Computer Security Symposium 2016
11 - 13 October 2016

マルウェアによる対仮想化処理の傾向についての分析

大山 恵弘[1]

概要：マルウェアの中には，自身が仮想マシンモニタなどの仮想化機構によって作られた環境で動作しているかどうかを推定し，もしそうであれば実行を終了して解析を妨害するものがある．そのような処理（対仮想化処理）を実行するマルウェアの存在は広く知られている．しかし，現在の世界のマルウェアのうち，どの程度の割合のマルウェアがどのような対仮想化処理を行い，それが解析の妨害にどの程度効果的であるかについては，知見が不足している．本研究では，マルウェアの動的解析結果のデータセットである FFRI Dataset を分析し，2016 年に収集されたマルウェアによる対仮想化処理の傾向を明らかにする．

キーワード：マルウェア，仮想化，仮想マシンモニタ，ハイパバイザ，ログ解析

Analysis of Trends in Anti-virtualization Operations by Malware

Yoshihiro Oyama[1]

Abstract: Some malware presumes whether it is running in an execution environment created by a virtualization mechanism such as a virtual machine monitor, and if it determines that it is, it terminates the execution to prevent analysis. The existence of malware that executes such operations (anti-virtualization operations) is widely known. However, insufficient knowledge has been collected about (1) how much ratio of current malware in the world executes anti-virtualization operations, (2) what types of anti-virtualization operations it executes, and (3) how effective they are to prevent analysis. We analyze the FFRI Dataset, a dataset of dynamic malware analysis results, and clarify the trends in anti-virtualization operations executed by malware samples collected in 2016.

Keywords: Malware, virtualization, virtual machine monitors, hypervisors, log analysis

1. はじめに

自身の解析を難しくするための処理を実行するマルウェアの存在が広く知られている．主要な処理の一つは，実行環境が仮想マシンか実機かを推定し，仮想マシンである可能性が高いと判定したら実行を終了するというものである．本研究では，このような，解析者による仮想化機構の利用に対する，マルウェア側からの対策処理（**対仮想化処理**）を扱う．

対仮想化処理については多くの研究が行われ，プログラムが仮想化機構の有無を推定する手法や，検出を回避するための仮想化機構の実装方式については，多くの知見が集まっている．しかし，残念ながら，世界に現在広まっているマルウェアに関して，以下の問いに対する答は明らかにはなっていない．

- どの程度の割合のマルウェアが対仮想化処理を実行するのか．
- 多くのマルウェアが実行する対仮想化処理は何か．一方，ほとんど実行されない対仮想化処理は何か．
- 複数の対仮想化処理を実行するマルウェアが多いのか，それとも単一の対仮想化処理だけを実行するマルウェアが多いのか．最も多くの対仮想化処理を実行するマルウェアは，どの程度の数の処理を実行するのか．
- 解析システム上で対仮想化処理を実行するマルウェアは，仮想化機構を検出できているのかいないのか．

解析が難しくなっている高度化されたマルウェアに対抗す

[1] 筑波大学
University of Tsukuba

るためには，これらの問いに対する知見を積み上げることが重要であると考えている．

本研究の目的は，現実のマルウェアによる対仮想化処理の傾向を明らかにすることである．本稿では，マルウェアデータセット FFRI Dataset 2016 [17] に含まれる 8243 のマルウェアの動的解析結果を対象にして傾向を分析した結果を報告する．

本研究は特に以下の点に関して有益な情報を提供する case study であると考えている．

- 対仮想化処理に対して解析システムが対策を講じなかった場合に，どの程度の割合のマルウェアの解析が難しくなる可能性があるか．
- セキュリティシステムが優先的に対処すべき対仮想化処理は何か．
- 広範な対仮想化処理を実行するマルウェアを解析するためには，解析システムはどの程度多くの対策を講じる必要があるか．
- 対仮想化処理を実行する現実のマルウェアを，動的解析システムはうまく解析できているか．
- 既存のマルウェア動的解析システムには具体的にどのような制限があるか．
- マルウェアの動的解析ログのデータセットのみを用いた分析では，どういう知見は得られて，どういう知見を得るのは難しいか．

2. 対象とする対仮想化処理

対仮想化処理の実装に利用できる主な情報を以下に示す．

(1) ファイル：仮想マシンモニタとの連携のために仮想マシン上で動く OS 上に作られるプログラムファイルや設定ファイルの有無や内容など
(2) ハードウェア情報：ハードディスクなどのデバイスの型番，BIOS から返されるハードウェアの情報など
(3) 入出力操作結果：in 命令や out 命令などの入出力デバイス操作命令の実行結果など
(4) 命令実行結果：仮想マシンモニタが厳密なエミュレーションを行わないなどの理由により，仮想マシン上と実機上で挙動が変わる CPU 命令の実行結果など
(5) 時間情報：特定の処理の実行に要する時間や，プロセスへの CPU 時間の割り当てパターンなど

