JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol. 10, Issue.1, January 2022 E-ISSN: 2347-2693

SecurityTAG'S

Boddepalli Kiran Kumar[1*], Korla Swaroopa[2]

[1,2]Associate Professor, Dept. of CSE, Aditya College of Engineering, Surampalem Affiliated to Jawaharlal Nehru Technological University Kakinada, East Godavari, India

**Corresponding Author: kirankumar_cse@acoe.edu.in, Tel.: +91-9030029990*

DOI: https://doi.org/10.26438/ijcse/v10i1.4144 | Available online at: www.ijcseonline.org

Received: 21/Jan/2022, Accepted: 25/Jan/2022, Published: 31/Jan/2022

***Abstract*—** In this paper, initially we describe the present antivirus in aspects like –memory[1] they are consuming, and how efficiently they are protecting the system. In the next section of this paper we briefly discuss the design methodologies that are practiced presently, their drawbacks and limitations. Finally we describe an effective design methodology which uses SecurityTAG to protect the system. SecurityTAG is generated by the SecurityTAG generator which takes some parameters as inputs and gives the SecurityTAG as the output. This gives better protection against any virus and detection of infected files is very easy and effective.

***Keywords*—**SecurityTAG, generator, key, virus

I. INTRODUCTION

PRESENT ANTIVIRUS PACKAGES

All software, including ready-to-wear software, should be sufficiently reliable and secure in delivering the service that is promised of them. There are various ways in which this reliability and security can be achieved in practice, such as the use of various validation and verification techniques in the software development phases, statistical testing of the final product before delivery, issuance of patches and service releases for the product in operation, as well as the use of software fault/intrusion tolerance techniques. The fault tolerance techniques can range from simple "wrappers" of the software components [2] to the use of diverse software products in a fault-tolerant system [3]. This latter strategy of implementing fault tolerance was historically considered prohibitively expensive, due to the need for development of multiple bespoke software versions. However, the wide proliferation of ready-to-wear software for various applications has made the use of software diversity an affordable option for fault tolerance against either malicious or non-malicious faults.

In this section we aim at describing the present antivirus package, collected from various resources available on the web and experimental studies we did on various packages in the following aspects.

i. Memory space utilized by the AV package.
ii. Efficiency of the AV package.

Memory space utilized by the AV packages:
Here we give you information about 29 AV package. All are free- ware and shareware.

Table 1: Memory usage table[4]

AV package name	Memory usage during idle(KB)	Memory usage during scan(KB)
PC Tools AntiVirus Free Edition 4.0.0.26	5680	23948
Norton Antivirus 2009	6000	51312
Kaspersky Antivirus 2009 v8.0.0.357	6565	50892
Spyware Terminator v2.2.3.444	7624	64292
Quick Heal AntiVirus Plus 2008 v9.50	151660	201660
BitDefender 10 Free Edition	9668	42492
Avira AntiVir Personal 8	10072	70072
Kaspersky Internet Security 2009 v8.0.0.3578	16180	44216
Rising Antivirus 20.44	16252	59704
DriveSentry 3.1	16992	18504
avast! 4 Home Edition	23100	63416
CA Anti-Virus 2008	37756	47556
AVG Anti-Virus Free Edition 8.0.138	38244	88244
ESET NOD32 Antivirus 3.0.669	40364	46012
ESET Smart Security 3.0.669	42640	39284

Trend Micro AntiVirus plus AntiSpyware 2008	42680	63704
ZoneAlarm Antivirus 7.0.483	52772	97200
BitDefender Antivirus 2008 v11	56440	73720
Comodo AntiVirus 2.0.17.58	56668	76668
Trend Micro Internet Security 2008	59140	79688
F-Secure Anti-Virus 2008	61972	178824
Moon Secure Antivirus 2.2.2.163	71528	71528
Panda Antivirus 2008	76344	101416
McAfee VirusScan Plus 2008	87632	140484
Norman Antivirus & Antispyware	99856	201660
Windows One LiveCare 2.5	102868	107868
eEye Blink 4.04 Personal Edition	117972	131412
G DATA AntiVirus 2008	130544	175176

In the above table we have collected data from the freeware and shareware versions only as they are available freely but the professional versions may consume more memory compare to the free versions because of the extra features and protection system.

