Proceedings

MITA 2006

The 2006 International Conference on
Multimedia,
Information
Technology and its
Applications

July 4 - 6, 2006
Dalian University of Light Industry, Dalian, China

Korea Multimedia Society (KMMS), Korea

Dalian University of Light Industry, Dalian, China
Fang Liu
Software Institute of Dalian Jiaotong University

E-7. EFFICIENT AUTHENTICATION PROTOCOL BASED ON CERTIFICATELESS SIGNATURE IN VEHICULAR AD-HOC NETWORKS

Chae-Duk Jung*, Chul Su*, Kyung Hyune Rhee**
*Department of Information Security, Pukyong National University
**Department of Computer Science, Pukyong National University
***Division of Electric, Computer and Telecommunications Engineering, Pukyong National University

E-8. The Research of the PNG Image Encryption Algorithm on the Mobile Platform

Lin Feng*, Lijun Wang*, Tao Sun*, Yuze Sun*
*Institute of University Students’ Innovation, Dalian University of Technology

E-9. A STUDY ON INFORMATION SECURITY SYSTEM EVALUATION PARADIGM

Kab-seung Koo*, Sung-il Han*, Su-yoon Ahn**, Kyu-min Cho***, Gang-soo Lee*
*Dept. of Computer Engineering, Hannam University
**Korea Information Security Agency

E-10. THE VULNERABILITY ANALYSIS FOR BUFFER OVERFLOW USING CIL

*Division of Computer Science and Computer Engineering, SoonChunHyang University
**National Security Research Institute

E-11. AN EMPIRICAL STUDY ON RATING LEVELS IN RISK ASSESSMENT METHODS

Ik-hwan Bae*, Yeon-hee Ji*, Gang-soo Lee*
*Dept. of Computer Engineering, Hannam University

E-12. New Modular Exponentiations for Fast Montgomery Algorithm

Yilan Bao*, Ying Lu*, Xiaocong Wei*
*School of Information Science & Engineering, Dalian Institute of Light Industry

Theme F. Multimedia Networks and Communications

F-1. A FAIR CONTENT SHARING MODEL BASED ON FILE REPUTATION

Jung Hwa Shin*, Kyung Hyune Rhee**
*Dept. of Computer Science, Pukyong National University, Korea
**Division of Electronic, Computer and Telecommunication Engineering,
THE VULNERABILITY ANALYSIS FOR BUFFER OVERFLOW USING CIL

Doo-Soon Park*, Jae-Sin Hong*, Su-Hyong Son*, Kibom Kim**, Hyungwoo Kang**, Soonjwa Hong*, Min Hong*

*Division of Computer Science and Computer Engineering, SoonChunHyang University
**National Security Research Institute
E-mail: parkds@sach.ac.kr, zzuri@hotmail.com, sonsuhyong@nate.com, kibom@etri.re.kr, kanghw@etri.re.kr, hongjsj@etri.re.kr, mhong@sach.ac.kr

ABSTRACT

Over the past decade, the demands for building secure software have become a dominant goal for software development. Although great advances have been made in this area, the core problems of security vulnerability still exist. Buffer overflows are the most common source of security vulnerabilities in C program. To analyze these security vulnerabilities, we used a CIL (C Intermediate Language) that is written in OCaml (Objective Caml). We used well-known open test data such as Sendmail, Wu-ftpd, Bind, Perl on the Linux operating system. A C syntax is created to analyze the warnings and we implement the pushdown automata to check the generation of actual buffer overflows. But there was no actual buffer overflow during the source code analysis. Therefore, we conclude that the CIL does not provide the accurate buffer overflow measurement and an efficient new tool is essential for buffer overflow measurement.

Keywords: vulnerability analysis, buffer overflow, CIL, open source code, pushdown automata

1. INTRODUCTION

In knowledge-based information society, most advanced countries invest their concerns to nourish the information and knowledge industry and to secure the key technologies. In addition, the significance of information security to protect and iterate the data is no less than acquisition of the information and knowledge. Since the malicious virus, worm, spy-ware, and hacking are exponentially and swiftly spread in the recent internet world, the robust information security is becoming a hot issue in computer communications. These security strategies include firewalls, intrusion detection mechanisms, honey pots, port monitors, system security scanners, and e-mail content scanners [1].

