Malware Detection and Tracer Approach for Operating System

Dokuparthi Prasanthi[1], V. Rama Krishna[2]

[1]M.Tech student, Department of CSE, Anurag Group of Institutions, Hyderabad, India

[2]Assistant Professor, Department of CSE, Anurag Group of Institutions, Hyderabad, India

Abstract: ***Modern computer systems are built on a foundation of software components from a variety of vendors. While critical applications might undergo extensive testing and evaluation procedures, the heterogeneity of software sources hazardous the integrity of the execution environment for these trusted programs. For instance, if an attacker can be able to merge an application exploit with privilege increase vulnerability, the Operating System (OS) can become corrupted. Mandatory Access Control (MAC) in a commercial operating system to handle malware problem is a challenge but also a capable approach. The firmest barriers to apply MAC to defeat malware programs are the incompatible and unusable problems in existing MAC systems. The aim of our study is to address these issues design a novel Efficient Malware Detection and Tracer design (EMDT) using Hidden Markov model, which incorporates intrusion detection and tracing in an operating system. In this proposed approach conceptually consists of three actions: tracing, detecting and restricting deduced intruders .The novelty of the proposed study is that it leverages light-weight intrusion detection and tracing techniques to automate security label configuration that is widely acknowledged as a tough issue when applying a MAC system in practice. The other is that, rather than controlling information flow as a traditional MAC does, it traces intruders and restricts only their significant malware behaviours', where intruders characterize processes and executables that are potential agents of a remote attacker. Our prototyping and testing's on Windows operating system show that Tracer can effectively defeat all malware samples tested via blocking malware behaviours while not causing a significant compatibility problem.***

Keywords: Detection, intrusion, malware, tracing, vulnerability.

1. Introduction

Malicious software (i.e., Malware) is one of the most severe computer security problems today. A network of hosts which are cooperated by malware and controlled by attackers can cause a lot of damages to information systems. As a useful malware defence technology, Mandatory Access Control (MAC) works without relying on malware signatures and blocks malware behaviours before they cause security damage. Even if an unauthorized user manages to breach other layers of defence, MAC is capable of act as the last shelter to avoid the entire host from being compromised. However, MAC mechanisms built in commercial operating systems (OS) often go through from two problems which make general users unenthusiastic to assume them. One problem is that a built-in MAC is mismatched with a lot of application software and thus interferes with their running and the other problem is low usability, which creates it difficult to configure MAC properly. Our observations are as follows: The incompatibility problem is introduced because the security labels of existing MACs are not capable to distinguish between malicious and benign entities, which Causes a enormous number of False Positives (FP) (i.e., treating benign operations as malicious) thus avoiding many benign software from performing legal operations; the low-usability problem is launched, because existing MACs are not capable to automatically label the huge number of entities in OS and thus require tough configuration work at End users. With these investigation results, our main objective is to propose a novel MAC enforcement approach EMDT, this consists of three actions: Detection, tracing and restriction.

Each process or executable has two states, suspicious or benign. The contributions of this study are We introduce EMDT, a novel MAC enforcement approach which combines intrusion detection and tracing techniques to disable malware on a commercial OS in a compatible and usable manner. We have implemented EMDT to immobilize malware timely without need of malware signatures. We investigate the root reason so discover compatibility and low usability problems of existing MACs. Although not all the observations are brand new, we consider that understanding these reasons more comprehensively and illustrating them through the design of an actual system are useful for other MAC researchers.

