

Control-Flow Integrity(控制流完整性) 的提出与发展历程

原创 SuckerForPain 最后发布于2018-12-24 18:18:17 阅读数 794 ☆ 收藏

1 引言

1.1 控制流劫持

计算机经常受到旨在控制软件行为的外部攻击。这种攻击作为数据传输并驻留在程序存储器中，就触发预先存在的软件缺陷。通过利用这些漏洞并获得对软件行为的控制。

在二进制安全中，大部分的漏洞利用方式是劫持控制流，接着使程序按照攻击者的攻击思路运行下去。控制流劫持是一种危害性极大的攻击方式，来获取目标机器的控制权，甚至进行提权操作，对目标机器进行全面控制。当攻击者掌握了被攻击程序的内存错误漏洞后，一般会考虑发起控制

早期的攻击通常采用代码注入的方式，通过上载一段代码，将控制转向这段代码执行。例如，应用程序中的缓冲区溢出可能导致对敏感系统函数程序从未打算使用的函数。

通过栈溢出控制程序中某个函数的返回地址，然后在该地址上布置 shellcode，将控制流劫持至 shellcode。为了阻止此类攻击，在硬件的支持下，现增加了 DEP(Data-Execution-Prevention, NX or W \oplus X) 机制，通过限制内存页不能同时具备执行权限和写权限，即隔离数据与代码来阻止程序运行对数据。

为了突破DEP的防御，攻击者又探索出了代码重用攻击方式，他们利用被攻击程序中的代码片段，进行拼接以形成攻击逻辑。代码重用攻击包括Return ROP(Return Oriented Programming)、JOP (Jump Oriented Programming) 等。研究表明，当被攻击程序的代码量达到一定规模后，一般能够从中找到图灵完备的代码片段，绕过 DEP 保护，最后达到攻击者的目的。

1.2 早期的防范措施

已提出的漏洞缓解措施包括堆栈金丝雀[Cowan et al. 1998]，运行时消除缓冲区溢出[Ruwase和Lam 2004]，随机化和人工异质性[PaX Project 2004; 2003]，以及可疑数据的污染[Suh et al.2004年]。

其中，有些减缓技术是不切实际的。例如，依赖于硬件修改或施加高性能损失。在任何情况下，它们的安全利益都有争议：缓解范围通常有限，攻击者规避每个部署的缓解机制[Pincus and Baker 2004; Shacham et al. 2004; Wilander and Kamkar 2003]。

这些机制的局限性部分源于缺乏现实的攻击模式以及对非形式化推理和隐藏假设的依赖。为了实现可靠性，缓解技术应该赋予潜在攻击者的聪明才智以现的软件漏洞的丰富性 - 易于理解和强制执行，同时为强大的攻击者提供强有力的防御。另一方面，为了在实践中可部署，缓解技术应该适用于现有代码，甚至适用于传统二进制代码) 并且产生毕竟低的开销。

2 CFI的提出与发展

2.1 CFI的提出

为了抵御控制流劫持攻击，Abadi等人于2005年提出了控制流完整性 (Control Flow Integrity, CFI) 的防御机制，是一种通过保证控制流的完整性来: 改的方法 (基于插桩技术)。

其核心思想是限制程序运行中的控制转移，使之始终处于原有的控制流图所限定的范围内。它规定软件执行必须遵循提前确定的ControlFlow图 (CFG CFG可以通过分析 (源代码分析，二进制分析或执行分析) 来定义。这篇文章关注通过静态二进制分析得到的CFG。CFG也可以通过显式安全策略来: 写为安全自动机[Schneider 2000]。

具体做法是通过分析程序的控制流图，获取间接转移指令 (包括间接跳转、间接调用、和函数返回指令) 目标的白名单，并在运行过程中，核对间接转移是否在白名单中。控制流劫持攻击往往会违背原有的控制流图，CFI使得这种攻击行为难以实现，从而保障软件系统的安全。

