SIPConfusion: Exploiting SIP Semantic Ambiguities for Caller ID and SMS Spoofing

Qi Wang*, Jianjun Chen*✉, Jingcheng Yang*, Jiahe Zhang*, Yaru Yang*, Haixin Duan*
* Tsinghua University

***Abstract*—Session Initiation Protocol (SIP) is a cornerstone of modern real-time communication systems, powering voice calls, text messaging, and multimedia sessions across services such as VoIP, VoLTE, and RCS. While SIP provides mechanisms for authentication and identity assertion, its inherent flexibility poses the risk of semantic ambiguity among implementations that can be exploited by attackers.**

In this paper, we present SIPCHIMERA, a novel black-box fuzzing framework designed to systematically identify ambiguity-based identity spoofing vulnerabilities across SIP implementations. We evaluated SIPCHIMERA against six widely used open-source SIP servers—including Asterisk and OpenSIPS—and nine popular user agents, uncovering that attackers could spoof their identity via manipulating identity headers and circumvent authentication. We demonstrate the real-world impact of these vulnerabilities by evaluating five VoIP devices, seven commercial SIP deployments, and three carrier-grade RCS-based SMS platforms. Our experiments show that attackers can exploit these vulnerabilities to perform caller ID spoofing in VoIP calls and send spoofed SMS messages over RCS, impersonating arbitrary users or services. We have responsibly disclosed our findings to affected vendors and received positive acknowledgments. We finally propose remedies to mitigate those issues.

I. INTRODUCTION

Session Initiation Protocol (SIP) is a critical component for modern communication systems. It is the signaling standard behind Voice over IP (VoIP), Voice over LTE (VoLTE), Rich Communication Services (RCS), and many enterprise-grade telephony infrastructures. Whenever users make a mobile call over LTE, receive a business support call, or use an app like Linphone for VoIP, SIP is coordinating the session behind the scenes. In recent years, the adoption of SIP has become near-universal—even SMS is being replaced by SIP-based messaging through RCS, now used by over one billion users globally [1].

Among the serious threats to SIP-based systems is SIP identity spoofing, commonly exploited for caller ID and SMS spoofing attacks[2]. These attacks enable a wide range of malicious activities, including impersonation, telecom fraud, and large-scale scams. To address this, the community has developed various mechanisms, such as authentication and identity assertion, to mitigate such attacks[3], [4], [5]. Both industry and regulators have rolled out countermeasures such as the STIR/SHAKEN framework, which enforces cryptographic signing of caller identities using public key infrastructure (PKI) [6]. These developments have significantly raised the bar for traditional spoofing techniques, rendering traditional methods, such as simply modifying the caller ID number, ineffective in practice.

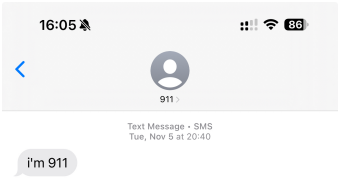


Fig. 1: A real-world example of SMS spoofing that impersonates 911 by exploiting SIP semantic ambiguities.

In recent years, increasing attention has been directed toward semantic ambiguities—subtle inconsistencies among implementations that can be exploited by attackers [7], [8]. These attacks arise not from the absence of security features, but from discrepancies in interpretation. An attacker can craft malicious SIP messages to cause different entities (e.g., proxies, user agents) to misinterpret identity-related headers, thus bypassing the authentication defense to execute spoofing attacks. In this paper, we systematically investigate such vulnerabilities in the SIP protocol. Specifically, our research aims to answer three key questions: 1) How can ambiguity-based spoofing vulnerabilities be systematically identified across SIP implementations? 2) What types of ambiguity-based vulnerabilities exist in SIP implementations, and how prevalent are they? 3) What are the real-world security implications of such vulnerabilities in operational SIP infrastructures?

To address these questions, we present SIPCHIMERA, a novel fuzzing framework designed to systematically identify spoofing vulnerabilities arising from SIP semantic inconsistencies. Using SIPCHIMERA, we conducted a large-scale evaluation of six popular SIP proxy servers and nine popular SIP user agents. Our analysis of the results revealed a wide range of semantic inconsistencies that affect nearly all server-client combinations, covering scenarios where the attacker either possesses or lacks valid credentials.

To assess the real-world impact, we conducted proof-of-concept experiments on three representative SIP-powered

✉ Corresponding author: jianjun@tsinghua.edu.cn

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.230116
www.ndss-symposium.org

platforms, including five VoIP devices, seven commercial SIP services, and three RCS-based SMS services in major carrier networks that have billions of users. Our evaluation revealed protocol ambiguities that can be exploited for identity spoofing, enabling caller ID and SMS spoofing attacks. Figure 1 shows an attack impersonating 911. We disclosed our findings to affected open-source projects like Asterisk, as well as affected commercial operators, receiving positive acknowledgments.

In summary, we make the following contributions:

- We design and implement SIPCHIMERA[1], a novel framework to automatically discover ambiguity-based identity spoofing vulnerabilities across SIP implementations.
- We conducted the first systematic study to identify ambiguity-based spoofing vulnerabilities in SIP implementations. Our evaluation of six open-source SIP servers and nine user agents revealed ambiguity-based spoofing vulnerabilities that affect nearly all server-client combinations, covering scenarios with both authenticated and unauthenticated attackers.
- We present real-world case studies in production VoIP and RCS-based SMS infrastructures showing how these semantic ambiguities lead to caller ID and SMS spoofing in practice. We responsibly disclosed our findings to affected vendors and received positive acknowledgments.

II. BACKGROUND

A. Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) is a signaling protocol widely used for initiating, managing, and terminating real-time communication sessions over the Internet. SIP operates at the application layer and is designed to be independent of the underlying transport layer, which means it can work over various transport protocols, including UDP and TCP.

SIP was designed by the Internet Engineering Task Force (IETF) and first standardized in 1999 under RFC 2543[9]. In 2002, the revised standard RFC 3261[10] was published, which remains the core specification for SIP. SIP uses a text-based message format similar to the Hypertext Transfer Protocol (HTTP), making it human-readable and easy to debug. For example, as illustrated in Fig. 2, the first line indicates the method, the target URI, and the SIP version. The subsequent consecutive lines form the headers, and after an empty line, there is the body of the message. It also supports a wide range of features and extensions, such as call forwarding, conferencing, and presence information, which can be implemented using additional SIP methods and headers.

SIP is designed to be a client-server protocol. The main components in a SIP architecture include User Agents (UAs) and SIP Servers. The UAs can be either a User Agent Client (UAC) or a User Agent Server (UAS). A UAC initiates SIP requests, while a UAS responds to these requests. SIP Servers, such as Proxy Servers and Registrar Servers, act as intermediaries in the communication process, each with

[1]SIPCHIMERA is available at https://github.com/EkiXu/SIPChimera

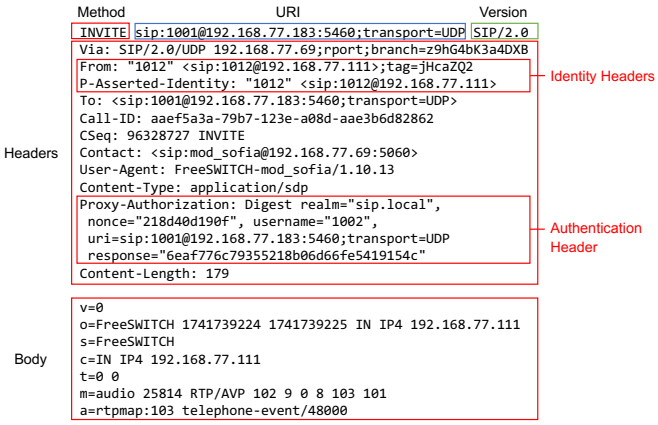


Fig. 2: Structure of an INVITE SIP Message.

distinct purposes. Proxy Servers forward SIP requests between UACs and UASs, often performing tasks like authentication, authorization, and routing. Registrar Servers are responsible for registering the location of UAs. When a user's device connects to a SIP network, it registers its location with the Registrar Server so that incoming calls can be routed correctly.

Fig. 3 shows an example where Alice makes a SIP call to Bob. The SIP message flow for a basic call is as follows: Alice (UAC) sends an INVITE request to Bob (UAS). This request contains information about the call, such as the media types the caller can support (e.g., audio, video). The INVITE request may pass through one or more Proxy Servers for routing and authentication purposes. Upon receiving the initial request, the proxy server responds with an Authentication Challenge to request credentials from the UAC and subsequently forwards the message to the UAS after identity verification success. The 100 Trying provisional response indicates that the SIP Proxy server has sent the INVITE request to the UAS and is waiting for the UAS to process it. The UAS then responds with a series of status codes. The final response, 200 OK, means that the UAS is willing to accept the call. The UAC then sends an ACK message to confirm receipt, and the call is established. Before and after the call is established, the UAC and UAS may transfer additional information through other Methods, such as REFER, UPDATE, and INFO. Besides calling, RFC 3428[11] defines the MESSAGE method for SIP-based instant messaging without setting up a session. MESSAGE requests carry text or binary data and, like INVITE, may traverse proxies. Upon receipt, the UAS responds with 200 OK if successful. MESSAGE enables real-time text communication within SIP networks, independently of media sessions.

B. SIP Authentication & Identification

As described in RFC3261[10], the authentication mechanism in SIP is primarily implemented using the digest challenge-response method. This approach verifies whether a user is authorized to access SIP services by requiring them to respond correctly to a nonce value using a shared secret. The credentials involved in this process, typically embedded

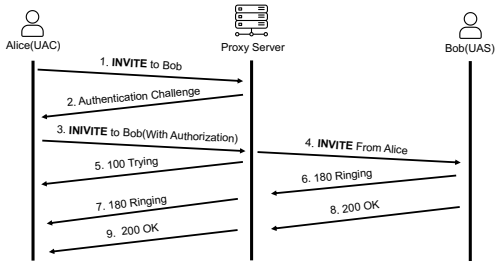


Fig. 3: Typical SIP INVITE Workflow.

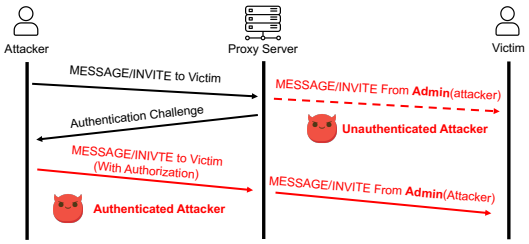


Fig. 4: Threat Model.

in the Proxy-Authorization or Authorization headers, are used to validate the identity of the user for service access control. The former is used when a User Agent directly authenticates with a User Agent Server (UAS), while the latter is used when authenticating with an intermediate Proxy Server.

Notably, SIP's identity mechanism operates independently from this authentication process. As illustrated in Fig. 2, a typical SIP message includes separate authentication and identity headers. The initial SIP protocol simply used the From header to indicate sender identity and does not mandate any binding relationship with the authentication credentials. Such a design could help protect the privacy of the sender, but attackers can easily forge the identity by tampering with the From header. To balance privacy and security, in 2002, an IETF draft[12] on SIP Privacy Mechanisms proposed the introduction of the *Remote-Party-ID* (RPID) header. This header keeps the actual identity of the sender within trusted networks and automatically gets removed when forwarded to untrusted entities. Such a mechanism was later formalized and improved in RFC 3325[13] as the *P-Asserted-Identity* (PAI) header. Users could use *P-Preferred-Identity* (PPI) to declare their preferred identity, and trusted entities would insert an appropriate PAI header to identify the user after authentication.

