Evaluation of the Adoption and Privacy Risks of Google Prompts

Christos Avraam[1] and Elias Athanasopoulos[2]

[1] School of Electronics and Computer Science, University of Southampton, Southampton, UK
ca2u19@soton.ac.uk
[2] Department of Computer Science, University of Cyprus, Nicosia, Cyprus
eliasathan@cs.ucy.ac.cy

Abstract. Internet services struggle with implementing better techniques for making authentication easier for the end-user by balancing those traits without sacrificing their security or privacy. One very recent such technology is Google Prompts, where users can authenticate by merely tapping a prompt to their mobile phone. In this paper, we attempt to understand how Google Prompts work and the extent to which current users adopt them. To this end, we build a collection system for estimating, using a completely transparent methodology, the fraction of users that have enabled Google Prompts in their accounts. Our collection system can infer the adoption of Google Prompts in the wild. Most importantly, we can use the system for performing a preliminary study of the privacy implications of Google Prompts.

Keywords: Authentication, Privacy, Measurements

1 Introduction

2-Step Verification Phone Prompts (most commonly abbreviated to Google Prompts) [5] is a mobile-based authentication method and was introduced relatively recently by Google. With Google Prompts enabled, a user needs to only enter their email for authenticating with Google and then complete a *Yes/No* notification prompt received on their phone. If the user selects the option *Yes*, they automatically gain access to the account; otherwise, the authentication process gets canceled and blocked. Since 79% of the human population between ages 18-44 carry their mobile phones 22 hours a day [22], an unauthorized person (attacker) who does not possess the mobile device cannot gain access to the account. As previously explained, a user can either opt-in for an entirely password-less authentication experience or even combine it with another security factor from Google's weaponry to form a more traditional and convenient 2FA mechanism. By selecting the latter and more commonly with the combination of a text-based password, the user is required to enter it before receiving the Google Prompt explicitly. Essentially, this means they trade off performance and usability for additional security. These two modes (a) password-less authentication and (b) 2FA-mixed are deeply integrated with currently available Google services and allow users to easily switch from mode to mode or entirely turn off Google Prompts and rely upon traditional authentication.

Although Google Prompts seems like a real-life problem solver and 2SV adoption seems exponential, according to Google [15], the actual user adoption, through not just enabling Google Prompts but also for its wide variety of authentication methods, is still questionable. It is also questionable how this system behaves under an adversarial setting. Precisely, we show how an attacker can

carefully issue silent probes to Google Prompts enabled accounts for receiving back signals about their geographical location. Since the user's privacy is considered a top priority aspect, we need to perform a proper vulnerability assessment. In this paper, we build a collection system for assessing the *current* use of Google Prompts in the wild. Our system can perform all measurements without violating users' privacy or *disturbing* them by any means. The measurement methodology is entirely transparent, i.e., users are not affected by our probes. Building such a collection system allows us to infer how Google Prompts are integrated with the current authentication mechanisms offered by Google as well.

In this paper, we make the following contributions.

1. We review Google Prompts, a fairly recent 2FA technology introduced by Google. Our main goal is to understand how Google Prompts are integrated into the current authentication system of Google.
2. Towards realizing this understanding, we design and implement a collection system that estimates how many users who have a Google account, have enabled Google Prompts. All measurements are carried out ethically without affecting the security and privacy of the analyzed users. All measurements are stored entirely anonymized.
3. Based on our experimental findings, we make a preliminary assessment of the privacy risks introduced by Google Prompts. Towards this, we suggest possible attack methodologies, where an adversary can carefully issue silent probes to Google Prompts enabled accounts for receiving back signals about their geographical location.

2 Google Prompts

Google has developed a mobile-based 2-Step Verification (2SV) method to enhance the process of accessing Google services. By enabling this authentication method as the default, the sign-in process goes as follows: the users after they input their email, they receive a notification prompt on a preconfigured trusted device, such as a smartphone. On that prompt, they can tap *Yes* to allow sign-in (see Figure 6) and gain access to the service immediately or *No* to deny/block the sign-in process. Since its primary function is sending a prompt on a device, for future reference, we will refer to it as *Google Prompts*. The default configuration of Google Prompts is to outsource authentication to it, transforming authentication into an entirely password-less experience. Google Prompts removes the need for remembering passwords or tokens, as it only requires the user's email address. It is considered more user-friendly and faster for the average user than other 2SV and 2-Factor Authentication (2FA) methods [9]. For instance, one-time codes received via SMS or the Google Authenticator app, not only require password input but also to transfer/copy them into Google's sign-in page manually. The prompt may also include details of the device/client initiating the authentication request: (a) Operating System, (b) current timestamp, and (c) the location if it could be determined.

