情報処理学会論文誌 Vol. 50 No. 8 1870–1882 (Aug. 2009)

推薦論文

ネットワークタイムスタンプによるリモート仮想マシンモニタ検出

嶋村 誠[†1] 河野健二[†1,†2]

仮想マシンモニタがセキュリティ上の重要性を増しており，仮想マシンモニタを応用したセキュリティシステムが多数提案されている．攻撃者は仮想マシンモニタを利用したホストに侵入し，攻撃の手口などの情報を提供してしまうのを回避するため，攻撃対象ホスト上の仮想マシンモニタの検出を行うことがある．現在の仮想マシンモニタの検出手法では，攻撃者は攻撃対象ホストに侵入した後に仮想マシンモニタの検出を行う．したがって，管理者は攻撃者が用いた攻撃メッセージを取得することができる．しかし，攻撃者がホストに侵入せずに，リモートスキャンによる仮想マシンモニタの検出ができると，管理者の用意したこれらのセキュリティシステムをより容易に回避できるようになってしまう．したがって，仮想マシンモニタを利用したセキュリティシステムの有用性を高めるために，リモートスキャンによる仮想マシンモニタの検出とその対策について議論する必要がある．本論文では，ネットワークタイムスタンプを用いることで，ホストに侵入することなく仮想マシンモニタを検出できることを示す．提案手法では，2 種類のタイムスタンプ（ICMP と IP のタイムスタンプ）を 1 パケットで同時に取得し，2 つのタイムスタンプのずれ方を調べることで，仮想マシンモニタを検出する．対象ホストが仮想マシンモニタを使用していない場合，2 つのタイムスタンプはほとんど同じ値を示す．しかし，対象ホストが仮想マシンモニタを使用している場合，2 つのタイムスタンプの処理の間に仮想マシンモニタが割り込んで処理を行うことがあるため，これらのタイムスタンプのずれ方に違いが生じる．実際に，様々な仮想マシンモニタについて，仮想マシンモニタが動作するホストと仮想マシンモニタが動作していないホストに対して実験を行い比較することで，仮想マシンの検出が可能であることを示す．

Remote Virtual Machine Monitor Detection Using Network Timestamp

Makoto Shimamura[†1] and Kenji Kono[†1,†2]

Virtual machine monitors (VMMs) play important roles in security and many researchers have proposed VMM-based security systems. To avoid being trapped by VMM-based security systems and providing information of attacks, attackers try to detect VMMs. In current VMM detection techniques, attackers detect VMMs after compromising the target, and thus administrators can obtain the used messages for an attack. However, if attackers start to exploit remote VMM detection, that can detect VMMs without compromising the target, they can easily avoid VMM-based security systems. Thus we must discuss remote VMM detection and countermeasures to improve the efficiency of VMM-based security systems. In this paper, we present a technique for remotely detecting a VMM without compromising the target using network timestamp. The technique examines discrepancy between two timestamps (ICMP and IP timestamps) stamped in one packet for evidence of the presence of a VMM. If the target host does not use a VMM, the timestamps indicates almost the same time. But if the target host uses a VMM, some of the timestamps will show discrepancies in an anomalous way because the VMM sometimes interrupts the timestamp operations of the target and does some operations. By comparing the experimental results for virtual hosts that use various VMMs and a real host, we show that remote VMM detection is feasible.

1. はじめに

仮想マシンモニタ[1)–3)]は，複数の仮想マシンを 1 つのハードウェア上で動作させるソフトウェアであり，サーバ統合などの目的で広く利用されている．また，近年では仮想マシンモニタを応用したセキュリティシステムが提案されている．たとえば，仮想ハニーポット[4)–8)]は，仮想マシンモニタを利用したおとりホストであり，仮想マシンモニタにおいて

†1 慶應義塾大学理工学部情報工学科
Department of Information and Computer Science, Faculty of Science and Technology, Keio University
†2 科学技術振興機構 CREST
CREST, Japan Science and Technology Agency
本論文の内容は 2008 年 5 月のコンピュータセキュリティ研究会にて報告され，同研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

攻撃の記録や検知を行うシステムである．これにより，仮想ハニーポットでは，システムの存在を攻撃者に気づかれることなく，攻撃の記録や検知を行うことができる．

しかし，攻撃者はこのような仮想マシンモニタを利用したセキュリティシステムを持つホストを攻撃することを避けようとする．なぜなら，攻撃者は自分の作成した攻撃に関する情報を管理者に与えたくないためである．たとえば，仮想ハニーポットに攻撃を行うと，攻撃の解析に必要な情報を管理者に与えてしまう．この情報をもとに，管理者は攻撃の検知に必要なパターンファイルを作成するなどの防御手段を講ずることができる．そのため，攻撃者は攻撃行動を行う前に，攻撃対象ホストが仮想マシンモニタを利用したセキュリティシステムを用いているかどうかを調査することが多い．実際に，Agobot[9] ワームは，攻撃対象のホストに侵入した後で，仮想マシンモニタやデバッガといった攻撃行動を監視するためのシステムの検出を試み，そのようなシステムを検出すると攻撃行動を終了する．なお，仮想マシンモニタの検出自体は必ずしも仮想マシンモニタを用いたセキュリティシステムの検出を意味するわけではない．しかし，攻撃者は仮想マシンモニタを検出することで，セキュリティシステムを回避することができる．

既存の仮想マシンモニタの検出手法は，ローカル仮想マシンモニタ検出とリモート仮想マシンモニタ検出の 2 種類に大別される．ローカル仮想マシンモニタ検出[10)–15)] では，攻撃者は攻撃対象ホストに侵入した後で仮想マシンモニタを検出する．このため，仮想ハニーポットは，攻撃者がローカル仮想マシンモニタ検出により仮想マシンモニタを検出し攻撃を中止したとしても，侵入のために使用された攻撃メッセージなどの情報を収集できる．したがって，攻撃者は管理者に情報を与えないために，ローカル仮想マシンモニタ検出ではなく，リモート仮想マシンモニタ検出によって事前に仮想マシンモニタを検出するようになると考えられる．

リモート仮想マシンモニタ検出では，攻撃者はリモートホストから対象ホストにアクセスを行い，仮想マシンモニタを検出する．既存のリモート仮想マシンモニタ検出手法では，複数のホストから取得したハードウェア情報を照合することで，同一ホスト上の仮想マシン群を検出する手法[16)] や，対象ホストに root 権限でアクセスし，攻撃者のマシンの時計を用いて対象ホスト上でのベンチマークの処理時間を計測することで仮想マシンモニタによるオーバヘッドを検出する手法[17)] が提案されている．しかし，これらの手法は，複数のホストの調査結果や対象ホストでのベンチマークプログラムの実行が必要である．このため，これらの手法がセキュリティシステムの管理者にとって脅威となる可能性は比較的小さい．

本論文では，リモートスキャンによる仮想マシンモニタ検出の一手法を提案する．リモートスキャンによる仮想マシンモニタ検出では，既存のリモート仮想マシンモニタ検出手法と異なり，対象ホストとの通信が可能でさえあればよい．リモートスキャンによる仮想マシンモニタ検出手法が実際に使用されるようになると，攻撃者が仮想マシンモニタを用いたセキュリティシステムを容易に検出できるようになってしまう．このため，仮想マシンモニタを利用したセキュリティシステムの有用性を高めるためには，リモートスキャンによる仮想マシンモニタの検出とその対策について議論する必要がある．

