IIR

Internet
Infrastructure
Review

Jun.2023
Vol.59

定期観測レポート

メッセージング

フォーカス・リサーチ(1)

CTOとCTO Function Listerを使ったマルウェアの解析方法

フォーカス・リサーチ(2)

クロスデバイスフローによる認証・認可

フォーカス・リサーチ(3)

IIJとDNSの30年

IIJ
Internet Initiative Japan

Internet Infrastructure Review

June 2023 Vol.59

エグゼクティブサマリ

前号の小誌で、OpenAIのChatGPTについて触れました。その後は毎日のように、生成AIや大規模言語モデルに関する情報が、IT業界のみならず、社会全体の話題をさらっています。企業における活用が進み、社会への浸透も急速に進んでいる様子が感じられます。

一方、生成AIがもたらす負の側面についても、言及されることが増えているようです。先に開催されたG7群馬高崎デジタル・技術大臣会合において「AIガバナンスの相互運用性を促進等するためのアクションプラン」が採択され、G7広島サミットの首脳コミュニケにおいても「ガバナンス、著作権を含む知的財産権の保護、透明性の促進、偽情報を含む外国からの情報操作への対応、これらの技術の責任ある活用」などが具体的に言及されています。

AIもひとつの技術である以上、最先端の技術を開発・利用する者に高い倫理観が求められることは言うまでもありません。技術者の一人としてそれをあらためて認識した次第です。

「IIR」は、IIJで研究・開発している幅広い技術を紹介しており、日々のサービス運用から得られる各種データをまとめた「定期観測レポート」と、特定テーマを掘り下げた「フォーカス・リサーチ」から構成されます。

1章の定期観測レポートでは、電子メールを中心とするメッセージングを解説します。インターネット上では様々なサービスが開発されてきましたが、電子メールは今でもインターネットに欠かせないサービスの1つであり、IIJも創業以来、電子メールを提供しています。そのような寿命の長い、重要なサービスではありますが、今でも機能改善が進んでいます。そのような状況における、IIJの電子メールサービスの停止、M^3AAWGのトピックとしてDMARC 2.0、送信ドメイン認証とSTARTTLSの普及について説明します。

2章のフォーカス・リサーチでは、IIJの社員が開発しているマルウェア解析のツールを紹介します。マルウェアはインターネットで大きな脅威となっており、被害の事例も数多く発生しています。筆者はIIJのマルウェア＆フォレンジックアナリストとして、顧客のインシデント対応を行うと共に、その経験を活かしてマルウェア解析のツールの開発を行っています。実際に解析を行っている立場から、必要と感じる機能を実装していることもあり、マルウェア解析の実態が垣間見られる記事になっています。

3章のフォーカス・リサーチでは、クロスデバイスフローによる認証・認可を取り上げます。インターネット上で提供されるサービスが社会のインフラとなる中、サービスを利用する際の認証・認可の重要性は高まるばかりです。クロスデバイスフローにより、多くの人が常時携帯しているスマートフォンを活用し、より安全で使いやすい認証・認可フローを実現できます。標準化された、あるいは、標準化に向けて作業中の複数のデバイスフローの仕様や違いについて述べています。

4章のフォーカス・リサーチは、前回のバックボーンネットワークに続き、DNSに関するIIJの取り組みを紹介します。DNSはインターネットの根幹を支える基盤であることは言うまでもなく、IIJも創業以来、様々な形でDNSに向き合ってきました。サービスや技術の観点から、社会とDNSのつながりも含め、IIJとDNSの30年間を振り返ってみました。現在とは大きく異なる商用インターネットの黎明期のDNS事情など、興味深い内容になっていると思います。

IIJは、このような活動を通してインターネットの安定性を維持しながら、日々、改善・発展させていく努力を行っています。今後も企業活動のインフラとして最大限にご活用いただけるよう、様々なサービスやソリューションを提供し続けてまいります。

島上 純一（しまがみ じゅんいち）

IIJ 常務取締役 CTO。インターネットに魅かれて、1996年9月にIIJ入社。IIJが主導したアジア域内ネットワークA-BoneやIIJのバックボーンネットワークの設計、構築に従事した後、IIJのネットワークサービスを統括。2015年よりCTOとしてネットワーク、クラウド、セキュリティなど技術全般を統括。2017年4月にテレコムサービス協会MVNO委員会の委員長に就任し、2023年5月に退任。2021年6月より同協会の副会長に就任。

メッセージング

1.1 電子メールと30周年

昨年30周年を迎えたIIJは、これまでにたくさんのサービスを作り、世に送り出してきました。とりわけ電子メールはインターネットにおけるインフラサービスの中でも、かなり古参の部類です。今も数々の新しいサービスが生まれていますが、始まりがあれば終わりもあり、新規サービスを作るよりも、今運用されているサービスをお客様への影響を最小限にしつつ、いかに安全に終了させるかの方が遥かに大変であることは、あまり知られていません。

本章の前半ではIIJが24年間提供し、昨年サービスを終了した「IIJポストオフィスサービス」を振り返ります。また、後半は送信ドメイン認証技術「DMARC」について、現在進行中の新たな議論と、IIJのメールサービスで観測している電子メールの経路暗号化について報告します。

1.2 IIJポストオフィスサービス

IIJポストオフィスサービスとは、お客様の独自ドメイン名を用いてメールの送受信が行える法人向けメールホスティングサービスです。

利用を開始するにはお客様のDNSサーバでMXレコードをIIJに向けていただくのみ、今やどのホスティング事業者も備えている機能ですが、記録によれば1998年7月にサービスを開始しています[*1][*2]。メールの保存容量は無制限、受信メールは14日間保存できる機能仕様でした。

IIJの高品質なバックボーンに直結し、安定したサービス提供をしており、お客様にご好評いただいていたのはもちろん、電子メールが企業における必須のビジネスツールであることから、担当営業からも「売りやすいサービス」として社内の知名度を上げていました。

その後も、次々と機能追加・関連サービスのリリースをしています(表-1)。

1.2.1 長寿命サービスの課題

どのような企業も、新たに生み出したサービスは、なるべく大きく売り上げ、利益を生むサービスに育て、それを長期間に渡って提供したいと思うのが自然です。IIJ ポストオフィスサービスは、まさにそのようなサイクルを生み、IIJの成長に大きく貢献したサービスの1つでした。

しかし、息の長いサービスを運用していると、遅かれ早かれ、次のような3つの課題に直面します。

■ (1)ソフトウエアが最新の技術に対応できない

ソフトウエアの技術革新は日進月歩で変化しています。読者がソフトウエア技術者であれば、日々肌で感じていることでしょう。開発当初は最新の技術だったものも、数年後にはまた新しい技術が生まれ、より良い仕組みが開発されています。

年月	項目
2001年7月	ウイルスプロテクション(ウイルス対策)機能追加*3
2002年12月	「IIJメールゲートウエイサービス」開始。メール監査オプションを追加*4
2003年3月	MailViewer(Webメール機能)を標準提供*5*6
2004年10月	迷惑メールフィルタオプション追加*7
2006年10月	「IIJセキュアMXサービス」開始*8
2010年1月	送信ドメイン認証技術DKIM標準対応*9
2010年6月	IPv6に標準対応

表-1 IIJポストオフィスサービスの変遷

*1 IIJ、「IIJポストオフィスサービス開始」(https://www.iij.ad.jp/news/pressrelease/1998/pdf/postoffice.pdf)。
*2 マイクロソフト社から Windows 98 がリリースされたのが、同月の1998年7月でした。
*3 IIJ、「IIJ、ウイルス対策サービスを7月1日より開始」(https://www.iij.ad.jp/news/pressrelease/2001/pdf/po-virusprotection.pdf)。
*4 IIJ、「IIJ、中堅企業向けの情報漏洩対策サービス『IIJ Mailゲートウェイ サービス』を開始」(https://www.iij.ad.jp/news/pressrelease/2002/pdf/iij-mgw.pdf)。
*5 IIJ、「IIJ、『IIJポストオフィスサービス』に新機能『MailViewer』を追加」(https://www.iij.ad.jp/news/pressrelease/2003/pdf/0327.pdf)。
*6 Webメールの代表格であるGmailが招待制で公開されたのが2004年です。
*7 IIJ、「IIJ、企業向け電子メールアウトソースサービスに迷惑メール対策機能を拡充」((https://www.iij.ad.jp/news/pressrelease/2004/pdf/0928.pdf)。
*8 IIJ、「IIJ、メールのあらゆるリスク管理を実現する『IIJセキュアMXサービス』を開始」(https://www.iij.ad.jp/news/pressrelease/2006/pdf/0905.pdf)。
*9 IIJ、「IIJ、『IIJポストオフィスサービス』において送信ドメイン認証技術『DKIM』に対応」(https://www.iij.ad.jp/news/pressrelease/2010/pdf/po_dkim_2.pdf)。

そのような中で古いコードに修正を加えながら、最新のトレンドにキャッチアップしていくのは大変難易度の高い業務であり、スキルもさることながら、高いモチベーションも必要です。

歴史の長いサービスは、刻々と変化するインターネットのセキュリティ要件や、お客様から頂戴する新機能の要望や改善にお応えするのが難しくなってきてしまいます。

■ (2)脆弱性対応に限界が見えてくる

ソフトウエアは開発したら終わりではありません。毎日のように報告されている脆弱性対応や、いずれ訪れるミドルウエアのEoLに対応する継続的な保守開発が必要であり、一般的にサービスのリリースサイクルは、新規開発よりも保守開発の方が圧倒的に長いことも重要なポイントです。

例えば、IIJポストオフィスサービスは、OSのサポート終了に伴って、6世代、7種類のOSを入れ替えながらサービスの提供を継続していました。こうした保守開発はサービス品質を維持する上で極めて重要で欠かせないものです。しかし、新機能開発と比べて目に見える変化が少なく、現行で動いているものに手を加えるため、新たな不具合を混入してしまうリスクもあり、お客様や担当営業、経営層にも伝わりづらい、地味な開発であることも事実です。

■ (3)開発当時の背景や経緯が分からない

古いIIJのサービスは「開発したメンバーが運用する」、というケースが少なくありませんでした。これには「最もサービスに精通したメンバーが運用しているので、障害が発生しても復旧が早い」といった良い点がある一方で、そのような「運用でカバー」されてしまった結果、ドキュメント化や新メンバーへのスキルの継承が進みづらいという課題もあります。IIJポストオフィスサービスもその1つでした。

従って、時間の経過とともに開発当初のメンバーは減少していき、サービスの立ち上げや企画に携わった人物ではないメンバーで開発・運用することになります。こうなるとドキュメント化されていない部分は、当時の状況を推し量ることしかできません。「なぜこうなっているのだろう?」という部分が増えていった結果、保守開発や、メンテナンスのハードルが極めて高いものになります。

1.2.2 サービス終了の決断とロードマップ

以上のような背景を抱えながら運用していましたが、ついに延命できない技術的な課題に直面してしまいました。

この状況をステアリングコミッティで共有し、2018年の運用・開発・サポート部門が一同する社内会議にて、「4年後にIIJポストオフィスサービスを終了する」ことを決断しました。このときのアクションプランは次のとおりです。

- サービス終了日の設定
- 後継サービス(IIJセキュアMXサービス)への移行支援体制確立
- IIJセキュアMXサービスに移行支援機能の開発・実装
- 社内アナウンス
- お客様アナウンス
- 担当営業への個別進捗確認

サービスの終了によってお客様が受けるビジネス影響を最小限にするため、今回はすべての担当営業に漏れなく連絡を取り、進捗を確認しました。大変泥臭い業務ですが、IIJ の都合でサービスを終了することになりますので、入念な事前調査をしたのちに、サービス終了日の約1年前から営業部門への呼びかけを開始し、全社横断で取り組みました。

1.2.3 24年間の歴史に幕を閉じる

そして2022年9月30日、「IIJポストオフィスサービス」は静かに提供を終了しました。本サービスの終了によって、お客様には大変ご迷惑をお掛けしたことをお詫び申し上げます。

また、普段私のような技術部門から、直接感謝の気持ちをお伝えする機会がございませんが、IIJポストオフィスサービスをご利用くださったすべてのお客様に、この場を借りて厚く御礼申し上げます。

長年のご愛顧をいただきまして、誠にありがとうございました。

なお、IIJポストオフィスサービスの後継サービスとして、IIJセキュアMXサービスをご用意しております。今後とも、IIJのサービスをよろしくお願いいたします。

1.2.4 IIJポストオフィスサービスをご利用だったお客様へのお願い

IIJポストオフィスサービスをご利用いただいたお客様に、最後のお願いがございます。

> もし、お客様ドメインのTXTレコードに
> **include:spf.po.2iij.net**
> が記述されている場合は、忘れずに削除してください。

サービス終了後の整理を進めており、まもなく、このSPFレコードを削除いたします。

万が一、お客様が電子メールで利用しているドメインにIIJポストオフィスサービス専用のSPFレコード "include:spf.po.2iij.net" が記載されたままですと、このレコードを削除したタイミングから、宛先で送信ドメイン認証の検証が失敗(permerror)することが予想されます。その結果、お客様ドメインでのなりすまし対策に不備が出る可能性があります。

連絡の取れるお客様には、担当営業を通じてご連絡を差し上げていますが、今一度ご確認をお願いいたします。

1.3 DMARC2.0について (M^{3}AAWG topic)

2023年2月、サンフランシスコにて3年ぶりにM^{3}AAWGが現地開催されました。M^{3}AAWG(Messaging, Malware and Mobile Anti-Abuse Working Group)は、2004年に設立され、メールをはじめとするメッセージング技術を中心とした分野に関する議論をする団体であり、世界各国から様々なメンバが参加をしています。IIJのようなMSP(Mailbox Service Provider:メールボックス事業者)や、ESP(Email Service Provider:メール送信事業者)、サーバホスティング事業者、アカデミアの分野からの参加、DNS Federation、アンチスパム/アンチウィルスエンジンなどを提供するセキュリティベンダ等々、近年はメールだけでなくSMSやSNSメッセージングなどの分野に関わる様々な企業や学術機関が参加しています。

M^{3}AAWGの国際会議は年に3回行われており、通例として毎年2月頃にはサンフランシスコ、6月頃にはヨーロッパ地域のどこか、10月頃には北アメリカ大陸の都市で実施されています。

今回のM^{3}AAWGでも様々なテーマについての議論が行われましたが、特に多くの意見が飛び交ったのがDMARC 2.0についてのセッションでした。本稿では、2023年4月時点でのDMARC 2.0[*10]についての情報をまとめたいと思います。

なお、M^{3}AAWGはプライベートミーティングであり、その場での議論の詳細についての公開は禁止されているため、あくまでも公開情報に基づいた内容となります。また、IETFのドキュメント内ではDMARC-bisという表現が使われていますが、本稿ではすべて統一して"DMARC 2.0"と記載します。

