如何双取情况变成双取漏洞：双取指令的研究

在Linux内核

王鹏飞， *国防科学技术大学;* 延Krinke， *大学学院*
伦敦; 凯露和李根， *国防科学技术大学;*
史蒂夫Dodier - 拉扎罗， *伦敦大学学院*

https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-pengfei

本文包含的论文集
第26届USENIX安全研讨会

8月16日至18日，2017年温哥华•加拿大不列颠哥伦比亚省

ISBN 978-1-931971-40-9

第26届USENIX安全研讨会论文集开放存取

由USENIX主办

如何双取情况变成双取漏洞：
双取操作数在Linux内核的研究

王鹏飞
国防科学技术大学

延Krinke
伦敦大学学院

卢启
国防科学技术大学

李根
国防科学技术大学

史蒂夫Dodier - 拉扎罗
伦敦大学学院

抽象

我们提出的第一个系统地检测在Linux内核中潜在的双重取漏洞静态的方法。使用基于模式的分析，我们identi网络连接的Linux内核编90分取。这些57发生在驱动程序，它以前的动态方法均非能够不访问对应硬件来检测。我们手动调查的90个事件，并推测其中双取出现三种典型的方案。我们讨论了他们每个人的详细介绍。我们进一步去发展咸静态分析的基础上，Coccinelle的匹配 - 荷兰国际集团的发动机，其检测可能会导致内核漏洞双取的情况。当应用到Linux，FreeBSD和Android的内核，我们的方法发现了6个以前未知的双取的错误，他们四人在驱动程序，这三个是可利用的双取vul- nerabilities。所有identi网络版bug和漏洞已经Rmed指和维护修补CON网络连接。我们的方法已经通过了Coccinelle的团队，目前正在整合到Linux内核补丁审批。根据我们的研究，我们还提供了双预测取的错误和vulnerabil-伊蒂埃斯实用所谓lutions。我们还提供自动补丁解决检测到重取的错误。

1引言

广泛使用的多核硬件正在concur-租金节目越来越普遍，尤其是在oper-阿婷系统，实时系统和计算密集型的系统。然而，并发程序也是臭名昭著的**二 FFI 邪教检测并发错误。现实世界的并发错误可分为三种类型：原子性违**章的错误，为了违章错误，和死锁[20]。

数据竞争是并行程序的另一个常见的情况。当两个线程都访问一个共享存储器位置，这两个接入中的至少一个它发生为写入，并且两个访问的相对顺序不被任何同步原语[30，15]执行。数据争通常导致并发错误，因为它们可能会导致原子-侵犯[22，21，23]或命令 - 违反[33，40]。除了两个线程之间发生，数据也种族可以在内核和用户空间发生。塞尔纳[32]是第一个使用的术语“dou- BLE取”来描述由于在内核取相同的用户空间中的数据的两倍的竞争条件一个Windows内核漏洞。双取时，内核读取并使用相同的值，在用户空间的两倍重边（希望它是identi- CAL两次）出现错误，而同时运行的用户线程可以在时间窗口修改值之**间的两个内核中读取。 FF ER在流人[1，32，14，37]。**

Jurczyk和冷风[14]是在第一个系统地研究dou- BLE取。他们动态的方法去tected通过追踪内存访问，他们发现了一系列的Windows内核的双获取漏洞的双取。然而，他们的动态方法只能达到有限的覆盖范围。特别是，它不能被应用到代码对应硬件需要被执行，因此，设备驱动程序不能没有访问该设备或它的模拟来分析。因此，他们的分析不能覆盖内核的全部。事实上，他们的做法并没有发现在Linux中，FreeBSD的OpenBSD的或[13]中任双取漏洞。是 - 边，Jurczyk和冷风带来的关注，不仅对如何连接**的第二也是对 *如何利用* 双取漏洞。关于如何利用dou- BLE取指令最近已成**为可公开获得的[11]。因此，审计仁，尤其驱动程序，为获取双安全漏洞已经成为当务之急。

设备驱动程序是关键的内核级程序，桥梁的硬件和提供的接口是 - 吐温的操作系统软件和设备连接到系统。驱动程序是最新operat-很大一部分

荷兰国际集团系统，例如，在Linux 4.5源的网络连接LES是 - 44％的长到驱动程序。司机被发现是特别容易出错的内核组件。Chou等。[7] empiri-卡利表明，错误率在设备驱动程序是约十倍比内核的任何其它部分高。雨燕等。[34]还发现，在Windows XP系统崩溃的85％，可以在驱动程序错误受到指责。Further-更多，Ryzhyk等。[29]发现，在驱动器的错误的19％是并发错误，并且其中大多数是数据争用或死锁。

由于驱动程序也是在内核失败的这样一个临界点，就必须分析安全vulnerabili-关系，即使其相应的硬件不利用 - 能。事实上，Linux内核源网络莱的26％属于哪个不能Jurczyk和冷风的基于x86的tech- NIQUE分析x86之**外的硬件架构。因此，动态分析不是一个可行的，一 FF ordable方法。因**此，我们开发了一个基于静态模式的方法，以确定在Linux内核的双读取，包括司机在内的完整空间。我们identi网络编90分取，我们再研究和categ o- rized到其中双取出现三种典型的方案。我们发现，大多数双取不取双因**错误虽然内核 *取* 相同的数据两次，它只 *使用* 从两次存取中的一个的数据**。因此，我们重新定义网络的静态基于模式的AP-proach检测出真正的双取错误和vulnera- bilities，并进行分析的Linux，Android和FreeBSD的吧。

我们发现，大多数在Linux下4.5双取发生在驱动器（57/90）等完成大部分identi网络的编双取臭虫（4/5）。这意味着动态分析方法不能检测出大部分的双取的错误，除非研究人员有机会获得硬件与他们分析内核兼容的完整范围。这是CON音响Rmed指通过Bochspwn，动态分析方法的比较，这是无法连接的第二任双Linux中3.5.0 [13]其中，我们的做法网络连接NDS 3读取错误。综上所述，我们做本文的以下贡献：

（1）在Linux内核中双取的第一个系统研究。 我们提出的第一个（据我们所知）在整个Linux内核的双读取，包括如何和为什么双取发生的分析研究。我们使用模式匹配自动 - matically识别Linux内核90双取的情况，并通过人工审核内核源调查的候选人。我们分类的identi-网络编双取到三种典**型的方案（ *类型选择，大小检查，浅拷贝* ）在其中双取易于发生，并且示**出了具有双详述取情况下分析各场景。大多数（57/90）的identi网络的编发生在司机双取。

（2）一种基于模式的双取错误检测AP- proach。 我们开发了一个基于静态模式的方法，来检测双抓取错误[1]。该方法已被宜施mented上Coccinelle的程序匹配和反式形成发动机[17]和已被改编为入住荷兰国际集团在Linux，FreeBSD和Android的内核。这是第一种方法能检测完整的内核，包括所有驱动程序和所有硬件架构双取漏洞的网络连接。我们的方法已经通过了Coccinelle的团队，目前正在集成到通过Coccinelle的Linux内核补丁程序审批。

六双取的错误（3）Identi科幻阳离子。 我们总共发现了6个实双取错误。四是在linux 4.5的驱动程序和他们三个都是利用的vulner-能力。此外，所**有四个驱动器相关的双取的错误属于同 *大小检查* 场景。该漏洞已经被Lin**ux维护者Rmed指CON连接，并在新的版本中，作为我们报告的结果已固定成本。一张大取漏洞在Android被发现

6.0.1内核，这是已经连接在新的Linux ker- NELS固定的。

（4）策略双取错误预防。 根据我们的研究，我们提出了网络解决方案已经预见到双取的错误，我们实施了strate-吉斯之一的工具，自动补丁双取错误。

凯莉在Linux驱动程序2个呈现在Linux的内存访问的相关背景，SPECI网络仲重刑，并在发生错误如何获取双：本文的其余部分安排如下。第3所介绍我们的方法来获取一倍的检测，包括我们的分析过程中，identi Fi的分类编双取成三种情况，以及我们从identi-音响学会编双取错误。第4节我们工作的评价，包括对人工分析数据和应用我们到Linux，FreeBSD和Android的内核方法的结果。第5所讨论检测到的错误，我们的方法的双取的错误预防准则，我们的网络ndings的解释，以及limi- tations的影响。相关工作是在第6节讨论，其次是结论。

2背景

我们提供的数据如何在Linux内核和它的驱动程序和用户空间之间EX-变化提醒读者，以及如何竞争条件和双取的bug可以在这个框架内进行。

[1] 我们的分析，请 https://github.com/UCL-CREST/ doublefetch

2.1内核/用户空间保护

在现代计算机系统中，内存分为内核空间和用户空间。内核空间是内核代码执行，并且其中其内部的数据被存储，而用户空间是正常的用户亲正如事实运行。每个用户空间程序驻留在它自己的地址空间，该空间内只能寻址内存。这些虚拟地址空间是通过在这样一种方式，单独的空间之间的异相关特征研保证内核映射到物理存储器。内核也有自己独立的地址空间。

