文章编号：1672-2892(2010)03-0364-04

一种恶意软件分析中检测虚拟环境的方法

吴发伟，方 勇，刘 亮

(四川大学 信息安全研究所，四川 成都 610065)

摘 要：安全厂商普遍使用虚拟环境来分析恶意软件，但是很多恶意软件都使用了检测虚拟机的技术来对抗对其的分析。文章介绍了3种主要的检测虚拟环境方法，给出了相应的对抗措施来防止对虚拟环境的检测。设计了一种新的基于性能比较的检查虚拟机和模拟器的方法，实验结果表明，该方法能够有效地检测出虚拟机和模拟器，如VMware软件和模拟器Qemu。

关键词：木马分析；虚拟机；模拟器；虚拟环境

中图分类号：TN915.08；TP309 **文献标识码**：A

A method to detect the presence of virtual environment in the analysis of malware

WU Fa-wei，FANG Yong，LIU Liang

(Institute of Information Security，Sichuan University，Sichuan Chengdu 610065，China)

Abstract：Security Companies usually apply virtual environment to analyze malware，whereas a large amount of current malware already adopts various VMware detection techniques in order to resist analysis. In this paper，three main methods for detecting the presence of virtual environment are presented，as well as their countermeasures. A performance related method to detect the presence of virtual machine or emulator is designed，which can successfully detect the presence of virtual environment，such as VMware and Qemu，etc.

Key words：malware analysis；virtual machine monitor；emulator；virtual environment

Internet上存在着许多恶意软件(如Malware,Malicious Software等)，对恶意软件的分析方法通常分为静态分析和动态分析。在虚拟环境下进行动态分析是分析恶意软件时普遍的选择。虚拟环境包括虚拟机(Virtual Machines Monitors，如VMware,Virtual PC等)和模拟器(Emulator，如Qemu,Bochs等)。用虚拟环境分析恶意软件的好处是不用担心恶意软件执行后对计算机造成破坏，还可以很方便地恢复到已知的干净状态。许多恶意软件分析工具(如文献[1]中使用的工具，以及文献[2]中的TTAnalyze)都是基于虚拟环境来分析恶意软件的。文献[1]所给出的方法能够检测出使用了Anti-VMware技术的恶意软件，文献[1]使用了基于Qemu的模拟器来分析恶意软件。恶意软件采取了许多措施来对抗分析，常见的措施就是检测是否正运行于虚拟环境，如果是，则采取死循环、退出程序、重启系统等其他行为来对抗分析。本文分析了3种主要检测虚拟机的方法及相应的对抗措施。已有检测虚拟环境的方法都存在着相应的对抗措施，已经不能有效地用于检测虚拟环境，本文提出了一种基于性能比较的检测虚拟机和模拟器的方法，能够有效地检查出虚拟机和模拟器。

1 虚拟机和模拟器的概念及检测技术

模拟器与虚拟机不同，模拟器模拟所有指令的执行，而虚拟机的一部分指令可直接在硬件上执行。已有的虚拟机检测技术可分为：1) 基于介质访问控制(Media Access Control，MAC)地址的检测方法；2) 基于描述符表(Descriptor Table)的检查方法；3) 基于虚拟机后门指令的检测方法。

1.1 基于MAC地址和注册表的检测

基于MAC地址检测虚拟机的原理是：虚拟机MAC地址都是有规律的[3]。通过获取计算机的MAC地址可以检测是否正在虚拟机中运行。

收稿日期：2009-11-16；修回日期：2009-12-28

1.2 基于描述符表的检测

汇编指令 SIDT 和 SGDT 以及 SLDT 可以得到 Interrupt Descriptor Table Register,Global Descriptor Table Register, Local Descriptor Table Register 的内容，在虚拟机里面和真实主机里面这些值的地址范围是不一样的。“Scoopy doo[4]使用 SIDT,SGDT 和 SLDT 得到相应寄存器的值，然后与 VMware 定义的值进行比较，判断是否正运行于虚拟机”。RedPill[5]使用 SIDT 来保存 IDTR 的值，然后与一个阈值进行比较，大于阈值就认为是在虚拟机中运行。类似的，文献[6]使用了 LDT 来检测虚拟机是否存在。

