J Supercomput
DOI 10.1007/s11227-015-1604-8

DTB-IDS: an intrusion detection system based on decision tree using behavior analysis for preventing APT attacks

Daesung Moon[1] · **Hyungjin Im**[2] · **Ikkyun Kim**[1] · **Jong Hyuk Park**[2]

Abstract Due to rapid growth of communications and networks, a cyber-attack with malicious codes has been coming as a new paradigm in information security area since last few years. In particular, an advanced persistent threats (APT) attack is bringing out big social issues. The APT attack uses social engineering methods to target various systems for intrusions. It breaks down the security of the target system to leak information or to destroy the system by giving monetary damages on the target. APT attacks make relatively simple attacks such as spear phishing during initial intrusion but a back door is created by leaking the long-term information after initial intrusion, and it transmits the malicious code by analyzing the internal network. In this paper, we propose an intrusion detection system based on the decision tree using analysis of behavior information to detect APT attacks that intellectually change after intrusion into a system. Furthermore, it can detect the possibility on the initial intrusion and minimize the damage size by quickly responding to APT attacks.

✉ Jong Hyuk Park
jhpark1@seoultech.ac.kr

Daesung Moon
daesung@etri.re.kr

Hyungjin Im
imhj9121@seoultech.ac.kr

Ikkyun Kim
ikkim21@etri.re.kr

1 Network Security Research Team, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

2 Department of Computer Science and Engineering, Seoul National University of Science and Technology (SeoulTech), Seoul, Republic of Korea

Published online: 31 December 2015

Keywords Advanced persistent threats (APT) · Behavior analysis · Decision tree · Intrusion detection

1 Introduction

The paradigm of cyber-attack is changing in accordance with the enhancement of security technology. The end user with accessibility to the important system, confidential data, and main employees that work in a company or institution is set as the goal of the initial intrusion. This is called an APT [1]. The purpose of an APT attack is to disturb or paralyze the normal operation by controlling the particular systems after intrusion to the main system. And it acquires confidentiality of government and main information of a company.

In the beginning, the APT starts to get the useful information about target system. For technical method, there is a method to find out the port that can be hacked through port scan. For non-technical method, there is a method of acquiring e-mail address of the person in charge, in-house job assignment, or security manager to security information through social engineering. Spear phishing based on the uses of e-mail input malicious code to the terminal of another end user. After acquiring the terminal route authority of the main employee, the account information (ID, Password, etc.) is taken through long-term monitoring, or remote control tool is used to take over the main system and network. The initial attack takes the simple attack method relying on the social engineering method. However, after intruding the system, the attack changes intellectually through zero-day attack, authority promotion, and backdoor establishment. Thus, the APT attack requires new detection method by accompanying big threat [2].

Therefore, we propose an intrusion detection system using decision tree to prevent destruction of system and long-term information leakage. The proposed system analyzes the behaviors of the malicious code and then sets rule through decision tree. Moreover, it classifies malicious status through analyzes execution file or installation file newly applied to the end user.

This paper is composed of 4 sections. Section 2 is about the research discussing about IDS classification, attack cases, and malicious code detection technology. In addition, Sect. 3 discusses about the proposed system. In Sect. 4, we summarize and conclude our research.

2 Related works

This chapter discusses the existing studies, APT attack cases, and core technologies related with our proposed system.

2.1 Intrusion detection system

Intrusion Detection System (IDS) is the system that detects malicious activity that occurs in the system. IDS can be classified according to the source of the data as follows [3–5].

Host-based IDS (HIDS) is the intrusion detection system for host such as personal computers or server computers. HIDS analyzes the resources such as the log records, files, folders, and detects and analyzes trace of infection. An APT attack uses malicious attachment files to infect the system. Thus, the role of HIDS is very important. The detection method of HIDS can differ according to product, but usually the hash value of the file is saved to check whether the system is detect the file and checks the change of file regularly. In addition, it detects the system call or vector table provided in the operation system to analyze and detect abnormal behavior. Most HIDS is composed of agent-type program within the computer.

Network-based IDS (NIDS) focuses on the external interface. The malicious activities that occurs within the network are detected through abnormality of the network traffic [6]. Service rejection attack, port scan, and packet snipping are detected through NIDS. NIDS is needed in all stages excluding the hiding stage of an APT attack. However, for the case of the APT attack, the complex malware is used for NIDS system of target to escape skillfully. Therefore, the technology to prevent this is being continuously researched.