どの情報も仮想化機構の検出には効果的であるため，マルウェアはどの情報の取得も試みる可能性がある．理想的には，上記の全ての情報について分析することが望ましい．しかし，FFRI Datasets は挙動情報として CPU 命令レベルのトレースを含まないため，FFRI Datasets を用いて全てを分析するのは難しい．本研究では，Windows API 呼び出し列から多くの情報を得られるもの，すなわち，(1) と (2) の対仮想化処理について分析することに集中した．

3. データセット

FFRI Datasets [17] は，仮想マシンモニタを用いたマルウェア解析用サンドボックスである Cuckoo Sandbox の上でマルウェアを実行して得られた動的解析結果のデータセットである．本研究では FFRI Dataset 2016 に含まれる Windows 10 (x64) 上での 8243 のマルウェアの解析結果を対象とする．これらのマルウェアは 2016 年の 1 月から 3 月までに収集されている．データセット内の情報によると，FFRI Dataset 2016 の作成には Cuckoo Sandbox 2.0-dev と VirtualBox が用いられた．

Cuckoo Sandbox はマルウェアの挙動の特徴とマルウェアの動作を照合する機能を提供している．各特徴は *signature* と呼ばれる．Cuckoo Sandbox が検出した signature は解析結果に記録される．各マルウェアに対して，各 signature が検出されたかどうかの情報が付与される．Signature を検出する処理は Cuckoo Sandbox 本体とは別の community と呼ばれるソースツリーで維持されている．FFRI Dataset 2016 には，community のコードが検出した signature が含まれている．

FFRI Datasets は，マルウェアファイルの静的情報やマルウェアの実行中に発生した通信の情報などの様々な情報を含む．本研究ではそれらのうち，signature の検出結果とマルウェアによる Windows API 呼び出しの列を分析する．

FFRI Dataset 2016 では全てのマルウェアの解析結果に `lsass.exe` というプロセスの情報が含まれているが，これはマルウェアのプロセスではないため分析対象から外す．

なお，API 呼び出し列に現れない処理をデータセットから把握するのは難しい．例えば，文献 [15] の手法のようにマルウェアが時間情報を利用して，API 呼び出し列に痕跡を残さずに仮想化機構の有無を判定していたとしても，それをデータセットからは把握できない．本研究の結果を解釈する上でその点には留意する必要がある．

4. 分析

4.1 検出された signature

Signature の検出数上位 15 件を**表 1** に示す．なお，signature の全検出数の総和は 41613（1 マルウェアあたりの平均は 5.05）であり，少なくとも 1 回は検出された signature は 90 種類ある．1 マルウェアから検出された signature の種類数は最大が 16，最小が 1 である．後述する対仮想化処理の signature は 15 位に 1 件入っている．

Cuckoo Sandbox には，`antivm_` および `antiemu_` という接頭語が付いた関数で検出される，対仮想化処理の signature がある．本稿では，関数名から接頭語を除いた文字列を signature 名として用いる．それらの signature のうち FFRI Dataset 2016 に含まれるものを，検出数の多い順に

表 1 検出されたマルウェアの数が多い signature
Table 1 Signatures detected in many malware samples

	Cuckoo Sandbox による signature の説明	該当マルウェアの数と割合
1	File has been identified by at least one AntiVirus on VirusTotal as malicious	8243 (100.0%)
2	Performs some HTTP requests	5332 (64.7%)
3	Allocates read-write-execute memory (usually to unpack itself)	4615 (56.0%)
4	Generates some ICMP traffic	4162 (50.5%)
5	One or more potentially interesting buffers were extracted, ...	2858 (34.7%)
6	Creates executable files on the filesystem	2214 (26.9%)
7	One or more of the buffers contains an embedded PE file	2116 (25.7%)
8	The executable has PE anomalies (could be a false positive)	2006 (24.3%)
9	Executed a process and injected code into it, probably while unpacking	1486 (18.0%)
10	Installs itself for autorun at Windows startup	1378 (16.7%)
11	This executable has a PDB path	1091 (13.2%)
12	Collects information to fingerprint the system ...	1029 (12.5%)
13	Code injection with CreateRemoteThread or NtQueueApcThread in a remote process	579 (7.0%)
14	A process attempted to delay the analysis task.	509 (6.2%)
15	Checks the version of Bios, possibly for anti-virtualization	443 (5.4%)

表 2 対仮想化処理の signature
Table 2 Signatures of anti-virtualization operations

Signature 名	Cuckoo Sandbox による signature の説明	該当マルウェアの数と割合
`generic_bios`	Checks the version of Bios, possibly for anti-virtualization	443 (5.4%)
`vmware_files`	Detects VMWare through the presence of various files	358 (4.3%)
`generic_scsi`	Detects virtualization software with SCSI Disk Identifier trick(s)	113 (1.4%)
`vbox_keys`	Detects VirtualBox through the presence of a registry key	34 (0.4%)
`vbox_files`	Detects VirtualBox through the presence of a file	29 (0.4%)
`vmware_keys`	Detects VMWare through the presence of a registry key	14 (0.2%)
`generic_disk`	Queries information on disks, possibly for anti-virtualization	11 (0.1%)
`generic_services`	Enumerates services, possibly for anti-virtualization	7 (0.1%)
`generic_firmware`	Detects Virtual Machines through their custom firmware	2 (0.02%)
`sandboxie`	Tries to detect Sandboxie	2 (0.02%)
`virtualpc`	Tries to detect VirtualPC	1 (0.01%)
`wine`	Detects the presence of Wine emulator	1 (0.01%)

