ウィルスのないインターネットに向けて
～オペレーティングシステムからのアプローチ～

河野　健二

1. 研究のねらい

インターネットの普及にともない、様々なコンテンツがインターネット上で配布されるようになっており、実行可能コンテンツと呼ばれる一種のプログラムを配布することも一般的になっている。実行可能コンテンツとは、受取り側の計算機環境で実行されるコンテンツであり、プラグイン、Active X、Java applet などがある。また、メールに添付して送られる PDF やワード形式のドキュメントも一種の実行可能コンテンツであり、Acrobat Reader や Microsoft Word などの閲覧プログラムによって解釈・実行されるものである。

インターネットを用いると、不特定多数のユーザが自由にコンテンツを配布できるため、悪意のあるユーザがウィルスやワーム等の攻撃用プログラムを仕込んだ実行可能コンテンツを配布することも容易になってしまっている。従来と同じウィルスであっても、インターネットを媒体とすればより広範囲により迅速に広めることができ、フロッピーディスク等の物理的なメディアを媒体として広まるウィルスよりもはるかに甚大な被害を与えることができる。

これまでに行われてきたウィルス対策は、ワクチンソフトによるものが主である。ワクチンソフトでは、既知のウィルスを類型化したデータベースを作成し、そのデータベースを参照してウィルスの検出を行っている。そのため、データベースの更新を怠ってはワクチンソフトも役に立たず、その上、まったく未知のウィルスに対してはその原理上ほとんど無力である。ワクチンソフトから身を隠すための手法も年々巧妙になっており、それを追いかける形でいたちごっこのようにワクチンソフトが開発されている。

本研究のねらいは、未知のウィルスに対しても有効に機能する保護機構を実現することにある。従来のオペレーティングシステムは、インターネットのような高い開放性を有する環境を想定して設計されたものではないため、悪意のあるプログラムに対してはきわめて脆弱である。本研究では、オペレーティングシステムの基本的な設計手法から見直すことによって、ウィルスなどの悪質なソフトウェアに対する耐性の強いオペレーティングシステムを実現することを目指している。

2. 研究成果

本研究の成果は、1) 実行可能コンテンツを安全に実行するためのさまざまな要素技術を確立した点と、2) 高い堅牢性を有するインターネット・サーバの実現手法を提案した点のふたつに要約できる。第二の成果は、安全性の高いオペレーティングシステムの実現法を考察する中で、副産物として得られた成果である。以下、それぞれの成果についてまとめる。