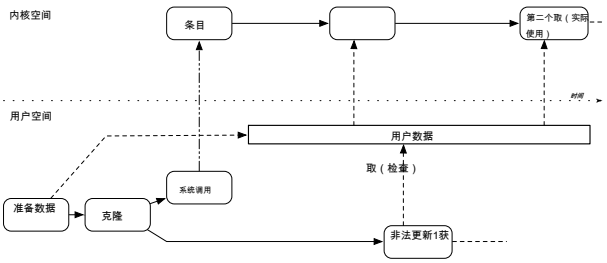

图1：主要双人取竞争条件

特别方案是由操作SYS-TEM提供给内核和用户空间之间的数据交换。在Windows中，我们可以使用该设备的输入和输出控制（IOCTL）方法，**或一个共享内存对象方法内核和用户空间之间交换数据** [2]

这是非常相似的共享存储器区域。 在Linux和FreeBSD，功能提供给内核空间和用户空间，我们称之为之间安全**地传送数据** ***传输等功能。*** **例如，Linux的有四个经常使用的传递函数， 调用copy_from_user（ ），copy_to_user（ ）， GET_USER（ ） 和 put_user（ ）** 该复制单个值或数据并从用户空间以安全的方式的任意量。传输功能不仅交换内核和用户空间，还提供针对无效的内存访问，如非法地址或页面故障的保护机制之间的数据。因此，任何在Linux的双取将涉及的传递函数多PLE调用。

2.2内存访问的驱动器

设备驱动程序负责烯内核组件abling的内核并使用连接到系统的硬件设备进行通信。驱动器具有典型特征，例如用于同步和异步操作的支持并且能够被多次打开[8]。驱动程序是对安全是 - 在他们的事业关键的故障可导致在给予整个系统的控制漏洞。最后，司机经常不得不从用户空间复制变量类型或可变长度的消息到硬件，并且，我们将在后面看到，这往往导致双取的情况是导致漏洞。

在Linux中，所有设备都有一个文件表示可以从用户空间进行访问互动**与硬件的驱动程序。内核在/创建一个文件 开发 二教区长为每个驱动器，**通过该用户空间进程可以使用网络连接文件输入/输出系统调用交互。该驱**动程序提供所有网络连接文件相关的OP-操作的实现，包括 读（ ） 和 写（ ） 功能。在这样的功能，驾驶员需要从获取数据**

[2] https://support.microsoft.com/en-us/kb/191840

用户空间（在 写 ） 或复制数据到用户空间（在 读 ）。 驱动程序使用的传递函数这样做，并再次，任何双重取将涉及的传递函数多个invoca-蒸发散。

2.3双取

双取是发生在内核和用户空间之间的内存访问的竞争条件的特殊情况。这种类型的第一个漏洞是预由塞尔纳[32]在Windows上的报告双取漏洞sented。从技术上讲，双取指需要一个内核函数内的地方，如一个系统调用，其通过从用户模式的用户应用调用。如IL-图1 lustrated，内核函数从在用户空间中的同一存储器位置取两次的值，则第一个时间来检查和验证，并使用它（注意的是，事件是从时间轴的第二时间左到右，但用户数据是相同的对象中的所有时间）。同时，两个内核之间取的时间窗口内，一个同**时运行的用户线程作案网络上课的价值。然后， FF erent值，这不仅会造成在二 FF erent计算结果，但可能会导致一个BU FF 呃溢流，一个空指针崩溃**或更糟的后果。

为了避免混淆，我们使用术语 ***取翻番*** **要么** ***仔细取情况*** **在本文中以表示所有在其中内核读取超过一次相同的用户数据**的详细情况的，和所谓的双取可以是fur-疗法分为以下情况：

良性双取： 良性双取的是，不会造成伤害，由于附加的保护方案或因为双牵强值不用于两次（细节将在第5.3节中讨论）的情况。

有害双取： 有害双取还是 ***仔细取的bug*** **是双取，可能实际上会导致在**内核中SPECI网络Ç情况下，例如，可以由用户进程触发的竞争条件的故障。

双取漏洞： 双取的错误也可以变成一个 ***仔细取漏洞*** **一旦造成竞争状态**的形成机制，quence被利用，例如

```
140 INT cmsghdr_from_user_compat_to_kern ( 结构 指向msghdr * kmsg，141
                        无符号字符 * stackbuf， INT stackbuf_size )
142 {143
        结构 compat_cmsghdr __user * ucmsg; 144
        结构 的cmsghdr * kcmsg，* kcmsg_base; 145 compat_size_t
ucmlen;
。。。
149 kcmsg_base = kcmsg = ( 结构 的cmsghdr * ) stackbuf; 150 ucmsg = CMSG_COMPAT_FI
RSTHDR ( kmsg ) ; 151
        而 ( ucmsg ! = NULL ) {152
            如果 ( GET_USER ( ucmlen，&ucmsg-> CMSG_LEN ) ) 153
                返回 - EFAULT;
。。。
156         如果 ( CMSG_COMPAT_ALIGN ( ucmlen ) <157
                CMSG_COMPAT_ALIGN ( 的sizeof ( 结构 compat_cmsghdr ) ) )
158             返回 - EINVAL;
159         如果 ( ( ... ) ( ( ( CHAR __ 用户* ) ucmsg - ( 焦炭__ 用户* ) ... 160
                                + ucmlen ) > kmsg-> msg_controllen )
161             返回 - EINVAL;
。。。
166         ucmsg = cmsg_compat_nxthdr ( kmsg，ucmsg，ucmlen ) ; 167} 168 如果 ( kcmlen
== 0 ) 169

        返回 - EINVAL;
。。。
183 ucmsg = CMSG_COMPAT_FIRSTHDR ( kmsg ) ; 184 而 ( ucmsg ! =
NULL ) {185
        __get_user ( ucmlen，&ucmsg-> CMSG_LEN ) ; 186
        TMP = ( ( ucmlen - CMSG_COMPAT_ALIGN ( 的sizeof ( * ucmsg ) ) ) + 187
                CMSG_ALIGN ( 的sizeof ( 结构 的cmsghdr ) ) ) ;
188     kcmsg-> CMSG_LEN = TMP;
。。。
193     如果 ( 调用copy_from_user ( CMSG_DATA ( kcmsg ) ，194
                CMSG_COMPAT_DATA ( ucmsg )
195             ( ucmlen - CMSG_COMPAT_ALIGN ( 的sizeof ( * ucmsg ) ) ) ) )
。。。
212}
```

图2：在Linux的2.6.9双取漏洞

作为通过BU FF 呃溢流，造成特权escala-重刑，信息泄漏或内核崩溃。

在本文中，我们研究了有害的双重取和良性双取。即使是 - nign双取当前不容易，当代码被更改或更新在将来（当双提取的数据被重复使用）他们中的一些可以变成有害的。此外，一些良性双取它们会导致性能下降，当取之一是多余的（在第5部分中讨论）。

不仅出现在Windows内核[14]，而且在Linux内核漏洞双取。无花果URE 2只显示了一个双取臭虫在Linux中2.6.9，该报告为CVE-二〇〇五年至2490年。

在网络连接文件 compat.c，

当用户控制的内容通过复制到ker- NEL SENDMSG ( ) 相同的用户数据，而无需在第二时间执行健全性检查被访问两次。这可能会导致一个内核BU FF 呃溢流而目前，脱颖而出可能导致权限提升。功能 cmsghdr_from_user_compat_to_kern() 在两个步骤中的工作：它第一个检查在第一个循环中的参数（线151），并复制在第二回路（管线184）中的数据。然而，只有第一个取的（线152）

ucmlen 在使用前，检查（线156-161），而之后的第二抓取（线185）不存在的检查是 -

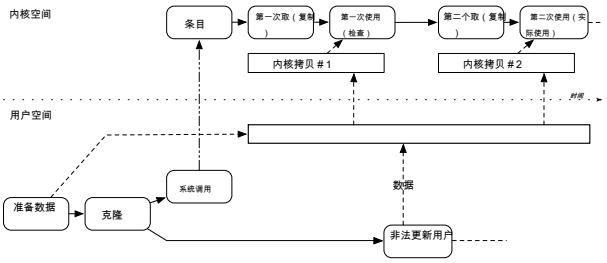

图3：双获取与传递函数

前使用，这可能会引起溢流在复制oper-通货膨胀，可以利用通过修改消息来执行任意代码（线195）。

的大量方法已经提出了在内存访问等级数据争用检测。静态接近ana- lyze运行它的programwithout [35，28，12，6，10，19，38]。然而，它们的主要缺点是它们产生漏电了大量的不实报道，由于缺乏程序的完整执行上下文。动态接近执行该程序，以验证数据争用[31，16，15]，检查是否比赛可在执行导致程序失败。动态方法通常控制活性线程调度器来触发SPECI音响Ç的交错，以增加一个错误表现[41]的概率。然而，运行时开销是一个严重的问题和驱动程序代码测试需要SPECI网络I2C硬件或专用模拟的支持。不幸，

（1）双取错误是由内核和用户空间之间的竞争条件引起的，这是二 FF 从共同的数据争erent因为争用条件是国家环保总局由内核和用户空间分级。对于数据的比赛，在同一个地址空间中存在的读写操作，大部分以前的**方法通过识别检测数据争 *所有* 读取和写入操作访问相同的内存位置。然而，事情迪 FF erent对于双取错误。内核只包含而写入驻留在用户线程两次读**取。此外，如果有一种可能性，即内核取和使用相同的存储器位置两次，因为恶意用户进程可以具体来说被设计为在网络之间写入RST和第二取双取存在错误。

