Resolution based metamorphic computer virus detection using redundancy control strategy

RUO ANDO, NGUYEN ANH QUYNH,YOSHIYASU TAKEFUJI
Graduate School of Media and Governance,Keio University,
5322 Endo Fujisawa, Kanagawa, 252 Japan
1-1538-11 Iriya Zama, Kanagawa, Japan 228-0024
{ruo.quynh,takefuji}@sfc.keio.ac.jp http://www.neuro.sfc.keio.ac.jp

Abstract: - In this paper we propose a resolution based detection method for detecting metamorphic computer virus. Our method is the application of formal verification using theorem proving, which deduce parts of viral code from the large number of obfuscated operations and re-assemble those in order to reveal the signature of virus. While previously many kinds of the symbolic emulation based methods have been applied for metamorphic virus, no resolution strategy based method is proposed. It is showed that the complexity of metamorphic virus can be solved if the obfuscated viral code is canonicalized and simplified using resolution based state pruning and generation. To make our detection method more feasible and effective, redundancy-control strategies are applied for the resolution process. In this paper the strategies of demodulation and subsumption are applied for eliminating the redundant path of resolution. Experiment shows that without these strategies, resolving metamorphic code into several simplified operations is almost impossible, at least is not feasible in reasonable computing time. The statistics of reasoning process in detecting obfuscated API call is also presented. We divide obfuscated API call into four modules according to the types of metamorphic techniques and compare the conventional resolution with our method applying redundancy-control strategy.

Key-Words: - Metamorphic virus, Resolution based detection, Theorem proving, First-order logic, Redundancy control strategy, obfuscated API call

1 Introduction

The number of security incidents is still constantly increasing, which imposes a great burden on both the server administrators and client users. Among these incidents, despite the short history, computer viruses have become a very important issue. Although it has been about one decade since computer viruses became expected occurrence, viruses, worms and Trojan damages personals, companies government. Recent viruses and worms are divided into two types.

One is to exploit the vulnerability of the latest and major software such as Nimda, MSBlaster and SQL Slammer.These worms recently are a valid example showing we suffer the great damege if we keep using the computer unpatched. Another evolution of virus writing is proceeded on the intention of challenging AV(anti virus) products. Thwarting virus scanning technique is first appeared in 1990's, called polymorphic computer virus. In 2002, Win32.Simile have a great impact on AV software company with sophisticated viral code hiding techniques such as EPO (entry point obfuscation).This kind of malicious code is called metamorphic virus about which Symantec Corporation published the paper in 2001[1].

