MIME is broken

Steffen Ullrich, genua GmbH
steffen_ullrich@genua.de

about:me

Steffen Ullrich

- 20+ years working at genua GmbH
  as IT security engineer, researcher, fellow
- Focus not on breaking things, but on protecting what's broken
- Firewall development focus application layer
- Collaboration with academia in research projects,
  focus defense against attacks via mail and web
- Involved in product and research strategies

about:us

genua GmbH

- 30 years old, 360+ employees
  Kirchheim b. München, Berlin, Leipzig, Cologne, Stuttgart
  independently operating subsidiary of Bundesdruckerei
- Security solutions for IT and OT
- Focus on sectors with higher security requirements:
  Public sector, critical infrastructure, regulated industry, eHealth, ...

genua is an Expert for the Protection of Complex and Critical Digital Infrastructures of Companies and Public Organizations.

OUR FIELDS OF USE
Public Sector >
Critical Infrastructure >
Industry >
Protection of Classified Information >

OUR PRODUCT AREAS
Firewalls & Gateways >
VPN >
Remote Maintenance >
All Solutions >

Your career with the leading IT security specialist. Munich – Berlin – Leipzig – Cologne – Stuttgart.

WORK AT GENUA
Professionals & Career Starters >
Training & Dual Studies >
Pupils & Students >

JOIN OUR TEAM
M jobs(at)genua.de
T
+ 49 89 991950-0
F
+ 49 89 991950-0

motivation of research

Supposed to follow standards of application protocols and formats
when implementing content analysis. But ...

- typical standards are unecessary **flexible** and **complex**
- leave too much **room for creative interpretation**:
  - underspecified in edge cases
  - undefined handling of protocol errors
  - SHOULD vs MUST
  - partly conflicting with previous standards
- this **conflicts with security**
  - different implementations have different interpretations in edge cases
  - attackers can use this to
    feed analysis system with seemingly harmless content
    but letting the final target eat the malicious payload

focus of research

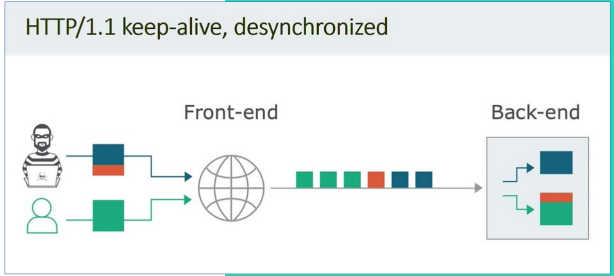
MIME is standard for „rich“ mail: structured, binary attachments, non-ASCII characters.
Using interpretation differences with MIME to bypass security systems

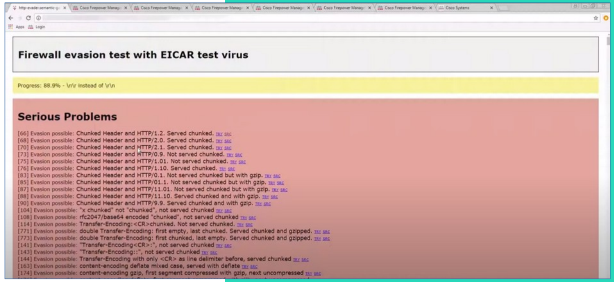
- analysis in mail filter, firewall, IDS, antivirus, ...
  vs. interpretation by mail user agent or web frontend
- bypass malware detection **by content**
  using EICAR test virus, but results relevant for URL detection too
- bypass attachment filtering **by file name**

Similar to research for HTTP/1

- Targeting servers
  HTTP desync attacks (popularized by portswigger, 2019)
- Targeting clients
  Bypassing majority of application firewalls with unexpected responses
  (http-evader, 2015 – fully automated test suite)

Research was done primarily in 2015..2018 (but recently updated)
in context of BMBF sponsored research project APT-Sweeper