現在、送信ドメイン認証技術としてインターネット上で利用されているDMARCは、RFC 7489[*11]にて国際規格として定義されています。DMARC 2.0では、いくつかの変更が現在検討されており、主な変更点は以下です。

- RFC 7489はInformational Categoryだったのに対して、DMARC 2.0は標準化を目指す
- DMARCポリシーの取得にPublic Suffix Listを使わずにDNS Tree Walkを利用するように変更
- 一部タグの廃止、並びに新しいタグの追加

1.3.1 Public Suffix ListとDNS Walk Tree

Public Suffix List[*12]とは、eTLD(Effective TLD)と呼ばれるドメインを管理する有志のリストです。Firefoxや Thunderbirdなどの製品で知られているMozillaが管理していたものが、現在はボランティアによって更新の運用がされています。

日本国内でよく見かけるドメインとしては、co.jpやne.jp、更には地方自治体で利用されているドメインなどが登録されています。

*10 IETF、Datatracker(https://datatracker.ietf.org/doc/draft-ietf-dmarc-dmarcbis/)。
*11 IETF、Datatracker(https://datatracker.ietf.org/doc/html/rfc7489)。
*12 github、「publicsuffix/list」(https://github.com/publicsuffix/list)。

RFC 7489の記述では、Public Suffix Listに登録されていないドメインに対して組織ドメインの探索や決定ができないことが問題視されていました。DMARC 2.0では、DMARC評価時の組織ドメインの決定とDMARCポリシーの探索に、DNS Tree Walkを利用することでその問題を解決したとのことです。

DMARCレコードを登録、公開するドメイン所有者視点では特に変更をする箇所はありません。それに対して、メールを受信してDMARCレコードを評価するIIJセキュアMXサービスのようなサービスを提供するIIJを含めたメールボックス事業者にとっては、ドメイン評価時のプログラムを改修する必要があると思われます。

1.3.2 タグの廃止、並びに新しいタグの追加

DMARC 2.0では表-2に示すタグの削除と追加が予定されています。ここで、IIJのようにDMARCレポートを受信していたり、DMARCレコードによるフィルタリングを実装している事業者が気になる点は、DMARC 2.0が導入された際に、新しいタグ並びに削除されるタグに対する対応をどのタイミングで開始・終了するべきなのか、という点です。DMARCレコードは、そのドメインを管理している組織によって実施されており、そのレコードの更新も各組織が任意のタイミングで実施します。そのため、削除予定であるタグへの処理対応をいつ止めるのか、追加予定であるタグへの対応をいつから開始するのかを見極める必要があると考えています。

1.4 送信ドメイン認証とSTARTTLS普及状況の報告

1.4.1 送信ドメイン認証統計

IIR Vol.55でも掲載をしたIIJ セキュアMXサービスで受信したメールについての送信ドメイン認証対応の割合をグラフにしています(図-1)。

SPF検証結果については、前回とほぼ同じ比率となっていますが、DKIM pass並びにDMARC passの比率はそれぞれ8ポイント程度増加しています。昨今のなりすましメール対策として、徐々にではありますが日本でもDKIM署名の実装やDMARCポリシーの指定を行う企業が増えていることが分か

追加/削除	タグ	説明
追加	np	存在しないサブドメインについてのポリシーを指定するフラグ(RFC 9091から抜粋)
追加	psd	Public Suffix Domainであるかどうかを示すフラグ
追加	t	pctフラグ(後述)の一部機能を残したフラグ
削除	pct	DMARCポリシーの適用割合を宣言するフラグ
削除	rf	DMARC検証失敗レポートのフォーマット指定
削除	ri	DMARC集計レポートの作成要求間隔の指定(デフォルトは1日1回、それ以上は best effort)

表-2 DMARC 2.0にて追加/削除予定のタグ

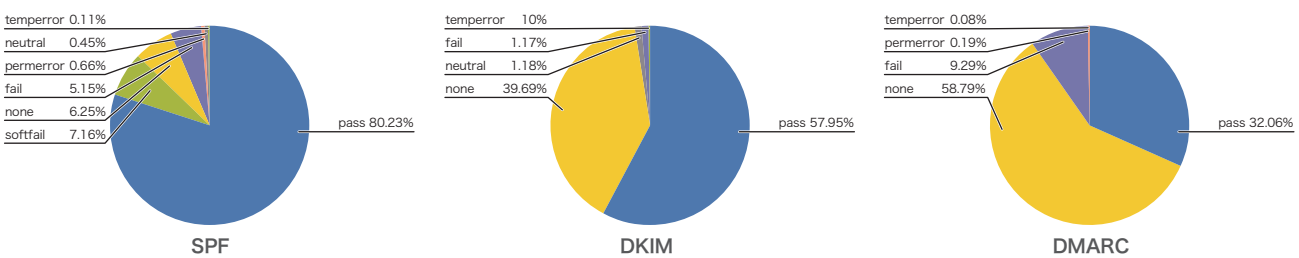

図-1 IIJセキュアMXサービスで受信したメールにおける送信ドメイン認証対応の割合

ります。2023年2月には、総務省よりクレジットカード会社各社に対して、DMARC導入によるフィッシングメール対策強化が要請されています[*13]。

1.4.2 受信/送信STARTTLS統計

数年前から、メールを取り巻くセキュリティ問題のトピックの1つとして、PPAP（パスワード付き暗号化ZIPファイルを添付したメールのやりとりの俗称）の廃止が挙げられることがよくあります。

以前のIIRでもトピックに挙げたEmotetと呼ばれるウイルスの流行を受けて、PPAP廃止の施策を進めている企業を取り上げた報道もありましたが、依然として日本のインターネットセキュリティを取り巻く問題として度々話題に上がっています。

1年前に発行したIIR Vol.55に、IIJもPPAPを廃止したというトピックがありましたが、今回は、添付ファイル暗号化ではなく通信経路の暗号化という観点から考察をしていきたいと思います。

企業においてPPAPは、実際にメールを送信される従業員の方並びにメール送信システムで人間が認識しやすい形で添付ファイルを暗号化するプロトコルですが、そもそもファイルを添付したメールの通信そのものが暗号化できれば、情報漏えいのリスクが軽減できるのではないか、ということで、今回は2022年4月から2023年4月のおよそ1年間、IIJセキュアMXサービスにおいてインターネットからのメール受信並びにインターネットへのメール送信の際に、どれくらいの割合のメールがTLS通信にてやりとりをされたのかを可視化しました。

IIJセキュアMXサービスでは、メールの送受信の際の経路暗号化に対応しています[*14]。

メールの送信プロトコルであるSMTPではTLS通信をする際にSTARTTLSという拡張プロトコルが使われます。対向のサーバに対して接続が確立したのちにTLS通信を施行し、対応しているTLSバージョン並びに暗号化方式でEnvelope From、Envelope To、DATA（メールのヘッダと本文データ）の通信を行います。対向サーバがTLSに対応していない場合は、そのまま平文でやりとりをしてメールを送信します[*15]。

まず、IIJセキュアMXサービスで2022年4月から2023年4月の約1年間で受信したメールについて、どれくらいの割合でSTARTTLSが利用されていたかを図-2に示します。

IIJセキュアMXサービスは、様々な企業のお客様に利用されているため、インターネット上の多様なサーバからの接続を観測しています。

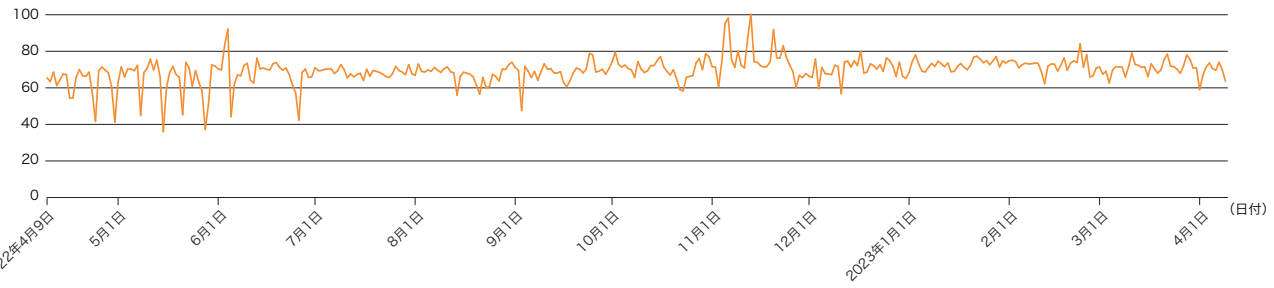

図-2 IIJセキュアMXサービスでの受信メールにおけるSTARTTLS利用の割合（2022年4月～2023年4月）

＊13 総務省、「クレジットカード会社等に対するフィッシング対策強化の要請」（https://www.soumu.go.jp/menu_news/s-news/01kiban18_01000184.html）。
＊14 IIJ、「経路暗号化」（https://www.iij.ad.jp/biz/smx/other.html#anc02）。
＊15 ietf.org、「SMTP Service Extension for Secure SMTP over Transport Layer Security」（https://www.ietf.org/rfc/rfc3207.txt）。

日によっては、特定のお客様宛に送信される標的型攻撃や、ばらまき型と呼ばれるフィッシングメールが観測されることもあります。

それらのメールは、インターネット上の様々なサーバから送られてきますので、STARTTLSを用いたSMTP通信の暗号化をせずにメールを送信するサーバも多くあると考えられます。

また、2022年4月から2022年5月の時期にSTARTTLSの割合が大きな幅で上下しているのは、おそらく当時流行していたEmotetがインターネット上の多くのサーバから送信されてきていたのではないかと推測できます。

次に、IIJセキュアMXサービス設備からインターネットへ送信したメールについてのSTARTTLS利用状況を図-3に示します。

IIJセキュアMXサービスからの送信メールについては、送信先サーバがSTARTTLSに対応していない場合を除いて経路暗号化が実施されるために、受信メールの状況と比べて高い比率で通信の暗号化がされている様子が窺えます。

IIJでも長年、添付ファイル自動暗号化機能を提供してきましたが、昨今のEmotetをはじめとする暗号化ZIPファイルを用いたウイルスへの対策として、数年のうちに機能提供の終了を予定しています。

通信経路の暗号化とメールに添付されているコンテンツの暗号化は一概に比較はできないと思いますが、このデータがPPAP脱却のお役に立てれば幸いです。

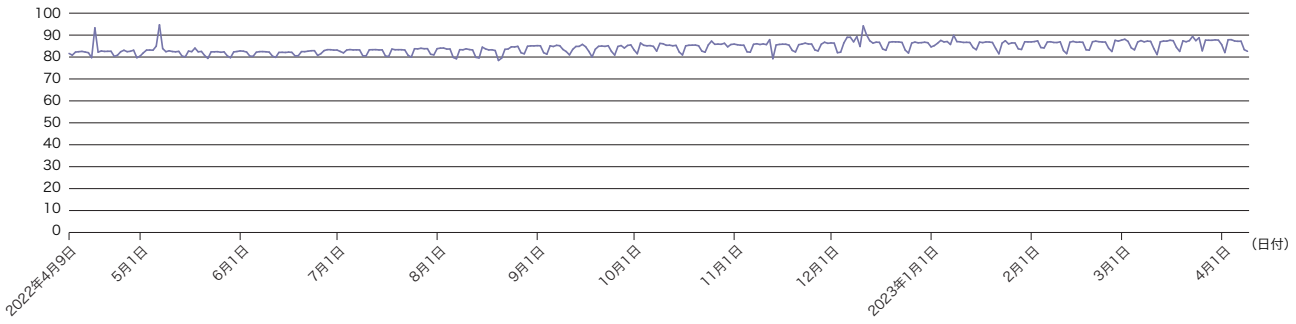

図-3 IIJセキュアMXサービスでの送信メールにおけるSTARTTLS利用の割合

執筆者:

1.1 電子メールと30周年、1.2 IIJポストオフィスサービス
古賀 勇（こが いさむ）
IIJ ネットワーク本部 アプリケーションサービス部 運用技術課 課長(兼)社長室。
2007年IIJ入社。メールサービスの運用業務に従事し、現場でメールに関する動向を調査。お客様のメールボックスを守るため、最新の攻撃手法や、迷惑メールのトレンド、対策情報などを発信。M^3AAWG、WIDE Project、openSUSEなどで幅広く活動中。

1.3 DMARC2.0について(M3AAWG topic)、1.4 送信ドメイン認証とSTARTLS普及状況の報告
今村 侑輔（いまむら ゆうすけ）
IIJ ネットワーク本部 アプリケーションサービス部 運用技術課 リードエンジニア。
2015年IIJ入社。メールサービスの運用業務に従事。IIJ Europeでの就業経験を活かし、日々グローバルに活躍中。

CTOとCTO Function Listerを使ったマルウェアの解析方法

2021年に行われたVirus Bulletin(VB2021 localhost)において、CTO及びCTO Function Listerというツールを発表しました[*1]。その後も断続的ではありますが、機能追加などの改善を行っています。本稿では、これらのツールをマルウェア解析のどの部分に適用しているのかを、実際の解析事例と共に紹介します。

なお、今回使用した検体はselfmake3という、SpiderPigと呼ばれるRATをダウンロードして実行するためのダウンロード型マルウェアで、標的型攻撃で使用されたものです。SHA256ハッシュ値は以下のとおりです。

```
7DA969010A55919AA66ED97A2D2D6D6A0BE3D8DC6151EEB6CEBC15E4F06D4553
```

2.1 起動方法と初期画面

CTO、CTO Function Listerは共にIDA Pro[*2]上でプラグインとして動作します。起動はEditメニューのPlugins、ツールバーのボタン、ショートカットキーのいずれかから行うことができます。図-1では、IDAウィンドウのツールバーの右端に中年男性風のアイコンがあるのが見えます。これらがCTOとCTO Function Listerのアイコンです。これらをクリックすることでウィンドウの左側にCTO Function Listerが表示され、右側にCTOが表示されます。CTOは主に関数呼び出しの親子関係を可視化するツールです。CTO Function Listerは関数一覧や、それぞれの関数が持つ特徴を抽出して保持し、フィルタリング機能でそれらの情報を一括検索するのが主な機能になります。図中では、それぞれのツールがIDAの逆アセンブルビュー(IDA View-A)で表示している"_WinMain"関数(正確にはMFCのAfxWinMain関数)のアドレスと同期し、そのアドレスの情報を各々表示しているのが分かります。