If you consider a PC, that now a days used will contain minimum of one GB of memory. Out this one GB, depending on the operating system used the memory consumption varies from 20% to 40% of the total memory. Which is very useful in providing good user interface and easy to work environment. This 20% to 40% is considerable for operating system-with out which we cannot use it. But coming to the AV package, here we are running a program which is not having any personnel use and only for the protection of our system from threats.

Efficiency of the AV package

Generally the time complexity of these will directly proportional to the memory space they are using and in some exceptional cases it depends on the code i.e., if it has so many recursive functions and loops, though it is a small program its time complexity is very high. If we consider the memory as the measure to compute the proportionality relational equation. complexity then we can find out the time complexity by using the If you observe the memory consumed in above table, in the idle time only they are resident in the considerable amount of memory which can degrade the performance of the system.

The other thing we have to observe in the AV package is, how far the AV package is defending the new viruses. Almost all the AV packages are able to find out the existing and known virus only. If the system is effected with the new virus then they are not able to detect them. This is the main drawback of the all the existing AV packages. The efficiency of different AV packages can be obtained from various web sites in all the aspects.

II. EXISTING METHODOLOGY

Antivirus is a good way to protect against viruses, but we still have disadvantages. First of all, an antivirus used signature in his database that means that he is unable to discover new attacks; this can be remedied by updating periodically the database. Beside antivirus stays helpless against different kinds of attacks like hijacking, Denial of Service, and other. That is why we need other software's, along with the use of antivirus

Mostly three techniques were used in the AV package:

i. Pattern matching technique.
ii Checksum method
iii Signature analysis

Pattern matching technique:

In this method[5] the previously known virus patterns are matched while scanning. If any similar pattern occurs the it will remove or modify it depending upon the level of virus effect.

A more elegant and more transparent solution to the pattern matching scan is a memory resident piece of software, which checks for viruses again by pattern matching each time an attempt to execute a program is made, or when a new removable disk is introduced to the system. This method is effective in stopping the spread of viruses, and has little performance overhead relative to the loading time of the program

Checksum method:

Check summing [5] is a method based on calculating CRC [Cyclic Redundancy Check] checksums and is a modification of signature analysis. The method was developed to overcome the main disadvantages of the signature method, large databases and frequent false alarms. Checksumming accounts for not only the search string [or, to be more precise, a checksum for the string] but the location of the string in the body of a malicious program. The location is used to calculate the checksums for the entire file. Thus, instead of a 10-12 byte search string [and this is a minimum size], the checksum takes four bytes and the location data also take four bytes. However, checksumming is more time consuming than signature analysis.

Signature analysis

A signature is a unique sequence of bytes that is specific to a piece of malicious code. Signature analysis, or a modification of it, was [and remains] one of the first methods used in anti-virus engines to detect viruses and other malware. Evident advantages of this method are its high speed [especially with the use of special algorithms] and the possibility of detecting several threats using one signature. On the other hand, a serious disadvantage of this method is that for reliable detection of malicious code, the signature must be large, at least 22-40 bytes [anti-virus producers usually use longer signatures, of up to 64 bytes, to improve the detection level]. So the size of the anti-virus database also increases. Another challenge to this method is that much contemporary malware is written in high level languages such as C++, Delphi or Visual Basic. These programs contain fragments of code that do not change [the so-called Run Time library]. If a wrong signature is used, this leads to false alarms, where a clean file is reported to be infected. The false alarm problem can be solved by using extremely large signatures, or by restricting detection to certain data areas like relocation tables or text strings, which is undesirable.

Techniques for detecting polymorphic viruses

Self-encryption and polymorphism are used in most types of virus to maximize the difficulty of their detection. Polymorphic viruses are extremely difficult to detect because they do not have signatures, that is, there's no constant fragment of virus specific code. In most cases, two samples of the same polymorphic virus do not have a single coinciding fragment.

There are many kinds of polymorphic viruses, from boot and DOS file viruses to Windows viruses, macro and script viruses. Polymorphic 'envelopes' are also used to hide Trojan programs.

