文章编号:1007-1423(2019)31-0015-06 DOI:10.3969/j.issn.1007-1423.2019.31.004

自动化漏洞利用研究进展

靳宪龙[1],黄雅娟[2]

(1.四川大学计算机学院,成都 610065;2.国防科技大学国际关系学院,南京 210012)

摘要:
网络空间安全斗争形式日趋复杂,针对软件安全的攻防博弈愈演愈烈。软件漏洞挖掘与利用的复杂性及专业性,使得大量工作仅能依靠安全专家完成。近年来,漏洞数量激增,仅依靠安全专家已无法有效应对。自动化漏洞利用应运而生,该方法在提升工作效率的同时降低人力成本,并一定程度上满足了自动化攻防的迫切需求。介绍自动化漏洞利用相关概念,对关键技术进行归纳与总结,梳理国内外主流的自动化漏洞利用系统。

关键词:
网络安全;二进制;漏洞利用;自动化

0 引言

近年来网络空间攻防对抗态势不断升级,网络空间安全的地位与作用进一步凸显。软件作为网络空间的基础之一,承载着网络空间中的各项服务,同时也充当着网络空间与现实世界之间的桥梁。在频发的各类安全事件中,无论是基于利益驱动的网络犯罪还是具有政治背景的 APT 行动,软件漏洞始终是对抗双方博弈的核心。

传统的漏洞挖掘与利用主要通过人工来进行,安全专家依靠完善的专业知识及丰富的安全经验,对软件中潜在的缺陷进行挖掘,判断其可利用性并制定修复方案。但随着各种半自动、自动化漏洞挖掘技术的提出,使得漏洞挖掘的效率大幅提升,仅谷歌的 OSS-Fuzz 框架每月平均挖掘 200 个漏洞。相较自动化漏洞挖掘的效率,以人工的方式对漏洞的可利用性进行判定已无法满足现实需求。

网络攻防也对漏洞利用的自动化提出了迫切需求。2013 年美国国防部高级研究计划局(DARPA)发起了 CGC(Cyber Grand Challenge)项目[1],旨在建立高性能的具备自动化攻击与防御能力的网络推理系统。获得 CGC 冠军的网络推理系统 Mayhem,在 2016 年的 DEFCON CTF[2]中与人类战队进行攻防对抗,虽最终落败,但已体现出自动化漏洞利用在网络攻防中的广阔前景。

1 自动化漏洞利用概述

自动化漏洞利用是指,在无人工干预的基础上,自动化挖掘软件内部缺陷,并利用该缺陷使软件实现非预期功能的能力。依据自动化漏洞利用的完整程度,可分为完整模式与受限模式。

完整模式,在自动化挖掘软件漏洞并生成利用样本的基础上,使得样本具备绕过现有安全机制的能力;受限模式,实现自动挖掘软件漏洞并生成漏洞利用样本,但不考虑安全机制绕过。完整模式的自动化漏洞利用难度较大,目前的研究主要集中在受限模式。

自动化漏洞利用主要包含以下环节[3]:

(1)漏洞挖掘。通过对软件源码或二进制代码进行分析,采用模糊测试、补丁比对等技术,挖掘软件内部潜在缺陷。

(2)漏洞分析。基于软件缺陷,通过污点分析、符号执行等技术,定位漏洞点、判明漏洞类型、澄清漏洞成因等关键信息。

(3)漏洞利用。在判明漏洞机理的前提下,利用该缺陷,使软件实现非预期的功能,通过恶意利用对安全造成直接危害。

(4)安全机制绕过。结合代码复用、信息泄露、堆风水等攻击技术,实现对 DEP、ASLR、CFI 等安全机制的绕过。

2 自动化漏洞利用相关技术介绍

随着软件分析技术的发展,提出了模糊测试、符号执行、污点分析等技术,这些技术已在现实工作中发挥了重要作用。

2.1 模糊测试

模糊测试是指向待测程序提供大量特殊构造的或是随机的数据作为输入,监视程序运行过程中的异常并记录导致异常的输入数据,基于导致异常的输入数据进一步定位软件中漏洞位置[4]。

依据测试用例的生成方法,模糊测试可以分为基于变异的模糊测试与基于生成的模糊测试。基于变异的模糊测试,通过对现有数据进行变异以生成新的测试用例。基于生成的模式测试通过对程序进行建模,从而生成新的测试用例。模糊测试已在漏洞挖掘领域发挥重要作用,但依然存在测试盲目性大、测试效率低、代码覆盖率不高等问题。