表 2 に示す．最も多くのマルウェアで signature が検出された仮想マシンモニタは VMware であり，次いで VirtualBox である．Sandboxie，VirtualPC，Wine の signature はほとんど検出されていない．

対仮想化処理の signature を検出するために Cuckoo Sandbox が用いている基準を**表 3** に示す．どの signature も，API 呼び出しの関数名や引数を特徴的なファイルアクセスやレジストリアクセスのパターンと照合することによって検出している．

FFRI Dataset 2016 の収集で用いられたと思われる community 2.0 のソースコードには，`antivm_` と `antiemu_` の接頭語のついた関数は，合計 18 個ある．表 2 のもの以外には以下の 6 種類の signature が提供されているが，FFRI Dataset 2016 の実行では 1 回も検出されていない．

`generic_ide`，`vbox_acpi`，`vbox_devices`，
`vbox_window`，`virtualpc_magic`，`vmware_in_insn`

Community の最新版（2016 年 8 月 12 日現在）のソースコードには，加えて，以下の 7 種類の signature が提供されているが，当然ながら，どれも 1 回も検出されていない．

`bochs_keys`，`computername`，`generic_cpu`，
`hyperv_keys`, `parallels_keys`, `vpc_keys`, `xen_keys`

これらの 7 種類の signature はデータセット収集環境には導入されていなかったと考えられるため，Bochs，Hyper-V，Parallels，Xen に関する signature が検出されなかったのは自然である．なお，これらの signature については，本研究において，最新版のソースコードに含まれる検出基準をデータセットに適用して signature の検出を試みたが，いずれも検出されなかった．ただ，API 呼び出しの多くの引数に，`Hyper-V` や `microsoft-hyper-v` という文字列が出現しており，Hyper-V に関しては，マルウェアが何らかの対仮想化処理を実行している可能性は排除できない．

表 2 の signature が各マルウェアから何種類検出されたかを調査した．結果を**表 4** に示す．1 マルウェアから検出される対仮想化処理の signature は 1 種類か 2 種類であることが多く，3 種類以上が検出されるマルウェアは全体の約 0.4%である．4 種類以上に至っては 8243 検体中 9 検体だけしか存在しない．広範な対仮想化処理を実行するマルウェアは少ないという結果になっている．

表 3 Signature 検出基準（部分的に省略および簡略化している）
Table 3 Signature detection standards (partly abbreviated or simplified)

vmware_files	アクセスされるファイルパスの末尾が vmmouse.sys, vmhgfs.sys, hgfs, vmci などにマッチする
vmware_keys	アクセスされるレジストリキーが \\SOFTWARE\\(Wow6432Node\\)?VMWare, Inc. などにマッチする
vbox_files	アクセスされるファイルやロードされる DLL のパスが VBoxVideo.[a-zA-Z]{3} などにマッチする
vbox_keys	アクセスされるレジストリキーの末尾が \\SOFTWARE\\Oracle\\VirtualBox Guest Additions などにマッチする
generic_bios	アクセスされるレジストリキーの末尾が SystemBiosVersion や VideoBiosVersion などにマッチする
generic_disk	引数に IOCTL_DISK_GET_DRIVE_GEOMETRY を与えて DeviceIoControl が呼び出されるなど
generic_scsi	アクセスされるレジストリキーが，指定の文字列パターン（Scsi や Disk などが含まれる）にマッチする
generic_firmware	引数に SystemFirmwareTableInformation を与えて NtQuerySystemInformation が呼び出される
generic_services	EnumServicesStatusA や EnumServicesStatusW が呼び出されるなど
sandboxie	アクセスされるファイルやロードされる DLL のパスの末尾が sbiedll(.dll)? にマッチするなど
wine	アクセスされるレジストリキーが \\HKEY_CURRENT_USER\\Software\\Wine にマッチするなど
virtualpc	MicrosoftVirtualPC7UserService... にマッチする文字列で識別される mutex が open されるなど

表 4 各マルウェアから検出された対仮想化処理の signature の数
Table 4 Number of anti-virtualization signatures detected in each malware sample

