Securing Telephone Communication with Trusted Services*

Francesco Buccafurri[1,*,†], Vincenzo De Angelis[2,†], Sara Lazzaro[2,†] and Carmen Licciardi[1,†]

[1]University Mediterranea of Reggio Calabria, Via dell'Università 25, Reggio Calabria, 89122, Italy
[2]University of Calabria, Via P.Bucci, Arcavacata di Rende (Cosenza), 87036, Italy

Abstract

In the landscape of cyber attacks, social engineering has often a strategic role for the attacker. Recently, social engineering attacks that exploit fake phone communication are increasingly on the rise. The state-of-the-art countermeasures that do not require changes of the telephone network infrastructure have still some limitations. Therefore, the problem merits further attention. In this paper, we choose a restricted but yet very meaningful use case, that is the case in which users communicate with trusted services (eg., their bank). In this case, there are two possible threats: vishing combined with number spoofing attacks and malware-based fake calls. Both are very dangerous because the victim can be easily deceived as they are inclined to trust the authority of the entity they are communicating with. In this context, we propose a practical solution whose goal is to authenticate the communicating trusted entity using a form of double-factor authentication. Our solution works both in VoIP and non-VoiP networks, is very lightweight, and does not require changes in the telephone infrastructure. A proof of concept of the core functions of the solution is also shown in the paper, which runs on Android and leverages Lightweight Directory Access Protocol (LDAP) to implement the out-of-band channel for the second authentication factor.

Keywords

Telephone Communication, Spoofing attacks, Vishing attacks

1. Introduction

Often cyber attacks exploit human factor weaknesses. Among other social engineering tactics, vishing is more and more used by attackers who impersonate an authority to deceive the victim and persuade them to perform specific actions. Possible actions are for example fraudulent money transfers, in the case the impersonated authority is the bank of the victim. In particular, the scammer exploits suitable techniques for phone-number spoofing [1] in such a way that in the smartphone of the victim the incoming call appears authentically coming from the bank. In this case, the victim is the callee. Impersonation of phone interlocutors can occur also when the victim is the caller. There is a very recent malware, called FakeCall. FakeCall malware operates as a banking trojan designed to deceive users by simulating legitimate banking or financial institution interactions. When activated, FakeCall hijacks a legitimate call that the victim starts to a certain trusted service, redirecting it to the scammer, but keeping legitimate number displayed on the screen.

In general, attacking the authenticity of phone communications with trusted services can result in very serious impact for the victim.

In the past literature, some efforts have been put in place against phone number spoofing. The Federal Communications Commission (FCC) has required U.S. telecom providers to implement STIR/SHAKEN, a solution developed by the industry that leverages digital signatures. However, scaling this system in the

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT

*You can use this document as the template for preparing your publication. We recommend using the latest version of the ceurart style.

*Corresponding author.

†These authors contributed equally.

bucca@unirc.it (F. Buccafurri); vincenzo.deangelis@dimes.unical.it (V. De Angelis); sara.lazzaro@dimes.unical.it (S. Lazzaro); carmen.licciardi@unirc.it (C. Licciardi)

0000-0003-0448-8464 (F. Buccafurri); 0000-0001-9731-3641 (V. De Angelis); 0000-0002-0846-4980 (S. Lazzaro); 0009-0002-9981-1569 (C. Licciardi)

telecommunications industry poses significant challenges, potentially making it unfeasible. Additionally, STIR/SHAKEN is limited to IP-based systems (e.g., SIP), leaving traditional non-IP networks (e.g., SS7) vulnerable. Some research proposals attempted to overcome the limitations of STIR/SHAKEN. However, they either assume a trusted telephone network infrastructure [2] or need a centralized server to support mutual authentication of callers and callees [3].

In this paper, we find a solution that does not assume the trustworthiness of the telephone network infrastructure and does not rely on centralized servers. Moreover, it is agnostic to the type of telephone system and, similarly to [3], uses a data channel to implement a double-factor authentication of the trusted service. Unlike [3], no handshake between communicating parties is established, making the solution much more lightweight and less prone to possible software bugs.

