APPBACS: AN APPLICATION BEHAVIOR ANALYSIS AND CLASSIFICATION SYSTEM

Himanshu Pareek[1], P R L Eswari[1] and Dr. Sarat Chandra Babu[2]

[1]Center for Development of Advanced Computing, Hyderabad
himanshupareek@gmail.com, prleswari@cdac.in
[2]Center for Development of Advanced Computing, Bangalore
sarat@cdac.in

ABSTRACT

Number and complicacy of malware attack has increased multiple folds in recent times. Informed Internet users generally keep their computer protected but get confused when it comes to execute the untrusted applications. In such cases users may fall prey to malicious applications. There are malware behavior analyzers available but leave report analysis to the user. Common users are not trained to understand and analyze these reports, and generally expect direct recommendation whether to execute this application on their computer. This research paper tries to analyze behavior and help the common users and analysts to quickly classify an application as safe or malicious.

KEYWORDS

Application Analysis, Trojan, Automated Malware Analysis

1. INTRODUCTION

Malware authors often package their Trojans as useful looking applications. They use various methods like infecting good websites, spam and search engine optimization to lure users to download their rogue applications. Quick heal Technologies in its latest report of January 2013 explains that malware (Trojan being most occurring) is the largest threat to PCs [1]. Though users install firewall and antivirus software on their computer and keep their Operating System and application software patched with latest updates, situations arise where users want a software tool for their work and they get it from Internet. They install the software which happened to be a Trojan and leaves the computer in a compromised state. In such cases, a behaviour analysis will be very useful to quickly have an indication about software's capabilities [2]. Malware analysts also come across many unknown applications and they do manual and automated analysis to find out whether the software is malicious or benign. Though there are tools which do automated analysis of a program [3][4] (like Zero Wine [5]) to capture the behaviour of an application, APPBACS does the analysis using a set of pre-defined rules of these behaviour patterns and gives verdict whether the application under consideration is malicious.

2. RELATED WORK

CWsanbox [6] presents the approach towards automating malware analysis and compares its report with other systems like TTAnalyze [7] and Norman Sandbox [8]. These tools provide a nicely formatted report of malware behavior and none does either analysis of collected behavior

DOI : 10.5121/ijcsit.2013.5205

patterns or present a score. Moreover, Arne Vidstrom [9] at ntsecurity.nu shows that emulators can be deceived by malware and nothing matches analysis on a real computer. Mandiant Red Curtain [10] checks certain static parameters of an executable like presence of packer signatures, entropy and PE anomalies. It then generates a score and if score is more than a threshold value then binary may be malicious. Moreover, Scott Treadwell demonstrates the use of scoring via weighted means to detect packed executables [11]. Konrad Rieck's MalHeur is most close to our work. MalHeur is primarily for clustering the malware binaries according to their families but also capable of clustering benign binaries too. [12].

3. OVERVIEW OF OUR APPROACH

APPBACS is implemented for Windows platform and evaluated on Windows XP SP3, Windows Vista and Windows 7. APPBACS provides the feature of deleting all the changes on system which gives freedom to professional analyst to quickly analyze untrusted application. The tool works in three steps which are explained below.

In **first** step, user needs to run APPBACS tool with path of un-trusted application as a command line. This starts the application and APPBACS captures the behavior of application. Such behavior traces can also be collected from other tools but then output should be in APPBACS understandable text format. But advantage with APPBACS is that all changes are undone after analysis is completed.

In **second** step, APPBACS takes these behaviors patterns and matches against pre defined rules. Data which is collected using APPBACS and pre-defined rules are explained in next subsection. For every detected pattern, a severity number is assigned.