リモートスキャンによる仮想マシンモニタ検出では，対象ホストをネットワーク通信のみを用いて調べることにより，仮想マシンモニタに起因するホストの振舞いの違いを検出する．提案手法では，複数のネットワークタイムスタンプを単一のパケットを用いて取得し，そのずれの有無で仮想マシンモニタの検出を行う．検査対象ホストが仮想マシンモニタを使用している場合，仮想マシンモニタによって仮想マシンの切替えが発生したり仮想マシンの時刻が修正されたりすることがあるため，タイムスタンプ間に大きなずれが生じることがある．一方，検査対象が仮想マシンではない場合，タイムスタンプ間のずれが生じることはない．したがって，タイムスタンプ間のずれを利用することで，仮想マシンモニタの有無を検出することができる．提案手法を Xen[1)]，VMware Workstation[2)]，VirtualBox[3)] 上で動作する Linux を用いた仮想マシンについて実験した結果，これらの仮想マシンモニタが提案手法により検出可能であることが分かった．

本論文では，まず 2 章で仮想マシンモニタについて説明する．次に 3 章でリモートスキャンによる仮想マシンモニタ検出について説明し，4 章でタイムスタンプを用いた仮想マシンモニタ検出手法について述べる．その後，5 章で Xen，VMware Workstation，VirtualBox 上の仮想マシンについての実験結果を示し，6 章で提案手法に対する防御方法について議論する．7 章で関連研究について述べ，8 章で本論文をまとめる．

2. 仮想マシンモニタ

2.1 仮想マシンモニタの定義

仮想マシンモニタとはハードウェア環境を忠実に仮想化するソフトウェアレイヤである．仮想マシンモニタによって提供される仮想的なハードウェア環境のインスタンスを仮想マシンと呼ぶ．仮想マシンモニタは複数の仮想マシンを並列に動作させることができるため，1 つのハードウェアで様々なオペレーティングシステムやそのアプリケーションを動作させることができる．本論文では，仮想マシンモニタを動かしている計算機環境全体をホストと呼ぶ．

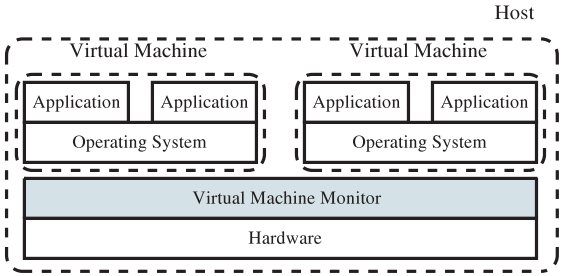


図 1　ハードウェア上で動作する仮想マシンモニタ
Fig. 1　Virtual machine monitor runs on bare hardware.

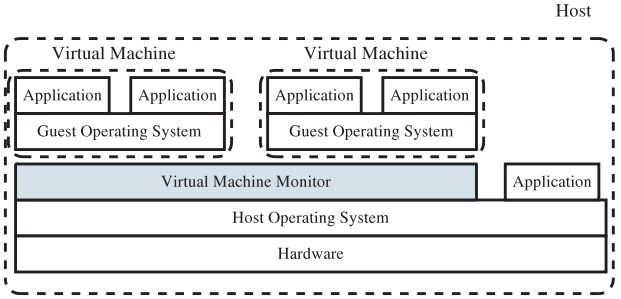


図 2　オペレーティングシステム上で動作する仮想マシンモニタ
Fig. 2　Virtual machine monitor runs on an operating system.

仮想マシンモニタには，仮想マシンモニタがハードウェア上で直接動作するタイプ（**図 1**）と，ホストのオペレーティングシステム（ホスト OS）上で動作するタイプ（**図 2**）がある．前者では，仮想マシンモニタが小さなオペレーティングシステムになっている．このタイプの仮想マシンモニタには，Xen [1]，VMware ESX Server [2] などがある．後者は，仮想マシンモニタはホスト OS を利用して計算機資源の管理や入出力を行う．このタイプの仮想マシンモニタには，VirtualBox [3] などがある．また，ホスト OS を改変して動作することで両方のタイプの特徴を持つ VMware Workstation [2] のよう仮想マシンモニタもある．

仮想マシンモニタは 1 つの CPU 上で複数の仮想マシンを切り替えながら動作させることができる．まず，CPU 上で動作し始めた仮想マシンに CPU 時間が割り当てられる．仮想マシンが与えられた CPU 時間を使い切ると，仮想マシンモニタが仮想マシンの動作を強制的に中断し，他の仮想マシンに CPU 時間を割り当てて動作させる．このようにして，複数の仮想マシンを 1 つの CPU 上で動かす．

また，仮想マシンモニタは入出力デバイスの仮想化を行う．たとえば，ネットワークデバイスの仮想化では，仮想マシンモニタはパケットの集配を行う．外部からパケットの入力があった場合，仮想マシンモニタはホストのネットワークデバイスからそのパケットを読み出す．その後，そのパケットの宛先アドレスから宛先の仮想マシンを判別し，パケットを配送する．一方，仮想マシンがネットワークデバイスに出力したパケットは，仮想マシンモニタがいったん受け取った後でホストのネットワークデバイスに出力する．

3. リモートスキャンによる仮想マシンモニタ検出

3.1 既存のリモート検出手法との比較

“リモートスキャン” とは，送信したメッセージに対する応答を解析することで，対象ホストの動作に関する情報を取得することである．本論文では，対象ホスト上で特定のプログラムが動作していることを仮定せずに，対象ホストとのネットワーク通信だけで，対象ホストが仮想マシンモニタを使用しているかどうかを判別することを目的とする．

既存のリモート仮想マシンモニタである Kohno ら[16] の手法では，複数の仮想マシンが同一のホスト上で動作しているときに，すべての仮想マシンの時計が基準時計に対して同じようにずれるということを利用して，仮想マシンモニタを検出する．この基準時計からのずれをネットワークタイムスタンプを用いて算出することで，リモート仮想マシンモニタ検出を行う．この手法は仮想マシン群が 1 つのハードウェアを共有していることを利用するため，複数の仮想マシンが同一ホストで動作しており，攻撃者にそれらの仮想マシンがアクセス可能な場合にしか検出できない．Franklin ら[17] の手法では，対象ホスト上で CR3 レジスタ[⋆1]を頻繁に操作する，仮想マシンであることが処理時間に強く影響するベンチマークプログラムを動作させ，その処理時間を攻撃者のマシンで観測することで仮想マシンモニタを検出する．ここで，攻撃者のマシンから観測を行うのは，対象ホストの時計が調整されている場合でも仮想マシンモニタを検出できるようにするためである．しかし，この手法では，CR3 のような特殊なレジスタを操作するために，対象ホストの root 権限でプログラムを動作する必要があるため，リモートスキャンとは異なる．

また，従来のリモートスキャン手法は，仮想マシンモニタを検出できる状況が非常に限定さ

⋆1 ページディレクトリを指定するためのコントロール・レジスタ．

れる．たとえば，Nmap [18] を用いたポートスキャンでは，902 番ポートで動作する VMware Server Console を検出することで，そのマシンが VMware ESX Server を用いていることが分かる．しかし，管理者は通常このようなサービスに対してインターネットからのアクセスを許可することはない．

3.2　仮想マシンと実マシンにおけるネットワーク動作の差異

仮想マシンは実マシンと比較して，ネットワーク通信に関する振舞いが異なる．これは，(1) 仮想マシンの時計の狂い，(2) 仮想マシンモニタによる仮想マシンへのパケット配送，(3) 仮想マシンモニタによるパケット操作，に由来する．以下では，リモートスキャンによる仮想マシンモニタ検出を実現するために，これらの 3 つの原因による仮想マシンの振舞いの変化について述べる．

まず，仮想マシンの時計の狂いによってネットワーク通信の振舞いは変化する．たとえば，VMware ではタイマデバイスのエミュレーションを行う [19] ため，時計のずれが実マシンに比べて大きくなる [16]．このずれは，ネットワークタイムスタンプや，接続のタイムアウトなどに表れる．