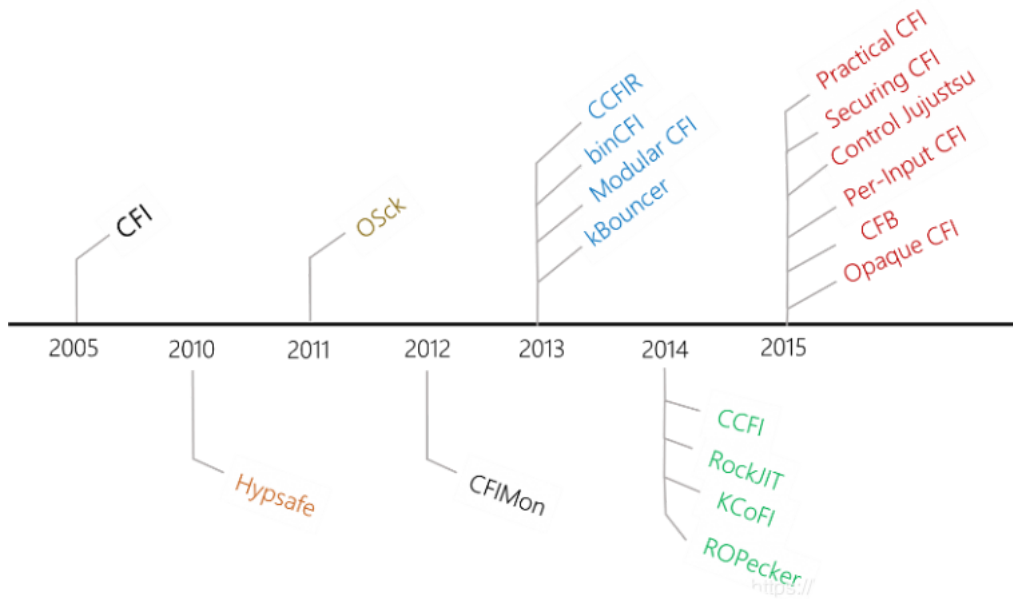
利用二进制重写技术向软件函数入口及调用返回处分别插入标识符ID和ID_check，通过对比ID和ID_check的值是否一致判断软件的函数执行过程是否: 而判断软件是否被篡改。

该方法虽能防范多种类型攻击，但实现复杂，且由于判断软件是否被篡改的全部逻辑都在该软件自身中存在，故该防护机制易于被攻击者发现和利用。

2.2 CFI的分类 (实现角度)

CFI从实现角度上，被分为细粒度和粗粒度两种。

细粒度CFI严格控制每一个间接转移指令的转移目标，这种精细的检查，在现有的系统环境中，通常会引入很大的开销。而粗粒度CFI则是将一组类似或标归到一起进行检查，以降低开销，但这种方法会降低CFI的保护效果。



2.3.1 基于插桩技术的CFI

2005年Martín Abadi和Mihai Budiu等人提出了控制流完整性 (Control-Flow Integrity, CFI) 的思路[1]。程序在其执行过程中, 应当遵循预先定义好 (Control Flow Graph, CFG), 以确保程序控制流不被劫持或非法篡改, 以防止背离程序编制时所设计的控制流转移关系。CFI在运行时检测程序的在控制流图中, 以识别是否遭遇了攻击。这种方法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。(参考第一节)

严格意义上的CFI, 需要做到对每条间接控制转移都做检查, 并确保每条转移指令只能转移到它自身的目标集合。如果要达到更好的防御效果, 则要做做的检查。然而, 这种检查机制虽然能够最大程度地提升系统的安全性, 但其开销过大, 难以得到实际部署。