In **third** step, APPBACS calculates the Power distance score [13] using these severity numbers and based on this score we classify the application under three categories: {Safe, Potentially Malicious, and Malicious}. User can restrain himself to run a malicious application and should not run potentially malicious application with administrator's privilege. These three steps are shown in figure 1:

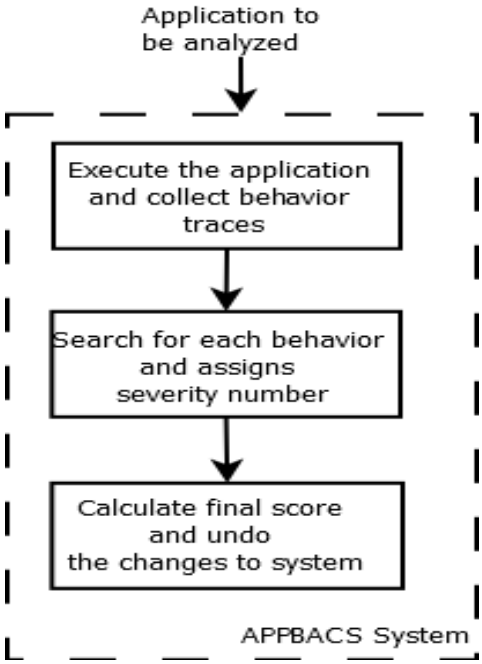

Figure 1. High Level Process of APPBACS

4. DATASETS

We did two honey-pot setups using Dionaea tool [14] at beam telecom (residential) and BSNL broadband in our lab. Dionaea tool emulates vulnerabilities and invites attacks and logs the malware. We also collected malware from other sharing websites like offensivecomputing.net, contagiodump.blogspot.com, malware.lu and virusshare.com. We decided to keep all these malwares in training set and keep on collecting new malwares from these sources and keep them in test set. We kept 2500 malware in the test set.

5. METHODOLOGY

We analyzed all the malware in training set manually and formulated a set of rules which can affect the usability of computer.

Following nominations are used onwards to explain the analysis process.

B_i = A behavior exhibited is represented as,
V = Vector of all pre-defined behavior rules where each element **v = {B_i, C_i}**
C_i = Category to which a behavior belongs. There are 10 unique categories listed below.
Z = Vector of Categories along with their Severity number and Power Quotient which controls the weight on **S**, where each element **z = {C_i, S, p}**
If behavior exhibited is Bi then it is searched in V to find which category C_i it belongs. Then C_i is searched in Z to find the severity of the exhibited behavior.
A short list of behaviour rules (B_i) is shown below.

1. Editing hosts file (Add DNS Mappings)
2. Edit services file (Adds services (port number) to TCPIP Services)
3. Edit LMHOSTS (Add Name Resolution mappings)

4. Edit dssec.dat (may change the Active Directory Settings)
5. Read/Edit SAM (Windows Password files)
6. Enumeration of processes
7. Writing other Process Memory
8. Reading other process memory
9. Install hidden services in svchost.exe
10. Adding URL Search hooks in IE
11. Antivirus notification settings are changed

Apart from such kind of rule matching, APPBACS also takes in consideration following things to analyze effectively.

1. Hash of the original application and created files (classification as whether executable or other)
2. DNS Queries made by the application
3. Network Connections
4. All the processes created by original program are analyzed in similar manner recursively

Using hashes of original and newly created files, APPBACS determines whether the application deletes or copies itself which indicates that application's intentions are not good. For logging DNS queries made by the application, we install fakeDNS [15] which sets the DNS to local host and INETSIM [16] simulates all the services to log all requests. We also run a thread to capture all the packets flown during runtime of application. This is implemented using PCAP Library [17]. APPBACS then analyzes the PCAP dump to search for NETBIOS queries made by the application [18].

Besides behaviour analysis, we make use of two static parameters also. APPBACS scans executable against well known packer signatures [19] to detect if it is packed using a packer program. APPBACS also scan executable against file magic signatures to detect whether it is an installer package [20]. If executable is packed using a known packer then chances of getting classified as malware increases and hence we put this feature in rule categories with a severity of 5. However if executable is packaged installer then, chances of getting classified as benign application increases. We keep this feature in rule of categories with a severity number of 1/3. We multiply the final score with this if executable is an installer package.

