IDENTIFICATION OF HETEROGENEOUS MALICIOUS NODES IN THE CRUCIAL DYNAMIC ENVIRONMENT FOR PROTECTING LOCAL AND REMOTE SYSTEMS

[1]A.EDWINROBERT, [2]M.HEMALATHA

[1,2]Department of Computer Science, Karpagam University, Coimbatore, India
Email: [1]edrt.edwin@gmail.com, [2]csresearchhema@gmail.com

Abstract: Computer Software applications are a heterogeneous emerging trend, third party software applications are increasing more and more today. Due to the enhancement in existing software applications, it must be integrated with COTS [Commercial off the Shelf] products. Sharing of information takes place between each Corporate for doing their concerns routine work for their development. As sharing of information takes place in their routine work, there is a chance for the intruders getting into the official Websites and also in the concern's database. Also the intruder injects different categories of malicious code through programs written in HLL [High Level Languages].As a consequence of this the kernel of the operating system, system registry and various disk partitions get crashed. This leads to memory duplication and inconsistency in maintaining official data. The proposed approaches such as FECB [Fish Eye Code Behavior], EMDT [Efficient Malware Detector and Tracer] and MCBA [Malicious Code Behavior Analysis] are the novel approaches that are implemented to identify the number of Malicious files in Local host and Packets in the Remote host and heal the malware. Detecting and tracing the Malware behavior based on virus signatures and finally the Malicious packets [encrypted or decrypted] delivered to and fro in the network can be identified, traced and healed automatically by the method of intrusion detection algorithm. From the above explained techniques the behavior of malicious codes can be broadly divided into Benign and Suspicious files respectively. These two files are segregated and stored separately in the buffer to count the loss of packets, calculate Detection Rate and for estimation of the fitness value.

Keywords: Intruders, Malicious files, Packets, Benign and Suspicious.

I. INTRODUCTION

Software applications are more spread today and there is a usage of third party software application that spreads the Malicious Code in any part of a software system or script that is intended to cause undesired effects, security breaches or damage to a system. Malicious Code is an application security threat that cannot be efficiently controlled by conventional antivirus software alone. Malicious Code describes a broad category of system security terms that includes attack scripts, viruses, worms, Trojan horses, backdoors, and malicious active content.

Malicious Code can take the form of:

- Java Applets
- ActiveX Controls
- Scripting languages
- Browser plug-ins
- Pushed content

Once inside your environment Malicious Code can enter network drives and propagate. Malicious Code can also cause network and mail server overload by sending email messages, stealing data and passwords, deleting document files, email files or passwords, and even re-formatting hard drives. The Anti-virus benchmark pointed out that there is no silver bullet to prevent spy ware, User awareness still remains the best way to prevent spy ware installation. An Error of commission or omission in a Software Application (SA) may allow protection mechanisms to be bypassed. Developer's awareness is also necessary, as several security flaws were discovered recently in banking applications that could take entire control of their devices, as all applications are running with root access.

II. FESR APPROACH

Obviously in FESR(Fish Eye State Routing) Algorithm,[11]spreading of malicious packet in the network via shortest path can be identified and minimized, but no chance of healing it entirely from the network or from the system from where it is affected. The heterogeneous nodes generated from the network may affect the overall system increase packet duplication but the behavior of the nodes cannot be predicted without segregation of the packets known to the user which causes various attacks over the network.

III. SLOW PATCHING

The software is designed to fix problems with, or update a computer program for the manufacturers to distribute embedded software devices, When a security update is required it is up to the manufacturers to provide it to their customers. So software applications distribution requires more time than for competitors like Apple. This slow patching process leads to a rise of Jailbreak devices, as users want access to the latest releases. As a consequence, the restrictions are completely bypassed and users are more exposed to spyware an application claiming to be the latest version of a famous Twitter client, which actually runs spyware in the background and uploads all private data to the attacker.

IV. FALSE POSITIVE AND NEGATIVE RATES

"False Positive" when an uninfected object (file, sector or system memory) triggers the anti-virus program. The opposite term "False Negative" means that an infected object arrived is detected. Tao etal proposed HLA [Homomorphic Linear Authenticator] Architecture does not give the information about how many packet is loosed, Gerald Tesauro et al, discovers the "False Positive" files when an uninfected object (file, sector or system memory) triggers the anti-virus program and "False Negative" means that an infected object arrived undetected uses On Demand Scanning. This technique also allows more number of infected processes increases the False Positive and False Negative rates

False positive rate = (No .of True Positive / (No. of True positives + No. of false Negatives))

Domain Type Entity:

Lee Badger et al. proposed DTE which groups processes and files into domains and types, respectively, and controls accesses between domains and types. This clearly gives the unique identification of malicious files but doesn't identifies encrypted malicious file. This technique does not block file which tend to spread more malicious files, which increases False Positive Rate leading to corruption of memory.

