Paying the Price for Disruption

How a FinTech Allowed Account Takeover

Vincent Haupert
Friedrich-Alexander University
Erlangen-Nürnberg
vincent.haupert@cs.fau.de

Dominik Maier
TU Berlin
dmaier@sect.tu-berlin.de

Tilo Müller
Friedrich-Alexander University
Erlangen-Nürnberg
tilo.mueller@cs.fau.de

ABSTRACT

This paper looks at N26, a pan-European banking startup and the poster child for young FinTech companies. We assess how security is treated by startups that provide disruptive technologies in the financial sector. In an area that has been committed to security, we find that FinTech companies have modern designs and outstanding user experience as their main priority. This strategy is rewarded by a rapidly increasing customer base but reveals a flawed understanding of security. We analyzed all aspects of security of N26, including the frontend, backend, protocols, human factors, and underlying design concepts, and found issues in all of them. We succeeded in leaking customer data, manipulating and carrying transactions and even could have entirely taken over foreign accounts. We reported these findings to N26 and did not disclose them before they were fixed. By publishing this case study, we hope to raise awareness about security considerations in the critical banking sector, especially for other FinTech startups.

CCS CONCEPTS

• **Security and privacy** → **Multi-factor authentication**; **Web application security**; *Software reverse engineering*; • **Social and professional topics** → Phishing;

KEYWORDS

FinTech, Mobile Banking, Security Breach, 2FA, PSD2

ACM Reference Format:
Vincent Haupert, Dominik Maier, and Tilo Müller. 2017. Paying the Price for Disruption: How a FinTech Allowed Account Takeover. In *ROOTS: Reversing and Offensive-oriented Trends Symposium, November 16–17, 2017, Vienna, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3150376.3150383

1 INTRODUCTION

Young FinTech companies strive to be disruptive in the financial sector. In other words, they try to invent technologies that displace existing banking technologies. Since FinTech companies operating in the consumer market focus on usability and user experience, well-established security standards of the banking sector are sometimes neglected. This paper presents a range of security vulnerabilities that were identified in N26, a FinTech company. N26, which currently has 500 000 registered users in countries across Europe, offers a smartphone-only bank account. Without access to a user's smartphone, we were able to not only reveal N26 customer data and manipulate transactions in real-time but also completely take over a victim's bank account. The bugs, taken together, would have eventually put the consumers' money and data at risk. We believe a security-focused development process would have discovered most of these bugs early on. This raises the question of whether startups tend to ignore well-established security concepts in favor of a quick market launch. Also, the focus on usability aspects over security plays a key role in the advancing downfall of conceptual security measures.

We demonstrate how little thought N26 put into security by presenting a variety of practical exploits against it. Those exploits were found in all parts of the N26 infrastructure and have since been fixed in the course of a responsible disclosure process. By combining those exploits, we show that even a complete account takeover was possible. We also provide an overview of the state of the art in mobile banking and shed light on the current state of transaction schemes used for banking. With the example of N26, we argue why basic concepts were flawed and how established banking standards were disregarded. We also discuss the current legal situation of banking in Europe with regard to transaction security. By highlighting the attacks described in this paper, we hope to raise awareness among investors, state actors, and founders alike in order to emphasize the need to include security analyses early on in the development process.

2 BACKGROUND

In this section, we elaborate on how online transactions and their security mechanisms evolved over time. Today, many FinTech companies like N26 are best described as app-only banks. Everything, including opening accounts, sending transactions, applying for loans and more is done through the central app. Of course, established banks have offered traditional online banking through the browser for much longer. Since N26 operates with a German banking license, we focus on the history and regulations of German online banking. Soon, however, supranational regulation will enforce similar rules throughout the European Union, as is outlined in the following section.

2.1 Online and Mobile Banking

In the German market, even in 1980 when online banking was first made public, every transaction had to be secured by an extra

ROOTS, November 16–17, 2017, Vienna, Austria

ACM ISBN 978-1-4503-5321-2/17/11…$15.00
https://doi.org/10.1145/3150376.3150383

transaction authentication number—the TAN. A TAN is a one-time password that a customer is required to enter after a transaction has been initiated. This means that on top of a customer's account number and password, single-use TANs are necessary as a second factor when performing credit transfers. The concept of TANs has been in place for over 30 years now and its security is steadily increasing. Technological advances in both attacks and defenses have led to more advanced TAN methods. The development in the recent past, however, has taken a different path, led by startups like N26. The emphasis is no longer on security but has shifted toward ensuring a better user experience at all costs, ignoring both conceptional and technical flaws.

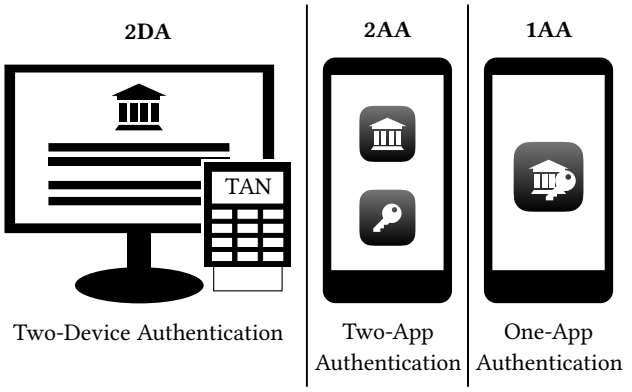

Figure 1: The three classes of two-factor authentication procedures used in digital banking in 2017.

