Forensic Science International: Digital Investigation 45 (2023) 301569

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS 2023 USA - Proceedings of the Twenty Third Annual DFRWS Conference

Factorizing 2FA: Forensic analysis of two-factor authentication applications

Jessica Berrios [a, *], Elias Mosher [a], Sankofa Benzo [a], Cinthya Grajeda [a], Ibrahim Baggili [b]

[a] Samuel S. Bergami Jr. Cybersecurity Center, Connecticut Institute of Technology, University of New Haven, 300 Boston Post Rd., West Haven, CT, 06516, USA
[b] Baggil(i) Truth (BiT) Lab, Center for Computation & Technology, School of Electrical Engineering & Computer Science, Louisiana State University, USA

ARTICLE INFO

Article history:

Keywords:
Digital Forensics
Artifacts
Two factor authentication (2FA)
Mobile applications
Windows
Android
iOS

ABSTRACT

Many sectors such as banking, academia, health care, and others have made Two-Factor Authentication (2FA) mandatory for all their registered users. The growth in the usage of 2FA technology demonstrates the need to understand how 2FA applications operate, the kind of information they store about their users, and the implications, if any, that may arise if malicious actors exploit them. Our work focuses on the forensic analysis of 15 2FA applications used by millions of people. Our analysis includes popular applications such as FreeOTP, Google Authenticator, Microsoft Authenticator, Twilio Authy, and more. The applications were tested on different operating systems (Android, iOS and Windows 10) and used with applications such as Facebook, Twitter and Instagram. Our methodology focused on not just forensically analyzing the devices' storage, but also the network traffic of all devices and the memory of the Windows machine. Results revealed that the majority of analyzed applications store encrypted/encoded and plain text information, such as secret keys, timestamps, account names, e-mail addresses, the application locking pin, and more. Consequently, we believe that the critical discovery of secret keys allows for the 2FA functionally to be bypassed and it is demonstrated in this work. Our results revealed that 14 of 15 applications stored the name of the social media application/account information, and 14 of 15 applications stored either plain text, or encoded/encrypted secret keys. Finally, 2 of 15 applications stored a pin in plain text used to lock the application and/or encrypt all information on the disk.

1. Introduction

User accounts are hijacked on a daily basis. As adversaries produce more sophisticated attacks, users continue losing billions of dollars from different types of internet scams. According to the 2022 Internet Crime Report released by the FBI, there have been 3.26 million cybercrime complaints from victims all over the world over the last five years and they have resulted in $27.6 billion in total losses (FBI, 2023). Account hijacking typically allows users to steal and control a user's data, money, or identity. Two-Factor Authentication (2FA) is a widely adopted approach for protecting user accounts. 2FA is an identity verification approach that requires two forms of identification to access a system (Microsoft, 2022). 2FA technology has been around for years, as it was first introduced in 1986 by RSA as a key fob (America, 2022). Since then, 2FA has rapidly become the standard for login security across many sectors. For instance, starting in 2021, Google began to automatically enroll 150 million of its users in 2-Step Verification (2SV), including 2 million YouTube creators. Consequently, 50% of those user accounts avoided being compromised (Kim, 2022). According to a study by Duo Labs, there has been a significant surge of 2FA implementations between the years 2017 and 2021. As of 2021, 79% of people reported using some form of 2FA in comparison to only 28% in 2017 (Childers, 2021).

2FA works by enabling a second factor of verification during the login process. Users first enter information they know such as a username and password. This is followed by a second factor and can either include (Fruhlinger, 2019):

- "Something you have" (e.g device)
- "Something you are" (e.g biometrics)
- "Somewhere you are" (e.g location based)

A few of the most common 2FA methods include text messages (SMS), Time-based One Time Passwords (TOTP), pre-generated

* Corresponding author.
E-mail addresses: jberr6@unh.newhaven.edu (J. Berrios), emosh1@unh.newhaven.edu (E. Mosher), sbenz14@newhaven.edu (S. Benzo), cgrajedamendez@newhaven.edu (C. Grajeda), ibaggili@lsu.edu (I. Baggili).

https://doi.org/10.1016/j.fsidi.2023.301569

codes, push notifications, and Universal Second Factor (U2F) security keys (Reese et al., 2019). Our work focuses only on 2FA applications that use TOTP, which in the majority of cases use time-sensitive generated codes that are six digits in length. Given the popularity and growth of 2FA, it will naturally attract adversaries.

Our research only focuses on artifacts left behind by various 2FA mobile and desktop applications and not on the applications being authenticated, i.e., Facebook. This is to understand the type of data discovered through different types of forensic acquisitions (disk, memory and network) and how that data can be utilized as evidence during an investigation or for malicious purposes given the situation. Therefore, our contributions are as follows:

- A primary digital forensics methodology and analysis of disk, network, and memory of fifteen 2FA applications presented in Table 1.
- A notable amount of relevant digital forensic artifacts shared on the open platform, the Artifact Genome Project(Grajeda et al., 2018).[1]

This paper is organized as follows: Section 2 presents related work. Section 3 explains the methodology, while Section 4 discusses the disk analysis and experimental results. Section 5 details the Windows memory and network results. Section 6 demonstrates 2FA Bypass with TOTP and conclusion and future work are presented in Section 7 and 8 respectively.

2. Related work

The popularity of 2FA is vast, as it is supposed to provide another layer of security to online information. The interest in discovering how to bypass it, however, has never been greater. A simple Google search on how to bypass 2FA produces hundreds of hits with blog posts, YouTube videos, articles, and so on. Grimes (2019) explains there are multiple ways to do this, for example, through session hijacking, man-in-the-endpoint attacks, SMS rogue recovery, brute force attack, duplicate code generators and more. One of the main goals in some of these attacks and our main goal in this paper is to acquire the secret “seed” or key as most applications refer to it, in order to be able to reproduce the six digit code to authenticate a user into an account.

At the time of writing and to the best of our knowledge, the methodology to obtain not only secret keys but other personal identifiable information from 2FA applications and a proof of concept evaluation on bypassing 2FA using those secret keys is a novel approach. Nevertheless, we acknowledge the work of (Ozkan and Bicakci, 2020) where they used an Android emulator to fetch secret keys from 2FA applications. In their results, they found five applications containing secret keys in storage and seven applications containing keys in memory. This work was accomplished by an entirely different method using basic reverse engineering techniques.

