6W-02

重要サービスの特定を困難化する通信処理制御法

奥田 勇喜† 佐藤 将也‡ 谷口 秀夫‡
†岡山大学工学部 ‡岡山大学大学院自然科学研究科

1 はじめに

マルウェアによる攻撃への対策として，重要サービスの動きを攻撃者に見せない（不可視化する）技術がある．我々は，重要サービスのプロセス情報やファイル操作を不可視化する技術を提案した [1][2]．本稿では，重要サービスの通信操作の不可視化，つまり重要サービスの特定を困難化する通信処理制御法を述べる．

2 通信処理制御法

2.1 要求と課題

重要サービスの特定を困難化する通信処理制御法に対し，次の要求がある．重要サービスの通信内容をもとにして重要サービスの存在が攻撃者に特定され，無効化される可能性がある．このため，重要サービスの通信内容を不可視化する（要求 1）必要がある．

また，攻撃を回避するために重要サービスのプログラムを変更すると，重要サービスを攻撃内容に合わせて個別に変更しなくてはならない．この変更の工数は大きいため，重要サービスのプログラムを変更せずに制御法を実現する（要求 2）必要がある．

通信処理制御法の確立には,（要求 1）を満足する形で，以下の課題に対処する必要がある．

（課題 1）重要サービスの OS への通信要求を捕捉できること
（課題 2）捕捉した通信要求の内容を取得できること
（課題 3）取得した内容に基づき，通信処理を代理実行できること
（課題 4）実行結果を重要サービスに返却できること

また,（要求 2）を満足するために，仮想計算機 (Virtual Machine，以降 VM) 上で OS や重要サービスを動作させ，仮想計算機モニタ（VM Monitor，以降 VMM）を変更することにより通信処理制御法を確立する．詳細は 2.2 節で述べる．

2.2 対処

2.2.1 基本構造

通信処理制御法の基本構造を図 1 に示す．保護対象 VM の OS 上では，重要サービスを提供するプロセス（以降，重要プロセス）を動作させる．また，管理 VM の OS 上では，システムコールを代理実行するプロセス（以降，代理プロセス）を動作させる．VMM は，保護対象 VM 上のシステムコールを捕捉し，重要サービスによる通信要求の内容を取得する．これを管理 VM 上の代理プロセスに転送し，システムコールの代理実行を依頼する．代理プロセスによる代理実行後，VMM は代理実行の結果を重要プロセスに返却する．

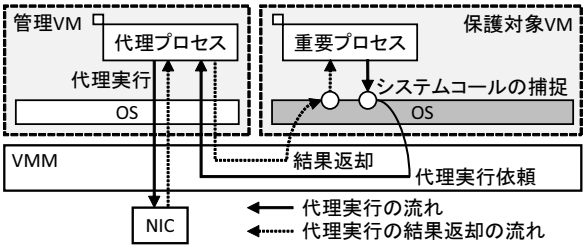

図 1 基本構造

2.2.2 対処内容

（課題 2）と（課題 3）については，図 1 の基本構造にすることにより，必要となる対処を VMM と管理 VM（代理プロセスを含む）で実現する．

（課題 1）への対処を以下に述べる．VMM は，デバッグレジスタを操作し，保護対象 VM 上で当該アドレスの命令が実行される直前にデバッグ例外が発生するように設定する．さらに，保護対象 VM の実行制御フィールドを操作し，デバッグ例外の発生により VM exit が発生するように設定する．

（課題 4）への対処を以下に述べる．代理プロセスは，システムコールの代理実行後，結果を VMM に返却する．VMM は，代理実行の返り値を保護対象 VM の rax レジスタに格納する．データを受け取る read() や recv() などのシステムコールを代理実行した場合は，文字列を重要プロセスのバッファの領域に格納する．この後，命令ポインタにシステムコール終了処理の先頭アドレスを設定し，保護対象 VM に処理を返却する．

2.2.3 通信処理制御法の処理流れ

通信処理制御法について，処理流れを図 2 に示す．

(1) 保護対象 VM におけるシステムコールの発行により VM exit が発生し，VMM に処理が遷移する．VMM に処理が遷移した際に，VM exit の発生原因がデバッグ例外であり，かつ例外の発生したアドレスがシステムコール処理の開始アドレスと一致した場合，本制御法の導入により発生したデバッグ例外として，システムコールの発行を検知する．

(2) システムコールを発行したプロセスが重要プロセスの場合は，システムコールの種類を判別する．重要プロセス以外は (15) に移行する．

(3) システムコールの種類に応じて分岐する．本制御法

Communication Control Method for Complicating Identification of Essential Services.
Yuuki Okuda†, Masaya Sato‡, Hideo Taniguchi‡
†Faculty of Engineering, Okayama University
‡Graduate School of Natural Science and Technology, Okayama University