次に，複数の仮想マシン間でホストのネットワークインタフェースが共有されているため，パケット配送のオーバヘッドが生じる．たとえば，Xen では，仮想マシンモニタがパケットを受信すると，宛先である仮想マシンのキューにパケットを配送する．そのキューは次に仮想マシンがスケジュールされたときに処理される．対して，実マシンではパケットが到着すると同時にパケット処理がなされる．この動作の違いにより，仮想マシンと実マシンではパケットの RTT に違いが生じる場合がある．したがって，この RTT の違いを利用し仮想マシンの検出ができると考えられる．しかし，RTT は通信経路上のルータなどで遅延が発生した場合にも影響を受けるため，RTT を利用した手法で正確に仮想マシンモニタを検出することは難しい．

最後に，仮想マシンモニタはパケット操作を行うことができるため，パケットの扱いに違いが生じる場合がある．たとえば，仮想マシンモニタを用いたセキュリティシステムで，仮想マシンの脆弱性を利用する攻撃パケットを拒否するようなシステムが考えられる．この場合，攻撃者は意図的に攻撃パケットを送信することでこのシステムを検知することができる．しかし，このような手法では一般の仮想マシンモニタを検知することはできない．

3.3　ネットワークタイムスタンプ

本論文ではネットワークタイムスタンプを用いて，リモートホストの時刻を取得し，仮想マシンの時計の狂いに由来する振舞いの違いを検出する．ネットワークタイムスタンプは対象ホスト上で打刻されるため，通信経路上のルータなどの影響を受けずに，対象ホストの時刻を正確に取得できる．また，ネットワークタイムスタンプは基本的に OS で処理されるものであり，現在の仮想マシンモニタはネットワークタイムスタンプに対して処理を行うことはない．

ネットワークタイムスタンプは，様々なネットワークプロトコルに存在するため，複数の種類のネットワークタイムスタンプを単一のパケットで同時に取得することができる．たとえば，ICMP タイムスタンプと IP タイムスタンプが単一のパケットに打刻されるようにすることができる．このとき，取得したタイムスタンプは通常同じ時刻を示す．これはタイムスタンプの精度に比べて非常に短い間隔で打刻されるためである．実際，ICMP，IP 両タイムスタンプの精度は 1 ミリ秒である．対して，Intel Core2 Duo 1.86 GHz，Linux 2.6.18 のマシン上で，ICMP タイムスタンプを打刻してから IP タイムスタンプを打刻するまでの処理時間は，平均 2 マイクロ秒であった．

まれに同一パケット上の複数のタイムスタンプが異なる時刻を示すことがある．これは，2 つのタイムスタンプを打刻している間に時刻の切替わりが発生するためである．たとえば，ICMP タイムスタンプと IP タイムスタンプを打刻する間に時刻の切替わりが発生した場合，これらの 2 つの値は 1 ミリ秒だけ異なる．

仮想マシンでは，2 つのタイムスタンプの値が異なる確率が実マシンに比べて高くなる．たとえば，仮想マシンが 1 つ目のタイムスタンプを打刻した直後に，仮想マシンモニタが仮想マシンを切り替えることによって，仮想マシンの処理を中断し，他の仮想マシンを動作させることがある．この場合，2 つ目のタイムスタンプが打刻されるのは処理が再開された後である．このようにタイムスタンプの打刻間隔が長くなることで，値が異なる確率が高くなる．提案手法では，仮想マシンモニタを検出するために，このタイムスタンプのずれを利用する．

通常，インターネット上で使用可能なタイムスタンプは以下のとおりである．

- **ICMP タイムスタンプ**　ICMP タイムスタンプは RFC 792 [20] において定義されている．受信タイムスタンプと送信タイムスタンプが定義されている．受信タイムスタンプはホストがタイムスタンプ要求を受信したときに設定され，送信タイムスタンプはホストがタイムスタンプ応答を送信するときに設定される．Linux では，この 2 つのタイムスタンプはつねに同じ値が設定される．タイムスタンプの値は，世界標準時の深夜 0 時を基準としたミリ秒単位の値である．
- **IP タイムスタンプ**　IP タイムスタンプは RFC 791 [21] において定義されている．IP

タイムスタンプはホストだけでなく，通信経路上のルータなどのタイムスタンプを得ることも可能である．タイムスタンプの値は，世界標準時の深夜 0 時を基準としたミリ秒単位の値である．

- **TCP タイムスタンプ**　TCP タイムスタンプは RFC 1323 [22] において定義されている．主な使用目的は RTT を計測することである．RFC 1323 ではタイムスタンプのクロックとして，システム起動時に 1 度だけ初期化される仮想的なクロックを用いることが推奨されている．Linux ではタイマ割込みの回数を数えるカウンタである jiffies を利用して実装されている．このため，Linux における TCP タイムスタンプは，カーネルの設定により 1 ミリ秒から 10 ミリ秒の精度を持つ．
- **アプリケーションタイムスタンプ**　アプリケーションタイムスタンプとは，アプリケーション層のサーバが用いるタイムスタンプである．たとえば，NTP タイムスタンプ[23] などがある．タイムスタンプの精度はアプリケーションに依存する．

この 4 種類のタイムスタンプでは，(1) ICMP タイムスタンプ + IP タイムスタンプ，(2) TCP タイムスタンプ + IP タイムスタンプ，(3) アプリケーションタイムスタンプ + TCP タイムスタンプ，(4) アプリケーションタイムスタンプ + IP タイムスタンプ，(5) アプリケーションタイムスタンプ + TCP タイムスタンプ + IP タイムスタンプが単一のパケットで取得可能である．以下では，単一のパケットで取得したタイムスタンプのペアを/で区切って示す．たとえば，ICMP タイムスタンプと IP タイムスタンプのペアを ICMP/IP タイムスタンプと呼ぶ．

表 1 に仮想マシンモニタを検出するためのタイムスタンプ・ペアに関する比較を示す．タイムスタンプ・ペアは組み合わせるタイムスタンプの精度が同等でないことがあり，その場合は最も低いタイムスタンプの精度でしかタイムスタンプのずれを検出できないため，検出が難しくなる．たとえば，TCP タイムスタンプが 10 ミリ秒の精度であるとき，TCP/IP タイムスタンプのずれは 10 ミリ秒単位でしか検出できない．また，あまり使われないタイムスタンプを用いる場合には防御が容易になる．たとえば，IP タイムスタンプは実際にはあまり使われないタイムスタンプであるため，攻撃者が IP タイムスタンプとのペアを用いる場合，管理者は IP タイムスタンプを検知することにより異常を検知できる．また，アプリケーションタイムスタンプとのペアを用いる場合，提案手法はアプリケーションに対する依存性を持つ．たとえば，NTP タイムスタンプを用いたペアは NTP サーバに対してしか使用できない．アプリケーション/TCP/IP タイムスタンプは 3 つのタイムスタンプを用いるため，アプリケーション/TCP タイムスタンプ，アプリケーション/IP タイムスタンプ，TCP/IP タイムスタンプの 3 種類の比較を同時に用いることができる．このため，タイムスタンプのずれが観測できる可能性は高くなると考えられる．しかし，上記の TCP/IP タイムスタンプの比較に関する問題点や，アプリケーションへの依存性が生じる．

表 1　タイムスタンプの組み合わせ方に関する比較
Table 1　Comparison of timestamp pairs.

