TIME IS ON MY SIDE:

Forward-Replay Attacks to TOTP Authentication

SocialSec 2023: 9th International Symposium on Security and Privacy in Social Networks and Big Data
University of Kent, Canterbury, UK | August 14-16, 2023

Giuseppe Bianchi
Lorenzo Valeriani

CNIT and University of Rome "Tor Vergata"

NSS-SocialSec 2023

Canterbury, UK
August 15, 2023

Brief Story

- At our lab, we are hardcore players of Candy Crush Saga.
- As such, we cannot wait for game lives to replenish.
- By manipulating the internal clock settings on the phone, we easily obtain free lives.

So we came up with a question...

Can a similar principle be exploited for malicious purposes by an external attacker?

Agenda

Introduction

- Two-factor authentication (2FA) is a widely used method to add an extra layer of protection against unauthorized access or fraud.
- One-time password (OTP) authentication is a very common second factor.
- TOTP has gained popularity in more recent years due to its convenience and usability. It requires the user to:
  1) Have a TOTP generator installed on their device.
  2) Have preliminarily shared with the server a secret key.

TOTP

$$\mathrm{TOTP}(K,T) = \mathrm{HOTP}\left(K, \left\lfloor \frac{\text{Current Unix time} - T_0}{X} \right\rfloor\right),$$

- Specified in the IETF RFC 6238.
- At the numerator is the number of seconds elapsed from a reference instant (e.g., midnight UTC of January 1, 1970).
- X is the interval (e.g., 30 seconds).

TOTP

- It just requires a loose time synchronization among the device and the server clocks.
- It ensures that even if an attacker obtains a TOTP code, it will be useless after a short time.
- The client and server can independently calculate the expected code at any given time.

Time Tracking on Mobile Devices

- Smartphones are highly complex devices.

- Mobile operating system's clock can be set in different ways:
  1) Manually
  2) NTP
  3) NITZ
  4) GNSS

Park, S., Shaik, A., Borgaonkar, R., Seifert, J.P.: White rabbit in mobile: Effect of unsecured clock source in smartphones. In: Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices. pp. 13–21

Vulnerabilities in TOTP Applications

- In [1], the storage of secrets and network traffic of 16 Android TOTP apps was analyzed.
- [2] deals with the backup process off 22 Android TOTP apps to assess the security of the stored data and how it was transmitted over the network.
- In [3], an analysis was performed on 11 Android TOTP apps to investigate whether root access could be utilized to extract secrets from storage and memory.

[1] Polleit, P., Spreitzenbarth, M.: Defeating the secrets of OTP apps, pp. 76–88. IEEE (2018)
[2] Gilsenan, C., Shakir, F., Alomar, N., Egelman, S.: Security and privacy failures in popular 2FA apps. prepublication.In: USENIX Security 2023 (2023)
[3] Ozkan, C., Bicakci, K.: Security analysis of mobile authenticator applications.
In: 2020 International Conference on Information Security and Cryptology (ISC-TURKEY), pp. 18–30. IEEE (2020)

Vulnerabilities in TOTP Applications

- Two blog posts adopt a conceptually similar approach to the one described in our work.
- They focus on specialized technologies (TOTP hardware tokens).

Deeg, M.: To the future and back: hacking a TOTP hardware token (SYSS-2021-007). https://blog.syss.com/posts/syss-2021-007
Huseynov, E.: TOTP replay attack - yubikey. https://medium.com/@eminhuseynov_37266/totp-replay-attack-yubikey-et-al-adde8e8c62d3

Positioning of this Work

- We deal with the widespread and general case of TOTP applications on Android smartphones.
- We do not aim to attack the secret.
- Our POC does not require any modifications at the operating system level, such as root privileges.
- We suggest to use the new term "forward-replay" so as to distinguish this attack from other forms of time traveler attacks.

