École polytechnique de Louvain (EPL)

IMAP proxy to sanitize attachments

Xavier Schul, MSc Student.
xavier.schul@student.uclouvain.be

June 25, 2018

Context

- Email is the most frequently used delivery mechanism for malware
- Most anti-viruses ineffective against 0-day malware
- New ways for viruses to spread and become less detectable

$\Rightarrow$ We need a complementary tool to anti-viruses

Context
PyCIRCLeanMail

- Open-source email sanitizer developed in Python
- Developed by the Computer Incident Response Center Luxembourg (CIRCL)

circl.lu
Computer Incident Response Center
LUXEMBOURG

This tool labels as dangerous files with active content

Context
PyCIRCLeanMail

We would like to sanitize the email before the user retrieves it

- User has no IT knowledge ⇒ the process should be transparent
- User does not have access to its mail server ⇒ need a proxy
- User does not have financial means for sophisticated sanitizing tools ⇒ need an open-source solution

⇒ Need an open-source IMAP transparent proxy
☹ Not available

Objectives

Implement an IMAP transparent proxy to

- Integrate PyCIRCLeanMail module
- Provide a **generic** and **scalable** IMAP transparent proxy to the open-source community

Outline

1 Context

2 Theoretical base
- 2.1 IMAP
- 2.2 Transparent proxy

3 Detect malicious attachments
- 3.1 Anti-malware problem
- 3.2 PyCIRCLeanMail

4 Proxy implementation
- 4.1 Modules

5 Demonstration

6 Performance

7 Conclusion

Theoretical base

IMAP

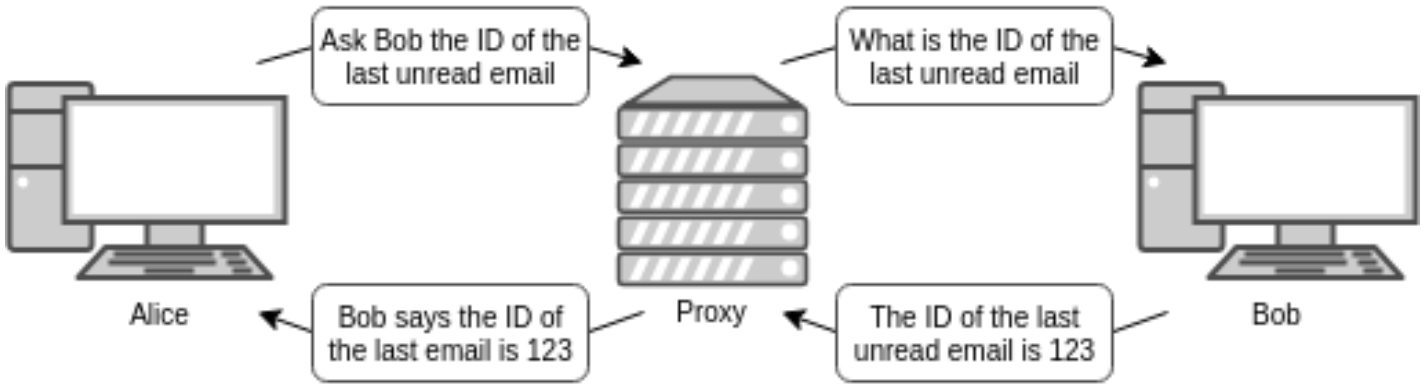
- **I**nternet **M**essage **A**ccess **P**rotocol defined by the RFC 3501
- Used by e-mail client to retrieve e-mail messages from a mail server

IMAP
Client commands

Tag + Command + Arguments

- **Tag** = alphanumerical identifier of the command
  - *imaplib* uses 4 same letters + digits (ZRTF2, ZRTF3, ZRTF4,...)
  - *Thunderbird* uses only digits (1, 2, 3,...)
  - *Outlook* generates a different tag for each command

Transparent proxy



- A **proxy** is an intermediary between a client and a server
  - IMAP proxies: *Imapproxy*, *Perdition*
- A **transparent proxy** is a proxy that does not modify the request or response beyond what is required for authentication and identification

Detect malicious attachments

Anti-malware problem
Methodologies

Most of anti-malwares use 2 methodologies:

- **Signature-based**
  - Analyze the file and generate its signature
  - Compare this signature to signatures in a database
  - If the signature is not present ⇒ safe

  ☹ Do not detect 0-day malwares

  ☹ Transformation techniques to by-pass detection
- **Behaviour-based**
  - Analyze the behaviour of files

