Lost in Translation: Text Message Spoofing via Email

Sumanth Rao Ye Shu Stefan Savage
Aaron Schulman Geoffrey M. Voelker Enze Liu[§]
UC San Diego [§]*Carnegie Mellon University*
{svrao, y2shu, ssavage, schulman, voelker}@ucsd.edu [§]enzeliu@andrew.cmu.edu

***Abstract*—This paper identifies and empirically validates a class of vulnerabilities in smartphone messaging systems that allows attackers to easily spoof the identity of participants, even injecting themselves into existing conversations. These issues arise from ambiguities at two levels of the messaging stack. First, that major carriers provide gateways between traditional open email services and the transport services (e.g. IMS) that manage the delivery of SMS/MMS messages in IP-based cellular networks. Second, that the standard messaging apps offered by popular smartphones support identity aliasing both across protocols (e.g., iMessage, RCS, SMS/MMS) and across identifier types (i.e., email addresses, phone numbers, etc.). Taken together, we show that carefully crafted email messages can be used to coerce carriers to issue spoofed IMS messages that will then be displayed as the sender of the attacker's choosing (e.g., arbitrary email address, phone number or short code) — including injecting forged messages into existing messaging threads. We demonstrate working versions of these attacks across a variety of phones (both Android and iPhone) and wireless carriers (including AT&T, Verizon, T-Mobile, and Google Fi), describe how they could be used to support more complex attacks and discuss the range of mitigations and defenses that would offer improved protection.**

1. Introduction

"Text messaging" via cell phone was popularized by the introduction of the Short Messaging Service (SMS) in the early 1990s and accelerated with cross-carrier support by the end of the decade. Supplemented by the Multimedia Messaging Service (MMS) in the early 2000s (to accommodate photos and video as well) and then integrated into IP networks via the IP Multimedia System (IMS), today all cellular carriers natively support these standard messaging features, and they are transparently integrated into the default messaging apps on every major smartphone brand.

Moreover, to help drive text messaging adoption in the U.S., most major carriers also introduced email to SMS gateways (e.g., Verizon's `vtext.com` service was introduced in 2001) to integrate existing SMTP-based email networks with text messaging. Thus, by sending email to `phonenumber@carriergateway.com`, the email content would be relayed, as a text message, to the recipient on the carrier's network with the associated phone number.

While certainly convenient, composing these two services—email and SMS—creates a new and unique attack surface as each messaging protocol has rich, but distinct, rules for addressing and formatting. Indeed, email to text services are not simply relays, but also protocol converters: they must transform the syntax and expected semantics of email headers into the associated fields supported by the messaging transport medium (e.g., IMS) for native text message delivery. Thus, any imperfection in how these gateways perform this mapping or in the assumed integrity of an email's fields, could allow outside parties to inject problematic text messages into their system. This is particularly true with respect to origin forgery since existing email authentication protocols (e.g., SPF, DKIM, and DMARC) are themselves complex and ambiguous. Further compounding this challenge is that composition ambiguity is not limited to email to text gateways, but can also be present in the handset messaging apps themselves, since these have evolved to unify both email and text-based message addressing.

In this paper, we investigate these issues by rigorously exploring both sources of composition ambiguity and their practical security impacts. Concretely, we make three principal contributions:

- **Carrier conversion mapping (sending side)**. We develop a methodology for inferring critical email to messaging conversion rules and test it empirically across the major U.S. cellular providers.[1]
- **Handset message interpretation (receiving side)**. We systematically explore and then validate the key IMS parsing and conversation mapping rules used by the messaging subsystems of popular Android and Apple smartphones. For example, we show how specially-crafted sender email addresses can exploit bugs in messaging apps to have the apps interpret the addresses as phone numbers.
- **Vulnerability assessment**. Finally, using the range of composition ambiguities that we have identified, we develop and empirically validate end-to-end spoofing attacks, demonstrating the range of capabilities available to unprivileged Internet email users and their application to a variety of potential threats. This includes

1. We have also assessed the gateways of national cellular carriers operating in Canada and Mexico, and we present an initial version of this extended analysis in Appendix B.

not only spoofing new messages from arbitrary users, but also injecting messages into existing messaging threads, as well as on behalf of "verified" third parties as implemented via RCS Business Messages (RBM).

Finally, we discuss these problems within the larger challenge of service evolution, composition and integration across legacy protocols, and explore the range of both idealized and practical mitigations available.

2. Background

This paper focuses on the interaction between phone-based messaging and email, and thus requires some basic background in each to contextualize the points of contact between the two services and the emergent opportunities for ambiguity.

2.1. Mobile Phone Messaging

Text messaging over mobile phones was first standardized in the mid-1980s via the Short Message Service (SMS), part of the Global System for Mobile Communications (GSM), and was offered commercially by the mid 1990s. As originally conceived, SMS encoded individual messages of up to 160 7-bit characters, transported as part of the Mobile Application Part (MAP) SS7 protocol inside cellular networks. In the early 2000s, the Multimedia Messaging Service (MMS) commercially introduced support for both longer messages, as well as video clips, images, and audio — encoded using MIME (adopted from the email world). Around the same time, the Third Generation Partnership Project (3GPP) standardized the IP Multimedia Subsystem (IMS), a SIP-based protocol for delivering cellular services over IP networks including voice and SMS/MMS. Thus, today most *native* SMS/MMS messages are sent via IMS.[2]

Against this background, so-called over-the-top (OTT) messaging services were built that operated largely outside the standard cellular ecosystem — notably Apple's iMessage.[3] This allowed such services to more quickly add new features such as location sharing, video calling, combined Wi-Fi/cellular handoff, etc. Partially in response, the GSM defined the Rich Communication Services (RCS) standard, which allowed the incorporation of similar features via a protocol operating in concert with IMS. Over time, the distinction between OTT and non-OTT messaging has become blurred — e.g., Apple's standard messaging app offers support for SMS/MMS and RCS over IMS in addition to its native iMessage transport, and Google's app, while ostensibly RCS-based, will bypass IMS and use native Google RCS servers if possible. Indeed, most default messaging apps attempt to hide the complexity of the underlying transport and will associate all messages purporting to belong to the same sender, regardless of the mechanism used to deliver the message. Critically, legacy SMS/MMS over IMS remains a common denominator service that is widely supported by all networks, phones and default smartphone messaging apps.

2. SMS and MMS *internetworking* between carriers is further complicated and can involve a range of other protocols (e.g., Diameter) and infrastructures (e.g., GPRS Roaming eXchange, aka GRX) but, for the most part, the details of their function are secondary to the issues discussed here.

3. iMessage uses the cellular system for phone number validation, but is otherwise run completely independently, although the Apple messaging app will integrate messages delivered to a phone number via other delivery mechanisms such as IMS.

2.2. Email to Text Message Gateways

Early SMS messaging was local to particular carriers, but by 1999 inter-carrier exchange and billing had been worked out, and thus phone numbers became a global identifier for messaging. In part to encourage this broad use and to integrate with the dominant form of Internet communications at the time — email — carriers introduced email to SMS (and then MMS) gateways. For example, in late 2001 Verizon introduced `vtext.com` which would accept emails of the form `number@vtext.com` (e.g., `9876543210@vtext.com`) and deliver them, via SMS, to the Verizon customer with that phone number (followed in 2003 by `vzwpix.com`, which could also gateway MMS-compatible content in this manner). Over time, all major US carriers maintained such gateways, as well as many independent Mobile Virtual Network Operators (MVNOs).

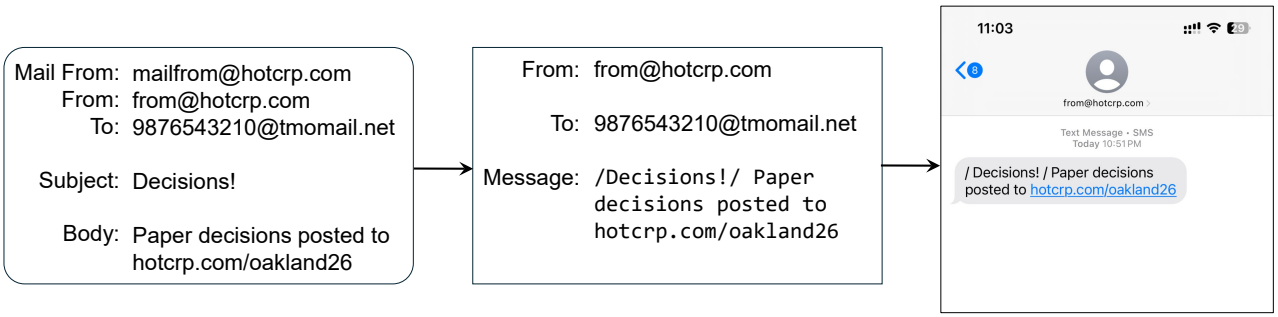
Such gateways operate by implementing standard Internet email service: senders look up the gateway via the Domain Name System (DNS), query the Mail Exchange (MX) record, look up its IP address, connect via TCP on port 25, and then submit email via the Simple Message Transport Protocol (SMTP). Part of the gateway's "job" is to validate that an email is well-formed: that the destination email address is formatted correctly and belongs to a customer served by the gateway, that the source email address has not been spoofed (i.e., using existing email anti-spoofing protocols including SPF, DKIM, and DMARC), and that the body does not contain malicious or unwanted content (typically via anti-spam filters and policy rules on acceptable MIME types for attachments). If any of these checks fails, then the gateway rejects the message and does not send it to the recipient.

Finally, an email must then be converted into a form compatible with the underlying message type (e.g., SMS, MMS, etc.) and transport (e.g., IMS). This conversion may include re-encoding text, images, and other attachments as well as segmenting, as needed, to meet the requirements of the particular transport modality. Moreover, standard email *To* and *From* headers must be mapped to identifiers that will be consistently interpreted by cellular messaging systems. Figure 1 depicts a high-level example of how such header mapping takes place in this workflow.

2.3. Related Work

This paper builds on prior work in both email security and SMS spoofing, which we highlight briefly here.

Email Spoofing. The original SMTP design did not provide any mechanism to guarantee the validity of a sender's identity — a lack which gave rise to widespread email sender



(a) Email to gateway (b) Gateway converts to SMS (c) Recipient phone (987) 654-3210

Figure 1: Example of T-Mobile's SMS gateway converting an email to an SMS message. The gateway has to select, parse, and authenticate the sender identity and recipient, and convert the subject and body of the email into an SMS message. Each of these operations can contribute to convincing spoofing attacks if not performed correctly.

spoofing. To address such abuse, a range of "add-on" email protocols were created and deployed to limit spoofing.

However, these approaches can be complex in practice. As such, one line of research (e.g., [4], [6], [9], [12], [13], [15], [20], [26], [28], [45], [54]) has empirically analyzed the deployment, management, and misconfiguration of the three major email authentication protocols: SPF, DKIM, and DMARC. Our methodology for measuring carrier deployment of SPF, DKIM, and DMARC builds on this prior work.

As well, there is a significant research literature focused on particular techniques for *defeating* such anti-spoofing measures. Among these, Hu et al. [27] conducted an end-to-end measurement of email spoofing attacks at 35 major email providers, Chandramouli et al. [10] studied email spoofing through header injection, Chen et al. [11] and Shen et al. [40] highlighted security vulnerabilities and possible spoofing attacks in the context of composing email services, Liu et al. [32] and Wang et al. [52] constructed email spoofing attacks by abusing forwarding, Wang et al. [53] and Wang et al. [55] demonstrated email spoofing vulnerabilities in the context of shared SPF, and Ma et al. [33] crafted email spoofing attacks by exploiting the *Sender* field. While our work is focused on strictly the interplay between email and SMS, we take advantage of several such previously identified mechanisms to bypass weak anti-spoofing controls used on the email side of these gateways.