2. Related Work

DTE proposed by Badger *et al*. (1995) is a classical MAC model to confine process execution, which group's processes and files into domains and types, respectively and controls accesses among domains and types. Tracer can be considered as a simplified DTE that has two domains (i.e., benign and suspicious) and four types (i.e., benign, read-protected, write-protected and suspicious). Moreover, Tracer can usually configure the DTE attributes (i.e., domain and type) of processes and files under the maintain of intrusion detection and tracing so as to develop usability. PRECIP Wang *et al*. (2008) addresses several practical issues that are significant to contain spyware that be determined to leak sensitive information. The risk-adaptive access controls (Kaspersky Lab, 2012). That targets to create access control more dynamic so as to attain a better tradeoff between risk and benefit. Most existing antimalware technologies are based on detection (Kirda *et al*., 2006; Martignoni *et al*., 2008). Tracer tries to combine detection and access control so that it not only can detect but also can block malware behaviours before their harming security. Antimalware technology that be similar to Proposed EMDT is behaviour blocking (Nachenberg, 2002) which can confine the behaviours of definite adverse programs that are profiled in advance. Many commercial antimalware tools (Kaspersky Lab, 2012; Viper Inc., 2012) also have a behaviour-based module to protect against unknown malware programs.

Problems in MAC

Incompatibility is a familiar problem when enforcing a MAC modeling operating system (Li *et al.*, 2007; Fraser, 2000; Wang *et al.*, 2008). To examine its core reason, in a secure network environment, we set up two mechanisms to run MAC enforced operating systems with MLS policy allowed and MAC module allowed. After a few days, we surveyed that these MAC systems produced a enormous number of log records about denied accesses, which specified that some applications failed and some acted irregularly. As the operation environment is secure without intrusion and malware, these denied accesses are thus "false positive." However, from the view of intrusion thwarting, these processes do not necessarily represent intruders so that their "read" or "write" accesses to the/tmp should not be merely denied. Although it is possible to resolve this problem by adding "hiding sub directories" under/tmp, it is still difficult to eliminate the FPs resulting from many other shared entities on an OS Relying on these labels, a MAC system habitually fails to make correct decisions on intrusion blocking which eventually results in many FPs. Low usability is another problem in a MAC-enabled system, as it often requires make difficult configurations and unconventional ways of usage.

3. Proposed System

Efficient Malware Detection and Tracer (EMDT): In this section, we present our EMDT approach that aims to immobilize malware in a OS by disagreeing malware behaviours. The adversaries of EMDT are malware programs that break into a host through the network or removable drives. As OS is the most popularly attractive to hackers, the description of EMDT is designed Appling it to operating systems with some changes.

4. Overview of EMDT

4.1 Overview

The design of an access control mechanism is to define the security label. We initiate a new form of security label called suspicious label for our EMDT approach. It has two values: suspicious and benign. Meanwhile, EMDT only allocates a suspicious label to a process or an executable, because a process is possibly the agent of an intruder and an executable determines the execution flow of a procedure which represents an intruder. When a process requests to access these entities, EMDT mainly uses their DAC information to make access control decisions, thus a vast amount of configuration work can be reduced while keeping traditional usage conventions unchanged.

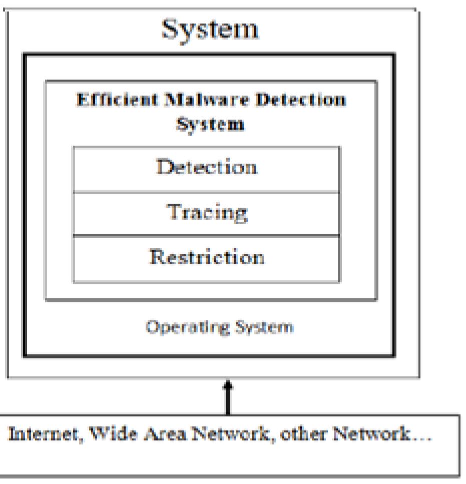
System
Efficient Malware Detection System
Detection
Tracing
Restriction
Operating System
Internet, Wide Area Network, other Network…

Figure 1: EMDT Overview

The above Fig. 1 gives an overview of EMDT which consists of three types of actions, tracing, detection and restriction. Each process or executable has two states, suspicious and benign. The restriction action forbids a suspected intruder to make malware behaviours in order to guard CIAP. That is to protect integrity, confidentiality and availability, as well as to stop malware propagation. The three actions study as follows: Once detecting a suspected process or executable, EMDT labels it as suspicious and traces its descendent and interacted processes, as well as its generated executables. EMDT does not restrict benign processes at all and permits suspicious processes to run as long as possible but stops their malware behaviours that would cause security damages.