However, such identity headers didn't get protected by cryptographic mechanisms and leave the opportunity for attackers to forge identities. To address these issues, RFC4474[14] introduced two new headers, Identity and Identity-Info, along with a new authentication service entity. The Identity header contained a signature of the From header and the message body, while the Identity-Info header was a URI from which the corresponding certificate could be obtained. Further RFCs, including RFC7340[15], RFC8224[16], RFC8225[17], and RFC8226[18] jointly proposed the Secure Telephony Identity Revisited/Signature-based Handling of Asserted information using toKENs (STIR/SHAKEN) scheme. This scheme abandoned the approach in RFC4474[14] and redefined the Identity header for storing signature-related information.

The high cost and complexity of cryptographic mechanisms have limited the adoption of STIR/SHAKEN primarily to North American operators [19], [20]. Consequently, most SIP deployments continue to depend on legacy standards. These standards often overlap, creating a fragmented ecosystem where different devices and servers implement varying subsets or interpretations, resulting in semantic ambiguities.

III. Overview

A. Threat Model

Generally, the process of Alice calling or sending messages to Bob via a SIP proxy can be divided into three distinct phases: (1) *Session Initiation:* The User Agent Client (UAC) of Alice sends an initial SIP request through a proxy server to Bob. At this point, the proxy server typically authenticates the originator before the request is processed. (2) *Authentication Challenge:* The UAC resubmits the request with valid authentication credentials. The proxy server verifies credentials and originator identity. (3) *Message Delivery:* If credentials and identity verification succeed, the proxy server forwards the request toward the User Agent Server (UAS) of Bob, potentially adding identity header fields to reliably convey the authenticated identity of Alice.

In this study, we define our threat model based on the remote attacker who sends malicious SIP messages to the victim from UAC, exploiting the inconsistencies between SIP proxies and user agents to spoof the chosen identity before the victim discovers the attacker's real identity. In such a scenario, attackers don't need to compromise or impersonate the proxy server, and the victim's UAS will implicitly trust messages forwarded from the proxy server without re-verifying the authentication.

We further consider two types of spoofing attackers according to their capabilities: *unauthenticated attackers* and *authenticated attackers*. As depicted in Fig. 4, the *unauthenticated attackers* do not have legitimate credentials or an account within the target SIP services, which means they must bypass the authentication challenge phase.

The *authenticated attackers* have gained access through the compromised account, either by registering accounts or using the guest account, or exploiting leaked credentials. Unlike bulk registration of ordinary user accounts for spam attacks, such attackers could impersonate high-value targets such as administrators or other trusted leadership figures, with significant implications for targeted social engineering attacks.

B. Motivating Example

As presented in Fig. 1, we identified a real-world SMS spoofing scenario that affected a major mobile operator with over one billion users. The attacker who owns a phone number could send texts as if from an arbitrary user e.g., 911. In Fig. 5, we show the details of this attack technique.

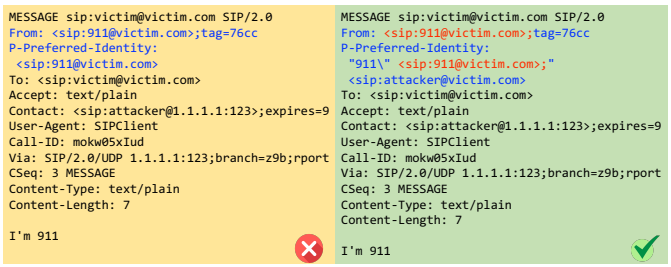


Fig. 5: A motivating example of spoofing involves an attacker sending text messages to a victim's phone, making it seem as though they are from 911. The malformed SIP request on the right bypassed the identification mechanisms of affected operators.

For anonymization purposes, we use `victim.com` to represent the operator's SIP domain, `attacker` for the attacker's phone number, `victim` for the victim's phone number, and `911` for the impersonated identity. The left-hand message reveals that the attacker simply replaced the account address in the identity headers with `911@victim.com` instead of `attacker@victim.com`. Normally, the server will reject such a request because the request's credentials do not match the identity header. However, we found some malformed messages, like the one on the right, could bypass the identification mechanism. Specifically, the SIP server interprets the slash as an escape character. Thus, the double-quote character following the slash is escaped. For the SIP server, it parses the result where `911" <sip:911@victim.com>` is the display-name, and the attacker's address is considered the real sender address, allowing the message to pass verification. However, on the victim's side, the double-quote character is not escaped. For the victim, the display name appears as `911\`, and the address seems to be from the 911 account. Using this technique, an attacker can send any text to a remote victim, making it appear to come from an arbitrary user.

C. Research Questions

From the above description of the attack principles, the key point of these attacks is to craft malformed SIP messages, which can lead to inconsistencies between the SIP servers and the SIP clients. Therefore, this paper aims to answer the following research questions:

RQ1: How to systematically identify ambiguity-based spoofing vulnerabilities in SIP implementations? SIP is a rich and extensible protocol for initiating and managing communication sessions. However, such flexibility also introduces potential for semantic ambiguity, particularly when SIP messages deviate from the ABNF-defined grammar. Manually identifying such ambiguities across diverse implementations is challenging due to the complexity of the protocol's standards. Although there are studies [21] focused on fuzzing SIP protocols, they are focusing on memory-related vulnerabilities such as Denial-of-Service (DoS) attacks and cannot detect spoofing cases, as these do not trigger memory crashes.

To address this gap, we propose a generative fuzzing framework, SIPCHIMERA, designed to uncover ambiguity-based spoofing vulnerabilities through systematically analyzing discrepancies across SIP implementations. SIPCHIMERA leverages grammar extracted from SIP-related specifications and employs both grammar-level and byte-level mutations to generate high-coverage test samples. These samples aim to provoke unintended and inconsistent behavior across implementations, thereby revealing underlying ambiguities. We will describe the details of SIPCHIMERA in Chapter IV.

RQ2: What types of ambiguity-based spoofing vulnerabilities exist across different implementations, and what is their prevalence? To investigate the diversity and prevalence of ambiguity-based spoofing vulnerabilities within the SIP ecosystem, we evaluated SIPCHIMERA across a comprehensive range of SIP implementations, encompassing both server-side and client-side components. Our testbed includes six widely used SIP servers and nine popular user agents. The evaluation reveals that over 80% (47 out of 54) of server and user agent combinations are susceptible to ambiguity-based spoofing attacks. We identified and categorized four distinct scenarios of spoofing vulnerabilities. By grouping these ambiguities based on their root causes, we offer insights into common misinterpretations of SIP specifications.

RQ3: What are the real-world security implications of ambiguity-based spoofing vulnerabilities? Bridging the gap between theoretical vulnerabilities and practical exploitation is essential for accurately assessing risk. We investigate this question by conducting responsible testing of discovered ambiguities against real-world SIP services, including seven public SIP services and RCS platforms provided by three large operators. The result shows that ambiguity-based spoofing is widespread in real-world SIP services. Additionally, we propose three strategies to mitigate these vulnerabilities.

IV. METHODOLOGY

A. Workflow

Fig. 6 shows the framework of SIPCHIMERA, which can be divided into six steps: (1) The *Extractor* module extracts ABNF/BNF rules from the RFC documents related to the SIP protocol and converts them to production rules. (2) The grammar rules will guide how the *Generator* module produces initial samples to get high-quality samples that follow the SIP standards. (3) After the *Corpus* module has obtained the initial samples, each of them will be mutated by the *Mutator* Module, so that the test samples can be slightly off track and hopefully trigger unexpected logical flow. (4) The mutated sample will be sent to the SIP server under test, and if the request gets accepted by the SIP server, it will naturally be forwarded to the SIP user agent. (5) To improve the testing efficiency, the *Scheduler* module will collect the response from the SIP server and choose which sample will be stored and wait for the next mutation turn in the *Corpus* module. (6) After the client has received the SIP request, the *Validator* will check whether the spoofing attack has succeeded.

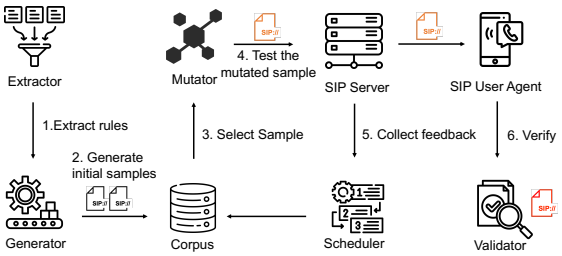


Fig. 6: SIPCHIMERA Workflow.

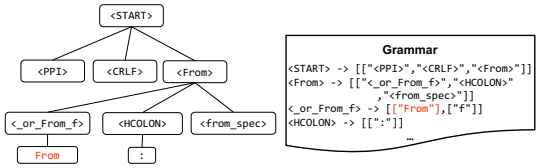


Fig. 7: Generating a new sample with grammar rules.

B. Extractor

The *Extractor* module parses ABNF (Augmented Backus-Naur Form) rules from SIP-related RFCs and converts them into a standardized rule format for test case generation. Each rule is identified by a non-terminal (e.g., `<A>`) and maps to one or more alternatives, where each alternative is a sequence of literals or other non-terminals. For example, the rule `A = "B" / "C"` becomes `<A>= [["B"], ["C"]]`. Non-quoted identifiers are interpreted as rule references, and will be converted to the corresponding non-terminal symbols that are enclosed in angle brackets.

The parsing process proceeds in four phases: tokenization, syntactic analysis, rule expansion, and deduplication. Tokenization removes comments and whitespace, producing a stream of normalized tokens. Syntactic analysis uses recursive descent with stack-based handling of nested structures. To support constructs including alternation, repetition, and optional, the parser introduces auxiliary rules to preserve grammar structure and maintain context-freeness. As illustrated in Fig. 7, alternation rules, which are denoted by the `/` operator in ABNF, are parsed into explicit branching choices. For instance, the rule `"From" / "f"` is transformed into `<_or_From_f> -> [["From"], ["f"]]`, enabling the generator to produce multiple syntactic variants for the same logical construct. All parsed rules will be automatically deduplicated and stored in a central dictionary.

C. Generator & Mutator

The *Generator* and *Mutator* modules are responsible for producing high-quality test samples. The generator will utilize the rule set extracted by the *Extractor* to generate corresponding testing samples. Specifically, the generator will accept a given initial derivation rule for the `<START>` symbol to generate testing samples. The generator will then start from the `<START>` node and, according to the input grammar derivation rules, randomly select one valid derivation for each node until all nodes reach terminal positions. Fig. 7 shows a part of the generation process that sample is expanded from the `start` node according to the derivation formula, when it encounters a repetition or alternation node, the generator randomly selects one rule to expand, for example, the `field-name` of the From header is selected among `From` and `f`. This process results in an initial syntax tree and its corresponding test sample. For the cryptographic header (et al., Identity), we follow the specification to calculate the corresponding value.

Leveraging the hierarchical information of the syntax tree nodes, the mutator can achieve grammar-level mutations by duplicating, deleting, or moving subtrees. At the byte-level, mutations are applied to terminal nodes by modifying their string content with deletion, insertion, or encoding transformations. These mutations produce test samples that intentionally deviate from the SIP specification in minor ways, enabling effective evaluation of SIP implementations' compliance and resilience against ambiguity-based spoofing attacks.