Other research that has been previously written in relation to 2FA technology focuses on the growing prevalence of 2FA, as well as a detailed analysis of their structures and how they work. Additionally, other research in general has focused on forensically investigating mobile applications ranging from social media to native device applications. Forensically investigating applications in multiple devices is also the focus of this paper. Therefore, the following subsections put the spotlight on related works conducted in the last ten years.

[1] https://agp.newhaven.edu.

Table 1
2FA applications tested.

Application	Version	Platform	Downloads
Aegis	2.0.3	A	100K+
FreeOTP	1.5	A	1M+
TOTP	1.89	A	100K+
Google	5.20R4	A/I	100M+
Microsoft	6.2207.4624	A/I	50M+
2FAS	3.17.0	A/I	1M+
Twilio Authy	4.8.8	A/I/W	10M+
Okta Verify	7.9.0	I	36K
Two-Factor	1.5	I	25
FIS Authenticator	4.4.6	I	120
Epic Authenticator	1.0	I	21
IBM Verify	2.5.2	I	72
Authenticator+	2.0.4	I	N/A
WinAuth	3.5.1	W	N/A
2 Factor Auth	2.5.1804	W	N/A

Key: A: Android, I: iOS, W: Windows.

2.1. Two-factor usability

The usability of 2FA has been researched in terms of what the best applications are for 2FA and how these applications are structured. Many of these studies specifically cover items such as the 2FA methods with the fastest response time, how user-friendly they are to set up, and how complicated they are to use (Reese et al., 2019). What is interesting to note is how 2FA is gaining more prevalence in many areas where security is becoming a larger concern. For instance, the incorporation of 2FA on Bitcoin wallets does not take the traditional route of using the five most common methods, but instead uses signatures to generate 2FA (Mann and Loebenberger, 2017). Another example of this involves implementing TOTP 2FA in private cloud services such as OpenStack. The advantages of adding this second level of security are discussed and compared to other major platforms that are already making use of 2FA such as Facebook, Microsoft, and Apple (Gordin et al., 2019). Additionally, 2FA is rapidly expanding and beginning to be incorporated into more of our daily tasks. For example, online transactions are becoming more popular every day and the growth of online stores poses a new challenge to securing online transactions. Due to this, new technologies such as SecurePay are beginning to appear to the public. SecurePay works to incorporate 2FA into a more secure online transaction application (Konoth et al., 2020).

In a study conducted on the comparative usability of 2FA, participants were asked what motivated them to begin using 2FA. The majority felt more secure using it. In addition, other forms of 2FA were analyzed and some drawbacks were mentioned. These drawbacks included a dip in productivity from companies who required 2FA access cards, which led to loss of revenue. The cause was lost time from problems such as employees losing or forgetting their access cards (De Cristofaro et al., 2013).

Consequently, 2FA can also be considered a security mechanism for IoT devices. Although many of the methods being applied to IoT devices do not follow the traditional methods for 2FA, they still provide an additional level of security. For example, a method being implemented on IoT devices includes sending a message to the server to which the IoT device seeks to connect, while the server responds with a one-time key (Gope and Sikdar, 2019).

Finally, one aspect that can cause hesitance when using 2FA is the risk of losing access to your account. If a user were to lose their device, then the user would lose access to their account. For this reason, many applications have a recovery option set in place. However, attackers can still use these mechanisms to gain unauthorized access to users’ accounts (Dimitrienko et al., 2014).

2.2. Other related works

Similar studies have been performed resulting in the findings of many digital forensic artifacts. Such works include but are not limited to investigations on extremist social media applications (Johnson et al., 2022), Android Auto and Apple Carplay Forensics (Mahr et al., 2022), video conferencing applications such as Zoom (Mahr et al., 2021), virtual reality applications (Yarramreddy et al., 2018), Chromecast forensic analysis (Sitterer et al., 2021), analysis of social networking applications (Al Mutawa et al., 2012), Dones (Clark et al., 2017), Amazon Alexa(Dorai et al., 2018) and many more.

3. Methodology

The forensic investigation of fifteen 2FA applications consisted of four main phases described in the following sections. These included scenario creation & testing, data acquisition, data analysis and 2FA bypass. It is important to note that out of the fifteen applications tested, six were only iOS platform compatible, three were Android, and two were Windows. One application was compatible with all three operating systems, while three were compatible with the Android and iOS operating systems.

The apparatus used to conduct this research is presented in Appendix A, Table A.4. No APK reverse engineering tools were utilized as that would retread the study by (Ozkan and Bicakci, 2020). Advanced analysis with decompilers/disassemblers is out of the scope of this research.

3.1. Scenario creation & testing

The first phase consisted of testing fifteen two-factor applications, listed on Table 1, along with the number of downloads (if available). Additionally, five social media and cloud storage service applications were selected to authenticate, the applications consisted of Facebook, Instagram, Dropbox, Twitter, and Snapchat. The 2FA applications were chosen based on two things, their popularity and their lack of it. The reason being to make sure there was diversity within the applications that could yield comparable results. This also allowed for a comparison to be made between applications that are created from well known organizations (i.e Google) to smaller less known organizations.

In order to implement realistic testing and acquire real-world datasets, the tests were performed on a rooted Galaxy S6 smartphone, a jailbroken iPhone 7 and a Windows 10 desktop set-up on a virtual machine (VM). As part of our controlled environment for testing, a WiFi hotspot was created to isolate all network traffic from all devices. The testing applications used were downloaded from the Google Play Store,[2] Apple App Store,[3] the Microsoft Store,[4] Github,[5] and the Authy website.[6]

Note, as expected, some of the applications automatically updated during the testing phase, however, this did not affect results in our process.

The process of testing all 2FA applications using five services to authenticate, which included four social media applications and one cloud service, consisted of five main steps with small differences between them. It is important to note that the method utilized here was to manually obtain the key from the application instead of scanning the QR code provided. To demonstrate this process, refer to Fig. 1 as it depicts the Facebook application being authenticated using the TOTP Authenticator application. The most common steps are highlighted below:

1. Set-up 2FA applications in devices and provided a phone number on one occasion.
2. Where optional, created a pin or password in order to lock the 2FA application and/or encrypt all its data in the device. Tests were conducted with both options.
3. Enabled 2FA in the service needing authentication (i.e., Facebook) and opted to use an authentication application, such as TOTP Authenticator.
4. Copied the secret key provided by the service needing authentication (i.e., Facebook) and entered it manually in the 2FA application in order to add the account. In some cases, to identify the account, the name was also added as well as the name of the service application.
5. The latter process created the six digit code required by the service needing authentication in order to be verified.
6. Logged in to the social media application using the six digit code.

