

情報セキュリティ技術動向調査

タスクグループ報告書

(2008 年下期)

2008 年 3 月

IPA[®] 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 2008 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.....	1
1. セキュリティ機能の移行技術.....	2
2. DNSSEC 技術動向	9
3. SYSLOG 技術動向.....	17
4. セキュア OS の動向.....	21
5. テンポラリファイルの扱い..... エラー! ブックマークが定義されていません。	
6. コンピュータウイルスの動向.....	37
7. ダークネット観測の技術動向と観測事例.....	40
8. IPsec 関連技術の標準化動向.....	47
9. Resource PKI の動向.....	52
10. アイデンティティ管理技術の動向.....	56

序 2008 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 2 回目の会合を 2008 年 12 月 19 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。今回から、井上大介委員が加わり、事務局にもグレン・マンスフィールドが加わった。

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・解説していただいで本書を取りまとめた。

概況

今期も新たなセキュリティ関連の規格・仕様が規定されていると共に、仕様としては既存のセキュリティ関連技術を実際に実装・配備する際にも戦略とそれを支援する技術的手法が必要とされつつある。また、セキュリティ関連のイベントを認識できるようにするための機能についての規格・仕様の改良や観測技術の実運用も行われてきた。さらに各オペレーティングシステムへのアクセス制御機能の移植も進められている。

規格・仕様の策定が活発なのはアイデンティティ管理関連のようであり、特に OpenID 周辺の規格の策定は、活発に行われている。既存の規格・仕様をバージョンアップする際に、セキュリティの観点から下位互換性をあえて断ち切る規格・仕様が策定されたのは TLS (Transport Layer Security) 1.2 である。実際に配備されて普及するように促すために、あえて軽便な規格・仕様を策定してきたのは、BTNS (Better-Than-Nothing Security) である。

インターネットのインフラストラクチャを構成している基盤的な技術に関して、DNSSEC の配備や、Resource PKI の配備の話題が台頭しているが、戦略性と共に支援技術の準備が求められる。

インターネット上でダークネット観測を行っているプロジェクトは複数あり、特定の脆弱性を狙ったワームについてのトラフィックが顕著に観測されている。今期は USB メモリを介して感染を広げるコンピュータウイルスの増加も注目された。

いまだに多くのセキュリティ脆弱性の原因となっているテンポラリファイルの操作に関するプログラミングミスについて、詳しく解説している。

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

1. セキュリティ機能の移行技術

宮川 寧夫

1 情報セキュリティ技術分野における「移行」の話題

今日、情報通信技術の「移行」に関する話題で、世間でも話題にのぼるのは、いわゆる「地デジ」への完全移行の話題であろう。「2011年をもってアナログ放送は終了いたします」と、連日、各テレビはCMを流している。目を転じて、情報セキュリティ関連の技術分野にも、衆目を集めている「移行」の話題が2008年にはあった。

例えば、内閣官房情報セキュリティセンター（NISC）は、『政府機関の情報システムにおいて使用されている暗号アルゴリズム SHA-1 および RSA 1024 に係る移行指針』を策定している[1]。この背景には、これらの暗号技術については、コンピュータのCPUの計算能力の向上に伴って、安全性が相対的に低下しつつあることがある。

また、米国連邦政府は、DNSシステムに関してDNSSECに移行・配備に関する施策を実施しており、商務省（Department of Commerce）がパブリックコメントを募集した[2]。この背景には、昨今、注目されたDNSシステムに対する毒盛り攻撃（Poisoning）に対しては長期的・本格的な対策が必要であることが認識されつつあることに拠る。

暗号技術の移行問題にせよ、DNSシステムのDNSSECへの移行問題にせよ、現在、稼働しているネットワークシステムを保ちながら、同時に相互運用可能性を確保しなければならないが、これは容易ではない。また、情報セキュリティ機能についての相互運用可能性の確保に関しては、後述するように下位互換性の制約という特別な論点も存在する。

2 ソフトウェア機能の「移行」問題の位置づけ

一般に、セキュリティ機能を備えるソフトウェアといえども、セキュリティの観点のみならず、他の観点からも、その品質を確保することが求められる。そこで、ソフトウェアの品質についての代表的な規格であるISO/IEC 9126-1[3]を踏まえて、一般に、どのような観点からソフトウェアの品質が語られているかを掲げておく。

- 機能性（Functionality）
- 信頼性（Reliability）
- 使用性（Usability）
- 効率性（Efficiency）
- 保守性（Maintainability）
- 移植性（Portability）

「機能性」の確保が重要であることは言うまでもないが、「相互運用可能性」について、

この規格は「機能性」の副特性として扱っている。あるソフトウェアに実装されたセキュリティ機能を移行させることによって相互運用可能性に支障をきたすとしたら、そのソフトウェアの品質には問題があることになる。

「移行」に際しては、そのソフトウェアに「保守性」があることが期待される。あるいは、新しいセキュリティ機能と共に新しいソフトウェアの方に、従来の他の機能を「移植」する事例もある。

本稿ではソフトウェアのセキュリティ機能の「移行」に際して利用可能な技術を下記の項目のもとに列記することを試みる。

- 前提とする技術（保守性・移植性）
- 検証技術（機能性 / 相互運用可能性）
- 配備技術

3 前提とする技術（保守性・移植性）

(1) プロトコル仕様による暗号アルゴリズム ID のオブジェクト ID の登録等

ネットワークを介して情報通信を行う際には、その当事者同士が、誰と、どのようなプロトコルで、何を渡すかについて、通信の最初に交換して共有する必要がある。このように共有すべき一連の情報、定義、仕様等が「オブジェクト」と呼ばれる。このような文脈の「オブジェクト」について、国際的に一意に識別できるようにするために、全世界で固有な値が登録・管理されている[4]。これが、オブジェクトID（オブジェクト識別子：Object Identifier）と呼ばれており、様々な規格がオブジェクトIDとして指されるようになっている。このようなオブジェクトIDのリポジトリは、今日、インターネット上で参照することができる²。

オブジェクト ID は、ツリー構造の採番体系になっており、ルートの直下には、下記のように ISO と ITU-T の両標準化機関と共に、両機関の合同が分岐している。

- itu-t (0) : ITU-T
- iso (1) : ISO および ISO/IEC
- joint (2) : 合同

概ね、国コードのもとに企業コードが登録されるようなディレクトリ構造になっているといえる。[5] 具体的には、わが国においては、企業コード等の登録・管理は、次世代 EDI

² OID Repository

<http://www.oid-info.com/index.htm>

推進協議会 (JEDIC) においても行われている。³

暗号アルゴリズムを応用するセキュアプロトコルの仕様においても、暗号アルゴリズム ID はオブジェクト ID として登録される。このオブジェクト ID が、セキュアプロトコルが相互運用可能性を確保しつつ動作する背後で働いているのである。

ここで注意を要する暗号アルゴリズムが、いくつかある。例えば、RSASSA-PSS や、RSA-OEAP というアルゴリズムについては、オブジェクト ID のみならず、それらのパラメータも交換・共有されることを要するので、プロトコル仕様がそのために領域を規定していなければならない。

後述するように、アプリケーションプログラムを保守する者自身は、移行するセキュアプロトコルをプログラムに実装する必要はない。むしろ保守性の観点から、適切なライブラリを見極めて呼び出すことの方が適切であるといえる。

(2) セキュリティの観点からの交換可能性の確保と下位互換性の制約

IETF (The Internet Engineering Task Force) のセキュリティエリア内の各 WG が、近年、共通の課題として取り組んできたのが「ハッシュ・アジリティ (Hash Agility)」と呼ばれたハッシュ関数の取り替え可能性の確保の課題である。SHA-1 アルゴリズムの将来的な安全性を懸念して、よりセキュアなアルゴリズムに移行できるようにする前提として、インターネットプロトコル仕様の中ではアルゴリズムを交換可能なものとして規定すべきとされた。SHA-1 の利用を本文中に明記してしまっていた仕様については、改訂作業が進められた。例えば、RFC 2560 が規定していた OCSP (Online Certificate Status Protocol) 等が挙げられる。

セキュリティエリアからは、2008 年 8 月に TLS (Transport Layer Security) 1.2 の仕様が RFC 5246 として発行された[6]。この仕様が検討された過程では、上記の交換可能性の確保の観点を越えて、興味深い議論が行われてきた。TLS の仕様では、cipher suite と呼ばれる複数の暗号技術の組み合わせの集合が規定されており、ハンドシェイクの一環で利用される cipher suite が交渉される。TLS の WG においては「これらの cipher suite の集合を、よりセキュアなものの集合に移行しなければならない」という観点から、従前は是とされてきた「下位互換性の確保」を断ち切らなければならないとする議論が行われてきた[7]。この話題は、ヴァンクーバー会合 (2007 年 12 月) からフィラデルフィア会合 (2008 年 3 月) にかけて議論されてきた。このように、プロトコル仕様を下位互換性を断ち切ることは、次項で述べる相互運用可能性テストにおいて、考慮すべきテスト項目も変容させる。

複数世代のプロトコル仕様を実装しているアプリケーションが、下位互換性を制約する

³ 次世代 EDI 推進協議会 (JEDIC)

<http://jedic.ecom.jp/jedic/about.html>

事例もある。ブラウザのひとつ、マイクロソフト社の Internet Explorer 7 が 2006 年末にリリースされて普及してきたが、このデフォルト設定として、SSL 2.0 プロトコルが無効とされている[8]。SSL 2.0 仕様には、cipher suite の交渉におけるダウングレード攻撃が可能という脆弱性があり、これらを解消した SSL 3.0 仕様は、1996 年に発行された。このデフォルト設定の変更による SSL 2.0 プロトコルの無効化には 10 年を要したことになる。

(3) 特定のプロトコルバージョンに強く依存しないプログラミング

アプリケーションプログラムを保守してプロトコルを移行させるために修正したり、新規に新しいプロトコルに対応するアプリケーションを開発する者は、将来のプロトコルの移行をも考慮して、プロトコルに依存しないコーディングを心がける必要がある。例えば、プロトコル固有の構造体の要素を自らポインタで操作するようなコーディングをしてしまうと、将来、機能性 / 相互運用可能性を確保できなくなる懸念があると共に、保守性が損なわれてしまう。

保守性を確保するためには、品質の高いライブラリが提供されるようになるのを待って、それらに関数として呼び出すようにする必要がある。この際に、いくつか注意を要する点があるが、これらの注意事項はセキュアプロトコルを対象とする場合に限られない。

まず、ライブラリが提供される際には、同様の機能を実現するために、複数の関数が候補となる可能性がある。この際に、保守性の観点からは、特定のバージョンに固有の関数を避けて、複数世代のバージョンを通じて汎用的に利用できるものを探して呼び出すようにすることが望ましい。

また、既存のソースコードが、何らかの理由によって、ライブラリ関数を呼び出さずに自らポインタで構造体を操作するようになっている場合、その背景と機能をよく理解した上で、必要な機能を実現するように修正を加える必要がある。

4 検証技術（機能性 / 相互運用可能性）

(1) 相互運用可能性テスト

新しい通信プロトコル仕様を策定する過程や、その仕様が確定して発行されて他者が実装したとき、加えて仕様の細部を追加的に取り決めようとするときなどに、相互運用可能性テストが実施される。暗号技術を応用するセキュアプロトコルの場合、比較的多くの設定項目が規定されるので、テストケースにおける「場合分け」が膨大となることがある。

相互運用可能性テストにおいては、通常、ソフトウェアツールと、テストデータの集合、加えて一連の手続きを解説したドキュメントが利用されるが、先行してリファレンスとする実装を決める必要がある。そして、そのソフトウェアツールは、各プロトコルに固有のものとなり、汎用的なものは見聞きしない。プロトコル仕様のバージョンが上がれば、リ

ファレンス実装やソフトウェアツールも更新する必要がある。

複数の組織から各実装を持ち寄ることになるので、中立的な団体が構成されたり、運営にあたることが多い。

(2) システム移行テスト

「システム移行テスト」は、旧システムから新システムへの移行を行う際に、円滑な作業を行うことができるように実施されるテストである。2009年から再編される情報処理技術者試験の出題範囲の体系においては「サービスマネジメント」の項で扱われる基本的な論点でもある。

移行の際に必要な一連の作業を、あらかじめドキュメントとして、とりまとめておき、リハーサルとしてのテストを行って、その結果を評価することによって、一連の作業が実際に成功するようにドキュメントを修正して本番に備える。一般に、運用中のシステムについては、長時間停止させることは難しいことが、前提として考慮される。

5 配備技術

(1) IPv4/IPv6 デュアルスタック等

とかく v6 対応機器の新規配備が話題になり易いが、既存のシステム/ソフトウェアを活かして移行させる技術についても注目したいところである。

「IPv4/IPv6 デュアルスタック」(以下、デュアルスタック)とは、ひとつの機器に IPv4 と IPv6 の両仕様に基づくプロトコルスタックを共存させる仕組みをいう。この仕組みを利用することによって、状況に応じて IPv4 または IPv6 の利用を選択できるようになる。

IPv4 環境と IPv6 環境の共存を可能にする仕組みとしては、この他にもデュアルスタック化が困難な機器を対象とする、IPv4 と IPv6 相互のプロトコル変換を目的とした「IPv6/IPv4 トランスレータ」と、IPv6 のパケットを IPv4 でカプセル化して転送する「IPv6 over IPv4 トンネリング」が挙げられる[9]。

(2) DNSSEC 配備

DNSSEC プロトコルが動作する DNS の配備も、今後、広く進めていく必要があると言われている課題であるが、この際の技術的な課題は、まだ整理しきれていないように見受けられる。これに関しては、DNSSEC Deployment Initiative⁴ が、技術的な課題とロードマップを取りまとめている。ここでは、DNSSEC の配備に影響を与えている論点として、次の論点が識別されている。

- Root の問題

⁴ <http://www.dnssec-deployment.org/>

- 鍵の更新 (rollover)
- ゾーン横断 (Zone walking)
- 性能
- アルゴリズムの更新
- アプリケーションの問題

以上

参考文献

[1] 内閣官房情報セキュリティセンター (NISC), 「『 政府機関の情報システムにおいて使用されている暗号アルゴリズム SHA-1 および RSA 1024 に係る移行指針』 の策定」, 2008 年 2 月 4 日.

<http://www.nisc.go.jp/conference/seisaku/dai16/pdf/16siryou0301.pdf>

[2] DEPARTMENT OF COMMERCE, "Enhancing the Security and Stability of the Internet's Domain Name and Addressing System"

<http://edocket.access.gpo.gov/2008/E8-23974.htm>

[3] ISO/IEC 9126-1:2001 - Software engineering -- Product quality

注：この規格は、一連の 25000 シリーズ SQuaRE (Software product Quality Requirements and Evaluation) として再編される予定である。

[4] ISO/IEC 9834-1:2005 - Information technology - Open Systems Interconnection, Procedures for the Operation of OSI Registration Authorities: General Procedures and ASN.1 Object Identifier tree top arcs

注：この規格は、ITU-T Rec. X.660 (2004) と同等のものである。

[5] ISO/IEC 6523-1:1998, "Structure for the identification of organizations and organization parts -- Part 1: Identification of organization identification schemes"

[6] T. Dierks, E. Rescorla, RFC 5246, "The Transport Layer Security (TLS) Protocol Version 1.2", 2008 年 8 月.