Chao Zhang 等人指出, CFI未被广泛应用的原因是插桩引入的开销过大, 需要额外的信息 (比如间接控制转移可能的目标集合) 的支持, 且不能进行: 署。为此, 他们提出了CCFIR[2] (Compact Control Flow Integrity and Randomization)。其策略是对间接调用指令和函数返回指令的目标进行区验证的返回指令跳转到敏感函数的行为。这些限制囊括了CFI保护的主要思想, 且不需要别名分析。出于效率和兼容性的考虑, 通过一个专用的 “Sprir section” 来实现间接分支转移, 并限定代码对齐, 对跳转目标进行限制。经SPECint2000进行测试, 其平均/最高性能开销分别为3.6%和8.6% (平均/进一步提升安全性, “Springboard section” 在程序启动时随机地变换允许的跳转目标, 进一步增加了控制流劫持的难度。CCFIR是一个纯粹的二进制不依赖源码或调试信息, 所依赖的仅是重定位表中的信息。受保护的代码可以被独立地验证, 也可以进行增量式地部署。

Mingwei Zhang 等人提出了一种针对COTS (Commercial-off-the-shelf) 程序的CFI机制——binCFI[3], 目标是使CFI直接应用于已有的商用应用上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, 对不同类型的间接控制转移, 收集它们合, 如返回指令的合法目标集合就是所有函数调用指令的下一条指令的地址, 导出符号地址集合就是ELF文件中.dynamic段中存储的地址。利用这种方以得到与已有CFI方案相当的安全性。通过新增代码段的方式, 不改变原有的代码, 达到完全透明的目的。实验表明, 论文中提出的方案可以应用于大程序中, 拥有不错的安全性。

相对于严格意义上的CFI, CCFIR[2]和binCFI[3]都属于粗粒度CFI机制。所谓粗粒度CFI, 就是放宽检查的条件, 减少比较目标集合。原来的CFI是一个一个目标集合。放宽后, 间接转移目标进行合并, 比如把所有的间接调用指令和间接跳转指令的目标集合合成一个目标, 所有的函数返回指令的目标集这种方式能够有效地降低CFI引入的开销, 但也存在安全性问题。

Enes Göktas在Overcoming CFI[4]中, 利用两种特殊的gadget——entry point (EP) gadget和call site (CS) gadget, 来绕过粗粒度CFI机制的防御。gadget满足检查的条件, 但是其实是并不符合真实的控制流。Enes Göktas对这两种gadget的有效性进行了论述, 并利用其构造ROP链、调用库函数, 攻击。

为了对抗Overcoming CFI对粗粒度CFI的攻击, Ali Jose Mashtizadeh 等人提出了一种通过对代码指针加密, 来增强CFI的方法, 称为CCFI[5]。粗粒质就在于归类使得攻击者有了可乘之机。CCFI就是要对代码指针做更细致地划分, 还不能回归到CFI原本定义的那种细粒度的分类。他们将代码指针细别是函数指针类、return指针类 (函数返回地址)、方法指针类、虚表指针类 (后两种都是针对C++语言特有的类)。每次在保存一个代码指针时, 将息加密保存。每类指针对应一些具有标识作用的运行时信息。在这些指针解引用 (间接转移) 时, 解密信息并比对。另外为了加快加解密的速度, CCF器中独特的指令 “AES-IN” 对AES加解密算法加速, 同时为了保证密钥的安全, 又将密钥保存在处理器中独特的寄存器中。他们修改了开源编译个编译器, 并用其编译了一些程序, 在实际的攻击环境中检测其有效性。实验证明, 他们的方法在安全性和性能方面取得了很好的折衷。

LucasDavi等人在Stitching the Gadgets[6]中详细分析了不同的CFI解决方案包括kBouncer、ROPecker、binCFI、ROPGuard和Microsoft EN的安全性不容乐观。他们首先重新思考了不同的粗粒度的CFI的假设, 对kBouncer、ROPecker、binCFI、ROPGuard和Microsoft EMET4.1进行了详细析, 特别地, 整合了这些现有粗粒度CFI, 形成了一个安全策略更为严格的CFI系统。在这个更为严格的条件下, 完成了图灵完整性的验证。为了更好地

