Formal method using demodulation against viral software metamorphism

Ruo Ando, Yoshiyasu Takefuji
Graduate School of Media and Governance,Keio University,
5322 Endo Fujisawa, Kanagawa, 252 Japan
http://www.neuro.sfc.keio.ac.jp

Abstract: - Recent cyber attacks and viruses become more sophisticated. Metamorphic virus such as Win32 simile have a great impact on anti-virus software developers, which evades signature matching by inserting redundant fake assembler operation codes. In this paper we present a formal method applying the equality substitution called demodulation for detecting metamorphic viral coding. Proposal technique is based on theorem proving, which is possible to take high degree of human expertise. ATP(Automated Theorem Proving) involves verifying the correctness of the target code using formal specification language reflecting the experience of skilled programmer. In the stage of scanning, SoS(Set of Support) strategy is employed. For the searching to be feasible and more effective, the resolution of more than one clauses not in SoS is inhibited in order to prevent the prover go into abundant searching place. Target code is added to the list of set of support with repeating equality substitution to generate unit conflict. Experiment shows the effectiveness of proposal formal method for detecting metamorphic viral coding.
Key-Words: - Metamorphic Virus, Formal method, Equivalence validation, Set of Support, Demodulation, Unit conflict, Assembly code formulation

1 Introduction

The number of security incidents is still constantly increasing, which imposes a great burden on both the server administrators and client users. Despite the short history, computer viruses is becoming one of the most important issue. Although it has been about one decade since computer viruses became expected occurrence, Viruses, worms and Trojan damages personals, companies government. Code red, nimda and Msblaster recently are a valid example showing we suffer the great damege if we keep using the computer unpatced. We have come to accept the updating a software regularly and uniformly. Besides, the cost of maintaining and updating vulnerable software is increasing gradually but certainly.

1.1 Metamorphism : viral code hiding

Recent cyber attacks and viruses become more sophisticated, while many users begin to equip anti virus scanners that is mainly relies on signature matching. Recently virus is improved rapidly so as to evade signature matching. Symantec Corporation published the paper[1] introducing metamorphic viruses against the impact of W32 simile computer virus, loading complex viral code hiding techniques. Viral code hiding techniques which avoid the string matching are not new phenomenon. This kind of technology is first appeared in early 1990's, called polymorphic computer virus. Polymorphic viral coding applies encryption for its body to nullify the virus scanning. Still now polymorphic virus is challenging to detect completely and effectively. However, there exists detection method and scanners that survives and improved from DOS 16bit days for Polymorphic coding attack[2].

Metamorphic virus can change whole its body with no encryption. Instead of encryption, metamorphic viral code are generated by inserting redundant assembler, replacing register and changing magic word so as not to disturb the same action of pre-morphed virus in targeting operating system. As a result, after metamorphic coding, the virus can change its body while keeping the same function. As we discuss in section 3.1, once some virus become metamorphic, little piece of viral code are scattered over the whole infected program body. Consequently virus scanner cannot detect it with the sequence matching.

Although metamorphic virus is not appeared newly like polymorphic viruses, with the rapid improvement and complication in 32 bit processors and operating system, metamorphism is applied studiously by virus writers. While there are actually the scanners that survived DOS polymorphic days, no one can find the effective way of detecting viral metamorphism. Besides, it is hard to automate the heuristic process of detecting metamorphic compared with another viral code hiding techniques. It is expected

1.2 Software verification and validation

As the information processing system become sophisticated and complicated, software verification is now a broad and complex disciple for software developers who aim to assure that their product satisfies all the expected requirements. On technical aspect, software verification could be classified into two fundamental approaches. Dynamic verification is processed on the execution of a program, or checks its behavior on simulation. Static verification is a process to inspect whether some requirements of software is satisfied by mathematical or logical method. Formal verification is based on later one, which verifies the equivalence of the target code and the given code described as correct. Formal methods takes advantages in the sense that these do not depend on traditional logic simulation or test vectors, consequently provide more complete coverage of the program behavior.

2 Related work

In this section we discuss about some technologies detecting computer viruses. Detection method could be categorized into two types: 1)signature matchin*g* dealing with knows misuse viral coding and 2)heuristic scan applying discover method to find unregistered computer virus[3]. Heuristic scan is almost as same meaning as anomaly detection adopting data-mining techniques for secure network [4]. Another type is adaptive protection, which is implemented compiler to inspect the integrity of call pointers when program is executed.

