Composition Kills:
A Case Study of Email Sender Authentication

Jianjun Chen, Vern Paxson, Jian Jiang

ICSI  UC Berkeley  F5 (shape security)
How Do You Verify the Email Sender?

Your Single Transaction Alert from Chase

from: Chase <no.reply.alerts@chase.com>
to: whucjj@gmail.com
date: Jun 28, 2020, 8:04 PM
subject: Your Single Transaction Alert from Chase
mailed-by: chase.com
signed-by: chase.com
security: Standard encryption (TLS) Learn more

Important according to Google magic.
A Case of Our Spoofing Attacks on Gmail (Fixed)
Background:
Sender & Authentication
Background: Who’s the Sender?

SMTP envelope

HELO helo.sender.com
MAIL FROM: <s@mfrom.sender.com>
RCPT TO: <bob@email.com>

From: Secure Bank <noreply@bank.com>
To: Bob <bob@email.com>
Subject: Account Alert: Suspicious Purchase

Dear Bob,

We are writing to inform you that…
Background: SMTP Lacks Authentication

The original SMTP has no built-in authentication mechanism
• Anyone can spoof any identity in HELO/MAIL FROM and From

Alice

Mail User Agent

Sending Service (a.com)

Receiving Service (b.com)

IMAP POP

Bob

Mail User Agent

HELO a.com
MAIL FROM: <alice@a.com>
From: <alice@a.com>
To: <bob@b.com>
Three Sender-Authentication Protocols

- **Sender Policy Framework (SPF, RFC 7208)**
  - verifying the IP address of the sending domain

- **DomainKeys Identified Mail (DKIM, RFC 6376)**
  - verifying the email is signed by the sending domain

- **Domain Message Authentication, Reporting and Conformance (DMARC, RFC 7489)**
  - “how to” policy for recipient based on SPF and DKIM
  - “fix” the alignment problem of SPF and DKIM
Sender Policy Framework (SPF)

1. Publish authorized IP lists via DNS

2. Query the domain in HELO and MAIL FROM to obtain the IP lists

3. Check if the sender's IP matches the IP lists
   • If yes, SPF pass

From: Alice <alice@a.com>
To: Bob <bob@b.com>
Subject: Hello from Alice

Dear Bob,
I’m Alice…

1.2.3.4
a.com

a.com TXT 1.2.3.0/24

Alice

HELO a.com
MAIL FROM: <s@a.com>
RCPT TO: <bob@b.com>

Receiving Services

Sending Services

DNS

MUA

Bob

DNS

MUA

1.2.3.4

b.com

a.com TXT 1.2.3.0/24

1.2.3.4 matches 1.2.3.0/24
DomainKeys Identified Mail (DKIM)

1. Publish public key via DNS

2. Generate DKIM-Signature with private key and attach it to the message.

   **HELO** ehlo.a.com
   **MAIL FROM:** <s@smfrom.a.com>
   **RCPT TO:** <bob@b.com>

   **DKIM-Signature:** ...
   d=a.com;
   s=any;...
   From: Alice <alice@a.com>
   To: Bob <bob@b.com>

3. Query “s._domainkey.d” (any._domainkey.a.com) to obtain public key

4. Validate DKIM signature with the public key

---

**Diagram:**
- **MUA** (Sending Services)
  - a.com
- **DNS**
- **MUA** (Receiving Services)
  - b.com
What's Wrong with SPF/DKIM?

HELO ehlo.attack.com
MAIL FROM: <s@mfom.attack.com>
RCPT TO: <bob@b.com>

DKIM-Signature: ...;d=attack.com;
s=2020;...
From: Alice <alice@a.com>
To: Bob <bob@b.com>
Subject: Hello from Alice

Dear Bob,
I'm Alice...

What SPF verifies
What DKIM verifies
What the end-user sees

Neither SPF nor DKIM validate the From header that is displayed to the end-user.
Domain Message Authentication, Reporting and Conformance (DMARC)

3. Receiving services perform **identifier alignment test** to check if the domain in From header matches SPF or DKIM-verified domain.
   - Exactly match (strict) or have the same registered domain* (relaxed, default mode)

4. The email passes DMARC authentication if:
   1. either SPF or DKIM show a positive result, and
   2. the From header domain passes the alignment test.

* Defined in public suffix list, https://publicsuffix.org/

---

**Diagram:**

1. **MUA** publish policy via DNS
   - **Sending Services** a.com
   - **DNS**
   - **Receiving Services** b.com
   - **MUA**
   - **Bob**
   - Query the domain in From header

Alice
Overview of Email Authentication Flow

What could possibly go wrong?
Bypassing the Authentication
Inconsistencies between different components could lead to security vulnerabilities.
Inconsistencies between different components could lead to security vulnerabilities.
**Exp. 1: Inconsistencies b/w DKIM and DNS**

**Ambiguity: What DKIM uses differs from what DNS queries**

HELO attack.com
MAIL FROM: <any@attack.com>

DKIM-Signature: ...;d=bank.com;
    s=attack.com\.x00.any;...
From: <sec@bank.com>
To: <victim@victim.com>

Dear Customer,

We are writing to inform you that...

