情報セキュリティ技術動向調査

タスクグループ報告書

(2009 年下期)

2010 年 3 月

独立行政法人 情報処理推進機構
セキュリティセンター

目次

序 2009 年下期の技術動向 - 今日のセキュリティエンジニアリングの話題

宮川 寧夫

背景

「情報セキュリティ技術動向調査 TG（タスクグループ）」[1]は、その第 4 回目の会合を 2009 年 12 月 25 日に開催した。情報セキュリティのエンジニアリングの分野において注目すべき動向を発表しあい、発表内容に基づいて討議した。

本報告書は、読者として IT 技術者を想定している。技術者によるエンジニアリング活動の中で、注目すべき動向もしくは話題を各委員独自の視点から紹介・解説していただいて本書を取りまとめた。

概況

今期、IETF の会合が広島において開催された。そのためか、今回はインターネットのプロトコルに関連する報告が約半分を占める。そこで、まず、広島会合におけるセキュリティ関連の話題を紹介するとともに、その場においても話題になった TLS renegotiation（再ネゴシエーション）における脆弱性と、その対策 RFC（Request For Comment）について解説する。次に、今後、IPv6 の導入とともに普及が期待されている Mobile IPv6 における位置情報プライバシの話題を採りあげた。さらに、IPv6 の導入・配備によって生じる各種のセキュリティ問題を整理することを試みる。今回はインターネット上における新たな脅威 Gumbler について報告すると共に対策技術の紹介を試みる。

また、近年、米国連邦政府の調達政策に関連する技術には示唆に富むものがある。今期、注目されたのは、オバマ政権の発足以降、民間の事業者が提供するサービスを連邦政府機関においても活用する動きが加速していることに関連した‘ Open Identity Solutions for Open Government ’という施策である。また、連邦政府職員の身分証明書カードに関する PIV 規格に基づく調達におけるテスト仕様書が改訂されつつある。

各国の政府調達における関心事でもある Common Criteria に関して、2009 年後半には OS（オペレーティングシステム）の評価に関連する Protection Profile が、軒並み‘ Sunset ’となった。‘ Sunset ’の日時以降に、当該 Protection Profile を用いて製品のセキュリティ評価を行っても承認されない。この背景を説明する。

セキュリティの観点からのプログラミングの話題として、今回は URI エスケープの話題を紹介する。

---

[1] http://www.ipa.go.jp/security/outline/committee/isec_tech1.html

1. 第 76 回 IETF 参加報告

木村 泰司

1. 7 年ぶりの日本開催

2009 年 11 月、広島市の ANA クラウンプラザホテルで第 76 回 IETF ミーティングが開催された。日本で開催されるのは 2 回目で、前回の 2002 年 7 月横浜開催以来、7 年ぶりとなる。参加登録者数はプレナリーでの発表時点（11 月 10 日）で 1,106 名、そのうちの日本からの参加者の割合は 34%で、米国の 27%を抜いて 1 位であった。参加人数の他にも、IPv6 の研究開発や標準化活動で知られる萩野純一郎氏にちなんで設立された Itojun Service Award の初の授賞式が行われたことや、インターネットへの接続が広帯域でかつ大変安定しており好評であったこと、そして参加者メーリングリストでの決め細やかな対応など、日本における開催という意味で、印象深い IETF ミーティングであった。

2. セキュリティエリア WG の動向

2.1 PKIX WG（Public-Key Infrastructure (X.509)）

PKIX WG は、インターネットにおける X.509 ベースの PKI について策定を行うことを目的として、1995 年に設置された WG である。近年の PKIX WG では、PKIX WG で扱っているプロトコルの暗号アルゴリズムの代替可能性（Algorithm Agility）の確保やトラストアンカーの証明書管理（Trust Anchor Management）の仕組みの確立といった課題がある。しかし WG においてはアクティブなのは、古参のメンバーが多く、チェアの Stephen Santesson 氏を含め、ベテランの参加者が I-D の作成に取り組んでいるという状況がある。

前回の第 76 回 IETF 以降から広島の PKIX WG ミーティング（2009 年 11 月）までの間に、以下のドキュメントが RFC となった。

○RFC となったドキュメント

- Elliptic Curve Cryptography Subject Public Key Information (RFC 5480) [1]
  電子証明書に入る Subject の公開鍵アルゴリズムとして、楕円曲線暗号を使えるようにする RFC。ECDSA（Elliptic Curve Digital Signature Algorithm）、ECDH（Elliptic Curve Diffie-Hellman）、ECMQV（Elliptic Curve Menezes-Qu-Vanstone）の 3 種類について定義されている。
- Other Certificates Extension (RFC 5697) [2]
  同一のエンドエンティティに対する複数の電子証明書を、Relying Party が識別

できるようにするため、関連する電子証明書を結びつける識別子を格納できる拡張フィールドを定義したRFC。

この他に、実装状況の報告によりRFC 5280をPS（Proposed Standard）からDS（Draft Standard）にするための”RFC 5280 Implementation Report”[3]がNIST（国立標準技術研究所）のTim Polk氏により報告された（検証作業はDavid Cooper氏）。NISTにおけるテストスイートであるPKITS(Public Key Interoperability Test Suites)[4]を用いて、PKIX WG MLで収集されたS/MIMEのメールを検証するなどの作業が行われた。今後、RFC 5280の大きな変更は行わず、Errataに基づく修正のみでDS化の提案が行われる。

2.2 SIDR WG（Secure Inter-Domain Routing WG）

SIDR WGは、RPSEC WGでまとめられたSecurity Requirementを踏まえ、インタードメイン・ルーティングにおけるセキュリティアーキテクチャの検討を行うWGである。このWGではRFCになったドキュメントはまだないが、第76回IETFの後、WG Last Callがかけられたドキュメントが現れてきた。

○WG Last Callがかけられたドキュメント

- An Infrastructure to Support Secure Internet Routing [5]
  セキュア・ルーティングを実現するためのアドレス資源の割り振り構造に従ったPKIの定義を行っているドキュメントである。後半にRPKI（Resource PKI）という用語が出てくる。
- Certificate Policy (CP) for the Resource PKI (RPKI) [6]
  RPKIのCP（証明書ポリシー）を定義したドキュメントである。

後半にWGドキュメントになっていない、2つの話題について議論された。IPアドレスの割り振り／割り当てとROA発行の関係について様々なケースを列挙したTerry Manderson氏のドキュメント[7]とTrust Anchorの管理手法に関するStephen Kent氏の提案である。Stephen Kent氏の提案は、RPKIの実装において、ローカルにTrust Anchorとなる電子証明書を用意し、プライベートアドレス等も扱えるようにするものである。

SIDR WGは、PKIX WGと同様にベテランの少数のメンバーが提案と議論を行っている状況であり、今後、より実現性の高めるために、ルーティングの技術者も議論に参加することが望まれている。

3. テクニカルプレナリー

テクニカルプレナリーは IRTF の報告や、IETF ミーティング参加者全体での技術的なトピックについて議論が行われる全体会議である。今回は、国際化ドメイン名に関する議論が行われた。台湾ではドメイン名の 35.2%が、韓国ではドメイン名の 13.7%が IDN として登録されているようで、今後コード体系が多いことによる問題、例えば Web のセキュリティやあいまいさを避けるために、複数のコードを混在させるのではなく、UTF-8 に限定するなどの対策が必要であるというプレゼンテーションがあった。この他に、.ARPA ゾーンにおける DNSSEC 導入のスケジュールが IAB のチェアから発表された。この発表によると 2009 年第 4 四半期に設定を開始し、2010 年第 2 四半期に本番投入できるように計画を進めているとのことであった。

4. IEPG にみる 2009 年のインターネット運用に関するセキュリティ動向

本稿は IETF ミーティングの参加報告であるが、2009 年度のインターネット運用のセキュリティに関わる出来事を振り返る意味で IEPG（Internet Engineering and Planning Group）ミーティングに触れておきたい。IEPG ミーティングは、IETF ミーティングの初日、IETF の受付が開始する前に行われるミーティングである。

IEPG ミーティングは、開催日程からして IETF ミーティング参加者を対象としたものであるが、ミーティングの趣旨はプロトコルの策定とは離れている。インターネット経路制御の実状や、IP アドレスと AS 番号の管理業務を国際的に行っている RIR（Regional Internet Registry）の活動状況など、国際的なインターネットの運用に関する話題が多い。今回の IEPG ミーティングのアジェンダの中でインターネットの運用上のセキュリティに関連するものを以下にまとめる。2009 年のインターネット運用上のセキュリティの話題を少しずつ取り出して集めたようなアジェンダであった。

- DNSSEC at the root, Joe Abley. ICANN
  ルートゾーンを提供するルート DNS サーバにおいて、DNSSEC を導入する計画の発表である。ICANN、米国商務省 電気通信情報局（NTIA）、VeriSign によるもので、2009 年 12 月 1 日に署名が行われ、2010 年 1 月から 7 月にかけて”Incremental roll out”（署名レコードの影響を考慮した段階的な提供）が行われる。2010 年 7 月に完了する予定である。
- RIPE Labs, Mirjam Kuehne, Robert Kisteleki. RIPE NCC
  ヨーロッパ地域の RIR である RIPE NCC で新たに設立された RIPE Labs と呼ばれる研究グループの紹介である。RIPE Labs は、IP アドレスに関する登録情報とトラフィックデータを元にした、様々な分析ツールを Web で開発・公開していく活動グループである。

2008 年にあった YouTube の経路ハイジャック事件以降、インターネット上のトラフィック状況などを世界地図にマッピングし、閲覧できるようにするプログラムの開発が進んできている。その活動のひとつとして捉えることができる。

- Stateless and desperate, Geoff Huston. APNIC

  DNS サーバにおける TCP の通信負荷を下げられるかどうか、という観点でステートレスの TCP に関する考案である。結局、考案したステートレス TCP は大きな効果が得られないことがわかったが、DNSSEC のように TCP が使われることが想定される DNS サーバにおいて TCP に工夫を凝らす着眼点として面白い。

- FIB aggregation, Lixia Zhang. UCLA

  インターネットに接続する BGP ルータにおいて、経路表の増大が問題となっており、この発表は経路表を扱うデータ構造である FIB（Forward Information Base）を集約（アグリゲート）し、メモリ消費量を減らすことについての考察である。一定の効果は見られるが、アグリゲートの処理をどのタイミングで行うか、といった課題がある。

- OpenDNSSEC, Stephen Morris. Nominet UK

  様々なネームサーバ・ソフトウェアと組み合わせ、ネームサーバにおいて DNSSEC を提供できるようにするオープン・ソフトウェアの紹介である。

- Root scaling study report, Jaap Akkerhuis. NLNetLabs

  DNS のルートゾーンにおける DNSSEC の導入に関する調査活動の中間報告である。国際化ドメイン名の導入や DNSSEC の導入により、転送するデータが格段に増加するルートゾーンを提供する DNS サーバの要件等を調べることを目的としている。

以上

参考文献

[1] “Elliptic Curve Cryptography Subject Public Key Information” (RFC 5480)
http://www.ietf.org/rfc/rfc5480.txt

[2] “Other Certificates Extension” (RFC 5697)
http://www.ietf.org/rfc/rfc5697.txt

[3] “Implementation Report for the Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile RFC 5280”
http://tools.ietf.org/html/draft-cooper-pkix-rfc5280-impl-report-00

[4] “Computer Security Division Computer Security Resource Center – PKI Testing”
http://csrc.nist.gov/groups/ST/crypto_apps_infra/pki/pkitesting.html

[5] “An Infrastructure to Support Secure Internet Routing”, draft-ietf-sidr-arch-09.txt
http://tools.ietf.org/html/draft-ietf-sidr-arch-09

[6] “Certificate Policy (CP) for the Resource PKI (RPKI)”, draft-ietf-sidr-cp-08.txt
http://tools.ietf.org/html/draft-ietf-sidr-cp-08

[7] “Use Cases and interpretation of RPKI objects for issuers and relying parties”, draft-manderson-sidr-usecases-01
http://tools.ietf.org/id/draft-manderson-sidr-usecases-01.txt

2. SSL/TLS の renegotiation（再ネゴシエーション）における脆弱性

木村 泰司

1. 概要

2009 年 11 月、SSL（Secure Sockets Layer)および TLS（Transport Layer Security）の脆弱性[1][2][3]が PhoneFactor 社の Marsh Ray 氏によって公表された[4]。脆弱性のある TLSv1.2 と SSLv3.0 は、Web ページのセキュアな閲覧の為に使われている最新のプロトコルで影響範囲が広い。

発見された脆弱性は、SSL/TLS のサーバとクライアントの間で、暗号アルゴリズムや暗号鍵の変更等のために使われる renegotiation（以下、再ネゴシエーションと呼ぶ）の手順(以下、ハンドシェイクと呼ぶ)にある。攻撃が成功すると、クライアント側から送信する通信データの中に任意の文字列を挿入することができ、Man-in-the-middle attack(以下、中間者攻撃)が可能になるというものである。

この脆弱性の発見を受けて、IETF の TLS WG では、対策を提案した RFC 5746[5] をわずか 3 ヵ月程でまとめ、2010 年 2 月に公表した。また SSL/TLS のライブラリで知られる OpenSSL は、脆弱性が発見されたのちに、急遽、再ネゴシエーション機能を無効化したバージョンがリリースされ[6]、その後 RFC5746 で提案された対策を実装した 0.9.8m-beta1 がリリースされた。

本稿では、インターネットにおける通信の安全性に大きく影響することを踏まえて、2009 年度後半の動向として、SSL/TLS の再ネゴシエーションにおける脆弱性について述べる。

2. 再ネゴシエーション・ハンドシェイクとその脆弱性

SSL と TLS は TCP の上位に位置するプロトコルである。TLS のセッション（以下、セッションと呼ぶ）は、TCP のコネクションが確立したのち、その通信路を利用して鍵確立などが行われる。TLS のネゴシエーション・ハンドシェイク（以下、ネゴシエーション）には、クライアントとサーバがセッションを新規に確立するための初期ネゴシエーションと、一度確立したセッションのパラメーターを変更するための再ネゴシエーションの 2 種類がある。

初期ネゴシエーションでは、クライアントとサーバが、使用できる暗号アルゴリズムの情報やそのパラメーターを伝達したり、通信データの暗号化に使われる共有鍵のためのランダム値を伝達したりする。このハンドシェイクが成功するとセッションが確立し、以降のクライアント・サーバ間の通信はネゴシエーションされた内容に基づいて暗号化される。

TLS の再ネゴシエーションは、TLSv1.0 および SSLv3.0 で新たに提案されたハンドシェイクである。一度確立したセッションに置き換わる新たなセッションを確立するために、クライアントとサーバの間で行われる。再ネゴシエーションは、既に確立しているセッションを使って、サーバからクライアントへの HelloRequest と呼ばれる要請のメッセージか、クライアントからの初期ネゴシエーションと同様の ClientHello のメッセージが送られることによって開始する。

再ネゴシエーションは、暗号化されているセッションを使って行われるため、盗聴してもその内容を知ることは基本的に難しい。また実際のハンドシェイクを偽造することも難しい。しかし Marsh Ray 氏によると、再ネゴシエーションの前後のセッションは、暗号化の状態が連続していても、認証の状態は不連続であり、後のセッションのトラフィックが予測可能である場合、攻撃者が予め用意しておいたセッションと入れ替えられると指摘されている[7]。2009 年 11 月、IETF TLS WG で同じ脆弱性を発見した Martin Rex 氏の Internet-Draft[8] では、この脆弱性を利用した攻撃パターンとして、以下の 3 つが挙げられている。

○再ネゴシエーションの脆弱性を利用した攻撃パターン

- クライアントの初期ネゴシエーションの入れ替え
  クライアントの初期ネゴシエーションが終わった後のセッションを、サーバの別の再ネゴシエーション後のセッションに入れ替える。

- サーバの初期ネゴシエーションの入れ替え
  クライアントの別の再ネゴシエーション後のセッションを、サーバの初期ネゴシエーションが終わった後のセッションに入れ替える。

- 別個のセッションから単一セッションの作成
  クライアントと攻撃者、攻撃者とサーバの間のセッションを、クライアントとサーバの間の単一のセッションにつなげる。攻撃者はクライアントとサーバの間の通信を完全に中継できる。

いずれのパターンでも、クライアントとサーバはネゴシエーションが成功しているように見えるため、基本的に攻撃されていることを検知することは難しい。

3. 脆弱性の影響

前回の第 76 回 IETF における TLS WG での発表資料[9] によると、この脆弱性を利用すると、以下のようなことが可能になると指摘されている。SSL/TLS 自体の脆弱性であるため、この他にも SSL/TLS を使った IMAP や LDAP 等の様々なプロトコルに影響する。

- クライアントに成りすましたデータ挿入
  クライアント認証が行われた TLS の通信において、認証手続きが行われていないセッションでのデータ挿入が可能である。

- クライアントの送信データの取得
  クライアント証明書なしに、TLS を通じてクライアントがサーバに向けて送信したデータを取得できる。

- 意図しない認証状態への変更
  本来はクライアント認証を必要としないにも関わらず、クライアント認証をサーバが要求するように、再ネゴシエーションを通じて本来と異なる認証状態にさせられてしまう可能性がある。