Spoofing in Mobile Networks. Mobile cellular networks have long had spoofing challenges as well. These include various non-messaging issues orthogonal to our work, such as spoofing caller ID in VoIP and VoLTE [14], [35], and spoofing emergency alerts in 4G [31] and 5G networks [7].

However, much closer to our research is the broad literature on text message spoofing (particularly driven by SMS "spam"). For example, attacks employing fake cellular base stations are readily able to spoof such messages [39], [62]. Similarly, it has been well-documented that the lack of authentication in 2G mobile networks enables spoofing [48], [63]. Kim et al. [29] have demonstrated similar attacks in LTE networks as well, and Zhao et al. [64] showed how spoofing attacks could be accomplished in RCS with physical access to the device of the victim. Yet others have explored spoofing vulnerabilities in the context of IMS. For example, Tu et al. [48] uncovered a variety of vulnerabilities of IMS-based SMS and crafted various attacks, including SMS spoofing. Shi et al. [42] showed additional spoofing possibilities in IMS with a different threat model (malware on the victim's mobile device). Wang et al. [56] further highlighted SMS spoofing possibilities by exploiting SIP semantic ambiguities. Finally, research by Tsunoda [1], [46], [47] identified spoofing vulnerabilities in *commercial* SMS gateways (e.g., SMSGlobal). Our work is most closely aligned with these latter efforts, but removes any requirement for privileged access to the IMS system, the victim's device, or the ability to deploy fake base stations. Instead, we build on a broadly and publicly accessible attack surface: the email to text gateways offered by mobile carriers.

3. Threat Model

In this paper, we focus strictly on spoofing attacks, where an attacker seeks to send a message that appears to originate from a *sender identity* that they do not control.[4]

Unlike previous work, which generally requires that the attacker have some privileged technical capability to inject messages (e.g., access to a custom SMS gateway [46], [47] or the ability to inject raw IMS messages [42], [48]), our threat model imposes only minimal technical requirements on the adversary: an Internet host capable of making SMTP connections (i.e., to carrier gateways), software (e.g., telnet or curl) for customizing the headers and body of email messages, and a domain under the attacker's control (e.g., `attacker.com`) with permissions to configure its DNS records such as SPF and DMARC.

In addition to these basic technical requirements, we also assume our attacker can obtain sufficient intelligence

4. In Section 6.4 we briefly discuss how such a capability might be amplified to *also* receive replies from the victim, but that is not a core focus of this work.

about their target to mount the attacks we describe. For most attacks, this information includes identifying the target's phone number, wireless carrier, and the carrier's email gateway (and a similar requirement for the sender identity the attacker seeks to spoof "as"). An attacker can learn the given person's phone number through various means, including through online search and phone directories, via a public breach dump, or through a personal connection. From a phone number, an attacker can typically infer the wireless carrier providing service by using public databases of phone prefixes (e.g., NANPA [36]) or using free services that perform cell carrier lookups [21]. Finally, an attacker can identify a carrier's email gateway by searching online for documentation (as we have done).

For more advanced attacks, an attacker may also need additional information specific to the attack. For example, when spoofing as a trusted entity (Section 6.2), an attacker may need to know special origin phone numbers (e.g., Gemini) or addresses (e.g., RCS Business Messages) that are used to signal that trust. Similarly, to spoof a group conversation (Section 6.3), an attacker will also require the email addresses or phone numbers of the other participants in the group. Finally, for particular kinds of spoofing (e.g., when spoofing as a specific *phone number*) the attacker may need to know if the target uses an iPhone or Android phone, as the attack details depend on the platform.[5]

4. Security Analysis of Carrier Gateways

This section identifies major U.S. email to text carrier gateways, analyzes their behavior in converting email to mobile messages and evaluates their security mechanisms. Specifically, we (1) infer how gateways derive the sender identity from SMTP headers, and (2) assess the extent to which gateways apply email security mechanisms to spoofed messages. These represent the limitations on what kinds of IMS messages can be *sent* via our attacks, a process that is independent of the destination phone. Subsequent sections will focus on how such messages are parsed and displayed on the handset and how these vulnerabilities can be combined to form complete attacks.

4.1. Identifying Gateways

We identified active gateway domains for twenty popular carriers, including the three dominant mobile network operators (MNOs) and popular mobile virtual network operators (MVNOs) in the U.S. (as per Section II.B.1 of the 2024 FCC Communications Marketplace Report [18]).[6] Since many of these carriers share gateways due to business relationships, we focus on the most well-known carrier that uses each of the shared gateways (we provide a full list of such carriers, grouped by their email to text gateways in Table 6 of Appendix C.1). Table 1 summarizes the advertised gateway domains that we evaluate, the representative carriers that use them, and the MX domain (i.e., returned from an MX record lookup of the gateway domain) they use for receiving email.[7] We consider gateway domains for SMS and MMS separately, as they exhibit different behavior. We purchased service and SIM cards for each of these carriers, and ensured that their gateways were active by resolving their MX records and verifying that port 25 at the associated MX domain accepted SMTP connections and delivered email messages to our phones.[8]

Carrier	Gateway Domain → MX Domain
AT&T SMS	txt.att.net → att-e2xms-west.mx.a.cloudfilter.net
AT&T MMS	mms.att.net → att-e2xms-west.mx.a.cloudfilter.net
Google Fi MMS	msg.fi.google.com → alt3.gmr-smtp-in.l.google.com
Verizon SMS	vtext.com → smtpin02.vzw.a.cloudfilter.net
Verizon MMS	vzwpix.com,mypixmessages.com → smtpin02-mms.vzw.a.cloudfilter.net
T-Mobile SMS	tmomail.net (SMS) → tmo-west.mx.a.cloudfilter.net
T-Mobile MMS	tmomail.net (MMS) → tmo-west.mx.a.cloudfilter.net

TABLE 1: U.S. wireless carrier SMS and MMS services, their gateway domains, and the MX domain they point to. Google Fi only delivers messages via MMS.

4.2. Inferring Gateway Conversion Strategies

The core functionality of an email to text gateway is converting an email message to a mobile message, which we refer to as their *conversion strategy*. After validating the sender, the gateway must carefully map the *sender identity* and *message content* from an email message format to a mobile message format. Different message types (SMS and MMS) have different standards, further adding to the complexity of this mapping. Indeed, while there are standards (3GPP TS 23.140 [16] and RFC 4356 [22]) that provide high-level guidelines for this conversion process, as we show later, the implementations vary across gateways.

5. To a first approximation, this distinction is readily available via Apple Messages which will "color code" messages being composed based on whether the target phone number has Apple Messages enabled or not.

6. While there are online lists of gateways [19], [59], we found them to be outdated: many gateways no longer work because the carrier has been acquired, merged with another, or went out of business. In most cases, such as Metro PCS's acquisition by T-Mobile [44], their customers migrated to the new business and, hence, now use the email gateway of the new business (i.e., former Metro PCS customers now use T-Mobile's gateways).

7. We note that all but Google Fi appear to have outsourced their mail handling to Cloudmark, a subsidiary of Proofpoint that sells secure messaging services to ISPs, mobile operators and other large customers.

8. We also purchased wireless service for a sample of four *other* MVNO carriers that share these gateways (e.g., Tracfone SMS uses the same gateway as Verizon SMS), and they all behaved the same in our experiments.

Carrier Gateway	Valid MF & FROM	Valid MF	Valid FROM
AT&T SMS	N/A	N/A	N/A
AT&T MMS	MF	MF	$\boldsymbol{\times}^{Rej}$
Google Fi MMS	MF	$\boldsymbol{\times}^{Rej}$	$\boldsymbol{\times}^{ND}$
T-Mobile SMS	From	MF	$\boldsymbol{\times}^{ND}$
T-Mobile MMS	From	MF	$\boldsymbol{\times}^{ND}$
Verizon SMS	From	$\boldsymbol{\times}^{ND}$	From
Verizon MMS	MF	MF	From

TABLE 2: Sender identity conversion behavior for carrier gateways. **N/A**: Reserved phone number is used as the sender information. **MF**: Message delivered and *Mail From* header used. **F**: Message delivered and *From* header used. $\boldsymbol{\times}^{Rej}$: Gateway rejects the message during the SMTP session. $\boldsymbol{\times}^{ND}$: Message not delivered to phone.

4.2.1. Methodology. Since gateway implementations are not public, we treated each gateway as a black box and empirically inferred the conversion strategy it employs. Then, using existing standards as a guide, we conducted a series of experiments which test the gateway's behavior under varying header or body conditions. These experiments not only helped us understand the normal conversion process, but also revealed how the gateway handles edge cases, which are not standardized and handled differently by providers. For both headers and message body, we started with common cases where all relevant fields are present, and then tested edge cases where one or more fields were missing (or empty if the field has to be present) or duplicated (if applicable).

Sender and recipient headers. For the sender identity, we started with common cases where both the standard SMTP *Mail From* and *From* email headers exist. We then tested three edge cases: 1) the *From* header is missing, 2) multiple *From* headers exist, and 3) the *Mail From* header is empty ("<>"). Similarly, for the recipient identity we started with the common case where one *To* header exists, and then tested edge cases where the *To* header is missing and multiple *To* headers exist.[9]

Subject and message body. We started with the common case where both the *Subject* header and the message body exist. We then tested edge cases where the *Subject* header is missing. In addition, for MMS we added common MIME attachments (e.g., images or text files) to the email message body to observe how the gateway handles them.

Capturing and parsing converted mobile messages. Finally, we established the mapping between the email message format and the converted mobile message. We first obtained the raw Protocol Data Unit (PDU [17], [37]) of each mobile message, which represents the raw byte stream received on the device after modem processing (Appendix C.2 shows an annotated example). Using raw PDUs allows us to analyze message content before it is parsed and rendered by a messaging app and thus most closely captures how it was sent via IMS.[10] We then inferred the conversion strategy by comparing PDU fields with the original email headers and content. For SMS, the sender's email address is encoded as part of the content in a standard format, indicating that it should be displayed as the sender of the message [17]. For MMS, we use the standard sender field, *From* [37].

9. Other headers such as *Sender* can also indicate the sender identity. However, from our tests gateways rarely rely on them compared to *From* or *Mail From*, even when these additional headers are present.

Carrier Gateway	MMS PDU FROM	MMS PDU SUBJECT	SMS/MMS PDU BODY
AT&T MMS	F	S	B
Google Fi MMS	F	S	B
T-Mobile SMS	-	-	F / S / B
T-Mobile MMS	F	S	B
Verizon SMS	-	-	F (S) B
Verizon MMS	F	S	B

TABLE 3: Message content conversion behavior for carrier gateways. F: *From*, S: *Subject*, B: *Body*

4.2.2. Results. Based upon our experiments, we describe the carrier gateway conversion strategies used by various providers to derive the sender identity and message content.

Sender Identity. Table 2 summarizes the different sender identity conversion strategies used by the carrier gateways. Regardless of the email address used in an email message, the AT&T SMS gateway only uses a pool of reserved phone numbers as the sender identity and wraps the original sender in the message body. We mark this as N/A in Table 2 and exclude it from further discussion since it directly prevents spoofing. The AT&T MMS gateway always uses the address in the *Mail From* email header and refuses to deliver a message if this header is empty.

Three other gateways, Verizon SMS, T-Mobile SMS, and T-Mobile MMS, all prioritize the *From* header over the *Mail From* header. However, they exhibit different fallback behaviors: Verizon SMS falls back to *From* when *Mail From* is empty and does not deliver a message when *From* is missing. In contrast, the T-Mobile SMS and MMS gateways fall back to *Mail From* when *From* is missing, and do not deliver a message when *Mail From* is empty.

Google Fi and Verizon MMS use the *Mail From* header as the sender when both *Mail From* and *From* are present. If either is missing, Google Fi does not deliver the message. In contrast, the Verizon MMS gateway falls back to the *From* header when *Mail From* is empty.

