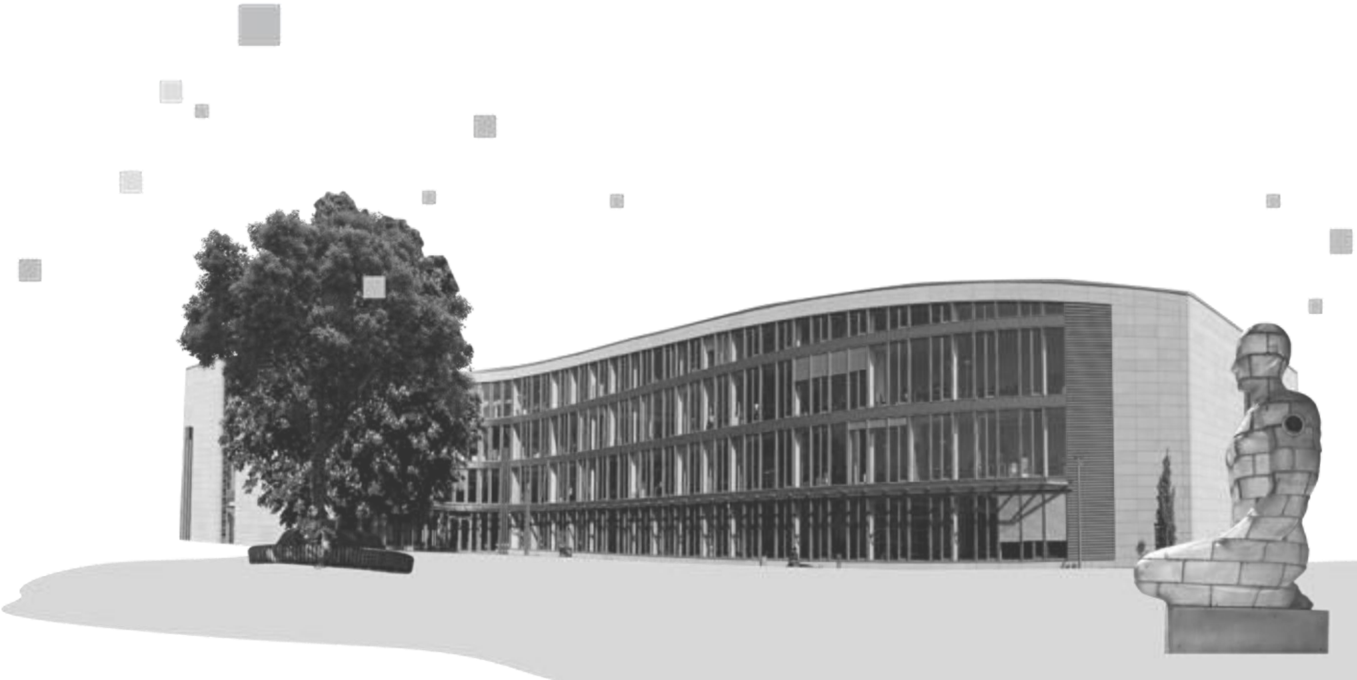
# Email Transport Security in Practice

## Email Client Selftest Service

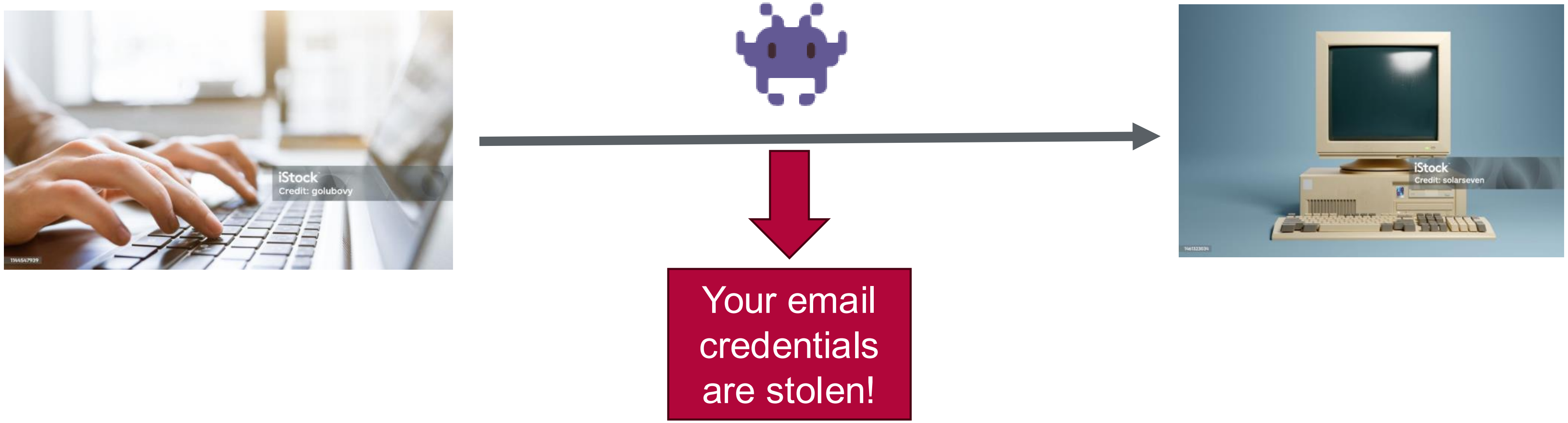
Jhannes Ernesto Reimann, Sofya Generalova



Design IT.
Create Knowledge.