会受到攻击的。他们提出了三种攻击攻击方法：一是利用堆上的漏洞来破坏栈上的callee-saved寄存器保存区域，使得callee-saved寄存器被劫：间和内核之间进行上下文切换的问题，来劫持sysenter指令，使控制跳转到攻击者想跳转的位置；三是通过泄露主栈的地址来泄露出来shadow进行攻击。

以往CFI方案的问题是只实施了控制流不敏感策略。上下文信息的缺少不可避免地降低了CFI的防御能力，从而给了攻击者利用空间。上下文敏感(sensitive CFI)是有望解决这一问题的方法，它依赖于上下文敏感的静态分析，将CFI不变量和CFG中的控制流路径联系到一起，运行时在执行时不变量。CCFI早在CFI提出之时已被认可，但因其在实际应用中不实际而被迅速废弃。Victor van der Veen 等人提出的Practical Context-Sensitive CFI是一个可用于现实应用程序的高效、可靠、实用的上下文敏感CFI方案：PathArmor。PathArmor依赖于商用硬件的支持，高效可靠地监控通往敏感控制流转移攻击)的执行路径；使用仔细优化的二进制插桩设计，在被监控路径上强制执行上下文敏感CFI的不变量。PathArmor的路径不变量是通过一定范围内的上下文敏感静态分析得到的，而且路径验证步骤使用了caching来提高验证效率。路径验证本身也是非常高效的，因为所有的CFI点处集中完成。

John Criswell将CFI做到操作系统内核中，使之免受经典的控制流劫持、return2user和代码段修改攻击，该系统称为KCoFI[9]。他们在基于标保护的基础上，加入一个运行时监控的软件层，负责保护一些关键的操作系统数据结构和监控操作系统进行的所有底层状态操作。并且还通过形个小子集的正确性。证明涵盖了页表管理、异常处理、上下文切换和信号分派等操作。实验表明，KCoFI阻止了攻击者将控制流向FreeBSD Kernel转移。与未修改的内核相比，KCoFI在运行server测试集时具有较小的开销，但运行文件系统操作密集型的测试集时，开销较大。

2.3.2 利用硬件来提升CFI的效率

为了能够在性能和防御方面取得更好的效果，一些研究着手于利用现有的硬件机制，来降低CFI的开销。

Vasilis Pappas提出利用硬件性能计数器，在运行时观察执行流的思路，该方法被称为kBouncer[10]。他们利用LBR (Last Branch Register) 来捕获跳转信息。具体做法是在敏感系统调用处，对捕获的16次跳转进行安全性判断，即return指令需要跳转到调用点的后继位置，indirect-call指令的目标是余跳转指令目标基本块长度不能全部少于20条指令。为了避免攻击者利用库函数调用来完成攻击，文章在所有的库函数调用点，来进行上述合法性检查。kBouncer的防御效果，作者对IE浏览器、Adobe Flash Player和Adobe Reader进行了实验(利用已知安全漏洞，组织ROPpayload攻击这三种应用)表明该方法能够有效缓解ROP攻击。同时，该方法的性能开销低于~4%。

Yueqiang Chen等人设计了一种与kBouncer类似的方法，称作ROPecker[11]，也是利用LBR捕获程序控制流的方式进行ROP攻击监测。但不同之处在于受ROP攻击的逻辑和触发监测的时机。1、判断逻辑：在运行时检测过去(利用LBR)和未来的执行流(模拟执行)中是否存在长gadget链(5个比较gadget)，若存在，则认为这是一次ROP攻击。Gadget信息是通过静态分析二进制程序和共享库得到的。2、运行时监测是事件驱动的，具体时机是调用和执行流跳出滑动窗口触发异常。ROPecker设计了一个滑动窗口，因为代码本身具有时间和空间的局部性，但是gadget链却是散列的，利用这一保证该窗口内的gadget数目不足以构成一次ROP攻击，窗口内的代码设置可执行权限，窗口外的代码不可执行，当执行流跳出滑动窗口时，便会触发运行时检测。该方法利用代码本身具有的时间和空间局部性，针对gadget链是散列的前提，提出了滑动窗口机制，使用事件驱动的检测方法，具有较高效率。为了验证该方法的安全性，ROPecker选取了有栈溢出的真实世界应用(Linux Hex-editor)进行攻防演练。实验结果证明，ROPecker能够有效攻击。同时，SPEC CPU2006benchmark显示了该方法的开销非常低(~2%)。