Such variations demonstrate the lack of standardization and complexity in conversion strategies. As we show later, some fallback behaviors lead to spoofing vulnerabilities.

Message Content. Table 3 shows the content conversion strategies for the carrier gateways. All carriers distinguish the subject and body syntactically, but have a wide variety

10. We obtained raw PDUs by installing a custom replacement for the default Android messaging app.

Carrier Gateway	Basic	Mismatched Logic	Missing Header	Parsing Error
AT&T MMS	—	—	—	—
Google Fi MMS	MF: attacker.com F: example.com	MF: example.com F: attacker.com	—	—
Verizon SMS	MF: attacker.com F: example.com	—	MF: <> F: example.com	—
Verizon MMS	—	MF: example.com F: attacker.com	MF: <> F: example.com	—
T-Mobile SMS	MF: attacker.com F: example.com	—	MF: example.com	MF: attacker.com F: @user,@example.com:bad@attacker.com
T-Mobile MMS	MF: attacker.com F: example.com	—	MF: example.com	MF: attacker.com F: @user,@example.com:bad@attacker.com

TABLE 4: Vulnerability of carrier gateways to various kinds of spoofing attacks (spoofing `example.com`).

of behaviors. All MMS gateways (Verizon, Google Fi, T-Mobile, and AT&T) use the *Subject* header in the MMS PDU, and transform the body as a plain text body part in the MMS PDU. The SMS gateways, on the other hand, have more restricted formats. Per specification [17], the email address of the sender should be prepended to the message body in the SMS PDU. SMS does not have a dedicated or standardized subject field, though, and we observe different formats among carrier gateways: T-Mobile's SMS gateway encodes the subject between two slashes, and Verizon's SMS gateway encodes the subject between parentheses. While carriers may use stylized subject formats to indicate that the message content is converted from an email messages, an attacker can suppress these formats and deliver messages that look like normal text messages (Appendix C.5).

4.3. Bypassing Sender Authentication

For an attacker to spoof the sender identity of an email message they need to bypass the sender authentication mechanisms of the gateway. Guided by prior work in email security, we developed a suite of nearly 125 spoofing attacks to evaluate a broad range of basic and advanced spoofing techniques for bypassing the gateway's sender authentication mechanisms. The source code for all scripts is available at https://github.com/ucsdsysnet/email2sms.

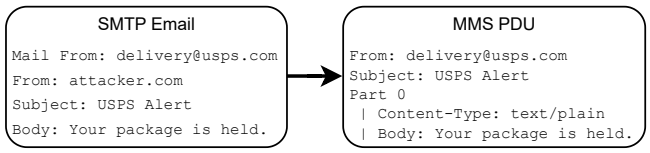
4.3.1. Methodology. Modern email systems use three common anti-spoofing mechanisms: SPF (authenticating sender IP via the *Mail From* domain), DKIM (cryptographically signing message content), and DMARC (aligning the *From* domain with SPF or DKIM results). If an email message fails DMARC, the owner of the *From* domain (typically the domain being spoofed) expresses a policy in its DMARC record that the gateway uses to determine the outcome: *Reject* or *Quarantine* the message, or *None* to deliver normally. Following prior work [13], [20], [27], we configured a domain with SPF, DKIM, and DMARC records and ran an authoritative DNS server to capture all DNS queries from gateways processing our emails (using unique subdomains with a TTL of 1s). For each gateway, we inferred support for these mechanisms and conducted spoofing tests to evaluate their implementation.

We performed two kinds of spoofing tests (basic and advanced) that model different degrees of attacker sophistication. Basic spoofing tests model cases where an attacker simply spoofs the *From* header of an email message without any additional modification. Inspired by Hu et al. [27] and Liu et al. [32], we spoofed four categories of domains: (1) domains that do not support SPF, DKIM, and DMARC, and domains that do using policies of (2) *Reject*, (3) *Quarantine*, and (4) *None*. For each category, we spoofed as a domain under our control as well as a top-ranked Tranco [30] domain. These choices of domains represent different levels of domain reputation to detect any potential special protection for often-targeted domains (observed by Liu et al. [32] and in our own experiments). We configured the spoofed email message to pass SPF and DKIM using a domain under our control (but not the same as the domain in the *From* header).

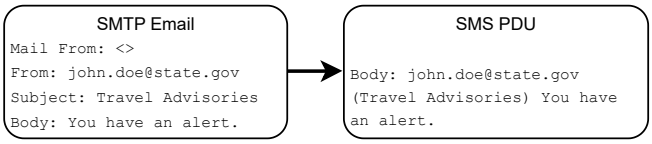
Advanced spoofing tests model a more sophisticated attacker. Inspired by attacks described in Chen et al. [11] and Shen et al. [40], our test suite includes nearly 75 spoofing techniques that broadly belong to four categories. These categories include missing or duplicate headers (e.g., multiple *From* headers), obsolete headers (e.g., routing addresses and comments in *From*), malformed headers (e.g., *From* header that is not an email address), and malformed email addresses (e.g., an email address with a control character in the local part, or with a TLD that does not exist). While our tests covered a broad range of techniques, they might not be complete due to the blackbox nature of the gateways.

4.3.2. Results. Our measurements indicate that all gateways support SPF, DKIM, and DMARC. All but one (AT&T MMS) reject email messages that naively spoof as a domain that has a DMARC policy of *Reject*. AT&T MMS ignores DMARC and relies solely on SPF authentication: email messages that pass SPF are delivered, while those failing SPF are blocked, regardless of DMARC alignment status.

Despite these security measures, the gateways are vulnerable to a variety of spoofing attacks that bypass these protections. Table 4 summarizes the attacks in five categories,



(a) Spoofing `usps.com` with invalid *From* on Google Fi's MMS gateway.



(b) Spoofing `state.gov` with empty *Mail From* on Verizon's SMS gateway.

Figure 2: Examples of carrier gateway conversions of sender identity and message content under (a) mismatched logic and (b) missing header attacks (parsed representation of the PDU shown for clarity).

each of which we detail below. In these attacks, the goal of the attacker is to have a gateway deliver a message spoofed as coming from `example.com`, a domain not under control of the attacker. We start with basic spoofing attacks, and then progress to more sophisticated versions.

Basic Spoofing Attacks. All three gateways that prioritize *From* over *Mail From* are vulnerable to a basic spoofing attack: spoofing any target domain that has a DMARC policy of *None* or *Quarantine*. An attacker can simply set the *From* header to the desired spoofed domain and their own domain `attacker.com` in the *Mail From* header.

The ability of attackers to spoof domains with a DMARC policy of *Quarantine* is particularly concerning, as this policy is widely regarded as strict. If the recipient is a Web mail service, for instance, the message is typically delivered to the recipient's spam or junk folder. However, because of the conversion process and the lack of a "spam" folder in mobile devices, the gateways deliver the message to the recipient's phone like a regular text message.

Mismatched Logic Attacks. Two gateways, Google Fi and Verizon MMS, decide whether to deliver a message using the outcome of DMARC authentication. However, while DMARC authenticates using the *From* header, the two carrier gateways use the address in the *Mail From* header as the sender. This mismatch allows an attacker to use a domain they control in the *From* header to pass DKIM and DMARC authentication, while placing a domain protected by a DMARC policy of *Reject* (e.g., `usps.com`) in the *Mail From* header, which the gateways display as the sender (Figure 2(a)). In addition, the attack also succeeds for Google Fi when using a malformed *From* header (e.g., an alphanumeric string that is not a valid email address).

Missing Header Attacks. Three gateways have vulnerabilities when authenticating an email message with a missing header, which allows an attacker to bypass DMARC authentication checks. As shown in Figure 2(b), Verizon's SMS and MMS gateways incorrectly authenticate an email message when the *Mail From* header is empty ("<>"). Both gateways deliver the message using the address in the *From* header as the sender identity without performing any authentication checks. This vulnerability allows an attacker to spoof arbitrary domains. In addition, Verizon's SMS gateway uses the content of the *From* header as is without any formatting checks, allowing an attacker to spoof as arbitrary alphanumeric sender identities. We show that this practice creates significant security issues in Section 5.

Lastly, T-Mobile's gateway incorrectly authenticates an email message when the *From* header is missing: the gateway falls back to using *Mail From* as the sender identity (Table 2) and does not perform any authentication checks. T-Mobile's gateway also exhibits special protections for popular domains (e.g., `usps.com`) but not for other DMARC policy of *Reject* domains (e.g., `box.com`). For popular domains, the gateway returns an SMTP error code *552 5.2.0 sender rejected AUP#SNDR*, suggesting the spoofed domain triggers a reputation-based filter. For less popular domains (e.g., lower-ranked Tranco domains and our own domain, both with a DMARC policy of *Reject*), the gateway delivers the message with the spoofed sender identity.

Parsing Error Attacks. First highlighted in [11], some email server implementations incorrectly parse complex email addresses that contain routing information. For example, the following address is a valid SMTP *From* header:

```
<@route1.com,@route2.com:user@site.com>
```

The first two domains, separated by a comma, are treated as routing information, while the delivery address is `user@site.com`. T-Mobile SMS and MMS gateways have issues when parsing this type of address. In particular, when the routing domain is malformed, the gateways fail to recognize it and incorrectly process the address. For example, with the following *From* header where the first domain is malformed (`user` instead of `user.com`):

```
<@user,@example.com:bad@attacker.com>
```

The T-Mobile SMS and MMS gateways incorrectly convert the above header to `user@example.com` as the sender and ignore the real address, `bad@attacker.com`. This vulnerability allows an attacker to spoof arbitrary domains.

Summary. The attacks identified in this section document a variety of ways to bypass the sender authentication mechanisms of the gateways. In the worst case, attackers can spoof email messages with an arbitrary local name and domain of the attacker's choosing. In one case (Verizon SMS), attackers can spoof arbitrary alphanumeric strings as well. Next, we show that these attacks can further exploit vulnerabilities on the receiving phones to also transform from an email address to a phone number or short code.

5. Security Analysis of Phone Messaging Stack

Next, we evaluate how control over the sender identity that email to text gateways provide enables novel spoofing

Parsing Vulnerability	Phone	Spoof As	Received As	Parsed As	Layer
Special Sequence	Apple	Phone Number	`9876543210=?@attacker.com`	+1 (987) 654-3210	Telephony
		Short Code	`54321=?@attacker.com`	54321	
		Text	`Chase=?@attacker.com`	Chase	
Space	Apple	Phone Number	`"9876543210 "@attacker.com`	+1 (987) 654-3210	Telephony
		Short Code	`"54321 "@attacker.com`	54321	
		Text	`"Chase "@attacker.com`	"Chase	
Invalid Address	Apple	Phone Number	`‘9876543210’` or `"9876543210"`	+1 (987) 654-3210	Telephony
		Short Code	`‘54321’` or `"54321"`	54321	
		Text	`Chase`	Chase	
Numeric Address	Android	Phone Number	`987@6543.210`	+1 (987) 654-3210	Google Msgs
		Short Code	`5@4.321`	54321	

TABLE 5: Parsing vulnerabilities in Apple and Android messaging stacks.

vulnerabilities in smartphone messaging stacks. We study the two most popular smartphone messaging stacks, Apple's iOS and Google's Android. Through brute force analysis, validated by reverse engineering, we discovered that carefully-crafted email addresses can confuse the stacks and cause them to convert email sender identities into spoofed phone numbers, short codes, or even alphanumeric codes. In most cases, the spoofed messages appear in the messaging app as coming from the legitimate sender, and in some cases insert into existing conversations, with only an indication that the conversation switched to SMS (and out of RCS or iMessage). We also performed a historical analysis of these vulnerabilities. The Apple vulnerability appears in their OS library binaries since at least 2012, and the Google vulnerability may have existed since at least 2017.