Forward-replay attack

- The attacker obtains access to the device at a given time, either manually or using malicious applications.
- The attacker is able to activate (or access a background running) TOTP authenticator app.
- The attacker manipulates the device date/time and sets it to a sequence of future time instants {t1 , t2 , · · · , tn}.
- For each sampled time instant, the attacker reads the TOTP generated code and creates a log of valid timestamp-code pairs {(t1 , OTP1), (t2 , OTP2), · · · , (tn , OTPn)}.

Forward-replay attack

- By having the ability to compute and collect future TOTPs, the attacker can "replay" them to the server at the exact times in which they will be considered valid, i.e. when the server time reaches the sample time of the OTP.

- It does not require changing the server clock or eavesdropping the communication channel.

- A quite powerful attack model is still necessary, as the attacker needs three capabilities:
  1) Moving the time forward
  2) Capturing the code
  3) Extracting the code from the device

Threat Scenarios

- Manual Access. An attacker may have physical access to a victim's device for a given interval of time.
- Malicious App. Can be manually installed or injected through ordinary offensive penetration techniques.
- Public USB Charging Sockets. They can pretend to be input-output or network devices or be exploited as a means to use the Android Debug Bridge (ADB).
- The victim device can be restored to an unaltered state after the generation of the codes. The codes cannot be traced back to the date in which they were gathered, making it difficult for the victim to detect that their codes have been compromised.

Proof of Concept

- To demonstrate the practical feasibility of an attack, we developed an Android application whose purpose is to trigger the generation of future codes and to log them.

- We initially targeted the codes generated by the Google Authenticator app (100MM+ downloads). The app was then expanded to support also the apps FreeOTP Authenticator (1MM+ downloads) and Duo Mobile (10MM+ downloads).

- Our app was successfully tested on a completely standard and non-rooted Google Pixel 7 Pro running Android 13 with the March 2023 security update.

Android Accessibility Services

- Accessibility services are intended to assist users with disabilities in using Android devices and applications. They run in the background and receive callbacks by the system when accessibility events are triggered.

- An accessibility service is capable of simulating a gesture on the touch screen as if it had been initiated by the user directly, and provide input as if typed on the keyboard.

