RB-002

Hussa: スケーラブルかつセキュアなサーバアーキテクチャ ～低コストなサーバプロセス実行権限変更機構～

Hussa: Hugely Scalable and Secure Server Architecture for Shared Servers – A Low-cost Mechanism for Changing Runtime Privilege of Server Processes –

原 大輔 †, ‡ HARA, Daisuke
中山 泰一 † NAKAYAMA, Yasuichi

1. まえがき

近年，個人でウェブサイトを公開するユーザが増加している．サイトの設置場所として，サイト専用のサーバを用意する場合と，共有型ホスティングサービス等の複数サイトで計算機を共用する環境を利用する場合に大別される．前者は計算機リソースを自由に利用可能というメリットがあるが，運用コストが高く，個人で利用することが困難なケースが多い．一方，後者は自由度や計算機リソースの容量が制限される代わりに低価格で利用することができるため，個人の利用に適しており，広く利用されている．複数サイトで計算機を共用する場合，従来の UNIX のパーミッションモデル “owner/group/other” と同一権限で動作するウェブサーバの組合せにより，計算機を共用する他のユーザからコンテンツファイルを盗視・改竄される危険性がある．

本研究では，ウェブサーバがリクエストを受信した際，(1)seteuid(), setegid() システムコールを用いて，サーバプロセスの実効ユーザ ID/実効グループ ID をサイト毎に異なる一般ユーザに変更した後リクエスト処理を実施 (2) ユーザのスクリプトが root 権限を奪取できないよう setuid(), setgid() 系のシステムコールを本システムのみから実行可能とすることでこの問題を解決する．seteuid(), setegid() システムコールを用いることで，プロセスを終了することなく，複数回実行権限を変更することができ，性能面で有利である．

本システムの基礎性能を評価した結果，Apache と比較して平均 0.5 %，最大 4.7 %の性能低下であり，実用に耐え得る性能を達成していることを確認した．

以下，2 章で実行権限に関する既存技術について述べる．また，3 章で本稿で提案するシステムの設計について述べ，4 章で本システムに対して行った評価結果について報告する．最後に 5 章で本稿をまとめる．

2. 実行権限に関する既存技術

本章では，UNIX 系 OS での実行権限及び，POSIX ACL，セキュア OS，コンテナ/VM，そして，我々がこれまでに提案してきたウェブサーバシステムである Harache/Hi-sap について述べる．

2.1 UNIX 系 OS での実行権限

従来の UNIX 系 OS には，プロセスの実行権限を変更するシステムコールとして実ユーザ ID/実効ユーザ ID を変更する setuid() 系 (setuid(), seteuid(), setreuid(), setresuid()) と実グループ ID/実効グループ ID を変更する setgid() 系 (setgid(), setegid(), setregid(), setresgid()) がある．root 権限で動作するプロセスが setuid(), setgid() システムコールを実行して一般ユーザの実行権限に遷移する処理は不可逆であり，それ以降，元の root 権限に戻ることはできない．一方，seteuid(), setegid() システムコールは実ユーザ ID は変更せずに実効ユーザ ID のみ変更するため，可逆である．

UNIX 系 OS のプロセスのうち，ユーザ毎に異なる実行権限で動作するものは通常 root 権限で生成された後，setuid(), setgid() システムコールを実行することで実ユーザ ID/実効ユーザ ID が各一般ユーザへ変更され動作する．例えば，シェルのプロセスはログインしたユーザ毎に異なる実行権限で動作する．一方，Apache [1] 等のウェブサーバのプロセスは伝統的に一律専用の一般ユーザ§(専用ユーザ) の実行権限で動作する．同一の計算機にアカウントを持つ複数のユーザがウェブサイトを開設する場合，各ユーザは各自のホームディレクトリ配下に各自が所有者であるコンテンツファイルを配置した上で，専用ユーザの権限で動作するウェブサーバのプロセスが各ファイルを読み書きできるように，UNIX のパーミッションモデルの other に read/write 権限を付与する必要がある¶(図 1(0))．other に権限を付与することの弊害は，計算機を共用する他のユーザからファイルを盗視・改竄される危険性が存在することである (図 1(1))．このような悪意のあるユーザは計算機にログインして直接 cp, rm 等のコマンドを実行したり (図 1(1-1))，自身のサイトにスクリプトを設置してスクリプト経由でコマンドを実行する (図 1(1-2)) ことでファイルを盗視・改竄する．