中間者攻撃の実現には、クライアントやサーバと SSL/TLS のセッションを確立できるような通信環境が必要であるが、本来 SSL/TLS はそのような攻撃が行われる環境においてもセキュアな通信を実現する目的で設計されたプロトコルである。インターネットを含む様々なネットワークでは、その想定の下で SSL/TLS が使われているという意味で、重大な脆弱性が発見されたといえる。

4. 対策

SSL/TLS の再ネゴシエーションの脆弱性に対して、IETF TLS WG では再ネゴシエーションの手続きを修正する根本的な対策が提案された。この脆弱性の対策は、RFC 5746[2]を反映した実装を利用することである。再ネゴシエーションが必要ない場合には、これを無効にすることも考えられる。なお、SSL/TLS のライブラリである OpenSSL では、RFC 5746 を反映したバージョンがリリースされている。

[2] このドキュメントは、広島で開催された第 76 回 IETF 期間中の TLS WG で発表されたが、同時に異例の速さで RFC にするためのスケジュールも発表された 11 月 30 日に WG Last Call を終え、12 月 22 日に IETF Last Call 完了、2 月 14 日に RFC Editor の作業を終えて RFC として公開するというものである。

この修正では、再ネゴシエーションに新たにふたつの変更が加えられた。ひとつはコネクション状態を示すフラグ等の追加である。TLSv1.2[10]の 6.1 節で定義されているコネクション状態において、セキュアにセッションを引き継ぐことを示す secure_renegotiation と呼ばれるフラグと、client_verify_data、server_verify_data と呼ばれる、直前のセッションで交換されたメッセージを含める値である。もうひと一つは、renegotiation_info と呼ばれる TLS 拡張である。TLS 拡張は、ネゴシエーションの際の ClientHello と ServerHello の末尾に付けられる値である。renegotiation_info には前述のコネクション状態を示す client_verify_data および server_verify_data が含まれており、再ネゴシエーションが行われる前後のセッションの連続性が確認できるようになっている。

5. まとめ

SSL/TLS は https をはじめ、SSL-VPN など、インターネットにおけるセキュアな通信を実現する方式として、最も広く使われているプロトコルのひとつであるといえる。その脆弱性の影響は大きいが、IETF の TLS WG ではいち早く根本的な解決方法を提案し、それは SSL/TLS のライブラリである OpenSSL にも反映された。

現代の、数多くのプログラムの脆弱性が発見される中で、特にプロトコルの脆弱性に対して必ずしも十分な速さで対策がとれるとは限らない。今後も、善意の技術者が迅速に解決に向けた活動を実施できることが重要であると考えられる。

以上

参考文献

[1] CVE-2009-3555,
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3555

[2] “Vulnerability Note VU#120541 SSL and TLS protocols renegotiation vulnerability”,
http://www.kb.cert.org/vuls/id/120541

[3] “JVNVU#120541:SSL および TLS プロトコルに脆弱性”,
http://jvn.jp/cert/JVNVU120541/

[4] “Authentication Gap in TLS Renegotiation” (Marsh Ray 氏のブログ),
http://extendedsubset.com/?p=8

[5] “Transport Layer Security (TLS) Renegotiation Indication Extension”,
http://www.ietf.org/rfc/rfc5746.txt

[6] “OpenSSL version 0.9.8l released” (0.9.8l の annouce.txt),
http://cvs.openssl.org/fileview?f=openssl-web/news/announce.txt&v=1.52
[7] “Renegotiating TLS”,
http://extendedsubset.com/Renegotiating_TLS.pdf
[8] “Transport Layer Security (TLS) Secure Renegotiation”
http://tools.ietf.org/id/draft-mrex-tls-secure-renegotiation-04.txt
[9] “TLS Renegotiation Vulnerability”
http://tools.ietf.org/agenda/76/slides/tls-7.pdf
[10] RFC 5246, “The Transport Layer Security (TLS) Protocol Version 1.2”,
http://www.ietf.org/rfc/rfc5246.txt

3. IP による移動通信における位置情報プライバシ

太田 耕平、キニ・ グレン・マンスフィールド

1. はじめに

移動通信の普及につれて位置情報プライバシへの関心が高まっている。近年では位置情報を積極的に用いたアプリケーションが新たな利便性を提供している一方で、IP アドレスと緯度経度の情報を結びつけるサービスも容易に利用可能であることから IP アドレスの漏洩が物理的な位置の漏洩にもつながる可能性がある。さらにインターネットでは通信トラフィック情報を収集、分析することによって利用者を”プロファイル”することが可能であることからその対策も課題となっている。

今後 IPv6 の導入とともに普及が期待されている Mobile IPv6 [1]では、これまで移動環境であっても実際にはポイントごとにしか利用できず断続的であったインターネット接続がシームレスなものとなる。そのため、プロファイルされる要素の中に「移動していること」およびその移動元、移動先、といった情報が知らず知らずのうちに漏洩する危険がある。さらには端末で利用されているアプリケーションの情報をあわせて分析することで、行動や生活様式、さらには個人の特定も可能となる危険がある。

また、これらの問題で警戒すべき対象として、第三者による傍受のみならず通信相手によるプロファイリングも含まれることが重要である。例えば、移動端末から利用する様々なサービス事業者がそれらの情報を収集、分析、利用する可能性がある。

本稿では、IP 通信での位置情報の利用と保護の問題のうち、Mobile IPv6 によって直面する IP アドレスとその変動情報の漏洩に起因する位置情報プライバシ問題と、その対策の現状について述べる。

2. MobileIPv6 における IP アドレスの位置情報プライバシ問題

MobileIPv6 における位置情報プライバシは、[2]で基本的な問題点が挙げられている。MobileIPv6 では、移動端末がそのアドレスとして固定的なホームアドレス（HoA：Home address）と移動先毎に変化する気付アドレス（CoA：Care-of address）を持ち、移動端末とホームアドレスをホストするホームエージェント間をトンネリング接続することでシームレスなインターネット接続を実現している。そのため、特定の HoA に対する CoA の変化を観測できればその端末（多くの場合は特定の個人と強い関係がある）が移動していることがわかり、それらを追跡し、さらなるプロファイルが可能となる。

それゆえ、Mobile IPv6 で、IP アドレスの位置情報プライバシを保護することは、移動

端末の CoA と HoA の関係を保護することとなる。現在の Mobile IPv6 で、CoA と HoA の関係が漏洩するのは以下のような場合である。

(1) 通信相手への CoA の開示

Mobile IP では移動端末-通信相手間は、原則としてホームエージェントを経由するリバーストンネルと呼ばれる経路で通信される。しかしこの経路は冗長であることから、Mobile IPv6 ではホームエージェントを経由せず、通信相手に CoA を開示して直接通信する最適経路モードを標準で備えることになっている。つまりこのモードを使用しているときには通信相手は移動端末の CoA を知ることができ、保持している HoA と CoA の関連情報を参照することで、移動端末が実際に移動したかどうかを知ることができる。

(2) 第三者への HoA の漏洩

移動端末-ホームエージェント間のトンネルが暗号化されていないとき、経路上の第三者はその通信に含まれている HoA と CoA を盗聴することができる。また、移動端末-通信相手間で経路の最適化モードを使用しているときには、経路上の第三者は移動端末が通信相手に HoA と CoA の関連付けのための通信を盗聴可能であり、やはり移動端末が実際に移動したかどうかを知ることができる。

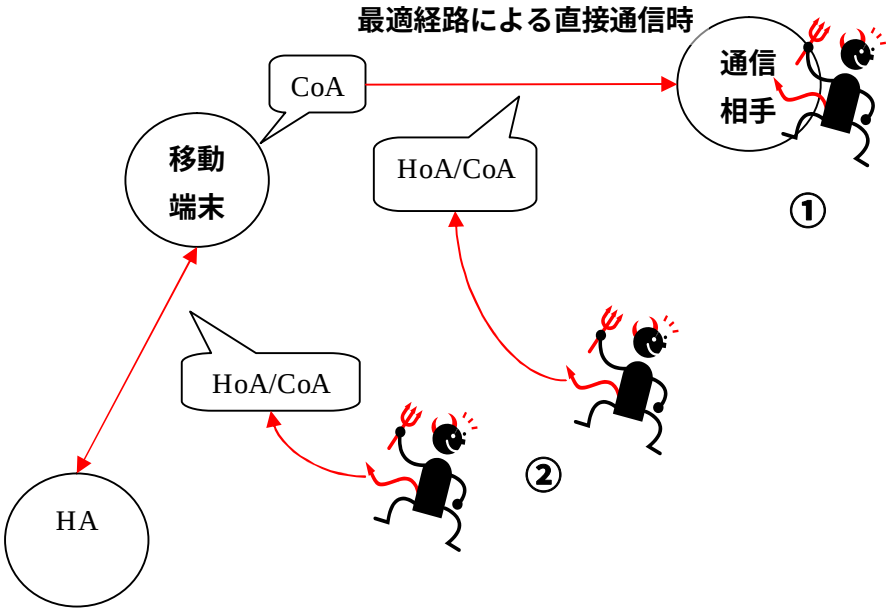

図 1：MobileIPv6 における IP アドレスの位置情報プライバシ問題

3. MobileIPv6 における位置情報プライバシの保護

移動端末は、移動端末-ホームエージェント間のトンネルを IPsec の ESP（(Encapsulating Security Payload）で暗号化することで、経路上の第三者への HoA の漏洩を防ぐことができる。また通信相手との通信に Mobile IPv6 で標準となった最適経路モードではなくホームエージェントを経由するリバーストンネルによる通信を利用すれば、暗号化するかどうかに関わらず通信相手は CoA を知ることはできないためアドレスから移動を知ることはできない。

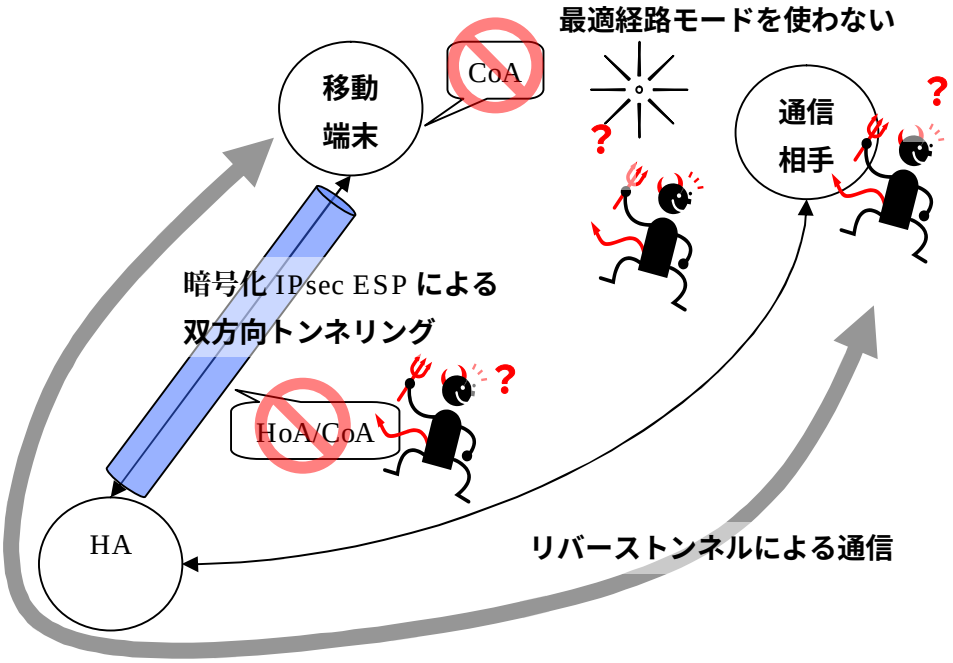

図 2： 既存の手法による位置情報プライバシの保護とその限界

しかし最適経路モードでは通信相手に CoA を開示する必要があるだけでなく、そのためのメッセージに HoA の情報が含まれ、それらは暗号化されていないため、通信相手と経路上の第三者の双方に移動端末の CoA と HoA の関係をさらすことになる。この問題を解決するには移動端末が CoA を開示しつつ HoA を保護することが必要となる。

4. RFC 5726 が提案する新しい保護

2010 年 2 月に IRTF（Internet Research Task Force）から発行された RFC 5726 [3] ではこれらの IP アドレスの開示の必要性と保護の両立の問題に対応し位置情報プライバシを

保護するふたつのシナリオが提案されている。

ひとつは通信相手が移動端末の本当の HoA を知ることができ、通信相手側から移動端末に対して通信を開始するケースであり、HoA を暗号化する手法である。もうひとつは移動端末側から通信を開始するケースであり、本当の HoA を通信相手にも開示しない手法である。

4.1 暗号化 HoA（リバーストンネルによる CoA の通知）

この手法では、最適経路モードで移動端末から通信相手に CoA を通知するメッセージ中の HoA の第三者への漏洩を防ぐために、メッセージ中の HoA を暗号化する。

問題はいかにして移動端末-通信相手間で暗号化のための鍵を共有するかであるが、通信相手から移動端末への通信を確立する際のリバーストンネル経由の Return Routability 手続きを利用し、その中で生成される鍵を利用することでこれを実現する。

これによって通信相手は通知された CoA に対する本当の HoA と暗号化された HoA を保持することになり、最適化された経路で交換されるパケットには暗号化された HoA を利用ことで経路上の第三者から本当の HoA を保護することができる。Return Routability 手続きで生成される鍵は通信相手によって異なるため暗号化された HoA もまた異なるものとなる。この手法は、既存の Return Routability 手続きを変更することなく実現できる。

4.2 擬似（Pseudo）HoA

通信相手から通信を開始する上記ケースでは通信相手に本当の HoA を開示する必要があるが、移動端末にとってより好ましいのは通信相手からも本当の HoA を保護することである。移動端末が CoA を通知する際に本当の HoA ではなく擬似的に生成した擬似 HoA を利用することでこれを実現する。

単なる暗号化とは異なり、擬似 HoA はホームエージェントまで実際に経路制御されるアドレスである必要があるため､ホームエージェントが存在するホームネットワークの Prefix とランダムなビット列を結合することで生成される。生成された擬似 HoA は Return Routability および通信相手への CoA の通知にも利用される｡これによって移動端末は本当の HoA を通信相手にも公開することなく通信が可能となる。また擬似 HoA は通信相手毎に異なるものを利用できるためプロファイリングを試みる盗聴者が HoA を固定的な ID として利用することを防ぐこともできる。

さらにこの手法では通信相手にとっては使われている HoA が本当の HoA か擬似 HoA か区別することができない。

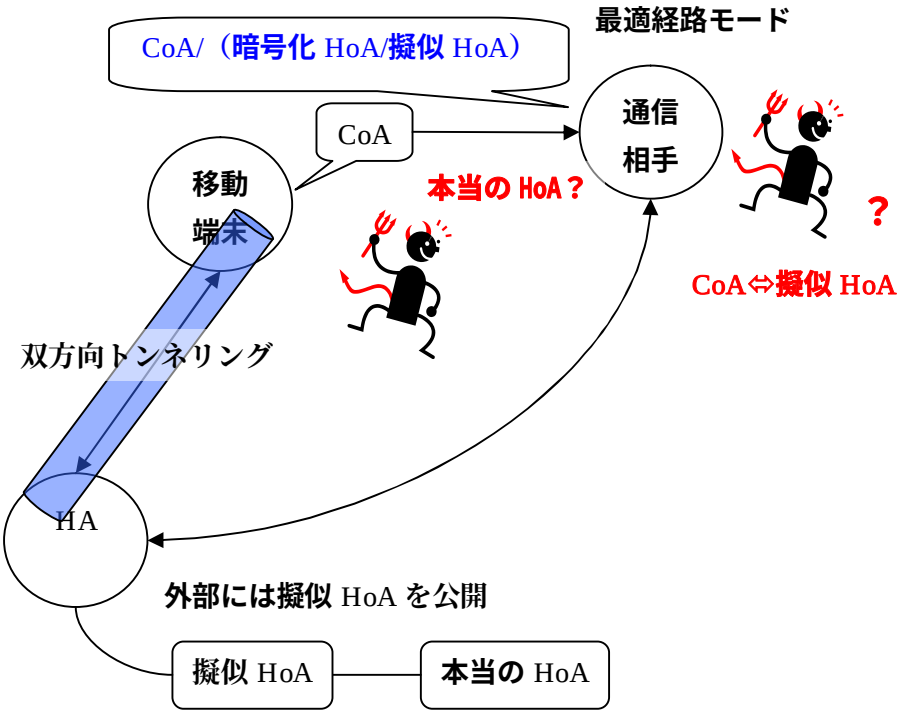

図 3： 暗号化 HoA と擬似 HoA による位置情報プライバシの保護

5. まとめ

Mobile IPv6 の位置情報プライバシを考える際には、いかにして HoA と CoA の関連を、通信相手、および盗聴者から保護するかが課題となる。これまでの Mobile IPv6 の枠組みの中では移動端末-ホームエージェント間のトンネル通信を暗号化し、CoA の開示につながる最適経路モードを利用しないことが推奨されることとなる。RFC 5726 では最適経路モードを利用しながら経路上の第三者と通信相手から真の HoA を保護する手法を考案し、CoA の開示が端末（利用者）の特定につながらない方法を提案している。一方で RFC 5726 はインターネットにおける位置情報プライバシという広範な問題の中で MobileIPv6 に固有な部分に関する解決案を提示している Experimental な RFC であり、インターネットの位置情報プライバシの利用と保護の問題は多くの課題があるホットな技術分野である。

以上

参考文献

[1] RFC 3775, “Mobility Support in IPv6”, June 2004.