Viruses are called polymorphic if their body is self-changing during replication to avoid the presence of any constant search strings. Polymorphic viruses can not be detected [or can be detected only with great difficulty] using so-called virus signatures or masks, sequences of unchanging virus-specific code. Polymorphism is achieved by encrypting the main code of the virus with non-constant keys containing random sets of decryption commands, or by changing the executable virus code. There are also other rather exotic examples of polymorphism. For example the DOS virus Bomber is not encrypted, but the sequence of instructions, passing control to the body of the virus, is completely polymorphic.

It is problematic to use signatures [sometimes called 'search strings'], as outlined above, to detect polymorphic viruses. Since the code changes with each infection, it becomes impossible to select the correct signature. Even a very large signature can not be used to identify an encrypted virus uniquely without giving false alarms. It's not difficult to see why. The polymorphic virus encrypts its body, converting the virus code into a variable. And variable code can not be selected for a signature.

So for detection of polymorphic viruses, additional techniques must be used.

Reduced masks

If the encryption algorithm used by the virus is not sufficiently advanced, it's possible to use elements within the encrypted body of the virus to take the encryption key out of the equation and obtain static code. The signature, or mask, can then be taken from the resulting static code.

Known plaintext cryptanalysis

Known plaintext cryptanalysis, another method for dealing with polymorphic viruses, works like this. Using the known original virus code and the known encrypted code [or suspicious code that looks like an encrypted virus body], the engine reconstructs the keys and the algorithm of the decrypting program. The engine then decodes the encrypted virus body by applying this algorithm to the encoded fragment. The use of a system of equations to decode an encrypted virus body is similar to the classical cryptographic problem of decoding an encoded text without keys.

However, there are two key differences. First, most of the data required for the solution is known. Second, the solution must be solved using available RAM and with limited time. In general, this method is less time consuming and uses a smaller amount of memory than emulation of virus instructions [see below]. However, this solution implies constructing a system of equations and it becomes rather complicated. The main problem is the mathematical analysis of the equation or the system of equations constructed.

III.. PROPOSED METHOD: SECURITYTAGS

We have proposed a method which can efficiently detect the new virus and it will take considerably less memory and computational time when compared to the previous AV packages. One of the major advantage of this SecurityTAG method is tracing the effected file is very easy, which we have demonstrated in the following sections.

What is a SecurityTAG?

Security TAG is a unique identification number generated by the Security TAG generator to each and every file, directory and system in a tree hierarchical order. The following figure will illustrate this clearly.

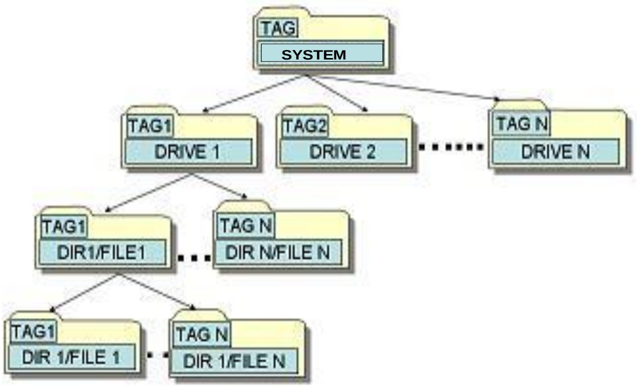

Figure:1 Hierarchical order of SecurityTAG generation

Security TAG generator:
Security TAG generator will generate a unique security TAG. This algorithm takes the parameters like size of the file, file extension, file checksum, and parity as inputs for generation of the Security TAG.