Signature の数	0	1	2	3	4	≥5
マルウェアの数	7678	168	353	35	9	0

4.2 仮想化機構検出処理が実行されるプロセス

実行中にプロセスを生成するマルウェアもある．対仮想化処理は，マルウェアの最初のプロセスが実行する場合もあれば，生成された子プロセスが実行する場合もある．

対仮想化処理の signature が検出されたマルウェアを対象に，signature 検出基準に合致する API 呼び出しの数（成功失敗を問わない）と，そのような呼び出しを含むプロセスのプログラム名と数を求めた．結果を**表 5** に示す．なお，FFRI Dataset 2016 では，マルウェアファイルの名前は「ファイルの SHA256 ハッシュ値.exe」となっている．表には API 呼び出し数上位の 8 件のみが書かれているが，以降は，iesecure.exe, splwow64.exe, verifiergui.exe, winhlp32.exe, eventvwr.exe, ... と続く．対仮想化処理は，マルウェアの最初のプロセスが実行することも多いが，他のプロセスが実行することも多いことがわかる．表中のハッシュ値ではないファイル名のプログラムは，どれも OS が提供しているユーティリティである．対仮想化処理は，これらのユーティリティが本来の処理として呼び出している可能性もあるが，これらのユーティリティを実行するプロセスにマルウェアがコードを注入して呼び出している可能性もある．

表 5 対仮想化処理を実行したプロセス
Table 5 Processes executing an anti-virtualization operation

プログラム名	API 呼び出し数	プロセス数
ハッシュ値.exe	957	234
explorer.exe	1915	734
hh.exe	189	13
helppane.exe	144	8
regedit.exe	125	9
bfsvc.exe	62	14
notepad.exe	54	10
write.exe	54	10

表 6 対仮想化処理に関する API 呼び出しの成否
Table 6 Success and failure of API calls related to anti-virtualization operations

Signature 名	成功呼び出し数	失敗呼び出し数
generic_bios	1503	345
vmware_files	0	751
generic_scsi	745	0
vbox_keys	0	174
vbox_files	0	101
vmware_keys	0	76
generic_disk	25	0
generic_firmware	10	6
generic_services	7	7
sandboxie	0	3
wine	0	2
virtualpc	0	1

4.3 個別の signature についての議論

4.3.1 概観

マルウェアにとって対仮想化処理が成功しているか失敗しているかを評価するために，各 signature に関する API 呼び出し全てについて，成功した数と失敗した数を調査した．Signature が検出されたマルウェアの全 API 呼び出しに対して，関数名と引数を Cuckoo Sandbox が用いているパターンと照合することにより数を求めた．Cuckoo Sandbox は，引数がパターンにマッチしていても，呼び出しが失敗した場合には signature を検出しないことがあるが，本研究では成功と失敗の両方を数えた．呼び出しが成功したか失敗したかは，データセットに含まれる返り値や NTSTATUS エラーコードの値と，各 API の仕様を元に決定した．

結果を**表 6** に示す．仮想化機構に関する呼び出し（vmware_*, vbox_*, sandboxie, wine, virtualpc）は 100%失敗している．一方，ディスクに関する呼び出し（generic_scsi と generic_disk）は，100%成功している．BIOS のバージョンの検査に関する呼び出し

(generic_bios)は, 成功することも失敗することもあった. ファームウェア情報の検査とサービスの列挙に関する呼び出し (generic_firmware と generic_services) は, 後述するが, 事実上は 100%成功している. 以下で各 signature について説明する.

4.3.2 BIOS のバージョンの検査

多くの仮想マシンモニタは固有の仮想 BIOS を提供している. マルウェアは BIOS のバージョンの情報により, 仮想マシンモニタの存在の有無や種類を, より高い精度で推定できる.

BIOS のバージョンは対仮想化処理以外でも利用されうるため, BIOS のバージョンを検査したからといって, 仮想化機構を検出しようとしていると即断することはできない. 実際, Cuckoo Sandbox でもこの signature は "possibly for anti-virtualization" という曖昧な表現で説明されている. 同様の表現は generic_disk と generic_services でも用いられており, これらに関する API 呼び出しは, 対仮想化処理であるとは限らない.

この signature は 443 と最も多くのマルウェアから検出されており, この signature に関する API 呼び出しの数も最多である. 呼び出しの成功と失敗は, アクセスしたレジストリキーで決まっている. HKEY_LOCAL_MACHINE\\HARDWARE の下にあるキーへのアクセスは全て成功しており, HKEY_LOCAL_MACHINE\\SOFTWARE の下にあるキーへのアクセスは全て失敗している. 成功した呼び出しでは, SystemBiosVersion として "LENOVO - 2020" か "LENOVO - 3220" を取得し, VideoBiosVersion として "Hardware Versio" を取得している. Cuckoo Sandbox は実機の BIOS とは異なる BIOS の情報を返すよう実装されている. SystemBiosVersion としては上記 2 種類の文字列の片方をランダムに返し, VideoBiosVersion としては "Hardware Version 0.0" を返す. これらの文字列は公開情報であることから, 自身が Cuckoo Sandbox の上で動いているかどうかを推定するプログラムを作成することは容易である. 今回用いたデータセットのマルウェアの中に, Cuckoo Sandbox を意識するものがあったとすると, そのマルウェアは Cuckoo Sandbox の存在を認識した可能性は高い. しかし, 4.3.8 節で述べるが, そのようなマルウェアは少なかったと推測している.