[2] RFC 4882, “IP Address Location Privacy and Mobile IPv6: Problem Statement”, March 2007.

[3] RFC 5726, “Mobile Ipv6 Location Privacy Solutions”, February 2010.

4. IPv6 セキュリティ

馬場 達也

1. はじめに

2011 年から～2012 年に IPv4 グローバルアドレスが枯渇するとの予測があり、それまでに、インターネットを介してサービスを提供しているシステムは IPv6 に対応させる必要がある。しかし、IPv6 環境におけるセキュリティ上の問題についてはあまり整理がされていないのが現状である。本稿では IPv6 の導入によって生じるセキュリティ問題について整理する。

2. IPv6 への取り組みの必要性

世界的に IP アドレスを管理する IANA（Internet Assigned Numbers Authority）の IPv4 グローバルアドレスの在庫が枯渇するのは 2011 年 9 月頃、地域の IP アドレス管理団体である RIR（Regional Internet Registry）の在庫が枯渇するのは 2012 年 10 月頃と予測されている（2010 年 2 月 3 日現在）。IPv4 を延命させるための取り組みもされているが、根本的な解決法は IPv6 への対応であると言われている。

3. IPv4 と IPv6 の違い

IPv4 と IPv6 の主な違いは以下の 4 点である。

(1) アドレスの長さと表記方法が異なる

IP アドレスの長さは、IPv4 の場合は 32 ビットであったが、IPv6 では 4 倍の 128 ビットとなる。これにより、アドレス空間を増やし、アドレス枯渇問題を根本的に解決している。また、表記方法も変更され、IPv4 では、”163.135.10.10”のように「ドット区切り 10 進数表記」であったのが、IPv6 では、” 2001:db8:615:10::1”のように、「コロン区切り 16 進数表記」となっている。

(2) リンクローカルアドレスが追加された

IPv6 では、インターネット上で利用可能な「グローバルアドレス」や、従来のプライベートアドレスに該当する「ユニークローカルアドレス」に加えて、同一セグメント内での

み利用可能な「リンクローカルアドレス」が追加された。

(3) 拡張ヘッダの概念が追加された

ソースルーティングやフラグメントなど、通常あまり使用することのない IP の機能は、拡張ヘッダで対応するように変更された。拡張ヘッダには、ルーティングヘッダや、フラグメントヘッダ、IPsec で使用する AH ヘッダや ESP ヘッダなどが存在する。

(4) ステートレスアドレス自動設定が追加された

IPv6 では、DHCPv6 によるアドレスの割り当て（ステートフルアドレス自動設定）以外に、ルータが配下のホストにネットワーク部（上位 64 ビット）を通知し、ホスト側がホスト部（下位 64 ビット）を自動生成して IPv6 アドレスを構成する「ステートレスアドレス自動設定」が追加された。

4. IPv6 環境におけるセキュリティ面での誤解

IPv6 環境におけるセキュリティ面の誤解として一番多いのが、「グローバルアドレスを持つと、外部から攻撃されてしまうのは？」ということである。IPv4 の場合は、通常は PC にはプライベートアドレスを付与し、インターネットとの接続点で NAPT（Network Address Port Translation）によってグローバルアドレスに変換をしているため、外部からはアクセスすることができない。しかし、IPv6 の場合においても、NAPT は使わないものの、ルータのファイアウォール機能で適切に設定すれば問題ない。

2 番目として「IPv6 は攻撃が少ないから安全」という誤解もある。現在は、IPv6 ユーザが少ないため、IPv6 ネットワーク上で動作するワームなどは発見されていないが、IPv6 が普及するにつれて、IPv6 を使用した攻撃やワームは増加すると考えられる。このため、IPv6 環境においてもセキュリティ対策は万全にしておく必要がある。

そして、3 番目として、「IPv6 通信は IPsec によって暗号化されるので安全」という誤解もある。これは、IPv4 の仕様では IPsec の実装はオプションであったが、IPv6 の仕様では実装必須とされていることに起因する。しかし、実装必須とされているからといって、IPv6 通信に IPsec が使われるとは限らない。実際、多くの OS では、IPv4 でも IPsec が利用できるように実装されているが、エンド・ツー・エンドの通信に IPsec を使うことはあまりなのが現状である。IPv6 の IPsec も IPv4 と基本的に変わらないため、IPv6 であるからといって、IPsec が使われるということはない。ただし、IPv6 の場合は、組織内のネットワークにおいてもグローバルアドレスが使用されるため、IPv4 の場合に問題となることの多かった、「NAT 越え」の問題は発生しない。このため、IPsec を利用しやすい環境には近づいているとはいえる。しかし、IPsec の鍵管理の煩雑さを解消しないと、エンド・ツー・エン

ドの通信に IPsec を使うのは難しいだろう。

5. IPv6 環境におけるプライバシー問題

IPv6 で問題となるのが、IPv6 アドレスによって行動をトレースできてしまうという、プライバシーに関する問題である。基本的に、IPv6 アドレスの下位 64 ビットは、EUI-64 フォーマットに従って、48 ビットの MAC アドレスから生成される。つまり、同じ端末であれば、ネットワークを移動しても、IPv6 アドレスの下位 64 ビットは同じになるということである。このため、サーバ側でアクセスログを取っていれば、ログ中の IPv6 アドレスの下位 64 ビットを突き合わせることで、行動履歴が取れてしまう。この問題に対する解決策として、IPv6 アドレスの生成に MAC アドレスを使用せず、ランダムに生成するという以下の仕様が存在する。

- RFC 3041, “Privacy Extensions for Stateless Address Autoconfiguration in IPv6”
- RFC 4941, “Privacy Extensions for Stateless Address Autoconfiguration in IPv6”

Windows XP SP2 では RFC 3041、Windows Vista/Windows 7 では RFC 4941 を採用しているが、その他の OS では採用されていない。

ただし、Windows であってもリンクローカルアドレスには RFC 3041/4941 は適用されないという問題がある。リンクローカルアドレスの上位 64 ビットは「fe80::」に固定されているため、どの環境にいても、MAC アドレスが同じであれば、リンクローカルアドレスは同一となる。このため、セミナ会場の無線 LAN 環境などで、参加者が同じセグメントに接続している場合に、事前に相手のマシンの MAC アドレスが分かっていれば、リンクローカルアドレスを得ることができるため、IP レベルでアクセスできてしまうという問題がある。

6. IPv6 環境でのネットワークスキャン

ワームなどは、攻撃対象を見つけるために、IP アドレスを変えながら順次アクセスを行う、いわゆるネットワークスキャン（ポートスキャン）を行う。IPv6 はアドレス空間が広大なため、IPv4 環境よりネットワークスキャンが困難となっている。しかし、RFC 5157 “IPv6 Implications for Network Scanning”でも指摘されているように、MAC アドレスから IPv6 アドレスを生成する仕組みの場合はスキャンが容易になってしまうという問題がある。これは、MAC アドレスの上位 24 ビットはベンダ ID となっているため、広く流通して

いる NIC のベンダ ID を調べてスキャンすれば、IPv6 アドレスの 128 ビット中 40 ビット分を固定にしてスキャンすることができるからである。また、Windows では、RFC 4941 の仕組みにより、IPv6 アドレスの下位 64 ビットをランダムに生成しているが、MAC アドレスから生成したアドレスも付与されて応答ができるようになっているため、この手法を回避することができない。

7. IPv6 環境での不正アクセス

アプリケーションレベルや TCP レベルの攻撃など、レイヤ 4 以上の攻撃は IPv6 でも IPv4 と同じ手法で行うことが可能である。IPv6 でアクセス可能なサイトにおいて、IPv6 に対応していない IDS（Intrusion Detection System）や IPS（Intrusion Prevention System）を使用している場合は、プロトコルを IPv6 にして攻撃すれば、攻撃を検知されずに IPv4 と同様の攻撃を行うことが可能となってしまう。このため、サーバを IPv6 に対応する場合には、同時に IDS や IPS などのセキュリティ機器も IPv6 に対応しなければならない。

また、マルウェア（ウイルス、ワーム、スパイウェアなど）については、現在のところ IPv6 対応のものは発見されていない。しかし、マルウェアの駆除は、ファイルベースで行われるため、従来通りウイルス対策ソフトなどによって検知・駆除することが可能である。

セキュリティ製品としては、ファイアウォール、IDS/IPS、アンチウイルスゲートウェイ、URL フィルタリング、WAF（Web Application Firewall）などが存在するが、全体的に IPv6 への対応が遅れており、早急な対応が求められる。

8. ステートレスアドレス自動設定のセキュリティ問題

IPv6 で追加されたステートレスアドレス自動設定には、DHCP による方法と比較して、(1)①アドレス割り当て時に DHCP 認証（MAC アドレス認証）ができない、(2)②割り当てのログが残らない、(3)③RA（Router Advertisement）の機能を悪用して盗聴することができてしまう、という問題がある。

しかし、(1)①の問題については、そもそも、MAC アドレスは偽ることが可能であり、認証を行いたい場合には、IEEE802.1X 認証を利用すべきである。また、(2)②の問題については、確かに、割り当てのログは残らないが、DHCP 環境であっても、勝手にアドレスを設定してしまえば、DHCP によって割り当てられなくても通信できてしまうという問題は依然として残る。(3)③の問題については、次に詳しく述べることとする。

(3)③の問題は、攻撃者が不正に RA を送出することによって、デフォルトゲートウェイに成りすますことで、通信の内容を盗聴するというものである（図 1）。

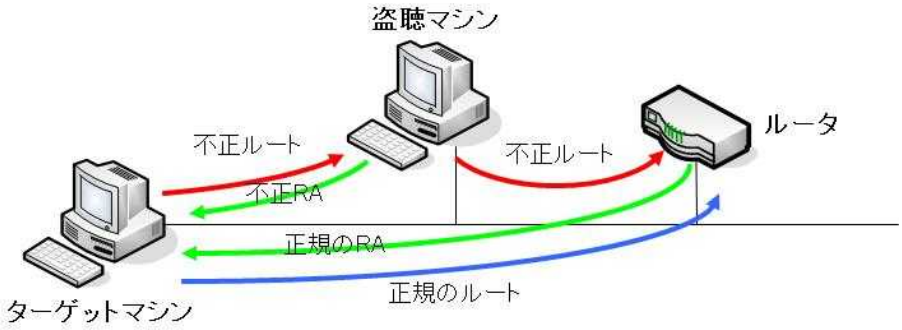

図 1： 不正 RA による盗聴

draft-ietf-v6ops-ra-guard-04.txt “IPv6 RA-Guard” に対策が列挙されており、SEND（RFC 3971）という RA の認証の仕組みを利用するか、レイヤ 2 スイッチで、ルータ接続ポート以外からの RA をフィルタリングするようにするという方法がある。しかし、SEND については、Cisco 社のルータでは実装されているものの、まだあまり普及していないという問題がある。また、レイヤ 2 スイッチで RA をフィルタリングする方法については、対応しているスイッチが少ないという問題がある。このため、当面の対策として、正規のルータからの RA の優先度（Router Preference）を High に設定しておき、デフォルトの優先度（Medium）の RA が送信されてきた場合は端末側が無視するようにしておくことが良い。しかし、攻撃者が RA の優先度を High にした場合には効果がないという問題がある。

9. トランスレータ利用時のセキュリティの考慮事項

サービス提供者側のシステムが IPv6 への対応ができない場合、トランスレータと呼ばれる機器によって、ユーザからの IPv6 アクセスを IPv4 に変換してシステム側に中継する場面が想定される（図 2）。この場合、ユーザの IPv6 アドレスがトランスレータの IPv4 アドレスに変換されるため、インシデントが発生した場合に発信元を追うためには、システム側のアクセスログとトランスレータでの変換ログを付き合わせる必要がある。これを実現するためには、システム側では、送信元 IPv4 アドレス（トランスレータの IPv4 アドレス）とともに、送信元ポート番号も取得する必要がある。ロードバランサの製品によっては、HTTP ヘッダに変換前の IPv6 アドレスを埋め込むことができる製品がある。システム側でこの HTTP ヘッダを見ることによって送信元 IPv6 アドレスが判明する。

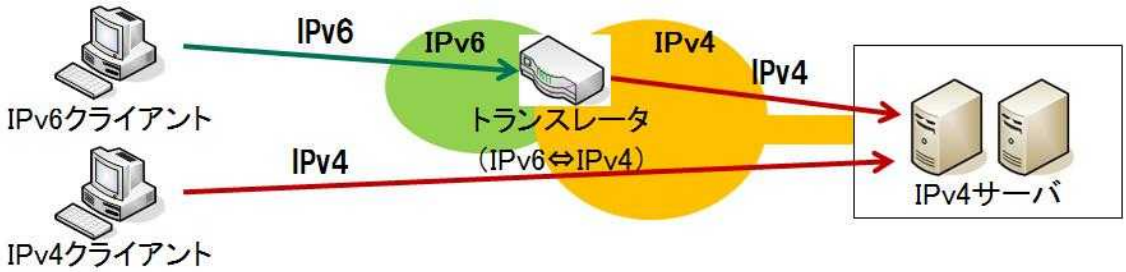

図 2： トランスレータによる IPv6/IPv4 変換

10. 拡張ヘッダに起因するセキュリティ問題

大量の Type 0 ルーティングヘッダ（ソースルーティング）を付加した IPv6 パケットを送信することにより、DoS 攻撃が可能となってしまうという「Type 0 ルーティングヘッダの問題」が存在したが、RFC 5095 の”Deprecation of Type 0 Routing Headers in IPv6”にて、Type 0 ルーティングヘッダは廃止になった。また、フラグメントを悪用した攻撃など、IPv4 にて問題となっていた攻撃は、IPv6 でも問題となっている。

11. まとめと今後の展開

IPv6 には、IPv4 のセキュリティ問題に加えて、IPv6 で追加された機能に起因するセキュリティ問題が追加されており、その問題を理解して IPv6 対応する必要がある。また、セキュリティ関連機器は IPv6 対応が遅れている場合が多いが、サイトが IPv6 対応した場合には IPv6 のセキュリティも万全にしておく必要があり、ベンダに対応を急がせる必要がある。

現在、IPv6 普及・高度化推進協議会や JNSA にて IPv6 セキュリティについて議論をしているが、議論を加速させ、成果を広く周知することが望まれる。

以上

参考文献

- RFC 4942, “IPv6 Transition/Coexistence Security Considerations”
- RFC 4943, “IPv6 Neighbor Discovery On-Link Assumption Considered Harmful”
- RFC 5095, “Deprecation of Type 0 Routing Headers in IPv6”
- RFC 5157, “IPv6 Implications for Network Scanning”
- draft-ietf-v6ops-ra-guard-04.txt “IPv6 RA-Guard”
- draft-ietf-v6ops-cpe-simple-security-08.txt “Recommended Simple Security Capabilities in Customer Premises Equipment for Providing Residential IPv6 Internet Service”

5. Web 媒介型攻撃 Gumblar の動向調査

井上 大介

1. 新たな脅威 Gumblar

2009年初頭から、日本国内の大手企業を含む様々なWebサイトの改ざんと、それらのWebサイトを閲覧したPCがマルウェアに感染するという事案が多発している。これらは、Gumblarと呼称されるWeb媒介型の攻撃手法によるものである。本稿では、このGumblarについて、その基本的な仕組みを概観するとともに、対策手法について述べる。

2. ドライブバイダウンロードと Web サイト改ざんの連鎖

ユーザがWebサイトを閲覧した際に、ユーザのWebブラウザもしくはWebブラウザが利用するプラグインやアプリケーションの脆弱性が悪用され、ユーザが気付かないうちにマルウェアのダウンロードと実行が行われる攻撃手法をドライブバイダウンロード（drive-by-download [1]）と呼ぶ。GumblarもユーザPCへのマルウェア感染の際に、このドライブバイダウンロードの手法を用いている。

Gumblar以前のドライブバイダウンロードは、攻撃の起点となるWebサイトを用意するための事前準備を必要としていた。例えば、攻撃者がSQLインジェクション等によって正規のWebサイトを攻撃し、マルウェアや不正なスクリプトをWebのコンテンツに埋め込むといったWebサイトの改ざんが、事前準備として行われることが多かった。その一方で、Webサイト側のSQLインジェクション対策が進み、SQL文中の特殊文字の適切なエスケープ処理やWAF（Web Application Firewall）の導入等が浸透するにつれ、攻撃者がWebサイト（特にサーバ側のセキュリティ対策を十分に行っている大手企業のWebサイト）を正攻法で攻略することは難しくなってきていた。