<http://www.ietf.org/rfc/rfc5246.txt>

[7] "Removing cipher suites for 1.2", 2007 年 11 月 2 日 (金)

<http://www.imc.org/ietf-tls/mail-archive/msg07472.html>

[8] “Release Notes for Internet Explorer 7”,
<http://msdn.microsoft.com/en-us/ie/aa740486.aspx>

[9] 「IPv4/IPv6 デュアルスタックとは」, JPNIC (日本ネットワークインフォメーションセンター)
http://www.nic.ad.jp/ja/basics/terms/IPv4_IPv6-Dual-Stack.html

2. DNSSEC 技術動向

グレン・マンスフィールド，太田 耕平

1 背景

DNS の欠点は 1990 年に Steve Bellovin によって発見され、論文としてまとめられたが、彼は、その影響の大きさを考慮して発表していなかった。1995 年に、その論文[1] が発表されたことによって DNS がシステムへの侵入に悪用できることが明らかになり、より安全な DNS の研究が始まった。

最初の「攻撃の実演」は 1997 年に Eugene Kashpureff によってなされた。世界の主要なネームサーバのキャッシュを汚染することで、www.internic.net に対する問い合わせが www.alternic.net に対するアドレスとして返されることを示し、それが非常に容易であることが明らかにした。

その後、1997 年 1 月に RFC 2065 “Domain Name System Security Extensions”が発行され、続いて 1999 年 3 月には RFC 2535 が発行されることで DNSSEC の準備が整い、最も普及している主要な実装である BIND9 でも実装された。しかし、認証のための鍵配布をどのように実現するかについての問題は大きく、その利用は事実上不可能だった。

そのような中、特定の 2 者間で DNS の登録情報を更新する際の認証技術が提案された。RFC 2845 は、DNS サーバに対する更新をクライアントからの認証するための Transaction Signature (以下、TSIG)を定義し、これによって DNS サーバ間および通常の DNS 要求の認証も可能となった。この技術は共通鍵暗号技術とハッシュ関数の暗号技術に基づくセキュアな方法で、DNS 更新に対する要求と応答が許可されているエンドポイントを特定し認証する手段を提供している。しかし、この TSIG は DNS 全体の問題を解決する DNSSEC ではない。TSIG は DNS に参加するエンティティ間で個別の認証を提供しているにすぎず、世界中に分散している DNS 全体の問題を解決することができないため、普及もしていない。

本来、DNS は、あらゆる要求に対してその発信者に関係なく同じ応答を返すように設計されている。その意味で、すべての DNS データは誰からも参照できるものである。したがって、DNSSEC は、機密性やアクセス制御リスト、あるいは発信者によって要求を区別するような方法を提供するものではない。

その意味で、DNSSEC は SSL/TLS などの認証・暗号化技術とは異なるものであり、DNS が返す情報の正当性を確認するには、それが提供しているサーバの認証ではなく、そのドメイン名が親ドメインによって正しく発行されていることを確認する必要があるのである。このことは木構造をなしている DNS 全体を考慮しなければ DNS 情報の正確さは保証できないことを示している。

DNSSEC の RFC 発行後、4 年を経て、IETF は計画を見直し、現在の DNSSEC 標準を構成する 3 つの RFC を発行した。

- RFC 4033, “DNS Security Introduction and Requirements”
- RFC 4034, “Resource Records for DNS Security Extensions”
- RFC 4035, “Protocol Modifications for the DNS Security Extensions”

これらの文書は、様々な問題に断片的に対処したそれ以前の RFC を見直したものである。それらは以下の RFC となる。:

RFC 2535, RFC 3008, RFC 3090, RFC 3445, RFC 3655, RFC 3658, RFC 3755, RFC 3757, RFC 3845 および

RFC 1034, RFC 1035, RFC 2136, RFC 2181, RFC 2308, RFC 3225, RFC 3007, RFC 3597, RFC 3226.

2 DNSSEC プロトコルの概要

DNSSEC は、クライアントが下記の事項を検証することを可能としている。

- DNS 応答の完全性
- DNS 応答の正当性

これらの検証は、応答の内容が、該当するドメインやリソースレコードがない場合でも同様に必要であり、具体的には DNSKEY、RRSIG および NSEC レコードを用いて以下のように実行される。

DNS サーバが特定のレコードに対する要求を受けとったとき :

その応答は通常 RR セットデータを含んでいる。DNSSEC の場合は、その応答に RR セットに対応する RRSIG レコードが追加される。

クライアントは、得られた応答から RR セットデータを取り出し、RRSIG レコードの中で参照されているアルゴリズムを利用して RR セットデータのハッシュ値を算出する。

次に、クライアントは当該ゾーンに対する DNSKEY レコードを取得し、格納されている公開鍵を利用して RRSIG 中のハッシュ値を復号し、RRSIG のハッシュ値を得る。

上記の 2 つのハッシュ値が同じであれば、その RR セットデータの完全性を確認できる。正当性を検証するためには、DNSKEY の正当性を確認する必要がある。

DNSKEY そのものは、DNSSEC 応答の中で additional section に格納されている。この DNSKEY レコードの正当性は、0 節で述べたように、その親のゾーンの公開鍵による署名によって検証する。その鍵はさらにその親の鍵で署名される、というように DNS ツリーを遡って信頼点 (Trust Anchor) までたどることで検証が完了する。これによって DNSKEY の正当性を検証することが可能となり、この信頼点が DNS ツリー上の root で

あれば完全で最終的な検証となる。

要求に対応するデータがない場合は、サーバは RRSIG レコードに対応する NSEC 応答あるいは RR 応答を返す。

DNSSEC では AD フラグおよび CD フラグを利用する。AD フラグは、ネームサーバがリゾルバに対してコンテンツが認証されているかどうかを通知可能とする。これはリゾルバが DNSSEC を（直接的に）サポートしていないが、そのネームサーバおよび通信を信頼している場合に有用なものとなる。一方の CD フラグは、リゾルバがネームサーバに対して自身が DNSSEC 検証を実行可能であり、ネームサーバによる DNSSEC 検証が不要であることを通知可能とする。

DNSSEC ではさらに DO フラグを利用する。DO (DNSSEC OK) は EDNS0 と呼ばれる拡張 DNS で用意されたフラグを利用しており、DNS 応答に DNSSEC に対応した RR セットデータを含ませるように要求する際に利用される。また、メッセージに複数の RRSIG が含まれることがある DNSSEC では、512 バイトに制限された従来の UDP ベースの DNS メッセージは小さすぎる。この問題は同じく EDNS0 で導入された最大メッセージサイズを 4096 バイトとする拡張を利用することで対応できる。

3 DNSSEC の技術課題

DNSSEC によって DNS の応答の安全性を確保できるが、DNSSEC の導入にともなう技術およびセキュリティ上の課題がある[2]。

- DNS の応答メッセージのサイズの増大
前述の EDNS0 で従来の 512 バイトから 4096 バイトにまで拡張されている。さらに UDP メッセージのサイズを超えた場合は、TCP にフォールバックする。
- トランザクション数の増加
ゾーンの公開鍵を探すためのトランザクションが増加する。またゾーンファイルのサイズが数倍になる
- 必要な処理能力の増大
電子署名、およびその検証、鍵管理、鍵更新など、処理量が多くなり、クライアント、サーバ双方とも必要な処理能力が大幅に増大する
- 設定と運用が難しくなる
設定ミスや鍵の期限切れなどの問題がおこりやすくなる
- 小さな要求が大きな応答を発生させ得る
DoS の増幅器となりえる
- 鍵更新 (Key rollover)
特に信頼点を root ゾーンとできるかどうかは運用面での現実的な課題となる

- 絶対時間についての仕様
時刻同期を必須とすることになる
- NSEC レコード
ゾーンファイルに内容と構造の記述が必要で、NSEC3 レコードは依然検討中

普及についても、すでに広く普及した大規模なシステムへの配備となるため、大きな抵抗にあっていいる。DNSSEC が対応しようとしている問題は、通常知られている脆弱性や攻撃などとは性質が異なるため、その価値の説明と理解は容易ではない。また近年まで DNS による攻撃は一般的ではないとされてきた。

さらに、部分的な配備が、配備を難しくしてしまうことも問題である。部分的な配備によって、root 以外の信頼点が発生し、DNS がシステムとして分断されてしまい、2 節で述べた検証の過程もそれぞれの信頼点で分断されてしまうため、クライアントはそれらすべての信頼点の情報を保持する必要がでてくるため、信頼点の管理という新しい問題が発生しています。

また、現実的な課題のひとつとしてアプリケーション側の対応の問題がある。DNSSEC での検証が失敗したときのアプリケーションの適切な振舞いは明確になっていない。例えば Windows 7 の DNSSEC では、DNS クライアントはクライアント自身の DNSSEC の検証に対応していない。クライアントは検証のために自身が利用している DNS サーバを信頼することになる。これは前述の AD フラグを利用した実装例となる。このことのメリットのひとつは、信頼ポイント自体の設定をクライアントが実行する必要はなく、それゆえ多くの手間が省けることであるが、これが適切な方法かどうかはあまり議論されていない。

次に上記問題の中でももっとも困難で、本質的な問題のひとつである鍵更新の問題について述べる。

3.1 鍵更新問題 (Key rollover)

署名に用いられる鍵は定期的に更新される必要がある。DNS のような分散システムではこれは容易ではない。特に信頼点が島状にちらばっている場合は、問題はさらに複雑になる。

事故などが無く安全に運用されてきた鍵については自動更新のシナリオもシンプルで、各鍵に利用期限を設定した上で、現在の鍵 K_n で、更新する新しい鍵 K_{n+1} に署名する。しかし、もし漏洩などの事故によって K_n の変更が必要となった場合は、もはや鍵 K_n が信頼できないため、その方法は使えない。そのため DNSSEC は独立した 2 つの鍵を使う。

- (1) DNSKEY レコードで発行され、ゾーンに署名する運用鍵 ZSK (Zone Signing Key)
- (2) 運用鍵 KSK に署名する保証鍵 KSK (Key Signing Key)

ZSK は、通常のセキュリティレベルで運用される。そして ZSK に事故などがあった場合に利用することになるバックアップ用の鍵が KSK となる。

ここで注意すべきなのは ZSK の運用と KSK の重要性である。原理的にすべての DNS 応答に署名することになるため、ZSK は非常に多くのメッセージに対して署名する。このことは鍵が特定されるリスクが高いことを意味しており、ZSK の更新頻度は慎重に検討される必要がある。さらに、そのようなリスクをもつ ZSK に対して署名する KSK の役割は単なるバックアップを超えて重く、KSK の運用は DNS とは独立に慎重に扱うことが重要となる。しかし、上記のポイントは十分に理解されているとはいえないのが現状である。

3.2 DNSSEC の運用に関する意識調査

ICANN の ccNSO は、2007 年 6 月現在の調査に DNSSEC に対する意識調査を実施した [3]。(国レベルおよび世界的なドメインの DNS を運用している ccNSO/wwTLD のメンバーから 61%の回答を得ている)

- | | |
|--------------------------|-----|
| • DNSSEC について知っている | 90% |
| • 運用している | 7% |
| • 試験運用している | 5% |
| • 運用を計画している (技術、情報の成熟待ち) | 85% |
| • 1 年以内の運用を計画している | 30% |
| • 具体的な計画はない | 30% |
| • root ゾーンへの署名を希望 | 84% |
| • ICANN による署名を希望 | 65% |
- (残りのうち、2%は ICANN 以外の NGO, 2%は政府機関による署名を希望)

DNS 運用の現場では、DNSSEC に対する関心、期待は大きいですが、現実的な運用面での課題がその普及を妨げていることがわかる。

3.3 DNSSEC の普及状況

DNSSEC は期待されたほど普及してはいないが、スウェーデンを中心として、徐々に利用が進んでいる [4]。

- 2005 年 9 月: .se (スウェーデンのルート) が ccTLD として初の署名
- 2006 年 4 月: .ru (ロシア) が署名
- 2006 年 8 月: .pr (プエルトリコ) が署名
- 2007 年 1 月: .bg (ブルガリア) が署名
- 2007 年 6 月: .br (ブラジル) が署名
- 2008 年 7 月: .org が最初の gTLD として署名

- 2008年8月：.govが2009年1月に署名することを発表
- 2008年9月：.cz（チェコ）が署名

国内では、JPRS（Japan Registry Services）が配備について取り組んでおり、2007年10月のICANN会議のDNSSECワークショップで試験運用とその結果について発表している[5]。

2008年11月28日、スウェーデンのトップレベルドメインを管理する.SE（The Internet Infrastructure Foundation）は1,000番目のDNSSEC .seドメイン（<http://www.sormlandsmusikoteater.se/>）を登録した[6]。

Kaminsky問題が明らかになって以来、DNSSECへの関心は大きくなっている。例えば、郵便電気通信庁や、選挙運営行政などの複数のスウェーデンの公的機関はDNSSECドメインとして署名することを選択しており、他も追随している。また、配備を促進するために.SEは2009年1月からDNSSECのための特別課金を廃止している。

2009年1月23日現在、.SEが管理しているDNSSECドメインはさらに増加し、1,563が署名されている。しかし、.SE全体では840,905のドメインが登録されており、最も熱心なスウェーデンでも普及の進行は遅々として進んでいないことがわかる。

また、監視サイトSecSpider⁵によると、2008年12月29日現在、全世界ではKSKとZSKの両方を配備したDNSSECゾーンは1952運用されており、そのうち1192は.SE配下となっている。世界的にはさらに遅れている。様々な情勢や圧力にも係らず、明らかに配備はうまくいっていないが、2009年1月末には上記のDNSSECゾーンの数は4,071まで急増しており関心は高まっている。

4 配備が遅れている理由

その重要性にも係らず、実際の普及が遅れていることの背景にはDNSSECを利用しないことによるリスクと利用したときの利益が明確ではないことが挙げられる。

一般的なabc.comやdef.netを運用しているドメインは、安全でないDNSのリスクやDNSSECによる利益を理解していないことが多い。abc.comドメインがハイジャックされると、まずは利用者が影響を受ける。しかしabc.comの所有者にとっては、オンラインショッピングサイトのようにその事業がそのユーザからのアクセスに依存する事業を運営していなければ、そのダメージはそれほど深刻ではないかもしれない。

一方で、abc.comの利用者は確かに影響を受けるかもしれないが、利用者自身によってできることは多くない。利用者はその安全でないサイトを避けるかもしれないが、実際のところ取りえる選択肢がなく、あるいはそのことに気づきもしないことも多い。

DNSSEC配備問題は、そのセキュリティ上の問題点が、エンドユーザにとっては、深刻

⁵ SecSpider the DNSSEC Monitoring Project,
<http://secspider.cs.ucla.edu/islands.html>

にみえないことが多いことが危機感の欠如につながり、結果として積極的に推し進める立場の人が不在である（ように見える）ことである。

4.1 商業的側面

DNSEC が、普及しない技術であるとか、あるいは普及するまでは意味がない、といった議論はある。そのため、それが配備されるまで、少なくともバックアップとして、各サイトはクライアントに対してその正当性を示すためのサイト証明書を利用しなければならない。

このことは証明書発行機関（CA）にとってはいいビジネスとなっているため、信頼点を提供することのできる CA が、DNSSEC に積極的になる理由が希薄になってしまう。最初のケースであるスウェーデンでも、バックアップとして、SSL 証明書を認証の手段とした。SSL 証明書は、CA によって直接的な収入の手段となるが、.org のようなトップレベルドメイン（TLD）での DNSSEC の場合には直接的な収入にならないことも DNSSEC が進まない理由の一つといえる。

しかし、エンドユーザにとっては、どの CA を信頼すべきかについて悩まされるとともに、多くの root 証明書を管理する必要性にせまられており、この傾向が続くと時期にはそれらを管理するのが非常に面倒という状況になる可能性がある。

また、多くのサイトでは証明書の購入について財政的、経営的負担をしたくないと考えることもありえる。

さらなる要因もある。DNSSEC はサイトそのものの認証ではない。DNSSEC は DNS 名の所有者としてのサイトを認証するのみである。つまり、サイト証明書は、登記簿などによって存在を保障する社会的に広く認知された法的手段があり、CA はそれらの情報を元に証明書を発行できるが、ドメイン名の場合は、その DNS 名を登録・確認できるのは親ドメインのみである。

つまり、CA がサイト証明書と同じようにドメイン名を認証するためには、認証したドメインの親ドメインのゾーン管理者にその存在を確認する必要がある。そしてその親ゾーンの存在を確認するためには、さらにその親と次々に確認をとる必要がある。そうやって確認作業をしてはじめてドメイン名を認証することができる。しかしこれは DNSSEC がやろうとしていることそのものであり、これを実現することが DNSSEC 配備問題の要点であり、ほとんどの部分でもある。

結局のところ、ルートゾーンは、まだ署名されておらず、そのため DNSSEC は例外的な処理なしには機能していない。そして CA による証明書に基づいたエンドポイントの認証も（同じ理由で）機能しない。