3.2. Data acquisition

In order to acquire disk images and network traffic from each mobile device tested, a diverse set of acquisition tools were utilized (see Appendix A, Table A4). This includes Magnet Acquire[7] to obtain physical forensic images of the Android and ArtEx[8] to acquire images from the iPhone. Note that no Secure Digital (SD) cards were acquired in this research as their storage was not necessary for the purpose of this investigation. The main disk in the mobile devices is where the applications were installed by default.

Additionally, in our preliminary analysis, network traffic was acquired from mobile devices using Wireshark.[9] However, it was discovered that the traffic was either encrypted or encoded which caused it to be out of the scope of this research. The tool Fiddler[10] was another network traffic capturing tool used on these devices to decrypt HTTPS traffic. The initial goal was to intercept the 2FA secret keys as they were generated by the social media applications. Unfortunately, the traffic could not be captured with Fiddler. This was likely due social media applications using certificate pinning (Telerik, 2014), which can block Fiddler-generated certificates. Fiddler was successfully used to monitor traffic from the 2FA applications themselves, but this produced nothing of value as these applications perform all operations related to calculating the OTPs locally (M'RaihiBellare, 2005). In terms of the Windows VM, to speed up the process, disk images were not needed and only a file system acquisition of the target locations was implemented. Network traffic was acquired using Fiddler, and memory dumps were completed using iDumpIt[11] after each major test per application.

4. Disk analysis & experimental results

During the setup process of the 2FA applications, it was

[2] https://play.google.com/store/games?hl=en&gl=US.
[3] https://www.apple.com/app-store/.
[4] https://apps.microsoft.com/store/apps.
[5] https://winauth.github.io/winauth/download.html.
[6] https://authy.com/download/.
[7] https://www.magnetforensics.com/resources/magnet-acquire/.
[8] https://www.doubleblak.com/.
[9] https://www.wireshark.org/.
[10] https://www.telerik.com/fiddler.
[11] https://www.magnetforensics.com/resources/magnet-dumpit-for-windows/#Our%20Blog.

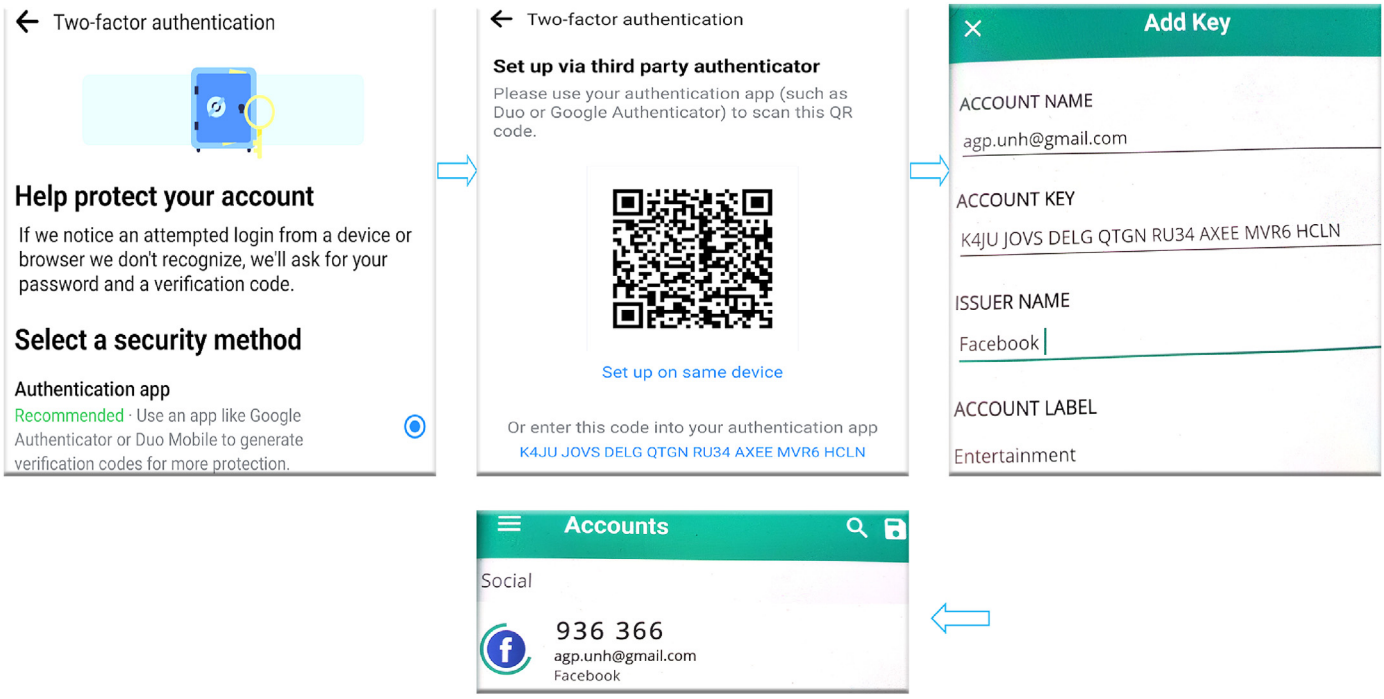


Fig. 1. Authentication process.

discovered that some of them had the option to lock the application via a four-digit pin or another type. Locking the application provided a layer of security to avoid unauthorized access. In some cases, this extra layer also encrypted the information stored in the device. In this phase, only four applications had this option and they were tested with and without assigning a pin. These configurations yielded different results which are highlighted in Table 3.

A granular breakdown of where these artifacts are stored in the device is presented in Table A.5, Appendix A. The most relevant results extracted from all devices in all tested applications are discussed in the following subsections. Refer to Table 2 for a summary of the number of applications containing this important information.

Consequently, it is critical to mention that twenty-five artifacts were discovered to store the most important information. Out of twenty-five artifacts, thirteen (50%) were extracted from the Android device and the remaining were extracted from the iOS device and the Windows VM.