しかしながらGumblarは、互いに独立した攻撃手法と考えられていたWebサイトの改ざん（サーバへの攻撃）とドライブバイダウンロード（クライアントへの攻撃）を巧みに連鎖させることで、Webサイトの改ざんとマルウェアの感染拡大の巨大なスパイラルを作り出している。

3. Gumblar 攻撃の流れ

Gumblar攻撃の流れを図1に示すとともに、以下に概説する。

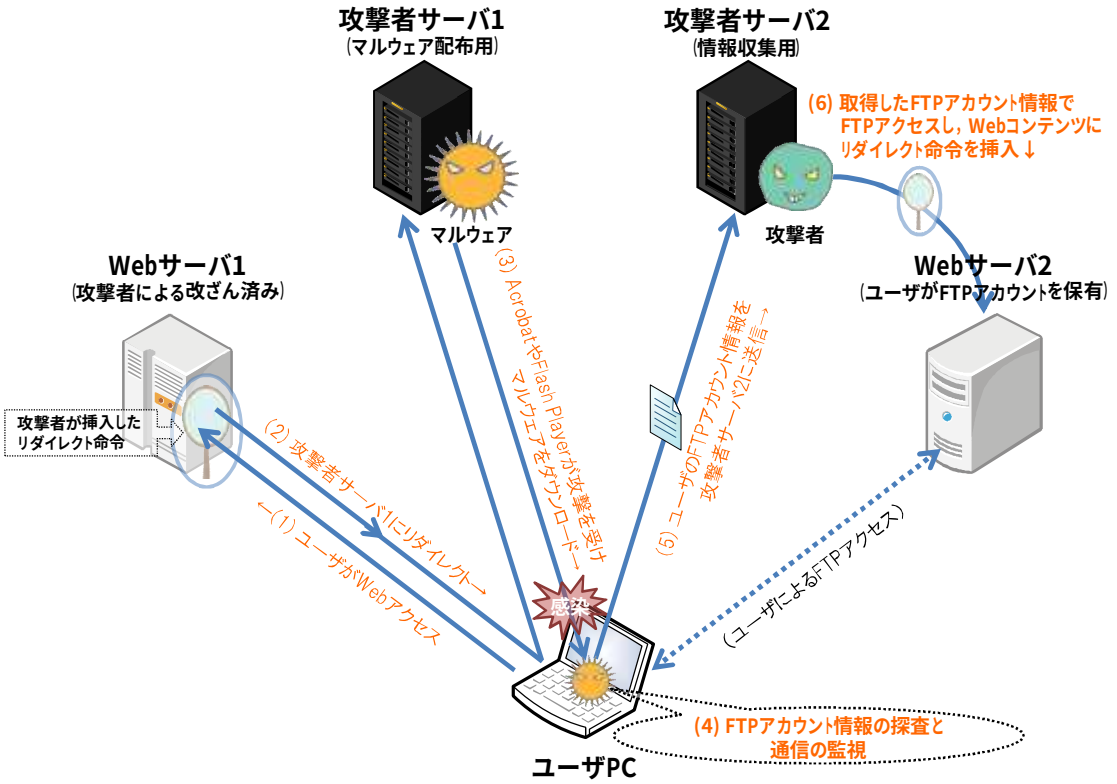

図1： Gumblar攻撃の流れ

(1) **ユーザPCがWebサーバ1（既に攻撃者によって改ざんを受けているサーバ）にアクセスし、Webコンテンツを閲覧する。**

(2) **Webサーバ1のWebコンテンツには、JavaScriptによる不正なリダイレクト命令が難読化された状態で挿入されており、ユーザPCは攻撃者の制御下にあるマルウェア配布用の攻撃者サーバ1にリダイレクト（誘導）される。**

(3) **ユーザPCが攻撃者サーバ1にアクセスすると、バージョンの古いAdobe Readerの脆弱性を攻撃するPDFファイルや、Flash Playerの脆弱性を攻撃するSWF（Small Web Format）ファイル等がダウンロードおよび実行され、ユーザPCの権限が奪取される。その後、複数のマルウェアがダウンロードされユーザPCに感染する。**

(4) **ユーザPCに感染したマルウェアは、ドライバとしてユーザPCの通信を監視するとともに、ユーザPCの中に保存されているFTPアカウント情報（ID、パスワード）を探査する[2]。**

(5) **マルウェアによる通信の監視中に、ユーザがWebサーバ2にFTPアクセスを行った場合（またはマルウェアがユーザPC内でFTPアカウント情報を発見した場合）、攻撃者の制御下にある情報収集用の攻撃者サーバ2に、ユーザのFTPアカウント情報が送信される。**

(6) 攻撃者は窃取したFTPアカウント情報を用いて、正当なユーザになりすましてWebサーバ2にFTPアクセスし、Webコンテンツにリダイレクト命令を挿入する。

このように、Gumblarとは特定のマルウェアの名称ではなく[3]、ドライブバイダウンロードによるマルウェア感染と、それに続くWebサイトの改ざんという一連の攻撃手法を指し示したものである。

4. Gumblarの巧妙さと継続的な進化

日本では複数の大手企業のWebサイトがGumblarによる改ざんの被害に遭っている。これは、大手企業のWebサイトをメンテナンスしている管理者（自社あるいは外注先の企業）のPCがマルウェアに感染し、Webサイト管理用のFTPアカウント情報が盗用されていることに起因している。攻撃者は堅牢なWebサーバを直接攻撃するのではなく、セキュリティ対策の甘いPCから攻略し、窃取した正規のアカウントでWebサーバにログインするという巧妙な手口を取っている。

攻撃者がWebコンテンツに挿入するリダイレクト命令は、多くの場合難読化されているため、Webサイトの管理者であっても改ざんに気付きにくい。しかも難読化前のURIを含む文字列は頻繁に更新されている[4]ため、難読化文字列のシグネチャ化やURIのブラックリスト化を難しくしている。

また、Gumblarの攻撃手法自体も進化を続けており、Gumblar.xなどの亜種と思われる手法や、8080.ru、8080.cnなど類似の手法も出現している[5]。これらの攻撃手法では、Adobe ReaderやFlash Player以外にも、Microsoft Data Access Components、Internet Explorer 7、JRE（Java Runtime Environment）、SnapShot Viewer、DirectShow等、様々なアプリケーションの脆弱性が利用されている[5]。また、Webコンテンツにリダイレクト命令を挿入する代わりに、Apacheの不正な設定ファイル（.htaccess）を特定のディレクトリに置くことで、検索エンジンからのアクセスや404エラー等の発生時に攻撃者サーバにリダイレクトする（Webサイトの管理者に気付かれにくい）手法も報告されている[6]。

ドライブバイダウンロードによってユーザPCへの感染に成功したマルウェアは、ユーザPCのローカルディスクの中から、FTPクライアント機能を有するアプリケーションが保存しているFTPアカウント情報を探索する。Gumblarが流行の兆しを見せ始めた2009年初頭には、ダウンロードされたマルウェアに、ローカルディスクからFTPアカウント情報を窃取する機能は含まれていなかった（FTP通信からの窃取のみ）[7]。しかし、ドライブバイダウンロードの原理上、攻撃者は配布するマルウェアを自由に変更することができ、このローカルディスクからのFTPアカウント情報の窃取をはじめ、Gumblarによって感染するマルウェアは今日までに様々な機能強化がなされてきている。なお、GumblarによるFTPア

カウント情報の窃取が確認されているアプリケーションは以下の通りである[8]。

ALFTP 5.2 beta1
BulletPloof FTP Client 2009.72.0.64
EmFTP 2.02.2
FFFTP 1.96d
FileZilla 3.3.1
FlashFXP 3.6
Frigate 3.36
FTP Commander 8
FTP Navigator 7.77
FTP Now 2.6.93
FTP Rush 1.1b
SmartFTP 4.0.1072.0
Total Commander 7.50a
UltraFXP 1.07
WinSCP 4.2.5

5. Gumblar 対策

ここまで述べたように、GumblarはユーザPCとWebサイトの両方に対して攻撃と改ざんを試みる。したがって、Gumblarによる攻撃を防ぐためには、ユーザ側（ユーザおよびユーザPCが属するネットワーク管理部門）とWebサイト側の両面からの対策が求められる。

5.1 ユーザの対策

ユーザが行い得るGumblar対策は、基本的には従来のユーザPCのセキュリティ対策と同様、以下の通りである。

(1) Windows OSの最新版へのアップデート
(2) アンチウイルスソフトの導入とシグネチャの更新
(3) 各種アプリケーションやプラグインのアップデート
   - Webブラウザ
   - Adobe Reader
   - Flash Player
   - JRE

- Microsoft Office製品　等

ここでは特に、OS更新とアンチウイルスソフト導入だけではなく、Webブラウザとそのプラグインや、各種アプリケーションのアップデートを行うことが重要である。なお、対策の詳しい実施方法については文献[5][9]等を参考のこと。

5.2 ユーザ PC のネットワーク管理部門の対策

ユーザPCが企業や学校等の組織に属している場合、その組織のネットワーク管理部門においては、以下の対策を行うことができる。

(4) マルウェア配布用サーバのIPアドレスによる検知/フィルタリング
(5) 情報収集用サーバへのHTTPリクエストによる検知/フィルタリング

対策(4)は、マルウェア配布用サーバへのリダイレクト（図1の(2)）を検知/フィルタリングするために用いられる[10]。ただし、2009年終盤に再流行したGumblarでは、マルウェア配布用サーバの台数が大幅に増加[11]したため、IPアドレスのブラックリスト化を難しくしている。対策(5)は、ユーザPCから情報収集用サーバにFTPアカウント情報を送信する際（図1の(5)）に用いられるHTTPリクエストが、RFC違反の特殊なものであることを利用した検知/フィルタリング手法である[11]。

5.3 Web サイト側の対策

Webサイトの管理者が行い得るGumblar対策として、以下が挙げられる。

(6) WebサーバへのFTPアクセスの制限
(7) Webコンテンツの改ざん検知
(8) Webコンテンツの異常検知

対策(6)は、WebサーバにFTPアクセスできるIPアドレスに制限を設け、既知のIPアドレス以外からのFTPアクセスを拒否することで、攻撃者による不特定のIPアドレスからのFTPアクセスを防ぐことができる。対策(7)は、Webサイト上のコンテンツを定期的にチェックし、Webサイトの管理者が意図しない不正な書き換え等の発生を検知する方法である。ここでは、Webコンテンツの改ざん検知技術[12][13]が利用できる。対策(8)は、Webサイト上のコンテンツを自動巡回し、不正なリダイレクト命令やマルウェア等の異常なコンテンツが含まれていた場合に、それらを検知する方法である。ここでは、クライアントハニーポットと呼ばれる、ユーザからのWebアクセスを模擬する能動的なハニーポット技術の研究開発と運用が進められている[14][15]。

6. まとめ

本稿では、Webを媒介とした新たな攻撃手法であるGumblar、およびその対策について概説した。Gumblarは依然進行中の事象であり、継続的な経過観測が必要である。

以上

参考文献

[1] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang and Nagendra Modadugu “The Ghost In The Browser: Analysis of Web-based Malware,” HotBots'07, April, 2007.

[2] JM Hipolito, “Stolen FTP Credentials Key to Gumblar Attack,” TrendLabs Malware Blog, Jun 2009.
http://blog.trendmicro.com/stolen-ftp-credentials-key-to-gumblar-attack/

[3] ノートン プロテクションブログ, “Gumblar (ガンブラー) ？Web 攻撃を解説,” Jan 2010.
http://communityjp.norton.com/t5/blogs/blogarticlepage/blog-id/npbj/article-id/45

[4] Mary Landesman, “Gumblar Morphs Again: Now Martuz.cn,” ScanSafe STAT Blog, May 2009.
http://blog.scansafe.com/journal/2009/5/18/gumblar-morphs-again-now-martuzcn.html

[5] So-net セキュリティ関連ニュース, “「ガンブラー」「サイト改ざん」めぐる基本のQ&A ～ 何が起きている？ 対策は？,” Jan 2010.
http://www.so-net.ne.jp/security/news/newstopics_201001.html

[6] ラック, “Gumblar（ガンブラー）ウイルスによる新たなホームページ改ざん被害を確認,” Mar 2010.
http://www.lac.co.jp/info/alert/alert20100303.html

[7] INTERNET Watch, “ボット、偽ソフト、ルートキット……「Gumblar」感染被害の実態,” Mar 2009.
http://internet.watch.impress.co.jp/docs/news/20100309_353620.html

[8] JPCERT/CC, “FTP アカウント情報を盗むマルウエアに関する注意喚起,” Feb 2009.
http://www.jpcert.or.jp/at/2010/at100005.txt

[9] Cyber Clean Center, “ホームページからの感染を防ぐために,”.
https://www.ccc.go.jp/detail/web/

[10] ラック, “Gumblar（ガンブラー）ウイルスの組織内感染拡大とホームページ改ざん

被害増加に伴う対策の確認," Dec 2009.
http://www.lac.co.jp/info/alert/alert20091225.html
[11] IIJ, "Internet Infrastructure Review," vol.006, pp. 11-12, Feb 2010.
http://www.iij.ad.jp/development/iir/pdf/iir_vol06.pdf
[12] Tripwire, http://www.tripwire.co.jp/solutions/guide2004/guide3.html
[13] WebS@T, http://www.kaizankenchi.jp/
[14] Honey Monkey,
http://research.microsoft.com/en-us/um/redmond/projects/strider/honeymonkey/
[15] gred, http://www.gred.jp/

6. Open Identity Solutions for Open Government

工藤 達雄

1. はじめに

本報告では 2009 年下半期のアイデンティティ管理技術に関する動向として、米国 Open Identity Solutions for Open Government の動きを概観する。

2. Open Identity Solutions for Open Government とは

米国ではとくにオバマ政権の発足以降、民間の事業者が提供するサービスを政府機関で活用する動きが加速しており、アイデンティティ管理の分野も例外ではない。

2009 年 9 月、FCIOC（Federal CIO Council：連邦政府 CIO 評議会）内の ISIMC（Information Security and Identity Management Committee：情報セキュリティおよびアイデンティティ管理委員会）の分科委員会である ICAMSC（Identity, Credential and Access Management (ICAM) Subcommittee）は、新たなイニシアチブとして「OpenIdentity Solutions for Open Government」を発表した[1] 。

本イニシアチブでは、市民に対する行政の透明性向上およびアクセスの容易化、そしてアプリケーションごとのクレデンシャル（いわゆるアカウント情報）発行の省力化を進めるために、民間の IdP (Identity Provider：たとえばポータル事業者や決済事業者など、市民に対して ID を発行・運用する組織)の発行したクレデンシャルを、連邦政府機関の市民向け Web サイト（以下、「行政 Web サイト」）へのアクセスに活用することを目指している。

民間の IdP は、RP（Relying Party）となる行政 Web サイトに対してアサーション（ここでは、行政 Web サイトにアクセスしようとする利用者に関する、認証結果や属性情報）を提供するにあたり、「TFP（Trust Framework Provider：オンライン・アイデンティティの信頼モデルを策定・認定する組織）」の信頼モデルに準拠しなくてはならない。この TFP も IdP と同様に民間の組織であり、ICAMSC が TFPAP（Trust Framework Provider Adoption Process）[2] に従って評価・認定する。

つまり本イニシアチブでは、連邦政府が直接民間の IdP を評価・認定するのではなく、連邦政府の認定を受けた民間の TFP（実際には、TFP の基準に合致する監査人）が IdP の評価・認定を行う。ICAM ではこのように民間のリソースを活用することで、コストの削減と、本モデルの普及の迅速化を狙っている。

TFPAP では、TFP 申請者の信頼モデルが行政 Web サイトに適合するかどうかのチェ

ックが、ふたつの観点から行われる。ひとつは Trust Criteria Assessment であり、NIST SP 800-63-1 に基づく技術的な評価項目との適合性と、プライバシ保護に関する適合性の確認からなる。もうひとつは Audit Criteria Assessment であり、IdP の監査を行う監査人に対して TFP 申請者が要求する、基準の適合性が確認される。

現時点では、以下の組織が TFP の認定を受ける予定である。

- OIX（Open Identity Exchange） (Open Identity Exchange)（暫定的承認）
- Kantara Initiative（暫定的承認）
- InCommon Federation （申請審査中）

一方の IdP は、以下の事業者がパイロットを実施中である。

- Google： OIX の信頼モデルを採用し、NIH (アメリカ国立衛生研究所) とパイロット評価を実施中
- Yahoo： OIX の信頼モデルを採用し、パイロット評価を実施中
- PayPal： OIX の信頼モデルを採用し、パイロット評価を実施中
- Wave
- Equifax

ICAMSC は、民間の事業者(IdP) と政府機関(RP)との間で利用するアイデンティティ技術についての認定も行う。この認定プロセス（Identity Scheme Adoption Process）[3] において、ICAM は業界標準のアイデンティティ仕様を行政 Web サイトに適用し利用するための取り決め（プロファイル）を作成する。プロファイルの作成においては、元となるアイデンティティ仕様の標準化団体などに仕様変更を要求することはない。