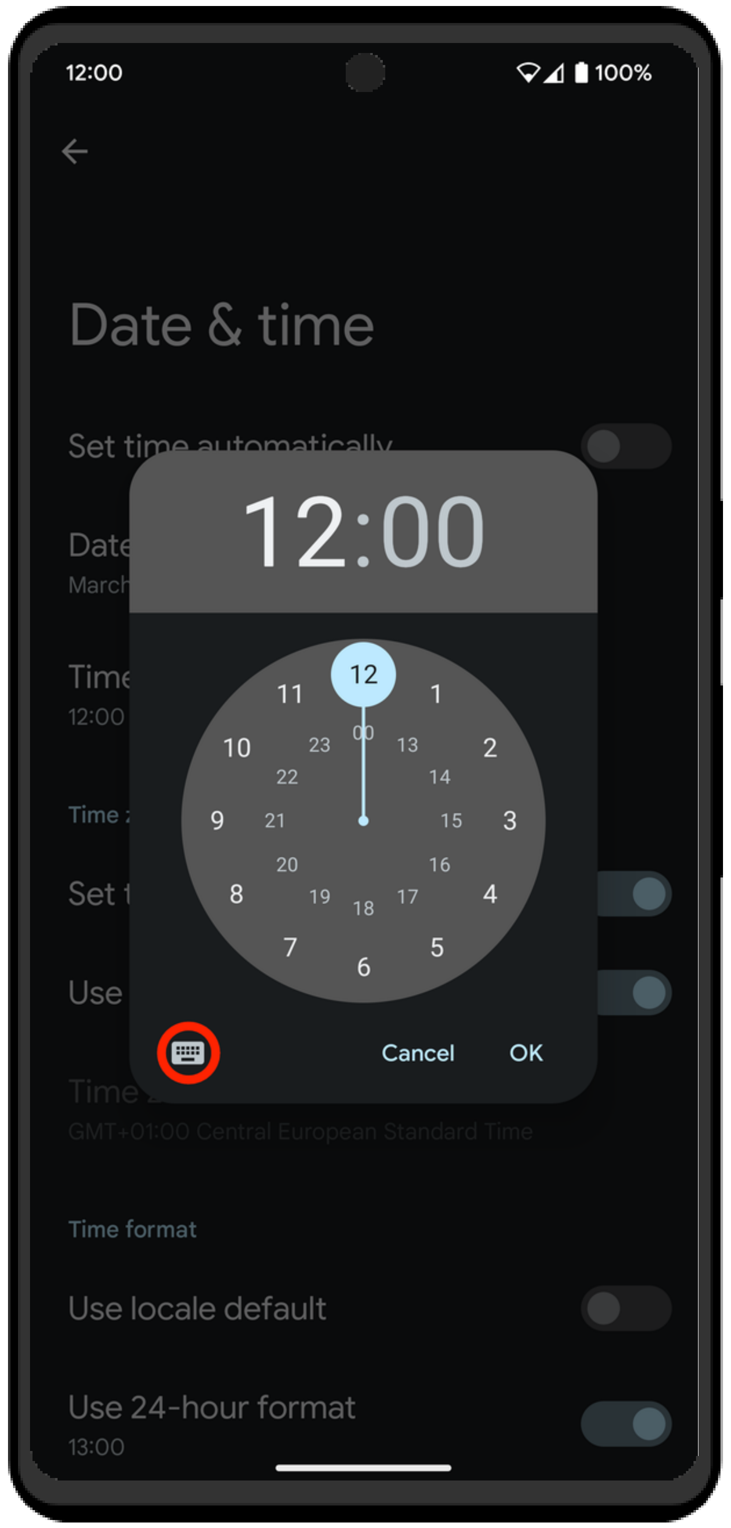
Activating the TOTP Authenticator App

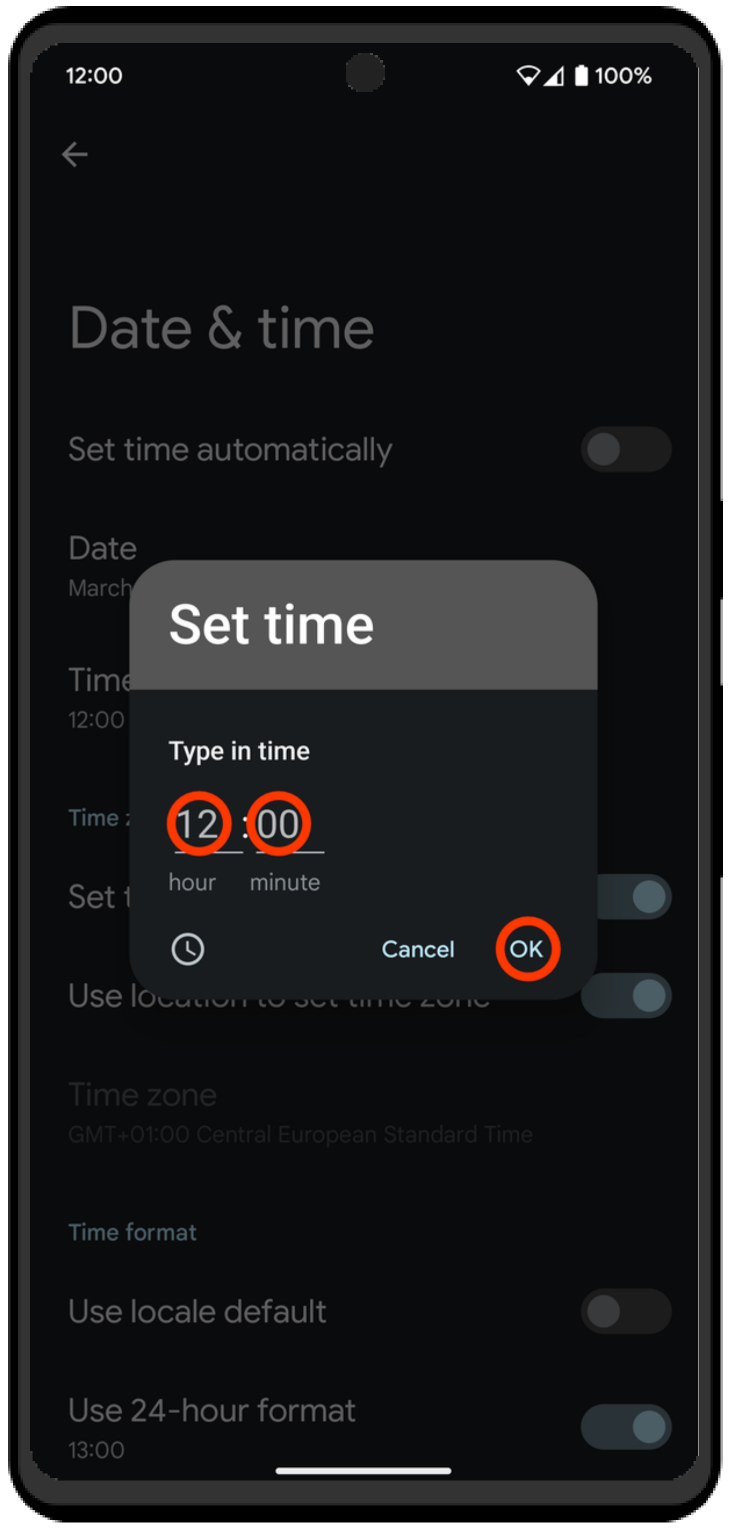
- To launch the app, the package manager can be interacted with to retrieve the launch intent associated with the package name of the OTP generator (e.g., com.google.android.apps.authenticator2).
- The same package name has to be declared in a <queries> element in the malicious app's manifest to comply with package visibility filtering of Android.