Today, TAN procedures can be categorized into three classes, as visualized in Figure 1: *two-device authentication* (2DA), *two-app authentication* (2AA), and *single-app authentication* (1AA). In a 2DA scheme, the customer uses one device (e.g. a PC or a smartphone) to initiate the transaction using their online banking login credentials and another device to generate or receive the TAN. The pinnacle of 2DA methods from a security standpoint is the *chipTAN* method. This uses the customer's personal bank card and a dedicated reader device to create the TAN. Since the reader offers an external display, is not connected to the Internet, and handles only very limited input, purely technical attacks are practically infeasible. Mobile phones have been used to display TANs delivered via SMS, but this method was proven to be insecure years ago [24]. The possibility to run software on mobile phones then led to the creation of app-based TAN procedures. Different variations exist that require the device to be online (receiving TANs via push notifications like smsTAN) or work offline (generating TANs on a device, much like chipTAN). In contrast to earlier TAN methods, app-based procedures do not necessarily require the user to transfer the TAN manually to the transaction-initiating channel; rather, they might confirm the transaction directly—without ever displaying a TAN—when both the banking app and the TAN app are running on the same device. While using banking services on a smartphone via a separate 2DA TAN procedure—like chipTan—represents an increase in availability without immediate drawbacks in security, the introduction of two- and single-app authentication schemes constitutes a turning point in the development of online transaction security for the worse. These apps openly abandon the strong security guaranteed by two physically independent devices. Instead of using one device for transaction initiation and another for confirmation, 2AA uses a dedicated banking app to issue a transaction and another app to confirm it. Both apps can operate on the same mobile device. The 1AA scheme, with the N26 app as an early example, takes this idea one step further. Here, the theoretical border of a second, sandboxed, app, is no longer present. Instead, the entire transfer process is tucked into a single app, leaving almost no trace of the transaction numbers of the past.

2.2 Legal Aspects

The financial sector in Europe is regulated. These regulations extend to bank credit transfers as well. While we already noted that German online banking has historically been concerned about security, the European Banking Authority (EBA) stipulated rules that apply throughout the European Union [8]. The hallmark of these guidelines is that transactions need to employ *strong customer authentication* (SCA). It should make use of two mutually exclusive authentication elements of the categories *knowledge*, *possession*, and *inherence*. Furthermore, these elements need to be independent in such a fashion that "the breach of one does not compromise the other(s)". As these rules were established by the end of 2014 and were drafted as early as 2010, it might be surprising that 2AA and 1AA procedures are allowed at all. This is due to a major exemption of these EBA guidelines. They only affect transactions issued using a web browser. This means that the same transaction does not need to follow the rules if it is executed using a custom-made banking app instead. This is a classic niche that a startup could use to disrupt the market. Traditional banks are also taking advantage of this flaw, even releasing banking apps that merely wrap the banks' web page in a WebView inside an app. Consequently, these apps are exempted from the requirement of SCA even though the same functionality through a web browser would have required SCA. The *Revised Payment Service Directive* (PSD2), however, explicitly accounts for mobile devices and forces them to make use of SCA. While an early draft [9] still indicated that PSD2 will force payment service providers to implement the elements of SCA in separate and trusted execution environments, the final draft [10] dismissed this requirement. The EBA even emphasizes that SCA on a single device will be compatible as long as they make "use of separated secure execution environments through the software installed inside the multi-purpose device". Whether this will prohibit 1AA schemes in their current form is not publicly known yet.

2.3 Related Work

As stated, the course taken by authentication schemes entirely operating on a single device poses a security risk. Such schemes offer a single point of failure: the integrity of the single multi-purpose device they are running on, most often an Android- or iOS-driven smartphone. Given the sluggish update policy of many Android vendors and emerging attacks like Drammer developed by Fratantonio et al. [29], the integrity of smartphones cannot be guaranteed. Malware running with high privileges can be placed even in official app stores [23]. Grassini et al. consider smartphones

to belong to the same device group as chipTAN—namely "multi-factor OTP devices" [15]. However, as they are connected to the internet and able to run arbitrary third-party code, we deem smartphones more vulnerable than chipTAN devices. By analyzing and attacking two popular app-based TAN procedures—the pushTAN and the photoTAN methods—we offer proof that these methods are a clear security downgrade and thus have a significantly broader attack surface. The first case that shows the decline of conceptional security is the photoTAN procedure [18]. In contrast to the pushTAN procedure, the photoTAN method is used to leverage a dedicated hardware device—comparable to the security properties of chipTAN—to authorize transactions, thus offering a secure and trusted display. However, a special implementation of the app-based photoTAN method is implemented as a 2AA procedure. With two elements on the same phone, the user no longer takes photos but these are passed into the photoTAN app directly behind the scenes. Second, we showed an attack against the pushTAN 2AA procedure based on a privilege escalation exploit that transparently tampered with a user-initiated and user-confirmed transaction without the user being able to notice it [17]. Konoth et al. propose similar attacks and label them as *2FA synchronization vulnerabilities* [21], as these exploit the heavy synchronization that recent versions of smartphone operating systems make use of. Bai et al. researched mobile payments that work offline, revealing major limitations of existing token-protection techniques. They also point out the additional attack surface through the web, much like the shift from chipTAN to app TANs for online payments [1]. Yang et al. show that payments on phones are secured even less in a non-banking context—namely in the case of in-app payments. They uncovered hundreds of apps with at least 100 000 users which had flawed payment processes that could cause financial loss as well [31]. Research on online banking with regard to security was already underway in the 1990s through the work of Arie et al. [26]. The security of online banking has increased steadily ever since. Aside from payment, as discussed in this paper, a security-second approach can be seen in other areas that nourish innovative and fast-moving companies. In contrast to the banking scenario, the newer trend of the internet of things (IoT) has no history of security improvements to build upon. Historically, security has been rather grim for connected devices. Large-scale studies of firmware by Costin et al. [5] and in-depth firmware analyses by Shoshitaishvili et al. [27] uncovered a range of errors in embedded devices. Still, new products in the IoT sector face the same problems as the latest FinTech products: not enough emphasis is placed on security. Fernandes et al. argue that security in the IoT is facing the same problems as in traditional computing [12]. Quite similarly to the gist of our paper, Dragoni et al. use the IoT as an example to state the same facts. They show that here too a shift in culture is needed. According to their paper, "The Internet of Hackable Things", only 48% of organizations focus on security from the beginning of the development phase. Furthermore, Dragoni et al. provide data indicating that startups take a far laxer approach toward security in the IoT field. Their study shows that the percentage of startup executives who feel their products are highly resilient to cyber-attacks is 16% lower than in traditional firms [7]. Further blurring the line between the IoT and our app scenario, the security assessment of smart home applications by Fernandes et al. [11] considers the devices as well as their apps, showing errors on the app side. Thus, they unveil a lack of security understanding in the producing firms.