Distributed IDS (DIDS) is the IDS where the various remote sensors sent information to the central management system. IDS has expansion but issue can occur in one point of the dispersion hierarchical type IDS. In addition, DIDS is centralized to be applied to cloud and lacks flexibility.

2.2 APT attack cases

An APT attack achieves purpose using various malicious code methods. Various modules such as antivirus detection evasion module, information collection module, and system's management authority module, and various zero-day vulnerabilities are used here.

Stuxnet set Iran's nuclear power plant as target to hack into supervisory control and data acquisition (SCADA) system. Malicious program was inserted to the Programmable logic controller (PLC) of nuclear reactor [7]. This destroyed the Uranium enrichment program and interrupted the creation of nuclear weapons. It has strong metastatic features because it is composed of components possible for 10 self-cloning. Stuxnet used Window shell LNK vulnerability to create ".dll" file that is loaded in accordance with search process (explorer.exe). This composed the environment where malicious code can be executed through NTDLL hooking. Furthermore, window server service vulnerability, window printer spooler vulnerability, and share network service vulnerability were used to acquire management authority. Stuxnet used USB for attack where the network net became separated to use the vulnerability that can deliver malicious code to even the system. When the system is not a target system, Stuxnet did not perform particular activity and used rootkit to remove own existence.

Duqu is known as the variant of stuxnet, but stuxnet and duqu have different purpose. If stuxnet was made after destroying the target system and making the system not able to operate, duqu has the purpose of information leakage to collect information of the target system. Initial intrusion was attempted through spear phishing attaching infected Micro Office Document. After being successfully hacked into the system and

installing back door, communication was executed with the control and command (C&C) server. In addition, duqu applies rootkit after 36 days from installed date to automatically delete and not leave trace of intrusion. Key logging component collects main information such as password. The collected information is used to acquire authority to access other system of the network by the invader [8].

Carbanak is the APT attack targeting financial institutions. This attack approaches the target system through phishing email. In addition, it used vulnerability in Microsoft office, Microsoft word. Carbanak was installed a backdoor on the user's machine, which was hidden in the "svchost.exe" process. They check whether the security program. Try to privilege escalation exploits a vulnerability in the Windows series of behavior if a security program on the target system to prevent malware that was undetected. In addition, the user's key stroke information with screenshot every 20 s the information was collected. They use Remote Desktop Protocol (RDP) to maintain a connection with an infected computer, thus they can steal money consistently [10].

2.3 Behavior-based detection system

Behavior-based detection system is the method that detects intrusion by the comparison and analysis of the attack patterns as well as activity in the process, registry, and network that occur in the system. It can overcome the limitation of signature-based analysis. Behavior-based detection method is classified into the system-based detection method which is based on the activities that occur within the computer system and the network detection method based on the activities that occur within all networks [11,12].

System-based detection method System-based detection method is based on the detection of various activities that occur within the computer system. Generally, this method is divided into integrity inspection method and behavior block method. The integrity inspection method checks whether the integrity status of the computer system is being continuously maintained. When the status of the system changes, this is considered as malicious change, and there is the advantage of detecting the malicious code that is not yet known. On the other hand, there is a weakness of difficult for early response because of the detection after being damaged. Behavior block method monitors all behaviors that occur in the system [13]. When behavior similar to malicious code occurs, the method detects and traces which execution file induced the behavior.

Network-based detection method Network is present in various behaviors because network may cause in number of traffic by the external and internal network. Excessive network access, IP change packet delivery, MAC change packet delivery, Address Resolution Protocol (ARP) change packet delivery, etc., correspond to check malicious traffic through single behavior. However, attack due to single behavior according to the development of information security became meaningless. Thus, malicious codes are developing to manipulate network behavior through various methods to not get caught to the rule of security product. These malicious codes analyze the behavior of

detailed items such as the number of session count, number of packet delivery target IP, total number of In/Out packet, and number of protocol.

2.4 Decision tree

Decision tree analyzes the records of the data collected in the past and makes the patterns of existing in between as tree form. The classification module classified new record and is used to predict the corresponding type of value. The decision tree analysis suggests various methods such as classification standard, stop rule, and pruning method. According to how they are combined the different decision trees are formed. Various algorithms are being researched to form decision tree accurately and quickly, so that newly developed algorithms are being announced.

The decision tree algorism always notifies the proof of classification or prediction. Thus, it is easy to understand the created pattern. Furthermore, even in the case when number of properties that compose data are unnecessary, the properties that do not influence the classification during module structure are automatically excluded for easy data selection [14].

