This Selfie Does Not Exist: On the Security of Electroneum Cloud Mining

Alexander Marsalek[1], Edona Fasllija[2] and Dominik Ziegler[3]
[1]*Secure Information Technology Center Austria, Vienna, Austria*
[2]*Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Graz, Austria*
[3]*Know-Center GmbH, Graz, Austria*

Keywords: Electroneum Cloud Mining, Cryptocurrency, Impersonation Attack, Image Manipulation.

Abstract: The Electroneum cryptocurrency provides a novel mining experience called "cloud mining", which enables iOS and Android users to regularly earn cryptocurrency tokens by simply interacting with the Electroneum app. Besides other security countermeasures against automated attacks, Electroneum requires the user to upload selfies with a predefined gesture or a drawing of a symbol as a prerequisite for the activation of the mining process. In this paper, we show how a malicious user can circumvent all of these security features and thus create and maintain an arbitrary number of fake accounts. Our impersonation attack particularly focuses on creating non-existing selfies by relying on Generative Adversarial Network (GAN) techniques during account initialization. Furthermore, we employ reverse engineering to develop a bot that simulates the genuine Electroneum app and is capable of operating an arbitrary number of illegitimate accounts on one Android device, enabling the malicious user to obtain an unfairly large payout.

1 INTRODUCTION

During 2017, the first and most popular cryptocurrency, Bitcoin, experienced an astonishing 13-fold increase in value (Corcoran, 2017). Since then, cryptocurrencies, as well as the underlying blockchain technology, have received increasing attention from both academia and industry. Nonetheless, several challenges related to the scalability and volatility of cryptocurrencies, energy consumption during mining, and the technological literacy required to fully comprehend the enabling technology have put a strain on their mass adoption as a mainstream form of payment.

The traditional process of cryptocurrency mining —i.e. verifying transactions in a blockchain— involves solving difficult cryptographic problems that call for significant computational power and often specialized hardware. Subsequently, miners seeking to increase their chances of solving these puzzles and getting rewards, choose to invest in powerful application-specific integrated circuit (ASIC) mining devices and oftentimes combine their computational resources over a network to form what is called a mining pool (Frankenfield, 2019). However, this mining approach is not considered viable in resource-constrained devices such as smartphones, as it can interfere with their overall performance and lead to battery damage or overheating (Comben and Rivet, 2019). Due to these effects, both Google and Apple prohibit cryptocurrency mining apps from their application stores (Apple, 2019; Google, 2018), limiting the accessibility of the average smartphone users to crypto mining. Despite this fact, several projects like Electroneum[1], Phoneum[2], Pi[3] and MIB[4] are committed to making mining possible on mobile devices. The mobile mining process of such projects mostly refers to the so-called "airdrop" of pre-mined token awards to the users of their dedicated mining app, sometimes based on the specifications of the phone. While this approach significantly relieves the strain put on the device, it introduces susceptibility to new device or app impersonation attacks, through which adversaries can emulate an arbitrary amount of accounts to illegitimately obtain tokens.

Electroneum was launched back in 2017 as an alternative cryptocurrency with a mobile-specific business model and use-case. The Electroneum team aimed at gaining widespread adoption by focusing on creating an easily-accessible, cryptocurrency-based mobile payment system that allows its users to store, send and receive digital coins via their smartphone alone. During its short history,

[1]https://electroneum.com/
[2]https://phoneum.io/
[3]https://minepi.com/
[4]https://www.mibcoin.io/

Marsalek, A., Fasllija, E. and Ziegler, D.
This Selfie Does Not Exist: On the Security of Electroneum Cloud Mining.
DOI: 10.5220/0009829303880396
In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications (ICETE 2020) - SECRYPT*, pages 388-396
ISBN: 978-989-758-446-6

the user basis of this cryptocurrency grew quickly up to over 3.4 million registered users (Electroneum, 2020). They initially released their *mobile mining* experience, which in essence operated as a stream of token rewards loosely based on the hashing power of the device. They later replaced mobile mining with *cloud mining* in order to make mining fairer for all mobile users, independent of their ability to always have Internet connectivity, and introduce new anti-bot technologies in their fight against fake Electroneum (ETN) miners.