4.3.3 仮想マシンモニタの検出

仮想マシンモニタの検出に関する API 呼び出しは必ず失敗していた. VMware に関するファイルやレジストリキーの open に失敗したのは, データセット収集環境が VirtualBox ベースであることから当然である. VirtualPC に関しては, 全マルウェアの実行中で 1 回だけ, MicrosoftVirtualPC7UserServiceMakeSureWe're TheOnlyOneMutex という引数での NtCreateMutant 関数の呼び出しに対してのみ, signature が検出されている. この呼び出しも失敗しているが, それは同じ理由で自然である. ただ, VirtualBox に関するファイルやレジストリキーの open にも必ず失敗していた. その理由は, ゲスト OS に VirtualBox Guest Additions が入っていなかったためと考えている. Cuckoo Sandbox 2.0-dev は, VirtualBox に関する signature の検出基準に, VirtualBox Guest Additions のファイルへのアクセスを用いる. そして, 今回用いたデータセットでは, そのアクセスは全て失敗している.

4.3.4 Sandboxie, Wine の検出

Sandboxie は仮想マシンモニタではないが, 信頼できないプログラムを動かすための仮想環境を作ることを可能にするソフトウェアである. ファイルなどの資源に対する操作は仮想化された資源に対して適用され, 実資源に影響を及ぼすことはない. Sandboxie に関する signature は 2 つのマルウェアの実行で検出されている. 1 つのマルウェアは Sandboxie のライブラリ SbieDll.dll を引数として LdrLoadDll 関数を 2 回呼び出している. もう 1 つのマルウェアは Sandboxie_SingleInstanceMutex_Control を引数として NtCreateMutant 関数を 1 回呼び出している. どの呼び出しも失敗している.

Windows エミュレータ Wine に関する signature は 1 つのマルウェアで検出されている. RegOpenKeyExW 関数により, HKEY_CURRENT_USER\\Software\\WINE というレジストリキーが 2 回アクセスされ, 両方失敗している.

データセット収集環境では Sandboxie も Wine も用いられていないと予想されるため, 全ての呼び出しが失敗したのは自然である.

4.3.5 ディスクハードウェア情報の検査

仮想マシンモニタは特定の仮想ディスクハードウェアを仮想マシンに提供していることが多い. 実行環境が仮想環境であるかどうかを判定する上で, ディスクハードウェアの情報は重要な手がかりとなる.

generic_scsi が検出される API 呼び出しは, SCSI デバイスの identifier に関するレジストリキー, および, ディスクのサービスに関するレジストリキーへのアクセスである. 通常, これらのキーには, 製品の型番などのディスクハードウェアを識別する情報が入っている.

Cuckoo Sandbox 2.0-dev は, 偽装したハードウエア情報を仮想マシンに提供する. 具体的には, SCSI デバイスのレジストリの値に vbox, vmware, qemu, virtual などの文字列が含まれる場合には, その値を Seagate のハードディスクを表す ST9160411AS という文字列やランダム文字列に置き換えている. 今回用いたデータセットでは, ST9160411AS という文字列は, 33 のマルウェアで API 呼び出しの引数に現れている.

generic_disk が検出されるのは, physicaldrive0 や scsi0 という文字列が含まれるファイルパスを与えての NtCreateFile 関数などの呼び出しや, 所定の制御コード

を与えての DeviceIoControl 関数などの呼び出しである．

これらの signature に関する API 呼び出しは全て成功していた．ディスクハードウェア情報を用いて仮想化機構の有無を推定するマルウェアがあったとすると，そのマルウェアが仮想化機構を検出した可能性は高い．上記のように，公知の偽装情報もいくつかのマルウェアには伝わっている．しかし，4.3.8 節で述べるが，そのようなマルウェアは少なかったと推測している．

4.3.6 サービスの列挙

動いているサービスを列挙する処理の signature が検出されたマルウェアの数は 2 と少なく，API 呼び出しの数も 14 と少ない．14 の呼び出しのうち，7 回は成功，7 回は失敗である．しかし，この失敗の呼び出しは，ERROR_MORE_DATA というデータの継続を意味するエラーコードを返すものであり，事実上は，それらも含めて全ての呼び出しは成功している．マルウェアはゲスト OS 上で動くサービスの情報を問題なく取得したと考えられる．

4.3.7 ファームウェア情報の検査

ファームウェア情報を検査する処理の signature が検出されたマルウェアの数は 2 と少なく，API 呼び出しの数も 16 と少ない．どちらのマルウェアも SystemFirmwareTableInformation という引数で NtQuerySystemInformation 関数を呼び出しており，それが signature として検出されている．呼び出しは 10 回が成功，6 回が失敗であったが，失敗は全て結果を格納するバッファのサイズの不足に起因するものであり，どの失敗の直後にも同じ関数の呼び出しが成功している．よって，ファームウェア情報を得る試みは全て成功したと考えられる．