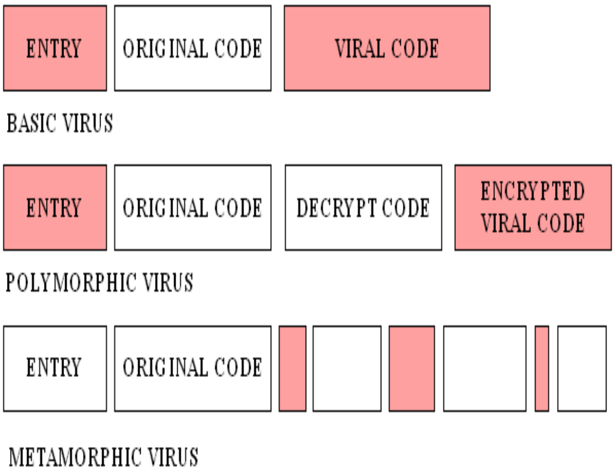

Fig. 1 Three kinds of viruses

Figure1 shows the comparison of three kinds of viruses. In basic virus, part of the entry point code is changed to derive program control to infected code. Detection is relatively easy if static signature matching can find viral code. Polymorphic virus applies encryption for its body to evade signature matching[2]. Still nowadays, no complete solution is proposed to detect polymorphic virus. However, there also exists heuristic method that survives and has been improved from DOS 16bit days for Polymorphic coding attack. Metamorphic virus reprogram itself with little pieces of viral code scattered. At the same time, the space between viral codes is filled withjunk code. This kind of virus shows different body in every infection although the generations are all functionally equivalent. Besides, what makes it more difficult to detect metamorphic virus than polymorphic is that the infection of entry point is hidden. This obfuscating technique is called entry point obfuscation. In polymorphic virus, entry point code is changed to lead the program control to the viral code regardless of situation of infecting file, which is not the case of metamorphic virus. After the evaluation since 16bit DOS days, AV scanner can improve the detection rate of polymorphic virus by finding the infected entry point. Signature based scanning is almost nullified for the metamorphic virus. Although as countermeasure for metamorphic virus the code emulation techniques are applied in AV products, tracing the value of registers or variables is not feasible enough to find this type of viral code. In this paper we introduce the resolution based symbolic emulation method for metamorphic virus detection. The effectiveness of resolution based detection for metamorphic virus is discussed.

2 Resolution based virus detection

Resolution based virus detection is one of applications of formal verification using theorem proving. To resolve the obfuscated code into simplified one, we need the adequate representation of disassembly code and state pruning and generation. In this section we discuss the methodology for resolution based detection.

2.1 Representation of disassembly code

Modern computer has Von Neumann architecture where the instruction is processed in sequential form. For every execution of instruction, the state of program loaded to memory is changing, which is expressed by the value of register, variables and address in memory. We formulate assembly code as follows:

-instruction_name(data_type(x),data_type(y),address, time(z))|
state(data_type(x),data_type(y),address,time(z)

Figure2 shows the FoL (First-order Logic) formulation of API GetModuleHandleA. Obfuscation and simplification of this operation is discussed in section 4. In generic assembly language such as GaS of X86, the instructions consists of opcode and operand. The first two arguments, data_type(x) and data_type(y) express opcode and operand. Address is the number of execution. The term Time(z) expresses how many this instruction is executed.

original code	**formulated code**
mov dword_1,A	**state(VAR(dword_1), const(A),v,Tim(1))**
mov dword_2,B	**state(VAR(dword_2), const(B),w,Tim(1))**
mov dword_3,0	**state(VAR(dword_3), const(0),x,Tim(1))**
push offset dword_3	**state_push(var(dword_3), y,Tim(1))**
call ds: GetModuleHandleA	**state(call (GetModuleHandle), z,Tim(1))**

Fig. 2 FoL formulation of disassembly code

2.2 State resolution and demodulation

In general, verification of the software is the process of explosion of the states of program. If the target program is infected, there has be a state transition to achieve some operations as part of malicious behavior such as calling API. To resolve these instructions from obfuscated code, resolution system need to store all state generated by the execution of every instructions.

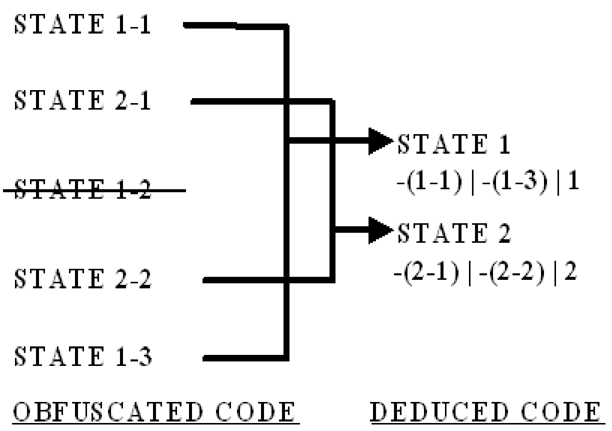

Fig. 3 State pruning and resolution

Figure 3 shows proposal method of the resolution based detection of obfuscated metamorphic code. Our method consists of two types of reasoning, resolution and demodulation. First, several instructions are deduced to one simplified instruction by applying hyper resolution. We apply the transition axioms to combine several states and generate new state for this resolution which means state pruning is executed at the same time. Transition axioms and state pruning is discussed in next section. Second, junk code such as NOP and xor(X,X) is crossed by a technique called demodulation. Demodulation is one of equality substitution methods to purge the information in a sense semantically redundant. Demodulation is discussed in next section. In the process of detection, we add the formulation of disassembly code of viral code besides the code under inspection. Reasoning program attempts to deduce the same as "signature clauses" from obfuscated metamorphic code. The reasoning program is terminated when the equivalence is found between two clauses (codes). In automated reasoning, this termination is called unit conflict. The automated reasoning event called unit conflict is generated when we get unit clauses with opposite in sign.