There are many algorithms that analyze decision tree and ID3 algorithm, was introduced first for decision tree. ID3 uses an information theory and its class is based on entropy. Since it cannot use a continuous attribute, C4.5 algorithm was developed to complement this. In addition, CART and CHAID algorithms based on statistical analysis methods such as Chi-Square, T-test, and F-test were developed. In this paper, our proposed system utilizes C4.5 algorithm, which can use continuous data and make accurate categorization even though there are many types [15].

The decision tree is fast and accurate in learning a huge dataset. Therefore, it is good for analyzing various data created in a system and numerous traffics that flow into the network like IDS system. In addition, it is easy to understand the created pattern with decision tree, so a security manager can easily monitor and edit a situation on his own. Lastly, since decision tree has excellent accuracy compared to other machine-learning techniques, this works as a great advantage in IDS where accuracy is an important element.

2.5 Other detection systems

Shahid Alam et al. creates behavioral signature in real-time and proposed the framework that can detect metamorphic malware. Annotated Control Flow Graph (ACFG) and Sliding Window of Difference and Control Flow Weight (SWOD-CFWeight) were suggested as new technology. In contrast to other technologies, ACFG provides quick comparison of CFG without change of detection accuracy. SWOD-CFWeight eases the current issue related to the regular change of opcodes such as the use of other compiler, optimization of compiler, operating system, and obfuscation [16].

Aziz Mohaisen et al. suggested the labeling system that resolved the existing weakness and malicious software analysis based on automated behavior called AMAL. AMAL is composed of two sub-systems including AutoMal MaLabel. AutoMal is the tool that collects behavior artifacts of memory, network, registry, and file system.

MaLabel uses the collected artifact to create feature and classifies the data based on the learned data [17].

Wang et al. uses Support Vector Machine (SVM) based on the behavior signature to develop automatic malware detection system. Cross validation scheme uses 60 actual malwares to solve the accuracy issue. The suggested system is composed of 3 types of modules including system backup, system monitoring, and system recovery. System backup provides backup for important system files in server and client. System monitoring uses the previous backup data of the system to recover operating system and user data. System recovery module uses reversing engineering technology to recover the process infected to the malicious software [18].

Ammar Ahmed E. Elhadi et al. suggested the malware detection system based on API call graph. To strengthen the detection of variety malicious software, API call graph has reliability on the data and also has advantage of the sequential profile. In addition, it does not use just one from signature and behavior analysis, but it uses both for execution [19].

Liu Wu et al. proposed detection algorism using Malware Behavior Feature (MBF) and developed detection system based on this. MBF based malware detection system is composed of batch job module, detection engine module, and behavior judgment module. Batch job module executes quick and overall detection to quickly handle the malicious software. Detection engine module is composed of dynamic link library file of Window. Behavior judgment module helps detecting detection engine [20].

Youngjoon Ki et al. analyzed the API call sequence of malware and proposed a system that judges whether a file is malicious compared to an existing benign program. The system acquires the information of API that occurs during program run-time and analyzes its DNA sequence information. The system suggested by Youngjoon Ki et al. has 0.999 for F-measure and 0.998 for accuracy, which are very high numbers [21].

Tian et al. [22]. used dynamic analysis method and classified based on the frequency of API. In addition, 1368 malwares were analyzed to proposed system that can classify malware family.

Ye et al. [23]. used silent analysis method. PE code was analyzed to extract sequence, and this was compared with signature DB to execute primary classification. Moreover, machine-running method was used for secondary classification. Sequence of API was used as feature to analyze 17,366 malware.

Sathyanarayan et al. [24] used IDA PRO to extract API. Frequency of API was calculated to define sequence based on this, and the rule of malware was created. Thus, when the API frequency of test file was given, it judges whether it is malware or normal file by comparing with sequence.

3 DTB-IDS

3.1 Architecture

In this paper, we propose an intrusion detection system based on decision tree (We call it as DTB-IDS). The DTB-IDS is a system that classifies the malicious code by learning decision tree about the malicious behavior in the network and system. DTB-