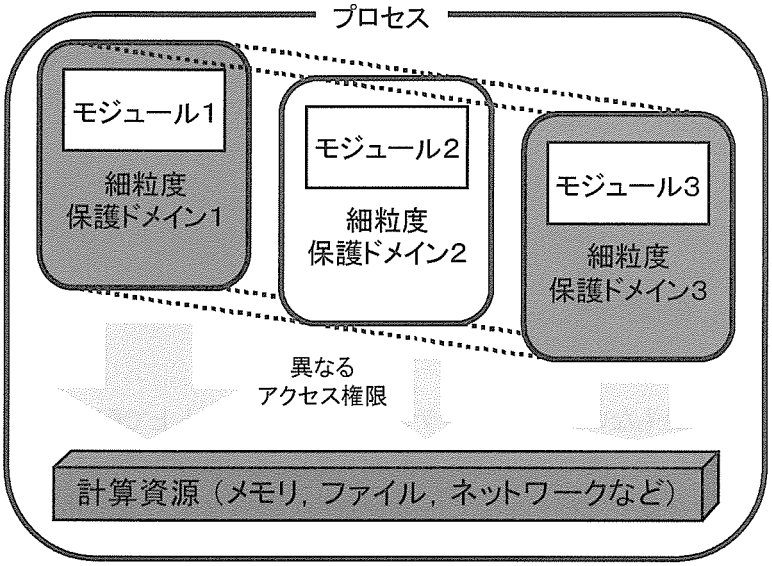

図 1 ： 細粒度保護ドメインによるプロセス・モデル

2．1．細粒度保護ドメインとその適用

インターネットからダウンロードした実行可能コンテンツによる不正攻撃には、(1) 不正な資源アクセスによる攻撃と (2) 資源の濫用による攻撃のふたつの攻撃形態があることが知られている。不正な資源アクセスによる攻撃とは、何らかの手段を用いて、実行可能コンテンツがアクセスしてはならない資源にアクセスする攻撃である。たとえば、管理者権限で動作するネットワークサーバのセキュリティ・ホールを突き、パスワードファイルを入手するといった攻撃である。資源の濫用による攻撃とは、大量の計算資源を消費することによって、他のプログラムの実行を妨げる攻撃である。たとえば、大量のメモリを消費し、他のプログラムをメモリ不足で実行できない状態に陥れるような攻撃である。本節では、前者の不正な資源アクセスによる攻撃に対する防御手法に関して得られた成果をまとめる。

2．1．1．実行可能コンテンツの分類

実行可能コンテンツは、その実行形式の違いから (1) プラグイン形式と呼ばれる形式と、(2) ヘルパアプリケーション形式と呼ばれる形式のふたつに分類できる。プラグイン型とは、別のアプリケーションにリンクして実行される形式のコンテンツであり、ウェブブラウザのプラグインや Active X などがある。それに対し、ヘルパアプリケーション形式のコンテンツとは、別のアプリケーションによって解釈・実行される形式のコンテンツである。たとえば、PDF 形式や PS 形式のドキュメントは、それぞれ Acrobat Reader や Ghostscript によって解釈・実行される実行可能コンテンツである。

プラグイン形式のコンテンツは、アプリケーションにリンクして実行されるため、従来のオペレーティングシステムでは、常にアプリケーションと同じアクセス権限で実行される。そのため、極めて容易に不正な資

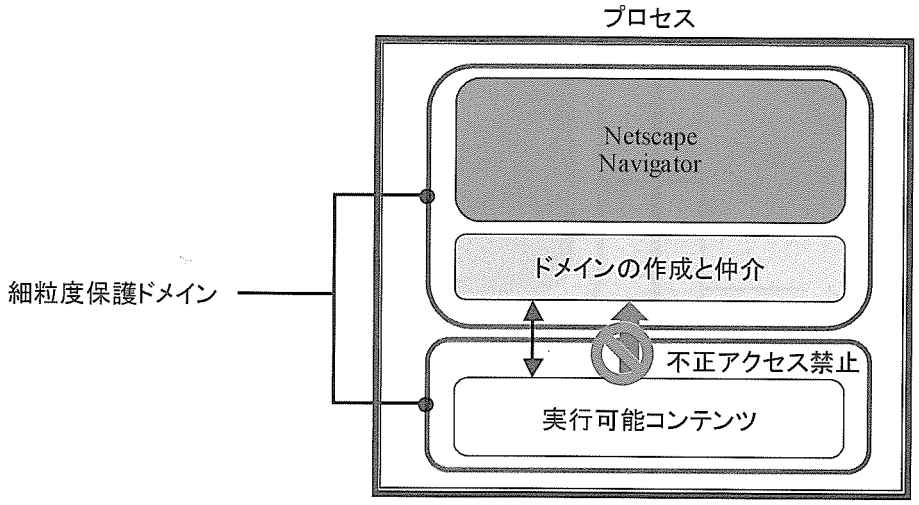

図２： プラグイン型コンテンツの安全な実行

源アクセスを行なうことができる。本研究で確立した保護機構を用いると、プラグイン型の実行可能コンテンツに対し、アプリケーション本体よりも制約のきつい保護を行なうことができ、プラグイン形式の実行可能コンテンツであってもより安全に実行することができる。

ヘルパアプリケーション形式のコンテンツは、ヘルパアプリケーションによって解釈・実行されるため、プラグイン形式のコンテンツに比べ、不正な資源アクセスを行なうことは難しい。しかしながら、ヘルパ・アプリケーション自体のセキュリティ・ホールを突くコンテンツを意図的に作成することができ、結果としてヘルパアプリケーションの持つ権限をのっとることができることが報告されている。

