RP2: Automated end-to-end email component testing

MSc. Security & Network Engineering

Kevin Csuka, Isaac Klop

October 16, 2018

University of Amsterdam
Supervisor: Michiel Leenaars, NLnet

Intro

Intro

- E-mail software is complex
- Large surface for human error

Intro

- E-mail software is complex
- Large surface for human error
- How do you know you did it right?
- Anxiety around managing own mail server

Intro

- E-mail software is complex
- Large surface for human error
- How do you know you did it right?
- Anxiety around managing own mail server
- Misses an automated end-to-end test

Research Question

To what extent can we prove a mail server is properly set up via end-to-end component testing?

Related Work

- End-to-end integration testing [Paul, 2001] [1]

Related Work

- End-to-end integration testing [Paul, 2001] [1]

- Internet.nl [2]
- mail-tester.com [3]
- MxToolbox [4]
- emailsecuritycheck.net [5]

Related Work

- End-to-end integration testing [Paul, 2001] [1]

- Internet.nl [2]
- mail-tester.com [3]
- MxToolbox [4]
- emailsecuritycheck.net [5]

- Not end-to-end
- Not automated

Method

Divided in 3 parts:

Method

Divided in 3 parts:

1. Taxonomy

Method

Divided in 3 parts:

1. Taxonomy
2. Design tests
    - End-to-end testing
    - Black box
    - RFC/Specifications/Best Practices

Method

Divided in 3 parts:

1. Taxonomy
2. Design tests
   - End-to-end testing
   - Black box
   - RFC/Specifications/Best Practices
3. Proof of concept
   - Python3
   - Modular
   - Continuous Integration / Continuous Deployment (CI/CD)

Results - Taxonomy

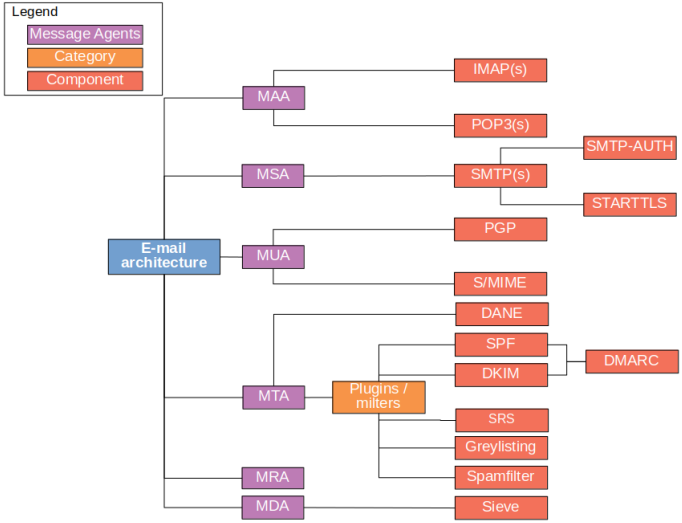

Figure 1: Taxonomy of the e-mail architecture

Results - Test Design

- Expected behaviour of components
- Refer to the respective RFC/Specification

Results - Test Design

- Expected behaviour of components
- Refer to the respective RFC/Specification

- E.g. SPF [6]
  - *HELO* domain, *MAIL FROM* domain, IP address
  - Is IP address authorized for domain?
  - Returns result code (i.e. *pass*, *fail*, *softfail* etc.)
  - RFC guidelines for result

Results - Test Design

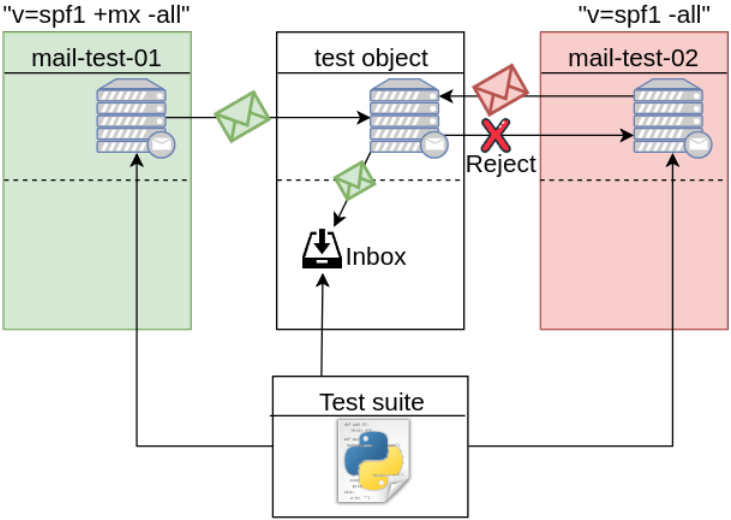

Figure 2: SPF test design example

Proof of Concept

- Multiple mail servers
  - Public IP address
  - Different configuration
  - Intentional flaws in configuration/DNS records
  - Automated via Ansible