4.3.8 対仮想化処理の認識

FFRI Dataset 2016 のマルウェアの中には，Cuckoo Sandbox を認識したものは少ないと推測している．その根拠は，対仮想化処理の実行後にすぐにプロセスを終了せずに，活動を続けているマルウェアが多いからである．各マルウェアプロセスの全 API 呼び出し列のうち，対仮想化処理の signature に関する最後に成功した API 呼び出しが出現した位置の平均を**表 7** に示す．ここで位置とは，API 呼び出し列において，当該の呼び出しが行われた順番の数を，列に含まれる全呼び出し数で割った値である．また，最後に成功した API 呼び出しからプロセス終了までに行われた API 呼び出し数の平均も，合わせて表 7 に示す．もしマルウェアが仮想化機構を検出した後にすぐに実行を終了する場合には，対仮想化処理は API 呼び出し列の終盤に実行される結果になるはずである．しかし，実際にはそうはなっていない．加えて，マルウェアは単なる実行終了処理に要すると考えられる数（数十程度）を超える数の API 呼び出しを実行している．すなわち，割合のみならず呼び出しの数の面でも，仮想化機構を検出していないマルウェアが多いと考えられる結果が出ている．

表 7 対仮想化処理 signature に関する，最後に成功した API 呼び出しの出現位置の平均と，その後に行われた API 呼び出し数の平均

Table 7 Average positions of the last successful API call related to an anti-virtualization operation and average numbers of API calls performed after the last call

Signature 名	出現位置	API 呼び出し数
generic_bios	18.9%	2995
generic_scsi	68.2%	1552
generic_disk	40.1%	6360
generic_firmware	59.5%	240
generic_services	19.6%	300

generic_disk, generic_firmware, generic_services が検出されたマルウェアについては，数が少ないため，最後の API 呼び出し後の挙動を個別に精査した．その結果，20 のうち 18 のマルウェアが，ファイルへの書き込み，レジストリキーへの書き込み，プロセス生成，別プロセス内へのスレッド生成のどれかを必ず実行していることがわかった．これらの処理は攻撃活動でよく用いられる．これら 18 のマルウェアは Cuckoo Sandbox を検出しなかったか検出できなかった結果，活動を続けたと考えられる．残りの 2 つのマルウェアは，攻撃とみなしうる処理を実行しないまま終了した．2 つのうち 1 つは，4.5.3 節で述べるマルウェアである．

4.4 見逃されている挙動

セキュリティソフトウェアがあらゆる対仮想化処理を検出するのは不可能である．FFRI Dataset 2016 の中にも，Cuckoo Sandbox が見逃していると思われる対仮想化処理が散見される．それらのうち主要なものを以下に述べる．

- 引数を検査される API 関数が網羅的ではない．例えば，LdrLoadDll 関数の引数は検査されるが，LdrGetDllHandle 関数の引数は検査されない．その結果，あるマルウェアは Sandboxie を特徴付ける文字列 sbiedll.dll を引数として LdrGetDllHandle を呼び出しているが，見逃されている．
- 一部の API 関数では，失敗した呼び出しの引数が検査されなくなる．例えば，NtCreateFile 関数の引数は，signature 検出のパターンにマッチしていたとしても，呼び出しが失敗すると照合対象から外されることがある．その結果，VirtualBox を特徴づける VBoxGuest や VBoxMiniRdrDN などの文字列を含む引数が全て見逃されている．解析者が対仮想化処理の挙動を漏れ無く把握できるように，呼び出しの成功失敗に関わらず，パターンにマッチする API 呼び出しを試みたことそれ自体を signature として検出すべきであると考える．
- ファイルやレジストリキーのパターンマッチが網羅的ではない．例えば，一部のマルウェアは vmmemctl と

いう VMware が用いるファイルを開こうとするが，この文字列は signature 検出のパターンに含まれないため，見逃されている（当該 API 呼び出しはファイルが無いことにより失敗しているので，二重の理由で見逃されている）．

4.5 個別のマルウェアの挙動

注目すべきいくつかのマルウェアについて以下に述べる．

4.5.1 ハッシュ値 8a1279...

Symantec と Microsoft による検査では，このマルウェアはそれぞれ `Infostealer.Limitail` と `TrojanSpy:Win32/Ursnif.HN` と判定されている．しかし，両社が公開しているこのマルウェアの挙動は，データセットからは確認できない．本節の他のマルウェアについても同様である．

最多の 16 種類の signature が検出されている．そのうち仮想化関連のものは `generic_bios` と `vmware_files` のみである．マルウェアの特徴を多く備えるからといって，必ずしも，広範な対仮想化処理を実行するとは限らないことが分かる．なお，このマルウェアは対仮想化処理の実行後にも，通信などの様々な処理を実行しており，仮想化機構を検出して実行を終了するような挙動は見せていない．

4.5.2 ハッシュ値 bef62f...

Symantec と Microsoft による検査では，このマルウェアはそれぞれ `Trojan Horse` と `Trojan:Win32/Toga!rfn` と判定されている．

Sandboxie に関する signature が検出された 2 つのマルウェアのうちの 1 つであり，ファームウェア情報の検査に関する signature が検出された 2 つのマルウェアのうちの 1 つであり，Wine に関する signature が検出された唯一のマルウェアである．

このマルウェアはファイル `VBoxGuest` を `NtCreateFile` 関数により開こうとしているが，4.4 節で述べた理由により，見逃されている．

4.5.3 ハッシュ値 f67724...