A proof of concept of the core functions of the solution is also shown in the paper, which runs on Android and leverages Lightweight Directory Access Protocol (LDAP) to implement the out-of-band channel for the second authentication factor.

We notice that the notion we use in this paper of trusted service is not coinciding with that of trusted service provider introduced in the European regulation eIDAS [4]. However, we expect that eIDAS trusted service providers can enrich their role in many directions, with the goal of securing society. Among other things, they could play some role also in the context of this paper, just as they themselves could be part of the telephone communication with citizens. Therefore, the need of securing phone communication could also affect eIDAS trusted services.

The structure of the paper is the following. In section 2, we investigate the related literature. The threat model adopted in this paper is defined in Section 3. We describe the proposed solution in Section 4. In Section 5, we describe the implementation of our solution and a proof of concept. The security analysis is provided in Section 6. Finally, in Section 7, we draw our conclusion.

2. Related work

In this section, we provide a brief overview of the work related to our proposal. The problem we face in this paper has been recently considered in the research literature. Moreover, it has been also subject of attention in the realm of Internet Society, in which a number of RFCs were defined to introduce a mechanism, called STIR/SHAKEN [5, 6, 7, 8], aimed to mitigate the risk of phone-number spoofing. Interestingly, the Federal Communications Commission (FCC) has required U.S. telecom providers to implement STIR/SHAKEN. Unfortunately, as stated in [2], the solution is not scalable in real-life contexts and, importantly, it is not applicable to any type of telephone system, as it works only in SIP (i.e., VoIP system).

In the research literature, the state-of-the-art paper is [2]. In this work, the authors propose a caller verification system, called CIV. CIV uses a challenge-response protocol that works entirely withing the telephone system without requiring PKIs and external channels. Importantly, it overcomes the major limitation of STIR/SHAKEN, as it works in every telephone system (both VoIP and non VoIP). However, it has in turn two serious drawbacks. The first is that the system is vulnerable in a threat model in which the attacker is able to intercept calls. The authors themselves note that a powerful adversary can do this through the Law Enforcement Monitoring Facility (LEMF), SIM swap, and SS7 hacking. The second limitation is that the verification process has a very high latency (from 12 to 29 seconds) for both landline and cellular phones, unless a massive change of some features of the telephone system has been applied. In contrast, in our paper, the security of the solution is not affected even in the case the attacker is able to intercept calls. Moreover, the detection is done via a very lightweight LDAP request, so with latency with several orders of magnitude lower than dozens of seconds.

Among approaches that exploit an external channel to perform authentication, the most representative is certainly [3]. In this paper, they propose AuthentiCall, a system designed to support end-to-end mutual authentication and call content integrity over all telephone systems (e.g., landline, cellular, or VoIP). The system uses Internet for data communication, with the scope of establishing a strong cryptography-based handshake between caller and callee when the call is starting, to block it in the

case of failure. Moreover, the audio content of the call is authenticated. Also our paper uses an external Internet connection for authentication. However, our paper overcomes some drawbacks of [3]. The first is that AuthentiCall relies on a trusted server whose role is crucial in the authentication protocol. In contrast, we do not leverage any third trusted party and there is no intermediary between caller and callee. However, such as [3], we rely on a standard PKI. The second aspect that appears advantageous in our paper with respect to [3] is that our approach is much more lightweight and less prone to software security bugs. Indeed, the authentication process only relies on an LDAP request (and a very fast local verification of the downloaded information) instead of the strong cryptography-based handshake of AuthentiCall.

Other approaches that exploit external channels, such as [9, 10, 11], are less meaningful as they use public blockchain (very expensive) or are limited to VoIP systems.

3. Threat Model

In this section, we describe potential malicious attacks that could target an individual in the absence of security measures. Furthermore, we present the security and privacy goals of this work.