www.hpi.de

# Imagine writing an email in 2025



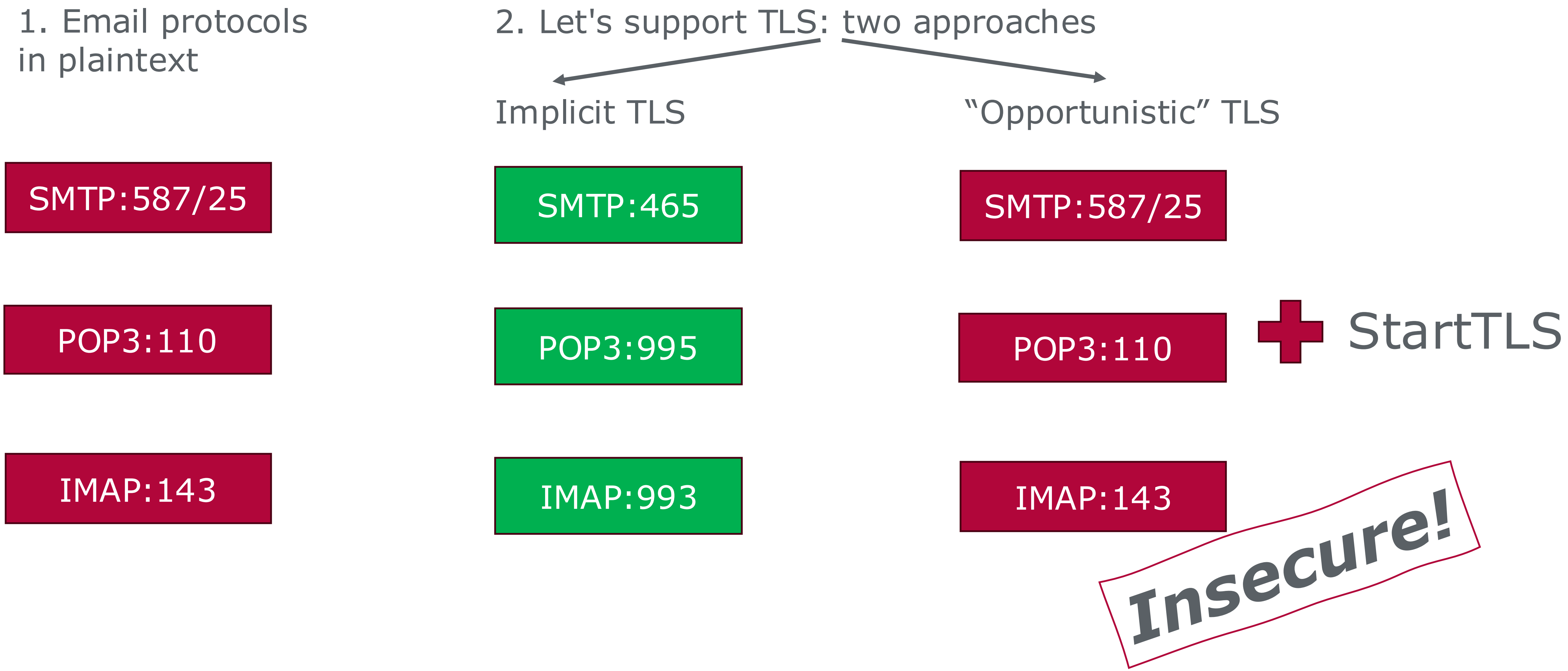
Your email credentials are stolen!

How Is That Possible?

# Table of contents

1. Vulnerability
2. Vulnerable servers
3. Vulnerable clients
4. Original paper's methodology of testing clients
5. Selftest Service
6. Demo
7. Future Directions

# Protocols: Historical context



1. Email protocols in plaintext

SMTP:587/25

POP3:110

IMAP:143

2. Let's support TLS: two approaches

Implicit TLS

SMTP:465

POP3:995

IMAP:993

“Opportunistic” TLS

SMTP:587/25

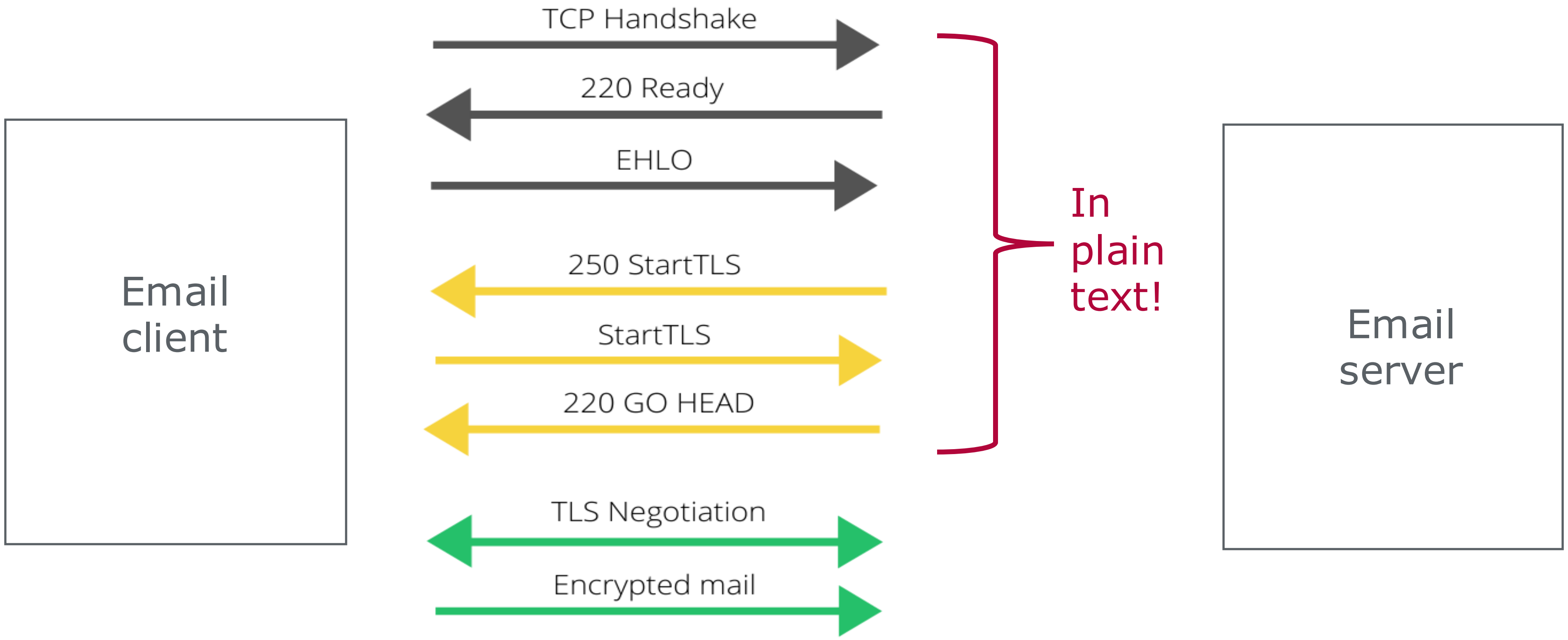
POP3:110

IMAP:143

StartTLS

Insecure!