2.4 N26

N26 is a Berlin-based startup bank that focuses on bringing all financial services to the customer's smartphone. We chose N26 as our case study because it is the poster child for startups in the financial sector. N26 acquired over 500 000 customers since its launch in 2015 and is one of the few European FinTech companies operating on a full European banking license [14]. The success of N26 is based on its having all services and offerings entirely within a single app that provides outstanding usability. On top of common Single Euro Payments Area (SEPA) credit transfers, N26 also enables customers to perform direct payments to other N26 users, request an overdraft, or take out an insurance policy—all from within the app. While it is undoubtedly worthwhile to create intuitive and easy-to-use solutions, it is particularly important for a bank to repose business confidence by keeping their customers' money and data safe. In the past, established banks took a conservative approach by valuing security features very high. This strategy led to a steady increase in transaction security, but it came at the cost of user experience. Today, even established banks provide app-based authentication schemes operating on only a single device. This shift toward user-focused technologies can partly be accredited to disrupting new players like N26 which are changing the market. As a matter of fact, N26 was the first German bank that implemented the entire transaction process in one app [20], with others slowly following suit, despite the obvious security drawbacks. Implementing both, transaction initiation and confirmation, in a single app effectively eliminates two-factor authentication. In turn, strong vetting of the remaining infrastructure should be put in place. Following best practices with respect to technical security becomes increasingly important. The following sections, however, provide evidence that neither N26's Android and iOS apps nor their backend or underlying concepts could be described as secure. Seemingly, security has at no point been a focus of the young but well-funded company that handles the personal data and money of half a million people.

3 *APP*ALLINGLY INSECURE

This section describes practical exploits against the infrastructure of the evaluated startup, N26. We reveal a variety of dire security flaws, most of which could have been prevented by following best practices in the development process. All affected user accounts were part of the research group. The bugs have since been fixed in a professional manner as part of a responsible disclosure process. Still, for the sake of readability, we chose the present over the past tense.

3.1 Security Model

Before the presentation of the distinct security flaws, this section gives an overview of N26's security model and the most important security anchors. In comparison to other online and mobile banking systems, the N26 account's authentication scheme consists of more than two elements. All of them, however, relate to either knowledge or possession. The authentication factors inside the app that are based on knowledge are as follows:

Login Credentials. To log in, a user's email address and password are required. Both are given in a standard fashion during the registration process, at which time the user also automatically opens a bank account at N26. The password needs to satisfy the following policy: It should be at least seven digits long, consisting of a minimum of one digit, one special character, and one capital letter [25]. In our tests, however, it was not necessary to specify a special character. Providing a password of this length with only a capital letter and a digit was sufficient.
Transfer PIN. The transfer PIN is a four-digit numerical token that the user defines during the account activation process. The transfer PIN is mainly used to authorize transactions and cash withdrawals. The customer can change the transfer PIN at any time; this does not require entering the old pin but only the *MasterCard ID*, a token that is distinct to the N26 account and described below. For ATM withdrawals, the card is blocked after entering the wrong PIN three times.

Apart from these knowledge-based factors, the following elements aim at authentication through something only the user possesses:

MasterCard ID. This 10-digit numerical token is printed below the name on the customer's MasterCard (cf. Figure 2 for an example). This is *not* the credit card PAN. The MasterCard ID is properly the most important static authentication element of the N26 security scheme. A user needs it for the initial phone pairing, to change the transfer PIN, and during the unpairing process that is covered in greater detail in Section 3.3.4.
Paired Device. Through personalization, the customer's smartphone serves as a possession element relying on asymmetric cryptography. To define a paired device for the first time, the customer needs to supply the MasterCard ID and a one-time password delivered via SMS. After successful confirmation, the app sends the public key of a 2048-bit RSA key to the N26 backend, thus making this smartphone the paired device. Subsequently, this key is used in a challenge–response authentication system. There are some business cases that require the paired device—most notably transaction confirmation.
SIM Card. During the pairing process, a user also needs to supply a phone number that they want to pair the N26 app with. The client's phone number, however, does not serve as an authentication element. It is only required for unpairing.

Figure 2: The N26 MasterCard. The 10-digit number beneath the cardholder name is the *MasterCard ID*.

Besides the knowledge- and possession-based authentication elements, the N26 account also has other security features. Those have in common that they are not used for authentication but answer the purpose of user-specified bank account restrictions and allow the customer to quickly respond to abnormal incidents:

Immutable Data. While most of the data—including many security elements—can be changed by the users themselves only by knowing the account's login credentials, some data is immutable and can only be changed via the N26 customer service. This most notably includes the customer's email address that is used to recover a forgotten password. Further immutable data is the customer's name; the shipping address, however, can be changed at any time (even on an unpaired device).
Card Usage. A user can decide how one is allowed to use the N26 MasterCard. This includes switches that decide if online payments, ATM cash withdrawals, or payments abroad are permitted.
Card Limits. Like the card usage settings, a user can define a custom daily limit for cash withdrawals and payments. The range for cash withdrawals lies between €0 and €2 500, while for payments the minimum and maximum values are €0 and €10 000. Both withdrawals and payments have a fixed monthly limit of €20 000.
Lock Card. The N26 MasterCard can be locked, preventing its usage completely. The user might also unlock the card again—at any time—in the same fashion.
Push Notifications. On various occasions, the customer's smartphone receives real-time notifications. This includes successful and failed payments, incoming and outgoing transfers, and direct debit transfers. These notifications might help a customer to react quicker in the case of fraudulent transactions.

3.2 Frontend Flaws

This section is dedicated to the security flaws that we identified in the N26 frontends. The main means of interaction of N26 customers with their account is through the N26 app, which is available for Android and iOS. On top of this, N26 also offers a web app. The online version is limited as it does not allow the confirmation of transactions. It quickly became clear that all three clients talk to the same API endpoints. The Android app did not use any defensive mechanisms and was therefore a good choice as the main target. It allowed us to comfortably reverse-engineer the communication protocols and to examine the functionality up close. This yielded a device-independent vulnerability that one might abuse to carry out a transaction manipulation attack. On top of this, we found an Android-specific flaw that allows an attacker to inject arbitrary web content into the N26 app itself.