It is important to note that with the default configuration if a user for some reason has no internet access or even physical access to their trusted device, they have the option to select alternative authentication methods from the Google's available security features (e.g., text-based password, Google Authenticator app, SMS token). Illustratively, when the user has selected to enter their password instead of sending the prompt, in the next authentication session, the previously selected method (in this case, the password) is again *optionally* requested by default.

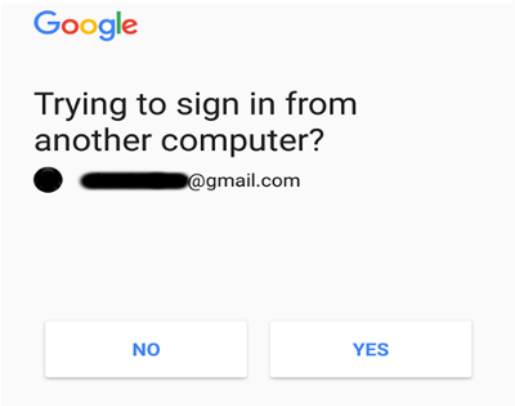


Fig. 1: Google Prompt as it is being received on the user's pre-configured device. By tapping *Yes*, it allows immediate access to the service otherwise *No* it blocks the unauthorized access.

Besides the above mode, there is another configuration where Google Prompts is used as a second factor (2FA) to trade-off speed with additional security. If this mode is enabled, the user enters compulsory their password or any other of the available Google's security factors. Then the Google Prompt is also *required* to be answered in order for the signing-in process to complete.

Moreover, when someone attempts an authentication for the first time on a new entity (e.g., the user is authenticating with Google using a new laptop), an extra step is added. A number appears on the browser, and if the device holder selects *Yes* to approve sign-in, the user has to select one of the three given numbers on the phone to match it. Otherwise, if the wrong number is selected, the authentication process gets canceled, regardless of the previous step selection.

Finally, depending on the operating system of the user's device (IOS/Android), compatibility requirements are slightly different. Android devices must keep Google Play Services up to date and have enabled screen-lock. In iPhones, screen-lock is already enabled because Touch ID requires it, and it is also necessary to have the Google Search app pre-installed since it delivers the prompts. Screen-lock is a core requirement, since it prohibits unlocked phones, accessible by anyone, to confirm Google Prompts.

2.1 Research Challenges

Assessing Google prompt's user adoption, was a challenging task since it was required to construct a collection system that can infer if a given e-mail account is associated with Google Prompts (black-box evalution). We needed to evade Google's defense mechanisms or at least deal with them in a proper manner. By resolving such issues, we can conclude to a better understanding of how exactly this new 2SV variant works. In this part, we review the significant difficulties we had to overcome.

Request Rate When a single entity issues about 5 to 6 consecutive requests on a given day for one account, Google Prompts gets disabled. Instead of sending a prompt on the preconfigured trusted device, as we have already discussed, a corresponding error message (see Figure 2 case A) appears

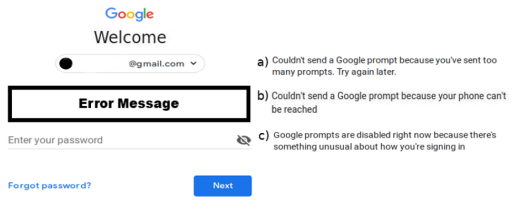


Fig. 2: Indication that Google Prompts are disabled with three possible error messages. Most imporntantly, the under analysis account does not receive any prompts. For case C, it occured explicitly when the routing was made through Tor.

on the browser that issued the request, and a password is required in order to sign-in. Therefore, each account should be tested only *once*. In addition to this limitation, we lessen the number of emails we can analyze per day, due to we should not be intrusive to the server itself. Google Prompts can also be disabled in cases where the users cannot use, reach (see Figure 2 case B), or lost their device. For such cases, Google recommends to its clients to take specific actions [3] in order to resolve any issues.

