OS の制御を奪われても機能するファイルアクセス制御システム

滝澤 裕二　　光来 健一　　柳澤 佳里　　千葉 滋

東京工業大学大学院
情報理工学研究科 数理・計算科学専攻
{takizawa,kourai,yanagisawa,chiba}@csg.is.titech.ac.jp

要 旨

従来のシステムでは OS のファイルアクセス制御により攻撃を受けた時の被害を限定していた。しかし、OS 自体に脆弱性が存在する場合には、そのファイルアクセス制御機能をバイパスされる危険性がある。この問題を解決するために、本稿では OS の制御を奪われた状況においてもファイルアクセス制御を強制するシステム SAccessor を提案する。SAccessor は攻撃対象の作業 OS とファイルアクセス制御を行う認証 OS をそれぞれ別の VM で動作させ、認証 OS に対して直接ユーザ認証させることによって安全なファイルアクセス制御を実現する。我々は SAccessor を Xen およびその上で動作する Linux 上に実装し、SAccessor のオーバーヘッドを測定する実験を行った。

1　はじめに

デスクトップ PC はファイアウォールによってインターネットからの直接攻撃から守られているため、攻撃者はメールの添付ファイルや P2P ソフトウェアを用いた間接的な攻撃を行うことが多い。これらの攻撃はアプリケーションの脆弱性やユーザの不注意を利用して、悪意のあるプログラムを実行させることによって行われる。従来は、OS のファイル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 によるファイルアクセス制御は 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、修正パッチが公開される前に行われるゼロデイアタックや、デスクトップ PC にパッチを当てていないことが問題になっている。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、OS の制御を奪うことでファイルアクセス制御そのものを無効化されたりする危険性がある。

この問題を解決するために、OS カーネルの制御を奪われても認証およびファイルアクセス制御を強制できるようにするシステム SAccessor を提案する。SAccessor では認証とファイルアクセス制御を攻撃対象の OS から分離するために、攻撃対象の作業 OS とファイルアクセス制御を行う認証 OS をそれぞれ別の仮想計算機（VM）上で動作させる。作業 OS は認証 OS と通信することでファイルアクセスを行う。信用できない作業 OS を介さずに安全に認証を行うために、認証 OS が認証ダイアログを画面に表示し、ユーザに直接パスワードを入力させる。

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ファイルアクセス制御システム SAccessor について述べる。4 章では SAccessor の実装について述べ、5 章では SAccessor のオーバーヘッドを測定した実験について述べる。6 章では SAccessor の安全性について議論を行う。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

デスクトップ PC に対して、メールの添付ファイルや P2P ソフトウェアを悪用したが攻撃が増えてきている。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。このようなプログラムは機密ファイルを攻撃者に送信させたり、不特定多数に対して公開したりしようとする。さらに、次に起動した時にもそのプログラムが実行されるようにシステムの設定ファイルを変更しようとする。

従来、このような攻撃は OS のアクセス制御機構によって防がれていた。ユーザのログイン時に OS が認証を行い、認証に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、プログラムを実行したユーザが一般ユーザである場合には、管理者権限が必要となる起動スクリプトの書き換えを行うことはできない。

さらに、信用できないコードを安全に実行できるようにするために、OS がサンドボックスと呼ばれるきめ細かいアクセス制御も提供されるようになってきている [4]。サンドボックスはアクセスできるファイルを制限した環境でプロセスを動作させ、システムに悪影響がおよぶのを防ぐ機構である。例えば、サンドボックス内でメールの添付ファイルを実行させることで、そのユーザの機密ファイルへのアクセスは禁止することができる。

このように従来のシステムは OS のアクセス制御によって攻撃を防いでいたが、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、パッチが公開される前の攻撃（ゼロデイアタック）に対しては対処できない。また、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当てると動作しなくなるアプリケーションがある場合には、パッチが当たっていないこともある。

このような OS カーネル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てしまう。さらに、OS 自体の制御を奪われたすると、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う。このような状況では、攻撃者に機密ファイルや起動スクリプトへのアクセスを許すことになる。