V. PROPOSED APPROACHES

FECB Approach: A Reverse engineering technique is proposed which is based on FECB (Fish Eye Code Behavior) Technique which is the enhancement of Fish Eye State Routing algorithm.

Fish Eye State Code Behavior Algorithm (FECB) is a proactive or table driven filtering algorithm. FECB is based on the traditional link state routing algorithm. Each and every node collects the information about the structure of the vertices from the neighboring nodes and calculates the filtering table. It then disseminates the information locally to the neighbouring nodes.

VI. FECB METHOD

- Initialize the structure of Node List, Host, Neighbour Node, Queue and Distance.
- Check if node is empty by ensuring the queue.
- Assign and link the neigbour node from source.
- Check the malicious nodes by comparing the nodes.
- Identify the Node weight in the Linked List.
- Perform transaction and record the transaction time for shortest path using disjktra's algorithm
- Minimize the number of malicious nodes by stopping the selfish nodes [spy].

The simplest method of this type is to let the interpolating function be a constant function (a polynomial of degree zero) that passes through the point $((a+b)/2, f((a+b)/2))$. Let a represent malware packets and b represent benign packets. This can be given by rectangular rule as given below

$$\int_a^b f(x)\,dx \approx (b-a)\, f\left(\frac{a+b}{2}\right).$$

Table 1. Results obtained from FECB

<table>
<tr><td rowspan="2">Parameters</td><td rowspan="2">FECB</td><td colspan="2">No of Nodes</td></tr>
<tr><td>Malware</td><td>Benign</td></tr>
<tr><td>Detected Malicious code</td><td>10</td><td rowspan="3">4</td><td rowspan="3">14</td></tr>
<tr><td>Healed</td><td>5</td></tr>
<tr><td>Packet Duplication</td><td>5</td></tr>
</table>

Figure.1 Propagation of Malware and Benign Packets.

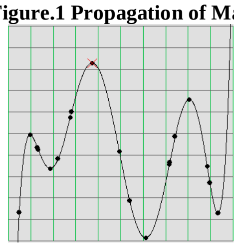
Figure .a

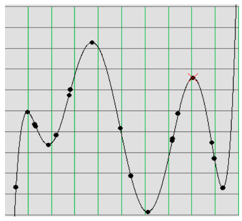
Figure.b

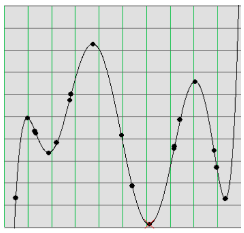
Figure .c

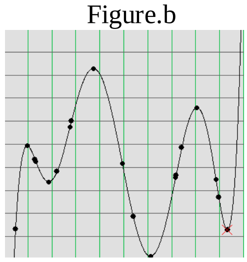
Figure .d

In the above figures a crossed line at each maximum and minimum level indicates propagation of malicious node that can be found when there is a sudden change in the dissemination of the total nodes observed for detecting and healing.

By applying the trapezoidal rule the interpolating function may be an affine function (a polynomial of degree 1) that passes through the malware at various destination in the local and remote hosts $(a, f(a))$ and $(b, f(b))$.

$$\int_a^b f(x)\,dx \approx (b-a)\,\frac{f(a)+f(b)}{2}.$$

For either one of these rules, we can make a more accurate approximation by breaking up the chunk of total nodes took for identifying malware behavior $[a, b]$ into some number n of subintervals, computing an approximation for each subinterval, then adding up all the results. This is called a *composite rule*, *extended rule*, or *iterated rule*. For example, the composite trapezoidal rule can be stated as

$$\int_a^b f(x)\,dx \approx \frac{b-a}{n}\left(\frac{f(a)}{2}+\sum_{k=1}^{n-1}\left(f\left(a+k\frac{b-a}{n}\right)\right)+\frac{f(b)}{2}\right)$$

where the subintervals of the benign and suspicious files have the form $[k\,h,\ (k+1)\,h]$, with $h = (b-a)/n$ and $k = 0, 1, 2, ..., n-1$.