1.3 基于虚拟机后门指令的检测方法

目前使用较多的检测虚拟机的方法是使用 VMware 后门指令。文献[4]和文献[7]都使用了 VMware 后门指令来检测 VMware 是否存在。后门指令检测 VMware 的关键代码如下：

```
mov eax, 'VMXh'
mov ebx, 0
mov ecx, 10          // get VMware version
mov edx, 'VX'        // port number
in eax, dx           // read port
```

使用上述关键代码，程序就可以检测出是否在虚拟机中运行。

2 对抗虚拟机检测的措施

2.1 对抗基于 MAC 地址和注册表检测的措施

通过获取计算机的 MAC 地址可以检测是否正在虚拟机中运行。修改虚拟机的 MAC 地址可以对抗基于 MAC 地址检测虚拟机的方法。

2.2 对抗基于描述符表检测和基于虚拟机后门指令检测的措施

文献[8]中的“REFORM”是一个 C++编写的交互式反汇编器(Interactive Disassembler, IDA)插件，在使用 IDA Debugger 时“REFORM”能够使程序自动检测是否存在对虚拟机的检测。“REFORM”能够扫描恶意软件的内存，如果发现程序有使用 VMware 后门指令或者类似于 SIDT 等描述符指令，就在相应的地方设置硬件断点。一旦硬件断点被触发，则自动对相应的寄存器和内存地址打补丁。例如，在执行 VMware 后门指令之后，ebx 将会从 magic value 变为等于 0。通过这种方式，动态地对抗那些基于描述符表和 VMware 后门指令的措施。即使使用了基于虚拟机技术的加壳软件(如 Themida 和 VMProtect 等)对恶意软件加壳之后，也不能绕过这种动态的检测方法，因为程序要执行，总是要在执行前对相应代码解密。

2.3 相关工作

通过修改 VMware.vmx 配置文件来禁止获取 VMware 版本的后门指令，Carpenter[9]给出了一系列的配置选项来防止对虚拟机的检测。

3 基于性能比较的虚拟环境检测技术研究

3.1 检测原理

本文提出一种基于性能比较的检测虚拟环境的方法，经过测试，该方法不仅能够检测出 VMware，而且能检测出基于 Qemu 的环境。

在虚拟环境中运行程序的效率没有在真实硬件上高，执行速度要比在真实环境中慢。因为虚拟机模拟硬件，中间有翻译或拦截，增加了流程，也增加了执行的时间[10]。这就为笔者提供了一种基于执行时间差异来检测虚拟环境的思路，但是使用绝对执行时间并不能够检测出是否正运行于虚拟机中。不同的硬件甚至不同的操作系统和配置都会导致绝对执行时间的变化。在真实主机中执行两条指令 A 和 B，记录它们执行时间之比 T_1，在虚拟机中也执行同样的指令 A 与 B，记录它们执行时间之比 T_2，如果 T_1 和 T_2 有显著的差异，那么就可以检测出是否运行于虚拟环境之下。因此，在基于性能比较的方法中，选择用于比较的指令就显得特别重要，关系到比较是否能够成功，可以选取 CR0 和 CR3 与 NOP 指令来做比较[11]。读写 CR0 和 CR3 都是特权指令，而 NOP 是一般指令，因此，这里实际上是用读写特权指令的时间与一般指令的时间作比较，如果执行的相对时间在模拟环境和真

实环境差异明显，则可以检测出当前执行的环境是否是模拟环境。CR0 与 CR3 是控制寄存器，CR0 可用于控制中断，CR3 是页目录寄存器，保存页目录的地址。在比较时要使用同样多的指令，使用 RDTSC 得到时间，读写 CR0 的示例代码如下(读写 CR3 的代码类似)：

```
CLI
RDTSC
MOV starttime,EAX
MOV EAX, CR0
MOV CR0, EAX
MOV EAX, CR0
MOV CR0, EAX
RDTSC
MOV endtime,EAX
STI
```