	仮想マシンモニタの検出容易性	実装容易性	防御容易性	対象ホストの存在割合
ICMP/IP	易	易	易	3 割以上
TCP/IP	難	難	易	3 割以上
アプリケーション/TCP	*	*	難	*
アプリケーション/IP	*	*	易	*
アプリケーション/TCP/IP	*	難	易	*

(*：アプリケーションに依存する)

本論文では，ICMP タイムスタンプと IP タイムスタンプを単一のパケットで取得することで，仮想マシンモニタ検出を行う．ICMP/IP タイムスタンプは，タイムスタンプの値の定義が同じであるため比較が容易であり，精度も 1 ミリ秒と細かい．提案手法では，対象ホストが ICMP/IP タイムスタンプ要求を受け付けられるように設定されている必要があり，そのように設定されていないホストに対しては仮想マシンモニタ検出が行えない．しかし，現在インターネット上のホストの 1/3 はこの要求を処理可能である[24),25)]．このため，攻撃者にとっては，リモートスキャンの対象となるホストの数は十分多いと考えられる．

4. タイムスタンプによる仮想マシンモニタ検出

4.1 タイムスタンプのずれによる検出

単一のパケット上に 2 つのタイムスタンプ T_A，T_B を打刻する場合，打刻の間に時刻が切り替わることでタイムスタンプがずれることがある．**図 3** にそのようなタイムスタンプ

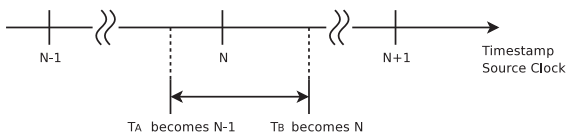


図 3　タイムスタンプのずれが生じるときのタイムスタンプの打刻と時刻の進みの関係
Fig. 3　Relationship between source clock and two timestamp operations, when a timestamp discrepancy occurs.

のずれが発生するときのタイムスタンプの打刻と時刻の進みの関係を示す．図 3 では，タイムスタンプ T_A が打刻された直後に，時計の時刻が進み，タイムスタンプ T_B に T_A より大きな値が打刻される．実マシンではこのタイムスタンプのずれの量は大きくても 1 単位である．これは，2 つのタイムスタンプ間の処理時間がタイムスタンプの単位時間に比べきわめて小さいためである．

仮想マシンの場合，T_A を打刻してから T_B を打刻するまでの時間は，(1) 仮想マシンの切替え，(2) 仮想マシンの時刻管理，の 2 つが原因で変化する．T_A が打刻された直後に仮想マシンモニタによって仮想マシンの切替えが発生すると，タイムスタンプ処理は他の仮想マシンが動作している間中断する．処理が再開したときにタイムスタンプの単位時間以上経過していると，T_A より大きい値が T_B に打刻される．

また，仮想マシンの時刻管理が原因で，2 つのタイムスタンプに大きなずれが生じることがある．仮想マシンの時刻は，タイマデバイスのエミュレーションや，仮想マシン切替えによって，実際の時刻からのずれが大きくなっていくことが知られている[16),17)]．そのため，仮想マシンモニタは仮想マシンの扱う時刻を定期的に修正する[19)]．この修正が T_A が打刻された直後に行われると，T_B に T_A とは大きく異なる値が打刻される．多くの仮想マシンモニタ上の仮想マシンでは実際の時刻に比べて時刻が遅れる方向にずれるため，T_B が T_A より大きくなる．しかし，一部の仮想マシンモニタ上の仮想マシンでは，タイマデバイスのエミュレーションによって，時刻が実際の時刻より進むことがあるため，修正時に T_B が T_A より小さくなることがある．

提案手法では，これらのタイムスタンプのずれを検出することで，仮想マシンモニタを検出する．たとえば，T_B が T_A より 2 単位以上大きいときに，対象ホストが仮想マシンであると見なす．また，T_B が T_A より小さいときにも，対象ホストが仮想マシンであると見なす．このような異常なずれは，仮想マシンモニタ上で動作する仮想マシンの台数が多いとより顕著に観測される．これは仮想マシンの台数が多いと仮想マシンの切替えがより頻繁になるためである．また，仮想マシンの切替えによる処理の中断が増加することで，仮想マシンの時刻のずれも大きくなる．提案手法は，仮想マシンモニタが仮想マシンの実行をその実行状態にかかわらず中断する仮想マシンモニタを検出できる．どのタイプの仮想マシンモニタであっても，基本的には仮想マシンの実行状態にかかわらず実行を中断する．したがって，提案手法はどのタイプの仮想マシンモニタでも検出できる．

4.2　ICMP/IP タイムスタンプのずれ

仮想マシン切替えと ICMP/IP タイムスタンプのずれの関連を調べるため，Xen 準仮想化ドメインにおいて，CPU のタイムスタンプカウンタ（TSC）を 4 つの計測点で計測した．ここで用いた計測点は，タイムスタンプを取得したとき（TSC-ICMPTS，TSC-IPTS），仮想マシンの切替えが発生したとき（DESCHED-TSC），仮想マシンが再び実行を開始したとき（RESCHED-TSC）である．また，各タイムスタンプの値（ICMPTS，IPTS）も記録した．これらの関係を**図 4** に示す．計測結果として得られたログの一部を**図 5** に示す．ログの下線のない部分では，TSC-ICMPTS と TSC-IPTS の間において仮想マシンの切替えが発生せず，ICMPTS と IPTS は同じ値を示した．一方，ログの下線部においては，TSC-ICMPTS と TSC-IPTS の間において仮想マシンの切替えが発生し，DESCHED-TSC と RESCHED-TSC の値がその間に入っている．また，ICMPTS の値と IPTS の値が 2 だけずれている．したがって，仮想マシン切替えによってタイムスタンプのずれが発生していることが分かる．

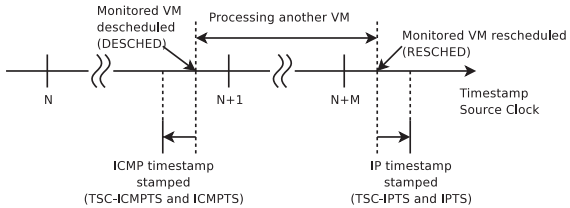


図 4　タイムスタンプカウンタを計測した点
Fig. 4　Monitoring points.

TSC-ICMPTS	TSC-IPTS	ICMPTS	IPTS	DESCHED-TSC	RESCHED-TSC
1489095607643	1489095611304	24722328	24722328	-	-
1489097468341	1489101221517	24722329	24722331	1489097474046	1489101213474
1489107997041	1489108008661	24722335	24722335	-	-

図 5　仮想マシンの切替えによって生じた ICMP/IP タイムスタンプのずれ
Fig. 5　Timestamp discrepancy between ICMP/IP timestamps caused by switching virtual machines.

また，仮想マシンの時刻管理とタイムスタンプのずれの関連を調べるため，Xen 完全仮想化ドメインにおいて，タイムスタンプを打刻する時点において計測を行った．Xen 完全仮想化ドメインは仮想マシンごとに TSC を保持しているため，計測では，各タイムスタンプを打刻する時点での仮想マシンの TSC（vTSC-ICMPTS，vTSC-IPTS），実際の TSC からのオフセット（OFF-ICMPTS，OFF-IPTS）を記録し，仮想マシンモニタの持つ TSC

vTSC-ICMPTS	vTSC-IPTS	OFF-ICMPTS	OFF-IPTS	rTSC-ICMPTS	rTSC-IPTS	ICMPTS	IPTS
698140796796	698140800919	276767605917	276767605917	974908402713	974908406836	26370399	26370399
698141354052	698146772212	276768962209	276763571293	974910316261	974910343505	26370399	26370402
698147288707	698147293208	276764808242	276764808242	974912096949	974912101450	26370402	26370402

図 6　仮想マシンの時刻修正によって生じた ICMP/IP タイムスタンプのずれ
Fig. 6　Timestamp discrepancy between ICMP/IP timestamps caused by time correction of a virtual machine.