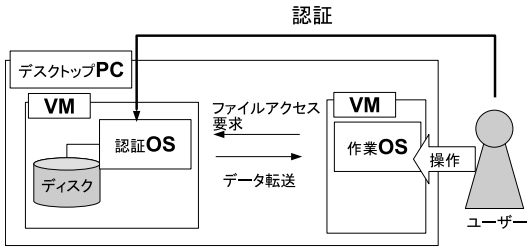

図 1: SAccessor の概観

3 SAccessor

我々は、OS カーネルの制御を奪われても認証およびファイルアクセス制御を強制するシステム SAccessor を提案する。SAccessor は VM を用いて認証とファイルアクセス制御機構を攻撃対象の OS から分離する。これにより、OS カーネルの脆弱性を攻撃されたとしても、アクセス制御を強制させることができる。SAccessor は従来のファイルアクセス制御を補完するものであり、OS の制御が奪われるまでは従来のファイルアクセス制御も行われる。

3.1 アーキテクチャ

SAccessor では VM を用いて、図 1 のように 2 つの OS 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 OS であり、もう一つは認証およびファイルアクセス制御を行う認証 OS である。SAccessor では 1 人のユーザが 1 台のマシンを使うデスクトップ PC を想定しており、ユーザには作業 OS の画面が見える。ユーザの使うアプリケーションは作業 OS で実行される。ファイルは認証 OS によって管理されており、作業 OS は認証 OS と通信してファイルにアクセスする。そのため、作業 OS は直接ファイルにアクセスすることはできない。また認証 OS への攻撃を防ぐために、認証 OS は作業 OS からのファイルアクセスに関する通信のみを許可する。

作業 OS がファイルにアクセスするときには必要に応じて認証 OS がユーザ認証を行う。認証に成功すると、作業 OS に対して、そのファイルへのアク

図 2: 認証ダイアログ

セスが許可される。認証は、ユーザが認証 OS に物理的につながれたデバイス（キーボードなど）を通して認証 OS と直接やりとりをして行う。この認証は、作業 OS を介さずに行うため、作業 OS の制御が奪われてもバイパスされることはない。

3.2 認証ダイアログ

認証は作業 OS の画面上に認証ダイアログと呼ばれるダイアログボックスを出し、パスワード入力させることにより行う。認証 OS が作業 OS の画面を管理し、、パスワード入力のための認証ダイアログを作業 OS の画面の上に重ねて表示する。認証ダイアログは認証 OS の管理下にあるために、作業 OS からは操作することはできない。そのため作業 OS 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。また、入力した認証のパスワードは作業 OS には渡らないので作業 OS にパスワードを盗まれることもない。パスワードが攻撃者に漏れないので、たとえ攻撃者が物理的にこの PC にアクセスできたとしても認証を成功させることは難しい。

ダイアログが認証 OS によって出されたことを保証するために、認証ダイアログにユーザのみが知っているシークレットを表示する。認証ダイアログは作業 OS の画面上に重ねて表示されるため、作業 OS に偽のダイアログを出された場合、区別するのが難しい。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、認証ダイアログにはあらかじめ認証 OS に登録しておいたシークレットを表示させて、認証ダイアログが認証 OS から出されたものであることをユーザに確認させる。作業 OS にはこのシークレットが分からないため、偽のダイアログを出されてもユーザは区別することができる。作業 OS は画面をキャプチャしたとしても認証ダイアログは認証 OS の管理下にあるのでシークレットは見ることができない。

図 2 は認証ダイアログが表示されている画面のスクリーンショットである。表示されている認証ダイアログは認証 OS によって出されたものであり、その後ろにあるターミナル画面は作業 OS の画面である。シークレットはダイアログのタイトルに表示されている文字列である。また、アクセスしようとしているファイルパスがダイアログ中央に表示されている。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右上のボタンをクリックすれば認証を失敗させることができる。

3.3 ファイルアクセス制御

アクセス制御はファイルに記述されたポリシーに基づいて行われる。ポリシーファイルにはファイルやディレクトリをまとめたグループ、それらのアクセス権を記述する。認証 OS ではファイルやディレクトリのグループを単位として認証を行う。ユーザはファイルやディレクトリを一つのグループとして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できる。一度グループに対して認証を行ったら、認証の有効期間内は認証が省略される。これはファイルアクセスの度に認証ダイアログを出して、作業効率を著しく落としてしまうのを防ぐためである。また、ユーザ単位の認証と比べてよりきめ細かいアクセス制御が可能になる。たとえ認証をしたあとで作業 OS の制御を奪われたとしても作業 OS は許可されたグループのファイルにしかアクセスできない。許可されたグループについても、認証の有効期間内しかアクセスできない。

