テキストメッセージから生成AIへ
～メールの歴史とセキュリティの進化をたどる～

(前半)誰も取り残さないメール・メールセキュリティ技術解説



JPAAWG 9th General Meeting　2026年11月5日

株式会社TwoFive 小嶋 彬彦

自己紹介

小嶋彬彦 (株式会社TwoFive 開発エンジニア)

2018年～

長崎県立大学(学部・修士)

情報セキュリティを専攻
卒業研究・修士論文ではフィッシングURL検知についての論文を執筆

2024年～

株式会社TwoFive

業務では自社製品の開発・運用
自社環境の監視や関連情報の調査にも携わる

© TWOFIVE ALL RIGHTS RESERVED.

目次

1. はじめに
2. メール送受信の仕組み
3. メールの誕生
4. 電子メールとして標準化
5. インターネットの普及に伴うメールの進化
6. SPAMの増加とセキュリティ技術の導入
7. 送信ドメイン認証の登場
8. メールセキュリティの現在地
9. 前半まとめ

© TWOFIVE ALL RIGHTS RESERVED.

はじめに

この発表はメール初心者・学生さんなどを対象としています

- **メールって歴史の長い技術だから難しそう**
- **メールの方が自分より年上だからよく分からない**
- **ITエンジニアだけどメールはちょっと苦手意識がある**
- **メールを普段使っているがどのような仕組みで動作しているか知らない**

といった方が対象です

初学者の方にメール技術の大枠を理解していただく事を優先し、技術の詳細を省略・簡略化している部分があります
温かい目で見守っていただけますと幸いです

© TWOFIVE ALL RIGHTS RESERVED.

メール送受信の仕組み

メールに影響を与えた現実の郵便システム

- メールの仕組みは現実世界の郵便システムによく似ている

現実の郵便	メール
手紙を書く人	メール送信者
郵便局	メールサーバ
郵便の配送	SMTPによる配送
封筒の宛先	エンベロープTo(配送に使用される宛先)
封筒の差出人	エンベロープFrom(返送に使用される差出人)
便箋に書かれた差出人	ヘッダーFrom(データとしての差出人)

© TWOFIVE ALL RIGHTS RESERVED.

メール送受信の仕組み

メールはどのようにして受信者のもとに届くのか

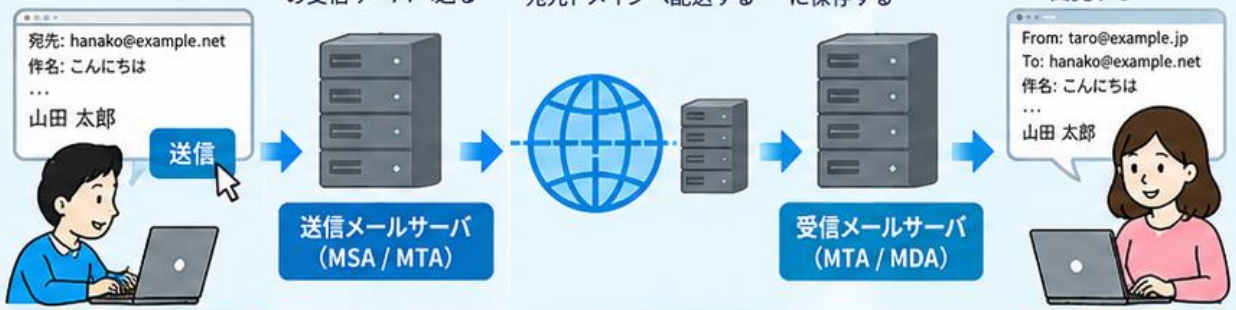
現実の郵便システム
（手紙を投函してから相手に届くまで）

① 差出し	② 集荷・引受	③ 輸送	④ 仕分け	⑤ 配達	⑥ 受取
手紙を書き、封筒に入れて切手を貼り、ポストに投函する	郵便局がポストから集荷し、差出元の郵便局で引き受ける	トラックや飛行機で宛先の地域にある郵便局まで輸送する	宛先の地域の郵便局で郵便番号や宛先に基づいて配達先ごとに仕分ける	配達員が各家庭のポストに配達する	受取人が自宅のポストから手紙を受け取る
差出人（手紙を書く人）	差出元の郵便局（引受局）	郵便の輸送	宛先の郵便局（仕分け局）	配達	配取人（手紙を受け取る人）
封筒の宛名と便箋の宛名を記載しポストに投函する	集めた郵便物をバーコードで処理し次の拠点へ送る	陸・空の輸送で大量の郵便物を宛先地域まで運ぶ	郵便番号・宛先に基づいて配達エリアごとに仕分ける	配達員が担当エリアの各家庭に配達する	自宅のポストで手紙を受け取る



メールの仕組み
（メールを送信してから相手に届くまで）

① メール作成・送信	② 送信サーバ（MSA）	③ インターネット（SMTPで配送）	④ 受信サーバ（MTA/MDA）	⑤ 受信・閲覧
メールソフトでメールを作成し、送信ボタンを押す	認証を行い、送信者のメールを受け取り、宛先ドメインの受信サーバへ送る	SMTPでメールをインターネット上の複数のサーバを経由して宛先ドメインへ配送する	宛先ドメインのメールサーバが受け取り、受信者のメールボックスに保存する	メールソフトやWebメールでメールを受け取り、閲覧する
送信者（メールを送る人・MUA）	送信者の認証（SMTP AUTH）を行いメールを受け取り、宛先ドメインのMTAへ送信する	DNSで宛先ドメインのMXレコードを参照し、複数のメールサーバを経由して配送する	受信したメールをウイルスチェック等の処理後、受信者のメールボックスに保存する	受信者（メールを受け取る人・MUA）
メールソフトでヘッダーや本文を作成し、送信サーバへ送信する				メールソフトやWebメールでメールを取得し、閲覧する（IMAP/POP）



© TWOFIVE ALL RIGHTS RESERVED.

メールの歴史

メールの誕生

- **1960年代 TSS(Time Sharing Service)上でユーザ間のメッセージ交換が始まる**
  一つのコンピュータを共同利用するTSSにおいてユーザ毎のメールボックスを用意し、そこへ他のユーザがメッセージを書き込んでコミュニケーションをとり始める
- **1969年 ARPANET開始**
  インターネットの前身となるARPANETが運用を開始
- **1971年 ネットワークを介した電子メールの登場**
  Raymond Samuel Tomlinsonが@で区切る形式のメールアドレスを用いて世界で初めて異なるコンピュータ間での電子メールを実現
- **1978年 最初期のスパムメール登場**
  とある企業が製品発表会の案内を行うために宣伝メールを400人ほどに送信



© TWOFIVE ALL RIGHTS RESERVED.