CAPTCHAs The combination of performing automated procedures in the Google API and assessing different accounts using one client had as a result of the service to project CAPTCHAs [8,23] after just a few requests. Therefore, we needed to perform all tests as cultivated/humanly as possible and in a more distributed fashion. We route all of our traffic through the anonymizing network service TOR [13], so we can send all requests using several exit nodes with different geolocations. Interestingly enough, using TOR gives us two notable benefits. Firstly, different exit nodes will issue requests that in most of the times will not be blocked by CAPTCHAs. Secondly, and most important, assuming the account under test has enabled Google Prompts, Google will refuse to send the prompt to the user under some under investigation circumstances (will be discussed further in Section 5). Instead, Google will issue a unique message that *prompts are disabled* for the particular account and requests for the user's password. The above case scenario allows us to infer if Google Prompts are enabled without actually annoying or disturbing *any* user. The received message for this particular case is depicted in Figure 2; notice the case *C)*.

Fail Validation Page As we have previously mentioned, there is a second configuration/mode where Google Prompts are used as a second factor. In this mode, for the authentication to be successful, the user's password is compulsory and entered before the validation with the phone prompt. Since we do not know the user's password to proceed further, we cannot examine and measure this configuration. Therefore we cannot differentiate between Password-only configurations and Password&Google Prompts configurations. However, we have noticed that we can deduce whether a user is using this particular mode by invoking the reminder-password process (Forget password? - see Figure 5b).

Upon initiating this process, Google attempts to validate the user's identity before allowing them to reset their password. The options presented to the user are last remembered passwords, sending an SMS to the user's device, and using another email address (previously defined). If the configuration we mentioned earlier, is enabled, one of the reminder-password validation options

involves sending a prompt to the user's pre-configured device; otherwise, it is not an option. This deduction was confirmed by enabling/disabling this configuration for a multiple of our owned test accounts and consequently checking all options presented to the user.

A significant drawback to this approach is that Google only allows the password-reminder process to occur when the device issuing the request has already been successfully authenticated in the past. In other cases, Google prohibits the password-reminder process and returns a *Fail Validation Block Page*, which we have not yet found a successful way to bypass it.

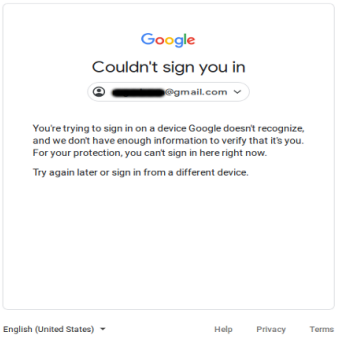


Fig. 3: *Fail Validation Block Page*. The above page may be displayed during the Password Reminder process and blocks the collection system from continuing forward.

3 Research Methodology

3.1 Dataset

According to Forbes [17], on the 9th of September 2014, almost 5 million Gmail addresses paired with passwords were leaked and published online on a Bitcoin Security forum. As well as other websites, Google announced about the leaked accounts and suggested that its clients take action to increase their account's security by adding extra verification layers [10]. Therefore, we assume that these emails should be enhanced with additional security measures. We apprehend this dataset with only the email addresses and storing each of them in our database to be checked. Some of its entries were not adhering to a correct email format, and we exclude those using an email regular expression. Also, some duplicates existed, which we excluded since we must check each email only once.

3.2 Collection System

The collection system is designed for a single purpose, to experimentally measure how many users with a Google Account have set Google Prompts as their default authentication method. It can navigate effectively through the whole authentication process and, depending on each account security configuration, follows a different evaluation path. The system comprises two major components: the interface, written in Python, and an evaluation script, written in JavaScript.