Symantec と Microsoft による検査では，このマルウェアはそれぞれ `Trojan Horse` と `TrojanSpy:Win32/Ursnif.HN` と判定されている．

このマルウェアからは，ファームウェア情報の検査，Sandboxie の検出，VirtualPC の検出という稀有な signature の全てが検出されている．このマルウェアは，これらの対仮想化処理を実行した後，すぐに実行終了処理に入っている．ところが，このマルウェアは Sandboxie と VirtualPC を検出していない．よって，もしこのマルウェアが仮想化機構を検出していたとすると，それはファームウェア情報や性能などの他の情報からである可能性が高い．ただ，現段階では，実行を終了した理由が仮想化機構の検出であるかどうか自体が特定できない．それを特定するには，検体や実行命令列などの他の情報を元に，さらに深い調査を行う必要がある．

VirusTotal（`https://www.virustotal.com/`）の情報によると，このマルウェアは Dr. Watson というソフトウェアのファイルへの書き込みや同ソフトウェアのプロセスの生成を行うが，データセットにはその挙動は見られない．

なお，signature として検出されてはいないものの，このマルウェアは他にも興味深い処理を実行している．例えば，`GetNativeSystemInfo` 関数で CPU 数を取得したり，100 ミリ秒ごとにカーソル位置を繰り返し取得したり，`C:\\Sandbox` というフォルダを開こうとしたりしている．

5. 関連研究

仮想化機構の有無をプログラムが推定するための技術については多くの研究が存在する．Garfinkel らの研究 [5] や Raffetseder らの研究 [13] では，多くの技術が比較されている．これらの研究では各技術を定性的に議論するにとどまっており，現実のマルウェアがどの技術をどの程度の割合で用いているかは明らかにしていない．

Chen らの研究 [2] では，APT 攻撃のマルウェアと一般のマルウェアの間で，デバッガや仮想マシンを意識した処理の傾向がどう異なるかを評価している．多くのマルウェア検体を用いて対仮想化処理の傾向を分析しているのは本研究と同じであるが，異なる点も多く存在する．まず，彼らの研究では対仮想化処理を静的解析でのみ検出しており，動的挙動の把握はしていない．また，`Sleep` 関数の呼び出しを対仮想化処理とみなすなど，本研究よりも粗い検出基準を採用している．さらに，彼らの研究は 2 種類のマルウェアの差異に焦点を当ててており，マルウェアが実行する対仮想化処理の詳細を明らかにしていない．

Chen らの研究 [3] では，マルウェアによる仮想マシンやデバッガを意識した挙動を分類するとともに，6222 のマルウェアを VMware Server の仮想マシン上と実機上で実行して，挙動の違いを評価している．本研究と異なり，この研究が示している実験結果は仮想マシン上で挙動を変えるマルウェアの割合のみであり，マルウェアが実行した具体的な対仮想化処理は明らかにされていない．

仮想化機構を検出して解析を回避するマルウェアを解析する手法については多くの研究が存在する [1,4,6–12,14,16]．これらの研究と本研究は補完的な関係にある．マルウェアが実行する対仮想化処理をより良く理解することにより，マルウェア解析手法のさらなる改良が見込める．逆に，洗練されたマルウェア解析手法を採用することにより，マルウェアの挙動をさらに詳しく分析することができる．

6. まとめと今後の課題

FFRI Dataset 2016 で観測された，仮想化機構を意識した処理を実行するマルウェアの傾向について報告した．本

研究で得られた知見を以下にまとめる．これらはあくまでFFRI Dataset 2016 のマルウェアデータセットと Cuckoo Sandbox 2.0-dev に見られる傾向であり，それがマルウェア全般や他の解析システムでも観測される一般的な傾向であるかどうかについては，さらなる調査が必要である．

- 8243 のマルウェア中，565（6.9%）のマルウェアは，対仮想化処理を少なくとも 1 つ実行する．それらのマルウェアは，本研究で対象とした対仮想化処理 12 種類のうち，1 種類か 2 種類だけを実行することが多く，5 種類以上実行するものは無い．
- 最も多くのマルウェアが意識する仮想マシンモニタは VMware であり，次いで VirtualBox である．Bochs，Hyper-V，Parallels，Xen に関する挙動については，おそらく，データセット収集環境における当該 signature 処理の未対応により，検出できていない．なお，Hyper-V に関する文字列が引数に与えられた API 呼び出しはデータセット中に多数存在する．
- 対仮想化処理はマルウェアのプロセス自身が実行することが多いが，マルウェアが生成した子プロセスが実行することも，無視できないほど多い．
- 固有のファイルやレジストリキーをアクセスして仮想マシンモニタを検出しようとするマルウェアの試みは全て失敗していると予想される．一方，ハードウェア情報やサービスの情報による仮想化機構の検出については，成功しているか失敗しているかは断定できない．
- 一部のマルウェアは，ハードウェア情報の取得の後，そのマルウェアが本来の目的としていると考えられる処理を実行せずに終了する．
- 検査すべき関数やマッチすべきパターンが網羅されていないことなどにより，Cuckoo Sandbox が見逃していると思われる対仮想化処理が存在する．