5.1. Address Parsing Vulnerabilities

Spoofing as a short code, phone number, or text string using a specially crafted email address requires a vulnerability in the messaging stack that will cause it to incorrectly convert an email address into these formats (e.g., converting `1234@attacker.com` into the short code 1234). Finding these vulnerabilities is challenging, as the messaging app as well as many libraries process SMS/MMS sender identities. Instead, we treated the sender identity parsing functionality as a black box and brute forced a variety of character patterns in email addresses. Then we validated the specific characters that caused a spoofing capability by reverse engineering libraries and applications to find special handling of these character patterns (Section 5.2). Table 5 summarizes the four kinds of vulnerabilities we discovered.

Methodology. We started by finding character patterns that caused the messaging stack to parse an email sender identity incorrectly. On discovering a vulnerable pattern, we then generated a set of test cases that determined the types of spoofing that this pattern enables (e.g., converting an email address to a phone number).

In the address local part, we tested all combinations of ASCII characters and extended UTF-8 bytes in runs of 1–3 characters (e.g., `hello&$?@attacker.com`). We also tested RFC-compliant double-quoted local parts (e.g., `"a b"@attacker.com`) and non-standard single-quoted variants, embedding spaces, Unicode, SMTP comment syntax (`'user(comment)'`), and delimiters illegal outside quotes (commas, colons, semicolons). We also tested email addresses with entirely numeric, truncated, or structurally incomplete registered domains and TLDs (e.g., `1@2.3`) with combinations of invalid local parts, domains, and TLDs.

We used Verizon's SMS gateway to deliver these test messages to both iOS and Android phones since Verizon had the most permissible sender identity gateway conversion attack (Section 4.3). We then compared the received sender identities with the ones that we sent from the cellular network to the phone based on the raw message PDUs stored in the device's local SMS/MMS databases (Section 4.2.1). For each vulnerable pattern that caused a visible change in the sender identity, we generated test cases that attempted to spoof as a phone number (both US and international numbers), a short code, or text. We also verified that the messaging app displays the spoofed sender identity stored in the database to the user when reading messages.

We tested if there are differences in how the OSes process messages on different models of devices using three versions of Apple iPhones[11] with iOS versions 17, 18.5, and 26, and three Android devices from vendors with the largest U.S. market share [43]: Google Pixel Fold with Android 16, Samsung Galaxy S22+ with Android 14, and Motorola Moto G Play with Android 13. We used the native messaging app on iOS and the two most popular messaging apps on Android: Google Messages for Google and Motorola, and Samsung Messages for Samsung Galaxy.[12]

Apple iOS Vulnerabilities. We discovered three vulnerabilities with iOS's messaging stack, each of which allows an attacker to spoof as an arbitrary phone number, short code, and alphanumeric number. First, iOS's messaging stack cuts off all characters in the sender identity after the MIME encoded-word start sequence "=?", which causes the messaging

11. Due to Apple's closed-source nature, we were unable to capture the SMS/MMS PDU on an iPhone. Instead, we used the equivalent message from Android as a proxy and assumed that the gateway delivers the same message to both Android and iPhone.

12. Starting with Galaxy S21, the default messaging app for Samsung phones is Google Messages [49]. Prior to this, the default messaging app was Samsung Messages. We tested both apps on the Galaxy S22+.

subsystem to interpret just the left portion as the sender identity (Section 5.2). Depending on what appears before the sequence, iOS converts the sender identity to a phone number, short code, or alphanumeric text. For example, if the email address is `9876543210=?@attacker.com`, iOS converts it to a phone number (+1 987-654-3210). A variant of this attack also works for international numbers by prepending the country code (e.g., `+919876543210=?`).

In addition to "`=?`", an ASCII space (" ") has similar behavior: iOS converts only text on the left side of a space to a sender identity. Note that this attack requires a double-quoted email address (e.g., `"54321 "@attacker.com`): iOS's messaging stack strips the leading double quote when the text looks like a phone number or short code, but keeps the double quote when the text consists of letters.

The last vulnerability pertains to how iOS's messaging stack parses the body of a raw SMS PDU. Recall that the gateway conversion process for SMS pushes the sender identity to the SMS PDU body (Section 4.2.2). Thus, an SMS parser in the messaging stack must parse the sender identity from the SMS PDU body, overriding the default originating address for email gateway conversions (Appendix C.2 shows an example Verizon SMS PDU). It is during this process that Apple's message stack skips validation if the sender identity in the SMS PDU is a valid email address, and instead parses the invalid address as alphanumeric text (e.g., "Chase"), phone number, or short code. In contrast, the Android messaging stack only overrides the originating address if it is a valid email address.

Android Vulnerabilities. We also identified a significant vulnerability with Android's Messaging stack when the Google Messages app [23] receives SMS messages. Android incorrectly deletes email address specific characters (i.e., '`@`' and '`.`') from email addresses that look like phone numbers (or short codes). This behavior enables an attacker to use an email address to spoof an arbitrary phone number or short code. For example, an SMS in Android's database contains the phone number "+1 (987) 654-3210" when sent from the email address `987@6543.210` (using country codes similarly spoofs international numbers).

Note that Android does not exhibit this vulnerability when the default messaging app is Samsung Messages on Samsung Galaxy phones.

5.2. Root Cause Analysis

Next, we find plausible validation for the root causes of the vulnerabilities identified in both iOS and Android's messaging stacks.

Methodology. For Android, we manually examined the open-source code of the Android telephony framework and used JADX to reverse-engineer parts of the closed-source Google Messages app to understand how it handles message sender identities. For iOS, we manually reverse engineered the SMS/MMS libraries CoreTelephony and IMCore using Ghidra for iOS 26.1 (23B85) released on March 11, 2025,

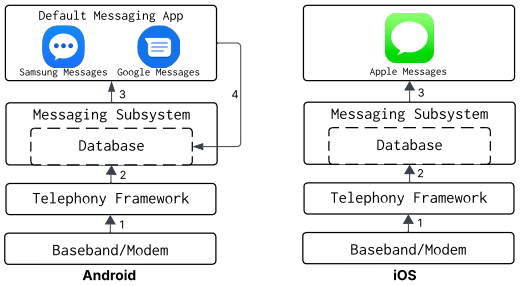


Figure 3: The four layers of the OS messaging stack on Apple's iOS and Google's Android.

using the ipsw tool [8] to obtain the latest version of the firmware. In both cases, we searched for the character patterns that caused the incorrect conversion of sender identities. We also searched for the code that processes email and phone number sender identities to see how they determine if a sender identity is an email or phone number, and how they process each differently.

Background. For both OSes, once the carrier delivers the PDU to the modem, the message goes through a series of layers of libraries and finally the messaging app where it is ultimately displayed to the user. The message can be filtered or manipulated at any of the four layers shown in Figure 3. However, the OSes differ in what order the messages are parsed in these layers: iOS has a messaging subsystem library that stores the messages in a database, then the messaging app just reads this database to display them. Android instead sends messages directly from the Telephony Framework to the *default* Messaging App, which then stores and reads them from the database in the messaging subsystem. This difference means that it is possible that a sender identity conversion can happen in the Messaging App on Android, but it can only happen in libraries on iOS.

Android's Messaging Stack. The Android SMS/MMS delivery architecture places a significant portion of the message processing logic in the default messaging app, rather than in the Android OS telephony framework. When the Android telephony framework receives a PDU, it passes it to the default messaging app (Google Messages in our case) through a broadcast intent [3]. The messaging app parses the message, displays it to the user, and persists it to the Android messaging database (`mmssms.db`) through the Telephony content provider APIs [2], [34].

We discovered that the raw PDU and the `SmsMessage` object parsed by the Android telephony framework preserves the sender as the original all-numeric email address (e.g., `987@6543.210`). However, when Google Messages receives the message, it checks if the sender address is an email address using a regular expression. Unfortunately, this regular expression has a bug that requires the email address domain to have a top-level domain (TLD) ending in an alphabetic character. Thus, an all-numeric email address—although syntactically correct per RFC 5322 [38]—fails this check and Google Messages treats it as a phone number.

Finally, once Google Messages considers the sender identity as a phone number, it sanitizes this "number" by stripping non-numeric characters (e.g., '@' and '.'), and stores the number in the Android messaging database. Figure 5 shows an example of this behavior, where we spoofed an email message with an all-numeric email address `987@6543.210` and Google Messages displayed it as the phone number "+1 (987) 654-3210".

We looked at historical versions of Google's messaging app and found that a similar email address matching regex appears in Google's Messaging apps dating back to at least 2016 (version 2.0.068).

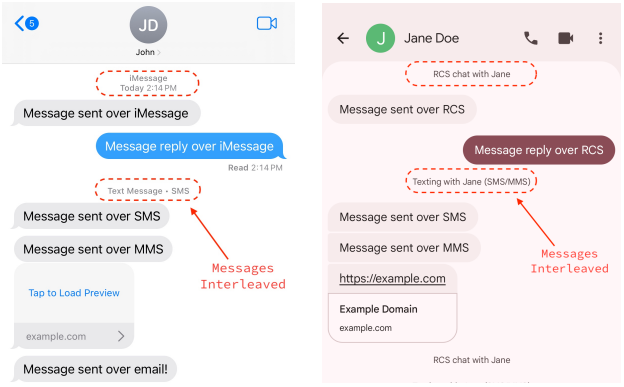
Apple's Messaging Stack. While Android's address parsing vulnerability is in the messaging app layer, Apple's vulnerability is in a different layer: the Core Telephony framework of Apple's messaging stack [58]. From reverse engineering the Apple libraries, we discovered that Apple's telephony layer infers whether the sender address is an email address or phone number based on the presence or absence of a single '@' character. If it infers the message to be an email message, it always treats it as an MMS message and therefore performs MIME decoding on the sender identity before storing it into the database. This behavior appears to explain why we discovered that the sequence "=?" (the special sequence indicating a non-ASCII, MIME-encoded string) causes the sender address to be truncated to just the string before the sequence. The code also indicates that if the sender address is not an email address, it is not MIME decoded and is passed through the telephony layer as is.

Although we are unable to execute the firmware in a debugger, results from black-box tests are consistent with our interpretations from reverse-engineering the libraries: (1) when sending email messages with a MIME encoded sender address (e.g., `=?UTF-8?B?w7w?=`), the displayed sender is MIME decoded (ü), and (2) when sending email messages with a plain text sender address (e.g., `Chase=?` with the special sequence but without an '@' character), the displayed sender is not MIME decoded and Apple Messages displays it as is (e.g., `Chase=?`).

We also tested and reverse engineered this behavior on a much older version of iOS, iOS 9 from 2012 using an iPad Mini 1st gen. We found it has consistent behavior across both versions of the firmware (iOS 9 and iOS 26), indicating that this is a long-standing vulnerability in how iOS handles email-formatted sender identities.

5.3. Interleaving Conversations

To improve user experience, messaging apps can coalesce messages to create a unified view of a conversation with another person: regardless of whether a person communicates from their work or personal number, through SMS, MMS, iMessage, or even by email, messages from this person can be grouped together. While convenient, this practice assumes that the sender identity of a message is trustworthy. However, this assumption does not hold in the presence of spoofing attacks, which enable attackers to inject spoofed



(a) iPhone (b) Android

Figure 4: Interleaving behavior of Apple Messages and Google Messages across contact methods and protocols.

messages into existing conversations and significantly boost the trustworthiness of the spoofed message.

Methodology. We systematically investigated how messaging apps coalesce messages from different contact methods and protocols. For each phone and messaging app we examined: (1) whether messages from the same phone number but different protocols (e.g., SMS, MMS, RCS, and iMessage) are merged; and (2) whether messages from different contact methods (phone number vs. email via carrier gateway) belonging to the same contact are merged.