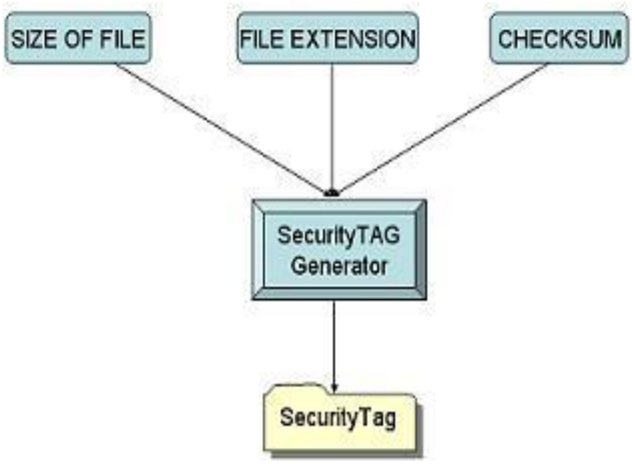

Figure: 2 SecurityTAG generator

As we are using the hierarchical order in generation of the tags, the whole system will have a unique Security TAG which is to be monitored continuously. These tags will be changing upon the modifications the files or directories, which will again lead to the change of the whole system SecurityTAG.

Detecting the threats and tracing the effected file:
First of all we will discuss the main characteristics of any virus

i. Replication of data.
ii. Modification of data.

These are the two main characteristics of any virus which will lead to degrade the performance of system and sometimes lead to crashing the system.

Using our SecurityTAG we can easily find any type of modification of the file as we are considering all the parameters of the file. Suppose any file is affected by some virus then ultimate target of the virus is to change the data or replicate it. If any change in the data then there will be a change in the corresponding SecurityTAG which will lead to the change of the SecurityTAG of the corresponding directory, that will lead to the change of the SecurityTAG of the system which is monitored continuously by monitoring engine.

Tracing of the virus is very easy as we are using the tree structured SecurityTAG generation. The following figure will demonstrate the procedure for tracing infected drive/file, which would be very easy when compared to previous techniques

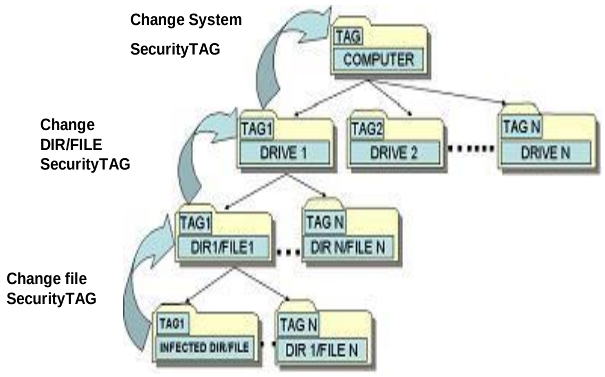

Figure:3 Tracing the infected file

Coming to the performance aspects of this SecurityTAG protection method, in the idle mode it is only monitoring the system SecurityTAG only which will take hardly 512 KB of memory and in the back end the generating algorithm is observing and modify SecurityTAG of the changed files which are used by the user and new tag is generated after the changes made by the user. This back end observing and modifying algorithm will take hardly 2 to 3 MB. So in idle this method is using hardly 3.5 MB. Initially it will consume some time to generate SecurityTAG for all the files in the system. Later on the monitor and generator will run as back end process continuously to protect the system.

Note: We are implementing the above method which may lead to some further minor modifications in the proposed algorithm.

IV. FUTURE WORK

To the above mentioned method if we add the known virus definitions and detecting them with the method of pattern matching and checksum methods then we can develop an effective AV package which gives better performance in all the aspects that are mentioned above.

REFERENCES

[1] Memory- memory refers to the main memory or physical memory.
[2] van der Meulen, M.J.P., et al. *Protective Wrapping of Off-the-Shelf Components*. in *the 4th International Conference on COTS-Based Software Systems (ICCBSS '05)*. Bilbao, Spain: Springer. 2005.
[3] Strigini, L., *Fault Tolerance Against Design Faults*, in *Dependable Computing Systems: Paradigms, Performance Issues, and Applications*, H. Diab and A. Zomaya, Editors. J. Wiley & Sons. p. 213-241, 2005.
[4] http://www.raymond.cc/blog/archives/2008/07/11/ -free-antivirus-is-the-lightest-on-system-memory-usage/
[5] Fred Cohen, “Computer Viruses”, *Computers €4 Security* 0, 22-35.
[6] *Computers* & *Security 7*, Specid Issue on Viruses.
[7] Marvin Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall.