SAccessor: デスクトップPCのための安全なファイルアクセス制御システム

滝澤裕二[†] 光来健一[†]
千葉滋[†] 柳澤佳里[†]

従来のシステムではOSによる認証やファイルアクセス制御により不正アクセスを防いできた。しかし攻撃者にOSの制御を奪われた場合には任意のファイルへのアクセスを許してしまう。この問題を解決するために本稿では、デスクトップPC上で仮想計算機を用いてファイルアクセス制御を隔離して安全に実行するシステムSAccessorを提案する。SAccessorはユーザが作業を行う作業VMと認証およびファイルアクセス制御を行う認証VMを同一ホスト上で動作させる。作業VMは認証VMと通信してファイルアクセスを行い、認証VMはユーザが利用しているディスプレイ上に直接ダイアログボックスを表示することで作業VMを介さない安全な認証を可能にする。また、ダイアログによる認証に適したアクセス制御を実現するために作業VMが用いるファイルシステムを拡張した。

SAccessor: A Secure File Access Control System for Desktop PC

Yuji Takizawa,[†] Kenichi Kourai ,[†] Shigeru Chiba [†]
and Yoshisato Yanagisawa [†]

Traditional systems prevent illegal accesses by the authentication and access control in operating systems. However, if attackers compromise an operating system itself, they can access arbitrary files. To solve this problem, we proposed SAccessor, which is a system that separates and performs file access control using virutal machines (VMs) on desktop PCs. SAccessor runs the working VM for users' work and the authentication VM for authentication and file access control on one host. The working VM performs file accesses by communicating with the authentication VM. The authentication VM directly shows a dialog box on the users' display to enable secure authentication via no working VM. We extended the file system used by the working VM to achieve access control with that dialog.

1. Introduction

ビジネスでデスクトップPC（またはラップトップPC）を使用し、重要なファイルがデスクトップPCに保存されていることが多くなっている。しかしデスクトップPCはメールの添付ファイルや、P2Pソフトなどを用いた情報漏えいの危険にさらされている。従来はOSのアクセス制御機構により、これらの攻撃の被害は限定されてきた。例えば、一般ユーザ権限で動く攻撃プログラムは管理者権限を必要とするファイルにはアクセスすることができない。

しかし、OSによるアクセス制御はOSカーネルの脆弱性を利用した攻撃を受けると機能しなくなる可能性がある。近年、OSカーネルの脆弱性は数多く見つかっており、デスクトップPCに修正パッチを未適用であることが問題になっている。デスクトップPCはサーバほど十分なセキュリティ対策がとられていないためである。OSカーネルの脆弱性を攻撃されると、ユーザ認証をバイパスして管理者権限を奪われたり、アクセス制御そのものを無効化される危険性がある。

この問題を解決するために、仮想計算機 (VM) を用いてファイルアクセス制御を隔離して安全に実行するSAccessorを提案する。SAccessorはユーザが作業を行う**作業VM**とファイルアクセス制御および認証を行う**認証VM**を同一ホスト上で動作させる。作業VMは認証VMと通信してファイルアクセスを行い、認証VMはポリシにしたがってダイアログボックスを出して認証を行う。

このような認証ダイアログの利用に適したアクセス制御を行えるようにするために我々は作業VMが用いるファイルシステムを拡張した。ダイアログが出さ

† 東京工業大学 情報理工学研究科
Tokyo Institute of Technology, Information Science and Engineering

れる頻度を抑えるためにグループ単位で認証を行い、認証の有効期間を設定できるようにした。また追記のみというパーミッションを追加することでログファイルに対応した。さらに、setuid プログラムが管理者権限の必要なファイルにアクセスする際に管理者の認証を求めず、かつ、安全に実行できるようにした。

以下、2 章では従来のファイルアクセス制御機構の問題点を示し、3 章ではその問題を解決するシステム SAccessor の基本的なアーキテクチャについて述べる。4 章ではシステムを構築する上で行ったファイルシステムの拡張について述べ、5 章では SAccessor の有効性について述べ、6 章で SAccessor のオーバーヘッドを測定した実験について述べ、7 章で関連研究について述べ、8 章で本稿をまとめる。