Firewall evasion test with EICAR test virus

Progress: 88.9% - \n\r instead of \r\n

Serious Problems

```
[66] Evasion possible: Chunked Header and HTTP/1.2. Served chunked. TRY SRC
[68] Evasion possible: Chunked Header and HTTP/2.0. Served chunked. TRY SRC
[70] Evasion possible: Chunked Header and HTTP/2.1. Served chunked. TRY SRC
[73] Evasion possible: Chunked Header and HTTP/0.9. Not served chunked. TRY SRC
[75] Evasion possible: Chunked Header and HTTP/1.01. Not served chunked. TRY SRC
[76] Evasion possible: Chunked Header and HTTP/1.10. Served chunked. TRY SRC
[83] Evasion possible: Chunked Header and HTTP/0.1. Not served chunked but with gzip. TRY SRC
[85] Evasion possible: Chunked Header and HTTP/01.1. Not served chunked but with gzip. TRY SRC
[87] Evasion possible: Chunked Header and HTTP/11.01. Not served chunked but with gzip. TRY SRC
[88] Evasion possible: Chunked Header and HTTP/11.10. Served chunked and with gzip. TRY SRC
[90] Evasion possible: Chunked Header and HTTP/9.9. Served chunked and with gzip. TRY SRC
[104] Evasion possible: "x chunked" not "chunked", not served chunked TRY SRC
[108] Evasion possible: rfc2047/base64 encoded "chunked", not served chunked TRY SRC
[114] Evasion possible: Transfer-Encoding:<CR>chunked. Not served chunked. TRY SRC
[771] Evasion possible: double Transfer-Encoding: first empty, last chunked. Served chunked and gzipped. TRY SRC
[773] Evasion possible: double Transfer-Encoding: first chunked, last empty. Served chunked and gzipped. TRY SRC
[141] Evasion possible: "Transfer-Encoding<CR>:", not served chunked TRY SRC
[143] Evasion possible: "Transfer-Encoding::", not served chunked TRY SRC
[144] Evasion possible: Transfer-Encoding with only <CR> as line delimiter before, served chunked TRY SRC
[163] Evasion possible: content-encoding deflate mixed case, served with deflate TRY SRC
[174] Evasion possible: content-encoding gzip, first segment compressed with gzip, next uncompressed TRY SRC
```

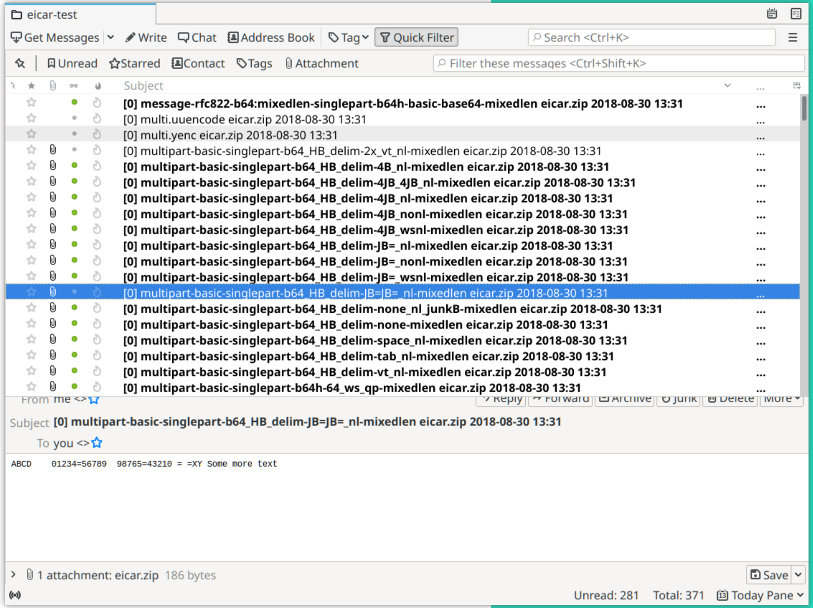
research method

script based generation of lots of test cases with many variations

- 372 mails for bypassing content analysis
- 176 mails for bypassing extension blocking
- exported as files, maildir, pcap

semi-automatic analysis of

- mail user agents
  Thunderbird, Outlook, Apple Mail, mutt, ...
- antivirus, mail filters – standalone and within SMTP
  ClamAV, amavisd-new, ...
- IDS, Firewall
  suricata, snort3, major FW
- libraries
  Perl MIME::Tools, Golang mime/multipart, Python email.parser



```
[0] message-rfc822-b64:mixedlen-singlepart-b64h-basic-base64-mixedlen eicar.zip 2018-08-30 13:31
[0] multi.uuencode eicar.zip 2018-08-30 13:31
[0] multi.yenc eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-2x_vt_nl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-4B_nl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-4JB_4JB_nl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-4JB_nl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-4JB_nonl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-4JB_wsnl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-JB=_nl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-JB=_nonl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-JB=_wsnl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-JB=JB=_nl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-none_nl_junkB-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-none-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-space_nl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-tab_nl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64_HB_delim-vt_nl-mixedlen eicar.zip 2018-08-30 13:31
[0] multipart-basic-singlepart-b64h-64_ws_qp-mixedlen eicar.zip 2018-08-30 13:31

From me <>
Subject [0] multipart-basic-singlepart-b64_HB_delim-JB=JB=_nl-mixedlen eicar.zip 2018-08-30 13:31
To you <>

ABCD    01234=56789  98765=43210 = =XY Some more text

1 attachment: eicar.zip 186 bytes
Unread: 281   Total: 371
```

MIME essentials

what is MIME

In the beginning ...