This paper focuses on analyzing the security measures for device verification and authorization employed by the *new* Electroneum app (Section 2). We investigate on the possibility of bypassing these security measures to successfully exploit the *cloud mining* process. In order to do so, we exploit Electroneum's account creation and device authorization protocol to mount a device emulation and app impersonation attack that does only require minimal interaction during its initialization phase. (Section 3). Our findings demonstrate that the newly employed security mechanisms are still vulnerable to this kind of spoofing attacks and that it is possible to generate an unlimited number of illegitimate accounts that take advantage of ETN's cloud mining token rewards (Section 4). We further discuss potential countermeasures to ensure execution of the mining application on genuine mobile devices in order to mitigate protocol exploitation and app impersonation attacks in Section 5. We subsequently discuss how the contribution of our paper compares with related work in Section 6. Lastly, we conclude by investigating the possibility to extend the scale of our attack by fully automating the account creation protocol.

2 ELECTRONEUM

Electroneum's origin traces back to September 2017, when it first came to life as a Monero[5] fork via an Initial Coin Offering (ICO). Labeling itself as "the mobile-cryptocurrency", Electroneum was launched with the unique mission of allowing anyone with a smartphone and Internet connection to participate in the global economy. From the start, the Electroneum team focused on making digital payments accessible to the 1.7 billion unbanked people around the world, by making ETN mining as simple as downloading an app and creating an Electroneum account. Furthermore, with a total supply of 21 billion coins, the creators of ETN aimed for easier mining and spending of whole Electroneum coins in day-to-day purchases.

The strength of Electroneum stems from its user-friendly mobile application launched in March 2018, with which users can mine, send, receive and store ETNs. The virtual *mobile mining* —now called *cloud-mining*— process has the form of an airdrop of token awards to the users of the application.

The upcoming section goes into the details of the Cloud Miner app and the security features employed by Electroneum when verifying accounts and authorizing devices for cloud mining.

2.1 Cloud Mining Process

The Electroneum Cloud Miner application was created using the Apache Cordova framework[6]. As such, it is mainly implemented in Javascript. Users of this application are only required to create an Electroneum account to be able to start mining right away. When creating the account, the user must provide a valid email address and choose a password. To activate cloud mining, the user is prompted to enter information about their country and confirm their age. Once the user provides this information, they are requested to submit their first selfie, posing in a specific gesture or holding a predefined drawing, as shown in Figure 1. After the selfie is submitted, the cloud mining process is started and stays active for 7 days. This comprises one of the main features that differentiates between cloud mining and the former mobile mining. Cloud mining does no longer rely on the device being always connected to the Internet, but instead only poses the requirement of connecting to the Electroneum services once in 7 days to extend mining.

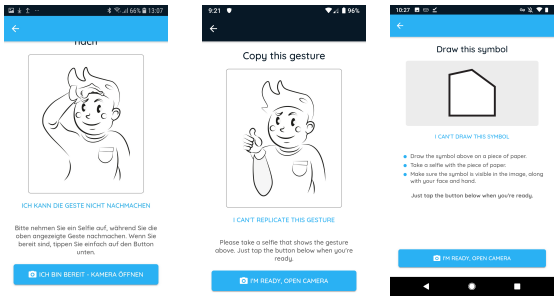
Creating an Electroneum account automatically triggers the creation of an Electroneum mobile wallet accessible via the app. To be able to store, send and receive ETN tokens, the user has to activate the wallet by completing a number of steps as follows: **First**, the user has to select a PIN as an additional factor of authentication for the app. **Second**, the user has to confirm their email address by entering a confirmation code sent by Electroneum. **Third**, following the email confirmation, the user is required to enter basic information such as their name and phone number. **Fourth**, the user has to acknowledge several security hints and warnings. **Fifth**, the user has to prove that they are a human and not a bot. For this, Electroneum uses a symbol-based CAPTCHA[7] test, as shown in Figure 2. The wallet is activated once the user solves the three consecutive CAPTCHAs. The user is rewarded with free ETN tokens for each verification step they complete.

