2016 3rd International Conference on Information Science and Control Engineering

An APT Trojans Detection Method for Cloud Computing Based on Memory Analysis and FCM

Liang Ge[1,a], Lianhai Wang[1,b], Lijuan Xu[1,c]

[1]Shandong Provincial Key Laboratory of Computer Networks,Shandong Computer Science Center(National Supercomputer Center in Jinan), Jinan, China 250014

e-mail: [a]gel@sdas.org, [b]wanglh@sdas.org, [c]xulj@sdas.org

***Abstract*—With memory information as characteristic, a classified method to detect APT（Advanced Persistent Threat） Trojans in cloud computing is proposed in this paper. Memory analysis and fuzzy C-means (FCM) algorithm based on the optimized initial cluster centers detects the similarity of the APT Trojans. Without influence normal operation of virtual machine in cloud, the classifier can determine whether it is malware or not. The method can overcome the shortage of feature scanning technology which could not recognize unknown Trojans, and could significantly improve the detection speed since it does not need to unpack, decrypt, and other complex operations. Experiment results show that the detection method has good accuracy, so there is a certain practical value.**

Keywords- cloud computing; APT Trojans; memory analysis; FCM

I. INTRODUCTION

In recent years, cloud computing based on the development of virtualization technology and high-speed network is regarded as the important revolution of the development of the Internet era in the future. However, with the landing of the cloud computing, the security issues faced by people in the internet become more and more serious. Hackers could carry out professional APT(Advanced Persistent Threat） attacks for cloud computing in order to get the core confidential data. APT attack is a computer crime which is the information theft or destruction for country or major commercial interests. And they have caused real damage for cloud computing, such as the attack for Sony. APT Trojans are the important tools for these crimes. They could hide and spread themselves in cloud computing.

Generally, Trojans for APT attacks cannot spread widely, which means that they are still unknown for antivirus and intrusion detection system (IDS) products in cloud computing [1]. Using Zero-Day vulnerabilities or valid digital signatures, Trojans can easily escape from the detecting of security software [2]. The IDS has the capability of automatically detecting abnormal traffic in a specified network, but for some special Trojans, it is almost incapable. And APT Trojans usually have the mechanism to check the working of security software and evade the detection of security software. Furthermore, APT Trojans usually use encryption, DLL hiding, http disguising techniques. So it is difficult for existing methods to detect unknown APT Trojans in cloud computing.

Memory forensics analysis is the forensic analysis of computer's memory dump. Its primary application is investigation of advanced computer attacks, which are stealthy enough to avoid leaving data on the computer's hard driver. Many researchers have developed techniques for extracting invalid information [3-4], such as registry command lines [5] and registry keys and values [6-8]. And with the development of memory forensics, it is possible to use this technique to detect Trojans [9].

In this paper we give a new APT Trojans detection method for cloud computing based on the memory analysis and fuzzy C-means (FCM). This method is following: firstly acquire the memory of cloud computing, then obtain the virtual machine memory by memory analysis; secondly analyze the virtual memory and select the important information of the running program as the feature, then cluster information through data mining and obtain the classifier for unknown Trojans; finally use the classifier to detect the unknown program to determine whether it is malware.

The rest of this paper is organized as follows. Section 2 presents our APT Trojans detection method in cloud computing based on the memory analysis and FCM. Section 3 shows empirical results of our method on some datasets. Finally, Section 4 presents our conclusions.

II. DETECTION BASED ON MEMORY ANALYSIS AND FCM

At present, VMware, XEN and KVM are the more popular virtual machine platform. Based on the platform all kinds of cloud environment can be built. The key of the APT Trojans detection in cloud computing is the Trojans detection of virtual machine in cloud computing. Here, we take the KVM virtual machine platform as an example to introduce our Trojans detection method. This method can also be applied to other mainstream virtual machine platform under necessary adjustment. The structure of detection method is shown in Figure 1. The method is composed of three parts: memory analysis, construction of classifier, and Trojan detection. The principle is following: Memory analysis modular is used to analyze the memory image from the cloud and obtain the training set for the Trojans detection;

Construction of classifier modular is used to extract feature information from the training set and store the data in the database, and it could use the FCM algorithm to cluster analysis for the feature of the training set; the Trojan detection modular is used to classify the unknown programs and determine whether it is the malware.