Yubin Xia等人设计的CFIMon[12]，也是采用性能计数器来捕获程序执行流，并进行合法性判断。但他们采用的是BTB (Branch Trace Buffer)，来捕获运行过程中所有跳转指令的信息。BTB与LBR不同之处在于，BTB可以把程序整个执行过程中所有的跳转指令的历史信息都记录下来，LBR只能记录16条需要CPU向指定的一个缓冲区内写入跳转信息，当缓冲区满时，CPU会触发异常交给操作系统处理(将缓冲区内容写入文件中)，LBR是循环的寄存器程序性能明显比LBR性能低。CFIMon检查BTB的时机在两个阶段：一是当缓冲区满时，操作系统将所有历史信息写入另一个进程，由另一个进程进行检查；二是当受保护进程执行敏感系统调用时，另一个进程也进行历史信息的合法性判断。合法性判断主要检查间接控制转移的跳转目标是合法目标集合内。控制转移的历史跳转目标在合法目标集合中，认为当前受保护进程没有收到攻击；如果有至少一个间接控制转移的历史跳转目标在合法目标集合中，则进程受到攻击。合法目标的集合是在线通过静态分析获得的，并且存储在检查进程中。

3 总结与展望

大多数的攻击都依赖于某种形式的控制劫持来重定向程序执行，CFI (Control Flow Integrity) 是一种运行时强制执行的技术，用以抵御代码注入、代也不易受信息泄露攻击。CFI在运行时强制执行预期的控制流转移，不允许执行应用程序控制流图(Control Flow Graph, CFG)中未出现的转移。精入大量开销，这一点激励了更为实际的、粗粒度的CFI变体的发展；粗粒度CFI通过放宽限制条件来降低开销，却也给攻击者提供足够的利用空间。另外也仅仅是用于阻止攻击者对控制流的劫持，有研究表明(如Control-Flow Bending, CFB[13])，攻击者可以仅仅通过控制函数调用的参数，完成攻击如执行任意代码、执行受限制的代码、和信息泄露。

从2005年CFI技术被提出，CFI技术已经历了10多年的发展。近两年涌现出的大量CFI相关的研究工作，极大推进了CFI技术的发展。但是，CFI技术未发展空间，仍然需要研究人员继续深入研究，实现一种高可靠、低开销的CFI技术。

参考文献

- [1]. Martín Abadi, Mihai Budiu, Úlfar Erlingsson, and Jay Ligatti. 2005. Control-flow integrity. In Proceedings of the 12th ACM conference on computer and communications security (CCS '05). ACM, New York, NY, USA, 340-353.
- [2]. Chao Zhang, Tao Wei, Zhaofeng Chen, Lei Duan, Laszlo Szekeres, Stephen McCamant, Dawn Song, and Wei Zou. 2013. Practical Control Flow Integrity and Randomization for Binary Executables. In Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP '13). IEEE Computer Society, Washington, DC, USA, 559-573. DOI=http://dx.doi.org/10.1109/SP.2013.44

[4]. Enes Göktaş,Elias Athanasopoulos, Herbert Bos, and Georgios Portokalidis. 2014. Out of Control: Overcoming Control-Flow Integrity. In Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP '14). IEEE Computer Society, Washington, DC, USA, 575-589.

[5]. 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, 952-963.