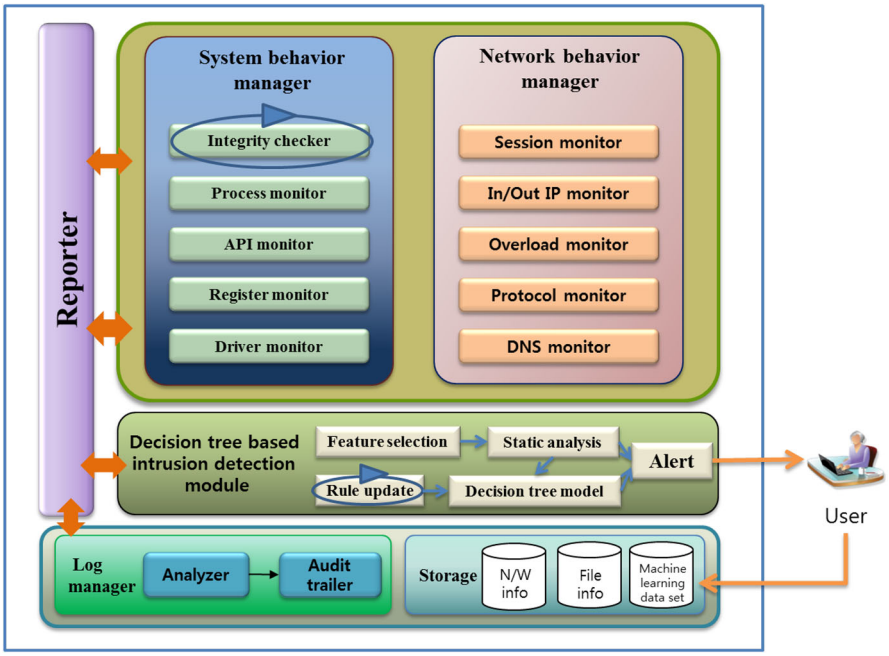

Fig. 1 Architecture of DTB-IDS

IDS is composed of 6 models as follows: Reporter, System behavior manager, Network behavior manager, Decision tree-based Intrusion detection module, Log manager, and Storage. Figure 1 shows architecture of the proposed system.

Reporter is in charge of mutual communication between modules. Each module must be given the information to detect decision tree-based intrusion detection module to detect abnormal behavior as mutual complementary relationship. In addition, the feedback is received from user based on the detected, saves the feedback, and then needs to update rule set again. This is where mutual communication is essential.

System behavior manager is composed of various things such as functions of integrity checker, process monitor, API monitor, register monitor and driver monitor. In the integrity checker, the behavior change of the system continuously changes. For API, Process, and Register, the live data changes continuously when the system operates, and it is changing after saving the hash value of basic files provided by OS other than this live data, event occurs to detect malignancy. When the event occurs, this is delivered to the recognition module. In addition, for the live data such as process, API, register, and driver, the log information is requested from Log manager when the main information is created or deleted and delivers to the detection module.

Network behavior manager is composed of session monitor, in/out IP monitor, overload monitor, protocol monitor, and DNS monitor. For the session monitor, the count is continuously accumulated when session occurs on the in/out direction per process.

When session occurs suddenly, then the event occurs. When IP Address/MAC/ARP change packet occurs in the in/out monitor, the number of occurrence is detected to create event. In the overload monitor, the increase and reduction of the total traffic are detected. When excessive network access occurs, event about this occurs. When excessive access of unnecessary protocol occurs, then the event generates. Last, for the dns monitor, event occurs while moving based on whitelist or blacklist. When event occurs in each monitor, the information of the process is delivered through detection module to judge the malignancy in the detection module.

Decision tree-based intrusion detection module has the following functions: rule update, feature selection, static analysis, decision tree model and alert. Rule update is the stage of learning the data to be applied to decision tree. It is updated when new malicious code is discovered or when new dataset is added. Feature selection is the function that extracts feature to classify or learn the events that occur in the system or network. It is executed first when receiving information from the other module. Based on this information, the pattern is created and the pattern is initially classified due to static analysis. When initial classification does not occur and it is delivered to the Decision Tree classifier to judge the malignancy through the algorithm learned earlier. Last, alert function checks the malignancy to the users. After the user decides malignancy, the result from here is delivered to the DTB-IDS so that assessment feedback can occur.

The log manager has received logs from the system behavior, network behavior manager, and decision tree-based intrusion detection module. As numerous logs are created, the analyzer normalizes the received logs to analyze and perform audit trail on them. When the decision tree-based intrusion detection module discovers malware, the audit trailer tracks its route and restores the changed data to the original state.

Storage is the module that saves various information used in decision tree-based intrusion detection module. N/W info saves network information that cannot be learned or categorized by decision tree, including internal network information, malicious network signature, and network behavior sequence. In addition, the file info stores data for static analysis such as the information of file installed in the user's system or malicious file signature. Finally, the assessment feedback received from the user is saved to the machine-learning dataset where it provides additional information when updating the rule in future detection module.