在虚拟环境执行与在真实主机运行时间会有较大差异，其原因在于真实主机在处理特权指令时比虚拟环境要做更多的事情，处理更多的任务。真实主机和虚拟机实现的不同导致了执行时间会有差异。设 T_0,T_3,T_n 分别表示使用上述代码操作 CR0,CR3,NOP 的时间，R_0 表示 T_0 与 T_n 的比值，R_3 表示 T_3 与 T_n 的比值。由上面的分析，可以得出一个合理的假定：1) 在虚拟机中 R_0 逼近 1，因为在虚拟机中，系统处理特权指令所做的事情比真实主机少得多；2) 在真实主机上，R_0 与 R_3 的比值逼近 1。

3.2 实验

实验中的数据采集方式如下：首先执行 1 000 次，得到平均值，然后再对这 1 000 个值取平均。Real 代表真实主机，VM 代表虚拟环境，Qemu 代表 Qemu 模拟的环境。注意，因为实验条件限制，只在 Vmware Workstation 下测试，并未对 Vmware esx 进行测试。

表1 AMD X2 240@2.81 GHz, 1 G RAM
Table1 AMD X2 240@2.81 GHz, 1 G RAM

Real/VM	T_0	T_3	T_n	R_0	R_3	R_3/R_0
Real	179	241	9	19.89	26.78	1.35
VMware	100	15 794	82	1.22	192.61	157.88

表2 Intel E5200@2.5 GHz, 2 G RAM
Table2 Intel Pentium E5200@2.5 GHz, 2 G RAM

Real/VM	T_0	T_3	T_n	R_0	R_3	R_3/R_0
Real	377	376	42	11.42	11.39	1.00
VMware	332	7 264	284	1.17	25.59	21.87

表3 Intel(R) T5800@2.00 GHz 2.00 RAM
Table3 Intel(R) T5800@2.00 GHz 2.00 RAM

Real/VM	T_0	T_3	T_n	R_0	R_3	R_3/R_0
Real	420	960	66	6.36	15.55	2.44
VMware	186	11 333	160	1.16	70.83	61.06

表4 AMD X2 240@2.81 GHz, 2 G RAM
Table4 AMD X2 240@2.81 GHz, 2 G RAM

Real/VM	T_0	T_3	T_n	R_0	R_3	R_3/R_0
Real	224	536	63	3.56	8.51	2.39
VMware	119	10 978	105	1.13	104.55	92.52

表 1~表 5 的每轮测试，都执行了 1 000 × 1 000 次，实验结果表明：在 VMware 中读写 CR0 要比在真实主机中快(这一点不适用于 Qemu)；在 VMware 和 Qemu 中读取 CR3 比在真实主机中慢得多，甚至有 10 倍以上的差距，而执行 NOP 指令，真实主机要比在 VMware 和 Qemu 中快一点。经过实验表明：在虚拟机中读写 CR3 的 T_3，远远大于真实主机中读写 CR3 的 T_3；在 Vmware 和 Qemu 中，R_3 与 R_0 的比值最小为 13.33，最大为 157.88，而在真实主机上，R_3 与 R_0 的比值最小为 1.00，最大为 2.44。在虚拟机中 R_3 与 R_0 的比值与真实环境相比变化很明显。

表5 AMD X2 240@2.81 GHz, 2 G RAM
Table5 AMD X2 240@2.81 GHz, 2 G RAM

Real/VM	T_0	T_3	T_n	R_0	R_3	R_3/R_0
Real	224	536	63	3.56	8.51	2.39
VMware	453	6 036	77	5.88	78.39	13.33

根据实验结果，可以提出以下用来检测虚拟机的方法：

1) 设置一个阈值 Threshold_T3，如果读取 CR3 的时间 T_3 大于 Threshold_T3，则可以判定运行在虚拟环境中，否则判定运行在真实主机中。本实验表明，可以选取 2 000 到 5 000 作为一个阈值，从表 1~表 5 中可以看出，在虚拟机中的 T_3 都大于 3 000，本文选择 3 000。

