セキュリティシステム保護のためのサンドボックスシステム

Sandbox System for Protecting Security Systems

尾上 浩一　　大山 恵弘　　米澤 明憲

Koichi ONOUE　Yoshihiro OYAMA　Akinori YONEZEWA

東京大学大学院 情報理工学系研究科

Graduate School of Information Science and Technology

The University of Tokyo

{koichi,oyama,yonezawa}@yl.is.s.u-tokyo.ac.jp

悪意あるプログラムコードによる不正操作やアプリケーションの脆弱性を利用した攻撃の被害を防ぐための手段として，侵入検知/防止システム，アンチウィルスシステムなどのセキュリティシステムの利用が有用である．しかしながら，近年，セキュリティシステム自体の安全性の問題が生じてきている．本論文では，セキュリティシステムの動作を監視・制御することによって，セキュリティシステムの安全性と頑健性を向上させるサンドボックスシステムを提案する．我々のシステムの利用者は，セキュリティシステムのプログラムコードを修正する必要はない．アンチウィルスシステムに我々のシステムを適用したところ，オーバーヘッドは約 1.3 倍以内におさまった．アンチウィルスシステムが異常終了したときに，我々のシステムが，異常終了を検出し，アンチウィルスシステムを再起動させることができることを確認した．

1　はじめに

悪意を持つ者による計算機システムへの不正侵入，コンピュータウィルス，トロイの木馬，ワームなどによる攻撃の脅威は依然深刻である．計算機システムへの攻撃を検知・防止する有用な手段として，アプリケーションの動作を監視・制御するセキュリティシステムがこれまでに数多く提案されている．その中には，サンドボックスシステム [11, 17, 19, 20, 22, 24]，侵入検知/防止システム [13, 16, 25]，アンチウィルスシステム [2, 3, 7] などが含まれる．これらのセキュリティシステムは現在広く普及しており，攻撃の検知・防止に大きな成果をあげている．

しかし，近年，これらのセキュリティシステム自身の安全性が脅かされるという問題が生じている．セキュリティシステム自身もプログラムであるため，脆弱性が含まれる可能性がある．たとえば最近では，アンチウィルスシステム ClamAV [1] に整数オーバフロー脆弱性 [9]，アンチウィルスシステム Sophos Anti-Virus [6] にヒープオーバフロー脆弱性 [10]，侵入検知システム Snort [5] にバッファオーバフロー脆弱性 [8] が発見されている．悪意を持つ者は，セキュリティシステムに含まれる脆弱性を利用することにより，セキュリティシステムを乗っ取ったり，強制終了させるなどの攻撃を行うことができる．セキュリティシステムへの攻撃が成功すると，攻撃を受けた計算機の防御は極めて弱いものになる．このため，セキュリティシステムを攻撃から保護することは，一般のプログラムを攻撃から保護することに比べて，はるかに重要である．

しかし，現在世界で利用されている計算機上において，多くの場合，セキュリティシステムは自身への攻撃に対する特別な防御がないままで実行されている．また，セキュリティシステム自身の安全性を高める技術については，近年研究が本格化したばかりであり，十分な知見が蓄積されていない．特に，どのような仕組みを用いて各セキュリティシステムを保護するべきか，また，保護した場合に各セキュリティシステムにどの程度の実行時オーバヘッドが加わるのか，といった点について，知見が極めて不足している．

本論文では，セキュリティシステムの安全性および頑健性を高めるためのシステムを提案する．我々の提案システムは，セキュリティシステムをサンドボックスの中で実行し，セキュリティシステムの動作を監視・制御する．提案システムは，セキュリティシステムが発行するシステムコールおよびセキュリティシステムに対して送られるシグナルを捕捉する．捕捉後に行われる処理は，提案システムの利用者が

記述するセキュリティポリシーによって指示される．提案システムは監視対象のセキュリティシステムの異常動作を検出し，強制終了や再起動などの対策を自動的に実行する．提案システムは，利用にあたりセキュリティシステムに修正の必要がないという利点を持つ．我々は，広く利用されているアンチウィルスシステム ClamAV に提案システムを適用し，性能および頑健性に関する実験を行った．

