Study of an Anti-Virus Framework

Ming Zhang[1,2] and Wei Chen[1]
[1]School of Information Engineering, Wuhan University of Technology, Wuhan, China
[2]Linyi University, Shandong, China

***Abstract*—Commercial anti-virus software has been extended to anti-virus engine matching signature works, this method can accurately detect the known viruses. The study of anti-virus technology of the foreign scholars focuses on the following aspects, improving the ability of antivirus software to detect unknown viruses, solving the deformation of polymorphic viruses against signature matching problem, and improving the efficiency of the signature matching algorithm. In order to prevent the virus cause damage, select the VMWare configuration of the Windows xp virtual machine as a test environment. Selecting gray pigeons virus spread widely for experiment.**

Keywords-anti-virus framework; intelligent feature codes; anti-virus; aotm-detection activities

I. INTRODUCTION

Commercial anti-virus software has been extended to anti-virus engine matching signature works, this method can accurately detect the known viruses. The study of anti-virus technology of the foreign scholars focuses on the following aspects, improving the ability of antivirus software to detect unknown viruses, solving the deformation of polymorphic viruses against signature matching problem, and improving the efficiency of the signature matching algorithm. Domestic scholars focus on the calculation model, the computer virus propagation model and evolution model, and other areas of the theory]. The research of antivirus engine makes the industry protect the core competitiveness. However, antivirus engine as antivirus software core modules in solving various kinds of problems of the anti-virus technology is involved in the design of the engine, it is hard to convert the theoretical research into practical applications. Therefore, the scholars need to pay attention to the study of the anti-virus engine. In literature [9], the disadvantage of the traditional antivirus engine is analyzed, which indicates that the traditional antivirus engine using fixed detection logic has its defects of lack of atomic testing activity, and the antivirus engine and signature data database has strong coupling relationship. In addition, the antivirus engine test control coupling between the two modules is higher, it is hard to update the detection module timely. At present, the computer virus makers have been fused variety of the attack techniques, constantly created behavior unique virus. Without timely extension of detection module, antivirus vendors often face the embarrassing issue for killing tools. Another aspect, the anti-virus technology showed the trend of the development of the embedded direction. Network equipment maker provides the antivirus module in the router, such as, H3C provides ASM (Anti-Virus Security Module) antivirus module. ASM is installed on the H3C router for on-line detection of through traffic and filters the information carried in viruses, worms, and Trojan virus. Because of hardening the antivirus engine code to the embedding device adds to the cost of the upgrade code, the requirement for antivirus engine is more flexible expansibility. In order to improve the scalability of anti-virus software, firstly, we make decoupling of the engine with signature database. Secondly, we decompose each test module of antivirus engine and design a fine-grained detection module. Literature mainly points out that the existing disadvantages of antivirus engine design and proves that the detection module has decomposability theoretically, but without the introduction of key technology and performance analysis.

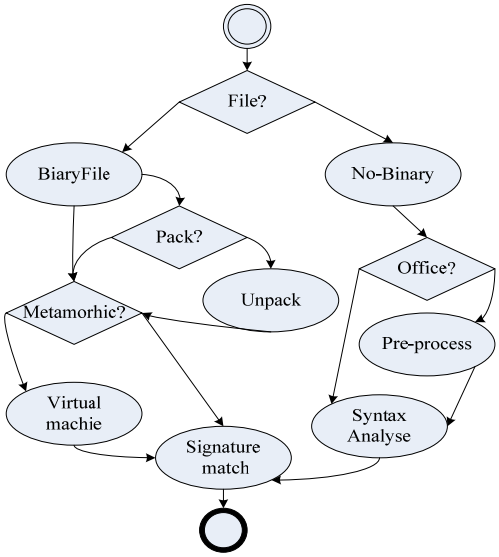

FIGURE I. FILE TYPE TESTING ACTIVITY DIAGRAM.

As shown in figure 1, let determine file type of activity is P1, unpack activities is P2, virtual machine testing activities is S1, signature matching activities is S2, sequence L1 = {P1, P2, S1, S2} denote a detection logic, L1 describe a complete process. As follows: first to determine the type of the file to be detected, then the judgment of pack and pack type, again by unpack activity processing, after the success of the unpack, if it is multiple state to send the virus to the virtual machine processing, the final signature matching. We can be intuitive to see the disadvantages of traditional antivirus engine from diagram.