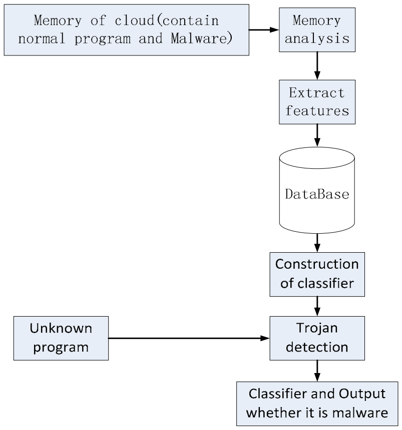

Figure 1. Structure of detection model

A. Memory analysis of KVM

The whole framework of memory analysis in a KVM environment is consisted by five phases: host machine information capture, virtual machine detection, VMCS data structure validation, virtual machine memory acquisition and virtual machine memory analysis. In [10], the first step to the fourth step has been given. Here, we mainly focus on the extraction of information in virtual machine. Taking the windows system for example, we present the processes of extracting prerequisite factors from the memory image, such as: process information, DLL loading information for each process, the API for each process, and so on.

Firstly we give the method for extracting the process information. Extraction of process information is based on the analysis method of KPCR structure. It could be divided into the following steps: 1) Acquisition of KPCR structure; 2) Acquisition the address of CR3 register; 3) Address translation; 4) search of kernel variable. After memory mapping technology, the base address of system process is obtained and the KPROCESS structure of each process is mapped. The running process can be obtained by traversing the double-linked list.