（rTSC-ICMPTS，rTSC-IPTS）を算出した．計測結果として得られたログを**図 6** に示す．ログの下線のない部分では，OFF-ICMPTS と OFF-IPTS は一致しており，ICMPTS と IPTS は同じ値を示した．これは ICMP タイムスタンプの打刻と IP タイムスタンプの打刻の間に実際の TSC からのオフセットが修正されていないことを示している．一方，ログの下線部では，OFF-ICMPTS と OFF-IPTS は異なる値になっている．これは ICMP タイムスタンプの打刻と IP タイムスタンプの打刻の間に仮想マシンモニタによる時刻の修正が行われたということを示す．この場合，ICMPTS と IPTS は 3 だけ異なる値を示しており，タイムスタンプのずれが発生したことが分かる．

この時刻修正の原因は，Xen 完全仮想化ドメインが 2 つの方式を用いて時刻管理を行っているためである．1 つ目に，仮想マシンの実行を中断し，他の仮想マシンに CPU 使用権を与えるときに，仮想マシンモニタは実行を中断する仮想マシンの TSC を保存し，その仮想マシンに再び CPU 使用権を与えるときに，保存されている TSC を書き戻す．2 つ目に，仮想マシンの TSC がハードウェアのタイマ割込みとずれないように，ハードウェア割込みがあったときに，仮想マシンモニタの TSC に基づいて仮想マシンの TSC を修正する．**図 7** に，仮想マシンの TSC がどのように修正されているかを計測した結果を示す．図 7 の点線は仮想マシンの TSC の進み方が仮想マシンモニタの TSC の進み方と一致する場合を示しており，十字の点は計測された仮想マシンの TSC の進み方と仮想マシンモニタの TSC の進み方の関係を示している．仮想マシンの TSC は 1 つ目の時刻管理方式によって，仮想マシンがスケジューリングされるたびに書き戻されているため，実際の時間より遅く進んでいる．これは十字の点を連続的につないだとき，傾きが点線の傾きより小さくなっている部分が多いことで分かる．その後，2 つ目の時刻管理方式によって仮想マシンの TSC が実マシンの TSC と一致するように大きな修正が行われている．これは，十字の点の傾きが極端に大きくなる点があることで分かる．この 2 つ目の修正の前後に ICMP/IP タイムスタンプの打刻処理が入ると，タイムスタンプのずれが発生する可能性が高い．

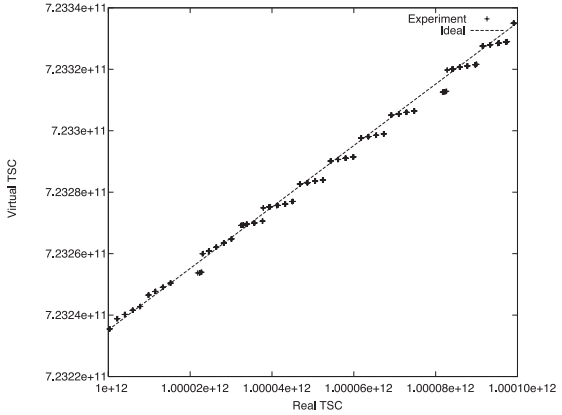


図 7　仮想マシン上のタイムスタンプカウンタの値（Virtual TSC）と実マシン上のタイムスタンプカウンタの値（Real TSC）の関係
Fig. 7　The relationship between virtual and real TSCs.

4.3　ICMP/IP タイムスタンプによる検出機構

提案手法による仮想マシンモニタ検出機構は ICMP/IP タイムスタンプを比較して仮想マシンモニタ検出を行う．以下では各タイムスタンプの値をそれぞれ ICMPTS，IPTS と呼ぶ．なお，以下の議論は Linux の実装に基づき，ICMPTS が先に設定されると仮定する．検出機構による比較結果は以下の 4 ケースを想定する．

- **ICMPTS = IPTS**　実マシンでも仮想マシンでも発生する．
- **ICMPTS + 1 = IPTS**　実マシンでも仮想マシンでもまれに発生し，仮想マシンの場合はその確率が高い．
- **ICMPTS + 1 < IPTS**　仮想マシンの場合にしか発生しない．検出機構はこの現象を検知すると，仮想マシン切替え，または仮想マシンの時刻の進み方が遅いことによって発生するずれを仮想マシンモニタが修正したと見なす．
- **ICMPTS > IPTS**　仮想マシンの場合にしか発生しない．検出機構はこの現象を検知すると，仮想マシンの時刻の進み方が速いことによって発生するずれを仮想マシンモニタが修正したと見なす．

提案手法による仮想マシンモニタ検出では，非常に低い確率の事象を検知する必要があるため，必要とするパケット数が多くなる．必要とするパケット数 n は，タイムスタンプのずれに異常が発生する確率 p と仮想マシンの検知精度 c に依存し，$c = 1-(1-p)^n$ となる．たとえば，

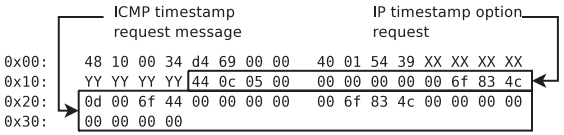


図 8　ICMP/IP タイムスタンプ要求メッセージの例
Fig. 8　Example of ICMP/IP timestamp request message.

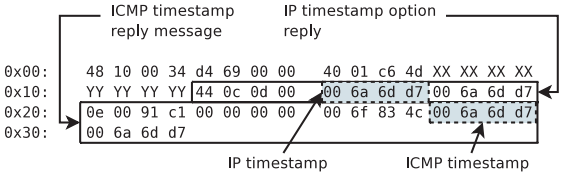


図 9　ICMP/IP タイムスタンプ応答メッセージの例
Fig. 9　Example of ICMP/IP timestamp reply message.

もし発生確率 p が 1/1,000,000 で検知精度 c が 0.9 である場合，$1-(1-1/1{,}000{,}000)^n > 0.9$ を解いて，必要パケット数は約 230 万パケットである．発生確率 p は仮想マシンモニタの実装やハードウェアの種類などに依存する．提案手法は必要とするパケット数が多いという問題があるものの，攻撃者は多数の検出対象マシンに対して，同時に多数のマシンを用いた分散スキャンや，数カ月にわたる低速スキャンを行うことで，十分な効率で仮想マシンを発見することができると考えられる．また，5 章で示すように，仮想マシンモニタ上の仮想マシンの台数が多い場合は発生確率 p が上昇し，必要なパケット数が少なくなる．

4.4 実　　装

提案手法による仮想マシンモニタ検出機構を x86 アーキテクチャ上の Linux に実装した．Linux では ICMP/IP タイムスタンプ要求パケットを送信するには，タイムスタンプ要求パケットを作成し，RAW ソケットを使って送信する必要がある．**図 8** に ICMP/IP タイムスタンプ要求パケットの例を示す．ただし IP アドレスは匿名化している．このパケットは，IP タイムスタンプオプションを持たせた ICMP タイムスタンプ要求パケットである．**図 9** にこの要求パケットに対する返答パケットの例を示す．IP タイムスタンプと ICMP タイムスタンプは，それぞれこのパケットのアドレス 0x18 と 0x2c から始まる 4 バイトである．また，ICMP/IP タイムスタンプ応答パケットを受信し処理するために，パケットキャプチャを行うライブラリである libpcap [26] を用いた．なお，本機構は，取得するタイムスタンプがネットワークレイテンシに影響されないため，ユーザレベルで実装できる．

