ドメイン名と証明書とTLS

自分でメールサーバなどを
動かす私人の立場で

藤原 和典
fujiwara@jprs.co.jp
株式会社日本レジストリサービス (JPRS)

Internet Week ショーケース 徳島
2022年6月24日

自己紹介

- 氏名:　　藤原和典
  - 個人ページ: http://member.wide.ad.jp/~fujiwara/
  - 勤務先:株式会社日本レジストリサービス (JPRS) 技術研究部
  - 業務内容:DNS関連の研究・開発 (標準化、試作、論文)
  - Internet Week プログラム委員 (2016～)
- 本日の立場: 自分でメールサーバなどを動かす私人
  - DNSSECはあたりまえ/ DNSSECの署名と検証は10年以上やっている
    - オランダ製ソフトウェアによるDNSSEC遊び, 21 July 2010, DNSOPS.JP BoF,
    - http://dnsops.jp/bof/20100721/dnsops-20100721.pdf
  - TLSとか証明書については素人

個人サーバの運用

- 1990年: UUCPで (当時は juice.or.jp 下流)
- 1992年: 大学(や会社)で個人サーバ運用 (→自宅ADSL/光)
- 1998年: 10人ほどで月9800円のサーバを借り、ドメイン名を登録し、webサーバ、メールサーバの運用開始
  - jpドメイン名はgr.jpの登録が面倒そうだったのでorgドメイン名とし、メンバーの一人が女の子の名前.pyonなんとかでドメイン名作りたいといったのでpyon.org (同じ時期に知人がpyon.comを登録)
- 2008年: VPSに移行 (月500円 → 移動 → 移動)
  - (電気通信事業者とみなされる可能性に気が付き)
  - 分担金を集めるのをやめ、メール、Webの転送のみ → 個人VPS

個人サーバの機能

- DNSサーバ
  - PrimaryをVPSで、Secondaryをフレッツ光の先の家で
    - フレッツ光では、固定IPv4アドレス必須 (→ 2022/6/1半固定に)
  - 2011年ごろからDNSSEC署名済 (2009年ごろにDLV遊び)
- メールサーバ
  - MTA (sendmail)
  - POP3 (dovecot, localhostのみbind,家からssh portforwardで接続)
- Webサーバ (Let's Encryptの証明書を得るため)
- CVSリポジトリ (home の一部)
- cronで適当なデータ収集

サーバ設定について (私見、一般論？)

- DNS関連: DNSSECは簡単という立場
  - **権威サーバのDNSSECの設定は簡単、署名して、DSをレジストラに登録するだけ**
  - **フルサービスリゾルバでのDNSSEC検証も簡単**
    - unboundいれて、auto-trust-anchor-fileを指定するだけ

- Webサーバ: いまどきはTLS必須
  - 数年前まではサーバ証明書は有料だった
  - 個人の場合は金をかけずに Let's Encrypt でサーバ証明書を入手
  - Trustの概念とか深く考えず、ブラウザから接続できれば十分
- メールサーバ
  - SPF (TXT "v=spf1 ip4:… ip6:… -all") ぐらいは書いておく
  - 多くのメールサーバがSTARTTLSに対応した (受けるときはサーバ証明書が必要)
    - 送るときも正しいサーバ証明書がいるか？
  - Gmailに迷惑メール判定されるようになった (関係は不明)

証明書の設定 (メールサーバ)

- **Webサーバ用の証明書はメールサーバなどでも使用できる**
- MXメールサーバ名とEHLOで返すサーバ名を同じにし、サーバ証明書を用意
  - メールサーバ名と同じWebサーバを用意 (Webサーバいらなくても動かすこと)
  - Let's EncryptでWebサーバ用にサーバ証明書を設定
  - dehydratedでは /usr/local/etc/dehydrated/certs/ドメイン名/ に証明書ファイル
- FreeBSD では ca_root_nss package をいれることで、/usr/local/share/certs/ca-root-nss.crt にCAリストが追加
- /etc/mail/sendmail.mc に以下を追加 (postfix, eximでもできるはず)
  - define(`CERT_DIR', `/usr/local/etc/dehydrated/certs/ドメイン名')dnl
  - define(`confSERVER_CERT', `CERT_DIR/cert.pem')dnl　サーバ証明書
  - define(`confSERVER_KEY', `CERT_DIR/privkey.pem')dnl　秘密鍵
  - define(`confCLIENT_CERT', `CERT_DIR/cert.pem')dnl　サーバ証明書
  - define(`confCLIENT_KEY', `CERT_DIR/privkey.pem')dnl　秘密鍵
  - define(`confCACERT', `/usr/local/share/certs/ca-root-nss.crt')dnl　ルートCAリスト
  - define(`confCACERT_PATH', `/usr/local/share/certs/')dnl