4.2 政治的関心

ルートゾーンが署名されるとしても、誰が署名するのか、が問題となる。それを実行す

るものには、巨大な権威と責任が伴う。これは政治的価値と商業的価値につながる。そのため誰が署名するのか、に対する回答にたどり着くのは自然と困難になる。これは別なもめごとにもなる。理想的にはなんらかの NPO がその責を果たすべきだが、現時点ではそれは検討中のひとつの提案にすぎない。多くは A と J の root サーバを運用する Verisign のような営利企業ではなく、ICANN のような非営利組織を希望している。

5 参考文献

[1] Steven M. Bellovin. “Using the Domain Name System for System Break-ins”, Proceedings of the Fifth Usenix UNIX Security Symposium, June 1995, Salt Lake City, UT,
<ftp://ftp.research.att.com/dist/smb/dnshack.ps>

[2] DNSSEC - The Theory, Geoff Huston, The ISP column,
<http://www.potaroo.net/papers/isoc/2006-08/dnssec.html>

DNSSEC - The Practice, Geoff Huston, The ISP column,
<http://www.potaroo.net/papers/isoc/2006-09/dnssec2.html>

DNSSEC - The Opinion, Geoff Huston, The ISP column,
<http://www.potaroo.net/papers/isoc/2006-10/dnssec3.html>

[3] DNSEC Survey Results,
<http://ccnso.icann.org/surveys/dnssec-survey-report-2007.pdf>

[4] DNSSEC News, DNSSEC Mailinglists, DNSSEC News,
<http://www.dnssec.net/news>

[5] Shinta Sato, Kentaro Mori, A Brief Introduction to JPRS' DNSSEC Implementation Research for TLD, DNSSEC Workshop - ICANN Los Angeles Oct. 29, 2007,
<http://losangeles2007.icann.org/files/losangeles/SatoICANN-LA-DNSSEC-shinta-v1.0.pdf>

[6] .SE Pressreleases, 1, 000 DNSSEC domains signed,
<http://www.iis.se/about/press?id=173>

3. SYSLOG 技術動向

太田耕平

1 はじめに

本稿では、古くから、そして現在も TCP/IP 環境でのログ収集のために広く利用されている SYSLOG プロトコルの標準化および技術動向について述べる。

2 SYSLOG 標準の現状

SYSLOG はインターネット初期からのロギングのためのデファクト標準となっており、UDP をトランスポートとし、そのシンプルさ、実装と運用の容易さから、純粋な記録以外にも、ウイルス・侵入検知などのセキュリティアラートなどの目的にも広く利用されている。

しかし、実際のところそのプロトコルは十分には文書化されておらず、現在のよりどころは INFORMATIONAL として発行されている RFC 3164, “The BSD syslog Protocol”[1] だけである。これはトランスポートに UDP を用いており、信頼性を保障せず、認証、暗号化などの一般的なセキュリティ対策もないものとなっており、すなわち、

- 暗号化がないことでシステム情報の盗聴は極めて容易であり、
- 認証がないことで、偽のログメッセージを簡単に挿入でき、
- 信頼性を保障していないため、重要なログの欠落を知ることも防ぐこともできない

このことは、現実問題として、現状では今日のセキュリティ状況に見合わない脆弱なロギングシステムが一般的ロギングシステムとして利用されていることを示している。

この状況を改善するため、IETF の SYSLOG WG では、RFC 3195 “Reliable Delivery for syslog”[2] を策定し、TLS と BEEP による安全な SYSLOG の標準化を目指した。しかし、この標準案は包括的なセキュリティ機能と引き換えに SYSLOG の美点であった簡便さを大きく損ない、従来の SYSLOG からのギャップも大きく多くの利用者・開発者に敬遠される結果となった。

そのため、RFC 3195 自体は、大幅に簡素化し draft-lear-ietf-syslog-rfc3195bis-01 の individual-draft として再提案することとなり、SYSLOG WG としては、要件をさらに精査することで必須部分を抽出し、それぞれの部分を以下のように分割して標準化を目指している。

- メッセージフォーマットを策定する本体部分
The syslog Protocol, RFC5424 として 2009 年 3 月に発行
- 従来の UDP による SYSLOG に相当する UDP トランスポート部分

- Transmission of syslog messages over UDP, RFC5426 として 2009 年 3 月に発行
- RFC3195 の必須部分を抽出し、TCP による信頼性を前提とした TLS トランスポート部分
 TLS Transport Mapping for Syslog, RFC5425 として 2009 年 3 月に発行
- 複数の Collector に SYSLOG メッセージを振り分ける際の署名部分
 Signed syslog Messages, draft-ietf-syslog-sign-24.txt

さらに、前述のように、SYSLOG はその重要性にもかかわらず、システムの稼働状況に対する管理が行えないことから、管理性についての標準化も進められており、最初のステップとして Textual Convention MIB (Textual Conventions for Syslog Management, <draft-ietf-syslog-tc-mib-08.txt>) の標準化が議論され RFC5427 として 2009 年 3 月に発行され、他の管理標準案からの参照と利用も始まっている。しかし、本体部分については議論が遅れており、新たに整備されつつある新しい SYSLOG 体系の中での大きな穴となっている。

図 1 に、2009 年 1 月現在の SYSLOG 関連プロトコルの標準化状況を示す。基本的なプロトコルのほとんどが IESG によるレビューを完了し、RFC 番号の割り当てを完了しており、2009 年 3 月より RFC として順次発行されている。

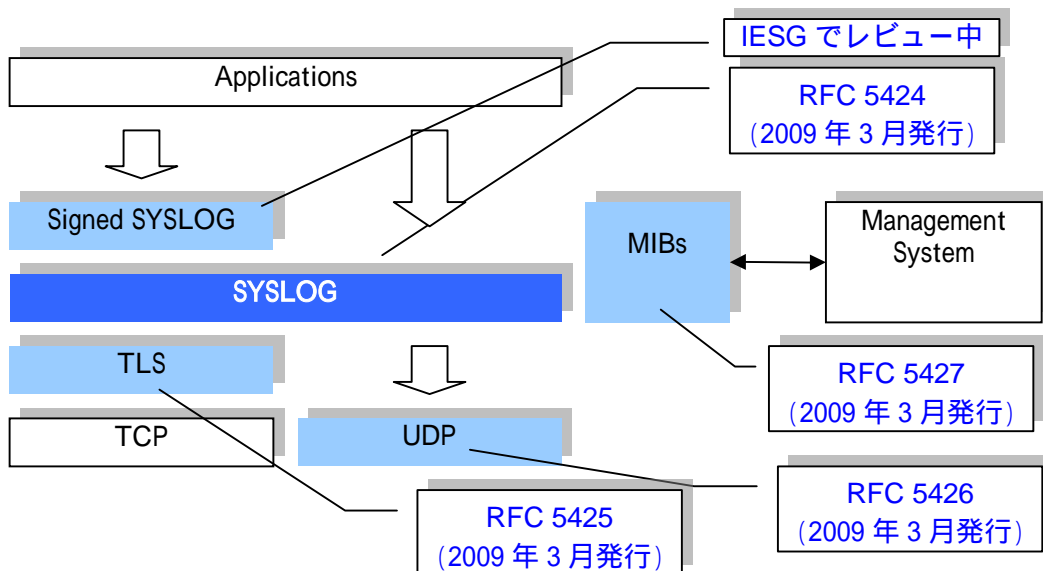


図 1 SYSLOG 関連プロトコルの標準化状況

3 SYSLOG 実装の現状

SYSLOG は、その重要性に対して、運用面での現実的な問題があまり考慮されてこなかったが、従来の技術的な運用管理の用途のみならず、監査などの用途の需要が高まるにつ

れ、その信頼性、完全性が大きな問題となっており、現在利用されている実装についても重要な調査がなされている。

Marcus Ranum らは、USENIX2005 および SANS (SysAdmin, Audit, Network, Security) のロギングについての包括的なチュートリアルで SYSLOG 運用の問題点についても RFC 3164 の内容と対比して指摘している。当時の一般的な構成で、現在標準的に使われている UDP を利用したネットワーク経由でのログ収集では、場合によっては 99% 以上のメッセージが失われることがありえることを指摘しており、一つの巨大なログサーバよりも、集約用の小型のログサーバを分散配備して、そこから中央のサーバに暗号化の上、信頼性のあるトランスポートで送信するようにすることが望ましいことを指摘している[3]。

また現在の一般的な環境で広く利用されている複数の実装でも、何らかの理由でメッセージ落ちが発生しても送信側、受信が双方でメッセージの欠落を知ることができない。最も伝統的な syslogd では日常的なメッセージ落ちの発生が報告されている[4][5]。メッセージ落ちの多くは秒間 1000 個以上の SYSLOG メッセージの発生時であり、その際は Severity などの重要性を考慮することなく発生順に破棄される。このようなログはディスク溢れやデーモンの設定ミスなどの障害時などにも起こりうる。

また、最新の実装として知られている syslog-ng や rsyslog では基本性能が大きく改善されるとともに TCP によるリモートロギングのサポートなどにより信頼性が大きく向上しているが、ログの欠落が発生するケースが依然として残っている。具体的には障害などによってネットワークが長時間不通になり、サーバとの TCP コネクション自体が切断されてしまうようなケースに対する対応は十分ではない。例えば、Linux の標準設定ではネットワークの不通状態が約 15 分を過ぎると TCP コネクションが切断されてしまうため、その期間に発生したメッセージはコネクションと共に失われてしまう。同様の問題は SYSLOG プロトコル本体の著者である Rainer Gerhards によっても指摘されており[6]、アプリケーションレベルでの信頼性を確保するために RELP (Reliable Event Logging Protocol)を提案し、そのライブラリ実装を公開している[7]。

重要な点は、ログ情報はあらゆる監査や障害に対応する際の最後の砦ともいえる役割を担っていることである。そのため、様々な手段・レベルで信頼性を確保することが重要である、さらに重要なのは、管理者がログ情報を信頼するために、ログ収集システムそのものの信頼性を知ることができることである。つまり信頼性の確保をどこまでどのようにやるかの問題の基礎として、ログ収集システムの運用状況を管理するための機能が不可欠であるといえる。

4 まとめ

本稿では、広く普及し、重要な役割を担っている SYSLOG プロトコルの現状について、標準化および技術の動向をまとめた。

長く議論されてきたその仕様は、2009年1月現在、複数の要素に分割され、その基本要素の策定がほぼ完了している。

セキュリティと信頼性を考慮した新しい SYSLOG 標準により、現在大きく欠けているセキュリティ上のギャップをうめ、システムの基盤として欠かすことのできないロギングシステムに他のシステムと同じ水準のセキュリティを期待することができるようになる。

一方で、トランスポートより上位の信頼性、および大規模システムでの適切な配備と管理の方法など、運用・管理面での標準化はまだ途上であり、残された課題となっている。

以上

参考文献

- [1] RFC 3164, "The BSD syslog Protocol, August 2001, C. Lonvick"
- [2] RFC 3195, "Reliable Delivery for syslog, November 2001, D. New, M. Rose"
- [3] Marcus Ranum, "System Logging and Log Analysis",
http://www.ranum.com/security/computer_security/archives/logging-notes.pdf
- [4] 細内 昌明, 山崎 謙太, 森田 滋, 中島 藤夫, 大辻 彰, 三瓶 英智, 「ミッションクリティカル向け syslog 改善機能の提案と実装」, Linux Conference 2008.
- [5] 角田 裕, 丸山貴史, 阿部 智, 太田耕平, 和泉勇治, キニ グレン マンスフィールド, 根元義章, 「ログの重要度に基づいた適応型送信制御による効率的なログ転送方式の提案」, 電子情報通信学会技術研究報告 Vol.107, No.530, CS2007-78, pp.27-32, Mar. 2008.
- [6] Rainer Gerhards, "On the (un)reliability of plain tcp syslog...",
<http://blog.gerhards.net/2008/04/on-unreliability-of-plain-tcp-syslog.html>
- [7] librelp – a reliable logging library,
<http://www.librelp.com/>

4. セキュア OS の動向

面 和 毅

1 OpenSolaris の新セキュリティ機構(FMAC)

1.1 FMAC(Flexible Mandatory Access Control)

FMAC(Flexible Mandatory Access Control)は、Flask アーキテクチャと Type Enforcement モデルを OpenSolaris 上に移植したものになる。

通常の Solaris では"Trusted Extensions"と呼ばれるパッケージを導入する事により、セキュリティの強化をはかることができる。今回 OpenSolaris 上で FMAC が採用される事により、OpenSolaris 上でも現行の Solaris 上のセキュリティと同等のレベルの物が実現できるようになる。

1.2 Type Enforcement モデルの概要

OS 上でのセキュリティモデルを考える際には、以下の 4 つの要素を定義する。

- アクセスの主体となる「サブジェクト(通常はプロセスなど)」
- アクセス対象となるシステムリソース「オブジェクト(ファイルやソケットなど)」
- システムリソースの操作となる「アクセス(読み込みや書き込み、生成など)」
- アクセスが許可されるか否かを判定する「リファレンスモニタ」

これらを用いて「あるサブジェクトがオブジェクトに対してどういうアクセスが「許可」されるか」という形でアクセス制御が行われる。この際に、サブジェクト/オブジェクトとは何か、アクセスの許諾を判定する際に使用される要素は何か、などの考慮が必要になる。そこで、これらを決定する指針となる「セキュリティポリシーモデル」が必要になる。

セキュリティポリシーモデルに関しては様々なものがあるが、Type Enforcement モデルでは

- サブジェクトには「ドメイン」を付与する
- 全てのシステムリソース(プロセスを含む)に「タイプ」を付与する
- ドメインとタイプの関係(書き込み、読み込みなど)をあらかじめ定義しておき、その関係が記載されている物だけ「許可」となる。明示的に関係を定義されていない物は全て「拒否」となる。

というルールに従う。

この Type Enforcement モデルを Linux 上で実装した物が、「SELinux」と呼ばれる実

装で、現在は Red Hat Linux などに標準で搭載されているものになる。

1.3 Flask アーキテクチャの概要

また、SELinux では、様々なセキュリティポリシーをサポートすることを目標に、その設計に「Flask」と呼ばれるアーキテクチャを採用している。この Flask アーキテクチャは

- セキュリティポリシーに基づいて判断を下すロジック
- セキュリティポリシーの判断結果を強制的に実施するメカニズム(オブジェクトマネージャ)

から構成されており、セキュリティポリシーを変更するとオブジェクトマネージャの働きを変更する事が出来る。

Type Enforcement モデル及び Flask アーキテクチャに関しては、「アクセス制御に関するセキュリティポリシーモデルの調査 報告書」⁶に詳しく記載されている。

1.4 プロジェクトスケジュール

OpenSolaris 上での実装「FMAC」は、Linux での実装の「SELinux」が使いづらいとユーザ側から認識されてしまっている事を教訓とし、ユーザビリティを向上したシステムを目指している。

FMAC のプロジェクト自体は、2008 年 2 月 14 日からスタートしており、概ね下記のスケジュールに従ってプロジェクトが動いている。

- Project の提案 02/14/2008
- Project のサイト作成 03/04/2008
- プレスリリース 03/13/2008
- Alpha 1 ソースコード 05/02/2008
- システムコールサポート 06/20/2008
- プロセスコンテキストサポート 07/10/2008
- Alpha 2 ソースコード 09/02/2008
- Alpha 3 ソースコード 10/30/2008

1.5 プロジェクトステータス

2008 年 10 月時点で、下記のユーティリティ / インターフェースが移植されている。

⁶ 「アクセス制御に関するセキュリティポリシーモデルの調査」, 2005 年 3 月
http://www.ipa.go.jp/security/fy16/reports/access_control/policy_model.html

ユーティリティ

- checkpolicy
- loadpolicy
- getenforce
- setenforce
- setfiles
- Pcon
- getfilecon
- setfilecon
- newrole
- fmacsetup

インターフェース

- security_load_policy()
- security_compute_av()
- security_check_context()
- security_getenforce(),
- security_setenforce()
- is_fmac_enabled()
- getcon(), getpidcon()
- getexeccon(), setexeccon()
- getprevcon()
- freecon()