現在認定済み、ならびに認定作業中のプロファイルは以下の通り。

- ICAM OpenID 2.0 Profile (認定済み)
- Kantara SAML 2.0 eGovernment Profile (認定済み)
- ICAM IMI 1.0 Profile (認定済み)
- ICAM WS-Federation (作業中)

このうち ICAM OpenID 2.0 Profile および ICAM IMI 1.0 Profile について、次節以降に述べる。

3. プロファイル

3.1. OpenID 2.0 Profile

OpenID 2.0 Profile [4] は、OpenID Authentication 2.0 仕様を適用するためのプロファイルである。OMB M-04-04 が定める 4 段階の保証レベル（LOA：Levels of Identity Assurance）のうち、レベル 1 に対応している。

本プロファイルの規定では、IdP が RP に提供するユーザ識別子として PPID（Private Personal Identifier：IdP と RP の組ごとに異なる「仮名識別子」）の使用を必須とすることで、RP 間での識別子のコリレーション（名寄せ）を防止している。なお、この RP から IdP への PPID の使用の指定は、OpenID 拡張仕様のひとつである PAPE（Provider Authentication Policy Extension）の認証ポリシーによって定義される。

また本プロファイルの規定では、RP は事前に認定された IdP のリストを自身で管理し、ユーザに OpenID 識別子を直接入力させるかわりに、そのリストの中からユーザに選択させる方法のみが許可されている。これは、いわゆる「ディレクテッド・アイデンティティ」と「IdP ホワイトリスト」を組み合わせたパターンである。

その他、OpenID 認証に関するすべての通信を SSL/TLS にすることや、「IdP ホワイトリスト」を ICAM が管理し、RP がこのホワイトリストを適宜参照することなどが定められている。

3.2. IMI 1.0 Profile

IMI 1.0 Profile [5] は、Identity Metasystem Interoperability 1.0 仕様を適用するためのプロファイルである。LOA レベル 1 から 3 に対応しており、RP はユーザのカード・セレクタに、LOA 1 / 2 / 3 のうち、いずれかの LOA を示すクレームを要求しなくてはならない。また OpenID 2.0 Profile プロファイルと同様に、本プロファイルにおいても PPID の使用が必須とされている。

Information Card はマネージド・カードのみが許可されており、かつ発行者は事前に認定された IdP でなくてはならない。また Information Card を要求する RP はカード・セレクタに対し自身の X.509 証明書を提示しなくてはならず、さらにカード・セレクタは、その RP の X.509 証明書を、IdP にトークンをリクエストする際に提示する必要がある。この結果 IdP は、トークンの提出先となる RP を認識することができるようになる。さらに IdP は同時に、返却するトークンを RP の X.509 証明書の公開鍵を用いて暗号化することで、指示された RP のみに、そのトークンの内容を開示することが可能となる。

なお、トークンの型式としては SAML 1.1 アサーションのみが利用を許可されている。IdP は自身がそのトークンを発行したことを RP に示すために、その SAML アサーションに自身の X.509 証明書を含め、かつ秘密鍵を用いて署名することが規定されている。RP が信頼する IdP は、OpenID 2.0 プロファイルと同様、ICAM が管理することになっている。

4. 現在の状況

前述の通り NIH が OpenID RP となるパイロットを現在行っており、適用対象のアプリケーションは会議登録、Wiki への認証、ライブラリへのアクセス管理などとなっている。また、その他、食品医薬品局(FDA)、一般調達局(GSA) などのサイトが構築中である[6]。

一方 TFP の動向としては、前述の通り、2010 年 3 月現在では OIX [7] と Kantara Initiative [8] が暫定的な承認を受けている。前者の OIX は OpenID Foundation と Information Card Foundation が共同参画する TFP であり、LOA レベル 1 の IdP として Google (ICAM OpenID 2.0 Profile), Equifax (同 IMI 1.0), PayPal (同 OpenID 2.0 および IMI 1.0) を認定した。一方の Kantara Initiative は LOA レベル 1 から 3 までの IdP の認定に関して ICAMSC から承認を受けており、現在パイロットを行っている。

5. まとめ

米国政府のオープン・ガバメント戦略の中ではクラウド・コンピューティングの活用や行政データの公開などが注目を集めているが、IT サービスの根本である ID 情報の源泉を民間に委ねるという本イニシアチブもまた、非常に野心的な試みであると言える。実運用はまだ始まっていないが、今後同様の動きが連邦政府以外の州や市、米国以外の政府機関、そして行政サービス以外のドメインにどのように波及していくのかが注目される。

以上

参考文献

[1] Open Identity Solutions for Open Government
http://www.idmanagement.gov/drilldown.cfm?action=openID_openGOV

[2] Federal Identity, Credentialing, and Access Management Trust Framework Provider Adoption Process (TFPAP) For Levels of Assurance 1, 2, and Non-PKI 3
http://www.idmanagement.gov/documents/TrustFrameworkProviderAdoptionProcess.pdf

[3] Federal Identity, Credentialing, and Access Management Identity Scheme Adoption Process
http://www.idmanagement.gov/documents/IdentitySchemeAdoptionProcess.pdf

[4] Federal Identity, Credential and Access Management OpenID 2.0 Profile
http://www.idmanagement.gov/documents/ICAM_OpenID20Profile.pdf

[5] Federal Identity, Credentialing, and Access Management Identity Metasystem Interoperability 1.0 Profile
http://www.idmanagement.gov/documents/ICAM_IMI_10_Profile.pdf

[6] OpenID pilot project for identity management starting up at NIH - IT Compliance Advisor
http://itknowledgeexchange.techtarget.com/it-compliance/openid-pilot-project-for-identity-management-starting-up-at-nih/

[7] Open Identity Exchange
http://openidentityexchange.org/

[8] Assurance Certification Application Overview - Certification Programs - Kantara Initiative
http://kantarainitiative.org/confluence/display/certification/Assurance+Certification+Application+Overview

7. 米国連邦政府 PIV 基準体系整備の進展

宮川 寧夫

1. 米国連邦政府 PIV の背景

米国の連邦政府における政府職員の身分証明用の IC カードの調達に関しては、PIV（Personal Identity Verification）という施策がある[1]。これは、HSPD-12 (Homeland Security Presidential Directive 12) という 2004 年 8 月の政策に拠る。FIPS （Federal Information Processing Standard）として FIPS 201 という規格が 2006 年 3 月に制定された。これは連邦政府職員および契約業者用の共通識別基準である。FIPS 201 に準拠した IC-ID カードの発行が義務づけられており、2000 万枚が発行予定である。

また、NIST（National Institute of Standards and Technology）によって NPIVP（the NIST Personal Identity Verification Program）[2] という認定制度が運用されている。同プログラムの目的は、下記のとおりである。:

- ふたつの PIV コンポーネント（PIV ミドルウェアおよび PIV カードアプリケーション）の SP 800-73 仕様への準拠性を検証する
- 「NPIVP によって検証された PIV ミドルウェアと PIV カードアプリケーションの組み合わせは、相互運用可能である」という保証を提供する

NPIVP のもとで行われるすべてのテストは、NVLAP によって設立された CST（Cryptographic and Security Testing）LAP（Laboratory Accreditation Program）のもとで認定された第三者の設備において扱われる。

2. PIV 関連の SP 800 仕様書体系

PIV 規格の体系は、複数の仕様書がから構成されている。その中で SP 800-73 は、PIV ミドルウェアのインターフェイス仕様を規定しており中核的な文書となっている。

- SP 800-73: “Interfaces for Personal Identity Verification”

SP 800-85 A は、ミドルウェアおよび PIV カードアプリケーションの準拠性テストのガイドラインであり、SP 800-85 B は、PIV データモデルの準拠性テストのガイドラインである。

- SP 800-85A: “Middleware and PIV Card Application Conformance Test

Guidelines”

- SP 800-85B: “PIV Data Model Conformance Test Guidelines”

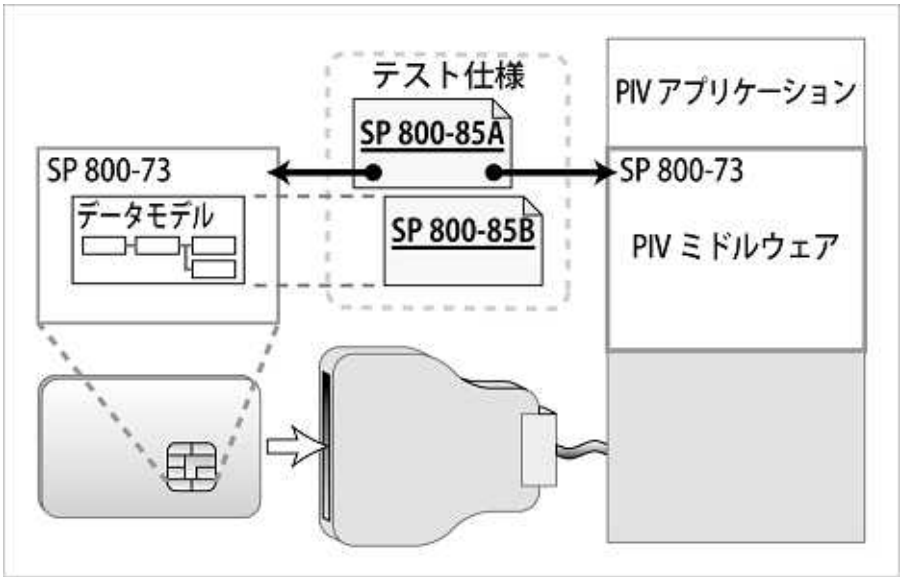

図 1：中核的仕様および準拠性テスト仕様

特殊論点として、SP 800-76 は、バイオメトリックデータ仕様を規定し、SP 800-78 は、暗号アルゴリズムおよび鍵長を規定している。

- SP 800-76: “Biometric Data Specification for Personal Identity Verification”
- SP 800-78: “Cryptographic Algorithms and Key Sizes for Personal Identity Verification”

バイオメトリクスに関して、PIV においては指紋と顔写真が使用される。SP 800-76 には、これらを格納するフォーマットとして CBEFF（Common Biometric Exchange Formats Framework）の利用が規定されているほか、指紋の登録方法や性能テストの実施方法についても規定されている。

暗号技術に関して、FIPS 201およびPIV 関連のSP 800仕様書体系において、PIV カードは、4つの暗号鍵が各用途に使用されるように規定されている。

- **本人認証用の公開鍵ペア**
- **カード認証用の鍵（公開鍵暗号と共通鍵暗号の両方を規定）**
- **ディジタル署名用の公開鍵ペア**
- **鍵管理用の公開鍵ペア**

暗号技術によって防護する対象には、下記のデータがある。

- **上記の各用途に対応したX.509公開鍵証明書**

- ディジタル署名されたCHUID（Card Holder Unique Identifier）
- ディジタル署名されたバイオメトリックデータ
- SP 800-73が規定するSecurity Object（これはディジタル署名されたハッシュテーブルである。）

制度運用に関する規定として、SP 800-79 として PIV 発行組織の運用承認についてのガイドラインと、SP 800-87 として連邦政府および外郭組織の識別についての規定がある。

- SP 800-79: “Guideline for Accreditation of PIV Issuer Organizations”
- SP 800-87: ”Codes for the Identification of Federal and Federally-Assisted Organizations”

3. 今期に発行された仕様書案

今期（2009 年 下期）に発行された仕様書の更新ドラフトには下記の 3 文書があり、中核文書が更新されていると言える。

(1) Draft SP 800-73-3: ”Interfaces for PIV” （2009 年 8 月 14 日）
(2) Draft SP 800-85B-1: “PIV Data Model Conformance Test Guidelines”（2009 年 9 月 14 日）
(3) Draft SP 800-78-2: “Cryptographic Algorithms and Key Sizes for Personal Identity Verification (PIV)”（2009 年 10 月 7 日）

注：これらのうち、(1)の SP 800-73-3 および(2)の SP 800-78-2 は、2010 年 2 月 22 日に正式に発行された。

これらの中で、(3) は 20 ページであるが、(1)は総計 127 ページ、(2) は 172 ページの文書であり大部な仕様書群となっている。これらのドラフトにおける改訂箇所を解説する。

(1) Draft SP 800-73-3: Interfaces の改訂箇所

これは、2008 年 9 月に発行された SP 800-73-2 の改定案であり、この仕様は 4 つのパートから成る。

- Part 1: End Point PIV Card Application Namespace, Data Model and Representation
- Part 1: PIV Card Application Interface
- PIV Client Application Programming Interface
- The PIV Transitional Data Model and Interfaces

Part 1 の冒頭に改訂箇所がまとめられるようになり、序文として「PIV ミドルウェアの更新設定管理」と「NVLAP 準拠性テスト」について説明されるようになったので、以前よりは文書としても読み易くなっている。

技術的には追加されたデータ項目がある。オプション項目として、鍵の履歴についてのオブジェクトとが追加された。鍵の履歴管理は、商用の IC カード仕様も備えている機能である。この項目に関連して、役目を終えた鍵管理用の鍵に対応する X.509 証明書もオプション項目として格納されることになっている。これらの項目は、特段のアクセス制御を要しないように選ばれているが、PIN（Personal identification number：暗証番号）による防護は備えている。

また、連邦政府以外が発行する PIV 準拠カードには、相互運用可能性を確保すべく UUID（Universally Unique Identifier）[3] を含めることが追加された。

(2) Draft SP 800-85B-1: PIV Data Model Conformance Test の改訂箇所

準拠性テスト仕様は、コマンドインターフェイスを扱う SP 800-85A と、このデータモデルを扱う SP 800-85B に分かれて規定されている。データモデルについてのテスト領域としては、下記の 4 つの領域が扱われる。

- すべてのデータオブジェクトについて BER-TLV （Basic Encoding Rules Tag-Length-Value）フォーマットの準拠性
- バイオメトリックデータについて CBEFF（Common Biometric Exchange Formats Framework）への準拠性
- ディジタル署名されたデータについて CMS（Cryptographic Message Syntax）への準拠性
- X.509 公開鍵証明書について FICC（Federal Identity Credentialing Committee）プロファイルへの準拠性

準拠性テストは、他の仕様書が改訂するたびに反映・修正される。

上記の中核文書の現行版である SP 800-73-2 の Part 1 中のオプション項目を反映して該当項目についての準拠性テストが追加され、最大サイズに関する準拠性テストは削除された。

また、下記の SP 800-78 の現行版を反映して、ディジタル署名についての準拠性テスト項目が更新され、証明書プロファイルについての準拠性テスト項目も更新された。

(3) Draft SP 800-78-2: Cryptographic Algorithms and Key Sizes の意義

本書は 2007 年 8 月に発行された SP 800-78 -1 の改訂案であり、使用する暗号アルゴリズムおよび鍵長が規定されている。目次構成は変わっていなが、計算機性能の向上に対する暗号技術の安全性の相対的低下の問題についての対策を反映して、より強い暗号アルゴ

リズムへの移行が規定されている。

ここでは、本人認証用の鍵と、ディジタル署名用の鍵についての規定を抜粋してみる。

表：暗号アルゴリズムおよび鍵長

鍵種別	使用期間	暗号アルゴリズムおよび鍵長
本人認証用	2013 年 12 月 31 日まで	RSA (1024 or 2048 bits) ECDSA (Curve P-256)
	上記の翌日以降	RSA (2048 bits) ECDSA (Curve P-256)
ディジタル署名用		RSA (2048 bits) ECDSA (Curves P-256 or P-384)

暗号アルゴリズム移行の必要性と相互運用可能性の観点の両面を熟慮した形跡が読み取れる。

4. 所感

PIV の施策においては、政府調達するミドルウェア仕様を規定したことによってテスト仕様も公開されるようになっている。2006 年以降の COTS 製品として PIV カードを調達できるようにしてきた施策が、より発展的に結実しつつある。PIV 仕様に基づく IC カード製品に関しては、北米市場が形成されつつあるという。

このような調達制度を わが国にも導入することは困難であるかもしれないが、技術的な観点からも参考になる事項を含んでいる仕様書群である。一連の仕様の中には先進的な要素が導入されつつあることも読み取れる。

以上

参考文献

[1] NIST, “About Personal Identity Verification (PIV) of Federal Employees and Contractors “, http://csrc.nist.gov/groups/SNS/piv/index.html

[2] “NIST's Personal identity verification program (NPIVP)”, http://csrc.nist.gov/groups/SNS/piv/npivp/index.html

[3] RFC 4122，"UUID URN Namespace"（2005 年），http://tools.ietf.org/html/rfc4122

8. OS 関連 ProtectionProfile の動向

面 和毅

1. Protection Profile

Protection Profile とは、Common Criteria(情報技術セキュリティを評価するための国際規格)を用いて評価対象製品（TOE）を評価する際に、その評価対象製品の種別に応じた、実装に依存しないニーズがまとまっているドキュメント(設計書の雛形)である。

例えば、ある OS のセキュリティを評価する際に、役割ベースのアクセス制御（Role-Based Access Control）の Protection Profile（RBAC-PP）を参照して評価し、その Protection Profile をどのレベルまで満たしているか（EAL と呼ばれる、7 段階ある保証要件）を検証して、セキュリティレベルを評価する。これにより、ある OS は RBAC-PP に対して EAL3 を取得しているが、別の OS は RBAC-PP に対して EAL4+を取得しているなど、セキュリティレベルが数値化されて比較しやすくなるため、製品導入の際に指標とすることが出来る。

