



KOREAN SOCIETY FOR INTERNET INFORMATION

The 15th Asia Pacific International Conference on Information Science and Technology (APIC-IST 2020)

July 5-7, 2020, Sheraton Seoul Palace Gangnam Hotel, Korea

<http://www.apicist.org/2020>

Proceedings of APIC-IST 2020

| Organized by |

**Korean Society for Internet Information (KSII)
The University of Queensland (Australia)**

| Sponsored by |



Contents



10-8	Research on Response to Malicious Insider Threats using Feature Importance Analysis Dong-Wook Kim, Gun-Yoon Shin, Sung-Sam Hong (Gachon Univ., ROK)	263-264
10-9	Evaluation of Weather data, Power Demand and SMP Prediction Model by using LightGBM Yun-Jae Shin, KangYoon Lee (Gachon Univ., ROK)	265-267
11-1	An Efficient Resource Provisioning Scheme based on Machine Learning for Heterogeneous Internet of Things Sa Math, Prohim Tam, Chaebeen Nam, Seokhoon Kim (Soonchunhyang Univ., ROK)	268-270
11-2	Financial Fraud Detection adopting Distributed Deep Learning in Big Data Priyanka Purushu (AT&T, USA), Jongwook Woo (California State Univ., Los Angeles, USA)	271-273
11-3	BP Neural Network based RFID Indoor Location Method Chengbo Zhang (Nanjing Univ. of Information Science and Technology, China), Haoran Pei (Anqing Normal Univ., China), Zilong Jin (Nanjing Univ. of Information Science and Technology, China), Lejun Zhang (Yangzhou Univ., China)	274-277
11-4	Gaussian Process for Uncertainty in Predictive Analysis Oakyoungh Han, Jaehyoun Kim (Sungkyunkwan Univ., ROK)	278-280
11-5	Cross-Architecture Linux Malware Classification Method based on Binary Lifting Hae-Seon Jeong, Jin Kwak (Ajou Univ., ROK)	281-284
11-6	Detecting Technique of Rowhammer Malware Based on Static Analysis Minkyung Lee, Deuk-Hun Kim, Jin Kwak (Ajou Univ., ROK)	285-289

Cross-Architecture Linux Malware Classification Method based on Binary Lifting

Hae-Seon Jeong¹ and Jin Kwak²

¹ISAA Lab., Department of Computer Engineering, Ajou University
Suwon, Republic of Korea

²Department of Cyber Security, Ajou University
Suwon, Republic of Korea

[e-mail: hsjeong.isaa@gmail.com, security@ajou.ac.kr]

*Corresponding author: Jin Kwak

Abstract

With the growth of Internet of Things(IoT) market consisting of embedded devices, the number of linux malware targeting embedded devices is also increasing. Although malicious code classification is necessary to determine the threat posed by linux malware and respond, it is difficult to apply binary code similarity-based malware classification methods to linux malware as linux malware use different sets of instructions consisting binary code due to the various architectures. However, binary code similarity-based malicious code classification can be performed if an instruction performing the same function can be identified regardless of architectures. Accordingly, this paper proposed cross-architecture linux malware classification method by using binary lifting method which converts binary code with various architectures into IR(Intermediate Representation) of same format and defining rules for identifying functions of instructions.

Keywords: Malware Classification, Cross-Architecture, Linux Malware, Binary Lifting

1. Introduction

Along with the growth of the IoT(Internet of Things) market, which consists of embedded devices, linux malware targeting embedded devices is continuously evolving and increasing. Malicious code classification is required to determine and respond to potential threats inherent in malicious code. However, linux malware consists of 10 more different architectures, such as [Table 1 \[1\]](#). Therefore, there are various sets of instructions that make up binary code. As a result, it is difficult to apply binary code similarity-based malicious code classification methods to linux malicious code classification [\[2\]](#).