また、ZFS/TEMPFS に対してのファイルセキュリティラベルとアクセス制御が出来るようになっている。

このように、FMAC on OpenSolaris プロジェクトは着々と開発が進んでいる。この FMAC が正式版になった暁には、SELinux の経験のあるエンジニアであればポリシ記述方法等はほとんど代わらないため、比較的導入しやすいセキュリティオプションとなるであろう。

2 SELinux の動向

2.1 Permissive Domain

これまで SELinux ではシステム全体で Enforcing/Permissive を切替えるようになって

いた。そのため、特定のサービス用に SELinux のポリシーを追加した際でも、システム全体のモードを変更するため、膨大なログが出力され、かつデバッグに手間が掛かっていた。

しかし新しい SELinux では、特定のドメインのみを Permissive にする事が出来るようになった。これにより、特定のドメイン用のポリシーを作成した際に、そのドメインのみを Permissive モードにする事が出来るため、ポリシーのデバッグが簡単に行えるようになっている。

この Permissive ドメインは Fedora9 から使用可能となっており、

```
# semanage permissive -a "ドメイン名"
```

でドメイン単位で Permissive モードにする事が出来る。

2.2 組み込みシステムへの取り組み

SELinux を用いたシステムはこれまで一般的な OS のセキュリティ強化として注目されてきた。また、他のいわゆるセキュア OS 製品に関しても、DMZ 等のセキュリティを確保すべきシステムの強化、また軍事関係などの特殊なセキュリティが要求される場所で使用されてきた。

しかし、今年になって SELinux やセキュア OS を用いて組み込み系システムを保護しようという動きが出て来ている。Linux だけに関しても、オタワで行われた LinuxWorld では

- SELinux

SELinux for Consumer Electronics Devices

(日立ソフト 中村氏)

- Smack

Smack in Embedded Computing

(Casey Schaufler 氏)

といったように、セキュア OS で保護する対象を組み込み系まで広げて来ている。組み込み系では一般的な OS と違い、元々特殊なインターフェースを使用しているため、セキュア OS のポリシーに起因する使い勝手の変更等が意識されにくいことから、比較的受け入れられやすいと思われる。

以上

5. テンポラリファイルの扱い

田中 哲

2008 年下期には、Perl の File::Path モジュールの `rmtree` 関数に関する CVE が 3 件発表された。(CVE-2008-2827, CVE-2008-5302, CVE-2008-5303) また、`symlink attack` に関する CVE は 100 件以上出ている。

テンポラリファイルの扱いに関する問題は古くからあるが、いまだに多くの問題が発生する。そこで本稿ではテンポラリファイルの扱いかたについて解説する。また、安全な削除に利用できる新しいシステムコールが提案されているので、それについても触れる。

1 テンポラリファイル

テンポラリファイルはプログラムが一時的に利用するファイルである。

Unix においては `/tmp` や `/var/tmp` というディレクトリが提供されており、すべてのユーザがそのディレクトリ下にテンポラリファイルを生成・削除するのが慣習である。本稿では、これらのディレクトリの代表として `/tmp` を例として用いる。

この慣習では、`/tmp` を複数ユーザで共有する。そして、マルチユーザシステムにおいては他のユーザは必ずしも信用できない。そのため、テンポラリディレクトリ内でのファイル操作には他のユーザが任意のタイミングで介入する可能性があり、通常のファイル操作以上に注意が必要である。不適切に操作を行うと、意図しないファイルの削除や書き換えといったセキュリティ問題を引き起こすことがある。

また、本稿では `/tmp` 直下にある他ユーザのファイルは `unlink`, `rmdir`, `rename` できないものと仮定する。これは `sticky bit` (`S_ISVTX` フラグ) の効果であるが、これについては後で述べる。

さらに、テンポラリディレクトリはローカルなファイルシステムであることを仮定する。つまり NFS 等のリモートファイルシステムについては考慮しない。

2 テンポラリファイルの作成

ディレクトリでない、通常のファイルを `/tmp` 直下に生成する場合、`open` システムコールにおいて、第 2 引数 `flags` に `O_EXCL` フラグを指定しなければならない。`O_EXCL` は、ファイルが既に存在したら失敗するという指定である。また、第 3 引数 `mode` には `0600` など、他ユーザはアクセスできないパーミッションを指定する。

たとえば、読み書き両用で `foo` という名前のテンポラリファイルの生成に挑戦する場合、以下のようなになる。

```
int fd;
fd = open("/tmp/foo", O_RDWR | O_CREAT | O_EXCL, 0600)
if (fd == -1) {
    失敗に対応する
}
```

もし /tmp/foo が存在した場合、O_EXCL の効果によってこの open は失敗する。これにより、例えば他ユーザが /tmp/foo から重要なファイル(/etc/passwd や ~/.ssh/authorized_keys など) へのシンボリックリンクを作るといった攻撃を行っても、重要なファイルを意図せず書き換えてしまうことを防ぐことができる。

また、パーミッションを 0600 とすることにより、umask の設定に依存せず、他ユーザがファイルを書き換えてしまうことを防げる。

テンポラリファイルを生成するという目的のためには、ファイルが存在して失敗したときには、ファイル名を変えて再挑戦する必要がある。言語や環境によってはこの再挑戦まで含めて安全に行う機能がライブラリとして用意されており、可能ならそれらを使ったほうがよい。そのようなライブラリにはたとえば以下がある。

- C 言語の mkstemp (Single Unix Specification)
- C 言語の tmpfile (ISO C)
- シェルスクリプト用の mktemp コマンド (OpenBSD)
- Perl の File::Temp ライブラリ
- Python の tempfile ライブラリ
- Ruby の tempfile ライブラリ
- PHP の tmpfile

なお、歴史的な経緯から、ファイル名を生成するだけの関数がライブラリに用意されていることがある。ファイル名を生成するだけの関数は、間違えた使いかたをしやすいため、基本的には避けるべきである。そのようなものにはたとえば以下がある。

- C 言語の mktemp (Single Unix Specification)
- C 言語の tempnam (Single Unix Specification)
- C 言語の tmpnam (ISO C)

3 テンポラリファイルの削除

/tmp 直下にテンポラリファイルが安全に作成された場合、削除は単に unlink システムコールを使えばよい。

たとえばテンポラリファイルの名前が `/tmp/foo` であれば、以下のように行う。

```
if (unlink("/tmp/foo") == -1) {  
    失敗に対応する  
}
```

4 テンポラリディレクトリの作成

テンポラリディレクトリを安全に作成するには `mkdir` システムコールを使用する。その際、第2引数 `mode` には `0700` など、他ユーザはアクセスできないパーミッションを指定する。

```
if (mkdir("/tmp/foo", 0700) == -1) {  
    失敗に対応する  
}
```

`open` とは異なり、`mkdir` にはフラグ引数がない。`mkdir` は指定されたファイルが既に存在したときには常に失敗する。なお、指す先が存在しないシンボリックリンクが存在した場合も同様に失敗する。したがって、`mkdir` が成功すれば、テンポラリディレクトリの作成に成功したことが保証できる。

また、パーミッションを `0700` とすることにより、`umask` の設定に依存せず、他ユーザがディレクトリを書き換えてしまうことを防げる。

5 テンポラリディレクトリの削除

自分で作ったテンポラリディレクトリを削除する場合には、`rmdir` システムコールを使用すればよい。例えば以下のように行う。

```
if (rmdir("/tmp/foo") == -1) {  
    失敗に対応する  
}
```

ただし、`rmdir` は空のディレクトリしか削除できないため、`rmdir` に先立って内部のファイルをすべて削除しておかなければならない。サブディレクトリがあるのであれば、その内部のファイルを事前に削除しなければならず、一般には再帰的な処理が必要である。

ここで以上の操作が安全なのは、信用できない他ユーザは書き込めないディレクトリを削除する場合である。そうではない場合、すなわち、信用できない他ユーザが書き込めるディレクトリを削除する場合には必ずしも安全ではない。これについて次に述べる。

6 他ユーザが書き込めるディレクトリの削除

他ユーザが書き込めるディレクトリを削除するにはかなりの注意が必要である。冒頭に触れた Perl の rmtree の CVE もこの状況を扱う場合の話である。

他ユーザが書き込めるディレクトリを扱う状況はたとえば以下が考えられる。

- テンポラリディレクトリを作成するときに、mkdir に指定するパーミッションとしてたとえば 0777 を指定してしまうと、umask によっては誰でも書き込めるディレクトリができてしまう。
- 複数ユーザでディレクトリを共有してファイルを管理する状況。なお、この想定はテンポラリでないファイルを扱う状況である。
- 一般ユーザのテンポラリディレクトリを管理者が削除する場合、管理者にとって、その一般ユーザという他ユーザが書き込めるディレクトリを扱うことになる。(なお、一般ユーザがディレクトリを削除するとき、そのディレクトリは管理者という他ユーザが書き込めるが、管理者を信用するのは前提であるため、これは危険とはみなされない。)

最初の状況を避けるのは簡単である。テンポラリディレクトリを生成するときに、mkdir に指定するパーミッションで 0700 を指定すればよい。

しかし、後のふたつの状況を避けることは困難である。複数ユーザで意図的にディレクトリを共有するのであれば、ときに削除が必要になる。また、管理者が一般ユーザのテンポラリディレクトリを削除するのは、/tmp を掃除するときに必要なことになる。

信用できない他ユーザが書き込めるディレクトリを削除するのが危険である理由を例によって説明する。

ユーザ A が /tmp/foo に以下のようなディレクトリおよびファイルを作って放置し、管理者が削除することを決定したと仮定する。

/tmp/foo	所有者がユーザ A のディレクトリ
/tmp/foo/passwd	所有者がユーザ A のファイル

rmdir システムコールでディレクトリを削除する場合、中身が空でなければならない。このためディレクトリを削除するツールは、削除対象のディレクトリから末端のファイルまでたどり、末端のファイルから削除を始める。

ディレクトリ構造をたどるには、ディレクトリとそれ以外のファイルを区別し、ディレクトリの場合は再帰的に削除を行うという判断が必要である。このとき、あるパス名が指す実体がディレクトリかどうかを判断するには lstat システムコールを使う。lstat はシ

シンボリックリンクをたどらないため、ディレクトリと、ディレクトリを指すシンボリックリンクを区別できる。

また、ディレクトリであると判明した後に、その直下に存在するファイル名を得るには `opendir/readdir/closedir` 関数を用いる。以下ではこれらを `opendir` とだけ記述する。

これを例に適用すると素朴には次の動作となる。

1. `/tmp/foo` がディレクトリかどうか `lstat` で調べる -> ディレクトリである
2. `/tmp/foo` ディレクトリの中身を `opendir` で調べる -> `passwd` がある
3. `/tmp/foo/passwd` がディレクトリであるかどうか `lstat` で調べる -> ディレクトリでない
4. `/tmp/foo/passwd` を `unlink` で削除する
5. `/tmp/foo` を `rmdir` で削除する

なお、ここで例にあげたディレクトリ構造ではディレクトリにひとつのファイルしか入っていないため、上記のようなシーケンシャルな動作になるが、一般にはひとつのディレクトリに複数のファイル・ディレクトリが入っているため、再帰的な動作が必要になる。

ところで、`/tmp/foo` はユーザ A が所有しているため、ユーザ A はいつでも書き換えられる。ここで、4 の直前、すなわち管理者が `/tmp/foo/passwd` を `lstat` でディレクトリでないと判断した後、`/tmp/foo/passwd` を `unlink` で削除する前に、以下の変更を加えた場合を考える。

3.1. ユーザ A が `/tmp/foo` を `/tmp/bar` に `rename` システムコールで改名する

3.2. ユーザ A が `/tmp/foo` から `/etc` へのシンボリックリンクを `symlink` システムコールで作成する

この場合、4 で管理者は `/tmp/foo/passwd` を削除するが、これは `/tmp/foo` が `/etc` へのシンボリックリンクになった後であるため、`/etc/passwd` を削除することになる。`/etc/passwd` は `/tmp/foo` 内のファイルでは無く、これは意図された削除ではない。`/etc/passwd` が削除されると、ログインできなくなる等、多くの問題が発生する可能性がある。つまり、ユーザ A によるシステムへの攻撃が成功する。

このように上述の素朴なディレクトリ削除は危険である。

同様に、複数ユーザでディレクトリを共有した場合、共有ディレクトリ内で素朴な削除を行った場合、共有していないユーザ個人のファイルの削除を引き起こす可能性がある。

この危険性は、`/tmp/foo/passwd` というパス名が指すファイルの実体が `/`, `/tmp`, `/tmp/foo` というパス内の上位の要素の変化に伴って変わりうる、という点が原因である。`/` と `/tmp` が指すファイルは管理者しか変更できないので問題ないが、`/tmp/foo` はユーザ A が変更

できるので現実的な問題となる。

上記の素朴な削除では、`/tmp/foo` がシンボリックリンクでない本物のディレクトリであることを一度は確認しており、その確認した時点では `/tmp/foo/passwd` を削除すべきであるという判断は正しい。しかし、`/tmp/foo` はユーザ A がいつでも変更できるので、実際に削除する時点でもその判断が正しいという保証はなく、そのずれが脆弱性を生んでいる。このように確認してから実際に行うまでの時間差の問題を TOCTTOU (Time of Check to Time of Use) 問題という。

これを安全に削除するひとつの方法は、`chdir` システムコールを用いてカレントディレクトリを移動し、`unlink` に与えるパスは `/` を含まない単純なファイル名にすることである。`chdir` でカレントディレクトリを移動した後、本当に削除対象のファイルがあるディレクトリに移動できたかどうかを検査し、意図どおりに移動できていなかったら中断する。`chdir` はファイルシステムを変更しない操作なので、ここで中断すればファイルシステムに悪影響を与えない。そして、`/` を含まない単純なファイル名での `unlink` は、カレントディレクトリ直下のファイルしか削除できない。このため、`unlink` によってどのファイルが削除されても、それは削除対象ディレクトリ直下のファイルであるから削除対象であり、削除して問題ない。

ここでカレントディレクトリが本当に意図したディレクトリであるかどうかという検査が必要になる。これは `lstat` システムコールで得た `st_dev` (デバイス番号) と `st_ino` (i ノード番号) を比較することで行う。削除対象のディレクトリのパス名で `lstat` した結果と、`chdir` した後にカレントディレクトリ `."` を `lstat` した結果で `st_dev` と `st_ino` がそれぞれ一致すれば、意図どおりのディレクトリに移動できたことがわかる。もし異なれば、`lstat` してから `chdir` するまでにディレクトリがシンボリックリンクにすり替えられる等の攻撃を受けていることになる。

この方法により前述の `/tmp/foo` を安全に削除すると以下ようになる。

1. `/tmp` を `lstat` して結果を記録する
2. `/tmp` に `chdir` で移動する
3. カレントディレクトリを `lstat` して `/tmp` の `lstat` の結果と比較する -> 異なっていたら中断する
4. `foo` を `lstat` してディレクトリかどうか調べる -> ディレクトリである
5. `foo` に `chdir` で移動する
6. カレントディレクトリを `lstat` して、`foo` の `lstat` の結果と比較する -> 異なっていたら中断する
7. カレントディレクトリの中身を `opendir` で調べる -> `passwd` がある
8. `passwd` を `lstat` でディレクトリかどうか調べる -> ディレクトリでない
9. `passwd` を `unlink` で削除する

10. 親ディレクトリ "." に `chdir` する
11. カレントディレクトリを `lstat` して、`/tmp` の `lstat` の結果と比較する -> 異なっていたら中断する
12. `foo` を `rmdir` で削除する

このように削除しているとき、ユーザ A が `/tmp/foo` を `/tmp/bar` に `rename` し、`/tmp/foo` を `/etc` へのシンボリックリンクにする攻撃を行うと考える。

この攻撃が 4 以前に行われると、`foo` はディレクトリでないと判断され、攻撃で作られたシンボリックリンク `foo` を `unlink` して終了する。つまり、`/etc/passwd` は `unlink` されず、攻撃は失敗する。