- ASCII only, maximum line length 1000 bytes

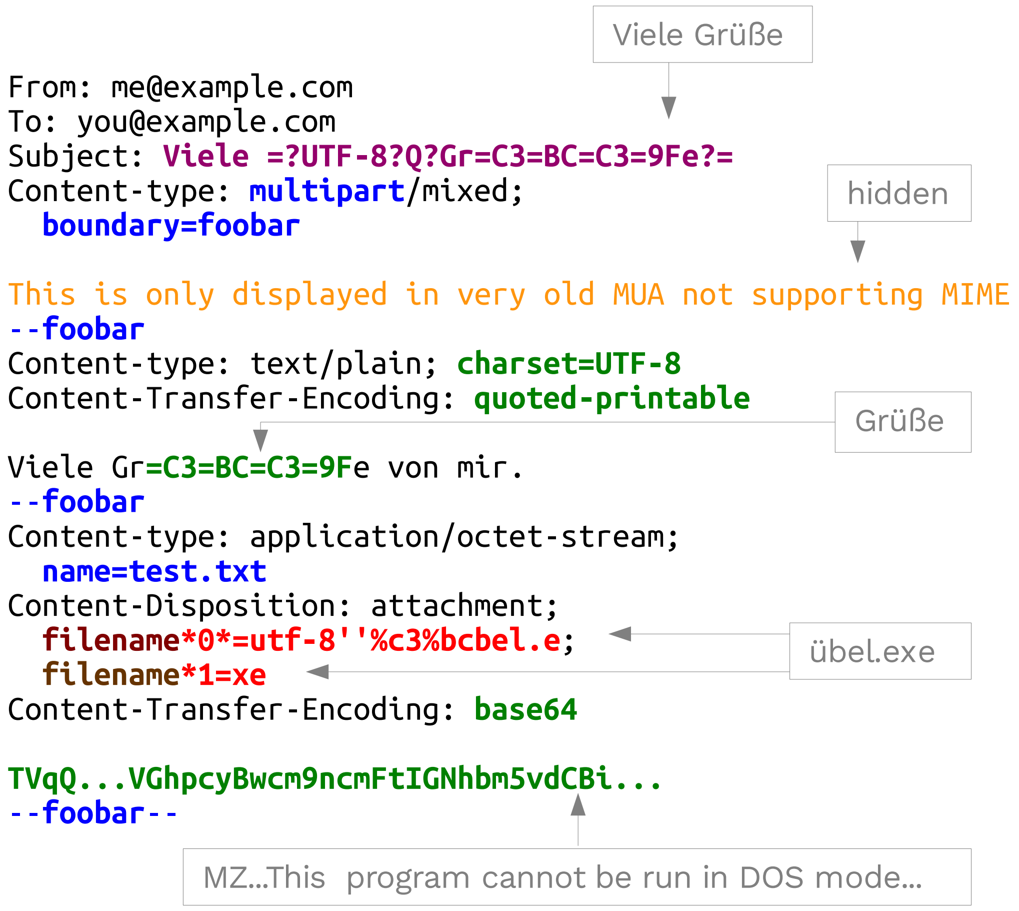
Enter MIME RFC 2045-2048 (1996) - serialization within the original limits

- Multipurpose Internet Mail Extension
- encoding of non-ASCII **characters** and **binary** data in body and header fields
- encoding of **structure**: MIME parts with various types and relations
- flexible, complex, underspecified, lots of room for creative interpretation, ...

Later (1997)

- RFC 2183: Content-Disposition
  context for MIME parts: inline|attachment, filename, date ...
- RFC 2231: **long non-ASCII parameter values** like for filename
  different encoding for unstructured (RFC2047) and structured (RFC2231) fields

MIME by example



```
From: me@example.com
To: you@example.com
Subject: Viele =?UTF-8?Q?Gr=C3=BC=C3=9Fe?=
Content-type: multipart/mixed;
  boundary=foobar

This is only displayed in very old MUA not supporting MIME
--foobar
Content-type: text/plain; charset=UTF-8
Content-Transfer-Encoding: quoted-printable

Viele Gr=C3=BC=C3=9Fe von mir.
--foobar
Content-type: application/octet-stream;
  name=test.txt
Content-Disposition: attachment;
  filename*0*=utf-8''%c3%bcbel.e;
  filename*1=xe
Content-Transfer-Encoding: base64

TVqQ...VGhpcyBwcm9ncmFtIGNhbm5vdCBi...
--foobar--
```

RFC 2046
Serializing structure, MIME parts
multipart/...; boundary=
Content-type: ...; name=

RFC 2045
Encoding binary, characters in body
Content-Transfer-Encoding: base64 | quoted-printable
Content-type: ...; charset=