3.2 Behavior analysis

System behavior Malware has system behavior that it mainly performs to install a malicious code and survive for a long time. The examples are: file creation, file deletion, file reading, file recording, file duplication, file route check, temporary file route, registry key creation, registry key open, registry key designation, registry deletion, process creation, process end, process information, and command execution. It is difficult to determine whether a process is a malicious behavior or normal behavior because a single behavior has occurred. Thus, this paper restructures features by analyzing the

system API created by a malicious file. To represent the patterns of all behavior, the proposed system in this paper is reconstructed by accumulating the frequency of the same feature event. The Table 1 shows a set of APIs in the system according to the behavior.

Network behavior Most researches analyze network behaviors during a fixed period of time. Since networks generally have a flow, their behaviors must be classified based on

Table 1 Behavior and APIs as feature

Behavior	APIs	Field
File creation	CreateFile, CreateDirectoryA, SetFileAttributes	API1
File delete	DeleteFileA, RemoveDirectoryA	API2
File read	ReadFile,findClose, GetFileSize, GetFileAttributes, FindFirstFileA	API3
File record	WriteFile, SetFilePointer	API4
File copy	CopyFile, CopyFileA, MoveFileA	API5
File path	GetModuleFileName	API6
File temporary path	GetTempPathA	API7
Registry key creation	RegCreateKey, RegCreateKeyEx, RegCreateKeyExA	API8
Registry delete	RegDeleteValueA, RegDeleteKeyA, RegDeleteValueW, RegDeleteKeyW	API9
Process creation	CreateProcess, CreateProcessA, CreateThread	API10
Process Termination	TerminateProcess, ExitProcessClose, HandleGetExit, CodeProces	API11
Process information	GetCurrentProcessProces, s32FirstW, Process32First, Process32Next, Process32NextW	API12
Command execution	WinExec, ShellExecuteA, GetCommandLineA	API13
Mutex	CreateMutexA, ReleaseMutex, OpenMutexW, OpenMutex	API14
Network data transportation	send, recv, WSASend, ntohs, inet_addr, inet_ntoa	API15
Network connection	InternetCrackUrlA, InternetOpenUrlA, HttpQueryInfoW, InternetConnectW, HttpOpenRequestW, HttpSendRequestExW, HttpAddRequestHeadersW, HttpAddRequestHeadersA, HttpEndRequestW, WinHttpOpen, InternetReadFileExA, InternetOpenA, InternetOpenW	API16
File Download	URLDownloadToFileW, URLDownloadToFile, URLDownloadToCacheFileA, URLDownloadToCacheFileW	API17
Memory Management	VirtualAlloc, GlobalFree, HeapAlloc, HeapFree	API18
Time function	GetTickCount, Sleep, WaitForSingleObjec	API19
Error handling	GetLastError, SetErrorMode, SetLastError	API20
String manipulation	lstrlenA, lstrcmpiA, lstrcpynA, lstrcatA, lstrcmpA, CharNextA, MultiByteToWideChar	API21
DLL Injection	CreateRemoteThread, WriteProcessMemory	API22

time period. In addition, network attacks must operate within a short period of time to become threatening to networks. However, some of the process behaviors do not have the concept of time. What is important is not the occurrence of a malicious behavior during a certain period but is actually whether it has occurred or not. Session count (in/out), number of times that IP modulation, MAC modulation, and ARP modulation have occurred, and the number of IP addresses to send packets are main examples whose behaviors must be restricted by time period. Time period is not set separately for each abnormal behavior, but one period is made to be applied the same for all behaviors. If it is 1 s, an abnormal behavior in which the session count is more than a few times within 1 s, the number of times that IP/MAC/ARP modulation have occurred, and the number of IP addresses to send packets become the signs of abnormal behavior. If the detection period for an abnormal behavior is set at 5 s, the process can be detected as an abnormal sign if external access occurs 1000 times within 5 s. The time period does not start unconditionally from a certain point. If there are 100 abnormal behaviors of external access during the time period of 5 s, the time for measuring 5 s starts from the moment that 1 external access is made. Assume that 5 s are counted from the first external access and that an abnormal behavior has occurred if there are 100 external accesses within 5 s. If the external access occurs 99 times and not 100 within 5 s, the abnormal behavior is considered not to have occurred, and the time is initialized to 0 s and the count of 99 times is also initialized to 0 s. After, if there an external access is made again after some time, the 5 s-count resumes. If the external access occurs 101 times within 5 s after the count starts, it can be recorded that the abnormal behaviors of 100 external accesses have occurred.