メールの歴史

電子メールとして標準化

- **1982年 SMTP・メッセージ形式の標準化**
  メール送信・配送に使用されるプロトコルSMTPが標準化(RFC821)、
  実際に送信されるメールデータの形式も標準化された(RFC822)
- **1984年 POPの標準化**
  メールの受信に利用されるプロトコルPOPが標準化(RFC918)
- **1986年 DNSを用いたメール配送技術の標準化**
  DNSに配送先を問い合わせる仕組みとしてMXレコードやその取り扱いが標準化
  (RFC 974)
- **1988年 IMAPの標準化**
  受信したメールをサーバ上で同期・管理するプロトコルIMAPが標準化(RFC1064)

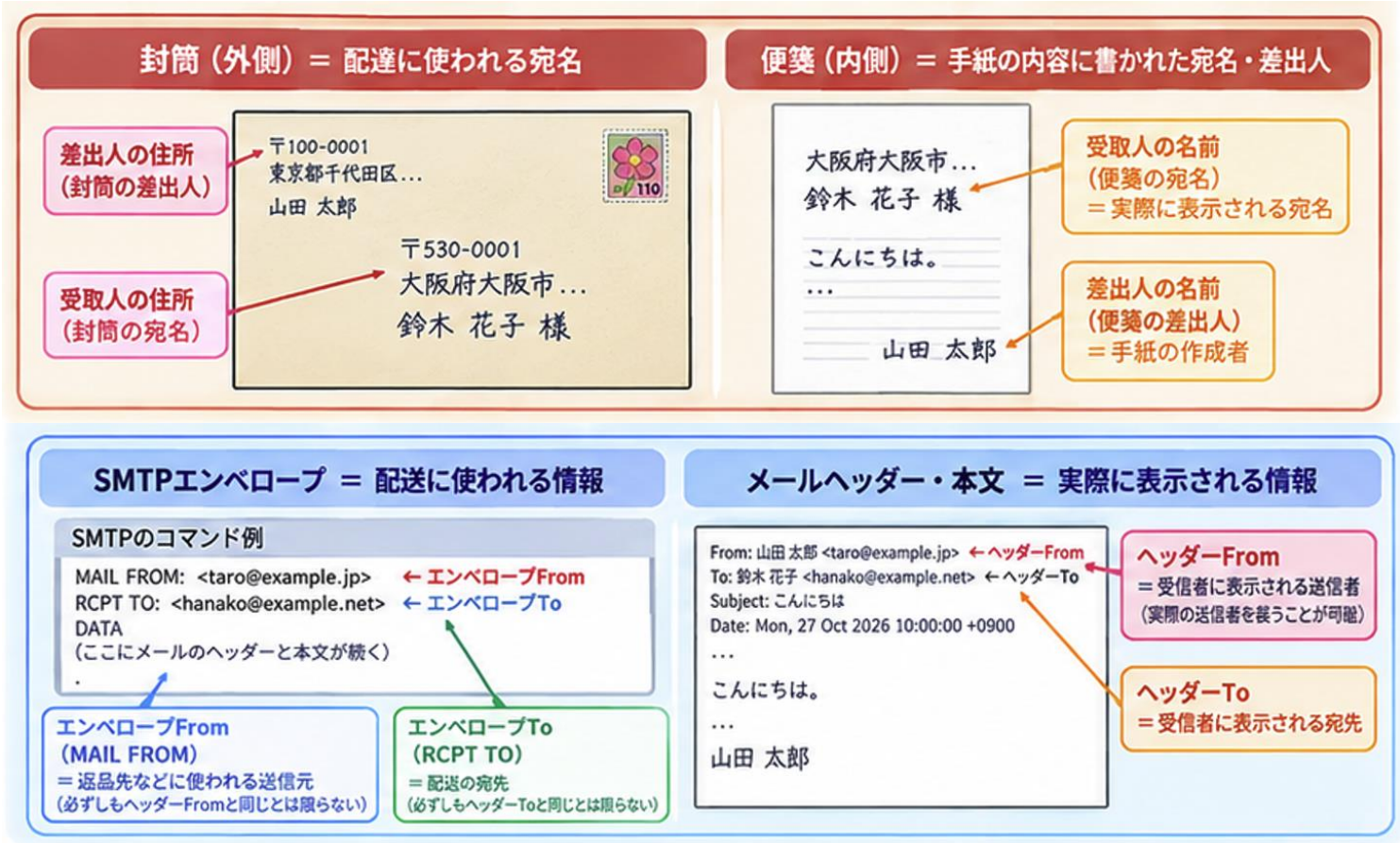
メールの普及に伴い、メールで使用する各種プロトコルが標準化された
この時点でのSMTPはメールを送信・配送する機能がメインであり、通信内容の暗号化や送信者の認証といった機能は標準で備わっていなかった

© TWOFIVE ALL RIGHTS RESERVED.

メール技術解説1

SMTPについて

- メールには宛名や差出人を示すデータが2箇所に存在する
- 配送に使うエンベロープFromと受信者に表示されるヘッダーFromは標準で一致する必要がないため、ヘッダーFromに他人のアドレスを記載する事で簡単になりすましを行うことができてしまう

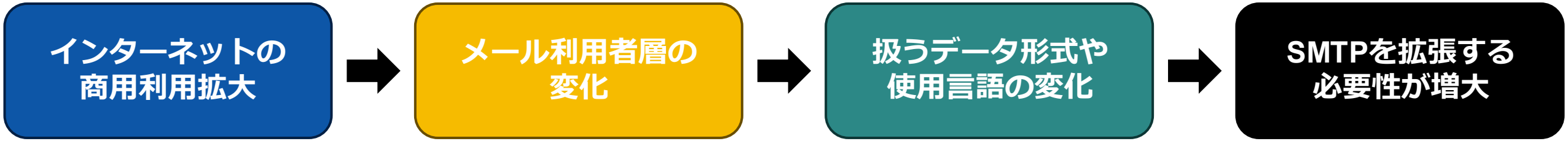


© TWOFIVE ALL RIGHTS RESERVED.

メールの歴史

インターネットの普及に伴うメールの進化

- **1992年 MIME(Multipurpose Internet Mail Extensions)の標準化**
  メールの表現形式を拡張するMIMEが標準化(RFC1341)
  非ASCII文字の使用や添付ファイル、多様なデータをメールで扱うことが可能に
- **1993年 ESMTPの標準化**
  SMTPの機能を拡張する仕組みとしてESMTPが標準化(RFC1425)
  従来のSMTPとの互換性を維持しつつ機能追加が可能に
- **1990年代後半 Webメールの登場と普及**
  インターネット接続とブラウザのみで利用できるWebメールが登場・普及



© TWOFIVE ALL RIGHTS RESERVED.

メールの歴史

SPAMの増加とセキュリティ技術の導入

- **1990年代後半~2000年代初頭 SPAMメールが急増**
  SMTPに送信者認証機能がなく、TCP/25に直接接続すると誰でもメールが送信可能
- **1998年 TCP/587サブミッションポートの標準化**
  メールサーバ間のメール配送で使用するポートと利用者がメール投稿に使用するポートを分離し、TCP/587ポートをメール投稿用として標準化(RFC2476)
- **1999年 SMTP STARTTLS、SMTP AUTHの標準化**
  SMTPを拡張し暗号化通信に対応したSMTP STARTTLSを標準化(RFC2487)
  メールの送信時にメールサーバがクライアント(送信者)を認証するSMTP AUTHが標準化(RFC2554)
- **2000年代前半 OP25Bの導入**
  一般利用者向け回線から直接TCP/25に接続しメールを送信することを防ぐためにISPや通信事業者が外部のTCP/25への接続を制限するOP25Bを導入

SMTP AUTHによる利用者の認証やSTARTTLSによる通信内容の暗号化、OP25Bによる不正な直接送信の制限が導入され、不正なメール送信や内容の盗聴を防止

© TWOFIVE ALL RIGHTS RESERVED.