攻撃が 4 と 5 の間で行われると、5 の `chdir` では `/etc` にカレントディレクトリが移動することになる。しかし、6 の検査で意図しないディレクトリに移動したことが検出されて中断する。つまり、`/etc/passwd` は `unlink` されず、攻撃は失敗する。

攻撃が 5 以降 12 以前に行われると、削除開始時点において `/tmp/foo/passwd` に存在したファイルの削除が行われる。`/etc/passwd` は `unlink` されず、攻撃は失敗する。なお攻撃が 9 以前である場合は、削除されるファイルは `/tmp/bar/passwd` という削除対象外のパスに移動している。しかし、削除開始時点においては `/tmp/foo/passwd` という削除対象であったため、これを削除することは間違いでない。

攻撃が 12 以降に行われた場合、`/tmp/foo` が存在しないため、`/tmp/foo` を `/tmp/bar` に `rename` することができない。このため、攻撃は失敗する。

このように、どのタイミングで攻撃が起きても、`/etc/passwd` は `unlink` されず、攻撃は失敗する。

一般には、`chdir` でカレントディレクトリに移動したときに、意図どおりに移動できたことを確認すれば、関係ないファイルの削除を防ぐことができる。

7 新システムコール: `openat` とその仲間

前節では他ユーザが書き込めるディレクトリを `chdir` を用いて安全に削除する方法について述べた。

しかし、`chdir` はカレントディレクトリというプロセス属性を変更するため、スレッドセーフでない。これは `rm` のような削除専用コマンドであれば問題にならないが、Perl などのように用途を限定しない言語のライブラリとしては問題がある。

スレッドセーフかつ安全に他ユーザが書き込めるディレクトリを削除するのは簡単でない。

カレントディレクトリに依存せずに `/tmp/foo/passwd` を削除するには、`/tmp/foo/passwd` というパス名を `unlink` に与えざるを得ない。したがって、他ユーザが `/tmp/foo` を変更できるとすれば危険性を排除できない。この危険性を排除するためには、

/tmp/foo を他ユーザが変更できないようにすることが考えられる。たとえば、/tmp/foo の所有者を自分に変更して、パーミッションを 700 に変えるなどである。

しかし、所有者を変更することは管理者しかできない。また、所有者やパーミッションの変更はそれ自身が symlink attack の対象になるため、注意深い実装が必要である。

スレッドセーフかつ安全に他ユーザが書き込めるディレクトリを削除することにはさまざまな困難があり、可能な限り避けたほうがよい。とくに、テンポラリディレクトリを生成・削除するだけであれば、生成時にパーミッションを 0700 に設定するだけで避けられるのでぜひそうやって避けるべきである。

ただし、最近、安全なディレクトリの削除にも利用できる新しい API が実装・提案されている。openat に代表されるこの API は、ファイルディスクリプタを活用して、パス名をたどってファイルの実体を得る操作を削減可能とする。まず、パス名を受け取る多くのシステムコールに対応して、加えてカレントディレクトリの代わりになるファイルディスクリプタも受け取るシステムコールが用意される。また、パス名を受け取る代わりにファイルディスクリプタを受け取るシステムコール・関数もいくつか用意される。

たとえば、open システムコールに対応して、openat システムコールが定義される。openat は open の引数に加え、ディレクトリを示すファイルディスクリプタを受け取り、パス名が相対パスだったときに、起点にカレントディレクトリでなく指定されたファイルディスクリプタを用いる。

また、これらのシステムコールの多くにはフラグ引数も追加され、動作を修飾できる。たとえば、指定したファイルディスクリプタでなく、プロセスのカレントディレクトリを用いるというフラグが指定ができる。また、fstatat システムコールでは、シンボリックリンクをたどるかどうかを指定でき、stat と lstat のどちらの動作も選べる。unlinkat システムコールはディレクトリの削除かどうかを指定でき、unlink と rmdir のどちらの動作も選べる。

さらに fdopendir 関数が用意され、open して得たファイルディスクリプタをもとに DIR 構造体を生成できる。これにより、ディレクトリを open した結果を、readdir/closedir に使うことができる。

これらにより、カレントディレクトリに依存せずに安全なディレクトリの削除を実装できる。つまり、カレントディレクトリの代わりにディレクトリを open したファイルディスクリプタを使えばよい。たとえば、前述した /tmp/foo/passwd の削除は以下のように実現できる。

1. /tmp を lstat して結果を記録する
2. /tmp を open する -> fd_tmp が得られる
3. fd_tmp を fstat して /tmp の lstat の結果と比較する -> 異なっていたら中断する

4. fd_tmp から相対で foo を fstatat してディレクトリかどうか調べる -> ディレクトリである
5. fd_tmp から相対で foo を openat する -> fd_foo が得られる
6. fd_foo を fstat して foo の fstatat の結果と比較する -> 異なっていたら中断する
7. fd_foo の中身を fdopendir で調べる -> passwd がある
8. fd_foo から相対で passwd を fstatat でディレクトリかどうか調べる -> ディレクトリでない
9. fd_foo から相対で passwd を unlinkat で削除する
10. fd_foo を close する
11. fd_tmp から相対で foo ディレクトリを unlinkat で削除する
12. fd_tmp を close する

なお、ここで fstatat はシンボリックリンクをたどらないようにフラグを指定して使用するものとする。

この方法は、chdir による方法と同様に安全であり、かつ、カレントディレクトリに依存していないのでスレッドセーフである。

openat は Solaris 9 で実装され、Linux 2.6.16 でも実装された。POSIX の次期バージョンにも提案されており、広く普及していくことが期待される。

8 sticky bit

本稿では /tmp 直下にある他ユーザのファイルは unlink, rmdir, rename できないものと仮定している。これは /tmp には sticky bit (S_ISVTX フラグ) が設定されていることを意味している。

Unix では原則的に、あるファイルの unlink や rename はそのファイル自身の所有者には関係なく、そのファイルが存在するディレクトリに書き込み権限があるかどうかによって決まる。

この原則を /tmp にそのまま適用すると、/tmp は誰でも書き込み可能であるから、直下に作ったテンポラリファイルは誰のファイルでも削除・改名できることになる。

この場合、自分が作ったテンポラリファイルを後で open して使ってはならない。たとえば他のコマンドにパス名を渡して使わせるのも危険である。そのコマンドがそのパス名で open するとき、そこにあったファイルは削除されたり、別のファイルにすり替えられている可能性がある。

このように自由に削除・改名できてしまうのは不適切であるため、ディレクトリには sticky bit を設定できる。

sticky bit が設定されているディレクトリ下のファイルは、その sticky bit の設定され

ているディレクトリもしくは削除対象のファイルの所有者か、あるいは特権があるユーザしか unlink はできない。unlink だけでなく、rename にも同様な制約がある。

この制約により、/tmp 直下にテンポラリファイルを生成すると、生成したファイルは信用できない他ユーザにすり替えられることはない。

9 環境変数 TMPDIR

ここまで述べてきたように、テンポラリファイルを安全に扱うには注意が必要である。しかし、そもそもそのように注意が必要なのは、ユーザ全員が /tmp を共有しているという理由が大きい。

もし、/tmp のような共有ディレクトリを用いず、個々のユーザ専用のディレクトリ内にテンポラリファイルを生成するのであれば、危険性はかなり抑制される。

そこで、テンポラリファイルを生成するアプリケーションは、生成するディレクトリを外部から指定できることが望まれる。

そして、外部からテンポラリファイルのためのディレクトリを指定するには環境変数 TMPDIR を使うという慣習がある。もし TMPDIR が設定されているなら、そのディレクトリ内にテンポラリファイルを生成することが期待される。

アプリケーションが TMPDIR をサポートしていれば、/tmp を避けることが可能となる。

ただし、TMPDIR を使うには TMPDIR 環境変数が悪意をもって設定されてはならない。従って、リモートから環境変数を任意に設定できる機能を持っているアプリケーションでは素朴にサポートしてはならない。

なお、C 言語の mkstemp, tmpfile 関数は TMPDIR をサポートしていないので、サポートするには別途コードが必要になる。また、OpenBSD の mktemp コマンドは -t オプションにより TMPDIR を使うようになる。

10 シェルスクリプト

シェルスクリプトでテンポラリファイルを安全に生成することは簡単でないので、ここで触れる。

シェルスクリプトでテンポラリファイルを作る伝統的な方法は /tmp/foo.\$\$ などといったファイル名を使うことである。\$\$ はプロセス番号に展開されるため、他のプロセスとファイル名は衝突しないという意図である。以下に例をあげる。これは日付を（無駄にテンポラリファイルを経由して）表示するスクリプトであるが、とくに意味はない。

```
date > /tmp/foo.$$
cat /tmp/foo.$$
rm /tmp/foo.$$
```

しかし、このテンポラリファイルの生成は危険である。

衝突しないというのは、他のユーザも協調してテンポラリファイルの名前にプロセス番号を使うときの話であり、悪意あるユーザにとって、そうでない番号をファイル名に使うことを防げる機構は存在しない。

たとえば、ユーザ baz が /tmp/foo.1000 に /home/foo/.ssh/authorized_keys を指すシンボリックリンクを作った場合、ユーザ foo が上記のように /tmp/foo.\$\$ というテンポラリファイルを生成し、偶然プロセス番号が 1000 だった場合、/home/foo/.ssh/authorized_keys が破壊されてしまう。

これを避けるためには先に述べたとおり、テンポラリファイルを生成するときに O_EXCL を指定すればよい。

これを POSIX シェルで行うには、set -C を使う。set -C は noclobber オプションを有効にする。noclobber が有効な場合、> によるリダイレクトでの open には O_EXCL が指定される。

これを使ってテンポラリファイルの生成に挑戦し、失敗したときは終了するには以下のようにする。また、ここでは TMPDIR が指定された場合には使うようにしている。

```
t="{TMPDIR:-/tmp}/foo.$$"
set -C
:> "$t" || exit 1
set +C
date > "$t"
cat "$t"
rm "$t"
```

失敗したときにファイル名を変えて再挑戦するのであれば以下のようにする。

```
set -C
n=1
t="{TMPDIR:-/tmp}/foo.$RANDOM$n$$"
while ! :> "$t"; do
  n=`expr $n + 1`
  t="{TMPDIR:-/tmp}/foo.$RANDOM$n$$"
done
set +C
```

```
date > "$t"
cat "$t"
rm "$t"
```

なお、ここで \$RANDOM と \$\$ を使っているが、これらは攻撃に対する防御として本質的なものではない。他ユーザの意図的な攻撃からの防御はこれまでに述べたとおり set -C による O_EXCL が本質である。ただし、\$RANDOM や \$\$ には意図的でない衝突の確率を減らす効果がある。

また、標準化されているものではないが、システムが mktemp コマンドを提供していればこれを使うこともできる。mktemp は引数に指定したテンプレートに従って安全にテンポラリファイルを生成し、生成したファイルのパスを出力する。この mktemp コマンドは OpenBSD が発祥である。また、GNU coreutils 6.10 でも提供されている。

```
t=`mktemp -t foo.XXXXXXXXXX`
date > "$t"
cat "$t"
rm "$t"
```

11 まとめ

テンポラリファイルの扱いとシンボリックリンク攻撃へ対処について述べた。この問題は古くから知られているが、まだ多くの問題が発生している。また、対処法についても openat など新しい技術が開発されている。

以上

12 参考文献

- IEEE Std 1003.1-2001 (POSIX)
- Secure Programming for Linux and Unix HOWTO: 6.10. 競合状態を避ける
<http://www.linux.or.jp/JF/JFdocs/Secure-Programs-HOWTO/avoid-race.html>
- IPA ISEC セキュア・プログラミング講座：C/C++言語編 第9章ファイル対策
<http://www.ipa.go.jp/security/awareness/vendor/programmingv2/contents/c801.html>

6. コンピュータウイルスの動向

武田 圭史

1 はじめに

本稿では 2008 年下半期に発生したインシデントの発生状況、特に攻撃等に用いられた攻撃手法について分析する。上半期の報告においても 2007 年あたりから利益を得ることを目的としていると思われる攻撃の増加について述べているが、この傾向は依然強まっていると考えられる。上半期に引き続き、SQL インジェクションにより Web サイトのコンテンツを改ざんし、ダイナミック DNS 等を用いて指定される海外のサーバよりマルウェアをダウンロードさせるといったタイプの攻撃が引き続き観測されている。

その一方で、USB メモリの自動実行機能を悪用して USB メモリをパソコンに挿入した際に感染を広げるタイプのコンピュータウイルスの被害も広がっている。

2 概要

2007 年までの、企業官公庁等組織における情報セキュリティインシデントの動向としては、暴露型ウイルス感染による匿名ファイル共有ネットワークへの自宅からの情報漏えい、パソコンや外部記憶媒体の紛失・盗難、電子メールの宛先の間違いや第三者のアドレス列挙による電子メールアドレスの漏えいなど、ユーザのミスや管理の不備に起因する事故が多く報告されてきた。

2007 年の後半から 2008 年上半期にかけては、SQL を使用してデータベースにアクセスする Web アプリケーションの脆弱性を攻略するというタイプのインシデントが多く確認された。これらの攻撃では、データベースに格納されているクレジットカードなどの情報を盗み出したり、Web ページの内容を改ざんし、アクセスしたユーザのパソコンのブラウザの脆弱性などを利用して悪意のあるプログラムをダウンロードさせるといった手口が多く見られた。2008 年下半期においても、Web アプリケーションの脆弱性攻略による不正アクセスが多く発生している。

2008 年夏ごろから、主にアジア圏を中心に被害が拡大していった USB メモリ等外部記憶媒体を介して感染を広げるタイプのウイルスが日本に多く上陸した。USB メモリを介して感染するウイルスに関しては次項において詳細に解説する。

3 USB メモリを介して感染を広げるコンピュータウイルス

以前より USB 等の外部記憶媒体を接続した際に自動実行する機能についてはその危険性が指摘されており、一部のセキュリティ専門家達による指摘はおこなわれていたものの、大規模なウイルス感染等の事態に発展することはなかった。これまで広く利用されてきた Windows XP では、CD-ROM の場合にはディレクトリ内に設置された Autorun.inf という設定ファイルに基づき、ドライブ挿入時の自動実行処理の内容が規定されていた。

しかし、通常の USB メモリでは、メモリをパソコンに挿入しただけではプログラムを自動実行する機能はなく、ユーザが当該ドライブをダブルクリックするか、メモリ領域の一部を CD-ROM としてパソコンに認識させる特殊な USB メモリを使用する必要があった。

一方、Windows Vista では、通常の USB メモリでも最上位のフォルダ内に Autorun.inf が設置されていればその内容にしたがって自動実行が行われる仕様とされた。この結果、Autorun.inf によって起動されるコンピュータウイルスを含む USB メモリを挿入されたコンピュータは、その実行ファイルを自動的に実行し、当該ホストをウイルスに感染させるだけでなく、他の USB メモリなど外部記憶媒体が挿入された場合においてはこれらの媒体に感染を広げることとなった。

なお、USB メモリの大容量化と低価格化が急激に進み、利便性の高さから利用機会が増えていること、外部記憶媒体におけるウイルス対策の必要性を意識していない利用者が多数存在していることが感染拡大の要因であると推測される。

夏以降、国内でも多くの感染がみられ年末に向けて感染を拡大していった同種のコンピュータウイルスは、関連するファイルに対して隠しファイルの属性を設定しており、隠しファイルを表示するよう設定を変更しても常に隠しファイルを表示しないように設定を変更し続ける機能を持っていたため、一般のユーザがこのファイルの存在に気づくことがないように工夫がされていた。USB メモリを介して感染を広げるコンピュータウイルスは、被害の拡散が局地的ではあるが、自宅のパソコンで使用した USB メモリがウイルスに感染し、その USB メモリを組織内に持ち込んで利用したことで、組織内のパソコンに感染が拡大した事例も確認されている。また、数多くの亜種が出現し、既存のウイルスに外部記憶媒体への感染機能を追加したものも確認された。このようなコンピュータウイルスは、急速にインターネット上で感染を広げるような最近のウイルスとは大きく傾向が異なったために当初ウイルス対策ソフトでの検知も難しい状態が続いた。