Adversary Model

In our threat model we consider the following attacks:

- **Attack 1**: impersonation of a trusted service through the combination of social engineering and phone number spoofing techniques;
- **Attack 2**: impersonation of a trusted service through the interception of a legitimate call from a customer to a trusted service;
- **Attack 3**: man-in-the-middle (MITM) in the communication channel between the user and a trusted service;
- **Attack 4**: modification of the data retained by a trusted service through injection techniques;

Attack 1. An attacker can use a combination social engineering and phone number spoofing techniques to impersonate a trusted service and persuade the victim to provide sensitive information, such as personal credentials, OTPs (One-Time Password) or PINs, or to take specific actions, such as performing bank transfers. These types of attacks can be executed through different communication channels, such as sending e-mails or messages, or making phone-calls that appear to originate from the legitimate entity.

Attack 2. The goal of this attack is the same as **Attack 1**, i.e., to impersonate a trusted service and persuade the victim to provide sensitive information. Differently from **Attack 1**, in this case the attacker compromises the smartphone of the victim, for example via a malicious app. Through this, the attacker can intercept the legitimate call made by the victim to a trusted service and redirect them to a fake trusted service operator.

Attack 3. An attacker can conduct a MITM attack in which they intercept communication between a trusted service and the victim. This attack is possible, for instance, due to the use of insecure communication channels such as open Wi-Fis. In this case, the victim believes they are communicating directly with a trusted service, but actually, data are intercepted by the attacker before reaching the receiver. Through this attack, the attacker can modify the data exchanged in transit between a trusted service and the customer. The goal of this attack is to provide the victim with information letting them trust the incoming call.

Attack 4. Another type of attack may target a trusted service directly. In this case, the attacker can use injection techniques to modify/delete customer information retained by a trusted service. This type of attack can aim to compromise customer information used by our protocol to authenticate the phone calls coming from a trusted service. The objective of the attacker is to block legitimate calls, thus negating the service (Dos attack).

Security and Privacy Goal

Our main security and privacy goal is to guarantee the confidentiality, integrity, and availability of user-sensitive data and, then of the service provided by a trusted service. By mitigating vishing attacks, we prevent malicious users from deceiving victims and obtaining sensitive data. Additionally, we preclude attacks by honest-but-curious individuals from learning about incoming calls to legitimate users from a trusted service. Furthermore, we prevent attackers from modifying the data managed by a trusted service or accessing it without authorization. As there are several types of attacks, our protocol allows us to identify which specific attacks have been carried out.

4. The Proposed Approach

In this section, we describe our solution to mitigate vishing attacks. Roughly, when a user receives a call apparently coming from their Trusted Service, an app installed on the smartphone detects the authenticity of the calling number by a proper online check. This check is done by finding a sort of credential providing that the Trusted Service has actually called the user at that time. Specifically, every time the Trusted Service calls a customer to their registered number, it makes available, via web, a privacy-preserving evidence of the call to that number at that time. This way, the app can detect that the call is vishing if such an evidence is not present. A similar mechanism is activated when a call is originated from the smartphone to the number of the Trusted Service. In this case, when the Trusted Service answers the call, it publishes the credential of the call. This way, if the call is fake, i.e., it is actually directed to the attacker instead of the Trusted Service, the call is blocked because the credential cannot be found.

Now, we formalize the protocol involving both Trusted Service and user's app in a generic fashion. A technology-aware definition of the protocol will be provided in Section 5.

In our scenario, we utilize the notations presented in Table 1.

Table 1
Notations

Notation	Description
TS	The Trusted Service
$\langle P_{TS}, S_{TS} \rangle$	Pair of public and private keys of an asymmetric cryptographic scheme associated with the Trusted Service
h	Cryptographic hash function
S_{TS}	Trusted Service server (a specific publicly readable server location)
n_{TS}	Trusted Service official phone number (assuming the Trusted Service has just one official number)
u	Customer
A_u	Customer's app
n_u	Customer's registered phone number
R_{TS}	Phone-number register of the app, storing the association between the Trusted Service official phone number and the server Trusted Service address
S	Secret seed shared between the Trusted Service and the user
PRNG	Secure pseudo-random number generator
r_k	k-th pseudo-random number generated by the PRNG starting from the seed S
s_i	i-th time slot
τ	Duration of a time slot
BF_i	Bloom filter associated with the i-th time slot
t_0	Starting time
c_t	Number of seconds counted after the starting time at the time of the call (sent or received)