メールの歴史

送信ドメイン認証の登場

- **2006年 SPFの実験的標準公開**
  メールの送信元IPアドレスがエンベロープFromのドメインによって許可されているIPアドレスであるかを確認して認証するSPFが実験的に標準化(RFC4408)
- **2007年 DKIMの標準化**
  メールに電子署名を付与し受信時に検証する事で内容が改竄されていないことや署名に使用されたドメインを確認するDKIMが標準化(RFC4871)
- **2015年 DMARCの標準化**
  SPF・DKIMで認証した結果とヘッダーFromが一致することを検証してなりすましを防ぎ、認証失敗時に指定した取り扱いを要求するDMARCが標準化(RFC7489)

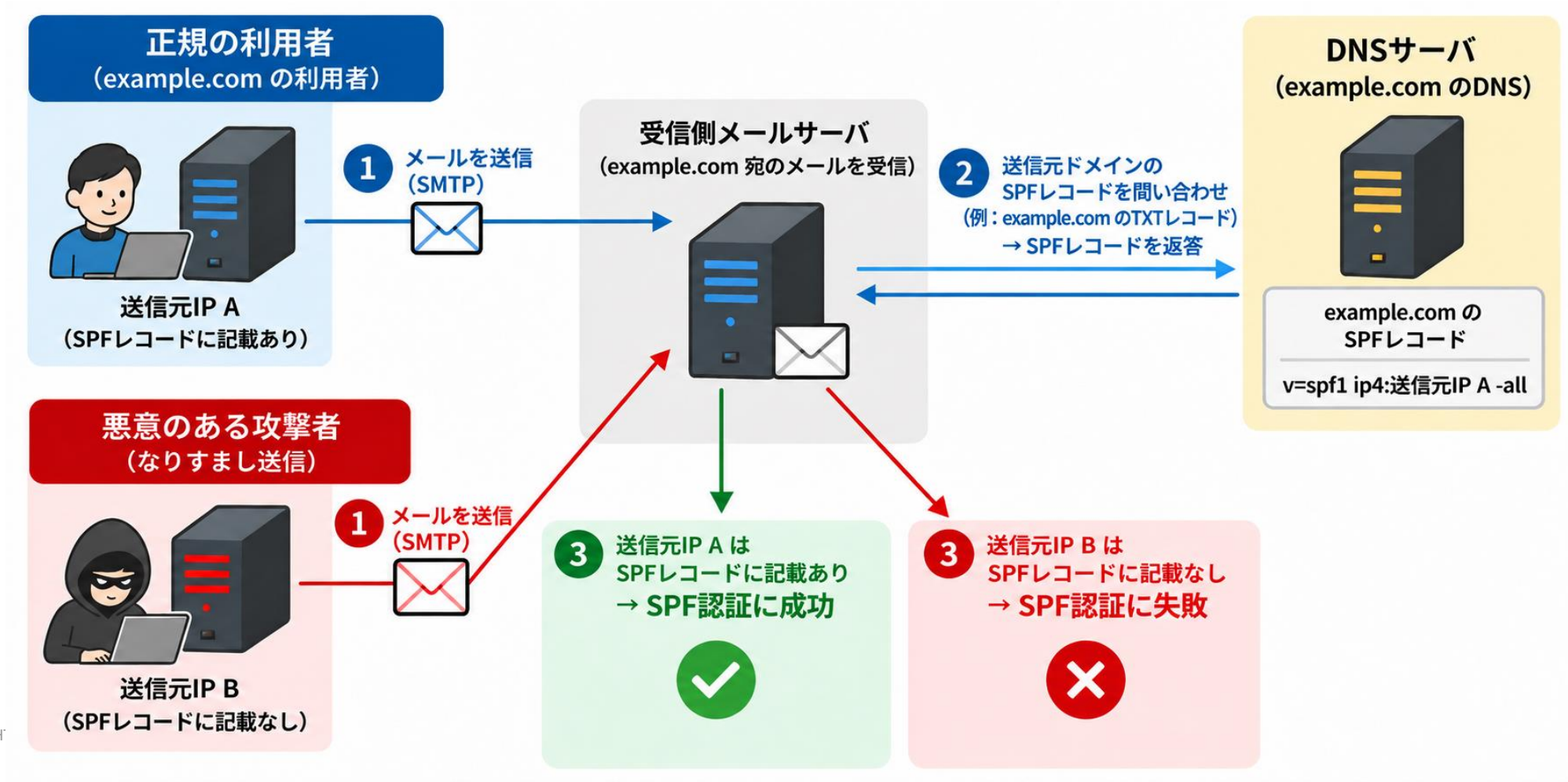
なりすましメールやフィッシングメールの出現により、受信者がメールに記載された送信者情報の正当性を検証する必要が生じた。
これを受けて送信元IPアドレスの送信許可を確認するSPF、署名対象ドメインの管理する鍵で署名され、署名対象の内容が改ざんされていないことを検証するDKIM、二つの結果とヘッダーFromの整合性を検証するDMARCが導入された

© TWOFIVE ALL RIGHTS RESERVED.

メール技術解説2

SPFについて

- 受信者は送信元IPアドレスをエンベロープFromのSPFレコードに照会する事で
  そのドメインにおいてメール送信が許可されている事を検証することが可能となる

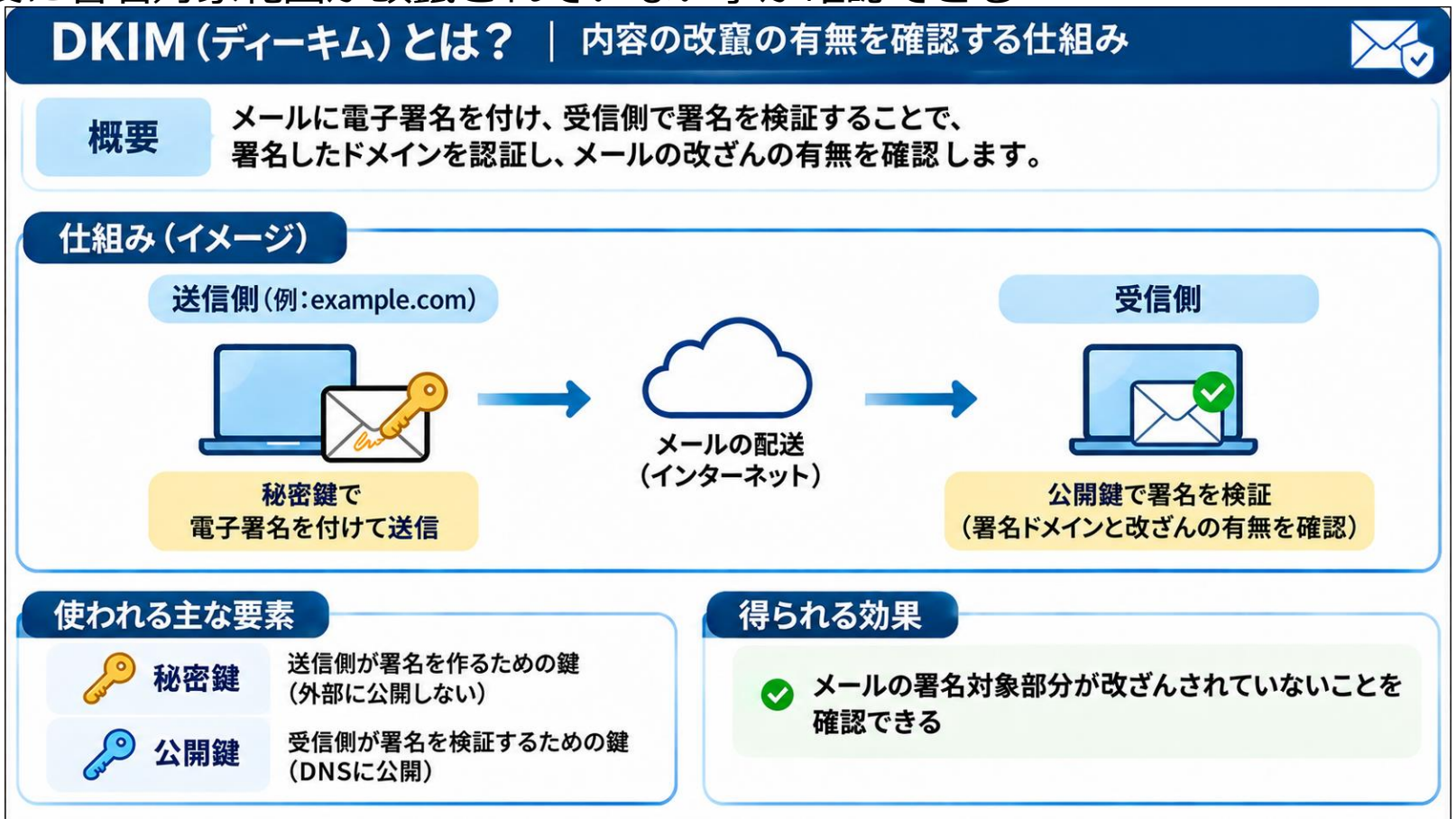


© TWOFIVE ALL RIGH

メール技術解説3

DKIMについて

- 送信するメールに対して送信者が秘密鍵で電子署名を付与する
- 受信時にDKIMヘッダーのdタグに記載されたドメインの公開鍵を用いて署名を検証し、受信したメールが署名に使用されたドメインによって作成されたものである事、作成後に署名対象範囲が改竄されていない事が確認できる



© TWOFIVE ALL RIGHTS RESERVED.