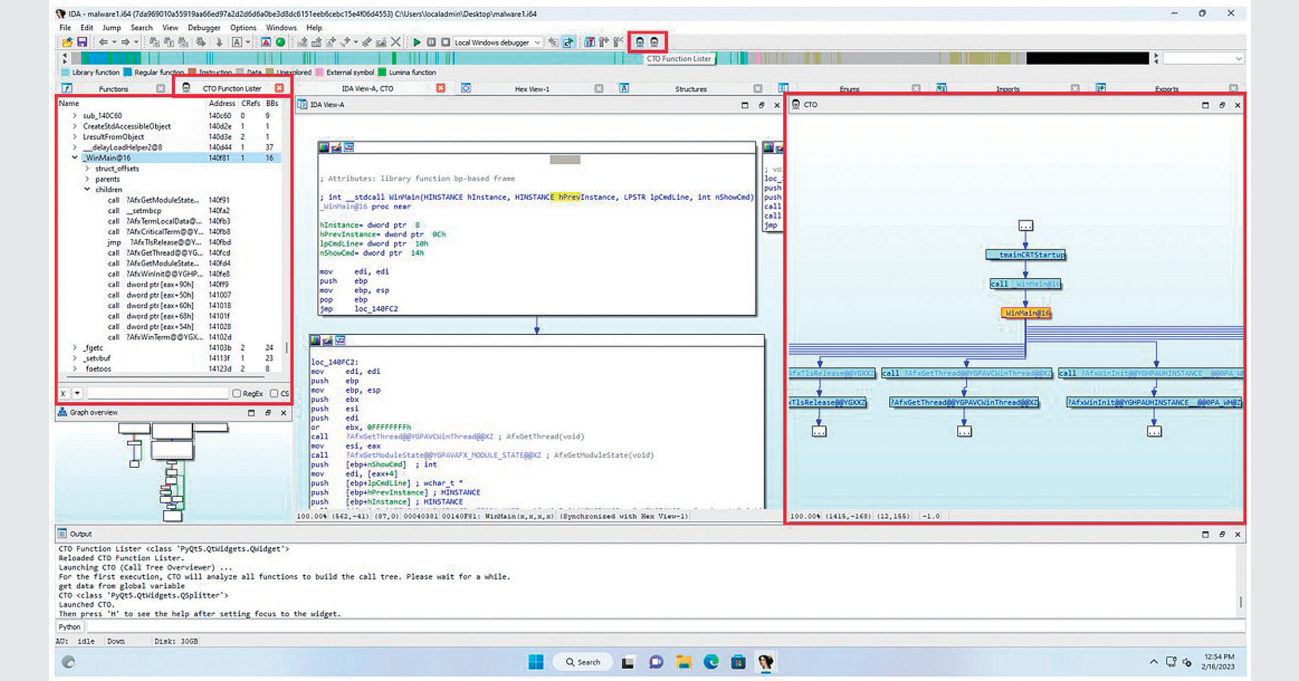

図-1 CTO及びCTO Function Listerの起動ボタンと各画面

＊1　VB2021 localhostでの発表は次のURLを参照のこと。CTO (Call Tree Overviewer) yet another function call tree viewer(https://vblocalhost.com/conference/presentations/cto-call-tree-overviewer-yet-another-function-call-tree-viewer/)。また、CTOとCTO Function Listerは筆者のgithubリポジトリで公開している(https://github.com/herosi/CTO)。

＊2　IDA Pro (https://hex-rays.com/ida-pro/)はマルウェア解析者にとって必須の逆アセンブラ、デコンパイラ。CTOやCTO Function ListerはIDAPython APIを使って記述をしている。

2.2 暗号化やエンコード/デコードルーチンの検出

マルウェア作者はコンフィグなどのデータや通信に暗号化やエンコードを行い、発見しづらくしていることが多いのは皆さんご存じのことと思います。その際、AESやRC4といった既存の暗号アルゴリズムが使われている場合と、簡易的にxor命令でカスタムエンコーディングが行われている場合があります。明示的にxorでカスタムエンコーディングをしている場合に加え、前述のアルゴリズムを含む多くの既知の暗号アルゴリズムにもxor命令が入っています。またCPUのレジスタ以上に長いデータを暗号処理するためには、必然的にループ構造も必要になります。そのためCTOでは、IDAが認識している関数を全走査し、xor命令を見つけ、それがループの中にあるかをチェックし、結果を一覧するビルトインコマンドがあります。また、該当した関数名がデフォルトから変更されていない場合は、"xorloop_"という接頭辞を付与して改名することで、関数名からも簡単に見つけられるようにもしています。

図-2はそのコマンドの実行方法です。CTOからもショートカットで実行することは可能ですが、ここではCTO Function Lister上のメニューからの実行方法について説明します。

まず、ドロップダウンメニューのボタンをクリックし、そこから、"Built-in scripts"、"Find xor instructions in a loop"を 選択します。解析中のプログラムサイズにもよりますが、本マルウェア(コードセクションのサイズが約280KB)であれば、2〜3秒程度で終わります。

結果はOutputウィンドウにも表示されますが、CTO Function Lister上で該当関数のみにフィルタして表示することも可能です。それを行うためには図-3のとおり、再度ドロップダウンメニューを開き、"Preset filters"、"xor instruction in a loop"を選択します。

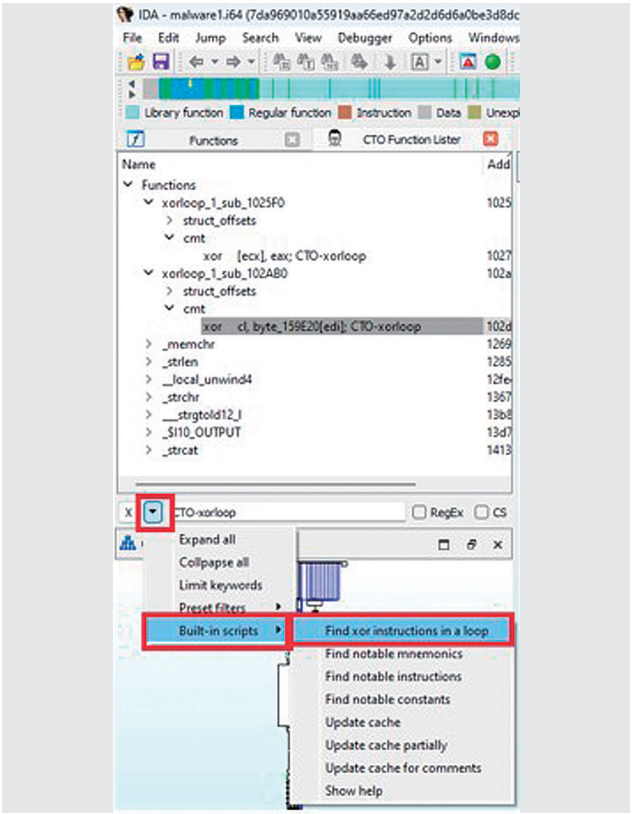

図-2 ループ内のXOR命令検出コマンドの実行方法

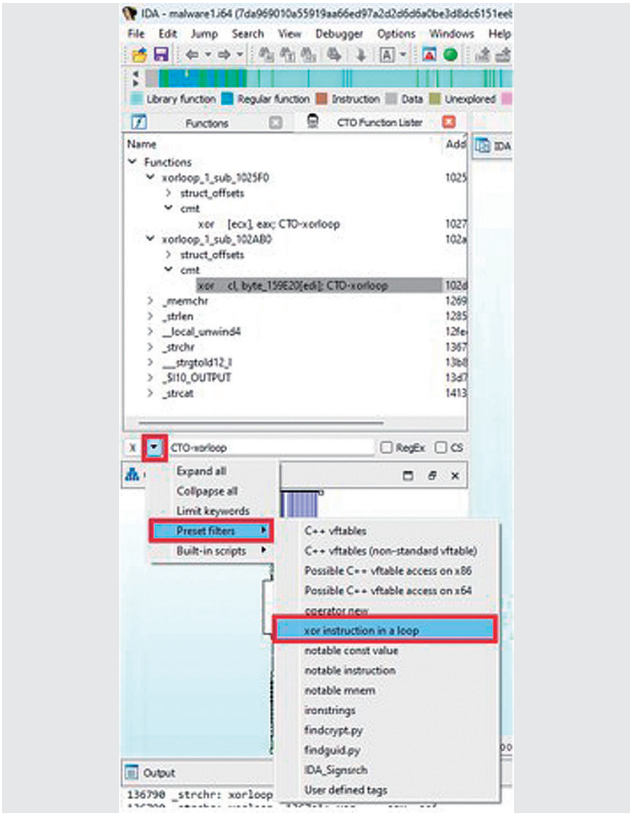

図-3 ループ内のXOR命令の検出結果表示方法

これにより、図-4のようにループ内にxor命令がある構造を持つ関数のみが列挙されます。IDAはFLIRTやLumina[*3]により、静的リンクされたC言語などのライブラリ関数の名前をある程度変更してくれます。ご覧いただいて分かるとおり、それらによって最初の2つの関数以外はすべて名前が付いています。よって、優先的に見なければならないのは最初の2つの関数（sub_1025F0とsub_102AB0）です。先ほどのコマンドは該当xor命令上に"CTO-xorloop"というコメントを付与しており、それをCTO Function Lister上でフィルタして表示しています。cmtというサブツリーがそれにあたり、CTO Function Lister上の該当行をクリックすることで、IDAの逆アセンブルビュー上で該当アドレスにジャンプすることができ、周辺のコードを確認することが可能です。

前述のコマンドで得られた2つの関数の周辺コードを確認したところ、1つは悪意のあるサーバからダウンロードしたペイロードを復号する際に使用するルーチン、もう一方は本検体に内蔵されているC&Cサーバのホスト名やIPアドレスなどのコンフィグデータを復号するルーチンでした。このような重要なコードを瞬時に見つけることができます。

今回はxor命令によるカスタムエンコーディングを例に取りましたが、AESなどの暗号アルゴリズムや、SHA256やMD5といったハッシュアルゴリズムは多くの場合、特徴的なmagic value[*4]やテーブルを持っています。そのような特徴を検出するために、findcrypt[*5]やIDA Signsrch[*6]などのサードパーティスクリプトやプラグインが公開されています。CTO

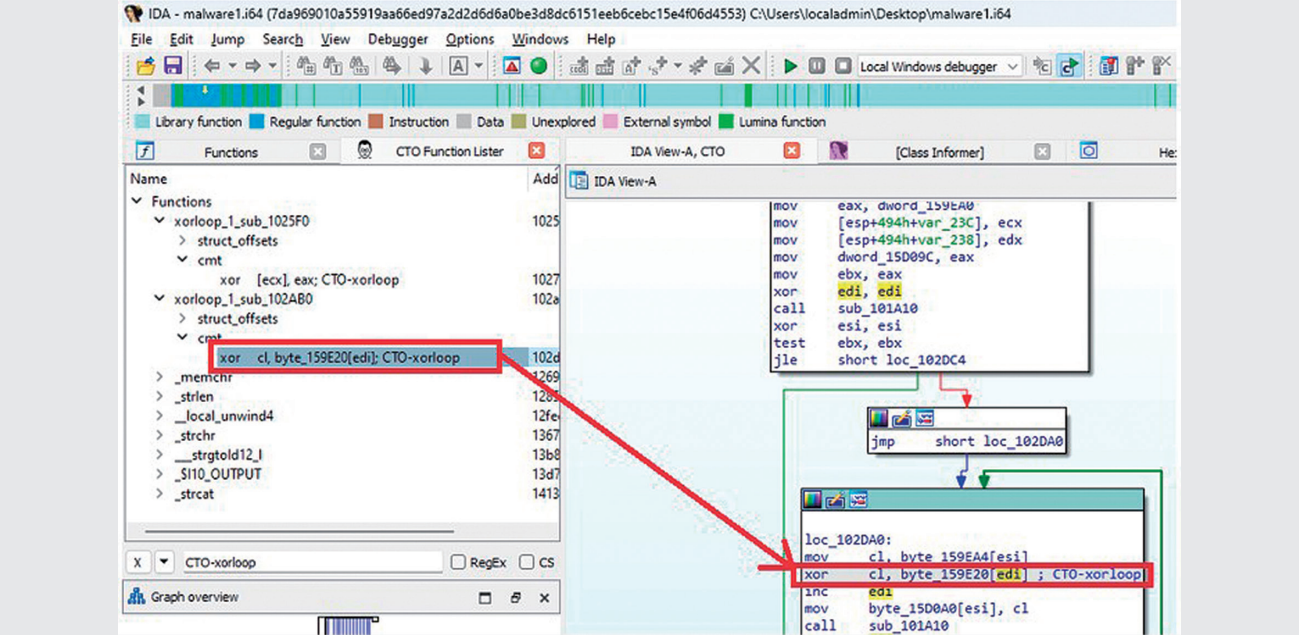

図-4 ループ内のXOR命令検出結果と、その付近のコード

＊3　FLIRTやLuminaはIDAの機能の一部で、どちらもパターンマッチングを行って既知の関数を検出し、関数名を変更することができる。FLIRTはローカルのデータベースを参照するのに対し、Luminaはクラウド上のデータベースを参照する。Luminaの場合、まずIDA利用者が命名した関数の情報をクラウド上のLuminaサーバに共有しておく。そして同じパターンを持つ関数が、他の利用者が解析中のバイナリ上で発見された場合、Luminaサーバ上の関数名が適用される。

＊4　magic valueとは、あるフォーマットのヘッダやフッタを一意に特定するための目印となる特定の文字列や数値。

＊5　findcrypt（https://github.com/you0708/ida/tree/master/idapython_tools/findcrypt）はIDA Pro用のサードパーティスクリプトで、著名な暗号アルゴリズムやハッシュアルゴリズムが持つテーブルやmagic valueをパターンマッチングにより検出することができる。findcryptには複数の実装があるが、このfindcryptはPythonで記述されており、拡張が容易であるため、筆者はこれを利用している。

＊6　IDA Signsrch（https://sourceforge.net/projects/idasignsrch/）はIDA Pro用のサードパーティプラグインで、findcryptと同様に暗号アルゴリズムやハッシュアルゴリズムを検出するために利用する。findcryptと似たようなツールだが、それぞれ守備範囲が異なる場合があるので、複数実行することがある。

Function Listerはこれらの結果も認識し、フィルタして表示することもできます。これらを組み合わせて活用することにより、暗号化/復号ルーチンやエンコード/デコードルーチンを効率良く発見し、迅速にその周辺コードを確認していくことができます。

2.3 パス探査

CTOは該当アドレスへの、もしくはそこからの経路を表示することができます。図-5はCTO Function Listerで発見したxor命令を右クリックして"Find the path(s) to this node"を選択した際の結果です。結果は画面右側にコールツリーグラフとして表示されます。

このグラフは、各関数アドレスやそれを参照するコードやデータの関連性を示してはいるものの、残念ながら純粋な実行パスではありません。なぜかというと、仮にある関数内に関数ポインタがあったとして、それが必ずその場で呼び出されるわけではないからです。例えば、関数ポインタをレジスタやHeapに格納して、かなり先の関数で呼び出すこともあるでしょう。またC++のvftableのような仕組みでは、間接呼び出しが多用されます。正確な実行位置を得るためにはクラスのインスタンスを追いかけ、すべてのアクセスを見つけ、vftableから関数ポインタを取り出して実行する処理を見つけなければならず、コードが非常に複雑になるからです。そのためCTOでは、コードが関数ポインタにアクセスした時点でそのアドレスを抽出して、このような親子関係のグラフを作っています。ただし、それでも十分役に立つものになっています。