II. FRAMEWORK DESIGN SCHEME

A. *The Structure of the Extensible Framework*

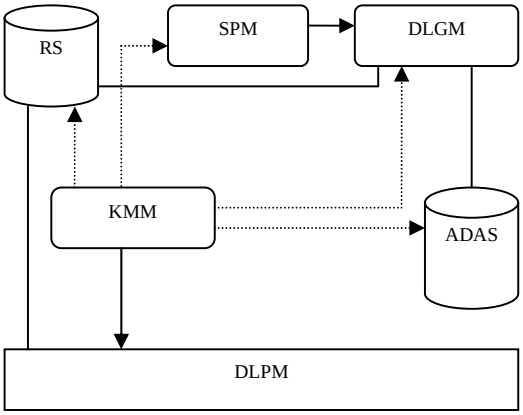

FIGURE II. FRAMEWORK ARCHITECTURE DIAGRAM.

DLGM is a key part in the whole extensible framework, is responsible for receiving the output results of the SPM, convert a disordered atomic testing activities subset into effective Detection Logic, new detection logic must meet the concurrency and dependence. RS (Rule Set) provide rule sets used by detection logic. DLPM (Detection Logic Perform Module) receive inspection activities of the output of the DLGM perform for a particular virus. ADAS (Aotm - Detection Activities Set) is the most basic testing Activities, receives commands from KMM management, can realize real-time updates.

In order to verify the validity of the design scheme, at the same time decrease the difficulty of the realization of the design, on the basis of the open source antivirus software AnSav prototype system is realized. AnSav written in Win32 assembly language, signature matching is realized by using classical BM algorithm [10]. Decomposing inspection activities on the basis of AnSav. For example, UPX.ASM unpack module realizes three classic ASPACK, UPX, PEcompact unpack algorithm respectively, encapsulates three unpack algorithm to form three DLLS, adding new unpack test need to add new DLL file.

B. *Intelligent Feature Codes*

Intelligent feature codes is the key to realize the engine decoupling, difference of intelligent feature codes and ordinary signature in carry the virus guide engine dynamic adjustment behavior information, using the behavior of the specific virus information dynamic adjustment detection logic to realize inversion of control, control coupling becomes interface coupling. Intelligent signature consists of three parts, respectively there are specific behavior code, engine control code, virus signature code. Specific behavior code describes the behavior characteristics of the virus, total 11 bytes. The first bytes for a sign, the rest of the four words represent the operation file system behavior of virus, operation the registry behavior, behavior and time behavior, the last word as extensions to retain. The second, it is the engine control code, as follows:

```
Virusbehavior struct
    Behavior flag    db ?
ModiFilebhInfo  word ?
    ModiRegInfo    word ?
    ProcInfo       word ?
    TimeInfo       word ?
    Reserved       word ?
Virusbehavior ends.
```

Engine control code of 3 bytes, respectively represent scanning object control information, engines work behavior control information and scan control information. Scanning the object control information is that the scope of the scanning, such as RAM, DBR, specific folders. Engine control information shows that how to deal with the testing results of the work behavior, such as alarm, isolation, directly deleted, restart the delete, etc. Degree of scan information indicates whether in-depth scanning.

```
EngineCtlInfo struct
    Objctlflag     db ?
    Behaviorflag   db ?
    Effectflag     db ?
EngineCtlInfo ends.
```

Virus signature code is the same as the traditional virus signature code, extracting the string of the virus code as matching basis. Therefore, intelligent feature codes as follows:

```
InteligentVirSig struct
    uVirBehavior  Virusbehavior <>
    uEngCtl     EngineCtlInfo <>
    uVirBody          Sig  VirusSig    <>
InteligentVirSig ends.
```

C. *The Key Algorithm Description*

(b) Detection logic generation algorithm

The algorithm is executed in the DLGM module, the use of rules and signature generate specific virus detection logic L. To traverse the first testing activities set SETa, system according to the type of testing activities establishes processing P, scanning th S, clear C three queue; Order according to the P, S, C merge into total queue QSet, and specific detection in traverse QSet activity; Adjust QSet according to the rules of the order of testing activities, the adjusted of queue QSet for a detection logic L.

Input: Sig, SET^a

Output: Detection logic L

```
(1)  Begin
(2)  For each aseti in SETa
(3)  Switch(aseti.type):
(4)    Case P:
(5)      enPqueue(aseti);break;
(6)    Case S:
(7)      enSqueue(aseti);break;
(8)    Case C:
```

(9) enCqueue(aseti);break;
(10)EndCase.
(11)EndFor
(12)For each Qi in QSet
(13) Mergequeue(Qi);
(14)EndFor
(15)For each aj in QSet
(16) For each ri in Ruleset
(17) IF Match(ri, aj)==True
(18) Adjust aj in QSet based on ri;
(19) EndFor
(20) EndFor
(21) L←QSet;

III. SIMULATION EXPERIMENT

In order to prevent the virus cause damage, select the VMWare configuration of the Windows xp virtual machine as a test environment. Selecting gray pigeons virus spread widely for experiment, because gray pigeons is a typical independent Trojan virus, representative; Second, only to start the file scan module can detect effectively independent Trojan virus, is suitable for analyzing the performance of the anti-virus engine.