3.2.1 Transaction Manipulation. The goal of this attack is the transparent real-time manipulation of a user-initiated transaction. In this scenario, the victim, Bob, is in the act of sending a transaction worth €10 to the beneficiary Alice. He fills in Alice's international bank account number (IBAN) and an amount of €10, and submits the transaction. Thereafter, Bob is prompted to confirm his transfer as the app displays the transaction details again. As the details seem sound, Bob confirms the transaction. The next day, however, Bob uses the browser-based online banking service of N26 on his computer and realizes that he did not transfer €10 to Alice; it turns out that he transferred €200 to an unknown IBAN. This attack becomes

possible because N26 did not sign the transaction data despite already making use of their private and public key infrastructure during the transaction confirmation on the paired device. They also did not make use of certificate pinning. Technically, a credit transfer works in the following way:

(1) The customer enters the transaction details—that is, the beneficiary's IBAN and the desired amount—in the N26 app and presses the send button.
(2) Next, the user needs to supply the four-digit numerical transfer PIN, which is sent along with the transaction order.
(3) If the provided PIN is correct, the transaction initiation is completed and the N26 backend sends a push notification to the customer's paired device. The payload of the push message contains a TAN encrypted with the public key of the paired device.
(4) After opening the push notification, the app displays the transaction details again and asks for the user's confirmation. If the user authorizes the transaction, the encrypted TAN is decrypted using the private key and sent back.
(5) If the TAN is correct, the transaction comes into effect.

If an attacker can launch a man-in-the-middle (MitM) attack, the transaction can be tampered with transparently. This becomes possible because of two reasons. First, even though the N26 apps make strict use of HTTPS, they do not make use of certificate pinning—a best practice that prevents unauthorized third parties from breaking the confidentiality and integrity of the transmitted data. Second, the push notification only contains the encrypted TAN but not the transaction details. Therefore, the app is forced to rely on and display the data that the user entered in the first step instead of presenting transaction details received through a second communication channel encrypted with the paired device's public key. In summary, an adversary can circumvent the three-step authentication procedure only by compromising the network layer. This can be achieved by successfully executing one of the following attack vectors:

– A user can be tricked into installing the certificate through phishing or other means of social engineering. The user is the weakest point in the system. So, the app should pin its certificates. In general, attacks using phishing or social engineering are particularly dangerous and have a high success rate [6].
– A trusted certificate authority (CA) issues the certificate. Vulnerabilities in CA validation processes sometimes allow an attacker to take hold of a certificate for domains they do not own.
– Both Android and iOS are frequently the prey of privileged malware—that is, malware that performs a privilege escalation exploit before executing their payload. It is a trivial task for privileged malware to place a certificate.

3.2.2 Popup WebView Injection. While reversing the Android app, we stumbled upon many deep links registered in the app manifest. Opening many doors is never a good idea and deep links are mostly considered to have a negative security impact [22]. In this case, we found one link—`number26://main/?tutorial=${url}`—which allows a popup injection. The web link enables an attacker to launch the N26 app and make it display a popup window containing a WebView rendering any user-provided content. The WebView

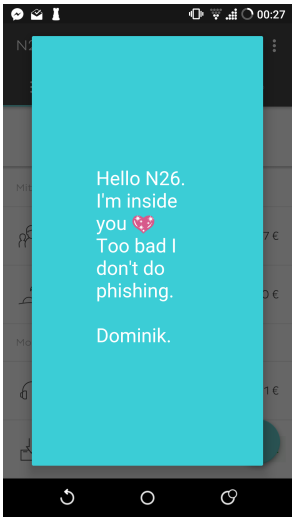

Figure 3: WebView Injection. An external deep link starts the app and allows it to display any web page inside it.

is code-enabled and the N26 app injects a Java callback object that allows sharing to Facebook, navigation inside the app, and more. The possibility to run JavaScript increases the attack surface, depending on the app and Android versions. Even worse, the popup poses a severe threat through advanced phishing attacks. Since the user logs into the app as usual before the popup is displayed, the app name is still N26 in the system and there is no way to distinguish this popup from a genuine app interaction. In contrast to known phishing attacks on Android that usually need malicious apps with appropriate permissions to be installed [4], this attackvector does not depend on any permissions. Instead, making matters worse, deep links starting the attack can even be triggered by a hostile web page while browsing. The size and look of the popup is completely dependent on the target URL. Figure 3 displays an example popup that shows the app in the background. The link URL indicates that the reason for the existence of the flaw must have been to allow a general way to display tutorials. This is a rather unfortunate choice that might have been noticed earlier had there been a larger focus on security during development.

3.3 Backend Flaws

In contrast to the previous section on the security issues of the N26 apps, this section deals with security flaws related to the N26 backend. As all N26 frontends make use of the same API calls—that is, there are no app-specific requests—these vulnerabilities are independent from the device that a customer uses. Apart from the internals already acquired from reverse-engineering the N26 apps, we learned the most about their communication protocol through HTTPS interception. While presenting the frontend flaws in the last section, we noted that the apps do not make use of certificate pinning. Hence, we were not even required to alter the app in order to intercept the communication using a MitM proxy. Through this analysis, we identified a broad variety of partly severe security issues.

3.3.1 Information Leakage. N26 also offers peer-to-peer payments called *MoneyBeams*. They are sent immediately and,—in contrast to SEPA credit transfers—, come into effect without delay. This feature, however, is only available if the beneficiary is also an N26 customer. Therefore, when a customer wants to send a MoneyBeam to a friend, the N26 app scans the user's address book for contacts that use N26 as well. Even though it is comprehensible that N26 wants to present other N26 users directly in the app, the technical implementation does not preserve privacy as the N26 app uploads *all* email addresses and phone numbers found in the customer's address book in plain text to the N26 backend. This is also illustrated in Figure 4. There is no hashing and even a zero-knowledge approach may be applied. Consequently, this feature can also be abused to identify if a given email or phone number is associated with an N26 account. In particular, having a set of email addresses which are known to be used for an N26 account is useful for launching a targeted attack. As a customer's address

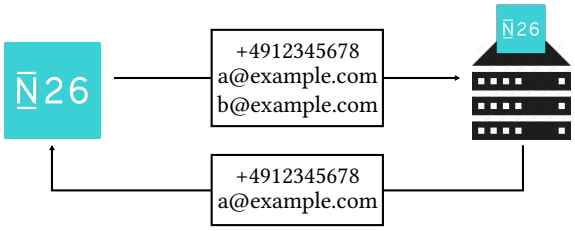

Figure 4: N26 customer identification.

book potentially contains many entries, it is easy to probe more than 1,000 candidates in a single request. Technically, N26 sends a POST request with a JSON array of email and phone number strings, and the backend looks for those entries which are associated with an N26 account and returns them in the response body. As this endpoint does not implement any rate-limiting, one might use it to scan massive data sets of email addresses and phone numbers for N26 accounts. Owing to the limited entropy of phone numbers, even a brute force attack on them would be feasible.