ポリシーファイルの各行には、<> で囲まれたグループ名または、ファイルパス、ディレクトリパスのいずれかとパーミッション、認証の有効期間の組を記述する。あるグループ名から、次のグループ名までの間が一つのグループとなる。設定できるパーミッションは読み込み可能 (r)、書き込み可能 (w)、追記のみ (a) である。各ファイルのパーミッションや認証の有効期間を省略した場合には、グループに対して定義したパーミッションや有効期間が適用される。同じファイルに対して、複数のポリシーを記述した場合には、上の行に書いたものが優先される。

ポリシファイルの記述例を次に示す。

```
<init> (w) (60)
/etc/rc.local
/etc/init.d/
```

図 3: 起動スクリプト用のポリシ

```
<config> (rw) (10000)
/etc/passwd (rw) (600)
/etc/shadow (r) (600)
/etc/
```

この例では config というグループが作られており、ファイル /etc/passwd と /etc/shadow、および、ディレクトリ /etc が同じグループに属している。このグループに対して認証に成功すると、/etc/passwd は読み書き可能、/etc/shadow は読み取り専用、/etc 以下のその他のファイルは読み書き可能となる。認証の有効期間は、/etc/passwd および /etc/shadow が 600 秒、/etc 以下のその他のファイルは 10000 秒となっている。

3.4 ポリシの例

起動スクリプトの書き換えを防ぐポリシは図 3 のようになる。Linux では /etc/rc.local と /etc/init.d 以下に起動スクリプトが置かれている。このポリシを適用すると、これらのファイルを書き換えようとした時に認証を要求される。ユーザが一旦認証したとしても、認証の有効期間は 1 分と短いため、攻撃者が起動スクリプトを書き換えられる可能性は低くなる。

また、攻撃者がユーザの機密ファイルにアクセスするのを防ぐポリシは図 4 のようになる。この例では、それぞれ異なる機密情報が置かれているディレクトリ /home/takizawa/secret と /home/takizawa/private を別々のグループに分け、それぞれのグループに対する認証の有効期間を 10 分に設定している。これらのディレクトリの下の機密ファイルを読み書きする時には別々に認証を要求されるため、攻撃者にすべての機密情報を同時に読み書きされてしまう可能性は低くなる。

```
<takizawa-secret> (rw) (600)
/home/takizawa/secret/

<takizawa-private> (rw) (600)
/home/takizawa/private/
```

図 4: 機密ファイル用のポリシ

4 実装

SAccessor は、VMM として Xen[3]、認証 OS、作業 OS として Linux をベースに、カーネル内の NFS サーバを改造して実装を行った。使用した NFS プロトコルのバージョンは 3 である。Xen において VM はドメインと呼ばれ、制御用のドメイン 0 とそれ以外のドメイン U が存在する。SAccessor は認証 OS、作業 OS をそれぞれドメイン 0、ドメイン U で動作させる。作業 OS は認証 OS のファイルシステムを NFS マウントして使い、作業 OS がファイルにアクセスするときには、認証 OS の NFS サーバが認証およびアクセス権のチェックを行う (図 5)。

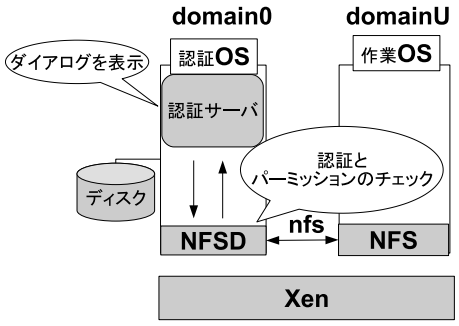

図 5: SAccessor のアーキテクチャ

4.1 認証

認証 OS 上には認証ダイアログを表示するための認証サーバが動いている。NFS サーバと認証サーバとの通信はデバイスファイルを用いて行う。認証サーバは起動すると、まず、認証ダイアログに表示するシークレットをユーザに入力させる。次にポリシファイルを読み込み、NFS サーバとの通信用のデバイスファイルを通してそのデータを渡す。その後、NFS サーバから認証の要求を受け取るまでスリープする。