２．１．２．細粒度保護ドメイン

細粒度保護ドメインは、一つのプロセスの中に複数個持つことができる保護ドメインである。保護ドメインとはアクセス可能な資源の集合を表す概念であり、従来のオペレーティングシステムでは、プロセスが保護ドメインの役目を兼ねている。細粒度保護ドメインはプロセスによる保護ドメインの部分集合として定義される。つまり、細粒度保護ドメインでアクセス可能な資源は、それが定義されたプロセスでアクセス可能な資源の中の一部になっている。

細粒度保護ドメインでは、アクセス制御を細かい単位で行なうことができる。例えば、メモリに対するアクセスは、ページ単位で細粒度保護ドメイン毎に異なる保護モードを設定することができる。また、ファイルやネットワークなどのシステム資源へのアクセスも任意の単位でアクセス権限を設定することができる。システム資源に対するアクセス制御の具体的な方式はオペレーティングシステムでは規定しておらず、プロセス毎にユーザレベルで実装されるポリシーモジュールと呼ぶコードにアクセス制御を委任する。従って、アプリケーションに合わせて様々な保護ポリシーを実現できる。

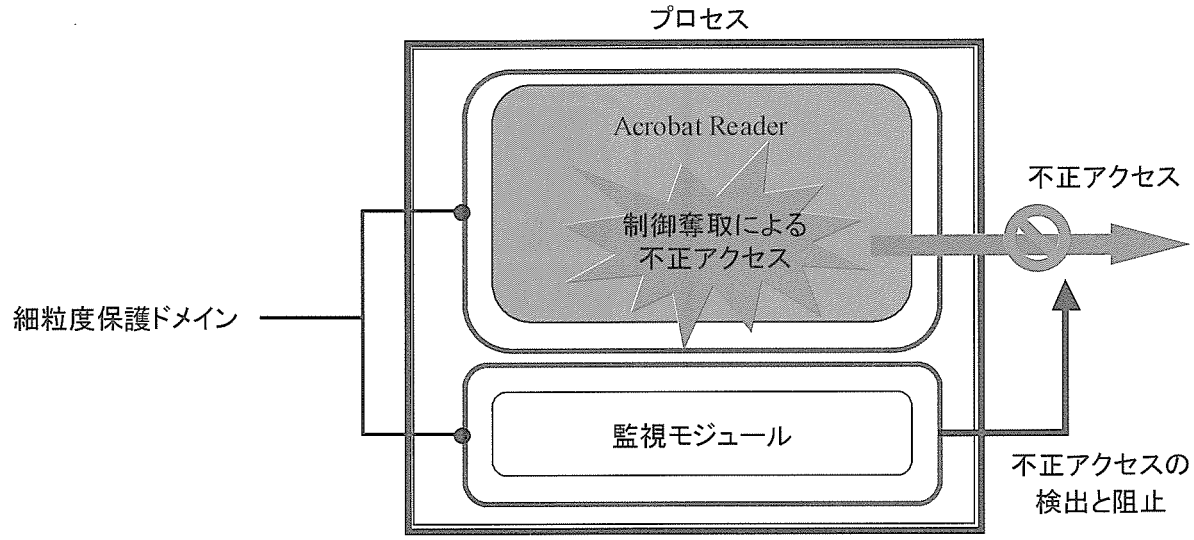

図 3 ： ヘルパアプリケーション型コンテンツの安全な実行環境

2．1．3．プラグイン型コンテンツの安全な実行環境

細粒度保護ドメインを用いると、プラグインやActive Xなどの実行可能コンテンツを安全に実行することができる。なぜなら、プラグインなどの実行可能コンテンツを細粒度保護ドメインの中に閉じ込めて実行することによって、悪意のあるコンテンツが不正アクセスを試みても、それを細粒度保護ドメインのレベルで阻止することができるからである。

本研究では、実際に Netscape Navigator を拡張し、Active X などの実行可能コンテンツを細粒度保護ドメインに閉じ込めて実行できる環境の構築を行った。図 2 に示すように、Netscape Navigator の持つ拡張機能を利用し、細粒度保護ドメインの中に実行可能コンテンツを閉じ込めるためのコードの追加を行なった。これによって、Netscape Navigator 本体に与えられた権限よりもさらに制限された権限の下でコンテンツを実行することができるようになり、悪質な実行可能コンテンツによる不正アクセスからローカルな計算機環境を保護することができるようになる。