Offset(P)	Name	PID	PPID	PDB	Time created	Time exited
0x00000000281fda0	WINWORD.EXE	2260	1492	0x23df5000	2015-06-12 01:25:02 UTC+0000	
0x0000000028439e0	XLLiveUD.exe	1820	504	0x21b82000	2015-05-29 13:45:11 UTC+0000	2015-05-29 13:45:12
0x000000002881768	XLLiveUD.exe	1532	384	0x24d64000	2015-06-01 00:36:40 UTC+0000	2015-06-01 00:36:41
0x000000002887020	jucheck.exe	3072	204	0x21028000	2015-05-29 13:46:22 UTC+0000	
0x000000002897338	IEXPLORE.EXE	2032	332	0x23d5d000	2015-06-04 09:44:20 UTC+0000	2015-06-10 07:37:46
0x00000000289e630	SGTool.exe	4020	2260	0x2eb34000	2015-06-12 01:30:12 UTC+0000	
0x0000000028d7cd0	XLLiveUD.exe	2264	384	0x14a16000	2015-06-10 07:38:02 UTC+0000	2015-06-10 07:38:09
0x0000000028d9020	se_wdtc.exe	3876	196	0x0c148000	2015-05-29 13:43:34 UTC+0000	2015-05-29 13:43:38
0x00000000290c020	ThunderPlatform	228	504	0x08a2c000	2015-06-01 00:34:03 UTC+0000	
0x000000002919b98	svchost.exe	2184	724	0x2ed6d000	2015-05-29 13:42:02 UTC+0000	
0x000000002ce2c08	services.exe	724	680	0x15f94000	2015-05-27 14:09:38 UTC+0000	
0x000000002ce3990	matlabserver.ex	1336	724	0x20705000	2015-05-27 14:10:10 UTC+0000	
0x000000002cefc70	TPAutoConnect.e	3708	2388	0x06506000	2015-05-29 13:42:09 UTC+0000	
0x000000002d20128	csrss.exe	648	580	0x14f5f000	2015-05-27 14:09:37 UTC+0000	
0x000000002d43020	SogouCloud.exe	3888	2260	0x09fdd000	2015-06-12 01:25:22 UTC+0000	
0x000000002d5f6f0	svchost.exe	1208	724	0x176ab000	2015-05-27 14:09:41 UTC+0000	
0x000000002d84758	xmp.exe	368	1744	0x24782000	2015-05-27 14:09:55 UTC+0000	
0x000000002e7d390	ZhuDongFangYu.e	1440	724	0x186f2000	2015-05-27 14:09:42 UTC+0000	
0x000000002e8f670	explorer.exe	1492	1468	0x1885e000	2015-05-27 14:09:43 UTC+0000	
0x000000002e92558	svchost.exe	1124	724	0x17554000	2015-05-27 14:09:41 UTC+0000	
0x000000002eaac08	vmacthlp.exe	904	724	0x168f7000	2015-05-27 14:09:39 UTC+0000	
0x000000002ecc8d0	jusched.exe	204	1492	0x1da84000	2015-05-27 14:09:50 UTC+0000	
0x000000002edc7d0	alg.exe	2568	724	0x04717000	2015-05-29 13:42:06 UTC+0000	
0x000000002f0e3c0	ctfmon.exe	152	1492	0x1dc0b000	2015-05-27 14:09:50 UTC+0000	
0x000000002f17978	smss.exe	580	4	0x1398f000	2015-05-27 14:09:35 UTC+0000	
0x000000002fd33b0	VMUpgradeHelper	620	724	0x298b1000	2015-05-27 14:10:27 UTC+0000	
0x00000000302e488	winlogon.exe	680	580	0x15965000	2015-05-27 14:09:37 UTC+0000	
0x000000003097c08	ThunderPreload.	1204	504	0x2a226000	2015-05-27 14:10:28 UTC+0000	2015-05-29 13:42:15
0x0000000030dcda0	spoolsv.exe	1672	724	0x1a18f000	2015-05-27 14:09:45 UTC+0000	
0x0000000030deda0	svchost.exe	936	724	0x16cc5000	2015-05-27 14:09:40 UTC+0000	
0x0000000030eb9b0	vmtoolsd.exe	212	724	0x27f9d000	2015-05-27 14:10:23 UTC+0000	
0x0000000030f18e8	KKV.exe	1904	1812	0x1c8b8000	2015-05-27 14:09:48 UTC+0000	
0x00000000317bda0	lsass.exe	736	680	0x15f9d000	2015-05-27 14:09:38 UTC+0000	
0x000000003223c10	svchost.exe	1028	724	0x1748d000	2015-05-27 14:09:40 UTC+0000	
0x000000003237020	Thunder.exe	384	1492	0x0b4bc000	2015-06-01 00:36:37 UTC+0000	
0x00000000323d848	CAJSHost.exe	1292	724	0x18e7a000	2015-05-27 14:10:10 UTC+0000	
0x000000003263020	svchost.exe	504	724	0x28023000	2015-05-27 14:10:24 UTC+0000	
0x0000000032719e0	TPAutoConnSvc.e	2388	724	0x30134000	2015-05-29 13:42:03 UTC+0000	
0x00000000327b888	conime.exe	2700	660	0x1a971000	2015-06-12 01:26:07 UTC+0000	
0x0000000032a5900	MATLAB.exe	460	936	0x2837a000	2015-05-27 14:10:24 UTC+0000	
0x0000000033b87c0	System	4	0	0x00039000		

Figure 2. Process information of system

Secondly we give the method for extracting the DLL loa ding information for process. Normally, when the LoadLibra ry function is called by the process, DLLs are automatically l oaded to the double-linked list consisting of LDR_DATA_T ABLE_ENTRY. So DLL loading information for each proce ss could be obtained by traversing the double-linked list poin ted by PEB's InLoadOrderModuleList. The steps are as follo wing:

- Find the address of EPROCESS structure for the process in the memory;
- acquire the corresponding PEB structure through EPROCESS;
- find the _PEB_LDR_DATA through the PEB;
- find the list header LIST_ENTRY loading the DLLs in the _PEB_LDR_DATA;
- obtain the loading DLL through traversing the _LDR_DATA_TABLE_ENTRY structure in LIST_ENTRY.

