ATLANTIS PRESS
Advances in Computer Science Research, volume 50
4th International Conference on Electrical & Electronics Engineering and Computer Science (ICEEECS 2016)

Research of ROP attack and defense technology based on ARM architecture

Qi Cai，Jingbo Guo

Pingdingshan Institute of Education, 467000, Pingdingshan Henan China

Keyword: ARM architecture, ROP, attack and defense,technical research

Abstract: In recent years, with the development of mobile technology,almost all the smart phones and panel personal computers using the ARM architecture, and security issues on this platform arealso getting more and more attention from researchers. Attack of malicious tampering program flow control is one of the major attacks for the platform. Attackers use thevulnerability of stack overflow to write long data in internal storage which make it beyond its buffer capacity, cover the return address, and thus transfer to the attacker's malicious control flow. To defend this type of attack,the operating system was added with W + X protection mechanisms,viz.dataExecution Prevention (DEP). The hardware was realized by IntelXD(Execute Disable bit)technology, ND(No-Execute Page-Protection) of AMD technology and XN(eXecute Never) ARMtechnology. Return-oriented programming (ROP) isbased on this proposed defense technology.

Introduction

Since the concepts of return-oriented programming waspproposed by Shacham in 2007. ROP technology has become a hot research for university institutions at home and abroad, including researchROP attacks on various platforms, such as the mainstream X86 platform, ARM platform, SPARC platform and so on.Tim Kornau first proposed to build the Turing completeness of ROP attacks on the ARM architecture and proposeda set of algorithm for automatic constructionROP chainbased on REILlanguage. Erik Buchanaz et al. improved the Turing completeness of generating ROP gadgets for the first time at the University of California, San Diego in 2008. Later, Ryan Roemer followed Erik Buchanan's research to further studythe implementation of ROP on X86 and SPARC architectures.

Review of related researches

For the research on return-oriented programming technology, domestic and foreign researchers mainly focus on three aspects: (1) studyon ROP attack principle on each platform; 2) Platform-independent automatic construction of ROP attack chain;(3)study on ROP defense technology. The following three aspects are the ways to analyze the research situationat home and abroad as below.

Study on ROP attack principle on each platform. Since the concepts of return-oriented programming waspproposed by Shacham in 2007. ROP technology has become a hot research for university institutions at home and abroad, including researchROP attacks on various platforms, such as the mainstream X86 platform, ARM platform, SPARC platform and so on.Tim Kornau first proposed to build the Turing completeness of ROP attacks on the ARM architecture and proposed thealgorithm of automatic constructionROP chainbased on REILlanguage. Erik Buchanaz et al.

improved the Turing completeness of generating ROP gadgets for the first time at the University of California, San Diego in 2008. Later, Ryan Roemer following Erik Buchanan's research, further studies the implementation of ROP on X86 and SPARC architectures.

Platform-independent automatic construction of ROP attack chain. Automatic construction of ROP chain isone of important parts in the ROP attack technology. Because the way of finding ROP chaininstruction is complex, so a set of automatic construction chain tool was very helpful for the research of available instruction sequence and the acceleration of vulnerability development. In the field of automatic construction of ROP attack chain. Ryan Glenn Roemer et al. from University of Californiafirstly realized the method of automatic construction. Later, Thomas Dullien et al. proposed a platform-independentgadget search framework by using the REIL intermediary language for static analysis the disassemblingcode, making the gadget searchout of the restriction of platform and applying more widely. Tim Kornau applied the platform-independentframeworkthat based on REILlanguage into the ARM architecture, realizing theautomatic search of ROP chain gadget on ARM architecture.

Study on ROP defense technology. In the field of ROP defense technology, researchers have proposed many response programs at all levels. Kaan Onarlioglu et al. started fromthe compiler level, eliminating the return instruction {(ret instruction of X86)ys}from compiler level, thus erasing the binary code of the gadget, but this scheme cannot defense the JOP attacks. Kangjie Luet al. started fromthe static analysis of binary instructions, transforming the ROP attack sequence into non-ROP attack sequence. From the perspective of detection, and according to the characteristics of ROP attack techniques, many researchers use the binary instrumentation tool to detect theexecution behavior during the program execution and achievethe purpose of detecting ROP attack, such as ROPdefender and DROP detection tools.