5. 実　　験

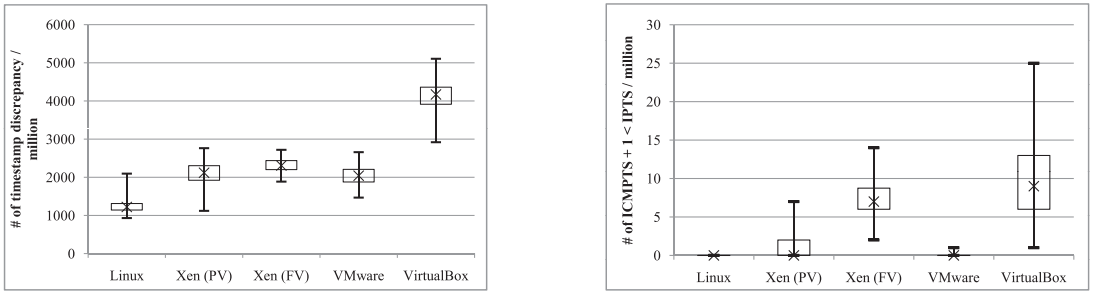
5.1 様々な仮想マシンモニタに対する実験

実装した仮想マシンモニタ検出機構を用いて，広く使われている仮想マシンモニタに対して実験を行った．提案機構は AMD Opteron 2.2 GHz，RAM 1 GB のマシンで動作しており，実験対象ホストは Intel Core2 Duo E6300 1.86 GHz，RAM 2 GB のマシン上で動作している．実験対象としたホストの環境は，(1) Xen 3.1.0a 準仮想化ドメイン（Xen (PV)），(2) Xen 3.1.0a 完全仮想化ドメイン（Xen (FV)），(3) VMware Workstation 6（VMware），(4) VirtualBox 1.5.2（VBox），(5) ハードウェア上の Linux（Linux）である．なお，これらのうち，Xen はハードウェアの上でオペレーティングシステムとして動作する仮想マシンモニタであり，VirtualBox はホスト OS の上で動作する仮想マシンモニタである．また，VMware Workstation はホスト OS を改変し動作する仮想マシンモニタである．すべての場合において，Linux カーネル 2.6.18 を用いた．これらの環境は同じハードウェアを使用したマルチブート環境として用意した．また，提案機構と対象ホストは 1 Gbps のスイッチを経由して接続した．

実験では，50,000,000 個のタイムスタンプ応答を収集し，1,000,000 応答あたりの ICMPTS < IPTS となる応答数と，ICMPTS + 1 < IPTS となる応答数を計測した．対象ホストのキューが詰まってしまうことを避けるため，タイムスタンプ要求は 1 ミリ秒ごとに送信した．実験時間は 1 ケースにつき，14 時間であった．

図 10 に実験結果を示す．図 10 (a) に ICMPTS < IPTS となる応答を示し，図 10 (b) に ICMPTS + 1 < IPTS となる応答の数を示す．図 10 (b) では，ICMPTS + 1 < IPTS となる応答は実マシンの場合には発生しなかった．一方，仮想マシンの場合には ICMPTS + 1 < IPTS となる応答が 1,000,000 パケットあたり 0.06 回から 10.14 回であった．したがって，提案機構が ICMPTS + 1 < IPTS を検出した場合，対象ホストが確実に仮想マシンであると判断できる．

また，図 10 (a) では，ICMPTS < IPTS となる応答の数が Linux の場合は平均 1,270 回なのに対し，仮想マシンの場合には平均 2,045 回から 4,118 回と増加した．これは，仮想マシンモニタによる仮想マシン切替えや時刻修正のため，ICMP/IP タイムスタンプのずれが生じる確率が増加しているためである．図 10 (a) は ICMPTS + 1 < IPTS となるようなずれが発見されなかった場合でも，ICMPTS < IPTS となる確率を見ることで仮想マシンモニタの検出ができる可能性があることを示唆している．たとえば，対象ホストと同じハード



(a) ICMP/IP タイムスタンプが ICMPTS < IPTS となった回数　(b) ICMP/IP タイムスタンプが ICMPTS+1 < IPTS となった回数

図 10　仮想マシンと実マシンについての実験結果
Fig. 10　Experimental results on virtual machines and a real machine.

表 2　仮想マシンを 90%の確度で判断するのに必要なパケット数
Table 2　Number of packets required to detect a virtual machine with 90% confidence.

仮想マシンモニタ	所要パケット数
Xen (PV)	2,200,000
Xen (FV)	320,000
VMware	39,000,000
VirtualBox	230,000

ウェアを攻撃者が用意できるとき，あらかじめ様々な仮想マシンモニタについて ICMPTS < IPTS となる確率を調べておいて，対象ホストの確率と比較することで仮想マシンかどうかを判断できる．

図 10 (b) に示した実験結果より計算した，提案手法によって各仮想マシンモニタ上の仮想マシンを 90%の確度で判断するのに必要なパケット数を**表 2** に示す．提案手法は VMware の検出に約 3,900 万パケットが必要なことが分かった．一方，VirtualBox では約 23 万パケットで検出可能である．これは，VMware が行っている仮想マシンの時刻管理が，他の仮想マシンモニタに比べてより正確であるためであると考えられる．提案手法は Xen 完全仮想化ドメインや，VirtualBox のような時刻管理があまり正確でない仮想マシンモニタの検出を比較的容易に行うことができる．

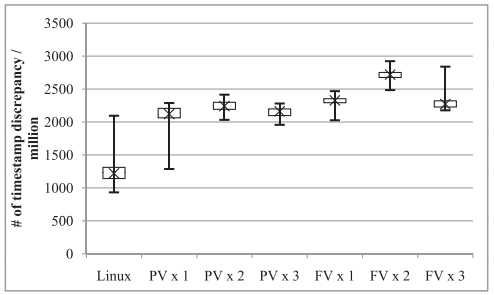
5.2　仮想マシンモニタ上の仮想マシンの台数との関係

仮想マシンモニタ上の仮想マシンの台数が ICMP/IP タイムスタンプのずれ方に与える影響を調査するため，実験を行った．5.1 節で示した実験を，Xen に対して，仮想マシンの台数を変えながら行った．**図 11** に実験結果を示す．なお，参考のために実マシンの Linux 上での実験結果と同時に示す．図 11 (a) は，ICMPTS < IPTS となる回数を示し，図 11 (b) では，ICMPTS + 1 < IPTS となる回数を示す．“PV x n” と “FV x n” は，それぞれ準仮想化ドメインが n 台動作している状況と，完全仮想化ドメインが n 台動作している状況を示す．

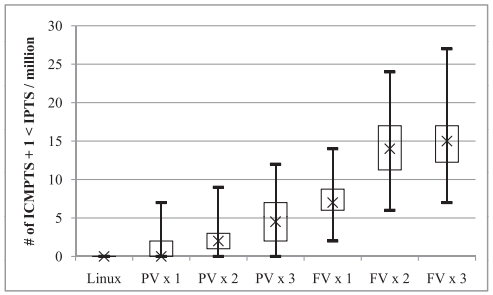
図 11 (a) に示した実験結果では，ICMPTS < IPTS となる回数は，すべての場合において実マシンと比べて上昇しているものの，仮想マシンモニタ上の仮想マシンの数には影響されないことが分かる．準仮想化ドメインでは，1,000,000 応答あたりの回数がそれぞれ，2,075 回（n = 1），2,243 回（n = 2），2,145 回（n = 3）であった．完全仮想化ドメインでは，2,314 回（n = 1），2,705 回（n = 2），2,287 回（n = 3）であった．

一方，図 11 (b) に示した実験結果をを見ると，仮想マシンの台数が増加するにつれて，ICMPTS + 1 < IPTS となる回数が上昇している．準仮想化ドメインでは，1,000,000 応答あたりの回数がそれぞれ，1.08 回（n = 1），2.31 回（n = 2），4.52 回（n = 3）であった．完全仮想化ドメインでは，7.37 回（n = 1），14.30 回（n = 2），15.10 回（n = 3）であった．

図 11 (b) に示した実験結果より計算した，提案手法によって仮想マシンを 90%の確度で判断するのに必要なパケット数を**表 3** に示す．提案手法は仮想マシンの台数が増えるほど，仮想マシンモニタを検出しやすくなっていることが分かる．



(a) ICMP/IP タイムスタンプが ICMPTS < IPTS となった回数



(b) ICMP/IP タイムスタンプが ICMPTS+1 < IPTS となった回数

図 11　仮想マシンの台数を変化させたときの実験結果
Fig. 11　Experimental results when the number of virtual machines are changed.

