社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－CSEC－41 (12)
2008／5／23

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検出

嶋村 誠　　河野 健二

慶應義塾大学 理工学部 情報工学科

E-mail: tima@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

仮想ハニーポットなどの，仮想マシンモニタ (VMM) 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．攻撃者はこのようなシステム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 VMM の検出を行う．現在の VMM の検出技術では，攻撃者は攻撃対象ホストに侵入した後に VMM の検出を行う必要がある．従って，攻撃者は管理者に攻撃の手口を監視されてしまうという危険がある．しかし，もし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の検出が行えると，攻撃者は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る．本論文では，ネットワークタイムスタンプを用いて，ホストに侵入することなく仮想マシン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これらのタイムスタンプのずれ方に生じる異常を検出することで仮想マシンを検出する．実際に，様々な VMM について，VMM 上で動作するホストに対する実験結果と VMM が動作していないホストに対する実験結果を比較し，仮想マシンの検出が可能であることを示す．

Remote Virtual Machine Scanning Using Network Timestamp

Makoto Shimamura　　Kenji Kono

Department of Information and Computer Science, Keio University
E-mail: tima@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

Many researchers have proposed VMM-based security systems that monitor detailed behaviors of trapped attackers. To avoid being trapped by VMM-based security systems, attackers would try to avoid compromising virtual machines. Currently, attackers detect VMMs after compromising the target, and thus attackers must send malicious attack messages to the target. However, if sophisticated remote VMM scanning is achieved, attackers will be able to detect VMMs without compromising the target. Then, VMM-based security systems will not be effective for trapping skillful attackers. We demonstrate a technique for remotely scanning VMM. The technique regards anomalies in the timestamp discrepancy between timestamps stamped in one packet as evidence of the presence of a VMM. A remote VMM scanner can obtain multiple timestamps from a single packet, which usually indicate the same time. But if the target is a virtual machine, some of the timestamps will show discrepancies in an anomalous way because the VMM sometimes interrupts the timestamp operations of the target. Our experimental results indicate that remote scanning of VMMs is feasible.

1 はじめに

仮想ハニーポット [1, 2, 3, 4, 5] などの，仮想マシンモニタ (VMM) を攻撃の解析に応用したシステムが提案されている．これらのシステムは，VMM を利用し，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．

攻撃者はこれらのシステムに攻撃を行うとデメリットを被る．例えば，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い，防御手段を講じられてしまう．従って，攻撃者は攻撃行動を行う前に，攻撃対象ホストが VMM 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 [6] ワームは，攻撃対象のホストに侵入した後で，仮想マシンやデバッガといった攻撃行動を監視するための環境の検知を試み，監視環境を検知すると攻撃行動を終了する．なお，仮想マシンの検出自体は直接仮想ハニーポットを検出していることにはならない．しかし，攻撃者は仮想マシンを検出することで仮想ハニーポットを回避することができる．

既存の仮想マシン検出手法はローカル仮想マシン検出とリモート仮想マシン検出の 2 種類に大別される．ローカル仮想マシン検出 [6, 7, 8, 9, 10, 11, 12] 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の検出を行う．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検出により仮想マシンを検出し攻撃を中止したとしても，使用された攻撃メッセージなどの情報が収集できる．従って，有用な情報を管理者に与えないためには，ローカル仮想マシン検出ではなく，リモート仮想マシン検出によって事前に仮想マシンを検出する必要がある．

本論文では，リモートスキャンによる仮想マシン検出の可能性を報告し，その一手法を提案する．既存のリモート仮想マシン検出では，対象ホストへのログインや，複数のホストの調査結果との比較を必

要とする．これに対し，リモートスキャンによる仮想マシン検出では，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検出では，対象ホストをネットワーク通信のみで調べることにより，仮想マシンに起因する振る舞いの異常を検出する．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で取得したときに，VMMによる仮想マシンの切り替えや，仮想マシンの時刻の修正によって発生するタイムスタンプ間の大きなずれを検出することで，仮想マシンを検知する．提案手法を様々なVMM上のホストに対して実験した結果，Xen [13]，VMware Workstation [14]，VirtualBox [15] 上で動作する仮想マシン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，まず第2章でリモートスキャンによる仮想マシン検出について説明する．次に第3章でタイムスタンプを用いた仮想マシン検出手法について述べる．その後，第4章で様々なVMM上の仮想マシンについての実験結果を示し，第5章で関連研究について述べ，第6章で本論文をまとめる．