D. Scheduler

The *Scheduler* module is designed to schedule the queue of test samples maintained by the *Corpus* module. In each test round, the module selects and dispatches the foremost test sample in the queue to the SIP server.

To improve the quality of test samples, this module utilizes SIP server response status codes as feedback. Specifically, a response with a success status code is interpreted as confirmation that the current sample has been accepted by the target SIP server. If a sample fails to traverse the SIP server, it will be removed from the queue, as it will not reach the user agents and thus cannot trigger ambiguity-based spoofing vulnerabilities. For those samples that successfully pass through the SIP server, the *Scheduler* module will further mutate them to enhance the likelihood of inducing resolution ambiguity. These promising samples are therefore reinserted into the test queue for subsequent mutation cycles. Moreover, when the testing queue is exhausted, the *Scheduler* module invokes the *Generator* module to synthesize new test samples and continue the testing process.

E. Validator

The *Validator* module is responsible for verifying whether the test case has successfully caused an ambiguity-based spoofing attack. We implement identification for both SIP-based SMS communication and telephone communication with a criterion that the test message is displayed in the message or call history of the forged user.

For SMS messages, to correlate the test sample with the client display messages, we add the serialized data of the test sample, along with the Call-ID of the network request, to the message, so that we can directly ascertain the original structure of the test sample and the actual network request to detect whether the samples are getting normalized by the SIP server.

In the context of telephony, the framework is programmed to transmit mutated call messages followed by a CANCEL

message upon receiving the 180 Ringing status code to simulate the dialing process and automatically terminate the call. By capturing the client's caller history, the time corresponding to successful test samples will be recorded so that the corresponding samples can be identified based on the timestamp, and in addition, these successful samples will be retested to prevent false alarms.

V. Evaluation and Findings

A. Evaluation

1) Core SIP Methods & Headers Selection: In our evaluation, we focus on SIP methods and headers that could affect the identity presentation on the victim's side. The selection process is guided by two main criteria: such methods or headers (i) contain caller identity information, and (ii) are triggered before victims answer calls or receive messages.

For method selection, we conducted a systematic analysis of all 14 standard SIP methods. The results are summarized in Appendix Table VII. Based on this analysis, we identified six SIP methods that are relevant to our threat model, including INVITE, MESSAGE, REGISTER, UPDATE, PUBLISH, and REFER. The remaining 8 methods are either passive responses, used after session establishment, for capability probing, or serve purposes unrelated to identity presentation when victims make trust decisions.

During the executor phase, only identity-related headers are generated, while the remaining components of the SIP request message are populated using a valid template. For header selection, we gather all SIP-related RFC documents and extract ABNF grammar rules. We modify the `<user>` definition to `<fake_sender>` and `<sender>` symbols, ensuring spoofing identity applies to all identity-related headers. Furthermore, we manually examined the specifications to uncover any additional headers that may carry identity information. As discussed in Appendix Table VIII and illustrated in Appendix Listing 1, all these headers are appended to the `<START>` symbol. If a sample lacks the `<fake_sender>` node, it is regenerated to ensure potential exploitation of ambiguity-based spoofing vulnerabilities.

2) Setup: To evaluate SipChimera and find inconsistencies in a controlled environment, we built our testbed with SIP proxy servers and user agents. By exploring the SIP topic[22] on GitHub, we selected six different open SIP proxy servers with three criteria: (1) run independently, (2) support both text and calls, and (3) verify user credentials. For the SIP user agents, we selected nine popular free softphones that support the sending and receiving of texts and calls based on a SIP service provider rank list[23]. We listed the test target SIP proxy server in I and SIP user agents in Appendix Table II.

In our evaluation, we register three types of accounts: the attacker account that is owned by the attacker, the victim account, and the admin account that plays the role of the who the attacker wants to impersonate.

For each SIP server pair under test, we register the victim accounts at SIP user agents. We configured the SIP servers under test to follow the best security practices written in their official documentation. Before the test, we use the attacker and the admin account to send probe messages to the user agents under test. These messages are used to mark the messages that the correct identity of the attacker and the admin should display on the victim's side. Following this initialization, the SipChimera executor logs into the SIP server with the attacker account. In each testing round, the executor sent the test message sequence to the user agents under test to start a call or send a message.

A spoofing attack was considered successful if the corresponding text or call appeared in the message history under the identity of the admin account, indicating that the user agent displayed the spoofed identity as valid. Specifically, for methods not directly used to REGISTER, PUBLISH, REFER, and UPDATE, we retest the success samples to confirm that the spoofing is caused by these types of methods rather than by others in the session.

We conducted our experiments on a Linux server equipped with a 2.5GHz 64-core CPU and 128 GB of RAM. Considering that some SIP servers handle the INVITE and MESSAGE methods differently, we conducted grouped testing based on the request method. In each test group, one specific SIP server and one method were selected. For each round, we run SipChimera with up to 50,000 samples.

3) Result: Across a total of 16,200,000 samples, 701,676 samples are successfully forwarded to the user agent, and we found 8442 instances that successfully triggered ambiguity-based spoofing attacks. We summarized the number of successful spoofing in Appendix Table IX.

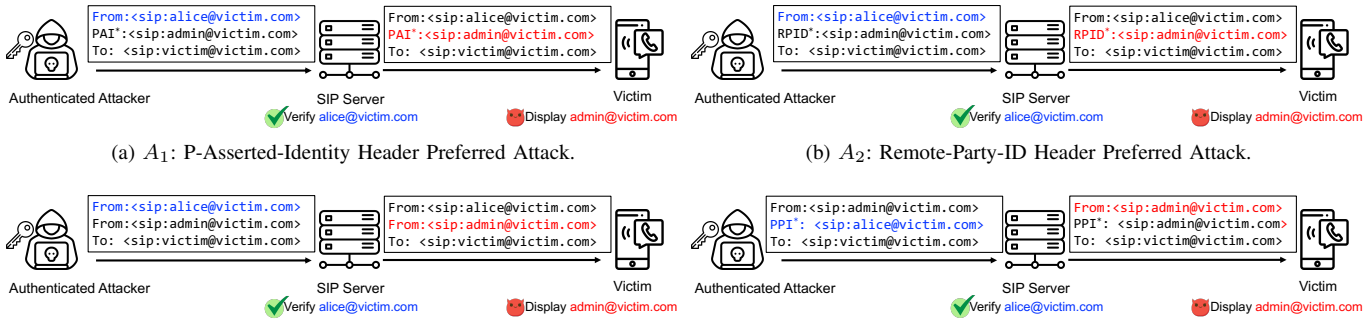
Our evaluation reveals that all tested SIP servers implement basic identity alignment to verify whether the identity specified in the identity headers matches the authenticated identity in the authentication headers — with the exception of message sending in FreeSWITCH, which allows the authenticated user to modify its From header freely. Notably, half of the tested SIP servers forward calls or messages without completing the authentication challenge when the sender's identity does

TABLE I: Tested Open-source SIP Proxy Servers.

#	Name	Version	Github Stars (2025.3)
1	ejabberd[24]	24.12	6.2k
2	FreeSwitch[25]	1.10.12	3.9k
3	Asterisk[26]	22.1.1	2.4k
4	Kamailio[27]	6.0.1	2.4k
5	Opensips[28]	3.4.1	1.3k
6	Flexisip[29]	2.4.0	160

TABLE II: Tested SIP User Agents

#	Name	Version
1	Zoiper[30]	5.6.6
2	Linphone[31]	5.2.6
3	MircoSIP[32]	3.21.5
4	MizuPhone[33]	4.0.24061.97
5	Jitsi[34]	2.10.5550
6	Jami[35]	latest stable (20250304)
7	Empathy[36]	3.25.90
8	Twinkle[37]	1.10.2
9	Blink[38]	5.9.1



(a) A_1: P-Asserted-Identity Header Preferred Attack.

(b) A_2: Remote-Party-ID Header Preferred Attack.

(c) A_3: First From Header Preferred Attack.

(d) A_4: P-Preferred-Identity Header Preferred Attack.

Fig. 8: Different cases of Identity Headers Mismatch Attacks.
*Abbreviations to indicate corresponding headers.

not belong to the hosted domain. This behavior enables unauthenticated attackers to bypass authentication and send spoofed requests. A summary of the identified vulnerabilities is presented in Table III.

Our analysis shows that the REGISTER and PUBLISH methods do not alter the identity of subsequent messages or calls, consistent with their defined roles: REGISTER handles user registration, and PUBLISH shares presence information. These methods are not designed to modify identity during sessions. Likewise, the UPDATE and REFER methods do not affect the identity of ongoing calls in the SIP servers and user agents we tested. UPDATE adjusts session parameters, and REFER redirects the recipient to a third party—neither changes the caller's identity in our evaluation. We observed that the From header is most commonly used by user agents to display caller identity; other headers, except for P-Asserted-Identity and Remote-Party-ID, were not applied by any user agents during our evaluation. Among proxy servers, only Flexisip was found to prefer the P-Preferred-Identity header for identity alignment.

B. Authenticated Attacks

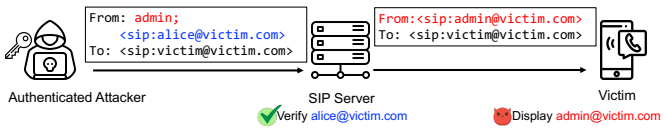
As stated in Section III-A, an authenticated attacker with a valid SIP account can pass the *Authentication Challenge* and be authorized within the target SIP domain. Although authenticated users are normally limited to their own identity, we discovered that, in all tested servers, an attacker can craft SIP requests that pass server validation and impersonate another user. By exploiting ambiguities in how SIP servers parse and prioritize identity headers, attackers can impersonate other users and bypass identity alignment mechanisms.

1) Identity Headers Mismatch: Due to the existence of multiple identification headers in the SIP protocol standard, such as P-Asserted-Identity, Remote-Party-ID, and From, etc., inconsistencies in how these headers are supported and prioritized across different SIP servers and user agents can introduce ambiguities in identity handling. These ambiguities may be exploited by attackers who manipulate under-prioritized or unsupported headers to spoof the identities of legitimate users. Through experiments, we found the following scenarios.

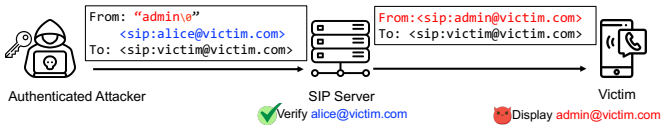
P-Asserted-Identity header preferred attack (A_1): In the example of Fig. 8a, an authenticated attacker sends a SIP request containing two conflicting identity fields: a legitimate From header and a forged P-Asserted-Identity header claiming to be `admin@victim.com`. Though P-Asserted-Identity was intended to be inserted only by trusted intermediaries after successful authentication, we discovered that half of the tested SIP servers, including Kamailio, OpenSIPS, and ejabberd, fail to enforce strict access control over the P-Asserted-Identity header, allowing end users to forge it freely. As a result, user agents such as Zoiper display the asserted identity, thus attackers can impersonate high-privilege users without breaking authentication.

Remote-Party-ID header preferred attack (A_2): Similarly, Fig. 8b demonstrates a spoofing attack that abuses the deprecated Remote-Party-ID (RPID) header to mislead the user agent about the caller's identity. In this attack, a legitimately authenticated user, Alice, includes a forged Remote-Party-ID header in the SIP request. The SIP server verifies the request based on the From header and correctly associates the message with `alice@victim.com`. However, the callee's client prioritizes the RPID field when rendering the caller's identity, thereby displaying the spoofed address `admin@victim.com`. As a mechanism for representing authenticated user identities in SIP, Remote-Party-ID was never formalized by the IETF. Nevertheless, we discovered that the RPID header remains supported in several SIP implementations and continues to influence display behavior on some user agents such as Zoiper.