メール技術解説4

DMARC誕生の経緯

- SPF・DKIMによって送信元の以下の要素が検証可能になった
    - 送信元IPアドレスがドメイン管理者によってメール送信を許可されている事
    - DKIMヘッダに記載されたドメインによって作成され、署名対象範囲については改竄されていない事
- 攻撃者がSPF、DKIMを設定した状態で**ヘッダFromを詐称してのなりすましは可能**
- SPF・DKIMが**失敗した場合のメールの取り扱いは受信側に一任**されており、なりすまされた側にはどうすることもできなかった
- SPF・DKIMは設定を誤ると正規のメールでも認証が失敗し、受信側の判定によってはメールが届かなくなる恐れがあり、認証結果をドメイン所有者が確認できる必要性が高まった

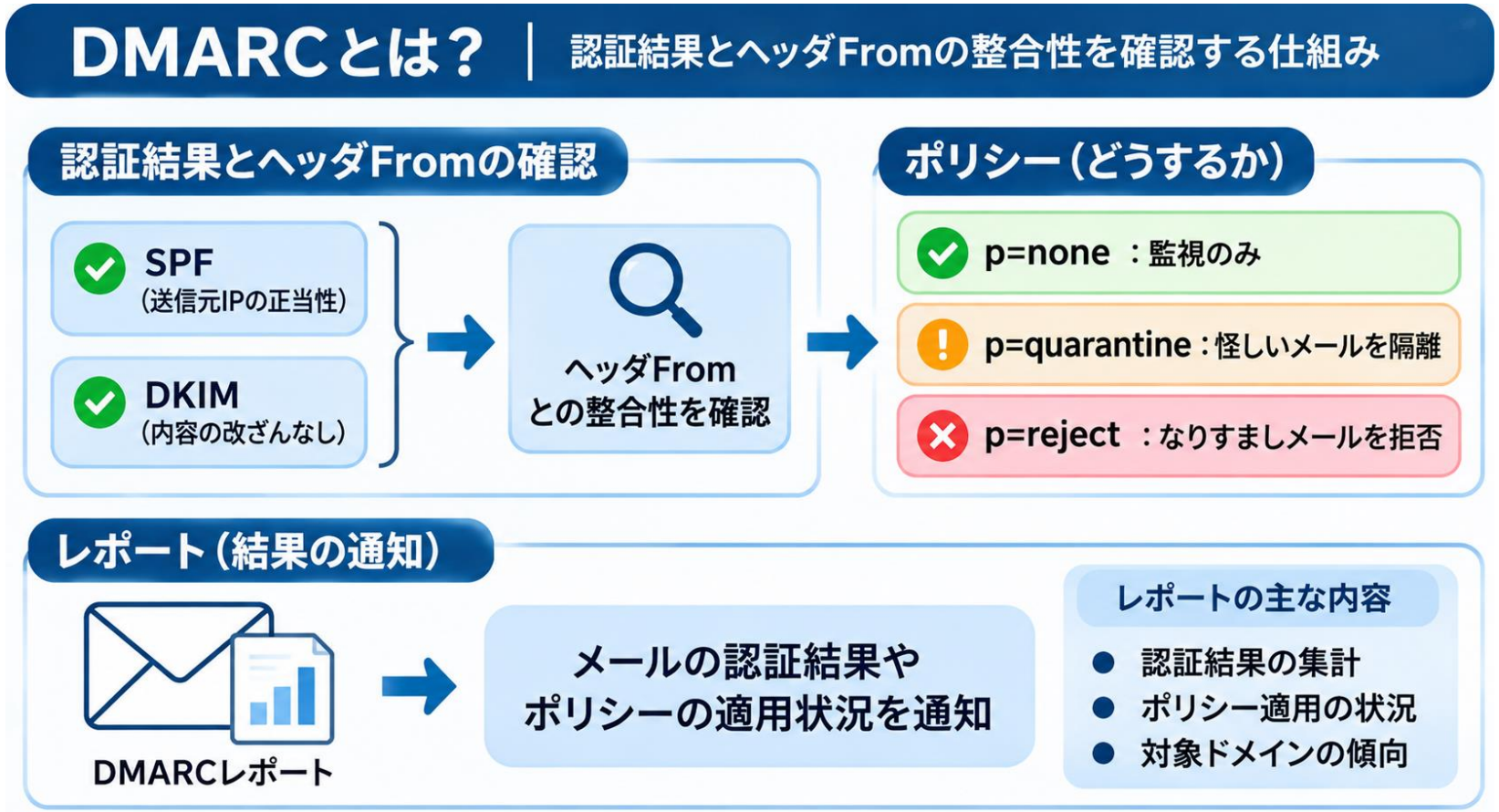
上記のような背景からSPF・DKIMが認証したドメインとヘッダFromのドメインの整合性を確認し、失敗時の取り扱い方針の公開とドメインのメール送信状況の把握に対応したDMARCが標準化されるに至った

© TWOFIVE ALL RIGHTS RESERVED.

メール技術解説4

DMARCについて

- DNSにポリシーとレポート送信先アドレスをDMARCレコードとして記載する
- 受信者はDMARCレコードを参照し認証失敗時には記載されたポリシーに従ってメールを取り扱う(ポリシーに従うことは強制ではない)



© TWOFIVE ALL RIGHTS RESERVED.

メールの歴史

メールセキュリティの現在地

- フィッシングメールの観測数は依然高い水準であり、被害も出続けている
- 送信ドメイン認証の普及によりフィッシングメールのトレンドは適当に取得したドメインにSPF・DKIM・DMARCを設定して認証をパスさせる手法になりつつある
- このような手法については送信ドメイン認証では防御が難しいため、別の技術やアプローチでの防御を行う必要がある
- 既存の送信ドメイン認証においても正しく設定をしていないことによる認証のすり抜けなどが発生している
- フィッシングメールの文面についても生成AIの発達により言語の壁はなくなり、**文面からフィッシングを見分けることは極めて難しくなりつつある**

© TWOFIVE ALL RIGHTS RESERVED.

前半まとめ

コンピュータ共用の時代から続いてきたメール技術がいよいよAI時代に突入

- 同じコンピュータ、ネットワーク利用者同士の情報交換から始まったメール技術は時代に適応しつつ様々なアップデートを続けている
- 時代と共に様々な機能・要素が追加され、それに伴い**メールを触る上で求められるハードルも上昇し続けている**
- 今後は生成AIの台頭などによりさらに技術の発展速度が上がることが予想される
- このあとは初学者向けに生成AIを用いて簡単にメールサーバを立てる方法のご紹介

© TWOFIVE ALL RIGHTS RESERVED.

