Effectiveness Evaluation of Software Protection based on MPN

Wang Huai Jun  
School of Information Science and Technology, Northwest University, NWU-Irdeto IoT-Information Security Joint Laboratory  
Xi’an, China  
hjwang006@163.com

Fang Ding Yi  
School of Information Science and Technology, Northwest University, NWU-Irdeto IoT  
Xi’an, China

Wang Ni  
School of Information Science and Technology, Northwest University, NWU-Irdeto IoT-Information Security Joint Laboratory  
Xi’an, China

Gu Yuan Xiang  
Irdeto, Holland  
NWU-Irdeto IoT-Information Security Joint Laboratory  
Ottawa, Canada

Abstract—There are many technologies of software protection to prevent software from being attacked, such as software piracy, tampering, and reverse engineering, but what about the effectiveness of them is hard to answer. This paper presents a new evaluation method considering attack cost as the metric to evaluate the effectiveness of software protection under the circumstances of fulfilling assumptions that all software can be attacked successfully. We propose a new attack model, called MPN, and deduce an approach to calculate attack cost with it. Moreover, we also verify the approach.

Keywords—software protection; attack model; Petri Net; effectiveness evaluation

I. INTRODUCTION

Software protection technologies were born to avoid threats of software piracy, tampering, and reverse engineering. But whether and how these protection technologies are effective are still an unsolved problem. Our motivation derives from how to evaluate the effectiveness of software protection by measuring attack cost. Now, there is no accepted approach was widely accepted as a common metric. But many researchers have done lots of relevant works, and we can classify them in two groups. ① Evaluation based on Theoretical Analysis[1][2][3][4], which measures or prove the effectiveness of software protection through theoretical analysis. ② Evaluation based on Attack[5][6][7][8][9], which measures or proof from the view of attack.

Technically, the evaluation approach in this paper belongs to the second group. Rather than doing manual attack experiment or designing common attack tools, we describe software attacks with an attack model. In this paper: Section 2 introduces an attack model called MPN. In Section 3, discuss the method to evaluate the effectiveness of software protection with the metric “attack cost”. We verify the evaluation method through a case in Section 4. Section 5 concludes this paper.

II. EFFECTIVENESS EVALUATION WITH PETRI NET

Attack Net model based on Petri Net was first proposed by McDermott[10]. The basic Petri Net[11] is a three-tuple (P, T, F). The flexibility of Petri Net makes that it can abstract attack process and refine it. It can indicate more information. So, we use Petri Net to model software attacks.

A. Primary elements in Software Attack

In an informal attack description, there are six types of elements are listed by Steffan[12]. According to that list, we made new description in following Table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker</td>
<td>Attacker in Software Attack.</td>
</tr>
<tr>
<td>Object</td>
<td>Object in Software Attack.</td>
</tr>
<tr>
<td>Goal</td>
<td>The purpose of a software attack process.</td>
</tr>
<tr>
<td>Method</td>
<td>The way to achieve Goal. There are many Methods in a process.</td>
</tr>
<tr>
<td>State</td>
<td>Intermediate process of software attack.</td>
</tr>
<tr>
<td>Technique</td>
<td>The attack technique to attack software in a attack process.</td>
</tr>
<tr>
<td>Sub-goal</td>
<td>The purpose of to achieve States.</td>
</tr>
<tr>
<td>Action</td>
<td>Performing Technique.</td>
</tr>
<tr>
<td>Precondition</td>
<td>The condition to perform the attack technique.</td>
</tr>
<tr>
<td>Influence</td>
<td>The consequence after performing the attack technique.</td>
</tr>
</tbody>
</table>

There are many Methods to finish a Goal, and each Method includes many States and Techniques.

B. Mapping Primary elements into Marked Petri Net

Fig. 1 is an example of Petri Net. $P={p_0, p_1, p_2, p_3, p_4}$ is the set of States, $T={t_0, t_1, t_2, t_3, t_4}$ is the set of Techniques.

![Example](Figure 1)

Then we can mapping the primary elements in Table I into the counterpart in Fig.1 as follows.

1). Attacker and Object

Attacker is the people executing attack based on Petri net. Object is the software that Attacker want to attack. And they both can be seen from the Petri net.