Table 1. Ratio of Linux Malware by Architectures

Architecture	Samples	Percentage
X86-64	3018	28.61%
MIPS I	2120	20.10%
PowerPC	1569	14.87%
Motorola 68000	1216	11.53%
Sparc	1170	11.09%
Intel 80386	720	6.83%
ARM 32-bit	555	5.26%
Hitachi SH	130	1.23%
AArch64 (ARM 64-bit)	47	0.45%
Others	3	0.03%

However, if the functions performed by the instructions can be identified in various architectures, binary code similarity-based malicious code classification can be performed. Accordingly, this paper introduces B2R2, a binary lifting tool that converts the sets of instructions of various architectures into the

same format in the related study of section 2. In section 3, features for identifying function of instructions are derived and the binary code similarity-based classification method is proposed and in section 4, conclusion is made.

2. Related Works

2.1 Binary Lifting

The technique of converting binary code consisting of a low level instruction set into high level IR(Intermediate Representation) is called binary lifting. By performing binary lifting, different sets of instructions can be represented in the same form of IR for various architectures.

2.2 B2R2

Among the binary lifting tools, B2R2 [3] supports seven linux architectures, including X86-64 and MIPS I, which account for a high percentage of linux malware. In addition, by using F#, a functional programming language, it can efficiently perform binary lifting by providing multi-core parallelism which simultaneously utilizes multiple CPU cores by not relying on the external environment.

Also, LowUIR, the IR of B2R2, stores informations about variables used and functions performed for efficient data analysis. The structure of the LowUIR is as shown in Fig. 1.

METADATA	μ	1100	ExprInfo * ConsInfo
			ExprInfo
ENDIAN	ϵ	1100	BEndian LEndian
UNOP	$\diamond_u$	1100	NEG NOT
BINOP	$\diamond_b$	1100	ADD SUB MUL DIV SDIV MOD SMOD SHL SHR SAR AND OR XOR CONCAT
RELOP	$\diamond_r$	1100	EQ NEQ GT GE SGT SGE LT LE SLT SLE
CASTOP	$\diamond_c$	1100	ZeroExt SignExt
EXPRESSION	exp	1100	Num value size
			Var name size
			PCVar name size
			TempVar name size
			Name name
			UnOp $\diamond_u$ exp μ
			BinOp $\diamond_b$ exp exp μ
			RelOp $\diamond_r$ exp exp μ
			Load ϵ size exp μ
			ITE exp exp exp μ
			Cast $\diamond_c$ size exp μ
			Extract exp pos size
			Undefined size
STATEMENT	$stmt$	1100	ISMark addr len
			IEMark addr
			LMark name
			Put exp exp
			Store ϵ exp exp
			Jump exp
			Clmp exp exp exp
			InterClmp exp exp exp
			SideEffect SideEffect

Fig. 1. Structure of the LowUIR

Fig. 2 is a result of the binary lifting of the ‘xor’ instruction in MIPS I architecture. It starts with ‘IsMark’ and ends with ‘IEMark’ STATEMENT. EXPRESSION used by ‘Put’ STATEMENT is ‘Var’ and ‘BinOp’ in order. BINOP ‘XOR’ used in ‘BinOp’ EXPRESSION means that the instruction performs xor operation.

```
[[ISMark (0UL,4u);
Put
  (Var (32,9,"R9",MIPSRegisterSet<200, 0>),
  BinOp
    (XOR,32,Var (32,9,"R9",MIPSRegisterSet<200, 0>),
    Var (32,10,"R10",MIPSRegisterSet<400, 0>),
    { HasLoad = false
      VarInfo = MIPSRegisterSet<600, 0>
      TempVarInfo = set [] },None));
IEMark 4UL]]
```

Fig. 2. Binary Lifting Result of ‘xor’ Instruction of MIPS I

3. Proposed Method

3.1 Process of Classification Method

This paper derived features that can identify the function of instructions from the LowUIR generated as a result of the binary lifting by B2R2, and proposed automated malicious code classification methods. This classification method targets the X86-64 and MIPS I architecture, which account for about 50% of linux malware, and the test environment of B2R2 is as shown in Table 2. The full process of the binary lifting-based linux malware classification method proposed by this paper is as shown in Fig. 3 and detailed process is as follows.