この例であれば、パスの最初はdynamic initializerという関数です。この関数は、CRT(C-Runtime)内のinitterm[*7]という関数で処理されます。このマルウェアはMFCを使って記述されていることがコードを読むことで分かります。MFCアプリケーションはメインアプリケーションクラスをグローバル変数として宣言する必要があります。その宣言により、dynamic

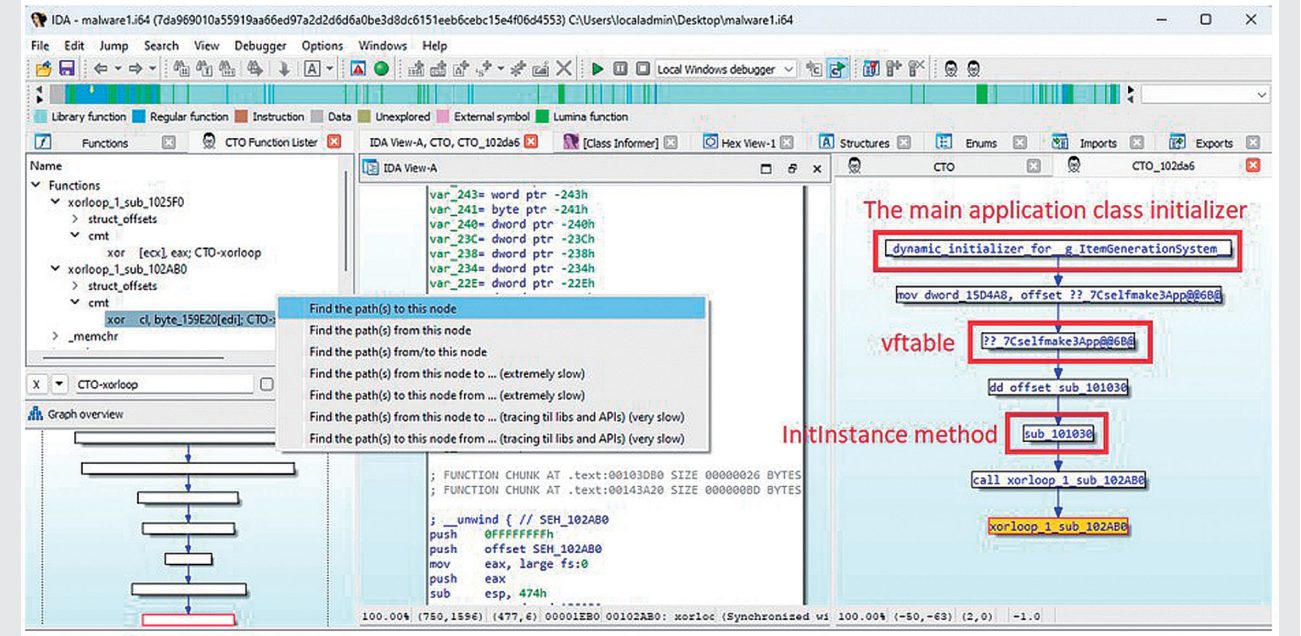

図-5 パス探査

*7 initterm (https://learn.microsoft.com/cpp/c-runtime-library/reference/initterm-initterm-e)はメイン関数の実行前に、CRT内でグローバルオブジェクトを初期化する関数。CRT内でinitterm関数呼び出し時にグローバル変数を第1、第2引数に取るので、IDAがこの関数を認識できていなくても、比較的見つけやすい。initermは2つの引数の間にある関数ポインタを順次実行する。各関数ポインタはdynamic initializer(https://learn.microsoft.com/cpp/c-runtime-library/crt-initialization)のコードによってカプセル化されている。そのコード内でグローバルオブジェクトのコンストラクタが実行され、そのクラスインスタンスがグローバル変数に格納される。

initializerでカプセル化されたメインアプリケーションクラスのコンストラクタがinittermから呼び出され、そのクラスインスタンスがグローバル変数に格納されます。表示されている関数名はLumina[*8]で自動的に付けられたものであり、for以降は明らかに間違った名称が付けられてしまっています。しかしCTOが表示したパスが示しているとおり、そのコンストラクタ内で、Cselfmake3Appというクラス名のvftableのアクセスが確認できます。またこのクラスがCWinAppクラスを継承していることがClass Informer[*9]の結果の1つである、クラス継承の階層構造からも確認できました。これらの事実から、Cselfmake3Appがこのマルウェアのメインアプリケーションクラスであることは明白です。

次にCselfmake3Appのvftableは、sub_101030という関数と接続しています。CTOは、関数内に存在するグローバル変数へのアクセスを抽出してキャッシュとして持っています。特にその中でも変数名の先頭やそのアドレスに付与されたコメントの最後にvftableやvtableという文字列を発見した場合、そのグローバル変数をvftableとして扱ってテーブルをパースし、一定の法則に従って関数ポインタ群をそのvftableに属するものであると認識します。IDAはRTTIを認識することが可能なため、このアドレスのコメントにvftableを含む文字列が付与されます。よってCTOの初回実行時にvftable解析処理が実行され、CTO内でsub_101030がこのvftableの一部であると認識済みです。そのためvftableに属する関数へのアクセスが発生すると、CTOはこのようにこの関数ポインタを仮想メソッドとして接続できるのです。図-6は、Cselfmake3Appのvftableのノード（上から3番目の「??_7Cselfmake3App@@6B@」）をCTO上でクリックしたときのIDAの画面を表示したものです。「IDA View-A」では

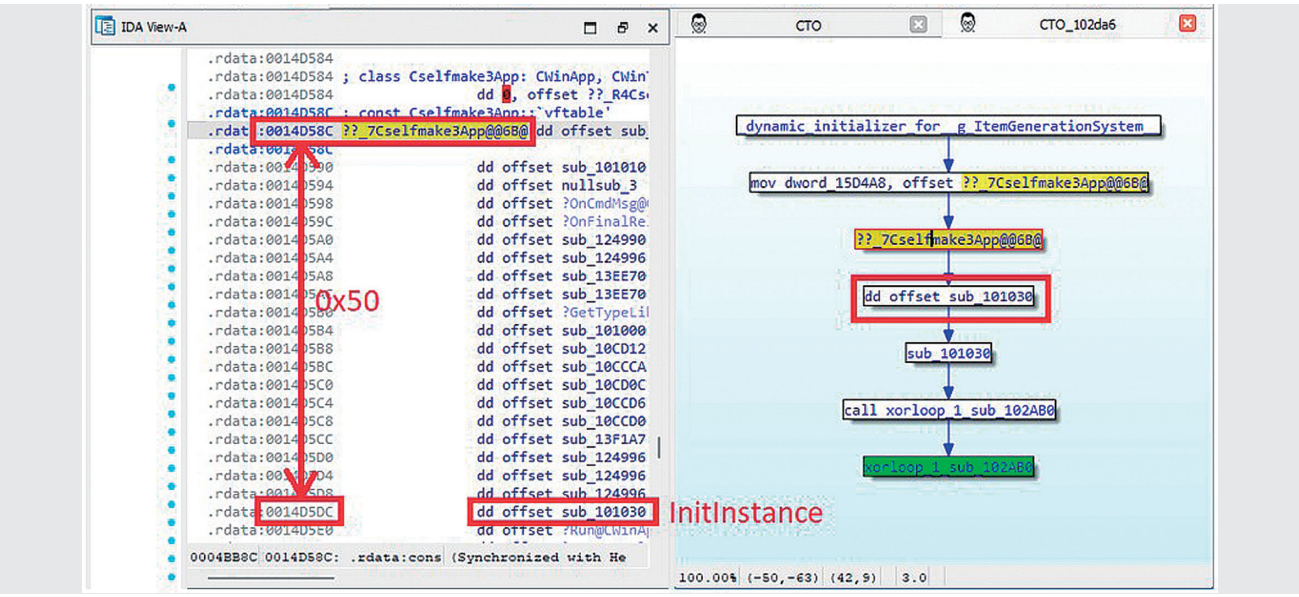

図-6 MFCメインアプリケーションクラスのvftableとInitInstanceの関数

*8 Luminaは前述のとおり、一般利用者が付けた名称が反映されてしまうため、名前の正確性は付けたユーザの技量に依存する。よって、あまり信用できないことが多く、参考程度にするのが良い。今回の例でも不正確な名前が付いている。

*9 Class InformerはIDA Proのサードパーティプラグインで、C++のRTTI（Runtime Type Information）を解析し、クラス名やクラス継承の階層構造を視認することが可能なツール（https://sourceforge.net/projects/classinformer/）。IDA 7.0からRTTIの解析自体はできているが、階層構造の表示やクラスの検索機能など、まだこのプラグインの方が優れた点があるため、現在でもこれを利用している。また、64-bit版IDAでもPE32上のクラス情報を復元できる改良版を筆者のgithubリポジトリで公開している（https://github.com/herosi/classinformer-ida8）。これは、IDAが8.0から32-bit版IDAを段階的に廃止し、64-bit版のみに移行し始めたためで、オリジナルのClass Informerは64-bit版IDA上でPE32を解析できなかったため、今後の動きをにらんでの処置。

vftableの先頭から関数ポインタが連続していますが、先頭から0x50の位置にsub_101030があることが分かります。ちなみに、32-bitのMFCメインアプリケーションクラスでは、vftableの0x50にInitInstanceという仮想メソッドが存在します。つまり、sub_101030はInitInstanceです。

MFCアプリケーションは前述のとおりCRT内でメインアプリケーションクラスのコンストラクタを処理した後、WinMain関数(厳密にはAfxWinMain)内でInitInstanceやRunなどのいくつかのメソッドを実行します。特にInitInstance関数は、MFCアプリケーションの規約上、必ずOverrideすることが定められており[*10]、ここが実質的にマルウェアのメイン関数となっていることが多いです。このマルウェアもInitInstance(sub_101030)が呼び出されており、その中でマルウェアのコンフィグをデコードするルーチン(sub_102AB0)を呼び出しているのがCTOのコールツリーグラフから簡単に分かります。

また、CTOのパス探査のもう1つの特徴は、クロスリファレンス[*11]さえあれば、グローバル変数(文字列を含む)であってもパスを作成できる点です。IDAにもProximity View (もしくはBrowser)という機能がありますが、関数にしか利用できません。これはCTOを使う利点の1つであると言えます。

注意点として、今回紹介したようにCTO Function Lister上でこの機能を使うためには、CTOをあらかじめ起動しておく必要があることを付け加えておきます。

2.4 std::string/std::wstringの検出

C++で書かれたマルウェアの多くは、std::stringやstd::wstringを文字列操作に使っています。これらのクラスはコンストラクタや一部のメソッドがインライン展開されてしまうため、一見ではこのクラスが使われていることを知るのが困難な場合があります。ただし、クラスレイアウトの初期化を行うコードには特徴的な初期値が使われるため、多少の誤検知はあるものの、単純なパターンマッチングで検出することが可能です。

これらの検出は、先ほども紹介したCTO Function Listerのドロップダウンメニューから"Built-in scripts"、"Find notable instructions"を選択することで行えます。また、フィルタしてコマンドの結果を確認するためには、同じくドロップダウンメニューから"Preset filters"、"Notable instruction"を選択すれば、表示することができます。

例として、マルウェアによってデコードされたコンフィグデータをパースしていくコードで使われていたstd::stringを見ていきます。図-7はCTOで検出したstd::stringの初期化部分のコードを表示したものです。図中1つ目の赤枠では、即値0xfでスタック変数が初期化されているのが分かります。これは、Visual Studioのstd::stringで長年使われている初期化コードの一部です。その2命令下(2つ目の赤枠)には、1バイト分のNULL文字でバッファの先頭部分(前述の0xfで初期化したアドレスより-0x14の位置)を初期化しているコードも見え

*10 次のURLにCWinAppを継承した場合にOverride可能なメソッドと、Overrideが必須なメソッドが記述されており、InitInstanceのみが必須となっている(https://learn.microsoft.com/cpp/mfc/overridable-cwinapp-member-functions)。
*11 クロスリファレンスはxrefsとも呼ばれ、IDAの最も重要な機能の1つ。特定のアドレスを参照するコードやデータを一覧してくれる。参照と被参照(xrefs fromとxrefs to)の2種類があり、IDAはどちらも表示、利用することが可能なのでクロスリファレンスと呼ばれている。CTOもこれを利用して親子関係を作り出している。

ます。これらがセットで存在した場合、筆者はstd::stringであると判断して構造体を適用するようにしています。

std::stringのクラスレイアウトはドキュメント化されていない内部的なものであり、我々が調べた限りでもVisual Studioのバージョンに応じて数パターンあります。このマルウェアはVisual Studio 2008でコンパイルされたものであることが分かりました。そのため、そのバージョンに対する適切な構造体をロードして、スタック上のstd::stringインスタンスの先頭にその構造体を適用することで、図-8のようにきれいにstd::stringを認識できます。