目前主流的模糊测试工具有 Peach[5]、Sulley[6]、AFL[7]等。

2.2 符号执行

符号执行是指采用抽象符号代替程序变量,依据程序语义遍历整个执行空间,程序的输出被表示为输入符号值的函数[8]。

符号执行可分为静态符号执行、动态符号执行与选择性符号执行。静态符号执行并不真正的执行程序,而是通过符号值模拟程序执行;动态符号执行是对具体执行与静态符号执行的整合,兼具了两种方法的优点;选择性符号执行可以对关键部分进行符号执行,其余部分采用真实执行。虽然符号执行技术具有较高的代码覆盖率,但面临着程序状态空间爆炸、复杂约束无法求解、资源消耗过大等问题。

目前,主流的符号执行工具有基于源码的 EXE[9]、KLEE[10]、DART[11]、CUTE[12],基于二进制的 angr[13]、SAGE[14]、Bitblaze[15]等。

2.3 污点分析

污点分析的核心思想是跟踪并分析污点数据在程序中的传播情况。污点分析可分为静态污点分析与动态污点分析。静态污点分析通过对源码或字节码中的语句进行分析,刻画语句或指令间的依赖关系并以此为依据,判断污点数据可能的全部传播路径。动态污点分析是在程序执行过程中,结合程序插桩跟踪污点数据的传播过程,相较于静态污点分析能够提供更精确的分析结果。虽然污点分析能够提供精确的分析结果,但其同样面临着隐式流问题、污点清除、性能开销过大等问题。

目前,主流的污点分析工具有 TaintCheck[16]、Dytan[17]、libdft[18]等。

3 主流自动化漏洞利用系统

国内外研究团队已针对自动化漏洞利用技术做了大量工作[19],本节按时间顺序对十款主流自动化漏洞利用系统进行介绍。

3.1 APEG

2008 年 D. Brumley 首次提出基于补丁比对的漏洞利用自动生成方法 APEG[20]。该方法在实现过程中,首先通过补丁比对定位漏洞点;随后分析补丁代码,澄清漏洞机理并生成能够触发漏洞的异常输入;最后,结合污点分析技术生成漏洞利用样本。由于该方案可操作性强,因此得到了安全研究者的普遍认可。

APEG 的不足主要体现在:首先,APEG 同时需要原程序与补丁程序,该前提一定程度上限制了 APEG 的应用场景。其次,由于软件复杂度的提升,简单的指令比对已无法快速定位漏洞点,同时与漏洞无关的补丁也对澄清漏洞机理造成了干扰;再次,APEG 所生成的漏洞利用样本以 DoS 为主,对漏洞的利用程度有限。

3.2 AXGEN

2009 年 Sean Heelan 首次提出基于异常输入的自动化漏洞利用方案 AXGEN[21]。该方法综合使用数据流分析、动态污点分析等技术,针对缓冲区溢出类漏洞,生成利用程序。

在实现过程中,AXGEN 首先对系统调用、线程创建、信号量等关键点进行 Hook;随后,采用污点分析对程序运行状态进行监控,分析指令寄存器 EIP 中污点数据情况,判断控制流是否被劫持;最后,生成约束条件,求解后得到漏洞利用样本。在测试过程中,AXGEN 对多个存在漏洞的应用程序成功生成漏洞利用样本。

该方法的局限性主要表现在,首先,AXGEN 不具

备漏洞挖掘能力;其次,不适用于较为复杂的漏洞类型;最后,由于采用了污点分析技术,因此 AXGEN 存在污点分析所固有的不足。

3.3 AEG

2011 年 T. Avgerinos 首次提出了基于源码的自动化漏洞利用方法 AEG[22]。该方法针对栈溢出、格式化字符串漏洞自动生成漏洞利用样本,是首个完整实现从漏洞挖掘到漏洞利用样本生成全流程的自动化解决方案。在实现过程中,AEG 首先对源码进行预处理,生成字节码与二进制程序;其次,通过前向符号执行挖掘程序中潜在的漏洞。根据挖掘到的漏洞信息构造路径约束并求解,生成能够触发漏洞的输入;再次,结合异常输入与动态分析,生成环境约束。最后,求解路径约束、环境约束,生成漏洞利用样本。在测试过程中,AEG 针对 14 款应用成功生成 16 个漏洞利用样本,其中包含两个 0 day。

该方法的局限性主要体现在:首先,依赖于程序源码,限制了 AEG 的应用场景;其次,所针对的漏洞限于简单的栈溢出与格式化字符串。

3.4 Mayhem

2012 年 S .K. Cha 提出了基于二进制的自动化利用方案 Mayhem[23],该方案可以看做 AEG 在二进制方向的拓展。在实现过程中,Mayhem 运用在线符号执行、离线符号执行以及基于索引的内存建模技术,使用 BAP[24]将汇编指令转换为中间语言并构造约束条件,通过对约束求解,最终生成漏洞利用样本。Mayhem 在 2016 年举办的 CGC 决赛中取得冠军。