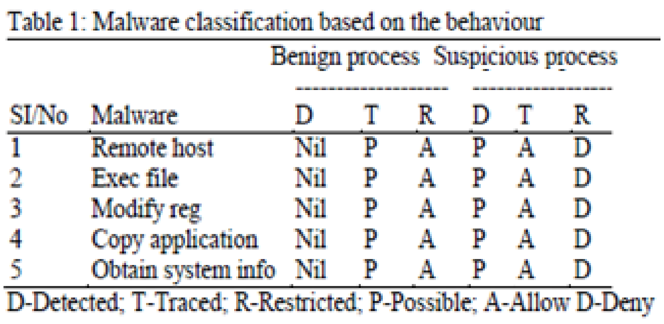
Table 1: Malware classification based on the behaviour

		Benign process			Suspicious process		
SI/No	Malware	D	T	R	D	T	R
1	Remote host	Nil	P	A	P	A	D
2	Exec file	Nil	P	A	P	A	D
3	Modify reg	Nil	P	A	P	A	D
4	Copy application	Nil	P	A	P	A	D
5	Obtain system info	Nil	P	A	P	A	D

D-Detected; T-Traced; R-Restricted; P-Possible; A-Allow D-Deny

The object and parameter signify the target and parameter of the operation, respectively. Specific malware behaviours monitored in the current version of EMDT, which includes the 30 critical malware behaviours shown in Table 1. Moreover, EMDT allocates dynamic addition of new behaviours. EMDT utilizes the subject label and behaviour to build a decision while normal MACs use the subject label , object label,operation and parameter. As behaviour consists of operation, object and parameter, EMDT actually uses the same four factors of normal MAC decision. Moreover, EMDT's decision procedure produces three possible access control results: "allow," "deny," and "change label," which be similar to those of normal MACs. The detailed decision logic of Tracer is shown in Table 1. The detection and tracing actions guide to the decision result "change label," while restriction action leads to "deny." All access requests not denied are allocated. As an online approach, Tracer be able to produce the FP rate lower than that of behaviour-blocking mechanisms in commercial antivirus software. This is attained as a MAC system, EMDT blocks a behaviour based simultaneously on the behaviour and security label (i.e., the suspicious label of the current process), rather than simplify the behaviour as done by a behaviour-blocking system.

4.2 Detecting intruders

The detecting action is liable for identifying all potential intruders. we design a light-weight intrusion detection algorithm that can identify all potential intruders but may have a relatively higher FP rate at the initial step. Tracing and restricting actions, will still agree to it to run rather than stop it immediately, but only avoid it from executing featured malware behaviours. As depicted in the above Fig. 1, the detection works at two levels: entrance and interior:

$$D(P) = \begin{cases} \text{Benign otherwise } \rightarrow \\ \textit{Suspicious if } s€p \end{cases} \qquad (1)$$

Where, D (P) is detection of process, signature s belongs to signature based, it comes in distrustful folder. The detection at entrance attempts to check all possible venues through which a malware program may break into the system.