2. 従来のファイルアクセス制御

近年、デスクトップ PC に対する攻撃が増えてきている。例えば、攻撃者はメールに悪意のあるプログラムを添付し、メーラの脆弱性やユーザの不注意を利用してプログラムを実行させる、P2P ソフトウェアの場合は、ユーザの関心を誘うような名前のファイルを共有し、ファイルをダウンロードさせて実行させる攻撃が行われる。このような攻撃を受けるとユーザの機密情報が流出する危険がある。また、実行ファイルを書き換えられてウィルスに感染したり、起動スクリプトを書き換えられてシステムの起動時にバックドアが起動するようになってしまう。攻撃者が PC への侵入に成功した場合には、ログファイルが削除されてしまい侵入の発見が困難になる。

従来、このような攻撃は OS による認証とファイルアクセス制御によってある程度防がれていた。ユーザのログイン時に OS が認証を行い、認証が成功すると特定のユーザ ID とグループ ID が与えられる。OS のファイルシステムはアクセス制御リスト（ACL）を管理しており、アクセス要求に対してユーザ ID およびグループ ID と ACL を比較してアクセスの可否を決定する。例えば、一般ユーザはパスワードファイルを読むことはできず、実行ファイルの書き換えもできない。

しかし、近年 OS にも脆弱性が見つかっている。例えば、ローカルユーザが管理者権限を取得できる脆弱性[1] が報告されている。また、OS カーネルをバッファオーバフローさせて、任意のコードを実行させることができる脆弱性[2] も報告されている。このような脆弱性に対しては修正パッチが公開されるが、デスクトップ PC を使うユーザのセキュリティ意識が低い場合や、パッチを適用すると動作しなくなるアプリケーションがある場合には、パッチが適用されていないことが多い。

このような OS の脆弱性を攻撃されると、従来のファイルアクセス制御は機能しなくなる可能性がある。悪意のあるプログラムに OS カーネルの脆弱性を利用して管理者権限を奪われると、管理者としてログインするための認証をバイパスされ、任意のファイルへのアクセスを許してしまう。さらに OS 自体の制御を奪われたとすると OS が提供しているファイルアクセス制御そのものを無効化されてしまい、たとえ SELinux[3] などのセキュア OS で管理者権限を制限していても任意のファイルへのアクセスを許してしまう。

3. SAccessor

前述の問題を解決するために、仮想計算機を用いてファイルアクセス制御を隔離して安全に実行するシステム SAccessor を提案する。SAccessor は同一ホスト上に、ユーザが作業を行う**作業 VM** と**認証 VM** を動作させ、作業 VM で行う従来の OS によるファイルアクセス制御とは別に、認証 VM でも独立してファイルアクセス制御を行う。これにより、作業 VM の OS の制御を奪われてもファイルアクセス制御を機能させることができる。3 章では SAccessor の基本的なアーキテクチャについて説明し、それを実用的にするために必要になったファイルシステムの拡張について 4 章で述べる。

3.1 アーキテクチャ

SAccessor では、図 1 のように 2 つの VM を同一マシン上で動かす。一つはユーザが作業を行う作業 VM であり、もう一つは作業 VM とは独立して認証とファイルアクセス制御を行う認証 VM である。作業 VM はファイルシステムとして NFS を使用し、認証 VM 上に置かれた作業 VM 用のディレクトリをルートディレクトリにマウントする。SAccessor はデスクトップ PC を対象としており、ユーザはマシンに直接アクセスして作業する。ユーザの使うアプリケーションは作業 VM 上で実行され、ユーザは作業 VM の画面だけを見ながら作業する。ファイルは認証 VM によって管理されている。そのため、作業 VM は直接ファイルにアクセスすることはできない。ファイルアクセスを行う場合には、まず作業 VM の OS により通常のアクセス権のチェックが行われ、その後、認証 VM 上でユーザ認証とアクセス権のチェックが行われるというように、作業 VM と認証 VM で二重に行われる。