The time is divided into slots of duration τ (in seconds), and every slot is identified by a progressive integer number s_i. Each call coming and received from the Trusted Service is assigned with two distinct contiguous time slots, to manage potential latency issues. Specifically, the slots are chosen as follows. Suppose the call arrives within the slot s_i at the time c_t (counting the number of seconds after the starting time of the system). If $c_t < s_{i-1} \cdot \tau + \frac{\tau}{2}$ (i.e., the call arrives before the half of the slot), then the call is associated with slots s_i and s_{i-1}. Otherwise (i.e., $c_t \geq s_{i-1} \cdot \tau + \frac{\tau}{2}$), the call is associated with slots s_i and s_{i+1} (i.e. the current slot and the next one).

In the setup phase, the Trusted Service TS registers the Customer's registered phone number n_u, and the customer u installs the app A_u, then enabling the sharing of a shared seed S and the proper configuration of a register R_{TS} on A_u. This register stores the association between the Trusted Service official phone number and the server Trusted Service address.

S is generated and stored in a secure memory area of the Trusted Service server S_{TS} and on the smartphone of u. Moreover, for each call to/from u, say it the $k-$th one, the k-th pseudo-random number r_k, generated by the PRNG starting from the seed S, is computed to distinguish possible multiple calls to the same user in the same slot.

We now describe the steps of the protocol by separating the server from the client side.

Server Side. When TS calls n_u or TS answers a call coming from n_u, simultaneously, TS publishes the credential of the call (as defined below) in S_{TS}. Furthermore, TS builds the Bloom filter BF_i by inserting in it the credentials of all the calls of the slot s_i. At the end of s_i, TS digitally signs and publishes BF_i into S_{TS}.

The credential is built as specified below. Consider the k-th call occurring in the i-th slot. The credential is computed through the following hash: $h(r_k||n_u||s_i)$, where r_k is the pseudo-random number, n_u is the phone number of the user, and s_i is the slot to which the call belongs.

Client Side. When u receives the call (in the case of incoming call) or their call is answered (in the case of outcoming call), A_u intercepts it and checks whether the calling/called phone number n_{TS} occurs in R_{TS}. If this is not the case, A_u does not make any operation on the call. Otherwise, A_u has to verify whether there is a vishing/fakecall attempt or a legitimate call.

To do so, A_u finds the address S_{TS}, identifies the two possible slots associated to the call, say s_A and s_B, and computes the credentials at the same way as the Trusted Service does. Then, A_u downloads the credentials published on S_{TS} until now in the involved slots. If one of the credentials is published by the Trusted Service on S_{TS}, this is the proof that the call is actually generated or answered by the Trusted Service. Otherwise, A_u detects the anomaly and blocks the call.

Then, A_u awaits the end of the two time slots and, downloads the corresponding Bloom filters BF_A and BF_B and checks the validity of the signature. After that, A_u queries both Bloom filters to check if one of the credentials is present. In the negative case, the call is not initiated or answered by the Trusted Service. This indicates that there was a vishing/fakecall attempt.

Otherwise, it means that a malicious actor modified the list of the credentials published by the Trusted Service at S_{TS}, by dropping the expected credential. This is then another type of anomalous situation, which can be identified as a DoS (Denial of Service) attempt. Indeed, the malicious actor caused the deny of the call (incoming or outcoming) truly originated/answered by the Trusted Service. The app sends a proper notification to the Trusted Service.

5. Implementation and Proof of Concept

In this section, we describe the software demonstrator we developed, which implements the core functions of the approach proposed in Section 4. This section aims to provide the reader with a proof of concept to show the technical feasibility of the proposal.

The general architecture of the solution is composed of TS, LDAP, and a client application.