2.2 POSIX ACL

POSIX ACL [2] は，UNIX のパーミッションモデルに加え，ユーザ単位でのアクセス制御を提供する．POSIX ACL を用い，コンテンツファイルの read/write 権限を other ではなく，専用ユーザに付与することで，直接 cp, rm 等のコマンドを実行されることによるファイルの盗視・改竄を防ぐことが可能となる．

また，通常，スクリプトは専用ユーザの権限で動作するため，スクリプトによる盗視・改竄は POSIX ACL 単体で防ぐことはできない．但し，スクリプトのうち，CGI は suEXEC [1, 3, 4] を組み合わせることでサイト毎に専用ユーザとは異なる権限で実行され，盗視・改竄を防ぐことが可能となる．suEXEC では，サーバプロセスが fork() システムコールにより子プロセスを生成し，

†電気通信大学 情報工学科
Department of Computer Science,
The University of Electro-Communications
‡現在，日本電信電話株式会社

§ウェブサーバ専用に用いられ，ログインシェルを持たない等の特徴がある．*apache*, *www-data*, *www* 等の名前が付けられている．

¶HTML，画像ファイル等は read 権限のみでよいが，wiki のデータファイルやログファイルは write 権限が必要．

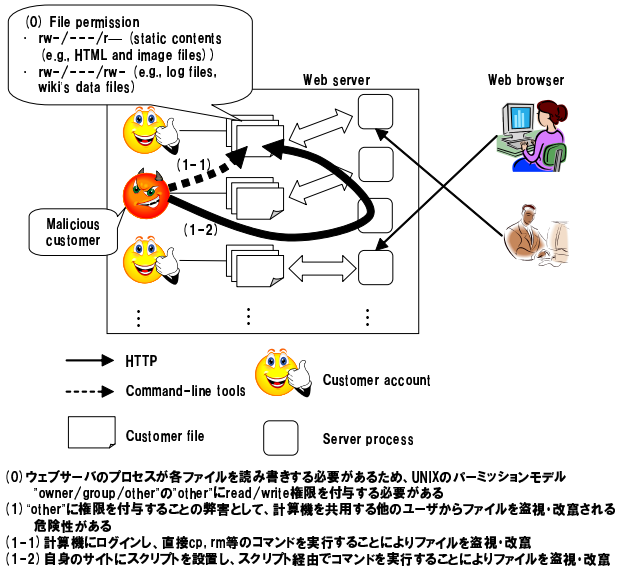

図 1: サーバプロセスが同一の実行権限で動作することの弊害

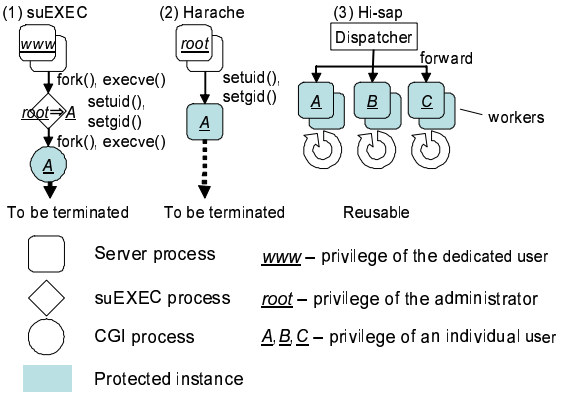

図 2: suEXEC, Harache, Hi-sap 概要

その子プロセスが “setuid bit” が設定されたバイナリを execve() システムコールにより root 権限で実行する．そのバイナリが setuid(), setgid() システムコールにより実行権限を変更した後，さらに子プロセスを生成して CGI を実行する (図 2(1)) ため，プロセス生成及びプログラム実行が 2 回必要となり，性能が低いという欠点がある．一方，Apache 等のウェブサーバのプロセスにスクリプト言語のインタプリタを内蔵し，スクリプト (組込みスクリプト) を高速に実行する組込みインタプリタ [5, 6, 7, 8] を用いる場合，組込みスクリプトは専用ユーザの権限で動作するため，盗視・改竄を防ぐことはできない．

2.3 セキュア OS

セキュア OS [9, 10, 11] は，強制アクセス制御 [12] と最小特権 [13] の機能を有する OS である．強制アクセス制御は全てのユーザとプロセスに例外なくアクセス制御を実施する機能であり，最小特権はユーザとプロセスに動作上必要最小限の権限を与えるというポリシである．従来の UNIX 系 OS は root 権限により OS の全ての操作が可能であったが，セキュア OS を用いることで，root 権限を制限することが可能となる．よって，サーバソフトウェアのセキュリティホールや設定誤り等により root 権限が奪取された場合でも，被害を最小限に抑えることができる．