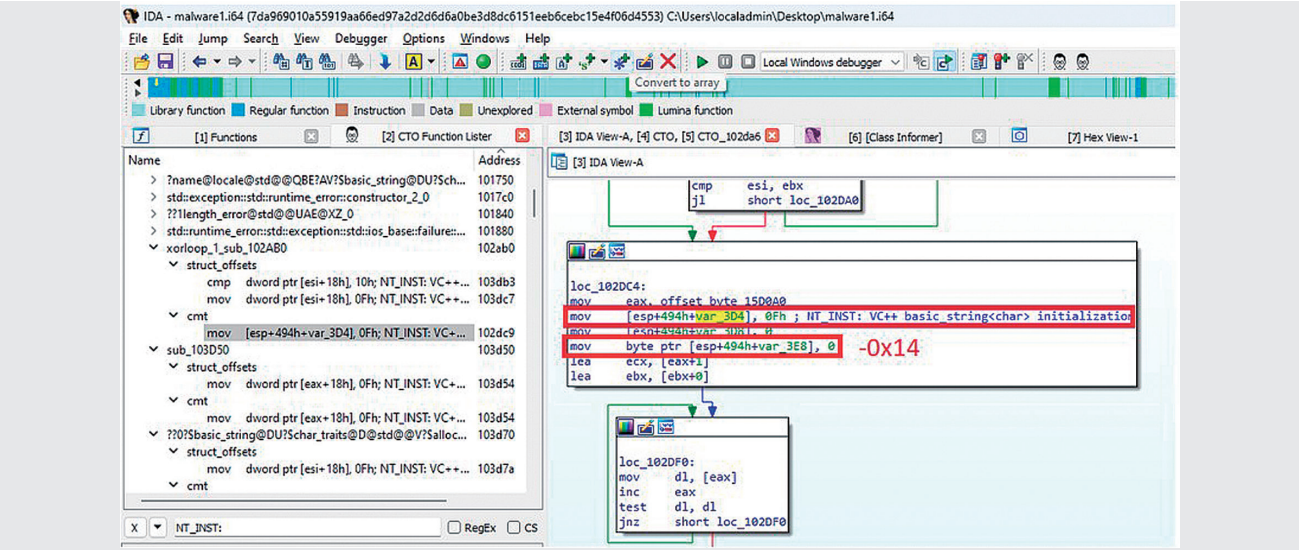

図-7 std::stringの検出

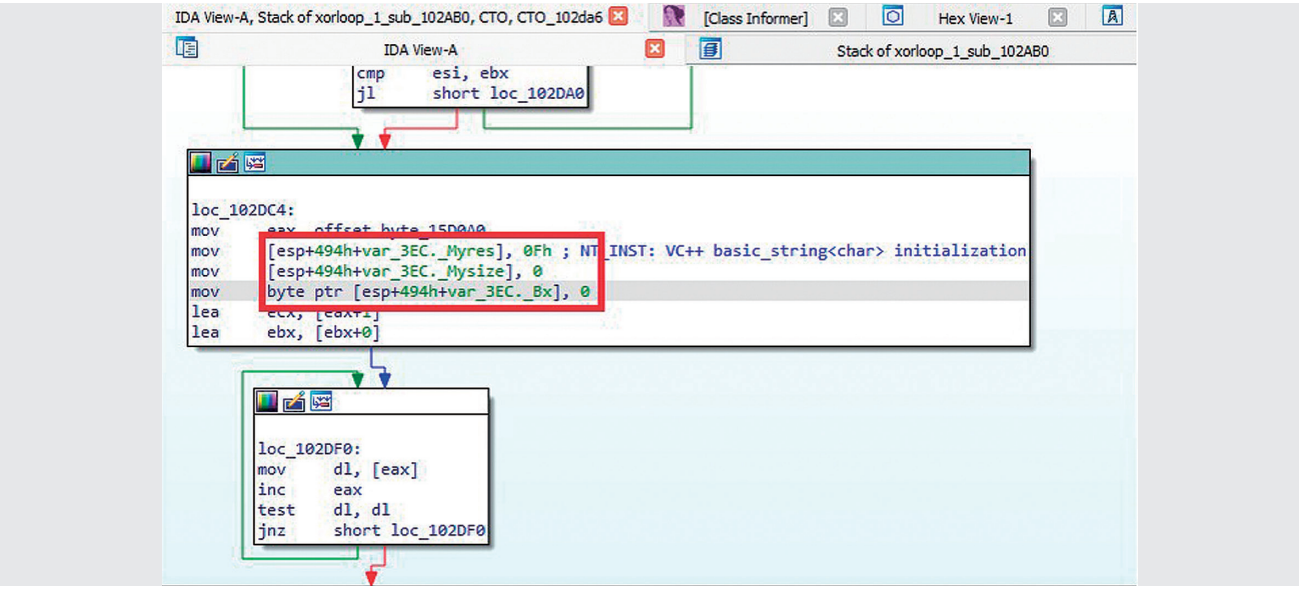

図-8 std::stringの構造体適用とメンバ変数の認識

2.5 実践 CTO/CTO Function Lister

紹介したIDAプラグインを使ってマルウェア解析をする講義を、2023年2月に行われたGCC 2023 Singapore[*12]で実施してきました。ランダムに選ばれた4〜6人で1チームを作り、講義の最後に本稿で示したマルウェア検体の特徴的な機能やコードをCTF形式で出題し、ゲーム感覚で解答してもらいながらマルウェア解析をしてもらいました。

受講生は各国から選抜された学生たちでしたが、IDAの使用やリバースエンジニアリング自体の経験がない人も多く、CTFを始める前に簡単なレクチャーを行いました。それでも、ここで紹介したようなテクニックを駆使して時短をしていくことで、優秀なチームはこのマルウェアであれば1時間半ほどで大部分の解析を終えられるようになりました。また三分の二以上のチームは3時間程度で重要な部分をほぼ解いていました。この講義ではリバースエンジニアリングのみを実施してもらうために実行ファイルそのものは渡さず、そのファイルをロードしたIDAのデータベースのみを渡して解析してもらいました。実行してしまえば、このマルウェアは動作が単純なので簡単に動作概要を把握できますが、コードを読んで正確にマルウェアの挙動を把握する能力も必要であるため、あえて厳しいやり方を受講生に課しています。そのような状況下でも、紹介したツールやテクニックを使って勘所を養うことで、急成長していく様子はとても感動的でした。

2.6 まとめにかえて

CTOやCTO Function Listerは、本稿で紹介した以外にも、過去のマルウェア解析で必要性を感じた機能を実装しています。これからも継続的に自動化などのアイデアを考えて実装していきます。これらのツールが皆さんのマルウェア解析の一助になれば幸いです。

執筆者:
鈴木 博志（すずき ひろし）
IIJ セキュリティ本部 セキュリティ情報統括室 マルウェア&フォレンジックアナリスト。
IIJのCSIRTチームであるIIJ-SECTのメンバーであり、社内、顧客のインシデント対応に従事。主にマルウェア解析とフォレンジック調査を担当。そこから得られた知見を元に、Black Hat（USA、Europe、Asia）、Virus Bulletin、FIRST TCなどの国際カンファレンスや、内閣サイバーセキュリティセンター（NISC）、総務省、法務省、IPA、産総研などで講演を行う。また、Black Hat USA、FIRST（Annual、TC）、Global Cybersecurity Camp、MWS、セキュリティキャンプ全国大会やサイバーコロッセオなど、国内外のカンファレンスや教育プログラムでの専門家や学生向けのトレーニング講師も兼ねる。特にBlack Hat USAでは日本人として初めてトレーニング講師に選ばれ、フォレンジック調査とマルウェア解析を使用したインシデントレスポンスに関するトレーニングを提供している。この分野では17年を越える経験を有する。

＊12 GCC（Global Cybersecurity Camp）（https://gcc.ac/）はアジアを中心に8か国（2023年2月現在）から選抜された学生などへの教育プログラム。2023年で5回目となり、2月にシンガポールで行われ、本文のとおり筆者たちもそこで講義を行った（https://gcc.ac/gcc_2023/lectures/#reverse-engineering-malware-written-in-c-with-ida-and-semi-automated-scripts）。IIJは業界に貢献する目的で、第1回から継続的にトレーニングを提供している。

クロスデバイスフローによる認証・認可

3.1 はじめに

スマートフォンの急速な普及と機能の進化は、私たちの暮らしに大きな変化を与え続けています。現在では、日常のあらゆる場面でスマートフォンが活用されるようになりました。インターネット上のサービスを安全に利用する上で欠かせない、認証・認可の手続きも例外ではありません。本稿では、近年、注目を集めている、スマートフォンを活用したクロスデバイスフロー(Cross-Device Flow[*1])と呼ばれる認証・認可方式について解説します。

クロスデバイスフローとは、あるサービスを利用する端末(PCやスマートテレビなど)と、その利用のための認証・認可を行う端末(スマートフォンなど)が分かれている認証・認可方式のことです。例えば、「スマートテレビ上で動画配信サービスを利用したい。しかし、テレビのリモコンではユーザIDやパスワードが入力しづらいので、サインイン手続きをスマートフォンで代わりに実行する」といった例が挙げられます。

このケースでは、「入力インタフェースに制約がある端末でサービスを利用したい」という課題をクロスデバイスフローで解決しています。この他にもクロスデバイスフローが必要とされる場面は数多くあり、例えば、「共用端末や公共端末など、機密情報の入力を避けたい端末でサービスを利用したい」「既存の認証・認可フローに多要素認証を追加したい」「複数の端末で同じ秘密鍵を用いた認証・認可を行いたいが、秘密鍵のコピーは避けたい」といったように幅広い活用方法が提案されています。

クロスデバイスフローを実現するための標準仕様は策定中のものを含め複数あり、それぞれユースケースが異なります。以下がその代表的な仕様です。それぞれ詳しく見ていきましょう。

- OAuth 2.0 Device Flow
- OpenID Connect CIBA Flow
- OID4VPのCross Device Flow
- SIOPv2のCross-Device Self-Issued OP
- CTAP v2.2のHybrid transports

3.2 OAuth 2.0 Device Flow

OAuth 2.0 Device Authorization Grant (RFC8628)[*2]はOAuth 2.0の認可フローの1つです。IETFによって2019年に標準化されました。一般にDevice Flowと呼ばれます。スマートテレビやデジタルフォトフレーム、プリンタなど、ユーザからの入力に制約があるデバイス上で実行するアプリケーションを、別デバイスで補助することを目的に設計されたクロスデバイスフローです。前節で紹介したスマートテレビで動画配信サービスアプリケーションを利用するケースは、まさにDevice Flowの代表的な使用例です。

Device Flowは認可のフローであり、クライアントアプリケーションがサービス(通常、APIとして提供される)を利用するためのアクセストークンを、認可サーバに発行してもらうためのプロトコルです。認証のフローではないので、Device Flowの仕様の範囲には、クライアントアプリケーションがエンドユーザを認証する機能(IDトークンの発行など、エンドユーザを識別する機能)は含まれていません。認証も行いたい場合は、OpenID Connectなどと組み合わせて実装する必要があります。

*1 Cross-Device Flows: Security Best Current Practice(https://datatracker.ietf.org/doc/draft-ietf-oauth-cross-device-security/)。
*2 RFC 8628:OAuth 2.0 Device Authorization Grant(https://datatracker.ietf.org/doc/rfc8628/)。

図-1はDevice Flowによる認可フローの例です。OAuth 2.0では、サービスを利用するアプリケーションをクライアント、承認を行うアプリケーション(通常はWebブラウザ)をユーザエージェントと呼びます。

1. エンドユーザが対象デバイス上のクライアントを起動します。
2. クライアントは認可サーバに対し、認可リクエストを送信します(a)。
3. 認可サーバはそのレスポンスとして、デバイス検証コード(デバイスコード)、エンドユーザ検証コード(ユーザコード)、及びエンドユーザにアクセスしてもらう検証用URLを返します。
4. クライアントは受け取ったユーザコードと検証用URLを画面に表示します。検証用URLは通常、QRコードとして表示されます。
5. エンドユーザはスマートフォンなどでQRコードを読み取り(b)、検証用URLを取得します。
6. 取得した検証用URLにユーザエージェントでアクセスします。認証が求められるので、サインインします。
7. サインイン後の画面に、ユーザコードが表示されます。(エンドユーザにユーザコードの入力を求めるパターンもあります)。
8. 一方、クライアントは、エンドユーザがユーザエージェントを操作している間、認可サーバに対してアクセストークン取得リクエストの送信を繰り返します。リクエストにはパラメータとしてデバイスコードを含めます。
9. エンドユーザはクライアントが表示しているユーザコードとユーザエージェントが表示しているユーザコードが一致していること、及びその他の注意事項を確認の上、承認します(c)。
10. 認可サーバはアクセストークンを発行し、アクセストークン取得リクエストに対するレスポンスとしてクライアントに返します(d)。

他のOAuth 2.0の認可フローとDevice Flowの大きな差異は、フロントチャネルの実現方法にあります。フロントチャネルとは、クライアントとユーザエージェント間の連携のことです。OAuth 2.0 Authorization Framework(RFC6749)[*3]で定義されているAuthorization Code FlowがOAuth 2.0の認可フローでは最もよく使われますが、Authorization Code Flowではフロントチャネルをリダイレクト(HTTPリダイレクト、もしくはディープリンク(iOSのUniversal LinksやAndroidのApp Links)のようなアプリケーション間連携の仕組みを用いたリダイレクト)によって実現します。しかし、クライアントとユーザエージェントが別デバイスで動作するDevice Flowではリダイレクトが使用できないので、代わりにQRコードの読み取りや目視と手入力などによるエンドユーザの仲介によって実現しています。

Device Flowのフロントチャネルの実現方法はシンプルで、特殊なハードウェアも不要な実装しやすいものである一方、セキュリティ的には強固とは言えない面があります。ソーシャルエンジニアリングや中間者攻撃によって、アクセストークンを詐取されたり、悪意のあるサイトに誘導される危険性が指摘さ

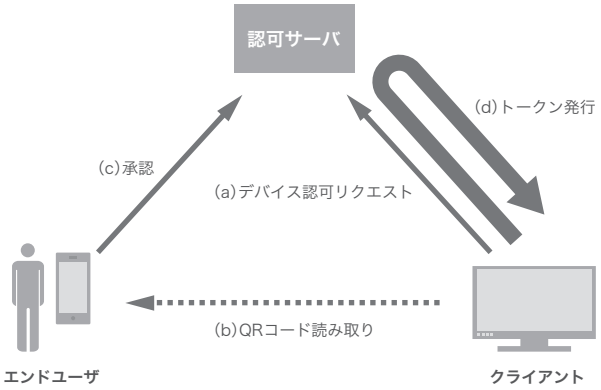

図-1 Device Flowによる認可フローの例

*3 RFC 6749 - The OAuth 2.0 Authorization Framework(https://datatracker.ietf.org/doc/rfc6749/)。

れています。従って、Device Flowは機密性、重要性の高いデータへのアクセスを伴うクライアントでの使用は避けるべきとされています。

3.3 OpenID Connect CIBA Flow

OpenID Connect Client-Initiated Backchannel Authentication Flow[*4]はOpenID Connectの認証・認可フローの1つで、略してCIBAと呼ばれます。OpenID Foundationによって2021年に標準化されました。CIBAもDevice Flow同様、サービスを利用するクライアントとその承認操作を別々のデバイスで実行可能とするクロスデバイスフローです。しかし、そのコンセプトは大きく異なります。Device Flowでは一人のエンドユーザが両方のデバイスを操作するケースがほとんどですが、CIBAはそれぞれのデバイスを別々のユーザが操作するケースを考慮して設計されています。これにより、CIBAは以下のようなユースケースを可能にします。

- コールセンターの担当者が電話口の顧客の情報を取得するケース。担当者は顧客から聞いた会員番号を顧客管理システムで検索する。すると、顧客のスマートフォンに通知が飛び、個人情報を開示して良いか確認する旨のメッセージが表示される。顧客が承認してようやく、顧客管理システムは検索結果として顧客の情報を表示する。この仕組みにより、担当者が無断で顧客情報を閲覧するような情報漏えい事故を防げる。
- 店舗でのクレジットカード決済の承認に使用するケース。顧客が店舗で買い物をする際、レジにてクレジットカードで支払おうとすると、顧客のスマートフォンに通知が飛び、支払い内容の確認メッセージが表示される。顧客が承認すると決済が完了する。サインや暗証番号の入力などよりも確実な本人確認と同意取得が実現できる。