# Opportunistic TLS



SMTP protocol with StartTLS

https://www.twilio.com/en-us/blog/insights/what-is-starttls

An active attacker on the network can strip StartTLS and force no-TLS

***if email server and client don’t explicitly disallow it***

if **email server** and client
don’t explicitly disallow it

# How many email servers would accept plaintext?

# Passive Measurement with Shodan

## Idea: scan Internet for email servers that advertise plain auth

SMTP example:

1. Client requests capabilities from server

```
EHLO test.com
```

2. Server lists capabilities

```
250-mail.nsipmail.de
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH PLAIN LOGIN
...
```

potentially effective before TLS → **250-AUTH PLAIN LOGIN**

**This does not guarantee server's vulnerability:**

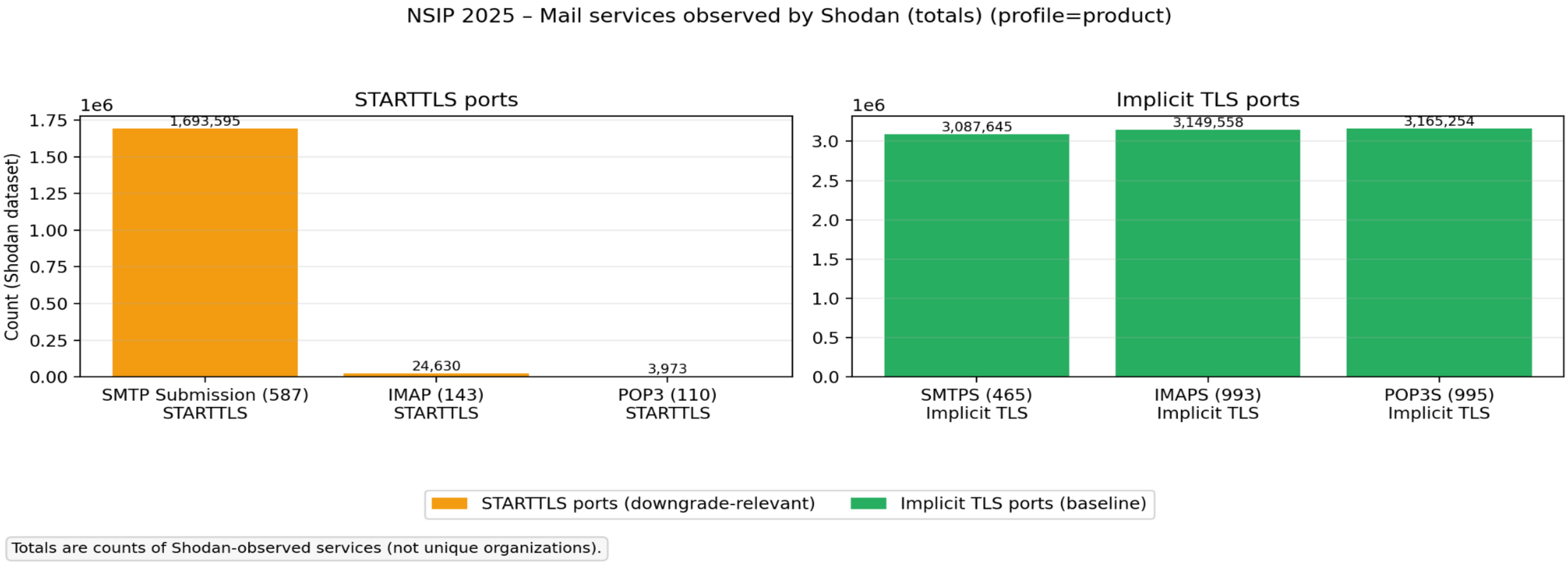
Client tries to login

```
C: AUTH PLAIN AGFsaWNlAHNlY3JldA==
```

Server rejects

```
S: 530 5.7.0 Must issue a STARTTLS command first
```

# Passive Measurement with Shodan: results



NSIP 2025 – Mail services observed by Shodan (totals) (profile=product)

Totals are counts of Shodan-observed services (not unique organizations).

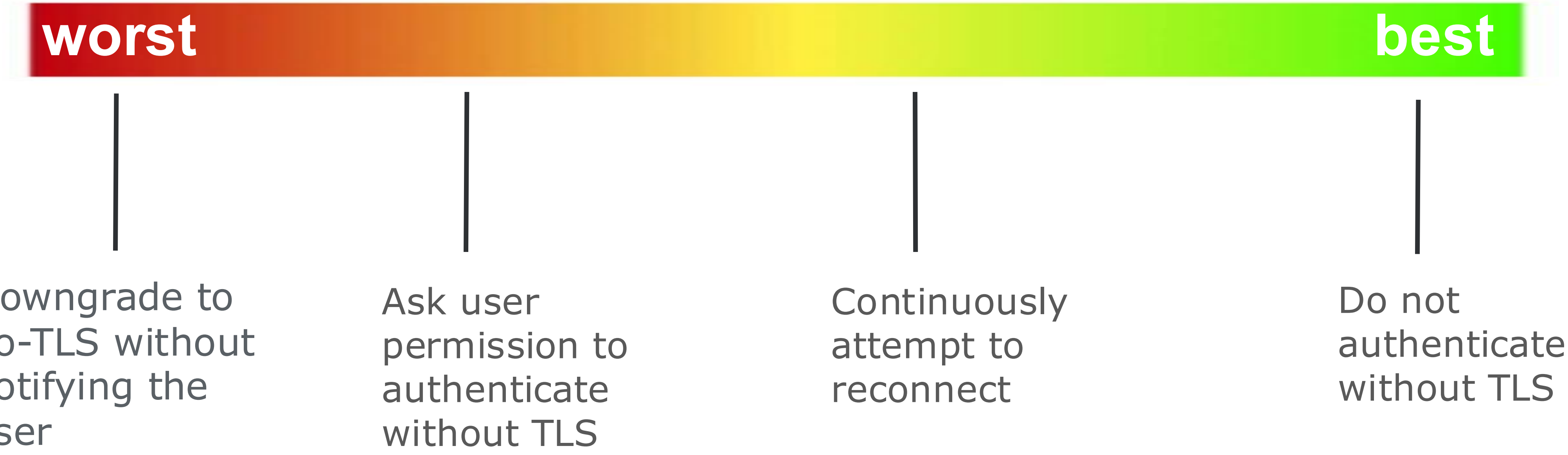
if email server and **client**
don’t explicitly disallow it