認証 VM への攻撃を防ぐために認証 VM への外部

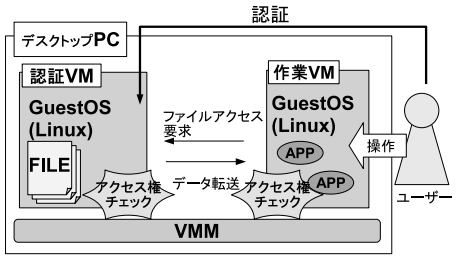

図 1　SAccessor のアーキテクチャ

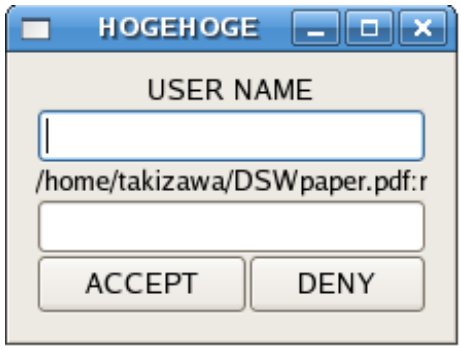

図 2　認証ダイアログ

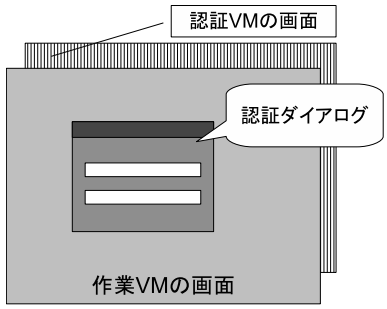

図 3　SAccessor の画面構成

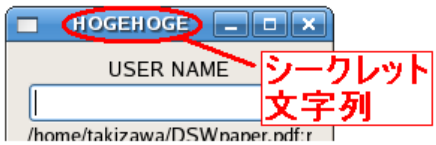

図 4　シークレット文字列

からのアクセスは禁止し、作業 VM からのアクセスは NFS に限定している。さらに、認証 VM には管理者しかログインできないようにしている。

3.2　SAccessor の認証

3.2.1　認証ダイアログ

作業 VM がファイルにアクセスするときにはポリシに応じて認証 VM がユーザ認証を行う。認証 VM は作業 VM の画面上に図 2 のような**認証ダイアログ**と呼ばれるダイアログボックスを表示し、ユーザにユーザ名とパスワードを直接入力させることによりユーザ認証を行う。SAccessor では作業 VM 上の OS の制御を奪われることを想定しているため、NFS リクエストで渡されるユーザ ID は信用せず、認証ダイアログで入力されたユーザ名を元にファイルアクセス制御で使用するユーザ ID を決定する。認証ダイアログにはアクセスしようとしているファイルのパス名とアクセスの種類も表示される。これはどのファイルに対して認証を行うか確認するためである。ユーザが不正アクセスと判断した場合には、DENY ボタンをクリックすることで認証を失敗させることができる。

認証ダイアログを用いる際に重要な点は、作業 VM に干渉されないこと、およびユーザに認証 VM の存在を意識させないことである。これらを実現するために SAccessor では図 3 のよう画面を構成する。認証 VM で X を起動させ、その画面をディスプレイに表示する。作業 VM でも X を起動し VNC を使って認証 VM の X 上にウィンドウを表示しフルスクリーンモードにする。認証ダイアログはこの VNC ウィンドウの上に重ねて表示される。

このようにすることでユーザには認証ダイアログが作業 VM から出されたように見える。その一方で認証ダイアログは認証 VM の管理下にあるために、作業 VM の内部からは操作することができない。そのため作業 VM 上で動く悪意のあるプログラムは認証を行うことができない。たとえリモート攻撃者に作業 VM の管理者権限を奪われても、攻撃者はこの認証を成功させることができない。また、認証ダイアログに入力したユーザのパスワードは作業 VM に渡らないのでユーザのパスワードを盗まれる危険性は低い。そのため、たとえ攻撃者が物理的にデスクトップ PC にアクセスできたとしても認証を成功させるのは難しい。

3.2.2　認証ダイアログの識別

SAccessor では認証 VM が認証ダイアログを作業 VM の画面上に重ねて表示するため、作業 VM に偽のダイアログを出された場合、本物の認証ダイアログと区別するのが難しい。攻撃者によって作業 VM から出された偽のダイアログにパスワードを入力してしまうと、攻撃者にパスワードを盗まれてしまう。