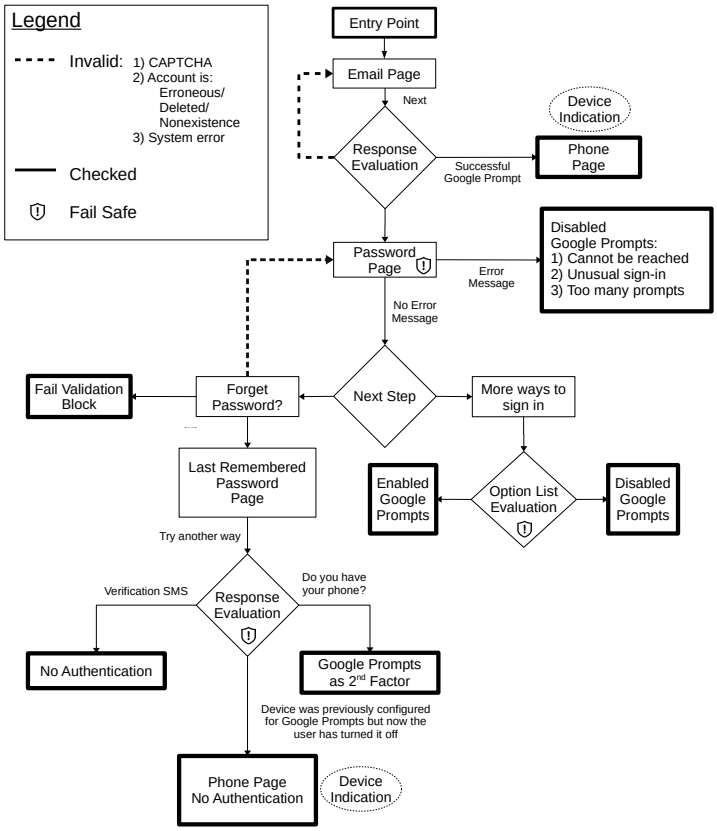
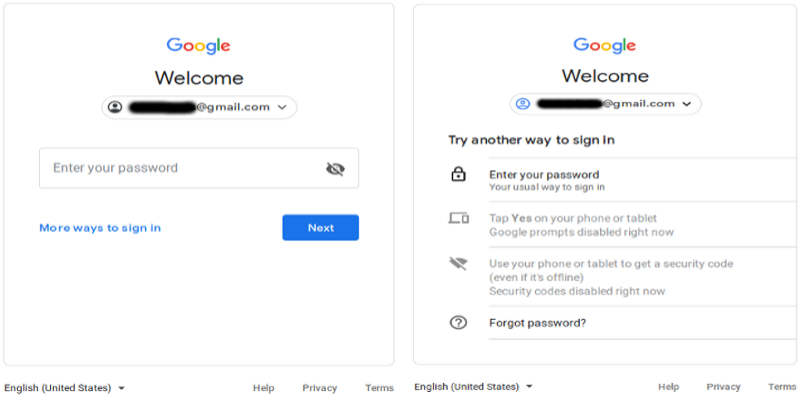
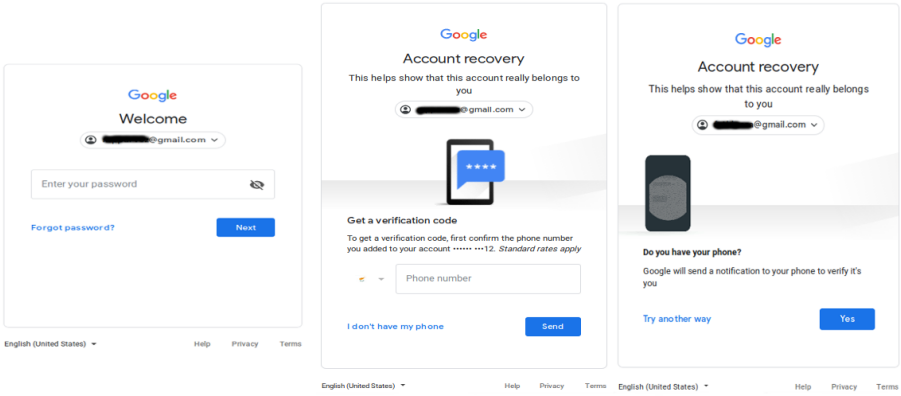


Fig. 4: Evaluation Decision Tree used by the collection system. The different states and their interconnections allow us to effectively understand how Google Prompts are realized and, using this understanding, to proceed and perform actual measurements related to the adoption of Google Prompts in the wild. Notice, that Google Prompts can be enabled in several ways, such as for *password-less authentication* (see states *Response Evaluation*, *Password Page* and their derived ones), as well as the second factor in 2FA (see states *Forget Password?*, *More ways to sign-in* and their derived ones).

The interface initially creates and connects to an SQLite database (see Section 3.1) and retrieves a specified amount of not yet analyzed email addresses. To achieve automation, we implement this component based on the Selenium WebDriver [2]. Selenium uses a browser-specific web driver which sends commands (sent in Selenese, or via a Client API) to an actual browser to fetch its results. However, any attempt performed by automated means was blocked by CAPTCHA and to remain under the radar without issuing too many requests from the same hosts, as explained in Section 2.1, we use a proxy which redirects all network traffic through Tor. For each distinct email address, we create a new TOR circuit by utilizing the Stem Controller [4], and we record the current IP



(a) *More ways to sign-in*. The user can gain access to the service through alternative means than the password. The available options are presented in a list and could be either enabled or disabled.