1. Attacker signs the message with his private key and sends the message

2. When receiving the message, DKIM should query ‘s._domainkey.d’ to obtain the public key: (attack.com\.x00.any._domainkey.bank.com)

3. But DNS takes \x00 as a terminator, and obtains public key from attack.com

4. DKIM **pass**, DMARC **pass**
Exp. 2a: Multiple From Headers

Ambiguity: What receiving server verifies differ from what MUA displays

- RFC 5322: Messages with multiple From should be rejected
- In practice: 19/29 accept (15 use first, 3 use last, 1 show both)
Exp. 2b: Multiple From Headers with Space

Three types of variants:
1) _From: a@a.com ; 2) From_: a@a.com; 3) From\n_ : a@a.com

From:
  : <any@attack.com>
From: <admin@bank.com>
To: <victim@victim.com>

Dear Customer,…

From:
  : <any@attack.com>
From: <admin@bank.com>
To: <victim@victim.com>

Dear Customer,…

Attacker
Server

Receiving
Services

Mail User
Agent

DMARC verifies attack.com

MUA display bank.com
Exp. 3: From Alternative Headers

- 7/19 MUAs display **Sender** or **Resent-From** header value when From header is absent

<table>
<thead>
<tr>
<th>From</th>
<th>Sender: <a href="mailto:admin@bank.com">admin@bank.com</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td><a href="mailto:any@attack.com">any@attack.com</a>&gt;</td>
</tr>
<tr>
<td>To:</td>
<td><a href="mailto:victim@victim.com">victim@victim.com</a>&gt;</td>
</tr>
<tr>
<td>Dear Customer,...</td>
<td></td>
</tr>
</tbody>
</table>

DMARC verifies attack.com  

MUA display bank.com
Email Parsing Process

Email Message

From: Secure Bank <admin@bank.com>
To: <victim@victim.com>

Dear Customer,…

From Header

From: Secure Bank <admin@bank.com>

Email Address

admin@bank.com
Complex From Header Syntax

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Comments</th>
<th>Route portion</th>
<th>Real address</th>
</tr>
</thead>
<tbody>
<tr>
<td>From: Secure (<a href="mailto:b@b.com">b@b.com</a>) Bank &lt;@c.com, @d.com:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="mailto:a@a.com">a@a.com</a> (<a href="mailto:e@e.com">e@e.com</a>) &gt; (<a href="mailto:f@f.com">f@f.com</a>)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A quick example of valid (!) From header

- **Multiple address lists.** [RFC 5322]
- **Encoding:** defined to support no-ascii character. [RFC 2047]
  - From: bob <b@b.com> is equal to
  - From: =?utf-8?B?Ym9i?= <b@b.com> in Base64 encoding
- **Quoted-pair:** use ‘\’ to escape special characters like ‘( ‘. [RFC 5322]
Exp. 4a: Exploiting Differences in Feature Support

From: <any@attack.com>, <admin@legitimate.com>

Mail server

Email client

From: <@attack.com, @any.com: admin@legitimate.com>

Mail server

Email client

From: bs64(<admin@legitimate.com>), <any@attack.com>

Email client

Mail server
Exp. 4b: Exploiting Parsing Inconsistencies

From: <admin@legitimate.com>, <any@attack.com>

Email client

Mail server

From: admin@legitimate.com, <any@attack.com>

Email client

Mail server

From: <any@attack.com> admin@legitimate.com

Mail server

Email client
### How Prevalent are UI-mismatch Vulnerabilities?

#### Table 2: Vulnerability of the tested email providers and MUAs to UI-mismatch attacks.

<table>
<thead>
<tr>
<th>Servers</th>
<th>MUAs</th>
<th>Web interface</th>
<th>Windows Mail</th>
<th>Windows Outlook</th>
<th>MacOS Mail</th>
<th>MacOS eM Client</th>
<th>Linux Thunderbird</th>
<th>Android Gmail</th>
<th>Android Outlook</th>
<th>iOS Mail</th>
<th>iOS Gmail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gmail.com</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>iCloud.com</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Outlook.com</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Yahoo.com</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Naver.com</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fastmail.com</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Zoho.com</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Tutanota.com</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Protonmail.com</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Mail.ru</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

“✓”: email server and MUA combination where we can expose an inconsistent interpretation.

“—”: email providers that don’t support third-party MUAs for our testing account.

- 43 out of 82 different combinations that could be exploited
- What we found only constitutes a subset of the problem
Exp. 5: Spoofing via an Email Service Account

**Ambiguity:** What sending server validates differ from what MUA displays

![Diagram showing the process of spoofing via an email service account]

- **Attackers with an email service account**
  - attacker@gmail.com tries to spoof admin@gmail.com

- **Sending services** should ensure that the From header matches authenticated username
  - But From header validation is error-prone because of complex syntax

- **We found 7 out of 8 email providers are vulnerable**
Attackers with an email service account
- Create deceptive content in body, To, and Subject, but not From header

Exp. 6: Combing Replay and Multiple-From Ambiguity (1/2)

1. Attacker emails himself through the email provider server.

RCPT TO: <attacker@gmail.com>

DKIM-Signature: ...; s=selector; d=gmail.com;...
From: <attacker@gmail.com>
To: <victim@victim.com>

Dear Customer,

We are writing to inform you that...
Exp. 6: Combing Replay and Multiple-From Ambiguity (2/2)

② Attacker replays the messages with an extra From header.

Ambiguity: What DKIM verifies differs from what MUA displays
• DKIM components verify the last header
• MUAs show the first header

RCPT TO: <victim@victim.com>

DKIM-Signature: ...; s=selector; d=gmail.com;...
From: <admin@gmail.com>
From: <attacker@gmail.com>
To: <victim@victim.com>

Dear Customer,
We are writing to inform you that…
Thinking on Defense

• Better parsing and protocol spec
  • “Be liberal strict in what you accept”
  • make protocol implementation-friendly
    • simple, well-typed/structured messages, reduce/avoid multiple party processing

• Better UI
  • UI needs more explicit security indicators

• For end-users
  • Don’t blindly trust the email sender displayed in email client
  • Use end-to-end authentication such as PGP
    • PGP may also have parsing ambiguities, but hopefully better than those in SPF/DKIM/DMARC.
We will make this tool publicly available at

https://github.com/chenjj/espoofer
Thank you!

See more demo videos on Youtube.