虽然在实际应用中 Mayhem 表现出色,但同样存在以下局限:首先,仅完成了针对部分系统调用及库函数的建模,因此无法处理复杂应用;其次,无法处理多线程应用;再次,由于使用了污点分析,因此具备污点分析所共有的不足。

3.5 CRAX

2012 年 Shih-Kun Huang 等人提出了基于异常输入的自动化漏洞利用方法 CRAX[25]。该方法采用回溯分析获取漏洞点的详细信息,在实现过程中,首先基于异常输入,在导致程序崩溃的路径上进行符号执行;同时监测指令寄存器 EIP 及内存中符号变量情况,并生成相应的约束条件;最后,求解约束条件并生成漏洞利用样本。在测试过程中,该方法针对 16 款小型应用、3 款中型应用成功生成漏洞利用样本。

该方法的局限性主要体现在:首先,不具备漏洞挖掘能力,漏洞利用样本的生成依赖于异常输入的质量;其次,面临路径选择、符号求解等问题;最后,程序发生异常的位置往往滞后于真实错误的位置,虽然 CRAX 采用了回溯的方法,但部分信息的丢失将影响生成漏洞利用样本的成功率。

3.6 PolyAEG

2013 年 M. H. Wang 等人提出了自动化漏洞利用方案 PolyAEG[26],侧重于生成高质量、多样性的漏洞利用样本。在实现过程中,PolyAEG 首先动态监控程序执行状态,提取相关信息构建污点传播流图 iTPG 及全局污点状态记录 GTSR;其次,分析潜在的控制流劫持点、可利用的指令、污点内存等,结合跳转指令链与污点内存构建不同的约束条件;最后,对约束求解,生成漏洞利用样本。测试过程中,该方案针对每个存在漏洞的应用程序均生成大量多样性漏洞利用样本,最高可达 4724 个。

该方法的局限性主要体现在污点分析所存在的误报、漏报及性能开销较大。

3.7 FlowStitch

2015 年 H. Hu 首次提出了面向数据流的自动化漏洞利用方案 FlowStitch[27]。该方案在不影响程序控制流的前提下,通过已知内存错误构造输入数据,完成对原有数据流中关键变量的篡改,实现权限提升、信息泄露等功能。在测试过程中,针对 8 个存在漏洞的应用,自动生成 19 个漏洞利用样本。所生成的漏洞利用样本均能绕过 DEP、CFI,其中 10 个漏洞利用样本能够对抗 ASLR。

该方案主要局限于 FlowStitch 需要程序中存在已知的内存错误,该前提限制了 FlowStitch 的应用场景。

3.8 Mechanical Phish

2016 年来自加州大学圣芭芭拉分校的安全团队 Shellphish 提出自动化漏洞利用方案 Mechanical Phish[28]。该系统在实现过程中,首先由基于 AFL 与动态符号执行的漏洞挖掘引擎 Driller[29]进行自动化漏洞挖掘;其次,由基于 angr 的漏洞利用生成引擎 Rex 自动化生成漏洞利用样本;最后,由补丁生成引擎 Patcherex 生成补丁程序。该方案在 DARPA 举办的 CGC 决赛中,取得了第三名。

该方案的局限性主要体现在：首先，方案的设计以CGC为背景，所使用的操作系统DECREE、针对的漏洞类型均与真实环境差距较大，因此该方案在现实环境中的表现有待完善；其次，Mechanical Phish使用angr作为符号执行引擎，因此无法处理大部分真实应用。

3.9 FUZE

2018年Wei Wu首次提出了基于内核UAF漏洞的自动化方案FUZE[30]。该方案综合运用模糊测试、符号执行、动态追踪等技术，针对内核UAF漏洞自动生成漏洞利用样本。在实现过程中，首先结合KASAN[31]与动态追踪技术，获取生成利用程序所必须的信息，如释放对象的基址、大小等；其次，通过内核Fuzzing识别引用悬垂指针（dangling pointer）的系统调用；再次，结合符号执行对可利用性进行判断，并计算堆喷数据；最后，生成完整的漏洞利用程序。通过对15个内核UAF漏洞进行测试，FUZE成功生成漏洞利用并一定程度上绕过保护机制。