ROP attack analysis under ARM architecture

ROP attack principle in ARM architecture. Kornau realized the ROP attack in ARM architecture for the first time in the literature. This paper provesexecutable code and system library code in search applicationenough to guide and finish the Turing completeness for any operation. And then Lucaset al. proposedthe use ofindirectjump instructionBLX reg to guide and finish JOP attack, and realized this attack in android 2.0system. In order to finish a ROP attack, the combination of instructionsegments exist in system library or executable region must be used. The most important feature of these instructionsegments is that the last instruction in each gadget is a jump instruction, such as “b1 1r”. Parameter between two gadgetsup and down operates instruction through internal storage, such as “ldmfdsp!, (r4, r7, lr)” , and it was completed by the last gadget writingstack parameter and the next gadget readstack parameter. Executive address and function parameter on the stack was meticulous designed to guide the gadgets to finish a completeattack. Since theARM cannot support the automatic updates of stackregister, therefore, the construction of gadget is needed to execute the update of stackregister. An example of ROP attack by applying ARMindirect jump instructionBLX, the whole process of ROP attack was shown as follow:

Process of ROP attack in ARMarchitecture. (1) First, theattackers use theprogram vulnerabilities to cover the address of gadget and return to the address for intercepting the program control flow. And regA = ULBaddressandregB=stack pointer were arranged. When the function returns, the program flow goes to the firstgadgetcontrolled by attacker, corresponding to steps 1 and 2; (2) The last instruction b regA of gadgetl jumps to ULB block (regA points to ULB address), since the last instruction in every gadget is an indirect jump, so a

trampolineULB(update-load-branch) is used to bridge two gadgets. This ULB block generally consists of 1-3 instructions, comleting the functions of updating stackregister (a), loading the next gadget block address (b), jumping (c), corresponding to the three steps. (3) ULB first updates the regB which makes the regBpoint to address 1, loading load regC, [regB], at this time, regC stores the address of gadget2, and branch regC skips to gadget2 for execution, corresponding to steps 4 and 5 in Figures 3-6.(4) Steps 6 and 7 are the repetition of steps 3 and 4, conducting above process successively untilROP chain is finished.

The algorithm of gadgets search

There is no public gadgets search tool based on ARM instruction set so far, Kornau proposed a platform-independent algorithm of automatic searchgadgets based on REIL language, but the whole algorithm is very complex.

By search methods of Kornau and Schacham, the realization under ARM platform in this paper was shown as follow: first, searching the net instruction flows in linklibrary ("bl/blxreg" [OxfflOx2flOxe 1], "pop{..pc}" [0x80\Oxbd\OxeB], these instructions equal to ret instructions in X86), 4 bytesassembleARM instruction stream reversely to arrive closing value or stop in no instruction. Since the fixed-length feature of ARM instruction, only the forward 4 byte was needed to search, and the search algorithm is as follow:

Search algorithm of availableinstruction (1):

Search_rret Sequence(code, depth) // Search rtet instructionCreat a trieseq

    For pos from 1 to code len// Search the whole code file

If3-bytes@pos inrret-seq //rretseq showed inbinary stream

    Call Buildwe Valid (pos, trie-seq, depth)

Search algorithm of availableinstruction (2):

Build-Valid(pos, root, depth)// assemble reversely indepthand length

for i=1 to depth

ifdis_arm(pos-4) is valid

    add-seq(pos, trie-seq)

Though the algorithm mentioned above can obtain theavailable sequence information for further analyze and combine these instructions and make it complete a particular operation gadget (such as mov/add/and/), lay the foundation of Exploit vulnerability in ROP chain.