Protection Profile/Common Criteria/EAL に関しては、IPA のサイト[3]からダウンロードできる Common Criteria（現在はバージョン 3.1 リリース 3）が詳しい。

2. 主要な OS 関連の Protection Profile が軒並み Sunset に

2009 年後半に、主要な OS 関連の Protection Profile が Sunset になった。Protection Profile の Sunset とは、以下のように定められている。

> Sunset: 関連した新しい Protection Profile のスタンダードが公開された際に、Protection Profile に「Sunset」の日程が付与される。そのため、sunset の日時以降で、その Protection Profile を用いて製品のセキュリティ評価を行っても承認されない。

これにより、現在 OS 関連の Protection Profile は下記の 2 つのみとなっている。

- COTS Compartmentalized Operations Protection Profile
- U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness

特に、OS 関連でのセキュリティの指標となっていた下記の Protection Profile がすべて Sunset となっている。

- CAPP（Controlled Access Protection Profile）

[3] http://www.ipa.go.jp/security/jisec/cc/index.html

- SLOSPP（Single-Level Operating System Protection Profile）
- MLOSPP（Multi-Level Operating System Protection Profile）

今のところ MLOSPP や CAPP などの OS の Protection Profile をカバーする、新しいスタンダードは存在していない。したがって、主要な OS のセキュリティ評価は、今後新しい Protection Profile が公開される迄できないことになる。なぜ、このようなことになったのであろうか。

2.1 CAPP（Controlled Access Protection Profile）

CAPP はオープンシステムの標準的なレベルである、ユーザ ID をベースとした任意アクセス制御であり、TCSEC の C2 レベルに相当する。個々のユーザが、個々のデータオブジェクト（ファイルやディレクトリ）に対して実行できるアクセスを制御することが要求される。また、システム内で発生するセキュリティに関連したイベントに対して、監査ログを取得できることが要求される。例えば前者の、データオブジェクトに対してのアクセスに関しては、監査ログに確実に記録されていくことが求められる。

2.2 Single-Level Operating System Protection Profile (SLOSPP)

米軍の COTS（commercial off-the-shelf：商用オフザシェルフ）戦略、即ち、都度軍から業者に対して特別に発注した品からではなく民生品を採用する戦略の一環。Medium Robustness 環境用である。（Robustness とは頑健さを言い、外部のノイズや設計誤差などの不確定な変動に対して，システム特性が現状を維持できることを言う。）単一レベルで認証／任意アクセス制御／監査と暗号化を要求している。CAPP の SuperSet になっているが、暗号部分が強化されている。

2.3 Multi-Level Operating System Protection Profile (MLOSPP)

米軍の COTS 戦略の一環。Medium Robustness 環境用。マルチレベルセキュリティ（MLS）で、強制アクセス制御（MAC：Mandatory Access Control）／強制インテグリティ制御（MIC：Mandatory Integrity Control）／暗号化を要求している。TCSEC の B2 レベルに相当する。

元々、マルチレベル用には LSPP と呼ばれる Protection Profile が存在した。LSPP は、Labeled Security Protection Profile の略で、すべてのサブジェクト（アクセス元。プロセスなど）とオブジェクト（アクセス対象。ファイルやディレクトリ、プロセスなど）すべてにラベルを付与して、情報の機密レベルや包含関係（部門や所属部隊など）により制御するもので、防衛システムなどで求められるセキュリティレベルである。これは、2007 年

に既に Sunset になっている。

MLOSPP は LSPP の SuperSet になっており、LSPP と比べて暗号化や MIC 部分が更に要求されるため、OS セキュリティの分野では MLOSPP を一定の EAL レベル（EAL4+など）で満たすように作成することが多くなってきていた。

現在の時点で、主要な OS が取得している Protection Profile とレベルを表 1 にまとめた。主要な OS はほぼ、IT 製品に対して高度なセキュリティ水準を求める政府機関などに対してシステムを納入するために、これらの Protection Profile で EAL4+の評価をとっている。

表 1：主要な OS が取得している Protection Profile とレベル

OS	Protection Profile	EAL
Solaris10+Trusted Extension	CAPP	4+
	RBACPP	4+
	CAPP	4+
	RBACPP	4+
AIX 5L™ 5300-05-02	CAPP	4+
AIX 5L 5300-05	RBACPP	4+
Windows 2003	CAPP	4+
Red Hat Linux 5	CAPP	4+
	RBACPP	4+
Novel SuSE Linux 8	CAPP	4+
	RBACPP	4+

4. NIAP(National Information Assurance Partnership)の方向性の変更

実際には、今回の修正は NIAP 全体の方向性の変更から来ている。NIAP（National Information Assurance Partnership）ではコモンクライテリアを利用した情報セキュリティ製品の検査を行い、Protection Profile と関連するテストセットの開発を行っている。

NIAP では、Basic/Medium Robustness の Protection Profile に関して、ベンダーや顧客からのコメントを受けて、現在の Protection Profile モデルを見直すという発表を 2009 年 3 月 16 日に行った。現行のモデルでは、すべての製品に対して同一の保証レベルを提供できると仮定していたが、インプリメンテーションによっては必要なテストプランを作成しておらず、異なったラボで一貫して評価するために、更にドキュメントが必要になっていた。そのため NIAP ではこの評価をもとに、「今後は Protection Profile とサポートドキュメントが無い限り評価を行わない」とし、評価方法を大きく見直している。

多くの技術のセキュリティ要件は、政府の数多くの機関や民間でも等しいことから、現

在 NSA が民間や Common Criteria のコミュニティと協力し、US 政府の Protection Profile に変わる、標準の Protection Profile を作成している。

将来には、それぞれの Protection Profile の EAL に関して、よりテストプランの作成やテスト結果、また評価に関する証拠をより公開していくことなどが求められ、新たに作成される Common Criteria 4.0 でもこれらのドキュメントサポートがより求められるようになっていく。

これらの変更により、最初のステップとして、2009 年 10 月から、NIAP では Basic/Medium の Robustness に関連する Protection Profile を一旦 Sunset とし、対応する新しい Protection Profile を Interim として作成していく予定になっている[4]。

OS に関しては、以上のような理由から SLOSPP/MLOSPP などが Sunset になっており、新しい OS の Protection Profile が現在 Interim としてファイナルレビュー中である[5]。

以上

[4] http://www.niap-ccevs.org/faqs/niap_evolution/
[5] http://www.niap-ccevs.org/pp/draft_pps/

9. URI のエスケープ

田中 哲

1. はじめに

Web において使われる URI（URL）には様々な情報が埋め込まれるが、その埋め込みの際にはエスケープ（パーセントエンコード）が行われる。埋め込まれた情報は取り出されるときにアンエスケープ（パーセントデコード）される。

たとえば、四則演算を行う電卓の Web アプリケーションを考える。この電卓は HTML の form でユーザに式の入力を求め、式が入力されてサーバに送られたらその結果を表示する。ここで、式は "http://calc.example.org?q=式" というような URI にブラウザがアクセスすることによって、サーバに送られるものとする。この形式は HTML の form で用いられる application/x-www-form-urlencoded という形式であり、HTML の規格で定義される。この場合、"1+1" という式を URI に埋め込むには "+" という文字を "%2B" にエスケープし、"http://calc.example.org?q=1%2B1" としなければならない。Web アプリケーション側では、URI から取り出した "1%2B1" をアンエスケープして "1+1" という式を得て計算を行うことになる。

ここで、エスケープとアンエスケープの処理が適合していないと、適切に情報を受け渡せない。たとえば上記の電卓で、送信側でエスケープを行わず "http://calc.example.org?q=1+1" とすると、application/x-www-form-urlencoded では "+" は空白（ASCII コード 0x20）を示すことになっているので、Web アプリケーションは "1 1" という文字列を受けとることになる。また、逆に Web アプリケーション側でアンエスケープを行わないと、"1%2B1" という不適切な式を受けとることになる。

エスケープ・アンエスケープの処理は様々な言語でライブラリが提供されている。ECMAScript (JavaScript) もそのひとつである。今期には ECMAScript 5th edition が承認され、そこでもエスケープ・アンエスケープのための 4 つの関数が定義されている。そこで、本稿では URI のエスケープの考え方を解説し、ECMAScript の関数について論じる。

2. エスケープの考え方

エスケープは表現したい情報を制限された形の文字列で表現するために行われる。

たとえば、C 言語の文字列リテラルは任意のバイト列をプログラムというテキスト内に埋め込むために、バックスラッシュを用いたエスケープ記法を持っている。たとえば、NUL 文字それ自体をプログラム内に直接埋め込むのは不適切であるが、エスケープ記法を用い

れば \0 という 2 文字により、NUL 文字それ自体は使わずに NUL 文字を表現することができる。また、バックスラッシュや文字列終端を示すダブルクォートを除いた表示可能文字はそのまま記述することができ、多くの場合、表現したい情報をそのまま記述できる。

URI でも、パーセントエンコーディングというエスケープ記法により、表現したい情報を URI の文法に従って埋め込める。たとえば、# という文字は query と fragment の間の区切りとして用いられているので、その文字自体を query 内の文字として表現したい場合は、エスケープして %23 としなければならない。つまり、# という文字それ自体を query に直接埋め込むことはできないが、エスケープ記法を用いれば %23 という 3 文字により、# という文字それ自体は使わずに # という文字を表現することができる。

2.1. URI の構造

URI は RFC 3986 で定義される。ただし、ここでは個々のスキーム（http, ftp, mailto など）は定義されず、それらについては別に定義される。

URI は資源の識別子を ASCII の表示可能文字のみからなる文字列で構成するようにデザインされている。ここで URI は紙に書かれることも考慮されており、そのため空白（0x20）は URI 中には使用できない。ASCII の表示可能文字は "!"（0x21）から "~"（0x7E）までの 94 文字であるが、その 94 文字の中でも、"^" などいくつかの文字は使用可能な文字から除かれている。使用できない文字はパーセントエンコーディングという %XX という形式を使って 16 進のコードで記述する。

URI の文法を RFC 3986 から抜粋すると以下のようになる。この文法は ABNF [RFC2234] という記法によって記述されている。

ABNF は伝統的な BNF（Backus-Naur Form）とよく似ている。ただし、選択はスラッシュで表現する。また、繰り返しの記法として <a>*<b>element があり、これは element の <a> 回以上 <b> 回以下の繰り返しを意味する。そして、<a> が省略された場合には 0, <b> が省略された場合には無限大を意味する。また、省略可能であることを示す記法として [elements] があり、ブラケット内が省略可能であることを意味する。なお、終端記号としてダブルクォートでくくられた形の文字列を使用できるが、これは大文字小文字を区別しないことに注意が必要である。あと、セミコロンから行末まではコメントである。

```
URI         = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

scheme      = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )

hier-part   = "//" authority path-abempty
            / path-absolute
            / path-rootless
```

```
            / path-empty

authority   = [ userinfo "@" ] host [ ":" port ]

userinfo    = *( unreserved / pct-encoded / sub-delims / ":" )

host        = IP-literal / IPv4address / reg-name

IP-literal = "[" ( IPv6address / IPvFuture  ) "]"

IPvFuture  = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )

reg-name    = *( unreserved / pct-encoded / sub-delims )

port        = *DIGIT

path-abempty  = *( "/" segment )
path-absolute = "/" [ segment-nz *( "/" segment ) ]
path-rootless = segment-nz *( "/" segment )
path-empty    = 0<pchar>  ; 空文字列

segment       = *pchar

segment-nz    = 1*pchar

query       = *( pchar / "/" / "?" )

fragment    = *( pchar / "/" / "?" )

pchar         = unreserved / pct-encoded / sub-delims / ":" / "@"

unreserved  = ALPHA / DIGIT / "-" / "." / "_" / "~"

sub-delims  = "!" / "$" / "&" / "'" / "(" / ")"
            / "*" / "+" / "," / ";" / "="
```

```
pct-encoded = "%" HEXDIG HEXDIG

HEXDIG         =  DIGIT / "A" / "B" / "C" / "D" / "E" / "F"

DIGIT          =  %x30-39 ; 0-9

ALPHA          =  %x41-5A / %x61-7A   ; A-Z / a-z
```

ここで ALPHA, DIGIT, HEXDIG は ABNF [RFC2234] から抜粋したものである。ABNF においてダブルクォートでくくられた文字列は大文字小文字を区別しないため、HEXDIG は小文字の "a", "b", "c", "d", "e", "f" も受け付ける。

また、IPv4address, IPv6address はそれぞれ IPv4 と IPv6 のアドレスを示す。これらの定義は煩雑であり、またそれらの部分ではパーセントエンコーディングが使えず本稿のテーマから外れるためここでは示さない。

ここで、pchar を展開して整理すると、文法を 3 つに分割できる。

- URI の大きな構造を定義する部分
  この部分により、URI が木構造として定義される。

```
URI         = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

hier-part   = "//" authority path-abempty
            / path-absolute
            / path-rootless
            / path-empty

authority   = [ userinfo "@" ] host [ ":" port ]

host        = IP-literal / IPv4address / reg-name

IP-literal = "[" ( IPv6address / IPvFuture  ) "]"
IPvFuture   = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )

path-abempty  = *( "/" segment )
path-absolute = "/" [ segment-nz *( "/" segment ) ]
path-rootless = segment-nz *( "/" segment )
```

```
path-empty    = 0<pchar>  ; 空文字列
```

- URI の要素の「文字列」を定義する部分

この部分は文字もしくはエスケープ（pct-encoded）が並んでいるという構造になっている。 つまり、C 言語の文字列リテラルから両端のダブルクォートを除いたような構造である。

ただし、scheme は単なる並びというわけではなく、 先頭の文字が英字であるという制約がある。また、前項目に入れた IPvFuture の一部にも文字列的な部分がある。

```
scheme      = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
port        = *DIGIT
userinfo    = *( unreserved / pct-encoded / sub-delims / ":" )
reg-name    = *( unreserved / pct-encoded / sub-delims )
segment     = *( unreserved / pct-encoded / sub-delims / ":" / "@" )
segment-nz  = 1*( unreserved / pct-encoded / sub-delims / ":" / "@" )
query       = *( unreserved / pct-encoded / sub-delims / ":" / "@" / "/" / "?" )
fragment    = *( unreserved / pct-encoded / sub-delims / ":" / "@" / "/" / "?" )
```

- 各要素に使われる文字の種類と個々のエスケープを定義する部分

この部分は文字の集合として unreserved と sub-delims を定義し、URI で用いられるエスケープの記法として pct-encoded を定義している。

```
unreserved  = ALPHA / DIGIT / "-" / "." / "_" / "~"

sub-delims  = "!" / "$" / "&" / "'" / "(" / ")"
            / "*" / "+" / "," / ";" / "="

pct-encoded = "%" HEXDIG HEXDIG

HEXDIG         =  DIGIT / "A" / "B" / "C" / "D" / "E" / "F"

ALPHA          =  %x41-5A / %x61-7A   ; A-Z / a-z

DIGIT          =  %x30-39 ; 0-9
```

ここで、文字列的な部分には segment や query などがあるが、使用できる文字種はそれ

ぞれで異なる。scheme と port は pct-encoded が使えず表現できる情報に制約があるのに対し、それ以外は pct-encoded によって任意の文字（バイト）を表現できる。

たとえば、"@" は userinfo には使えないが、segment には使用できる。この違いは、それぞれが利用される文脈が原因である。userinfo は authority で利用されるが、そこでは userinfo と host が "@" によって区切られている。そのため "@" を使用可能にすると、"@" が userinfo の内容なのか host との区切りなのか解釈が曖昧になってしまう。それに対して、segment が利用される path-absolute などでは "@" を区切りとして用いておらず、segment に "@" を使用しても曖昧にはならない。つまり、userinfo の定義に "@" が含まれないのは、authority の構造が原因である。

同様に、この文法の中で使われている区切り文字には ":", "/", "?", "#", "[", "]", "@" があり、これらについても各文字列部分において問題が起きなければ許されている場合がある。

- scheme では pct-encoded は許されず、使用できる文字は英数字と "+", "-", "." と制約されている。また、先頭文字はさらに英字だけに制限されている。

- port では pct-encoded は許されず、使用できる文字は数字のみである。これは、ポート番号は数字だけで表現できることがあきらかで、他の文字やパーセントエンコーディングを使用可能にする利点がないからだと考えられる。

- userinfo では ":" は許されるが、"/", "?", "#", "[", "]", "@" は許されない。
  - もし "/" が許された場合、"http://user/info@host" は authority が "user/info@host" と"user" のどちらなのか曖昧になる。後者の場合、path-abempty が "/info@host" になる。
  - もし "?" が許された場合、"http://user?info@host" は authority が "user?info@host" と"user" のどちらなのか曖昧になる。後者の場合、query が "info@host" となる。
  - もし "#" が許された場合、"http://user#info@host" は authority が "user#info@host" と"user" のどちらなのか曖昧になる。後者の場合、fragment が "info@host" となる。
  - もし "@" が許された場合、"http://user@info@host" は userinfo と host の区切りがわかりにくい。ただし、host には "@" が許されていないので、userinfo だけに "@" を許すのは曖昧ではない。
  - "[" と "]" は IPv6 アドレスを区切るためにしか許されておらず、userinfo は IPv6 アドレスが記述される host とは "@" で区切られている。そのため、"[" と "]" を許しても曖昧にはならない。