Moving the Time Forward

- In order to alter the system clock, Android applications necessitate specific permissions, such as those granted to system apps or devices with root access.
- We overcame this limitation by creating an accessibility service that mimics the touch inputs made by a user who is manually changing the time.

Moving the Time Forward

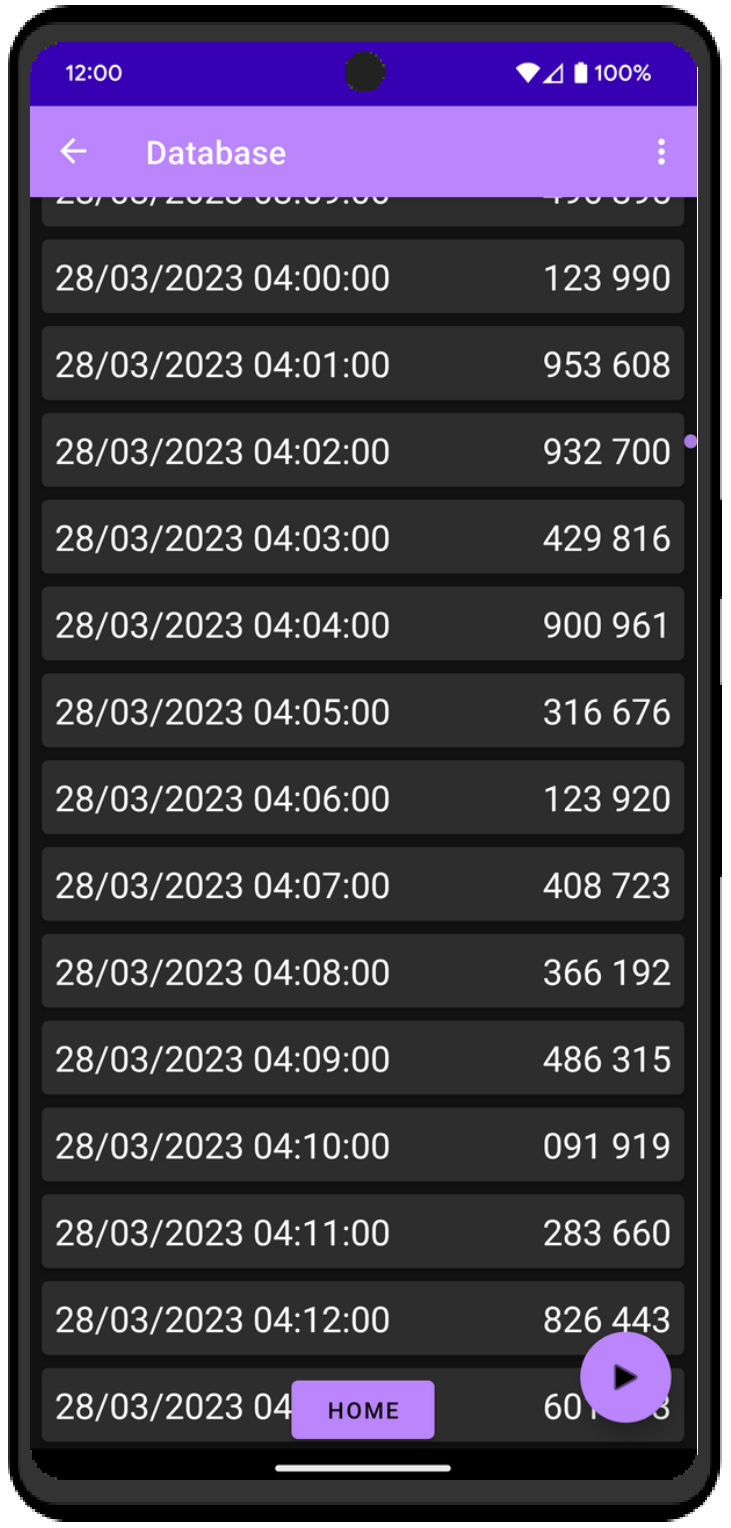




Capturing and Extracting the Codes

- When the TOTP is shown on the screen, an accessibility event is triggered and the service can capture the code.

- The code can then be saved to a database or file for later retrieval from the device or transmitted over the network.