The vulnerabilities of software security are formed from the mistakes in the stage of embodiment for function statements and in the stage of design or implementation or from the problems in operation or management. The technique to reduce the vulnerabilities of software detects and removes these weak points from the software development stage or utilization stage, so this process, technology, or tool minimize the attempts of attacking and provide the safe environments for software development and utilization. Therefore, software developers or users can detect and remove the vulnerable points in advance before taking advantages by hackers.

The tools for detecting the vulnerable points of software have been developed on various purposes. Although the big break through has been made for this filed, the security problems are still remained in the recent internet community. The most terrible and malicious vulnerability of software is a buffer overflow. Currently, many static code checkers that detect the buffer overflow are available. These checkers detect problematic code using the propriety heuristics to look for suspicious code segments, calls to specific utilities known to have vulnerability issues, or a combination of both. BOON [2], FlawFinder [3], ITIS4 [4], and MOPS [5] are these kinds of static code checker. All of these tools examine the source code and report the possible or potential weaknesses.

In this paper, we used a CIL program to test the well-known open codes such as Sendmail, Wu-ftpd, Bind (DNS system), Apache (web server), Perl (development language) on the Linux operating system. We double checked the warnings for actual occurrences of buffer overflow which are already detected by a CIL.

2. RELATED WORK

The static code analysis approach which is one of security vulnerability analysis methods for the source code detects potential security problems through the source code analysis. The static code analysis approach are focusing on the buffer overflow, race condition, malicious code insertion functions, format string problems, executable macro, and so on.

Buffer overflows are the most common source of security vulnerabilities in C programs. When a string of characters of unchecked length is entered into a program, buffer overflow attacks occur in programs. These attacks
modify the value of a return address from a function call and it causes the control to jump to malicious code which was also entered via the buffer overflow [3]. Some solutions are declaring all local variables in C as static to keep them from the stack [1]. One of useful tools is "StackGuard" which is a modification of the standard GNU C compiler gcc. StackGuard plays a role by inserting a "guard" value (called a "canary") in front of the return address. If a buffer overflow overwrites the return address, the canary value changes and the system detects this problem before using it [3].

The standard solution for preventing the buffer overflows in C programming is a utilization of standard C libraries which provide the protection about these problems. Standard C functions that do not range checking of character string inputs are vulnerable for the attacks of function algorithm. These functions include scanf(), gets(), sprintf(), vaprintf(), strcpy(), and strcat() [6].

The source codes are analyzed by a CIL tool based on the Linux operating system. Linux was developed by Linus Torvalds as an operating system kernel in 1991. This kernel could be combined with the FSF material and other components (in particular some of the BSD components and MIT's X-windows software) to produce a freely-modifiable and very useful operating system. The CIL is a high-level representation along with a set of tools that permit easy analysis and source-to-source transformation of C programs [8]. The main advantages of CIL is that it compiles all valid C programs into a few core constructs with a very clean semantics. Furthermore, the CIL front-end is able to process not only ANSI-C programs but also those using Microsoft C or GNU C extensions. CIL computes the types of all program expressions, and makes all type promotions and cast explicit. Thus CIL provides an integrated program representation that can be used with routines that require an AST, as well as with routines that require a CFG (e.g., dataflow analyses). CIL also supports even lower-level representations (e.g., three-address code) [8].

The most common way to use CIL is to write an Ocaml module containing your analysis and transformation. Objective Caml (OCaml) is a general-purpose programming language descended from the ML family. OCaml shares the functional and imperative programming features of ML, but adds object-oriented constructs and has minor syntax differences.

3. APPLICATION AND ANALYSIS OF CIL

In this section, we describe a system structure for the security vulnerability analysis to apply the CIL tool and explain about the open source codes as the test data. The processes to apply these source codes into CIL are described. After acquisition of warnings from the CIL, we confirmed the buffer overflows which are really occurred in the system. We implemented a C syntax which generates buffer overflows and also implemented the deterministic pushdown automata for checking. This deterministic pushdown automata confirms the real occurrences of buffer overflows for warning messages which were generated by a CIL tool. Figure 1 shows the system structure for the vulnerability analysis.

The CIL requires the specific environments for running such as a Windows operating system or Linux operating system and we selected the Linux for open source code testing. Open source code mean software which provides the source code for its program. Thus programmers can readily modify the source code without any restrictions and can obtain freely. A good example of open source codes is Linux. These kinds of open source codes are started from the operating systems and networking systems and are extended to the GUI systems and Office programs for general users.