テキストメッセージから生成AIへ
～メールの歴史とセキュリティの進化をたどる～

後半：生成AIをアシスタントに、メールサーバを作ってみる
Claude × Stalwart での構築と検証



JPAAWG 9th General Meeting　2026年11月5日

株式会社TwoFive 高崎 俊太郎

自己紹介

高崎 俊太郎（株式会社TwoFive）

2007年～
通信会社
ネットワークサービスや大規模SI案件で、提案・設計・構築・サポートを幅広く担当

その後 10年以上
法人向けメールサービス
メールサービスの開発・運用と、チームマネジメントに従事

2025年～
株式会社TwoFive
自社製品の開発・運用に加え、プリセールス/コンサルティングに従事

学習ツールの変遷（イメージ）

アイコンの大きさ＝使った比重、検索＝インターネット検索



© TWOFIVE ALL RIGHTS RESERVED.

目次

1 **背景と問い**
学び始める入り口が狭い中で、生成AIは助けになるか

2 **使ったソフトウェアと生成AI**
Stalwart、Claude＋Claude Code、これまでの構成との違い

3 **検証環境と進め方**
受信できない環境での検証の分担、任せる範囲と承認

4 **結果**
送信側と受信側の認証、作業の分担と時間

5 **落とし穴と人の判断**
実際にあった落とし穴、誤りの止まり方と減らし方、人が引いた線

6 **得られた知見と今後**
問いへの答えと、生成AIとの関わり方

© TWOFIVE ALL RIGHTS RESERVED.

背景

求められる水準は上がる一方、初心者・若手が学び始める入り口は狭くなっている

水準が上がった

大手メールサービスの送信者ガイドライン以降、SPF・DKIM・DMARCが整っていないメールは届きにくくなった

覚えることが多い

送受信の仕組み、DNS、TLS、送信ドメイン認証、スパム対策。何から手を付ければよいか分かりにくい

学べる場が少ない

メールサーバを一から組む機会も、教えられる経験者も減り、初心者・若手エンジニアが参入しづらい

© TWOFIVE ALL RIGHTS RESERVED.

このセッションの問い

生成AIをアシスタントにすれば、初心者・若手エンジニアや少人数のチームでも、“ちゃんとした”メールサーバを構築・検証できるか？

正しさ

AIの答えを確かめられるか

設定やコマンドが正しいかを、公式情報とテストで確かめる。SPF・DKIM・DMARCの判定が予想どおりに付くかを見る

安全さ

秘密情報や設定を守れたか

パスワードや鍵をAIとの会話に渡さない。変更は承認してから実行し、いつでも戻せる状態を保つ

人の判断

どこで人が線を引いたか

AIに任せる範囲と人が決めることを分ける。受信側のポリシーなど、正解が1つでない判断を誰がしたかを見る

© TWOFIVE ALL RIGHTS RESERVED.

目次

1 **背景と問い**
学び始める入り口が狭い中で、生成AIは助けになるか

2 **使ったソフトウェアと生成AI**
Stalwart、Claude＋Claude Code、これまでの構成との違い

3 **検証環境と進め方**
受信できない環境での検証の分担、任せる範囲と承認

4 **結果**
送信側と受信側の認証、作業の分担と時間

5 **落とし穴と人の判断**
実際にあった落とし穴、誤りの止まり方と減らし方、人が引いた線

6 **得られた知見と今後**
問いへの答えと、生成AIとの関わり方

© TWOFIVE ALL RIGHTS RESERVED.

使ったソフトウェアと役割

メールサーバはStalwartで構成し、生成AIはClaude＋Claude Codeを使った

v0.16.27

Stalwart

Rust製のオールインワン型メールサーバ。SMTP・IMAP・JMAPに加え、SPF・DKIM・DMARC・ARC、MTA-STS、スパムフィルタまでを1つで備える

生成AI

Claude＋Claude Code

Claudeで調査と設計の相談、Claude Codeで設定とスクリプトの作成・実行、テストの設計と実行、ログの解析までを担当

© TWOFIVE ALL RIGHTS RESERVED.

これまでのOSSの構成との違い

Postfix＋Dovecotは部品を組み合わせて作り、Stalwartは全部をまかなう

観点	Postfix＋Dovecot など	Stalwart
構成	役割ごとの部品を組み合わせる（送受信はPostfix、メールボックスはDovecot）	1つのサーバで、送受信からメールボックスまで
送信ドメイン認証	OpenDKIM・OpenDMARCなどの部品を追加して連携	SPF・DKIM・DMARC・ARC、MTA-STSなどを標準で搭載
スパム対策	rspamdやSpamAssassinを追加して連携	スパムフィルタを内蔵
設定と管理	部品ごとの設定ファイル。管理画面は別途	管理画面とコマンドラインで一元管理
アカウント管理	OSのユーザーや仮想ユーザー。LDAP・SQLとの連携を部品ごとに設定	内部ディレクトリで管理（今回）。LDAP・SQL・OIDCとも連携できる
実績と情報量	長年の実績。日本語の情報も多い	比較的新しく、更新が頻繁。日本語の情報は少ない
AIとの相性	学習データが多く、答えが安定しやすい	AIの知識が古いことが多く、公式情報での確認が必須
初心者にとって	部品の連携を理解する必要があり、最初の壁が高い	入り口は低いが、仕組みの理解は別に必要

© TWOFIVE ALL RIGHTS RESERVED.

生成AIの比較と選定理由

Microsoft Copilotの評価でも、今回の用途の第一候補はClaude＋Claude Code

ポイント	Claude＋ Claude Code	ChatGPT＋ Codex	GitHub Copilot	Gemini
未知の製品・技術の調査	◎	◎	○	◎
要件整理・構成設計	◎	◎	○	◎
Linux・CLI中心の構築	◎	◎	○	○
複数の設定ファイルの理解・編集	◎	◎	◎	○
コマンド実行を伴う構築作業	◎	◎	○～◎	○
エラー→ログ解析→修正の反復	◎	◎	○～◎	○
Bash・Ansibleでの自動化	◎	◎	◎	○
Gitリポジトリを使った開発	◎	◎	◎	○
技術的な壁打ち・相談	◎	◎	○	◎
長いドキュメント・設定の理解	◎	◎	○	◎
今回の用途への総合適性	◎ **第一候補**	◎ 第一候補級	○	○

評価：Microsoft Copilotの回答（今回の用途に対する評価で、製品の性能順位ではない）

今回の検証で確かめたこと

サーバを直接操作できる
SSHでの構築から記録・コミットまで任せられた

承認を細かく制御できる
コマンドごとに承認する運用にできた

ルールを書いて渡せる
CLAUDE.mdで秘密情報と記録のルールを守らせた

根拠を示して確かめる
前提がないと止まり、誤った指摘には確認を求めた

© TWOFIVE ALL RIGHTS RESERVED.

目次

1 **背景と問い**
学び始める入り口が狭い中で、生成AIは助けになるか

2 **使ったソフトウェアと生成AI**
Stalwart、Claude＋Claude Code、これまでの構成との違い

3 **検証環境と進め方**
受信できない環境での検証の分担、任せる範囲と承認

4 **結果**
送信側と受信側の認証、作業の分担と時間

5 **落とし穴と人の判断**
実際にあった落とし穴、誤りの止まり方と減らし方、人が引いた線

6 **得られた知見と今後**
問いへの答えと、生成AIとの関わり方

© TWOFIVE ALL RIGHTS RESERVED.