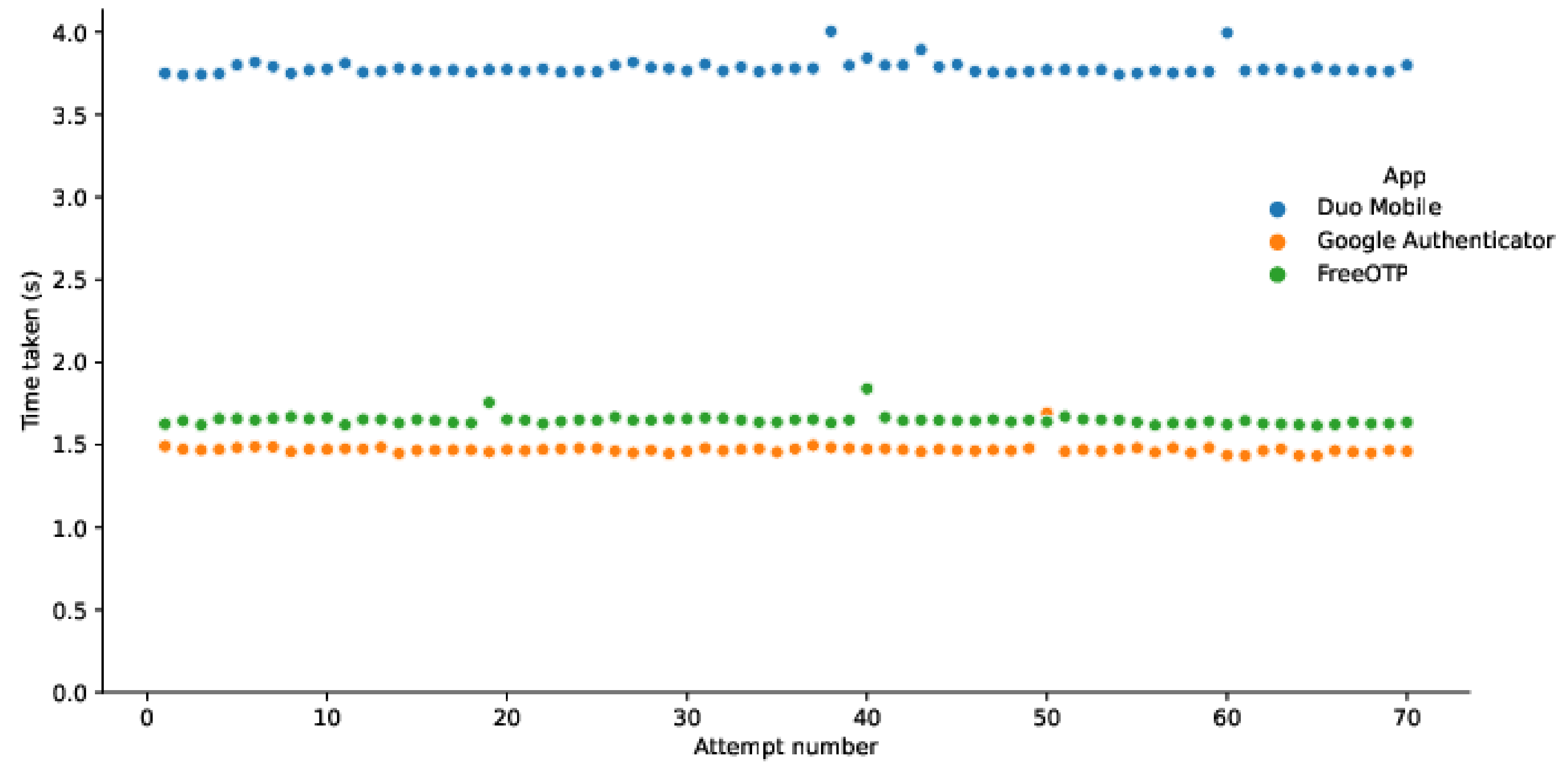
28/03/2023 04:00:00	123 990
28/03/2023 04:01:00	953 608
28/03/2023 04:02:00	932 700
28/03/2023 04:03:00	429 816
28/03/2023 04:04:00	900 961
28/03/2023 04:05:00	316 676
28/03/2023 04:06:00	123 920
28/03/2023 04:07:00	408 723
28/03/2023 04:08:00	366 192
28/03/2023 04:09:00	486 315
28/03/2023 04:10:00	091 919
28/03/2023 04:11:00	283 660
28/03/2023 04:12:00	826 443
28/03/2023 04	60 [illegible]

Adapting to Different TOTP Apps

- The core of the app remains valid, so limited actions are required to target different TOTP apps
- FreeOTP requires a tap on the code location to display it.
- Duo Mobile not only has a button to show the code but also implements a 30-second application-level timer to update it.

Numerical Results

App	Mean	Std. Dev	Min	Max
Duo Mobile	3.782	0.045	3.740	4.005
Google Authenticator	1.471	0.030	1.434	1.692
FreeOTP	1.648	0.030	1.616	1.840