Typical open source codes are Bind (DNS system), Sendmail (email), Apache (web server), X window system, Perl (development language), Mozilla (web browser), Linux (operating system) and so on. We tested open source codes with Sendmail, Wu-ftpd, BIND, and Perl in this paper. Sendmail, Wu-ftpd, BIND, and Perl include 7123, 15541, 31034, 911877 lines for source respectively.

To test Wu-ftpd, we go after following steps:
1) Download and install the OCaml.
2) Unzip a CIL file and a bufferaccess.tar.gz file which include a static analysis program and install them.
3) Unzip Wu-fp files.
4) Typing of "./build lnx" will create the executable files.
5) Type "./build lnx CC=cilly CLFLAGS=-dobufferaccess" Here CFILGS is optional and we don't need to delete some files which are generated during the process and the other process is
6) Type "./build lnx"
2) Type "./build inx CC=cilly > wu-flpd.result"

This is the steps for Sendmail source code:
1) Star zvxf sendmail.8.13.5.tar.gz
2) Sod libsm
3) $ ./build
4) $ ./build cc=cilly > sendmail.result

Although testing of Sendmail source code with CIL creates some errors, there was no problem for testing results and the source of errors was libsm.

Here are steps for Perl source code:
1) $ rm -f config.sh policy.sh
2) $ sh configure -de
3) $ make
4) $ make test
5) $ make install

But it fails to achieve the results. For successful testing, we only used a following commend:
1) $ make cc=cilly > perl.result

Bind source code can be tested as following:
1) $ ./configure
2) $ make
3) $ make cc=cilly > bind.result

Using these results which are bunch of warnings from the CIL, we checked the spots which generate the warnings to confirm that they really create buffer overflows using the pushdown automata. This is the part of C syntax in our testing.

While (conditional expression) {statement; }

Conditional expression has a permission for relational expression and statement can be iteration statement, if statement, and assignment statement. Each variable is allowed to use array and pointer operation. The deterministic pushdown automata judge the actual buffer overflow in our test program. When the left value of relational operator in conditional expression is same with the left value of assignment statement in statement, the actual buffer overflows are not occur. Otherwise, the actual buffer overflows are happened.

For example, give C program is:

```
While (x[i++]==" V"){
  x[i++]=y[j++];}
```

or

```
While (*x++==" V"){
  *x++=*y++;
```

In this kind of C program, actual overflow is generated when the conditional statement which is in a while loop did not check the size of buffer for variable x. Therefore, the x variable in conditional statement and left variable of assign statement in body of while loop are not same variables.

When we tested with a Sendmail, around 3,000 warnings were occurred. We analyzed the source code for these warnings, but there was no actual buffer overflow. The wu-flpd also tested using 4 different cases: including library and flag, including library only, including flag only, without library and flag. Absence of library does not affect for the testing result, but absence of flag causes the different testing result. The wu-flpd created around 700 warnings, but there were no actual buffer overflow during the source code analysis.

Similarly although Perl and Bind generated 2,400 and 5,400 warnings respectively, there were no real buffer overflows. Therefore, we conclude that the CIL does not provide the accurate buffer overflow measurement and the efficient tools are essential for buffer overflow measurement.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we analyzed the security voluntarily of buffer overflow using the CIL tool. The characteristics of CIL are described in this paper and the source code of well-know open sources, Sendmail, Wu-flpd, Bin, Apache, Perl, are analyzed. To confirm the warning detected from the CIL, whether the actual buffer overflows are occurred or not, buffer overflows are generated using C syntax. The deterministic pushdown automata double check the real occurrences of buffer overflows.

According to our experimental results, the warnings of buffer overflows in CIL did not actually happen from our testing so sophisticate improvements are required to correctly utilize the CIL program. Therefore, a new enhanced analysis tool which can broadly be applicable for open sources codes is essential for correctly test the buffer overflow.

5. REFERENCES

variable of the same type.

And 3,000

type

does not affect

the

type.

The

and 700

of

during

400

and

buffer

do not

and the

overflow

K

arity of

istics of

t and

Bind,

arnings

flows

using a

double

arnings

ation from

related to

a new

dicable

uest the

for the

flware

ference

7-202,

en, "A

over-

ework

(DSS),

 Unix

Oct.


Static Vulnerability Scanner for C and C++ Code,


Properties(MOPS), Computer Science Division,

University of California Berkeley,


Analysis and Transformation of C Programs",

Conference on Compiler Construction, pp213-228,

2002

- 171 -