Generating Various Patterns of Intrusion using IGA

Ja-Min Koo and Sung-Bae Cho

Dept. of Computer Science, Yonsei University
Shinchon-dong, Seodaemoon-ku
Seoul 120-749, Korea
{icicle, sbcho}@sclab.yonsei.ac.kr

ABSTRACT

As the computer environment changes significantly, several mechanisms such as firewall are developed and intrusion detection system (IDS) is the representative among them. However, because many IDSs are modeled and evaluated with known intrusion patterns, they have inertia not to detect the intrusions which are unknown or transformed. In this paper, we propose an IGA-based method to generate the novel intrusion patterns. The fitness values of generated intrusion patterns are evaluated by users, and we apply the generated intrusion patterns to LinSTAT, one of the most famous misuse detection systems. As a experimental results, various and available intrusion patterns are generated about 73%.

1. INTRODUCTION

An intrusion detection system (IDS) is a system that attempts to identify intrusions, which we define to be unauthorized uses, misuses or abuses of computer systems by either authorized users or external perpetrators [1]. IDSs detect intrusions by analyzing information about user activity from sources such as audit records, system tables and network traffic summaries. IDSs have been developed and used at several institutions. Some example IDSs are AT&T's Computer Watch [2], SRI International Intrusion Detection Expert System (IDES) [3].

As more and more organizations depend on IDSs as integral components of their computer security systems, techniques for evaluating IDSs are becoming more important. However, evaluating an IDS is a difficult task. First of all, it can be difficult or impossible to identify the set of all possible intrusions that might occur at the site where a particular IDS is employed. In addition, intruders can discover previously unknown vulnerabilities in a computer system, and then use new intrusion techniques to exploit the vulnerabilities. Second, an IDS can be affected by various conditions in the computer system. For example, even if an IDS can ordinarily detect a particular intrusion, the IDS may fail to detect that same intrusion when the overall level of computer activity in the system is high. Lastly, since recent security accidents are happened with transformed or unknown intrusions, it is difficult to evaluate the IDS with known or existed intrusion patterns. Moreover, it takes to long time to generate or transform the intrusion one by one manually.

In this paper, we propose the method to generate automatically various intrusion patterns using Interactive Genetic Algorithm (IGA) which is one of artificial intelligence techniques. We utilize the method proposed by MIT Lincoln Lab [4] to generate intrusion patterns, and use the UNIX tool EXPECT [5].

2. RELATED WORKS

2.1. Representation of Intrusion Patterns

To generate the intrusion patterns, we have to model the primitive of intrusion before. There are many kinds of methods to represent the intrusion, and Attack trees, state transition and rule-based methods are typical.

- Attack trees: It has existed in various forms, and under various names, for many years, but has been most recently described as a systematic method to characterize system security based on varying attacks [6]. They refine information about attacks by identifying the compromise of enterprise security or survivability as the root of the tree. The ways that an attacker can cause this compromise iteratively and incrementally are represented as lower level nodes of the tree. An enterprise typically has a set or forest of attack trees that are relevant to its operation. The root of each tree in a forest represents an event that could significantly harm the enterprise's mission. Each attack tree enumerates and elaborates the way that an attacker could cause the event to occur. Each path through an attack tree represents a unique attack on the enterprise.
- State transition: It is an approach to misuse detection using state transition diagrams for representing and detecting known penetration scenarios [7]. In this approach, a penetration is modeled as a sequence of

actions performed by an attacker that leads from some initial state on a system to a target compromised state. The initial state corresponds to the state of the system prior to the execution of the penetration, and the compromised state corresponds to the state of the system resulting from the completion of the penetration.

● Rule-Based: This approach is a traditional, general purpose rule-based expert system. The basic idea is that every detection rule in the rule base encodes a particular scenario and is responsible for detecting its occurrence by pattern matching the corresponding intrusion signature against the audit trial. When using this approach, RUSSEL [8]-predictive language-is widely used and forms *Condition*→*Action*.