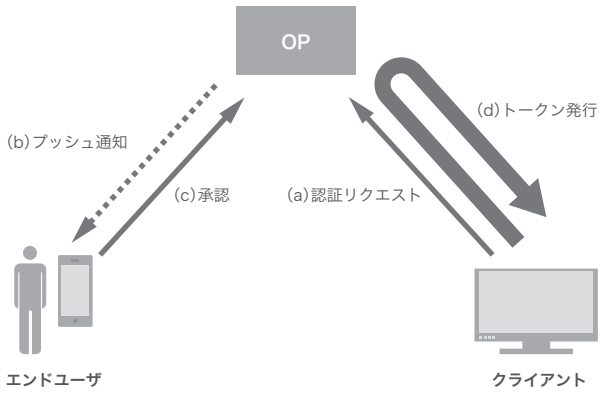

図-2 CIBAによる認証・認可フローの例

それでは、このような認証・認可をどのようにして実現しているか、CIBAの詳細を見ていきましょう(図-2)。まず、用語を整理しておきます。CIBAでは、クライアントを実行するデバイスを消費デバイス(Consumption Device)、エンドユーザが承認操作を実行するデバイスを認証デバイス(Authentication Device)と呼びます。認証デバイスは通常、スマートフォンです。なお、認証デバイス上で承認操作を行うアプリケーションを指す言葉はCIBAでは定義されていませんが、本稿では便宜上、これを認証アプリケーションと呼ぶことにします。また、CIBAはOpenID Connectの認証・認可フローなので、アクセストークンと共にIDトークンも発行するプロトコルです。これらを発行するサーバをOpenID Provider(OP)と呼びます。

1. クライアントがOPに対して、認証リクエストを送信します(a)。リクエストには、対象のエンドユーザを指定する識別子をパラメータとして含めます。
2. OPは認証リクエストに対し、レスポンスとして認証リクエストIDを返します。
3. OPはエンドユーザデータベースから、エンドユーザに紐付く認証デバイスを検索し、その認証デバイスに同意取得メッセージを送信します(b)。多くの場合、プッシュ通　知(Apple Push Notification Service や Firebase Cloud Messaging)が使用されます。
4. 同意取得メッセージを受信した認証デバイスは認証アプリケーションを起動し、画面にメッセージを表示します。
5. エンドユーザは同意するか拒否するか選択し、回答をOPに送信します(c)。
6. エンドユーザが同意した場合、OPはアクセストークンとIDトークンを発行します。

*4 OpenID Connect Client-Initiated Backchannel Authentication Flow - Core 1.0(https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html)。

7. クライアントはトークンエンドポイントをポーリングし、トークンを取得します(d)。このときのリクエストにはパラメータとして認証リクエストIDを含めます。クライアントが通知用のエンドポイントを公開できる場合は、ポーリングせずにトークン発行時に通知を受信するオプションもあります。

なお、OPと認証デバイス間で行うやり取りのプロトコルは、CIBAの仕様では定義されていません。通信方法も転送するメッセージ仕様も実装者による設計に委ねられています。

CIBAがDevice Flowや後ほど触れる他のクロスデバイスフローと大きく異なるのは、フロントチャネルを使わず、バックチャネルのみで完結するフローである点です。バックチャネルとはクライアントとOP間、及び認証アプリケーションとOP間のやり取りを指します。クライアントと認証アプリケーション間の直接のやり取りがないので、先のコールセンターの例のように、消費デバイスと認証デバイスが地理的に離れたユースケースに対応できます。

もう1つ、CIBAの大きな特徴はクライアント主導型(Client-Initiated)の認証・認可フローであるという点です。他のOAuth/OpenID Connectのフローの場合、クライアントが任意のタイミングでエンドユーザのリソースにアクセスしようとすると、一度認証・認可を済ませた後、その後長期間、クライアントがトークンを保持し続けることになります。CIBAでは、クライアントからの要求があるごとに短期間有効なトークンを発行するという運用が可能になるので、より柔軟なリソース保護ができます。

このように、CIBAは他の認証・認可フローでは対応が難しいユースケースをサポートする貴重なフローです。特に金融業界から注目を集めており、OpenID Foundationが普及を図っているFAPI(金融などの強固なセキュリティを必要とする分野向けのOAuth/OpenID Connectのプロファイル)[*5]にも組み込まれています[*6]。

3.4 OID4VPのCross Device Flow

本節では、現在、OpenID Foundationによって策定が進められているOpenID for Verifiable Presentations[*7](略して、OID4VPと呼ばれます)について解説します。まずは、OID4VPの前に、OID4VPが取り扱う対象であるVerifiable Credentialについて簡単に説明しておきましょう。

Verifiable Credential(VC)とは、検証可能なデジタル資格証明書のことです。例えば、パスポート、卒業証明書、社員証、等々を電子化したものです[*8]。発行者がデジタル署名を行い、第三者が検証できるようになっています。標準化されたVCの規格は複数あり、デジタル運転免許証用のISO/IEC 18013-5 Mobile driving licence(mDL)[*9]や汎用のデータ形式であるW3C Verifiable Credentials[*10]などがあります。

VCは通常、資格所有者のウォレットと呼ぶアプリケーションに格納します。しかし、mDLやW3C Verifiable CredentialsはVCのデータ仕様に過ぎないので、発行者から取得してウォレットに格納するプロトコルや、ウォレットから検証者に提示するプロトコルは定義されていません。これらのプロトコルの設計は実装者に委ねられています。例えば、医療情報用のVC(W3C Verifiable Credentials形式)を扱う仕様であるSMART Health Cards(SHC)[*11]はその一例です(ちなみに、デジタル庁が提供している新型コロナワクチン接種証明書アプリもSHCを採用しています[*12])。OpenID FoundationはVCの普及を進めるため、これらのプロトコルの標準化を進めています。VCを発行するプロトコルであるOpenID for Verifiable

*5 FAPI 2.0 Security Profile(https://openid.bitbucket.io/fapi/fapi-2_0-security-profile.html)。
*6 FAPI:Client Initiated Backchannel Authentication Profile(https://bitbucket.org/openid/fapi/src/master/Financial_API_WD_CIBA.md)。
*7 OpenID for Verifiable Presentations(https://openid.net/specs/openid-4-verifiable-presentations-1_0.html)。
*8 Verifiable Credentials Use Cases(https://www.w3.org/TR/vc-use-cases/)。
*9 ISO/IEC 18013-5:2021 - Personal identification — ISO-compliant driving licence — Part 5:Mobile driving licence(mDL)application(https://www.iso.org/standard/69084.html)。
*10 Verifiable Credentials Data Model v1.1(https://www.w3.org/TR/vc-data-model/)。
*11 SMART Health Cards(https://smarthealth.cards/en/)。
*12 デジタル庁、「よくある質問:接種証明書の記載内容について」(https://www.digital.go.jp/policies/vaccinecert/faq_06/)。

Credential Issuance（略して、OID4VCIと呼ばれます）[*13]とVCを提示するプロトコルであるOID4VPです。OID4VCIもOID4VPもVCのデータ仕様からは独立しており、mDLでもW3C Verifiable Credentialsでもその他の形式でも扱えるようになっています。

では、本節の主眼であるOID4VPについて見ていきましょう。そもそも、VCを提示する、とはどういうことでしょうか。例えば、お酒の購入に年齢確認を求められるシナリオを想像してください。物理的な身分証明書の場合は、実店舗で店員に対面で提示するようなケースです。一方、VCは電子データですから、対面である必要はありません。オンラインショッピングで利用できます。

1. 酒店のWebサイトで酒類をカートに入れて、購入ボタンをクリックする。20歳以上であることを証明するVCを提示するよう求められる。
2. 提示ボタンをクリックすると、ディープリンクによってウォレットが起動し、酒店にVCを提示して良いか確認するメッセージが表示される。
3. 承諾するとリダイレクトで酒店のWebサイトに戻り、VCが酒店側に渡る。このとき、Selective Disclosureという仕組みにより、VCに含まれる生年月日以外の不必要な情報は渡さないようにできる。
4. 酒店のWebサイトは取得したVCを検証し、年齢を確認し、20歳以上であれば購入を許可する。

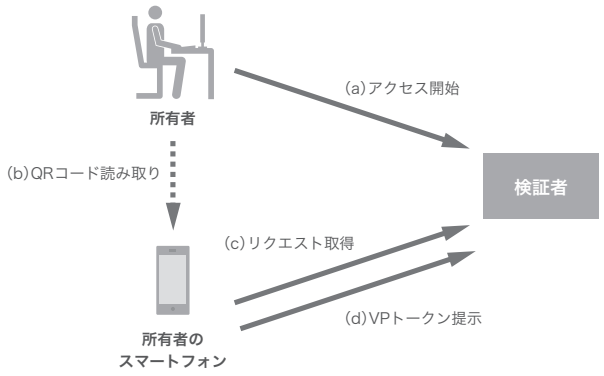

図-3 Cross Device Flowの認証例

上記は、OID4VPを実行するソフトウェアとウォレットが同一デバイス上に存在する、つまりアプリケーション間のリダイレクトが使用できる場合のフローで、Same Device Flowと呼ばれます。OID4VPにはCross Device Flowも存在します。先の例で言うなら、PCでオンラインショッピングをする際に、スマートフォンのウォレットに格納されているVCを使用するケースです。Cross Device Flowではリダイレクトの代わりにQRコードを使用して、両デバイス間を連携させます。

では、OID4VPのCross Device Flowを詳しく見ていきましょう（図-3）。OID4VPでは、VCを持つエンドユーザを所有者（Holder）、VCを提示する相手を検証者（Verifier）、VCを検証者に提示する際のデータ形式をVPトークンと呼びます。VPトークンには複数のVCを含めることができます。検証者のアプリケーションはHTTPSリクエストを受信するためのサーバが必要です。

1. 所有者がPCで検証者のサービスにアクセスします（a）。
2. 検証者アプリケーションがリクエスト取得用URIをQRコードに変換して画面に表示します。
3. 所有者はスマートフォンのウォレットでQRコードをスキャンします（b）。
4. ウォレットは取得した検証者サーバのリクエスト取得用URIにアクセスします（c）。
5. 検証者サーバはリクエストの詳細をレスポンスとして返します。リクエストの詳細には提示を求めるVCに関する詳細な要求事項が記述されています。
6. ウォレットは取得したリクエストに従って、提示するVCの内容について所有者に同意を求めるメッセージを表示します。
7. 所有者は内容を確認の上、VCの提示を承認します。
8. ウォレットは検証者サーバにVPトークンを送信します（d）。
9. 検証者によるVCの検証後、所有者はPCで検証者のサービスの利用を続行します。

*13 OpenID for Verifiable Credential Issuance（https://openid.bitbucket.io/connect/openid-4-verifiable-credential-issuance-1_0.html）。

OID4VPのCross Device Flowでは、最初のQRコードによるURI取得以降、検証者とウォレット間の通信はインターネットを使用する前提になっています。これを拡張し、インターネットが使えない環境でOID4VPを使用するためのOpenID for Verifiable Presentations over BLE[*14]という仕様も現在、策定中です。例えば、大規模コンサート会場や地下のライブハウスなど、スマートフォンが安定的にインターネットに接続できない環境で、代わりにBLE (Bluetooth Low Energy)による無線通信によって、電子チケットをVCとして提示するようなケースが想定されています。

VCの利用が想定されるユースケースは、日常のあらゆる場面に渡ります。OID4VPの標準化が完了し、本格的な普及が進めば、そのCross Device Flowを使う機会も非常に身近になることでしょう。

3.5 SIOPv2のCross-Device Self-Issued OP

Self-Issued OpenID Provider v2[*15](略してSIOPv2と呼ばれます)はOpenID Foundationが策定を進めている仕様で、OpenID Connectを拡張し、エンドユーザ自身がIDトークンを発行できるようにするための仕様です。以前の仕様(v2が付かないSIOP)はOpenID Connect Core 1.0[*16]の仕様の一部でしたが、現在、SIOPv2という独立した仕様として標準化作業が進められています。

OpenID Connectでは、OpenID Provider(OP)がエンドユーザの身元を証明するIDトークンを発行し、エンドユーザを認証したい第三者(Relying Party(RP))に提示します。その利用方法の代表例は、Webサービスを利用する際のソーシャルログイン(GoogleやAppleなどのアカウントでログイン)です。このとき、GoogleやAppleなどがOPとなり、Webサービスが RPとなります。SIOPは、このOPの役割をエンドユーザ自身が担い、自らのIDトークンを発行する仕組みです。

SIOPのメリットは、巨大プラットフォームによる中央集権的なID管理から離れ、エンドユーザ自身がID管理を行える点にあります。ソーシャルログインを使用すれば、どのRPを使用したかの情報がOPに収集されます。また、OPのアカウントが停止されてしまえば、RPの利用もできなくなります。こういった中央集権的なID管理の弊害を克服するため、SSI(Self-Soverign Identity、自己主権型アイデンティティ)という考え方が普及し始めています。SIOPはOpenID ConnectをSSIに対応させるための仕様です。

SIOPv2のプロトコルでは2つのフローを定義しています。1つが従来からあるSame-Device Self-Issued OPで、RPのクライアントアプリケーションとOPが同一端末で動作するフローです。RPとOP間の連携にリダイレクトを使用します。もう1つが、SIOPv2で新たに追加されたCross-Device Self-Issued OPで、OPが別デバイス(通常、スマートフォン)で動作するフローです。それでは、Cross-Device Self-Issued OPのフローについて見ていきましょう(図-4)。

1. エンドユーザがRPにアクセスします(a)。
2. RPが自己発行リクエストURIを画面に表示します。通常、QRコードとして表示されます。
3. エンドユーザはスマートフォンでQRコードをスキャンします(b)。自己発行リクエストURIはOPを起動するディープリンクになっています。

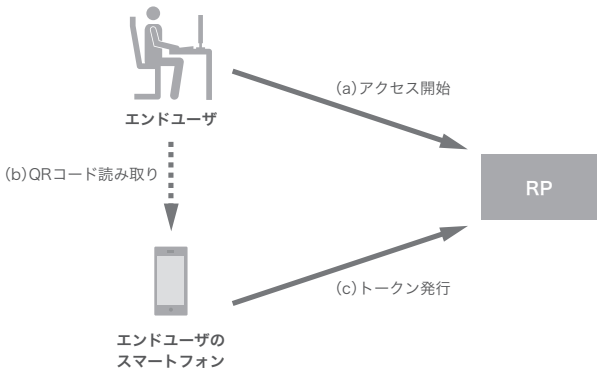

図-4 Cross-Device Self-Issued OPによる認証例

*14 OpenID for Verifiable Presentations over BLE(https://openid.bitbucket.io/connect/openid-4-verifiable-presentations-over-ble-1_0.html)。
*15 Self-Issued OpenID Provider v2 - draft 12(https://openid.bitbucket.io/connect/openid-connect-self-issued-v2-1_0.html)。
*16 Final: OpenID Connect Core 1.0 incorporating errata set 1(https://openid.net/specs/openid-connect-core-1_0.html)。

