ISSN 0361-7688，编程和计算机软件，2019年，卷。 45，第7号，第365-371。 ©昴出版公司，2019。

动态检测释放后使用错误免费

S. A. Asryanb，\*，S. S. Gaissaryana，C，E，F，\*\*，SH。 F. Kurmangaleeva，\*\*\*，A. M. Aghabalyand，\*\*\*\*，

N. G. Hovsepyand，\*\*\*\*\*和S. S. Sargsyand，\*\*\*\*\*\*

一个Ivannikov研究所系统编程，俄罗斯科学院，莫斯科，俄罗斯109004 b问题研究所在信息与自动化，科学亚美尼亚国家科学院，

埃里温，0014亚美尼亚

计算数学与控制论，莫斯科国立大学法律系，

莫斯科，俄罗斯119991

埃里温州立大学，埃里温，0025亚美尼亚

物理科学与技术，Dolgoprudnyi，莫斯科州的莫斯科研究所，141700俄罗斯˚F州立大学 - 高等经济学院，莫斯科，俄罗斯101000

\*电子邮件：asryan@ispras.ru

\*\*电子邮件：ssg@ispras.ru

\*\*\*电子邮件：kursh@ispras.ru

\*\*\*\*电子邮件：anna.aghabalyan@ispras.ru

\*\*\*\*\*电子邮件：narekhnh@ispras.ru

\*\*\*\*\*\*电子邮件：sevaksargsyan@ispras.ru

收到2019年2月15日;修订后的2019年2月15日;接受2019年2月15日

描述摘要-A用于检测使用释放后基于程序动态分析错误新颖方法。在存储器不安全的编程语言，如C或C ++，这个类的错误主要发生在程序试图访问已被已释放动态分配的存储器的区域。对于每一个程序的执行路径中，该方法检查分配，回收和访问操作的校正。由于使用了动态分析，错误只能在实际执行的代码的部分中均可发现。符号程序执行与SMT（满足性模理论）解算器的帮助下被使用。这使我们能够生成其中，所述处理的新生成的执行路径的数据。 DOI：10.1134 / S0361768819070028

1.引言

软件经常包含的类型•释放后使用 - （UAF）和错误

•缓冲或堆溢出。

由于软件的很大一部分中极为重要的应用程序时，错误可能会导致严重的后果。有一些工具来帮助解决使用静态[1，2]和动态[3-7]分析这个问题。

静态分析检查代码，但不执行。这种方法的缺点是，没有关于程序的执行（寄存器，程序执行追踪，输入数据等）时的状态的信息是可用的，这导致了大量的假阳性。出于这个原因，静态分析主要用于动态分析之前以显示可以潜在含有缺陷的程序的片段。

该工具在[1]设计用于检测UAF错误执行类似于可用表达式的分析（表达式x + y为可在点p，如果该表达式被计算从进入点到p和X任何路径上的分析描述和y的最后一个这样的计算之后保持不变直至p [8]）。在程序中的所有路径被检查和检查它的使用前的状态对象被定义。如果该条件不被满足，则存储器用途被认为是错误的，并且产生一个消息。

GUEB [2]基于所述程序的二进制代码的检查。分析的过程被分成两个主要阶段。在第一阶段中，堆存取和地址分配操作被跟踪（指针和堆元件之间的对应关系）。所述双{地址，大小}被存储在集alloc\_set和free\_set当存储器被分配和释放，分别。

在第二阶段中，检测UAF错误。使用在程序中的每个点的第一阶段收集到的数据，该工具构造集access\_heap包含所有对{地址，大小}可用在这一点上。如果集合的交集access\_heap和free\_set非空，则一个错误UAF被登记。

空隙顶部（字符输入[4]）{INT CNT = 0;如果（输入[0] ==“B”）CNT ++;如果（输入[1] ==“a”）的CNT ++;如果（输入[2] ==“d”）CNT ++;如果（输入[3] ==“！”）CNT ++;如果（CNT> = 3）中止（）; //错误