2). Goal, Sub-goal, State and Technique

In Fig.1, $p_0$ is the original State, and $p_4$ is the last State which is the Goal of the attack. $t_0$, $t_1$, $t_2$, $t_3$ and $t_4$ are Techniques in the software attack process. $p_1$, $p_2$, and $p_3$ are Sub-goals of Technique.

3). Method
One Method contains States and Techniques, which form a path from original State to last State. In Fig.1, we can find three Methods, \( p_0 \rightarrow t_0 \rightarrow p_1 \rightarrow t_1 \rightarrow p_2 \rightarrow t_2 \rightarrow p_3 \rightarrow t_3 \rightarrow p_4 \); \( p_0 \rightarrow t_0 \rightarrow p_1 \rightarrow t_2 \rightarrow p_3 \rightarrow t_4 \rightarrow p_4 \); \( p_0 \rightarrow t_0 \rightarrow p_1 \rightarrow t_1 \rightarrow p_4 \).

Obviously, a Method is a path in Petri Net from original State to last.

4) Preconditions, Actions and Influence

The elements introduced above belong to static properties. The difference between Marked Petri Net\(^{13}\) and the basic one is that the Marked Petri Net has a Token, which is a dynamic object. In the attacking process, Token is moved from one State to next one. If the condition is satisfied, an event will be enabled, that means in its input there are enough Tokens. Firing event will change the number and position of Token. Enabling event is just the Precondition; firing event is just the Action; changing Token is the influence of the Action.

C. Software Attack Modeling

**Definition 1 MN:** which is software attack modeling based on Marked Petri Net, including six elements \((P, T, F, Path, Rate, Cost)\), among which, first three elements come from the basic Petri Net; \( Path \) is a finite set of Methods. \( Rate \) and \( Cost \) respectively stand for attack rate and attack cost.

Based on the software attack model MPN and software attack process, we give the following definition.

**Definition 2 Attack Path:** Some Methods are included in a MN, and each method is called an Attack Path. (See the description of “Method” in detail from Table I)

**Definition 3 Attack Rate:** Be used to quantify the rate of attack progress in one Attack Path. It stands for Precondition of each Technique, and it will be changed dynamically when attacking.

We quantify attack process with Attack Rate, and it is instantiated as Token in MN. Let \( R(p_i) \) stands for Attack Rate’s value of reaching State \( p_i \), and then \( R(p_i) \) is more than or equal to 0, also less than or equal to 1. While \( R(p_i) \) is equal to 1, the Goal of attack will be achieved, and \( R(p_i) \) = 1, and it can be seen as the minimum increment of Rate caused by firing \( t_i \) in actual attacking.

**Definition 4 Attack Cost:** Reflect the cost of \( T(\text{Technique}) \) in Attack Path. In the attack process, \( Cost(p_i) \) is used to calculate the attack cost produced when reaching the State \( p_i \). And if \( p_i \) is the last State, \( Cost(p_i) \) is the attack cost of the Method. Attack Cost is quantifiable, and we use \( sac \) as its unit.

D. The Token in MN

Since MN = \((P, T, F, Path, Rate, Cost)\) is the process model to describe software attack in detail. In Attack Path \( Path_a = \{P_0, T_0, P_1, T_1, \ldots, T_{m-1}, P_m\} \), every \( T \) in Attack Path one by one, and ends at \( P_m \). After performing \( T \), the number of Token included in \( P_m \) will be decided whether \( T \) will be enabled. So, in the attack process, the Sub-goal of each technique is to get enough Token needed in following State. Next we analyze \( t_i \) in \( Path_a \) in the view of Attack Rate. The condition to enabling \( t_i \) is whether the value of \( R(p_i) \) after firing \( t_i \) is big enough. Obviously, when firing \( t_i \), the value of \( R(p_i) \) will be increased.

Then, in MN Token can be instantiated with Attack Rate. Next, we give a definition:

**Definition 5 Threshold Value (\( \mu \)):** it stands for the condition of enabling \( t_i \). Where is \( \mu > \mu_i \).

So Precondition of \( T \) is: Token in \( P_i \) \( \geq \mu_i \), and the Influence of \( T \) increase the Token in \( P_{i+1} \). At first, Token is assigned 0 in \( P_i \). After performing \( T \), the Token = \( R(T) \) + \( R(t_i) \) in \( P_i \). Figure 2 presents the progress of attack along \( Path_a \). \( R(t_i) = R(p_i) - R(P) \) is the increment of Rate caused by firing \( t_i \) in actual attacking.