(2a) 内核的参与使得双取虫二 FF 从访问数据的方式数据争erent。

在Linux中，从获取用户数据

空间到内核空间依赖于传递给传递函数的SPECI音响参数C（例如， 调用 copy_from_user ( )

和 GET_USER ( )) 而不是直接取消引用用户的指针，这意味着常规赛的数据去tection方法的基础上指针引用不适用了。

（3）此外，在Linux上双取的错误比普通的数据竞争或Windows双取错误是比较复杂的。如图3所示，在Linux的双取错误需要第一个获取该复制数据，通常随后是第一个检查或使用所复制的数据，然后进行第二次再获取该复制相同的数据，和第二使用的相同的数据。虽然dou- BLE取可以通过匹配读取操作的模式进行定位，利用获取的数据的差异很大。例如，除了被用于验证，第一个取出的值可被可能复制到其他地方，供以后使用，这意味着第一个使用（或检查）可以是在时间上不存在。此外，所获取的值可以作为参数传递给其它功能用于进一步使用通过。因此，在本文中，**我们去网络网元 *采用* 在双取为一个条件检查（compar- ISON读取的数**据），分配给其他变量，函数调用参数传递，或者使用所取得的数据的计算。我们需要考虑到这些双取特性。

由于这些原因，确定双取错误重新奎雷斯一个专门的分析和以前的方法**要么是不适用或不è FF ective。**

2.4 Coccinelle的

COCCINELLE [17]是一个程序的匹配和transforma-灰发动机与在C代码中指定期望的匹配和变换的专用语言SMPL（Seman-抽动补丁语言）。Coccinelle的最初焦油geted抵押演变Linux驱动程序，但现在被广泛用于系统代码的网络nding和FI兴错误。

COCCINELLE的策略为横动控制 - 溢流曲线图是基于时序逻辑CTL（计算树逻辑）[3]，并且在实施COCCINELLE图案匹配是路径敏感，这实现更好的代码覆盖。COCCINELLE被高度优化的IM证明当详尽地遍历所有的执行路径的性能。此外，Coccinelle的是不敏感的换行符，空格，评论，等。此外，图案 - 基于分析被直接施加到源代码，因此是定义为宏德音响操作，如

GET_USER（）要么 __ GET_USER（）匹配，这有利于基于的传递函数的identi音响阳离子双取的检测期间不会被扩大。因此，Coccinelle的是我们开展我们的基于模式匹配双取的研究合适的工具。

3次提取以Linux内核

在本文中，我们在Linux内核中双取的研究分为两个阶段。如图4所示，在第一个阶段，我们分析了Linux内核与

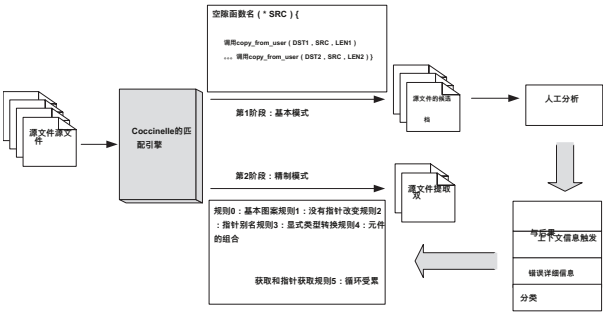

图4：我们的两相Coccinelle的基于双取状况检测过程概述

使用identi音响ES当一个函数的传递函数的多个调用一个基本双取图案COCCINELLE发动机。然后我们手动调查易拉罐didate科幻LES通过模式匹配发现，归类，其中双取发生的情况，当双取错误或漏洞很容易恰巧基础上，是有关错误的上下文信息。在第二阶段的基础上，从手动分析所获得的知识，我们开发使用Coccinelle的发动机系统检测整个内核，我们也可以用来分析额外FreeBSD和Android的双取的错误和漏洞更精确的分析。

3.1基本模式匹配分析

有些情况下，双取是难以避免的情况下，并且存在大量的Linux内核的两倍获取相同的数据的功能。根据DE音响定义，当相同的用户数据是一个短的时间间隔内取两次的双重取可发生在内核。因此，我们可以得出结论，我们将用它来匹配所有潜在的双重取仰卧uations一个基本格局。的图案，其中一个核函数是使用传递函数来两次取至少从相同的用户存储区的数据的**情况相匹配。在Linux内核的情况下，比赛的传递函数主要是 GET_USER（）和 调用copy_from_user（）在其所有的变种。该模式允许复制的目标，并且所复制的数据的大小为二 FF erent，但副本（在用户空间中的地址）的**源必须是相同的。如图4所示，我们实现在发动机Coccinelle的基本模式匹配。

我们的方法检查所有源代码连接LES Linux内核和检查的核函数中含有杂质的来自相同用户的指针取数据传送功能的两个或更多个调用是否。从39906 Linux的源网络莱，17532网络莱属于驱动程序（44％）和10398网络莱属于非x86硬件architec-

不能用Jurczyk和冷风的基于x86的技术来分析功能（26%）。我们人工分析匹配的内核函数来推断知识上双取，即用户数据是如何转移到并在内核中使用的特性，这有助于我们开展的分类双取的情况下，在我们讨论3.2节。该手册分析还帮助我们重新连接的NE我们的模式匹配的方法和更精确地检测出真正的双取的错误，如第3.3节解释。

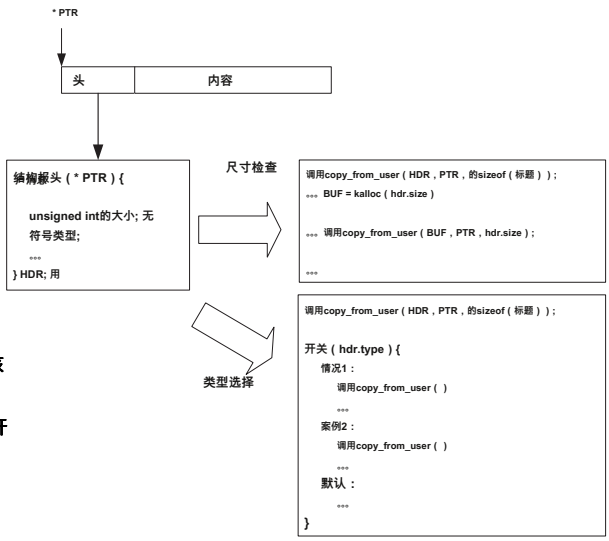

图5：如何留言结构导致双取指令

在调查过程中，我们注意到，有很多地方的传递函数从迪获取数据的情**况下 FF erent地址或来自同一个地址，但与二 FF erentØ FF 集。例如，内核**函数可提取一个特定的C语言结构中的元素的整个结构复制到内核分别代**替。通过添加二 FF erentØ FF 套到struc-自命的起始地址，内核取迪 FF 分开**struc- TURE，这导致多个取的erent元件。AN-其他常见的情况是加入一个**固定的Ø FF 集到源指针，以便处理很长的消息得非常好国家环保总局，或**者仅仅使用自增量（++），以在一个循环中自动处理的消息。所有这些情况造成基本模式匹配的误报，和226的情况下我们的初步报告被鉴定网络版误报，已经因为他们不被视为双取的情况和已经自动在我们重新连接斯内德相中移除不能引起双取错误，因为每个单件的消息仅取一次。

图5示出的情形：从用户空间到内核（驱动程序）中复制的一个消息空间通常由两个部分组成，报头和主体。标题包含有关该消息的一些元信息**，例如消息的指示符 *类型* 或者**

***尺寸* 消息体。由于消息有二 FF ER- ENT类型和消息长度也可以变化，内核通常取（拷贝）的标题第一个决定哪些BU FF 要创建ER型需求或howmuc**h空间需要分配的完整信息。仲 - OND取然后复制完整的消息到AL-位于B**U FF 呃的SPECI网络版类型或大小的。该仲OND取不仅复制体，还会将自**身复制完整的消息包括已获取已经头。因为该消息的报头被取出（复制）**两次，双取的情况出现。双取状况变成双取错误时从尺寸或类型的信息 第二 取被用作用户可能已经改变了两次存取之间的尺寸或类型的信息。如果，例如，尺寸信息被用于控制BU FF 呃接入，双取虫变成了漏洞。**

我们的研究精矿对环境的非derstanding的第一个阶段，在双取容易的事情发生，而不是详尽网络nding宝结下双取错误。尽管分析和定性不完全自动的，它不仅造成在需要手动调查，只用了几天来分析它们，使得所需**手册C 90名候选人 FF 我们的方法ORT可以接受的。**

3.2双取分类

由于我们人工检查双重取考生，我们注意到，有中双取易于发生三种常见**的场景，我们归类为 *类型选择*，*大小检查* 和 *浅拷贝*。现在，我们在详细**讨论这些。

大部分时间，从用户空间到内核空间复制数据是经由一个传递函数的单一调用简单。然而，当数据具有可变类型或可变长度，取决于数据本身的**事情变得复杂。这样的数据通常与开始 *头*，其次是数据的 *身体*。在下文中，我们认为这些数据是 *消息*，因为我们凭经验发现，可变数据被经常使用**的驱动程序将消息传递给从用户空间的硬件。

其中一个消息头被复制两次的双取的情况下可以很容易地通过仅复制消息主体在第二提取，然后加入与主体头部来避免。然而，在第二个步骤复制完整的消息是更方便，目前，脱颖而出在Linux内核中经常出现这样的双取局面。此外，Linux内核的大部分都是旧的，即，他们已经取双安全漏洞是已知的或了解之前开发的。因此，我们将讨论在内核中这样的双取情况