  ☺ Can detect 0-day malwares

  ☹ Reports much more false-positives

  ☹ Malwares can behave as normal application

Anti-malware problem
Experiments

Several experiments [1][2][3] show that

- Detection rate ranges from 40% to 80%
- Only 50% of scanners detect 0-day malwares
- Two days are required to detect new malware sample
- After a year, 10% of the scanners still do not detect some 0-day malwares

PyCIRCLeanMail

Analyzes the files statically and without running their content in order to label as dangerous each file containing or likely to contain active content:

- Commonly used malicious extensions
- Extensions that do not correspond to the MIME type
- Files without extension
- Libreoffice, Windows Office and PDF documents with active content

Proxy implementation

Proxy implementation

The client thinks he is talking to the server

- Client first authenticates to the proxy
- Proxy authenticates to the server with the client's credentials
- Proxy **swap tag** and transmits requests and response

```
[-->]:  CLIENT1 CLOSE
  [-->]:  SERVER1 CLOSE
  [<--]:  SERVER1 OK CLOSE completed
[<--]:  CLIENT1 OK CLOSE completed
```

Proxy implementation
Features

- Compatible with *Thunderbird*, *Outlook*,...
- Compatible with IPv6 only networks
- Support non-secure **and** secure connection
- Possibility to display IMAP payload
- Scalable and modular

Proxy implementation
Scalability and modularity

- Easy to intercept new commands

```
def logout(self):
    """ Logout and stop listening the client """
    self.listen_client = False
    self.transmit()

def select(self):
    """ Select a mailbox """
    self.set_current_folder(self.client_flags)
    self.transmit()
```

```
def move(self):
    """ Move an email to another mailbox """
    misp.process(self)
    self.transmit()

def fetch(self):
    """ Fetch an email """
    pycircleanmail.process(self)
    self.transmit()
```

- Easy to support new authentication mechanisms

```
def authenticate(self):
    """ Authenticate the client and call the given auth mechanism """
    auth_type = self.client_flags.split(' ')[0].lower()
    getattr(self, self.client_command+"_"+auth_type)()

def authenticate_plain(self):
```

Proxy implementation
Scalability and modularity

- Adding a new *Capability* might require some changes in the code

```
CAPABILITIES = (
    'IMAP4',
    'IMAP4rev1',
    'AUTH=PLAIN',
    'UIDPLUS',
    'MOVE',
    'ID',
    'UNSELECT',
    'CHILDREN',
    'NAMESPACE'
)
```

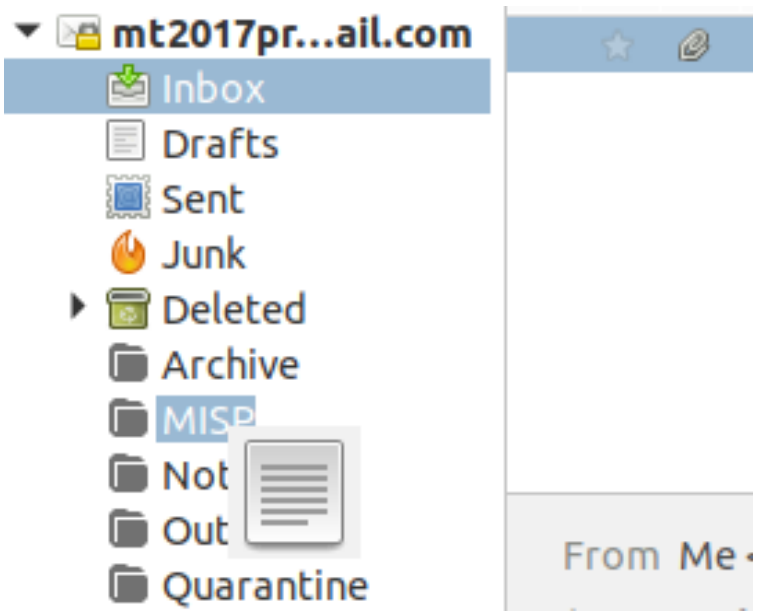
If the user wants to FETCH an email

- The proxy first FETCH the `X-CIRCL-Sanitizer` signature of this email
- If the email contains the signature and has a correct value $\Rightarrow$ already sanitized
- Else:
  - Fetch the entire email
  - Sanitize the email
  - Append the sanitized version of the email in the current folder
  - Append a copy of the original email in a *Quarantine* folder
  - Remove the original email