# How many email clients would authenticate in plaintext?

# A Multifaceted Study on the Use of TLS and Auto-detect in Email Ecosystems (2025)

## Email client behavior*:

*can differ depending on email protocol and TLS downgrade test case



worst

best

Downgrade to no-TLS without notifying the user

Ask user permission to authenticate without TLS

Continuously attempt to reconnect

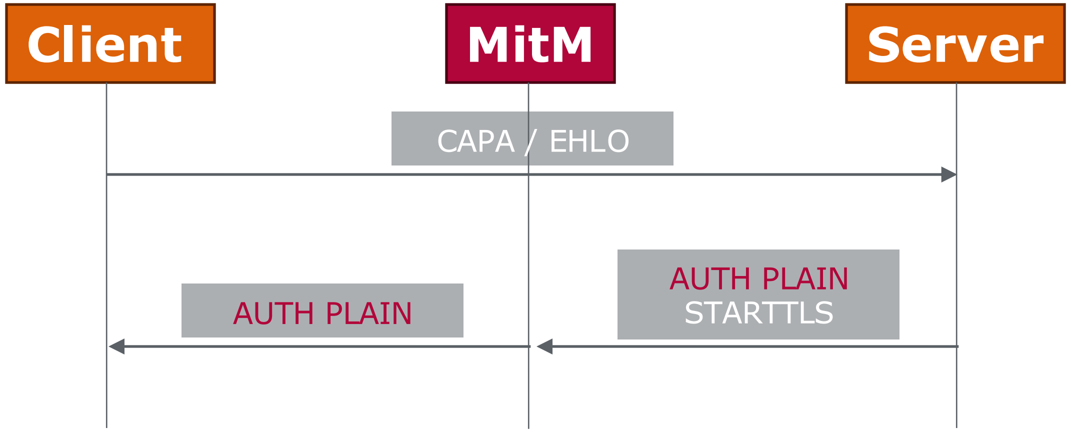
Do not authenticate without TLS