更多的细节和还强调，我们的人工分析过程中发现三种情况。

3.2.1类型选择

当消息报头用于类型选择，其中双取出现一个常见的情况是。换句话说，该消息的报头被取出第一个识别消息类型，然后将整个消息被取出和处理取决于identi-音响版型。我们已经观察到，这是很常见的Linux内核，在一**个驱动程序中的一个功能的设计通过使用来处理多个类型的消息 开关 语句**结构，其中每一个特定的消息类型被取出，然后被处理。的网络连接的R**ST抓取（消息类型）的结果是在所使用的 开关**

语句的条件，并在每一个的情况下， 开关，
该消息随后由一第二复制取到本地BU FF 一个特定的C型的ER（再加工）。

图6示出一个双重取situ-通货膨胀的一个例子，由于在FI文件类型选择 cxgb3_main.c，
网络驱动程序的一部分。功能 cxgb_extension_ioctl（ ） 第一个取出的消息的类型（用于连接的硬件的命令）到 CMD 从指针到用户空间 useraddr 在行2136它然后决定基于 CMD 该键入消息具有（例如， CHELSIP_SET_QSET_PARAMS，CHELSIP_SET_QSET_NUM 要么 CHELSIO_SETMTUTAB） 并复制完成消息到对应的结构（类型的 ch_qset_params，ch_reg，ch_mtus，...）。 **该类型的信息将被取出的第二时间作为整个消息的一部分（线2149**，2292，2355 respec- tively）。只要消息的报头部分未被再次使用时，双取在这种情况下不引起双取错误。然而，如果第二报头部分（类型/命令）获取将被再次使用，因为恶意用户可能已经改变了两次存取之间的标题**可能会出现问题。如果是 cxgb_extension_ioctl（ ） 手动investi- gation揭示在BU没有用的类型的部分 FF ERS T，EDATA， 男， ... 这里的双取的情况**不会导致双取漏洞。

我们发现有11次出现这种双重取cate-血腥的，他们的9驱动程序。11种现象没有使用的第二的头部部分提取，并且因此，它们不引起双取错误。

3.2.2尺寸检查

发生另一种常见情况，当该消息的实际长度可以变化。在这种情况下，该消息头是用于标识完整的消息的大小。消息头被复制到内核中第一个得到消息的大小（第一个提取），检查它的有效性，

```
2129 静态INT cxgb_extension_ioctl（ 结构 net_device的* dev的，
                                                    无效__ 用户* useraddr ）
2130 {
。。。
2133 U32在cmd;
。。。
2136        如果（ 调用copy_from_user（CMD，useraddr，的sizeof（CMD））） 2137
              返回 - EFAULT;
2138
2139        开关（ CMD ）{2140
            案件 CHELSIO_SET_QSET_PARAMS : {
。。。
2143            结构 ch_qset_params吨;
。。。
2149            如果（ 调用copy_from_user（＆T，useraddr，的sizeof（T））） 
2150                返回 - EFAULT;
2151            如果（ t.qset_idx> = SGE_QSETS ）
2152                返回 - EINVAL;
。。。
2238            打破;
2239}
。。。
2284        案件 CHELSIO_SET_QSET_NUM : {2285
            结构 ch_reg EDATA;
。。。
2292            如果（ 调用copy_from_user（EDATA，useraddr，的sizeof（EDATA））） 
2293                返回 - EFAULT;
2294            如果（ edata.val <1 ||
2295                    （edata.val> 1 &&！（.....））） 
2296                返回 - EINVAL;
。。。
2313            打破;
2314}
。。。
2345        案件 CHELSIO_SETMTUTAB : {2346
            结构 ch_mtus米;
2355            如果（ 调用copy_from_user（米，useraddr，的sizeof（米））） 
2356                返回 - EFAULT;
2357            如果（ m.nmtus！= NMTUS ）
2358                返回 - EINVAL;
2359            如果（ m.mtus [0] <81 ）
2360                返回 - EINVAL;
。。。
2369            打破;
2370}
。。。
2499}
```

图6：双取状况属于选型类别中 cxgb3的main.c

分配一个本地BU FF 必要的大小的ER，则仲OND取如下的整个消息，其中还包括头部，复制到所分配的BU FF 呃。如使用只要仅第一个提取的大小，而不是从第二首标检索提取，在这种情况下，双取不会引起双取漏洞或错误。然而，如果大小从第二提取和使用的首标检索，内核变得恶意用户可能已经改变了报头的大小元件vul- nerable。

一个这样的双取错误（CVE-2016-6480）的网络连接文件被发现 commctrl.c 在Linux 4.5的的Adaptec RAID CON组troller驱动程序。图7示出了再spo**nsible功能 ioctl_send_fib（ ） 其中取数据从用户空间指出由指针 ARG 通**过

调用copy_from_user（ ） 两次在线81和线116。

```
60 静态INT ioctl_send_fib ( 结构 aac_dev * dev的 ,
                                        无效__ 用户* ARG )
61 {62
        结构 hw_fib * kfib;
。。。8
1       如果 ( 调用copy_from_user ( ( 无效* ) kfib , 阿根廷 , 的sizeof ( ... ) ) ) {82
            aac_fib_free ( fibptr ) ; 83
            返回 - EFAULT; 84}

。。。
90大小= le16_to_cpu ( kfib-> header.Size ) + 的sizeof ( ... ) ;
91      如果 ( 大小<le16_to_cpu ( kfib-> header.SenderSize ) ) 92
            大小= le16_to_cpu ( kfib-> header.SenderSize ) ; 93
        如果 ( 大小>的dev-> max_fib_size ) {
。。。
101         kfib = pci_alloc_consistent ( dev亡> PDEV , 大小 , &DADDR ) ;
。。。
114} 115
116
        如果 ( 调用copy_from_user ( kfib , 精氨酸 , 大小 ) ) {117
            RETVAL = -EFAULT; 118
            去 清理; 119} 120 121


        如果 ( kfib-> header.Command == cpu_to_le16 ( ... ) ) {
。。。
128} 否则{
129         RETVAL =
                aac_fib_send ( le16_to_cpu ( kfib-> header.Command ) , ...
130                     le16_to_cpu ( kfib-> header.Size ) , FsaNormal ,
131                     1 , 1 , NULL , NULL ) ;
。。。
139}
。。。
160}
```

图7：在双取漏洞 commctrl.c
（CVE-2016-6480）

第一个取出的值被用于计算一个BU FF 呃尺寸（线
90），以检查的大小（线93）的有效性，并AL-定位BU FF 所计算的尺寸（线101）的ER，而第二拷贝（线116）获取与计算出的尺寸在整个消息**。注意变量 kfib 指着内核BU FF ER存储所述消息中的线被重复使用**

101.该消息的报头是大和各种首部的勒发言：被使用的消息一直后取出的**第二时间（例如， kfib-> header.Command**

在线121和129）。该函数还使用尺寸EL- EMENT的报头的第二时间线13**0，使双取漏洞恶意用户可能已经改变了 尺寸 网络连接视场的两次存取之**间的标题的。

我们观察到30次出现这样的尺寸检查双取易受其中22个发生在司机，其中4（所有驱动）的情况下。

3.2.3浅复制

最后特例双取场景中，我们identi-网络版就是我们所说的 *浅拷贝的问题。* 用户和内核空间之间的浅拷贝发生在一个BU FF ER（在第一个BU FF ER）在用户空间复制到ker- NEL空间，BU FF 呃包含一个指向另一个

```
55 静态INT sclp_ctl_ioctl_sccb ( 无效__用户* user_area ) 56 {57 结构 sclp_ctl_sccb ctl_sccb; 58 结构
 sccb_header * SCCB; 59 INT RC; 60 61 如果 ( 调用copy_from_user ( &ctl_sccb , user_area ,
                                                    的sizeof ( ctl_sccb ) ) )
62          返回 - EFAULT;
。。。
65 SCCB = ( 无效* ) get_zeroed_page ( GFP_KERNEL | GFP_DMA ) ; 66 如果 ( ！ SCCB ) 67

            返回 - ENOMEM; 68 如果 ( 调用copy_from_user ( SCCB , u64_to_uptr ( ctl_sccb.sccb )
 ,                                                  的sizeof ( * SCCB ) ) ) {
69          RC = -EFAULT; 70
            去 out_free; 71} 72 如果 ( sccb->长度> PAGE_SIZE || sccb->长度<8 ) 73


            返回 - EINVAL; 74 如果 ( 调用copy_from_user ( SCCB , u64_to_uptr ( ctl_sccb.sccb ) ,
                                                    sccb->长度 ) ) {
75          RC = -EFAULT; 76
            去 out_free; 77}

。。。
81 如果 ( copy_to_user ( u64_to_uptr ( ctl_sccb.sccb ) , SCCB ,
                                        sccb->长度 ) )
82          RC = -EFAULT;
。。。
86}
```