本研究の貢献を以下に示す．

- セキュリティシステム自身を監視・制御しながら実行するためのシステムを設計・実装した．
- 上記システムの利用によって加わる実行時オーバヘッドに関する実験結果を提示した．

本論文の構成を以下に示す．2 章で提案システムの設計について述べる．3 章で提案システムの ClamAV への適用について述べる．4 章で提案システムの実装について述べる．5 章で議論する．6 章で提案システムの実験結果を示す．7 章で関連研究を述べる．8 章でまとめと今後の課題について述べる．

2 設計

提案システムが保護するセキュリティシステムは，既存のサンドボックス，侵入検知/防止システム，アンチウィルスシステムである．提案システムの利用者は，既存のセキュリティシステムのバイナリを修正することなく，提案システムをセキュリティシステムに適用することが可能である．

提案システムでは，セキュリティシステムが発行するシステムコールおよびセキュリティシステムへ送られるシグナルを捕捉する．捕捉後のセキュリティシステムの動作は，提案システムの利用者が記述したセキュリティポリシーに基づいて制御される．セキュリティシステムが複数のモジュールに分割されているとき，提案システムはモジュールごとに動作の監視・制御を行う．

セキュリティシステムを監視・制御する提案システムを図 1 に示す．図 1 中では，3 つのモジュールからなるセキュリティシステムを示している．本論文では，セキュリティシステムの各モジュールを対象モジュールと呼ぶ．

提案システムは 2 種類のモジュール (管理モジュール，監視モジュール) を用いて，セキュリティシステムの監視・制御を行う．管理モジュールは，すべての監視モジュールを管理するモジュールであり，監視モジュール の起動や異常終了の検出，依存関係のある監視モジュール間の通信の仲介を行う．監視モジュールは，提案システムの利用者が記述したセキュリティポリシーに基づき，対象モジュールの動作制御を行う．監視機構を複数に分割することにより，提案システムでは，異常動作をした監視モジュールの影響が及ぶ範囲が小さくなる．

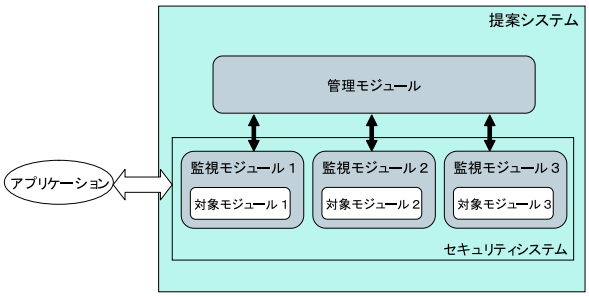

図 1: 提案システムの構成

管理モジュールは，すべての対象モジュールの制御が可能な実行権限で動作する．異なるユーザ権限で対象モジュールが動作する場合，多くの場合，管理者権限が必要となる．監視モジュールは対象モジュールと同じ実行権限で動作する．

提案システムの枠組みは既存のサンドボックスや侵入検知/防止システムと特別な違いはない．提案システムの特長は，セキュリティシステムを監視・制御し，セキュリティシステムの安全性・頑健性を向上させることである．

3 ClamAV への適用

3.1 ClamAV

ClamAV は Linux，FreeBSD，Mac OS X などで利用可能なアンチウィルスシステム ClamAV では，ウィルス検査に Aho-Corasick パターンマッチングアルゴリズム [12] を応用したシグネチャマッチングを利用している．ClamAV はウィルス検査対象のファイルとして，メール関連のファイル（`mbox`，`Mail` ディレクトリなど），アーカイブファイル・圧縮ファイル（`tar`，`gzip`，`bzip2`，`RAR`，`OLE2`），実行ファイル（`UPX`，`FGS`，`Petite`）に対するウィルス検査をサポートしている．ClamAV は，現在，37000 以上のウィルス，ワーム，トロイの木馬などを検出するためのシグネチャを保持している．ClamAV は，ウィ