Before the payout amount is reached, which is currently set to 100 ETN, the user is requested to submit a

[5]https://www.getmonero.org/

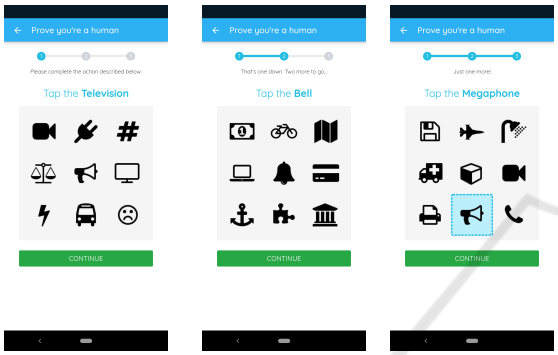
[6]https://cordova.apache.org

[7]Completely Automated Public Turing test to tell Computers and Humans Apart.



(a) (b) (c)

Figure 1: Symbols to draw and gestures to replicate during the two selfie creation steps.



(a) (b) (c)

Figure 2: The three CAPTCHAs to solve in order to activate the wallet.

second selfie. If both selfies are accepted the user will get the payout shortly after reaching the payout limit.

2.2 Security Features

Electroneum employs several client-side and server-side security mechanisms to prevent unauthorized usage of their API and the cloud mining application. The security features described in this section are particularly relevant for us, as they influence the design and methodology of our attack. These features were observed by first decompiling and decrypting the Android app, analyzing the network traffic and iteratively improving the attack.

Account Verification. During account creation, the user-provided email address is verified. In contrast to the mobile mining app, the phone number is no longer verified by sending an SMS-verification challenge.

CAPTCHAs. Another necessary step for the activation of an Electroneum account involves solving multiple CAPTCHAs to prove the user is human. Electroneum chooses to employ self-created CAPTCHAs, where the user is asked to tap a specific symbol, rather than relying on state-of-the-art CAPTCHAs like Google's reCAPTCHA[8].

[8]https://www.google.com/recaptcha

Maximum Number of Devices. Electroneum does not restrict logging into the same account with multiple devices. However, the number of devices does not influence the amount of tokens the user receives. In our experience, the performance and other metrics specific to a device, do not have an impact the amount of earned ETN tokens.

Device Authorization. In the process of analyzing the Electroneum API, we observed that a randomly generated device hash is assigned to each authorized device. In case the user wants to log in to a new device, an additional device authorization link is sent to the user's email address.

Selfie. To start cloud mining, it is necessary to submit a selfie with a predefined gesture or drawing, chosen by the server. Figure 1 shows three examples of selfie requirements. Figures 1a and 1b illustrate examples where the user is requested to take a selfie in a predetermined gesture, namely by touching their forehead or taking a thumbs up pose. In Figure 1c the user is requested to draw the symbol sent by the Electroneum server and take a selfie with the drawing. If a proposed gesture or drawing cannot be replicated, the user can request up to three new gestures or drawings. Every account needs to provide at least two selfies, one when the mining process is started for the first time and one before the first payout is received. If one or both of the submitted selfies do not get approved, the user is requested to create a new selfie and try again.

App Encryption. As a countermeasure against reverse engineering, Electroneum protects their application's source code via the "Cordova Crypt File Plugin"[9], which encrypts files during the build process and decrypts them at runtime.

Tor. Having different IPs and devices for each impersonated account is a desirable feature from the attackers perspective as it allows the attack to remain unnoticed for longer. The former Electroneum Mobile Mining service allowed access via Tor[10] IPs. Ziegler et al. (2018) exploited this vulnerability by setting fixed exit nodes per account. Electroneum now blocks exit node IPs for cloud mining.

Emulator Protection. The Electroneum app detects if it is executed on an Android Emulator and sends this information to the Electroneum server.

3 ATTACK

In this section, we will describe how an attacker can circumvent all security features and create an arbitrary number of fake miner accounts on a single device and thus earn an unfair amount of Electroneum tokens. The

[9]https://github.com/tkyaji/cordova-plugin-crypt-filext
[10]https://www.torproject.org

idea of the attack is to have an initial setup phase to create and activate the accounts that requires minimal interaction. Afterward, we rely on a completely automated cloud mining extension phase for which we develop a bot that emulates the Electroneum app and regularly maintains the accounts to earn Electroneum tokens.