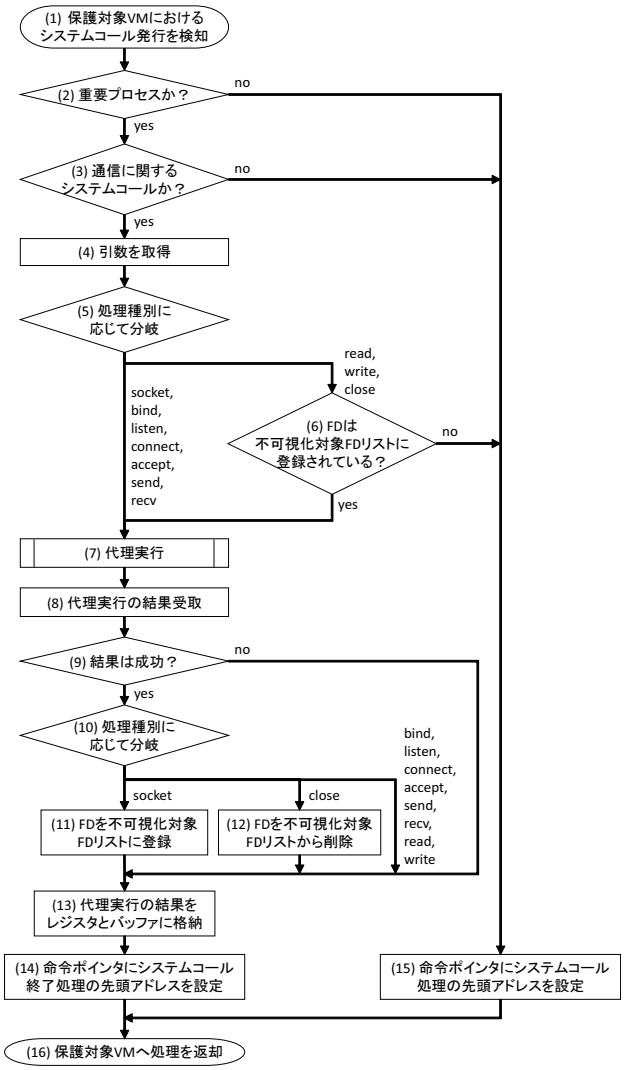

図 2　システムコール制御の処理流れ

では，通信に着目しているため，通信に関するシステムコールの場合は (15) に移行する．
(4) システムコールの引数を取得する．
(5) 通信に関するシステムコールのうち，socket の場合は (7) に，それ以外の場合は (6) に移行する．
(6) 引数で与えられた FD の値が不可視化対象 FD リストに登録されている場合は (7) に，登録されていない場合は (15) に移行する．
(7) 代理プロセスに代理実行を依頼する．
(8) 代理プロセスから代理実行の結果を受け取る．
(9) 結果が成功の場合は (10) へ移行する．結果が失敗の場合は (13) へ移行する．
(10) システムコールの種類別に分岐する．socket の場合は (11) に移行する．close の場合は (12) に移行する．それ以外の場合は (13) に移行する．
(11) 重要プロセスの新たなコネクションが確立されたため，FD を不可視化対象 FD リストに登録する．
(12) 重要プロセスのコネクションが削除されたため，FD を不可視化対象 FD リストから削除する．
(13) 代理実行の結果をレジスタに格納する．必要な場合は，保護対象 VM 上のバッファに結果を格納する．
(14) 命令ポインタにシステムコール終了処理を設定する．これにより，保護対象 VM では，システムコール処理の開始直後に終了処理が呼び出される．
(15) システムコール処理を保護対象 VM 上で継続するために，保護対象 VM の命令ポインタにシステムコール処理の先頭アドレスを設定する．
(16) 保護対象 VM に処理を返却する．

表 1　代理実行にかかるオーバヘッド

システムコール	平均値 [μs]	最大値 [μs]	中央値 [μs]	最小値 [μs]	分散
socket()	13.49	16.54	12.95	12.28	1.34
sendto()	18.68	22.39	18.40	17.87	0.63

3　評価

3.1　内容

システムコールの代理実行にかかるオーバヘッドを測定した．具体的には，システムコールを代理実行せずに返り値を返却するように改変した代理プロセスを用いて，重要プロセスが発行するシステムコールの発行直前から直後までの実行時間を測定した．なお，システムコールを 100 回発行し，送信データサイズは 1KB とした．

評価は，VMM として Xen 4.2.3 を用い，管理 VM と保護対象 VM 上で動作する OS として Debian 7.3（Linux 3.2.0 64bit）を用いた．管理 VM は Xen の仕様のため準仮想化し，7GB のメモリ，3 コアの仮想 CPU を割り当てた．保護対象 VM は VT-x を用いて完全仮想化し，1GB のメモリ，1 コアの仮想 CPU を割り当てた．8GB のメモリと Intel Core i7-2600（3.4GHz，4 コア）の CPU を搭載した計算機を使用した．

3.2　考察

測定結果を表 1 に示す．平均値が socket() は約 13μs，sendto() は約 18μs であるため，sendto() は socket() よりオーバヘッドが大きい．これは，VMM が sendto() を捕捉したときにバッファをコピーするためであると考える．

4　おわりに

通信内容をもとにした重要サービスの特定を困難化する通信制御法について述べた．具体的には，2 つの要求を満たすための 4 つの課題と対処を示し，処理の流れを述べた．評価より，sendto() は socket() よりオーバヘッドが大きいことを示した．

謝辞　本研究の一部は JSPS 科研費 16K16067，16H02829 の助成を受けたものです．

参考文献

[1] Masaya Sato, Toshihiro Yamauchi, Hideo Taniguchi: Process Hiding by Virtual Machine Monitor for Attack Avoidance, Journal of Information Processing, Vol.23, No.5, pp.673–682 (2015).
[2] 佐藤 将也，山内 利宏，谷口 秀夫：仮想計算機を用いた重要ファイル保護手法，情報処理学会シンポジウムシリーズ　コンピュータセキュリティシンポジウム 2017（CSS2017）論文集，Vol.2017，No.2，pp.1302–1308 （2017）．