图8：在双取错误 SCLP ctl.c（ CVE-2016-6130 ） _

BU FF 呃在用户空间（第二BU FF ER）。转印FUNC-重刑只复制了第一个BU FF ER（浅表副本）和第二BU FF ER必须由传递函数的第二次调用被复制（以执行深层副本）。Some-有时需要将数据从用户空间复制到内核空间，对数据采取行动，并复制数据返回到用户空间。这样的数据通常被包**含在所述第二BU FF 呃在用户空间，并指出，一些在第一个BU指针 FF ER在含有附加数据的用户空间。传递函数执行在BU指向浅拷贝，因此数据 FF**呃通过转移FUNC-重刑复制必须明确复制为好，以便进行深拷贝。这种深层副本会引起的传递函数的多个IN-职业为每个传递函数与一个二调用其不**必是双取 FF erent BU FF 呃被复制。我们的司机观察到这样的情况下31，1**9人。

在执行与传递函数深层副本，只有做浅拷贝会导致程序员引入错误的复**杂性，我们发现了一个这样的错误在网络文件 sclp_ctl.c 的IBM S / 390 S**CLP控制台驱动，这里的错误是由浅拷贝问题（CVE-2016-6130）引起的。功能

sclp_ctl_ioctl_sccb 在图8中进行的数据结构的一个浅拷贝从用户空间指向

user_area 成 ctl_sccb（ 线61）。做一个深层副本，然后它通过具有复制从用户空间指向另一个数据结构 ctl_sccb.sccb。 然而，的大小

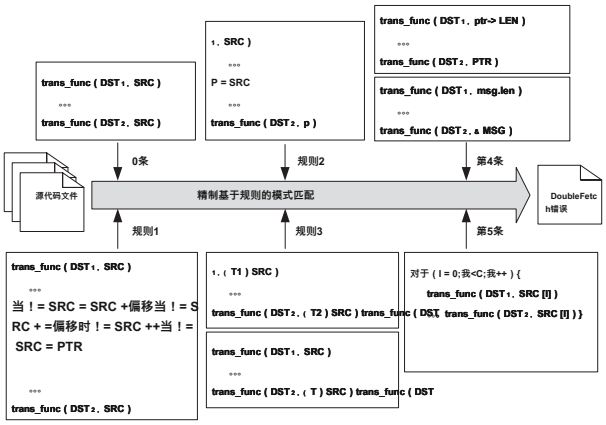

图9：重新定义网络Coccinelle的基于双取错误检测

数据结构是可变的，使 *大小检查* SCE-nario的。为了复制数据，其第一个提取的数据结构的首部到新创建的内核空间指向 SCCB（ 线68 ），以获得在数据长度

sccb->长度 这是检查在线路72有效性然后，根据 sccb->长度， 它复制整个CON组帐篷采用在管线74第二取最后，在线81，该数据被复制回用户空**间。尽管它看起来像的传递函数中的行74二者调用和81使用相同的长度 sccb->长度， 线81 actu-烯丙基使用该值作为线74复制（第二提取）而线7**4使用从网络连接的值取RST。

再次，这是因为用户可能在线路68和74但是已经改变了两次存取之间的值的双取错误，这种双重取错误不造成一个漏洞，因为既不可以在内核可以由无效大小坠毁给出的传递函数，也不能将信息泄漏发生时，内核拷**贝备份数据超出了它收到较早，因为复制的BU大小 FF 呃位于其自己的内**存页。触发错误将简单地在系统中呼叫的终止，结束与一个错误代码在管线82的双读取错误的尝试已经消除在Linux中4.6。

3.3重新定义科幻双取错误检测

在本节中，我们提出它采用定义重新连接我们研究的第二阶段双取即再次基础上，Coccinelle的匹配引擎的错误检测方法。虽然我们研究的第一个阶段是确定和在双取发生CAT-egorize方案中，仲OND阶段利用从第一个阶段获得的知识来设计针对SPECI Fi的凯莉识别双取的错误和改进的分析漏洞。

如图9所示，除了基本的双取模式匹配规则（ 规则0 ）这是时触发

当两个读取来自同一源位置的获取数据复位此输出，我们将以下五个科幻额外的规则，以提高精确度，以及发现角落的情况。该Coccinelle的引擎**通过一个分析源网络莱时应用这些规则之一。双取错误可能涉及二 FF ere**nt传递函数，因此，我们不得不采取从用户空间复制数据的四个传递函数**（ GET_USER ( ) ，__get_user ( ) ，调用copy_from_user ( ) ，__cop**y_from_user ( ) ）成CON组闪电状发病。我们用 trans_func ( ) 在图9中，以在Linux内核表象发送的任何可能的传递函数。

规则1：任何指针的变化。 在检测双取错误的最关键规则保持用户指针两次存取之间保持不变。否则，二 FF erent数据被提取的每个时间，而不是相同的数据是双牵强，和误报可以引起。如从规则1在图9中可以看出，这**种改变可能包括自增量（ ++ ）的情况下，添加邻 FF 集，或其他值的分配，**以及相应的子牵引的情况。

规则2：指针别名。 指针别名是COM的星期一双取的情况。在一些情况下，用户指针被分配给另一个指针，因为origi- NAL指针可能会改变（例如，通过处理一个循环内部分长mes-先贤部分），同时使用两个指针是更方便的，一个用于检查数据，另一个用于使用所述数据。由于可以从第2图9可以看出，这种分配可能会出现在一个函数的开头或两个取的中间。缺少别名情况可能会导致假阴性。

规则3：显式类型转换。 明确的指针类型转换被广泛使用时，内核从用户空间fetch- ING数据。例如，在大小入住ING场景中，消息指针将被转换为一个头指针以获得在FI头RST取，然后在第二取再次用作消息指针。如从规则3在图9中可以看出，任何两个源指针的可能涉及类型转换。缺少类型转换的情况下可能会导致假阴性。此外，显式指针类型转换通常与指针别名结合，导致相同的MEM ORY区域由两个类型的指针进行操作。

芸香4：元件的组合提取和指针取。

在一些情况下，用户指针既用于获取整个数据结构，以及通过解引用的指针数据结构的元素取只是一部分。例如，在大小入住ING场景中**，用户指针是第一个使用来获取消息长度 GET_USER (LEN，ptr-> LEN ），然后到整个消息在由所述第二获取复制**

调用copy_from_user (味精，PTR，LEN ），这意味着这两个取不使用完全相同的指针作为传递函数的参数，但它们覆盖相同

语义值。正如我们可以从规则4在无花果URE 9看到，这种情况可以使用用户指针或数据结构的作为转印FUNC-蒸发散的参数的地址。这种情况通常有明确的指针类型转换出现，如果这种情况是错过了假阴性可以造成的。

第5条：回路的参与。 由于Coccinelle的是路径 - 敏感，当循环出现在代码，在一个循环中一个传递函数的通话将被报告为两个电话，这可能引起误报。此外，如可看到的从第5图9中，当有两个取在一个循环中，第二取最后一次迭代的和第一个提取下一迭代将被匹配为双取。这种情况下，应为假阳性被删除，因为穿越iter- ations当用户指针应该已经改变，这两**种取越来越迪 FF erent值。而且，使用数组箱子复制二 FF 在循环中erent值**也引起误报。

表1：基本双人获取分析结果种类

	出现次数		在驱动程序
大小检查30		33%22	73%
选型11		12%9	82%
浅复制31		34%19	61%
其他	18	20%7	39%
总	90	100%57	63%
蟮	五	6%4	80%

表2：定义双取虫再Fi检测结果的内核

	总文件中报告				
		文件也是如	错误	大小校验。	类型SEL。
Linux的4.5	39906	53	五	23	6
安卓6.0.1 35,313		48	3	18	6
FreeBSD的	32830	16	0	8	3

4评价

在本节中，我们提出我们的研究，这包括两个部分的评价：当应用于三种开源内核的人工分析的统计数据，并重新定义网络方法的结果：Linux的Android和FreeBSD的。我们获得了可在分析的时候最先进的最新版本。

4.1统计与分析

在Linux的4.5，有52881网络莱总与他们的39906是源网络LES（以.c或.H的一个文件扩展名），这是我们的分析目标（其它网络文件被忽略）。

17532源连接LES属于驱动器（44%）。源连接LES和人为UAL检查以去除假阳性的基本模式匹配之后，我们获得了90双取候选连接LES作进一步**检查。我们分类的考生进入三双取场景 *尺寸检查*， *类型选择* 和 *浅表副本*。 他们是如何，而用户空间中的数据复制到内核空间，以及如何然后将数**据在内核中使用双取时最常见的情况。我们已经讨论了与真实的细节，这些场景双前一节中获取错误的例子。如表1所示，我们发现90名候选人，30个相关的大小检查的情况下，11条是关于类型选择方案中，和31分别与所述shal-低拷贝方案中，占33%， 12%和34%。18名候选人没有网络连接牛逼到三种情况之一。

此外，57出90名候选人是Linux驱动程序的一部分，其中，22人大小检查有关，9人类型选择相关的19例浅拷贝相关。

最重要的是，我们发现网络连接已经以前未知的双取错误，这些错误包括四个尺寸检查SCE-narios并且也渴望是 - 对大小检查方案一个浅拷贝的情况。他们三人都是利用的漏洞。该网络已经虫子已经重新移植，他们都已经被培养–ERS Rmed指CON网络连接，并同时被固定成本。从统计结果中，我们可以注意以下几点：

1. 57出来的候选人驾驶员相关的90（63%）和22个30（73%）的尺寸的检查的情况下，9出来的类型选择例11（82%）和19个31（浅拷贝的情况下61%）发生在驱动程序。