セキュア OS 単体で，直接 cp, rm 等のコマンドによるファイルの盗視・改竄を防ぐことが可能だが，スクリプトによる盗視・改竄は防ぐことはできない．

2.4 コンテナ/VM

コンテナ [14, 15, 16, 17] は OS レベルの仮想化技術であり，サーバソフトウェアを実行するコンテナを同一 OS 上で複数同時に動作させることができる．サイト毎にコンテナを割り当てることで擬似的な専用サーバを構築可能であるが，仮想化のオーバヘッドにより，OS 当たりに収容可能なコンテナ数は数百程度に限定される．共有型ホスティングサービスに求められる，計算機当たり 1000 サイト規模の収容は困難である．また，例えば Linux-VServer [17] を利用するためにはカーネルの修正が必要であるが，カーネル修正はカーネルの版数が上がる毎に必要となり，追従するコストは高い [18]．追従を諦めた場合，最新の機能やデバイスが利用できないという不都合が生じる．

仮想計算機 (VM) [19, 20, 21] は同一計算機上で複数の OS を同時に動作させる仮想化技術である．例えば，VMware ESX Server 上に複数の OS を収容し，ウェブやメール等のサーバプログラムを動作させた場合，OS 当たり 200 MB 程度のメモリを消費することが報告されている [20]．計算機当たり 1000 サイト収容するためには単純計算で 200 GB のメモリが必要となり，コスト面で現実的でない．また，準仮想化 [21] を利用するにはカーネル修正が必要となり，コンテナ同様，追従コスト等の問題がある．

2.5 Harache/Hi-sap

複数サイトで計算機を共用する場合に計算機を共用する他のユーザからコンテンツファイルを盗視・改竄されるという問題に対し，我々はこれまでに Harache [22] 及び Hi-sap [23] という 2 つのウェブサーバシステムを提案してきた．両システムとも，サーバプロセスをサイトを所有するユーザの実行権限に変更することで，ファイルパーミッションを owner のみに付与した状態でのサービス提供を可能としている．

Harache は組込みインタプリタや WebDAV [24] 等のサーバ組込みプログラムをセキュアかつ便利に利用可能とすることを目的としたシステムである．suEXEC の欠点であったスループット性能を解決するため，setuid bit が設定されたバイナリ経由でスクリプトを実行するのではなく，サーバプロセスをあらかじめ root 権限で起動しておき，ブラウザからのリクエストを受信した際にリクエストされたサイトを所有するユーザの実行権限に setuid(), setgid() システムコールにより遷移し，リクエスト処理を行う (図 2(2))．一度リクエスト処理を行ったサーバプロセスは他のユーザの実行権限に遷移できないため，他のサイトの処理を行うことができず，HTTP セッション満了とともに終了する．Harache は suEXEC