According to the algorithm, the simulation experiment was carried out under the ARM platform in this paper, And the test environment was built as follows:

Host environment: intel(R) Core (TM) i5-2300 CPU @ 2.80GHz , 4.00 GB of memory

Virtualmachine environment:

VMware? Workstation 7.1.2build-301548, ubuntu-10.10-desktop-1386, android-sdk-r18-linux emulator, android 2.3of simulation platform. Table 2 shows thedistribution of rret instruction in ARM library of android 2.3 system.

Table 1 Conclusion of rret distribution in ARM system library

Linkbase	#bx instruction	#blx instruction	#pop{pc} instruction	Gadgets number
lihandroid.sn	17	1 239	317	1 587
lilx.so	812	620	R4R	3 779
lihdl.so	85	42	18	180
lihm.sn	180	27	200	577
IIhtPTT17.S0	0	196	43	316
lihutils.so	6	949	l 015	2 537

Implementation and test of automatic search algorithm

Implementation of defense algorism. We implement the algorithms on Intel X86 to randomize the object code base. Due to the open source of Linux and a variety of specification, we can exactlydisassembling the objectsource code. We use disasm library to decompile and identify short sequences ahead of ret and jmp transfer instruction, and implementrandomization according to above algorithm. In this process, three possible file modifications were involved, namely, short sequence ahead of replaceabletransfer instruction, padding of NOP instruction and replacement of function position. In the replacement process of function, our strategy is to try our best to replace the neighboring function. This will not influence theoffset of absolute address before and after the two functions. Meanwhile, some functions, taking f1 and f2 as a assumption, if the magnitude of the two functions satisfy the following conditions: sizeof(f1)⩽sizeof(NOP)+sizeof(f2),sizeof(f2) ⩽ sizeof(NOP)+sizeof(f1), then its position can be changed without causing the offset change of subsequent function caused a chain reaction.

After locating the address of function precisely, we use Libelf [Koshy, 2010] library to parse object code file, and replace the position of short sequences, filling NOP instruction as well as change the overall position for thenecessaryfunction. In the process of moving the wholefunction, all the reference address involved in this functionneeds to be changed, and thus the cost will be greater. So in our algorithm, we try to avoid this step to randomize theshort sequence, maximizing thereplacement of neighboringinstructions in short sequence and filling NOP, to ensure theequivalent of semantic function of object code before and after randomization.

After analyzing the dependency of No. 1, 2 instructions, the positioncan be changed, while the last two NOP instructions can be used to fill. From a security perspective, the filling of NOP instruction sequence can be implemented simultaneously after the replacement of replaceablesequences. During the modification of LibELF library on object code, we modify the file layout of executable code based on ELF, and modifying it from the semantics to ensure the semantic correctness of modified code.

Testing. We randomly tested multiplecodes library to analyze the position, type and percentage of these randomized sequences, which including three parts randomization, viz. firstly the serialization of short sequence, followed by the random padding of NOP and functions randomization. Our experimental data showed that mostof short sequences were randomized by thetwo former ways. The higher distributions of the two former, the better effect it gets, and the version of randomization more stable and reliable. But the randomization of functions can solve all the remaining problems in the short sequences randomization.

Table 2 Results of randomization test

Program	Size（kb）	Ins	NO.	Short sequence	NOP	Function	Short sequence and total NOP proportion
AdobeReader8	15676	Ret	5264	2724	107	2433	53. 78
		Jmp	18966	17448	1	1517	92. 00
libc. so. 6	1713	Ret	1192	796	99	297	75. 08
		Jmp	503	497	3	3	99. 40
libm. so. 6	173	Ret	92	81	6	5	94. 57
		Jmp	15	15	0	0	100
ld-linux. so	134	Ret	52	35	5	12	76. 92
		Jmp	35	32	0	3	91. 43
1 ibcrypto. so	1734	Ret	779	413	109	257	67. 00
		Jmp	221	219	0	2	99. 09
libglib. so	1014	Ret	758	520	53	185	75. 59
		Jmp	133	129	0	4	96. 99

Meanwhile, we have tested the existing ROP attacks of Adobe Reader[Schultz,2010] and ApacheTomcat[Liu,2011]. This two ROP attacks were terminated by operating system because of the failure of control flow in their own sequence. But the modified performance of Adobe Reader had no influence. Since the modification was static, so it will not impact thedynamic performance of the program. And the current module can be carried out after the program installation.