RFC 2047
Encoding characters in header
base64 | quoted-printable
charset

RFC 2183
Content-Disposition
inline | attachment; filename=

RFC 2231
Encoding characters in parameter
charset, language
URL encoding of non-ASCII
split long parameter values

genua.

bypass content analysis

selected examples

conflicting Content-Transfer-Encoding I

```
Content-Transfer-Encoding: base64
Content-Transfer-Encoding: quoted-printable

Zm9vYmFyCg==
```

first field

Thunderbird, Outlook, Apple Mail

ClamAV, amavisd-new, ~~suricata~~, ~~snort3~~, FW
MIME::Tools, mime/multipart, email.parser

```
Content-Transfer-Encoding: quoted-printable
Content-Transfer-Encoding: base64

Zm9vYmFyCg==
```

last field

mutt

ClamAV, amavisd-new, suricata, ~~snort3~~, FW
~~MIME::Tools~~, ~~mime/multipart~~, ~~email.parser~~

conflicting Content-Transfer-Encoding II

```
Content-Transfer-Encoding:
  base64, quoted-printable

Zm9vYmFyCg==
```

```
Content-Transfer-Encoding:
  quoted-printable, base64

Zm9vYmFyCg==
```

first field element	last field element	no encoding
Thunderbird, mutt		Outlook, Apple Mail
ClamAV, amavisd-new, suricata[1], ~~snort3~~, FW ~~MIME::Tools~~, ~~mime/multipart~~, ~~email.parser~~	ClamAV, amavisd-new, suricata[1], ~~snort3~~, FW ~~MIME::Tools~~, ~~mime/multipart~~, ~~email.parser~~	

[1] base64 fine, but fails to completely decode and analyze quoted-printable for files

conflicting multipart boundary I

```
Content-Type: multipart/mixed; boundary=bar
Content-Type: multipart/mixed; boundary=foo

--foo
--bar
Content-type: text/plain

foobar
--bar--
--foo--
```

first field

Thunderbird, Outlook, Apple Mail

~~ClamAV~~, amavisd-new, ~~suricata~~,
snort3, FW
MIME::Tools, mime/multipart,
email.parser

```
Content-Type: multipart/mixed; boundary=foo
Content-Type: multipart/mixed; boundary=bar

--foo
--bar
Content-type: text/plain

foobar
--bar--
--foo--
```

last field

mutt

ClamAV, ~~amavisd-new~~, suricata,
~~snort3~~, FW
~~MIME::Tools~~, ~~mime/multipart~~,
~~email.parser~~

conflicting multipart boundary II

```
Content-Type: multipart/mixed;
  boundary=bar; boundary=foo

--foo
--bar
Content-type: text/plain

foobar
--bar--
--foo--
```

first field

Thunderbird, Outlook, mutt

ClamAV, ~~amavisd-new~~, ~~suricata~~, ~~snort3~~, FW
~~MIME::Tools~~, ~~mime/multipart~~, email.parser

```
Content-Type: multipart/mixed;
  boundary=foo; boundary=bar

--foo
--bar
Content-type: text/plain

foobar
--bar--
--foo--
```

last field

Apple Mail

~~ClamAV~~, amavisd-new, ~~suricata~~, ~~snort3~~, FW
MIME::Tools, ~~mime/multipart~~, ~~email.parser~~

padding in the middle of base64

```
Content-Transfer-Encoding: base64
Zm9vYg==
YXI=
```

- converting 3 bytes binary to 4 bytes ASCII
- less than 3 bytes → padding with „=“

foob	(2 bytes padding)
ar	(1 byte padding)

foob	foobar	foob<garbage>
mutt	Thunderbird, Apple Mail ~~ClamAV~~, amavisd-new, ~~suricata~~, ~~snort3~~, ~~FW MIME::Tools~~, ~~mime/multipart~~, ~~email.parser~~	Outlook

RFC 2045 section 6.8

*Because it is used only for padding at the end of the data, the occurrence of any "=" characters **may** be taken as evidence that the end of the data has been reached*

encoding yEnc – historic greetings from usenet news

```
Content-Transfer-Encoding: x-yencode

=ybegin line=128 size=51 name=file.bin
... nearly binary stuff ...
=yend size=51
```

not encoded	yEnc encoded
Outlook, Apple Mail, mutt	Thunderbird ~~ClamAV~~, ~~amavisd-new~~, ~~suricata~~, ~~snort3~~, ~~FW MIME::Tools~~, ~~mime/multipart~~, ~~email.parser~~

https://en.wikipedia.org/wiki/YEnc

There is no RFC or other standards document describing yEnc. The yEnc homepage contains a draft informal specification and a grammar (which contradict RFC 2822 and RFC 2045), although neither has been submitted to the Internet Engineering Task Force.