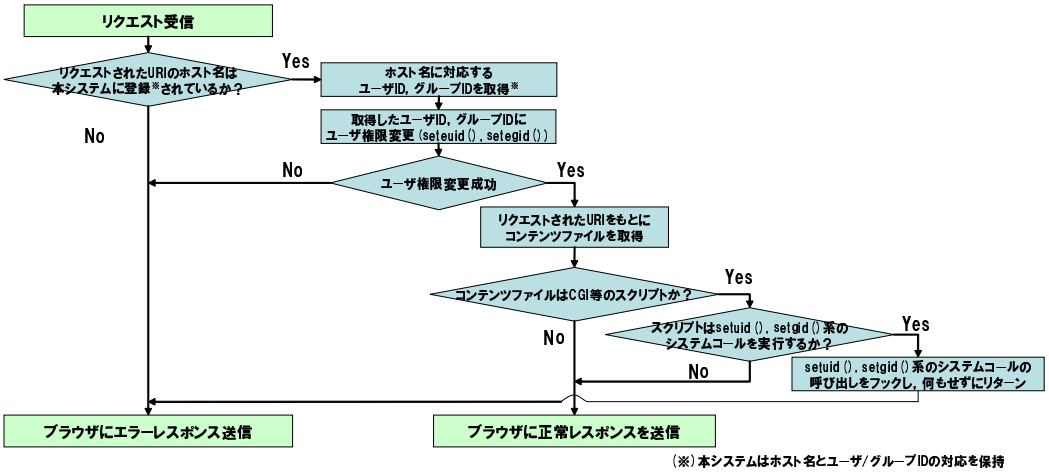

図 3: 本システムのフローチャート

と比較して最大 1.7 倍のスループットを確認したが，サーバプロセスの終了により，組込みインタプリタの性能が十分発揮できないという課題がある．

Hi-sap は組込みインタプリタの性能を十分に発揮させることを目的としたシステムである．suEXEC や Harache のようにリクエストに応じて動的に実行権限を変更するのではなく，あらかじめサイト毎にそれぞれ異なる実行権限で動作するウェブサーバ (worker) を用意しておき，フロントエンドで動作する dispatcher がリクエストを受信すると，該当のサイトを担当する worker にリクエストを転送（リバースプロキシ）する (図 2(3))．また，リクエストのないサイトを担当する worker は停止しておき，リクエスト発生時に動的に起動することにより，計算機当たり 1000 サイトまでを収容可能であることを確認した．Hi-sap は suEXEC と比較して最大 14.3 倍のスループットを確認した．また，Hi-sap は通常の Apache と比較して，平均 2.0 %，最大 6.9 %のスループット性能の低下にとどまり，低オーバヘッドであることを確認したが，遅延時間が平均 1.24 倍，最大 1.55 倍であり，課題を残した．

3. 設計

本章では，我々が提案する実行権限変更機構の設計について述べる．本設計は UNIX 系 OS を対象とする．

実行権限変更

2.5 節で述べた Harache と同様，ウェブサーバのプロセスをあらかじめ root 権限で起動しておく．そして，ブラウザからのリクエストを受信した際にリクエストされたサイトを所有するユーザの実行権限に seteuid(), setegid() システムコールを実行することで遷移する．リクエスト処理を行い，ブラウザにレスポンスを送信した後，再度 seteuid(), setegid() システムコールを実行し，root 権限に戻る．

Harache/Hi-sap 同様，ファイルパーミッションを owner のみに付与した状態でのサービス提供が可能であり，セキュリティを確保する．また，setuid(), setgid() システムコールを用いる Harache と異なり，seteuid(), setegid() システムコールを用いることにより，HTTP セッション満了の度にサーバプロセスを終了する必要はなく，複数回実行権限を変更することができ，スループット性能及び遅延の面で有利である．