NFS サーバがファイルアクセスのリクエストを受

け取った時に認証が必要だと判断したら、認証サーバに認証ダイアログを表示するように通知する。そのファイルのグループに対してまだ認証をしていない場合や、認証の有効期間を過ぎた時に認証が必要だと判断される。通知を受け取った認証サーバは認証ダイアログを表示し、ユーザにパスワードを入力させる。そして、受け取ったパスワードが正しいかどうかを調べ、その結果を NFS サーバに渡し再びスリープする。NFS サーバは認証に成功していれば処理を続行し、失敗していればエラーを NFS クライアントに返す。

SAccessor では認証 OS が認証ダイアログを作業 OS の画面に重ねて表示するために、以下のように画面を構成する。認証 OS 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 OS 上でも X を起動し、VNC を使って認証 OS の X 上にウィンドウを表示し、最大化させる。このように表示すると、ユーザには作業 OS の画面だけ見える。認証ダイアログはその VNC のウィンドウの上に重ねて表示する。

4.2 アクセス権のチェック

ファイルへのアクセス権のチェックは認証 OS の NFS サーバが read、readdir、write 要求を受け取る度に行う。NFS バージョン 3 では、サーバはクライアントの状態を保持しないため、要求毎にチェックを行う必要がある。NFS サーバが read または readdir 要求を受け取った時に、認証に失敗するか、ポリシでそのファイルまたはディレクトリの読み込みが許可されていなければエラーを返す。write 要求を受け取った時も同様にチェックするが、ポリシで追記のみを許可している場合は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、つまり、書き込みがファイルの末尾かをチェックする。これは NFS サーバにはファイルが追記モードでオープンされているかどうかが分からないためである。

5 実験

提案手法によりファイル 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 3.0.2 その上の VM で動く OS には Linux 2.6.12.6 を用いた。作業 OS が動く VM には 256MB のメモリを割り当てた。

SAccessor のオーバーヘッドを調べるために、ファイルの書き込みと読み込みのスループットを測定した。スループットは、100MB のファイルに対する読み込み、または書き込みを行って測定した。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 を用いた場合について行った。SAccessor を用いた場合については、ポリシの数を 0、100、500 の場合について測定した。それぞれ一度に読み書きするデータブロックサイズは 1KB、2KB、4KB、8KB の場合について測定した。読み込みについては、作業 OS にファイルキャッシュがある場合とない場合について測定した。