[6]. Lucas David, Ahmad-Reza Sadeghi, Daniel Lehmann, and Fabian Monrose. 2014. Stitching the gadgets: on the ineffectiveness of control-flow integrity protection. In Proceedings of the 23rd USENIX conference on Security Symposium (SEC '14). USENIX Association, New York, USA, 401-416.

[7]. Mauro Conti, Stephen Crane, Lucas David, Michael Franz, Per Larsen, Marco Negro, Christopher Liebchen, Mohaned Qunaibit, and Ahmad-Reza Sadeghi. 2015. Losing Control: On the Effectiveness of Control-Flow Integrity under Stack Attacks. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15). ACM, New York, NY, USA, 952-963.

[8]. Victor van der Veen, Dennis Andriesse, Enes Göktaş, Ben Gras, Lionel Sambuc, Asia Slowinska, Herbert Bos, and Cristiano Giuffrida. 2015. Context-Sensitive CFI. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15). ACM, New York, NY, USA, 927-940.

[9]. John Criswell, Nathan Dautenhahn, and Vikram Adve. 2014. KCoFI: Complete Control-Flow Integrity for Commodity Operating Systems. In Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP '14). IEEE Computer Society, Washington, DC, USA, 292-307.

[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, 447-462.


[11]. Yueqiang Cheng, Zongwei Zhou, Miao Yu, Xuhua Ding and Robert H. Deng. 2014. ROPecker: A Generic and Practical Approach for Defense Against ROP Attacks. In Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS '14).

[12]. 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, 1-12.

[13]. Carlini N, Barresi A, Payer M, Wagner D and Gross TR. 2015. Control-Flow Bending: On the Effectiveness of Control-Flow Integrity. In Proceedings of the 24th USENIX Security Symposium (USENIX Security 15). USENIX Association, Washington, D.C., USA

[14]. 武成岗, 李建军. 2016. 控制流完整性的发展历程.


点赞 收藏 分享 ...



SuckerForPain

发布了19 篇原创文章 · 获赞 8 · 访问量 9432

私信



世界网游的排名是什么

排名前十名网游

评论点赞，都要勇往直前....

Control-Flow Integrity(控制流完整性) 的原理 阅读数 2064

本文讨论的原理基于Control-Flow Integrity Principles, Implementations, and App... 博文 来自: Sucker F...

浅析A-CFI技术：如何利用硬件防御ROP 阅读数 103

0x00 前言随着漏洞缓解技术的不断发展，常用的一些漏洞利用手段如ROP变得越来越... 博文 来自: weixin_3...

控制流完整性简介 阅读数 778

控制流完整性简介0x00. 基础知识0x01. 控制流完整性发展历程0x02. CFI 机制的比较... 博文 来自: lsy67380...

FlowControl: PauseFrame 阅读数 1905

FlowControl: PauseFrame当端口的Rx 部分接收大流量且Rx FIFO 队列已满时，端... 博文 来自: laoding...



消防安全知识100题，不知道你就亏大了

消防考试考哪些

举报

[译] Android 内核控制流完整性 阅读数 46

原文地址: Control Flow Integrity in the Android kernel原作者: Android Devel... 博文 来自: weixin_3...

CSDN

首页 博客 学院 下载 论坛 问答 活动 专题 招聘 APP VIP会员 搜博主文章

Q

创作

目录一、Android内核漏洞概览访问控制seccomp sandboxing不需要权限在userlan... 博文 来自: think_ycx...

CFI (Common Flash Interface) 详解

【什么是CFI】CFI (Common Flash Interface) , 是JEDEC (Joint Electron Device ... 博文 来自: EMMA3S...

阅读数 4040

学Python后到底能干什么? 网友: 我太难了

感觉全世界营销文都在推Python, 但是找不到工作的话, 又有哪个机构会站出来给我... 博文 来自: CSDN学院

阅读数 1万+

大学四年自学走来, 这些私藏的实用工具/学习网站我贡献出来了

大学四年, 看课本是不可能一直看课本的了, 对于学习, 特别是自学, 善于搜索网上... 博文 来自: 帅地

阅读数 63万+

sd敢达无限v1.1.1
sd敢达ol重新开服

在中国程序员是青春饭吗?

