1. 相关研究

控制流完整性（Control Flow Integrity，简称CFI）最早由Abadi等人提出[5]，该方法假设程序在其执行过程中，应当遵循预先定义好的控制流图（Control Flow Graph，简称CFG），以确保程序控制流不被劫持或非法篡改。具体作法是在控制流转移指令前插入检验代码，来判断目标地址的合法性。CFI技术分为细粒度和粗粒度两种类型。

Abadi等[5]提出的细粒度的CFI技术，预先计算出所有间接转移指令的可能的目标地址，并为之赋予唯一的ID；在检查时，在每条间接跳转指令执行前增加检查，一旦违反控制流图的间接跳转就会报错。Abadi等[6]利用CFI思想保证控制流的正确性，并结合内存访问控制、影子栈等，将传统的应用程序包装成XFI模块，在运行时提供保护与监控，并通过设置外部接口与操作系统交互。虽然这些细粒度的CFI技术能够提升系统的安全性，但其开销过大，难以得到实际部署。

粗粒度的CFI技术将一组或相近类型的目标归到一起进行检查，可在一定程度上降低开销。Zhang等[7]提出了CCFIR，该方法对间接调用指令和函数返回指令的目标进行区分，阻止未经验证的返回指令跳转到敏感函数的行为。Zhangn等[8]提出了binCFI，该方法将间接转移指令的操作数分为代码指针、异常处理程序入口、导出符号地址和返回地址等类型，通过精细的静态分析，得到不同类型的间接控制转移的合法目标集合，并基于此进行攻击检测。Mashtizadeh等[12]提出了CCFI方法对代码指针进行更细致的划分。该方法将代码指针细分为函数指针类、return指针类、方法指针类、虚表指针类，通过对代码指针进行加密来增强CFI方法。以上方法都是只实施了控制流不敏感策略，然而上下文信息的缺少会降低方法的有效性。因此Veen等[13]提出了上下文敏感的CFI技术，该方法能够监控通往敏感函数（可用于实施控制流转移攻击）的执行路径，通过使用二进制插桩涉及，在被监控路径上强制执行上下文敏感CIF的不变量。

以上的CFI技术均是基于二进制插桩的，需要修改源码或者通过反汇编重写二进制代码，这给这些技术的使用带来额外的工作量；并且会引入较大的性能开销或者检测效果不够精确。本文提供的PerfCFI方法并不属于该类别，不依赖于源代码，同时可以直接作用于传统程序，但本文提供的检测框架可以和基于二进制程序的方式结合实现基于CFG的CFI检查，该方法将作为我们未来的研究方向之一。

为了能够在性能和防御效果方面取得更好的效果，一些研究引入硬件机制来降低CFI技术的开销。

Pappas等提出了kBouncer方法[10]，该方法利用LBR（Last Branch Register）捕获最近的16次跳转信息，在敏感系统调用处对捕获的16次跳转进行安全性判断。Cheng等提出ROPecker方法[2]，该方法也是利用LBR捕获程序程序流的方式进行ROP攻击检测。该方法在运行时检测过去和未来的执行流中是否存在长gadget链来进行攻击检测，还通过滑动窗口的机制来进一步提高准确性和高效性。但是，这两种方法都是一次性针对16个间接分支进行检查，会面临历史覆盖不足的问题，导致检测精度下降。Xia等提出了CFIMon[11],该方法采用BTB（Branch Trace Buffer）来捕获程序运行过程中跳转指令的信息。虽然BTB能够程序整个执行过程中的所有跳转指令的历史信息都记录下来，但使用BTB的程序性能明显比LBR性能低。本文提供的方法属于该类别，但相比于其它基于硬件的ROP检测方法，如kbouncer，PerfCFI检查的粒度更小，同时新增了long nop-gadget检查以及系统调用检查，能够有效避免kBouncer被历史覆盖的问题。

除了CFI方法，另一个ROP攻击防御方面的研究热点是随机化技术[14-18]，该类技术通过将函数、内存页、基础块或指令排布等随机化，让攻击者不能准确预测所需gadget的位置，无法进行ROP攻击。我们的perfCFI可以和这些一起作用，用于阻止攻击者对应用程序进行ROP攻击。

1. 总结及展望

本文针对当今软件安全领域面临的主要安全威胁之一ROP攻击展开研究，提出了PerfCFI ROP攻击检测方法，有效解决了传统的CFI方法面临着两大弊端（之一）：1）引入较大的性能开销；2）需要修改源码或者通过反汇编重写二进制代码。实验表明，该方法能够有效的检测ROP攻击，并且仅会引入的较小的性能开销。