Components	Implemented
IMAP	✓
SMTP	✓
SMTP-AUTH	✓
TLS	✓
DANE	✓
SPF	✓
DKIM	✓
DMARC	partial
SRS	✗
Greylisting	partial
Spamfilter	partial
Sieve	✓

Table 1: Components which the test suite can and cannot verify

Proof of Concept - Limitations

- Guidance from RFC/specification is limited
  - SPF softfail [6]
  - Greylisting [7]
  - Various errors
- DMARC sending report
- SRS
- Spamfilter

Proof of Concept

```
kcsuka@desktop-27: ~/end2end_email_components_testing
kcsuka@desktop-27: ~/end2end_email_components_testing 80x24
ok
test_51_dane_receive_notlsa (__main__.EmailTestCase) ... [b'17']
ok
test_52_dane_send_badtlsa (__main__.EmailTestCase) ... [b'']
Cannot read the mail in mailbox: INBOX with: 153918368917321813
ok
test_53_dane_send_notlsa (__main__.EmailTestCase) ... [b'22']
ok

======================================================================
FAIL: test_41_dmarc_receive_policy (__main__.EmailTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "./testunit.py", line 551, in test_41_dmarc_receive_policy
    "policy but did not reject the email")
AssertionError: Email delivered to target that should have been rejected as per
DMARC policy. Target correctly identified the failed DMARC policy but did not re
ject the email

----------------------------------------------------------------------
Ran 18 tests in 77.780s

FAILED (failures=1, skipped=3)
→  end2end_email_components_testing git:(master) ✗
```

Figure 3: Test suite - test run

Conclusion

- Tool assures administrator components work properly
- Limitations

Discussion

- Not all test cases covered - no complete taxonomy
- Opinionated (RFC often states SHOULD)
- End-to-end testing vs. unit/integration testing

Future Work

- Complete topology/taxonomy of e-mail infrastructure components
- Spam filter
- Expand current tests, e.g. ARC [8], edge cases
- Form of authentication for the test mail-servers
- Comparison study

Questions?

?

References i

- R. Paul, "End-to-end integration testing," in *Quality Software, 2001. Proceedings. Second Asia-Pacific Conference on*. IEEE, 2001, pp. 211–220.
- Dutch Internet Standards Platform, "About the email test," https://en.internet.nl/test-mail/, Accessed, Oct. 10 2018.
- MailPoet & AcyMailing., "Test the Spammyness of your Emails," https://www.mail-tester.com/, Accessed, Oct. 10 2018.
- MxToolbox, Inc., "MxToolbox," https://mxtoolbox.com/SuperTool.aspx, Accessed, Oct. 10 2018.
- Byteplant, "Free Email Security Check," https://www.emailsecuritycheck.net/index.html, Accessed, Oct. 15 2018.

References ii

- S. Kitterman, "Sender policy framework (spf) for authorizing use of domains in email, version 1," Internet Requests for Comments, RFC Editor, RFC 7208, April 2014, http://www.rfc-editor.org/rfc/rfc7208.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7208.txt
- M. Kucherawy and D. Crocker, "Email greylisting: An applicability statement for smtp," Internet Requests for Comments, RFC Editor, RFC 6647, June 2012, http://www.rfc-editor.org/rfc/rfc6647.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6647.txt
- Tim Wicinski, https://datatracker.ietf.org/doc/draft-ietf-dmarc-arc-protocol/?include_text=1, 2018, Accessed, Oct. 09 2018. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-dmarc-arc-protocol/?include_text=1

SRS

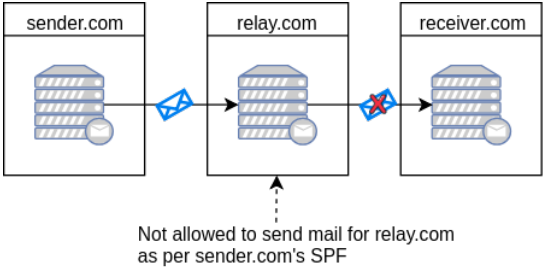

Figure 4: SPF breaking e-mail forwarding