}

图1从[10]程序的一个例子。

一种用于动态分析的普及的原因是它的检查在运行时，它可以访问寄存器和存储器内容的程序的能力。该工具雪崩[3]基于动态二进制翻译迭代地分析程序的执行代码。在分析过程中，该工具计算以便自动遍历程序中的所有可达到的路径，并检测其异常终止该程序的输入数据。

在DangNull [4]和FreeSentry [5]中，重点是在检测和该存储器被释放之后，从而防止错误UAF抵消指针动态分配的存储器。这两种工具使用的程序静态仪器。

Undangle [6]也防止UAF错误。该工具标记每个内存分配函数的返回值，并使用被污染数据的分析，跟踪这些标签。当存储器被释放，其中存储器位置与对应的标签相关联的和工具检查检测悬空指针（即，引用一个已释放的存储器位置的指针）。

该工具混乱[7]是基于错误检测的二进制代码，它结合离线的方法和在线接近的节目的符号执行。离线方法依次检查程序的执行路径。在各试验期间，该工具仅包含一个执行路径。一个缺点是，几个路径的公共初始片段在程序的每次运行反复执行。在线方法检查同时所有可能的执行路径，这可能会导致内存不足的情况。

这两者的组合如下方法作品。当达到存储器消耗的限制值，则控制点的创建，和的一些路径检查已被暂停，以及有关的执行，符号执行上下文，和特定的输入数据的当前状态的数据被保存。当资源变得可用（的某些路径检查已完成），则控制点中的一个被恢复（即，执行到这点使用所保存的数据再现）。接着，符号执行上下文被加载，并开始新的路径的分析。这种方法能够避免程序的重复符号执行到控制点。

在本文中，我们考虑基于动态分析和动态检测[9,10]的方法。我们描述了用于检测错误UAF方法，该方法检查使用指针的程序的所有可能的执行路径的正确性。该方法是基于在SAGE [11]所使用的代码覆盖算法，它使用动态分析仪的Triton [10]的基础设施。

在这项工作中，我们修改的Triton，其中显著改善其性能使用的代码覆盖算法，我们加入的程序，输入数据的工作从文件中读取分析的支持;这个功能是不是在海卫的实现支持。

在本文中的第二部分中，我们描述的Triton和在所提出的修饰中使用的代码覆盖算法。在第三部分中，我们将讨论初步实现UAF错误检测及其与动态代码覆盖组合。结果在第四部分介绍。

2.代码覆盖率算法

2.1。代码覆盖率的Triton

在本文中，我们使用的算法最大化Microsoft开发和使用SAGE [11]代码覆盖。该算法在曲通部分实施。它由两个阶段组成：

初始输入数据和为每个程序执行路径gatheringconstraints的选择;

通过求解由聚集在第一阶段约束logicexpressions产生新的输入数据。

考虑在图1中示出的程序的例子。

为了检验这一方案的所有路径，它必须提供与输入字符串坏！为了产生所需的数据，则该算法通过运行与初始输入字符串，它被放置在输入数据的列表的程序启动。在第一次运行后，将约束集<I0≠B，I1≠一个，I2≠d，I3≠！ >，其中I0，I1，I2和I3分别是输入存储器位置[0]，输入[1]，输入[2]，并输入[3]，分别被获得。

在该算法中执行的过程中，这些约束求解，以产生用于在输入数据，其后代数据满足这些约束的列表中的每个元件;所生成的数据被放置在输入数据的列表。该方案是该列表中的每个元素再次运行，算法的工作继续进行。

这个过程继续，直到在输入数据的列表中的所有元素被检查（该算法的伪代码示于图3）。通过与初始输入字符串良好应用该算法，以在图中所示的程序。1，我们得到图2中所示的解决方案的列表。

由于该算法的工作要求程序运行多次，海卫实现节能程序状态的选项。这显著

提高了算法的性能。注意，SAGE算法[11]设计成减小输入数据集的一部分中的Triton没有实现。为此，该工具乘运行在不打开新的执行路径输入数据所分析的程序。在这项工作中，我们添加了新功能的代码覆盖率算法，大大提高了性能。