2 リモートスキャンによる仮想マシン検出

2.1 既存のリモート検出手法との比較

本論文において，“リモートスキャンによる仮想マシン検出”とは，対象ホストが仮想マシンであるかどうかを，対象ホスト上で特殊なプログラムの動作を必要とせず，ネットワーク通信だけで決定することである．

この定義に従うと，既存のリモート仮想マシン検出手法はリモートスキャンではない．Kohnoら [16] の手法では，複数の仮想マシンのハードウェアフィンガープリントの一致を行うことで，仮想マシン群を検出できる．しかし，この手法は，複数の仮想マシンが同一VMM上で動作している場合にしか検出可能ではない．また，Franklinら [17] の手法では，ホスト上で仮想マシンの特徴が強く出るプログラムを動作させ，処理時間をリモートで観測することで仮想マシンを検出する．しかし，この手法は，対象ホストにroot権限でアクセスできる必要がある．

また，従来のリモートスキャン手法は，仮想マシンを検出できる状況が非常に限定される．例えば，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社製のVMMを用いていることがわかる．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．

2.2 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に表れる仮想マシンに起因する振る舞いの異常を検出する．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．

複数の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．例えば，ICMPタイムスタンプとIPタイムスタンプは単一のパケットで取得できる．この場合，これらの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 1.86GHz，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り替わりが発生するためである．例えば，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り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．例えば，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．以下では，仮想マシンを検出するために，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通りである．

- **ICMPタイムスタンプ**　ICMPタイムスタンプはRFC 792において定義されている．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 において定義されている．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 において定義されている．Linux では，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．例えば，NTP タイムスタンプなどがある．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 タイムスタンプと IP タイムスタンプ，(2) TCP タイムスタンプと IP タイムスタンプ，(3) アプリケーションタイムスタンプと TCP タイムスタンプ，(4) アプリケーションタイムスタンプと IP タイムスタンプが単一のパケットで取得可能である．

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検出を行う．以下では，単一のパケットで取得した ICMP タイムスタンプと IP タイムスタンプを ICMP/IP タイムスタンプと呼ぶ．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る．

3　タイムスタンプによる仮想マシン検出

3.1　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．例えば，タイムスタンプ T_A が打刻された直後に，時計の時刻が進んだ場合，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 を打刻するまでの処理時間は，(1) 仮想マシンの切り替え，(2) 仮想マシンの時刻の調整が原因で変化する．もし，T_A が打刻された直後に仮想マシンの切り替えが発生すると，タイムスタンプ処理は他の仮想マシンが走行している間中断される．処理が再開した時にタイムスタンプの単位時間以上経過していると，T_B が T_A より大きい値に打刻される．

また，2つのタイムスタンプのずれは，仮想マシンの時刻の調整も原因となる．仮想マシンの時刻はタイマーデバイスのエミュレーションや，仮想マシン切り替えによって，実際の時刻と比べてずれる [16]．従って，VMM は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 が異なる値に打刻される．多くの VMM では実際の時刻に比べて仮想マシンの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 VMM では，タイマーデバイスのエミュレーションによって，仮想マシンの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．

提案手法は，これらのタイムスタンプの異常なずれを検出することで，仮想マシンを検出する．例えば，T_B が T_A より2単位以上大きい時に，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 より小さい時にも，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の台数が増えるとより顕著に観測される．これは仮想マシンの切り替えが仮想マシンの台数が多くなるとより頻繁になるためである．また，仮想マシンの切り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．

3.2　ICMP/IP タイムスタンプのずれ

仮想マシン切り替えと 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，主要な時点において CPU のタイムスタンプカウンタ (TSC) を計測した．計測点は，タイムスタンプを取得したとき (TSC-ICMPTS, TSC-IPTS)，仮想マシンの切り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．図1に示すログでは，下線部において，仮想マシン切り替えによってタイムスタンプのずれが発生している．

また，仮想マシンの時刻の調整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

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 1: 仮想マシンの切り替えによって生じた ICMP/IP タイムスタンプのずれ

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 2: 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

メインにおいて同様の計測を行った．完全仮想化ドメインでは仮想マシン毎に TSC を保持するため，タイムスタンプ時点において，仮想マシンの TSC (vTSC-ICMPTS, vTSC-IPTS) とオフセット (OFF-ICMPTS, OFF-IPTS) を記録し，実際の TSC (rTSC-ICMPTS, rTSC-IPTS) を算出した．図 2 に示すログの下線部では，オフセットが修正され，タイムスタンプのずれが発生した．