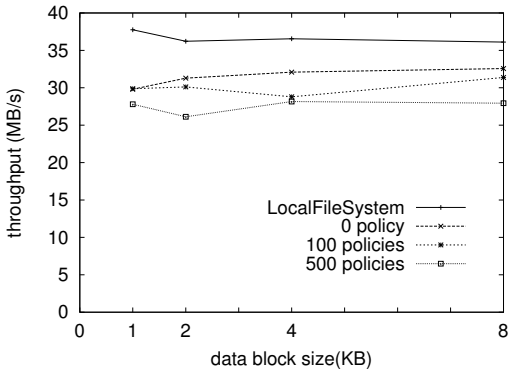

図 6: ポリシー数とファイル読み込み速度

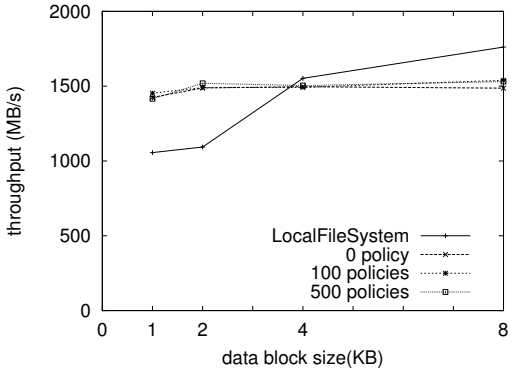

図 7: ポリシー数とファイル読み込み速度（キャッシュあり）

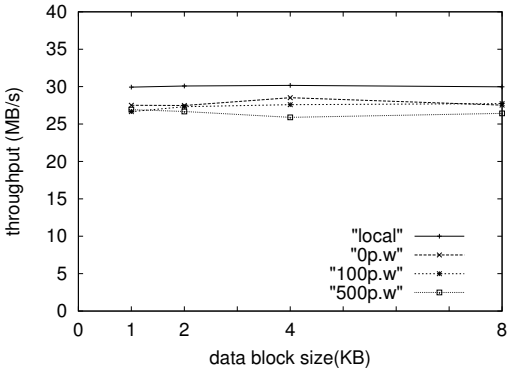

図 8: ポリシー数とファイル書き込み速度

5.1 読み込み速度

図 6 から、作業 OS にキャッシュがない場合のファイルの読み込みは、ポリシ数が増えるにしたがってスループットが低下していることが分かる。これは、現在の実装ではポリシを線形探索しているためである。ローカルファイルシステムに比べ、10%~27%のオーバーヘッドであった。図 7 は作業 OS がファイルのキャッシュを保持している場合の読み込み速度である。ファイルキャッシュを持つ場合はポリシ数によらずスループットはほぼ一定である。これは、ファイルアクセスはファイルキャッシュに対して行われるために、NFS サーバへのアクセスは行わず、アクセス権のチェックも行われないからである。したがって、ローカルファイルシステムとの違いは VM を使用していることだけである。一度に読み込むデータサイズが 1 KB、 2 KB のときに SAccessor よりもローカルファイルシステムの性能が劣る原因は調査中である。

5.2 書き込み速度

図 8 はファイルの書き込み速度の比較である。ディスクに全てのデータが書き込まれるまでの性能を測定した。ポリシ数が多くなるほどスループットが多少低下しているが、オーバーヘッドは約 10%であった。

6 議論

6.1 認証の頻度

SAccessor が行う認証の頻度にはセキュリティと利便性のトレードオフがある。ファイルを細かくグループに分けたり、認証の有効期間を短くしたりすればセキュリティを高めることができるが、認証を求められる頻度が増すため、ユーザの利便性を低下させてしまう。これは一回の認証で許可されるファイル数が少なくなり、一旦許可されたファイルに認証なしでアクセスできる期間が短くなるからである。逆に、グループに含まれるファイル数を多くしたり、認証の有効期間を長くしたりすれば、認証の頻度が減るためユーザの利便性は向上するが、その分セキュリティは低下する。セキュリティを保ちつつ、認証の頻度が適度になるようにポリシを設定する必要がある。

きめ細かく認証を行うようにすると、認証ダイアログを用いたサービス妨害（DoS）攻撃を行われる可能性も出てくる。セキュリティを高めるために多くのファイルグループを作ると、作業 OS に侵入した攻撃者は認証ダイアログを頻繁に出すために、いろいろなファイルにアクセスするという攻撃を行うことができる。ユーザはそれらの認証ダイアログに対して 1 つずつ対処しなければならないため、他の作業をすることが難しくなる。ただし、このように認証ダイアログがたくさん出るのは DoS 攻撃を受けた場合であるので、作業 OS 停止させて対処すればよい。

6.2 ファイルキャッシュ

作業 OS が認証 OS 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。認証の有効期間を過ぎたファイルがファイルキャッシュに残っていると、作業 OS は認証なしでファイルを読むことができてしまう。この問題を解決するには、認証 OS から作業 OS に認証の有効期間が過ぎたことを通知し、作業 OS のファイルキャッシュを強制的にフラッシュさせることが考えられる。作業 OS の制御が奪われたとしても、その時点でファイルキャッシュに残っているファイルは認証の有効期間内のものだけである。作

業 OS はこのファイルには認証なしでアクセスできるので、ファイルキャッシュに残っていてもセキュリティが低下することはない。

NFS は性能向上のためにクライアントのファイルキャッシュに書き込まれたファイルをしばらくしてからサーバに書き戻すため、この時間差を利用した攻撃が可能である。例えば、攻撃者が管理者権限を奪ってあるファイルに書き込むと、作業 OS のファイルキャッシュへの書き込みは成功する。このファイルを他のプロセスが読むと、ファイルキャッシュ上にある改ざんされたファイルを読むことになる。この問題を解決するには、ファイルのオープン時または最初の書き込み時に同期的に認証 OS と通信して認証させるようにすればよい。ただし、作業 OS の制御を奪われた場合にはこのような機構を強制させることはできないため別の仕組みが必要である。

また、ファイルが認証 OS の NFS サーバに書き戻される時に認証を行うことになるため、作業 OS でファイルに書き込んだ時点では認証ダイアログが表示されない。後から認証ダイアログが表示されると、どの操作に対して出された認証ダイアログであるかがユーザにはわかりにくくなり、アクセスを許可するかどうかの判断が難しくなる。その上、作業 OS でファイルに書き込んだ時にはファイルキャッシュに書かれて成功したように見えるが、後で認証に失敗すると書き込みに失敗することになるため、アプリケーションのセマンティクスに影響する可能性がある。この問題もファイルのオープン時または最初の書き込み時に認証すればよいと考えている。