2.2。在覆盖算法的改进

在最初的实现中SAGE算法[11]中的Triton，程序总是获得从输入数据的每次迭代后的列表中的最后一个元素，而不考虑该程序的基本块的数量使用该元素打开。其结果是，随着影响的代码覆盖率的输入数据一起，那没有打开任何新的路径输入的数据进行处理。

由于算法生成每个输入元素的子孙数据，输入数据的列表的长度显著增加。因此，为了有效地执行该算法，将所生成的输入数据应该优先。

在本文提出的算法的修改，在输入数据的列表中的每个元素被分配一个权重等于由该元素打开基本块的数量。在算法开始时，输入数据被赋予零权重。在算法的第一次迭代中，初始输入数据的权重被计算。

所检查的元件的权重被传递给后代元素（通过求解逻辑方程获得）：在每次迭代之后，所述权重被如下更新。因此，当使用输入数据的层次遍历。该程序的每次运行前，具有最大重量的元素是从输入数据的列表中选择。此显著简化解决方案的列表作为示于图4。

它在该图中看出，添加权重之后进行检查的输入数据的量的两倍几乎降低。这显著提高了算法的性能（在某些测试中，表现了90％，几乎增加）。

的Triton的另一个缺点是只接受在仅输入命令行参数的程序的支持。为了扩展程序的范围

可以分析，我们增加了程序的支持，使用文件作为输入数据的来源。我们还增加了确定将被标记为在分析的过程中象征性的那些输入数据的范围的能力。

在计算上述权重的方法是不是唯一可能的一个，并且在今后的研究，用于确定权重的其它技术将被考虑。

BUG检测

程序的动态分析是基于在其执行过程中的程序进行分析。这使得它可以分析并考虑到具体的执行条件和指针的使用特定值的方案。动态分析的缺点是有一个良好的代码覆盖率的要求。在许多情况下，然而，对于所检测到的错误，输入数据在其上的错误被再现可以生成。该UAF错误是由两个连续的事件造成的：

创建一个悬空指针;

使用悬挂指针存储器访问。

UAF的一个例子示于图5中检查线路3的条件之后，由指针PTR引用的存储器被释放（第4行），然后将控制转移到线12，其中相同的存储器位置被释放再次。

海卫一的错误检测算法

使用该程序的仪器，该算法跟踪存储器分配（malloc的）和释放（免费）的功能。在算法的开始，创建两套（allocSET和freeSET）。它们被用来跟踪程序执行过程中分配，然后释放该内存位置。这些集合的元素是对（地址，大小）。

每个malloc的或免费被调用时的时间，该套allocSET和freeSET是通过添加或删除与对应的地址和所分配的存储器的大小元件更新。当的malloc被调用时，创建一个新元素（address\_2，SIZE\_2）和添加到集合allocSET和来自第二集合中移除，如果它被包括在freeSET（即，两者的地址和大小一致）。

如果只有一个元素匹配的地址，则该地址和大小被附加更新集（如果SIZE\_2 <SIZE\_1，则该元件（address\_2 + SIZE\_2，SIZE\_1之前处理 - SIZE\_2）加入到freeSET当函数自由被称为时，相应的元件被从所述集allocSET到freeSET移动。

当存储器访问指令被执行，则检查指针是否在这两组。一个UAF错误登记在两种情况下：元素在freeSET发现，但不包括在allocSet;在freeSET一个且相同的元件中出现超过一次。该算法的工作示于图6中。

所提出的方法

为了提高错误检测的效率，我们建议上述两种算法结合起来。将合并的方法允许人们发现上发生的，由于在该程序的更深，非平凡零件检查各种执行路径UAF错误。图，其中双解除分配的错误不能没有使用关于代码覆盖信息（由于条件控制转移的存在，如果程序在特定的输入数据运行将只执行）中找到的程序图7示出的例子。

对于图7中所示的示例中，如果程序执行分别到达所述第一和第二编程线21和23，发生UAF错误。所提出的方法使得有可能找到输入数据保证的是，在代码所需的块到达（线20-21和22-23在图7），然后检查UAF错误代码的这一部分。