3.3.2 Unrestrained Siri Transactions. Since the release of iOS 10, third-party apps can use Siri to interact with the user. N26, being an agile company, adapted this feature early on. Hence, users of iOS 10 can dictate to Siri a transaction's beneficiary and amount (with a limit of €25 per transaction and a maximum of €200 per day). Siri asks the user for confirmation of the details and executes the transfer. The user is not required to enter the login credentials but might only perform these transactions on the paired device. Here, we can see once more the strong focus on usability. Figure 5 seems to confirm this restriction and suggests that N26 uses the same mechanism for Siri transactions as for regular or peer-to-peer transactions—that is, it confirms them with a TAN. As it turns out, however, N26 created a new endpoint for unverified low-amount transactions that, as the name suggests, do not require a TAN to become effective. Besides, not allowing Siri transactions from an unpaired device is solely a client value. An attacker who directly communicates with the N26 API does not have this restriction and may therefore perform arbitrary transactions. Enforcing client-side security clearly ignores best practices. After a successfully executed

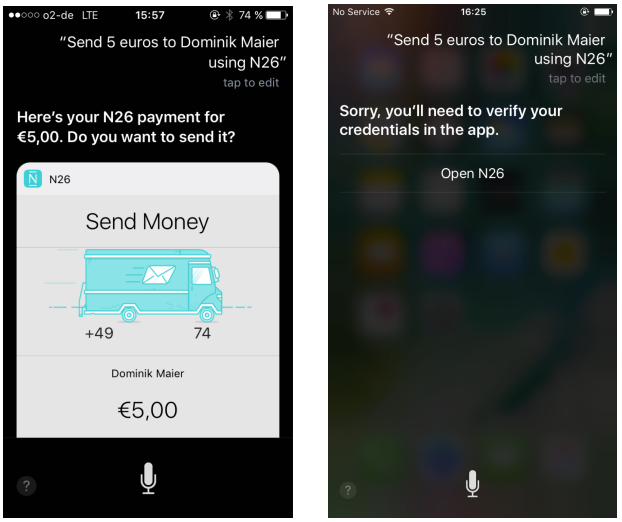

Figure 5: A Siri transaction on the paired device on the left succeeds while the same request is denied on an unpaired phone.

Siri transaction, we could withdraw the sent money from an ATM immediately. This makes attacks attractive. If an attacker were able to steal a single MasterCard and its transfer PIN, he could consequently use it as an *exit account*.

3.3.3 Permissive Risk Analysis. Taking the vulnerabilities presented so far into account, one might wonder if exploiting them in practice is not significantly more difficult thanks to transaction risk analysis. This is not far-fetched as even N26 claims intelligent algorithms that "immediately detect irregularities" would be able to prevent fraud before it even occurs. We tested these algorithms by automatically performing over 2,000 Siri transactions in a 30-minute period. Each of these transactions went through without an issue. We were surprised that no limits were applied as issuing this amount of transactions in the given time frame using voice commands is impossible and therefore definitely irregular. Nevertheless, we were confident that N26 would reach out to us shortly. However, this happened more than three weeks later when the N26 support reached out to us seeking an explanation for the unusual amount of transactions and informed us that this activity could result in account cancellation. Even though this reaction at first seems reasonable, N26 did not contact the sender; rather, it reached out to the receiver of the amount. Apart from the clearly manual processing of irregular activity, this may indicate that N26 reacted not because of security concerns but following a violation of their terms of service, probably assuming a business usage of the account. Furthermore, our entire analysis of N26 security can in no way be described as stealth. In summary, we doubt that N26 could have reacted appropriately to a real-world attack with higher amounts of money, possibly distributed over multiple accounts.

3.3.4 Unpairing. Even in case an attacker is able to obtain the login credentials as well as the transfer PIN, performing SEPA credit transfers with an arbitrary amount is still not possible because of the unavailable private key of the paired device. The goal of this attack

is to gain the possibility to initiate *and confirm* attacker transactions. To achieve this, the attacker will first unpair the victim's device and pair an attacker-controlled device right after. The *pairing* process is straightforward and requires the user only to supply a mobile phone number. N26 sends a four-digit numerical token as a text message to the given number. After the user enters the token into the app, the pairing process is completed. The *unpairing* process, however, is secured by a multi-step and multi-factor procedure. The steps one must perform to unpair an already existing pairing are as follows:

(1) Start the unpairing through the app or web interface. An email with a link (that contains a token as a parameter) to start the actual unpairing is sent to the user's email account.
(2) After following the link, the user needs to supply the transfer PIN.
(3) The user is prompted to enter the MasterCard ID.
(4) N26 sends a five-digit token to the number which was used in the previous pairing.

This conception of the unpairing process would require an attacker to (1) have access to the victim's email account, (2) know the transfer PIN, (3) have the MasterCard available, and (4) be able to receive the victim's SMS. Even though it is generally commendable that N26 identified the unpairing as particularly critical, each of these steps can be circumvented by knowing only the victim's login credentials:

(1) When the user starts the unpairing process, this technically leads to an HTTP request which causes the backend to send out an email. But instead of only sending the link to start the unpairing with the customer's email address, N26 also sends it as a response to the HTTP request. As a result, an attacker does not need access to a victim's email account.
(2) During the actual unpairing process, one is first prompted to enter the transfer PIN. Given that the PIN can be reset by only knowing the MasterCard ID,—which is required in the next step—, this step is redundant and does not offer any additional security. Hence, an adversary is not required to know the transfer PIN.
(3) The MasterCard ID ordinarily would require an attacker to possess the victim's MasterCard. Once again, it is noteworthy that this ID is in no way derived from the PAN nor is it directly connected to the PAN. Despite the PAN, which is probably masked in any response body, the MasterCard ID is part of *every* transaction performed with the card because it serves as a prefix in the unique identifier (UID) of transactions. Consequently, any card action, whether—buying something online, withdrawing cash, or making a payment on a POS terminal—, causes the MasterCard ID to leak in the transaction history. As noted in the previous step, the MasterCard ID also allows one to reset the transfer PIN to an attacker-controlled value.
(4) The last step in the unpairing process would require the attacker to either possess the victim's SIM card or be able to intercept messages sent to it. We were also unable to trick the backend into sending the unpairing token to a phone that is under the adversary's control and changing the phone number is an action that requires calling the customer service. The backend endpoint that validates the five-digit numerical token,—in contrast to the endpoint used for login—, does not implement rate-limiting. As we were on average able to probe 160 candidates per second, guessing the right token takes approximately five minutes.