2.1 Signature matching

Signature matching is searching the particular byte sequences in files according to each format, .COM file, .EXE file, .out file, scripts and macros. If predefined sequences are found, some action such as alarm, nullifying is triggered. In the sense that signature matching hardly generate false positive alarm, this technique is still core mode of anti virus detection. Signature matching, while made more flexible by pre-qualifying files and type of infections, and using wild cards, still requires exact matches between infection and signature. Also [5] shows the effectiveness in anomaly detection of process behavior by tracing system call sequence

Although the Anti virus software have relies much on signature-based techniques including regular expression and wild card, it could be pointed out that this method is sometimes CPU intensive, and costs a lot in frequently managing signature. And to ensure signature definitions, these should be updated from server of each vendor regularly and uniformly. Consequently time lag of updating could be the cost and cause to be exploited when it is not updated.

2.2 Heuristic scan

Recently anti virus software began to equip heuristic scan. Heuristic scanning is the operation to complement signature matching in finding potentially malicious code (or actual viral code) that have not been released and corresponded by anti virus software vendor[6]. Instead of looking for specific strings, heuristic scanning deal with higher information such as assembler operation code or commands in order to find uncategorized viruses or possibility of malicious code.

The word heuristic (hyu-RIS-tik), which is originated from Greek word heuriskein, means the way to determine something in a methodic or experimental way. A skilled programmer can notice the sign of malicious operation from normal one when he inspects the program carefully by some debugging tools. Heuristic scan is applied so that the experience or knowledge of debug expert in to a anti virus software. These scanning techniques are now available in many popular anti virus software although there still many way proposed to evade heuristic scanners. When it is executed, heuristic scanner searches hundreds of operation code, instructions and behaviors that viral code may include and calculate possibility according to the threshold the user has set up. Nowadays, it is summarized by AV vendors that about 70-80% of unknown virus can be detected in heuristic scan.

2.3 Formal methods

Formal method provides an mathmatical solution to malicious code detection. The advenatages of thie technique is for improving performance to detect, reducing code size and increase confidence in the correctness of target code. Also, formal method is categorized into static and dynamic inspection. Static method searches the property usually performed without executions of a program[7][8] whereas dynamic method finds the property for a specific input with partially execution of programs. Concerning static method, there are two methodologies applied. Control graph to determine whether the target program satisfies the given properties[9]. Formal representation defines correct or suspicious property of the program. This is also called proerty checking, which could be classified into theorem proving, equivalence checking, and model checking. Model checking has been researched strenuously over the last decade. This mehods is inspecting for the existence of a FSM (finite state machine) in another FSM[10].

3 Metamorphic viral coding

As we discussed in section 1.2, there are two methods to evade the string template matching:polymorphic and metamorphic coding. Although the polymorphic viruses are hard to prevent still now, there exists a countermeasures that have been improve since DOS 16 bit era. Polymorphic virus must have a executable code section that operating system can recognize. Then, once decrypted or decrypting engine is discovered, this could be manageable by signature scanner and eradiated. Through the last decade when the architecture of 32 bit processors or operating system becomes sophisticated and complicated, metamorphism is studiously applied by virus writers. As matters stand, there is no decisive technique for detecting metamorphic viral coding. Anti virus software companies says that less than 70- 80 % of viral metamorphism could be detected. In this section, we discuss four types of metamorphic viral coding, which are the same in mutating operating code and magic word form the same higher action.

3.1 Register replacement

As some simple techniques of metamorphic coding, we can exploit the exchangeability of some registers in IA 32 architecture.

```
POP EDX
MOV EDI, 0008H
MOV ESI,EBP
MOV EAX 000DH
ADD EDX, 005FH
MOV EDX,[EDX]
MOV [ESI+EAX*0000CCC9,EBX]

POP EAX
MOV EDX,0008H
MOV EDX,EBP
MOV EDI,000DH
ADD EAX,005FH
MOV ESI,[EAX]
MOV [EDX+EDI*0000CCC9],ESI
```

List1. Register replacement

List 1 shows the metamorphic coding of generating two different forms by replacing register. In this case, edx is replaced by eax, ebx by edi, edi by ebx, and esi by ebx. As a result, when this kind of code is translated in machine language, string template is changed.