Base	Size	LoadCount	Path
0x7c920000	0x93000	0xffff	C:\WINDOWS\system32\ntdll.dll
0x7c800000	0x11e000	0xffff	C:\WINDOWS\system32\kernel32.dll
0x77da0000	0xa9000	0xffff	C:\WINDOWS\system32\ADVAPI32.dll
0x77e50000	0x93000	0xffff	C:\WINDOWS\system32\RPCRT4.dll
0x77fc0000	0x11000	0xffff	C:\WINDOWS\system32\Secur32.dll
0x74480000	0xaf000	0xffff	C:\WINDOWS\system32\LSASRV.dll
0x71a90000	0x12000	0xffff	C:\WINDOWS\system32\MPR.dll
0x77d10000	0x90000	0xffff	C:\WINDOWS\system32\USER32.dll
0x77ef0000	0x49000	0xffff	C:\WINDOWS\system32\GDI32.dll
0x76db0000	0x12000	0xffff	C:\WINDOWS\system32\MSASN1.dll
0x77be0000	0x58000	0xffff	C:\WINDOWS\system32\msvcrt.dll
0x5fdd0000	0x55000	0xffff	C:\WINDOWS\system32\NETAPI32.dll
0x76770000	0x13000	0xffff	C:\WINDOWS\system32\NTDSAPI.dll
0x76ef0000	0x27000	0xffff	C:\WINDOWS\system32\DNSAPI.dll
0x71a20000	0x17000	0xffff	C:\WINDOWS\system32\WS2_32.dll
0x71a10000	0x8000	0xffff	C:\WINDOWS\system32\WS2HELP.dll
0x76f30000	0x2c000	0xffff	C:\WINDOWS\system32\WLDAP32.dll
0x71b70000	0x13000	0xffff	C:\WINDOWS\system32\SAMLIB.dll
0x743a0000	0x68000	0xffff	C:\WINDOWS\system32\SAMSRV.dll
0x76760000	0xc000	0xffff	C:\WINDOWS\system32\cryptdll.dll
0x5cc30000	0x26000	0x1	C:\WINDOWS\system32\ShimEng.dll
0x58fb0000	0x1ca000	0x1	C:\WINDOWS\AppPatch\AcGenral.DLL
0x76b10000	0x2a000	0x2	C:\WINDOWS\system32\WINMM.dll
0x76990000	0x13e000	0x4	C:\WINDOWS\system32\ole32.dll
0x770f0000	0x8b000	0x2	C:\WINDOWS\system32\OLEAUT32.dll
0x77bb0000	0x15000	0x1	C:\WINDOWS\system32\MSACM32.dll
0x77bd0000	0x8000	0x1	C:\WINDOWS\system32\VERSION.dll
0x7d590000	0x7f4000	0x2	C:\WINDOWS\system32\SHELL32.dll
0x77f40000	0x76000	0x4	C:\WINDOWS\system32\SHLWAPI.dll
0x759d0000	0xaf000	0xc	C:\WINDOWS\system32\USERENV.dll
0x5adc0000	0x37000	0x3	C:\WINDOWS\system32\UxTheme.dll
0x76300000	0x1d000	0x2	C:\WINDOWS\system32\IMM32.DLL
0x62c20000	0x9000	0x1	C:\WINDOWS\system32\LPK.DLL
0x73fa0000	0x6b000	0x1	C:\WINDOWS\system32\USP10.dll
0x5d170000	0x9a000	0x1	C:\WINDOWS\system32\comctl32.dll
0x4d200000	0xe000	0x1	C:\WINDOWS\system32\msprivs.dll
0x71c70000	0x4c000	0x2	C:\WINDOWS\system32\kerberos.dll
0x77c40000	0x25000	0x3	C:\WINDOWS\system32\msv1_0.dll
0x76d30000	0x18000	0x3	C:\WINDOWS\system32\iphlpapi.dll
0x767c0000	0x29000	0x7	C:\WINDOWS\system32\schannel.dll
0x765e0000	0x95000	0x7	C:\WINDOWS\system32\CRYPT32.dll
0x742e0000	0xf000	0x1	C:\WINDOWS\system32\wdigest.dll
0x68000000	0x36000	0x1	C:\WINDOWS\system32\rsaenh.dll