Following is a list of categories along with severity value and power for calculating power distance score used by APPBACS.

Table 1. Severity Number and Weights assigned to each behavior

Behaviour Detected	**Severity**	**Weight**
AvoidableFileSystemActivity	2	count
AvoidableRegistryActivity	2	count
NonAcceptableFileSystemActivity	7	count
NonAcceptableRegistryActivity	7	count
DeleteSelf	7	2
CopySelf	7	2
AbnormalURL	5	2
AbnormalRequestToURL	5	2
ListenActivity	5	2
EnumerateProcesses	3	1
ReadProcessMemory	3	1
WriteProcessMemory	5	2
PackedExecutable	5	2

For behaviour categories 5-13 **power** is kept a fixed value of 1 or 2 based on their importance. For behaviour categories 1-4 **power** is dynamically calculated **count**. The variable **count** is number of behaviours depicted from the category. For determining an abnormal URL, n-gram approach is used [21]. After determining number of behaviours depicted by application and their corresponding severity numbers, we calculate Power Distance Score as per following formula.

$$PD = \sqrt{\sum_{i=0}^{i=N} S_i^{\,p}}$$

Where N is total number of behaviours depicted by the application and Si is a severity number of a particular behaviour. If Power Distance (PD) score falls below 1, the application is classified as safe. A pictorial view is shown in figure 2.

Figure 2. Power Distance Risk Indicator

PD values from 1 to 4 are considered potentially unsafe and user should not run with administrator privileges. Values more than 4 are considered malicious. These threshold values are determined by analyzing many applications using tool and calculating PD. Severity numbers are based on expert opinion. For example **DeleteSelf** is given high priority which indicates high probability of application being malicious. However some readers can argue to assign a higher severity number or power quotient to behavior category **AbnormalURL**.

6. IMPLEMENTATION

Though major contribution of this research is the analysis of behavior patterns and giving a score to an application to determine whether it is benign or malicious, implementation of such a software tool will be appreciated when this does behavior capture by itself. We capture the behavior file system activity using mini filter driver [22] and registry callbacks [23]. Both are well documented developer interfaces in Windows OS. APPBACS also register a callback function using DDI PsSetCreateProcessNotifyRoutine which notifies the process creation and process deletion events [24]. We use detour library [25] [26] to hook a few Win32 API and monitor some of the above mentioned behaviors in Table 1. Detour library is well documented interface from Microsoft. Following Table 2 shows function hooked using detour library and behavior monitored.

Table 2. Win32 Functions hooked using Detour Library

Win32 API Hooked	**Behaviour Monitored**
EnumerateProcesses	Whether application enumerates all running processes in system
WriteProcessMemory	Whether application writes to other process' memory
ReadProcessMemory	Whether application reads other process' memory

Design of APPBACS and its capturing system is depicted in Figure 3.