3.2 Magic number permutation

Some metamorphic virus mutates a new form by changing magic word. List2 shows the substitution of magic word into ESI is permutated. The line 1 is malformed by using register EDI and EDX. And in line 2, substitution of 110000FFH is translated through EDX and EBX.

```
MOV DWORD PTR [ESI] ,11000000H
MOV DWORD PTR [ESI+0004],110000FFH

MOV EDI,11000000H
MOV [ESI],EDI
POP EDI
PUSH EDX
MOV DH,40
MOV EDX,110000FFH
PUSH EBX
MOV EDX,EBX
MOV [ESI+0004],EDX
```

List2. Register permutation

Compared with the case 1, which could be detected by crafted string matching such as half-byte wild cards, the next case go further to change magic value 11000FFH.

```
MOV EDX,11000000H
MOV [ESI],EBX
POP EDX
PUSH ECX
MOV ECX,11000000H
ADD ECX,000000FFH
MOV [ESI+0004],ECX
```

List3. Magic number permutation

List3 shows dividing the magic word 110000FF into 11000000 and 000000FF. Consequently, wild card based string matching become disable to find the magic number.

3.3 Reordering instructions

Compared with polymorphic viruses which decrypt themselves to a constant virus body in memory, this type of metamorphic does not come to be constant because jump instruction is inserted at random.

```
INSTRUCTION_A
INSTRUCTION_B
INSTRUCTION_C:

LABEL_2:
INSTRUCTION B
JMP
FAKE INSTRUCTIONS
START:
LABEL_3:
INSTRUCTION_C:
LABEL_1:
JMP
FAKE INSTRUCTIONS
```

List4. Reordering instructions

List4 shows the obfuscation of entry point to avoid the searching of the beginning of the executable code section. As a result, signature is scattered in amongst the original code. Furthermore, in this technique virus can inserts fake instruction between core instruction and jump code. In extremely case of this kind of method, Zperm virus generates millions of iterations to surpress the anti viral emulation speed.

These four types of metamorphism are the same in the sense that there could be translated as the certain abstraction from both core and fake instructions regardless of its various malforming forms. In other words, whatever the code is permutated, the function that each morphed code has to achieve is the same, consequently a kind of higher action can be logged as event in device driver. With the example in section 1.2 overflow is finally occurred despite its malformation of assembler code. Adversely, as long as we can only investigate on the assembler instruction level, we cannot go out of heuristic or data mining frameworks. From the next sections, we propose a insertion of new layer, called driver based protection layer, to obtain the highly abstracted action of metamorphic code.

3.4 Complexity of metamorphism

In this section we discuss about the complexity to detect the morphed viral code. Through last decade, operating system and CPU have become more sophisticated and complicated accompanying with implementation of many function, consequently we do not use all instructions and operations at the same time for one purpose. Adversely, many combinations of routines come to be possible to achieve the same function. Metamorphic viruses are exploiting this point of modern computer system. Morphed virus writer implements n functions in order to generate n! variations. For example, if Win32 metamorphic virus such as W32/ghost has 16 routines, the combination could be:

Combination = 16! = 20922789888000

Besides, the computer viruses choose the unpredictable one among these combinations using random number using some value of TLB (thread information block).

Selection = random (seed)

Seed : FS:Och /EIP

Also another register transition that is usually unpredictable could be the seed of random number.

```
MOV EAX DWOR PTR _XXX$[EBP]
PUSH EAX
MOV ESI,[EDX](*)
LEA ECX, DWORD PTR _BUF$[EBP]
MOV EDI,[EBX-04X](*)
PUSH ECX
NOP(*)
CALL _STRCPY
ADD ESP,8
```

List5. Inserting fake operations

List5 shows the list inserting fake instructions for redundant state in order to evade the signature matching. In IA32 architecture, there are eight generic registers available for programmers, all of which are not used in one operation. Particularly, ECX, EDX, ESI and EDI are often applied for auxiliary use. It follows that at factorial of 4 combinations is possible without accounting order of fake instructions.