4.1. Account information

Account information is extremely useful to find as it identifies the user of the account. During the set-up process of the 2FA applications and while adding a service to authenticate, certain information was required. Out of the fifteen applications examined, twelve (80%) stored account names in plain text (see Fig. 2). The account name could have been anything the user decided to name it, such as a username, email address or simply the name of the service being authenticated. Moreover, out of the fifteen applications tested, fourteen (93%) returned the issuer name of the application being authenticated, for example Facebook or Instagram. Other similarities found include the Universally Unique Identifier (UUID) which was stored in five out of the fifteen (33%) applications tested. The UUID serves as an identifier for each social media application tested with 2FA. Timestamps were also found in eight out of the fifteen (53%) applications tested. These timestamps were found across all three devices and presented the date and time that an account, such as Facebook, was added to the 2FA application.

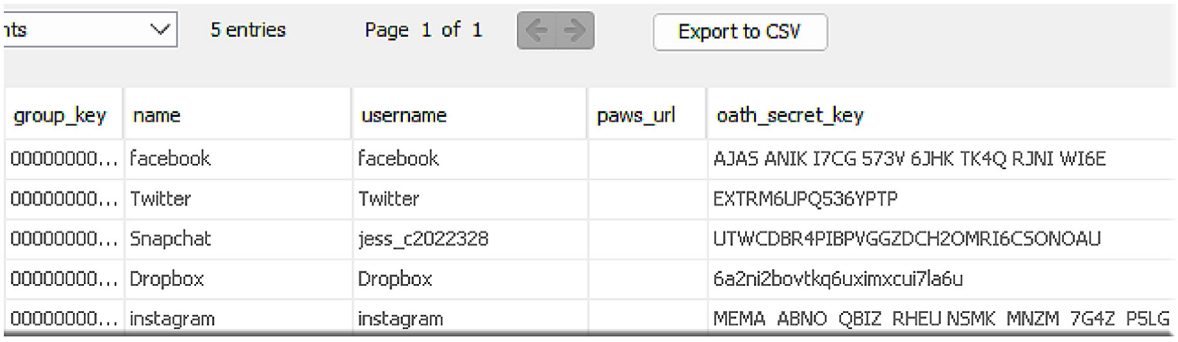
Additionally, the Twilio Authy application was the only one that required a phone number to be entered as part of the initial 2FA account setup. The phone number was found within the disk image of the Android device. Other important information found in the disk of the Windows VM stored by the Twilio Authy application included timestamps and identifiable information about the device such as IP addresses, region, city, and country. It also contained information about all of the devices registered under the phone number that was given at the creation of the account. All incidences of when Twilio Authy was downloaded and used with the particular phone number given while testing were logged in this central file. Each device logged in this application was given a unique identification number(UID).

Table 2
2FA artifact commonalities.

Artifact	Total App Number	Percentage
Issuer Name	14/15	93%
Account Name	12/15	80%
Secret Key	10/15	67%
Timestamps	8/15	53%
Encrypted Secret Key	7/15	47%
UUID	5/15	33%
Email	5/15	33%
Application Pin	2/15	13%
IP Address	2/15	13%
Location	2/15	13%

4.2. Secret/encrypted keys

Encrypted and non-encrypted secret keys were stored in fourteen out of the fifteen (93%) 2FA applications tested. The keys were found in either an encrypted format or in plain text (see Fig. 2). Any



nts 5 entries Page 1 of 1 Export to CSV

group_key	name	username	paws_url	oath_secret_key
00000000...	facebook	facebook		AJAS ANIK I7CG 573V 6JHK TK4Q RJNI WI6E
00000000...	Twitter	Twitter		EXTRM6UPQ536YPTP
00000000...	Snapchat	jess_c2022328		UTWCDBR4PIBPVGGZDCH2OMRI6CSONOAU
00000000...	Dropbox	Dropbox		6a2ni2bovtkq6uximxcui7la6u
00000000...	instagram	instagram		MEMA ABNO QBIZ RHEU NSMK MNZM 7G4Z P5LG

Fig. 2. Artifact from Microsoft Authenticator.

encrypted keys found were only decrypted if a pin was found to do so, any other attempts to perform advanced decryption were out of the scope of this paper.

For instance, the Twilio Authy application stored both, the plain-text and encrypted keys. Twilio Authy was one of the four applications that had the option of using a pin to lock the application. This pin also encrypted the secret keys.

Tests on the other three applications that had the option to use a pin rendered different results from the Twilio Authy application. Initially, it was believed that by setting a pin the information in the artifacts would be encrypted immediately, but this was not the case. It was discovered that six artifacts found in the disk stored encrypted keys without the need for a pin to be set. These six artifacts were spread between three applications. The 2FAS application for example, allowed for a pin to be set, but returned encrypted keys with and without the pin. The WinAuth application also did not require a pin to be set, and when the disk was analyzed, encrypted secret keys were found to be stored. Meanwhile, the application Aegis, allowed the user to set a pin and rendered only encrypted keys when the pin was in use, compared to plain text secret keys when the pin was not employed.

5. Windows memory & network results

As previously discussed, only the Windows VM was used to acquire its memory and capture the network traffic after each test from the three applications. Two memory captures were taken for each application, one while the 2FA application was opened during the testing process and one when the application was completely closed in the system. The idea was to test the concept of verifying that when the application is opened in the system and running, traces would be left in memory and more results would be found. However, the case might not be the same when the application is closed, and thus we wanted to see what information would remain in memory after the fact. This was proven to be true as the majority of the significant results that were found came from the memory capture that was taken while the application was opened.

When inspecting the memory and the network, there was no ambiguity about what had to be found. In terms of analyzing the memory, the tool Bulk Extractor was initially used, however, it did not provide any targeted relevant artifacts. On the other hand, Strings was the preferred tool to use as it allowed to search for targeted text strings in the memory dumps. This included secret keys and more.

Finally, network forensics was performed on the Windows VM by capturing the traffic with the Fiddler tool. The tool not only successfully decrypted HTTPS traffic, but also made it very convenient to search for the exact information that needed to be found in the network packets, such as secret keys. The results found are presented in Table 3.

5.1. Personally identifiable information(PII)

The memory and network both returned PII. In the memory, this included timestamps, location data, IP addresses, phone numbers, and email addresses. The network returned a phone number. Across the three applications tested on the desktop, the phone numbers and email addresses were returned for all of them. The timestamps, location data, and IP addresses were only returned for Twilio Authy.