2) 设置一个阈值 Threshold_R30，如果读写 CR3 和 CR0 的比值 R_3/R_0 大于 Threshold_R30，则可以判定为运行在虚拟环境中，否则判定运行在真实主机中。本实验表明，选取 5 到 10 之间的值会是较好的值，本文选择 8。从表 1~表 5 可以看出，在虚拟机中运行的结果表明 CR3 与 CR0 的比值都大于 8。

因此，可以通过 T_3 以及 R_3 与 R_0 的比值来检测虚拟环境，该方法不仅可以用来检测 Vmware，而且可以用于检测 Qemu，本文的原理也同样适用于检测其他虚拟环境，如 Virtual PC,Bochs 等，能够绕过基于文献[1]和[2]的恶意软件分析工具。

4　结论

本文分析了 3 种主要针对虚拟机的检测方式，对每种检测方法，分析了针对它们的对抗措施，用来防止对虚拟机的检测。这 3 种方法都存在着相应的对抗措施，已经不能有效地检测出虚拟环境。本文提出了一种基于性能差异来检测虚拟环境的方法，实验结果表明，通过比较不同指令的运行时间差异，能够判定当前是否运行于虚拟机中。该方法能够有效地检测出当前是否运行于虚拟环境中，可以对抗基于虚拟环境(如 VMware 和 Qemu)的恶意软件分析。本文的原理也适用于检测其他虚拟环境，如 Virtual PC 和 Bochs 等。

参考文献:

[1] David Yu Zhu,Erika Chin. Detection of VM-Aware Malware[EB/OL]. (2007-12-11)[2009-10-20]. http://radlab.cs.berkeley.edu/w/uploads/3/3d/Detecting_VM_Aware_Malware.pdf.

[2] Ulrich Bayer. TTAnalyze:A Tool for Analyzing Malware[D]. Vienna:Information Systems Institute and at the Institute of Computer Aided Automation Technical University of Vienna, 2005.

[3] Zknk Den. Detecting Vmwares Remotely[EB/OL]. [2009-10-20]. http://www.secniche.org/papers/Detecting_Vmwares_Remotely.pdf.

[4] Tobias Klein. Scoopy doo Vmware fingerprint suite[EB/OL]. (2003)[2009-10-20]. http://www.trapkit.de/research/vmm/scoopydoo/index.html.

[5] Rutkowska Joanna. Red Pill…or how to detect VMM using (almost) one CPU instruction[EB/OL]. (2004-11)[2009-10-20]. http://invisiblethings.org/papers/redpill.html.

[6] Danny Quist,Val Smith. Detecting the presence of virtual machines using the local data Table[EB/OL]. (2005)[2009-10-20]. http://www.offensivecomputing.net/dc14/vm.pdf.

[7] Elias Aka Lallous. Detect if your program is running inside a virtual machine[EB/OL]. (2005-04-04)[2009-10-20]. http://http://www.codeproject.com/KB/system/VmDetect.aspx.

[8] Li Sun,Tim Ebringer,Serdar Boztas. An automatic anti-anti-VMware technique applicable for multi-stage packed malware[C]// 3rd International Conference on Malicious and Unwanted Software(Malware'08). Washington,DC,USA:IEEE Computer Society, 2008:17-23.

[9] Matthew Carpenter,Tom Liston,Ed Sloudis. Hiding virtualization from attackers and malware[J]. IEEE Security & Privacy, 2007,5(3):62-65.

[10] Popek G J,Goldberg R P. Formal requirements for virtualizable third generation Architectures[J]. Communications of the ACM, 1974,17(7):412-421.

[11] Thomas Raffetseder,Christopher Kruegel,Engin Kirda. Decting System Emulators[EB/OL]. [2009-10-20]. http://www.cs.ucsb.edu/~chris/research/doc/isc07_detection.pdf.

作者简介:

吴发伟(1985-)，男，四川省德阳市人，在读硕士研究生，主要研究方向为网络系统与信息安全.email:ted.wu@qq.com.

方　勇(1966-)，男，四川省盐源县人，教授，博士，主要研究方向为网络系统与信息安全.

刘　亮(1982-)，男，四川省叙永县人，硕士，主要研究方向为网络系统与信息安全.