Figure 3. DLL information of some process

Thirdly we give the method for extracting the API for process. The API loaded by the process can be obtained by the analysis of PE file. However the packing and encrypting technologies are used by Trojans to avoid static analysis. Since these, we use the way that obtain the assembly code of loaded process in the memory to deal with these difficulties. The steps are following: a) acquire the assembly code of process in the memory; b) analyze the code to find the address of import function table; c) obtain the API of process through the structure of table.

```
API                            Process        Address
--------------------------------------------------------
AssocQueryKeyW                 Explorer.EXE   0x010048A2
PathFindFileNameW              Explorer.EXE   0x01005D6F
AssocQueryStringW              Explorer.EXE   0x01021C87
PathParseIconLocationW         Explorer.EXE   0x010236D9
SHRegGetValueW                 Explorer.EXE   0x0100655B
StrDupW                        Explorer.EXE   0x01008C88
SHRegDuplicateHKey             Explorer.EXE   0x010094CD
AssocCreate                    Explorer.EXE   0x01009A2F
StrRChrW                       Explorer.EXE   0x01009A3B
StrToIntW                      Explorer.EXE   0x01021C87
PathUnquoteSpacesW             Explorer.EXE   0x010236D9
```

Figure 4. API information of some process

Similarly, we can also obtain the registry information and system path information of process from the memory which would not be shown here.

B. APT Trojans classifier based on FCM

First we will give the feature extracting method of APT Trajans. Similar to most normal program, Trojans also need to call the interface and API to carry on a variety of operations. But they usually call some special API which is few called by normal program to destroy. The special operations include: read or write the specific registry key, information hide, encryption of communication, modify the critical path of the system, inject the hook into the process of the system, and so on. So the detection whether these special APIs are included in the process can be regarded as the features of APT Trojans. Of course since these APIs also are called by the normal programs, it would be low accuracy that only uses APIs to detect the APT Trojans. Compared with the data, the same type of Trojan has a certain similarity in function. Although they often use the technologies of packing and encrypting to hide themselves, the process of the API call for special function is often similar. If an unknown program is similar to the information in the feature database in DLL information, registry information, API information, and system path, it may be a variant of malware.

Above all, we would extract the below information as the feature: called DLL file name and number, called special API function name and number, special register information and system path in process memory. Part of the feature vector is shown in TABLE I.

Category	Feature
DLL	Kernel32.dll, User32.dll, Shell32.dll, Advapi32.dll, GDI.dll, Imm32.dll, Ole32.dll, Msvert.dll, Comdlg32.dll, Oleaut32.dll, Setupapi.dll, Urlmoon.dll
API	Accept, AdjustTokenPrivileges, AttachThreadInput, bind, BitBlt, CallNextHookEx, CerOpenSystemStore, CheckRemoteDebuggerPresent, connect, ConnectNamePipe, ControlService, CrypyAcquireContext, EnumProcess, FindResource, GetKeyState
Key registry informatio n	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run, HKCU\Software\Microsoft\Windows\CurrentVersion\Run, HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\[Shell], HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\[AppInit_DLLs]
System path	C:\windows\system32, C:\windows\system32\drivers, C:\windows\system32\dllcache, C:\Documents and Settings\Administrator\Application Data

TABLE I. FEATURES OF SAMPLES

Then we would give the method of constructing the classifier. In this paper, define the sample set $U=\{U_1,U_2,\ldots,Un\}$, the vector of each sample is $U_i=\{x_{i1},x_{i2},\ldots, x_{im}\}(i=1,2,\ldots,n)$, where x_{im} is the m^{th} feature of sample i. We will use the FCM to classify the sample set. The FCM clustering algorithm was first introduced by Dunn [11] and later was extended by Bezdek [12]. The algorithm is an iterative clustering method that produces an optimal c partition by minimizing the weighted within group sum of squared error objective function J_{FCM}

$$J_{FCM}=\sum_{k=1}^{n}\sum_{i=1}^{c}(y_{ik})^{p}d^{2}(u_k,v_i),$$

where $y_{ik}\in[0,1]$, c is the number of clusters with $2\leqslant c\leqslant n$, y_{ik} is the degree of membership of u_k in the i^{th} cluster, p is a weighting exponent on each fuzzy membership, v_i is the prototype of the centre of cluster i, $d^2(u_k, v_i)$ is a distance measure between object u_k and cluster centre v_i. The solution of the object function J_{FCM} can be obtained via an iterative process, which is carried out as follows:

a) Set values for c, p, and ϵ;
b) Initialize the fuzzy partition matrix $Y=[Y_{ik}]$;
c) Set the loop counter b=0;
d) Calculate the c cluster centers $\{v_i^{(b)}\}$ with $Y^{(b)}$:

$$v_i^{(b)}=\frac{\sum_{k=1}^{n}(y_{ik}^{(b)})^{p}u_k}{\sum_{k=1}^{n}(y_{ik}^{(b)})^{p}};$$

e) Calculate the membership $Y^{(b+1)}$. For k=1 to n, calculate the following: $Ik=\{i|1\leqslant i\leqslant c, dik=\|uk-vi\|=0\}$; for the kth column of the matrix, compute new membership values:

if $Ik=\Phi$, then $y_{ik}^{(b+1)}=\dfrac{1}{\sum_{j=1}^{c}(\frac{d_{ik}}{d_{jk}})^{\frac{2}{(p-1)}}}$;

else $y_{ik}^{(b+1)}=0$ for all $i\notin I$ and $\sum_{i\in I_k}y_{ik}^{(b+1)}=1$; next k.

f) If $\left\| y^{(b)} - y^{(b+1)} \right\| < \varepsilon$, stop; otherwise, set b=b+1 and go to step d).

Although FCM algorithm has good clustering effect, the deficiency of this algorithm is sensitive to its initial values. In most cases, it is easy to the local optimum for some initial values. So we need to improve the FCM algorithm to avoid this case. We give the following method. The principle is to choose the initial clustering center to make the distance between the initial clustering centers more than the set threshold. Then the algorithm would solve the clustering problem in several feasible regions. It could avoid the case that the algorithm is convergence to local optimum. The choice of initial center is as following:

a) compute the distance between any two samples and generate the distance matrix D. Set the nearest two samples to one class and take the midpoint of the two samples as the first clustering center.
b) select the distance threshold a. Use the distance matrix D to find all of the samples whose distance to the first category is greater than a. Set the nearest two samples to one class and take the midpoint of the two samples as the second clustering center.
c) similarly find all the samples whose distance to the found samples is greater than a. Set the nearest two samples to one class and take the midpoint of the two samples as the clustering center.
d) repeat c) until C class is found.

The above algorithm can make the clustering center to reasonably distribute in the sample space. It can avoid the excessive concentration and effectively improve independence of the classifications. It could help the FCM to achieve the global clustering effect.

C. Detection of unknown APT Trojans

By clustering the training set, C clustering center is obtained. Let's give the definition for sample similarity. The similarity of U_i, U_j is:

$$D_{U_iU_j} = \frac{\sum_{k=1}^{m} \frac{|x_{ik} - x_{jk}|}{\max(x_{ik}, x_{jk})}}{n}$$

When the similarity of two samples is high, the value of D is near to 1; conversely it is near to 0.

The system could compute the similarity between each detected sample and the clustering center. If the similarity reaches a certain threshold, we could think it belongs to a class and we can know whether it is malware. If the sample is not match to all the clustering, it will be a new clustering center and marked as a failure identification class stored in feature base. For all the unknown class, the system could determine whether it is the Trojan combined with their behavior characteristics, and decide to remain it in the feature base or remove it. In order to maintain the timeliness of the information in the feature base and reduce time consumption, the system need to periodically delete entries in database that the number of clusters is few and no new features for a long time.

III. EXPERIMENT

We would carry on the test on the build inspur server. The server configuration is as following: CPU models: Xeon E5-2620 GHZ 2GHZ*6; Memory capacity: 8G DDR3*2; The host operating system: Fedora 16(3.6.11-4.Fc16.X86_64), KVM virtual machine (qemu-KVM-0.15.1); Install the virtual machine operating system: Windows XP SP3 32-bit, Windows 7 64-bit. There are 200 samples for experiments, which are divided into normal programs and malicious programs. The distribution is shown in TABLE II. The normal programs are selected from the new virtual machine with windows XP or windows 7 in KVM platform. The malicious programs are collected from internet, including Trojans, backdoor procedures, etc.