A. The Effectiveness of the Extensible

First of all, design verification prototype system scalability experiment. Ideas are as follows: to modify the first gray pigeons simulation code variants of the virus, and then add a new atomic testing activity to variant virus. If to detect a new virus without modifying the engine code layer, it is in line with the software design of the open closed principle, prototype system has a good scalability.

First of all, through the new packer and encryption algorithm generates gray pigeons variant. Using the prototype had no packer WWPACK32 pack, then its encryption, to ensure that the variations of gray pigeons G_Server.dll can avoid prototype system hulling and decryption. In order to detect variations of gray pigeons, new module must be extension of prototype system, the modification process is as follows: first, the new unpack module Unpack04.dLL and the decryption module Decrptor2.dll is added to the prototype EASet, second, to increase the three detection rules into RS, as follows:

R1:IF Sig.Pack==04 then <a031,S1>;

R2: IF Sig.Encryp == 02 then <a037,S1>

R3:IF Sig=pack AND Sig= Encryp Then <aunpack,adecrp>

Third, to the virus signature database to increase the behavior of the new virus signature, such as Sig = <... , 04,... 02 >. Serial number 04 in the vector Sig is packers algorithm, packer said the virus algorithm for WWPACK32, 02 said encryption algorithm sequence number. Pretreatment testing activities a031 in the rules of the R2 corresponds with Unpack04.dll module, pretreatment testing activity a037 corresponds with Decrptor2.dll, S1 corresponding signature matching module. 3.4.2 prototype implementation algorithm generated after detection logic < < a031, a037 >, S1 >, the detection logic execution process is as follows: first, call Unpack04.dll hulled, again call Decrptor2.dll, finally by the detection of virus signature matching module. From the above example shows that detect a new virus or variations virus process only need to update the corresponding feature database, rule base and add new test module, the antivirus engine does not make any modify code layer, illustrate the engine frame has good expansibility.

B. Performance Analysis

Because the engine can be adjusted dynamically, improve the scalability of the system, and the scalability of performance for validation. Prepare for the test data, the choice is between 30 k to 1 M 2000 size of executable files, in 200 files random add case processing, polymorphic deformation process and the write packers encryption processing; Written in the guarantee the packers cryptographic operations can't use the current shell shell program, you must use the hulling and decryption program written by preprocessing before detection. Will gray pigeons, such as more than 10 kinds of virus in the 2000 executive file, than the average in a single file Tave for performance parameters, Ttotal to full scan time, F for scanning the number of files, such as type 1.

$$T^{ave} = T^{total} / F \tag{1}$$

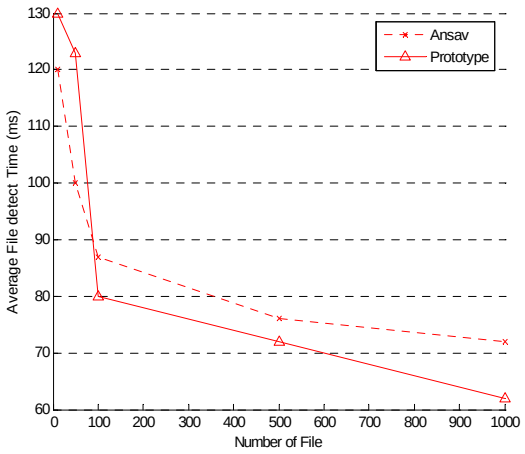

FIGURE I. SINGLE FEATURE CODES AND DETECT FILE CASES, PERFORMANCE COMPARISON.

Unable to modify commercial software, so can't compare with commercial software performance. With AnSav and as a representative of the traditional engine prototype system. Prototype system of a single file scanning time T is divided into two parts, the pattern matching time Tm and engine tuning is Te, such as type 2.

$$T = T^{0}_{m} + \sum_{i=0}^{n} T^{i}_{e} + \sum_{i=0}^{n} T_{m}^{i}$$

Tm is the time signature matching, Tei (0 < I < n) is an intelligent signature Sigi detection logic time adjust engine, n

for behavior characteristic code number. In order to study the effect of adjusting time of the prototype engine on performance, to a single virus signature scanning multiple executable experiment. Figure 1 shows, at the beginning of the test (that is, the test file number less than 100) prototype testing time is longer than conventional engine so, because traditional antivirus engine is a static model, there is no engine adjustment time. This phenomenon is consistent with the theoretical analysis results. In later period, along with the increase in testing the number of files, in part through packer, encryption processing of the prototype system of file to be detected, Tm less than Tm of Ansav engine dynamic adjusting and optimizing the virus detection logic, so the detection performance was improved.