2. 4双取出虫子出来的5（80%），我们发现里面的驱动程序和属于尺寸检查cate-血腥。

总体来说，这导致一个结论：大部分双取不引起双取的错误，并且双取更可能在司机发生。然而，一旦双取是由于尺寸检查，开发商必须要小心：四出22尺寸检查的情况在司机竟然是双取错误。

三种开源内核的4.2分析

基于双取基本模式匹配和人工分析，我们重新定义科幻我们取双模式，开发了一种基于Coccinelle的发动机新的双取的错误检测分析。为了充分evalu-吃了我们的方法，我们分析了三种流行的开源内核，即Linux的Android和FreeBSD的。结果示于表2中。

对于Linux内核，该实验是在4.5版本，这是最新版本进行该实验时进行。分析了大约10分钟，报道53候选人科幻LES。53名候选人的调查显示网络连接已经真正的双取的错误，这也是由以前的手工分析发现。已报告的科幻LES，23个大小检查有关，和6类型选择关联。

对于Android，尽管它使用Linux作为其ker- NEL还有，我们分析这是基**于Linux 3.18 6.0.1版本。还有迪 FF 之间erences Android的内核和原始Lin**ux内核：一个为Android内核是一个主流Linux内核，与用于SPECI音响Ç Android装置附加的驱动程序，以及其他附加功能，如增强的进行功耗管理换货或更快的图形支持。我们分析了约9分钟和48报道候选人科幻莱，其中包括7名科幻LES未包括在原来的Linux内核报告。已报告的考生，三是真正的双取的错误，包括用上面的Linux 4.5报告共享两个，一个是只报道了Android系统。在这些结果，18名候选人分别为大小检查有关，和六个候选人类型选择关联。

对于FreeBSD，我们需要改变传送FUNC-蒸发散 调用copy_from_use
r（ ）和__ 调用copy_from_user（ ）
到FreeBSD中相应的， COPYIN（ ）和
copyin_nofault（ ）。 我们从主分支获取源代码 3。这种分析了约2分钟，并且只有16个网络莱进行了报道，但没有人竟然是一个脆弱的双取错误。其中再移植候选人，8个人大小检查有关，三人类型选择关联。这是有趣的是，5这些16科幻LES的是良性的双读取，这将有双取的错误，但都是预先VENTED通过额外的检查方案。FreeBSD下的培养–ERS似乎更加了解双取的错误，并试图积极地阻止他们。相比较而言，对于Linux，只有5个的53份报告额外的检查机制保护。

在这个实验中，我们只计算规模入住ING和类型选择的情况下，因为重新连接斯内德模式匹配的方法丢弃浅拷贝案件无法引起双取错误。我们的方法的双取模式相匹配，从相同的存储器区域，其忽略了第一个BU取数据**FF ER中提取的浅拷贝的情况下，并只考虑多取相同的第二BU FF 呃。这样**的浅拷贝的情况下通常与其他方案，如大小检查和类型选择相结合。在表2中，尺寸检查Linux内核的情况下，还包括发生在浅表副本场景一例。

3 从GitHub截至7月2016（ https://github.com/freebsd/freebsd ）

5讨论

在本节中，我们将讨论发现双取bug和漏洞的Linux 4.5以及如何获取双能双的情况下获取的存在可以防止错误。我们也理解我们的网络ndings和我们的方法的局限性。

5.1检测到的错误和漏洞

根据我们的方法，我们发现在总共六个双取错误。其中有五个是以前没有（报道先前未知的漏洞CVE-2016-5728，-6130， -

6136，-6156，-6480）和第六个（CVE-2015年至1420年）是一种双取存在于基于一个较旧的Linux内核的最新的Android（版本 - 锡永6.0.1）（版本 - 错误锡永3.18）包含的bug，已网络连接的主线Linux内核因为Linux 4.1固定的。在网络连接的三五个新的错误是可利用的双取vulnerabili-关系（CVE-2016-5728，-6136，-6480）。在网络连接的四五个都在驱动程序（CVE-2016-5728，-6130，-6156，-6480）。所有的错误都被报告给Linux内核谁拥有CON组fi Rmed指他们main- tainers。所有这些报告的bug都是固定成本为4.8的Linux的。我们没有网络连接ND在FreeBSD的任何新的双取错误。所检测到的错误的细节示于表3中。

所提出的方法identi科幻ES大批的只有少数是双取的情况下双取的错误（甚至漏洞）。如何 - 以往，即使在情况下，我们称之为良性双取的情况目前还不是错误的，他们可以很容易地变成一个双取错误或漏洞的代码时没有特别注意的双获取最新情况。调查双取错误CVE-2016-5728的补丁历史时，我们观察到这样的情况发生。当显影剂从MIC主机驱动器到超过的Virtio的PCIe（VOP）驱动移动的功能，因此引入一个双取错误已引入的仲OND取出值的重用。我们今后工作的一个重要组成部分将是防止这种良性的双重演变成有害的获取情况。

我们没有网络连接的第二任漏报，同时手动检查的Linux内核源代码网络莱随机样本。

5.2比较

只有少数系统的研究已经进行了双取进行。Bochspwn [14，13]是唯一的方法类似足以保证与比较。Linux的3.5.0与Bochspwn的anal- ysis没有网络连接的第二任双取的bug，同时产生高达两倍获取日志200KB。在相同的内核，我们的做法identi-

表3：Identi网络连接的描述ED双取错误和漏洞（*）

标识	文件	描述
CVE-2016-5728 *	mic_virtio.c MIC架构VOP（虚拟I / O在PCI e）的驱动程序 *Linux的4.5*	**在比赛中状态 vop_ioctl 功能允许本地用户从内核存储器获取敏感信息或引起通过改变某些头拒绝服务**（存储器心病ruption和系统崩溃），又名“双重取”漏洞。 *属于大小检查方案。*
CVE-2016-6130	sclp_ctl.c IBM S / 390 SCLP控制台驱动 *Linux的4.5*	**在比赛中状态 sclp_ctl_ioctl_sccb 功能允许本地用户通过改变一定长度值，以获得从内核存储器敏感信息**，又名“双重取”漏洞。 *属于大小检查方案。*
CVE-2016-6136 *	auditsc.c Linux审核子系统 *Linux的4.5*	**在比赛中状态 audit_log_single_ execve_arg 功能允许本地用户绕过预期的字符集限制或改变某些字符**串破坏系统调用audit- ING，又名“双读取”漏洞。
CVE-2016-6156	cros_ec_dev.c Chrome OS的嵌入式控制器驱动程序 *Linux的4.5*	**在比赛中状态 ec_device_ioctl_xcmd 功能允许本地用户导致拒绝服务（外的界定数组访问）通过改变**一定大小值，又名“双重取”漏洞。 *属于大小检查方案。*
CVE-2016-6480 *	commctrl.c Adaptec的RAID控制器驱动程序 *Linux的4.5*	**在比赛中状态 ioctl_send_fib 功能允许本地用户导致拒绝服务（外的界定访问或系统崩溃）通过改变一**定大小值，又名“双重取”漏洞。 *属于大小检查方案。*
*CVE-2015 - 1420 **	fhandle.c 文件系统 ***安卓6.0.1，（Linux的3.18）***	**在比赛中状态 handle_to_path 功能允许本地用户导致拒绝服务（外的界定数组访问）通过改变一定大**小值，又名“双重取”漏洞。 *属于大小检查方案。*

网络编3超过了上面讨论的6个双取臭虫（其它3级的错误，我们发现在网络连接LES中不存在在Linux中3.5.0）。

很可能Bochspwn不能科幻ND这些缺陷，因为它们存在于驱动程序。事实上，动态方法不能支持不对应于继电器硬件或硬件的模拟驾驶。Bochspwn重新移植的只有28％，为ker- NEL的指令覆盖，而我们的方法静态分析完整的源代码。

至于为电子商务 FFI ciency，我们的方法只需几MIN-茨进行整个Linux内核源代码的路径敏感的探索。相比之下，Bochspwn引入了严重的运行时开销。举例来说，他们需要的模拟器15小时开机的Windows内核。

虽然只用了几天调查90双取的情况下，Jurczyk和冷风没有报告他们需要调查的双重获取由他们的模拟器生成的日志的200KB的时间。

5.3双取虫防治

尽管我们提供了一个分析，以检测双取漏洞，开发者仍然必须了解他们是如何发生的，并先发制人地防止重复读取错误。HU-人的错误是与通向新的双取的情况下可变信息打交道时，在驱动程序的开发可以预期的。

（1）不要复制标头的两倍。 双取situa-蒸发散可以完全如果第二只获取拷贝邮件正文和不完整的消息，该消息的副本的第二时间报头避免。例如，在Android的6.0.1（Linux的3.18）双取漏洞在Linux的4.1通过仅复制所述主体在所述第二提取解决。

（2）使用相同的值。 双取局面变成一个错误时，有来自读取操作的使用“相同”的数据，因为（恶意软件），用户可以改变两个取之间的数据。如果培养–ERS仅从取的一个使用数据，避免问题。根据我们的调查，大部分的双取的情况下是良性的，因为他们只用了第一个获取价值。

（3）覆盖的数据。 也有在其中数据需要抓取和使用情况的两倍，例如导出，完整的消息被传递给一个二 FF erent FUNC-重刑处理。解决这种情况，并消除双取错误的一种方法是覆盖从第二头与被取出第一个头部取。即使恶意用户更改了两次存取之间的标题，变化就没有影响。这种方法是在FreeBSD的代码被广泛采用，如 SYS的/ dev / AAC / aac.c 和