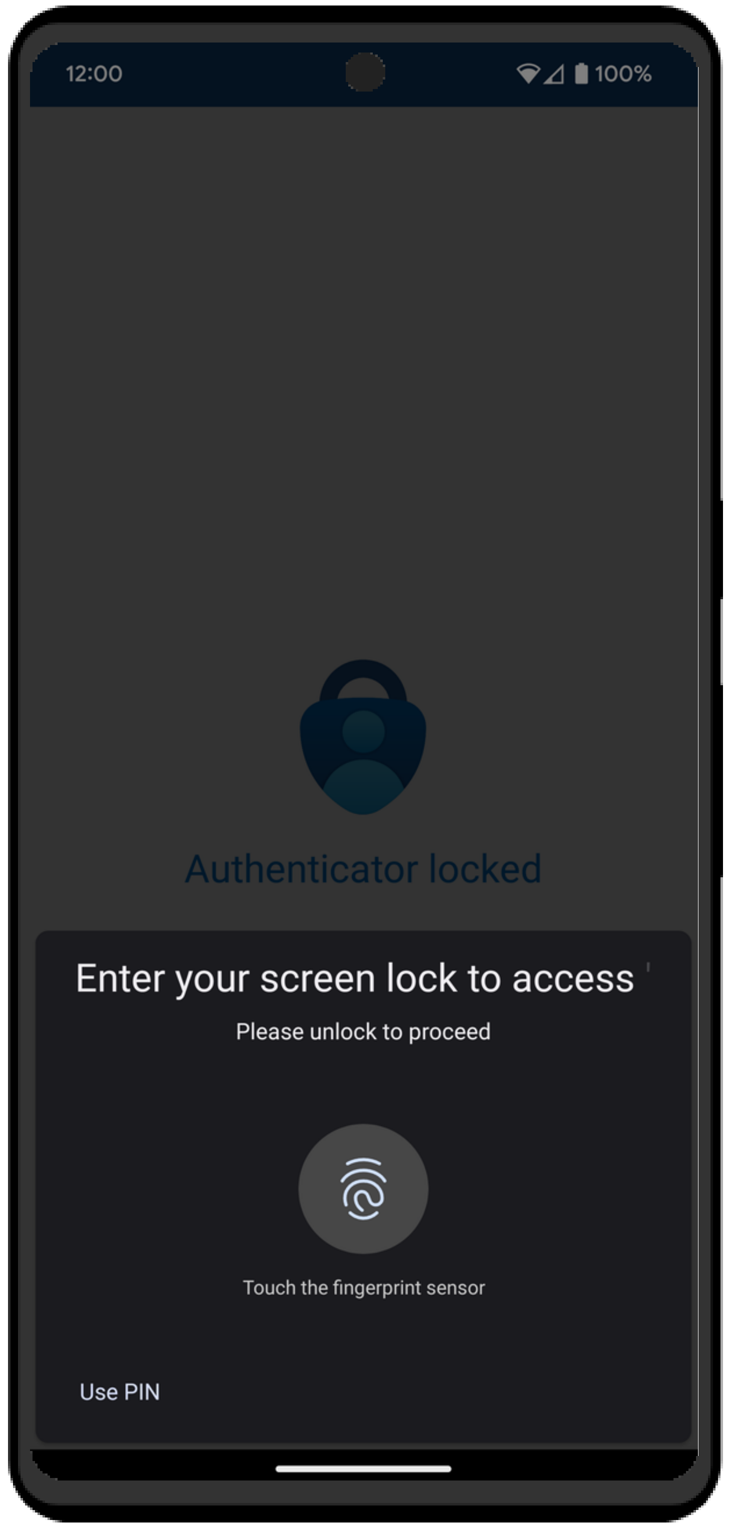
We were able to obtain TOTP codes from Google Authenticator for every minute of an entire day in about 35 min of execution. The one-minute interval for generating TOTP codes is determined by both the granularity allowed by manual time setting and the recommendation in the TOTP RFC to consider both the current and previous OTPs as valid for usability reasons. This effectively creates a time window of valid codes that spans one minute

Attack Improvements

- Investigating other means to carry out the attack (e.g USB communication).
- Implementing methods to make the attack more covert like injecting the application into a seemingly harmless one.
- Exploring other ways to change the time on a device, such as the NTP and NITZ protocols.

Possible Mitigations

- Using strong passwords and lock screen methods.
- Forcing online code generation with a cryptographic synchronization exchange.
- Resorting to key rotation approaches.
- Showing biometric authentication dialogs whenever the app switches from background to foreground.
- Relying on dedicated hardware or trusted execution environments for certifying the time.


12:00
100%
Authenticator locked
Enter your screen lock to access
Please unlock to proceed
Touch the fingerprint sensor
Use PIN

Thank You for Your Attention!

Questions?

Lorenzo Valeriani - lorenzo.valeriani@cnit.it