In the process of the experiment, some instructions required additional processing, like some non-operand instructions, which will not set the value of the general register, but to set theflag bit in the register. At present, we processed these instruction sequences in a non-conservative way. However, it has effect on the condition flag, so, special handling was needed. A certain number of this kind of sequences appears in Adobe Reader, but it rarely appears in other object code. However, this kind of code sequenceis almost useless for ROP. Therefore, this problem is not a problem for the defense of ROP. But, this also a research direction for the future, while in the instruction set of logical and mathematical calculations, our currently practices are more effective

The current implementation version still has to be improved. Currently, the analysis for the short sequences dependencyrequiresfiner granularity to identify more accurate dependency, especially the influence of register and instruction type on %efiag register in each flag bit. At present, many dependencyanalysis of the register are facing this problem. Although it has no influence on the function, it will affect the optimizing and conditional forecasting of instruction sequences, which need the optimizing in compile phase based onarchitecture. Secondly, the replacement of function position involves more complex condition, and the reliability and security is difficult to guarantee. Therefore, in our randomized algorithms, we try to improverandomization of short sequence and hit rate of NOP padding based on greed principle, reducing the call of randomized function. Next, we are going to set this kind of variable in the compile phase, providing infrastructure to support this randomization, and enhance thefeasibility and accuracy of function replacement.

At the same time, we considered that some basic facilities were reservedin the compile phase, such asoffset function, and increasing the number of patch code, and executing these codes when loading, in order to conduct the dynamic randomization of the address of the function and improve the granularity of randomization. Meanwhile, we believed that the current randomized algorithms can be placed in the last step ofinstallation program to improve a wider range of the safety of the procedure. At the same time, it will provide more help for our current algorithm by deliberately

conducting useful instruction selection and sorting for the sequence of randomization in the compile phase.

Conclusion

The randomization of address almost make the ROP attack useless, but the attack and defense technology is a game of "cat and mouse", the in-deep research on ROP have some practical significance, especially in the field ofcross-platform, such as ARM, SPARC, PowerPC and so on. Since the differences between platforms, how to develop a set of platform-independent automation ROP chain tools to acceleratethe process of Exploit has great practical significance, and research directions will focus on theautomatic construction of ROP chain in the future.

Reference

[1]PiotrBania. Security Mitigations for Return-Oriented Programming Attacks.Kryptos Logic Research .http://www.kryptoslogic.com. 2010.

[2] ZhiJun Huang, Tao Zheng, Jia Liu. A Dynamic Detective Method against ROPAttack on ARM Platform[C]. 2012 2nd International Workshop on SoftwareEngineering for Embedded Systems (SEES), Zurich, Switzerland. 2012:51-57.

[3]HristoBojinov, Dan Boneh, Rich Cannings, IliyanMalchev. Address SpaceRandomization for Mobile Devices. WiSec' 11，Hamburg, Germany. 2011; 14-17.

[4] Vasilis Pappas, MichalisPolychronakis, Angelos D. Keromytis. Smashing theGadgets: Hindering Return-Oriented Programming Using In-Place CodeRandomization.IEEE Symposium on Security and Privacy (SP). 2012: 601-615.

[5] Intel. 64 and IA-32 Architectures Software Developer's Manual Volume 2A:Instruction Set Reference, A-M.http://download.Intel.com/products/processor/manual/3253 83.pdf.

[6] Ma Xiangyu. Research on the key technology of binary translation. Shanghai: Institute of Computing Technology, Chinese Academy of Sciences.2014.

[7]Yu Xiaohong, Lu Yao. Buffer overflow research based on address space randomization in Linux. Computer Knowledge and Technology. 2011; 17: 90-93.