ウイルスによる被害が拡大すると、USB メモリを挿入しただけでは内容を自動実行しないはずの Windows XP でも多くの被害が観測された。これは、Windows XP では USB 挿入時に表示されるアイコンをダブルクリックすると Autorun.inf に指定された内容を実行するという機能によるものであった。多くのユーザが挿入した USB メモリの内容を閲覧するために表示されたアイコンをダブルクリックするためである。Autorun.inf に指定された内容を実行せずにフォルダの内容を閲覧するためには、右クリックから「開く」を

選択するように操作しなければならない。

このようなリスクの回避策として、Windows Vista においては外部記憶媒体接続時の自動実行機能の設定を停止する方法が推奨されている。また Windows XP においては、USB メモリアイコンをダブルクリックした際の Autorun.inf に指定される内容の実行を停止するためのレジストリ設定の変更などが推奨されている。ただし、これらの設定変更をおこなった場合に一部の製品等で本来想定していた自動実行が行われなくなるために製品の挙動が想定されるものと異なるなどの影響を受ける場合がある。また、使用している OS(Windows)のバージョンや修正プログラムの適用状態によって動作が異なるなどの症状が報告されている。

4 まとめ

本稿では、2008 年下半期におけるインシデントの新たな動向として、USB メモリを介して感染を広げるコンピュータウイルスの増加をとりあげた。本事象については広く利用されている Windows プラットフォームにおいて Windows XP の際に危険性が予見できたはずの USB メモリコンテンツの自動実行という機能が Windows Vista において実装されたために発生したものであった。また、自動実行機能の仕様が Windows XP の時点と Windows Vista においても細かな点まで様々な変更点があり挙動に相違が生じているため、対応の手法の伝搬などにおいても様々な混乱を生じている。すでに USB メモリ等を介した感染とネットワークを通じた感染の両方を感染経路とするハイブリッド型のコンピュータウイルスが出現しており、今後の技術的展開の動向に注意をしていく必要がある。

以上

参考文献

「コンピュータウイルス・不正アクセスの届出状況[11 月分]について」, 情報処理推進機構
<http://www.ipa.go.jp/security/txt/2008/12outline.html>

7. ダークネット観測の技術動向と観測事例

井上 大介

1 はじめに

ダークネットとはインターネット上の未使用の IP アドレス空間のことを示す。ダークネットに到来するパケットを観測することで、インターネットを経由して感染を広めるマルウェア（以下、リモートエクスプロイト型マルウェア）の活動傾向などを把握することが出来る。本稿では、ダークネット観測の技術動向ならびに、2008 年度下半期の観測事例について示す。

2 ダークネット

ダークネットとは、インターネット上で到達可能かつ未使用の IP アドレス空間のことを指す。未使用の IP アドレスに対しパケットが送信されることは、通常のインターネット利用の範囲においては稀なはずであるが、実際にダークネットを観測してみると相当数のパケットが到着することが分かる。これらのパケットには

- リモートエクスプロイト型マルウェアが次の感染対象を探查するためのスキャン
- マルウェアが感染対象の脆弱性を攻撃するためのエクスプロイトコード
- 送信元 IP アドレスが詐称された DDoS 攻撃を被っているサーバからの応答であるバックスキャッタ

などが含まれ、インターネット上での何らかの不正な活動に起因している。そのため、ダークネットに到着するパケットを受動的に観測することで、インターネット上で発生している不正な活動の傾向把握が可能になる。ダークネット観測の最大の利点は、パケットを正・不正で区別する処理を必要とせず、全てのパケットを不正なものを見なして分析することが出来る点にある。また、パケットの送信元に対して能動的に適切な応答をすると、さらに詳細な攻撃情報やマルウェアの検体を捕獲することも可能である。

3 ダークネットセンサ

ダークネット観測を行なう場合、パケット収集・応答用のセンサを設置する。センサは、パケットの送信元に対する応答の程度によって次のように分類される。

- **ブラックホールセンサ**：パケットの送信元に対し、全く応答を行なわないセンサ。メンテナンスが容易であり大規模なダークネット観測に向く。無応答であるため、外部

からセンサの存在を検知することが困難であるという利点もある。ただし、マルウェアの感染活動の初期段階であるスキャンやコネクションレスの 익스プロイトコードなどは観測可能であるが、それ以降の挙動を観測することは出来ない。

- **低インタラクションセンサ**：パケットの送信元に対し、一定レベルの応答を返すセンサ。TCP の SYN パケットに対して SYN-ACK パケットを返すセンサや、OS の既知の脆弱性を模擬する低インタラクションハニーポットがここに含まれる。リッスンしているポートや応答の傾向などからセンサの存在が露呈し易く、アドレスが連続した大規模なダークネットでの運用には不向きである。
- **高インタラクションセンサ**：実マシン、もしくはそれに準じた応答を返すセンサ（いわゆる、高インタラクションハニーポット）。マルウェアの本体やその感染時の挙動、攻撃者が不正アクセスを試みた際の行動履歴など多様な情報が取得可能である。ただし、安全な運用を行うためのコストは非常に高く、大規模運用には不向きである。

4 ダークネット観測プロジェクト

世界各国でダークネット観測プロジェクトが進行中である。ここでは、主要なダークネット観測プロジェクトについての概要を記す。

- **Network Telescope** :
<http://www.caida.org/research/security/telescope/>
米国の CAIDA (Cooperative Association for Internet Data Analysis) によるダークネット観測プロジェクト。16 万アドレス以上のダークネットを観測し、バックスキャッタやワームによるトラフィックのデータセットを公開している。
- **REN-ISAC** :
<http://www.ren-isac.net/monitoring.html>
米国の研究教育ネットワーク (REN: Research and Education Networking) におけるセキュリティ情報の共有・分析プロジェクト。Internet2 で観測されたトラフィックを分析し、観測結果を公開している。
- **Internet Motion Sensor (IMS)** :
(現在、Web での観測情報の公開は停止している)
米国のミシガン大学による /8 ネットワークを含む 1700 万アドレス以上の大規模ダークネット観測プロジェクト。観測された TCP SYN パケットの一部にセンサ側から

SYN-ACK を返すことで TCP コネクションの確立を試み、コネクション確立後の最初のパケットのペイロードを収集・分析する機能を持つ。

- **Leurre.com** :
<http://www.leurrecom.org/>
フランスの研究機関 Eurecom による分散型ハニーポットを用いた情報収集・分析プロジェクト。観測対象の IP アドレス数は比較的少数であるが、観測地域は世界各国に分散している。第1世代の Leurre.com v1.0 は低インタラクションセンサの Honeyd を使用していたが、第2世代の Leurre.com v2.0 では SGNET を使用して情報収集能力の向上を図っている。
- **Internet Storm Center (ISC)** :
<http://isc.sans.org/>
米国の SANS (SysAdmin, Audit, Networking, and Security) による、セキュリティ情報の収集・分析プロジェクト。厳密にはダークネット観測ではないが、50 万アドレス以上のファイアウォールログを、DSHield と呼ばれるシステムに集約し、統計情報やボランティアによる分析レポートを公開している。
- **Worldwide Observatory of Malicious Behaviors and Attack Threats (WOMBAT)** :
<http://www.wombat-project.eu/>
欧州委員会 (European Commission) のセキュリティ関連プロジェクト。フランステレコム、Eurecom、ウイーン工科大学、シマンテックなど複数の研究機関が参画し、ダークネットトラフィックやマルウェアなどのデータ収集と、それらの分析・原因究明を目指している。

日本国内では下記のプロジェクトが進行中である。

- **ISDAS (JPCERT/CC)**
<http://www.jpcert.or.jp/isdas/>
- **@police (警察庁)**
<http://www.cyberpolice.go.jp/detect/observation.html>
- **MUSTAN (情報処理推進機構)**
http://mustan.ipa.go.jp/mustan_web/

- WCLSCAN (三菱総合研究所ほか)

<http://www.wclscan.org/>

- nicter (情報通信研究機構)

(現在、Web での観測情報の公開は行っていない)

5 最近のダークネット観測事例

以下では、2008年9月～11月の3ヶ月間のダークネット観測事例を示す。この事例は、情報通信研究機構が研究開発を進める nicter システムが、/16 ダークネット (65,536 個の未使用 IP アドレスブロック) に設置したブラックホールセンサによって得られた観測結果に基づいている。

(1) パケット数とユニークホスト数

図は、nicter の観測ポイントの1つである /16 ダークネットに、2008年9月1日から11月30日までの3ヶ月間に到着したパケットの、1日あたりの総数 (青実線) および1日あたりの送信元ホストのユニーク数 (赤破線) を示している。この /16 ダークネットでは、1日平均で約250万パケット、約3.2万台のユニークホストを観測した。

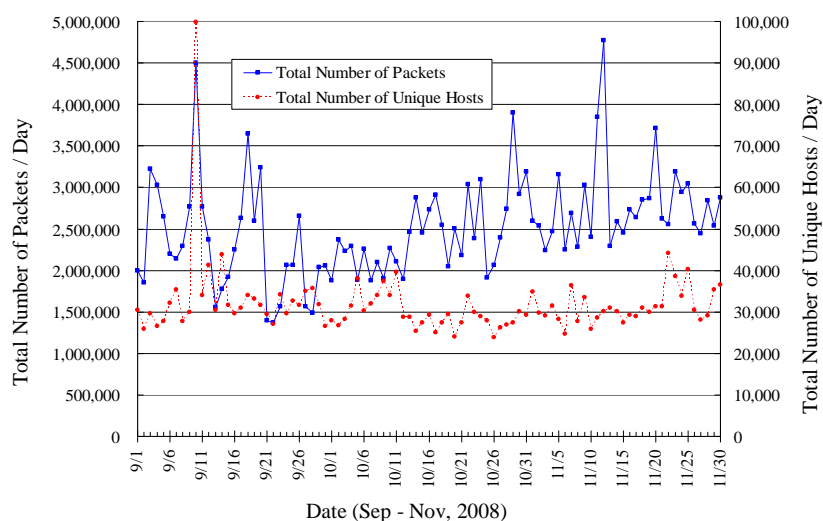


図 1 /16 ダークネットのパケット数とユニークホスト数

(2) プロトコル別のパケット数

ダークネットの観測結果を分析する際の常套手段として、まず全てのパケットを TCP、UDP、ICMP のプロトコル別に分離することが行われる。図 は、/16 ダークネットに到着した 1 日あたりのパケット数をプロトコル別に示したものである。パケット数では TCP (青実線) が平均約 148 万個と最も多く、次いで UDP (赤破線) の約 77 万個、ICMP およびその他のプロトコル (黄破線) は約 28 万個のパケットを観測した。

TCP のパケット数は変動が大きく何箇所かピークも見られる。3 ヶ月間で最も大きいピークは 11 月 12 日に観測されており、これは中国の特定のサーバからの大量のバックスキヤッタの影響である。

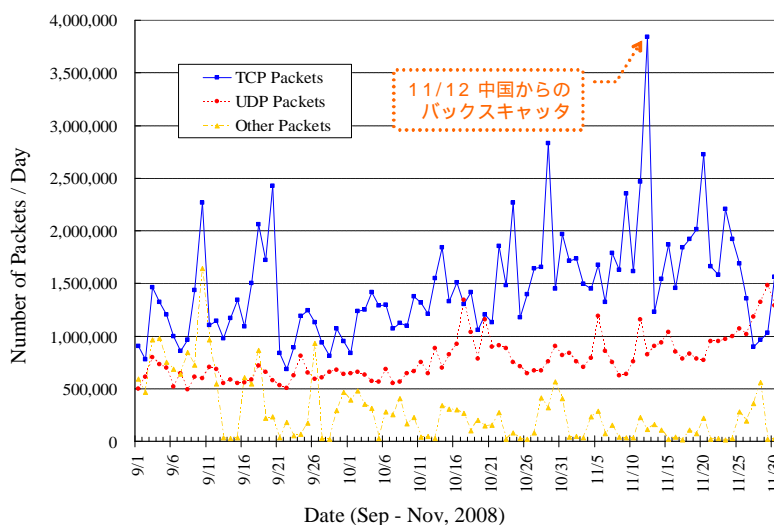


図 2 プロトコル別のパケット数

(3) プロトコル別のユニークホスト数

プロトコル別に分離したユニークホスト数も、時に重要な情報を与えてくれる。図 はそれぞれ、TCP (青実線)、UDP (赤破線)、ICMP およびその他 (黄破線) のプロトコルでパケットを送信したユニークホスト数を示している。/16 ダークネットにおいて、TCP は 1 日平均約 1.8 万台、UDP は約 1 万台、ICMP およびその他は約 3 千台のユニークホストからのアクセスを観測した。

9 月 10 日には TCP で顕著なピークが現れている。これは、ボットネットからの TCP 1433 番ポートへのスキャン活動の影響である。このスキャン活動は約 14 時間継続し、合計約 7

万台のユニークホストが観測された。一般に、ボットネットの活動では多数のホストが一斉動作するため、大規模なダークネットユニークホスト数を観測することで、その活動を検知することが可能である。

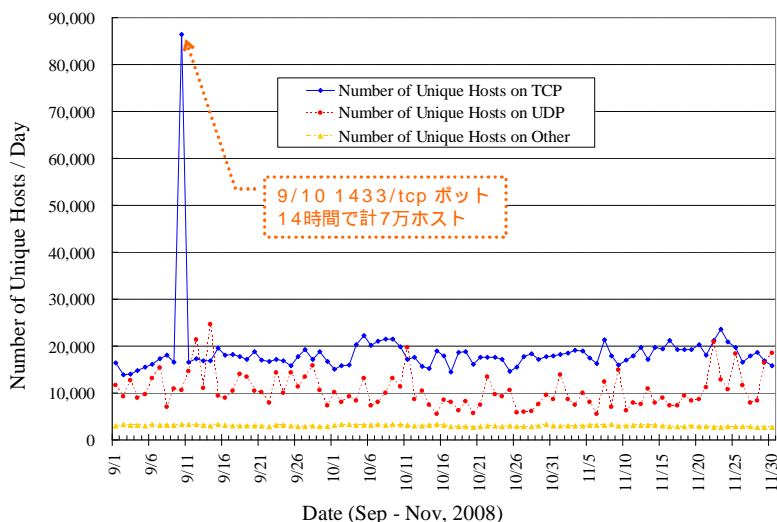


図 3 プロトコル別のユニークホスト数

(4) TCP 445 番ポートの観測結果

さらに詳細な分析を行うためには、TCP と UDP のポート番号ごとにパケットを分離することが有効である。図は、/16 ダークネットに到着したパケットのうち、宛先ポート番号が TCP 445 番ポートであるパケットを抜き出し、1 日あたりのパケット数（青実線）およびそれらパケットの送信元ホストのユニーク数（赤破線）を示したものである。図より、9 月中旬からパケット数、ユニークホスト数ともに増加傾向を示していることが分かる。

これは、Windows サーバサービスの脆弱性 MS08-067 を狙ったワームである W32.Downadup（別名 W32/Conficker.worm）が感染を拡大していることが原因と考えられる。その増加傾向は 2009 年 1 月現在も依然として続いており、2003 年 8 月に大規模感染を引き起こした Blaster の再来とも言われている。Microsoft は 2008 年 10 月 24 日に MS08-067 を修正する緊急 Windows アップデートを行ったが、その 1 ヶ月以上前からこの脆弱性を悪用したワームの活動の兆候が、ダークネットにおいて観測されていた。