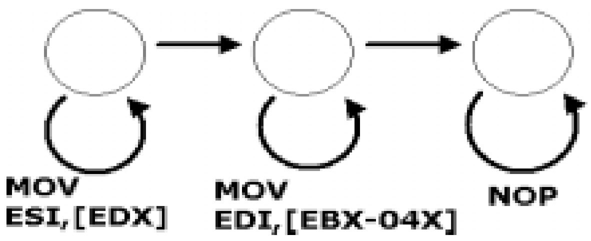

Fig. 1 Redundant states(loop) of list 5

Concerning the magic number, every magic number in windows operating system can be decomposed arbitrarily into 32 bit memory address number. However, the hexadecimal numbers from 0xBFFFFFFF to 0x00000000 such as 0x004235CC is preferable because the address from 0xFFFFFFFF from 0xC0000000 is number in kernel mode that is inclined to be hooked in heuristic scanning. Thus, metamorphic viral coder can generate the vast number of derivations using the large memory space and abundant availability of operations in IA and Win 32 complex architecture.

4 Proposal method

4.1 Validation using demodulation

In this paper we propose a detection using formal method against metamorphic computer viruses which checks the equivalence between pre-obscuration signature and morphed code.
Our formal method is based on theorem proving using demodulation for simplification and canonicalization[11][12]. In the normal cases of hardware and software checking, we prepare the formal specification, and prove or disprove the correctness of a system regarding the formulation of system. In this paper, instead we formulate the pre-obscuration signature as correct code, and inspect the equivalence between this code and metamorphic code. We can view these kind of detection method just discussed as a translation between computer languages.

Metamorphic virus writers mainly focus to avoid signature matching using fake and redundant instructions. On the other hand, virus scanners only attempts to check if the target code has functionally infected. The verification consists of proving that two expressions are equivalent, where the first expression is in the builder's language and the second is in the user's language.In this context, we defined the scanner's language as canonical or simplified form. To inspect the equivalence of these two assembly code, the theorem proving strategy called demodulation is applied in this paper.

Demodulation is designed to enable a theorem prover to simplify and canonicalize information. To be specific, this process applies unit equality clauses in order to rephrase, rewrite, simplify and canonicalizing expressions. A demodulator DM of the form EQUAL(A,B) for terms A and B applies to term C if and only if B is an instance of A or C is an instance B. Clauses that represent information semantically redundant (not syntactically redundant) can be purged by this procedure.

4.2 Unit conflict

The termination condition, that is detection condition for formal verifier that succeeds in the target assembly code is that of finding a proof of contradiction. In the methodology of formal verification, proving that certain property follows from a set of properties, facts and definitions is directed by assuming the desired conclusion false. Consequently, if expected property follows the remaining clauses, assuming it false in principle leads to a contradiction. This kind of contradiction is called unit conflict in the context of theorem proving based validation. The definition of unit conflict would be described as:

Definition: Unit conflict
The unit conflict has been found when the two clauses contains a single literal with opposite in sign, and can be unified, where a clauses unit clause is the one contains a single literal.

According to the definition above and proposal analysis framework, unit conflict means the equivalence between original assembly code and metamorphic code. In other words, in the proposal system, the generation of unit conflict leads to succeeding of detection of morphed viral code. Two clauses termed contradictory unit clauses if and only if each two clauses contains a single literal, the two are opposite in sign, and the two literals can be unified.

4.3 Set of support

Set of support was introduced by L.Wos, S.Robinson and Carson in 1965[13]. If the clause T is retrieved from S, SOS is possible with the satisfiability of S-T. Set of support strategy enable the researcher to select one clause characterizing the searching to be placed in the initializing list called SOS. For the searching to be feasible and more effective, the resolution of more than one clauses not in SOS is inhibited in order to prevent the prover go into abundant searching place.

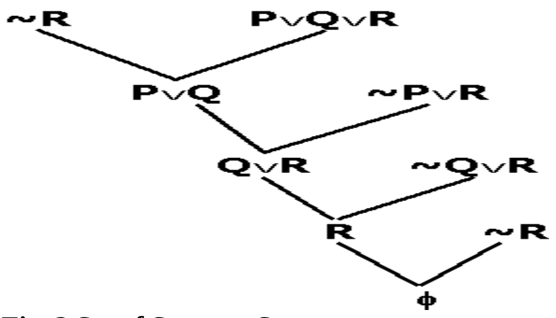

Fig. 2 Set of Support Strategy

Figure2 show the resolution process in set of support strategy, where
S={P and Q and R, ~P and R, ~Q and R, ~R}.
The restriction imposes the reasoning so that the program do not apply an inference rule to a set of clauses that are not the complement of set of support.