Therein, the telephone system of the TS (assumed integrated with the information system) manages the interactions with customers by intercepting the outcoming calls over the telecommunications network, and simultaneously interfaces with the LDAP server to publish hashes and Bloom filters,

which are provided to the users. The same happens for incoming calls (originated from the customers) when the TS operator answers the call.

The LDAP server acts as a repository, stores hashes and Bloom Filters, and makes them accessible to the customer application. The customer application, located on their device, receives/starts calls via the telecommunication network. Simultaneously, it connects to the LDAP server through the Internet to download the published hashes and check the validity of the call.

The demonstrator consists of three modules: (1) the back-end module, (2) the server-side module, and (3) the client-side (mobile) application. At this stage, our proof of concept is focused on the anti-vishing solution that intercepts only incoming calls in the client-side module.

5.1. Back-end module

We assume that the set-up phase is composed of a first step in which the TS registers the phone number of the customer by using a secure communication procedure (in principle, even in person) that is outside the function of our software system. Therefore, we focus on the implementation of the second step of the set-up phase, which consists just of the generation of the seed, as we skip for simplicity the sharing with customer.

The implementation is developed in Java, uses cryptography hashes, and a probabilistic data structure based on a Bloom Filter. To both salt hashes and authenticate messages, we use a secure Pseudo-Random Number Generator (PRNG), implemented through the class `SecureRandom`, configured with Deterministic Random Bit Generator (DRBG) that uses a defined seed (chosen compatible with the used PRNG). The `java.security.MessageDigest` library is used to compute SHA-256 digests. To manage and verify the integrity of the slots, we use a signed Bloom Filter, which is implemented through the Google Guava library (com.google.common.hash.BloomFilter e com.google.common.hash.Funnels). This library is configured to store up to 10,000 items with a false positives probability of 10^{-7}. This results in a Bloom filter size of 40.95KiB.

The credentials are generated by combining a sequence of 256 bytes (derived from the PRNG), the customer's phone number, and the current temporal slot. The resulting hashes are added to the Bloom Filter and, at the end of each time slot, the filter is serialized in a compact format using the java.io.ByteArrayOutputStream class, which allows us to store it on an external server.

The system synchronization is guaranteed by a mechanism that aligns operations with the start cycle of the next time slot, thus guaranteeing the correct resource allocation timing.

5.2. Server-side module

For managing the publication of hashes, we use a Lightweight Directory Access Protocol (LDAP) server, which is a protocol designed for directory administration and remote consultation. Specifically, we use OpenLDAP, which is an open-source and complete implementation of LDAP.

The use of LDAP is motivated by several key advantages, including ease of implementation and interoperability, with a view towards a potential extension of the system into a multi-organization and open-environment framework. LDAP provides integrity assurance and authenticity of information through mechanisms such as message authentication codes, digital signatures, and PKI. This way, it reduces dependence on the security of software modules, such as API, to allow access to data. Furthermore, the protocol prepares the solution for open-environment scenarios, where, similarly to the CRL of X.509 certificates, the LDAP address can be integrated into a certificate to guarantee its authenticity. Also, it is designed to be highly scalable, supporting large directories with thousands or millions records, while maintaining high performance by optimising for frequent read operations and efficient caching. Although for the scenario currently considered in this paper, this aspect could appear little meaningful, it can be important when we move to multi-organization scenarios in which scalability becomes a key factor. Finally, LDAP is lightweight, suitable for modest hardware, and integrates easily with existing LDAP infrastructures, making it an ideal choice for complex, distributed solutions.

The LDAP structure represents a hierarchical organization, with a basic domain identified as *dc=mydomain,dc=local*, which acts as an access point for all searches and represents a top-level organizational entity. Within the domain, the individual TS is represented by an organizational unit (OU), which further subdivides the resources into logical subunits denoted as *slot1* and *slot2*. Each unit is described through attributes that describe its type (*objectClass*) and other details. Digests are stored as *description* attributes. Specifically, there is one description per hash (i.e., per call).

5.3. Client-site application

The client-side application is a mobile Android application written in Kotlin [12], which implements an automatic incoming-call detection system designed to identify and block vishing attacks. We chose Android for the availability of libraries and software development platforms. Obviously, a similar implementation could be done in iOS.