(b) *Reminder Password*. The user can recover their account through a series of sequential steps where they must prove their identity. The steps displayed are depended on the account's security configurations.

Fig. 5: Additional steps/options a user can perform on the Google's sign-in page

address of that TOR exit node. Before that, we ensure that it has a different address than the previously analyzed email; otherwise, we send a signal *NEWNYM* to create another TOR circuit. Since TOR exit nodes are located worldwide, the messages and texts displayed are in different languages; so, we used the URL parameter "hl=en-GB" to always display text in English in the progression of being appropriately evaluated by the system. The Selenium handler is suited better for sites that use browser detection (i.e., Google), since it ensures cross-browser consistency and has more options than other more lightweight equivalents, such as CasperJS [7]. One of these options is selecting between multiple browser drivers,e.g., Firefox and Chrome. We chose the Firefox driver (geckodriver) since it allows disabling specific parameters, such as Cookies and History trackers, which helped us to disguise Selenium further. Also, in the progression of making the whole process more similar to a real-time authentication attempt by an ordinary user, we emit random delays (time sleeps) between each execution of the Selenium commands that interact with the browser.

Next in line, the system enters the email address to the Google sign-in page, automatically clicks the button *Next* to receive the response. Here we perform most of the processing using the evaluation script component, which scans the HTML document for specific tags to determine its next move. We have constructed an *evaluation decision tree*, which we depict in Figure 4, and based on that, we reach a verdict regarding the default authentication method the user has set. In essence, in Figure 4, rectangle nodes represent distinct pages in the Google Authentication system, whereas those emphasized in bold indicate the inferred authentication method for each valid account, which we mark as *Checked*. On the other hand, dotted lines mean that the account was marked as *Invalid* due to one ore more of the following reasons: 1) response contains a CAPTCHA (some instances may still occur for yet unknown reasons), 2) the email account under testing is either erroneous, deleted, or does not exist, 3) a system/network error occurred. Additionally, nodes with dotted circles can indicate the type of the user's device, which is considered a privacy leak and will be discussed further in Section 5. Recall that a user can decide to use Google Prompts in two modes (see Section 2). For inferring the alternative authentication option change-up we invoke the *More ways to sign* (see Figure 5a) process. On the other hand, for deducing the second mode (see Section 2.1), we invoke the *Reminder Password* (see Figure 5b) process as well.

Moreover, one significant enhancement we added to our collection system was the implementation of Fail Safes. They have an active role in ensuring that the system is running smoothly without any interruptions. They are located strategically in specific points in the EDT to maintain a debug log, which indicates necessary specific system changes (new/modified JavaScript elements) and to handle possible exceptions due to network or driver failure. Finally, after each email analysis, we store the Status (Checked/Invalid/Null), Exit IP Address, Current Time, Authentication method, and a Debug Log in our database.

3.3 Ethics

Before starting our research, we had to confirm that our approach does not annoy or affect the owner of the account in any case. We did this by replicating our experiments on various Google accounts created by us for this purpose. Generally, if we do not reroute our script through TOR and when Google prompts is the default authentication method, the user usually receives the notification on his trusted device. However, if we do reroute through TOR, in most cases, Google Prompts are disabled, as explained in Section 2.1. Then the process terminates, and as a result, the user is never notified by the service for an authentication attempt. At the same time, we count this in our evaluation as a Google Prompt authentication.

The number of active registered users in Google is over 425 million [1], and the service receives over 106 million requests per month, which is over 3.5 million requests per day. In order not to be intrusive to the service, we apply a limit of an average of 1000 requests per day. Thus, the number of requests we issue per day is about 0,028% of the total daily requests it receives, and in any case, it can not be considered intrusive.

Furthermore, as far as the collecting data is concerned, we do not store any privacy-related information of the user (e.g., images, phone numbers), and all other information is anonymized. We only store data that we categorized as observations on which method of authentication a user has set as default. After that, we use this data to store counters which represent our measurements.

4 Results

4.1 Initial Evaluation

Our initial evaluation consisted only of the password-less configuration mode. Therefore inferring if a particular Google account is using this mode, requires only a single request with the e-mail address as the sole input. We initially analysed 30,319 Google accounts from our dataset of which 27,286 (90.00%) were valid (*Checked*). The others were *Invalid* for reasons described in Section 3.2.