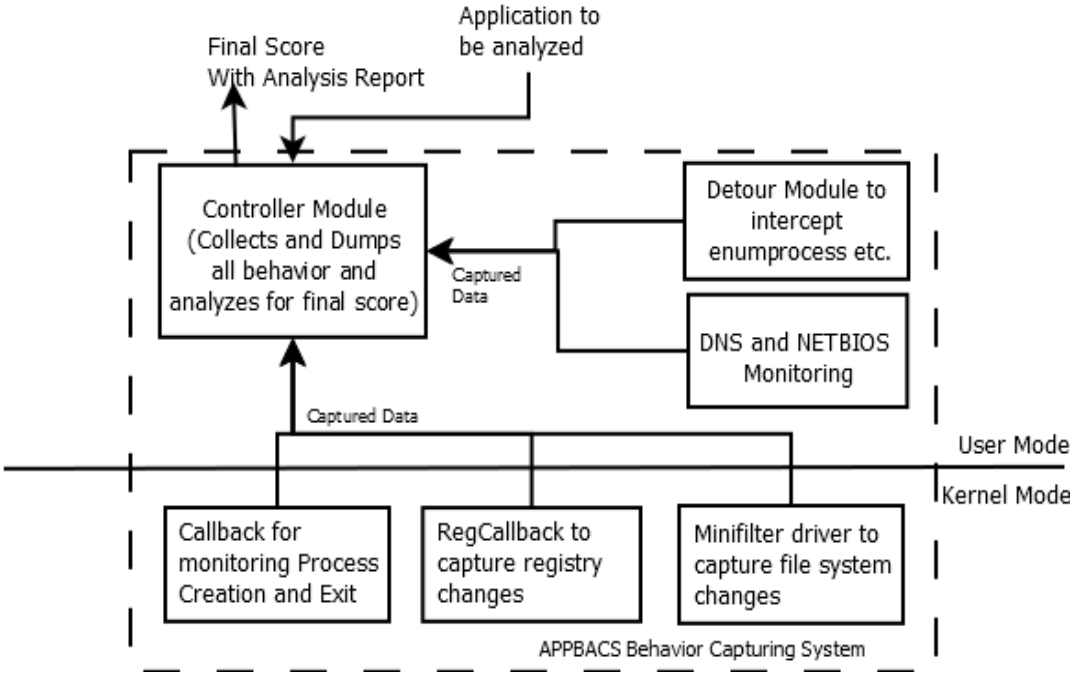

Figure 3. APPBACS Behaviour Capture System

Controller Module picks the application file path from command line argument and starts the process in suspended mode. It then sends PID to registry callback driver (which monitor registry operations) and minifilter driver (which monitor file system operation). Controller Module starts all the monitoring modules, resumes the process and then starts collecting the data from all monitoring modules. After all processes have exited (or they are terminated after maximum one minute) Controller Module does the analysis of collected behaviour patterns.

7. EVALUATION

We evaluated APPBACS on collected malware data set. Results were obtained which shows that tool is quite capable of finding out if an application is malware or benign. The success rate of this tool on malware applications is 99.56% while false positive rate is 1%. This is shown in table 3. We tested on many bots, fake AV, Trojans etc. and their score was never less than 7. Table 4 lists scores of some well known malware. Malware **PWS: Win32/Coced.2_25** was classified as potentially malicious as it is used to only generate cracks. Popular **DbgView** Application scored only 1.732 and installer of Free Download Manager (Lite) version scored 1.41.

Table 3. True Positive and False Positive Results

	Number	Classified Correctly	Classified Wrong
Malware Applications	2500	2489	11
Benign Applications	2600	2564	26

Table 4: PD Scores of some well know malware

Malware Name	MD5 Hash	PD Score
Backdoor:Win32/Sdbot	0783505871f4d862b78d4a709827d42d	20.5183
Trojan:Win32/Tibs.gen!A	90cf84122ac774166ac1ee54bf6801cd	16807
Backdoor.Win32.Rbot.gen	e732b2745e87b07c97107976e8434062	907.51
Worm:Win32/Spybot	bcad3a938531825c022b62e51b1daa72	129.996
PWS:Win32/Coced.2_25	d790d8c5ccc9c0c0caf1a4026807311e	1.73205
Backdoor:Win32/Gaobot	bed69cde9d96440c31704627cb556e9c	22.0227
Worm:Win32/Gaobot	210979b918a10fc44e57381dffb501b1	9.16515
Backdoor:Win32/Gaobot.AR	101e54b61e9c1e515a4a952ec4f6ddd5	7.1414
Worm:Win32/Gaobot	c8ae98259a3d11aaaad08bffd40d6228	9.16515
Worm:Win32/Gaobot	c54c286ec9ca3b01e498b7d4b570afaf	10.3923
Backdoor:IRC/Evilbot	54faf63f7833cfad9c1422087e9f767e	7.81
Backdoor:Win32/Gobot.A	ce04d8a9ff9295e9dec176bcf4fe0ebc	49
Backdoor:Win32/Msbot.C	8be95e4a8c3ca421aa1fa8d8592e7b82	5.65685
Win32/Oderoor.gen!A	8d96002e1374351936da764f65708740	8.27
Virus:Win32/Parite.B	b9ab75ba5ca97017d81db76c9022386b	49.9099
PWS:Win32/Abot	8163647cc12d36e1d95b0ef3096f876c	18.8149
Trojan:Win32/Skyboot	ae3abdfbc67945fc5b62dfb9e508149b	5
Backdoor:Win32/Rbot	03da32043abfbe9f454ce0d65090e5b7	129.958
Backdoor:Win32/Sdbot.0_4	d04fc8f3f6cd01aa78e2a28d8066b73d	49.295

Figure 4 shows the comparison of PD scores of malicious and benign applications. Malicious applications score considerably high as compared to benign applications.

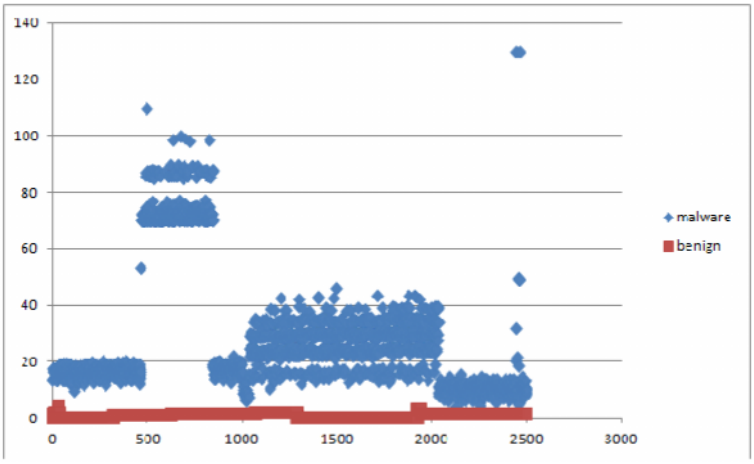

Figure 4. Comparison of PD Score of malicious and benign applications

For the above set of malware, we also calculated the Euclidean distance and compared with PD score. As expected PD score showed high degree of variation and helped in clearly classifying benign and malicious binaries. This is shown in figure 5. We removed two PD score of 16807 and 907 from graph data so that variation is visible.

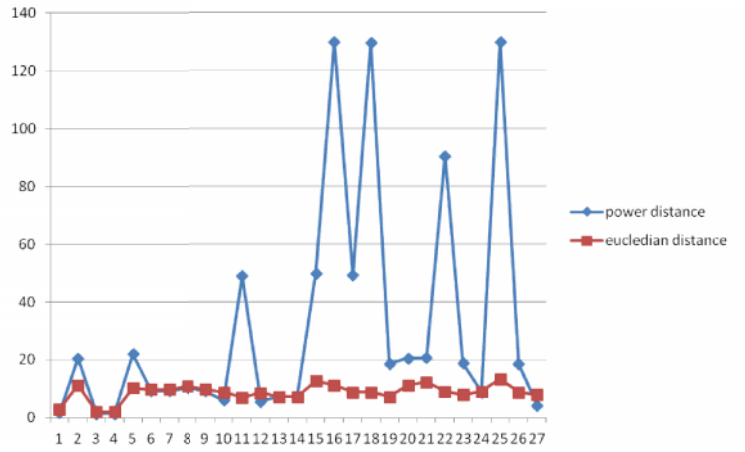

Figure 5. Comparison of ED and PD of Malware

8. CONCLUSIONS

APPBACS captures the behavior of the application in various aspects like file activities, registry activities, network activities and other kind of un-common behaviors and analyzes the reports automatically to give a final score on application. This score gives direct indication to user if application is malicious. I distributed the tool in various trainings and users were satisfied with such a tool. As part of future work, we will try to replicate the same scoring algorithm by capturing behavior using emulation of binary instructions. We intend to add more static parameters and detect binaries which checks for emulation software.

9. REFERENCES

[1] Quick Heal Technologies Report, January 2013, available at http://www.quickheal.com/blog/news20/
[2] Lenny Zeltser, “Three phases of malware analysis”, October 2010, available at http://computer-forensics.sans.org/blog/2010/10/11/3-phases-malware-analysis-behavioral-code-memory-forensics/
[3] Lenny Zeltser, “Free Toolkits for Automating Malware Analysis”, October 2010, available at http://blog.zeltser.com/post/1284687696/malware-analysis-tool-frameworks
[4] Lenny Zeltser, “Free Automated Malware Analysis Services”, August 2010, available at http://zeltser.com/reverse-malware/automated-malware-analysis.html
[5] Zero Wine, Malware Behavior analysis, http://zerowine.sourceforge.net/
[6] Willems, C.; Holz, T.; Freiling, F., "Toward Automated Dynamic Malware Analysis Using CWSandbox," Security & Privacy, IEEE , vol.5, no.2, pp.32-39, March-April 2007
[7] U. Bayer, C. Kruegel, and E. Kirda, “TTanalyze: A Tool for Analyzing Malware,” Proc. 15th Ann. Conf. European Inst. for Computer Antivirus Research (EICAR), 2006, pp. 180–192.
[8] Norman Sandbox, http://enterprise.norman.com/analysis
[9] Arne Vidstrom, Evading the Norman Sandbox Analyzer, http://ntsecurity.nu/onmymind/2007/2007-02-27.html
[10] Mandiant Red Curtain, http://www.mandiant.com/products/free_software/red_curtain/
[11] Treadwell, S., Zhou, M., “A heuristic approach for detection of obfuscated malware”, Proceedings of the 2009 IEEE ISI, Richardson, Texas, USA, June 08-11, pp. 291–299 (2009).

[12] K. Rieck, P. Trinius, C. Willems, and T. Holz. "Automatic Analysis of Malware Behavior using Machine Learning". Journal of Computer Security 2011.
[13] Power Distance, http://www.statsoft.com/textbook/cluster-analysis/#d
[14] Dionaea, dionaea.carnivore.it
[15] FakeDNS, Malcode analyst Pack, iDefense Labs
[16] INETSIM, Internet Services Simulation Suite, www.inetsim.org
[17] PCAP Library, http://www.winpcap.org/
[18] Himanshu Pareek, "Scanning a PCAP dump to find DNS and NETBIOS queries", available at http://www.codeproject.com/Tips/465850/Scanning-a-PCAP-dump-to-find-DNS-and-NETBIOS-queri
[19] Packed Executable Signatures, http://code.google.com/p/pefile/wiki/PEiDSignatures
[20] TrID, File Identifier, http://mark0.net/soft-trid-e.html
[21] Mike Geide Zscaler Inc., "N-gram Character Sequence Analysis of Benign vs. Malicious Domains/URLs", Available at http://analysis-manifold.com/ngram_whitepaper.pdf
[22] Microsoft Inc. "File System Mini filter driver", http://msdn.microsoft.com/en-us/library/windows/hardware/ff540402%28v=vs.85%29.aspx
[23] Microsoft Inc, "Filtering Registry Calls", http://msdn.microsoft.com/en-us/library/windows/hardware/ff545879%28v=vs.85%29.aspx
[24] Microsoft Inc., "PsSetCreateProcessNotifyRoutine Routine", http://msdn.microsoft.com/en-us/library/windows/hardware/ff559951%28v=vs.85%29.aspx
[25] Microsoft Inc., Detour Library, Software Package for re-routing Win32 APIs underneath applications, http://research.microsoft.com/en-us/projects/detours/
[26] Galen Hunt and Doug Brubacher, "Detours: Binary Interception of Win32 Functions", In Third USENIX Windows NT Symposium, USENIX, July 1999