4. ディープリンクからOPを起動します。画面にはIDトークン発行の承認を求めるメッセージが表示されます。
5. エンドユーザが承認すると、OPはRPのバックエンドサーバに対して、発行したIDトークンを送信します(c)。

Cross-Device Self-Issued OPの他にも、SIOPv2では以前の仕様から以下のような機能強化が行われる予定です。

- IDトークンに含めるエンドユーザ識別子は、以前はエンドユーザの公開鍵のフィンガープリントを使う仕様でした。v2ではこれに加え、DID(Decentralized Identifier)を使用できるようになります。これにより、外部の検証可能データレジストリ(Verifiable Data Registry)が使用できるようになります。
- OID4VPと組み合わせて、VCをIDトークンと共に提示できるようになります。RPはVCを検証することで、信頼できる発行者が発行したVCとIDトークンを紐付けることができるようになります。VCの検証処理はRP(=検証者)側で完結しますので、VCの発行者に情報が収集されるということもありません。

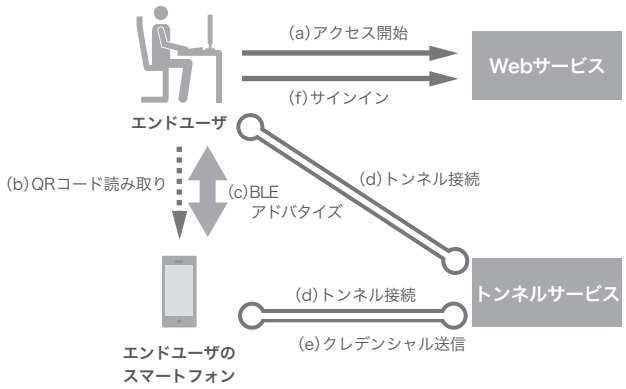

図-5 Hybrid transportsを使用したサインイン手順例

3.6 CTAP v2.2のHybrid transports

FIDO2[*17]はFIDO Allianceが推進している、パスワードレスでWebサービスにサインインするための認証技術です。FIDO2はW3C Web Authentication(WebAuthn)[*18]とそれを補完するClient to Authenticator Protocol(CTAP)[*19]から構成されています。WebAuthnはFIDO Allianceの協力の下、W3Cによって標準化された仕様で、認証器(Authenticator)と呼ばれる生体認証やセキュリティキーなどによる認証を使用して、Webサービスへのサインインを実現するための仕様です。CTAPはFIDO Allianceによって標準化された仕様で、USBやNFCで接続する、端末に組み込みでない外部の認証器を使用するための仕様です。

現在、策定中のCTAP v2.2では、Hybrid transportsという、スマートフォンを外部認証器として使用するためのプロトコルが提案されています。つまり、PCなどでWebサービスにサインインする際の認証をスマートフォンで行うことができるようになります。類似のソリューションは既にいくつかの事業者によって提供されていますが、いずれも独自仕様による実装でした。FIDO Allianceはこれを標準化しようとしています。Hybrid transportsを使用した認証はFIDO Cross-Device Authentication flow(CDA)[*20]と呼ばれることになるようです。ちなみに、似た語感のMulti-Device FIDO Credentials[*21]というものもありますが、これはエンドユーザが所有する端末間でクレデンシャル(認証資格情報)を同期する仕組みのことで、Passkeys[*22]とも呼ばれます。CDAとPasskeysは独立した仕様であり、Passkeysによってクレデンシャルが同期されていないPCとスマートフォン間でもHybrid transportsは利用可能です。

図-5はHybrid transportsを使用したサインイン手順の例です。

＊17 User Authentication Specifications Overview - FIDO Alliance(https://fidoalliance.org/specifications/)。
＊18 Web Authentication: An API for accessing Public Key Credentials - Level 3(https://www.w3.org/TR/webauthn-3/)。
＊19 Client to Authenticator Protocol(CTAP)(https://fidoalliance.org/specs/fido-v2.2-rd-20230321/fido-client-to-authenticator-protocol-v2.2-rd-20230321.html)。
＊20 Terms - passkeys.dev(https://passkeys.dev/docs/reference/terms/#cross-device-authentication-cda)。
＊21 White Paper:Multi-Device FIDO Credentials - FIDO Alliance(https://fidoalliance.org/white-paper-multi-device-fido-credentials/)。
＊22 Passkeys(Passkey Authentication)(https://fidoalliance.org/passkeys/)。

1. PCでFIDO2対応のWebサイトのサインイン画面を開きます(a)。
2. 使用する認証器の選択ダイアログが表示されるので、スマートフォンを選択します。
3. 画面にQRコードが表示されます。
4. スマートフォンでQRコードを読み取ります(b)。
5. スマートフォン側で認証アプリケーションが起動します。
6. このとき、フィッシングのリスク削減のため、BLEアドバタイズを用いてPCとスマートフォンが近接していることの確認が行われます(c)。
7. エンドユーザが指紋認証などを行います。
8. スマートフォンの認証アプリケーションとPCのWebブラウザ間にWebSocketを用いた「トンネル」と呼ぶ信頼できる安全な通信経路を確立します(d)。トンネルの仕様は実装者に委ねられています。
9. 認証アプリケーションはトンネル経由でWebブラウザにクレデンシャルを提供します(e)。
10. Webブラウザはクレデンシャルを使用してWebAuthnによるサインインを実行します(f)。

一度、トンネルによってリンクした後、次回以降の認証ではQRコードの読み取りの手順は省略されます。

FIDO2はWebサイトへのサインイン手順の置き換えに特化しているので、OAuth/OpenID Connectと組み合わせて利用することが可能です。従って、既存のOAuth/OpenID Connectがカバーしている領域の大半に、Hybrid transportsによるクロスデバイスフローを採用できる見込みがあります。現在、まだ策定中の段階ですが、実用化されれば、非常に大きな可能性を持つ仕様と言えるでしょう。

3.7 おわりに

以上、標準化済み、及び現在標準化に向けて策定中のクロスデバイスフローの仕様をご紹介しました。それぞれ特徴があり、対象となるユースケースは異なります。しかし、いずれもスマートフォンの持つ特性や機能(高い普及率、常時携帯、先進的な生体認証、QRコード対応、プッシュ通知対応、等々)を活用して、より安全で使いやすい認証・認可フローを実現しようという狙いは同じです。今後、クロスデバイスフローの普及が更に進むことで、オンラインサービスや取引の安全性は向上し、より快適なユーザ体験がもたらされることが期待されます。

執筆者:
四谷 兼三(よつや けんぞう)
IIJ 技術研究所 技術開発室。
次世代認証・認可関連技術の研究・開発に取り組んでいます。

IIJとDNSの30年

4.1 はじめに

従来はもっぱら学術機関での利用に限られていたインターネットに、商用サービスとして日本で初めて接続できるようにしたのがIIJでした。1993年11月のことです。それから今年で30年になります。本稿では、この30年の歩みをDNSの視点で振り返ります。

4.2 1990年代:接続サービスと共に

■ DNSはなかった

IIJは、1992年12月にインターネットイニシアティブ企画として設立され、翌5月に現在の社名に改められました。7月には「UUCPサービス」が、そして11月、ついに国内初の商用インターネット接続サービスである「インターネット接続サービス」が開始されました。

DNSはホスト名からIPアドレスを調べるサービスであり、インターネットを利用するには必須のものです。そのため、接続サービスを契約するとキャッシュDNSサーバがセットで付属してくるものとして扱われるのが一般的です。その現在の常識からすれば、IIJにおけるDNSの歴史もこのときに始まったものと考えそうですが、実はそうではありませんでした。提供されたのはあくまで専用線による接続サービスだけであり、キャッシュDNSサーバは含まれていなかったのです。

国内初のサービスとはいえ、利用者は学術ネットワークとしてのインターネットを知っている人が主でした。インターネットとは中央集権ではなく分散システムである。参加者は平等であり、必要なものは自分で何とかするか、足りなければ参加者同士の相互扶助で補完する。IIJとユーザの双方がこのような認識を共有していたと思われます。IIJが提供するのはあくまでインターネットに接続するところまで、DNSやメールやネットニュースなど接続した後の利用に必要になるものは自分で用意してほしいというのがIIJの立場であり、ユーザもそれが当然と受け止めていたようです。

■ 最初のDNSサーバ

ユーザが利用できるキャッシュDNSサーバが最初に提供されたのは、1994年5月の「ダイアルアップIPサービス」からでした。専用線を使った常時接続ではなく、電話回線を使って必要なときだけ接続するというもので、小規模な法人や個人でも利用できました。NTTの電話料金夜間定額サービスであるテレホーダイはまだ始まっておらず(1995年から)、接続したら必要な情報だけ収集してすぐ切断するという、当時主流だったパソコン通信と同じような使われ方でした。専用線接続と異なりユーザが必要なサーバを用意するにはそぐわない利用形態のため、キャッシュDNSだけはIIJ側で提供することになったようです。

キャッシュDNSサーバを提供するようになってからも、権威DNSサーバについてはユーザが用意してくださいという立場は変わりませんでした。とはいえ、DNSは教科書どおりであればプライマリ/セカンダリの2台構成にするものであり、当時はその両方をユーザで構築するのも簡単でなかったことから、セカンダリだけはIIJで引き受けるケースもありました。これは商売ではなく、インターネットの対等な参加者としての相互扶助の一環として行われていたようです。そのセカンダリDNSとしてゾーンを収容していたのは、なんとダイアルアップIPサービス用のキャッシュDNSサーバでした。現在の常識では考えられない構成ですが、当時は規模が小さく、相乗りさせることが合理的な判断だったようです。

相互扶助の精神で運用されていたセカンダリDNSとして重要なものに、JPドメインがあります。IIJに依頼されることになった経緯が分かる文書は今でもJPNIC上で公開されています[*1]。

IIJでは1994年からJPドメインのセカンダリを引き受けるようになり、以来現在に至るまで運用が続いています。その後は単なる扶助を越え、2001年にはいち早くIPv6に対応したり、2004年には海外拠点を設置してAnycastに対応するなど、先

＊1　JPNIC、「DNS管理グループ作業報告(一部、議題の元を含む)」(https://www.nic.ad.jp/ja/materials/committee/1994/0510/shiryou-2-4-1.html)。

進的な機能も積極的に取り入れ、JPドメインの安定運用のお手伝いをさせていただいています。

■ インターネットの普及

1990年代半ばは、インターネットが個人の手に届くようになった時期です。1994年9月、インプレスよりINTERNET magazineが創刊され、1995年11月にはTCP/IPを標準搭載したWindows95が日本でも発売されました。IIJ以外にも多くのISPが次々と誕生したのもこの時期です。

1996年12月、個人向け接続サービスの需要に応えて始まったのが「IIJ4U」です。大規模な個人向けサービス専用に設備構成を設計しており、可用性確保のため2台のキャッシュDNSサーバを異なるネットワークセグメントに置くという、今となっては常識の構成もここで初めて採用されました。

その頃には法人向け接続サービスでもキャッシュDNSサーバの提供が始まっていましたが、基本はお客様が自前で構築・運用するという点は変わらず、どうしてもそれができない場合にのみ使ってくださいという形態だったようです。これが標準提供になったのが、1997年11月に開始された「IIJエコノミー」からです。

SOHOという語は既に死語になった感はありますが、Small Office・Home Officeのことで、自宅や小規模な事務所などを仕事場にする働き方のことです。このSOHO向けの廉価版専用線サービスがIIJエコノミーで、従来の法人向け接続サービスとは異なり、ユーザが自分でサーバを設置、運用するような利用を前提としていませんでした。つまり、キャッシュDNSサーバもIIJが用意することになったのです。

このようにしてキャッシュDNSサービスの基礎が確立しました。この後ADSLや光回線、VPN、モバイルなど様々な形態での接続サービスがリリースされていきますが、回線側が変わっても、キャッシュDNSの方は設備の増強などをしつつも基本は変わりません。

4.3 2000年代：DNS単独のサービス開始

1999年、アメリカの株式市場はインターネット関連企業を中心に大きく値上がりしました。インターネットとはまったく無関係の企業でも、「ドットコム」をつけた社名に変更するだけで株価が2倍になったと言われています[*2]。この異常な高値は2001年の暴落で幕を閉じ、今では「ドットコム・バブル」として知られています。このドットコムとは、もちろんgoogle.comやyahoo.comなど、ドメイン名末尾の".com"のことです。

日本では1991年のバブル崩壊以降、平成不況、就職氷河期が依然と続いており、アメリカほどのバブルは起きませんでした。しかし、ソフトバンクの株式時価総額がトヨタ自動車に次いで2位になるなど、この時期に大きく成長した企業の多くがIT関連でした。

株価の上昇は一時的なバブルでしたが、自社専用ドメインを持つ習慣は完全に定着しました。大手企業やインターネット関連企業だけでなく、それ以外の企業も取得するのが当然になりました。ドメインを取得しても権威DNSに登録しなければ使えません。IIJはそれまで権威DNSが必要ならば自前で運用してくださいというスタンスでしたが、それには相応の専門知識が必要です。インターネットが一部の「分かっている人」向けだった時代はこのとき既に過去のものになっており、サーバ運用のノウハウを持たない人でも権威DNSに情報を載せたいという需要は大きくなっていました。

これに応えるため、2000年3月、ついにDNS単独のサービスを開始します。それが権威DNSの運用全般とWebからのゾーン情報の編集を可能にする「DNSアウトソースサービス」、セカンダリサーバだけを引き受ける「DNSセカンダリサービス」、そしてドメインの登録維持管理を行う「ドメイン管理サービス」です。

それまでJPドメインにはco.jp(企業)やac.jp(学術機関)などといった属性に基づく分類がありました。ドメインとは組織を表すものであり、だからこそ1組織につき1ドメインしか登

＊2　バートン・マルキール『ウォール街のランダム・ウォーカー』第4章。

録できませんでした。しかし2001年になって、組織に紐づかず、登録ドメイン数にも制限のない汎用JPドメインの制度が開始されました。これにより組織ごとのドメインだけでなく、商品やブランドごとに専用ドメインを登録する事例も増え、ドメイン登録数、そしてDNSアウトソースサービス/セカンダリサービスの利用は年々拡大していきました。

また、組織に属さない個人でもドメインを持つことは珍しくなくなりました。そのため、2002年3月、独自ドメインによるメールとWebそしてDNSのホスティングを個人向けに機能を抑えるかわりに低廉な料金で利用できる「IIJmioパーソナルドメインサービス」を、更に2003年3月にはDNSホスティングのみの「IIJmioシンプルDNSサービス」の提供を開始しました。