3.3 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検出機構は ICMP/IP タイムスタンプを比較して仮想マシン検出を行う．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，仮想マシン切り替え，または仮想マシンの時刻の修正が発生したと見なす．
- **ICMPTS > IPTS**　仮想マシンの場合しか発生しない．検出機構はこの現象を検知すると，仮想マシンの時刻の修正が発生したと見なす．

提案手法による仮想マシン検出では，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．例えば，もし発生確率 p が 1/1,000,000 で検知精度 c が 0.95 である場合，$1-(1-1/1,000,000)^n > 0.95$ を解いて，必要パケット数は約 300 万パケットである．発生確率 p は VMM の実装やハードウェアの種類などに依存する．必要パケット数が多いことは攻撃者にとって大きな問題ではない．攻撃者は多数のマシンを利用した分散スキャンや数ヶ月にわたる低速スキャンを行えばよい．

3.4 実装

提案手法による仮想マシン検出機構を RAW ソケットと libpcap [20] を用いて Linux 上に実装した．RAW ソケットは ICMP/IP 要求タイムスタンプを送信するために用いた．また，libpcap は ICMP/IP タイムスタンプ応答パケットを処理するために用いた．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．

4 実験

4.1 様々な VMM に対する実験

実装した仮想マシン検出機構を用いて，広く使われている VMM に対して実験を行った．提案機構は AMD Opteron 2.2 GHz，RAM 1GB のマシンで動作しており，対象ホストは Intel Core2 Duo E6300 1.86 GHz，RAM 2GB のマシン上で動作している．対象ホストのベースとなる VMM は VMware Workstation 6, Xen 3.1.0a, VirtualBox 1.5.2 オープンソース版である．それぞれの VMM 上に Linux 2.6.18 をインストールし，検出対象ホストとした．また，比較のため Linux 2.6.18 を直接インストールした．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．実験対象とした

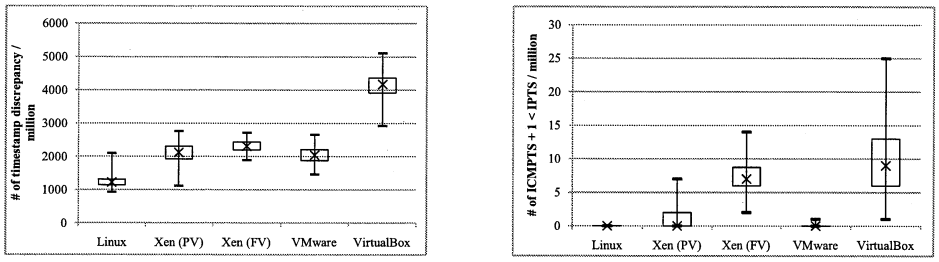

図 3: 仮想マシンと実マシンについての実験結果

ホストは (1) Xen 準仮想化ドメイン (Xen(PV)), (2) Xen 完全仮想化ドメイン (Xen(FV)), (3) VMware Workstation (VMware), (4) VirtualBox (VBox), (5) ハードウェア上の Linux (Linux) である．また，提案機構と対象ホストは 1Gbps 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，100 万応答あたりの ICMPTS + 1 ≠ 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 ミリ秒毎に送信した．実験時間は 1 ケースにつき，14 時間であった．図 3 に実験結果を示す．実マシンの場合には ICMPTS + 1 < IPTS となる応答は発生しなかった．対して仮想マシンの場合には ICMP + 1 < IPTS となった回数が 100 万パケットあたり 0.06 回から 10.14 回であった．従って，提案機構が ICMPTS + 1 < IPTS を検出した場合，対象ホストが仮想マシンであると判断できる．

4.2 ホスト上の仮想マシンの台数との関係

ホスト上の仮想マシンの台数が実験結果に与える影響に対して調査するため，実験を行った．4.1 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 4 に実験結果を示す．“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．実験結果では，仮想マシンの台数が動作するにつれて，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，100 万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．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システムの検出が容易であることを示す．

5 関連研究

リモート仮想マシン検出は，リモートホストから対象ホストに何らかのアクセスを行うことで，仮想マシンを検出する手法である．Kohno らの手法 [16] では，同一ホスト上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．この手法では，フィンガープリントとして，基準時計に対するリモートホストの時計のずれの速度を用いており，この速度をタイムスタンプを利用して取得する．この手法では，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，同じハードウェアフィンガープリントを持つホストを探す必要がある．対して，提案手法では単一のホストを調査し，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，対象ホスト上で CR3 レジスタへのアクセスを利用した，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，動作時間をリモートマシンの時計で測定することで仮想マシンを検出する．しかし，この手法は対象ホストに対する root 権限でのアクセスが必要なため，攻撃者にとっては容易に用いることができない．対して，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い．