2．1．4．ヘルパアプリケーション型コンテンツの安全な実行環境

ヘルパアプリケーション型のコンテンツは、PDF 形式のドキュメントなど他のプログラム(PDF の場合は Acrobat Reader)によって解釈・実行される形式のコンテンツである。プラグイン型のコンテンツとは異なり、一見したところヘルパアプリケーション型のコンテンツは直接実行してしまってもセキュリティ上の問題はないように思われる。しかし、コンテンツを解釈・実行するヘルパアプリケーションのセキュリティ・ホールを巧みに攻撃する悪質なコンテンツを作成できることが知られており、Acrobat Reader, Ghostscript, PowerPoint などのヘルパアプリケーションに脆弱性があることが報告されている。

細粒度保護ドメインを用いると、脆弱性のあるヘルパアプリケーションであっても安全に利用することが

できるようになる。図 3 に示したように、ヘルパアプリケーション自身を細粒度保護ドメイン内に閉じ込めてやることによって、悪質なコンテンツによってアプリケーションが乗っ取られてしまっても、その不正アクセスを検出し、未然に防ぐことができるようになる。本研究では、実際にヘルパアプリケーションを細粒度保護ドメイン内に閉じ込めて実行することのできる環境を構築し、Acrobat Reader や Ghostscript などのヘルパアプリケーションを安全に実行できることを確認した。実際、Acrobat Reader の脆弱性を突いて Acrobat Reader の制御を乗っ取る PDF ファイルを作成し、その PDF ファイルを処理する実験を行った。この実験の結果、悪質なコンテンツを処理してもローカルの計算機環境に何ら被害が及ばないことが確認できた。

2．1．5．特権プログラムの安全性向上

UNIX システムにおける setuid プログラムは、以前から不正アクセスの手段として狙われやすいことが知られている。特に所有者が root（管理者）である setuid プログラムは、実行したユーザに関わらず管理者権限で動作するため、脆弱な setuid プログラムのバグを突いて制御を乗っ取る手法などにより、管理者権限を不正に取得できてしまう。このような不正アクセスに対して脆弱な setuid プログラムの報告は現在でも跡を絶たず、2001 年には CERT/CC Vulnerability Notes で 5 件、SecurityFocus Vulnerabilities では 65 件に上っている。

所有者が root である setuid プログラムはプログラム全体が root 権限で動作するが、実際に root 権限が必要なのはプログラム中の一部のコードのみであることが多い。例えばユーザのパスワードを変更する passwd コマンドでは、root 権限が必要なのはパスワードファイルを編集するコードのみである。一方、不正アクセスの手段となる脆弱性は，コマンドライン処理やユーザの入力処理などのコードに多く存在し，root 権限が不要である場合が多い。従って、root 権限で動作するコードを必要最低限のコードにのみ制限することによって、setuid プログラムが不正アクセスの手段となる可能性を減らすことができる。

これまでにも、プログラム全体の権限を制限するサンドボックスなどの機構は数多く提案されている。しかしサンドボックスではプログラム中の一部のコードに対してのみ権限を制限することは難しい。setuid プログラムを安全に書くためのガイドラインなども知られているが、近年のプログラムの巨大化・複雑化にともなって、プログラム全体を安全に書くことはますます困難になってきている。

本研究の成果である細粒度保護ドメインの機構を利用すると、setuid プログラム中の root 権限で動作するコードを最小化することができる。この手法では、細粒度保護ドメインの機構を用いて root 権限が必要なコードにのみ root 権限を与え、root 権限が不要なコードは一般ユーザの権限で動作させる。これによって、root 権限が不要なコードを利用した管理者権限の不正取得を防止することができる。

2．2．資源横取り方式による資源濫用攻撃からの防御

不正なプログラムによる攻撃の一形態に、計算資源を意図的に占有し、他のプログラムが処理を継続できないようにする攻撃が知られている。このような形態の攻撃を「資源濫用によるサービス拒否攻撃」という。資源濫用によるサービス拒否攻撃は現実の脅威であり、たとえば、大量の資源を消費するアプレットを作成し、それをウェブ経由で配布することが簡単にできる。