偽のダイアログと本物の認証ダイアログを区別するために、認証ダイアログのタイトルにシークレット文字列を表示させる (図 4)。シークレット文字列はあらかじめ認証 VM に登録しておくユーザのみが知っている文字列である。ユーザはダイアログに表示されたシークレット文字列を見て、ダイアログが認証 VM か

ら出されたものであることを確認する。シークレット文字列は認証 VM に保存されているので、作業 VM に侵入した攻撃者にはこのシークレット文字列はわからない。また作業 VM の画面をキャプチャしたとしても、認証ダイアログは認証 VM の管理下にあるので作業 VM に侵入した攻撃者からはシークレット文字列を見ることができない。そのためユーザは作業 VM から偽のダイアログを出されても本物の認証ダイアログと区別することができる。

4. ファイルシステムの拡張

3 章で述べたような認証ダイアログを使ったアクセス制御を単純に用いると実用的にならない。我々は認証ダイアログのりように適したアクセス制御を行えるようにするために、作業 VM が用いるファイルシステムを拡張した。

4.1 認証頻度の制御

全てのファイルアクセスに対して毎回ダイアログを出して認証を行うのは実用的ではない。何故なら、ファイルアクセスの度にダイアログによる認証を行うと非常に多くのダイアログが表示され著しく利便性が落ちてしまうからである。その一方で認証をログインのときにしか求めないようにするとログイン中に作業 VM の OS の制御を奪われた時にそれ以降は任意のファイルにアクセスされてしまう。

4.1.1 認証の有効範囲

SAccessor では、ファイルやディレクトリをまとめたグループ単位でアクセス制御を行う。ポリシファイルにグループを定義することができ、それぞれのグループに対してアクセスを許可するパーミッションと認証の有効期間を設定する。一度グループに対して認証を行ったら、認証の有効期間内ではグループのファイルへのアクセスに対する認証は省略され、認証の回数を減らすことができる。また、従来のログイン時のみの認証と比べるとよりきめ細かいアクセス制御が可能になる。たとえ認証を行ったあとで作業 VM 上の OS の制御を奪われたとしても、攻撃者は認証が有効になっているグループのファイルにしかアクセスできない。

図 5 は SAccessor のポリシ例である。タグで囲まれた範囲が一つのグループになり、その中で羅列されるファイルのパス名がグループに属するファイルになる。それぞれのパス名の後ろには許可したいアクセス権を記述する。r、w はそれぞれ読み込みと書き込みを表している。グループタグの隣に書かれた数字は認証の有効期間（秒）を表している。ユーザが `budget.xls` を編集しようと認証を行うと `Report` グループに属するファイルへのみアクセスが可能になる。もし、`budget.xls` がウィルスに感染していてプログラムの制御を奪われたとしても、ウィルスがファイル `id_rsa` にアクセスしようとすると、新たに認証を求められる。ウィルスはこの認証を成功させることができず `id_rsa` にアクセスできない。

```
<Report> [3000]
/home/takizawa/Report/budget.xls  (rw)
/home/takizawa/Report/reference.pdf (r)
</Report>

<ssh>[10]
/home/takizawa/.ssh/id_rsa (r)
</ssh>
```

図 5 SAccessor ポリシ例

4.1.2 ファイルキャッシュの制御

作業 VM が認証 VM の NFS サーバから取得したファイルはファイルキャッシュとして保持され、その後のファイルアクセスはそのファイルキャッシュに対して行われる。作業 VM にファイルキャッシュが残っていると、作業 VM は認証 VM での認証なしでファイルを読むことができてしまう。例えば図 5 のポリシにおいて、初めてユーザが `budget.xls` を読み込もうとしたときには認証を求められる。認証に成功すると実際にファイルを読み込み作業 VM のメモリにキャッシュする。認証の有効期間が過ぎた後に再び `budget.xls` を読もうとした場合、認証 VM と通信をして認証を行うべきだが、作業 VM にファイルキャッシュが残っている場合には、認証 VM とは通信せず作業 VM のファイルキャッシュに直接アクセスしてしまい、認証を強制できない。