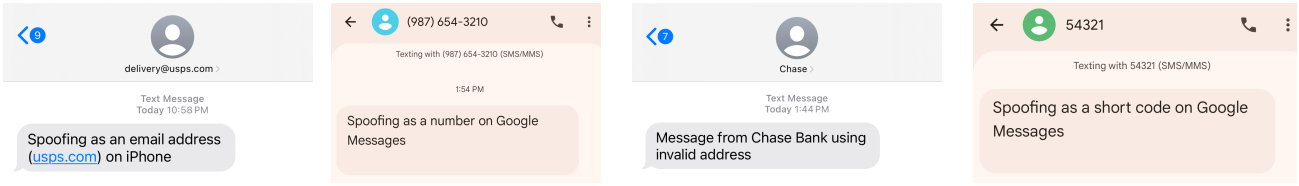
Results. All three applications we tested (Apple Messages, Google Messages, and Samsung Messages) only indicate when a protocol change occurs (e.g., RCS to SMS, or iMessage to SMS) by displaying small labels "Text Message * SMS" or "Texting with Jane (SMS/MMS)" as shown in Figure 4. However, the apps only display this indication for the first message of a protocol change, and a quick succession of spoofed messages from the attacker can quickly scroll such an indicator off the screen.

For different contact methods, Apple Messages exhibits the most aggressive merging behavior: it uses a single conversation thread that interleaves messages sent from all phone numbers and email addresses associated with a contact, regardless of protocol. In addition, Apple Messages does not indicate to the user the contact method and channel a message was received from (e.g., email vs. phone number).

This merging strategy, while convenient for users, makes spoofing attacks more powerful on the iPhone. An attacker need only spoof a contact's email address, and Apple Messages will interleave the spoofed message with an ongoing thread with the contact's phone number. In contrast, Google Messages and Samsung Messages on Android use a separate thread for every contact method, even if they belong to the same contact.

6. End-to-End Attacks

The previous sections focused separately on vulnerabilities in the email gateways and phone messaging stacks. In



(a) Email Address on iPhone (b) Phone Number on Android (c) Text on iPhone (d) Short Code on Android

Figure 5: Spoofed sending attacks on Apple Messages and Google Messages.

this section, we explore how an attacker can exploit these vulnerabilities end-to-end in real-world scenarios. We start with generic spoofing attacks that only require knowledge of the victim's phone number and corresponding gateway (Section 6.1). We then progress to more sophisticated attacks, where the attacker utilizes special treatment of certain sender identities (Section 6.2) or knowledge of the victim's contacts (Section 6.3) to increase the effectiveness of the attacks. Finally, we discuss how sender spoofing can be combined with phishing content to deceive victims (Section 6.4).

For each attack, we sent an email message with a spoofed sender identity through a carrier gateway (Section 4.3), triggering vulnerabilities in the phone's message parsing (Section 5.1), and observed how the default messaging app renders the message to evaluate success.

6.1. Generic Spoofing

The simplest set of attacks that an attacker can perform is spoofing as a sender of the attacker's choice, such as an email address, phone number, short code, or text. This category of attacks only requires knowledge of the victim's phone number and corresponding carrier gateway.

Email Address. Figure 5(a) shows an example attack spoofing as an email address (`delivery@usps.com`) on iPhone. By exploiting vulnerabilities in Section 4.3.2, an attacker can spoof as an arbitrary email address for all but one carrier gateway (AT&T MMS). This spoofed address will render as is on both Android and iPhone.

Phone Number, Short Code, or Text. Figures 5(b), 5(c), and 5(d) show examples of attacks spoofing as a phone number (+1 987-654-3210), text (Chase), and short code (54321) on iPhone and Android. The vulnerabilities in Section 5.1 collectively allow an attacker to spoof as an arbitrary phone number or short code for all carrier gateways on iPhone; arbitrary text for all but one carrier gateway (AT&T MMS) on iPhone; lowercase text for the AT&T MMS gateway on iPhone; and an arbitrary phone number and short code for all but one carrier gateway (AT&T MMS) on Android. Table 7 in Appendix C.3 details which vulnerabilities an attacker can exploit through each of the carrier gateways.

6.2. Trusted Brands Spoofing

Next, we describe spoofing attacks that take advantage of special treatment of high-profile sender identities by messaging apps. Such special treatment is designed to ensure trustworthiness of messages from important organizations. Ironically, an attacker can exploit such special treatment to boost the effectiveness of their spoofing attacks. We identify two types of special treatment: (1) Verified RCS Business Messages (RBM) from organizations on iPhone; and (2) Chat with Gemini on Android.

Verified RBMs on iPhone. Verified RCS Business Messages (RBM) builds on top of Rich Communication Services (RCS), a protocol supporting rich media and interactive communication (e.g., to notify when a package is delivered) [24]. Starting with iOS 18, Apple integrated RCS into Apple Messages and the support for RBM followed closely after. When the message app receives an RBM message, it renders the sender identity as coming from an organization, and not from a short code, an email address, or a phone number. Figure 6(a) shows an example RBM from Microsoft, where the app displays the Microsoft logo at the top. If the user taps on the sender, Apple Messages displays contact information for the organization together with a verified badge (Figure 6(b)).

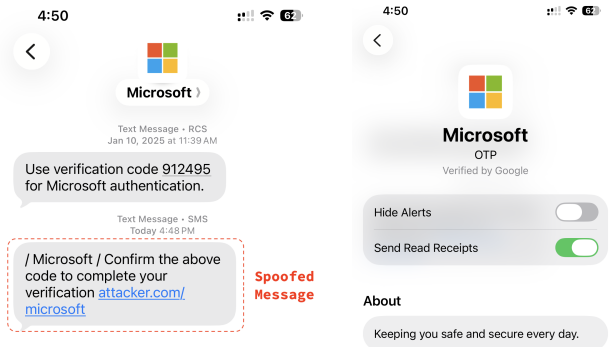
Apple Messages treats a message as an RBM if the sender identity is an email address of a specific format (e.g., `microsoft_abcdef@rbm.goog`[13] in Figure 6(a)), regardless of whether it is sent over RCS, SMS, or MMS. It uses such addresses to look up the organization's branding information (e.g., logo). However, Apple Messages never validates that the message was actually sent over RCS—it relies solely on the format of the email address.[14]

An attacker can therefore spoof an RBM by simply spoofing the sender identity as an email address of the specific format. Such email addresses are straightforward to obtain from a message dump of an iPhone that includes legitimate RBMs from the organization. This vulnerability undermines a key goal of RBM: preventing anyone but the organization from sending messages under its identity.

Chat with Gemini on Android. On Android, users can chat with Gemini in the Google Messages app. Similar to RBM on iPhone, Google Messages treats a message as being from Gemini if the sender identity is a special phone number (`833-991-3448`), regardless of whether the message is sent over RCS, SMS, or MMS. Hence, on

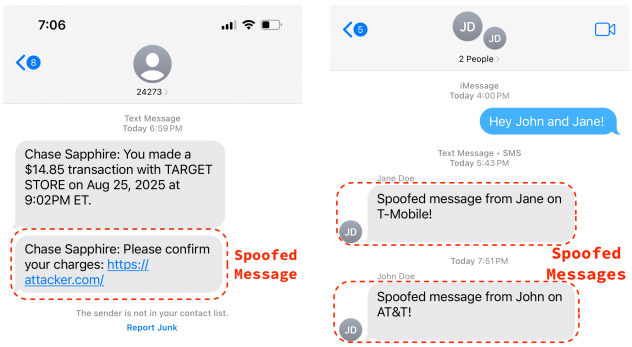
13. The domain `rbm.goog` has a neutral SPF policy and a DMARC policy of *None*, making it trivial to spoof.

14. The attack requires no prior relationship with the business to spoof an RCS Business Message in Apple Messages.



(a) Conversation (b) Contact Info

Figure 6: Spoofed RBM injected into Microsoft chat.



(a) Chase Bank (b) Group Conversation

Figure 7: Injecting into existing conversations: spoofed messages in the Chase Bank short code thread (24273) on iPhone, and in an Apple Messages group where the adversary impersonates both participants in an ongoing thread.

Android an attacker can spoof messages from Gemini by simply spoofing the number.

6.3. Targeted Spoofing

Attackers can mount more targeted spoofing attacks by using knowledge specific to the victim, such as information about the victim's contacts or existing conversations. We document two types of such targeted spoofing attacks. The first type of attack injects spoofed messages into existing conversations that the victim already has. The second type allows an attacker to spoof multiple participants in a group conversation that the victim is a part of.

Injecting as a Short Code. Figure 7(a) shows an attacker injecting a spoofed message into an ongoing conversation with Chase bank's short code (24273) on iPhone. This attack represents cases where the conversation is with a sender identity that is not in the victim's contact book. In such cases, the attacker needs to spoof as the exact sender identity to inject a message on both iPhone and Android.

Injecting as a Saved Contact. As discussed in Section 5.3, Google Messages merges only messages sent from phone numbers into a single conversation thread. Consequently, to inject a message into an existing thread with a saved contact on Android, the attacker must spoof that contact's phone number. Apple Messages, on the other hand, merges messages from multiple contact methods—including email, iMessage, SMS, and MMS—into the same conversation thread (Figure 4(a)). As a result, on iPhone an attacker can inject a message into an existing thread by spoofing any of the contact methods associated with the target.

Group Spoofing. When an attacker has knowledge of multiple contacts saved on a victim's device, they can construct a fake group conversation involving the victim and those contacts. The attacker can then assume the identity of any participant, giving the impression of a legitimate multi-party chat. Figure 7(b) shows an example of such group spoofing on Apple Messages, where all participants are contacts in a victim's contact book. Under such a setting, the attacker has two additional abilities: (1) they can iteratively control all sides of a group conversation by cycling through the headers, and (2) they can inject a message into an ongoing group conversation without having the message delivered to the other members of the group. In short, the attacker can effectively spoof as all parties involved in a group conversation, for both Android and iPhone.

Such attacks are possible because both Verizon MMS and AT&T MMS allow an email message to specify multiple recipients using additional *To* headers in an email message, with no apparent verification of the additional addresses. In both Google Messages and Apple Messages, an email message with multiple recipients results in the message being displayed as a group message. In both cases, the gateway delivers the message to the target (the first address in the *To* header), and the additional recipients are parsed as phone numbers and participants in a group conversation.

6.4. Phishing Content

Spoofing the sender identity is often only one part of a broader attack that aims at prompting the victim to take some action that benefits the attacker. In this section, we examine how attackers can utilize sender spoofing with carefully crafted phishing content to achieve higher-level malicious objectives. We discuss three types of phishing content that synergize well with sender spoofing, presented in order of increasing sophistication: content that advertises scams, content that employs app redirection, and content that facilitates conversation hijacking.

Advertising Scams. Particularly for the carrier gateways supporting MMS, it is straightforward for attackers to spoof messages to targets as yet another method for advertising any of a wide variety of common scams. However, unlike previous message abuse on phones, by spoofing scam messages an attacker can take advantage of trust relationships associated with the spoofed sender identity to increase the chance that a target engages.

Using standard MIME headers and base64-encoded attachments, we systematically documented the kinds of media the MMS carrier gateways support (Appendix C.4), as well as the display behavior of the attachments in the phone messaging apps. All carriers have gateways that deliver MMS messages containing URLs or media attachments, and the messaging apps on the phones make it very convenient for users to tap on the URLs to preview and start to engage with the scam an attacker is advertising.

Redirecting to Other Apps. In addition to traditional MIME content, an attacker can further take advantage of URL behavior specific to phones. In particular, an attacker can place URLs in spoofed email messages that redirect users to other apps on the phone. For instance, an attacker can use a URL with a phone number prepended by `tel:` in a spoofed message, and a user who taps on it will open the default phone application. Similarly, if a user taps on a URL to a WhatsApp invite `https://wa.me/19876543210`, the WhatsApp app will open and prompt the user to respond to that number, as shown in Figure 8(a).