メールサーバの動作確認 (サーバ証明書)

- telnet g.pyon.org 25

220 g.pyon.org ESMTP Sendmail 8.16.1/8.16.1; Fri, 29 Oct 2021 15:17:01 +0900 (JST)
250-8BITMIME
250-STARTTLS
250 HELP

- openssl s_client -starttls smtp -host g.pyon.org -port 25

Certificate chain
 0 s:CN = g.pyon.org
  i:C = US, O = Let‘s Encrypt, CN = R3　　Let’s Encryptから発行された証明書
略
EHLO local
250-g.pyon.org Hello g.pyon.org, pleased to meet you
250-8BITMIME
250 HELP

Gmailにメールを送ってみる

- echo "test" | Mail -s "arrive ?" fujiwara3x@gmail.com
- 結果、Gmailで迷惑メールと判定されなかった → DownloadしてReceived確認

ARC-Authentication-Results: i=1; mx.google.com;
spf=pass (google.com: domain of fujiwara@g.pyon.org designates 2001:e42:102:1804:160:16:204:207 as permitted sender) smtp.mailfrom=fujiwara@g.pyon.org;
dmarc=pass (p=NONE sp=NONE dis=NONE) header.from=pyon.org
Received: from g.pyon.org (g.pyon.org. [2001:e42:102:1804:160:16:204:207])
by mx.google.com with ESMTPS id q12si2475570pjv.168.2021.10.28.23.48.57
for <fujiwara3x@gmail.com>
(version=TLS1_3 cipher=TLS_AES_256_GCM_SHA384 bits=256/256);
Thu, 28 Oct 2021 23:48:58 -0700 (PDT)

- Receivedヘッダより、GmailのサーバにTLSでメールを送信できている
  - ARC-Authtntication-Resultsヘッダより、spf=pass, dmarc=pass も確認できる

Gmailからのメールを確認

- GmailでReplyしてみた
  - Received: from mail-il1-x12f.google.com (mail-il1-x12f.google.com [IPv6:2607:f8b0:4864:20:0:0:0:12f]) by g.pyon.org (8.16.1/8.16.1) with ESMTPS id 19U8Pumv034746 (using TLSv1.3 with cipher TLS_AES_256_GCM_SHA384 (256 bits) verified OK) for <fujiwara@g.pyon.org>; Sat, 30 Oct 2021 17:26:02 +0900 (JST)
  - 結果: sendmail がGmailのサーバ証明書を検証できている

メールに関する動向: メールでもTLS必須に

- RFC 6698 The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA, 2012/8
  - DNSSECにTLSサーバ証明書を入れる仕組み
  - DNSSECで守られたところに自己署名証明書を置くと、CAいらないという主張あり
  - 信頼としては、ドメイン名文字列に対応する証明書であること(だけ)を示す
- RFC 7672 SMTP Security via Opportunistic DNS-Based Authentication of Name Entities (DANE) Transport Layer Security (TLS), 2015/10
  - SMTP STARTTLSのサーバ証明書をDNSSECで守られたドメイン名に載せる仕組み

- RFC 8461 SMTP MTA Strict Transport Security (MTA-STS), 2018/9
  - 指定したドメイン名のメールサーバがTLS(STARTTLS)でしかメールを受け取らないことを示す
    - _mta-sts.example.com. IN TXT "v=STSv1; id=20160831085700Z;“
- RFC 8689 SMTP Require TLS Option, 2019/11
  - EHLOの応答にREQUIRETLS追加、MAIL FROMにもREQUIRETLS追加

メールに関する動向 (2)

- US政府がMTA-STS(RFC 8461)を要件としようとしているらしい
- それに対し、DANE SMTP (RFC 7672)派が問題点を指摘しているらしい
  - MTA-STSはWebPKIのTrust modelを使用するが、WebPKIにはUS政府に敵対する可能性のある国家によって運営されているCAが存在する
  - TLSではないメールにフォールバックする可能性がある
- どちらも、サーバ証明書を使ってメールサーバ間のSMTPをSTARTTLSでTLSにすることは同じ

WebサーバでのTLSA RRの書き方

- _443._tcp.ドメイン名 IN TLSA CertUsage Selector MatchingType Certificate
  - CertUsage 0..CA証明書, 1..サーバ証明書(有効なCA発行), 2..trust anchorの証明書, 3..サーバ証明書(自己署名でもいい)
  - Selector 0..FullCertificate, 1..SubjectPublicKeyInfo
  - MatchingType 0..証明書そのもの 1..SHA256 hash, 2..SHA512 hash
- 作り方
  - openssl x509 -inform PEM -outform DER -in 証明書.pem | sha256 を用意
  - 例: openssl x509 -inform PEM -outform DER -in /usr/local/etc/dehydrated/certs/g.pyon.org/cert.pem | sha256

d3829e7c648f0f447df37486ad371219e985dd2db9c5a5768555e2461d6c51c6

  - _443._tcp.ドメイン名 IN TLSA 3 0 1 ハッシュ値
  - 自己署名かもしれないサーバ証明書の、Full CertificateのSHA256ハッシュ
  - DNSSEC必須
  - (GnuTLS danetool でもTLSA RRを生成できるらしい)