Table 2. Test Environment of B2R2

Tool	Version	Function
B2R2	0.3.0	Binary Lifting
.Net Core SDK	3.1	OS Independency
Ubuntu	18.04	OS

Step 1. Binary Lifting by B2R2

Binary Lifting is carried out using B2R2 that efficiently performs binary lifting and generates IR that reflect data flow. The binary code of the X86-64 and MIPS I is represented by the same format of LowUIR as a result of binary lifting

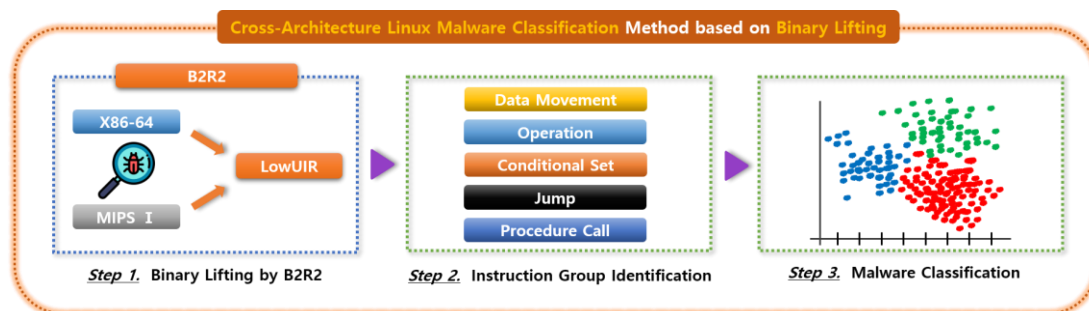


Fig. 3. Process of Classification Method

Step 2. Instruction Group Identification

It is difficult to identify function of instructions only with LowUIR generated as a result of binary lifting because even instructions with same function are expressed in various LowUIR STATEMENTS depending on the architectures. Thus, instructions performing common function [4,5] are classified into the groups of Table 3, and features for identifying each group are derived from the LowUIR. Features that can identify the instruction group are derived, excluding 'IsMark' and 'IEMark' STATEMENTS.

Table 3. Groups of Instructions

Group Name		Instructions of Architectures	
		X86-64	MIPS I
Data Movement		cmove, cmovne, cmovs, cmovns, cmovg, cmovge, cmovl, cmovle, cmova, cmovae, cmovb, cmovbe, mov, lea	mflhim, mflo, lw, mtlo, mthi, move
Operation	Binary	add, adc	add, addi, addu, addiu
		sub, sbc	sub, subu
		mul, imul	mult, multu
		div, idiv	div, divu
		xor	xor, xori
		and	and
		or	or, ori
	Shift	sal, shl	sll, sllv
		shr, shr	srl, sra
Conditional Set		setl, setnge, setne, setnz	slt, sltu, srl
Jump		jmp, je, jz, jns, jg, jne, jc, jnc	jr
Procedure Call		call	jal

Features for Identifying Data Movement

The LowUIR of the instructions that performs the data movement function only include 'Put'

STATEMENT, and EXPRESSION within 'Put' STATEMENT starts with 'Var' and ends with 'Var'. Since data movement function can be performed in other groups performing other function, it is identified as a data movement group when it satisfy each features, without falling under the other instruction groups.

Features for Identifying Operation

- Binary Operation : The lowUIR of the instruction performing binary operation has the 'Binop' EXPRESSION of 'Put' STATEMENT and has the corresponding operation value as the BINOP value : 'ADD', 'SUB', 'MUL', 'DIV', 'SDIV', 'XOR', 'AND' and 'OR'. The corresponding BINOP value of the operation is used only once, and 'BINOP' is not used as EXPRESSION within 'BINOP' EXPRESSION.