encoding uuencode – from a world before MIME

```
Content-Transfer-Encoding: x-uuencode

begin 644 file.bin
M04)#1`DP,3(S-#TU-C<X.2`@.3@W-C4]-#,R,3`@/2`]6%D@4V]M92!M;W)E
&('1E>'0*
end
```

<u>not encoded</u>	<u>uuencode encoded</u>
Apple Mail, mutt	Thunderbird[1], Outlook[2]
	ClamAV[3], amavisd-new[4], ~~suricata~~, snort3[2], FW[4] MIME::Tools, ~~mime/multipart~~, email.parser[5]

[1] Also „x-uue“ and „uuencode“, with begin/end and without
[2] Also „uuencode“, „end“ can be skipped
[3] Also „x-uue“ and „uuencode“, only „end“ can be skipped
[4] all variations
[5] also „uue“ and „x-uue“ and „uuencode“, but both begin and end are required

comments in wild places

```
Content-Type: multipart/mixed;
  boundary=(boundary=foo)bar

--bar
...
```

boundary „bar“	no clue
Outlook	Thunderbird, Apple Mail, mutt
ClamAV, ~~amavisd-new~~, ~~suricata~~, ~~snort3~~, FW ~~MIME::Tools~~, ~~mime/multipart~~, email.parser	

RFC 2822 section 3.2.3:

... There are several places in this standard where comments and FWS may be freely inserted ...

bypass filtering filename

selected examples

filename for attachments - RFC2231

```
Content-Disposition: attachment;
   filename=file.png;
   filename*1=zip; filename*0=file.
```

file.zip

Thunderbird[3], Apple Mail, mutt[2]

amavisd-new, ~~suricata~~, ~~snort3~~, ~~FW~~
MIME::Tools, mime/multipart,
email.parser

file.png

Outlook[1], mutt[2]

RFC 2231

.... the mechanism MUST NOT depend on ***parameter ordering*** *since MIME states that parameters are not order sensitive.*

[1] does not implement RFC2231 at all
[2] RFC2231 name does not take preference,
will recognize RFC2231 if plain filename is not given,
will even recognize if indices don't start with 0 and have gaps
[3] supports RFC2231 even for *boundary* parameter in Content-Type

filename for attachments - RFC2047

```
Content-Disposition: attachment;
   filename="=?us-ascii?B?ZmlsZS56aXA=?="
```

file.zip

Thunderbird[1], Outlook[2], Apple Mail[3]

amavisd-new[3], ~~suricata~~, ~~snort3~~, FW/~~FW~~[1] MIME::Tools[3], ~~mime/multipart~~, email.parser[2]

`=?us-ascii?B?...`

mutt

RFC 2047 section 5

- *An 'encoded-word' MUST NOT appear within a 'quoted-string'.*
- *An 'encoded-word' MUST NOT be used in parameter of a MIME Content-Type or Content-Disposition field, or in any structured field body except within a 'comment' or 'phrase'.*

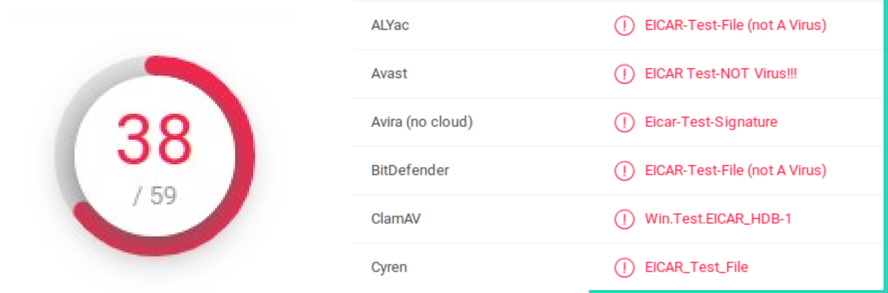
[1] no UTF-16* [2] also UTF-16LE, no UTF-16BE [3] no UTF-16LE, but UTF-16BE
Also: MS-Exchange wrongly transforms filename from RFC2231 to quoted RFC2047

applying knowledge to bypass most antivirus

step by step bypass vscan virustotal – I (ground truth)

```
From: me@example.com
To: you@example.com
Subject: plain
Content-type: multipart/mixed; boundary=foo

--foo
Content-type: text/plain

Virus attached

--foo
Content-type: application/zip; name=whatever.zip
Content-Transfer-Encoding: base64

UEsDBBQAAgAIABFKjkk8z1FoRgAAAEQAAAAJAAAAZWljYXIuY29tizD1VwxQdXAMiDaJCYiKMDXR
CIjTNHd21jSvVXH1dHYM0g0OcfRzcQxy0XX0C/EM8wwKDdYNcQ0O0XXz9HFVVPHQ9tACAFBLAQIU
AxQAAgAIABFKjkk8z1FoRgAAAEQAAAAJAAAAAAAAAAAAAAAAC2gQAAAABlaWNhci5jb21QSwUGAAAA
AAEAAQA3AAAAbQAAAAAA
--foo--
```