Finally, proxy transmits the original request to the server

Malware **I**nformation **S**haring **P**latform

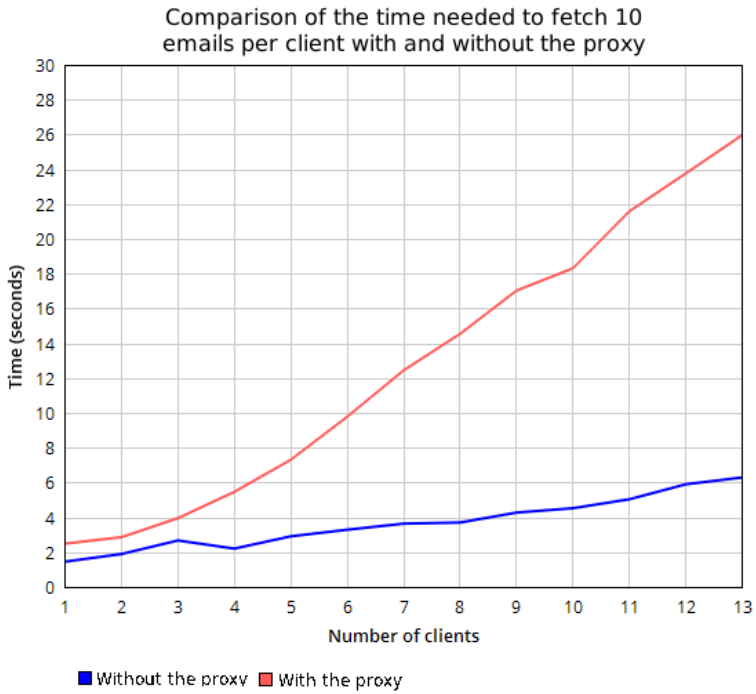
- Users drag and drop an email to *MISP* folder
- This email is sent to *MISP* SMTP server

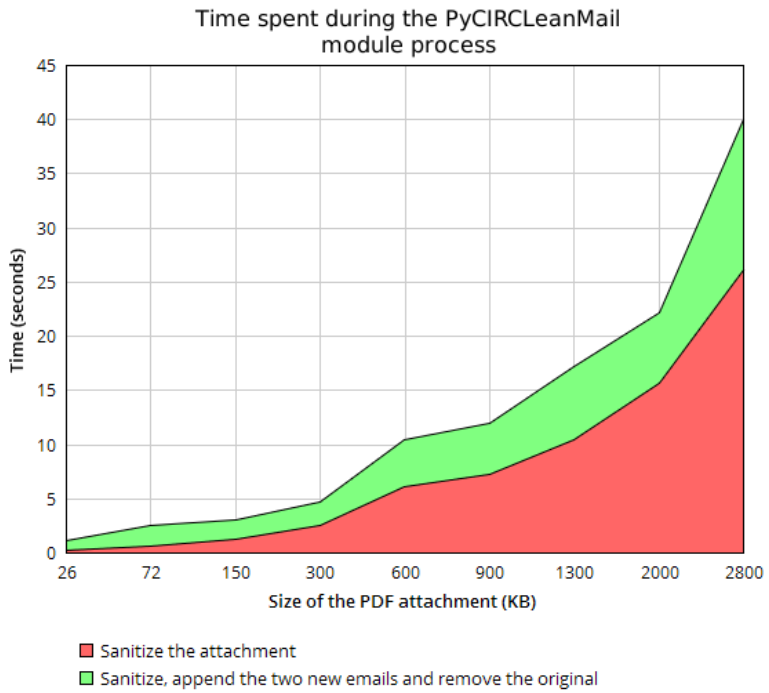


Demonstration

Performance

Performance
Time





- Not adapted to continuously listen to multiple clients
- Time seems exponential depending on the size of a PDF
  - But some detections are instantaneous (extensions blacklist)

Conclusion

- Scalable and modular IMAP transparent proxy
- PyCIRCLeanMail detects active content and complements anti-malwares
- Not adapted for multiple concurrent users

Bibliography

1. Q. K. A. Mirza, G. Mohi-Ud-Din, and I. Awan (2016). "A Cloud-Based Energy Efficient System for Enhancing the Detection and Prevention of Modern Malware".
2. Giovanni Vigna (2014). "Antivirus Isnt Dead, It Just Cant Keep Up".
3. A. Zarghoon et al. (2017). "Evaluation of AV systems against modern malware".