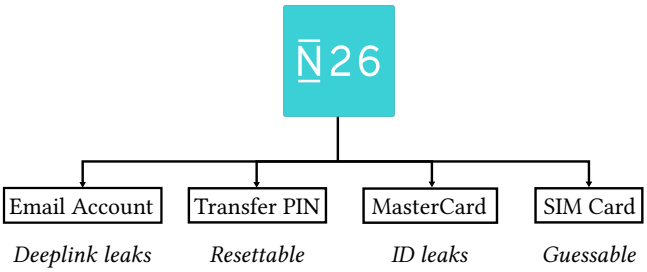

Figure 6: The authentication elements involved during the unpairing.

As summarized in Figure 6, an attacker can unpair a victim's phone without having access to the email account, without knowing the transfer PIN, and without possessing either the MasterCard or the SIM. After a successful attempt at unpairing, an attacker can simply pair a new phone. Although this attack even in its worst case only takes approximately 10 minutes and an attacker would likely execute it at night, the described unpairing attack generates a total of three emails and one SMS that the victim inevitably receives. There is one email containing the unpairing link, another two emails informing the victim about the reset of the transfer PIN and the successful unpairing, and an SMS with the token this attack performs a brute force attack on. After a successful unpairing attempt, a victim has no possibility to regain control of the account without calling the customer service. If this attack is executed outside of N26's office hours, even a victim who notices the attack on time can do nothing but watch.

3.3.5 Overdraft Abuse. N26 also offers a real-time overdraft. This feature is particularly interesting because N26 immediately grants an overdraft ranging from €50 to €2, 000 depending on the client's credit score. N26 is aware that using credit is a particularly critical process; hence, an overdraft might only be applied by using the customer's paired phone in a similar fashion as legitimizing transactions. Therefore, as the previous Section 3.3.4 demonstrated, an attacker can gain control of the paired device by only knowing the user's login credentials. Since the money is available immediately after the overdraft application, an attacker can steal even more money than the victim has available. In the best case,—from an attacker's point of view—, this could be as much as an additional €2, 000.

4 THE ROAD TO COMPLETE ACCOUNT TAKEOVER

So far, we have described various flaws of the N26 frontends and backend. However, we are yet to discuss how an attacker could gain access to a victim's login credentials or how they could use these flaws to launch a feasible attack. After outlining how to obtain the login data, this section describes an attack scenario that involves constructing a large-scale attack based on a spear-phishing attack and combining several of the described vulnerabilities. Later,

we briefly describe an impersonation attack by calling the N26 customer service.

4.1 Obtaining the Login Credentials

A precondition of taking control of the entire N26 account is that the attacker has either already obtained the login credentials or is able to sniff out an *access token*, which is a longer-lasting login token used for fingerprint or pattern login. There are many possibilities for gaining access to this information, among them:

- Gain access to the victim's email account. This is a weak point of the N26 password-recovery procedure. In general, N26 has a solid password policy which requires the user to supply at least seven characters containing at least one digit, one capital letter, and one special character. In conjunction with the employed rate-limiting, this virtually defeats brute force attacks. The password can be restored only by having access to the user's email account, which effectively reduces the password policy to the one the mail provider enforces. This could be as bad as 1234. Apart from this, email accounts themselves are subject to attacks and databases with email addresses are leaked from time to time [19].
- Password reuse. The email address is often a semi-public token. Owing to the vast numbers of accounts everybody owns and uses every day, passwords are frequently affected by reuse. So, it is likely that a customer of N26 is not using their password exclusively [30]. This is even more likely because N26's password policy is slightly below current accepted standards and consequently the customer does not need to invent a new password (or modify it) to fulfill the password restrictions.
- A victim falls for a phishing attack. Phishing mails are a serious threat because it is hard to mitigate them. Plenty of literature and papers available deal with the alarming success rates of phishing, which has not declined over time. A recent example is the phishing attack against Raiffeisen Bank [28]. We do not know if phishing is already an issue for N26 but the company could become a more attractive target as its customer numbers increase.
- An attacker can easily hijack a victim's account if he is able to obtain an access token. This could happen, for example, due to a forgotten logout.

Apart from the login credentials, the victim is required to already have used the MasterCard at least once to have the MasterCard ID available in the transaction history. The type of usage—for example, card payment or cash withdrawal—does not matter. Considering that the N26 MasterCard is not only automatically issued and sent to the customer but also an important part of the N26 business model, this is a weak precondition.

4.2 Large-scale Attack

In this section, we want to outline an attack that makes use of several of the presented security flaws and by combining these could have been able to take over thousands of accounts. The previous section already noted that a phishing attack is an effective way to gain control of a victim's login credentials. At first glance, N26 is an attractive phishing target because the only personalization they use in their business emails sent to the clients is the customer's first name. Frequently, the emails also contain clickable links. However, N26's total number of clients is still relatively low. Hence, a phishing mail sent out to random email addresses will probably be highly ineffective. An improved form of phishing is spear phishing, which is basically a more targeted version of phishing: By gaining additional information about the targets, the attackers send emails containing more details that the customer might relate to; thus, this is more effective than a regular phishing attack [2]. Owing to the information leakage vulnerability described in Section 3.3.1, we can effectively identify the N26 customers who use a given set of email addresses or phone numbers. In the past year, a huge database of over 65 million email addresses and password hashes obtained from a security breach at Dropbox was made publicly available [13]. We evaluated the *entire* data set exploiting the given information leakage vulnerability and identified more than 33,000 email addresses that are used to log into an N26 account. An attacker could have used this information to launch a highly targeted spear phishing attack that, for example, prompts the user to change their N26 password because they are affected by the Dropbox leak, as using the same password for both accounts could pose a security risk. Furthermore, Section 3.2.2 outlined an attack against the Android version of the N26 app that could be used to inject arbitrary content inside the N26 app. Therefore, the attacker email could also have asked the customer to click on a link to change the N26 password inside the app. This would further increase the victim's confidence in the legitimacy of the phishing mail. After an attacker knows the login credentials, the rest of the attack is straightforward: An adversary can now already make use of Siri transactions because, as we uncovered, they do not require the paired device. They are, however, limited to €200 per day per account. Even though this could fetch a significant amount of money, pairing an adversary-controlled phone by performing our unpairing attack would allow the attacker to gain access to all of the victim's money and even more. As a paired device also allows one to apply for an instant overdraft, an attacker could transfer money that the customer does not even own. This seems a particularly useful part of the attack because it is rumored that many people open an N26 account only to use it occasionally rather than as their main account because of its attractive credit card conditions (e.g. no account management fees, no transaction fees, a limited number of free cash withdrawals). In this case, a victim might not have a lot of money in the account but exploiting the overdraft can make up to €2,000 accessible. Last but not least, Section 3.3.3 showed that the transaction risk analysis that N26 performs is too permissive, which supports the idea that the depicted attack scenario would be highly successful.