EICAR inside

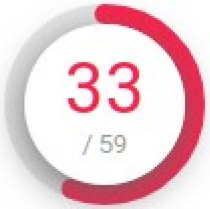


38 / 59

ALYac	EICAR-Test-File (not A Virus)
Avast	EICAR Test-NOT Virus!!!
Avira (no cloud)	Eicar-Test-Signature
BitDefender	EICAR-Test-File (not A Virus)
ClamAV	Win.Test.EICAR_HDB-1
Cyren	EICAR_Test_File

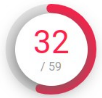
step by step bypass vscan virustotal – II (conflicting CTE)

```
From: me@example.com
To: you@example.com
Subject: b64-64qp
Content-type: multipart/mixed; boundary=foo

--foo
Content-type: text/plain

Virus attached

--foo
Content-type: application/zip; name=whatever.zip
Content-Transfer-Encoding: base64
Content-Transfer-Encoding: quoted-printable

UEsDBBQAAgAIABFKjkk8z1FoRgAAAEQAAAAJAAAAZWljYXIuY29tizD1VwxQdXAMiDaJCYiKMDXR
CIjTNHd21jSvVXH1dHYM0g0OcfRzcQxy0XX0C/EM8wwKDdYNcQ0O0XXz9HFVVPHQ9tACAFBLAQIU
AxQAAgAIABFKjkk8z1FoRgAAAEQAAAAJAAAAAAAAAAAAAAAAC2gQAAAABlaWNhci5jb21QSwUGAAAA
AAEAAQA3AAAAbQAAAAAA
--foo--
```



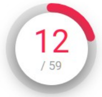
Order of CTE does not matter much

Switched Order



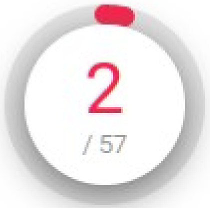
Some antivirus seems to apply heuristics for detecting base64

CTE xxxx

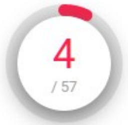


step by step bypass vscan virustotal – III (chunked base64)

```
From: me@example.com
To: you@example.com
Subject: b64eq-64qp
Content-type: multipart/mixed; boundary=foo

--foo
Content-type: text/plain

Virus attached

--foo
Content-type: application/zip; name=whatever.zip
Content-Transfer-Encoding: base64
Content-Transfer-Encoding: quoted-printable

UEs=AwQ=FAA=AgA=CAA=EUo=jkk=PM8=UWg=RgA=AAA=RAA=AAA=CQA=AAA=ZWk=Y2E=ci4=
Y28=bYs=MPU=Vww=UHU=cAw=iDY=iQk=iIo=MDU=0Qg=iNM=NHc=dtY=NK8=VXE=9XQ=dgw=
0g0=DnE=9HM=cQw=ctE=dfQ=C/E=DPM=DAo=DdY=DXE=DQ4=0XU=8/Q=cVU=VPE=0PY=0AI=
AFA=SwE=AhQ=AxQ=AAI=AAg=ABE=So4=STw=z1E=aEY=AAA=AEQ=AAA=AAk=AAA=AAA=AAA=
AAA=AAA=ALY=gQA=AAA=AGU=aWM=YXI=LmM=b20=UEs=BQY=AAA=AAA=AQA=AQA=NwA=AAA=
bQA=AAA=AAA=
--foo--
```