3.1 Initial Setup

The initial setup phase mainly consists of creating the Electroneum accounts and activating cloud mining. The steps required for setting up an account are detailed under Section 2.1. While these steps could also be automated, they only have to be done once and we decided to put our efforts on automating the regular task of extending the mining process instead.

To obtain the email addresses for our generated accounts, we decided to use Gmail addresses, which we could easily create for free using the private mode of our browser with different IP addresses. This decision was also motivated by the several problems that we encountered with email activation or device authorization links when using disposable email providers. Furthermore, the initial setup of the fake accounts was facilitated by the fact that Electroneum does not verify mobile phone numbers via SMS-challenges or check for correctness of the data entered by the user at this point.

The next big challenge from the attackers perspective concerns the generation of fake selfies to be submitted for the activation of cloud mining. We delve into the details of how to create selfies of non-existing persons in the next section.

3.2 Selfie Creation

To allow one person or attacker, to create multiple accounts and minimize the detection risk, it is necessary to create selfies of different looking personas. We follow a three-step approach to generate selfies of non-existing persons: **First**, we start by generating a face of a non-existing person. For this, we use the generative adversarial network (GAN) called StyleGAN (Karras et al., 2019a) and the improved version StyleGAN2 (Karras et al., 2019b) created by Karrays et al.[11]. StyleGAN and StyleGAN2 generate photo-realistic pictures of non existing persons[12]. Figure 3 shows on the left side, the faces that have been generated using these GANs. **Second**, we take a selfie of us in the required gesture or with the requested drawing. **Third**, we swap the generated face with the one of our selfies. While many apps and tools provide a face swapping functionality we found the Microsoft App "Face Swap" (Microsoft, 2019) to yield the best results. The center and the right columns in Figure 3 display our results.

[11]In the beginning we used StyleGAN, but after the release of StyleGAN2 we switched to the improved version.

[12]Samples of images generated with StyleGAN2 can be viewed on the website https://thispersondoesnotexist.com.

3.3 Automated Attack

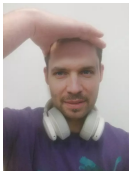
After creating the account and activating cloud mining, the user is supposed to regularly extend the mining by starting the Electroneum app and pressing the corresponding button. This can be done the earliest after 24 hours and the latest 7 days after the last extensions step. If not extended in this time period, the mining process is stopped. In this section we describe how this extension process can be fully automated for an arbitary number of accounts using a single device. For this, we implemented an Android app, which sends all required HTTP-requests to the Electroneum server. The bot app was implemented by reverse engineering the Electroneum app. Even though Electroneum employs app encryption, their encryption approach proved out to be insufficient as we discovered that the encryption key that is randomly generated at build time is then hard-coded into the Android application. An attacker can therefore easily extract it and decrypt the source code files. Algorithm 1 shows the necessary steps to extend the cloud-mining process in pseudo-code. First, it loads the stored account data, like username, password and PIN. Afterward, it generates device information, like serial number, model, etc. using a pseudo-random algorithm. This ensures that each account will always get the same values. The remainder of the algorithm follows the workflow of the official app. Line 5 verifies that no update is necessary and aborts otherwise. Next, in line 7 the algorithm checks, if a not expired JSON Web Token is stored for this account. If not, a new token is requested using a special refresh token, which is received after a successful login. If no refresh token is available the algorithm falls back to a login with the stored username and password and refreshes the JSON Web token afterward. After that, in line 17, a handshake token is requested. This call also verifies whether the cloud mining service is active. In line 18 the new `sessionData` is stored and finally the mining process is extended in line 20. A second, non printed, algorithm chooses the next account to be extended, ensuring, no account is being extended with an IP address used by one of our other accounts. If necessary a new IP address is retrieved by toggling the airplane mode.

4 RESULTS

In order to evaluate our approach, we created 5 accounts for non existing persons and let our bot maintain them.

(a) Generated face for Peter.

(b) Requested gesture for first selfie of Peter.



(c) First selfie for Peter.

(d) Requested symbol for second selfie of Peter.

(e) Second selfie for Peter.