Definition: Unit conflict
The unit confict is a event where two clauses contains a single literal of which signs are opposite and can be unified. These two clauses are called contradictory unit clauses.

In proposal method, detection is succeeded if unit conflict is occurred between deduced clauses and signature clauses. We prefabricate assembly code formulation that calls some API and add it to the list in the opposite (negative) sign. Then, reasoning program is proceeded, in order to occur the unit conflict by resolving the same clauses of original viral code.

3 ATP strategy

FoL resolution presented in this paper is one of the techniques of ATP (automated theorem proving).In this section we discuss ATP strategies to make resolution faster. These strategies are designed to reduce the redundancy by the retained information.

3.1 Set of support

Set of support was introduced by L.Wos, S.Robinson and Carson in 1965[9]. [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 set of support. 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.

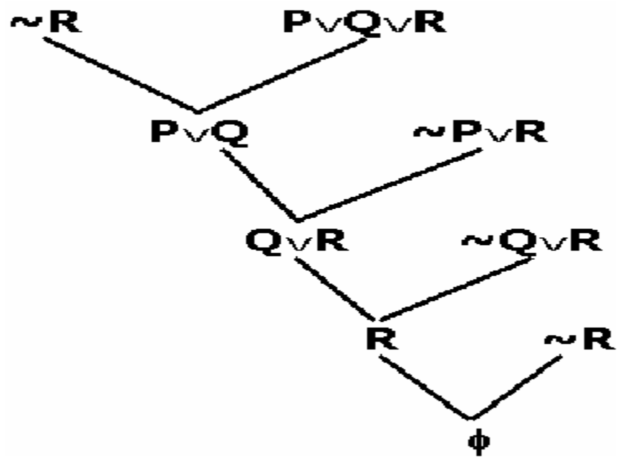

Fig. 4 Set of support strategy

Figure4 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. The further discussion of this restriction strategy can be referred in [8].

3.2 Redundancy control strategy

Redundancy-control strategy is basically designed to reduce the obstacle for reasoning program within the retained information. In this paper we apply two strategies, subsumption and demodulation.

3.2.1 Subsumption

Subsumption is the process of discarding a specific statement. The clause that duplicated or is less general is discarded in the already-existing information. As a result, subsumption prevents a reasoning program from retaining clauses that is obviously redundant, especially is logically captured by more general clauses. For example,

OLDER(father(x),x)
Subsumes
OLDER(father(Ann),Ann).

Definition. The clause A subsumes the clause B when B is the instance that is logically captured by B.
The clause
P(X)

Subsumes the clause
P(a).

There is a variation of subsumption called back subsumption in the newly generated clauses that is more general. In this paper we apply only forward subsumption. As we discuss in section 2, to deduce
state(var(dword_2),var(dword_1),x,time(y))
from the clauses

-state(reg(edx),var(dword_1),x,time(y))|
-state(var(dword_2),reg(edx),x,time(y)).

There are several paths and aximos that could be applied. Subsumption strategy is effective when the same or more specific clause in the present of already-existing clause is generated. The clause is crossed and the generated clauses on the process of resolution is also eliminated. The effectiveness of this strategy is presented in experimental results.

3.2.1 Demodulation

In the automated reasoning, one of the procedures of simplify or canonicalize information is called demodulation[8]. A unit equality applying for rewriting or rephrasing expressions to canonical form is called demodulator. A demodulator is a positive unit clause with an equality predicate to simplify the information. Demodulation is the effective way to eliminate garbage instructions.

EQUAL(nop,crossed).
EQUAL(mov(reg(eax),reg(eax)),crossed).

These demodulators eliminate the junk insertion. In proposal method, for every insructions, the state clause is generated. Among those generated clauses, the state generated by junk instruction is demodulated to the state "crossed", which is not translated to another state any more.Demodulation is also applied for the prevention of the nested substitution.

const(const(x))=const(x).
const(var(x))=var(x).

These demodulators block the resolution program from endless substitution of constant(x) and var(x). Demodulation can be applied for the junk code insertion when the target clause is identified as dead code in one line.