4.4 Assembly code formulation

As we discussed before, canonicalizaion using demodulation is applied in proposal system in order to check the equivalence of two assembly code. There could exist three ways to verify assembly code. One is to translate the assembly code to logic-level language. Another is to translate the logic-level formulation to assembly code. A third possibility is to translate both signature and matching assembly code into a third one, and inspect if the same result is obtained from those two formulations. In this paper we adopt the third technique.As is the usual case of viral code detection, proposal method is aimed for processing assembly language. For formulate the example we discuss in the next section, the clause is
ASM(I3,MOV(ESI,ECX))
Where ASM means the assembly code can be described in this manner and I3 is the number of instruction order. In the process of verification we are going to demodulate this clause to
ASM(expressions)
This could be expected to match the pre-obscuration code.
As framework of verification, demodulation is controlled by set of support strategy in following steps.
ASM(I3,MOV(ESI,ECX))
STEP1: To formulate the pre-obscuration code and morphed code by ASM function discussed above.
STEP2: Put the two ASM function in the set of support with choice of forward demodulation as inference rule.
STEP3: Instruct the reasoning program OTTER to translate the two ASM formulation.If demodulation generates unit conflict, then the equivalence of two function is validated, which means the metamorphic code is detected.

As attempted, because of the way the assembly code has been formulated, a successful detection will signed by unit conflict between the two ASM clauses.

5 Sample test

To test the effectiveness of our formal method, we used open source software called Otter (Organized Techniques for Theorem-proving and Effective Research) to deduct the equality of metamorphic viral coding. Otter is a forth-generation Argonne National Laboratory deduction system to prove theorems stated in FOL with Knuth-Bendix completion, weighting, and strategies for directing and restricting searches.Sample test code is proccessed by otter on Linux Kernel 2.6. The target code is composed by using the technique of register replacement and reordering instructions discussed in section 3. List6 shows an example code fragments of metamorphic coding selected we pick up in this paper as mutated to a new form in a new generation of the same virus.

```
Loop:
POP ECX
NOP
JMP ROUTINE1
ROUTINE3:
CALL EDI
XOR EBX,EBX
BEQZ N2
N2:
JMP Loop
JMP L4
ROUTINE2:
NOP
MOV EAX,0D601H
POP EDX
POP ECX
NOP
JMP ROUTINE3
```

```
ROUTINE1:
JECXZ Illegal_Code
XOR    EBX,EBX
BEQZ N1
N1:
MOV ESI ECX
JMP ROUTINE2
ROUTINE4:
```
List6. Sample metamorphic code

Here we apply the expression ASM(x) which means the assembly code can be described in this manner to translate the routine into the list below.

```
ROUTINE1:
ASM(I4(I3,JECXZ(Illegal_Code)))
ASM(I5(I4,XOR(EBX,EBX)))
ASM(I6(I5,BEQZ(N1)))
```
List7. Fragment of assembly code formulation

Demodulation rewrite the list6 in two phases. In the first phase, the substitution from left to right is operated according to the literal I1 to I19. By doing this, the instruction order obscured by inserting redundant JMP is translated from top to bottom. In the second phase we could tell this to an automated reasoning program by giving it the demodulator to delete the redundant instruction or JMP operation code.

```
ASM(x,JMP(y))=ASM(x).
ASM(x,BEQZ(y))=ASM(x).
ASM(x,XOR(EBX,EBX))=ASM(x).
ASM(x,NOP)=ASM(x).
ASM(NOP,x)=ASM(x).
```
List8. Demodulators

If this code fragment is submitted to an automated reasoning program in the set of support and forward demodulation as the inference rule, list demodulated in 31 steps to conflict to pre-obscured code of list6. Thus code fragment given by list can be translated in to list, proving that these two assemble code is equivalent.