表 3　仮想マシンの台数と 90%の確度で判断するのに必要なパケット数の関係
Table 3　Relationships between number of virtual machines and number of packets required to detect a virtual machine with 90% confidence.

ドメインのタイプ	仮想マシンモニタ上の仮想マシンの数	所要パケット数
準仮想化	1	2,200,000
準仮想化	2	1,000,000
準仮想化	3	510,000
完全仮想化	1	320,000
完全仮想化	2	160,000
完全仮想化	3	150,000

5.2.1 考　　察

図 11 (a) では，仮想マシンモニタが制御する仮想マシンの台数と ICMPTS < IPTS となる回数の間に関連性は確認できなかった．これは，Xen のスケジューラが各仮想マシンに CPU 使用権を必ず一定時間割り当てるようになっていることが原因であると考えられる．各仮想マシンに CPU 使用権が十分割り当てられている場合，ICMP タイムスタンプと IP タイムスタンプの打刻を行う間に，仮想マシンモニタが仮想マシン切替えを行う確率は大きく変わらない．

図 11 (b) では，仮想マシンモニタが制御する仮想マシンの台数が増加するにつれ，ICMPTS + 1 < IPTS となる確率が増加した．これは，仮想マシン切替えによる処理の中断が仮想マシンの台数の増加にともない長くなることが原因であると考えられる．この結果は仮想ハニーポットのような多数の仮想マシンを動かす仮想マシンモニタの検出が容易であることを示す．

6. 提案手法に対する防御手法

提案手法は，ファイアウォールによって ICMP/IP タイムスタンプの取得要求パケットを通過させないようにすることで，容易に防御が可能である．しかし，攻撃者は提案手法を他のタイムスタンプの組合せを使うようにすることが考えられる．本論文では提案手法を ICMP/IP タイムスタンプについてしか実装していない．しかし，TCP タイムスタンプとアプリケーションタイムスタンプを組み合わせると，タイムスタンプ要求パケットがファイアウォールを通過できる可能性が高くなる．

また，仮想マシンモニタや OS を書き換え，同時に取得される 2 つのタイムスタンプのずれを検出した場合にはタイムスタンプの値を改変するようにすることで，防御が可能である．たとえば，ICMP/IP タイムスタンプの値を必ず揃えるように修正することにより，提案手法による仮想マシンモニタ検出への対策が可能である．しかし，このような対策をとると，タイムスタンプの打刻する時刻の正確性が失われる．したがって，NTP のような，アプリケーションがタイムスタンプの時刻に強く依存するアプリケーションの場合は，動作に影響が出る可能性がある．また，ICMP/IP タイムスタンプの値を 2 以上ずれていた場合に限り修正を行うことも考えられる．しかし，5 章で示した実験結果（図 10 (a)）は，ICMP/IP タイムスタンプの値のずれが 1 しかない場合でも，ずれの発生確率を見ることによって仮

想マシンモニタ検出が行える可能性を示唆している．

このような制限は提案手法に特有のものではなく，多くのネットワークスキャナも同様の制限を持つ．たとえば，ポートスキャナはファイアウォールを適切に設定することによって攻撃者には使えなくなってしまう．しかし，攻撃者はこのような対策のなされていない，スキャナの利用できるホストから攻撃対象を探す．したがって，提案手法も攻撃者が攻撃対象の候補とするマシンの中から仮想マシンを利用したセキュリティシステムを用いていないホストを探すために使われる可能性がある．

7. 関 連 研 究

7.1 リモート仮想マシンモニタ検出

リモート仮想マシンモニタ検出は，リモートホストから対象ホストにアクセスを行うことで，仮想マシンモニタを検出する手法である．Kohno らの手法[16] では，同一ホストの仮想マシン群が同一のハードウェアを用いることを利用し，仮想マシン群を検出することで，仮想マシンモニタを検出する．この手法では，ハードウェアが同一であることを判断する基準として，対象ホストの時計の基準時計に対する一定時間でのずれの大きさを用いており，そのずれを TCP タイムスタンプや ICMP タイムスタンプを利用して算出している．この手法を用いて対象ホストが仮想マシンであるかどうかを判断するには，対象ホストの存在するネットワーク上の複数のマシンを調査し，同じハードウェアを持つホストを探す必要がある．提案手法では単一のホストを調査するだけで，そのホストが仮想マシンであるかどうかを判断する．

Franklin らの手法[17] は，仮想マシン上で非常に低速になるプログラムを動作させ，プログラムの動作時間を測定者のマシンの時計で測定することで，対象ホストが仮想マシンモニタを使用しているかどうかを判断する．論文中では，そのようなプログラムの例として，CR3 レジスタへのアクセスを利用することを提示している．しかし，CR3 レジスタにアクセスするためには対象ホストの root 権限でプログラムを動作させる必要がある．したがって，この手法は攻撃者が攻撃の予備調査段階で用いることはできない．一方，提案手法では対象ホスト上でプログラムを動作させる権限は必要なく，ネットワーク通信だけで仮想マシンモニタを検出できるため，攻撃の予備調査段階において使用できる．

ネットワークタイムスタンプを用いたリモート仮想マシンモニタ検出手法として，Liston ら[27] による手法が存在する．この手法は本論文におけるリモートスキャンの定義を満たすと考えられる．しかし，この手法の技術的な詳細は明らかにされていない．また，この手法は TCP タイムスタンプと ICMP タイムスタンプという単一のパケットで同時に取得できないタイムスタンプを使用するため，我々の提案手法とは異なる可能性が高い．

また，最近，Chen らによるリモート仮想マシンモニタ検出手法が提案された[28]．この手法では，ホストが仮想マシンのときに，得られた TCP タイムスタンプの実時間からのずれ方が実マシンと異なることを利用し，リモートスキャンによる仮想マシンモニタ検出を実現している．しかし，この手法では Xen を用いた仮想マシンから得たデータと実マシンから得たデータの差が非常に小さくなってしまっており，Xen の検出は難しい．提案手法では，Xen について，データを取得する仮想マシンが準仮想化ドメインと完全仮想化ドメインの両方の場合について，仮想マシンモニタの存在を正しく検出できている．

7.2 ローカル仮想マシンモニタ検出

ローカル仮想マシンモニタ検出[9)–15)] では，攻撃者は攻撃対象ホストに侵入した後で仮想マシンモニタの検出を行う．これらの手法は，主に対象ホストの CPU の特徴を調べる．たとえば，仮想マシンモニタと通信するための特殊な命令が存在することや，割込みに関する処理が変更されていることを検知する．提案手法は，リモートからプログラムを実行せずに仮想マシンモニタを検出するため，これらの手法とは異なり，ホストに侵入せずに仮想マシンモニタを検出できる．

Ferrie[14]，および Raffetseder ら[15] は，仮想マシンだけでなく，QEMU[29] などのエミュレータの検出手法も提案している．本論文ではエミュレータに対する実験は行っていないものの，提案手法は以下の 2 つの理由により，リモートスキャンによってエミュレータを検出できる可能性が高い．第 1 に，多くのエミュレータは CPU のクロックを正確にエミュレートするため，時間の進み方が遅くなる．このため，エミュレータの時刻を実際の時刻に合わせるためには頻繁な時刻の修正が必要であり，提案手法で検出できる．時刻の修正を行わないエミュレータの場合は，タイムスタンプの値が実際の時刻と一致しなくなるため，容易に検出可能である．第 2 に，エミュレータはホスト上のプロセスとして動作するため，他のプロセスとの頻繁なコンテキストスイッチが行われる．したがって，仮想マシンの切替えの場合と同様に，2 つのタイムスタンプのずれが発生するため，提案手法で検出できる．