4 Experimental results

In this section we discuss the experimental results of detecting metamorphic virus applying proposal method. The effectiveness of redundancy-restriction strategy in simplifying the metamorphically obfuscated code is presented. To pick up the sample, we construct the model of typically metamorphic virus. Figure5 shows the basic structure of metamorphic virus such as W32.evol, W32.simile, and W32.Zmist. Infected code is scattered and spread over its body. Infection of entry point code which direct program control to viral code is obfuscated and hidden. The scattered viral codes are reordered by inserting branch instructions such as JMP and JE. Besides, in some cases, redundant loop is inserted to change its signature for every infection. From the view point of implementation, the techniques applied for metamorphic virus is divided into three types, register substitution, magic number permutation and reordering instructions. The experiment is divided according to these techniques.

To test the effectiveness of our method, we used open source software called OTTER (Organized Techniques for Theorem-proving and Effective Research) to simplify the obfuscated viral code. Otter is a forth-generation of Argonne National Laboratory deduction system to prove theorems stated in FoL with Knuth-Bendix completion, weighting and strategies for directing and restricting searches.

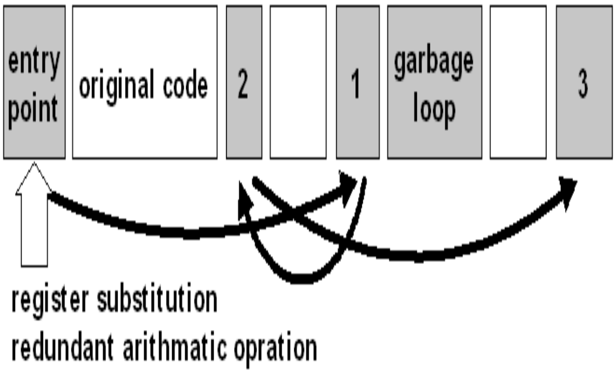

Fig. 5 Basic structure of metamorphic virus

Table 1 and 2 shows the result of identifying hiding technique called register substitution.Register substitution is the technique exploiting the exchangability of some registers in IA-32 architecture. Usually, most of operations consists of malicious behavior is not completed in one instructions.We can select arbitrary register to some operation of substitution. This technique generates different implementations for the same operation such as insert value A into address B.

original code	2 clauses	
obfuscated code	14 clauses	
	no strategy	proposal method
clauses given	59	45
clauses generated	93	60
Hyper_res generated	93	60
demod & eval rewrites	29	19
clauses forward subsumed	0	16
subsumed by SOS	0	5
clauses kept	92	43

Table. 1 Register substitution I

original code	2 clauses	
obfuscated code	8 clauses	
	no strategy	proposal method
clauses given	*	25
clauses generated	*	24
hyper_res generated	*	24
demod & eval rewrites	*	8
clauses forward subsumed	*	6
subsumed by SOS	*	1
clauses kept	*	17

Table. 2 Register substitution II

Table 1 is result of simplifying the instruction No 2,3 in Figure 2. In experiment, this instruction is obfuscated by inserting 12 redundant instructions. Table 2 is result of simplifying the instruction No 4,5 in Figure 2. In experiment, this operation is obfuscated by inserting 6 redundant instructions. It is validated that this type of obfuscation cannot be simplified with feasible computing time without redundancy control strategy.

In developing software, a constant number programmer sometimes remember is called "magic number". For example, in Win32. the base address of KERNEL32.DLL is fixed as magic number. Magic number permutation even changes constant DWORD values by adding redundant arithmetic operation. By using this technique, magic DWORD values are changed in subsequent generations of the virus. No wildcard based string matching is effective for this obfuscation method Table 3 shows the result of simplifying the instruction No 1 in Figure 2. In experiment, this code is obfuscated by inserting 4 redundant instructions.