VII. EMDT AND MCBA ARCHITECTURE

The proposed architecture is the Efficient Malware Detection and Tracer design (EMDT) using intrusion detection algorithm that can identify all potential intruders and Tracer technique reduces False Positive Rate and focuses on the tagging of executables while ignoring non executables and directories to overcome the problems found in Tao etal and Gerald Tesauro et al approach which does not address the problem of how to decrease False Positive and False Negative rates. EMDT using Hidden Markov model, incorporates intrusion detection and tracing in a commercial operating system which leverages efficient coding and authentication schemes with our proposed approach conceptually consists of three actions: detecting, tracing and restricting suspected intruders[10].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 [Mandatory Access Control] system in practice.

VIII. MCBA ANALYSIS

It detects encrypted type Malicious file which is the extension of Lee Badger et al's approach [5].It does not address the problem of how to identify, detect and heal the type of file that spreads through the network in an encrypted format.

It is assumed that the value of a function f defined on $[a, b]$ is known at equally identified malwares x_i, for $i = 0, \ldots, n$, where $x_0 = a$ and $x_n = b$. There are two types of Newton–Cotes formulae, the "closed" type which uses the function value at all nodes, and the "open" type which does not use the function values at the endpoints. The closed Newton–Cotes formula of degree n is stated as

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n} w_i\, f(x_i)$$

where $x_i = w_i + x_0$, with w (called the *encrypted malware*) equal to $(x_n - x_0)/n = (b - a)/n$. The w_i are called *weights* assigned to the malwares belong to this category.

This means they directly depend only on the x_i and not on the function f.Indirectly, the weights do depend on f. Let $L(x)$ be the interpolation polynomial in the Lagrange form for the given data points $(x_0, f(x_0)\square), \ldots, (x_n, f(x_n)\square)$, then

$$\int_a^b f(x)\,dx \approx \int_a^b L(x)\,dx = \int_a^b \left(\sum_{i=0}^{n} f(x_i)\,l_i(x)\right)dx = \sum_{i=0}^{n} f(x_i)\underbrace{\int_a^b l_i(x)\,dx}_{w_i}.$$

The open Newton–Cotes formula of degree n is stated as

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n-1} w_i\, f(x_i).$$

The weights are found in a manner similar to the closed formula.

Table :2 Results of Encrypted Malwares by Newton – Cotes

Malware types	Weights	End Points	Type	Method
Worm	20	WindowsXP	Closed	MCBA
Trojan	5	WindowsNT	Open	EMDT
P2P worm	7	Windows 7	Open	EMDT
RootKit	3	Windows 2000	Closed	MCBA
Boot Sector	10	Windows 98	Closed	MCBA

Epidemics Dynamics of MCBA: In this analysis we have simulated two approaches one is Secured Memory and the other is Unsecured Memory. In the First method, a trusted component (e.g., the MCBA) applies pattern recognition techniques to validate the integrity of the application's memory format during execution. In the Second method, the OS from root kits by securely storing a cloned image of the kernel at boot time, the trusted VMM [Virtual Memory Monitor] creates a copy of the kernel in a portion of memory that is inaccessible to the guest OS.

To integrate the Analysing function f we apply the following transformation

$$\int_0^\infty f(x)\,dx = \int_0^\infty f(x)e^x e^{-x}dx = \int_0^\infty g(x)\,e^{-x}dx$$

where $g(x) := e^x f(x)$. For the last integral one then uses Gauss-Laguerre quadrature. Note, that while this approach works from an analytical perspective More generally, one can also consider integrands that have a known x^α power-law singularity

$$\int_0^{+\infty} x^\alpha e^{-x} f(x)\,dx.$$

at x=0, for some real number $\alpha > -1$, leading to integrals of the form: This allows one to efficiently evaluate such integrals for polynomial or smooth $f(x)$ even when α is not an integer.

Our design offers these applications a resilient execution environment that protects the application from external threats.

$$\text{Vulnerability} = \frac{\alpha * \beta \sqrt[4]{d * 4}}{\alpha} \ \text{Strength}$$

Where α is the external virus from the network and β is corrupted virus due to internal virus affecting the kernel. Also d is Corrupted network data.