3.3 Service scenario

This chapter discusses a service scenario for the proposed DTB-IDS. The acronyms used in the scenario are shown on Table 2. The scenario is composed of two types of detection for known malware and unknown malware as Fig. 2.

Type A. Detection scenario of known malware

Step 1. U_Agent → SysBM: SysB_Info
An user agent sends the system behavior information to system behavior manager while an user uses a personal system.

Step 2. U_Agent → NetBM: NetB_Info
The user agent sends the network behavior information to network behavior manager while the user uses a personal system.

Step 3. SBM → DTB-IDSM : Anal_SB()
The system behavior manager analyzes the system behaviors then sends the detailed results (time, APIs, DLL information, etc.) of analysis to the decision tree-based IDS module. We discuss the results in Sect. 3.4 in details.

Step 4. NBM → DTB-IDSM : Anal_NB()
The network behavior manager analyzes the network behaviors then sends the detailed results of analysis to the decision tree-based IDS module.

Step 5. DTB-IDSM : Feature_Sel(), Static_Anal(), Dec_Mal()

Table 2 Definition of acronyms

Term	Explanation
U_Agent	User agent for monitoring system and network
SysBM	System behavior manager
NetBM	Network behavior manager
DTB-IDSM	Decision tree-based IDS module
SysB_Info	System behavior information
NetB_Info	Network behavior information
Alt_Msg	Alert information for malware decision
Anal_SB()	Analysis function for the system behaviors
Anal_NB()	Analysis function of the network behavior
Feature_Sel()	Feature selection function
Static_Anal()	Static analysis function
Class_DT()	Classification function by decision tree

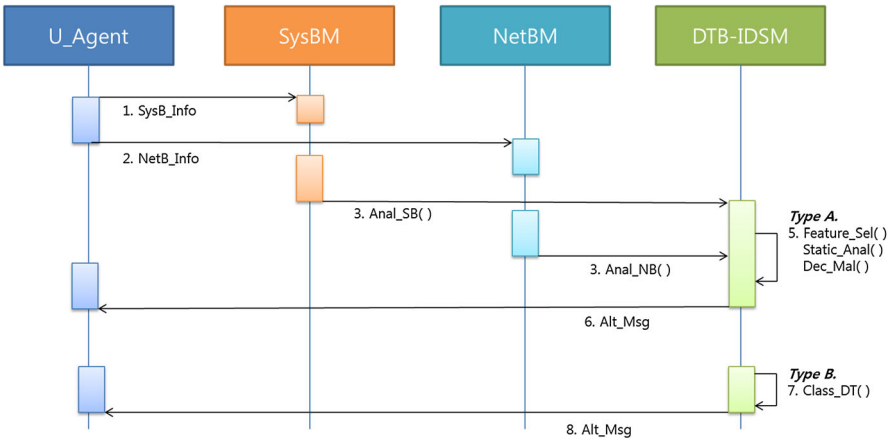

Fig. 2 Detection scenario for known and unknown malware

The decision tree-based IDS module selects features from the results then performs static analysis such as blacklist analysis and reputation-based detection, and so on. In addition, the module decides if the system or network is infected by malwares.

Step 6. DTB-IDSM → U_Agent: Alt_Msg

The decision tree-based IDS module sends an alert message about the decision to the user agent.

Type B. Detection scenario of unknown malware

Step1 ∼ 6: These steps are the same contents in type A.

Step 7. DTB-IDSM: Class_DT()
If the decision tree-based IDS module cannot detect malicious codes in static analysis, the module inputs the features of system behaviors and network behaviors to decision tree. Then, the module decides if the system or network is infected by malwares. We discuss the results in Sect. 3.4 in details.

Step 8. DTB-IDSM → U_Agent: Alt_Msg
The decision tree-based IDS module sends an alert message for the decision to the user agent.

3.4 Experiment and analysis

Data collection is very important to analyze the feature of malicious code. To collect the feature of API per behavior commonly used in the malicious code that was previously mentioned in chapter 2. In this experiment, we installed a virtual machine and ran 100 malicious codes and 30 normal files, and selected feature using a collector. API monitor uses collect the feature of malicious behaviors [25]. Programs such as notepad and calculator that were originally installed on windows were used for normal files, and malicious codes are collected from Malshare [26]. For data collection, all kinds of works are performing together to observe behaviors for 5 min. Table 3 shows the information collected from API monitor.

We analyzes the information of 3000 behavior events that is previously collected, represents the occurrence frequency of behaviors that are often used in malicious codes, and learn this through a decision tree.