4.3 Impersonation Attack

While the previous section showed an attack scenario that scales very well and combines multiple vulnerabilities to create a complete attack, it is also possible to perform a more individual-oriented attack after getting to know the victim's login credentials through impersonation. Section 3.1 already noted that owing to security considerations some of the data of an N26 account is immutable. Most importantly, this includes the customer's email address, which is required for login, and the phone number, which is required during unpairing. However, there are legitimate cases where a customer

may need to change the phone number because, for example, they have lost access to the original number. As N26 seeks to be a pure mobile bank, they do not have any bank branch that a customer could visit. Therefore, a client needs to call the N26 customer service to enter the new phone number. Naturally, N26 authenticates a customer prior to any action. However, the answer to each of the three questions the N26 customer support asks over the phone is either already available from the vulnerabilities described so far or can be extracted:

(1) After obtaining a customer's name, the customer service executive asks for the MasterCard ID. The unpairing attack already revealed that this token can be leaked as it is used as a prefix of the UID of transactions performed with the MasterCard.
(2) Next, they ask for the place of birth. Even though this is neither visible through the app nor processed, the place of birth leaks out in the queries the app performs.
(3) The last authentication question relates to the current account balance. It is sufficient to just have an idea of the account balance. As the exact values are always available with the login credentials, this does not pose a problem anyway.

After answering these questions correctly, an attacker has full control of the account and might request to change arbitrary data. This includes the phone number, which would allow an adversary to pair their own phone and gain control over all of the victim's money including the overdraft accessible. Even though this attack does not scale well since it requires human interaction, it has the advantage of absolute stealth because the N26 customer service does not notify the victim by email or SMS that crucial account data has been changed.

5 RESPONSIBLE DISCLOSURE

We chose a responsible disclosure process to give N26 enough time to address all the issues, so as not to put any customer at risk. As we were unsure about how N26 might react from a legal perspective, we asked the Chaos Computer Club to establish the contact. The contact with N26, however, was professional and friendly, and the technical staff was not only thankful for our reporting of the issues but also interested in improving their development process. After we granted a three-month period to address all the issues, we first presented our results at the 33rd Chaos Communication Congress to the public [3, 16]. To the best of our knowledge, N26 addressed all our reported findings prior to this talk.

6 CONCLUSION

We showed various exploits against N26, a well-funded, fast-growing startup in the vital banking sector. Taken together, these exploits could help attackers to steal the customers' money, thereby potentially pushing the company out of business. At the root of those exploits, we identify a strong focus on user experience combined with the high expectations of investors in the FinTech startup scene. This leads to the pressure of a quick market launch. Admittedly, if no analyst looks deeper into it, there is no immediate return on investment for security owing to its non-functional nature. But it is necessary to keep in mind the damage to customers, the loss of reputation for the company, and more generally the decline of confidence the public has in the bank's responsible use of their money. Once a security flaw becomes public, however, it justifies embracing security from the beginning even from an economic point of view. To summarize, we showed attacks against different points in the infrastructure of N26, including the frontend, backend, protocols, and underlying design concepts. After our revelations, N26 is indeed placing greater focus on security, but they could have avoided most of the problems from the beginning. We hope to raise awareness among the founders of and investors in other FinTech startups about the importance of including security analyses early on in the development process. We also outlined the ever-increasing evolution of online banking security and its recent decay. In general, we argue that multiple apps running on the same system, separated only by the integrity of the operating system, cannot be considered as secondary factors. This is also true for banks other than N26, including well-established banks that have shifted to the same technology in the recent past. Moreover, we see an obligation among governmental institutions in this regard. Particularly the vetting process in place for banking requires an understanding that is tailored to the digital era. In our opinion, in-depth security analysis and penetration tests that not only are based on reading a company's documentation but also perform practical attacks must be included in the development process.

ACKNOWLEDGMENTS

We wish to thank Felix Binder for the insightful discussions on the N26 security conception and implementation as well as the anonymous reviewers for their helpful comments.

REFERENCES

[1] Xiaolong Bai, Zhe Zhou, XiaoFeng Wang, Zhou Li, Xianghang Mi, Nan Zhang, Tongxin Li, Shi-Min Hu, and Kehuan Zhang. 2017. Picking Up My Tab: Understanding and Mitigating Synchronized Token Lifting and Spending in Mobile Payment. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 593–608. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/bai
[2] Zinaida Benenson, Freya Gassmann, and Robert Landwirth. 2017. Unpacking Spear Phishing Susceptibility. In *Targeted Attacks (Financial Cryptography and Data Security Workshops) (FC '17)*. Springer, 1–17.
[3] Chaos Computer Club. 2016. Shut Up and Take My Money! (30 12 2016). Retrieved October 01, 2017 from https://fahrplan.events.ccc.de/congress/2016/Fahrplan/events/7969.html
[4] Qi Alfred Chen, Zhiyun Qian, and Z. Morley Mao. 2014. Peeking into Your App without Actually Seeing It: UI State Inference and Novel Android Attacks. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, San Diego, CA, 1037–1052. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/chen
[5] Andrei Costin, Jonas Zaddach, Aurélien Francillon, and Davide Balzarotti. 2014. A Large-Scale Analysis of the Security of Embedded Firmwares. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, San Diego, CA, 95–110. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/costin
[6] Rachna Dhamija, J. D. Tygar, and Marti Hearst. 2006. Why Phishing Works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. ACM, New York, NY, USA, 581–590. https://doi.org/10.1145/1124772.1124861
[7] Nicola Dragoni, Alberto Giaretta, and Manuel Mazzara. 2017. The Internet of Hackable Things. (08 2017).
[8] European Banking Authority. 2014. *Final guidelines on the security of internet payments.* Technical Report. Canary Wharf, London, UK. https://www.eba.europa.eu/documents/10180/934179/EBA-GL-2014-12+(Guidelines+on+the+security+of+internet+payments)$_{R}$ev1
[9] European Banking Authority. 2016. *EBA consults on strong customer authentication and secure communications under PSD2.* Technical Report. Canary Wharf, London, UK.
[10] European Banking Authority. 2017. *EBA paves the way for open and secure electronic payments for consumers under the PSD2.* Technical Report. Canary Wharf,