Hijacking Existing Conversations. One major limitation of sender spoofing attacks is that the attacker cannot receive replies from the victim, as replies go to the legitimate sender being impersonated. One way to overcome this limitation is injecting the attacker's contact details onto the victim's device, enabling them to receive replies. To achieve this goal, an attacker can include a Virtual Contact File (VCF) in their message. A VCF is a MIME attachment supported by all MMS gateways and contains contact information.

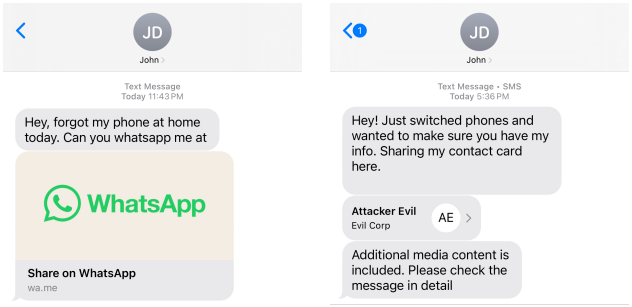
Figure 8(b) shows an example of a VCF attachment injected into an ongoing conversation on T-Mobile's MMS gateway.[15] When received, both Android and iPhone messaging apps present the VCF as a contact card. If the victim imports information from this card, their phone's address book is updated to include the attacker's contact, enabling the attacker to receive replies. Moreover, if an attacker does not know of a contact on a target's phone, the ability to send VCF creates an opportunity for an attacker to bootstrap one.

7. Discussion

In general, the problems highlighted in this paper are the product of semantic gaps that arises when composing disparate kinds of communication services. Absent clear end-to-end origin integrity, such compositions founder around the ambiguities of translating the addressing semantics from one domain into another (i.e., from email to IMS).

In Appendix A, we discuss our ethical considerations and coordinated disclosure to affected stakeholders. In what follows, we discuss short-term mitigations and longer-term fixes for the range of deficiencies identified in this paper; all require changes by an array of stakeholders (sadly, there is no silver bullet here).

15. A MIME attachment can cause the gateway to insert a short notice into the delivered message (e.g., "Additional media content is included..." in Figure 8(b)).



(a) WhatsApp Redirect (b) VCF Injection

Figure 8: Hijacking a conversation via VCF or WhatsApp redirect, causing the victim's replies to a spoofed identity to reach the attacker.

Perhaps the simplest approach is for each carrier to discontinue email to messaging gateways (as AT&T has recently done [5]). Indeed, it is unclear if consumer use of such services is still widespread and there are multiple alternatives for automating commercial messaging use (e.g., Twilio). However, while shutting down email gateways would eliminate the email spoofing vector we have surfaced, it would not address ambiguities in *other* kinds of messaging gateways — which may have similar vulnerabilities. The next most restrictive approach would be to configure gateways to require explicit SMTP Authentication [60] and then filter outbound mail to only allow sender information consistent with the authenticated user's credentials. However, this would also foreclose using such gateways via existing MTAs (e.g., such as gmail). A yet more permissive approach would be for gateways to require a validated *Mail From* domain (e.g., via SPF) and to *only* use the *Mail From* field in constructing a message (i.e., eliminate all other "fallback" behavior). This approach effectively delegates responsibility to the sending MTA, both to correctly configure SPF and appropriate authenticate their users. An even more open approach is simply "wrap" the address, subject and body in a message originating from a reserved phone number associated with the gateway (as AT&T did with SMS). This would make the origin of such messages clear, but would also impact user experience as legitimate messages would no longer be recognized as being associated with an existing contact. Lastly, all such gateways should also further tighten message parsing to limit the allowable syntax (and, thus, opportunities for confusing app parsers). However, there is no clear consensus on what that syntax ought to be, and additional study may be required.

On the handset side, app parsers should similarly be tightened to avoid ambiguities that "short circuit" display. As well, app developers may want to reconsider using mild indicators when the mode of transport has changed (i.e., the small type SMS/MMS indicator text) as we suspect few users understand its security importance. Indeed, at very least for modes of communication that have existing end-to-end security guarantees (e.g., RCS, iMessage), we

would recommend that intervening messages delivered via an insecure transport should either create a new thread and/or be highlighted as a potential security concern.

In addition, because spoofed text messages are often part of larger attacks (e.g., phishing), existing defenses, such as warning [57], user training [41], SMS trace analysis [61], and out-of-band verification [51], should account for spoofed messages in their threat models.

Finally, the ideal solution to this situation would be to standardize on messaging transports that provide end-to-end integrity and abandon others. However, we recognize that this is more of an economic and logistical challenge than a technical one, and thus may be slow to deploy.

Acknowledgments

We thank the anonymous reviewers and our shepherd for their helpful feedback. We are particularly grateful to Weihaw Chuang for facilitating the disclosure process with Google and for his feedback on the paper. Many thanks also to Cindy Moore and Jennifer Folkestad for operational and administrative support of our research as part of UCSD's Center for Networked Systems. We are also indebted to Katherine Izhikevich, Paul Chung, Seoyoung Kweon, Lisa Huang, and Jessie Johnston (via Mia Minnes) for volunteering their time and devices for testing. Funding for this work was provided in part by the Irwin Mark and Joan Klein Jacobs Chair in Information and Computer Science, the CSE Professorship in Internet Privacy and/or Internet Data Security, and the Paul Jacobs Chancellor's Endowed Faculty Fellowship for Next Generation Wireless. Enze Liu was supported by a Google Academic Research Award.

References

[1] Akaki. Analysis and Reproduction of Spoofed SMS-DELIVER — Akaki I/O. https://akaki.io/2022/analysis_and_reproduction_of_spoofed_sms-deliver, March 2022. [Online; accessed 2025-11-08].

[2] Android Open Source Project. Src/com/android/messaging/receiver/SmsReceiver.java — platform/packages/apps/Messaging — Git at Google. https://android.googlesource.com/platform/packages/apps/Messaging/+/de315b762312dd1a5d2bbd16e62ef2bd123f61e5/src/com/android/messaging/receiver/SmsReceiver.java, February 2025.

[3] Android Open Source Project. Src/java/com/android/internal/telephony/InboundSmsHandler.java — platform/frameworks/opt/telephony — Git at Google. https://android.googlesource.com/platform/frameworks/opt/telephony/+/307806916047d68377ff74674f95ca499243e068/src/java/com/android/internal/telephony/InboundSmsHandler.java, March 2025.

[4] Md Ishtiaq Ashiq, Weitong Li, Tobias Fiebig, and Taejoong Chung. SPF Beyond the Standard: Management and Operational Challenges in Practice and Practical Recommendations. In *Proceedings of the 33rd USENIX Security Symposium*, pages 3081–3098, 2024.

[5] AT&T. Say goodbye to email-to-text and text-to-email. https://www.att.com/support/article/wireless/KM1061254/, June 2025. [Online; accessed 2025-11-08].

[6] Nathaniel Bennett, Rebekah Sowards, and Casey Deccio. SPFail: Discovering, Measuring, and Remediating Vulnerabilities in Email Sender Validation. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, pages 633–646, 2022.

[7] Evangelos Bitsikas and Christina Pöpper. You have been warned: Abusing 5G's Warning and Emergency Systems. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pages 561–575, 2022.

[8] Blacktop. ipsw. https://github.com/blacktop/ipsw, November 2025.

[9] Birk Blechschmidt and Ben Stock. Extended Hell(o): A Comprehensive Large-Scale Study on Email Confidentiality and Integrity Mechanisms in the Wild. In *Proceedings of the 32nd USENIX Security Symposium*, pages 4895–4912, 2023.

[10] Sai Prashanth Chandramouli, Pierre-Marie Bajan, Christopher Kruegel, Giovanni Vigna, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. Measuring E-Mail Header Injections on the World Wide Web. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC)*, pages 1647–1656, 2018.

[11] Jianjun Chen, Vern Paxson, and Jian Jiang. Composition Kills: A Case Study of Email Sender Authentication. In *Proceedings of the 29th USENIX Security Symposium*, pages 2183–2199, 2020.

[12] Stefan Czybik, Micha Horlboge, and Konrad Rieck. Lazy Gatekeepers: a Large-scale Study on SPF Configuration in the Wild. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, pages 344–355, 2023.

[13] Casey Deccio, Tarun Yadav, Nathaniel Bennett, Alden Hilton, Michael Howe, Tanner Norton, Jacob Rohde, Eunice Tan, and Bradley Taylor. Measuring Email Sender Validation in the Wild. In *Proceedings of the ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pages 230–242, 2021.

[14] Haotian Deng, Weicheng Wang, and Chunyi Peng. Ceive: Combating Caller ID Spoofing on 4G Mobile Phones Via Callee-Only Inference and Verification. In *Proceedings of the ACM Conference on Mobile Computing and Networking (MobiCom)*, pages 369–384, 2018.

[15] Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, and J Alex Halderman. Neither Snow Nor Rain Nor MITM... An Empirical Analysis of Email Delivery Security. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, pages 27–39, 2015.

[16] European Telecommunications Standards Institute. 3GPP TS 23.140: Multimedia Messaging Service (MMS); Functional Description; Stage 2. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=794, March 2009.

[17] European Telecommunications Standards Institute. 3GPP TS 23.040: Technical Realization of the Short Message Service (SMS). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=747, April 2024.

[18] Federal Communications Commission. 2024 Communications Marketplace Report. FCC 24-136, December 2024. https://www.fcc.gov/document/fcc-releases-2024-communications-marketplace-report.

[19] Martin Fitzpatrick. Email to SMS - Send Free SMS via Email. https://email2sms.info/, 2017.

[20] Ian D Foster, Jon Larson, Max Masich, Alex C Snoeren, Stefan Savage, and Kirill Levchenko. Security by Any Other Name: On the Effectiveness of Provider Based Email Security. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 450–464, 2015.

[21] FreeCarrierLookup. Free carrier lookup service. https://freecarrierlookup.com/, August 2025.

[22] Randall Gellens. Mapping Between the Multimedia Messaging Service (MMS) and Internet Mail. RFC 4356, January 2006.

[23] Google. Google Messages — Apps on Google Play. https://play.google.com/store/apps/details?id=com.google.android.apps.messaging&hl=en_US, August 2025.

[24] Google. How RCS Business Messaging works. https://developers.google.com/business-communications/rcs-business-messaging/guides/get-started/how-it-works, August 2025.

[25] GSM Association. Cybersecurity Document Library. https://www.gsma.com/solutions-and-impact/technologies/security/cybersecurity-knowledge-base/cybersecurity-document-library/, March 2026.

[26] Hang Hu, Peng Peng, and Gang Wang. Towards Understanding the Adoption of Anti-Spoofing Protocols in Email Systems. In *Proceedings of the IEEE Cybersecurity Development Conference (SecDev)*, pages 94–101, 2018.

[27] Hang Hu and Gang Wang. End-to-End Measurements of Email Spoofing Attacks. In *Proceedings of the 27th USENIX Security Symposium*, pages 1095–1112, 2018.

[28] Olivier Hureau, Jan Bayer, Andrzej Duda, and Maciej Korczyński. Spoofed Emails: An Analysis of the Issues Hindering a Larger Deployment of DMARC. In *Proceedings of the International Conference on Passive and Active Network Measurement (PAM)*, pages 232–261, 2024.

[29] Hongil Kim, Jiho Lee, Eunkyu Lee, and Yongdae Kim. Touching the Untouchables: Dynamic Security Analysis of the LTE Control Plane. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pages 1153–1168, 2019.

[30] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS)*, 2019.

[31] Gyuhong Lee, Jihoon Lee, Jinsung Lee, Youngbin Im, Max Hollingsworth, Eric Wustrow, Dirk Grunwald, and Sangtae Ha. This is Your President Speaking: Spoofing Alerts in 4G LTE Networks. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services (Mobisys)*, pages 404–416, 2019.