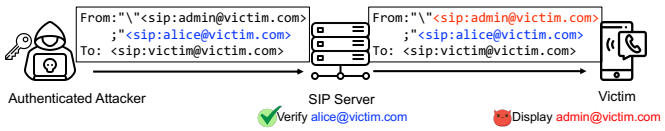
First From header preferred attack (A_3): Although RFC 3261 defines the syntax and semantics of the From header and recommends using multiple headers only if the header uses comma-separated values, it does not explicitly prohibit the presence of multiple From headers in a SIP message. The lack of a strict constraint allows different SIP implementations to adopt diverging parsing and display strategies. As shown in Fig. 8c, an authenticated attacker sends a SIP request containing two From headers where the first From is valid and used by the SIP server for authentication, and the second



(a) B_1: Semicolon Truncation Attack.



(b) B_2: NULL byte Truncation Attack.



(c) B_3: Quoted display-name Mismatch Attack.

Fig. 9: Different cases of Malformed Identity Headers Attacks.

is appended to mislead the recipient. We found that while most of the SIP servers will reject requests that have multiple From headers, ejabbered and OpenSIPS will forward any number of From headers and only verify the first From header. And some clients, including Jami, MicroSIP, Blink, and Empathy, will display the last occurrence.

P-Preferred-Identity preferred attack (A_4): As defined in RFC 3325, the P-Preferred-Identity header allows a user agent to suggest a preferred identity to the network, typically for generating a corresponding P-Asserted-Identity header. It is not intended to be shown to end users, but rather to inform identity assertion for SIP intermediate servers within trusted domains. As illustrated in Fig. 8d, the authenticated attacker sends a SIP request with a spoofed From and a valid P-Preferred-Identity header. We discovered that the Flexisip server preferred the P-Preferred-Identity header for identification and authentication alignment, while forwarding the attacker-controlled From header without normalizing it. As a result, all the user agents we tested display the unverified From identity, falsely attributing the call and message to a spoofed user.

2) Malformed Identity Headers: As it defined in RFC 3261, the From header field consists of two components: `name-addr` and `addr-params`. Fig. 11 illustrates the detail of the From header structure. The `name-addr` component identifies the initiator of the SIP request and comprises two parts: an optional `display-name`, which is used to present the caller's name on the callee's interface, and an `addr-spec`, which specifies the SIP address of the sender. These two parts are separated by a space. The `addr-params`, separated from `name-addr` by a semicolon character, serve to further distinguish SIP messages that share the same Call-ID header. With the help of the mutation and fuzzing process, we effectively explore the structure of these identity headers. After evaluation, we identified three cases of identity spoofing scenarios caused by inconsistent parsing of delimiters and escape characters, as illustrated in Fig. 9.

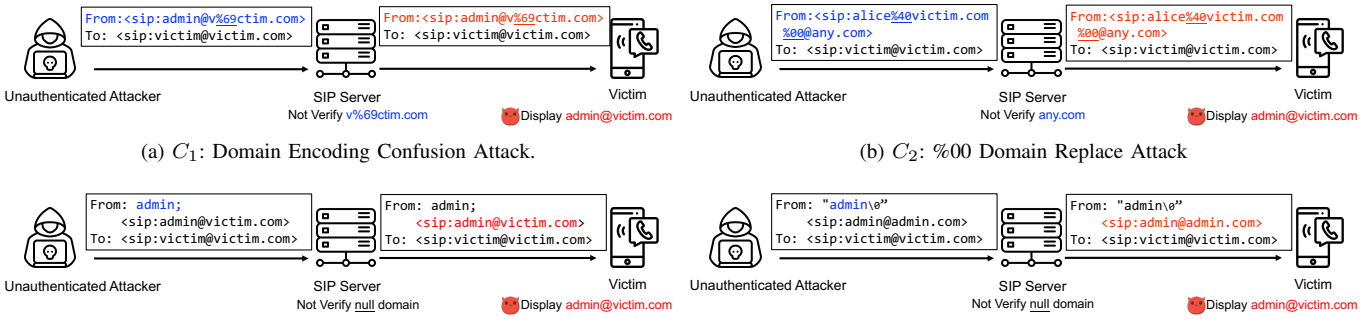
Semicolon truncation attack (B_1): This case involves an incorrect handling of the semicolon character. As Fig. 9a, we found that during the *Authentication Challenge* phase, Asterisk treats the semicolon character as part of `display-name`, but not the separator between `name-addr` and addr-params. However, when Asterisk delivers and forwards the message, the pjsip module will consider the semicolon character as the separator and remove the "addr-params". The message will further be normalized to the malformed "identity" (e.g. `admin` into `admin@victim.com`), thus resulting in the spoofing.

NULL byte truncation attack (B_2): This attack exploits ambiguity in interpreting the NULL byte (`0x00`). Fig. 9b shows that during authentication, the Asterisk server also allows the NULL byte as part of the `display-name`. However, when normalizing the message, the pjsip module interprets the NULL byte as marking the end of the From header, thereby considering the string `"admin"` before the NULL byte character as the identity of the sender. The server then regularizes the From header and forwards the message to the SIP client, leading to identity spoofing. The B_1, B_2 attacks highlight how internal inconsistencies in SIP servers can enable ambiguity-based identity spoofing.

Quoted display-name mismatch attack (B_3): As shown in Fig. 9c, this attack exploits inconsistent interpretations of the SIP From header between SIP proxies and user agents (UAs), leveraging a syntactic ambiguity defined in RFC 3261. The attacker constructs a From header in which the display-name is a quoted string that syntactically resembles a full SIP URI ending with a semicolon, and is followed by a legitimate address. When properly parsed, the quoted `display-name` should be treated as a nickname, and the subsequent `addr-spec` should be used for authentication. However, this input is interpreted inconsistently between the SIP Server and the user agent. The SIP server properly parses the quoted string and extracts the real addr-spec for authentication in the *Authentication Challenge* phase. Meanwhile, the user agent fails to interpret the outer quotes correctly, instead treating the inner "address" as the actual `addr-spec`. Additionally, the semicolon after the spoofed address leads the user agent to prematurely terminate parsing before reaching the real `addr-spec`. Consequently, the attacker can impersonate another user without violating authentication checks enforced by the SIP server.

C. Unauthenticated Attacks

In this study, we also consider unauthenticated attackers, those who do not possess valid credentials within the target SIP domain. In general, such attackers are expected to be blocked by SIP servers during the `Authentication Challenge` phase. However, our evaluation reveals a common behavior among SIP servers, including Kamailio, OpenSIPS, and ejabberd: When handling requests from a non-local domain, they simply forward the request without enforcing authentication. Through fuzzing-based exploration, we discovered that attackers can exploit this behavior by crafting ambiguously



(a) C_1: Domain Encoding Confusion Attack.

(b) C_2: %00 Domain Replace Attack

(c) C_3: Semicolon Truncated Confusion Attack.

(d) C_4: NULL byte Truncated Confusion Attack.

Fig. 10: Different cases of Domain Confusion Attacks.

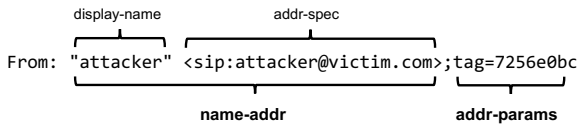


Fig. 11: From Header Structure.

structured SIP messages that appear to originate from a foreign domain to bypass the authentication of the SIP server, but are interpreted by the receiving user agent as belonging to the local domain. These inconsistencies between different SIP servers and user agents in how to parse and trust identity fields enable the attacker to spoof identities without any valid credentials.

1) Domain Confusion Attacks: One way to leverage this flaw is to craft malformed From headers that ambiguously encode the domain part of the SIP URI, so that attackers can deceive UAs into rendering a trusted identity within the local domain, while bypassing verification at the server side. These attacks introduce a semantic gap between server-side processing and client-side display, enabling unauthenticated identity spoofing. As illustrated in Fig. 10, our analysis reveals multiple such vectors, including escaped characters, null bytes, and delimiter misuse.

Domain Encoding Confusion Attack (C_1): As specified in RFC 3261 [10], the SIP protocol adopts the `% HEX HEX` mechanism for character escaping, following the rules defined in RFC 2396. For instance, a username like `"j@s0n"` must be encoded as `"j%40s0n"` to escape the `'@'` character. However, we discovered that several SIP user agents, including Blink, Empathy, and MizuPhone, fail to enforce the RFC's constraints on escaping, particularly about the host component of the URI. As illustrated in Fig. 10a, an attacker can encode the character `i` in the domain name `"victim.com"` as `'%69'`. While this is a non-equivalent, non-standard domain, the SIP server fails to reject the malformed request and forwards it to the recipient. Meanwhile, the user agent of the victim incorrectly decodes the escaped domain for display, rendering the sender identity as `admin@victim.com`. This misleads the user into believing the message originated from a trusted internal administrator.

%00 Domain Replace Attack (C_2): In another case, the attacker manipulates the username part without escaping the domain. RFC 3261 specifies that SIP URIs must conform to strict syntactic and escaping rules, and that each component, especially the domain, must be fully validated and treated as opaque by SIP elements that are not authoritative for them. However, as illustrated in Fig. 10b, we uncover a critical flaw in several user agents and SIP server implementations when handling null-byte encodings within the domain portion of a SIP URI. In this attack, an unauthenticated attacker crafts a From header such as `sip:admin%40victim.com%00@any.com`, where `'%40'` encodes `'@'` and `'%00'` represents a null byte character. We found that half of the tested SIP servers interpret the URI as syntactically acceptable and forward it to the target recipient. On the client side, Jami and Twinkle mishandle the null byte during display rendering, treating it as a string terminator, which causes the identity to be presented as `admin@victim.com`. As a result, the attacker successfully forged the identity of the admin account, misleading the recipient into trusting a spoofed origin.

Semicolon truncated confusion attack (C_3): As illustrated in Fig. 10c, an attacker crafts a malformed From header. SIP servers such as OpenSIPS incorrectly interpret the semicolon as the end of the URI and omit victim.com from domain validation. Meanwhile, many SIP clients, including Jitsi, MicroSIP, and Blink, process this header as a valid display-name + URI structure, and render `admin@victim.com` as the caller identity. This inconsistency enables attackers to spoof trusted identities by injecting a syntactically ambiguous display name, exploiting a mismatch between server-side URI handling and client-side rendering.

NULL byte truncated confusion attack (C_4): While RFC 3261 permits quoted strings as display names in SIP headers, control characters, including the null character, are explicitly excluded and must not appear unescaped. As shown in Fig. 10d, an unauthenticated attacker crafts a From header with a display name that contains a null character. The SIP server, OpenSIPS, misinterprets the display name portion, failing to extract and validate the correct domain, and forwards

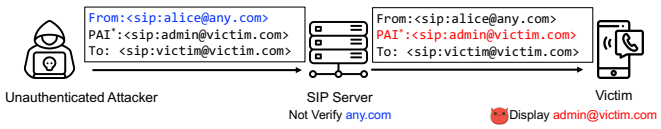
the message unchanged. However, we found that many UAs will tolerate the null bytes and display the SIP URI in the UI, resulting in the forged identity appearing to originate from a trusted administrator.