今年, 我也32了, 为了不给大家误导, 咨询了猎头、圈内好友, 以及年过35岁的几位... 博文 来自: 启舰

阅读数 23万+

超全Python图像处理讲解 (多图预警)

文章目录Pillow模块讲解一、Image模块1.1 、打开图片和显示图片1.2、创建一个简... 博文 来自: ZackSock...

阅读数 1万+

weixin_34129696
4577篇文章
关注 排名:千里之外

Smilence_Isy
10篇文章
关注 排名:千里之外

_UPDATA
13篇文章
关注 排名:千里之外

为什么猝死的都是程序员, 基本上不见产品经理猝死呢?

相信大家时不时听到程序员猝死的消息, 但是基本上听不到产品经理猝死的消息, 这... 博文 来自: 曹银飞的...

阅读数 14万+

毕业5年, 我问遍了身边的大佬, 总结了他们的学习方法

我问了身边10个大佬, 总结了他们的学习方法, 原来成功都是有迹可循的。 博文 来自: 敖丙

阅读数 16万+

推荐10个堪称神器的学习网站

每天都会收到很多读者的私信, 问我: “二哥, 有什么推荐的学习网站吗? 最近很浮... 博文 来自: 沉默王二

阅读数 26万+

企业邮箱免费试用, 购买还送专属公司域名!
企业邮箱怎样申请

强烈推荐10本程序员必读的书

很遗憾, 这个春节注定是刻骨铭心的, 新型冠状病毒让每个人的神经都是紧绷的。那... 博文 来自: 沉默王二

阅读数 8万+

为什么说程序员做外包没前途?

之前做过不到3个月的外包, 2020的第一天就被释放了, 2019年还剩1天, 我从外包... 博文 来自: dotNet全...

阅读数 10万+

柏林艺术家用行为艺术骇了一次谷歌地图

用一辆装了99部智能手机的手拉车, 一位柏林艺术家在一条空荡荡的街道上, 骇了一... 博文 来自: 德国IT那...

阅读数 2637

B 站上有哪些很好的学习资源?

哇说起B站, 在小九眼里就是宝藏般的存在, 放年假宅在家时一天刷6、7个小时不在话... 博文 来自: 九章算法...

阅读数 13万+

给新手程序员的一点学习建议


我是一个有几年经验的程序员, 之前对于自己的发展却是一头雾水, 不知道主流技术... 博文 来自: JAVA圈的...