ルス検査デーモン，コマンドライン検査，インターネット経由のウィルスデータベースの自動更新のためのツールなどを利用者に提供する．

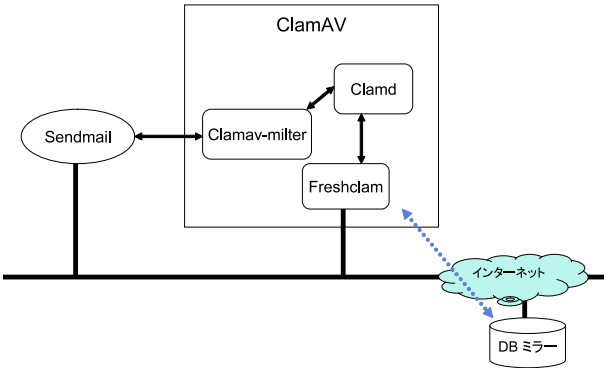

図 2: Sendmail への適用

本研究では，ClamAV が提供するツールを Sendmail [4] に適用する (図 2)．Sendmail により，メールの送信および LAN 内からのメールの受信が行われたときに，ClamAV が提供するツールがメールのウィルス検査を行う．ClamAV と Sendmail を連携させるために milter 機能を有効にする．

図 2 中の ClamAV が提供するツールについて以下に述べる．各ツールはユーザ (`clamav`) 権限で動作する．

Freshclam HTTP 経由でインターネット上のミラーサーバと通信し，周期的にウィルスデータベースを更新するデーモン．

Clamd Clamav-milter からの要求に応じて，ウィルス検査を行うサーバ．Clamav-milter との通信には UNIX ドメインソケットを利用．

Clamav-milter Sendmail と Clamd を仲介するデーモン．Sendmail からのメールの送受信時に，Clamd にメールのウィルス検査を要求．Clamd からの検査結果を Sendmail に返信．Sendmail との通信には UNIX ドメインソケットを利用．

3.2 ClamAV の保護

提案システムにより ClamAV を保護する例を図 3 に示す．提案システムは，ClamAV が提供するツールを対象モジュールとみなし，監視モジュールで監視・制御を行う．管理モジュール，監視モジュールはユーザ (`clamav`) 権限で動作する．

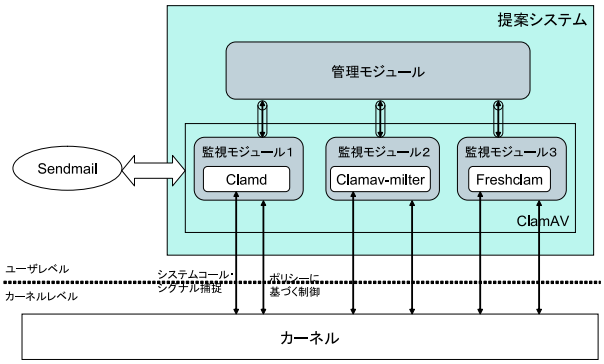

図 3: 提案システムによる ClamAV の保護

管理モジュール，監視モジュールのセキュリティポリシーを設定するために，提案システムでは 2 種類のポリシーファイルを利用する．

```
/* セキュリティシステムが監視するアプリケーション名 */
Sendmail

/* [対象モジュール名]  [ポリシーファイル名] */
   Clamav-milter       clamav-milter.pol
   Clamd               clamd.pol
   Freshclam           freshclam.pol
   ...

/* 対象モジュール間の依存関係 */
   Freshclam  <---  Clamd
   ...
```

図 4: 管理モジュールが利用するポリシーの例

管理モジュールが利用するポリシーファイルの例を図 4 に示す．ポリシーファイルに，提案システムの利用者がセキュリティシステムが監視・制御するアプリケーション名，保護するセキュリティシステムの対象モジュールの管理情報を記述する．提案システム利用者は，対象モジュールの管理情報として，識別名，ポリシーファイル名を記述する．さらに，必要があれば，2 つの対象モジュール M_1，M_2 間の依存関係を記述する．依存関係は対象モジュールを再起動するときに利用される．依存関係 $M_1 \longleftarrow M_2$ は，M_2 が M_1 に依存していることを表す．依存関係があるときに M_1 を再起動する場合には，まず，M_2 を制御する監視モジュールが M_2 を一時停止させる．その後，M_1 を制御する監視モジュールが M_1 を再起動し，最後に M_2 を制御する監視モジュールが M_2 を再開させる．