Result		Checked Accounts	
		Total	Percentage
Google Prompts	Successful	3	0.01%
	Disabled	1,024	3.75%
Unknown Cases		26,259	96.24%
Total		27,286	100.00%

Table 1: The number of Google accounts, where Google Prompts are enabled in password-less mode only.

Out of the valid accounts, we conclude that 1,027 (3,76%) of Google users, grouped as *Google Prompts* in Table 1, have Google Prompts enabled with the password-less configuration method. In the 3.75% of the cases, Google Prompts were disabled (password was requested instead with an error message) for the reasons we discussed in Section 2.1 and displayed the Figure 2. Since Google does not show the error message for an account that has not configured Google Prompts *in any mode*, we can filter out all of those accounts. More importantly, the accounts mentioned above *do not receive* an actual prompt, while we perform the measurements, and our method is relatively transparent. Nevertheless, in the sporadic three cases (0.01%), Google displayed a message indicating that it successfully sent the prompt. We are thoroughly discussing the reason behind this in Section 5.

For the remainder accounts, 26,259 (96.24%), there was no error message, and only the password form was presented. Since our collection system could not proceed further, at first, we assumed that these cases were not having an enhanced authentication mechanism enabled and *relied only on password*. However, we were wrong in our assumption, and more accurate measurements are depicted below.

4.2 Final Evaluation

We later discovered there are additional ways to deduce whether an account has Google Prompts enabled. In particular, a user can:

- Enable Google Prompts with the first configuration (password-less) but chooses to use the password (e.g., because their device is inaccessible during authentication). In the next authentication session, a password form is used as default but is *optional* since the option for more ways to sign in is still enabled (sign in only with the prompt).
- Enable Google Prompts as a second factor (second configuration). The password is *compulsory*, and to complete the authentication the prompt must be accepted (tap *Yes*) as well. To deduce

this configuration of Google Prompts, our collection system must interfere with the password-reminder process (see Section 2.1).

Both these states are presented in Figure 4 and in the Table 2 under *More ways to sign in* and *Forget Password - 2nd Factor*) paths, accordingly.

Authentication		Checked Accounts	
		Total	Percentage
Google Prompt	Successful	13	0.05%
	Disabled	1129	4.17%
More ways to sign in	Enabled	7	0.03%
	Disabled	440	1.62%
Forget Password	2nd Factor	242	0.89%
	No authentication	1670	6.17%
	Fail Validation Block	23585	87.07%
Total		27086	100.00%

Table 2: More accurate measurements obtained with the inclusion of the second configuration mode of Google Prompts and the additional cases of the first one. We observe that the number of Google Prompts accounts almost doubled (6.76%).

In this evaluation, we analyzed an additional amount of 30,014 Google accounts (distinct from the previous evaluation) from our dataset, of which 27,086 (90.24%) were valid (*Checked*). The others were *Invalid* for reasons described in Section 3.2. For the cases which have Google Prompts enabled in both configuration modes (1st mode: *Google Prompt* and *More ways to sign in*, 2nd mode: *Forget Password - 2nd Factor*) we have a total of 1831 (6.76%) accounts, of which Google Prompts were enabled in either of the two configuration modes. It is a small percentage, but it is almost double the previous estimate in our initial evaluation. In addition, notice that the Fail Validation Block section which covers a large percentage of our results. Similarly, as the previous evaluation (*Unknown Cases*), we were not able to navigate further to discover whether they were using Google Prompts as a second factor or have not set an authentication at all(see Section 2.1).

In summary of both evaluations, we can conclude that our collection system can successfully infer that 2858 (5,25%) accounts have adopted Google Prompts out of the total of 54372 valid Google accounts we checked.

5 Privacy Leaks

While measuring the adoption of Google Prompts in the wild, we tried to be as stealthy as possible to avoid users' annoyance. We performed all the measurements in such a way that they will not be receiving the actual prompts. Nevertheless, there was *still* a tiny fraction of Google Prompts received by users (about 16 in more than 50,000 valid accounts), which we could not mitigate further due to the black-box evaluation. We stress here that for those few successful Google Prompts, we did not further analyze if the prompts were answered positively. Security implications are out of the scope of this paper. Still, in such cases, if the targeted user accidentally confirmed the prompt, our

collection system or generally an attacker can freely access the corresponding account. Therefore a possible attack scenario is to send phishing emails to the victims (real-time) to manipulate them to accept the prompt. Beyond security, it is also essential to emphasize the protection of the user's privacy. Thus we focused here on finding possible privacy violations.