5.2. Secret/encrypted keys

The secret key created by the application being authenticated (i.e. Facebook) was stored unencrypted in the memory and over the network. Encrypted secret keys were only found in the memory for Twilio Authy and WinAuth. In all three of the applications tested on the Windows VM, the plain-text secret keys were only discovered for the Facebook application being authenticated and not the rest of the applications tested.

6. Bypassing 2FA with TOTP

Although 2FA has proven to be successful in providing a second layer of security to user accounts, we have previously established that it is not completely secure against all threats. Take the case of Comcast Xfinity, where, according to several media outlets, hackers were able to bypass the 2FA mechanism on user accounts. It is believed the bad actors implemented credential stuffing attacks in order to establish login credentials for Xfinity accounts. Then they used a privately circulated OTP bypass tool to falsify 2FA verification requests that successfully allowed the 2FA to be bypassed and provided access to user accounts (Abrams, 2022).

In order to understand 2FA TOTP, it is first necessary to briefly introduce the Hash-Based Authentication Code (HMAC), the fundamental underlying algorithm that powers TOTP. HMAC uses a function that hashes input data and outputs a hash of fixed length, but unlike a standard hashing function such as SHA256, HMAC also hashes a secret key value along with the input data. The hash created using the data and secret key is a form of verification for two-factor authentication alongside a standard password (KrawczykBellare, 1997).

TOTP is a form of HMAC One Time Password (HOTP), which uses the current time (UTC or GMT) as input data hashed alongside a secret key to create the one-time password used for 2FA. In most 2FA applications, this takes the form of a 6-digit one-time password

Table 3
2FA important artifacts found across all acquisition types.

	Issuer Name	Account Name	Email	Secret Keys	Timestamps	Encrypted Secret Keys	Salt	Phone Number	Application Lock Pin
Aegis Locked				A			A		
Aegis Unlocked	A	A	A	A					
Authenticator+	I	I		I	I				
Epic Authenticator		I		I					
FIS Authenticator	I	I		I					
FreeOTP	A	A				A			
Google Authenticator	A	A			A	A			
IBM Verify	I	I		I					
Microsoft Authenticator	A	A		A					
Okta Verify	I	I							
TOTP Locked	A	A	A		A	A			A
TOTP Unlocked	A	A	A		A	A			
Twilio Authy Locked	A		A	A	A	A	A	A	A
Twilio Authy Unlocked	A		A	A	A/W	A	A	A	
Twilio Authy Memory*	W		W			W		W	
Twilio Authy Network*	W		W	W					
Two-Factor	I	I		I	I				
WinAuth	W			W	W				
WinAuth Memory*	W		W			W			
WinAuth Network*	W			W					
2FAS Locked					A	A			
2FAS Unlocked	A/I	A/I			A	A			
2 Factor Authenticator Memory*	W	W	W	W		W			
2 Factor Authenticator Network*	W			W					

Key: A: Android Mobile, I: iOS Mobile, W: Windows, Memory* or Network*: Memory or network acquisition only.

(OTP). This password is recalculated every 30 s using the new current time variable. The password is calculated approximately simultaneously by both the 2FA application of the user and the service (such as social media application) on which they have activated 2FA. Back in 2011, TOTP was the most common and popular form of 2FA application (M'RaihiMachani, 2011) and it still continues to be highly prevalent today, with Google Authenticator, Twilio Authy, and Microsoft being the most used TOTP applications. However, in recent years, push notification applications such as Duo Mobile have become the most popular 2FA Authentication method, and as of 2019, it makes up 68% of 2FA users (DataProt, 2021).

When a user enables 2FA on a service, such as a social media account, the user is provided with a secret key wrapped inside a QR code, alongside additional account metadata. This QR code can then be scanned or decoded, and the secret key is then loaded into the 2FA application, which can then begin calculating the TOTP. Some services also support giving the secret key directly without a QR code, which is the option used in this research.

Because all TOTP applications use the same HMAC algorithm (an RFC under review by the IETF), the theory was that if a plain-text HMAC secret key could be forensically extracted from a disk image, researchers could input that key into any TOTP application, produce an identical one time password to that of the original, and use it to gain access to a 2FA protected social media account. The work produced in (Ozkan and Bicakci, 2020) established that it is possible to extract a valid secret key from an application's file system using reverse engineering tools. These extracted secret keys could then be used to log into a 2FA-protected service.

The general method of copying TOTP through acquiring the secret key is one that the security community has been aware of since the creation of HMAC (M'RaihiMachani, 2011), which is why IETF urges developers to protect and encrypt their secret keys in any system that uses TOTP (such as a 2FA application). The work laid out in (Ozkan and Bicakci, 2020) showed where these keys are stored and how to extract them using reverse engineering and memory forensic techniques. We sought to learn from their methodology but from the angle of disk, memory and network forensics (with an emphasis on artifact location), and extend the scope of the study to not only Android applications but also iOS and Windows, all while testing additional applications on all three platforms.

Therefore, we developed the following proof of concept to verify the validity of output OTPs from different 2FA applications using extracted keys from different acquisition methods. This concept was tested with multiple 2FA applications and different secret keys to prove that using any key created by the same algorithm can bypass 2FA on a user account, even if it was not created with the same user account and application. See Fig. 3 for a high-level demonstration along with the steps below.

- Use any unencrypted secret key that was forensically extracted from either the disk images, network traffic, or memory from each application.
- Input each secret key into a different 2FA application, not the one originally used to create the key.
- This new 2FA application will now generate a valid 2FA OTP code every 30 s.

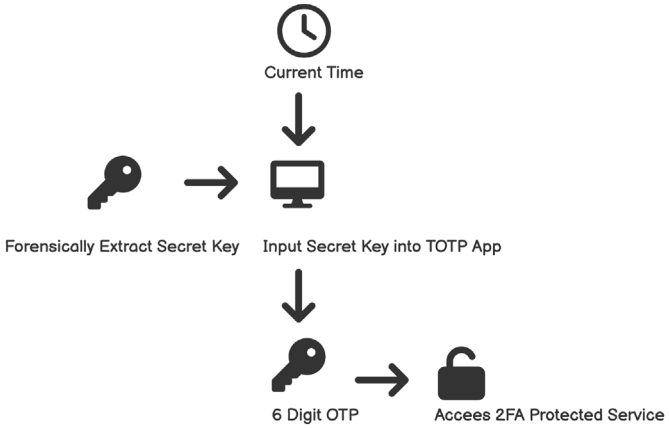


Fig. 3. 2FA bypass process.