Data-mining tool Weka is used, and the experiment is conducted by applying C4.5 algorithm. Figure 3 is a classification table of decision tree showing the rules of malicious files. The following results are drawn through behaviors such as process ending behavior, download behavior, registry key creation, access to temporary file place, file deletion, and the collection of process information.

Recently researches on analyzing and detecting malicious codes using API in the system are constantly increasing. As shown by some research results [7–9], research on detecting APT attack-related malicious codes is more difficult than general malicious codes such as virus, worm, botnet, and trojan. The existing researches [24–26] that were examined in the other detection systems in Sect. 2 were focused on detecting

Table 3 Information collected from API monitor

Collected information	Explanation
Time	Feature event-occurrence time
Module	Name of the DLL that made the API call
Category	A classificatory division in API
API	API system calls made by applications and services
Result	Return value

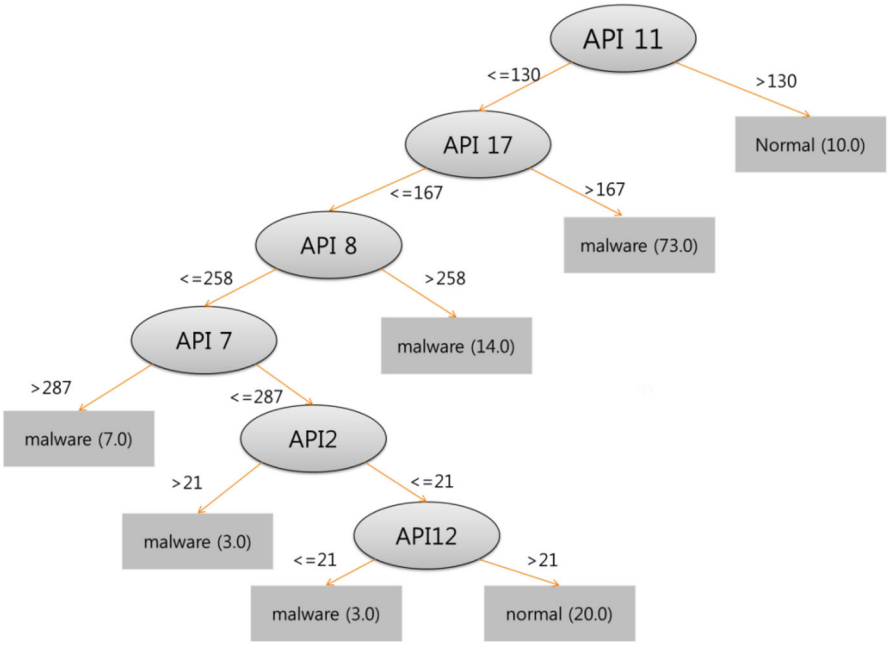

Fig. 3 Visualize decision tree results

general malicious codes, and showed accuracy of 89.1 % on average. The accuracy of detection is 84.7 % in our experiment. Maybe our proposed system seems to have a lower accuracy than other detection systems. However, considering that it is the detection of malicious codes related APT attack as we discussed, the detection accuracy of our proposed system is comparatively excellent.

4 Conclusion

An APT attack has its own purpose and goal. The APT attack carried out continuous and elaborate attack by deciding the target in contrast to the malware of the same type such as Bot, Trojan, and Worm. While the traditional attack is an attack based on special people by showing symptoms to the execution, the APT attack evades the detection where the invader leaks wanted information. Generally, the APT attack analyzed the information of target company such as antivirus within company, goal within the company, and personnel information and technical information about the target system based on the social engineering method to find the vulnerability of the target company. Then, the initial intrusion is attempted where prevention of initial intrusion is difficult with Antivirus.

In this paper, we proposed an intrusion detection system to minimize the damage from the APT attack that changes intellectually by detecting the possibility of initial intrusion. The proposed system analyzed the process and behavior of network to learn through decision tree, and detected intrusion based on behavior information.

The proposed system can be support a minimized damage size by executing quick detection from APT attacks that newly occurs along with the existing malware.

In the future work, there are some needs to study about the distributed APT security methods in network and system by standardizing the main behaviors that are created in process.

Acknowledgments This work was supported by Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. B0101-15-1293, Cyber targeted attack recognition and trace-back technology based on long-term historic analysis of multi-source data)

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this paper.