(f) Generated face for Lea.

(g) Requested symbol for first selfie of Lea.

(h) First selfie for Lea.

(i) Requested symbol for second selfie of Lea.

(j) Second selfie for Lea.

(k) Generated face for Karl.

(l) Requested gesture for first selfie of Karl.

(m) First selfie for Karl.

(n) Requested symbol for second selfie of Karl.

(o) Second selfie for Karl.

Figure 3: Generated faces and selfies.

Algorithm 1: extendMining.

Input: UserData := ⟨username, password, pin⟩
Output: {*success*, *error*, *tryLater*}

```
#Derive device information (uuid, model, serial) from username using a pseudorandom alghorithm.
deviceData ← generateDeviceData(username)
sessionData ← loadSessionData(username)
#Check if update is necessary
if checkAppUpdateNecessary() = true then
    return failed
if tokenIsNullOrExpired() = true then
    if refreshTokenIsNotNull() = true then
        getTokenFromRefreshToken()
    else
        login()
        getTokenFromRefreshToken()
#Do a handshake to get the new handshake token
result ← doHandshake()
if result.getStatus() = error then
    return error
sessionData.handshake ← result.getHandshake()
store(sessionData)
#extend the mining process and return the result
return extendMiningHTTP(deviceData, sessionData)
```

Figure 3 shows the generated faces, requested gestures and drawings, as well as the submitted selfies. All of our accounts/selfies were accepted and received their payouts.

Figure 4 shows the pending balance over time for our test accounts and one control account that we maintained as a regular, honest user, using the official Electroneum app. Figure 4 shows that both, our legitimate account as wells as our fake accounts earn Electroneum tokens at the same rate. Figure 5 shows the payout of the reference account for a longer period of time. It is eye-catching that the Electroneum team reduced the amount of tokens earned per day in this time-span. In June 2019, it took 12 days to reach the payout limit of 100 ETN, while in December and January, it took already 42 days to reach the same amount. Figure 6 shows the received ETN tokens per week. All accounts received roughly the same amount per week. Furthermore, Figure 6 shows that the payout rate has been further reduced in week four and afterward slightly increased.

5 DISCUSSION

In this paper, we have shown that the cloud mining process employed by Electroneum is susceptible to spoofing attacks. We showcased how to circumvent all employed security mechanisms successfully. Our accounts have been active for several months without being detected.

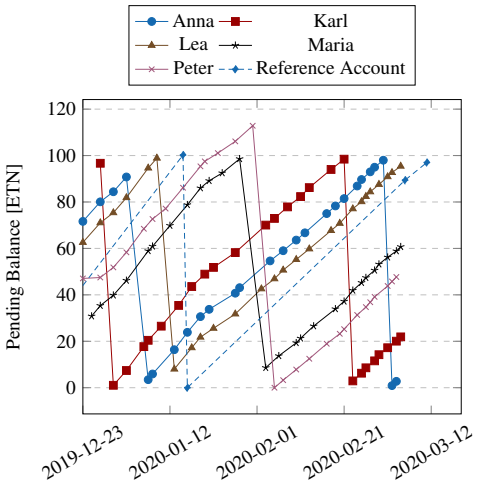


Figure 4: Pending balance over time. The payout threshold was 100 ETN.

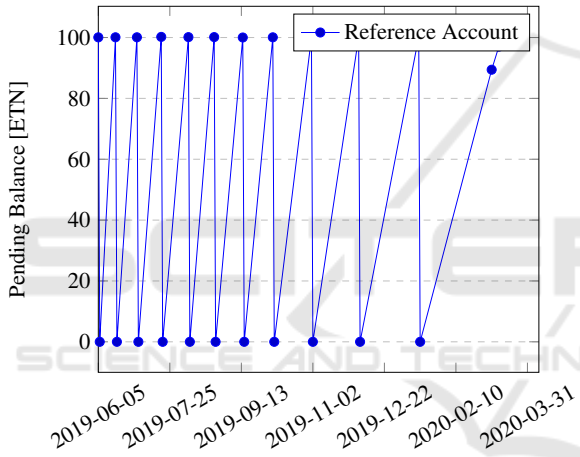


Figure 5: ETN pending balance over time.