ユーザスクリプトによる実行権限変更制限

ユーザのコンテンツのうち，CGI 等のスクリプトは，本システムと同様に setuid(), setgid() 系のシステムコールを実行することが可能であり，悪意のあるユーザに root 権限を奪取される危険性がある．これを防ぐため，スクリプトが実行する setuid(), setgid() 系のシステムコールをフックして無効化することにより，setuid(), setgid() 系のシステムコールを実行できるのを本システムのみに制限する．

本システムのフローチャートを図 3 に示す．本システムはホスト名とユーザ/グループ ID の対応を保持し，それに基づいて実行権限を変更する．

表 1: 実験環境

	クライアント＆サーバ
CPU	AMD Opteron 240EE 1.4 GHz ×2
Memory	4 GB
OS	CentOS 5.3 (Linux 2.6.18)
NIC	Broadcom BCM5704C 1 Gbps

4. 評価

本章では，本システムに対して行った基礎性能評価及びセキュリティ評価の結果を報告する．

4.1 基礎性能評価

本節では，本システムに対して行ったスクリプトへのリクエスト処理における基礎性能評価実験について述べ，性能が実用に耐え得ることを確認する．実験環境は，表 1 に示す一台の計算機でクライアントとサーバを兼ねた．

本システムの実行権限変更機構を Apache HTTP Server 2.2.10 のモジュールとして実装した．通常の Apache HTTP Server 2.2.10 を比較対象とし，本シス

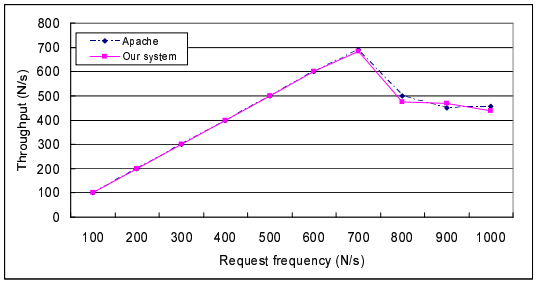

図 4: 基礎性能評価実験

テム，Apache とも設定ファイルにおいて初期状態の設定値を用いた．性能評価には，httperf ベンチマーク [25] 0.9.0 を用いた．

リクエスト頻度を変えながらスクリプトに対してリクエストを送信し，レスポンスの際のスループットを計測した．リクエスト対象には，サーバ計算機にインストールされている PHP 処理系の各種情報を出力する phpinfo() 関数を呼び出す PHP スクリプトを用いた．1 回のリクエストのデータ転送量は 40 KB 程度である．図 4 に測定結果を示す．横軸は 1 秒間に送信したリクエストの回数を表しており，縦軸は 1 秒間に受信したレスポンスの回数を表している．本システムは通常の Apache と比較して平均 0.5 %，最大 4.7 %の性能低下であり，非常に小さなオーバヘッドである．

以上の基礎性能評価実験により，本システムが実用に耐え得る性能を達成していることを確認した．

4.2 セキュリティ評価

従来，サーバプロセスを root 権限で動作させることは危険と考えられていた．Samba 等，本システムと同様，root 権限で起動した後，リクエストに応じて実効ユーザ ID/実効グループ ID のみを変更して処理を行うサーバプログラムは存在するが，従来の UNIX 系 OS は root 権限により OS の全ての操作が可能であり，特に root 権限のままユーザのスクリプト等サーバプログラム以外のコードが実行される期間はセキュリティが保障できなかった．

しかし，2.3 節で述べたセキュア OS を用いることで，root 権限を制限することが可能となる．つまり，サーバプロセスが操作可能な範囲を限定することができ，万一サーバプロセス奪取されたとしても，その被害を局所化可能である．よって，本システムを SELinux [9] 等のセキュア OS と組み合わせて動作させることにより，安全なサーバ環境を提供可能である．