	Sample set	Training set	Testing set
Normal program	100	80	20
Malicious program	100	80	20
total	200	160	40

TABLE II. EXPERIMENTAL SAMPLE DATA

The main purpose of the experiment is to verity the classification and recognition of the unknown Trojans and other malicious programs in the virtual machine of the cloud platform. There are four results for the detection, including: 1) the normal program is considered as normal, recorded as NY; 2) the normal program is considered as malicious, recorded as NN; 3) the malicious program is considered as normal, recorded as MY; 4) the malicious program is considered as malicious, recorded as MN. The variable H1 is the probability of correct classification for test sample; the variable H2 is the probability that the normal program is considered as malicious. The formula is as following:

$$H_1 = \frac{NY + MN}{NY + NN + MY + MN}$$

$$H_2 = \frac{NN}{NY + NN + MY + MN}$$

The results of the experiment are as following:

category	number	H1	H2
Normal program	20	92.9%	3.7%
Malicious program	20	90.1%	4.7%

TABLE III. RESULT OF CLASSIFIER DETECTION

In TABLE III, we can see that the detection method in this paper has high accuracy and low false rate. So it can have good effect in practice.

IV. CONCLUSION

In this paper, a new detection method for APT Trojans in cloud is present. The method has high accuracy and low false rate for Trojans detection. It could obtain the program information by analyzing the memory of the cloud platform and use the improve FCM algorithm to detect and identify the APT Trojans in cloud. The memory analysis method could extract the DLL information, API information, registry information, system path of system without affecting the virtual machine users. It could effectively deal with the difficulties that packing and encrypting technology challenge to detect. The improved FCM algorithm instead of the traditional feature code are used to classify and detect for the program, which could effectively find unknown Trojan. The method could be used to detect the implanted APT Trojans and reduce their damage and unnecessary losses. It would be a good solution for cloud security.

ACKNOWLEDGMENT

The work was supported by the National Natural Science Foundation of China (61572297), by the Shandong Province Outstanding Young Scientists Research Award Fund Project(Grant No. BS2013DX010), by the Natural Science Foundation of Shandong Province, China(Grant No. ZR2014FM003, ZR2013FQ001, ZR2013FM025, ZR2015YL018), and by the Shandong Academy of Sciences Youth Fund Project(Grant No. 2013QN007) .

REFERENCES

[1] Y. Liang, G. Peng, etc, “An unknown Trojan detection method based on software network behavior,” Wuhan university Journal of Natural Sciences, Vol 18, 2013, pp.369-376.

[2] A. Matrosov, E. Rodionov, D. Harley, etc, “Stuxnet under the micros cope,” http://www.eset.com/us/resources/white-papers/stuxnet_under_the_mciroscope.pdf

[3] S. Hejazi, C. Talhi, M. Debbabi, “Extraction of forensically sensitive information from windows physical memory,” Digital Investigation, vol 6, 2009, pp. S121-S131.

[4] O James, L. Gilbert, “Windows operating systems agnostic memory analysis,” Digital Investigation, vol 7, 2010, pp. S48-S56.

[5] M. Richard, C. Eoghan, “Extracting Windows command line details from physical memory,” Digital Investigation, vol 7, 2010, pp. S57-S63.

[6] M. Vivienne, T. Theodore, S. Iain, “The Windows Registry as a forensic artefact: Illustrating evidence collection for Internet usage,” Digital Investigation , vol3, 2006, pp. I66-I73.

[7] D. Brendan, ” Forensic analysis of the Windows registry in memory,” Digital Investigation, vol 5, 2008, pp. S26-S32.

[8] S. Zhang, L. Wang, L. Zhang, “Extracting windows registry information from physical memory, Computer Research and Development,” ICCRD, 2011, pp. 85-89.

[9] M. Zhao, L. Wang, ”Research on Trojan detction method of computer memory mirroring,” Applied Mechanics and Materials, 2014, 701-702, pp. 1013-1017.

[10] S. Zhang, L. Wang, X. Han, “A KVM virtual machine memory forensics mehtod based on VMCS,” CIS2014, pp657-661.

[11] J. Dunn, “A fuzzy relative of the ISODATA process and its use in detecting compact well separated clusters,” Journal of Cybemetics, Vol 3, 1974, 32-57.

[12] J. Bezdek, L. Hall, L. Clarke, “Review of MR image segmentation techniques using pattern recognition,” Med. Phys., vol 20, 1993, pp. 1033-1048.