Table 4 show the result of identifying the simple redundant loop.Metamorphic virus, the W95/Zperm family appeared in June and September 2000. These viruses inserts jump instructions into its code and add redundant loop module. In experiment, the loop consists of three instructions. To identify this loop, 43 clauses is generated. However, by using redundant-control strategy, the number of generated clauses is reduced to 24.

original code	1 clauses	
obfuscated code	5 clauses	
	no strategy	proposal method
clauses given	*	19
clauses generated	*	25
hyper_res generated	*	25
demod & eval rewrites	*	12
clauses forward subsumed	*	9
subsumed by SOS	*	2
clauses kept	*	15

Table. 3 Magic number permutation

original code	0 clauses	
obfuscated code	3 clauses	
	no strategy	proposal method
clauses given	24	14
clauses generated	43	24
hyper_res generated	43	24
demod & eval rewrites	18	9
clauses forward subsumed	0	2
subsumed by SOS	0	2
clauses kept	43	22

Table. 4 Junk code and loop insertion

Table 5 shows the result of resolving all modules above into simplified API call operation as shown in Figure3.2. Calling GetModuleHandleA is obfuscated by the techniques of register substitution, magic number permutation and junk code and loop insertion. In experiment, these codes is obfuscated by inserting 21 redundant instructions. It is validated that obfuscating API call cannot be revealed in reasonable computing time without redundancy-control strategy.

original code	**5 clauses**	
obfuscated code	**26 clauses**	
	no strategy	**proposal method**
clauses given	*	**203**
clauses generated	*	**293**
hyper_res generated	*	**293**
demod & eval rewrites	*	**143**
clauses forward subsumed	*	**112**
subsumed by SOS	*	**25**
clauses kept	*	**180**

Table. 5 Compilation: obfuscating API call GetModuleHandleA

5 Conclusion and further work

In this paper we present the resolution based technique for detecting metamorphic computer virus. In proposal method, scattered and obfuscated viral code is resolved and simplified to several parts of malicious code. Compared with emulation based method, this formal verification based method is effective for metamorphic virus which applies anti-heuristic techniques such as register substitution or permuting magic number. Our method is one of misuse detections, so it takes advantages in the probability rate of false positive. To make resolution program detect metamorphic virus faster and more feasible, we apply some redundancy-restriction control such as demodulation and subsumption. Demodulation is a equality substitution technique enabling a program to simplify and canonicalize statements by using rewrite rules called demodulator. This strategy is effective to remove garbage and junk code for obfuscating viral code. In experiment, these two kinds of strategies are coordinated to simplify the obfuscated API call operation. Metamorphic viral code is divided into four modules according to the technique of register substitution, magic number permutation and junk loop insertion. Experiment show that two modules cannot be simplified to detect without redundancy-control strategy. Although the other two modules can be verified as a part of viral code, it is showed that the proposal method using redundancy-control strategy is effective to make the reasoning program faster. It is also validated that without these strategies resolution program cannot reveal obfuscated API call in reasonable computing time.

References:

[1]Peter Szor and Peter Ferrie.Hunting for Metamorphic. Virus Bulletin Conference,September 2001: 123-144.

[2]Stephen Pearce, "Viral Polymorphism", paper submitted for GSEC version 1.4b,2003.

[3]Diomidis Spinellis. :Reliable identification of bounded-length viruses is NP-complete. IEEE Transactions on Information Theory,January 2000 :280-284.

[4] Static Analysis of Executables to Detect Malicious Patterns (2003) Mihai Christodorescu and Somesh Jha12th USENIX Security Symposium, August 2003

[5]Hao Chen, Drew Dean, and David Wagner.Model checking one million lines of C code. In Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS), pages 171--185, San Diego, CA, February 2004.

[6]O.Sheyner, J.Haines, S.Jha, R.Lippmann, and J. M. Wing, "Automated Generation and Analysis of Attack Graphs", IEEE Symposium on Security and Privacy , April 2002.

[7]Arun Lakhotia, Eric Uday Kumar: Abstracting Stack to Detect Obfuscated Calls in Binaries. SCAM 2004: 17-26

[8]Larry Wos, George A. Robinson, Daniel F. Carson, Leon Shalla: The Concept of Demodulation in Theorem Proving. J. ACM 14(4),1967:698-709

[9]Larry Wos, George A. Robinson, Daniel F. Carson: Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. J. ACM 12(4), 1965:536-541

[10]William McCune: OTTER 3.3 Reference Manual CoRR cs.SC/0310056,2003

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