8. ま と め

本論文では，リモートスキャンによるリモート仮想マシンモニタ検出が可能なことを示すため，一例として，複数のタイムスタンプを同時取得し，タイムスタンプのずれ方の異常を検出することによる仮想マシンモニタの検出手法を提案した．また，実験により，Xen[1)]，

VMware Workstation [2]，VirtualBox [3] が検出可能なことと，仮想マシンモニタ上で動作する仮想マシンの台数が多いほど仮想マシンモニタの検出が容易なことを示した．リモートスキャンによるリモート仮想マシンモニタ検出手法は，攻撃者が仮想マシンモニタを利用したセキュリティシステムを回避するために使われる可能性がある．したがって，システムの設計者はリモートスキャンによる仮想マシンモニタ検出に対して，防御策を講ずる必要があるだろう．

今後の課題として，仮想マシンモニタの種類の特定を行う手法や，他の仮想マシンモニタ検出手法の検討を行い，仮想ハニーポットのようなセキュリティシステムに適した対処方法を確立する必要がある．

参 考 文 献

1) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the art of Virtualization, *Proc. 19th ACM Symposium on Operating Systems Principles* (*SOSP '03*), pp.164–177 (2003).
2) Sugerman, J., Venkitachalam, G. and Lim, B.-H.: Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor, *Proc. 2001 Annual Usenix Technical Conference*, pp.25–30 (2001).
3) Sun Microsystems: VirtualBox. http://www.virtualbox.org
4) The Honeypot Project: Know Your Enemy: Defining Virtual Honeynets (2003). http://www.honeynet.org/papers/virtual
5) Jiang, X. and Xu, D.: Collapsar: A VM-Based Architecture for Network Attack Detention Center, *Proc. 13th Usenix Security Symposium* (2004).
6) Vrable, M., Ma, J., Chen, J., Moore, D., Vandekieft, E., Snoeren, A.C., Voelker, G.M. and Savege, S.: Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfirm, *Proc. 20th ACM Symposium on Operating Systems Principles* (*SOSP '05*), pp.148–162 (2005).
7) Asrigo, K., Litty, L. and Lie, D.: Using VMM-based Sensors to Monitor Honeypots, *Proc. 2nd USENIX International Conference on Virtual Execution Environments* (*VEE '06*), pp.13–23 (2006).
8) Jiang, X. and Wang, X.: "Out-of-the-box" Monitoring of VM-based High-Interaction Honeypots, *Proc. 10th International Symposium on Recent Advances in Intrusion Detection* (*RAID '07*) (2007).
9) F-Secure: F-Secure Computer Virus Information Pages: Agobot. http://www.f-secure.com/v-descs/agobot.shtml
10) Rutkowska, J.: Redpill: Detect VMM using (almost) One CPU Instruction (2004). http://invisiblethings.org/papers/redpill.html
11) Kato, K.: VMware's Back. http://chitchat.at.infoseek.co.jp/vmware/
12) Klein, T.: Jerry – A(nother) VMware Fingerprinter (2003). http://www.trapkit.de/research/vmm/jerry/index.html
13) Klein, T.: Scooby Doo – VMware Fingerprint Suite (2003). http://www.trapkit.de/research/vmm/scoopydoo/index.html
14) Ferrie, P.: Attacks on Virtual Machine Emulators, *Proc. 9th Annual Association of anti Virus Asia Researchers International Conference* (*AVAR '06*) (2006).
15) Raffetseder, T., Kruegel, C. and Kirda, E.: Detecting System Emulators, *Proc. 10th Information Security Conference* (*ISC '06*), pp.1–18 (2007).
16) Kohno, T., Broido, A. and Claffy, K.: Remote physical device fingerprinting, *Proc. 2005 IEEE Symposium on Security and Privacy* (*S&P '05*) (2005).
17) Franklin, J., Luk, M., McCune, J.M., Seshadri, A., Perrig, A. and Doorn, L.V.: Remote Detection of Virtual Machine Monitors with Fuzzy Benchmarking, Technical report, CMU-CyLab-07-001 (2007).
18) Fyodor: Remote OS Detection via TCP/IP Stack Fingerprinting (1998). http://www.nmap.org/nmap/nmap-fingerprintingarticle.html
19) VMware: Timekeeping in VMware Virtual Machines (2005). http://www.vmware.com/resources/techresources/238
20) Postel, J.: RFC 792: INTERNET CONTROL MESSAGE PROTOCOL. http://tools.ietf.org/html/rfc792
21) Postel, J.: RFC 791: INTERNET PROTOCOL. http://tools.ietf.org/html/rfc791
22) Jacobson, V., Braden, R. and Borman, D.: RFC 1323: TCP Extentions for High Performance. http://tools.ietf.org/html/rfc1323
23) Mills, D.L.: RFC 1305: Network Time Protocol (Version 3) Specification, Implementation and Analysis. http://tools.ietf.org/html/rfc1305
24) Medina, A., Allman, M. and Floyd, S.: Measuring the Evolution of Transport Protocols in the Internet, *Computer Communication Review* (2005).
25) Fonseca, R., Porter, G.M., Katz, R.H., Shenker, S. and Stoica, I.: IP Options are not an option, Technical Report EECS-2005-24, Electrical Engineering and Computer Sciences. University of California at Berkeley (2005).
26) Lawrence Berkeley National Laboratory's Network Research Group: TCPDUMP/LIBPCAP public repository. http://www.tcpdump.org/
27) Liston, T. and Skoudis, E.: On the Cutting Edge: Thwarting Virtual Machine Detection (2006). http://handlers.sans.org/tliston/ThwartingVMDetection_Liston_Skoudis.pdf
28) Chen, X., Andersen, J., Mao, Z.M. and Bailey, M.: Towards an Understanding of Anti-virtualization and Anti-debugging Behavior in Modern Malware, *Proc. 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*

(*DSN '08*) (2008).
29) Bellard, F.: QEMU, a Fast and Portable Dynamic Translator, *Proc. 2005 Annual Usenix Technical Conference*, pp.41–46 (2005).

(平成 20 年 10 月 24 日受付)
(平成 21 年 5 月 13 日採録)

推　薦　文

1 パケットで 2 種類のタイムスタンプを取得し，そのタイムスタンプのずれ方の傾向を調べることによって，検査対象となるシステムが仮想マシンモニタ上で稼動しているシステムかどうかをリモートから検出する手法を提案している．また，関連研究との比較を丁寧に論じており，提案手法の優位性を明確に示していること，提案手法を実装し，複数の仮想マシンモニタにおいて提案手法を適用した結果を示しており実践性が高いことから，推薦する．

（コンピュータセキュリティ研究会主査　寺田真敏）

嶋村　　誠（学生会員）

1982 年生．2005 年電気通信大学情報工学科卒業．2007 年慶應義塾大学大学院理工学研究科開放環境科学専攻修士課程修了．現在，同専攻博士課程在学中．オペレーティングシステム，システムソフトウェア，インターネットセキュリティに興味を持つ．IPSJ，IEEE，ACM 各学生会員．

河野　健二（正会員）

1993 年東京大学理学部情報科学科卒業．1997 年東京大学大学院理学系研究科情報科学専攻博士課程中退，同専攻助手に就任．現在，慶應義塾大学理工学部情報工学科准教授．博士（理学）．平成 11 年度情報処理学会論文賞受賞．平成 12 年度山下記念研究賞受賞．オペレーティングシステム，システムソフトウェア，インターネットセキュリティに興味を持つ．IEEE/CS，ACM，USENIX 各会員．