**19 out of 49 tested clients may silently downgrade to no-TLS**

# A Multifaceted Study on the Use of TLS and Auto-detect in Email Ecosystems (2025)

Testing email client behavior: 4 variations of disruptions

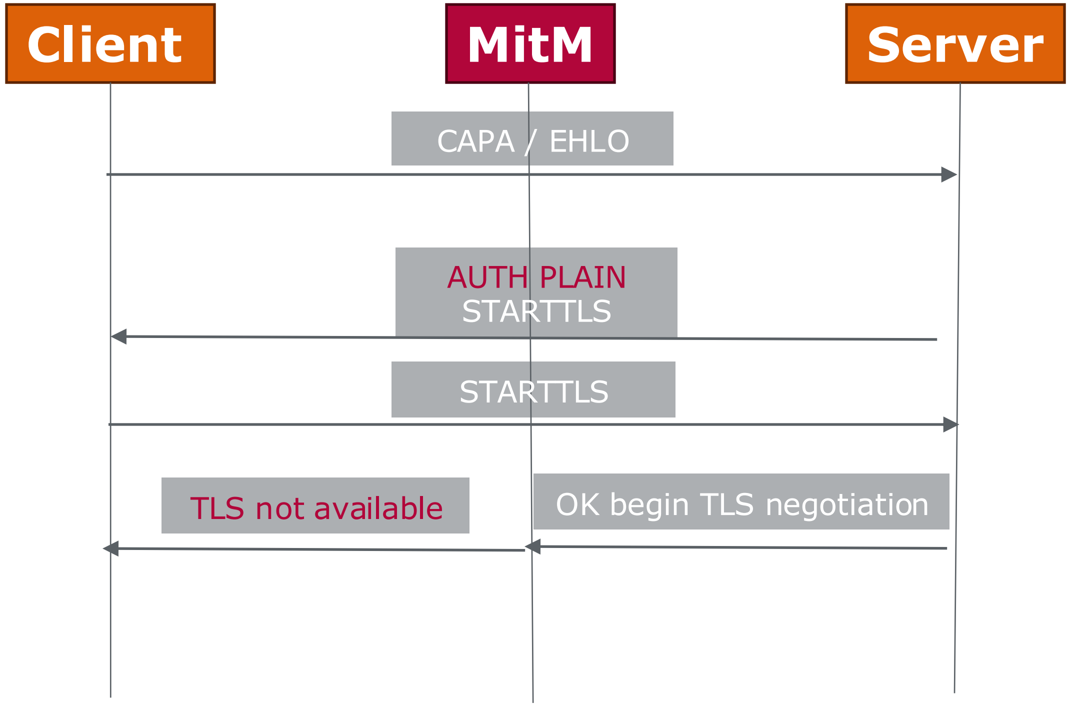
T1:
"classic" strip
StartTLS test



# A Multifaceted Study on the Use of TLS and Auto-detect in Email Ecosystems (2025)

## Testing email client behavior: 4 variations of disruptions

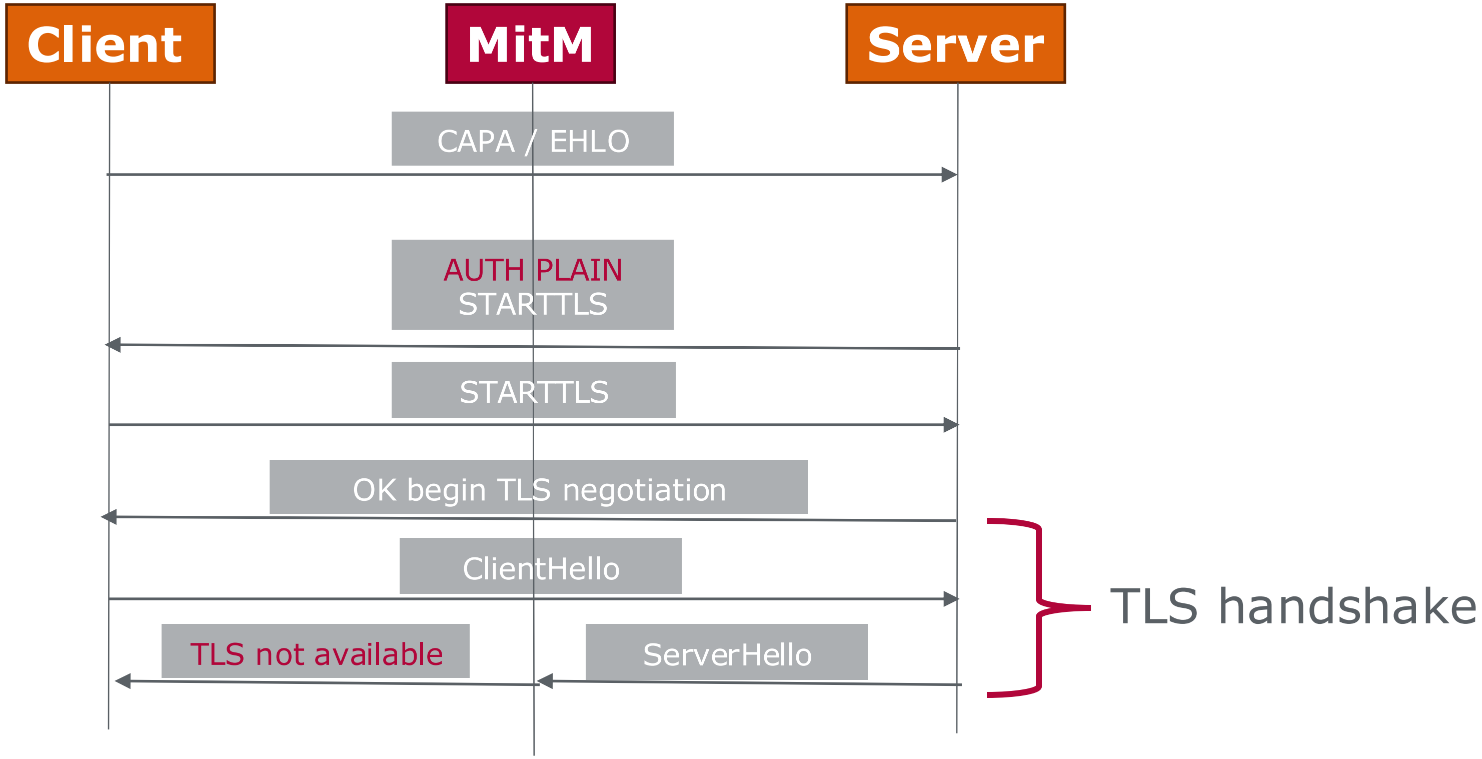
T3:
disrupt after client selects StartTLS