[32] Enze Liu, Gautam Akiwate, Mattijs Jonker, Ariana Mirian, Grant Ho, Geoffrey M Voelker, and Stefan Savage. Forward Pass: On the Security Implications of Email Forwarding Mechanism and Policy. In *Proceedings of the 8th IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 373–391. IEEE, 2023.

[33] Jinrui Ma, Lutong Chen, Kaiping Xue, Bo Luo, Xuanbo Huang, Mingrui Ai, Huanjie Zhang, David SL Wei, and Yan Zhuang. FakeBehalf: Imperceptible Email Spoofing Attacks against the Delegation Mechanism in Email Systems. In *Proceedings of the 33rd USENIX Security Symposium*, pages 1243–1260, 2024.

[34] Scott Main and David Braun. Getting Your SMS Apps Ready for KitKat. https://android-developers.googleblog.com/2013/10/getting-your-sms-apps-ready-for-kitkat.html, October 2013.

[35] Hossen Mustafa, Wenyuan Xu, Ahmad-Reza Sadeghi, and Steffen Schulz. End-to-End Detection of Caller ID Spoofing Attacks. *IEEE Transactions on Dependable and Secure Computing*, 15(3):423–436, June 2016.

[36] North American Numbering Plan Administration. NPA Reports. https://www.nanpa.com/reports/npa-reports, August 2025.

[37] Open Mobile Alliance. Multimedia Messaging Service Encapsulation Protocol. https://www.openmobilealliance.org/release/MMS/V1_3-20110913-A/OMA-TS-MMS_ENC-V1_3-20110913-A.pdf, September 2011.

[38] Pete Resnick. Internet Message Format. RFC 5322, October 2008. https://www.rfc-editor.org/info/rfc5322.

[39] Merve Sahin, Aurélien Francillon, Payas Gupta, and Mustaque Ahamad. SoK: Fraud in Telephony Networks. In *Proceedings of the 2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 235–250. IEEE, 2017.

[40] Kaiwen Shen, Chuhan Wang, Minglei Guo, Xiaofeng Zheng, Chaoyi Lu, Baojun Liu, Yuxuan Zhao, Shuang Hao, Haixin Duan, Qingfeng Pan, and Min Yang. Weak Links in Authentication Chains: A Large-Scale Analysis of Email Sender Spoofing Attacks. In *Proceedings of the 30th USENIX Security Symposium*, pages 3201–3217, 2021.

[41] Steve Sheng, Bryant Magnien, Ponnurangam Kumaraguru, Alessandro Acquisti, Lorrie Faith Cranor, Jason Hong, and Elizabeth Nunge. Anti-Phishing Phil: The Design and Evaluation of a Game That Teaches People Not to Fall for Phish. In *Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS)*, page 88–99, 2007.

[42] Jingwen Shi, Sihan Wang, Min-Yue Chen, Guan-Hua Tu, Tian Xie, Man-Hsin Chen, Yiwen Hu, Chi-Yu Li, and Chunyi Peng. IMS is Not That Secure on Your 5G/4G Phones. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 513–527, 2024.

[43] Statcounter. Mobile Vendor Market Share United States Of America. [Online; accessed 2025-11-08].

[44] T-Mobile USA. T-Mobile USA and MetroPCS To Combine, Creating Value Leader In U.S. Wireless Marketplace. http://www.t-mobile.com/news/press/t-mobile-usa-and-metropcs-to-combine-creating-value-leader-in-us, October 2012.

[45] Dennis Tatang, Florian Zettl, and Thorsten Holz. The Evolution of DNS-based Email Authentication: Measuring Adoption and Finding Flaws. In *Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, pages 354–369, 2021.

[46] Akaki Tsunoda. Demonstrating Spoofability of an Originating Number when sending an SMS using SMPP. *Digital Threats: Research and Practice*, 5(1):1–13, 2024.

[47] Akaki Tsunoda. Investigating Threats Posed by SMS Origin Spoofing to IoT Devices. *Digital Threats: Research and Practice*, 5(4):1–12, 2024.

[48] Guan-Hua Tu, Chi-Yu Li, Chunyi Peng, Yuanjie Li, and Songwu Lu. New Security Threats Caused by IMS-based SMS Service in 4G LTE Networks. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 1118–1130, 2016.

[49] The Verge. Google Messages on Samsung S21 Ultra One UI Redesign. https://www.theverge.com/2021/4/28/22407361/google-messages-samsung-s21-ultra-one-ui-redesign, April 2021.

[50] Verizon. Ensure Picture / Video Message does not Exceed File Size Limit. https://www.verizon.com/support/knowledge-base-14641/, August 2025.

[51] Chenkai Wang, Zhuofan Jia, Hadjer Benkraouda, Cody Zevnik, Nicholas Heuermann, Roopa Foulger, Jonathan A. Handler, and Gang Wang. VeriSMS: A Message Verification System for Inclusive Patient Outreach against Phishing Attacks. In *In Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI)*, 2024.

[52] Chenkai Wang and Gang Wang. Revisiting Email Forwarding Security under the Authenticated Received Chain Protocol. In *Proceedings of the ACM Web Conference (WWW)*, pages 681–689, 2022.

[53] Chuhan Wang, Yasuhiro Kuranaga, Yihang Wang, Mingming Zhang, Linkai Zheng, Xiang Li, Jianjun Chen, Haixin Duan, Yanzhong Lin, and Qingfeng Pan. BreakSPF: How Shared Infrastructures Magnify SPF Vulnerabilities Across the Internet. In *Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS)*, 2024.

[54] Chuhan Wang, Kaiwen Shen, Minglei Guo, Yuxuan Zhao, Mingming Zhang, Jianjun Chen, Baojun Liu, Xiaofeng Zheng, Haixin Duan, Yanzhong Lin, and Qingfeng Pan. A Large-scale and Longitudinal Measurement Study of DKIM Deployment. In *Proceedings of the 31st USENIX Security Symposium*, pages 1185–1201, 2022.

[55] Chuhan Wang, Chenkai Wang, Songyi Yang, Sophia Liu, Jianjun Chen, Haixin Duan, and Gang Wang. Email Spoofing with SMTP Smuggling: How the Shared Email Infrastructures Magnify this Vulnerability. In *Proceedings of the 34th USENIX Security Symposium*, 2025.

[56] Qi Wang, Jianjun Chen, Jingcheng Yang, Jiahe Zhang, Yaru Yang, and Haixin Duan. SIPConfusion: Exploiting SIP Semantic Ambiguities for Caller ID and SMS Spoofing. In *Proceedings of the 33rd Annual Network and Distributed System Security Symposium (NDSS)*, 2026.

[57] Yizhu Wang, Haoyu Zhai, Chenkai Wang, Qingying Hao, Nick A Cohen, Roopa Foulger, Jonathan A Handler, and Gang Wang. Can You Walk Me Through It? Explainable SMS Phishing Detection using LLM-based Agents. In *Twenty-First Symposium on Usable Privacy and Security (SOUPS)*, pages 37–56, 2025.

[58] The Apple Wiki. CoreTelephony. https://theapplewiki.com/wiki/CoreTelephony, December 2024.

[59] Wikipedia. SMS Gateway. *Wikipedia*, October 2025. https://en.wikipedia.org/wiki/SMS_gateway.

[60] Wikipedia. SMTP authentication. *Wikipedia*, March 2026. https://en.wikipedia.org/wiki/SMTP_Authentication.

[61] Guanhua Yan, Stephan Eidenbenz, and Emanuele Galli. SMS-Watchdog: Profiling Social Behaviors of SMS Users for Anomaly Detection. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection (RAID)*, pages 202–223, 2009.

[62] N Yomna. Gotta Catch'Em All: Understanding How IMSI-Catchers Exploit Cell Networks. *Electronic Frontier Foundation: San Francisco, CA, USA*, 2019.

[63] Yiming Zhang, Baojun Liu, Chaoyi Lu, Zhou Li, Haixin Duan, Shuang Hao, Mingxuan Liu, Ying Liu, Dong Wang, and Qiang Li. Lies in the Air: Characterizing Fake-base-station Spam Ecosystem in China. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 521–534, 2020.

[64] Jinghao Zhao, Qianru Li, Zengwen Yuan, Zhehui Zhang, and Songwu Lu. 5G Messaging: System Insecurity and Defenses. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, pages 37–45, 2022.

Appendix A. Ethics Considerations and Disclosure

There are two classes of ethical questions that result from a study such as ours. The first is the extent to which the experiments themselves implicated the interests of any stakeholders whose services or equipment we used. To that end, our experiments involve sending emails to gateways that are explicitly and publicly advertised as being available for that purpose. Moreover, we send messages destined for devices that we own (or have permission to use for this purpose), and which operate with SIMs associated with either the particular carrier or MVNO operating the email gateway, or a carrier that an MVNO depends on for service. To wit, while the content of our messages is atypical, we are otherwise using them for the purpose that they were designed for, with customers they were designed for reaching, using handset equipment under our control. Moreover, we were careful to rate limit our experiments to not impose undue load on any individual service or provider.

The second issue is one of disclosure and whether there was sufficient opportunity for vulnerable stakeholders to mitigate these issues prior to any publication of results (and hence potential secondary harms). We disclosed to all vulnerable carriers discussed in this paper in advance of submission. Each responded promptly and validated the significance of the issue, and, as of this writing, all four carriers had either deployed an explicit fix in their gateways or discontinued the operation of their email to text gateways altogether (AT&T [5]).[16] At the request of the reviewers, we have *also* disclosed to the GSM Association (GSMA) on February 17th, 2026, who will be able to appropriately distribute this information to carriers in other countries who may have similar or related flaws. On March 30th, 2026, GSMA's panel determined to update their FS.30 Security Manual [25] that details security guidelines for network operators, with details of our attacks and mitigations on how to tighten email to text gateway policies.

On the handset side, we disclosed the sender parsing vulnerabilities to Google Messages on August 22nd, 2025, and to Apple on September 3rd, 2025. Google confirmed the vulnerability and released a fix in Google Messages version 20251009_00_RC00, publicly available as of October 21st, 2025. Apple confirmed the sequence injection vulnerability on October 22nd, 2025, subsequently assigned CVE-2025-46311 to it, and addressed it in iOS/macOS/iPadOS 26.3. Apple was unable to reproduce the space injection attack in their testing and made no code changes. Apple determined the interleaving behavior to be correct by design, with no security implications, but indicated they may make enhancements in the future. Finally, an Apple fix for the RCS Business Messaging spoofing vulnerability is expected in a subsequent release. Taken together, we believe remaining risks associated with publication are minimal, especially when weighed against the benefits of improving our understanding of these kinds of composition risks (which are likely manifest in a range of enterprise systems employing similar kinds of functionality).

Appendix B. Extending beyond U.S. Carriers

Reviewers requested that we assess whether the vulnerabilities we identified extend beyond U.S. carriers. While it is beyond the scope of this paper (and our operational ability) to exhaustively evaluate all global carriers, we extended our analysis to explore nation-wide mobile carriers in Mexico and Canada that offer email to text gateways.

Mexico. We tested all Mexican carriers with national coverage that had documented email to text gateways. Of these, only Telcel's gateway (`itelcel.com`, whose MX record resolves to `mx1.itelcel.com` and points to an IP in Sixsigma Networks Mexico) accepted our connections. However, all messages sent to this server were returned with bounce messages reporting `550 Recipient IS NOT OK`, and we were unable to confirm whether the gateway was operational for legitimate use.