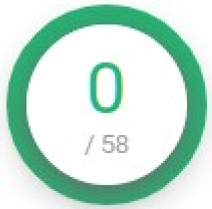


chunked base64
by its own, without
duplicate CTE

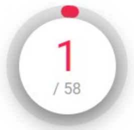


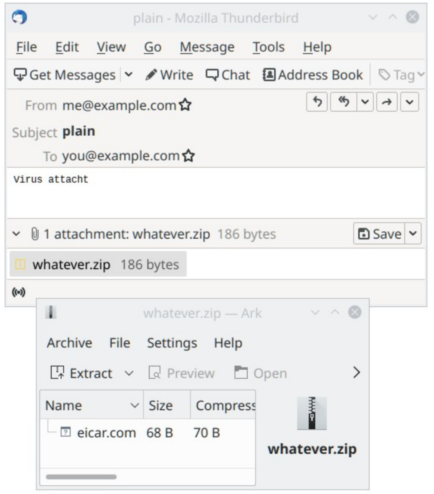
step by step bypass vscan virustotal – IV (double boundary)

```
From: me@example.com
To: you@example.com
Subject: b64eq-64qp-bd:good,bd:bad
Content-type: multipart/mixed; boundary=foo
Content-type: multipart/mixed; boundary=bar

--foo
Content-type: text/plain

Virus attached

--foo
Content-type: application/zip; name=whatever.zip
Content-Transfer-Encoding: base64
Content-Transfer-Encoding: quoted-printable

UEs=AwQ=FAA=AgA=CAA=EUo=jkk=PM8=UWg=RgA=AAA=RAA=AAA=CQA=AAA=ZWk=Y2E=ci4=
Y28=bYs=MPU=Vww=UHU=cAw=iDY=iQk=iIo=MDU=0Qg=iNM=NHc=dtY=NK8=VXE=9XQ=dgw=
0g0=DnE=9HM=cQw=ctE=dfQ=C/E=DPM=DAo=DdY=DXE=DQ4=0XU=8/Q=cVU=VPE=0PY=0AI=
AFA=SwE=AhQ=AxQ=AAI=AAg=ABE=So4=STw=z1E=aEY=AAA=AEQ=AAA=AAk=AAA=AAA=AAA=
AAA=AAA=ALY=gQA=AAA=AGU=aWM=YXI=LmM=b20=UEs=BQY=AAA=AAA=AQA=AQA=NwA=AAA=
bQA=AAA=AAA=
--foo--
```



Without CTE confusion




plain - Mozilla Thunderbird
File Edit View Go Message Tools Help
Get Messages Write Chat Address Book Tag
Save
whatever.zip — Ark
Archive File Settings Help
Extract Preview Open

From me@example.com
Subject plain
To you@example.com

Virus attacht

1 attachment: whatever.zip 186 bytes

whatever.zip 186 bytes

Name	Size	Compress
eicar.com	68 B	70 B

whatever.zip

MIME vs. cryptography

bypassing DKIM signatures with bad MIME - I

- DKIM major part of DMARC phishing protection. Basic idea:
  - outgoing mail server for domain signs mail header and body
  - recipient can get public key from DNS and check signature
    `DKIM-Signature: .. d=domain; s=20140901 -> dig txt 20140901._domainkey.domain`
  - if signature valid and domain aligned (From: user@domain)
    → DMARC pass, i.e. sender domain verified and not spoofed
- Broken standard and implementations
  - no requirements which header fields should be protected,
    only **insufficient recommendations**
  - able to prevent critical header fields to be added by attacker,
    but **no actual requirement** to do so
  - **implementations usually fail** to protect critical headers
  - ability to sign only part of body
    warns of security problems, but nevertheless allows it

bypassing DKIM signatures with bad MIME - II

```
DKIM-Signature: v=1; l=1850; d=dhl.com; s=20140901;
  h=date:from:to:message-id:subject:mime-version;
  b=...; bh=...
Date: Thu, 24 Sep 2017 19:08:23 +0800 (MYT)
Date: Thu, 14 Jan 2016 19:08:23 +0800 (MYT)
From: DHL Customer Support <support@dhl.com>
To: somebody@example.com
To: auftrag@original-company-not-shown
Message-ID: <9953648784.9145749@dhl.com>
Message-ID: <1453648784.9145749.1452769703900.JavaMail...dhl.com>
Subject: DHL Shipment Digest
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=BAD
Content-Type: multipart/mixed; boundary=----=_Part_9145747_2082645767.1452769703900

------=_Part_9145747_2082645767.1452769703900
Content-type: text/plain

The real DHL Shipment Digest ...
------=_Part_9145747_2082645767.1452769703900
--BAD
Content-type: text/plain

This is a faked mail with valid DKIM signature from DHL.
--BAD--
```

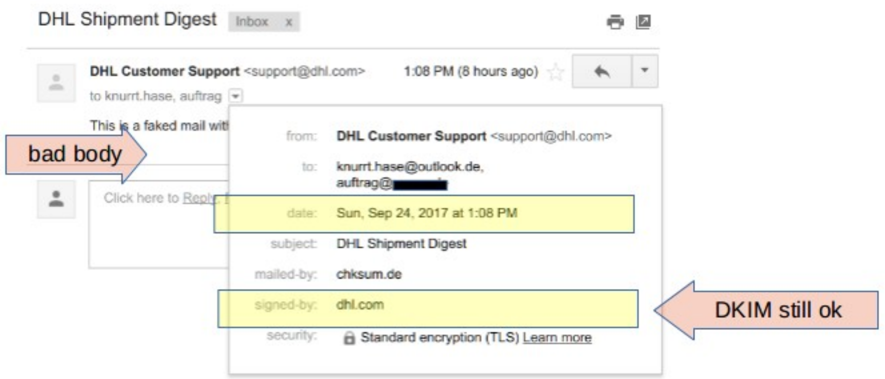
only part was signed, new data can be added after that

only orginal *date* and *to* are included in signature
signature takes fields from bottom, MUA from top