# A Multifaceted Study on the Use of TLS and Auto-detect in Email Ecosystems (2025)

Testing email client behavior: 4 variations of disruptions

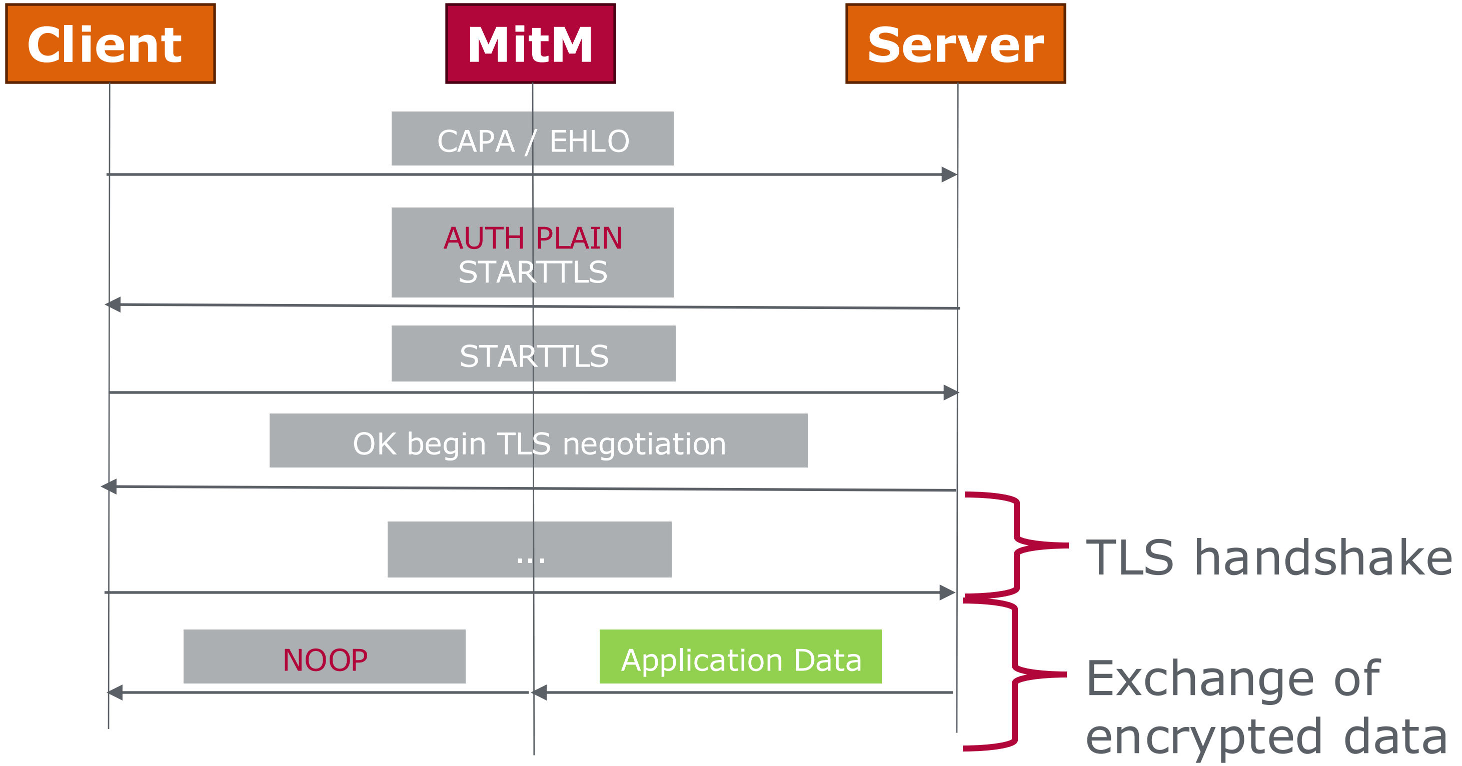
T2:
disrupt during
TLS handshake



# A Multifaceted Study on the Use of TLS and Auto-detect in Email Ecosystems (2025)

## Testing email client behavior: 4 variations of disruptions

T4:
disrupt
ongoing TLS
session

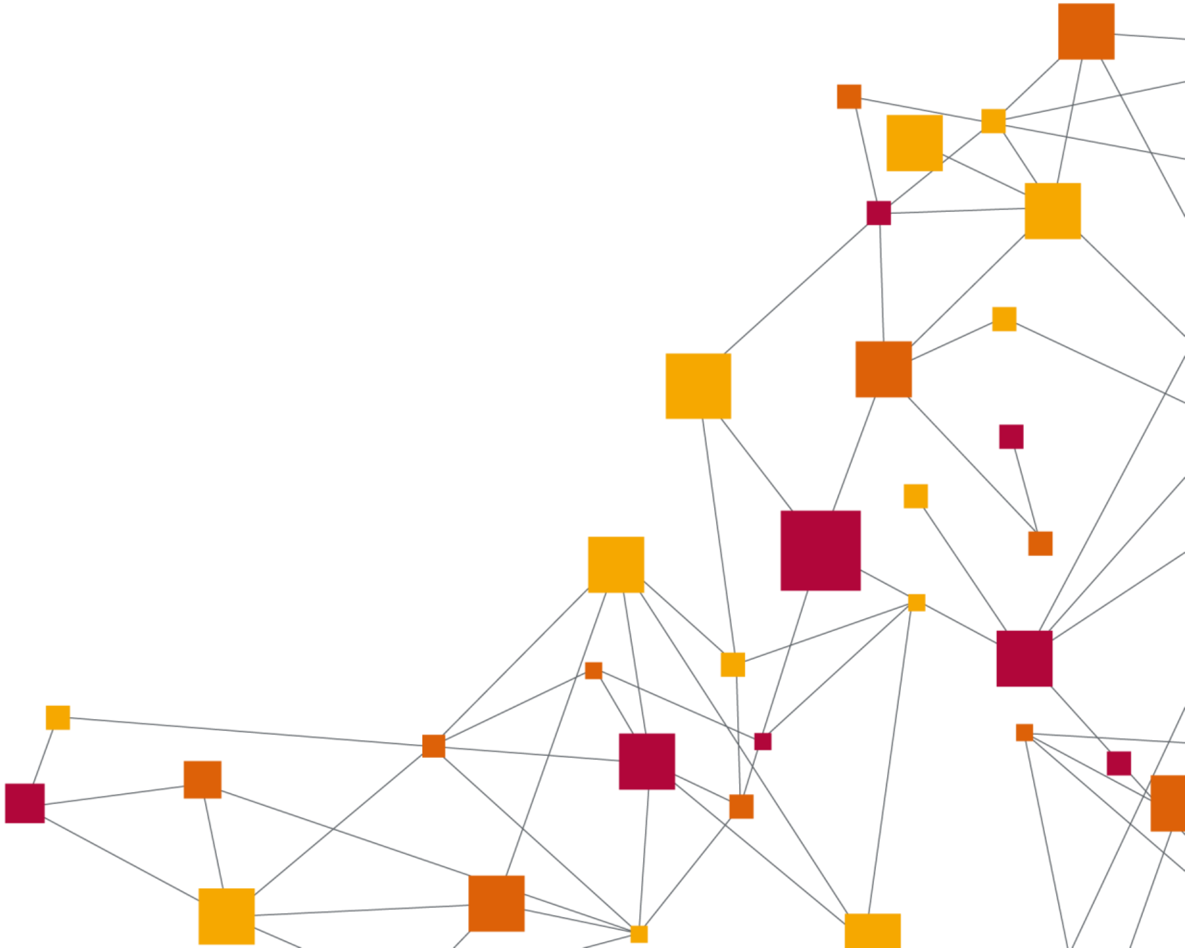


# Is your email client vulnerable?

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

# Selftest Service

A way to test your email client in use.



**Design IT.
Create Knowledge.**

www.hpi.de

# What did we build?

- Public, server-side **mail client self-test** for **SMTP + IMAP**
- Tests whether a client can be coerced into **plaintext authentication** when TLS protection is disrupted
- Implements **baseline + T1–T4** behaviors (paper-inspired)