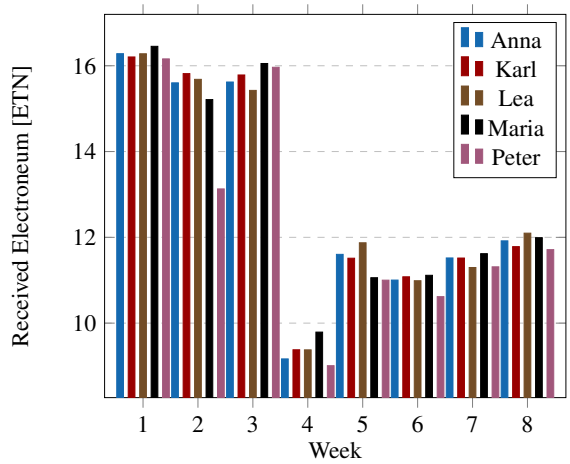


Figure 6: Received Electroneum per week.

Using our developed bot, an arbitrary number of accounts can be automatically maintained to earn a disproportional amount of ETN tokens. In this section we will discuss why this attack is possible and how it could be mitigated. The next section focuses on the identified issues and section 5.2 presents possible mitigations.

5.1 Issues

Based on these findings and the performed security analysis, we have identified four main issues: **First**, we found out that Electroneum does not verify phone numbers to protect against bots. Indeed, Electroneum requires a phone number during registration. The goal is to bind each account to a unique phone number. However, our tests revealed that the number entered is never checked. As a result, an arbitrary phone number can be used to create a new account. **Second**, Electroneum does not perform an identity check based on, e.g., government-issued documents. Admittedly, the application mandates to upload two selfies copying predefined gestures. To the best of our knowledge, these pictures are manually verified. Still, in our attack, we could circumvent this security mechanism with a dataset of (non-existent) human faces generated by a GAN. **Third**, no certificate pinning is used, which allows attackers to easily intercept or modify the network traffic using a standard man-in-the-middle attack. **Fourth**, the Electroneum app is not protected against manipulation, it does not check its integrity. Hence, the application can be modified at will or even be emulated entirely. In our attack, we could achieve both: we modified and recompiled the application, and emulated the client.

Summarizing, Electroneum suffers from several flaws that allow attackers to exploit their cloud mining process. However, while the first three problems could easily be fixed, by, e.g., enforcing phone number verification and more strict identity checks the fourth problem cannot easily be solved. We believe that the core problem does not lie in the cryptocurrency itself. Instead, the presented attack results from missing control over users' devices. Electroneum, or more generally speaking developers, have hardly any control over their application, once published. Ensuring unique devices and accounts can thus become a difficult endeavor. In fact, just recently Keybase[13], an end-to-end encrypted messenger suffered from a similar problem. Keybase offered a free airdrop program where they were giving out free crypto tokens for the Stellar[14] open-source protocol. Despite their efforts to limit the airdrop to benign users, it was quickly shut down due to issues with fake accounts. As the initiators put it, they were hit with fake accounts "far beyond the capacity of

[13] https://keybase.io
[14] https://www.stellar.org

Keybase or SDF to filter" (Keybase, 2019). In contrast to Electroneum, Keybase did not require the user to submit a selfie.

5.2 Mitigation

As stated, protecting the app against modifications cannot easily be prevented and heavily depends on the capabilities of the device and operating system. One solution, though, is to verify the integrity of mobile client applications at runtime remotely. Currently, only Android[15] provides such a mechanism via attestation, as shown by Prünster et al. (2019). Using Android's key attestation capabilities, it is possible to remotely establish trust in unmanaged environments and ensure that the app is executed on an unmodified, unrooted Android device. Furthermore it is possible to ensure that an unmodified copy of the application is executed and that only one Electroneum account per device can be used for mining. This approach needs no additional hardware or modifications to the operating system. A disadvantage of this approach is that not yet all Android devices provide hardware-backed attestation capabilities.

It would also be possible to mitigate this attack by automatically analyzing the submitted selfies for modifications and flag suspicious ones as proposed by Rossler et al. (2019).