- reg-name では ":", "/", "?", "#", "[", "]", "@" は許されない。reg-name は登録されたホスト名の意味であり、ホスト名・ドメイン名を表現するのに使われる。なお、ホスト名・ドメイン名には使用できる文字に制約があるが、この文法はその制約を表現していない。
  - もし ":" が許された場合、"http://host:80" は host が "host:80" と "host" のどちらなのか曖昧になる。後者の場合、port が "80" となる。
  - もし "/" が許された場合、"http://ho/st" は host が "ho/st" と "ho" のどちらなのか曖昧になる。後者の場合、path-abempty が "/st" となる。
  - もし "?" が許された場合、"http://ho?st" は host が "ho?st" と "ho" のどちらなのか曖昧になる。後者の場合、query が "st" となる。
  - もし "#" が許された場合、"http://ho#st" は host が "ho#st" と "ho" のどちらなのか曖昧になる。後者の場合、fragment が "st" となる。
  - もし "@" が許された場合、"http://userinfo@ho@st" は userinfo と host の区切りがわかりにくい。ただし、userinfo には "@" が許されていないので、host だけに "@" を許すのは曖昧ではない。
  - もし "[" と "]" が許された場合、"http://[v1.xxxx]" は reg-name（登録された名前）が "[v1.xxxx]" なのか、IPvFuture（未来の IP アドレス）が "v1.xxxx" なのか曖昧になる。

- segment と segment-nz では ":", "@" は許されるが、"/", "?", "#", "[", "]" は許されない。segment と segment-nz は階層的な構造を表現する部分であり、ファイルシステムを URI にマップする時に、ディレクトリ階層の個々のファイル名を対応させるなどと使われる。
  - もし "/" が許された場合、"http://host/foo/bar" は "foo/bar" が単一の segment なのかふたつの segment なのか曖昧になる。
  - もし "?" が許された場合、"http://host/foo?bar" は segment が "foo?bar" と "foo" のどちらなのか曖昧になる。後者の場合、query が "bar" となる。
  - もし "#" が許された場合、"http://host/foo#bar" は segment が "foo#bar" と "foo" のどちらなのか曖昧になる。後者の場合、fragment が "bar" となる。
  - "[" と "]" は IPv6 アドレスを区切るためにしか許されておらず、segment と segment-nz は IPv6 アドレスが記述される host とは "/" で区切られている。そのため、"[" と "]" を許しても曖昧にはならない。

- query では ":", "@", "/", "?" は許されるが、"#", "[", "]" は許されない。
  - もし "#" が許された場合、"http://host?foo#bar" は query が "foo#bar" と "foo" のどちらなのか曖昧になる。後者の場合、fragment が "bar" になる。

    - "[" と "]" は IPv6 アドレスを区切るためにしか許されておらず、query は IPv6 アドレスが記述される host とは "?" で区切られている。そのため、"[" と "]" を許しても曖昧にはならない。

- fragment では ":", "@", "/", "?" は許されるが、"#", "[", "]" は許されない。
    - もし "#" が許された場合、"http://host?query#foo#bar" は query と fragment の区切りがわかりにくい。ただし、query には "#" が許されていないので、fragment に "#" を許すのは曖昧ではない。なお、fragment に "#" が許されていないことにより、URI の最後から左に向かって最初の "#" を探すことにより fragment の範囲を決定できる。
    - "[" と "]" は IPv6 アドレスを区切るためにしか許されておらず、fragment は IPv6 アドレスが記述される host とは "#" で区切られている。そのため、"[" と "]" を許しても曖昧にはならない。

なお、"[" と "]" が IPv6 アドレスを区切るためだけに使われているのは、RFC 2396 では IPv6 アドレスが許されておらず、RFC 2732 でその部分が拡張されたためと考えられる。

2.2. スキーム依存な URI の構造

前節で述べた URI の構造は RFC 3986 で定義される、すべての URI に共通する構造である。

個々のスキームでは URI の中にさらに構造を導入できる。たとえば、ftp URI は RFC 1738 で定義されるが、転送にバイナリモードを使うことを指示するのに "ftp://host/dir1/dir2/filename;type=i" というように、";type=i" という指定を使用できる。URI の構文からすると "filename;type=i" の部分が segment に対応する。つまり、segment の中身を ";" で区切って、転送のモードを指定するという規則を ftp スキームが定義している。

このように、各スキームでは segment をさらに区切って構造を定義できる。このように構造を定義できる対象は segment に限らず、文字列的な構造の部分であればどこでも可能である。

ここで追加された区切り文字を区切りでなく使いたい場合、パーセントエンコーディングを用いる。たとえば、上記の ftp の例で filename 内に ";" を使いたければ、"%3B" を用いる。

従って、ftp URI の filename の中では ";" と "%3B" の意味は異なる。このように、ある文字とそのパーセントエンコーディング表現で意味を異なるものとしてよい文字は RFC 3986 で決まっており、どの文字でもそうしてよいわけではない。以下に示す unreserved

に属する文字はそのように意味を異ならせてはならない。つまり、"0" と "%30" の意味は常に一致しなければならない。

```
unreserved  = ALPHA / DIGIT / "-" / "." / "_" / "~"
```

それに対し、以下の reserved に属する文字は意味を異ならせてもよい。

```
reserved    = gen-delims / sub-delims

gen-delims  = ":" / "/" / "?" / "#" / "[" / "]" / "@"

sub-delims  = "!" / "$" / "&" / "'" / "(" / ")"
                  / "*" / "+" / "," / ";" / "="
```

ここで reserved は gen-delims と sub-delims にわかれているが、gen-delims は RFC 3986 で定義される URI 一般の部分で使われている区切りであり、sub-delims はそれ以外の区切りである。

各スキームでは文法として許される範囲で reserved の文字を区切りとして採用できる。たとえば、segment の中で ";" を区切りとして構造を定義することはできるが、"/" を区切りとすることはできない。これは、segment 自体が "/" で区切られた構造であるため、その中で "/" を区切りとすることは曖昧になってしまうからである。このことは、segment の文法で "/" が許されていないことからもわかる。

また、区切りを定義するのはスキームだけとは限らない。http では、Web ブラウザは URI をエスケープされたままの形で Web サーバに送信する。つまり http というプロトコル自体はパーセントエンコーディングかどうかという違いの意味を定義しない。そのため、パーセントエンコーディングの使い方はブラウザとサーバの合意次第ということになる。

この合意のひとつの例が HTML の form である。HTML の form を submit する方法のひとつに form に入力した内容を application/x-www-form-urlencoded という形式でまとめたものを URI の query とし、その URI を http でリクエストするというものがある。この application/x-www-form-urlencoded では "k1=v1&k2=v2" などというように、キーと値のペアのリストを "=" と "&" で query としてまとめる。ここでキーや値に "=" や "&" を使いたい場合、パーセントエンコーディングにより "%3D" や "%26" を用いる。また、"&" のかわりに ";" を使えることも推奨されているので、";" という文字自体には "%3B" を用いる。これは、URI が規定しなかった query 内の文字の意味を application/x-www-form-urlencoded という形式で定義し、Web ブラウザと Web サーバ

（Web アプリケーション）が form の submit における情報伝達にその定義を採用しているとみなせる。

なお、application/x-www-form-urlencoded には空白（0x20）を表現するのに "+" を用い、"+" を表現するには "%2B" を用いるという規則もある。

この "+" による空白は application/x-www-form-urlencoded で定義されているものなので、URI の application/x-www-form-urlencoded 以外の部分（segment など）では使えない。そのような部分には segment など query 以外の部分がある。また query でも application/x-www-form-urlencoded でない用途に使われている場合は、application/x-www-form-urlencoded の規則は適用されない。

3. ECMAScript の URI 関数

ECMAScript では以下の 4 つの関数を URI のエスケープ・アンエスケープ用として提供している。

- encodeURI
- decodeURI
- encodeURIComponent
- decodeURIComponent

どの関数も文字列を受け取り、文字列を返す。なお JavaScript の文字列は Unicode の文字列であり、UTF-16 による表現が想定されている。

encodeURI と decodeURI は URI 全体に適用するもので、encodeURIComponent と decodeURIComponent は URI の要素に適用するものと意図されている。

これらの関数は ECMAScript 3rd edition で導入された関数である。そのため、URI の仕様として RFC 3986 でなく、古い RFC 2396 を参考に設計されている。本稿では、RFC 3986 と照らし合わせて論じるが、RFC 2396 の影響についてもその都度述べる。

それぞれの関数の動作を以下に述べる。

3.1. encodeURI

encodeURI は URI 全体を文字列として受け取って、エスケープして返す関数である。具体的には、引数中の文字で以下の文字以外は UTF-8 としてパーセントエンコードした文字列を返す。

- 英数字 (A-Z a-z 0-9)

- - . _ ~
- : / ? # @
- ! $ & ' ( ) * + , ; =

URI の定義と照らし合わせると、この文字のリストは unreserved および reserved から "[", "]" を除いたものである。ここでとくに、"%" はあげられていないため、encodeURI("%") は "%25" となる。

このため、生成できない URI が存在する。たとえば、encodeURI("#") は "#" がリストに含まれているため "#" となる。encodeURI("%23") は "%" がリストに含まれていないため "%2523" となる。

つまり、encodeURI は "%23" を含む結果は生成できない。("#" の ASCII コードは 0x23 で、"%" の ASCII コードは 0x25 である。) このように生成できない URI が存在するため、encodeURI は任意の URI を構成する用途には使えない。

そもそも、URI は構成した時点で区切りが曖昧にならないようにエスケープが必要であり、URI を構成した後でエスケープするというのは不適切である。

3.2. decodeURI

decodeURI は URI 全体を文字列として受け取って、アンエスケープして返す関数である。

具体的には、引数中のパーセントエンコーディングを UTF-8 としてアンエスケープするが、アンエスケープした結果が以下の文字のどれかのときはアンエスケープせずにパーセントエンコーディングのまま残す。

- : / ? # @
- $ & + , ; =

URI の定義と照らし合わせると、この文字のリストは gen-delims から "[", "]" を除いたものおよび sub-delims の一部である。

ここで、decodeURI("%23") と decodeURI("%2523") はどちらも "%23" となる。前者は "%23" をアンエスケープした "#" がリストに入っているためであり、後者は "%25" をアンエスケープした "%" がリストに入っていないためである。つまり、decodeURI を適用すると、元の URI 内にあった "%23" と "%2523" を区別できなくなる。

decodeURI は encodeURI の結果を逆変換するのに使用できる。ただし、この逆変換には後述する decodeURIComponent も使用できる。decodeURIComponent は decodeURI よりも単純な動作の関数であり、逆変換のためには上記のパーセントエンコーディングを残す動作は不要である。

そもそも、URI は URI 全体をひとつの文字列とする限り区切りが曖昧にならないよう

にエスケープが必要であり、URI のままアンエスケープするというのは不適切である。

3.3. encodeURIComponent

encodeURIComponent は文字列を受け取って URI の要素として適切なエスケープを行って返す関数である。具体的には、引数中の文字で以下の文字以外は UTF-8 としてパーセントエンコードした文字列を返す。

- 英数字 (0-9 A-Z a-z)
- - . _ ~
- ! ' ( ) *

URI の定義と照らし合わせると、この文字のリストは unreserved および sub-delims の一部である。

この関数が対象としている URI の要素というのは、URI 中の文字列的な部分と考えられる。つまり、ある文字列を URI を通して伝達する場合、送信側で文字列を encodeURIComponent でエスケープして URI に埋め込み、受信側で URI から取り出して decodeURIComponent でもとの文字列を得る。この埋め込み・取り出しが確実に行えるためには、URI で用いられる区切り文字が encodeURIComponent の結果に現れてはならない。もし区切り文字が現れると、本当の区切り文字と encodeURIComponent の結果内のものを誤認して、埋め込んだものが取り出せない可能性が発生する。

URI 内で実際に区切り文字として使われている文字は、URI の各箇所において異なるが、unreserved の文字は区切り文字としては使ってはならないので、encodeURIComponent の結果に使うことは問題ない。しかし、上記のリストにある sub-delims の文字は区切り文字として使うことが許されているため、情報の伝達に問題が生じる可能性がある。

なお、上記の "!", "'", "(", ")", "*" が sub-delims の文字であるが、これらは RFC 2396 では unreserved に属していた。そのため、リストにこれらの文字が入っているのは不思議ではない。

3.4. decodeURIComponent

decodeURIComponent は URI の要素を文字列として受け取って、アンエスケープして返す関数である。具体的には、引数中のパーセントエンコーディングを UTF-8 としてアンエスケープする。この関数は decodeURI と異なり、すべてのパーセントエンコーディングを例外なくアンエスケープする。

この関数は他の関数と異なり、例外的な動作の無い、単純な仕様になっている。

decodeURIComponent は encodeURI と encodeURIComponent の両方の逆変換に使用できる。これは decodeURIComponent がすべてのパーセントエンコーディングをアン

エスケープするためである。encodeURI と encodeURIComponent の違いはパーセントエンコードする文字の集合の違いであるが、decodeURIComponent はどの文字でもエスケープされていればアンエスケープするので逆変換となる。

3.5. ECMAScript の URI 関数の問題

この節では ECMAScript で提供されている 4 つ URI 関数を説明し、問題点を指摘した。まとめると、以下のようになる。

- encodeURI, decodeURI: URI 全体をエスケープ・アンエスケープするという考え方がおかしい
- encodeURIComponent: URI の最新の仕様に追随しておらずエスケープが足りない
- decodeURIComponent: 問題なし

4. ありえたかもしれない仕様

前節で述べたように、ECMAScript の関数にはいくつかの問題がある。この節ではパーセントエンコーディングを扱う関数の仕様について論じる。

4.1. パーセントエンコーディングでエスケープする文字種

エスケープを行う関数では、どの文字をエスケープするかが問題となる。このことで考慮すべき要素には以下のものがある。

- 人間が URI を読みやすいように、なるべくエスケープしないほうがよい
- 伝えたい情報から URI に埋め込む形に変換する場合、"%" は常にエスケープしなければならない
- エスケープした結果を埋め込む箇所において許されていない区切り文字は使ってはならない
- URI を定義する RFC は改版を重ねるに従って reserved は多くなっている

まず、URI は人間にとって覚えやすいものであるようにデザインされているため、その趣旨を考えると、なるべくエスケープはしないほうが良い。エスケープをすると人間には内容を理解することが困難になる。たとえば、冒頭の電卓の例では、"1+1" という文字列を "1%2B1" とエスケープする必要があったが、普通は "1%2B1" が "1+1" であると読み取ることは難しい。

しかし、"%" をエスケープするのは、アンエスケープによって元の情報を取り出すため

には必須である。もし、"%" をエスケープしなければ、"%23" は "#" をエスケープした結果なのか、もともと "%23" だったのか曖昧になり、エスケープ前の情報を再現できなくなってしまう。

また、URI で使われている区切り文字を使うのも同様に曖昧な結果をもたらす。もし、URI の query 部分に埋め込む文字列を生成するのに"#" をエスケープしない変換を用いると、生成された "http://host/?foo#bar" という URI からもとの文字列は得られない。これはもとの文字列が "foo" と "foo#bar" のどちらの可能性も否定できないためである。

ここで、URI の区切り文字は URI の各部分によって異なるが、区切り文字になりうる文字は reserved に属する文字だけである。この理由は、unreserved に属する文字はパーセントエンコードするかどうかで意味を変えてはならないため、それを区切り文字とすると、区切りではない文字としてその文字を表現することが困難になるからである。

そして、URI の仕様は RFC の改版にともなって、reserved に属する文字が増える傾向にある。たとえば、"!" は RFC 2396 で unreserved に属していたが、RFC 3986 では reserved に属している。
これに関しては後述する。

これらの点を考慮すると ECMAScript の encodeURIComponent は RFC 2396 の古い定義の reserved/unreserved を使っているいうことを除けば、良い仕様であるといえる。unreserved 以外をすべてエスケープするというのは、 reserved をすべてエスケープすることになるため、URI のどの箇所に埋め込んでもそこで使っている区切り文字と衝突することがない。また、unreserved をエスケープしないことにより、人間にとっての可読性を最大限保っている。もちろん "%" もエスケープするので、元の情報を確実に取り出すことができる。

唯一の問題は、RFC の変化に対応できなかったことである。これに対応するには RFC 2396 の unreserved ではなく、RFC 3986 の unreserved を使って動作を定義すれば良い。または、将来 RFC が変化してさらに文字が unreserved から reserved へ移ることを想定するなら、（さすがに英数字は unreserved のままだと期待して）unreserved のかわりに英数字のみをエスケープせずに残すという仕様も考えられる。

4.2. エスケープしたものはエスケープ済みだと分かるようにする

URI に関わるセキュリティ問題として、多重エスケープ・アンエスケープがある。これは、パーセントエンコード・デコードを複数回適用してしまうというものである。

これが問題になるのは、不適切な入力の検査に失敗することがあるためである。

たとえば、ディレクトリトラバーサルという脆弱性を防ぐために必要な検査のひとつに、URI に含まれている segment が ".." というファイル名でないことを調べる検査がある。これは、一般にファイル名は "%" が任意に含まれるなど、segment としては適切でないため、segment はファイル名をエスケープしたものとなる。ここで検出の対象である ".."