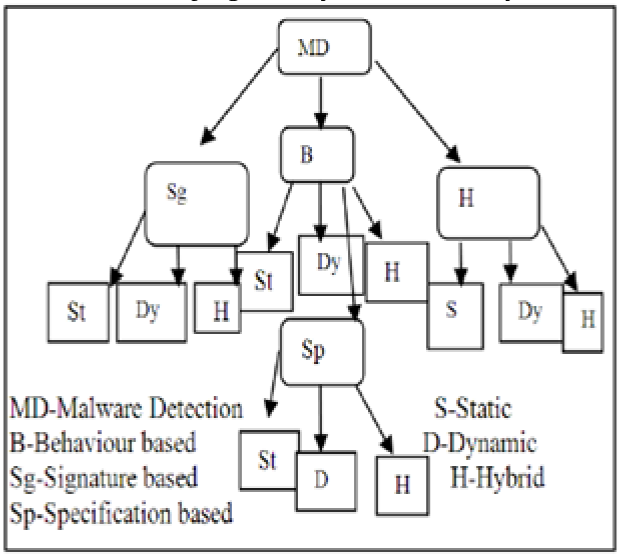

Figure 2: The mechanism to dynamicaly detecting the malware behaviours to OS

4.3 Tracing Intruders

To track intruders within an operating system, one can utilize OS-level information flow as done in King and Chen (2003) and Goel *et al.* (2005). However, a main challenge for leveraging OS level information flow to trace suspicious entities is that, file and process tagging normally leads the entire system to be floated with “suspicious” labels and thus earns too many FPs. To address this issue, we suggest the following two methods to limit the number of tagged files and processes in a single OS while averting malware programs from evading the tracing as much as possible. For tagging files, unlike the approaches in King and Chen (2003) and Goel *et al.* (2005) the schemes of several malware detection and MAC systems (Fraser, 2000; Wang *et al.*, 2008) that trace information flow on OS level, Tracer simply focuses on the tagging of executables while ignoring non executables and directories. This is because an executable signifies the possible execution flow of the process loading it, thus it ought to be deemed as an inactive intruder while a process is treated as an active intruder (Fig. 2). For tagging processes, we observed that the excessive number of tags mainly come from tracing Interposes Communication, i.e., spotting a process as suspicious if it obtains IPC data from a suspicious process. To address this issue, Tracer only tags a process receiving data from dangerous IPCs that can be exploited by a malware program to acquire control of the process to make arbitrary malicious behaviours.

4.4 Restricting Intruders

In order to disable malware programs on a host, the restricting action monitors and blocks intruders’ requests for executing critical malware behaviours listed in Table 2. To follow the principle of complete mediation for building a security protection system, Tracer further restricts two extensive

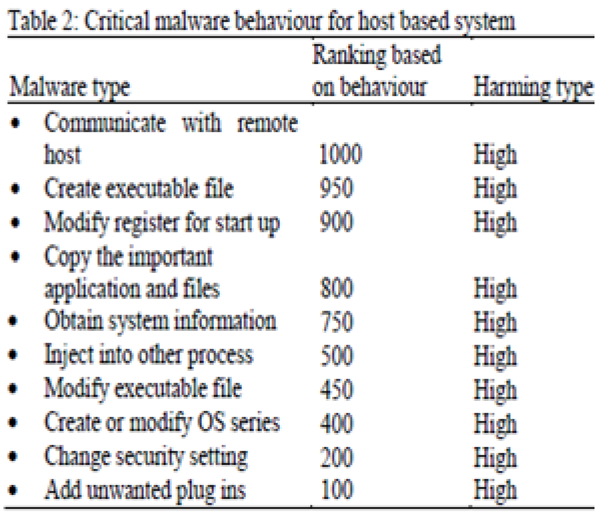

Table 2: Critical malware behaviour for host based system

Malware type	Ranking based on behaviour	Harming type
• Communicate with remote host	1000	High
• Create executable file	950	High
• Modify register for start up	900	High
• Copy the important application and files	800	High
• Obtain system information	750	High
• Inject into other process	500	High
• Modify executable file	450	High
• Create or modify OS series	400	High
• Change security setting	200	High
• Add unwanted plug ins	100	High

behaviours, called generic malware behaviours, to guard security more widely. The first one is “Steal confidential information,” which stands for all illegal reading of confidential information from files and registry entries. The other is “Damage system integrity,” which be an illegal modifications of the files and registry entries that need preserving integrity. All behaviours restricted are listed on the column “restrict” in Table 2. In summary, the restricting action consists of three rules (Fig. 4):