Canada. We similarly tested all national Canadian carriers with documented email to text gateways, and found only Telus Mobility to have a working gateway. Telus operates two gateway domains (`msg.telus.com` for SMS, and `mms.telusmobility.com` for MMS) both of which resolve MX to `msgspamblk.glbpr.telus.com`, which points to IP addresses operated by Telus. Via a contact in Canada, we tested the Telus MMS gateway[17]

16. Google Fi classified the vulnerability at its high severity rating and awarded us a bug bounty, T-mobile deployed a fix within 24 hours of disclosure and Verizon validated the bug in 24 hours and had deployed a patch within five days.

17. We tested on Koodo Mobile, which is a Telus MVNO that uses the Telus MMS gateway.

Carriers	SMS+MMS Gateway Domains
T-Mobile Mint Mobile Assurance Wireless US Mobile (LS) Red Pocket (GSMT)	tmomail.net (both SMS & MMS)
Google Fi	msg.fi.google.com (MMS only)
Verizon	vtext.com + vzwpix.com
Tracfone Straight Talk Simple Mobile Spectrum Mobile Xfinity Mobile Visible US Mobile (Warp) Red Pocket (CDMA)	vtext.com + mypixmessages.com
AT&T Consumer Cellular US Mobile (Dark Star) Red Pocket (GSMA) h2o Wireless	txt.att.net + mms.att.net

TABLE 6: The email to text gateways we identified for twenty popular wireless carriers in the U.S. We group the carriers by the gateways that they share.

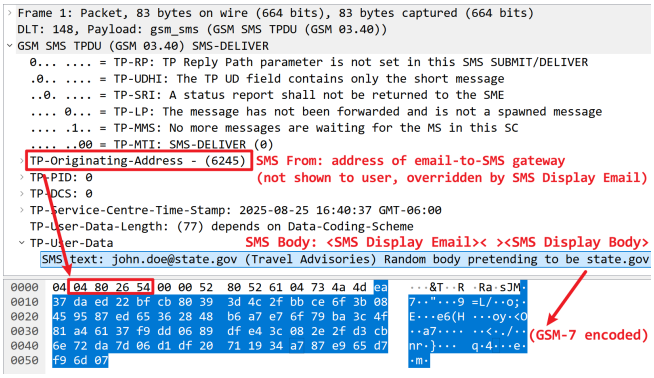


Figure 9: The raw SMS PDU byte stream corresponding to the parsed representations shown in Figure 2.

and found that it exhibits the same "Missing Header" vulnerability as Verizon MMS (Table 4): with *Mail From* empty and *From* set to a spoofed domain (e.g., `delivery@dhl.com`). Similar to T-Mobile, we observed popular domains (e.g., `usps.com`) being rejected with an SMTP error code `421 4.2.0 mail accepted for delivery AUP#BL`, suggesting the domain triggered a similar reputation-based filter as part of the carrier's anti-spam measures. For less popular domains (e.g., lower-ranked Tranco domains and our own domain, both with a DMARC policy of *Reject*), the gateway delivers the message with the spoofed sender identity. In the course of investigating this issue we identified a further parsing vulnerability in the Google Messages platform, which we disclosed to them. We also reported the gateway vulnerability to Telus Mobility but have not received a response despite multiple attempts to contact them.

Appendix C. Additional Material

C.1. Carrier Gateway Sharing

As discussed in Section 4.1, we identified the email to text gateways for twenty popular U.S. carriers. Table 6 shows these twenty carriers grouped by the gateways that they share. When carriers use separate gateway domains for SMS and MMS service, we include both domains.

C.2. Raw Data for SMS and MMS PDUs

To help reproduce our findings, we include the original binary stream of the SMS PDU shown in Figure 2 here. Figure 9 shows the raw bytes of an SMS PDU captured from Verizon, in which we use Wireshark's `gsm_sms` dissector to parse the SMS PDU. Note that although the SMS PDU contains an Originating Address field of `6245`, the phone display logic will override it to show the *SMS Display Sender* in the message body. The MMS PDU is similar, but shows the *From*, *Subject*, and *Data* (body) fields of the MMS message separately, where the *From* field shows the spoofed email address (`delivery@usps.com`).

C.3. Parsing Vulnerabilities per Carrier Gateway

Table 7 shows the details of the parsing vulnerabilities of the sender identity discussed in Section 6.1. The table annotates the vulnerabilities with the carrier gateway that they are specific to, and it documents the parsing behavior for alphanumeric sender identities (i.e., if they are type-cast to lowercase or retained in their original case).

C.4. MIME Attachments

While carriers impose a limit on the size of MMS messages [50], they are lenient on the type of media that can be sent. To infer which media types each carrier gateway allows, we enumerate attachments across a range of MIME types (Table 8). While most carriers allow all MIME types we tested, Google Fi strips PDF, ZIP, and DOCX attachments. Most carriers reject delivering riskier content, such as shell scripts or executable files.

C.5. Additional Conversion Details

Multiple Recipient Headers. The *To* header can have multiple addresses separated by commas. Some carrier gateways remove all addresses but the first when converting the email message to a PDU. However, both AT&T and Verizon retain all addresses and populate the PDU *To* header with them. The benign use case is that ultimately the messaging

Method	Phone/App	Spoof As	Carrier Gateway	Received As	Displayed As
Special Seq.	Apple	P	All	`9876543210=?@a.com`	+1 (987) 654-3210
		SC	All	`54321=?@a.com`	54321
		LCTXT	AT&T MMS, VZ-MMS	`lowerUPPER=?@a.com`	lowerupper
		TXT	GFi, TMO, VZ-SMS	`lowerUPPER=?@a.com`	lowerUPPER
Space	Apple	P	VZ-SMS	`"9876543210@a.com`	+1 (987) 654-3210
		SC	VZ-SMS	`"54321 @a.com`	54321
		TXT	VZ-SMS	`"lowerUPPER@a.com`	"lowerUPPER
Invalid Addr.	Apple	P	VZ-SMS	`"9876543210"`	+1 (987) 654-3210
		SC	VZ-SMS	`"54321"`	54321
		TXT	VZ-SMS	`lowerUPPER`	lowerUPPER
Numeric Addr.	Google Msgs	P	GFi, TMO, VZ	`987@6543.210`	+1 (987) 654-3210
		SC	GFi, TMO, VZ	`5@4.321`	54321

TABLE 7: Extended version of the parsing vulnerabilities in Table 5 annotated with the carrier gateway that they are specific to. P: phone number, SC: short code, TXT: arbitrary text, LCTXT: lowercase text, `a.com`: short for `attacker.com`.

MIME Type	VZM	TMO	ATT	GFI
Media				
PNG/JPEG	✓	✓	✓	✓
GIF	✓	✓	✓	✓
MP3	✓	✓	✓	✓
MP4	✓	✓	✓	✓
Contact & Docs				
VCF (vCard)	✓	✓	✓	✓
PDF	✓	✓	✓	S
ZIP Archive	✓	✓	✓	S
DOCX	✓	✓	✓	S
Executable Content				
Shell Script (.sh)	✘	✓	✓	✘
Executable (.exe)	✘	✘	✘	✘
Dynamic (.dll)	✘	✘	✘	✘
Web Content				
Links	✓	✓	✓	✓
HTML	✓	✓	✓	✓

TABLE 8: MIME type filtering and delivery inconsistencies across carrier messaging services. ✓: Delivered. ✘: Not Delivered. S: Delivered, but attachments stripped.

app on the phone can use the multiple addresses to associate the message with a group conversation thread.

However, the AT&T MMS and Verizon MMS gateways do not verify if the additional addresses are valid. As a result, an attacker can use arbitrary target addresses (e.g., spoofed email addresses or phone numbers) as additional addresses when crafting email messages to their gateways. Since the additional recipients are not verified, an attacker can include other phone numbers, belonging to other carriers, in the email *To* header. Section 6.3 describes combining this vulnerability with other UI rendering attacks to inject a message into an existing group conversation.

When converting email to IMS messages, by default carrier gateways will prepend the *Subject* header to the email body, often with simple formatting added. For example, many providers add a newline to separate the *Subject* from the body, T-Mobile SMS/MMS wraps it in forward slash tags, and Verizon SMS surrounds it with parentheses. To more stealthily inject a spoofed message into a conversation, an attacker will want to suppress the *Subject* header so that the email body appears like any other direct message.

Subject Suppression. To determine to what extent an attacker can suppress the *Subject* header, we tested sending an email message (1) without any *Subject* header, and (2) with five edge cases: using ASCII space, backspace and null byte characters, CRLF sequence, and Unicode NBSP character as the *Subject*. Table 9 shows how the carrier gateways convert the *Subject* header under these conditions. Sending a message without a *Subject* is the least attractive technique: all providers prepend a default string to the message body (e.g., T-Mobile prepends "`no subject`"). In many cases, a message without a *Subject* also leaks additional headers (e.g., *X-CMAE-Envelope*).

Scenario	VZ-MMS	VZ-SMS	TMO-SMS	ATT-MMS	GFI-MMS
Default	**Subj:S**↩b	(S) b	$/S/$ b	**S**↩b	**Subj:S**↩b
No Header	b*	b*	/no subj/ b*	b*	b
ASCII	b	b	/no subj/ b	b	b
Backspace	✘	b	$/$ $/$ b	b	**Subj:**
Null byte	b	b	/no subj/ b	b	**Subj:**
CRLF seq.	?	b	/no subj/ b	?	b
Unicode	?	(b)	$/?/$ b	?	**Subj:?**

TABLE 9: Suppressing the subject header. ↩: a newline between subject and body. b: body text. Subj: subject text. *: additional headers leaked. ✘: not delivered. ?: '?' character in subject.

However, with every carrier gateway except T-Mobile, an attacker can use one of several other syntactical techniques (e.g., a single ASCII space) to completely suppress *Subject*. In doing so, the gateway delivers the converted email message with just the email body as the message content. With T-Mobile, the most effective technique is a *Subject* header with a backspace character, in which case T-Mobile just prepends the string "`//`" to the body.

Appendix D. Meta-Review

The following meta-review was prepared by the program committee for the 2026 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

D.1. Summary

The paper identified two classes of vulnerabilities that enable end-to-end text message spoofing attacks that use email as the source. First, the authors describe several vulnerabilities that attackers can use against email to text message gateways operated by popular carriers, which allows email content to be converted and sent as text messages. Second, the authors describe bugs in implementations of messaging platforms such as iMessage and Google messages that hide the real source of attacks and enable impersonation. Experiments show that major MNOs in U.S. implement different email authentication mechanisms/policies, and they rely on different email headers for constructing the sender identity. On top of these, the messaging stacks of mainstream smartphone OSes are found to employ unsound parsing heuristics and display formats for handling various types of sender identities, which further enable the possibility of deceit. The paper then demonstrates how an attacker can take advantage of all these quirky behaviors to launch convincing and indiscernible SMS/MMS spoofing attacks at a fairly low cost. Google and Apple have already made plans to fix their implementations after responsible disclosure, but a complete mitigation will require carriers to tighten their implementations, and possibly intervention from standardization bodies.

D.2. Scientific Contributions

- Independent Confirmation of Important Results with Limited Prior Research.
- Identifies an Impactful Vulnerability.
- Provides a Valuable Step Forward in an Established Field.
- Establishes a New Research Direction.

D.3. Reasons for Acceptance

1) The paper identifies multiple impactful vulnerabilities to conduct realistic and powerful attacks through spoofed text messages.
2) The experiments were executed well, and the effort of reverse engineering the messaging stacks of smartphones to find implementation-level root causes is commendable.
3) The paper does a good job of building up the attack chain, and explains the steps through which the attack was tested.

D.4. Noteworthy Concerns

1) Multiple reviewers raised issues about the completeness and the generalizability of the identification of attack vectors and vulnerable networks is limited: the paper mostly uses ad-hoc methods in identifying vulnerabilities, and only tests major carriers within the United States.
2) The paper does not provide explicit details about the disclosure process, including efforts to inform carriers outside the US.