SAccessor では認証 VM から作業 VM に対して認証の有効期間が過ぎたことを通知し、作業 VM がファイルキャッシュをフラッシュする。作業 VM のゲスト OS の制御が奪われるとこの機構は機能しなくなるが、それによりセキュリティが低下することはない。その時点で作業 VM 上に残っているファイルキャッシュは認証の有効期間内のファイルのものだけであるためである。作業 VM はこれらのファイルにその有効期間は認証なしでアクセスできる。

認証 VM 上で動いている認証ダイアログを表示させるサーバプログラムは認証に成功すると認証の有効期間をカウントするスレッドを作り、認証の有効期間が切れたときに作業 VM へ有効期間が切れたグルー

プのファイルのリストを送る。作業 VM 上の OS はオープンしたファイルのリストを保持しておき、認証 VM から受け取ったファイルのリストに対応するファイルキャッシュをフラッシュさせる。

4.2 パーミッションの拡張

ログファイルなどユーザが明示的にアクセスしなくても、システムによってアクセスさるものがある。システムによってアクセスされるファイルに対して認証ダイアログを表示させても、ユーザにはそれが攻撃者によるファイルアクセスなのかの判断が難しい。ファイルによっては管理者のパスワードが必要になることもある。

SAccessor では読み込みと書き込みのパーミッションの他に、追記のみというアクセス権を設定できる。このパーミッションを活用し、ポリシに認証の有効期間を設定しなければ認証は省略できる。例えば下に示すようなログファイルを追記のみ認証なしで許可するというようなポリシを記述することで、ロギング等に対してダイアログがでないようにすることができ、認証ダイアログを出さずにログファイルの消去を防ぐことができる。

```
<log>
/var/log/message (a)
</log>
```

NFS ではオープン時に追記モードで開くかわからないので書き込み要求を受け取るときに書き込むオフセットがファイルサイズと一致しているかどうか調べ、書き込み先がファイルの末尾かどうか確認し、アクセスの可否を決定する。

4.3 一般ユーザによる setuid プログラムの実行

パスワード変更などのために setuid されたプログラムを実行する場合には一般ユーザが管理者のファイルにアクセスすることがある。従来のシステムの場合 setuid されたプログラムのユーザ ID はそのプログラムの所有者の権限で実行される。しかし、ではプログラムに関わらずファイル所有者のユーザ名とパスワードを要求する。そのため一般ユーザが setuid されたプログラムを使って管理者のファイルにアクセスする場合でも、管理者のパスワードが必要になってしまう。例えば一般ユーザがパスワード変更を行うときは passwd コマンドを実行する。passwd は/etc/shadow ファイルを修正しようとするが、このファイルは管理者のファイルであるので、SAccessor からは管理者のパスワードを入力しなければファイルのアクセス権が与えられない。

SAccessor は setuid されたプログラムは認証 VM 上で行う。そのために作業 VM の setuid されたプログラムは認証 VM へプログラムの実行を依頼するプログラムに置き換えておく。認証 VM は作業 VM からリクエストを受け取ると認証ダイアログによりユーザ認証を行い、認証に成功した場合には、対応するプログラムを認証 VM 上で実行する (図 6)。このときに入力するユーザ名とパスワードは一般ユーザのものである。このような手法をとることで、管理者のファイルに対する認証を行うことなく setuid されたプログラムを実行することができ、作業 VM からは setuid されたプログラムが使うファイルへのアクセスを制限することができる。例えば/etc/passwd ファイルは読み取り専用にすることができる。ファイルにアクセスするのは認証 VM 上のプログラムなので、ファイルキャッシュが作業 VM に残ることもない。