![Figure 2. The transition of Token in Patha](image)

According to Fig.2, there is: \( \mu_i \geq R(p_i) \) \( \geq \mu_{i+1} \).

So, \( \mu_i - R(p_i) \) is the minimum increment of Rate caused by firing \( t_i \) besides it is also the minimal Token to reach \( P_{i+1} \). Because first Technique fired in \( Path_a \) is \( t_0 \), \( \mu_0 = 0 \); In addition, because the last Technique is \( t_{m-1} \), after \( t_{m-1} \) fired, \( R(p_m) \) = \( \mu_m = 1 \).

Also, we can get: \( \Delta \mu = \mu - \mu_0 \), and it can be seen as the importance of \( t_i \) according to \( \Delta \mu \). If \( t_i \) is more important, \( \Delta \mu \) will be larger. In practice, it is affected by many factors, for example, attack scene, attack experience. So, we need to determine it through analyzing \( t_i \) in depth.

Above all, we can get following conclusion: the effect of Method displayed by an Attack Path Patha consist of firing every \( t \) in Patha.

As the only dynamic behavior in Attack Path, firing \( t \) is the reason that Attack Cost is produced. Then, in Attack Path Patha, Cost(Patha) is made up of the attack cost produced by firing each \( t \).

III. Effectiveness Evaluation with SAPMNON

From the above mentioned, when evaluating effectiveness the metric can use the attack cost.

A. Metric Calculation

Let \( c \) be the attack cost produced by firing Technique \( t_i \). Sub-goals of different \( t \) may be the same, but the \( c \) of these \( t \) may be yet different. Determining \( c \) is difficult, because we need to analyze and understand the Technique in depth. However, there are also many methods to acquire \( c \), most of which based on expert or attack experience. In Attack Path Patha = \{\( P_0, t_0, \ldots, t_i, \ldots, t_{m-1}, \ldots, t_m, P_m\}\), there is,

\[
Cost(P) = \sum_{i=0}^{m-1} c(t_i)
\]

To get more accurate results, we decompose the Technique based on Software Security Patterns\(^{14}\) into some smaller attack units. We named the small attack units as ae, main attack units are shown in Table II.
TABLE II. Main attack units

<table>
<thead>
<tr>
<th>ae</th>
<th>Class</th>
<th>Meaning</th>
<th>Attack Information</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>ae1</td>
<td>Disassembling by tools.</td>
<td>disassembly code</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>ae2</td>
<td>Memory spying</td>
<td>Spy the operation to memory of PE file</td>
<td>Number of memory blocks being spied</td>
<td>B</td>
</tr>
<tr>
<td>ae3</td>
<td>Differential attack</td>
<td>Compare difference</td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>ae4</td>
<td>Collusion attack</td>
<td>Multiple attackers cooperate to attack.</td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>ae5</td>
<td>Process snooping</td>
<td>Run program to analyze character.</td>
<td>snooping time</td>
<td>E</td>
</tr>
<tr>
<td>ae6</td>
<td>Interactive debugging</td>
<td>interact with debugger.</td>
<td>Time of user interference</td>
<td>E</td>
</tr>
<tr>
<td>ae7</td>
<td>Control flow analysis</td>
<td>Analyze control flow</td>
<td>numbers of control flow instructions</td>
<td>D</td>
</tr>
<tr>
<td>ae8</td>
<td>Instruction replacement</td>
<td>Modify instruction in the program.</td>
<td>Numbers of replacement instructions</td>
<td>E</td>
</tr>
<tr>
<td>ae9</td>
<td>Branch interference</td>
<td>Modify the condition jump.</td>
<td>Modify time</td>
<td>D</td>
</tr>
<tr>
<td>ae10</td>
<td>Control flow modifying</td>
<td>Modify control flow</td>
<td>Numbers of modified control flow</td>
<td>C</td>
</tr>
<tr>
<td>ae11</td>
<td>Code lifting</td>
<td>Code reuse.</td>
<td>Times (code size )</td>
<td>C</td>
</tr>
<tr>
<td>ae12</td>
<td>Data lifting</td>
<td>Data reuse.</td>
<td>Times (data size)</td>
<td>C</td>
</tr>
<tr>
<td>ae13</td>
<td>Data file replacement</td>
<td>Replace data file</td>
<td>Time of data file replacement</td>
<td>D</td>
</tr>
<tr>
<td>ae14</td>
<td>Program file replacement</td>
<td>Replace code file</td>
<td>Time of program file replacement</td>
<td>D</td>
</tr>
</tbody>
</table>