The main class, `MainActivity`, extends the custom interface `BroadCastReceiver.OnIncomingCallListener`. Furthermore, during initialization (done by the method `onCreate`), the application requires the necessary permission to access phone state and call logs. A `BroadcastReceiver` is registered to intercept events related to incoming calls. Subsequently, the app verifies whether it has the required permission to access `READ_PHONE_STATE` and `READ_CALL_LOG`. If this permission is not granted, then the app requires it from the user. When a call is received, the activated `BroadcastReceiver` forwards the phone number to the `onIncomingNumber` function. If the number is associated with the bank, then the application computes two time slots. The current time slot is the one associated with the timestamp recorded by the application at the moment of the call is identified. This time slot is computed by using the timestamp Unix, which is the number of seconds since 1 January 1970. As we set the slot duration time to 5 minutes, the timestamp Unix is divided by 300 (i.e., 5 minutes).

Then, as described above, the application chooses the previous or the next slot, depending on where the exact calling time is placed within the slot. Then, a new (deterministic) random number is generated using the `SecureRandom` library Subsequently, as described above, the application computes the two digests using the SHA-256 hashing function.

At this point, the application verifies whether at least one of the locally computed hashes occurs in the list of hashes published in the LDAP server. If none of the computed hashes is found, then the application shows a push notification alerting the user that the incoming call is anomalous, as shown in Figure 1.

In this case, at the end of the next time slot, the app deserializes and checks the Bloom filters associated with the two identified time slots. Deserialization and searching are done through the Google Guava library. If at least one digest is found, the app shows another push notification to the user to signal that a DoS attack has been detected. Otherwise, no further notification is shown.

6. Security Analysis

In this section, we analyze how the proposed protocol mitigates attack scenarios of our threat model (Section 3).

6.1. Attack 1: Impersonation via Social Engineering and Phone Number Spoofing

Attack Description. An attacker impersonates a trusted service using phone number spoofing and social engineering to deceive the victim into revealing sensitive information or performing unintended actions, such as transferring money.

Mitigation by the Protocol. Our protocol effectively prevents such attacks through the use of unforgeable credentials:

- When the attacker initiates a spoofed call, the app A_u intercepts the incoming call and checks if the calling number n_{TS} is listed in the trusted register R_{TS}.

Figure 1: Push notification signaling a malicious attempt.

- If n_{TS} is present in R_{TS}, the app queries the Trusted Service's server S_{TS} to verify the presence of the credential associated with the call in the related slots s_A and s_B.
- Since the attacker lacks access to the shared seed S and the pseudo-random numbers r_k, they cannot generate any correct credential.
- The absence of valid credentials leads to the detection of the attack, and the app blocks the call.

6.2. Attack 2: Smartphone Compromise and Redirected Calls

Attack Description. The attacker compromises the victim's smartphone via a malicious app, enabling them to intercept and redirect legitimate calls made by the victim to a fake trusted service.
Mitigation by the Protocol. This attack is thwarted as follows:

- Even if the attacker redirects the call to a fake operator, the app A_u on the victim's smartphone will verify the authenticity of the trusted service's response.
- Upon connection, A_u identifies the corresponding slots s_A and s_B and computes the credentials.
- The app then queries S_{TS} for the presence of these credentials. Since the fake operator cannot access the seed S or generate the pseudo-random numbers r_k, the credentials will be absent.
- In such cases, the app terminates the call and notifies the user of a potential redirection attempt.

6.3. Attack 3: Man-in-the-Middle (MITM) Attacks

Attack Description. The attacker intercepts communication between the victim and the trusted service, potentially modifying data to mislead the victim into trusting a fake call.
Mitigation by the Protocol. The protocol is designed to counteract MITM attacks as follows:

- Even if the attacker intercepts the communication between A_u and S_{TS}, they cannot generate valid credentials due to the lack of access to S and r_k. Thus, vishing is prevented.