Table 3: Vulnerable Data obtained from Local and Remote Host

Malware types	Category	End Points	Method
Communicate with remote host	Corrupted Network Data	Windows NT	FECB
Obtain system information	External virus	Windows7	EMDT
Add unwanted plug ins	External Virus	Windows Server2003	MCBA
Inject into other process	Internal Virus	Wndows2000	MCBA
Create or modify OS series	Internal Virus	Windows98	VMM

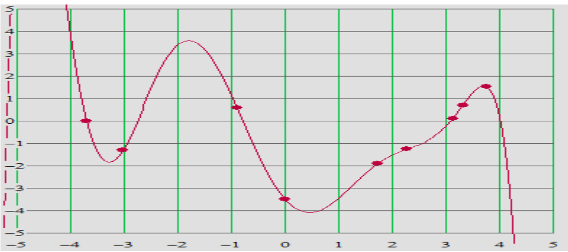

Figure.2 Corrupted data of local and remote host

If the critical application sends or receives data across a network interface, the guest OS kernel has an opportunity to modify this data in transit. The corrupted nodes figure.2 is represented in dots at various segments in local and remote host. Note that the node at the maximum level denotes the affect of remote host has more weight due to its failure rate which is more whereas the nodes at the local host lies at the minimum level because it failure rate is less and less weight is assigned. It displays various categories of viruses based on its behavior.

CONCLUSION AND FUTURE ENHANCEMENT

In this work, the researcher proposed three novel approaches have been suggested for protecting applications against malwares that crashes the entire system. All these approaches have identified the degree of malware strength and its associative behaviors at the local and remote host. Based on that we have identified the fitness value, node behaviors and vulnerability strength have been identified.

In future enhancement we need to implement the same needs to be implemented in Neural networks and also in Linux based systems.

REFERENCES

1. Z. Wang, X. Jiang, W. Cui, and P. Ning, “Countering Kernel Rootkits with Lightweight Hook Protection,” Proc. 16th ACM Conf. Computer and Comm. Security (CCS), pp. 545-554, 2009.
2. A. Seshadri, M. Luk, N. Qu, and A. Perrig, “SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity Oses,” Proc. 21st ACM SIGOPS Symp. Operating System
3. Gerald Tesauro, Jeffrey O. Kephart, Gregory B.Sorkin, "Neural Network for ComputerVirus Recognition," IEEE Expert, vol. 11, no. 4, pp. 5-6, Aug 1996.
4. M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario," Automated classification and analysis of internet malware", In Proceedings of the 10thSymposium on Recent Advances in Intrusion Detection (RAID’07), (2007), pp 178–197.
5. 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.
6. 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.
7. 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.
8. A. Edwin Robert and M.Hemalatha,G.Mohanasagam and N.Sasirekha, “Reverse Engineering for Malicious Code Behavior Analysis using Virtual Security Patching”, “International Journal of Computer Applications (0975 – 8887),Volume 26– No.4, July 2011
9. A. Edwin Robert and M.Hemalatha, “Behavioral and performance analysis model for Malware detection techniques”, International Journal of Computer Engineering and Technology (IJCET), ISSN 0976 – 6375(Online) Volume 4, Issue 1, Page No:141-151,January- February (2013)
10. A. Edwin Robert and M. Hemalatha, “Efficient Malware Detection and Tracer Design for Operating System”, “Research Journal of Applied Sciences, Engineering and Technology”, 6(11): 2052-2060, 2013 ISSN: 2040-7459.
11. Sunil Kumar Senapati, Pabithra Mohan Khilar , “Securing FESR against Data packet dropping by Malicious Nodes in MANET”, “International Journal of Computer Engineering, technology and sciences”(IJ-CA-ETS), Volume 1 ,Issue 2,PageNo: 440-442, ISSN: 0974-3596 ,April 2009
12. R. Srinivasan, Protecting Anti-Virus Software Under Viral Attacks, Master Degree of Science, Arizona State University (2007).
13. J. Cock, Computer Viruses and Malware, Springer (2006)
14. E. Skoudis and L. Zeltser, Malware: Fighting Malicious Code, Prentice Hall (2003)
15. P. Szor, The Art of Computer Virus Research and Defense. Addison Wesley, (2005)
16. http://www.veracode.com/security/malicious-code
17. http://en.wikipedia.org/wiki/Gauss-Laguerre_quadrature
18. http://en.wikipedia.org/wiki/Numerical_integration
19. http://en.wikipedia.org/wiki/Newton%E2%80%93Cotes_formulas

★★★