2) Header Smuggling: Another way to bypass the authentication challenge and spoof identity as a trusted identity within the local domain is by smuggling an extra local domain identity header. Although both *Identity Headers Mismatch* attack and *Header Smuggling* attack utilize multiple identity heads, the subtle difference between them is that *Identity Headers Mismatch* attack utilizes multiple identities within the local domain, whereas *Header Smuggling* attack is smuggling in identity through unauthenticated requests. After the evaluation, we discovered three scenarios as depicted in Fig. 12.

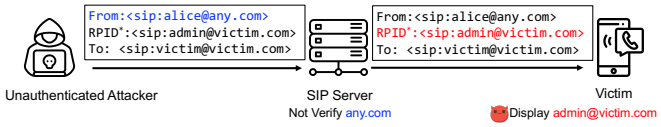
P-Asserted-Identity header smuggling attack (D_1): As defined in RFC 3325, the P-Asserted-Identity (PAI) header is intended for use within trusted administrative domains to assert a verified identity. However, as illustrated in Fig. 12a, we find that all the tested SIP servers fail to strip or validate PAI headers from unauthenticated, cross-domain requests. In this attack, an unauthenticated attacker sends a SIP request with a cross-domain From header and a forged P-Asserted-Identity header within the domain. The server directly forwards the request containing the attacker-controlled PAI header without validating any.com as a trusted peer. As a result, some of the user agent, such as Zoiper, displays the forged asserted identity (`admin@victim.com`), allowing the attacker to impersonate a trusted user despite originating from an unrelated domain.

Remote-Party-ID header smuggling attack (D_2): Similar to the P-Asserted-Identity header, the deprecated Remote-Party-ID (RPID) header remains supported by various SIP user agents for backward compatibility. As shown in Fig. 12b, the attacker includes an RPID header with a spoofed identity in an unauthenticated request. SIP servers that do not validate or sanitize the presence of the RPID header simply forward the header to the client. Because many UAs prioritize the RPID header over the standard From field for rendering caller identity, the spoofed address is displayed despite the request originating from a non-trusted domain. This attack underscores the lingering risks of legacy header support and the importance of default header sanitization.

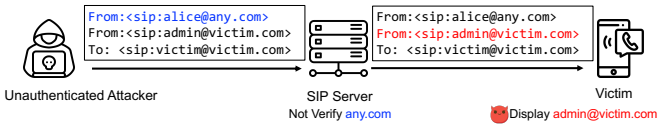
Multiple From header smuggling attack (D_3): Similarly to the A_3 attack, some SIP servers and user agents do not strictly enforce the constraint that there should be only one From header. As shown in Fig. 12c, the attacker sends a message containing two From headers: one with a benign identity and another with a forged one. Although the SIP parser on the server may only consider the first instance for routing or validation, some user agents will render the last occurrence for display. This discrepancy allows the attacker to smuggle a spoofed identity past the proxy and manipulate how it is presented to the recipient. This variant demonstrates the danger of non-compliant SIP implementations that tolerate duplicated headers and inconsistently resolve conflicts.



(a) D_1: P-Asserted-Identity Header Smuggling Attack.



(b) D_2: Remote-Party-ID Header Smuggling Attack.



(c) D_3: Multiple From header Smuggling Attack.

Fig. 12: Different cases of Header Smuggling Attacks
*Abbreviations to indicate corresponding headers.

VI. Real World Impact

While protocol-level vulnerabilities and semantic discrepancies can be systematically demonstrated in testbed scenarios, it is essential to assess whether these issues also manifest in operational services that are actively used in production. To evaluate the practical implications of our findings, we extend our analysis beyond virtual environments and conduct experiments against real-world SIP devices and deployments. In this study, we focus our analysis on two categories of SIP-based services that are representative of widespread usage in both enterprise and consumer contexts: public SIP services and Rich Communication Services (RCS) platforms.

A. Commercial VoIP Hardware Devices

Real-world VoIP networks typically rely on dedicated hardware devices rather than software-only solutions. And devices like Session Border Controllers (SBCs) are implemented to secure and manage SIP-based VoIP calls in enterprise-grade deployments. To evaluate whether vulnerabilities exist in commercial VoIP devices, we deployed VoIP systems in our lab.

1) Setup: In this evaluation, we utilized ejabberd, the most popular SIP server on GitHub, as the proxy server and examined two scenarios: one without an SBC and the other with an SBC. As depicted in Fig. 13, in the latter scenario, both attacker and victim devices are linked to the proxy server via the SBC. We chose representative VoIP devices from top market vendors[39], [40], including phones like the Yealink T30, Motorola IP9910, and Polycom VVX1500D, as well as SBCs such as the Mediant 4000b and ProSBC.

2) Results: We summarized the results in Table IV. We found that all the VoIP devices are vulnerable to ambiguity-based spoofing attacks to some extent. The results demonstrate that VoIP hardware is also vulnerable to the ambiguity-based spoofing vulnerabilities we discovered. We found that some of our test cases will be normalized after being forwarded by the

TABLE III: Spoofing Vulnerabilities between open-source SIP servers and SIP user agents.

User Agents	Method	Asterisk	FreeSWITCH	Kamailio	OpenSIPS	ejabberd	Flexisip
Linphone	INVITE						A_4
	MESSAGE	$B_{1,2}$	Not Aligned				A_4
Zoiper	INVITE			$A_{1,2}, D_{1,2}$	$A_{1,2}, C_1, D_{1,2}$	$A_{1,2}, C_1, D_{1,2}$	$A_{1,2,4}$
	MESSAGE	$B_{1,2}$	Not Aligned		$C_{1,3,4}$	C_1	A_4
Jami	INVITE				A_3, C_2, D_3	A_3, C_2, D_3	A_4
	MESSAGE	$B_{1,2}$	Not Aligned	C_2	$A_3, C_{2,3,4}, D_3$	A_3, C_2, D_3	A_4
Jitsi	INVITE						A_4
	MESSAGE	$B_{1,2}$	Not Aligned		C_3		A_4
MicroSIP	INVITE	B_3			A_3, D_3	A_3, D_3	A_4
	MESSAGE	$B_{1,2}$	Not Aligned	B_3	A_3, B_3, C_3, D_3	$A_3, B3, D_3$	A_4
Blink	INVITE				A_3, D_3	A_3, D_3	A_4
	MESSAGE	$B_{1,2}$	Not Aligned		C_4		A_4
MizuPhone	INVITE	B_3		B_3	B_3, C_1	B_3, C_1	A_4
	MESSAGE	$B_{1,2}$	Not Aligned	B_3	$B_3, C_{1,3,4}$	B_3, C_1	A_4
Empathy	INVITE				C_1	C_1	A_4
	MESSAGE	$B_{1,2}$	Not Aligned		C_1	C_1	A_4
Twinkle	INVITE				A_3, D_3	A_3, D_3	A_4
	MESSAGE	$B_{1,2}$	Not Aligned	C_2	C_2	C_2	A_4

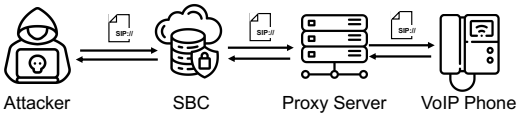


Fig. 13: Evaluation setup for actual VoIP devices with SBC.

TABLE IV: Spoofing Vulnerabilities in Commercial VoIP Hardware Devices.

Telephone	No SBC	ProSBC	Mediant 4000b
Yealink T30	C_1	✓	✓
Motorola IP9910	$A_{1,2}, C_1, D_{1,2}$	$A_{1,2}$	A_1
Polycom VVX1500D	$A_{1,2}, D_{1,2}$	$A_{1,2}$	A_1

SBC. For example, in Case C_1, the URL-encoded header will be normalized to the original header. However, the attacker can still smuggle some identity headers and make the spoofing attack successful.

B. Public SIP Services

Public SIP services offer free or open-access VoIP infrastructure for users to register accounts and perform SIP-based communication without operating their own servers. We searched and selected seven popular public SIP services available on the network that allow anyone to register. The detailed information is presented in Table V.

TABLE V: Tested Public SIP services. *N/A* means the information is not available.

#	Name	Domain	Country	Users
1	Linphone[41]	sip.linphone.org	France	731,278
2	Linhome[42]	sip.linhome.org	France	19,765
3	IPTEL[43]	iptel.org	German	N/A
4	SIP2SIP[44]	sip2sip.info	Netherlands	N/A
5	Antisip[45]	sip.antisip.com	France	456,713
6	Sylk[46]	sylk.link	Netherlands	N/A
7	OnSIP[47]	onsip.com	United States	over 100,000

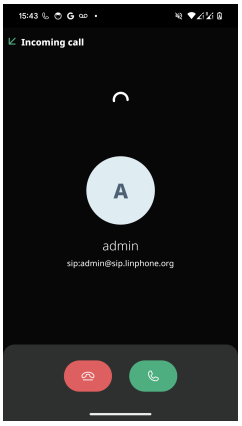


Fig. 14: An example of Caller ID spoofing that impersonates the admin account in the official Linphone public SIP service.

1) Setup: Our testing methodology was carefully designed to ensure minimal ethical risk and zero disruption to external users. We registered two accounts on the target SIP service, one acting as the attacker and the other as the victim. All messages were sent from the controlled "attacker" account and delivered exclusively to the "victim" account under our control. No unsolicited traffic was sent to third-party users, and no service functionality outside of normal SIP communication was exercised. All cases we tested are mentioned in section V and did not trigger crashes, DoS conditions, or resource exhaustion on any of the tested servers. Therefore, we are confident that the inputs used in this phase posed no operational risk to the public SIP infrastructure under test.

2) Results: Our evaluation uncovered a widespread presence of identity spoofing vulnerabilities caused by protocol semantic ambiguity across public SIP services and clients. Notably, although all the public SIP services we tested had the basic identity alignment mechanism that rejects requests with different usernames in the From header and Authentication header, attackers can leverage the semantic ambiguity

to launch spoofing attacks. As summarized in Table VI, we confirmed that the semantic inconsistencies identified in our controlled experiments are also prevalent in real-world deployments. We discovered that real-world deployed services such as Linphone and Linhome are affected by the *P-Preferred-Identity preferred attack (A_4)*, which could put almost all users at risk of identity spoofing. We also discovered that over half of our testing public services will directly forward cross-domain SIP requests without authentication, which could be abused by an attacker to launch unauthenticated attacks, making it harder to trace the attacker.

C. Rich Communication Services

Rich Communication Services (RCS) was developed to upgrade SMS, offering enhanced capabilities for messaging, voice, video, and file transfer to meet the demands of modern systems. According to GSMA's report [48], as of 2020, RCS had been rolled out by 90 cell operators across 60 countries worldwide. Moreover, over 1 billion users globally utilize RCS for their daily conversations [49]. Recognizing the potential for enhanced subscriber experiences, operators turn to RCS to offer more advanced and engaging communication services. The SIP protocol functions as the underlying signaling protocol for RCS services and is crucial for establishing and managing communication sessions.