- Shift Operation : The lowUIR of the instruction performing shift operation has the 'Binop' EXPRESSION of 'Put' STATEMENT and has corresponding operation value as the BINOP value : 'SHL', 'SHR'

Features for Identifying Conditional Set

The lowUIR of the instruction performing the function of setting the value according to the condition has 'RelOp' EXPRESSION.

Features for Identifying Jump

The lowUIR of the instruction performing the function of jump only has 'InterJump' or 'InterCJump' STATEMENT.

Features for Identifying Procedure Call

The lowUIR of the instruction performing the procedure call starts with 'Put' STATEMENT and ends with 'InterJump' STATEMENT.

Step 3. Malware Classification

Classify linux malware by learning the sequence of calls from identified instruction groups using a deep learning model. As a result, automated classification can be performed based on binary code similarity for linux malware in various architectures.

3.2 Performance Comparison

Various researches have been proposed to deal with challenges of linux malware classification caused by cross-architecture. However, most of them deal with the features of ELF file, not the decompiled instruction set [2,6].

ELF is an executable file of linux OS. However, there are challenges of using features of ELF file in Linux malware classification. If Header of ELF has been modified, ELF file cannot be parsed and thereby cannot be analyzed. Also, if ELF file has been obfuscated or packed, then ELF file cannot be analyzed for linux malware classification [1].

However, by using decompiled instruction set on linux malware classification, challenges caused by ELF file can be handled. CW Tien, SW Chen, T Ban and SY Kuo has classified Linux malware by using ELF features and decompiled instruction set at the same time [7]. However, they didn't identified all instruction functions and not all instruction set. However, by using this proposed method, all common instruction functions of the X86-64 and MIPS I can be identified and all instruction set can be identified into proper groups. Thereby, this method can be used for cross-architecture linux malware classification using instruction set.

Table 2. Results of Comparison

Linux Malware Classification Research	Instruction Set Used
J. Hwang and T. Lee [2]	X
D. Jeon and D. Park [6]	X
CW Tien, SW Chen, T Ban, and SY Kuo [7]	△ (Partial Identification)
Proposed Method	○

Although this paper addressed only the X86-64 and MIPS I, it is scalable to various architectures by applying method presented by this paper to other architectures other than X86-64 and MIPS I. It is also meaningful that existing binary code similarity-based malicious code classification methods can be used by automatically identifying the function of instructions for various architectures.

References

- [1] E. Cozzi, M. Graziano, Y. Fratantonio and D. Balzarotti, "Understanding Linux Malware", *2018 IEEE Symposium on Security and Privacy*, pp. 161-175, May. 2018.
- [2] J. Hwang and T. Lee, "Study of Static Analysis and Ensemble-Based Linux Malware Classification", *Journal of The Korea Institute of Information Security & Cryptology*, Vol. 29, No. 6, pp. 1327-1337, Dec. 2019.
- [3] M. Jung, S. Kim, H. Han, J. Choi and S.K. Cha, "B2R2: Building an Efficient Front-End for Binary Analysis", *2019 NDSS Workshop on Binary Analysis Research (BAR)*, Jan. 2019.
- [4] L. Chris, "Introduction to x64 Assembly", *Intel*, Mar. 2012.
- [5] C. Price, "MIPS IV Instruction Set", *Carnegie Mellon University*, Sep. 1995.
- [6] D. Jeon and D. Park, "Real-time Linux Malware Detection Using Machine Learning", *Journal of Korean Institute of Information Technology*, Vol. 17, No. 7, pp. 111-122, Jul. 2019.
- [7] CW Tien, SW Chen, T Ban and SY Kuo, "Machine Learning Framework to Analyze IoT Malware Using ELF and Opcode Features", *Digital Threats: Research and Practice*, No.5, Mar. 2020.

4. Conclusions

In this paper, the features that can identify the instruction group are derived from binary lifting result by B2R2 and it proposed the malware classification method that receives them as input.