From aligned with DKIM domain → DMARC pass

BAD boundary is active
previous content treated as MIME preamble
→ newly added unsigned content shown

bypassing DKIM signatures with bad MIME - III



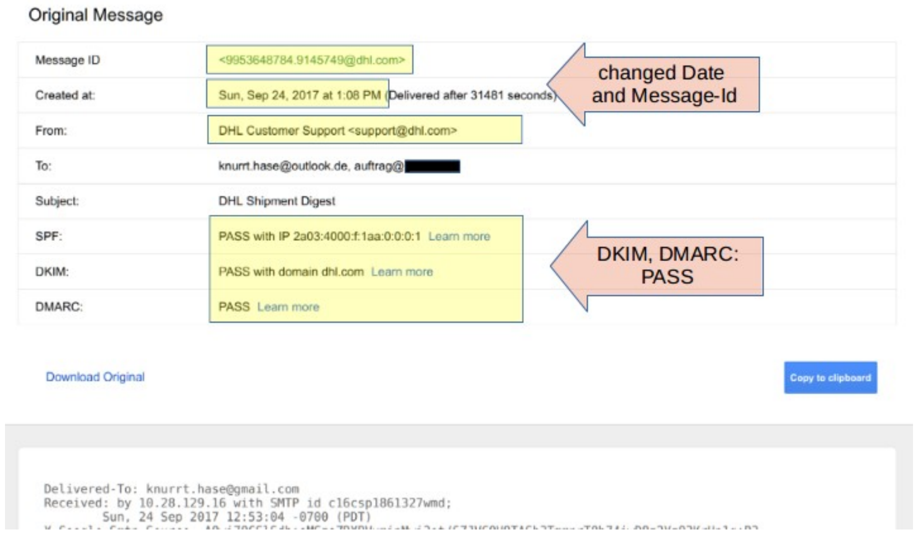
DHL Shipment Digest Inbox x

DHL Customer Support <support@dhl.com> 1:08 PM (8 hours ago)
to knurrt.hase, auftrag

This is a faked mail wit

Click here to Reply

from:	DHL Customer Support <support@dhl.com>
to:	knurrt.hase@outlook.de, auftrag@[redacted]
date:	Sun, Sep 24, 2017 at 1:08 PM
subject:	DHL Shipment Digest
mailed-by:	chksum.de
signed-by:	dhl.com
security:	Standard encryption (TLS) Learn more



Original Message

Message ID	<9953648784.9145749@dhl.com>
Created at:	Sun, Sep 24, 2017 at 1:08 PM (Delivered after 31481 seconds)
From:	DHL Customer Support <support@dhl.com>
To:	knurrt.hase@outlook.de, auftrag@[redacted]
Subject:	DHL Shipment Digest
SPF:	PASS with IP 2a03:4000:f:1aa:0:0:0:1 Learn more
DKIM:	PASS with domain dhl.com Learn more
DMARC:	PASS Learn more

```
Delivered-To: knurrt.hase@gmail.com
Received: by 10.28.129.16 with SMTP id c16csp1861327wmd;
        Sun, 24 Sep 2017 12:53:04 -0700 (PDT)
```

https://noxxi.de/research/breaking-dkim-on-purpose-and-by-chance.html

final words

solutions?

Problems are hard to fix

- zillions of MIME implementations and scripts in the wild, often broken
- no „monopoly“ implementations to enforce quality, like we have with browsers

Blocking invalid and edge-cases cause unbearable collateral effects

- too much junk in real world which works sufficiently enough
  (i.e. with a specific MUA in mind)
- operation beats security: „it worked before we installed the firewall“

Sanitizing (rewriting) content might cause problems with cryptographic signatures

- DKIM, PGP, S/MIME

Logging problems

- hope someone cares about logs

MIME is broken

bonus

customer story: but it worked w/o firewall

- customer complained that mail was blocked by firewall
  mail was created by script, using `uuencode --base64`

```
Content-Transfer-Encoding: base64

begin-base64 644 file012.pdf
JVBERi0xLjcNJeLjz9MNCjc2MiAwIG9iag08PC9MaW5lYXJpemVkIDEvTCA
...
```

- reason for blocking: invalid base64 characters
- worked before only, because
  - invalid base64 characters are ignored by MUA
  - 24 valid base64 characters are multiple of 4
    and decode to 18 bytes junk prefixing the real PDF file
  - leading junk will be ignored by PDF reader