- Restricting critical malware behaviours
- Restricting generic malware behaviours
- Restricting behaviours bypassing Tracer

By mediating all these behaviours, Tracer is able to safeguard system security and prevent a malware program from propagating itself in the system. To be exact, confidentiality is mainly accomplished by blocking the generic behaviour “Steal confidential information;”

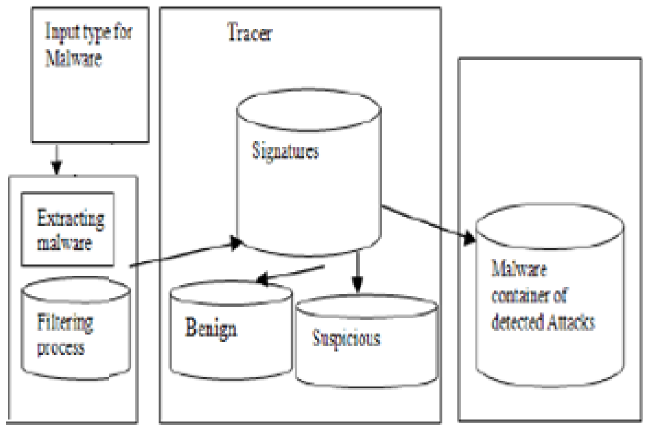

Figure 4: Dynamically restricting and detecting the malware behaviors using EMDT process

integrity is generally protected by blocking the generic behaviour "Damage system integrity;" availability is guarded by blocking the behaviours listed in Table 2 with the capital letter A attached. The algorithm 1 may impose a relatively elevated overhead only on the malware processes that frequently exhibit file copying behaviours but not on benign processes and the suspected processes that are actually gentle.

Algorithm 1:

```
Monitoring the Application Process:
 Input: File to be read,
Buffer reader Process: If (File! = Copying Behaviour)||
(Current Process = = Benign)
Return Operation To Buffer
For (Node of file = Read list of Buffer)
 If (File = = Node)
Statement: Attach the File in the Buffer reader
 Else Statement: copy the File into Node (Stack) for Blocking
Then Copy the file into buffer Return (permit the File to monitor)
```

Algorithm 2 is given below for detection that correlate read and writes operations by comparing buffer contents are more complicated to be circumvented than the other candidate algorithms, e.g., comparing buffer addresses. In the worst case that a malware program successfully circumvents the algorithms, EMDT still can tail it by monitoring related behaviours, e.g., "Create executables," since file-copying behaviours require to create executables.

Algorithm 2:

```
Detecting the Malware Process:
 Input: File to be read,
Buffer writer Process:
If (File ! = Copying Behaviour)||
(Current Process = = suspicious)
 Return Operation To Buffer
 For (Node of file = Read list of Buffer)
If (File==Node)
Statement: Attach the File in the Buffer writer
 Else Statement: Blocking file from Corruption
Then Copy the malware type into bufferwriter
Return (Malware type to buffer)
```

4.5 Dynamic changes of malware behaviours detection process

EMDT can be able to dynamically add in new behaviours to monitor. Behaviour consists of object, operation and parameter. For example, the operation create-file corresponds to two system calls: NtOpenFile and NtCreateFile. In contrast, a single system call might contain more than one operation. In each concerned system call, we set up one or more checkpoints, each of which is dependable for checking the behaviours belonging to the same