- Use the generated OTP/code to access 2FA protected service/account.

The challenges to this 2FA bypass primarily have to do with encryption. If a 2FA application encrypts its secret keys, either by default or as an optional setting, those keys first need to be decrypted before they are used as it is the precise series of characters (then hashed via HMAC) which generates the OTP (M'RaihiMachani, 2011). Another challenge lies in verifying the legitimacy of the secret key. If 2FA has been disabled, or disabled and then re-enabled, the secret key will have been changed by the service; thus the 2FA applications key will be out of date and no longer be generating valid OTPs which can be used for authentication.

6.1. TOTP bypass implementation & results

In order to verify the 2FA bypass could still be valid, tests were conducted using TOTP secret keys created through Twitter and Facebook accounts. The keys were then loaded into a selection of Android, iOS, and Windows 2FA apps that we already tested for artifact extraction.

The one-time password generated on the applications were then compared to one another across all devices and platforms to verify they were identical. Once verified, each account was logged onto each of the 2FA-protected social media accounts to verify the one-time password was valid.

Consequently, the 2FA bypass was executed successfully with identical results on all 2FA applications tested. All OTPs generated were identical if using the same secret key. This in turn allowed us to use known credentials to log into the 2FA-protected social media accounts. It is important to note that in order for the bypass to be successful, the bypasser must have the target's login credentials in order to authenticate. Credentials could be obtained in different ways, for instance by extracting them from disk or even memory. Some login information was discovered on our disk investigation, such as username or email address. Another example could include credential stuffing as mentioned in (Abrams, 2022). This proof of concept assumes the algorithm creating the keys was identical in all applications and that, if given identical input, keys would output identical OTPs.

From a criminal perspective, to successfully perform this bypass locally, and assuming the target device is securely locked, the attacker would to have obtain access to the target's mobile or desktop device in order to acquire the 2FA secret key through the file system (if accessible) or forensic acquisition of an image.

The far more practical and concrete application for this methodology is with regard to forensic examiners. The ability to extract a 2FA secret key allows the examiner to generate a valid 2FA OTP without the use of the original device thereby reducing the need to directly handle the device and potentially altering the data. By having your own copy of the key, the original device can be transported, handed over to other examiners, or stored away and you will still have the valid credentials needed to access the 2FA-protected service. This applies even in a disastrous scenario in which the original device is wiped or destroyed. As long as the 2FA feature of the protected service has not been disabled, the extracted key will remain valid for authentication.

7. Conclusion/discussion

With an increase in the digitization of different sectors, including critical institutions, two-factor or multi-factor authentication has been established as a standard for securing information. Still, it is always better to use multiple layers of security than nothing at all. Two-factor authentication is a large step towards having a more secure environment in a user's personal and professional life. Nevertheless, while these technologies reduce risk, the research discussed in this paper demonstrates that more work needs to be done to improve such technologies and that organizations need to strongly consider using encryption when transferring and storing user data.

The forensic analysis performed on these applications (Table 1) discovered an abundant amount of user or account information stored in plain text or encrypted on disk in the device and some in the memory and network traffic of the Windows VM. The diversity of devices used with different operating systems offered a better insight into how these different technologies store and transfer user data. Our results demonstrated that out of the fifteen applications tested, Android devices contained the most user/account information overall with about (50%). In terms of overlap between the applications tested, only Twilio Authy was tested on all three devices with no other overlap between the Windows machine and the mobile phones. The iPhone and Android devices did see more overlap with four applications in total being tested on both. The results included account names (80%), issuer/service name (93%), IP addresses (13%), locations (13%), four-digit pins (13%) and most importantly, secret keys (73%). All of these artifacts can be freely accessed and downloaded from the Artifact Genome Project.[12]

Consequently, the discovery of plain text secret keys allowed us to bypass 2FA and gain access to an account without needing the device from which the two-factor authentication was originally set-up on. From a criminal perspective, in order to bypass a 2FA-protected application, a criminal would need to launch a certain type of attack to be successful as highlighted in Grimes (2019). As previously mentioned the practical use of this bypass would have a larger appeal to forensic examiners. This bypass could be a critical method to use in the event law enforcement needed to triage a criminal event where access to a user's account was the main goal.

The prevalence of PII in plain text within these 2FA applications should be noted. While the information collected in these applications is necessary for their functionality, the lack of secure storage of the data is a privacy and security concern. In the case that these applications are exploited by an external attacker, it could lead to the identification of users and the compromise of their accounts. It is recommended that companies and developers making 2FA applications take steps to secure their users' data by way of encryption.

Expanding further, this is why it is not necessarily invalid to explore the plain text artifacts as opposed to trying to seek out encrypted data and decrypt it. Not only are encryption algorithms difficult to break, requiring many hours and sophisticated computer technology, but many companies do not encrypt a lot of their information to begin with. Instead of focusing on a practice that is seldom used, we focused on the plain text data because of its prevalence within the applications.

8. Future work

Work must continue in terms of analyzing future versions of the applications originally investigated, as well as new applications especially to verify that perhaps more attention is being put on encrypting secret keys and PII in storage. Research could involve implementing the different types of attacks discussed in this paper. Furthermore, focus should be put on the other types of multi-factor authentication as they were not explored in this research.