2.2. Interactive Genetic Algorithm (IGA)

Genetic algorithm (GA), first proposed by John Holland in 1975, is an attractive class of computational models that mimic natural evolution to solve problems in a wide variety of domains. It applies some of natural evolution mechanism such as crossover, mutation and survival of the fittest to optimization and machine learning. In addition, it provides very efficient search method working on population, and has been applied to many problems of optimization and classification [9]. The procedure of a simple genetic algorithm is as follow:

```
t=0;
InitializePopulation P(t);
Evaluate P(t);
while not done do
        t=t+1
        P'=SelectParents P(t);
        Recombine P'(t);
        Mutate P'(t);
        Evaluate P'(t);
        P=Survive P, P'(t);
end_while
```

IGA is the same as GA except the way of assigning the fitness value. In IGA user gives fitness to each individual instead of fitness function. In this way IGA can 'interact' with the user, and also can perceive user's emotion or preference in the course of evolution. For this reason, IGA can be used to solve problems that cannot be easily solved by GA, such as design and art [10, 11, 12].

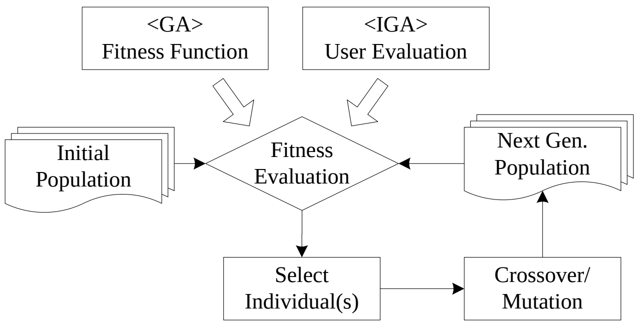


Fig. 1. Comparison of GA & IGA

Likewise this paper, we apply IGA because of the difficulty of evaluating intrusion patterns by mathematical fitness function. Therefore, generated intrusion patterns are evaluated by users with generated code and dependency of the behavior of each phase.

3. PROPOSED METHOD

3.1. Behavior Combination

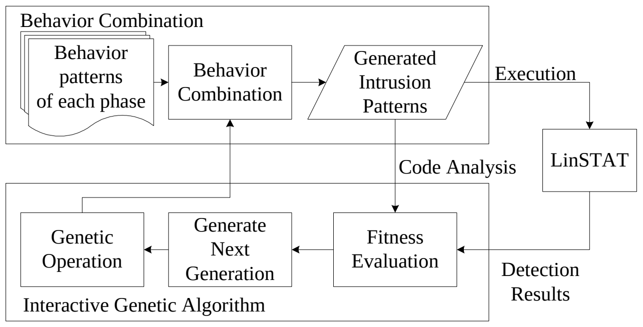


Fig 2. Proposed method

The proposed method as shown Figure 2 can be divided into 2 phases, behavior combination and interactive genetic algorithm.

Figure 3 shows the actual procedure of the intrusion pattern generation. There are many kinds of methods to represent the intrusion pattern, but we have applied the method proposed by MIT Licoln Lab [4]. In [4], they had divided 5 phase the sequence happening intrusion, and classified behaviors. For each phase, the behavior is the step to intrude some victim system likewise each node in attack tree

Much larger search space is necessary to generate various intrusion patterns, In this paper, we have added many behaviors analyzing existed exploits. Figure 4 shows the chromosome encoding of intrusion pattern, and search spaces are 2^{19} with it

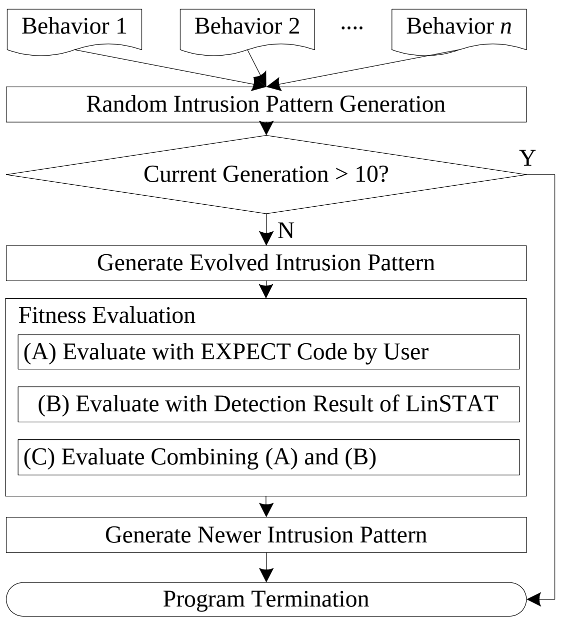

Behavior 1
Behavior 2
····
Behavior *n*
Random Intrusion Pattern Generation
Current Generation > 10?
Y
N
Generate Evolved Intrusion Pattern
Fitness Evaluation
(A) Evaluate with EXPECT Code by User
(B) Evaluate with Detection Result of LinSTAT
(C) Evaluate Combining (A) and (B)
Generate Newer Intrusion Pattern
Program Termination