検証環境とその制約

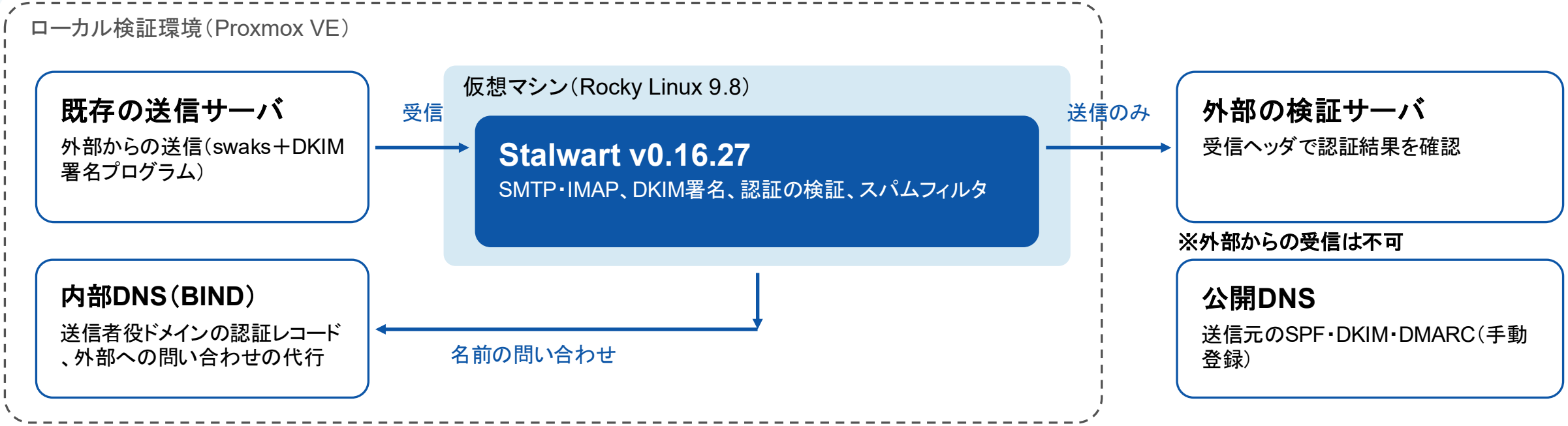
現場でよくある「あまり理想的ではない環境」を用意

項目	内容
基盤	ローカル検証環境のProxmox VE上の仮想マシン1台
OS	Rocky Linux 9.8（クリーンインストール状態。SELinuxは無効、firewalldは停止）
外部への送信	可能。自分で設定できるPTRはなく、宛先は外部の検証ドメインに限る
外部からの受信	不可。受信のテストはローカル内で外部の送信者を再現する
DNS	公開DNSは手動登録、内部DNS（BIND）はテスト用のゾーンを持ち、外部への問い合わせも代行

© TWOFIVE ALL RIGHTS RESERVED.

検証環境の構成

受信経路　ローカル内の既存サーバ → Stalwart
送信経路　Stalwart → 外部の検証サーバ



swaksでのテスト送信

Stalwartの25番ポートへ直接送信（SMTP認証なし）。エンベロープFromとヘッダFromを送信者役ドメインにし、テストごとに署名の有無・本文の改ざん・なりすましを切り替える

DKIM署名プログラム

送信者役ドメインの鍵で署名してから、swaksで送る。公開鍵は内部DNSに登録し、本文を改ざんするテストは署名の後で書き換える

IMAPでの結果確認

Claude CodeがIMAPで全フォルダを読み取り専用で開き、認証の結果と届いたフォルダを期待値と比べて合否を判定（点数は表示のみ）

© TWOFIVE ALL RIGHTS RESERVED.

外部からメール受信できない環境での検証の分担

外部の検証サーバーでは送信側を、ローカル内では受信側を確かめた

前提：外部からは受信できず、自分で設定できるPTRもない。そのため、送信側は外部の検証ドメインで、受信側はローカル内で外部の送信者を再現して確かめる

確かめたいこと	選んだ観点	方法	外部の証跡
送信側の認証	自分のドメインから送ったメールが、外部でSPF・DKIM・DMARCのpassになるか	外部の検証ドメインへ送り、受信ヘッダを確認	**人がIMAPで取得**
受信側の認証チェック	外から届くメールの認証を、Stalwartが正しく判定できるか	既存の送信サーバが外部の送信者を再現	ローカル内
なりすまし・改ざん	送信ドメイン認証が防ぎたいものを、failとして見分けられるか	内部DNSで送信者役ドメインを定義し、passとfailを再現	ローカル内
スパム判定（内蔵フィルタ）	認証の判定が、点数と配送先にどう効くか	点数の内訳と、DNSBLの判定を確認	ローカル内

© TWOFIVE ALL RIGHTS RESERVED.

Claudeに任せる範囲と、人が持つ判断

実行はClaudeに任せ、変更の承認と方針の決定は人が持つ

Claudeに任せる

・構成案と手順の作成
・設定ファイル・スクリプトの作成と実行
・テストケースの設計と実行
・ログ・受信ヘッダの解析
・作業内容の記録

人が承認・判断する

・パッケージ導入・設定変更の承認
・メール送信の承認（宛先と内容）
・秘密情報の投入と公開DNSの登録
・役割分担や方針の決定
・公式情報での最終確認

© TWOFIVE ALL RIGHTS RESERVED.

進め方

まっさらなOSから6つの段階で設定・構築を進め、実行した内容はスクリプトとして残した

0 準備

・スナップショットを取得
・CLAUDE.mdに承認と記録のルール
・パスワードなしのsudoを用意

1 OSの初期設定

・ホスト名と時刻を整える
・設定はスクリプトにして残す

2 Stalwartの構築

・v0.16.27を導入
・内部CAで証明書を発行
・管理画面で初期設定（人）

3 DNSと認証

・SPF・DKIM・DMARCを設定
・DKIMはed25519とRSAの二重署名
・生成と照合は自動、登録は手動

4 テストの自動化

・受信45項目・外部への送信・TLS
・swaksで送り、IMAPで結果を確認
・1コマンドで合否まで

5 再構築の計測

・スナップショットから戻す
・残したスクリプトで作り直す
・時間・承認・手作業を数える

© TWOFIVE ALL RIGHTS RESERVED.

目次

1 **背景と問い**
学び始める入り口が狭い中で、生成AIは助けになるか

2 **使ったソフトウェアと生成AI**
Stalwart、Claude＋Claude Code、これまでの構成との違い

3 **検証環境と進め方**
受信できない環境での検証の分担、任せる範囲と承認

4 **結果**
送信側と受信側の認証、作業の分担と時間

5 **落とし穴と人の判断**
実際にあった落とし穴、誤りの止まり方と減らし方、人が引いた線

6 **得られた知見と今後**
問いへの答えと、生成AIとの関わり方

© TWOFIVE ALL RIGHTS RESERVED.

送信側の結果

外部の受信サーバで、SPF・DKIM（ed25519とRSA）・DMARCがすべてpass

外部の検証ドメインで受信したヘッダ（抜粋）

```
Authentication-Results: [検証ドメインの受信サーバ];
  spf=pass smtp.mailfrom=tksk@example.jp;
  dkim=pass header.d=example.jp;    ← ed25519
  dkim=pass header.d=example.jp;    ← RSA
  dmarc=pass header.from=example.jp
```

テスト（6項目）	確かめること	判定	結果
DNSの照合（2）	自ドメインの20件（内部DNS 8件、公開DNSの権威サーバ2台×6件）と送信者役の14件が、期待値と一致するか	自動	**OK**
受け付け・配送・署名（3）	Stalwartが受け付け、外部へ配送し、DKIM署名（2つ）を付けたか	自動	**OK**
宛先側のヘッダ（1）	受信ヘッダで、SPF・DKIM（2つ）・DMARCの4つがpassか	人が確認	**すべてpass**

送信の条件は3通り（最初の鍵／DKIM鍵を手動で切り替えた後／DNSの問い合わせ先を内部DNSに変えた後）。いずれもすべてpass

© TWOFIVE ALL RIGHTS RESERVED.