该方案的局限性主要体现在：首先，未考虑开启KASLR的情况；其次，在使用符号执行时，对前置约束条件的设置使得FUZE存在漏报的可能性。

3.10 Revery

2018年Yan Wang提出了针对堆漏洞的解决方案Revery[32]。该方案在实现过程中，首先分析漏洞位置和相关内存布局，建立异常对象内存布局图、内存布局贡献者图；其次，以异常对象的内存布局为导向，采用定向Fuzzing探索替代路径，并结合污点分析技术，在替代路径中探索可利用状态；最后，确定拼接点、拼接路径，组合生成漏洞利用路径。求解路径约束、漏洞利用数据约束等约束条件，生成漏洞利用。在测试过程中，Revery以CTF赛题为样本，成功生成多个针对堆漏洞的漏洞利用样本。

该方案的局限性主要体现在：首先，Revery基于angr开发，因此无法处理大部分真实应用；其次，无法自动进行堆的原子化操作，不能自动化构造内存布局。

4 结语

自2008年首次提出APEG后，自动化漏洞利用历经了十年的发展。所挖掘的软件形式从源码、字节码扩展至二进制代码，能够处理的漏洞类型也逐渐复杂。虽然相关研究取得了明显进展，但依然存在一些关键问题需要解决。如符号执行所面临的瓶颈问题，使得自动化漏洞利用仅适用于小型软件或代码片段；其次，无法处理基于堆的复杂漏洞、逻辑漏洞、结合多个漏洞的漏洞链等；再次，目前的研究主要集中于受限模式，所生成的漏洞利用样本无法绕过已有的安全机制。

从上述分析来看，自动化漏洞利用距离实用还存在一定距离，但现实中的迫切需求，已指明了今后的发展方向，并将进一步推动自动化漏洞利用的发展与完善。

参考文献：

[1]DAPRA. DARPA Cyber Grand Challenge. https://github. com/CyberGrandChallenge.

[2]DEFCON. https://www. defcon. org.

[3]刘剑，苏璞睿，等. 软件与网络安全研究综述[J]. 软件学报，2018，29(1):42-68.

[4]吴世忠. 软件漏洞分析技术[M]. 科学出版社，2014.

[5]Peach. http://peachfuzzer. com.

[6]Sulley. http://code. google. com/p/sulley.

[7]AFL. http://lcamtuf. coredump. cx/afl/.

[8]Boyer R S，Elspas B，Levitt K N. SELECT-A Formal System for Testing and Debugging Programs by Symbolic Execution[J]. ACM SIGPLAN Notices，1975，10(6):234-245.

[9]Cadar C，Ganesh V，Pawlowski P M，et al. EXE:Automatically Generating Inputs of Death[C]. Proceedings of the 13th ACM Conference on Computer and Communications Security，CCS 2006，Alexandria，VA，USA，Ioctober 30-November 3，2006. ACM，2006.

[10]Cadar C，Dunbar D，Engler D R. KLEE:Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs[C]. 8th USENIX Symposium on Operating Systems Design and Implementation，OSDI 2008，December 8-10，2008，San Diego，California，USA，Proceedings. USENIX Association，2008.