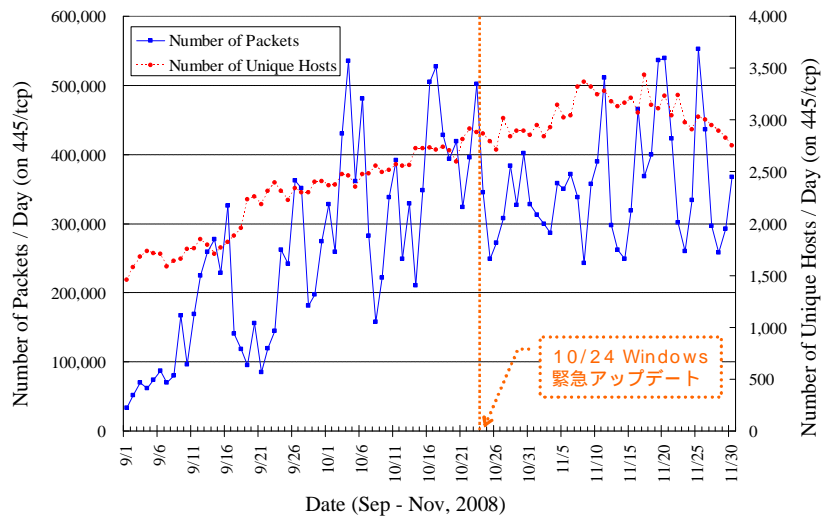


図 4 TCP 445 番ポートのパケット数とユニークホスト数

6 まとめ

本稿ではダークネット観測の技術動向と関連プロジェクト、および 2008 年度下半期の観測事例について述べた。大規模なダークネット観測により、インターネット上での不正な活動を迅速に検知することが可能であるが、観測結果から如何に原因を究明し、有効な対策につなげるかが課題となっている。

以上

8. IPsec 関連技術の標準化動向

馬場 達也

1 はじめに

2008 年下期 (7 月 ~ 12 月) における、注目すべきネットワークセキュリティ分野のトピックとしては、IETF (Internet Engineering Task Force) の IPsec (IP Security Protocol) 関連ワーキンググループ (WG) における標準化動向がある。IETF には、IPsec 関連の 3 つの WG があるが、この期間に新たな WG が承認され、既存の 2 つの WG において、WG のメインとなる文書が RFC として発行された。今回はこれらの標準化動向について報告する。

2 IPsec の概要

IPsec は、IP (Internet Protocol) にセキュリティ機能を追加するためのプロトコルスイートである。IPsec は、ESP (Encapsulating Security Protocol)、AH (Authentication Header)、IKE (Internet Key Exchange) の 3 つのプロトコルからなっており、ESP がデータの暗号化とメッセージ認証、AH がメッセージ認証、IKE が暗号化やメッセージ認証に必要な共通鍵の交換を行う。IPsec の最初のバージョン (IPsec バージョン 1) は、1995 年に RFC 1825 ~ RFC 1829 として発表され、鍵管理プロトコルである IKE (IKEv1) やいくつかの機能追加がなされたバージョン (IPsec バージョン 2) が 1998 年に RFC 2401 ~ RFC 2412, RFC 2451 として発表された。そして、2005 年に、リモートアクセス機能などが強化された IKEv2 を含むバージョン (IPsec バージョン 3) が RFC 4301 ~ RFC 4303, RFC 4305 ~ RFC 4307 として発表された。

IPsec (バージョン 3) の動作の手順は図 1 のとおりである。最初に、通信を行うノードの間で IKEv2 による交換が行われる。ここでは、IKE_SA_INIT 交換によって、IKEv2 の通信を保護するための IKE SA (Security Association) を確立し、次の IKE_AUTH 交換によって、データを保護するための IPsec SA (CHILD_SA) の確立と、相手認証を行う。その後は、確立された IPsec SA を使用して、データを IPsec の ESP または AH を使用して保護する。IPsec が提供する保護サービスは、使用するプロトコルによって異なり、表 1 のようになる。



図 1 IPsec の手順

表 1 IPsec が提供するサービス

提供サービス	サービスを実現する技術	サービスを提供する IPsec プロトコル
アクセス制御	セキュリティポリシーに従ったパケットフィルタリング	AH または ESP
データの完全性確保	メッセージ認証コード (MAC)	AH または ESP
データ送信元の認証	メッセージ認証コード (MAC)	AH または ESP
リプレイ防御	シーケンス番号のチェック	AH または ESP
データの機密性確保	共通鍵暗号による暗号化	ESP
トラフィック情報の機密性確保	共通鍵暗号による暗号化とトンネリング	ESP

3 IETF における IPsec に関する標準化の取り組み

現在、IETF には、IPsec 関連の WG として、IPSECME (IP Security Maintenance and Extensions)、MSEC (Multicast Security)、BTNS (Better-Than-Nothing Security) の 3 つが存在する。

- IPSECME WG

IKEv2 を含む IPsec に関する仕様の明確化や改良、機能追加を議論するためのものであり、2005 年に終了した IPSEC WG を引き継ぐ形で活動している。

- MSEC WG

ユニキャストにしか対応していない現在の IPsec/IKEv2 に対して、マルチキャストへの対応を議論している。

- BTNS WG

既存の IPsec では相手認証のための設定が煩雑であることから、相手認証をせずに、通信を保護するための仕組みを検討している。

2008 年下期には各 WG においていか次の動きがあった。

- (1) IPSECME WG が 2008 年 7 月に正式な WG として承認された。
- (2) MSEC WG において議論されていたメイン文書である “Multicast Extensions to the Security Architecture for the Internet Protocol” が RFC 5374 として発行された。
- (3) BTNS WG において議論されていたメイン文書である “Better-Than-Nothing Security: An Unauthenticated Mode of IPsec” が RFC 5386、” Problem and Applicability Statement for Better-Than-Nothing Security (BTNS)” が RFC 5387 として発行された。

4 IPsec によるマルチキャスト通信の保護 (MSEC WG)

ユニキャストが 1 対 1 の通信を行うものに対して、マルチキャストは 1 対多の通信を行うものである。マルチキャストは、インターネット上では実現が難しいが、NTT 東西のフレッツ網や NGN (フレッツ光ネクスト) 網、NTT コミュニケーションズやソフトバンクテレコムなどの IP-VPN 網において利用することができる。マルチキャスト通信は、会議の様相や経営層からのメッセージの中継や、売上データなどの一斉ファイル配信など、ビジネスにおいても有効に利用することができ、その際に暗号化などの IPsec の機能が有効となる。

従来の IPsec では、マルチキャスト通信の保護に対して、主に次のような問題があった。

マルチキャストにおける鍵管理

現状の IKE (IKEv1/IKEv2) では、1 対 1 のユニキャスト通信における鍵管理を想定しているため、1 対多の通信を行うマルチキャストグループでの鍵管理を行うことができない。

送信元の認証

IPsec では、パケットの受信者が付与されているメッセージ認証コード (MAC : Message Authentication Code) を検証することで、送信者が自身と同じ認証鍵を持っているということを確認する。ユニキャストの場合は、同じ認証鍵を持っているのは送信者と受信者のみであるため、メッセージ認証コードを検証することで、送信者を特定することができる。しかし、マルチキャストの場合は、グループで同じ認証鍵を共有するため、送信者が受信者と同じ認証鍵をもっていることを確認できたとしても、実は違うメンバが送信している可能性がある。

このため、MSEC WG では、1 対 1 の通信を想定している IPsec の SA を拡張した GSA (Group Security Association) を新たに定義し、GSA を確立するための鍵管理プロトコルを開発した。GSA は図 2 のように、レジストレーション SA、リキー SA、データ SA の 3 つから構成される。また、送信元を個々に認証する必要がある場合を想定し、共通鍵ベ

ースのメッセージ認証コード (MAC) ではなく、公開鍵ベースのデジタル署名を使用して送信元を認証する仕組みを開発している (RFC 4082, RFC 4359)。

マルチキャスト用 IPsec では、ユニキャスト用 IPsec と比較して SA だけでなく、SPD (Security Policy Database) PAD (Peer Authorization Database) といったデータベースも拡張され、それぞれ GSPD (Group Security Policy Database) GPAD (Group Peer Authorization Database) として再定義している。また、鍵管理プロトコルについては、IKEv1/IKEv2 の代わりに、GDOI (Group Domain of Interpretation) GSAKMP (Group Secure Association Key Management Protocol) などがある。しかし、ESP と AH のプロトコル自体には変更はないようにしている。表 2 にユニキャスト用 IPsec とマルチキャスト用 IPsec の違いをまとめる。

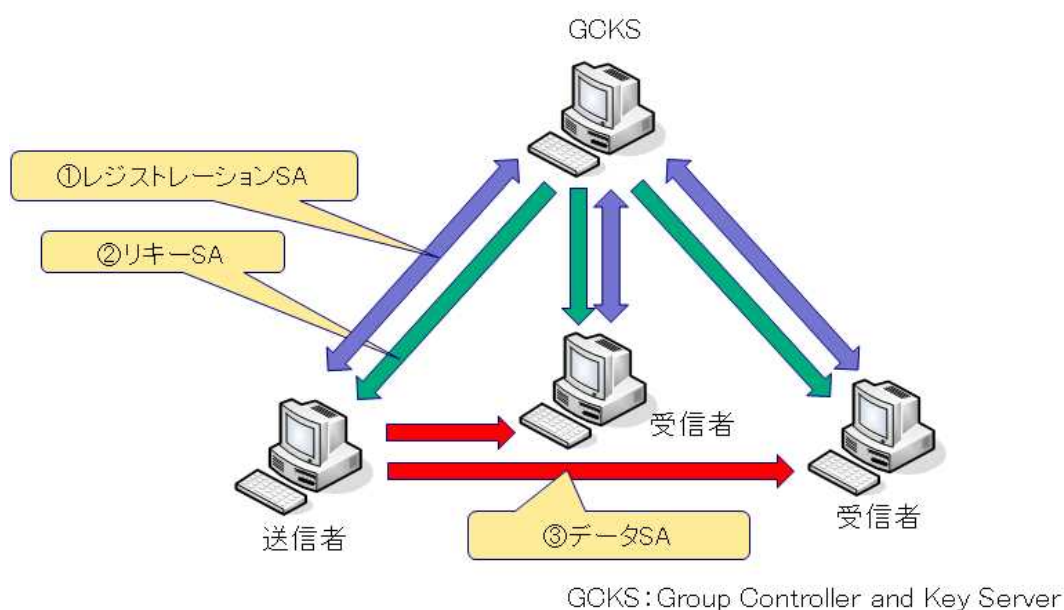


図 2 Group Security Association (GSA)

表 2 ユニキャスト用 IPsec とマルチキャスト用 IPsec の違い

ユニキャスト用 IPsec	マルチキャスト用 IPsec
SA (Security Association)	GSA (Group Security Association)
SPD (Security Policy Database)	GSPD (Group Security Policy Database)
PAD (Peer Authorization Database)	GPAD (Group Peer Authorization Database)
IKEv1 (Internet Key Exchange version 1)	GDOI (Group Domain of Interpretation)
IKEv2 (Internet Key Exchange version 2)	GSAKMP (Group Secure Association Key Management Protocol)

5 認証なし IPsec (BTNS WG)

IPsec では、相手認証にデジタル署名認証方式が推奨されており、エンドツーエンドの通信に IPsec を適用するためには、各端末用に公開鍵証明書を発行しなければならず、管理が非常に負担になるという問題がある。このため、この相手認証の処理を行わないことで、容易に IPsec を使えるようにするための検討が開始された。認証の処理を行わないため、通信相手が意図している相手かどうかを確認することはできないが、暗号化やメッセージ認証などの機能は利用することができるという意味で、“Better-Than-Nothing-Security” (無いよりはまし) という名称が付けられている。

また、相手認証の機能は IPsec のレベルで行わず、上位層プロトコルに任せるということも考えられる。例えば、ネットワークファイルシステムのプロトコルである NFSv4 では、認証はアプリケーションレベルで行い、暗号化やメッセージ認証は BTNS を利用することを検討している。

BTNS では、IPsec で定義されているデータベースである PAD と SPD のエントリに、それぞれ“PUBLICKEY”、“BTNS_OK”という新たな識別子を導入することで、認証なしの IPsec 通信を行うためのポリシーを追加している。そして、相手認証を行う IKEv2 において、認証の処理をバイパスさせればよいのであるが、バイパスさせようとすると IKEv2 のプロトコル仕様を変更する必要がある。このため、実際には、認証の処理をバイパスせずに、自己署名証明書や RSA 公開鍵など、認証局 (CA) による証明書発行の処理の必要のない認証情報を送信する。この時、通常のデジタル署名認証方式では、証明書が有効であることを確認するために、ルート証明書などによる公開鍵証明書の正当性の確認を行うが、この処理は行わない。つまり、送信されてきた公開鍵を使用してデジタル署名の検証は行うが、公開鍵の正当性は確認しないため、相手認証をしていないのと結果的には同じということになる。これにより、IPsec や IKEv2 のプロトコルを修正せずに、認証局による公開鍵証明書の発行などの管理の必要なしに、IPsec が利用できるようになる。

6 まとめと今後の展開

2008 年下期に、マルチキャストトラフィックを保護する仕組みと、相手認証処理を省略することで簡易にトラフィックを保護する仕組みが RFC 化された。これにより、NGN (Next Generation Network) などにおけるマルチキャストトラフィックの保護に IPsec を利用することや、証明書発行などの面倒な処理を必要とせずに、エンドツーエンドで IPsec を利用することが可能となる。

今後は、これらの仕組みを実装した製品が出現するのを待つことになるが、マルチキャスト通信を保護する必要性や、エンドツーエンド通信を IPsec レベルで保護する必要性がユーザに認知されるのかもポイントになると考えられる。

以上

9. Resource PKI の動向

木村 泰司

1 はじめに

Resource PKI (以下、RPKI と呼ぶ) は、2011 年と予測されている IPv4 アドレスの在庫枯渇⁷に備えた、IP アドレスの利用権利を担保する PKI として、またインターネット経路制御のセキュリティで使われる PKI として注目されている。

2008 年度は、RPKI のプロトコル策定が行われている IETF SIDR WG のドキュメント整備作業が大きく進み、RPKI の仕様の全体像が見えるようになってきた。また、ライブラリなどの実装を行ってきた APNIC と RIPE NCC では、プロトタイプシステムが完成し、この PKI の利用実験が開始された。

本節では、「PKI 関連およびインターネット経路制御のセキュリティ関連のトピック」として、RPKI を概説し、また国際動向について述べる。

2 RPKI とは何か

RPKI (Resource PKI) とは、アドレス資源⁸の利用権利を担保するための PKI (Public-key Infrastructure) である。RPKI における認証局は、アドレス資源を管理しているインターネットレジストリ運用される。

RPKI を使うと、IP アドレスや AS 番号が正しく割り振り、または割り当てされていることを証明する「リソース証明書」の発行が実現できる。RPKI のための認証局は、技術的には、インターネットレジストリだけでなく、アドレス資源が割り振られた組織が運用することもでき、IP アドレスの再割り振り⁹等が行われても、RPKI が IP アドレスの管理体制から乖離しないようになっている。

3 RPKI の用途

2008 年度の段階では、RPKI の用途は現在 2 つ考えられている。ひとつはアドレス資源の利用権利を示す「リソース証明書」であり、もうひとつはインターネット経路制御のためのセキュリティフレームワークである。以降、各々について述べる。

⁷ IPv4 Address Report

<http://ipv4.potaroo.net/>

⁸ アドレス資源 - アドレス資源とは、IP アドレスや AS 番号といったインターネットに接続するために必要な識別子のことで、インターネットレジストリと呼ばれる、国際的に体系づけられた複数の団体によって管理されている。

⁹ 再割り振り - ISP 等がネットワーク運用を委託している場合などに、IP アドレスの割り振り先組織が、再度割り振りを行うこと。

4 リソース証明書

リソース証明書とは IP アドレスや AS 番号が記載された X.509 形式の電子証明書で、記載されたアドレス資源が正当に割り振られたものであることを証明するものである。リソース証明書には割り振り先組織の名称等は記載されず、アドレス資源そのものが正しいものであることを証明する。

アドレス資源の割り振り・割り当て情報は、インターネットレジストリのデータベース¹⁰に登録されている。しかし IPv4 アドレスの在庫が枯渇する時期になると、この登録情報とは無関係に売買などが行われ、もしくは然るべき移転が行われることなく、不正に使われてしまう危険性が高まると考えられる。アドレス資源が不正に使われると、本来の割り振り先組織（ISP 等）がインターネットとの接続性を失う被害を受けるだけでなく、インターネットで利用できない IP アドレスの維持料を支払わざるを得ないような経済的な損失を被る可能性がある。

IP アドレスが記載されたリソース証明書は、その IP アドレスが正当に割り振られたものであることを証明するため、リソース証明書を提示できない組織の、IP アドレスの利用を差し止める手掛かりとして利用できる。