# Why did we build it?

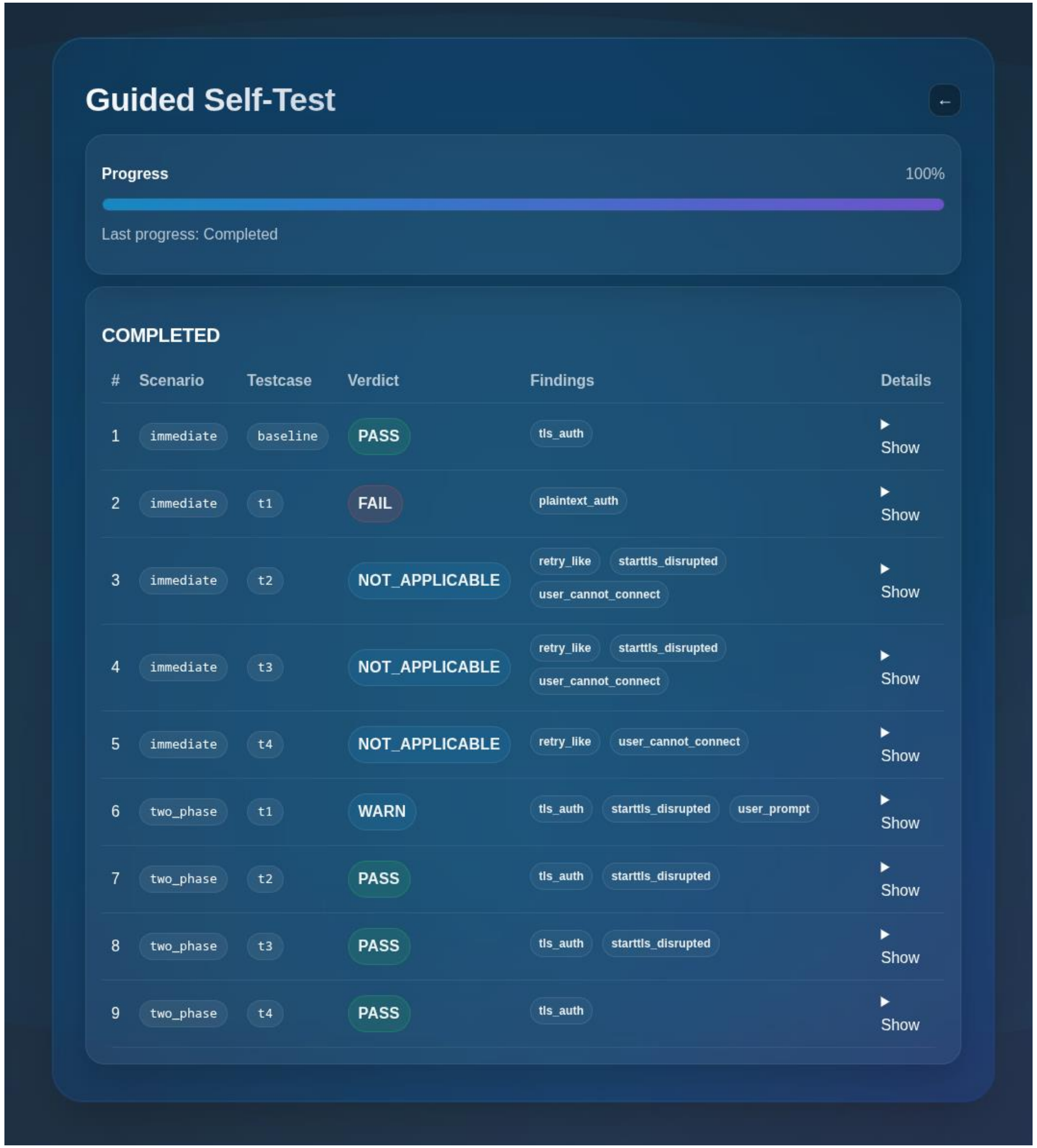
- Paper setup assumes a **MITM/lab environment** (hard to reproduce for normal users)
- We want a **“click** → **configure account** → **observe result”** workflow
- No local proxy, no custom CA, no special network setup required

# Demo

**Design IT.**
**Create Knowledge.**

**www.hpi.de**

# Result example



## Guided Self-Test

Progress 100%

Last progress: Completed

### COMPLETED

| # | Scenario | Testcase | Verdict | Findings | Details |
|---|---|---|---|---|---|
| 1 | immediate | baseline | PASS | tls_auth | ▶ Show |
| 2 | immediate | t1 | FAIL | plaintext_auth | ▶ Show |
| 3 | immediate | t2 | NOT_APPLICABLE | retry_like<br>starttls_disrupted<br>user_cannot_connect | ▶ Show |
| 4 | immediate | t3 | NOT_APPLICABLE | retry_like<br>starttls_disrupted<br>user_cannot_connect | ▶ Show |
| 5 | immediate | t4 | NOT_APPLICABLE | retry_like<br>user_cannot_connect | ▶ Show |
| 6 | two_phase | t1 | WARN | tls_auth<br>starttls_disrupted<br>user_prompt | ▶ Show |
| 7 | two_phase | t2 | PASS | tls_auth<br>starttls_disrupted | ▶ Show |
| 8 | two_phase | t3 | PASS | tls_auth<br>starttls_disrupted | ▶ Show |
| 9 | two_phase | t4 | PASS | tls_auth | ▶ Show |

# Possible Verdicts

- **PASS**: Auth/login happened only with TLS (no plaintext credentials seen).
- **FAIL**: Auth/login happened without TLS (plaintext credentials exposure).
- **INCONCLUSIVE**: No auth/login observed (client aborted / got stuck / never reached auth).
- **WARN**: Client showed a security prompt/downgrade warning (user-reported; no plaintext proven).
- **NOT_APPLICABLE**: Test step couldn’t be executed (cannot connect; user-reported).
- **SKIPPED**: Step was skipped in Guided mode

# Possible Automatic Findings

- **plaintext_auth**: We observed an auth/login attempt without TLS (tls=false).
- **tls_auth**: We observed an auth/login attempt with TLS (tls=true).
- **retry_like**: Many reconnects / retries but no auth seen (client appears stuck).
- **starttls_disrupted**: STARTTLS was refused / failed / dropped (e.g., “TLS not available”, handshake wrap failed).

# Why we simulate (and why we don’t run a proxy/MITM)

**Why no proxy on the client:**

- as little setup as possible

**Why no proxy on the server (we simulate instead)**

- For our research question, what matters is the **client’s view / decision point**

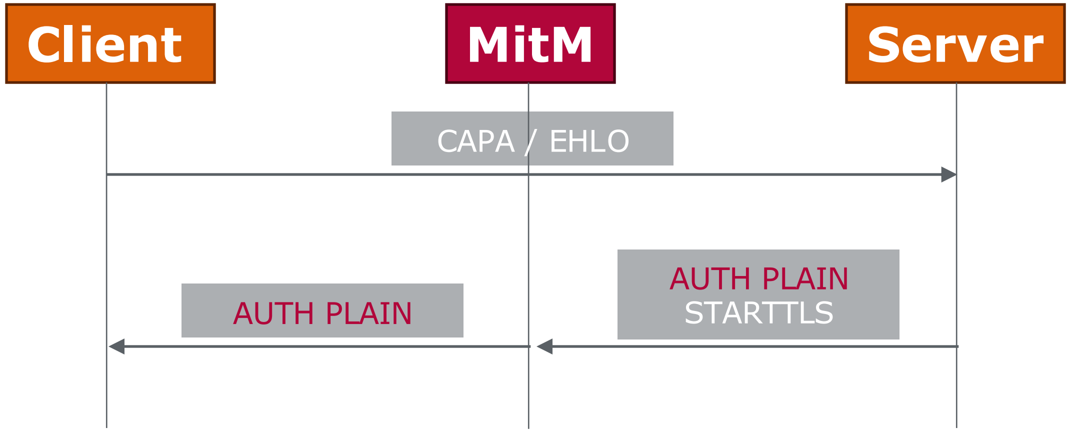
  A test server can create the **same observable conditions** by responding “as if a MITM was present”

  Result: we can still measure whether the client ever sends **AUTH/LOGIN without TLS**