SYS的/ dev /的aacraid / aacraid.c。

（4）比较数据。 以解决双取虫的另一种方法是将数据从比较中第一个使用它之前获取到的第二获取数据。如果数据是不一样的，操作必须被终止。

（5）同步抓取操作。 防止双取虫的最后的方法是使用同步方法，以保证两个不可分割的操作，如锁或关键部分的原子。只要我们瓜拉尼发球不能两者之间取来改变所获取的价值，那么无可厚非会fetch-荷兰国际集团多次了。然而，这种做法会招致性能惩罚内核的，同步的临界区介绍。

由于 *比较数据* 方法并不需要修改非常多的源代码，大部分identi-网络的编双取，我们发现已经以这种方式被Linux开发人员（修补漏洞CVE-2016-5728， -

6130，-6156，-6480）。如果从两个取重叠数据部分是不一样的，内核将重新现在把一个错误。有人可能会说，这将是更好的避免双重获取与任何其他网络的报头的前三个建议。然而，COM的配对数据有两个好处：它不仅允许恶意用户去tecting攻击，但也从其中的数据没有被恶意更改的情况来保护（例如，通过在用户空间代码一些bug）。

我们已经实现了 *比较数据* 在Coccinelle的方法作为自动补丁注入代码从网络连接的数据进行比较RST与来自第二数据取在地方取其中的双取错误已被找到。它能够自动修补所有尺寸检查双取的错误，这占大部分identi网络版的bug（5/6）的。

5.4解读结果

双取对内核去velopment一个根本性的问题。流行的操作系统，如Windows，Linux和Android和FreeBSD的所有在过去的双取的错误和漏洞。双取问题有着悠久的历史，和一个错误，我们identi网络版（CVE-2016-

6480）已经存在了十几年。

双取很流行，有时难免内核。我们从我们检测到的事件分为三种典型的双取场景。这些双取的63％以上发生在驱动程序，这意味着司机是重灾区。四出音响的五个新bug属于尺寸检查方案，表明审核双取错误的可变长度消息处理的需求。

在Linux内核，双取的错误更COM的丛超过inWindows因为传递函数在双取虫独立于应用的读取，mak- ING更难弱势双取分离良性的。先前的动态方法，并没有发现任何双取臭虫在Linux中，我们的静态方法发现了一些，这表明一个简单的静态分析的强大功能。

我们的方法需要手动检查，然而，人工检查不必重复了完整的内核为未来的分析可以限于改变网络LES。此外，开发自动identi网络上课静态分析双取精度更高的错误将花费更多的时间比开发我们目前的做法，在迪运行**它 FF erent仁，和手动结果调查一起。此外，我们的分析和分类之前，它不**是在哪些情况下双取的错误发生在需要以设计出更加精确的静态双取错误分析Linux内核知识闻名。随着重新连接斯内德的办法，一个只会不得不看53潜在的双重取的错误，而不是在所有90个双取的情况。因此，我们的AP-proach的人工分析部分是不可避免的，但高度好处网络官方。

作为预防，所有四个大小检查的bug被修补 *比较数据* 方法，表示双取操作不能完全避免，因为修补的情况下仍然中止客户端程序通过返回错误。此外，即使是良性的双取的情况并不安全，因为他们可以很容易变成有害的。一个这样的bug（CVE-2016-5728），是由一个代码更新良性双取的情况介绍。如何 - 以往，大多数的这些潜在的案件没有固定成本，因为他们目前不容易。

即使双取是良性的，也就是说，是不是vulnera- BLE，可以考虑性能问题，因为取（的传递函数调用）中的一个是多余的。

5.5局限性

我们专注于在双取出现的Linux的源代码的基于模式的分析，分析的情况。然而，分析的性质防止在一个较低的水平时发生双取，例如，在预处理或编译的代码的检测。

双取的错误，甚至可以发生在宏。在一个这样的情况下，[24]，宏读取**指针两次，第一个时间到试验 空值 而第二次使用它。然而，由于两个取之**间的电位变化的指针，空指针碰撞可能引起的。

双取的错误，也可以通过编译器优化介绍。然后，它发生在编译的二进制而不是源代码。威廉[37]最近发现了这样的在Xen管理程序，这是因为该**指针共享存储器区不标记为编译器生成的双取错误 *易挥发*， 允许编译器把**单个存储器访问成多个AC-正如事实在二进制级别，因为它假设该存储器将不会改变。

6相关工作

到目前为止，研究双取分析进行了专注于动态分析，而我们提出了一个静态分析方法。除了在Bochspwn [14，13]已经讨论过的工作，也有一些相关的研究如下。

威廉[37]中使用的类似的方法来Bochspwn分析半虚拟去恶习的后端组件的存储器访问模式。他分析identi网络编39潜在的双重取问题和安全关键后端的COM ponents发现三种新的安全漏洞。其中发现的漏洞并不在源代码中存在，但通过COM的堆垛机优化（参见5.5节的讨论）出台。此外，在源代码中的另一个发现的漏洞通常是不可利用，因为编译器OPTI-mizes的方式的代码，所述第二取指令替换为Fi的值的重用RST取。

双获取竞争条件是非常相似的时间 - 中检查到越时间的使用（TOCTOU）引起的检查条件和使用检查的结果（通过该条件不再之间发生变化的竞争条件）。在TOCTOU的数据不一致通常是由一个争用条件，从不当重新sults同步并发访问共享对象引起的。有共享对象的品种中的任何计算机系统，如网络连接LES [2]，插座[36]和存储器位置[39]，因此，一个TOC**TOU可以在存在二 FF 整个系统erent层。TOCTOU竞争条件往往出现在的**文件系统和众多的AP-proaches [5，9，18，4，27]提出了解决这些问题，但仍然没有通用，安全的方式为应用程序访问网络文件在比赛系统-高速公路。

沃森[36]曾利用上的包装，来自系统调用间位置concur- rency漏洞。他专注于包装的漏洞，这将导致安全问题，如特权esca-特征研和审计绕行。通过识别资源rel-

埃文特访问控制，审计或其他安全FUNC-族体正在跨越信任边界并发访问，他发现从包装的漏洞，并展示了利用与实例技巧。他还分类的越时间的审核，以时间使用和时间中置换到越时间的使用问题，除了时间中检查到越时间的使用问题。然而，他专注于系统调用介入的安全性也可扩展，而非内核，因为我们做。他做的，他是如何发现这些漏洞要么不亲韦迪细节。

杨等人。[39]编目在野外并发攻击通过研究46二 FF erent类型的漏洞和预先的sented自己的特点。他们指出，并发攻击的风险是成正比的易受攻击性窗口的持续时间。此外，他们还发现，以前TOCTOU检测与防御技术太SPECI网络c和无法检测或防止一般CON组货币的攻击。

Coccinelle的[17]，该程序匹配和transfor- mation发动机我们在我们的方法使用，最初焦油geted在Linux驱动程序附带的演变，但现在被广泛用于系统代码的网络nding和FI兴错误。随着Coccinelle的，Nicolas等。[26，25]进行的所有2003年和2011年间发布的Linux版本的研究，经过十多年的Chou等人的作品。[7]，是谁给了第一个在Linux中发现的故障进行深入研究。Nicolas等。指出，那种CON组十年前sidered故障仍然是有意义的，并且仍然存在于新的和现有的网络连接LES。他们还发现，所考虑的各种故障率在驱动程序目录，其中支持Chou等人在下降。

7结论

这项工作提供了第一个（据我们所知的）在Linux内核中双取的静态分析。这是第一种方法能检测双获取漏洞的完整的内核，包括所有驱动程序和所有硬件架构（这是impos-锡布尔赫丁使用动态方法）的网络连接。基于我们的图案 - 基于静态分析，我们分为三个典型SCE-narios中双取容易出现。我们还提供推荐的解决方案，SPECI网络C到我们在研究中发现典型的双取的情况，避免重复抓取错误和漏洞。一个解决方案是用来自动修补双取的错误，这是能自动修补的尺寸检查情况发生的所有发现的漏洞。

凡在Linux，FreeBSD和OpenBSD的内核的已知的动态分析，没有发现双取的错误，我们的静态分析发现，运用实物双取漏洞，网络五个这些都是以前未知的漏洞，以及三个是可利用的双取vulnerabili - 联系。所有报告的bug都已经CON网络Rmed指和

固定成本由维护者。我们的方法已经通过了Coccinelle的团队，目前正在整合到Linux内核补丁审批。

致谢

笔者想真诚地感谢所有的观众重新您对本文时间和专业知识。你有见地的意见帮助我们改进这项工作。这项工作部分是由中国的国家重点研究发展计划（2016YFB0200401）的支持，该方案教育部新世纪优秀人才支持计划，由国家Sci- ENCE基金会（NSF）中国61402492，61402486，

61379146，61472437，并通过实验室预研基金（9140C810106150C81001）。

参考

[1]错误166248 - CAN-2005年至2490年SENDMSG compat的堆叠过溢流。 https://bugzilla.redhat.com/show bug.cgi ? ID = 166248。

[2]乙 ishop， M.，d ilger， M.， 等。 检查在竞争条件网络文件访问。 *计算系统2，* 2（1996），131-152。[3]乙 runel， J.，d oligez， D.，H 安森， RR，L 一堵墙， JL， 和中号 uller， G.一种用于溢流基于程序匹配的基础：我们 - 荷兰国际集团时序逻辑和模型检测。在 *在第36届ACM SIGPLAN-SIGACT学术研讨会编程语言（popl等）的原则论文集（* 2009）。