As we have previously mentioned in Section 2, location tracking is part of the Google Prompts authentication, but we do not know at what extend it is being utilized. For that reason, we tried to investigate the variables enabling the successful reception on the user's device for that small percentage of users we have found. Recall that our collection system leverages exit nodes that are distributed across the globe, and in each email analyzation, we switched from node to node. Since the location was the primary variable change, successfully sent prompts are indicating a common signal between the triggering node (i.e., the TOR exit node) and the under analysis account. For instance, a TOR exit node that is closely (geographically) located to the user may successfully trigger a Google Prompt, which in turn can reveal the user's location. On the other hand, a Google Prompt that fails to be delivered indicates that the user is likely *not* geographically close to the particular node. As a result, determined attackers can orchestrate probes from different TOR exit nodes to infer the areas where the victim user (targeted Google account) is not currently located. Sequentially, they might discover the actual location by the process of elimination. Conducting a study to demonstrate such issues is beyond the scope of this paper, and is likely beyond the common sense of ethical research.

Of course, we understand that the location is not the only crucial factor here. Nevertheless, we alert the reader that at this point, Google Prompts, as designed, may be potentially used for extracting other private information as well. For example, our collection system during email analyzation was able to view data such as device brands and operating systems, user's first names, and profile pictures if they were previously set. In Figure 6, which is the default HTML response the client receives when the user has enabled Google Prompts, we can see the illustration of these examples. The above information can be obtained in the final pages with the indication of *Device Indication* in Figure 4.

6 Related Work

2FA hardens authentication for protecting users with stolen credentials; however, its efficacy is still questionable [20]. Initially, 2FA was a promising defense layer against simple, but highly effective attacks, such as phishing [12]. Nevertheless, today we are aware of advanced attacks that can bypass 2FA by *phishing* the second factor, aswell [14]. Despite the shortcomings of 2FA, researchers seek to invent and propose new 2FA-based systems that utilize second factors that can be reasonably user-friendly by requiring little or no user interaction. For instance, Sound-proof [18] uses as a second factor the proximity of the users phone to the authenticating device. At the same time, Wi-Sign [21] leverages perturbations in the WiFi signals incurred due to the hand motion while signing. In this paper, we do not assess the effectiveness of or exploring attacks that can bypass 2FA. Still, we investigate the mechanics of Google Prompts, a very new 2FA system offered by Google. To this aspect, this paper is closer to a similar study about quantifying the adoption of 2FA [19]. In this particular study, the researchers found out that 6.39% of about 100,000 email accounts of Google had enabled the 2-Step Verification method. Acemyan et al. [6] compare the usability, efficiency, effectiveness, and satisfaction measures among four of Google's 2FA mechanisms. Included in that set is Google Prompts which they found is more usable than the authenticator app.

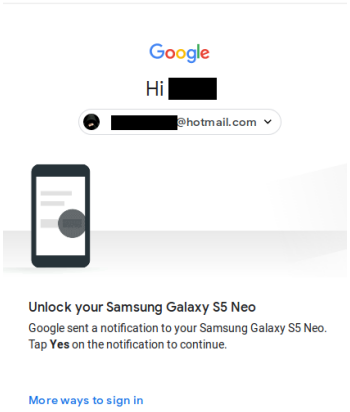


Fig. 6: Default HTML response the client receives when Google Prompts are enabled. We can see some information that can be linked to the user (first name, profile picture, device operating system and brand)

As far as other 2FA measurements are concerned, Weir et al. [16] performed a user case study asking e-banking customers to rate different 2FA methods regarding security, quality, and convenience. Overall, they found that users preferred usability above all and did not see the need for additional security. Ganson et al. asked mobile banking users to rate a single-factor and two 2FA schemes for telephone banking [24]. They found that the average user took 20 more seconds to complete each 2FA process than the single-factor one, and 2FA appears to users as a more secure solution but less easy-to-use. In a similar study, De Cristofaro et al. asked by various 2FA-familiar users to rate the usability of the three most popular 2FA solutions with different forms, that is, email or SMS sent to the user, a mobile app used an authenticator and a hardware token that produces OTP codes [11]. They observed that people who use 2FA for work prefer the mobile app option, while those who use it for personal and financial reasons prefer sending a text.