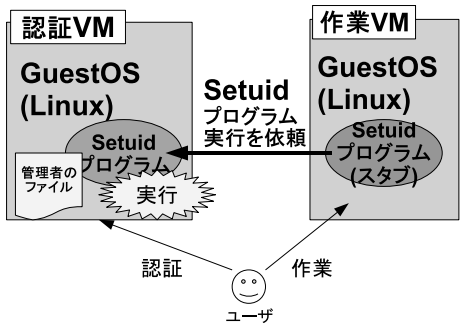

図 6 setuid されたプログラムの実行

setuid されたプログラムは認証 VM の表示するウィンドウ上で実行され、プログラムに対する標準入出力はこのウィンドウを通して直接ユーザから受け取る。標準入出力を作業 VM を介さずに行うために、passwd コマンドで新しいパスワードを入力してもパスワードが作業 VM にもれることはない。作業 VM と同じディレクトリ構造で setuid プログラムを実行するために、作業 VM 用のディレクトリに chroot して実行する。図 7 は passwd コマンドの実行画面である。画面前面に表示されているウィンドウは認証 VM が表示する xterm であり、このウィンドウは passwd の実行が終了したら破棄される。

この仕組みを用いて我々はポリシ編集のための setuid プログラムを新たに用意した。作業 VM に認証 VM へポリシ編集プログラムの起動を依頼するプログラムを用意しておき、認証 VM の表示する専用エディタを通してポリシの編集を行う。このポリシエディタはポリシファイルを編集するためのプログラムであり、ポリシファイル以外のファイルは開くことはできない。

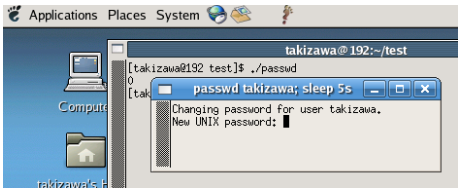

図 7 setuid プログラムのスクリーンショット

5. SAccessor のポリシ例

SAccessor を使うことで、バイナリファイルの書き換えを防ぐことができる。図 8 はこの攻撃を防ぐためのポリシ例である。グループ emacs と emacs-update は emacs パッケージのファイルをまとめたグループである。このポリシでは、emacs に関するファイルの読み込みの認証は省略されるが、ファイルの書き換えを行うときには認証を求められる。このポリシを適用すると、ユーザが emacs を使って作業するときには認証なしで作業ができる。管理者がパッケージのアップデートを行うときには、認証ダイアログがでるので認証を成功させればパッケージのアップデートを行うことができる。一方、攻撃者が emacs のプログラムを書き換えようとすると認証を求められるので、その認証を拒否することでプログラムファイルの書き換えを防ぐことができる。起動スクリプトに対しても同様のポリシを記述することで、攻撃者による書き換えを防ぐことができる。

SAccessor を使えば機密ファイルの漏洩も防ぐことができる。図 9 はブラウザに記録されているパスワードファイルを守るポリシ例である。`signons.txt` には web サイトなどへのログインパスワードが保存されている。`signon.txt` のアクセスには認証が必要である。ユーザがブラウザを開いていないのに認証を求められた場合には、攻撃者が `signon.txt` を盗み見ようとしているので認証を拒否すればファイルを見られることはない。ユーザがウェブサイトにログインしようとしたときに認証を成功させればよい。

setuid されたプログラムは認証 VM で実行されるため、作業 VM はそのプログラムが参照するファイルのアクセス権を必要としない。図 10 は sudo のポリシ例である。sudo は`/etc/sudoers` という設定ファイルを見るが、sudo は setuid プログラムなので作業 VM では`/etc/sudoers` にアクセスできるようにする必要がない。作業 VM からのアクセスを全面禁止にすることで sudo のポリシが読み書きされるのを防ぐことができる。

```
<emacs>
/usr/bin/emacs (r)
/usr/libexec/emacs/* (r)
/usr/share/pixmaps/emacs.png (r)
</emacs>

<emacs-update> [60]
/usr/bin/emacs (w)
/usr/libexec/emacs/* (w)
/usr/share/pixmaps/emacs.png (w)
</emacs-update>
```

図 8 プログラムファイルの書き換えを守るポリシ例

```
<firefox-password>[10]
/home/.../firefox/.../signons.txt (rw)
</firefox-password>
```