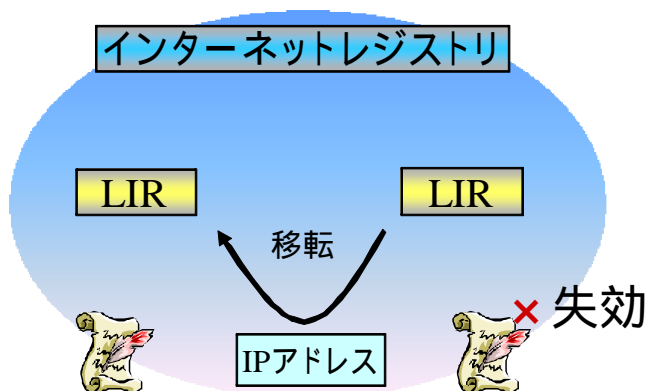


図1 IP アドレスの移管に伴う利用権利の担保

5 インターネット経路制御のためのセキュリティフレームワーク

インターネット経路制御¹¹は基本的に運用者の善意に基づいたシステムである。設定ミスなどにより、偽の IP アドレスの情報が交換されると、インターネットに接続している多くのルータがそれを取り入れてしまい、本来のネットワークとは異なるネットワークに IP パケットが転送されてしまうことがある。これは経路ハイジャックと呼ばれ、例として

¹⁰ 一般的に whois サービスを使うことで閲覧できる。

¹¹ インターネット経路制御 - インターネット経路制御とは、インターネットに接続するルータで行われる処理または手続きの一つで、ISP や大学、企業などの IP ネットワークがインターネットに接続するために行われる。インターネット経路制御では、ルータが BGP などのルーティングプロトコルを用い、IP アドレスや AS 番号の情報を交換する。

前期の報告書で紹介した YouTube の事件がある。

リソース証明書は、経路ハイジャックの中でも影響が大きい prefix hijacking や origin AS の詐称(なりすまし)を防ぐために利用できる。リソース証明書を使って ROA (Route Origination Authorization) と呼ばれる電子署名付きのデータを生成し、Secure BGP と呼ばれるルーティングプロトコルで使うと、不正な経路情報をフィルタリングすることが可能になる。リソース証明書の利用方法としては、Secure BGP の他に、専用の検証サーバを設けてルータに情報提供したり、Internet Routing Registry (以下、IRR) における電子署名のために使われたりすることが考えられている。[1]

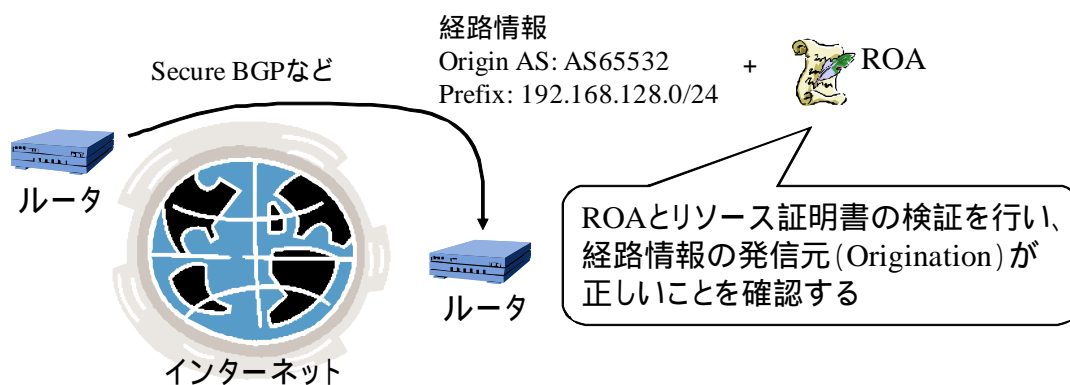


図2 経路ハイジャックを防ぐためのリソース証明書の用途

6 2008 年度の技術動向

RPKI に関わる、2008 年度の技術動向として注目すべきことは、プロトコルの策定動向と、RIR¹²を中心としたプロトタイプシステムの開発の2点である。

7 RPKI のプロトコル策定動向

2008 年度は、ROA のデータフォーマットが具体化し、経路情報との比較方法に至る議論が行われた。ルータ内の処理については、Cisco や Juniper、国内の複数の大手 ISP が議論に参加し始めた。RPKI に関わるプロトコルの策定を行う SIDR WG が設立された2006年4月当時、PKIX WG で策定された RFC 3779 [2]と RPSEC WG で議論されてきた“BGP Security Requirements” [3]に基づいた基本的な議論が行われていた。

8 RIR における RPKI のプログラム開発

2008 年度、APNIC と RIPE NCC は各々のメンバー (IP アドレスの割り振り先組織)

¹² Regional Internet Registry

特定地域内の IP アドレスの割り当て業務を行うレジストリのこと。

<http://www.nic.ad.jp/ja/basics/terms/rir.html>

に対して、RPKI の利用実験を開始した。RIPE NCC のベータテストプログラム¹³は、メンバーでなくても利用でき、利用イメージを共有することで、RIPE コミュニティでの議論を助ける役割を担っている。

これまで、RPKI のプログラム開発は、APNIC を中心として RIPE NCC と ARIN で行われてきた。APNIC では、2007 年度にアーキテクチャの検討やプログラム開発、RPKI のトラストアンカーに関する議論などを、Wiki¹⁴にまとめており、2008 年度の開発はこれらに基づいて行われた。

一方、RIPE NCC は IP アドレス管理業務への影響を調査するためにチームを作るなど、より運用面に着目した現実的な評価を行ってきた。

9 考察

IPv4 アドレスの在庫枯渇への注目度合いが高まるに伴って、不正な IP アドレスの利用についての関心も高まってきた。一方、使われていない IP アドレスを融通する「IP アドレスの移転」が不正に行われることで起こる、IP アドレスハイジャックの危険性の高まりを懸念する声も挙がっている。

これらを踏まえると、アドレス資源の利用権利を担保する RPKI は、IPv4 アドレスの在庫枯渇時期にあって大変重要な位置づけにあるが、RPKI がどのように提供されるべきで、またシンプルかつ稼働していることが重要であるインターネット経路制御において、どの程度依存すべきであるのかを、今後明らかにしていく必要がある。

以上

参考文献

[1] “Securing RPSL Objects with RPKI Signatures”

<http://www.ietf.org/internet-drafts/draft-ietf-sidr-rpsl-sig-00.txt>

[2] RFC 3779, “X.509 Extensions for IP Addresses and AS Identifiers”

<http://www.ietf.org/rfc/rfc3779.txt>

[3] “BGP Security Requirements”

<http://www.ietf.org/internet-drafts/draft-ietf-rpsec-bgpsec-10.txt>

¹³ Certification of Internet Number Resources

<https://certtest.ripe.net/>

¹⁴ The APNIC Resource Certification page – Project Documentation

<http://mirin.apnic.net/resourcecerts/>

10. アイデンティティ管理技術の動向

工藤 達雄

1 はじめに

本報告では、OpenID を中心に、2008 年下半期のアイデンティティ管理技術に関する動向を概観する。

2 概況

筆者の主観ではあるが、2008 年のアイデンティティ管理技術に関する議論の多くは、直接的・間接的に関わらず、OpenID を意識して行なわれてきたように思われる。

その背景には、OpenID の爆発的な普及がある。2007 年 12 月に コア仕様である OpenID Authentication 2.0 が確定して以降、大量の ID 情報を保有するサイトが相次いで OpenID プロバイダ機能の提供を開始し、一方 OpenID に対応したサイト（リライティング・パーティ）も、米 JanRain 社の集計によれば、2008 年の 1 年間で約 3 倍の 31,000 サイト以上に増加しているという[1]。

そして普及に伴い、OpenID の利活用を進める上での仕様の改版・拡張や、実運用での工夫などに進展がみられる。

3 OpenID 仕様の動向

3.1 PAPE（Provier Authentication Policy Extension）仕様の確定

前回の報告において、PAPE 仕様を策定するためのワーキング・グループ（WG）が結成されたことを述べたが、その後、2008 年 12 月、OpenID Foundation 会員の投票を経て、PAPE バージョン 1.0 が OpenID 仕様として承認された[2]。

PAPE は OpenID Authentication に対する拡張仕様であり、OpenID プロバイダとリライティング・パーティとの間での、OpenID Authentication プロセスにおける認証ポリシーの要求・表明の手順を定義している。手順の大まかな流れは以下の通り。

- (1) OpenID プロバイダは、自身がサポートする認証ポリシーを、エンドユーザの XRDS 文書に記述し、リライティング・パーティに広告(advertise)する。以下の例では、OpenID プロバイダは当該エンドユーザを、フィッシング耐性のある(phishing-resistant)方法によって認証を行なうことが広告されている。

```
<xrd>
  <Service>
    <Type>http://specs.openid.net/auth/2.0/signon</Type>

  <Type>http://schemas.openid.net/pape/policies/2007/06/phishing-resistant</Type>
    <URI>https://example.com/server</URI>
  </Service>
</xrd>
```

あるいは以下の例のように、OpenID プロバイダがどの保証フレームワークに準拠するかを、独自の「名前空間」として広告することも可能である。

```
<xrd>
  <Service>
    <Type>http://specs.openid.net/auth/2.0/signon</Type>

  <Type>http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf</Type>
    <URI>https://example.com/server</URI>
  </Service>
</xrd>
```

OpenID Authentication のプロセスにおいては、リライディング・パーティは OpenID プロバイダのエンドポイントの場所を入手するために、エンドユーザが指定した User-Supplied Identifier (「自身を識別する URL/XRI」あるいは「OpenID プロバイダの識別子」) に基づき XRDS 文書を取得する。このとき XRDS 文書に「OpenID プロバイダがサポートする認証ポリシー」が記載されていることで、リライディング・パーティは、エンドユーザに関するアサーションをどの OpenID プロバイダにリクエストするかを、認証ポリシーに基づいて判断することができるようになる。

- (2) リライディング・パーティは OpenID プロバイダに OpenID 認証リクエストを行なう際、そのメッセージのパラメータとして、アサーション発行時に適用してほしい「認証ポリシー」や「保証フレームワークの名前空間」などを含める(PAPE リクエスト)。
- (3) 認証リクエストを受け取った OpenID プロバイダは、認証ポリシーに従い、ユーザを認証する。

- (4) OpenID プロバイダは認証結果（アサーション）をリライティング・パーティに返却する際、そのメッセージのパラメータとして、アサーション発行時に適用した「認証ポリシー」や、「保証フレームワークの名前空間」、「実際の保証レベル」などを含める（PAPE レスポンス）。なお OpenID プロバイダは、リライティング・パーティからの PAPE リクエストの有無にかかわらず、PAPE レスポンスを返却することができる。
- (5) リライティング・パーティは OpenID プロバイダからの PAPE レスポンスの内容、すなわち適用された認証ポリシーを判断し、エンドユーザからのアクセスの可否を決定する。

3.2 OpenID Authentication 仕様の次期バージョンの策定作業が開始予定

2008 年 11 月、OpenID Authentication の次期仕様（バージョン番号は 2.1）を策定するための WG 開設が提案された[3]。

OpenID Authentication 2.1 は現行のバージョン 2.0 との後方互換を前提に、現仕様の「バグフィックス（語句の修正、図版の追加、読みやすさの向上、解釈のぶれの防止）や、セキュリティ・ガイドラインの追加、ディスカバリ機能の強化（XRD（Extensible Resource Descriptor）対応）、ひとつの識別子に対する複数 OpenID プロバイダの記述の許容）などが行われる予定となっている。

3.3 拡張仕様の新規提案

コアとなる OpenID Authentication 仕様に対し、オプションとなる「拡張仕様」に関しても、WG を設立し、策定しようという動きがいくつかある。予定されている主な拡張仕様は次のふたつである。

- OpenID and OAuth Hybrid Extension：ブラウザベースの ID 連携である OpenID と、ユーザのアクセス権限を考慮したサービス連携である OAuth とを組み合わせるための拡張。これにより、「OpenID による ID 情報（認証結果や属性）の提供」と「OAuth によるアクセス権限の委譲」に関するユーザの同意確認が同時に実施できるようになる[4]。
- Contract Exchange Extension：任意のパーティ間で「契約書案（contract proposal）」および、さらに双方が電子署名した「契約（contract）」を交換するための拡張。これにより、OpenID プロバイダとリライティング・パーティとの間で、重要な情報（例：クレジットカード番号や個人識別情報）の要求・提供を、あらかじめ合意された契約に基づいて行なうことができるようになる[5]。

この他、AX (Attribute Exchange) Extension 2.0 (現行の AX 1.0 の更新)、SREG (Simple Registration) Extension 1.1 (現行の SREG 1.0 の更新)、ディスカバリ関連仕様の検討などのための WG が開設される見込みである [6]。

3.4 OpenID の実運用の動向

OpenID Authentication 2.0 仕様により導入された「ディレクテッド・アイデンティティ」によって、リライング・パーティは、エンドユーザに自身の識別子 (URL もしくは XRI) を入力してもらう代わりに、主要な OpenID プロバイダをエンドユーザに提示し、その中から選択してもらうことができるようになった。現在では多くのリライング・パーティが、この後者の手法を用いているとみられる。

一方の OpenID プロバイダ側では、リライング・パーティ間での名寄せを防ぐため、リライング・パーティ毎に異なる「仮名」を払い出す運用が一部で広まっている。この方法は以前から OpenID.ee (エストニア eID を用いた OpenID プロバイダ) が採用していたが、2008 年 10 月に OpenID プロバイダのサービスを開始した Google および Microsoft Windows Live ID (現時点では試験運用) が、相次いで同様の仮名払い出しをサポートしている [7][8]。このような「メガ OpenID プロバイダ」の動きは、他の OpenID プロバイダや、リライング・パーティにも影響を与えるものとみられる。

また 2008 年 8 月にミクシィが OpenID プロバイダのサービスを開始したが、同サービスでは一般的なユーザ認証に加え、エンドユーザが特定のグループに属しているかどうかの確認を OpenID Authentication を用いて行なう、「マイミクシィ認証 (ある別のエンドユーザの友人かどうか)」および「コミュニティ認証 (あるコミュニティに属するかどうか)」という機能も提供している。

4 まとめ

2008 年下期から、PAPE の確定や Contract Exchange WG の提案など、OpenID をベースに重要な情報を流通させるための仕様策定の動きが加速している。また実運用においても、リライング・パーティ間での名寄せを防止するための仮名払い出しを大手 OpenID プロバイダが実装するなど、プライバシー保護への考慮も進みつつある。従来は「ブログへのコメント」のようなカジュアルな利用が一般的であった OpenID であるが、今後、より広い分野への適用が期待される。

以上

参考文献

- [1] Janrain Blog: Relying Party Stats as of Jan 1st, 2009
<http://blog.janrain.com/2009/01/relying-party-stats-as-of-jan-1st-2008.html>

- [2] Final: OpenID Provider Authentication Policy Extension 1.0
http://openid.net/specs/openid-provider-authentication-policy-extension-1_0.html

- [3] OpenID Wiki / OpenID_Authentication_2_1
http://wiki.openid.net/OpenID_Authentication_2_1

- [4] OpenID Wiki / OpenID and OAuth Hybrid Extension
<http://wiki.openid.net/OpenID-and-OAuth-Hybrid-Extension>

- [5] OpenID Wiki / Working_Groups:Contract_Exchange_1
http://wiki.openid.net/Working_Groups%3AContract_Exchange_1

- [6] OpenID Wiki / Working Groups
<http://wiki.openid.net/Working-Groups>

- [7] OpenID for Google Apps - Account Authentication API - Google Code
<http://code.google.com/intl/ja/apis/accounts/docs/OpenID.html>

- [8] Windows Live ID Becomes an OpenID Provider - Windows Live
<http://winliveid.spaces.live.com/Blog/cns!AEE1BB0D86E23AAC!1745.entry>

- [9] mixi Developer Center (ミクシィ デベロッパーセンター)
OpenID Membership Authentication Method Draft-1
<http://developer.mixi.co.jp/draft/openid-membership-authentication-method-draft-1>