Then calculate cost(t) of Technique by calculating cost of ae. If Technique ti is performing by \{ae0, ...,aei\}, we can get

\[
\text{cost}(t_i) = \sum_{j=0}^{i} c(ae_j)
\]

(2)

These data are obtained from attack log. For example, technique “DUMP” is decomposed by “Memory spying” and “Data lifting”, The attack log is as follows.

DUMP: ae1: memory operation times
ae4: code size

So, \(\text{cost}(t_{\text{DUMP}}) = c(ae_2) + c(ae_4)\). Next, divided ae into five ranks, and value of ae is set according to attack experience.

- **Rank A**: 1(sac). attack automatically absolutely.
- **Rank B**: 5(sac). it is feasible to attack automatically absolutely by designing a special tool.
- **Rank C**: 10(sac). attack manually with some assistant tools, which could provide much help.
- **Rank D**: 15(sac). attack manually with some tools which could only provide less help.
- **Rank E**: 20(sac). attack manually, there is no tools to provide help.

In which, sac is the unit of software attack cost. Then, calculate \(c(ae)\) by (3) in following.

\[
c(ae) = \text{Grade}(ae) \times \text{Value}(ae)
\]

(3)

In (3), \(\text{Rank}(ae)\) is the function to get ae’s Rank and \(\text{Value}(ae)\) is the function to calculate ae’s information value. But, it needs to normalize \(\text{Value}(ae)\) based on the data normalization theory, and \(\text{Value}(ae) = \log(\text{Value}(ae))\).

According to (1),(2),(3), we can get:

\[
\text{Cost}(P) = \sum_{i=0}^{5} \sum_{j=0}^{i} \text{Grade}(ae_j) \times \log(\text{Value}(ae_j))
\]

(4)

This attack modeling will be perfected with appearance of new Techniques in future and the Rank of attack units may be changed.

IV. A CASE STUDY

A. Background

1). Serial Numbers

This is often used to make user use the software legally [15]. If user had paid for the software, the developer will provide a unique Serial Number. Of course, in software there is an algorithm to certify the input number.

2). Control Flow Flattening

This [4] is a control obfuscation technique, which can obfuscate original control logic into a flatten control flow. The original software is “SerialNum.exe”, and “SerialNum-Obf.exe” is the protected software protected by the tool introduced in [16].

3). Attacks on Serial Numbers

First, we find some programs like “Crackme”, then attack them manually. Then, we modeled the attack introduced in [15] adding other methods with new codes into the program. At last, the MPN can be constructed for attacking against checking program by Serial Number.

![MPN for attacking against Serial Number](image)

The meaning of each State and Technique is shown in Table III.

<table>
<thead>
<tr>
<th>P/T</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_0)</td>
<td>Original program</td>
</tr>
<tr>
<td>(P_1)</td>
<td>The codes had been assembled</td>
</tr>
<tr>
<td>(P_2)</td>
<td>The Basic Segments have been obtained</td>
</tr>
<tr>
<td>(P_3)</td>
<td>Obtain the performed Basic Segments</td>
</tr>
<tr>
<td>(P_4)</td>
<td>Obtain the Key Segment</td>
</tr>
<tr>
<td>(P_5)</td>
<td>Obtain the cracked program</td>
</tr>
<tr>
<td>(S_0)</td>
<td>Disassemble the program</td>
</tr>
<tr>
<td>(S_1)</td>
<td>Partition codes in some Basic Segments</td>
</tr>
<tr>
<td>(S_2)</td>
<td>Mark performed Basic Segments</td>
</tr>
<tr>
<td>(S_3)</td>
<td>locate the Key Segment and modify Key Instruction among it</td>
</tr>
<tr>
<td>(S_4)</td>
<td>Monitor memory change and terminate when generating the address of Fail block, and Key Block is the terminated block.</td>
</tr>
<tr>
<td>(S_5)</td>
<td>Add New Codes into the program; Key Block always jumps to New Codes.</td>
</tr>
</tbody>
</table>

Obviously, two attack paths are included in Fig. 3.