はファイル名であるので、segment をまずアンエスケープしてファイル名に変換し ".." と比較するというのが正しい方法である。もし、アンエスケープせずに segment のまま ".." と比較すると、"%2E.", ".%2E", "%2E%2E" を検出できない。これらはアンエスケープすると ".." となるので、この間違いはディレクトリトラバーサル脆弱性の原因となる。

ここで、正しい方法を使っても、つまりアンエスケープしてから比較しても、まだ注意しなければならないことがある。比較した後でさらにアンエスケープしてはならないという点である。これが多重アンエスケープの問題である。もし、これを行ってしまうと、segment として "%252E%252E" が与えられた場合に、アンエスケープして "%2E%2E" が得られ、これは ".." とは異なるので問題は検出されないが、それをもう一度アンエスケープした ".." をファイル名として使ってしまう。

この間違いがおこるのは、プログラム内で、アンエスケープの前も後も単なる文字列であり、その区別が明らかでないからである。とくに、"abc" などエスケープ・アンエスケープで変化しない文字列は、値をみてもエスケープ済みかどうかわからない。このため、"%2E%2E" など、一回エスケープすると ".." などのアンエスケープで変化しない文字列になるデータでテストしても、多重アンエスケープ問題は発見できない。

この類の間違いを発見しやすくするデザインとしては以下のようなものが考えられる。

- application/x-www-form-urlencoded 専用のエスケープ・アンエスケープ関数を作る
  application/x-www-form-urlencoded は "K1=V1&K2=V2&..." という形式である。
  ここで、K1,K2, ... は HTML form 内のコントロール名であり、V1, V2, ... は対応する値である。
  K1, V1, K2, V2, ... は先に述べたようにエスケープされていなければならない。
  ここで、エスケープをする前のコントロール名と値のペアのリスト [[k1, v1], [k2, v2], ...] を受け取り、各文字列をエスケープした後で "=" と "&" で連結して "K1=V1&K2=V2&..." という形式の文字列を生成する関数が考えられる。この関数を仮に encodeWWWForm という名前とすると、以下のように動作する。

  - encodeWWWForm([["k1","v1"], ["k2", "v2"]]) => "k1=v1&k2=v2"
  - encodeWWWForm([["q","1+1=2"]]) => "q=1%2B1%3D2"
  - encodeWWWForm([[" :/?#[]@!$&'()*+,;=-._~", " :/?#[]@!$&'()*+,;=-._~"]]) => "+:/?%23%5B%5D@!$%26'()*%2B,%3B%3D-._~=+:/?%23%5B%5D@!$%26'()*%2B,%3B%3D-._~"

この関数には以下の利点がある。

  - ふたつ (以上) の文字列がひとつの文字列に合成されるので、未エスケープの文字

列とエスケープ済みの文字列を取り違えにくい

- 結果の文字列に必ず "=" が含まれるので、値を調べるだけでだいたい合成済みだと分かる(未エスケープの文字列に "=" が含まれている場合を考えると、確実に区別できるわけではない)
- encodeURIComponent よりもエスケープが少なく人間が結果を読みやすい

同様に、分解とアンエスケープを同時に行う関数も考えられる。つまり、"K1=V1&K2=V2&..." という文字列を受け取り、[[k1, v1], [k2, v2], ...] を返すというものである。この関数を仮に decodeWWWForm という名前とすると、以下のように動作する。

- decodeWWWForm("k1=v1&k2=v2") => [["k1","v1"], ["k2", "v2"]]
- decodeWWWForm("q=1%2B1%3D2") => "q=1%2B1%3D2"
- decodeWWWForm("+:/?%23%5B%5D@!$%26'()*%2B,%3B%3D-._~=+:/?%23%5B%5D@!$%26'()*%2B,%3B%3D-._~") => [[" :/?#[]@!$&'()*+,;=-._~", " :/?#[]@!$&'()*+,;=-._~"]]

この関数にも同様の利点がある。

- ひとつの文字列がふたつ (以上) の文字列に分解されるので、エスケープ済みの文字列と未エスケープの文字列を取り違えにくい
- 結果の文字列には "=" が含まれないことが多いので、値を調べるだけでだいたい分解済みだと分かる（未エスケープの文字列に "=" が含まれている場合を考えると、確実に区別できるわけではない。）

- HTML の属性専用のエスケープ関数を作る

  URI と同様にエスケープがセキュリティ問題になり得るケースとして、HTML の生成がある。

  HTML の生成には URI とは異なるエスケープが必要である。たとえば、"&" という文字を表現したければ "&" や "&" といったものに変換しなければならない。そして、URI と同様に、エスケープが必要となる文字は HTML の各部分によって異なる。HTML の地の文 (#PCDATA) では、ダブルクォートをそのまま記述できるが、属性（たとえば <a href="http://example.org"> の "http://example.org" の部分）をダブルクォートで括って記述する中では、ダブルクォートは直接は記述できず """ などとしなければならない。これは、ダブルクォートを直接記述すると属性の終端と混同して曖昧になってしまうからである。

  また、属性を括るにはシングルクォートも使用できるが、この場合はシングルクォ

ートを直接には記述できなくなる。

ここで、Web アプリケーションがユーザ入力を元に HTML を生成する場合、エスケープを行う必要がある。エスケープを忘れると XSS（Cross Site Scripting）という脆弱性の原因となる。

ここで、属性についてエスケープを忘れない方法として、
URI の application/x-www-form-urlencoded と同様の方法が考えられる。つまり、文字列をエスケープした後にダブルクォートで括った文字列を返す関数を用意するのである。たとえば、"a&b" という 3 文字の長さの文字列をこの関数を使ってエスケープすると、 "a&b" という 7 文字の長さの文字列ではなく、"\"a&b\"" という 9 文字の長さの文字列を生成する。

この関数には以下の利点がある。

- エスケープ済みの文字列にはダブルクォートが付加されるので、文字列をみるとエスケープ済みかどうかだいたい分かる（未エスケープの文字列の最初と最後にダブルクォートがついている場合を考えると、確実に区別できるわけではない。）
- エスケープ済みの文字列を再度エスケープすると、ダブルクォートが """ に変化するので間違いに気がつく
- HTML の開始タグを生成するところで、属性を括るダブルクォートを記述しなくて済む

5. URI の変遷

URI は RFC で定義されるが、前述したように定義の内容は変化している。この節では文字種の分類について、その変遷をまとめる。

まず、URI を定義する RFC は以下のように変遷している。

- RFC 1738　Uniform Resource Locators (URL)
- RFC 1808　Relative Uniform Resource Locators
- RFC 2396　Uniform Resource Identifiers (URI): Generic Syntax
- RFC 2732　Format for Literal IPv6 Addresses in URL's
- RFC 3986　Uniform Resource Identifier (URI): Generic Syntax

最初に RFC 1738 で絶対 URL が定義され、RFC 1808 で相対 URL が定義された。そして、RFC 2396 で URL と URN（Uniform Resource Name）を統合するものとして URI が定義された。URN は RFC 2141 で定義されるが、文法的には URI のスキームのひとつとみなせるので、本稿では触れない。RFC 2732 では IPv6 のアドレスを記述する方法が追加された。そして、RFC 3986 が現時点（2010 年 1 月）における最新版である。

以下の表は ASCII の表示文字それぞれについて、各 RFC がどの文字種に分類しているかを示したものである。文字の性質として重要なのはそれが unreserved に属するか、reserved に属するか、あるいはどちらでもないかである。unreserved と reserved といった文字種の定義はそれぞれの RFC で以下のようになっている。... という省略は文字のリストが記述されている部分である。

```
RFC3986:
  reserved    = gen-delims / sub-delims
  gen-delims  = ...
  sub-delims  = ...
  unreserved  = ALPHA / DIGIT / ...
  ALPHA       = ...
  DIGIT       = ...

RFC2396:
  reserved    = ...
  unreserved  = alphanum / mark
  alphanum    = alpha / digit
```

```
  alpha       = lowalpha / upalpha
  mark        = ...
  digit       = ...
  lowalpha    = ...
  upalpha     = ...
  delims      = ...  ; reserved と unreserved には使われていない
  unwise      = ...  ; reserved と unreserved には使われていない

RFC1738,1808:
  reserved    = ...
  unreserved  = alpha / digit / safe / extra
  alpha       = lowalpha / hialpha
  digit       = ...
  safe        = ...
  extra       = ...
  lowalpha    = ...
  hialpha     = ...
  national    = ...  ; reserved と unreserved には使われていない
  punctuation = ...  ; reserved と unreserved には使われていない
```

たとえば、RFC 3986 の gen-delims には ":" が含まれているので、":" は gen-delims に属し、gen-delims は reserved の一種である。これを表では gen-delims(r) と記述する。(r) は reserved の意味である。

また、同様に unreserved は (u) と記述する。

また、ASCII という文字コードは ISO 646 という国際規格の米国版である。ISO 646 は 7bit コードの定義であり、一部のコードを各国で変えてよいという仕組みになっている。以下の表の「可変」はその変えてよい文字である。(本稿では、ISO 646 自体でなく、その日本版の規格である JIS X 0201 - 1997 を参考にした。)

また、英数字はそれぞれすべて扱いが同じなため、代表して "0", "A", "a" を記載する。

				URI	IPv6	URI	URL	ISO646
oct	dec	hex	**文字**	RFC 3986	RFC 2732	RFC 2396	RFC 1738,1808	
041	33	21	!	sub-delims(r)		mark(u)	extra(u)	
042	34	22	"			delims	punctuation	
043	35	23	#	gen-delims(r)		delims	punctuation	**可変**: NUMBER SIGN / POUND SIGN
044	36	24	$	sub-delims(r)	reserved(r)	reserved(r)	safe(u)	**可変**: DOLLER SIGN / CURRENCY SIGN
045	37	25	%			delims	punctuation	
046	38	26	&	sub-delims(r)	reserved(r)	reserved(r)	reserved(r)	
047	39	27	'	sub-delims(r)		mark(u)	extra(u)	
050	40	28	(	sub-delims(r)		mark(u)	extra(u)	
051	41	29	)	sub-delims(r)		mark(u)	extra(u)	
052	42	2A	*	sub-delims(r)		mark(u)	extra(u)	
053	43	2B	+	sub-delims(r)	reserved(r)	reserved(r)	safe(u)	
054	44	2C	,	sub-delims(r)	reserved(r)	reserved(r)	extra(u)	
055	45	2D	-	unreserved(u)		mark(u)	safe(u)	
056	46	2E	.	unreserved(u)		mark(u)	safe(u)	
057	47	2F	/	gen-delims(r)	reserved(r)	reserved(r)	reserved(r)	
072	58	3A	:	gen-delims(r)	reserved(r)	reserved(r)	reserved(r)	
073	59	3B	;	sub-delims(r)	reserved(r)	reserved(r)	reserved(r)	
074	60	3C	<			delims	punctuation	
075	61	3D	=	sub-delims(r)	reserved(r)	reserved(r)	reserved(r)	
076	62	3E	>			delims	punctuation	
077	63	3F	?	gen-delims(r)	reserved(r)	reserved(r)	reserved(r)	
100	64	40	@	gen-delims(r)	reserved(r)	reserved(r)	reserved(r)	**可変**
133	91	5B	[	gen-delims(r)	reserved(r)	unwise	national	**可変**
134	92	5C	\		unwise	unwise	national	**可変**
135	93	5D	]	gen-delims(r)	reserved(r)	unwise	national	**可変**
136	94	5E	^		unwise	unwise	national	**可変**
137	95	5F	_	unreserved(u)		mark(u)	safe(u)	
140	96	60	`		unwise	unwise	national	**可変**
173	123	7B	{		unwise	unwise	national	**可変**
174	124	7C	|		unwise	unwise	national	**可変**
175	125	7D	}		unwise	unwise	national	**可変**
176	126	7E	~	unreserved(u)		mark(u)	national	**可変**
060	48	30	0	DIGIT(u)		digit(u)	digit(u)	
101	65	41	A	ALPHA(u)		upalpha(u)	hialpha(u)	
141	97	61	a	ALPHA(u)		lowalpha(u)	lowalpha(u)	

RFC 1738,1808 から RFC 2396 では、"$", "+", "," が unreserved から reserved へ変化し、"~" が national から unreserved に変化している。RFC 1738,1808 の national は reserved でも unreserved でもなく、使用できない文字である。

RFC 2396 から RFC 2732 では、"[", "]" が unwise から reserved へ変化している。RFC 2396 の unwise は reserved でも unreserved でもなく、使用できない文字である。ここで、"[", "]" は IPv6 アドレスを括るために許されたが、それ以外の部分では使えないままである。なお、RFC 2732 では IPv6 アドレスのために reserved と unwise を修正するという形で定義したため、RFC 2732 で記述されていない部分は RFC 2396 と同じである。

RFC 2732 から RFC 3986 では、"#" が delims から reserved へ変化し、"!", "'", "(", ")", "*" が unreserved から reserved へ変化している。ここで、"#" の変化は、RFC 2732 までは fragment の部分が URI の一部とはみなされていなかったのが、RFC 3986 で URI の一部とみなされるようになったことに伴う変化であり、実際には以前から "#" は fragment に前置する区切りとして使えていた。

URI で使われる文字は、reserved, unreserved およびパーセントエンコーディングの一部としての "%" である。それ以外の文字は使えない。使えない文字の理由として、RFC 1738 では、"<", ">", "\"", "#" は URI を自体を区切るのに使われるために禁止されると述べられている。また、"{", "}", "|", "\", "^", "~", "[", "]", "`" の 9 文字はゲートウェイなどによって変更されることがあるので禁止されると述べられている。これはこの 9 文字がすべて ISO 646 の各国版で変化する文字であることと、その 9 文字が national という名前で参照されていることから、ISO 646 の各国版の違いに起因すると考えられる。ただし、ISO 646 で変化可能な部分すべてが禁止されているわけではなく、"$" と "@" は RFC 1738 の時点で許されているし、その後 RFC 2396 で "~"、RFC 2732 で "[" と "]"、RFC 3986 で "#" が許されるように変化している。このように、URI は使える文字が増えていく傾向にある。

また、記号が unreserved から reserved へ変化することがある。RFC 2396 では、"$", "+", "," が unreserved から reserved へ変化している。RFC 2732 では、"[", "]" が unwise から reserved へ変化している。RFC 3986 では、"!", "'", "(", ")", "*" が unreserved から reserved へ変化している。このように、URI の定義では、reserved が増えていく傾向にある。

6. まとめ

本稿では、ECMAScript の関数を題材として URI のエスケープについて論じた。

decodeURIComponent は問題なく使える関数である。encodeURIComponent は RFC

の変化に追随していないため、用途によっては問題が生じうる。encodeURI **と** decodeURI **は使わない方がよい。**

一般に、エスケープの正しい動作は用途に依存する。このため、エスケープ関数は用途をはっきりと決めて設計しなければならない。そして、その用途は関数名やドキュメントでわかるようにすべきである。また、多重エスケープ防止のため、可能なら、未エスケープの状態とエスケープ済みの状態を値で区別できることが望ましい。

以上

参考文献

[RFC1738]
Uniform Resource Locators (URL), T. Berners-Lee, L. Masinter, M. McCahill, December 1994,
http://www.ietf.org/rfc/rfc1738.txt

[RFC1808]
Relative Uniform Resource Locators, R. Fielding, June 1995,
http://www.ietf.org/rfc/rfc1808.txt

[RFC2396]
Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter, August 1998,
http://www.ietf.org/rfc/rfc2396.txt

[RFC2732]
Format for Literal IPv6 Addresses in URL's, R. Hinden, B. Carpenter, L. Masinter, December 1999,
http://www.ietf.org/rfc/rfc2732.txt

[RFC3986]
Uniform Resource Identifier (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter, January 2005,
http://www.ietf.org/rfc/rfc3986.txt

[RFC2141]
URN Syntax, Moats, R., May 1997,
http://www.ietf.org/rfc/rfc2141.txt

[RFC2234]
Augmented BNF for Syntax Specifications: ABNF, D. Crocker, Ed., P. Overell,

November 1997,
http://www.ietf.org/rfc/rfc2234.txt
[HTML4.01]
HTML 4.01 Specification, December 1999,
http://www.w3.org/TR/1999/REC-html401-19991224/
[JISC]
JIS X3010:2003 **プログラム言語** C
[JISX0201]
JIS X 0201 - 1997 7 **ビット及び** 6 **ビットの情報交換用符号化文字集合**
[ISO646]
ISO/IEC 646:1991 Information technology -- ISO 7-bit coded character set for information interchange
[ECMAScript]
Standard ECMA-262 ECMAScript Language Specification, 5th edition (December 2009),
http://www.ecma-international.org/publications/standards/Ecma-262.htm