[4]Ç Al， 十，G Ul， Y.， 和 Ĵ ohnson， R. UNIX环境与开发通过算法复杂度攻击网络文件系统的比赛。在 *在安全和隐私（30 IEEE Sympo- sium* 2009年），第27-41。

[5]Ç 母鸡， H。， 和 w ^ 瓦格纳， D. MOPS：用于检查的软件安全性的基础设施。在 *计算机和通信安全第九届ACM会议论文集（* 2002），页235-244。[6]Ç 母鸡， J.， 和 中号 AC d onald，S.迈向静态和动态分析测试并发程序更好的合作。在 *并行第六届研讨会论文集和分布式SYS-TEMS测试，分析和调试（* 2008），页。8。

[7]Ç 侯， A.，Y 昂， J.，C HELF， B.，H allem， S.， 和 É 文格勒， 操作系统错误D.实证研究。在 *第十八ACMSymposium作业系统原理（SOSP）论文集（* 2001）。

[8]Ç orbet， J.，R ubini， 一种。，和 ķ roah- H artman， G。 *Linux设备驱动程序。* O'Reilly Media公司，2005。[9]Ç owan， C.，B eattie， S.，W 对， C。，和 ķ roah- H artman， G. RaceGuard：从临时文件科幻比赛vulner-能力的内核保护。在 *Use nix安全专题讨论会（* 2001），页165-176。

[10]é 文格勒， D.， 和 一种 shcraft， K. RacerX：电子 FF ective，静态检测的竞争条件和死锁。在 *9幅teenth ACM研讨会作业系统原理论文集（SOSP '03）（* 2003），ACM，第237-252。

[11 1 H ammou， S. 开发的Windows驱动程序：双取竞争条件漏洞，2016年 HTTP：//资源。infosecinstitute.com/exploiting-windows-driversdouble-fetch-race-condition-vulnerability/。

[12 1 H uang， J.， 和 ż 拴， 并发访问异常C.说服力预测。在 *2011年国际对称 - posium对软件测试和分析程序（* 2011），第144-154。

[13：J urczyk， M.， 和 C oldwind， G。 Bochspwn： Identify-通过全系统的存储器存取模式analy- SIS荷兰国际集团0天。黑帽2013，2013。 http://vexillium.org/dl。PHP ? BH2013 Mateusz Jurczyk Gynvael Coldwind.pdf。

[14：J urczyk， M.， 和 C oldwind， G.识别，并通过内存访问模式利用WIN-DOWS核心竞争条件。技术。代表，谷歌研究，2013。 http://research.google.com/酒吧/存档/ 42189.pdf。

[15]ķ asikci， B.，Z amfir， C。，和 C andea， G.数据种族与种族的数据错误：告诉迪 FF 与预示着erence。在 *在体系结构支持第十七届国际学术会议论文集编程语言和操作系统（ASPLOS）*（2012），第185-198。[16]ķ asikci， B.，Z amfir， C。，和 C andea， G. RaceMob：众包的数据争用检测。在 *第二十四届ACM研讨会作业系统原理论文集（* 2013年），第406-422。

[17]大号 一堵墙， J.，L aurie， B.，H 安森， RR，P 阿利克斯， N.， 和 中号 uller， OpenSSL中G.查找错误处理的错误使用Coccinelle的。在 *欧洲可信计算大会（EDCC）（* 2010），第191-196页。[18]大号 熙， K.-S.， 和 C hapin，科幻的SJ检测LE-基于种族CON组扬长避短。 *国际期刊信息安全4，* 1-2（2005），105-119。[19]大号 U， K.，W U， Z.，W 昂， 十，C 母鸡， C。，和 ż 侯， 十RaceChecker：电子 FFI cient identi网络的有害数据争阳离子。在 *2015年第23届的Eu romicro国际会议并行，分布式，以及基于网络的处理（* 2015年），第78-85。

[20]大号 U， S.，P 方舟， S.，S EO， E.， 和 ż 侯， Y.学习从错误：对现实世界的并发错误character-羟基苯乙的综合研究。在 *在体系结构支持第13届国际会议论文集编程语言和Operat- ING系统（ASPLOS）（* 2008），页329-339。

[21]大号 U， S.，P 方舟， S.， 和 ż 侯， Y.查找通过不可序列化的交织测试原子违章错误。 *IEEE交易软件工程38，* 4（2012），844-860。[22]大号 U， S.，T ucek， J.，Q 在， F。，和 ż 侯， Y. AVIO：经由接入交织不变量检测原子违规。在 *ACM SIGARCH计算机系统结构新闻（* 2006年），第二卷。34，第37-48。

[23]大号 ucia， B.，C 结， L.， 和 小号 trauss， K. Colorsafe：用于调试和动态避免多变量原子侵犯架构支持。 *ACM SIGARCH计算机体系结构的消息38，* 3（2010），222-233。[24]中号 C ķ enney， PE 清单：修复双取指针在 hlist进入安全（），2013。 HTTPS：//lists.linuxfoundation。组织/ pipermail /集装箱/ 2013-月/ 031996.html。

[25] p 阿利克斯， N.，T homas， G.，S 啊哈， S.，C alv` ES， C.，L 一堵墙， J.， 和中号 uller， G.故障在Linux中：十年后。在 *在体系结构支持第十六届国际会议论文集编程语言和操作系统（ASPLOS）*（2011年）。[26] p 阿利克斯， N.，T homas， G.，S 啊哈， S.，C 阿尔维斯， 厘米 uller， G。，和大号 一堵墙， J.故障在Linux 2.6中。 *ACM交易的电脑系统（TOCS）32，* 2（2014），4[27] p 艾尔 M.， 和 G 罗斯， TR由用户空间缓存的文件科幻元抵御TOCTTOU种族的应用。在 *在Vir-图阿尔执行环境第八届ACM SIGPLAN / SIGOPS会议的亲ceedings（* 2012），第215-226。

[28] p ratikakis， P.，F 奥斯特， JS， 和 H icks， M. LOCKSMITH：区域动脉灌注化疗蒂卡尔静态竞赛检测C. *ACM交易在计划 - 明语言和系统（TOPLAS）33，* 1（2011），3。29] R yzhyk， L.，C HUBB， P.，K UZ， 一世。，和 H EISER， G.丁哥：驯服去副司机。在 *计算机系统4 ACMEuropean会议论文集（* 2009年），第275-288。

[30] S avage， S.，B urrows， M.，N 埃尔森， G.，S obalvarro， P.，和 一种 nder-
儿子， T.橡皮擦：对多线程程序的动态数据竞争检测器。 *ACM交易的电脑系统（TOC S）15，*
4（1997），391-411。[31] S 恩， K.赛执导的并发程序的随机测试。

ACM SIGPLAN声明43， 6（2008），11-21。[32] S 尔娜， FJ MS08-061：内核模式的
情况下，双取，2008年。
https://blogs.technet.microsoft.com/
SRD / 2008/10/14 / MS08-061-的情况下的最-内核模式双取/。

[33] S 喜 Y.，P 方舟， S.，Y 在， Z.，L U， S.，Z 侯， Y.，C 母鸡， W.，和 ž 恒
W.我用错了去网络nition？：化解：去网络nition使用的用于检测并发和顺序错误IN-变
种。在 *ACM SIGPLAN通知（* 2010），第二卷。45，ACM，第160-174。

[34] S wift， MM，B 艾尔沙德， BN，和 大号 小艾， HM提高商品的操作系统的可靠性。 *ACMT rans。COMPUT。SYST。23，* 1（2005年2月），77-110。[35]垂直 翁荣南， JW，J 哈拉， R.，和 大号 erner， S.继电器：导数百万行代码的静态种族去tection。在 *欧洲软件工程confer- ENCE和软件工程的基础的ACM研讨会SIGSOFT的第六次联合会议的论文集（* 2007），页205-214。

[36] w ^ atson， RN并发环境与开发中的漏洞的系统调用包装。在 *在o首先USENIXWorkshop* FF
ensive Technolo-吉斯（WOOT）（ 2007）。

[37] w ^ ilhelm， F.跟踪特权的存储器访问发现软洁具漏洞。硕士论文，卡尔斯鲁厄研究所'F
乌尔Tech-
nologie，2015年[38] w ^ U， Z.，L U， K.，W 昂， X。，和 ž 侯， 十，协作技术并发错误检测。 *国际杂志并行编程43，* 2（2015），260-285。[39 -] Y 昂， J.，C UI， A.，S tolfo，S.，和 小号 ethumadhavan， S.并发攻击。在 *第四USENIX会议上平行热门话题论文集（* 2012）。

[40】Z 挂， M.，W U， Y.，L U， S.，Q 一世， S.，R 恩， J.，和 ž 恒 W.艾：容忍并发错误的轻量级系统。在 *软件工程的基础的第22届ACM SIGSOFT国际研讨会的亲ceedings（* 2014），ACM，第330-
340。

[41】Z 挂， W.，S 联合国， C。，和 大号 U， S. ConMem：通过电子检测严重CON组货币的错误 FF ECT为本的方针。在 *ASPLOS的架构支持的第十五版的Proceed-英格斯的编程语言和操作系统（ASPLOS*
XV）（ 2010），页179-192。