Furthermore, verifying the possession of the submitted phone number would increase an attackers effort. Additionally using certificate pinning and source code obfuscation would hamper potential malicious users.

In the next section, we present related work and discuss how they relate to this paper.

6 RELATED WORK

We can distinguish between two areas in literature, closely related to our work: Analysis of mobile mining and mobile application security. In this section, we will first discuss previous work on mobile mining. Subsequently, we will present work that focuses on detecting or preventing attacks on mobile applications.

Using smartphones for cloud mining is a relatively new concept. As a result, there exist hardly any studies on the security of such approaches. The first analysis of mobile mining was presented by Ziegler et al. (2018). In their 2018 paper, they analyze the security of Electroneum Mobile Mining. They demonstrate how the security mechanisms can be circumvented, and the reward system be exploited. The authors conclude that without a consensus algorithm, such as Proof-Of-Work, current-generation Android smartphones cannot provide sufficient security for a mobile mining process. Since the release of the paper, Electroneum has adapted several of the suggestions presented in the original paper. They have also redesigned the mining process and introduced a new reward system. In contrast to the original paper, our paper analyzes this new process. We present how to circumvent all new security mechanisms. We, however, also present a solution on how mobile mining could work without a consensus algorithm on current generation smartphones, by relying on Android's attestation features.

We now turn to previous approaches that allow to detect and prevent attacks on mobile applications. In literature, several approaches dealing with device and application integrity have been presented. For example, the works of Nauman et al. (2010), Bente et al. (2011) or Googles own *SafetfyNet* (Google, 2019) all demonstrate attestation mechanism which allows detecting if an application has been tampered with. However, most of these solutions are purely academic and are not integrated into the Android operating system. Furthermore, Prünster et al. (2019) proof that specific root mechanisms like Magisk[16], can circumvent the security features of *SafetfyNet*. The only solution available today seems to be the approach by Prünster et al. (2019). It relies on Android's key attestation capabilities and can thus be adopted quickly. The solution allows service providers to verify whether applications have been tampered with at runtime remotely.

On the other hand, detecting malware or malicious applications on Android has been researched intensively over the past years. As a result, we will only discuss the most prominent works. For example, already in 2009, Enck et al. (2009) proposed a certification process for applications. Their approach checks applications on installation based on predefined security rules. Vidas and Christin (2014) present an approach to dynamically detect malware based on four metrics: differences in behavior, performance, hardware and software components. Other approaches, such as the works presented by Desnos and Gueguen (2011); Jung et al. (2013); Zhou et al. (2013); Ren et al. (2014) focus on malicious app repackaging attacks. The goal is to find malicious repackaged applications of well-known applications.

Summarising, there exist a variety of tools to detect malware or repackaged applications. However, they target a scenario where an adversary tries to attack mobile applications. As such, they cannot prevent attacks such as application repackaging or application emulation, where the user, in this case also the attacker, has full control over the device.

[15]The feature is supported on devices running Android 8.0 and above with hardware-backed keystores such as a TEE or HSM.

[16]https://github.com/topjohnwu/Magisk

7 CONCLUSIONS AND FUTURE WORK

This work analyzed the newly employed security mechanisms of the Electroneum cloud mining app. The analysis identified several vulnerabilities related to the in-place security measures that allow for protocol exploitation and device/app impersonation attacks. We successfully mounted an account creation and device emulation attack to generate illegitimate accounts that exploit the cloud mining process and earn ETN reward tokens. On a technical level, the attack consists of an initial account setup phase that circumvents Electroneum's selfie-based account verification mechanism via generated selfies, and a fully-automated cloud mining extension process that reconstructs the network protocol and emulates a cloud-miner. We evaluated our approach by creating multiple fake accounts with generated selfies with predetermined gestures or symbols and developing a bot that emulates the Electroneum app and regularly maintains these accounts by extending their cloud-mining process. The generated selfies for each of these accounts were approved by the Electroneum team, and our automated mining-extension protocol successfully maintained the accounts and enabled receiving payouts at the same rate as legitimate reference accounts. We further propose potential mitigation techniques for preventing application repackaging or emulation attacks.