Fig. 3. The procedure of Pattern Generation

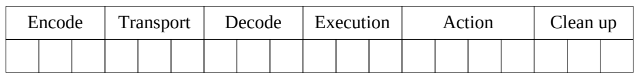


Fig. 4. Chromosome encoding

3.2. EXPECT

In this paper, we induce the script language called EXPECT to execute automatically. This package is based on another UNIX package called Tcl (Tool command language).

The Tcl package provides an interpreter for a simple programming language that includes variables, procedures, control constructs such as "if" and "for" statements, arithmetic expressions and other features.

The EXPECT package also provides a programming language interpreter. The core of the EXPECT interpreter is the Tcl interpreter, but EXPECT extends the Tcl command set to include several commands for controlling interactive programs. The command "spawn" creates an interactive process (such as telnet). The command "expect" waits to receive a specified string pattern (such as "login: ") from the process. The command "send" sends a string to the process. Thus, a script containing several expect/send" sequences and general programming constructs can control an interactive session and thereby simulate a human computer user. Figure 5 illustrates the operation of EXPECT. The following is a simple EXPECT script that controls a brief rlogin session:

```
#Spawn an rlogin process.
spawn rlogin ComputerName -l UserName
#Expect the password prompt, then send the password.
expect "Password: " send "ActualPassword \r"
#Expect the shell prompt, then send commands.
#The shell prompt is specified in a regular expression.
expect -re ".*%|.*|.*#" send "whoami \r"
expect -re ".*%|.*|.*#" send "ls \r"
expect -re ".*%|.*|.*#" send "logout \r"
```

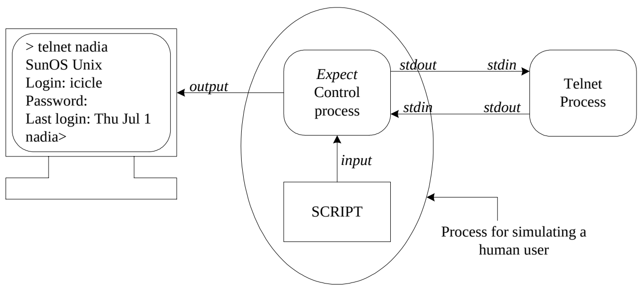


Fig. 5. Simulation of a human user's activity

3.3. LinSTAT

LinSTAT applied to evaluate fitness in this paper had been developed in Santa Barbara. It is one of STAT (State Transition Analysis Technique) [13]. LinSTAT and WinSTAT are two host-based systems similar to USTAT. It uses the event logs produced by the Snare Linux kernel module. This system is an interesting example of component reuse to implement similar functionality in different environmentsplatforms. The event providers for USTAT, LinSTAT and WinSTAT are obviously different. However, some of the entities used in scenarios are the same, and so are some of the scenarios.

3.4. Pattern Generation using IGA

It is easy to be generated meaningless intrusion patterns with high fitness value when using generic GA. Therefore, we employ IGA to evaluate intrusion pattern with generated code and detection result of LinSTAT. After, new population is generated influenced previous fitness value, and then genetic operations are applied. With these steps repeatedly, we can generate newer and various intrusion patterns

4. INTRUSION PATTERN GENERATING SYSTEM

We have developed the intrusion pattern generation system based on VC++ 6.0 to evaluate conveniently, and Figure 6 shows the system.

Operating at the first time, it generates 10 individuals randomly. Users can execute generated intrusion patterns by pressing button written "exe" and validate the EXEPECT code pressing "visualize." When pressing "exe," intrusion pattern is executed and the detection results of LinSTAT is displayed whether detected or not.

After, users choose the proper score from 10 to 50 (5 degrees) with detection result and code. When pressing "next," evolved new individuals are generated, the evolution procedure are terminated when the number of generation is 10.