London, UK. https://www.eba.europa.eu/-/eba-paves-the-way-for-open-and-secure-electronic-payments-for-consumers-under-the-psd2

[11] E. Fernandes, J. Jung, and A. Prakash. 2016. Security Analysis of Emerging Smart Home Applications. In *2016 IEEE Symposium on Security and Privacy (SP)*. 636–654. https://doi.org/10.1109/SP.2016.44

[12] Earlence Fernandes, Amir Rahmati, Kevin Eykholt, and Atul Prakash. 2017. Internet of Things Security Research: A Rehash of Old Ideas or New Intellectual Challenges? *IEEE Security & Privacy* 15, 4 (2017), 79–84. https://doi.org/doi.ieeecomputersociety.org/10.1109/MSP.2017.3151346

[13] Samuel Gibbs. 2016. Dropbox hack leads to leaking of 68m user passwords on the internet. (31 08 2016). Retrieved August 24, 2017 from https://www.theguardian.com/technology/2016/aug/31/dropbox-hack-passwords-68m-data-breach

[14] N26 GmbH. 2017. N26 increases number of customers to over 500.00. (21 08 2017). https://n26.com/content/uploads/2017/08/mobile-bank-with-strong-organic-growth.pdf Press release.

[15] Paul A Grassi, Michael E Garcia, and James L Fenton. 2017. Digital Identity Guidelines. *NIST Special Publication* 800 (2017), 63–3.

[16] Vincent Haupert. 2017. N26. (01 02 2017). Retrieved October 01, 2017 from https://www1.cs.fau.de/n26

[17] Vincent Haupert and Tilo Müller. 2016. Auf dem Weg verTAN: Über die Sicherheit App-basierter TAN-Verfahren. In *Sicherheit 2016 (Sicherheit, Schutz und Zuverlässigkeit Bonn 05.4. - 07.4.2016) (SICHERHEIT 2016)*, Michael Meier, Delphine Reinhardt, and Steffen Wendzel (Eds.). Gesellschaft für Informatik e.V. (GI), Bonn, 101–112.

[18] Vincent Haupert and Tilo Müller. 2016. *On App-based Matrix Code Authentication in Online Banking*. Technical Report. Freidrich-Alexander-Universität Erlangen-Nürnberg.

[19] Troy Hunt. 2017. Pwned websites. (2017). Retrieved August 26, 2017 from https://haveibeenpwned.com/PwnedWebsites

[20] Maik Klotz. 2017. Online-Banking: Digitaler Stillstand since 1980. (10 08 2017). Retrieved August 21, 2017 from https://paymentandbanking.com/online-banking-digitaler-stillstand-since-1980/

[21] Radhesh Krishnan Konoth, Victor van der Veen, and Herbert Bos. 2017. *How Anywhere Computing Just Killed Your Phone-Based Two-Factor Authentication*. Springer Berlin Heidelberg, Berlin, Heidelberg, 405–421. https://doi.org/10.1007/978-3-662-54970-4$_2$4

[22] Fang Liu, Chun Wang, Andres Pico, Danfeng Yao, and Gang Wang. 2017. Measuring the Insecurity of Mobile Deep Links of Android. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 953–969. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/liu

[23] D. Maier, T. Müller, and M. Protsenko. 2014. Divide-and-Conquer: Why Android Malware Cannot Be Stopped. In *2014 Ninth International Conference on Availability, Reliability and Security*. 30–39. https://doi.org/10.1109/ARES.2014.12

[24] Collin Mulliner, Ravishankar Borgaonkar, Patrick Stewin, and Jean-Pierre Seifert. 2013. SMS-based one-time passwords: attacks and defense. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, Berlin, Heidelberg, 150–159.

[25] N26 GmbH. 2017. Password For Your N26 Account. (2017). https://support.n26.com/read/000001288?locale=en

[26] Arie Segev, Jaana Porra, and Malu Roldan. 1998. Internet Security and the Case of Bank of America. *Commun. ACM* 41, 10 (Oct. 1998), 81–87. https://doi.org/10.1145/286238.286251

[27] Yan Shoshitaishvili, Ruoyu Wang, Christophe Hauser, Christopher Kruegel, and Giovanni Vigna. 2015. Firmalice-Automatic Detection of Authentication Bypass Vulnerabilities in Binary Firmware.. In *NDSS*.

[28] Lukas Stefanko. 2017. Phishing attack at Raiffeisen Bank by MazarBot. (08 2017). Retrieved August 26, 2017 from https://b0n1.blogspot.de/2017/08/phishing-attack-at-raiffeisen-bank-by.html

[29] Victor van der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clementine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. 2016. Drammer: Deterministic Rowhammer Attacks on Mobile Platforms. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. Vienna, Austria.

[30] Ding Wang, Zijian Zhang, Ping Wang, Jeff Yan, and Xinyi Huang. 2016. Targeted Online Password Guessing: An Underestimated Threat. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. ACM, New York, NY, USA, 1242–1254. https://doi.org/10.1145/2976749.2978339

[31] Wenbo Yang, Yuanyuan Zhang, Juanru Li, Hui Liu, Qing Wang, Yueheng Zhang, and Dawu Gu. 2017. Show Me the Money! Finding Flawed Implementations of Third-party In-app Payment in Android Apps. *24th Annual Network and Distributed System Security Symposium, NDSS* (2017).