7 Conclusion

In this paper, we review and analyze Google Prompts, a recently enabled authentication mechanism by Google. Towards realizing our understanding, we developed the first collection system that can estimate how many users across the world have adopted Google Prompts. We have analyzed more than 60,000 email accounts, of which more than 50,000 were valid Google accounts. Out of the valid accounts, we have successfully inferred that 2858 (5,25%) accounts had Google Prompts enabled in either of its two configuration modes. However, due to the many obstacles we faced since the evaluation was black-box based, we could not determine with certainty the authentication method used for a large percentage of our results. Moreover, we showed how an attacker could carefully issue silent probes to Google users for receiving back signals about their geographical location.

.

References

1. Gmail now has 425 million active users. http://www.theverge.com/2012/6/28/3123643/gmail-425-million-total-users.
2. Selenium. https://www.seleniumhq.org/.
3. Sign in faster with 2-step verification phone prompts. https://support.google.com/accounts/answer/7026266?co=GENIE.Platform%3DAndroid&hl=en.
4. Stem docs. https://stem.torproject.org/.
5. Sign in faster with 2-Step Verification phone prompts. 2019.
6. C. Z. Acemyan, P. Kortum, J. Xiong, and D. S. Wallach. 2fa might be secure, but its not usable: A summative usability assessment of googles two-factor authentication (2fa) methods. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 62, pages 1141–1145. SAGE Publications Sage CA: Los Angeles, CA, 2018.
7. R. Agarwal. Choosing automated testing frameworks phantomjs / casperjs vs selenium, 2015. https://www.algoworks.com/blog/choosing-your-automated-testing-frameworks-phantomjscasperjs-vs-selenium/.
8. L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. in european cryptology conference (eurocrypt), 2003.
9. D. Bisson. Two-factor authentication (2fa) versus two-step verification (2sv), 2016. https://www.grahamcluley.com/factor-authentication-2fa-versus-step-verification-2sv/.
10. O. O. S. Blog. Cleaning up after password dumps. http://googleonlinesecurity.blogspot.gr/2014/09/cleaning-up-after-password-dumps.html.
11. E. De Cristofaro, H. Du, J. Freudiger, and G. Norcie. A comparative usability study of two-factor authentication. in proceedings of the workshop on usable security (usec), 2014.
12. R. Dhamija, J. Tygar, and M. Hearst. Why phishing works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, SIGCHI, 2006.
13. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM, 2004.
14. N. Gelernter, S. Kalma, B. Magnezi, and H. Porcilan. The password reset mitm attack. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 251–267, May 2017.
15. E. Grosse and M. Upadhyay. Authentication at scale. ieee security and privacy 11 ,15-22, 2013. http://www.computer.org/cms/Computer.org/ComputingNow/pdfs/AuthenticationAtScale.pdf.
16. N. Gunson, D. Marshall, H. Morton, and M. A. Jack. User perceptions of security and usability of single-factor and two-factor authentication in automated telephone banking.computers & security, 2011.
17. K. Hill. Google says not to worry about 5 million gmail passwords leaked. http://www.forbes.com/sites/kashmirhill/2014/09/11/google-says-not-to-worry-about-5-million-gmail-passwords-leaked/.
18. N. Karapanos, C. Marforio, C. Soriente, and S. Čapkun. Sound-proof: Usable two-factor authentication based on ambient sound. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC15, page 483498, USA, 2015. USENIX Association.
19. T. Petsas, G. Tsirantonakis, E. Athanasopoulos, and S. Ioannidis. Two-factor authentication: Is the world ready?: Quantifying 2fa adoption. In *Proceedings of the Eighth European Workshop on System Security*, EuroSec '15, pages 4:1–4:7, New York, NY, USA, 2015. ACM.
20. B. Schneier. Two-factor authentication: too little, too late. *Communications of the ACM*, 48(4):136, 2005.
21. S. W. Shah and S. S. Kanhere. Wi-sign: Device-free second factor user authentication. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous 18, page 135144, New York, NY, USA, 2018. Association for Computing Machinery.

22. A. Stadd. 79% of people 18-44 have their smartphones with them 22 hours a day, April 2 2013. https://www.adweek.com/digital/smartphones/.
23. C. M. University. Captcha: Telling humans and computers apart automatically, 2000-2010. http://www.captcha.net/.
24. C. S. Weir, G. Douglas, T. Richardson, and M. A. Jack. Usable security: User preferences for authentication methods in ebanking and the effects of experience. interacting with computers, 2010.