[11]Godefroid P,Klarlund,Nils,Sen,Koushik. DART:Directed Automated Random Testing[J]. ACM Sigplan Notices,2013,40(6):213-223.
[12]Sen K,Marinov D,Agha G. CUTE: a Concolic Unit Testing Engine for C[C]. ACM SIGSOFT Software Engineering Notes. ACM,2005,30(5):263-272.
[13]Shoshitaishvili Y,Kruegel C,Vigna G,et al. SOK:(State of) The Art of War:Offensive Techniques in Binary Analysis[C]. 2016 IEEE Symposium on Security and Privacy (SP). IEEE Computer Society,2016.
[14]Godefroid P,Levin M Y,Molnar D. SAGE:Whitebox Fuzzing for Security Testing[J]. Communications of the ACM,2012,55(3):40-44.
[15]Song D,Brumley D,Yin H,et al. BitBlaze:A New Approach to Computer Security via Binary Analysis[J],2008.
[16]Newsome J,Song D X. Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software[C]. NDSS. 2005,5:3-4.
[17]Clause J A,Li W,Orso A. Dytan:A Generic Dynamic Taint Analysis Framework[C]. Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis,ISSTA 2007,London,UK,July 9-12,2007. ACM,2007.
[18]Kemerlis V,Portokalidis G,Jee K,et al. libdft:Practical Dynamic Data Flow Tracking for Commodity Systems[C]. ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments. ACM,2012.
[19]和亮,苏璞睿. 软件漏洞自动利用研究进展[J]. 中国教育网络,2016,(Z1):46-48.
[20]Brumley D,Poosankam P,Song D,et al. Automatic Patch-Based Exploit Generation is Possible:Techniques and Implications[C]. 2008 IEEE Symposium on Security and Privacy(sp 2008). IEEE,2008.
[21]Heelan S. Automatic Generation of Control Flow Hijacking Exploits for Software Vulnerabilities[D]. University of Oxford,2009.
[22]Avgerinos T,Sang K C,Rebert A,et al. Automatic Exploit Generation[J]. Communications of the ACM,2014,57(2):74-84.
[23]Cha S K,Avgerinos T,Rebert A,et al. Unleashing Mayhem on Binary Code[J],2012.
[24]Brumley D,Jager I,Avgerinos T,et al. BAP:A Binary Analysis Platform[C]. International Conference on Computer Aided Verification. Springer,Berlin,Heidelberg,2011.
[25]Huang S K,Huang M H,Huang P Y,et al. CRAX:Software Crash Analysis for Automatic Exploit Generation by Modeling Attacks as Symbolic Continuations[C]. Software Security and Reliability(SERE),2012 IEEE Sixth International Conference on IEEE,2012.
[26]Wang M,Su P,Li Q,et al. Automatic Polymorphic Exploit Generation for Software Vulnerabilities[C]. International Conference on Security and Privacy in Communication Systems. Springer International Publishing,2013.
[27]Hu H,Chua Z L,Adrian S,et al. Automatic Generation of Data-Oriented Exploits[C]. 24th USENIX Security Symposium(USENIX Security 15). 2015:177-192.
[28]Brooks T N. Survey of Automated Vulnerability Detection and Exploit Generation Techniques in Cyber Reasoning Systems[J],2017.
[29]Stephens N,Grosen J,Salls C,et al. Driller: Augmenting Fuzzing Through Selective Symbolic Execution[C]. NDSS. 2016,16:1-16.
[30]Wu W,Chen Y,Xu J,et al. {FUZE}:Towards Facilitating Exploit Generation for Kernel Use-After-Free Vulnerabilities[C]. 27th USENIX Security Symposium(USENIX Security 18). 2018:781-797.
[31]KASAN. https://github. com/google/kasan/wiki.
[32]Wang Y,Zhang C,Xiang X,et al. Revery:From Proof-of-Concept to Exploitable[C]. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM,2018:1914-1927.

作者简介:
靳宪龙(1988-),男,山东章丘人,硕士,研究方向为二进制安全、漏洞挖掘
黄雅娟(1987-),女,甘肃天水人,硕士,研究方向为基础理论
收稿日期:2019-10-08 修稿日期:2019-10-30

Research Progress of Automated Exploitation

JIN Xian-long[1], HUANG Ya-juan[2]

(1. College of Computer Science, Sichuan University, Chengdu 610065;
2. School of Graduate, National University of Defense Technology, Nanjing 210012)

Abstract:
The situation of cyberspace security is becoming more and more complex, and the offensive and defensive game against software security is intensifying. The complexity and technical of software vulnerability mining and exploitation make a lot of work only rely on security experts to complete. In recent years, the number of vulnerabilities has soared, and it has been impossible to respond effectively by means of manual means. Automated exploitation have emerged. This method reduces labor costs while improving work efficiency and meets the urgent need for automated attack and defense. Introduces the concept of automated exploitation; generalizes and summarizes the key technologies; sorts out the mainstream automated exploit systems at home and abroad.

Keywords:
Network Security; Binary; Exploit; Automation

(上接第 5 页)

Research on Differential Evolution Parameter Optimization of Artificial Potential Field Method for Path Planning

ZHAO Wen-yu[1], PENG Cheng[1,2]

(1. School of Electronic & Information Engineering, North China Institute of Science and Technology, Sanhe 065201;
2. Key Laboratory for Safety Production of Coal Safety Monitoring and Control Technology, Ministry of Emergency Management, Sanhe 065201)

Abstract:
The parameter values of the traditional artificial potential field method are usually set by experience, if the parameters are set improperly, the planned path will be jagged and not smooth enough, the calculation efficiency is low, and the target point is unreachable. In response to the above problems, the shortest path to the planning is targeted, the three parameters of the traditional artificial potential field method are optimized by differential evolution algorithm: The gravitational field positive proportional gain coefficient, the repulsive field gain coefficient, and the influence distance of the obstacle. The obstacles are expanded into a circular shape, and the simulation environment is constructed using MATLAB software. The simulation results show that the artificial potential field method after parameter optimization can plan a smooth path and verify the effectiveness of the algorithm.

Keywords:
Path Planning; Artificial Potential Field Method; Differential Evolution Algorithm