今後の課題としては，まず，性能情報や CPU 命令の実行結果も組み合わせて用いて，API 呼び出し列のみからは把握できない挙動を明らかにすることがある．また，今回用いたデータセットには同一マルウェアの亜種が含まれている可能性があるので，亜種の存在が分析結果に与えた影響を評価することも必要である．

謝辞 FFRI Datasets を提供して下さった (株)FFRI および MWS 組織委員会に感謝します．本研究は JSPS 科研費 26330080 の助成を受けたものです．

参考文献

[1] Balzarotti, D., Cova, M., Karlberger, C., Kruegel, C., Kirda, E. and Vigna, G.: Efficient Detection of Split Personalities in Malware, *Proc. of the 17th Annual Network and Distributed System Security Symposium* (2010).

[2] Chen, P., Huygens, C., Desmet, L. and Joosen, W.: Advanced or Not? A Comparative Study of the Use of Anti-debugging and Anti-VM Techniques in Generic and Targeted Malware, *Proc. of the 31st IFIP International Conference on ICT Systems Security and Privacy Protection*, pp. 323–336 (2016).

[3] Chen, X., Andersen, J., Mao, Z. M., Bailey, M. and Nazario, J.: Towards an Understanding of Anti-virtualization and Anti-debugging Behavior in Modern Malware, *Proc. of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 177–186 (2008).

[4] Dinaburg, A., Royal, P., Sharif, M. and Lee, W.: Ether: Malware Analysis via Hardware Virtualization Extensions, *Proc. of the 15th ACM Conference on Computer and Communications Security*, pp. 51–62 (2008).

[5] Garfinkel, T., Adams, K., Warfield, A. and Franklin, J.: Compatibility is Not Transparency: VMM Detection Myths and Realities, *Proc. of the 11th Workshop on Hot Topics in Operating Systems* (2007).

[6] Kang, M. G., Yin, H., Hanna, S., McCamant, S. and Song, D.: Emulating Emulation-Resistant Malware, *Proc. of the 2nd ACM Workshop on Virtual Machine Security*, pp. 11–22 (2009).

[7] Kirat, D., Vigna, G. and Kruegel, C.: BareCloud: Bare-metal Analysis-based Evasive Malware Detection, *Proc. of the 23rd USENIX Security Symposium*, pp. 287–301 (2014).

[8] Lengyel, T. K., Maresca, S., Payne, B. D., Webster, G. D., Vogl, S. and Kiayias, A.: Scalability, Fidelity and Stealth in the DRAKVUF Dynamic Malware Analysis System, *Proc. of the 30th Annual Computer Security Applications Conference*, pp. 386–395 (2014).

[9] Lindorfer, M., Kolbitsch, C. and Comparetti, P. M.: Detecting Environment-Sensitive Malware, *Proc. of the 14th International Symposium on Recent Advances in Intrusion Detection*, pp. 338–357 (2011).

[10] Nguyen, A. M., Schear, N., Jung, H., Godiyal, A., King, S. T. and Nguyen, H. D.: MAVMM: Lightweight and Purpose Built VMM for Malware Analysis, *Proc. of the 2009 Annual Computer Security Applications Conference*, pp. 441–450 (2009).

[11] Oyama, Y., Kawasaki, Y. and Takahashi, K.: Checkpointing an Operating System Using a Parapass-through Hypervisor, *Journal of Information Processing*, Vol. 23, No. 2, pp. 132–141 (2015).

[12] Pektaş, A. and Acarman, T.: A dynamic malware analyzer against virtual machine aware malicious software, *Security and Communication Networks*, Vol. 7, No. 12, pp. 2245–2257 (2014).

[13] Raffetseder, T., Kruegel, C. and Kirda, E.: Detecting System Emulators, *Proc. of the 10th Information Security Conference*, pp. 1–18 (2007).

[14] Zhang, F., Leach, K., Stavrou, A., Wang, H. and Sun, K.: Using Hardware Features for Increased Debugging Transparency, *Proc. of the 36th IEEE Symposium on Security and Privacy*, pp. 55–69 (2015).

[15] 宮本久仁男，田中英彦：特徴データベースを用いない効率的な仮想マシンモニタ検出方式の提案，情報処理学会論文誌，Vol. 52, No. 9, pp. 2602–2612 (2011).

[16] 大月勇人，瀧本栄二，齋藤彰一，毛利公一：マルウェア観測のための仮想計算機モニタを用いたシステムコールトレース手法，情報処理学会論文誌，Vol. 55, No. 9, pp. 2034–2046 (2014).

[17] 高田雄太，寺田真敏，村上純一，笠間貴弘，吉岡克成，畑田充弘：マルウェア対策のための研究用データセット〜MWS Datasets 2016〜, 情報処理学会研究報告, Vol. 2016-CSEC-74 (2016).