1) Setup: To access commercial RCS proxies, we first set up a custom Man-in-the-Middle (MITM) proxy to intercept and manipulate SIP traffic between the attacker's phone and the RCS servers. We performed an MITM attack on the attacker's phone by installing certificates and rerouting the RCS traffic. This lets us obtain a legitimate RCS account with credentials and caller ID formats (e.g., sip:phone-number@rcs-domain), thus gaining the ability to test SIP messages. Next, we can construct the victim's caller ID with the caller ID formats, only knowing their phone number and operator, and send spoofed RCS messages to the RCS server with the recipient set to the victim. This requires only the victim's phone number—no access to or modification of the victim's device or traffic.

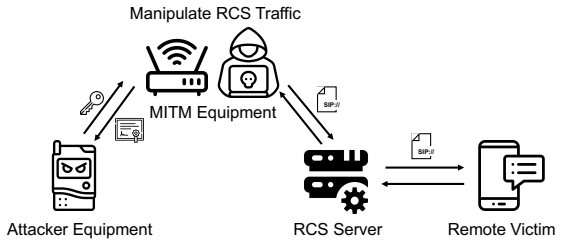


Fig. 15: Evaluation setup for RCS attack.

2) Results: For ethical considerations, we did not run the full fuzzer on commercial operators, but used the PoCs we found in the previous section for limited tests. Our tests show that all three operators will reject the spoofed messages that simply modify the identity headers. However, we discovered that all of them are vulnerable to A_4 attack, and one of them is vulnerable to B_3 attack. Furthermore, our observations indicate that when a victim's device lacks RCS support or is not configured to use RCS, the spoofed message is still delivered via traditional SMS, thereby impacting users across 2G, 3G, 4G, and 5G networks. As illustrated in Fig. 1, these vulnerabilities allow an attacker, possessing only a legitimate RCS account, to remotely dispatch a spoofed message that appears to originate from any chosen address.

VII. Discussion

A. Root Cause Analysis

The attacks we uncovered stem from a broader thematic pattern: semantic ambiguities that manifest as inconsistencies across different SIP implementations. We highlight two primary sources of such inconsistencies: inconsistencies in what identity should be trusted and inconsistencies in how the identity headers should be parsed.

Trust Inconsistency: RFC 3261 defines a trust model where user agents inherently trust proxy servers to modify message features necessary for SIP operations, such as adding Via headers. This trust is based on the assumption that proxy servers are honest intermediaries, but it does not prevent ambiguity-based spoofing attacks, which can occur even with recommended security measures like TLS in place. The A and D series attack revealed that the overlapping and redundant identity headers standards lead to different trust decisions among different implementations. For example, in the A_2 and D_2 attacks, the Remote-Party-ID header is displayed by the user agent based on the assumption that it has been verified by the proxy server. However, the proxy server, adhering to current standards, does not verify or alter this deprecated header, leading to spoofing despite the proxy not being at fault.

Parsing Inconsistency: Another inconsistency lies in the inherent complexity and flexibility of text-based protocol, which introduces further variability. The From header, for instance, supports a rich feature set, yet implementations often support only subsets, and do so differently. Moreover, the permissiveness of the syntax amplifies the likelihood of divergent parsing behavior, particularly in the face of non-conformant inputs. The B and C series attacks uncovered specific parsing inconsistencies that enable spoofing attacks, including the semicolon truncation attack (B_1) and %00 Domain Replace Attack (C_2). Such inconsistencies cause the user agents to display wrong identities unintentionally, even if the proxy server and user agents have the same trust strategy.

B. Mitigation

We propose several mitigations to fix these issues and help bridge the semantic gap across different SIP network entities.

Align authentication and identification headers. Specifically, SIP servers should ensure that the identity indicated in all the identity headers, including the deprecated headers consistent with the authenticated identity, e.g., via Authorization headers. Requests should be rejected if these identities do not align, except in cases where the server maintains an explicit allowlist mapping authenticated identities to permissible display

TABLE VI: Spoofing Vulnerabilities in public SIP service.
N/A: Method not supported.

User Agents	Method	sip.linphone.org	sip.linhome.org	iptel.org	sip2sip.info	sip.antisip.com	sylk.link	sip.onsip.com
Linphone	INVITE	A_4	A_4					
	MESSAGE	A_4	A_4	N/A				
Zoiper	INVITE	A_4	A_4	$A_{1,2}, D_{1,2}$	C_1	$A_{1,2}, D_{1,2}$	C_1	$A_{1,2}, C_1, D_{1,2}$
	MESSAGE	A_4	A_4	N/A	C_1		C_1	C_1
Jami	INVITE	A_4	A_4	C_2	C_2		C_2	A_3, C_2, D_3
	MESSAGE	A_4	A_4	N/A	C_2, D_3	C_2	C_2, D_3	A_3, C_2, D_3
Jitsi	INVITE	A_4	A_4					
	MESSAGE	A_4	A_4	N/A				
MicroSIP	INVITE	A_4	A_4		D_3		D_3	A_3, D_3
	MESSAGE	A_4	A_4	N/A	D_3	B_3	D_3	A_3, D_3
Blink	INVITE	A_4	A_4		D_3		D_3	A_3, D_3
	MESSAGE	A_4	A_4	N/A				
MizuPhone	INVITE	A_4	A_4		C_1	B_3	C_1	C_1
	MESSAGE	A_4	A_4	N/A	C_1	B_3	C_1	C_1
Empathy	INVITE	A_4	A_4		C_1		C_1	C_1
	MESSAGE	A_4	A_4	N/A	C_1		C_1	C_1
Twinkle	INVITE	A_4	A_4		D_3		D_3	A_3, D_3
	MESSAGE	A_4	A_4	N/A	C_2	C_2	C_2	C_2

identities. This prevents attackers from asserting misleading identities and victims from trusting wrong identities.

Normalize the forwarded message. According to RFC 3325, only trusted SIP entities can insert or modify headers such as P-Asserted-Identity and Remote-Party-ID. Consequently, SIP servers should strip or overwrite these headers if received from untrusted sources or endpoints. Additionally, the SIP server should normalize and rewrite From and P-Asserted-Identity headers to reflect a canonical, server-validated identity with valid identities from P-Preferred-Identity or Authentication headers. This step helps eliminate discrepancies that attackers could use to obfuscate their real origin.

Strict cross-domain handling. Additionally, we identify a broader protocol design issue: the lack of standardized mechanisms for inter-domain authentication. SIP's trust model often implicitly assumes intra-domain trust, creating opportunities for cross-domain header smuggling attacks. To mitigate this, servers that do not support inter-domain authentication schemes, e.g., STIR/SHAKEN, should reject requests with identities not from their administrative domain by default. Alternatively, STIR/SHAKEN or similar cryptographic schemes should be employed to verify identity assertions, providing end-to-end assurance across domain boundaries.

VIII. Related Work

A. Caller ID Spoofing & SMS Spoofing

Caller ID and SMS spoofing are longstanding and impactful attack techniques in which attackers falsify the origin of text or calls for phishing attacks and telecom fraud. According to Tu et al. [2] and Şahin et al. [50], telephone spam costs United States consumers $8.6 billion annually, and global telecom fraud caused revenue losses worth $38 billion in 2015. Kim et al. [4] showed that attackers could forge the caller ID by altering the From header in SIP INVITE messages. In a similar vein, Tu et al. [51] demonstrated that the From and P-Preferred-Identity headers in SIP MESSAGE requests can be replaced to send spoofing text.

In response to these threats, researchers have proposed various detection and prevention strategies. Mustafa et al. [5] developed a challenge-response protocol utilizing a covert channel accessible only to legitimate call participants. Deng et al. [52] introduced a callback-based verification method that checks the consistency of incoming and outgoing call states. Sheoran et al. [53] detected spoofing by inspecting the data exchanged between data-plane gateways and call control session functions. There are also some research [54], [55], [56], [57] that focus on cryptographic authentication mechanisms, introducing public key infrastructure (PKI) or cryptographic protocols to authenticate both parties.

Despite the technical sophistication of these approaches, their adoption in practice remains limited due to implementation complexity and concerns regarding user experience. Moreover, commercial services [58], [59] continue to offer Caller ID and SMS spoofing functionalities in real-world settings. Distinct from prior work, our research systematically investigates multi-party semantic ambiguities within SIP networks. We demonstrate that, even in the presence of cryptographic authentication mechanisms, inconsistencies or misinterpretations in the processing of authenticated fields can result in authentication bypasses and spoofing vulnerabilities. Our methodology is broadly applicable across all SIP-based environments, including carrier-grade and private telephony systems, without controlling other entities in the network.

B. Other SIP Attack

As the cornerstone of Internet telephony and multimedia communications, the Session Initiation Protocol (SIP) has been widely deployed in real-world networks in the past two decades. Extensive research has explored various security aspects of SIP implementations and deployments. Keromytis [60] analyzed 245 VoIP security publications, identifying denial of service and service abuse as requiring significant attention from the research community. Peng et al. [61], [62] identified authentication bypass, authorization fraud, and

accounting volume inaccuracy vulnerabilities in mobile data charging systems. Kim et al. [63] further revealed that cellular data charging mechanisms can be bypassed through protocol manipulation and enable free-riding attacks in operational networks.

However, these works rely on manual testing and suffer from limited scalability and systematic exploration. There are some works et al. [21], [64], [65] applying fuzzing approaches to SIP implementations and cellular networks, but they have primarily targeted memory-related vulnerabilities.

In general, our identified spoofing attacks fall into the broader category of semantic gap attacks, where discrepancies arise between the protocol specification and its diverse real-world implementations. Similar inconsistency problems also exist in other scenarios, such as email protocols [66], [67], [8], HTTP protocols [68], [69], [70], [71], CDN systems [72], [73], [74], [75], web applications [76] and zip files [77], etc. Through differential testing across multiple real-world SIP implementations with our fuzzing framework, we automatically and systematically identify inconsistencies that can be exploited for spoofing attacks rather than memory vulnerabilities. This class of attacks harder to detect through conventional fuzzing or vulnerability scanning, and tends to persist across multiple vendor implementations, contributing new attack vectors and insights to the SIP security landscape.

IX. Conclusion

In this study, we systematically investigated identity spoofing vulnerabilities in the Session Initiation Protocol (SIP) and conducted comprehensive testing. With SipChimera, we discovered that ambiguity-based spoofing vulnerabilities exist widely in open-source SIP servers and popular SIP user agents. Through real-world case studies on spoofing attacks, we found that a significant number of SIP-based services can be exploited by attackers to forge identities, enabling them to conduct social engineering attacks through the victim's trust in the fake identity. This highlights the severity and prevalence of identity spoofing attacks in real-world scenarios.

We responsibly disclosed the vulnerabilities to relevant parties and proposed mitigation strategies. Our goal is to raise awareness among the community and the public about ambiguity-based identity spoofing vulnerabilities in the SIP ecosystem, promote standardized protocols, and reduce the potential for semantic ambiguities that can lead to telecom fraud attacks. By shedding light on these critical issues, we hope to inspire collective efforts to enhance the security and integrity of SIP-based communication systems.

X. Ethics Considerations

Throughout the entirety of our research, we adhered strictly to relevant ethical guidelines and best practices. During the fuzzing phase, all experiments were conducted exclusively on self-deployed SIP servers and SIP User Agents within a fully controlled and isolated network environment. This approach ensured that no unintended traffic or disruptive effects were introduced into external systems.

For real-world testing, we selected only those attack samples that had been verified in our controlled environment to pose no availability risks to the real-world SIP servers. All attack traffic was confined to controlled interactions between simulated attacker and victim accounts, both of which were under our direct management. No traffic was directed toward or had any potential impact on unrelated users or services. Furthermore, to ensure the safety and integrity of external systems, we proactively communicated with relevant operators to conduct any real-world tests. This coordination ensured that our testing activities would not negatively affect their operational networks.