DNSアウトソースサービス、DNSセカンダリサービス、ドメイン管理サービスの3サービスはその後も機能追加や設備増強を繰り返しながら20年以上も継続する長寿サービスになりました。このうち特に大きかったのがDNSSEC対応と、オプションサービスである「サイトフェイルオーバーオプション」でしょう。

DNSはインターネットの利用に欠かせない要素技術の1つですが、1980年代に設計されたプロトコルであるため、当初想定されていなかった、あるいは無視して問題ないと思われていた欠点も含んでいます。その1つが、キャッシュポイズニング攻撃や中間者攻撃などにより応答パケットが偽造されたとしてもそれを検知するのが難しいという点です。詳細については長くなるのでここでは割愛しますが[*3]、DNSの登録情報に電子署名を付与し、応答を受け取った側はその署名を検証して情報の真正性を確認できるようにしたのがDNSSECです。

2010年7月、DNSのルートサーバでDNSSEC対応が開始され、同年12月にJPゾーンがDNSSEC署名されました。IIJのドメイン管理サービス、DNSアウトソースサービスでは2011年1月からDNSSECに対応しています。DNSSECは署名鍵の生成やゾーンへの署名など、これまでのDNS運用にはなかった煩雑な作業が必要で、そのため10年以上経った現在でも敬遠されがちな技術ですが、これらの作業をIIJのサーバ側が自動で行い、手間をかけずにDNSSECを利用できるようにしました。

DNSに登録された情報は静的なもので、応答を変更するには人間がそのたびに情報を書き換える必要がありました。2015年3月にDNSアウトソースサービスのオプションサービスとして提供開始された「サイトフェイルオーバーオプション」では、Webサーバなどの状態を外部から監視し、何らかの障害が発生したWebサーバをDNS応答からすみやかに取り除いたりスタンバイサーバに切り替えたり、またWebサーバが復旧したらDNSに自動で戻すことができるようになり、Webサーバの可用性向上を実現しました。

4.4 2010年代：攻撃との戦い

■ DDoS攻撃の拡大

2010年代はbotnetによるDDoS（分散型サービス拒否）攻撃が大規模化し、その対策に翻弄された時代になりました。

多数の端末から同時に多数のリクエストを送り付けることでサーバの処理能力を飽和させて可用性を失わせるのがDDoS攻撃です。コンピュータウイルスは2000年代に入ると不特定の端末に感染して広範囲に感染を広げるワームへと急速に進化し、更に組織化してbotnetとなりました。botnetを構成する多数のbotが攻撃者の指示を受け取ってDDoS攻撃する事件が各地で頻発するようになったのです。

その一方でインターネットのトラフィックは急速に増大を続けており、効率的なWebコンテンツ配信を目的とするCDN（Content Delivery Network）の利用も広がり、大規模配信を想定した構成のWebサーバであれば、ちょっとやそっとのDDoS攻撃でサービスが停止することはなくなりました。

ところで、DNSは1回の通信ではせいぜい数百バイトがやりとりされるに過ぎず、また効率的なキャッシュの仕組みも存在するため、CPUやメモリ、ネットワーク帯域といったリソースへ

＊3　例えば「DNSSECとは」（https://jprs.jp/dnssec/doc/dnssec.pdf）などを参照してください。

の負荷はWebやメールといったプロトコルと比べると微々たるものです。そのため、Webサーバが高性能化、広帯域化される一方で、DNSサーバについては最低限のリソースしか確保されない状態が長く続いていました。

多くの場合、攻撃者の狙いはWebサイトを閲覧不能にすることです。その目的を果たせるのであれば、攻撃目標がWebサーバである必要はありません。WebにアクセスするためにはまずそのサイトのIPアドレスを知る必要があり、ならばIPアドレスを入手するための仕組み、すなわち権威DNSサーバを機能不全に陥れることでも目的は果たせるのです。CDNに守られて大量の攻撃リソースを費やしても屈しないWebサーバを狙うよりも、そこそこの負荷で容易に飽和してしまう権威DNSサーバを攻撃する方が攻撃者にとって効率的です。

この代表的な事例が、Dyn(後にOracleにより買収)が2016年10月に受けたDDoS攻撃です[*4]。Dynは権威DNSサービス提供事業者としては超大手で、TwitterやSpotifyなど世界中の名だたるWebサービスがDynにホスティングしていました。このDynが数十万台の端末から6時間にわたってDDoS攻撃を受けて応答を返せない状態になり、Dynを利用していた多数のドメインにアクセスできなくなりました。

このDynへの攻撃に先立つこと4年、2012年にIIJも権威DNSサーバに対する大規模なDDoS攻撃を受けています。攻撃対象はWebとDNSをIIJにホスティングしていたお客様ドメインです。最初はWebサーバが攻撃されましたが、性能には十分な余裕があり、落とせないと認識した攻撃者が権威DNSに目標変更したのです。IIJの権威DNSサーバは当時としては潤沢なリソースを確保していましたが、広帯域化するWebサーバを落とすつもりのリソースを費やしてくるDDoS攻撃にはまったく足りておらず、応答を返しづらい状態に陥ったのです。

この事件はIIJのDNSサーバの設計思想を大きく転換させるきっかけとなりました。これ以前と以後で、DNSサーバのネットワーク構成が大きく変わっています。飽和攻撃に耐えられる十分な帯域を確保すること、仮にそれに耐えられなくてもAnycastにより被害を局所化する仕組みを備えること、またDNS以外のサービスに影響を波及させないようDNSサーバ群を専用ネットワークに分離することなど、多重の防御策を講じました。これらの対策を取り入れた設備構成変更はDNSアウトソースサービスなどの権威DNSサーバ、接続サービスなどで利用されるキャッシュDNSサーバ双方で順次実施されました。

■ オープンリゾルバ問題

DDoSは被害を受けるばかりではありません。残念ながら、IIJのDNSサーバが踏み台にされDDoS攻撃の「砲台」として利用されることもありました。

DNSは下位層のプロトコルとして主にUDPを用いますが、UDPはTCPに比べてクライアントによるIPアドレスの詐称が容易です。また、DNSの応答のパケットサイズは問い合わせの数十倍～数百倍に大きくすることができます。これを利用すると、悪意のある攻撃者は送信元IPアドレスを攻撃対象のものに詐称した問い合わせを踏み台となるDNSサーバに送り、そのIPアドレスにパケットサイズを増幅させた応答を返させるようにすることで、攻撃対象のネットワーク帯域を飽和させることができます。このような攻撃を「DNS amplification attack(DNS amp)」あるいは「DNS reflection attack」と呼びます(図-1)。

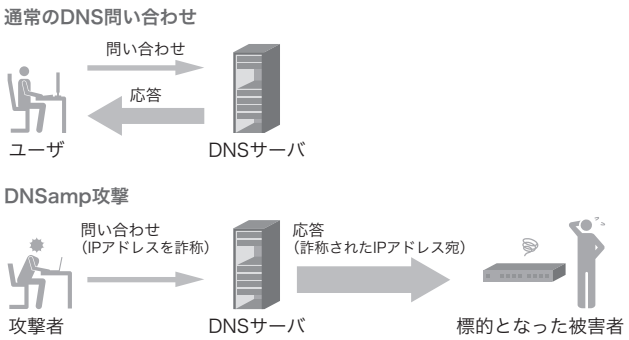

図-1 DNS amp

*4 Wikipedia、「DDoS attacks on Dyn」(https://en.wikipedia.org/wiki/DDoS_attacks_on_Dyn)。

DNS ampが成立するのは、DNSサーバが詐称されたIPアドレスに応答を返してしまうためなので、詐称されたIPアドレスからの問い合わせに応答を返さないようにすることで防ぐことができます。しかし、UDPを利用している都合上、詐称を見破るのは困難です。

本来キャッシュDNSサーバは組織ごとに用意するものであり、その組織内のユーザだけが利用できれば十分です。組織外からの問い合わせには応答しないようになっていれば、送信元IPアドレスが詐称されていたとしても、組織外に大きな応答を返すことがなくなり、DNS ampの踏み台として利用することができなくなります。しかし、インターネットが相互扶助の精神で動いていた時代から長い間キャッシュDNSサーバにはそういった制限をかける慣習は一般的ではありませんでした。IIJのキャッシュDNSサーバも例外ではなく、アクセス制限がないオープンリゾルバとなっていました。

盛んにDDoS攻撃が行われるようになったこの時期、悪意ある攻撃者はこれらオープンリゾルバに目を付け、踏み台として利用するようになりました。アクセス制限をすれば踏み台として利用できなくなるのは分かっているのですが、そうすると不正目的ではなく本来の用途でIIJのサーバを利用していただいているユーザも使えなくなってしまいます。長い間このジレンマに苦しんできましたが、ついに2013年12月、IIJのキャッシュDNSサーバに対してIIJ網外からのアクセスができなくなるよう設定を変更しました。

オープンリゾルバ対策はこれで終わりではありません。IIJが提供しているキャッシュDNSサーバの対策は完了しましたが、ユーザが自分で設置していたキャッシュDNSサーバやルータのDNS機能がオープンリゾルバになっていて攻撃の踏み台にされていたケースも多数あり、その該当ユーザに連絡して対処を要請するという活動が続きました。IIJだけでなく世界的に地道な対策が進んだことで、DNSを踏み台にしたDDoS攻撃は次第に下火になっていきました。

4.5 2020年代：更なる発展

■ 暗号化DNS

DNSは公開情報です。そのため、当初は盗聴されないこと（機密性）よりも改ざんされないこと（完全性）が重視されていました。2010年から始まったDNSSECも完全性を保証するための仕組みです。しかし、米NSAが大量の個人情報を収集していたことが暴露された2013年のいわゆるスノーデン事件[*5]をきっかけに、インターネット技術の標準化を推進する任意団体IETF（Internet Engineering Task Force）は「広範な監視は攻撃である」と宣言し[*6]、今後策定されるインターネット上のプロトコルは広域監視に耐え得る仕組みを具備するよう求めました。スノーデン事件ではDNSも監視対象になっていたこともあり、公開情報とはいえどんな情報を欲しがっているのかは個人のプライバシーであり、DNSにも機密性は必要であるという結論に達しました。

そこで標準化されたのがDNS over TLS（DoT）とDNS over HTTPS（DoH）です。従来のDNSはUDPまたはTCPの上に直接DNSメッセージを載せていましたが、DoT/DoHはそれぞれTLS、HTTPSのレイヤーの上にDNSメッセージを載せることで、第三者による盗聴を防ぎます。

2018年から2019年にかけて、Google Public DNSやCloudflare 1.1.1.1などのパブリックDNSサービスが相次いでDoT、DoHに対応し、またWebブラウザやOSなどクライアント側での対応も続きました。

現時点ではDoT/DoHで暗号化されるのはクライアントとキャッシュDNSサーバの間だけで、キャッシュDNSサーバと権威DNSサーバの間は暗号化の対象になっておらず、また

＊5　Wikipedia、「エドワード・スノーデン」（https://ja.wikipedia.org/wiki/%E3%82%A8%E3%83%89%E3%83%AF%E3%83%BC%E3%83%89%E3%83%BB%E3%82%B9%E3%83%8E%E3%83%BC%E3%83%87%E3%83%B3）。
＊6　RFC7258 Pervasive Monitoring is an Attack。

DoT/DoHサーバの設定を自動化する仕組みが普及していないなど、現状の利用にはまだ課題が残る部分はありますが、将来的には重要な位置を占めるだろうと考えられています。

そのため、IIJでは技術検証のための実験サービスとして2019年5月より「IIJ Public DNSサービス」を開始しました。DoT、DoHはUDPを使わずDNS ampの踏み台に利用される恐れがないため、オープンリゾルバとしてIIJ以外のユーザにも開放しています。その後接続サービスで利用されるキャッシュDNSサービスにも順次DoT/DoHの対応が拡大しています。

■ 新しい権威DNSサービス

インターネット黎明期から、権威DNSサーバは複数台構成にして可用性を向上させるべきものとされてきました。DNSを踏み台にしたDDoS攻撃は減りましたが、その後もDDoS攻撃自体はむしろ増えていることもあり、近年はゾーンを複数のDNS事業者に分散して設置することにより、特定の事業者が障害で応答できなくなった場合でも名前解決を継続できるようにするという考え方がとくに大規模なサイトでは一般的になりつつあります。

2000年にサービス開始されたDNSアウトソースサービス、DNSセカンダリサービスは機能強化を随時繰り返してきましたが、「DNSの問い合わせを受けてそれに答える」という基本機能に特化しており、複数の事業者で連携するには機能が足りませんでした。そのため、その他管理機能も強化するなどサービスを一新し、2019年11月、「IIJ DNSプラットフォームサービス」としてリリースしました。IIJをプライマリサーバにして他事業者をセカンダリにしたり、ユーザのプライマリサーバから転送されたゾーンにIIJサーバでDNSSEC署名するなど、従来のサービスではできなかった構成を自由にデザインできます。サイトフェイルオーバーオプションの後継サービスである「IIJ DNSトラフィックマネージメントサービス」も2022年3月に開始されました。

4.6 まとめ

IIJの30年の歴史をDNSの観点から駆け足で振り返りました。紹介しきれなかったエピソードなども多くありますが、誌面の都合上割愛せざるをえませんでした。

DNSは古くから存在するプロトコルですが、古いままではなく、常に進化を続けています。そして、インターネットの根幹を支える土台の1つであるということに関しては昔も今も変わりありません。IIJは今後も先進的な機能を積極的に取り入れつつ、堅牢かつ柔軟なDNSサービスを提供していく予定です。

執筆者:
山口 崇徳（やまぐち たかのり）
IIJ ネットワーク本部 アプリケーションサービス部。DNSサービスの開発などに従事。

June 2023 Vol.59

IIJ
Internet Initiative Japan

株式会社インターネットイニシアティブ(IIJ)について

IIJは、1992年、インターネットの研究開発活動に関わっていた技術者が中心となり、日本でインターネットを本格的に普及させようという構想を持って設立されました。
現在は、国内最大級のインターネットバックボーンを運用し、インターネットの基盤を担うと共に、官公庁や金融機関をはじめとしたハイエンドのビジネスユーザに、インターネット接続やシステムインテグレーション、アウトソーシングサービスなど、高品質なシステム環境をトータルに提供しています。
また、サービス開発やインターネットバックボーンの運用を通して蓄積した知見を積極的に発信し、社会基盤としてのインターネットの発展に尽力しています。

本冊子の情報は2023年6月時点のものです。

IIJ-MKTG019-0059

株式会社インターネットイニシアティブ
〒102−0071 東京都千代田区富士見2-10-2　飯田橋グラン・ブルーム
E-mail：info@iij.ad.jp　URL：https://www.iij.ad.jp