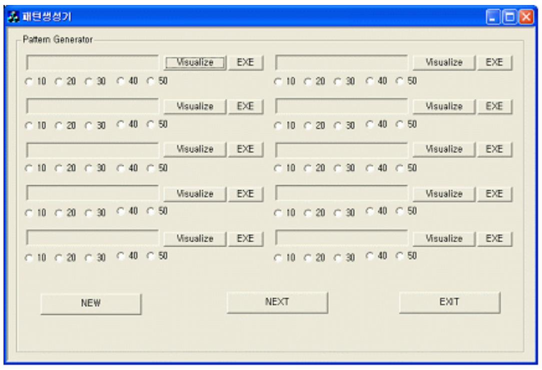


Fig. 6. Generating System

5. EXPERIMENTS

5.1. Experimental Environments

For each behavior is implemented by EXPECT script language, and we use generating system mentioned before to generate and execute automatically. The parameters of the experiment are set as shown in Table 1. For genetic operation, 1 point crossover, bit-flip mutation and elitism are used. Roulette wheel selection is adopted as selection mechanism, 5 evaluation degrees that is one the most widely used is applied [14].

Table 1. Parameters of experimental environments

Genetic Operator	Value
Population Size	10
Maximum Generation	10
Crossover rate	0.90
Mutation rate	0.05

5.2. Experimental Results

● Convergence

We have conducted with 3 kinds of fitness evaluation, only human, only the detection result of LinSTAT, and combining above them, and Figure 7 shows the changes of fitness on average and the best evolved 70 times.

In Figure 7, despite of shrinking the average and best fitness value, as generations are repeatedly, two values grows gradually after 4 generation. These values shrink first 2, 3 generation, because intrusion patterns generated former generation are detected by LinSTAT, so they are scored lower. However, as generations are repeatedly, they have higher fitness value than before. It is regarded generated intrusion patterns later as newer one and available. Exceptionally, the derivation of fitness value is unusual when evaluating only LinSTAT, intrusion patterns are just scored whether detected or not despite of not available.

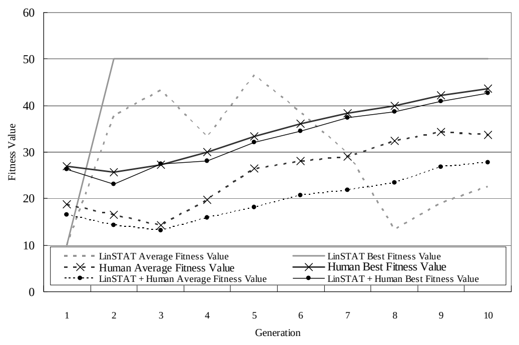


Fig. 7. Fitness changes

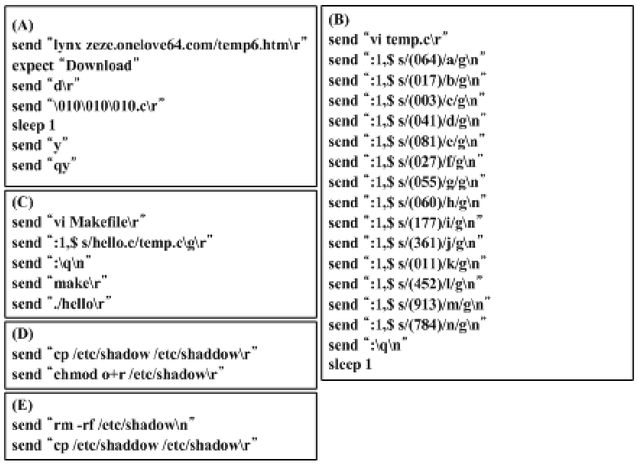
● Variety

Figure 8 shows an example of generated intrusion by using proposed method. In (A), temp6.htm, 010.c encoded previously are transported using HTTP protocol to victim system. (B) decodes the encoded files, (C) makes exploit code to execute. The exploit codes are executed simultaneously when specific file is complied by executing Makefile included exploit file name. In (D), the first line indicates the procedure to copy original file, the permission of original file is changed by the second line. When intruder achieves their aims, he/she must destroy their traces like (E).

The identical intrusion patterns are about 27% (about 130) as shown Table 2. Therefore, we can notice that various and newer intrusion patterns are generated about 73% using proposed method.