After completing the evaluation, we followed responsible disclosure practices by notifying the affected vendors about the vulnerabilities identified during our study. The vendors' response details are listed below.

Asterisk. We notified them of the vulnerability on March 20. They resolved the issue by March 21 and released the updated version on May 22.

FreeSWITCH. We informed them of the vulnerability on August 2, 2024. They thanked us for the report and indicated they would incorporate it as a security enhancement.

Linphone. We submitted our report on spoofing vulnerabilities in Flexisip and their public SIP service on April 14. They addressed the issue by April 18.

OpenSIPS. We sent our vulnerability report on April 21. They acknowledged the report and are working on a fix.

OP-I & OP-II & OP-III. They appreciated our work and evaluated our report as a critical (highest-level) vulnerability. They formed an expert team and had a in-depth discussion with us about the details. All three operators have fixed the vulnerabilities.

Others. We have contacted other relevant vendors and are looking forward to receiving their feedback.

Acknowledgment

We would like to express our sincere gratitude to the anonymous reviewers for their valuable comments and suggestions that greatly enhanced the quality of this paper. This work was supported by the National Natural Science Foundation of China (grant #62272265).

References

[1] S. Ahari, "New features to celebrate messages' 1 billion rcs users," 2023, last accessed: 2025-07-16. [Online]. Available: https://blog.google/products/android/7-new-messages-features

[2] H. Tu, A. Doupé, Z. Zhao, and G.-J. Ahn, "Sok: Everyone hates robocalls: A survey of techniques against telephone spam," in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 320–338.

[3] V. Buriachok, V. Sokolov, and M. T. Dini, "Research of caller ID spoofing launch, detection, and defense," *CoRR*, vol. abs/2004.00318, 2020. [Online]. Available: https://arxiv.org/abs/2004.00318

[4] H. Kim, D. Kim, M. Kwon, H. Han, Y. Jang, D. Han, T. Kim, and Y. Kim, "Breaking and fixing volte: Exploiting hidden data channels and mis-implementations," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 328–339. [Online]. Available: https://doi.org/10.1145/2810103.2813718

[5] H. Mustafa, W. Xu, A.-R. Sadeghi, and S. Schulz, "End-to-end detection of caller id spoofing attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 423–436, 2018.

[6] F. C. Commission, "Combating spoofed robocalls with caller id authentication," last accessed: 2025-07-18. [Online]. Available: https://www.fcc.gov/call-authentication

[7] PortSwigger, "Http desync attacks: Request smuggling reborn," last accessed: 2025-07-16. [Online]. Available: https://portswigger.net/research/http-desync-attacks-request-smuggling-reborn

[8] C. Wang, C. Wang, S. Yang, S. Liu, J. Chen, H. Duan, and G. Wang, "Email Spoofing with SMTP Smuggling: How the Shared Email Infrastructures Magnify this Vulnerability," in *34th USENIX Conference on Security Symposium*, 2025.

[9] H. Schulzrinne, E. Schooler, J. Rosenberg, and M. J. Handley, "SIP: Session Initiation Protocol," RFC 2543, Mar. 1999. [Online]. Available: https://www.rfc-editor.org/info/rfc2543

[10] E. Schooler, J. Rosenberg, H. Schulzrinne, A. Johnston, G. Camarillo, J. Peterson, R. Sparks, and M. J. Handley, "SIP: Session Initiation Protocol," RFC 3261, Jul. 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3261

[11] H. Schulzrinne, J. Rosenberg, B. Campbell, D. M. Gurle, and C. Huitema, "Session Initiation Protocol (SIP) Extension for Instant Messaging," RFC 3428, Dec. 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3428

[12] B. Marshall, "SIP Extensions for Network-Asserted Caller Identity and Privacy within Trusted Networks," Internet Engineering Task Force, Internet-Draft draft-ietf-sip-privacy-04, Mar. 2002, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-sip-privacy/04/

[13] J. Peterson, C. F. Jennings, and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks," RFC 3325, Dec. 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3325

[14] J. Peterson and C. F. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)," RFC 4474, Aug. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4474

[15] J. Peterson, H. Schulzrinne, and H. Tschofenig, "Secure Telephone Identity Problem Statement and Requirements," RFC 7340, Sep. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7340

[16] J. Peterson, C. F. Jennings, E. Rescorla, and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)," RFC 8224, Feb. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8224

[17] C. Wendt and J. Peterson, "PASSporT: Personal Assertion Token," RFC 8225, Feb. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8225

[18] J. Peterson and S. Turner, "Secure Telephone Identity Credentials: Certificates," RFC 8226, Feb. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8226

[19] F. T. Commission, "Stir/shaken broadly implemented starting today," 2021, last accessed: 2025-07-16. [Online]. Available: https://docs.fcc.gov/public/attachments/DOC-373714A1.pdf

[20] E. Priezkalns, "Global stir/shaken is dead; what comes next?" 2024. [Online]. Available: https://commsrisk.com/global-stir-shaken-is-dead-what-comes-next/

[21] H. J. Abdelnur, R. State, and O. Festor, "Kif: a stateful sip fuzzer," in *Proceedings of the 1st International Conference on Principles, Systems and Applications of IP Telecommunications*, ser. IPTComm '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 47–56. [Online]. Available: https://doi.org/10.1145/1326304.1326313

[22] Github, "Sip topic," last accessed: 2025-07-16. [Online]. Available: https://github.com/topics/sip

[23] VoIP.ms, "Softphones," last accessed: 2025-07-16. [Online]. Available: https://wiki.voip.ms/article/Softphones

[24] "Ejabberd," ProcessOne, last accessed: 2025-07-18. [Online]. Available: https://github.com/processone/ejabberd

[25] "Freeswitch," SignalWire, last accessed: 2025-07-18. [Online]. Available: https://github.com/signalwire/freeswitch

[26] "Asterisk," Asterisk, last accessed: 2025-07-18. [Online]. Available: https://github.com/asterisk/asterisk

[27] "Kamailio," Kamailio, last accessed: 2025-07-18. [Online]. Available: https://github.com/kamailio/kamailio

[28] "Opensips," OpenSIPS, last accessed: 2025-07-18. [Online]. Available: https://github.com/OpenSIPS/opensips

[29] "Flexisip," Belledonne Communications, last accessed: 2025-07-18. [Online]. Available: https://github.com/BelledonneCommunications/flexisip

[30] "Zoiper," Zoiper, last accessed: 2025-07-18. [Online]. Available: https://www.zoiper.com/

[31] "Linphone," Belledonne Communications, last accessed: 2025-07-18. [Online]. Available: https://www.linphone.org/

[32] "Microsip," MicroSIP, last accessed: 2025-07-18. [Online]. Available: https://www.microsip.org/

[33] "Mizuphone," Mizu Tech, last accessed: 2025-07-18. [Online]. Available: https://www.mizu-voip.com/Software/Softphones/Windowssoftphone.aspx

[34] "Jitsi," Jitsi, last accessed: 2025-07-18. [Online]. Available: https://jitsi.org/

[35] "Jami," Jami, last accessed: 2025-07-18. [Online]. Available: https://jami.net/

[36] "Empathy," GNOME, last accessed: 2025-07-18. [Online]. Available: https://gitlab.gnome.org/Archive/empathy

[37] "Twinkle," Twinkle, last accessed: 2025-07-18. [Online]. Available: https://www.twinklephone.com/

[38] "Blink," AG Projects, last accessed: 2025-07-18. [Online]. Available: https://icanblink.com/

[39] "Landline phones market size, share, growth, and industry analysis, by types (cordless telephones, corded telephones) , applications (household use, commercial use) and regional insights and forecast to 2033," Global Growth Insights, 2025, last accessed: 2025-07-18. [Online]. Available: https://www.globalgrowthinsights.com/market-reports/landline-phones-market-117455

[40] R. Carter, "The Top Session Border Controller Vendors in 2025: Securing Your Enterprise Voice," Jul. 2025. [Online]. Available: https://www.uctoday.com/unified-communications/the-top-session-border-controller-vendors-in-2025-securing-your-enterprise-voice/

[41] "Linphone free sip service," Belledonne Communications, last accessed: 2025-07-18. [Online]. Available: https://subscribe.linphone.org

[42] "Linhome free sip service," Belledonne Communications, last accessed: 2025-07-18. [Online]. Available: https://subscribe.linhome.org/

[43] "Iptel free sip service," IPTel, last accessed: 2025-07-18. [Online]. Available: https://www.iptel.org/

[44] "Sip2sip free internet communications," AG Projects, last accessed: 2025-07-18. [Online]. Available: https://sip2sip.info/

[45] "Antisip free sip service," AntiSIP, last accessed: 2025-07-18. [Online]. Available: https://sip.antisip.com/

[46] "Sylk suite," Sylk, last accessed: 2025-07-18. [Online]. Available: https://sylkserver.com/

[47] "Business voip phone system," onsip, last accessed: 2025-07-18. [Online]. Available: https://www.onsip.com/

[48] GSMA, "Global forecast for rcs growth." [Online]. Available: https://www.gsma.com/solutions-and-impact/technologies/networks/rcs/global-launches/

[49] M. Gatford, "Rcs active users to surpass 1 billion for first time in 2024," Juniper Research, Tech. Rep., 2023. [Online]. Available: https://www.juniperresearch.com/press/rcs-active-users-to-surpass-1bn-2024/

[50] M. Şahin, A. Francillon, P. Gupta, and M. Ahamad, "Sok: Fraud in telephony networks," 04 2017, pp. 235–250.

[51] G.-H. Tu, C.-Y. Li, C. Peng, Y. Li, and S. Lu, "New security threats caused by ims-based sms service in 4g lte networks," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1118–1130. [Online]. Available: https://doi.org/10.1145/2976749.2978393

[52] H. Deng and C. Peng, "Combating caller id spoofing on 4g phones via ceive," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 846–848. [Online]. Available: https://doi.org/10.1145/3241539.3267725

[53] A. Sheoran, S. Fahmy, C. Peng, and N. Modi, "Nascent: Tackling caller-id spoofing in 4g networks via efficient network-assisted validation," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 676–684.

[54] H. Tu, A. Doupe, Z. Zhao, and G.-J. Ahn, "Toward standardization of authenticated caller id transmission," *IEEE Communications Standards Magazine*, vol. 1, no. 3, pp. 30–36, 2017.

[55] B. Reaves, L. Blue, H. Abdullah, L. Vargas, P. Traynor, and T. Shrimpton, "AuthentiCall: Efficient identity and content authentication for phone calls," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 575–592.

[Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/reaves

[56] I. Dacosta and P. Traynor, "Proxychain: developing a robust and efficient authentication infrastructure for carrier-scale voip networks," in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIXATC'10. USA: USENIX Association, 2010, p. 10.

[57] B. Reaves, L. Blue, and P. Traynor, "Authloop: practical end-to-end cryptographic authentication for telephony over voice channels," in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. SEC'16. USA: USENIX Association, 2016, p. 963–978.

[58] SpoofTel, last accessed: 2025-07-16. [Online]. Available: https://www.spooftel.com

[59] SpoofCard, last accessed: 2025-07-16. [Online]. Available: https://www.spoofcard.com

[60] A. D. Keromytis, "A comprehensive survey of voice over ip security research," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 514–537, 2012.