本研究では優先度付き資源管理機構を提案し、各プロセスの資源優先度を制御するだけで、資源を濫用するプロセスから資源を奪い返したり、資源濫用の危険性のあるプロセスをあらかじめサンドボックスに閉じ込めて、その影響を押さえることができる。たとえば、アプレットを実行する Java 仮想機械をサンドボックスに入れておけば、大量の資源を消費するアプレットを起動してもウェブブラウザはその影響を受ける

ことがない。また、サンドボックスの外で動作するプロセスが資源を濫用し、計算機がフリーズした場合でも、そのプロセスの資源優先度を実行時に低く設定しなおすことによって、攻撃を仕掛けているプロセスから資源を奪い返すことができる。

2. 2. 特定領域言語による安全性の高いインターネット・サーバの実現法

情報交換や情報共有の基盤としてインターネットが広く利用されるようになるにつれて、インターネット上でサービスを提供する様々なアプリケーションが次々と開発されている。インターネット上で動作するこれらのアプリケーションは、アプリケーション層プロトコルを用いて通信を行う。アプリケーション層プロトコルとはトランスポート層の上位プロトコルであり、各アプリケーションに固有の通信規約である。アプリケーション層プロトコルには、ウェブページの閲覧に用いるHTTP、メールの送受信に用いるSMTPやPOPなど多くのプロトコルがある。これらのプロトコルは時系列に沿ったメッセージのやり取りとして定義されており、それぞれのメッセージは文字列を主体として構成されている。たとえば、POPにおける認証では、クライアントからサーバに USER kono¥r¥n という文字列を送信し、それに対し+OK¥r¥n または-ERR¥r¥n という文字列が返信される仕組みになっている。

こうしたメッセージのやり取りを実現するプロトコル処理部は、煩雑で退屈なプログラミングを要する。それぞれのメッセージが文字列を主体として構成されているため、メッセージの解析や構成、文字列とデータ構造の相互変換など、多くの文字列処理を要する。こうした文字列処理は、C 言語等の標準ライブラリで提供されている文字列処理関数を用いて実装されており、些細なプログラミング上の誤りを犯しやすい。プロトコル処理における些細な誤りは、しばしば致命的なセキュリティ・ホールにつながることはよく知られている。 また、こうしたプロトコル処理はシステムコール・レベルでの通信プリミティブを用いて実装されていることが多く、オペレーティングシステムに関する詳細な知識を必要とし、プロトコル処理部の可搬性を低下させる一因となっている。

本研究では、クライアント・サーバ型のアプリケーション層プロトコルを対象に、プロトコル処理部を自動生成するコード生成器 April の実現を行った。April はプロトコルの仕様を与えると、そのプロトコルを処理するC言語のコードを自動的に生成する。生成されたコードはRPCやRMIにおけるスタブと同様の役割を果たし、アプリケーション・プログラマは C 言語の関数を呼び出すだけで、メッセージの送受信を行なうことができる。通信プリミティブの扱いや煩雑な文字列処理は自動生成されたコード中に隠蔽される。そのため、アプリケーション・プログラマはプロトコル処理に特有の煩雑で単調なプログラミングから解放され、より本質的なプログラミングに専念できるようになる。

April におけるプロトコルの仕様は、1)メッセージ・フォーマットの定義、2)時系列に沿ったメッセージのやり取り、3)システムの状態遷移のみを記述すればよい。プロトコルの仕様を記述する April 言語は、アプリケーション層プロトコルの記述に特化した記法を持ち、様々なアプリケーション層プロトコルの仕様を宣言的に記述することができる。実際、April 言語を用いてPOP、HTTP、SMTPなどのプロトコルの記述を行った。また、POPサーバのひとつであるqpopperのプロトコル処理部をAprilによって自動生成されたコードに置き換え、元来のqpopperとの性能比較を行った。この性能比較では、Aprilによって自動生成されたプロトコル処理部のオーバーヘッドは無視できる程度であった。