References

1. Nikos V, Gritzalis D, Apostolopoulos T (2013) Trusted Computing vs. advanced persistent threats: can a defender win this game?. In: Ubiquitous Intelligence and Computing, 2013 IEEE 10th international conference on and 10th international conference on autonomic and trusted computing (UIC/ATC). IEEE, pp 396–403
2. Nikos V, Gritzalis D (2013) The big four-what we did wrong in advanced persistent threat detection?. In: Availability, Reliability and Security (ARES), 2013 Eighth International Conference on. IEEE
3. Seresht NA, Azmi R. MAIS-IDS: a distributed intrusion detection system using multi-agent AIS approach. Eng Appl Artif Intell 35:286–298
4. Hung-Jen L, Lin C-HR, Lin Y-C, Tung K-Y (2013) Intrusion detection system: A comprehensive. J Netw Comput Appl 36(1):16–24
5. Gaur Madhu Sharma, Pant Bhaskar (2015) Trusted and secure clustering in mobile pervasive environment. Human-centric Comput Inf Sci 5(32):19
6. Kang H-S (2015) A real-time integrated hierarchical temporal memory network for the real-time continuous multi-interval prediction of data streams. J Inf Process Syst 11(1):39–56
7. Falliere N, Murchu Liam O, Chien E (2011) W32.Stuxnet Dossier Version 1.4 (February 2011). Symantec Corporation
8. Bencsáth B, Pék G, Buttyán L, Félegyházi M (2012) Duqu: analysis, detection, and lessons learned. ACM Eur Workshop Syst Secur (EuroSec) 2012
9. (2013) The 'Red October' Campaign—an advanced cyber espionage network targeting diplomatic and government agencies. GReAT, Kaspersky Lab
10. (2015) CARBANAK APT THE GREAT BANK ROBBERY", Version 2.1. Kaspersky lab
11. Modi C, Patel D, Borisaniya B, Patel H, Patel Avi, Rajarajan Muttukrishnan (2013) A survey of intrusion detection techniques in Cloud. J Netw Comput Appl 36(1):42–57
12. Nissim N, Moskovitch R, Rokach L, Elovici Y (2014) Novel active learning methods for enhanced PC malware detection in windows OS. Expert Syst Appl 41(13):5843–5857
13. Ahn Hosang, Kim Hanna, Park Jae Roh (2014) Smart Monitoring of indoor asbestos based on the distinct optical properties of asbestos from particulate matters. J Converg 5(4):11–14
14. Pradhan B (2013) A comparative study on the predictive ability of the decision tree, support vector machine and neuro-fuzzy models in landslide susceptibility mapping using GIS. Comput Geosci 51:350–365
15. Jidiga GR, Sammulal P (2014) Anomaly detection using machine learning with a case study. In: 2014 IEEE international conference on advanced communication control and computing technologies (ICACCCT), pp 1060–1065
16. Alam S, Horspool RN, Traore I, Sogukpinar I (2015) A framework for metamorphic malware analysis and real-time detection. Comput Secur 48:212–233
17. Mohaisen A, Alrawi O, Mohaisen M (2015) Amal: high-fidelity, behavior-based automated malware analysis and classification. Comput Secur 1–16

18. Wang P, Wang Y-S (2015) Malware behavioural detection and vaccine development by using a support vector model classifier. J Comput Syst Sci 81(6):1012–1026
19. Elhadi AAE, Maarof MA, Barry BIA, Hamza H (2014) Enhancing the detection of metamorphic malware using call graphs. Computer Secur 46:62–78
20. Wu LIU, Ping REN, Ke LIU, Hai-xin DUAN (2011) Behavior-based malware analysis and detection. In: 2011 first international workshop on complexity and data mining, pp 39-42
21. Ki Y, Kim E, Kim HK (2015) A novel approach to detect malware based on API call sequence analysis. Int J Distrib Sensor Netw 2015(Article ID 659101):9
22. Tian R, Islam MR, Batten L, Versteeg S (October 2010) Differentiating malware from cleanware using behavioural analysis. In: Proceedings of the 5th International Conference onMalicious and Unwanted Software (MALWARE '10). Nancy, France, pp 23–30
23. Ye Y, Wang D, Li T, Ye D (2007) IMDS: intelligent malware detection system. In: Proceedings of the 13th ACM SIGKDD International conference on knowledge discovery and data mining. ACM, pp 1043–1047
24. Sathyanarayan VS, Kohli P, Bruhadeshwar B (2008) Signature generation and detection of malware families. In: Information Security and Privacy. Springer, Berlin
25. API Monitor. http://www.rohitab.com/apimonitor, Accessed 30 Nov 2015
26. Malshare. http://malshare.com/, Accessed 30 Nov 2015