監視モジュールが利用するポリシーファイルを図 5 に示す．ポリシーファイルには，対象モジュールを実行するユーザ名，起動時の実行コマンドが記述され

```
clamav                  /* 実行ユーザ名     */
clamd -c clamd.conf     /* 起動時のコマンド*/

/* システムコールに関するポリシー */
System call:
-------------------------------------
Default:     allow
execve       deny
...

/* シグナルに関するポリシー */
Signal:
-------------------------------------
Default:     deny
SIGKILL      restart
SIGTERM      restart
...
```

図 5: 監視モジュールが利用するポリシーの例

る．記述されたユーザ名は実行権限を表す．さらに，各システムコール，シグナルを捕捉したときの対象モジュールの制御方法も記述する．現在，提案システム では，対象モジュールが発行したシステムコールの許可・拒否，対象モジュールの強制終了・再起動が可能である．ひとつの監視モジュールが制御している対象モジュールを強制終了した場合，セキュリティシステムの機能の一部が欠如した状態になる．このため，現在提案システムでは，他の対象モジュールも各監視モジュールによって強制終了させている．

4 実装

4.1 管理モジュール

管理モジュールは，図 4 で示したようなポリシーファイルに基づき，子プロセスとして各監視モジュールを起動する．提案システムでは，pipe(2) を利用したプロセス間通信により，管理モジュールと監視モジュールの通信をう．管理モジュールは，生成した通信経路を利用して，異なる監視モジュール間の通信の仲介，監視モジュールの状態管理を行う．

対象モジュール M_1 を再起動する場合，M_1 を制御する監視モジュール は管理モジュールに M_1 に依存した対象モジュールがあるか確認する．管理モジュールは，ポリシーファイルに M_1 に関する依存関係の記述があった場合，管理モジュールは依存した対象モジュールを制御する監視モジュールに停止通知を発行する．管理モジュールは，停止を確認後，M_1 を制御する監視モジュール に再起動が可能であることを通知する．最後に，管理モジュールは，M_1 の再起動後を確認後，停止中の対象モジュールを制御する監視モジュールに再開が可能であることを通知する．

提案システムでは，現在，管理モジュールが監視モジュール起動時に pipe(2) により確保した通信経路の切断を認識することにより，監視モジュールが異常終了したことを検知する．検知後，管理モジュールは，必要があれば，異常終了した監視モジュールを再起動させる．

4.2 監視モジュール

ptrace(2) を利用することにより，監視モジュールが，対象モジュールが発行するシステムコールをユーザレベルで捕捉できる．監視モジュールは，waitpid(2) を利用し，システムコールおよび対象モジュールに送られるシグナルを捕捉する．監視モジュールは，対象モジュールが実行中に新たに生成したプロセスの監視・制御も行う．システムコールおよびシグナルを捕捉すると，監視モジュールは，図 5 で示したようなセキュリティポリシーに基づき，対象モジュールを制御する．

システムコールおよびシグナルの制御に関して，提案システムは，現在，監視モジュールのセキュリティポリシーよりも管理モジュールからの要求を優先する．たとえば，管理モジュールから対象モジュールの強制終了要求があった場合，SIGKILL に関して再起動するようなセキュリティポリシーが記述されていても，提案システムでは対象モジュールを強制終了させる．

対象モジュール M_1 を制御する監視モジュールが M_1 を再起動する場合，M_1 に依存する対象モジュール M_2 があるか，管理モジュールに確認する．依存した M_2 がある場合，M_2 を監視する監視モジュールは，管理モジュールから M_2 の停止を通知される．M_2 を監視する監視モジュールは，M_2 を停止させ，管理モジュールを経由して，M_1 を制御する監視モジュールに停止完了を通知する．そして，M_1 を制御する監視モジュールは，対象モジュールを再起動させ，管理モジュールを経由して，M_2 を制御する監視モジュールに再起動完了を通知する．最後に，M_2 を制御する監視モジュールが M_2 を再開させる．

監視モジュールが対象モジュールを強制終了させる場合，管理モジュールを経由して，すべての監視モジュールに対象モジュールを強制終了するように通知する．通知を受けた各監視モジュールは対象モジュールを強制終了し，管理モジュールが提案システムを終了させる．