3. 今後の展開

細粒度保護ドメインという新しい保護機構の有用性はある程度検証できたと考えており、今後は本研究の成果を実用化すべく一般に還元していきたい。細粒度保護ドメインの機構を拡張した Linux カーネルは、現在一般公開のためのライブラリの整備やドキュメントの作成などを進めている。一般公開を通じて実用化する上での問題点や、新しい適用分野などを見出していきたいと考えている。

資源濫用攻撃に対する防御手法は、まだ考えるべき点が多く残されている。これは、従来のオペレーティングシステムにおける資源管理手法が、現在の複雑なアプリケーションの構成法とは乖離したものになってしまっており、小手先の改良では対処しきれない問題が多く見出されたためである。これは新しい研究のシーズを見つけたとも言え、今後、さらに考察を深めていきたい。

4. 成果リスト

1. T. Shinagawa, K. Kono, and T. Masuda: Fine-grained protection domains based on segmentation mechanism, Presented at JSSS SPA 2000.
2. 金子 済、河野 健二、益田 隆司: 悪意ある外部プログラムによるリソース濫用の防止 ― ファイルキャッシュのケーススタディ ―、情報処理学会研究会報告 (2000-OS-43), pp. 205 -- 212, 2000.
3. Kenji Kono and Takashi Masuda: Efficient RMI: Dynamic Specialization of Object Serialization, In Proceedings of IEEE International Conference on Distributed Computing Systems, pp. 308- 315, 2000.
4. T. Shinagawa, K. Kono and T. Masuda: Exploiting segmentation mechanism for protecting against malicious mobile code, Technical Report (TR00-02), Department of Information Science, University of Tokyo.
5. 河野 健二、益田 隆司 : オブジェクト整列化の動的特化による効率的な RMI の実現、情報処理学会論文誌、41 巻 10 号、pp. 2916 - 2925, 2000.
6. 品川 高廣、河野 健二、益田 隆司 : Web ブラウザのための安全なプログラム実行環境の実現、情報処理学会研究会報告 (2001-OS-87), pp. 121 - 128, 2001.
7. 金子 済、河野 健二、清水 謙太郎 : メモリ占有 DoS 攻撃の防止: 優先度付きメモリ管理、情報処理学会研究会報告 (2001-OS-87), pp. 113 -- 120, 2001.
8. 品川 高廣、河野 健二、益田 隆司 : ヘルパアプリケーションの安全な実行環境、情報処理学会研究会報告 (2001-OS-88), pp. 67-- 74, 2001.
9. 品川 高廣、河野 健二、益田 隆司 : 実行可能コンテンツの安全な実行環境、日本ソフトウェア科学会 SPA 2002.
10. 品川 高廣、河野 健二、益田 隆司 : 実行可能コンテンツの安全な実行環境、情報処理学会論文誌、43 巻 6 号、pp. 1677-- 1689, 2002.
11. 品川 高廣、河野 健二、益田 隆司 : 細粒度保護ドメインによる軽量なサンドボックスの実現、情報処理学会研究会報告 (2002-OS-90), pp. 87-- 94, 2002.
12. Luciano P. Barreto, Gilles Muller, Julia L. Lawall, and Kenji Kono: A framework for simplifying the development of kernel schedures: Design and performance evaluaton, Technical Report (02/8/INFO), Ecole des Mines de Nantes.

13. Wataru Kaneko, Kenji Kono, and Kentaro Shimizu: Preemptive Resource Management: Defending against Resource Monopolizing DoS, Submitted for publication.
14. Kenji Kono: Exploiting Dynamic Specialization for Efficient RMI, Submitted for publication.
15. 河野 健二 : アプリケーション層プロトコルの実現を容易にするフレームワーク、投稿中.
16. 品川 高廣、河野 健二、益田 隆司 : 細粒度保護ドメインを用いた setuid プログラムの特権コード最小化、日本ソフトウェア科学会 第 19 回大会論文集。
17. Takahiro Shinagawa, Kenji Kono, and Takashi Masuda: Minimizing privileged code in setuid programs, submitted for publication.
18. Kenji Kono: April: An IDL for the programming of application-layer protocols, submitted for publication.