Much research can be done in this field, as prior studies have

[12] https://agp.newhaven.edu.

focused on the usability or specific attacks that can be performed on it. With this future research, law enforcement may have a new way to collect evidence that can be useful in their investigations that have a 2FA element.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Number 1900210. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Appendix A. Apparatus & Relevant Artifacts Found in Disk

Table A.4
Apparatus

Hardware/Software	Use	Company	Software Version
Galaxy S6	2FA Accounts	Samsung	7.0
iPhone 7	2FA Accounts	Apple	14.7.1
Android Debug Bridge (ADB)	Data Extract	Android Studio Developers	N/A
Magnet Acquire	Data Extract	Magnet Forensics	2.56.0.31667
FTK Imager	Data Extract	AccessData	4.7.1
ArtEx	Data Extract	DoubleBlak	2.4.12
Autopsy	Data Analysis	Basis Technology	4.19.3
Wireshark	Network Extract/Analysis	Wireshark, Inc	3.6.6
Fiddler	Network Extract/Analysis	Telerik, Inc	v5.0.20211
DB Browser for SQLite	File Analysis	DB	N/A
Instagram	2FA Testing	Meta Platforms, Inc.	254.0.0.19.109
Facebook	2FA Testing	Meta Platforms, Inc.	386.0.0.35.108
Twitter	2FA Testing	Twitter, Inc.	9.61.0-release.0
Dropbox	2FA Testing	Dropbox, Inc.	298.2.2
Snapchat	2FA Testing	Snap, Inc.	12.01.0.33
VMware Fusion	Platform to Host VM	VMware, Inc.	12.2.4
Windows 10 Education	Windows VM for Testing	Microsoft, Inc.	21H2
Bulk Extractor	Memory Analysis	N/A	2.0.0
Linux Strings Command	Memory Analysis	N/A	N/A

Table A.5
Important Data Path Directories and Files Found in Disk Across Devices

File ID	Path	App	OS	Description
1.1	/vol_vol20/data/com.beemdevelopment.aegis/files/aegis.json	Aegis*	Android	Encrypted Secret key, Salt & Nonce, UUID
1.2	/vol_vol20/data/com.beemdevelopment.aegis/files/aegis.json	Aegis	Android	Secret Key, Issuer Name, Account Name, UUID
1.3	/vol_vol20/data/org.fedorahosted.freeotp/shared_prefs/tokens.xml	FreeOTP	Android	Account Name, Issuer Name, and Encrypted secret keys
1.4	/vol_vol20/data/com.google.android.apps.authenticator2/databases/databases	Google	Android	Account Name, Issuer Name, and Secret Keys
2.1	/vol_vol20/data/com.azure.authenticator/databases/PhoneFactor	Microsoft	Android	Account Name, Issuer Name, Secret Keys
2.2	/private/var/mobile/Containers/Data/Application/72DAEA2F-0067-4D16-882F–03197441C97C/Documents/PhoneFactor.sqlite	Microsoft	iOS	UUID
2.3	/vol_vol20/data/com.authenticator.authservice2/shared_prefs/TOTP_Authenticator_Preferences.xml	TOTP*	Android	2FA Pin, Issuer Name, Account Name, and Secret key
2.4	/vol_vol20/data/com.authenticator.authservice2/shared_prefs/TOTP_Authenticator_Preferences.xml	TOTP	Android	Issuer Name, Account Name, and Secret key
3.1	/vol_vol20/data/com.twofasapp/shared_prefs/SecurePrefernces.xml	2FAS*	Android	Encoded Pin
3.2	/private/var/mobile/Containers/Shared/AppGroup/786DC57B-8EA6-4BC5-9436-C6F302CCF5C4/TwoFAS.sqlite	2FAS	iOS	Account Name, Issuer Name, Timestamps, and Secret Keys
3.3	/vol_vol20/data/com.twofasapp/shared_prefs/com.twofasapp_preferences_encrypted.xml	2FAS+	Android	Encrypted secret keys
3.4	/vol_vol20/data/com.authy.authy/databases/authy_database	Authy	Android	Private RSAKey
4.1	/vol_vol20/data/com.authy.authy/shared_prefs/com.authy.authy.storage.UserInfoStorage.xml	Authy	Android	User phone number, email
4.2	/vol_vol20/data/com.authy.authy/shared_prefs/com.authy.storage.authenticator_password_manager.xml	Authy*	Android	Timestamp 2FA password
4.3	/vol_vol20/data/com.authy.authy/shared_prefs/com.authy.storage.tokens.authenticator.xml	Authy	Android	Issuer name, Salt, and Secret/Encrypted keys
4.4	/vol_vol20/data/com.authy.authy/shared_prefs/tokenConfig.xml	Authy	Android	Issuer name
5.1	/private/var/mobile/Containers/Data/Application/EC0F1491-9034-436B-8AB9-B75D578DFDF2/Library/Preferences/com.okta.mobile.plist	Okta	iOS	Account Name and Issuer Name
5.2	/private/var/mobile/Containers/Data/Application/9AEE80C4-6616-426A-AB13-896D6F13B046/Library/Application Support/Main_sqlite	Two-Factor*	iOS	Account Name, Issuer Name, Timestamps, UUID, and Secret Keys
5.3	/private/var/mobile/Containers/Data/Application/E2444C86–B1B9–43C6–B99E-42F6DF8DCF1F/Library/LocalDatabase/fisauth.db	FIS	iOS	Account Name, Issuer Name, and Secret Keys
5.4		Epic	iOS	Account Name and Secret Keys

Table A.5 (*continued*)

File ID	Path	App	OS	Description
	/private/var/mobile/Containers/Data/Application/42F6A8C9-8458-4386-9C28-D0919FD4A4A7/Documents/tokens			
6.1	/private/var/mobile/Containers/Data/Application/F651A073-ECEA-4D25-9B3C−B82A71FAE610/Library/Application Support/61D927A9-CC77-45BD-B4C9-43CE70925E15	IBM	iOS	Account Name, Issuer Name, and Secret Keys
6.2	/private/var/mobile/Containers/Data/Application/F651A073-ECEA-4D25-9B3C−B82A71FAE610/Library/Application Support/ADAD21C0-FA72-460D-8744-195E5B31AD09	IBM	iOS	Account Name, Issuer Name, and Secret Keys
6.3	/private/var/mobile/Containers/Data/Application/37D75154-4AAA-45FD-9D43-9C533DB10A5F/Library/LocalDatabase/auth	Auth+	iOS	Account Name, Issuer Name, Timestamps, and Secret Keys
6.4	C:\Users\"Username"\AppData\Roaming\Authy Desktop\Local Storage\leveldb\000004	Authy	Windows	Locations, Device Type, Device Name, Timestamps, and Secret Keys
7.1	C:\Users\"Username"\AppData\Roaming\Authy Desktop\IndexedDB\file_0.indexeddb.leveldb\000003	Authy	Windows	Registration Status, Registration Time, Refresh Token
7.2	C:\Users\"Username"\AppData\Roaming\WinAuth\winauth.xml	WinAuth	Windows	Social Media Name, ID, Time Created, Secret Key

Key: Artifacts from Pin Activated Applications *.