阅读数 4181

消防安全知识100题, 不知道你就亏大了
消防考试考哪些

举报

	首页	博客	学院	下载	论坛	问答	活动	专题	招聘	APP	VIP会员	搜博主文章		创作
我有个学弟，在一家小型互联网公司做Java后端开发，最近他们公司新来了一个技术... 博文 来自: HollisChu...														
字节跳动的技术架构 阅读数 3万+														
字节跳动创立于2012年3月，到目前仅4年时间。从十几个工程师开始研发，到上百人... 博文 来自: 作一个独...														
在三线城市工作爽吗？ 阅读数 8万+														
我是一名程序员，从正值青春年华的 24 岁回到三线城市洛阳工作，至今已经 6 年有... 博文 来自: 沉默王二														
这些插件太强了，Chrome 必装！尤其程序员！ 阅读数 2万+														
推荐 10 款我自己珍藏的 Chrome 浏览器插件 博文 来自: 沉默王二														
抱歉，我觉得程序员副业赚钱并不靠谱 阅读数 1万+														
我最近看到不少关于程序员副业赚钱的文章，其中出的点子有这些：1. 在网上找项目... 博文 来自: 码农翻身														
@程序员：GitHub这个项目快薅羊毛 阅读数 1万+														
今天下午在朋友圈看到很多人都在发github的羊毛，一时没明白是怎么回事。后来上... 博文 来自: dotNet全...														
删库了，我们一定要跑路吗？ 阅读数 1万+														
在工作中，我们误删数据或者数据库，我们一定需要跑路吗？我看未必，程序员一定... 博文 来自: 平头哥的...														
“金三银四”，敢不敢“试”？ 阅读数 6849														
临近3月份，到了“金三银四”换工作的高峰期，往年可能会3、4月份，今年特殊，多... 博文 来自: 铭毅天下...														
又一程序员删库跑路了 阅读数 2万+														
loonggg读完需要2分钟速读仅需 1 分钟今天刷爆朋友圈和微博的一个 IT 新闻，估计... 博文 来自: 非著名程...														
再不跳槽，应届生拿的都比我多了！ 阅读数 1万+														
跳槽几乎是每个人职业生涯的一部分，很多HR说“三年两跳”已经是一个跳槽频繁与... 博文 来自: 九章算法...														
我以为我学懂了数据结构，直到看了这个导图才发现，我错了 阅读数 1万+														
数据结构与算法思维导图 博文 来自: 龙跃十二														
String s = new String(" a ") 到底产生几个对象？ 阅读数 1万+														
老生常谈的一个梗，到2020了还在争论，你们一天天的，哎哎哎，我不是针对你一个... 博文 来自: 宜春														
技术大佬：我去，你写的 switch 语句也太老土了吧 阅读数 3万+														
昨天早上通过远程的方式 review 了两名新来同事的代码，大部分代码都写得很漂亮，... 博文 来自: 沉默王二														
当年，非典SARS真的是我们战胜的吗？ 阅读数 5026														
这里是小汤山医院。医院早拆了，只剩一片芦苇荒地，和四周悄然兴建的温泉别墅。... 博文 来自: 纯洁的微笑														
学历低，无法胜任工作，大佬告诉你应该怎么做 阅读数 1万+														
微信上收到一位读者小涛的留言，大致的意思是自己只有高中学历，经过培训后找到... 博文 来自: 沉默王二														
和黑客斗争的 6 天！ 阅读数 3万+														
互联网公司工作，很难避免不和黑客们打交道，我呆过的两家互联网公司，几乎每月... 博文 来自: 纯洁的微笑														
讲一个程序员如何副业月赚三万的真实故事 阅读数 7978														
loonggg读完需要3分钟速读仅需 1 分钟大家好，我是你们的校长。我之前讲过，这年... 博文 来自: 非著名程...														
一大波硕士即将来袭 阅读数 6964														
前几天有一个读者朋友，也是程序员，在微信和我说：研究生扩招了，他要不要把专... 博文 来自: siyuanwa...														
别再自己抠图了，Python用5行代码实现批量抠图 阅读数 3万+														
前言对于会PhotoShop的人来说，弄一张证件照还是非常简单的，但是还是有许多人... 博文 来自: ZackSock...														
没有项目经验怎么办？ 阅读数 8502														举报
职场和学校最大的不同就是：你在学校，老师给一本书，然后你考试。如果没有通过... 博文 来自: 微信公众...														



SuckerForPain

TA的个人主页 >

原创19

粉丝8

获赞8

评论1


访问9432

等级: 博客 2

周排名: 28万+

积分: 281

总排名: 24万+

勋章: 

关注

私信

最新文章

模式识别与机器学习 | 概率图模型 | HMM


模式识别与机器学习 | 概率图模型 | 无向图


模式识别与机器学习 | 概率图模型 | 有向图模型


计算机算法设计与分析 | 网络流


模式识别与机器学习 | 机器学习简介


分类专栏

git1篇

software1篇

mongo

笔记13篇

机器学习14篇

展开

归档

2019

1月8篇

2018

12月

11月

8月

创作





















举报