- The first path is: \(\{P_0, t_0, P_1, t_1, P_2, t_2, P_3, t_3, P_4, t_4\}\), which attacks the software by destroying Serial Number checking based on modifying the key instruction.
- The second path is: \(\{P_0, t_0, P_1, t_1, P_2, t_2, P_3, t_3, P_4, t_4\}\), it stands for the attack which destroys Serial Number checking by adding new codes into the program.

TABLE III. meaning of each P and T in Fig. 3
Some researches or experiments have prove the Control Flow Flattening is effective\[15, 16, 17\]. Next, we will evaluate the technique with our proposed method.

B. Effectiveness Evaluation of Control Flow Flattening

Firstly, determine $ae$ in each Technique $t_i$ which are as follows:

<table>
<thead>
<tr>
<th>Technique</th>
<th>$t_0$</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ae$</td>
<td>$ae_1$</td>
<td>$ae_2$</td>
<td>$ae_3$</td>
<td>$ae_4$</td>
<td>$ae_5$</td>
<td>$ae_6$</td>
</tr>
</tbody>
</table>

Then, calculate Value of $t_i$. PEiD\[18\] is used to analyze $SerialNum.exe$ and $SerialNum-Obf.exe$ to obtain sizes of text sections. Some scripts and plug-ins in IDA\[19\] to collect data, which includes: amount of jmp and conditional jump instruction; amount of the re-performed instructions; amount of Basic segments performed; checking the redundant space in Object. The value of each $ae$ is shown in Table IV.

**TABLE IV.** Results of attack units $ae$

<table>
<thead>
<tr>
<th>Object</th>
<th>$ae_1$</th>
<th>$ae_2$</th>
<th>$ae_3$</th>
<th>$ae_4$</th>
<th>$ae_5$</th>
<th>$ae_6$</th>
<th>$ae_7$</th>
<th>$ae_8$</th>
<th>$ae_9$</th>
<th>$ae_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SerialNum.exe$</td>
<td>21371</td>
<td>206</td>
<td>417</td>
<td>1026</td>
<td>1566</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SerialNum-Obf.exe$</td>
<td>23315</td>
<td>263</td>
<td>657</td>
<td>2938</td>
<td>3938</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

With the data in Table II and V, calculate $Cost(Path_1)$ and $Cost(Path_2)$, the results are shown in Table V.

**TABLE V.** Value of attack cost calculation

<table>
<thead>
<tr>
<th>Object</th>
<th>$Cost(Path_1)$</th>
<th>$Cost(Path_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SerialNum.exe$</td>
<td>134.3 (sac)</td>
<td>116.3 (sac)</td>
</tr>
<tr>
<td>$SerialNum-Obf.exe$</td>
<td>148.4 (sac)</td>
<td>130.4 (sac)</td>
</tr>
</tbody>
</table>

C. Analysis

In Path1 and Path2, both of attack costs of attacking on $SerialNum.exe$ are lower than $SerialNum-Obf.exe$. Obviously, we can quantize the effectiveness of the Control Flow Flattening by attack cost which is the evaluation metric we proposed. $Diff$ is defined as the difference of attack Cost between unprotected and protected program.

$$Diff(Path2) = 130.4-116.3 = 14.1\text{ (sac)}$$

So we can evaluate the protection by compare the $Diff$, the bigger, the better. But if we consider the performance of the software, the result may be different, because some protection technologies bring little performance penalty, but can improve security of the software too much.

V. Conclusion

In the paper, a new approach is presented to evaluate effectiveness of software protection, and in which attack cost is considered as the evaluation metric. An attack model is introduced with Marked Petri Net, and give a way to calculate the attack cost in attack process with MPN. At last, the feasibility and reasonability of our method are tested and verified through evaluating effectiveness of Control Flow Flattening. We can see that attack cost can quantize the effectiveness of the protection according to the experiment results. In the way, we can evaluate effectiveness of software protection according to $Diff$. 

ACKNOWLEDGMENTS

The paper was supported by the National Natural Science Foundation of China (No.61070176, No.61170218, No.61272461), Research and Industrial Project of Shaanxi Province under Grant (No. 2011K06-07).

REFERENCES