図 9 ユーザの機密ファイルを守るポリシ例

```
<sudo>
/usr/bin/sudo (r)
/etc/sudoers ()
</sudo>
```

図 10 suid されたプログラムのためのポリシ例

5.1 制　　限

ユーザが認証を成功させると作業 VM に対して認証の有効範囲内でのアクセス権が与えられ、認証の有効期間が切れるまでは認証なしでアクセスできる。従って、作業 VM の OS の制御が奪われる前に認証し、その認証が有効なファイルを攻撃者から守ることはできない。また、作業 VM の OS の制御が奪われた後に認証したファイルも守ることはできない。また、作業 VM にあるファイルキャッシュが攻撃者によって書き換えられると、ユーザは攻撃者によって書き換えられた偽のファイルの内容を読み込んでしまい、ファイルの改ざんと同じ被害を受ける可能性がある。

6. 実　　験

我々は SAccessor を用いることによりファイル I/O にどの程度のオーバーヘッドが生じるかを調べる実験を行った。実験対象の計算機には、PentiumD 3.0GHz の CPU、メモリ 1 GB の計算機を用いた。VMM には Xen 3.0.4、その上の VM で動く OS には Linux2.6.16.33 を用いた。作業 VM の動く VM には 256MB 割り当てた。

まず、ファイルの書き込みと読み込みのスループッ

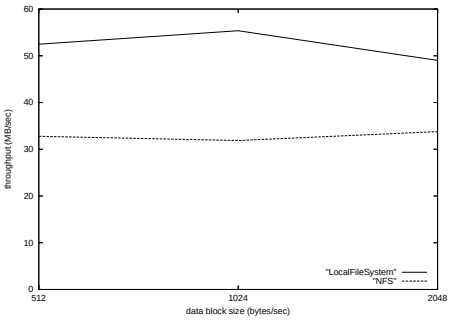

図 11 読み込み性能 (LocalFileSystem and NFS)

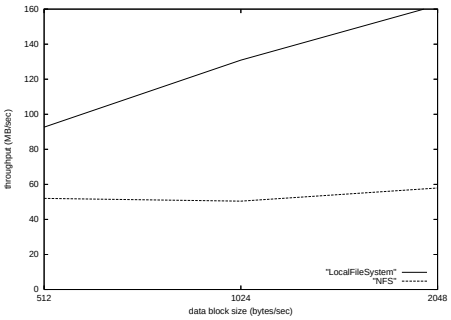

図 12 書き込み性能 (LocalFileSytem and NFS)

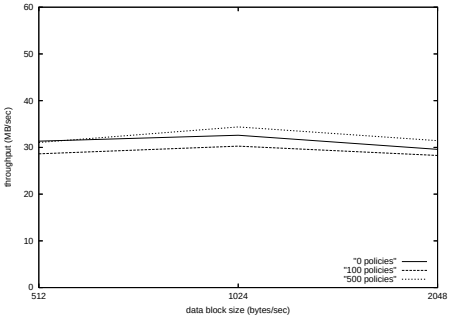

図 13 読み込み性能 (SAccessor)

トの測定を行った。スループットは 100MB のファイルに対する読み込み、および書き込みに対して測定した。この実験は (1) VM を用いずにローカルファイルシステム (ext3) に対して行った場合と、(2) SAccessor を用いた場合、(3) VM 間で NFS を用いた場合について行った。それぞれ一度に読み書きするデータブロックサイズは 512byte、1024byte、2048byte の場合について測定した。

実験結果は図 11、図 12 のようになった。縦軸はファイル I/O のスループット、横軸は読み込みあるいは書き込みのブロックサイズである。ローカルファイルシステムと比べて、SAccessor を使用した場合の読み込み性能は最悪の場合で 54 %、書き込みの性能は

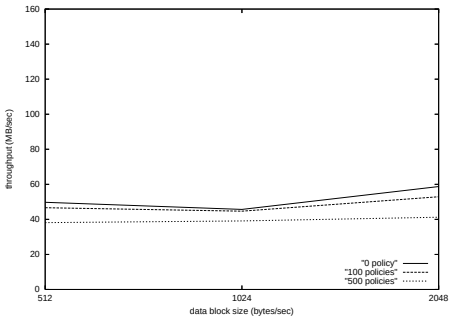

図 14 書き込み性能 (SAccessor)