5 議論

対象モジュール M_1, M_2 間で通信セッションを維持する必要がある場合，対象モジュールの再起動が問題となる．M_1 の再起動を行うと，通信セッションが切断されてしまうため，多くの場合，M_2 も再起動する必要がある．

本章では，提案システム M_2 を再起動させずに，M_1 を再起動させる手法について述べる．ただし，以下の手法は再起動前後で，M_1, M_2 間で実行状態を保持する必要がある場合には有効でない．

図 4，5 で示したポリシーファイルに対象モジュール間のセッションに関する記述を追加する．ポリシーファイルの記述により，対象モジュールは起動時に対象モジュール間でセッションを維持する必要があるか確認する．通信セッションを維持する必要がある場合，対象モジュールが `socket(2)`，`connect(2)` などのシステムコールの発行前に，対応する監視モジュールが，プロセスの複製を行うようにする．プロセスの複製のときに，監視モジュールが `CLONE_FILES` を設定し，対象モジュールに `clone(2)` を呼び出させる．`CLONE_FILES` を設定することで，親子プロセス間で開いたファイルディスクリプタを共有できる．プロセス複製後，監視モジュールは子プロセスを停止させ，対象モジュールの動作を親プロセスで監視する．

対象モジュールの再起動が必要になったとき，監視モジュールは対象モジュールの処理を停止している子プロセスに切り換え，親プロセスを終了する．子プロセスは，上記と同様の方法で `clone(2)` を用いてプロセスを複製し，生成された子プロセスを停止し，親プロセスを再開させる．

多くの場合，対象モジュールの再起動は，対象モジュールが異常動作をしたときに行われる．このため，`clone(2)` 呼び出しでメモリ空間の共有は行わない．このため，再起動前後で状態を維持することはできない．

6 実験

提案システムの実装を行い，以下の 2 つの実験を行った．実験環境は CPU Pentium 4 3.0 GHz (Hyper Threading 有効)，メモリ 2 GB，OS Linux 2.6.12 である．実装に利用した Sendmail，ClamAV のバージョンは各々 8.13.4-3，0.84 である．

		提案システム 無	提案システム 有	比 (有/無)
clamscan	Files 1	7.16	7.32	1.02
	Files 2	1.18	1.54	1.31
clamdscan	Files 1	6.75	6.84	1.02
	Files 2	0.86	1.07	1.24

表 1: clamscan，clamdscan の実行時間 (秒)

6.1 実行時オーバーヘッド

提案システムは，監視モジュールが対象モジュールの発行するシステムコールおよびシグナルを捕捉し，ポリシーファイルの記述に従って対象モジュールを制御する．提案システムの利用に伴い，セキュリティシステムの実行時にオーバーヘッドがどの程度加わるかを計測するための簡単な実験を行った．

オーバーヘッドを測定するために，ClamAV が提供する以下の実行コマンド検査ツールを利用した．

clamscan コマンド 引数で与えたディレクトリまたはファイルに対し，ウィルスデータベースを解凍・展開し，ウィルス検査を行う．この場合，clamscan を対象モジュールとみなし，オーバーヘッドを測定した．

clamdscan コマンド 引数で与えたディレクトリまたはファイルに対し，Clamd との通信によりウィルス検査を行う．Clamd は周期的にウィルスデータベースを解凍・展開する．そして，clamdscan コマンドからの要求時には，clamd は展開したウィルスデータベースを利用してウィルス検査を行う．このため，clamdscan コマンドは clamscan コマンドよりも効率的にウィルス検査を行なえる．この場合，clamdscan，Clamd を対象モジュールとみなし，オーバーヘッドを測定した．

検査対象のファイルとしては以下の 2 種類のファイルを利用した．

Files 1 ウィルスを含まない 100 個の正常な実行ファイル GoogleEarth.exe(約 40MB).