6.4. Attack 4: DoS Attack Targeting the Trusted Service

Attack Description. The attacker targets the Trusted Service by modifying or deleting customer information, disrupting the protocol's ability to verify calls.
Mitigation by the Protocol. The protocol mitigates this risk as follows:

- The use of digitally signed Bloom filters ensures that even if data is deleted or modified by an attacker, A_u can detect discrepancies. If the expected credential is present in BF_A or BF_B, it means that it was previously removed by the attacker. Then, the app raises an anomaly flag and notifies the Trusted Service of a Dos Attack at the end of the time slot.

7. Conclusion

In this paper, we propose a solution to face the problem of fake communication between customers and trusted services. The presented approach uses an external data communication channel to perform a sort of double-factor authentication of the call (either incoming or outcoming) and is secure also in the case of compromised telephone communication. We present also a proof of concept of the solution based on the implementation of its core functions. Even, for simplicity, we assume in this paper that the system works for one trusted service and its customers, in principle, no methodological changes are necessary if we move from a single trusted service to multiple trusted services. However, we plan to extensively develop the multi-organization model as a future work. Other future work is to complete the implementation and to formally prove the security of our approach.

Acknowledgments

This work is partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU and partially supported Agenzia per la Cybersicurezza Nazionale under the programme for promotion of XL cycle PhD research in cybersecurity – C36E24000080005. The views expressed are those of the authors and do not represent the funding institutions.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. Bidgoli, J. Grossklags, "hello. this is the irs calling.": A case study on scams, extortion, impersonation, and phone spoofing, in: 2017 APWG Symposium on Electronic Crime Research (eCrime), IEEE, 2017, pp. 57–69.

[2] S. Wang, M. Delavar, M. A. Azad, F. Nabizadeh, S. Smith, F. Hao, Spoofing against spoofing: Toward caller id verification in heterogeneous telecommunication systems, ACM Transactions on Privacy and Security 27 (2023) 1–25.

[3] B. Reaves, L. Blue, H. Abdullah, L. Vargas, P. Traynor, T. Shrimpton, {AuthentiCall}: Efficient identity and content authentication for phone calls, in: 26th USENIX Security Symposium (USENIX Security 17), 2017, pp. 575–592.

[4] European Union, eIDAS: electronic IDentification, Authentication and trust Services, 2014. URL: http://data.europa.eu/eli/reg/2014/910/oj, (Accessed 20 February 2024).

[5] J. Peterson, C. F. Jennings, E. Rescorla, C. Wendt, Authenticated Identity Management in the Session Initiation Protocol (SIP), RFC 8224, 2018. URL: https://www.rfc-editor.org/info/rfc8224. doi:10.17487/RFC8224.

[6] C. Wendt, J. Peterson, PASSporT: Personal Assertion Token, RFC 8225, 2018. URL: https://www.rfc-editor.org/info/rfc8225. doi:10.17487/RFC8225.
[7] J. Peterson, S. Turner, Secure Telephone Identity Credentials: Certificates, RFC 8226, 2018. URL: https://www.rfc-editor.org/info/rfc8226. doi:10.17487/RFC8226.
[8] C. Wendt, M. Barnes, Personal Assertion Token (PaSSporT) Extension for Signature-based Handling of Asserted information using toKENs (SHAKEN), RFC 8588, 2019. URL: https://www.rfc-editor.org/info/rfc8588. doi:10.17487/RFC8588.
[9] D. Hou, H. Han, E. Novak, Taes: Two-factor authentication with end-to-end security against voip phishing, in: 2020 IEEE/ACM Symposium on Edge Computing (SEC), IEEE, 2020, pp. 340–345.
[10] Y. Chen, Y. Wang, Y. Wang, M. Li, G. Dong, C. Liu, Callchain: Identity authentication based on blockchain for telephony networks, in: 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, 2021, pp. 416–421.
[11] I. M. Tas, S. Baktir, Blockchain-based caller-id authentication (bbca): A novel solution to prevent spoofing attacks in voip/sip networks, IEEE Access (2024).
[12] D. Jemerov, S. Isakova, Kotlin in action, Simon and Schuster, 2017.