7 関連研究

NFS サーバを用いるような構成にすればファイルアクセス制御をクライアントの OS から分離することはできるが、それだけでは認証を分離することができていない。NFS ではクライアントの OS の制御を奪われたら、認証をバイパスするのは容易である。Unix 認証を行っている場合、クライアント OS のユーザ ID とグループ ID を操作するだけで、任意のユーザになりすますことができる。Kerberos 認証を行っていても、クライアント OS にキーロガーを仕掛けることで容易に Kerberos パスワードを盗むことができる。SAccessor ではユーザが NFS サーバに対して直接認証できるようにすることでこの問題を解決している。

Plan9[5] 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、VM を用いてファイルサーバとクライアントを同一マシン上で動かし、認証はその場でシームレス行えるようにしている。

S4 [7] は OS が攻撃されたとしても、データの永久的な消去、ファイルアクセスの痕跡の消去を防ぐストレージシステムである。ストレージデバイスでリクエストの監視や、ファイルのバージョニングを行っている。S4 は攻撃者の侵入後に管理者の行う診断と回復の作業を助けるシステムであるのに対し、SAccessor はファイルの不正アクセス自体を防ぐシステムである。SIDS[6] は OS に依存せず、ストレージデバイスで侵入検知を行うシステムである。SIDS はストレージデバイスへのリクエストを基にクライアントの振る舞いを監視する。しかし、侵入者による攻撃と正規ユーザによるアクセスを区別することは難しいため、一般に管理者に警告を出すことしかできず、攻撃を防ぐのは難しい。

8 まとめ

本稿では、OS の制御を奪われてもファイルアクセス制御を強制できるシステム SAccessor を提案した。SAccessor では、VM を用いて一台のマシン上に認証 OS と作業 OS を動作させ、認証 OS がアクセス制御を行う。作業 OS の画面上に認証 OS が重ねて認証ダイアログを表示し、パスワードを入力させることで認証を行う。ファイルアクセス制御はファイルやディレクトリをまとめたグループごとに行い、一旦認証すれば有効期間内は再び認証する必要はない。

現在の実装では、ファイルアクセス時にポリシを線形探索しているためにオーバヘッドが大きくなっているので、探索アルゴリズムを改良することを考えている。また、作業 OS のファイルアクセスは NFS バージョン 3 を基に実装しているが、バージョ

ン 4 を用いることでサーバにファイルをオープンしたまま保持させることができ、性能を改善することができる。さらに、NFS は汎用の TCP/IP 通信を用いているためオーバーヘッドが大きい。これを Xen が提供している共有メモリによるドメイン間通信を用いるようにすることでさらにオーバーヘッドを削減できると考えている。

6 章で述べたように、作業 OS のファイルキャッシュを考慮した実装を行い、安全性を高める必要がある。また、作業 OS が別のホスト上にあるファイルサーバを使っている場合でも安全に認証を行えるように SAccessor を拡張することも検討している。作業 OS がファイルサーバのファイルにアクセスする時には、認証 OS がファイルサーバに対して認証を行う。このような構成にすることで、作業 OS が攻撃されたとしてもファイルサーバのパスワードの漏洩を防ぐことができる。

参考文献

[1] CVE-2005-0750.

[2] Cve-2005-1263. `http://cve.mitre.org/`.

[3] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer. Xen and the art of virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, October 2003.

[4] Ian Goldberg, David Wagner, Randi Thomas, and Eric A. Brewer. A secure environment for untrusted helper applications. In *Proceedings of the 6th Usenix Security Symposium*, San Jose, CA, USA, 1996.

[5] Bell Labs. Plan9:plan9 from bell labs. `http://plan9.bell-labs.com/plan9/`.

[6] A. Pennington, J. Strunk, J. Griffin, C. Soules, G. Goodson, and G. Ganger. Storage-based intrusion detection: Watching storage activity for suspicious behavior. In *Proceedings of the 12th USENIX Security Symposium*, pages 137–152, August 2003.

[7] John D. Strunk, Garth R. Goodson, Michael L. Scheinholtz, Craig A. N. Soules, and Gregory R. Ganger. Self-Securing storage: Protecting data in compromised systems. In *Proceedings of the 4th USENIX OSDI Symposium*, pages 165–180, October 2000.