5 Conclusion and further work

The conventional anti-virus software, and file scanner are all based on stored signatures. Consequently these schemes have the limitation against the new derivation using metamorphic coding discussed in section 3. In this paper we present a formal method applying equality substitution called demodulation for detecting metamorphic viral coding. As we discussed in section 2.3, there are three techniques for formal method: model checking, equivalence checking and theorem proving. This paper shows that theorem proving is effective for searching metamorphic code in the sense that it can reflects high degree of human expertise in its detection. Theorem proving involves verifying the correctness of mathematic theorems using a formal specification language. This is semi-automatic and consequently able to reflect the experience of skilled programmer. In the stage of scanning, SoS(Set of Support) strategy is employed. For the searching to be feasible and more effective, the resolution of more than one clauses not in SOS is inhibited in order to prevent the prover go into abundant searching place. Target code is added to the list of set of support with repeating equality substitution to generate unit conflict, which shows the effectiveness of proposal formal method for detecting metamorphic viral coding. Demodulators could be described from the experience specified for skilled developer. This system enabled us to detect metamorphic code effectively in the point that we control scanning with demodulators heuristically adopted.

For further work, instead of translating the assembly code to another, instructing reasoning program to determine which morphed code can be generated satisfying the property of original code is expected. Compared with demodulation technique, this method can select the best form among possible generation. It means that even if some clauses are missed, it may be possible to detect by semi-automatic deduction. Demodulation has a close relationship to paramodulation both in origin and in purpose. Each, if successful, causes an equality substitution to take place. While demodulation requires the equality literal to be in a unit clause, paramodulation does not. Another vital difference is that, while demodulation allows a nontrivial variable replacement only in the argument of the equality literal and in the term into which the substitution is being attempted.

References:

[1] Szor, Peter and Ferrie, Peter. "Hunting for Metamorphic." Virus Bulletin Conference, September 2001.
[2] Stephen Pearce, “Viral Polymorphism”, paper submitted for GSEC version 1.4b,2003.

[3]Ruo Ando, Yoshiyasu Takefuji, "Two-stage quantitative network incident detection for the adaptive coordination with SMTP proxy", Computer Network Security, Lecture note in computer science Springer,pp424-428,2003

[4] Dimitris A. Karras, Vasilis Zorkadis, "Neural Network Techniques for Improved Intrusion Detection in Communication Systems" WSEAS CSCC,2001,pp318-323

[5] Kosoresow, Andrew P. and Steven A. Hofmeyr, "Intrusion Detection Via System Call Traces", IEEE Software, Sept.–Oct. 1997, pp 35-40.

[6]Symantec Bloodhound Technology http://www.symantec.com

[7]J. Bergeron, M. Debbabi, J. Desharnais, M. M. Erhioui, Y. Lavoie and N. Tawbi, “Static Detection of Malicious Code in Executable Programs”. Proc. of the International Symposium on Requirements Engineering for Information Security, 2001.

[7]Diomidis Spinellis. Reliable identification of bounded-length viruses is NP-complete. IEEE Transactions on Information Theory, 49(1):280-284, January 2003.

[8] Mihai Christodorescu, Somesh Jha,”Static Analysis of Executables to Detect Malicious Patterns”,In 12th USENIX Security Symposium, Washington, DC, August 2003

[9] O.Sheyner, J.Haines, S.Jha, R.Lippman, and J.M.Wing, "Automated Generation and Analysis of Attack Graphs", in proceedings of the IEEE symposium on Security and Privacy, (Oakland, CA.), May 2002.

[10]Hao Chen, Drew Dean and David Wagner. “Model Checking One Million Lines of C Code.”, Proc. of 11th Network and Distributed System Security Symposium, 2004.

[11]Larry Wos, "The Problem of Demodulation During Inference Rule Application", J. Autom. Reasoning 9(1),pp141-143,1992

[12]Larry Wos, George A. Robinson, Daniel F. Carson, Leon Shalla, "The Concept of Demodulation in Theorem Proving". J. ACM 14, pp698-709,1967

[13] Wos, L.; Robinson, G.; Carson, D., "Efficiency and completeness of the set of support strategy in theorem proving" J. ACM,pp. 536-541,1965

[14]Khaled.E.A.Negm,"Secure Mobile Code Computing in Distributed Environment", WSEAS TRANSACTIONS ON COMUTERS,2003,pp506-513

[15]Diomidis Spinellis. Reliable identification of bounded-length viruses is NP-complete. IEEE Transactions on Information Theory, 49(1):pp 280-284, 2003.