Table 2. Frequency of identical patterns

No.	Genotype	Frequency
1	01101101100001 01001	53 (10.6%)
2	10010110010111 00011	35 (7.0%)
3	10110010110000 00001	27 (5.4%)
4	01000001011000 00011	12 (2.4%)
5	11101111111010 10101	7 (1.4%)



```
(A)
send "lynx zeze.onelove64.com/temp6.htm\r"
expect "Download"
send "d\r"
send "\010\010\010.c\r"
sleep 1
send "y"
send "qy"
```

```
(B)
send "vi temp.c\r"
send ":1,$ s/(064)/a/g\n"
send ":1,$ s/(017)/b/g\n"
send ":1,$ s/(003)/c/g\n"
send ":1,$ s/(041)/d/g\n"
send ":1,$ s/(081)/e/g\n"
send ":1,$ s/(027)/f/g\n"
send ":1,$ s/(055)/g/g\n"
send ":1,$ s/(060)/h/g\n"
send ":1,$ s/(177)/i/g\n"
send ":1,$ s/(361)/j/g\n"
send ":1,$ s/(011)/k/g\n"
send ":1,$ s/(452)/l/g\n"
send ":1,$ s/(913)/m/g\n"
send ":1,$ s/(784)/n/g\n"
send ":\q\n"
sleep 1
```

```
(C)
send "vi Makefile\r"
send ":1,$ s/hello.c/temp.c\g\r"
send ":\q\n"
send "make\r"
send "./hello\r"
```

```
(D)
send "cp /etc/shadow /etc/shaddow\r"
send "chmod o+r /etc/shadow\r"
```

```
(E)
send "rm -rf /etc/shadow\n"
send "cp /etc/shaddow /etc/shadow\r"
```

Fig. 8. An example of generated code

6. CONCLUDING REMARKS

In this paper, we have proposed method to generate intrusion pattern automatically by interactive genetic algorithm. There are various kinds of domains which use IGA to solve the problem suck as arts or design. However, this paper is the first one that proposes the method using IGA in computer security. The proposed process generates intrusion patterns using IGA and evaluates them with the detection results of LinSTAT and human. Experiments show that generated intrusion patterns are available and various ones are generated. However, generated intrusion patterns are limited to LinSTAT. Therefore, we have to devise to generate general, available and various intrusion patterns..

ACKNOWLEDGEMENT

This work was supported by 21st Century Frontier R&D Program of Korea Ministry of Science and Technology**.**

REFERENCES

[1] B. Mukherjee, L. T. Heberlein and K. N. Levitt, "Network intrusion detection," *IEEE Network,* vol. 8, no. 3, pp. 26-41, May/June 1994.

[2] C. Dowel and P. Ramstdet, "The computer watch data reduction tool," *In Proc. Of the 13th National Computer Security Conf.,* pp. 99-108, Wahsington DC, USA, Oct. 1990.

[3] T. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood and D. Wolber, "A network security monitor," *In Proc. of the 1990 IEEE Symp. On Research in Security and Privacy,* pp. 296-304, 1990.

[4] J. McHugh, "The 1998 Lincoln laboratory IDS evaluation a critique," *Recent Advances Intrusion Detection 2000,* pp. 145-161, Oct. 2000.

[5] D. Libes, *Exploring expect: A Tcl-based toolkit for automating interactive programs,* O'Reilly & Associates, 1994.

[6] B. Schneier, *Secrets and lies: Digital security in a networked world,* john Wiley & Sons, Aug. 2000.

[7] P. A. Porras, "A state transition analysis tool for intrusion detection," *Master's Thesis,* University of California, Santa Barbara, 1992.

[8] A. Baur and W. Weiss, "Audit trail analysis tool for systems with high demands regarding security and access control," *Technical Report,* Nov. 1988.

[9] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning,* Addison-Wesley: Reading, MA, 1989.

[10] J. A. Biles, "GenJam: A genetic algorithm for generating jazz solos," *In Proc. of Int'l Computer Music Conf.,* pp. 131-137, 1994.

[11] C. Caldwell and V. S. Johnston, "Tracking a criminal suspect through face-space with a genetic algorithm," *In Proc. of 4th Int'l Conf. Genetic Algorithm,* pp. 416-421, 1991.

[12] Y. Nakanishi, "Capturing preference into a function using interactions with a manual evolutionary design aid system," *In Genetic Programming,* 1996.

[13] K. Ilgun, R. Kemmerer and P. Porras, "State transition analysis: A rule-based intrusion detection system," *IEEE Trans. on Software Engineering,* vol. 2111, no. 3, pp. 181-199, Mar. 1995.

[14] H. Takagi, "Interactive Evolutionary Computation: Fusion of the Capacities of EC Optimization and Human Evaluation," *Proc. of the IEEE,* vol. 89, No. 9, pp. 1275-1296, 2001.