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検出手法として，Liston ら [21] 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．

ローカル仮想マシン検出 [6, 7, 8, 9, 10, 11, 12] 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．例えば，VMM と通信するための特殊な命令が存在することや，割り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

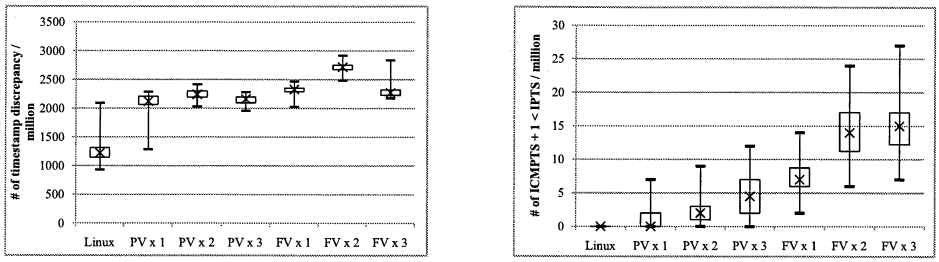

図 4: 仮想マシンの台数を変化させた時の実験結果

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を検出できる．

6 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検出を提案した．また，実験により，様々な仮想マシンモニタ上の仮想マシンが検出可能なことと，仮想マシンの台数が多いほど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検出手法は，攻撃者が仮想機械技術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．従って，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] : Know Your Enemy: Defining Virtual Honeynets (2003). `http://project.honeynet.org/papers/virtual/index.html`.

[2] Jiang, X. and Xu, D.: Collapsar: A VM-Based Architecture for Network Attack Detention Center, *Proc. of the 13th Usenix Security Symposium* (2004).

[3] Vrable, M., Ma, J., Chen, J., Moore, D., Vandekieft, E., Snoeren, A. C., Voelker, G. M. and Savage, S.: Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfirm, *Proc. of the 20th ACM Symposium on Operating Systems Principles* (*SOSP '05*), pp. 148–162 (2005).

[4] Asrigo, K., Litty, L. and Lie, D.: Using VMM-based Sensors to Monitor Honeypots, *Proc. of the 2nd USENIX International Conference on Virtual Execution Environments* (*VEE '06*), pp. 13–23 (2006).

[5] Jiang, X. and Wang, X.: "Out-of-the-box" Monitoring of VM-based High-Interaction Honeypots, *Proc. of the 10th International Symposium on Recent Advances in Intrusion Detection* (*RAID '07*) (2007).

[6] : F-Secure Computer Virus Information Pages: Agobot, `http://www.f-secure.com/v-descs/agobot.shtml`.

[7] Rutkowska, J.: Redpill: Detect VMM using (almost) One CPU Instruction, `http://invisiblethings.org/papers/redpill.html` (2004).

[8] Kato, K.: VMWare's Back, `http://chitchat.at.infoseek.co.jp/vmware/`.

[9] Klein, T.: Jerry - A(nother) VMware Fingerprinter, `http://www.trapkit.de/research/vmm/jerry/index.html` (2003).

[10] Klein, T.: Scooby Doo - VMware Fingerprint Suite, `http://www.trapkit.de/research/vmm/scoopydoo/index.html` (2003).

[11] Ferrie, P.: Attacks on Virtual Machine Emulators, *Proc. of the 9th Annual Association of anti Virus Asia Researchers International Conference* (*AVAR '06*) (2006).

[12] Raffetseder, T., Kruegel, C. and Kirda, E.: Detecting System Emulators, *Proc. of the 10th Information Security Conference* (*ISC '06*), pp. 1–18 (2007).

[13] 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. of the 19th ACM Symposium on Operating Systems Principles* (*SOSP '03*), pp. 164–177 (2003).

[14] Sugerman, J., Venkitachalam, G. and Lim, B.-H.: Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor, *Proc. of the 2001 Annual Usenix Technical Conference*, pp. 25–30 (2001).

[15] : VirtualBox, `http://www.virtualbox.org`.

[16] Kohno, T., Broido, A. and Claffy, K.: Remote physical device fingerprinting, *Proc. of the 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] : Timekeeping in VMware Virtual Machines (2005). `http://www.vmware.com/resources/techresources/238`.

[20] : TCPDUMP/LIBPCAP public repository, `http://www.tcpdump.org/`.

[21] Liston, T. and Skoudis, E.: On the Cutting Edge: Thwarting Virtual Machine Detection, `http://handlers.sans.org/tliston/ThwartingVMDetection_Liston_Skoudis.pdf` (2006).