25 %になった。VM を用いて NFS を使用した場合の性能はローカルファイルシステムと比較して、書き込みが 36 %読み込みが 57 %の性能であった。従って、SAccessor の性能低下の大部分はバックエンドに NFS を使用しているのが原因であると考えられる。

次に SAccessor を用いた場合についてポリシ数が 0、100、500 の場合に対して測定した。この実験結果は図 13、図 13 のようになった。結果から、オーバーヘッドはほぼ一定であることがわかった。ポリシ数増えると必ずオーバーヘッドも増えるという結果にはならなかったが、この原因は調査中である。

7. 関 連 研 究

SVFS[6] もユーザの作業とファイルアクセス制御を別々の VM で行うシステムである。SVFS では Normal VM と Admin VM があり、重要ファイルへの書き込みは Admin VM からのみに制限されている。そのため、重要ファイルの書き換えを伴う管理タスクは、通常作業を行う VM とは別の AdminVM から行わなければならない。SAccessor では認証ダイアログを出すことで作業 VM から可能である。

Proxos[5] は通常のアプリケーションを実行する Commodity VM とは別に Private VM と呼ばれる VM を用意し、重要ファイルにアクセスするアプリケーションを Private VM 上に隔離して実行する。Private VM 上で実行されるアプリケーションによる重要ファイルに関する操作は Private VM で実行し、それ以外の操作は Commodity VM に転送する。しかし、どの操作を Private VM で実行するのかというポリシをプログラマが記述しなければならない。

分散ファイルシステムを使いファイルサーバでファイルアクセス制御を行うことは可能である。しかし、一度認証してしまうと再度認証を求められないので認証をした後に管理者権限を奪われてしまうと攻撃者に任意のファイルのアクセスを許してしまう。また、ク

ライアントで入力する認証パスワードを盗まれてしまうと攻撃者が正規ユーザになり済ませてしまう。

Plan9[4] もファイルサーバを用いているが、管理者権限を必要とするファイルにアクセスするためには、ファイルサーバの物理的なコンソールから操作する必要がある。これは管理者権限を必要とするファイルに対してアクセス制御と認証を分離していることになるが、管理者権限を必要とするたびにファイルサーバの前まで行って操作するのは不便である。SAccessor ではユーザの利便性を向上させるために、デスクトップ PC 上でシームレスに認証を行えるようにしている。

SELinux[3] のような Trusted OS は管理者の権限を制限して、攻撃者からの被害を限定するシステムであるが、OS の脆弱性を攻撃することで無効化できる危険がある。それに対し SAccessor は攻撃者に OS の制御を奪われた場合にもアクセス制御が機能する。

8. まとめと今後の課題

本稿では、VM を用いてファイルアクセス制御を隔離して安全に実行するシステム SAccessor を提案した。SAccessor は同一ホスト上で作業 VM と認証 VM を動作させ、認証 VM が認証とふぁいっるアクセスを行う。認証 VM がディスプレイ上に認証ダイアログを表示し、ユーザに直接パスワードを入力させることで安全に認証を行うことができる。認証ダイアログの利用に適したアクセス制御を行うために、作業 VM が用いるファイルシステムを拡張した。この拡張により、認証の頻度を抑え、一般ユーザによる setuid プログラムの安全な実行が可能になった。

今後の課題は、作業 VM と認証 VM の間で用いる NFS を Xen に特化させて高速化することである。

参 考 文 献

1) CVE-2005-0750.
2) CVE-2005-1263.
3) SElinux http://www.nsa.gov/selinux/.
4) Bell Labs. Plan9:plan9 from bell labs. http://plan9.bell-labs.com/plan9/.
5) Richard Ta-Min, Lionel Litty, and David Lie. Splitting interfaces: Making trust between applications and operating systems configurable. In *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006)*, November 2006.
6) Xin Zhao, Kevin Borders, and Atul Prakash. Towards protecting sensitive files in a compromised system. In *SISW '05: Proceedings of the Third IEEE International Security in Storage Workshop (SISW'05)*, pp. 21–28, Washington, DC, USA, 2005. IEEE Computer Society.