Files 2 ウィルスを含む 300 個の実行ファイル clam.exe，clam.exe.bz2，clam.zip (約 330KB)

実験結果を表 1 に示す．提案システムの実行時オーバーヘッドは約 1.3 倍以内におさまり，提案システムを利用して，セキュリティシステムを保護すること

は，ある程度現実的であると考えられる．この実験では，clamscan，clamd，clamdscan が発行するシステムコールを捕捉したときに行われる，セキュリティポリシーの検査が単純であるため，オーバーヘッドがある程度小さく抑えられていると考えられる．セキュリティシステムが発行するシステムコール列やシステムコール引数の検査を追加した場合，オーバーヘッドが加わる可能性がある．

6.2 対象モジュールの異常回復の確認

提案システムの回復機能を検証するために，図 3 で示した ClamAV の保護に関する簡単な実験を行った．

この実験では，我々は管理モジュールが利用するセキュリティポリシーファイルに Clamd が Freshclam に依存しているという記述をし，提案システムを起動させた．そして，コマンドラインから Freshclam を強制終了させるためのシグナルを送った．このとき，以下の手順で提案システムが，Freshclam の異常終了を検知し，再起動を行っていることを確認した．

Freshclam を制御する監視モジュール M_f は，Clamd を制御する監視モジュール M_c に Clamd を停止するように通知する．M_c は Clamd を停止し，M_f に Clamd の停止完了を通知する．そして，M_f は Freshclam を再起動させ，M_c に Clamd の再開が可能であるという通知を行う．最後に，M_c が Clamd を再開させる．

7 関連研究

アプリケーションを安全に実行するためのサンドボックスについては非常に多くの研究が存在する [11, 17, 19, 20, 22, 24]．しかし，そのアプリケーションがセキュリティシステムである場合に，どのようなサンドボックスを構築すればセキュリティシステムを安全かつ頑健に高い性能で運用できるかという問題を扱った研究は極めて少ない．本研究はその問題を解決する試みの第一歩である．

セキュリティシステムを仮想マシン内に隔離して運用する方式が最近提案されている [15, 18, 21]．これらの方式を用いると，セキュリティシステムが乗っ取られた場合でも，攻撃の被害は仮想マシンの中に限定される．仮想マシンを用いる方式の問題は，大きいオーバヘッドである．本研究は，システムコールやシグナルの監視という，仮想化よりも軽量な処理を通じてセキュリティシステムの安全性を高めることを目標としている．また，本研究では，実行環境を隔離するだけではなく，セキュリティシステムの各動作に対して提案システムが取る反応を，セキュリティポリシーを通して設定できるようにしている．

吉野らによる SeRene [26] は，自身が攻撃されたり，自身にバグがあっても，できる限り稼動を続けられるような仕組みを備えたリファレンスモニタである．SeRene は独立したモジュールの集合体として実現される．各モジュールは監視を受けながら実行され，異常な動作を行ったりハングアップしたモジュールは強制的に再起動される．SeRene の研究では，セキュリティシステムのコード中に自己保護の機構を組み込んでいる．一方，本研究は，広範囲のセキュリティシステムを保護できるような汎用のシステムを提案している．

文献 [14] では，ネットワークベース侵入検知システム Bro [23] に対して処理量を爆発させる攻撃を仕掛けることにより，Bro の監視能力を著しく低下させられることが報告されている．提案システムは，現在，セキュリティシステムの性能を低下させたり，セキュリティシステムをハングアップさせたりする攻撃には対処できない．それらの攻撃への対策は今後の課題である．

8 まとめと今後の課題

本論文では，侵入検知/防止システムやアンチウィルスシステムなどの既存のセキュリティシステムを保護するシステムを提案した．提案システムは，利用にあたり，セキュリティシステムのバイナリを修正する必要がない．ClamAV に提案システムを適用したところ，実行時オーバーヘッドは約 1.3 倍以内におさまった．ClamAV が提供するツールが，異常終了から回復していることの確認を行った．

今後の課題としては，まず，提案システムの安全性や頑健性の向上させたい．管理モジュールは管理者権限で動作する可能性があり，監視モジュールの管理を行なっているため，提案システムでは管理モジュールを保護することがより重要となる．次に，より多くの既存のシステムに提案システムを適用し，有用性を確かめたい．そして，実用的にするために必要なセキュリティポリシーの考察を行いたい．また，オペレーティングシステムを拡張することにより，管理モジュールと監視モジュールが行える操作をより柔軟にしたい．

参考文献

[1] *Clam AntiVirus*. http://www.clamav.net/.