5. 結び

本稿では，UNIX 系 OS の実行権限に関する問題点を明確化し，それを解決するための実行権限変更機構を提案した．提案システムの基礎性能を評価した結果，実用に耐え得る性能を達成していることを確認した．

今後，実アプリケーションを用いた性能評価及び，他のサーバプログラムへの適用の検討を進める予定である．

参考文献

[1] Apache HTTP Server. `http://httpd.apache.org/`

[2] Andreas Grunbacher: POSIX Access Control Lists on Linux, *Proc. FREENIX Track: 2003 USENIX Annual Technical Conference*, pp.259–272 (2003).

[3] Nathan Neulinger: CGIWrap: User CGI Access. `http://cgiwrap.sourceforge.net/`

[4] Sebastian Marsching: suPHP. `http://www.suphp.org/`

[5] PHP: Hypertext Preprocessor. `http://www.php.net/`

[6] mod_ruby. `http://modruby.net/`

[7] mod_perl. `http://perl.apache.org/`

[8] mod_python. `http://www.modpython.org/`

[9] Peter Loscocco, et al.: Integrating Flexible Support for Security Policies into the Linux Operating System, *Proc. FREENIX Track: 2001 USENIX Annual Technical Conference*, pp.29–40 (2001).

[10] LIDS. `http://www.lids.org/`

[11] Toshiharu Harada, et al.: Task Oriented Management Obviates Your Onus on Linux, *Proc. Linux Conference 2004* (2004).

[12] John McLean: The algebra of security, *Proc. 1988 IEEE Symposium on Security and Privacy*, pp.2–7 (1988).

[13] Jerome H. Saltzer, et al.: The protection of information in computer systems. *Proc. the IEEE*, Vol.63, No.9, pp.1278–1308 (1975).

[14] Poul-Henning Kamp, et al.: Jails: Confining the omnipotent root, *Proc. the 2nd International System Administration and Networking Conference* (2000).

[15] Parallels Virtuozzo Containers. `http://www.parallels.com/products/virtuozzo/`

[16] Jeff Dike: A user-mode port of the linux kernel, *Proc. the USENIX Annual Linux Showcase and Conference* (2000).

[17] Linux-VServer. `http://linux-vserver.org/`

[18] Peter Suranyi, et al.: General Virtual Hosting via Lightweight User-level Virtualization, *Proc. the 2005 International Symposium on Applications and the Internet (SAINT 2005)*, pp.229–236 (2005).

[19] Paul Barham, et al.: Xen and the Art of Virtualization, *Proc. the ACM Symposium on Operating Systems Principles (SOSP 2003)*, pp.164–177 (2003).

[20] Carl A. Waldspurger: Memory Resource Management in VMware ESX Server, *Proc. the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002)*, pp.181–194 (2002).

[21] Andrew Whitaker, et al.: Denali: Lightweight Virtual Machines for Distributed and Networked Applications, Technical Report 02-02-01, University of Washington (2002).

[22] 原 大輔，尾崎 亮太，兵頭 和樹，中山 泰一：Harache：ファイル所有者の権限で動作する WWW サーバ，情報処理学会論文誌，Vol.46, No.12, pp.3127–3137 (2005).

[23] Daisuke Hara and Yasuichi Nakayama: Secure and High-performance Web Server System for Shared Hosting Service, *Proc. the 12th International Conference on Parallel and Distributed Systems (ICPADS 2006)*, pp.161–168 (2006).

[24] Yaron Y. Goland, et al.: HTTP Extensions for Distributed Authoring – WEBDAV, *RFC 2518* (1999).

[25] David Mosberger, et al.: httperf—A Tool for Measuring Web Server Performance, *Proc. 1st Workshop on Internet Server Performance*, pp.59–67 (1998).