受信側の結果

認証の判定は想定通りの結果だったが、異常なメールのうち4通は受信箱に届いた

テスト（45項目）	確かめること	SPF	DKIM	DMARC	点数	届いた先
01 正常	正しいメールが受信箱に届くか	pass	pass	pass	-0.80	受信箱
02 SPF fail・署名あり・逆引きなし	DKIMがそろえばDMARCはpassか	fail	pass	pass	5.90	迷惑メール
03 SPF fail・署名なし・逆引きなし	両方そろわなければDMARCはfailか	fail	なし	fail	6.70	迷惑メール
04 本文の改ざん	署名後の改ざんをDKIMで見分けるか	pass	neutral	pass（SPF）	-0.60	**受信箱 ※1**
05 なりすまし（p=reject）	p=rejectのなりすましをどう扱うか	pass	なし	fail	4.20	**受信箱 ※2**
06 本文のURLがDNSBLに該当	本文のURLをDNSBLで判定できるか	pass	pass	pass	3.70	**受信箱 ※3**
07 同じ組織ドメインのなりすまし	別のサブドメインもそろうとみなすか	pass	なし	pass	1.20	**受信箱 ※4**

45項目＝7通×6項目（受け付け・DKIM・SPF・DMARC・iprev・届いたフォルダ）＋06のDBL判定1、DNSの照合2。点数は判定に含めない。迷惑メールの境目は5.0点

※1 04 本文の改ざん
改ざんされた署名は、点数の上では「署名なし」と同じ（0点）。許可されたアドレスから送りSPFがpassしたため、DMARCもpass

※2 05 なりすまし（p=reject）
拒否するかは点数ではなく設定（dmarcVerify）で決まる。既定のrelaxedでは拒否せず、DMARC fail（p=reject）の+4.00点だけで5.0点に届かない

※3 06 本文のURLがDNSBLに該当
DBLの判定で+4.50点。ただし認証のpassで-0.90点され、3.70点で5.0点に届かない

※4 07 同じ組織ドメイン
DMARCの既定の比べ方（relaxed）では、同じ組織ドメインのサブドメインはそろったとみなす。aspf=s・adkim=sで厳密になる

© TWOFIVE ALL RIGHTS RESERVED.

受信側の点数の内訳

迷惑メール判定されたのは、複数の条件でスコア値に悪影響を与えたケース

スコア値に影響した項目	ルール名	01	02	03	04	05	06	07
認証がpass（DMARC・DKIM・SPF）	*_ALLOW	-0.90	-0.70		-0.70	-0.20	-0.90	-0.70
SPF違反のメールが直接届いた	VIOLATED_DIRECT_SPF		3.50	3.50				
送信元IPに逆引きがない	RDNS_NONE		2.00	2.00				
SPFがfail	SPF_FAIL		1.00	1.00				
DMARCがfail（p=reject）	DMARC_POLICY_REJECT					4.00		
DMARCがfail（p=none）	DMARC_POLICY_SOFTFAIL			0.10				
本文のURLがブロックリストに該当	DBL_SPAM						4.50	
Fromのドメインに A・MX がない	FROMHOST_NORES_A_OR_MX							1.50
エンベロープとFromが違う	FORGED_SENDER					0.30		0.30
Receivedヘッダがない	RCVD_COUNT_ZERO	0.10	0.10	0.10	0.10	0.10	0.10	0.10
合計	境目は5.0点	**-0.80**	**5.90**	**6.70**	**-0.60**	**4.20**	**3.70**	**1.20**
届いた先		**受信箱**	**迷惑メール**	**迷惑メール**	**受信箱**	**受信箱**	**受信箱**	**受信箱**

02と05の逆転

02の5.90点の大半は、SPFのfailに関わる3項目（6.50点）。Receivedなし・逆引きなしはテストの条件。05はp=rejectのfailの4.00点だけで5.0点に届かない

04（受信箱）と03（迷惑メール）

04は許可されたアドレスから送りSPFがpass。改ざんされた署名は「署名なし」と同じ0点で、加点されない。03は許可外・逆引きなしでSPFの3項目が付く

拒否は点数ではなく設定

p=rejectで拒否するかはdmarcVerifyで決まる。strictなら、p=rejectのfailだけを受信時に550で拒否する（p=none・quarantineは拒否しない）

© TWOFIVE ALL RIGHTS RESERVED.

Claude Codeと人の作業分担（再構築）

Claude Codeはスクリプト16本・コマンド77回、人は承認26回と手作業12件

Claude Code

スクリプトの実行 — 16本
OSの初期設定からDNSの照合・テストまで。うち直接実行は13本・35回

コマンドの実行 — 77回
サーバ上で47回、手元で30回（読み取り・git、記録と手順書の更新）

テストの実行と判定 — 60項目
受信45・外部への送信6・TLS 9。IMAPとログで合否を判定

調べと直し — 8回
スクリプト2本と手順書・記録を修正（4ファイル・8回）、サーバの設定2件

人

承認 — 26回
サーバでの変更と送信25回、証明書の登録方法の選択1回

秘密情報の配置と後片付け — 5件
鍵やパスワードの配置と、作業後の削除

管理画面での操作 — 5件
ウィザード、管理者パスワード、APIキー、送信用アカウント、APIキーのIPの制限

その他 — 2件
スナップショットの復元、宛先のメールボックスでの確認

© TWOFIVE ALL RIGHTS RESERVED.

構築にかかった時間と手作業

最初の構築は約350分。手順書とスクリプトでの再構築(2回目)は約73分になった

最初の構築

約350分

・作業記録（R-002～R-008）の合計
・検証・自動化の準備を含め 約10時間10分
・記録の間・休憩は含まない

再構築 (2回目)

約73分

・人の作業 約30分
・承認待ちと確認 約25分
・調査/修正 約10分
・Claude Codeの実行 約8分

さらに短縮（見込み）

約25分

・Claude Codeの実行 約8分
・人の作業 約10分（API化）
・承認待ちと確認 約5分（まとめる）
・調査/修正 0分（反映済み）

最後まで残った手作業

12件

・秘密情報の配置・後片付け 5件
・管理画面での操作 5件
・スナップショットの復元 1件
・外部での確認 1件

最初の構築の時間は作業記録上の合計。短縮の見込みは再構築の内訳からの試算（合計 約23分）

© TWOFIVE ALL RIGHTS RESERVED.

目次

1 **背景と問い**
学び始める入り口が狭い中で、生成AIは助けになるか

2 **使ったソフトウェアと生成AI**
Stalwart、Claude＋Claude Code、これまでの構成との違い

3 **検証環境と進め方**
受信できない環境での検証の分担、任せる範囲と承認

4 **結果**
送信側と受信側の認証、作業の分担と時間

5 **落とし穴と人の判断**
実際にあった落とし穴、誤りの止まり方と減らし方、人が引いた線

6 **得られた知見と今後**
問いへの答えと、生成AIとの関わり方

© TWOFIVE ALL RIGHTS RESERVED.

見えてきた落とし穴

もっともらしいが危うい場面が、準備から検証までにいくつか発生した

準備中に実際にあった

AIの知識が古い

ソフトウェアの対応OSを古い前提で判断していた。公式情報を確認して訂正

なぜ：AIの知識は学習した時点まで。Stalwartは更新が速く、知らない部分を推測で補ってしまう

リリースノートから発見

DKIM鍵の自動切り替え

DNSを手動管理にすると、鍵は1回だけ作られてすぐ署名に使われ、切り替わらない。DNSに登録するまで受信側は検証できない

なぜ：切り替えは、新しい鍵をDNSに載せ、見えるのを待ってから使う仕組みで、DNSの自動管理が前提。今回は手動管理にし、登録・確認・切り替えをスクリプト化した

構築中に実際にあった

秘密情報が画面に出た

「値を表示しない」ルールがあっても、確認コマンドの出力に管理者パスワードが表示された

なぜ：ルールは結果を禁じただけで、出力に値を含むコマンドを選ばない、までは決めていなかった

調査で見つけ、検証で確かめた

公開リゾルバのまま

DNSが8.8.8.8だと、Spamhausのブロックリストが応答しない。内部DNS経由に変えて判定が効くことを確認

なぜ：Spamhausは、大量の問い合わせが集まる公開リゾルバからの問い合わせに応答しないため

© TWOFIVE ALL RIGHTS RESERVED.

誤りはどこで止まったか

誤りの多くはエラーにならず、照合とClaude Code自身の確認で止まった

	最初の構築	再構築
Bashの実行回数	267	77
誤りの数	**34**	**10**
Claude Code／人／環境・仕様	20／5／9	9／1／0
うちエラーになったもの	6	2
気づいたきっかけ		
照合	9	5
Claude Codeが自分で	8	3
コマンドのエラー	13	2
人の指摘・報告	4	0
承認での却下・修正の指示	3（13回中）	0（26回中）

代表的な例

管理者パスワードを表示した
最初の構築・Claude Codeの誤り
→ Claude Codeが自分で気づいた

/etc/stalwart の所有者が変わった
再構築・チャットで渡したコマンドの誤り
→ Claude Codeがログで気づいた

RSAのDKIMレコードが k=ed25519
最初の構築・人の登録ミス
→ Claude Codeの照合で気づいた

© TWOFIVE ALL RIGHTS RESERVED.

誤りを減らすには

最初の構築の30件のうち約20件は、作業の前の決めごとで防げた

1. **最初に環境と前提を確かめる**
   手元のシェル、ファイルの権限、DNSを問い合わせる場所を、作業の前に決める
2. **確認はスクリプトにして使い回す**
   その場で書いたコマンドの誤りをなくす。承認の前に --help で書き方を確かめる
3. **人に値を手で写させない**
   ファイルはまるごと渡し、APIキーを使う。登録後はdigでの確認を手順に入れる
4. **状態は、確かめた出力からだけ書く**
   「既定」「残っている」を推測で書かない。版はパッチ版まで一覧にして決める
5. **秘密は数と有無だけを見る**
   値を伏せる処理に頼らず、最初から値を出さない確かめ方を選ぶ

防げたもの

約20件

Claudeの誤りのほとんどと、人が値を書き写す作業による誤り

防げないもの

約5〜7件

ドキュメントと実際の動作の違い。使い捨ての仮想マシンで先に試し、見つける時期を早める

再構築では、コマンドのエラー 13→2、照合 9→5、自分で気づいた 8→3 に減った。1と4は、その後CLAUDE.mdのルールに追記

© TWOFIVE ALL RIGHTS RESERVED.

人が線を引いたところ

何でもAI任せにせず、基準を決めて線を引いた

判断	決めたこと
DNSの登録	登録は手動、登録内容の生成と照合は自動。照合が人の登録ミスを止めた
権限	パスワードなしのsudoを許可し、歯止めはコマンドごとの承認に置いた（戻せる検証環境のため）
秘密情報	送信用アカウントとパスワードは人が管理画面で作成。管理操作はAPIキーで行わせた
送信先	家庭向け回線のIPで届かない可能性が高いため、Gmailには送らない
環境の設定	SELinux無効・firewalld停止は変えず、検証範囲をメール機能に絞った

© TWOFIVE ALL RIGHTS RESERVED.

AIをアシスタントにするときのコツ

AIに設定を聞くだけでなく、確かめ方と任せ方を決めておく

1	**公式情報で確かめさせる**	AIの知識は古いことがある。参照したURLを示させ、自分でも確認する
2	**変更の前に承認する**	何をなぜ変えるかを示させ、1つずつ許可する。自動で実行するモードは使わない
3	**秘密情報は渡さない**	パスワードや鍵は会話に貼らない。値ではなく置き場所だけを伝える
4	**テストで確かめる**	予想を先に決め、結果と照らし合わせる。テスト自体の誤りも疑う
5	**記録とスクリプトを残す**	何を頼み、何を直したかを残す。次は短い時間で作り直せる

© TWOFIVE ALL RIGHTS RESERVED.

目次

1 **背景と問い**
学び始める入り口が狭い中で、生成AIは助けになるか

2 **使ったソフトウェアと生成AI**
Stalwart、Claude＋Claude Code、これまでの構成との違い

3 **検証環境と進め方**
受信できない環境での検証の分担、任せる範囲と承認

4 **結果**
送信側と受信側の認証、作業の分担と時間

5 **落とし穴と人の判断**
実際にあった落とし穴、誤りの止まり方と減らし方、人が引いた線

6 **得られた知見と今後**
問いへの答えと、生成AIとの関わり方

© TWOFIVE ALL RIGHTS RESERVED.

得られた知見

1. AIの知識は古い前提で、バージョンや仕様は必ず公式情報で確かめる

2. 手順をスクリプトとテストに残すと、再構築は約5分の1になり、AIと人のミスも止められた。
ただし秘密の扱いなど12件は人の手に残る

3. 認証の判定が正しくても、受信側のポリシーの決め方は人の判断が要る

© TWOFIVE ALL RIGHTS RESERVED.

問いへの答え

問い：生成AIをアシスタントにすれば、初心者・若手エンジニアや少人数のチームでも、“ちゃんとした”メールサーバを構築・検証できるか？

答え：できる。ただし、確かめる力と判断は人が持つ必要がある

知識はAIで補える

・送受信の仕組みや設定の意味を、その場で聞きながら学べる

・設定・スクリプト・テストを作り、ログやヘッダを読み解ける

・手順とテストを残せば、少人数でも同じ環境を作り直せる

初心者が陥りがちなこと

・AIの古い知識や誤りを、そのまま信じる

・認証がpassすれば安全だと思い込む

・秘密情報を会話に貼る、画面に出してしまう

・公開リゾルバのままなど、環境の前提に気づかない

経験と判断が要ること

・受信側のポリシー（拒否するか、何点で迷惑メールか）

・承認と権限の線引き、秘密情報の扱い

・送信元IPの評判など、運用で得る知見

・AIの答えが正しいかの見極め

© TWOFIVE ALL RIGHTS RESERVED.

今後に向けて

生成AIで入り口を広げ、経験と判断を次の世代へつないでいく

今後の課題

完全自動化
Terraform＋Ansibleで仮想マシンから

役割ごとのエージェント化
Claudeを構築・テスト・確認の役割に分け、水平展開できる仕組みに

Stalwartの詳細な評価
より詳細な検討とテストで、Postfix・Dovecotの代替になりうるかを見極める

生成AIとエンジニアとしての広がり

初心者 × 生成AIを使いこなす
・調べる・作る・試すを速く回し、設計から自動化まで早く経験できる
・確かめる力がないと、誤りに気づかないまま広がってしまう

経験者 × 生成AIを使いこなせない
・判断と知見は強み。誤りを見抜き、線を引ける
・速さと扱える範囲で差がつき、知見を広げて渡す機会を逃す

目指す：両方を持つエンジニア
・初心者は、AIで経験を前倒ししながら確かめる力を育てる
・経験者は、判断をテストと手順にしてAIと次の人に渡す

生成AIは経験を置き換えるものではなく、経験を広げて受け継ぐ道具。
確かめる力と判断を持つ人を増やし、メールの安全と信頼を業界全体で前へ進めていく

© TWOFIVE ALL RIGHTS RESERVED.

ご清聴ありがとうございました