[2] *McAfee - Antivirus Software and Intrusion Prevention Solutions*. http://www.mcafee.com/us/.

[3] *Norton AntiVirus*. http://www.symantec.com/nav/nav_9xnt/.

[4] *Sendmail Home Page*. http://www.sendmail.org/.

[5] *Snort - the de facto standard for intrusion detection/prevention*. http://www.snort.org/.

[6] *Sophos Anti-Virus*. http://www.sophos.com/.

[7] *TREND MICRO*. http://www.trendmicro.com/en/home/us/personal.htm/.

[8] *Buffer overflow in Snort RPC preprocessor*, March 2003. http://www.kb.cert.org/vuls/id/916785/.

[9] *ClamAV Library Rem0te Heap Overflows Security Advisory*, July 2005. http://www.rem0te.com/public/images/clamav.pdf/.

[10] *Sophos AntiVirus Products Remote Heap Overflow vulnerability*, July 2005. http://www.frsirt.com/english/advisories/2005/1244/.

[11] A. Acharya and M. Raje. MAPbox: Using Parameterized Behavior Classes to Confine Untrusted Applications. In *Proceedings of the 9th USENIX Security Symposium*, Denver, August 2000.

[12] A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, Vol. 18, No. 7, 1975.

[13] K. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. Keromytis. Detecting Targeted Attacks Using Shadow Honeypots. In *Proceedings of the 14th USENIX Security Symposium*, Baltimore, July 2005.

[14] S. Crosby and D. Wallach. Denial of Service via Algorithmic Complexity Attacks. In *Proceedings of the 12th USENIX Security Symposium*, Washington, DC, August 2003.

[15] G. Dunlap, S. King, S. Cinar, M. Basrai, and P. Chen. ReVirt: Enabling Intrusion Analysis through Virtual Machine Logging and Replay. In *Proceedings of the 5th Symposium on Operating Systems and Implementation*, Boston, December 2002.

[16] D. Gao, M. Reiter, and D. Song. Gray-box Extraction of Execution Graphs for Anomaly Detection. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, Washington, DC, October 2004.

[17] T. Garfinkel, B. Pfaff, and M. Rosenblum. Ostia: A Delegating Architecture for Secure System Call Interposition. In *Proceedings of the 11th Annual Network and Distributed System System Security Symposium*, San Diego, February 2004.

[18] T. Garfinkel and M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *Proceedings of the 10th Annual Network and Distributed Systems Security Symposium*, San Diego, February 2003.

[19] I. Goldberg, D. Wagner, R. Thomas, and E. Brewer. A Secure Environment for Untrusted Helper Applications. In *Proceedings of the 6th USENIX Security Symposium*, San Jose, July 1996.

[20] K. Kato and Y. Oyama. SoftwarePot: An Encapsulated Transferable File System for Secure Software Circulation. In *Software Security – Theories and Systems*, Vol. 2609, pp. 112–132, February 2003.

[21] K. Kourai and S. Chiba. HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection. In *Proceedings of the 1st ACM/USENIX International Conference on Virtual Execution Environments*, Chicago, June 2005.

[22] Z. Liang, V. Venkatakrishnan, and R. Sekar. Isolated Program Execution: An Application Transparent Approach for Executing Untrusted Programs. In *Proceedings of 19th Annual Computer Security Applications Conference*, Las Vegas, December 2003.

[23] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks (Amsterdam)*, Vol. 31, No. 23–24, pp. 2435–2463, 1999.

[24] N. Provos. Improving Host Security with System Call Policies. In *Proceedings of the 12th USENIX Security Symposium*, Washington, DC, August 2003.

[25] D. Wagner and D. Dean. Intrusion Detection via Static Analysis. In *Proceedings of the 2001 IEEE Symposium on Security and Provacy*, Oakland, May 2001.

[26] 吉野, 大山, 米澤. 自己修復型リファレンスモニタの設計と実装. 第 7 回プログラミングおよび応用のシステムに関するワークショップ (SPA2004) 論文集, 長野県, March 2004.