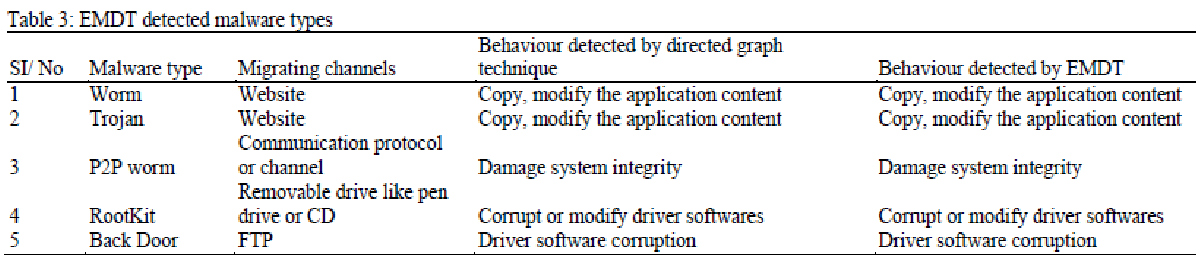

Table 3: EMDT detected malware types

Sl/ No	Malware type	Migrating channels	Behaviour detected by directed graph technique	Behaviour detected by EMDT
1	Worm	Website	Copy, modify the application content	Copy, modify the application content
2	Trojan	Website	Copy, modify the application content	Copy, modify the application content
3	P2P worm	Communication protocol or channel	Damage system integrity	Damage system integrity
4	RootKit	Removable drive like pen drive or CD	Corrupt or modify driver softwares	Corrupt or modify driver softwares
5	Back Door	FTP	Driver software corruption	Driver software corruption

Operation with the support of a modifiable behaviour list in memory.

5. Evaluation Results

Table 3 is given below explains the detailed test results of 5 selected malware samples. We can see that all the malware samples are successfully disabled via the restriction of their malware behaviours. For example, the worm "Worm." downloaded from the local website has the following main steps for function: it first copies itself, i.e., regsv.exe, to hard drive in OS, then runs regsv.exe as a new process, the new process then inserts a value under registry key regsv.exe so that it can be initiated when the system restarts, finally listens at port 113 to accept commands from a remote attacker. On a host without EMDT allowed, all above steps are successfully executed. However, after activating the EMDT protection, the malware behaviour "Copy itself" is blocked, i.e., the malware cannot generate a new copy of itself in the system folder. Consequently, the rest of the behaviours do not emerge anymore because these behaviours depend on the new process launched from the malware's copy. In other words, the worm is disabled.

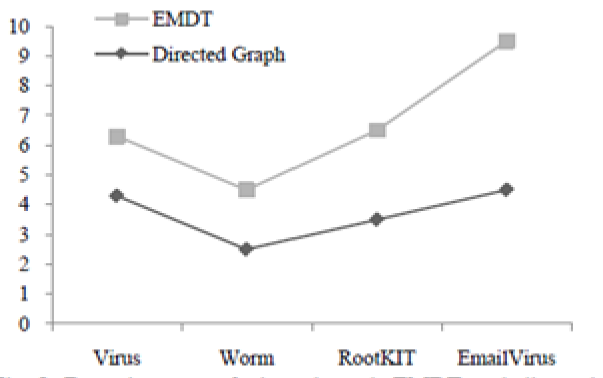

Fig. 5: Detection rate of virus through EMDT and directed graph

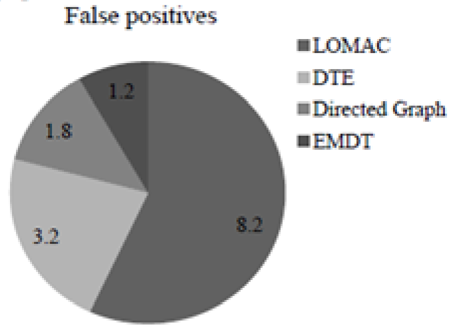

Fig. 6: Comparing the false positive rate of EMDT with other existing technique

fundamental reason is that the antimalware tools identify a suspicious behaviour only supported on the behaviour itself while Tracer further regard as the suspicious label of the process requesting the behaviour (Fig. 5 and 6).