[61] C. Peng, C.-y. Li, G.-H. Tu, S. Lu, and L. Zhang, "Mobile data charging: new attacks and countermeasures," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 195–204. [Online]. Available: https://doi.org/10.1145/2382196.2382220

[62] C. Peng, C.-Y. Li, H. Wang, G.-H. Tu, and S. Lu, "Real threats to your data bills: Security loopholes and defenses in mobile data charging," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 727–738. [Online]. Available: https://doi.org/10.1145/2660267.2660346

[63] H. Hong, H. Kim, B. Hong, D. Kim, H. Choi, E. Lee, and Y. Kim, "Pay as you want: Bypassing charging system in operational cellular networks," in *Information Security Applications*, D. Choi and S. Guilley, Eds. Cham: Springer International Publishing, 2017, pp. 148–160.

[64] N. Bennett, W. Zhu, B. Simon, R. Kennedy, W. Enck, P. Traynor, and K. R. B. Butler, "Ransacked: A domain-informed approach for fuzzing lte and 5g ran-core interfaces," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 2027–2041. [Online]. Available: https://doi.org/10.1145/3658644.3670320

[65] T. Yang, S. M. M. Rashid, A. Ranjbar, G. Tan, and S. R. Hussain, "ORANalyst: Systematic testing framework for open RAN implementations," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 1921–1938. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/yang-tianchang

[66] J. Chen, V. Paxson, and J. Jiang, "Composition Kills: A Case Study of Email Sender Authentication," in *29th USENIX Conference on Security Symposium*, 2020.

[67] J. Zhang, J. Chen, Q. Wang, H. Zhang, C. Wang, J. Zhuge, and H. Duan, "Inbox invasion: Exploiting mime ambiguities to evade email attachment detectors," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 467–481. [Online]. Available: https://doi.org/10.1145/3658644.3670386

[68] J. Chen, J. Jiang, H. Duan, N. Weaver, T. Wan, and V. Paxson, "Host of Troubles: Multiple Host Ambiguities in HTTP Implementations," in *23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[69] K. Shen, J. Lu, Y. Yang, J. Chen, M. Zhang, H. Duan, J. Zhang, and X. Zheng, "Hdiff: A semi-automatic framework for discovering semantic gap attack in HTTP implementations," in *52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2022)*. Baltimore, MD, USA: IEEE, 2022, pp. 1–13. [Online]. Available: https://doi.org/10.1109/DSN53405.2022.00014

[70] K. Mu, J. Chen, J. Zhuge, Q. Li, H. Duan, and N. Feamster, "The Silent Danger in HTTP: Identifying HTTP Desync Vulnerabilities with Gray-box Testing," in *34th USENIX Conference on Security Symposium*, 2025.

[71] Y. Liang, J. Chen, R. Guo, K. Shen, H. Jiang, M. Hou, Y. Yu, and H. Duan, "Internet's Invisible Enemy: Detecting and Measuring Web Cache Poisoning in the Wild," in *31th ACM Conference on Computer and Communications Security*, 2024.

[72] J. Chen, J. Jiang, X. Zheng, H. Duan, J. Liang, K. Li, T. Wan, and V. Paxson, "Forwarding Loop Attacks in Content Delivery Networks," in *Proceedings 2016 Network and Distributed System Security Symposium*, 2016.

[73] R. Guo, J. Chen, Y. Wang, K. Mu, B. Liu, X. Li, C. Zhang, H. Duan, and J. Wu, "Temporal CDN-Convex Lens: A CDN-Assisted Practical Pulsing DDoS Attack," in *32th USENIX Conference on Security Symposium*, 2023.

[74] L. Zheng, X. Li, C. Wang, R. Guo, H. Duan, J. Chen, and K. Shen, "ReqsMiner: Automated Discovery of CDN Forwarding Request Inconsistencies with Differential Fuzzing," in *Proceedings 2024 Network and Distributed System Security Symposium*, 2024.

[75] Q. Wang, J. Chen, Z. Jiang, R. Guo, X. Liu, C. Zhang, and H. Duan, "Break the wall from bottom: Automated discovery of protocol-level evasion vulnerabilities in web application firewalls," in *2024 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2024, pp. 128–128. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00129

[76] E. Wang, J. Chen, W. Xie, C. Wang, Y. Gao, Z. Wang, H. Duan, Y. Liu, and B. Wang, "Where URLs Become Weapons: Automated Discovery of SSRF Vulnerabilities in Web Applications," in *2024 IEEE Symposium on Security and Privacy*, 2024.

[77] Y. You, J. Chen, Q. Wang, and H. Duan, "My ZIP isn't your ZIP: Identifying and Exploiting Semantic Gaps Between ZIP Parsers," in *34th USENIX Security Symposium*. Seattle, WA: USENIX Association, Aug. 2025. [Online]. Available: https://www.usenix.org/conference/usenixsecurity25/presentation/you

APPENDIX

A. Methods and Core Headers Selection

Listing 1: Additional rules for generating samples.

```
{
    "<START>": [
        ["<_rep_identity_header_s>"]
    ],
    "<_rep_identity_header_s>":[
        [
            "<identitiy_header>",
            "<CRLF>",
            "<_rep_identity_header_s>"
        ],
        ["<identitiy_header>"]
    ],
    "<identitiy_header>":[
        ["<From>"],
        ["<PPreferredID>"],
        ["<PAssertedID>"],
        ["<Remote-Party-ID>"],
        ["<Identity>"],
        ["<Identity-Info>"],
        ["<Contact>"],
        ["<Call-Info>"],
        ["<Organization>"],
        ["<Refer-To>"],
        ["<Referred-By>"]
    ],
    "<user>": [
        [ "<fake_sender>"],
        [ "<sender>"]
    ]
}
```

TABLE VII: SIP Method Selection Analysis

SIP Method	Applicable	Primary Reason
INVITE	✓	Core to call/session initiation; central to our attack scenarios before victims answer calls.
MESSAGE	✓	Core to messaging; central to our attack scenarios when victims receive messages.
REGISTER	✓	Related to attacker identity binding; may lead to identity spoofing during incoming call/message.
UPDATE	✓	Used to modify session state before establishment, which could be exploited by attackers to manipulate caller display information.
PUBLISH	✓	May update attacker's identity status, potentially leading to identity spoofing before call setup or message delivery.
REFER	✓	Enables call transfers by the caller or callee, presenting opportunities for identity spoofing.
ACK	×	Response to session initialization from receiver. As a passive response message, it cannot be actively initiated for testing attack scenarios.
PRACK	×	Similar to ACK, used with reliable provisional responses. As a passive response message, it cannot be actively initiated for testing attack scenarios.
BYE	×	Used to terminate established sessions. It cannot be used for caller ID spoofing as it does not involve identity presentation.
CANCEL	×	Cancels pending requests after initiation. It cannot be used for caller ID spoofing as it does not involve identity presentation.
OPTIONS	×	Used for querying endpoint capabilities and status, not for session establishment. It cannot be used for caller ID spoofing as it does not involve identity presentation.
INFO	×	Used for transferring additional information (e.g., DTMF tones) only after the receiver answers the call. It cannot be used for caller ID spoofing as it occurs after session establishment.
SUBSCRIBE	×	Used for subscribing to status information, not related to caller identity presentation. Cannot be used for caller ID spoofing attacks.
NOTIFY	×	Paired with SUBSCRIBE for event notifications between UAS and Proxy, not related to caller identity presentation. Cannot be used for caller ID spoofing attacks.

TABLE VIII: SIP Identity-Related Headers Analysis.

Header	Source	Selection Reason
From	RFC 3261	ABNF Definition has the user field, Core identity header
P-Asserted-Identity	RFC 3325	ABNF Definition has the user field, Trusted network identity assertion
P-Preferred-Identity	RFC 3325	ABNF Definition has the user field, User agent preferred identity suggestion
Remote-Party-ID	Draft-ietf-sip-privacy-04	ABNF Definition has the user field, caller identity representation
Contact	RFC 3261	ABNF Definition has the user field, which contains the user identity for direct communication
Identity	RFC 4474 / RFC 8224	Cryptographic identity verification, related to identity mechanism
Identity-Info	RFC 4474	Cryptographic identity verification, related to identity mechanism
Call-Info	RFC 3261	Provides additional information about the caller, including photos and business cards for enhanced caller identification
Organization	RFC 3261	Provides the organization name of the caller
Refer-To	RFC 3515	Used by caller or callee to implement call transfer, which may be used for identity spoofing.
Referred-By	RFC 3892	Used by callee to indicate the caller identity when implementing call transfer.

B. Evaluation Results

TABLE IX: Evaluation Results of 50,000 Samples for Each Method Combination between SIP Proxy Servers and User Agents

User Agents	Method	Asterisk	FreeSWITCH	Kamailio	OpenSIPS	ejabberd	Flexisip
Linphone	INVITE	0	0	0	0	0	32
	MESSAGE	4	211	0	0	0	41
	PUBLISH	0	0	0	0	0	0
	REGISTER	0	0	0	0	0	0
	UPDATE	0	0	0	0	0	0
	REFER	0	0	0	0	0	0
Zoiper	INVITE	0	0	65	293	215	277
	MESSAGE	4	342	0	390	108	159
	PUBLISH	0	0	0	0	0	0
	REGISTER	0	0	0	0	0	0
	UPDATE	0	0	0	0	0	0
	REFER	0	0	0	0	0	0
Jami	INVITE	0	0	0	414	167	87
	MESSAGE	3	465	6	426	171	107
	PUBLISH	0	0	0	0	0	0
	REGISTER	0	0	0	0	0	0
	UPDATE	0	0	0	0	0	0
	REFER	0	0	0	0	0	0
Jitsi	INVITE	0	0	0	0	0	87
	MESSAGE	4	311	0	3	0	98
	PUBLISH	0	0	0	0	0	0
	REGISTER	0	0	0	0	0	0
	UPDATE	0	0	0	0	0	0
	REFER	0	0	0	0	0	0
MicroSIP	INVITE	3	0	13	0	53	112
	MESSAGE	3	471	17	243	112	142
	PUBLISH	0	0	0	0	0	0
	REGISTER	0	0	0	0	0	0
	UPDATE	0	0	0	0	0	0
	REFER	0	0	0	0	0	0
Blink	INVITE	0	0	0	71	7	104
	MESSAGE	4	276	0	10	0	72
	PUBLISH	0	0	0	0	0	0
	REGISTER	0	0	0	0	0	0
	UPDATE	0	0	0	0	0	0
	REFER	0	0	0	0	0	0
MizuPhone	INVITE	2	0	13	81	72	132
	MESSAGE	5	527	17	131	68	112
	PUBLISH	0	0	0	0	0	0
	REGISTER	0	0	0	0	0	0
	UPDATE	0	0	0	0	0	0
	REFER	0	0	0	0	0	0
Empathy	INVITE	0	0	0	5	6	112
	MESSAGE	3	316	0	8	12	117
	PUBLISH	0	0	0	0	0	0
	REGISTER	0	0	0	0	0	0
	UPDATE	0	0	0	0	0	0
	REFER	0	0	0	0	0	0
Twinkle	INVITE	0	0	0	32	42	92
	MESSAGE	4	217	12	8	6	87
	PUBLISH	0	0	0	0	0	0
	REGISTER	0	0	0	0	0	0
	UPDATE	0	0	0	0	0	0
	REFER	0	0	0	0	0	0