In future work we plan to further extend and improve our attack by fully automating the initial account setup phase. The complete automation of the selfie generation process would consequently increase the efficiency and scale of the attack and therefore enable the attackers to obtain significantly large amounts of ETN payouts.

Responsible Disclosure. We adhered to responsible disclosure guidelines and informed the Electroneum team about our findings.

REFERENCES

Apple (2019). App store review guidelines. https://developer.apple.com/app-store/review/guidelines/.

Bente, I., Dreo, G., Hellmann, B., Heuser, S., Vieweg, J., von Helden, J., and Westhuis, J. (2011). Towards Permission-Based Attestation for the Android Platform. In *Lecture Notes in Computer Science (including subseries LNAI and LNBI)*, volume 6740 LNCS, pages 108–115.

Comben, C. and Rivet, C. (2019). How to do cryptocurrency mobile mining. https://finance.yahoo.com/news/cryptocurrency-mobile-mining-100019158.html.

Corcoran, K. (2017). Bitcoin is climbing on the last day of 2017. https://www.businessinsider.com/bitcoin-price-value-increasing-on-final-day-of-2017-2017-12/commerce-on-business-insider?r=DE&IR=T.

Desnos, A. and Gueguen, G. (2011). Android: From Reversing to Decompilation. *Proc. of Black Hat Abu Dhabi*, pages 1–24.

Electroneum (2020). Electroneum roadmad - our vision mapped out. https://electroneum.com/journey/.

Enck, W., Ongtang, M., and McDaniel, P. (2009). On Lightweight Mobile Phone Application Certification. *Proceedings of the 16th ACM conference on Computer and communications security - CCS '09*, page 235.

Frankenfield, J. (2019). Mining pool. https://www.investopedia.com/terms/m/mining-pool.asp.

Google (2018). Let's build the world's most trusted source for apps and games. https://play.google.com/about/developer-content-policy-print/.

Google (2019). Safetynet attestation api. https://developer.android.com/training/safetynet/attestation.

Jung, J. H., Kim, J. Y., Lee, H. C., and Yi, J. H. (2013). Repackaging attack on android banking applications and its countermeasures. *Wireless Personal Communications*, 73(4):1421–1437.

Karras, T., Laine, S., and Aila, T. (2019a). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2019b). Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*.

Keybase (2019). The big stellar space drop. https://keybase.io/a/i/r/d/r/o/p/spacedrop2019.

Microsoft (2019). Face swap. https://www.microsoft.com/en-us/garage/profiles/face-swap/.

Nauman, M., Khan, S., Zhang, X., and Seifert, J.-P. (2010). Beyond Kernel-Level Integrity Measurement: Enabling Remote Attestation for the Android Platform. In Acquisti, A., Smith, S. W., and Sadeghi, A.-R., editors, *Trust and Trustworthy Computing*, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg.

Prünster, B., Palfinger, G., and Kollmann, C. (2019). Fides – unleashing the full potential of remote attestation. In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications*, volume 2: SECRYPT, pages 314—-321. SciTePress - Science and Technology Publications.

Ren, C., Chen, K., and Liu, P. (2014). Droidmarking: Resilient SoftwareWatermarking for Impeding Android Application Repackaging. *29th ACM/IEEE international conference on Automated software engineering*, pages 635–646.

Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., and Niessner, M. (2019). Faceforensics++: Learning to detect manipulated facial images. In *The IEEE ICCV*.

Vidas, T. and Christin, N. (2014). Evading Android Runtime Analysis via Sandbox Detection. *Proceedings of the 9th ACM symposium on Information, computer and communications security - ASIA CCS '14*.

Zhou, W., Zhang, X., and Jiang, X. (2013). AppInk: Watermarking Android Apps for Repackaging Deterrence. In *Proceedings of the 8th ACM SIGSAC symposium on*

Information, computer and communications security - ASIA CCS '13, New York, New York, USA. ACM Press.

Ziegler, D., Prünster, B., Marsalek, A., and Kollmann, C. (2018). Spoof-of-work evaluating device authorisation in mobile mining processes. In *ICETE 2018 - Proceedings of the 15th International Joint Conference on e-Business and Telecommunications*, volume 2, pages 380–387, Portugal. SciTePress.