6. Conclusion

In this study, we propose a novel MAC enforcement approach that integrates intrusion detection and tracing to guard against malware in a commercial OS. We have extracted 30 critical malware behaviours and three common malware characteristics for the incompatibility and low usability problems in MAC, which will benefit other researchers in this area. Based on these studies, we propose a novel MAC enforcement approach, called EMDT using Hidden markov model, to disable malware timely without need of malware signatures or other knowledge in progress. The novelty of Tracer design is two- fold. One is to utilize intrusion detection and tracing to regularly configure security labels. EMDT system doesn't restrict the suspected intruders right away but permits them to run as long as feasible except blocking their critical malware behaviours. This design generates a MAC system with good compatibility and usability. We have implemented Tracer in several OS and the evaluation results show that it can successfully guard against a set of real-world malware programs, including unknown malware programs, with much small FP rate than that of commercial antimalware techniques. In future we are going to initiate this study for a large web server runs the application front-end logic and data are outsourced to a database or file server where there is increase in application and data complexity.

References

[1] Badger, L., D.F. Sterne, D.L. Sherman, K.M. Walker and S.A. Haghighat, 1995. Practical domain and type enforcement for UNIX. Proceeding of the IEEE Symposium on Security and Privacy (S&P), pp: 66-77.
[2] Fraser, T., 2000. LOMAC: Low water-mark integrity protection for COTS environments. Proceeding of the IEEE Symposium on Security and Privacy (SP' 00), pp: 230-245.
[3] Goel, A., K. Po, K. Farhadi, Z. Li and E. Lara, 2005. The taser intrusion recovery system. Proceeding of the 20th ACM Symposium on Operating Systems Principles (SOSP '05), pp: 163-176.
[4] Kaspersky Lab, 2012. Retrieved from: http://www.kaspersky.com/.
[5] King, S.T. and P.M. Chen, 2003. Backtracking intrusions. Proceeding of the 19th ACM Symposium on Operating Systems Principles (SOSP '03), pp: 223-236.
[6] Kirda, E., C. Kruegel, V.G. Banks and R.A. Kemmerer, 2006. Behavior-based spyware detection. Proceeding of the 15th Conference on USENIX Security Symposium (USENIX-SS '06).
[7] Li, N., Z. Mao and H. Chen, 2007. Usable mandatory integrity protection for operating systems. Proceeding of the IEEE Symposium on Security and Privacy (SP '07), pp: 164-178.
[8] Martignoni, L., E. Stinson, M. Fredrikson, S. Jha and J.C. Mitchell, 2008. A layered architecture for detecting malicious behaviors. Proceeding of the 11th International Symposium on Recent Advances in Intrusion Detection, pp: 78-97.
[9] Microsoft, 2012. Mandatory Integrity Control. Retrieved from:http://en.wikipedia.org/wiki/MandatoryIntegrityControl.
[10] Nachenberg, C., 2002. Behaviors Blocking: The Next Step in Anti-Virus Protection. Retrieved from: http://www.securityfocus.com/infocus/1557.
[11] Shan, Z., X. Wang and T. Chiueh, 2011. Tracer: Enforcing mandatory access control in commodity os with the support of light-weight intrusion detection and tracing. Proceeding of the 6th ACM Symposium on Information, Computer and Communication Security, pp: 135-144.
[12] Viper Inc., 2012. Retrieved from: http://www. vipre. com/vipre/, 2012.
[13] Wang, X., Z. Li, J.Y. Choi and N. Li, 2008. PRECIP: Towards practical and retrofittable confidential information protection. Proceeding of the 15th Network and Distributed System Security Symposium.

Author Profile

Dokuparthi Prasanthi received the MCA degree from Acharya Nagarjuna University Vijayawada in 2012 and pursuing M.tech degree in Computer science and Engineering from Anurag Group of Institutions (Formerly CVSR College of Engineering) JNTU Hyderabad, India.

V. Rama Krishna working as Assistant Professor in Computer Science and Engineering from Anurag Group of Institutions (Formerly CVSR College of Engineering) JNTU Hyderabad, India.