动态分析方法的比较

表1比较了在本文中与混乱和Triton使用的方法所描述的方法。

结果

所提出的方法是在手动生成的基准包括图中所示的那些测试。 1和7;测试结果示于图8这些结果表明，在手动生成的基准的性能提高了约80％的Triton执行比较。在分析现实生活中的项目的试运行表明，在大多数情况下，象征式的数量是如此之大，海卫的代码覆盖算法不能解决这些方程的所有路径。

为了测试所提出的方法，我们特意注射UAF错误到现实生活中的项目的代码。我们分析了从包gvgen项目

表1的比较结果

缺陷可发生在管线26，并且在线路27中的第二示例。

graphviz的，从封装libjasper运行时碧玉，和从封装giflib gif2rgb。在这些项目中，在不同的注射水平发现的错误。在gvgen的情况下，缺陷在许多相互调用的功能进行了注射（最大喷射深度为三个级别（功能））。注入的代码是一个条件表达式取决于输入

图8.比较结果的分析执行时间。

数据和错误本身的代码。这种情况的满意度导致了错误的发生。

在碧玉和gif2rgb项目，注射只在一个函数由于象征式的复杂性来完成。注入的代码是完全错误代码。我们在包含UAF错误上所提出的方法能够发现漏洞现实生活中的程序也选择特定的代码片段。

结论

描述了一种用于检测存在的由于动态存储器指针的不正确处理的释放后使用-（UAF）的错误的方法。该方法是使用曲拉通基础设施[10]根据[11]中描述的算法和UAF错误检测算法上实现。修改和现有实现的改进使我们能够显著提高分析性能。

资金

这项工作是由俄罗斯基础研究基金会支持，项目编号。 17-01-00600。

参考

杜威，D.，掠夺，B.，和特雷纳，P.，Uncoveringuse释放后条件在编译代码，在PROC。对可用性，可靠性和安全性（ARES），2015年，第90-99第10届国际会议。

费斯特，J.，穆尼耶，L.，和Potet，M.-L.，免费的二进制代码，J. COMPUT后静态检测的用途。病毒黑客技术，2014年，第一卷。 10，没有。 3，第211-217。

伊萨耶夫，I.K.，西多罗夫，D.V.，格拉西莫夫，A.Yu.和Ermakov说，M.K.，雪崩：对错误的自动检测程序中动态分析的应用程序使用网络套接字，特鲁迪研究所。 SIST。程序。拉斯。 AKAD科学，2011，第一卷。 21，第55-70。

李，B.，宋，章，张，Y.，和王，T.，Preventinguse释放后，与悬摆指针无效，在PROC。的网络和分布式系统安全研讨会，2015年。

https://www.ndss-symposium.org/ndss2015/ndss2015-programme/preventing-use-after-free-danglingpointers-Noneification/

尤南，Y.，FreeSentry：防止使用-afterfree漏洞由于悬摆指针，在PROC。网络和分布式系统安全研讨会，2015年https://www.ndss-symposium.org/ndss2015/ndss2015-programme/freesentry-protecting-against-useafter-free-vulnerabilities-due-dangling-pointers/

卡瓦列罗，J.，GRIECO，G.，栗子，M.，和纳帕，A.，Undangle：在useafter - 自由和无漏洞双悬摆指针，在PROC的早期检测。的2012国际研讨会上的软件测试和分析，2012年，第133-143页。

桑基尔加丹茶，Thanassis Avgerinos，亚历山大Rebert，A.，和Brumley，D.，发动MAYHEM上的二进制码，在PROC。的2012 IEEE研讨会安全和隐私，2012年，第380-394。

阿霍，A.，乌尔曼，J.，塞西，R.，和林，M. S.，编译原理，技术和工具，波士顿：艾迪生韦斯利，2006年，第2版。

针 - 动态二进制指令Tool.https：//software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool

海卫 - 动态二进制分析Framework.https：//triton.quarkslab.com/

戈德弗鲁瓦，P.，莱，M. Y.，和莫尔纳，D.自动白盒模糊测试，在PROC。对网络和分布式系统安全（NDSS'2008），2008年，第151-166会议。

德莫拉，L。和Bjørner，N.，Z3：一个高效SMTsolver，在PROC。对工具和算法系统，2008年，第337-340建设与分析第14届国际会议。

由A. Klimontovich翻译