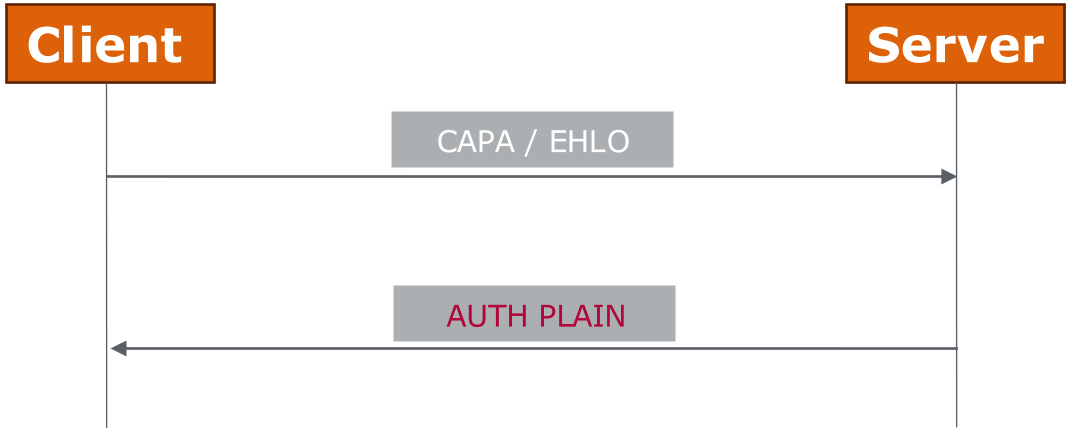
**Could we still do it with a server-side proxy?**

- Build a **relay proxy**: client → proxy → real upstream mail server

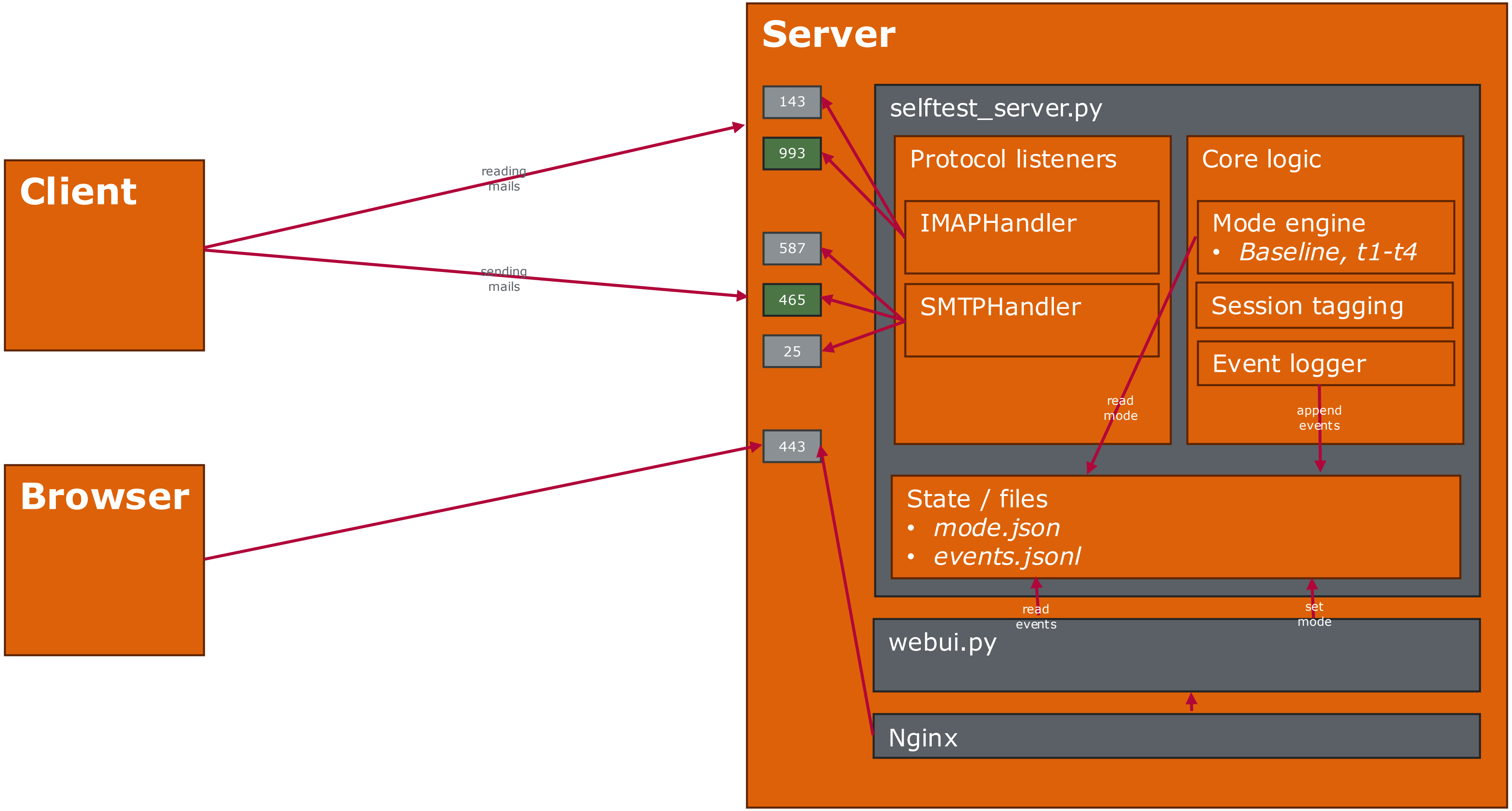
# "Simulation" Diagram of T1



# "Simulation" Diagram of T1



# Selftest-Service architecture



# Why no POP3?

- User friction & configuration effort
- Result: adding POP3 would likely reduce completion rate for a public self-test
- For the initial service, IMAP + SMTP submission covers the most common real-world configurations

# Future directions

**Selftest-Service:**

- POP3 extension
- Stronger session isolation + user accounts
- Lab-grade “real MITM” + real mailbox

**Admin/server-side test tool**

- CLI/script for mail admins to test their own SMTP/IMAP/POP3 configuration against T1–T4 downgrade conditions

**Manual vs autodetect behavior**

- Systematically investigate differences between manual configuration and autodiscovery-driven configuration paths.