尽管本文仅提出了先验知识的CFI检测方法，但本文的检测框架可以和基于CFG的CFI技术相结合，有效降低对控制流图检测的频率，进而降低基于CFG的CFI技术的性能开销，这也是我们未来的研究方向之一。

**References**:

1. Davi L, Sadeghi A R, Winandy M. ROPdefender: A detection tool to defend against return-oriented programming attacks,Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 40-51, 2011.
2. Yueqiang Cheng, Zongwei Zhou, Miao Yu, Xuhua Ding, and Robert H Deng. ROPecker: A generic and practical approach for defending against rop attacks. NDSS14, 2014.
3. Ping Chen, Hai Xiao, Xiaobin Shen, Xinchun Yin, Bing Mao,and Li Xie. DROP: Detecting return-oriented programming malicious code. In Proceedings of the 5th International Conference on Information Systems Security (ICISS), 2009.
4. T. Bletsch, X. Jiang, V. W. Freely and Z. Liang, Jump-oriented programming:a new class of code-reuse attack, in Proceedings of the 6th ,4CM Symposium on Information, Computer and Communications Security,New York, NY, USA, pp. 30-40, 2011.
5. M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti. Control-flow integrity: Principles, implementations, and applications. In ACM Conference on Computer and Communications Security (CCS), 2005.
6. M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti.Control-flow integrity principles, implementations, and applications. ACM Transactions on Information and System Security (TISSEC), (1), Nov. 2009
7. Zhang, T. Wei, Z. Chen, L. Duan, L. Szekeres,S. McCamant, D. Song, and W. Zou. Practical Control Flow Integrity & Randomization for Binary Executables. In IEEE Symposium on Security and Privacy, 2013.
8. M. Zhang and R. Sekar. Control flow integrity for COTS binaries. In USENIX Security Symposium, 2013.
9. B. Niu and G. Tan. Modular control-flow integrity. In PLDI’14, 2014.
10. Vasilis Pappas, Michalis Polychronakis, and Angelos D. Keromytis. 2013. Transparent ROP exploit mitigation using indirect branch tracing. In Proceedings of the 22nd USENIX conference on Security (SEC'13). USENIX Association, Berkeley, CA, USA, 447462.
11. Yubin Xia, Yutao Liu, Haibo Chen, and Binyu Zang. 2012. CFIMon: Detecting violation of control flow integrity using performance counters. In Proceedings of the 2012 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (DSN '12). IEEE Computer Society, Washington, DC, USA, 112.
12. Ali Jose Mashtizadeh, Andrea Bittau, Dan Boneh, and David Mazières. 2015. CCFI: Cryptographically Enforced Control Flow Integrity. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15). ACM, New York, NY, USA, 941951.
13. Victor van der Veen, Dennis Andriesse, Enes Göktaş, Ben Gras, Lionel Sambuc, Asia Slowinska, Herbert Bos, and Cristiano Giuffrida. 2015. Practical ContextSensitive CFI. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15). ACM, New York, NY, USA, 927940.
14. L. Davi,A. Dmitrienko, S. Nűrnberger,A-R. Sadeghi. Gadge me if you can: Secure and efficient ad-hoc instruction-level randomization for x86 and arm. In 8th ACM SIGSAC symposium on Information, computer and communications security ,2013.
15. Wartell, R., Mohan, V., Hamlen, K.W., Lin, Z. Binary stirring: self-randomizing instruction addresses of legacy x86 binary code. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 157–168, 2012.
16. M. Backes, S. Nurnberger, Oxymoron: Making Fine-Grained Memory Randomization Practical by Allowing Code Sharing, in the Proceedings of the 23rd USENIX Security Symposium,2014
17. V. Pappas, M. Polychronakis, and A. D. Keromytis. Smashing the gadgets: Hindering return-oriented programming using in-place code randomization. In IEEE Symposium on Security and Privacy, 2012.
18. 詹珣. 针对ROP攻击的块粒度地址空间随机化防御技术的研究[硕士学位论文]. 南京. 南京大学软件学院 2014
19. Pinghai Yuan, Qingkai Zeng, Xuhua Ding. Hardware-Assisted Fine-Grained Code-Reuse Attack Detection. International Workshop on Recent Advances in Intrusion Detection. Pp.66-85
20. “HT Editor 2.0.20 Buffer Overflow（ROP PoC）,” http://www.exploit-db.com/exploits/22683/.
21. “JAD Buffer Overflow（ROP PoC）,” <http://www.exploit-db.com/exploits/42357/>.