References

Abrams, L., 2022. Comcast xfinity accounts hacked in widespread 2fa bypass attacks. https://www.bleepingcomputer.com/news/security/comcast-xfinity-accounts-hacked-in-widespread-2fa-bypass-attacks/. **URL:**.

Al Mutawa, N., Baggili, I., Marrington, A., 2012. Forensic analysis of social networking applications on mobile devices. Digit. Invest. 9, S24−S33.

America, N. (2022), 'What is two-factor authentication (2fa)?'. URL: https://www.newamerica.org/in-depth/getting-internet-companies-do-right-thing/case-study-2-offering-two-factor-authentication/#:~:text=Though%20there%20have%20been %20several,a%20key%20fob%20in%201986.

Childers, D., 2021. State of the auth 2021. https://duo.com/assets/ebooks/state-of-the-auth-2021.pdf. URL:

Clark, D.R., Meffert, C., Baggili, I., Breitinger, F., 2017. Drop (drone open source parser) your drone: forensic analysis of the dji phantom iii. Digit. Invest. 22, S3−S14.

DataProt, 2021. Two-factor authentication statistics. DataProt. **URL:**. https://dataprot.net/statistics/two-factor-authentication-statistics/.

De Cristofaro, E., Du, H., Freudiger, J., Norcie, G., 2013. A Comparative Usability Study of Two-Factor Authentication. *arXiv preprint arXiv:1309.5344*.

Dimitrienko, A., Liebschen, C., Rossow, C., 2014. Security analysis of mobile two-factor authentication schemes. Intel Secur. J. 18 (4), 138−161.

Dorai, G., Houshmand, S., Baggili, I., 2018. I know what you did last summer: your smart home internet of things and your iphone forensically ratting you out. In: Proceedings of the 13th International Conference on Availability, Reliability and Security, pp. 1−10.

FBI, 2023. Internet crime report 2022. https://s3.documentcloud.org/documents/23707016/2022_ic3report.pdf. **URL:**.

Fruhlinger, J., 2019. 2fa explained: how to enable it and how it works. https://www.csoonline.com/article/3239144/2fa-explained-how-to-enable-it-and-how-it-works.html. URL:

Gope, P., Sikdar, B., 2019. Lightweight and privacy-preserving two-factor authentication scheme for iot devices. IEEE Internet Things J. 6 (1), 580−589.

Gordin, I., Graur, A., Potorac, A., 2019. Two-factor authentication framework for private cloud. In: 2019 23rd International Conference on System Theory, Control and Computing (ICSTCC), pp. 255−259.

Grajeda, C., Sanchez, L., Baggili, I., Clark, D., Breitinger, F., 2018. Experience constructing the artifact genome project (agp): managing the domain's knowledge one artifact at a time. Digit. Invest. 26, S47−S58.

Grimes, R., 2019. 12+ ways to hack multi-factor authentication. https://www.knowbe4.com. https://www.knowbe4.com/hubfs/12+_Ways_to_Hack_Two-Factor_Authentication-1.pdf. URL:

Johnson, H., Volk, K., Serafin, R., Grajeda, C., Baggili, I., 2022. Alt-tech social forensics: forensic analysis of alternative social networking applications. Forensic Sci. Int.: Digit. Invest. 42, 301406.

Kim, G., 2022. Making you safer with 2sv. https://blog.google/technology/safety-security/reducing-account-hijacking/. URL:

Konoth, R.K., Fischer, B., Fokkink, W., Athanasopoulos, E., Razavi, K., Bos, H., 2020. Securepay: strengthening two-factor authentication for arbitrary transactions. In: 2020 IEEE European Symposium on Security and Privs, pp. 569−586.

Krawczyk, Bellare, C., 1997. Hmac: keyed-hashing for message authentication. RCF. **URL:**. https://www.ietf.org/rfc/rfc2104.txt.

Mahr, A., Cichon, M., Mateo, S., Grajeda, C., Baggili, I., 2021. Zooming into the pandemic! a forensic analysis of the zoom application. Forensic Sci. Int.: Digit. Invest. 36, 301107.

Mahr, A., Serafin, R., Grajeda, C., Baggili, I., 2022. Auto-parser: Android auto and apple carplay forensics. In: 'Digital Forensics and Cyber Crime: 12th EAI International Conference, ICDF2C 2021, Virtual Event, Singapore, December 6-9, 2021, Proceedings'. Springer, pp. 52−71.

Mann, C., Loebenberger, D., 2017. Two-factor authentication for the bitcoin protocol. Int. J. Inf. Secur. 16 (2), 213−226.

Microsoft, 2022. What is two-factor authentication? https://www.microsoft.com/en-us/security/business/security-101/what-is-two-factor-authentication-2fa. URL:

M'Raihi, Bellare, H.N.R., 2005. Hotp: an hmac-based one-time password algorithm. RCF. **URL:**. https://datatracker.ietf.org/doc/html/rfc4226.

M'Raihi, Machani, P.R., 2011. Totp: time-based one-time password algorithm. RCF. **URL:**. https://www.rfc-editor.org/rfc/rfc6238.

Ozkan, C., Bicakci, K., 2020. Security analysis of mobile authenticator applications. In: '2020 International Conference on Information Security and Cryptology (ISCTURKEY)', pp. 18−30.

Reese, K., Smith, T., Dutson, J., Armknecht, J., Cameron, J., Seamons, K., 2019. A usability study of five two-factor authentication methods. In: Proceedings of the Fifteenth Symposium on Useable Privacy and Security.

Sitterer, A., Dubois, N., Baggili, I., 2021. Forensicast: a non-intrusive approach & tool for logical forensic acquisition & analysis of the google chromecast tv. In: 'The 16th International Conference on Availability, Reliability and Security', pp. 1−12.

Telerik, 2014. Http(s) sniffing doesn't work in some app. Tele. URL: https://www.telerik.com/forums/http-s-sniffing-doesn't-work-in-some-app.

Yarramreddy, A., Gromkowski, P., Baggili, I., 2018. Forensic analysis of immersive virtual reality social applications: a primary account. In: '2018 IEEE Security and Privacy Workshops (SPW)', IEEE, pp. 186−196.