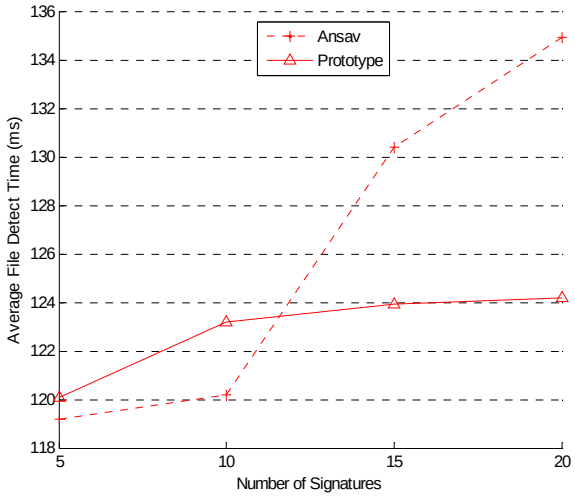

FIGURE II. SINGLE TEST FILE WITH THE SIGNATURE CASES, PERFORMANCE COMPARISON.

Second, in a single file for more signature scanning performance experiment. Experiment prototype system in intelligent feature codes and ordinary signature each accounted for 50%, AnSav are all common feature codes. Prototype system testing time T, such as type 3,

$$T = \sum_{j=1}^{l} T^{j}{}_{m} + \sum_{i=1}^{n} T^{i}{}_{e} + \sum_{i=1}^{n} T_{m}{}^{i}$$

L is the common feature code number, signature number n is intelligence, AnSav single file scan time. Figure 2 shows that the detection performance of conventional engine has nothing to do with signature number linear growth, but the quantity of prototype system of intelligent signature has a larger effect on the performance. Because, increases linearly with the increase of intelligent signature number n. Single file signature matching is the worst, seen by the result of the experiment of extensible engine performance bottleneck is a dynamic adjustment of the engine parts. Reality is need to modify the detection engine of the virus in the minority, most of the virus can still use fixed detection process implementation killing. According to related statistics, new Trojan virus in the first half of 2008, 2008, the company launched just 12 virus killing tool. On the basis of the third performance comparison experiment, the experiment of intelligent characteristic code number in the minority. The number of files to be detected for 100-2000, the proportion of ordinary signature with intelligent feature codes for 100:5, the total number of feature codes for 105.

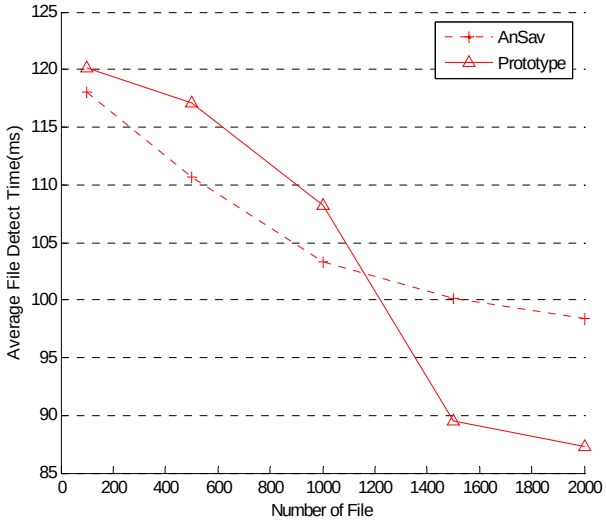

FIGURE III. PERFORMANCE COMPARISON OF RATIONAL DISTRIBUTION OF INTELLIGENT FEATURE CODES.

Figure 3 shows the comparison results, when testing the number of files more than 1000 more than the detection performance of AnSav prototype system, shows that when the number of files in delay can adjust engine process, so the prototype system of the overall detection performance is better than that of AnSav. Anyhow, simulation reflects the prototype system can be extended to enhance and improve the detection capability of a new virus, under the condition of rational distribution of intelligent characteristic code number system overall performance is improved.

ACKNOWLEDGMENT

Project supported by the Natural Science Foundation of Shandong Province, China(Grant No.ZR2014FL008) and the Scientific and Technical Development Project of Linyi(201412016).

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” *Phil. Trans. Roy. Soc. London*, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, *A Treatise on Electricity and Magnetism*, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in *Magnetism*, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized”, *J. Name Stand. Abbrev.*, in press.

[6] M. Young, *The Technical Writer's Handbook*. Mill Valley, CA: University Science, 1989.