メールサーバでのTLSA RRの書き方

- _25._tcp.ドメイン名 IN TLSA CertUsage Selector MatchingType Certificate
  - CertUsage 0..CA証明書, 1..サーバ証明書(有効なCA発行), 2..trust anchoの証明書, 3..サーバ証明書(自己署名でもいい)
  - Selector 0..FullCertificate, 1..SubjectPublicKeyInfo
  - MatchingType 0..証明書そのもの 1..SHA256 hash 2..SHA512 hash
- 作り方
  - Openssl x509 –inform PEM –outform DER –in 証明書.pem | sha256 を用意
    - 証明書のpemファイルをバイナリDER形式に変換すると証明書の生データが得られる
    - 生データのsha256 hashをTLSA RRに書くこととする
  - 例: openssl x509 –inform PEM –outform DER –in /usr/local/etc/dehydrated/certs/g.pyon.org/cert.pem | sha256

d3829e7c648f0f447df37486ad371219e985dd2db9c5a5768555e2461d6c51c6

  - _25._tcp.ドメイン名 IN TLSA 3 0 1 ハッシュ値
  - 自己署名かもしれないサーバ証明書の、Full CertificateのSHA256ハッシュ
  - DNSSEC必須

TLSA RRの設定確認

- Shumon Huque さんのチェックサイトを使わせていただく
- https://www.huque.com/bin/danecheck
- ドメイン名とポート番号、アプリケーション名をいれて Check
  - 例: ホスト名, ポート番号443, Application NONE (https)
  - 例: ホスト名, ポート番号25, Application SMTP

サーバ証明書+TLSA設定の結果

- 一般ユーザがブラウザでみると
  - https:// で、Let's Encryptに発行された証明書でアクセスできる
- DNSSEC/DANE, DANE SMTP派との通信
  - _443._tcp.ドメイン名 TLSA を検索し、httpsのサーバ証明書と一致することを確認
  - _25._tcp.ドメイン名 TLSAを検索し、STARTTLSでのサーバ証明書と一致することを確認
- 普通のメールサーバ、MTA-STS / REQUIRETLS 派との通信
  - CAに発行されたサーバ証明書を要求するが、Let's Encryptに発行された証明書をMTAに指定しているので通信可能

証明書、TLSA RRの定期更新

- Let's Encryptの証明書は有効期間90日なので、定期更新が必要
- 以下のようなShell Scriptで解決できる

```
#!/bin/sh
CERTに証明書のパス, TLSAFILE, TLSAFILEprev, TLSAFILEnewに　TLSA RRを書くファイル名
if /usr/local/bin/dehydrated -c ; then exit 0; fi  # 証明書の更新を行い、変化がなかったら何もしない
mv $TLSAFILEnew $TLSAFILEprev # 一つ前のTLSAを残しておく
HASH=`openssl x509 -inform PEM -outform DER -in $CERT | sha256`　# 証明書のハッシュ値を計算
(echo "_443._tcp.HOSTNAME. $TTL IN TLSA 3 0 1 $HASH";
 echo "_25._tcp.HOSTNAME. $TTL IN TLSA 3 0 1 $HASH") > $TLSAFILEnew # TLSA RR作成
cat $TLSAFILEprev $TLSAFILEnew | sort | uniq > $TLSAFILE # 新旧TLSA RRをまとめたファイルを作成
/etc/nsd/dnsseczonetool sign DOMAIN # TLSAFILEをincludeするゾーンファイルを署名して、reload
/usr/local/etc/rc.d/apache24 restart; /etc/rc.d/sendmail restart # Webサーバ,メールサーバ再起動
exit 0
```

まとめ

- いまどきは、Webサーバ、メールサーバなどにサーバ証明書を設定する必要がある
  - Let's Encryptなどで発行したサーバ証明書はメールサーバでも使用できる
- 現実派としては、
  - Trust の詳細よりは、普通のブラウザで開けるhttpsのURLや、Gmailなどにメールを受け取ってもらえることが重要である
  - まともな証明書 = (Let's Encryptも含む)普通のブラウザが信頼しているCAに発行された証明書
- 複数の考え方の人がいるが、どの派閥の設定からでもつなげられることをすすめたい
  - 多くのブラウザで信用されたCAに発行されたサーバ証明書を使用
  - TLSA RRを作成しておく / 定期更新を忘れずに