基于脆弱点特征导向的软件安全测试

笔记本： 论文
创建时间： 2019/11/1 10:00 **更新时间：** 2019/11/1 19:05
作者： Mrs.Lee

书名	基于脆弱点特征导向的软件安全测试	作者	欧阳永基
出版社	清华大学学报	阅读日期	2019.11.1

摘要（关注基于域收敛的路径遍历策略算法格式）

为克服模糊测试方法具有盲目性和覆盖率不高的缺点，缓解当前符号执行方法所面临的空间爆炸问题，该文提出一种基于脆弱点特征导向的软件安全测试方法。该方法结合模糊测试和符号执行方法的特点，针对缓冲区溢出，精确分析了具备该脆弱点特征的代码，并以此为测试目标，力图提高测试针对性；通过域收敛路径遍历策略生成新测试数据进行测试。

正文

为了促进深层次软件脆弱点的发掘，本文在分析已有缓冲区出错原因的基础上，总结缓冲区溢出的特征，获取可疑的敏感代码空间［13］，以此为待测目标，设计了一种基于域收敛的可疑代码遍历策略，力求提高软件路径遍历的针对性。

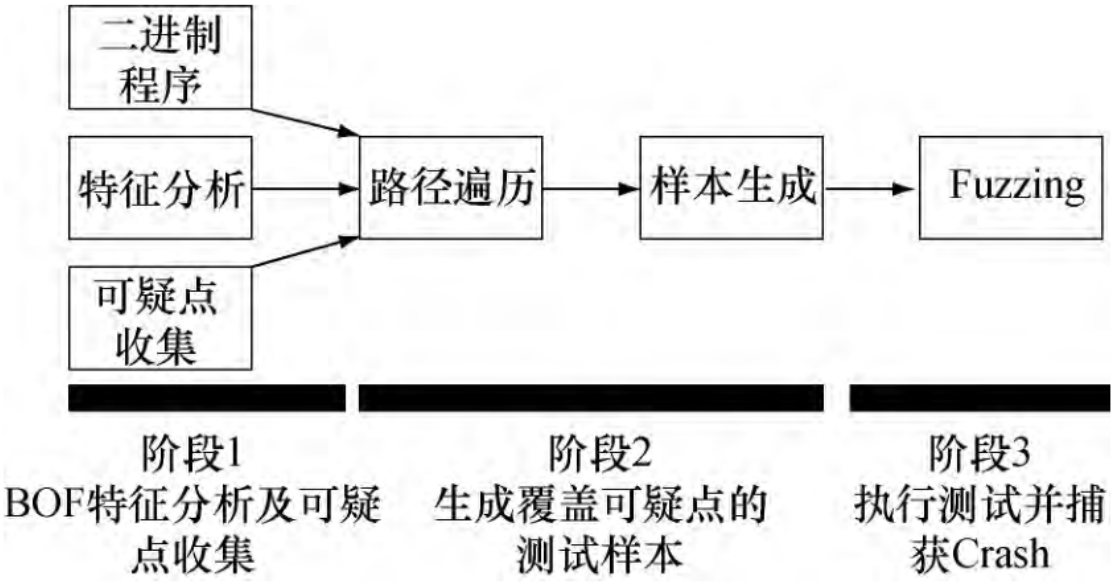

图 1 测试框架

一 脆弱点特征分析与可疑点收集

对于无边界检查的 BOF 脆弱点的特征，可以归纳为如图 4 所示的模型：一是它发生在一个循环结构中；二是在该循环中存在读写内存操作；三是控制读写的程序指针能被程序输入污染，即能被用户控制。若满足这 3 个条件，则可以控制 Pointer 访问非法的地址，从而发生错误。

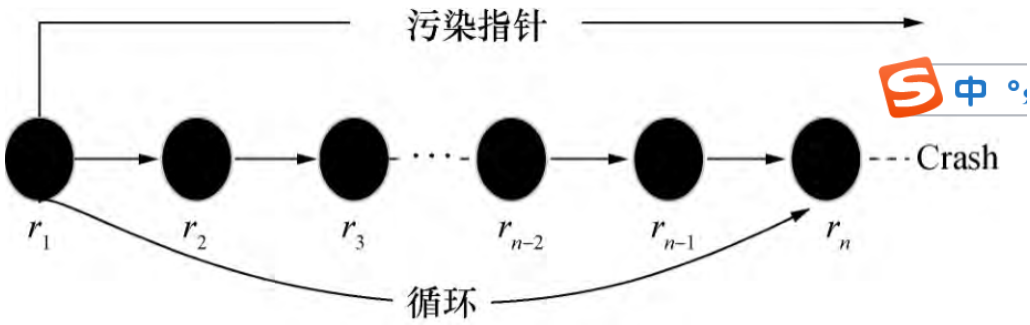

图 4　无边界检查 BOF 特征模型

对于有边界检查 B O F 脆弱点，虽然进行了边界检查，但是由于检查方式存在缺陷，使得可以利用索引指针访问到数组外的数据，进而产生错误。该类 B O F 具体特征如图 6 所示：一是与有边界 B O F 类似，它发生在数组的循环访问操作中；二是它存在指针越界处理指令；三是越界处理指令的守护条件能被污染。

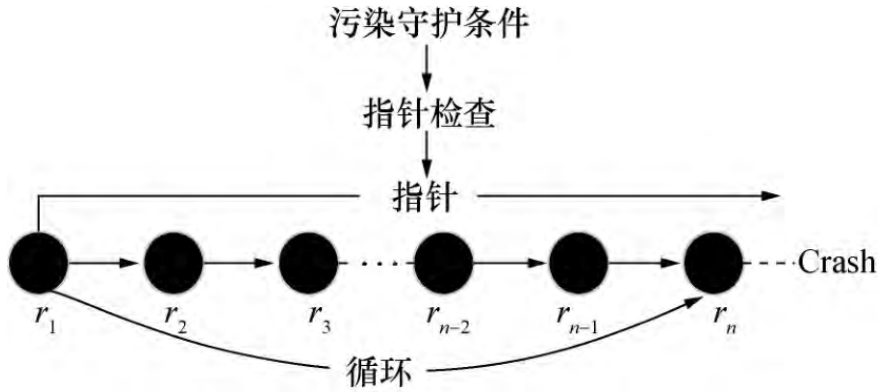

图 6　有边界检查 BOF 特征模型

二 基于域收敛的路径遍历策略

一个二进制程序由许多基本块组成，敏感点包含在这些代码块或几个基本块的组成中。如果将一个代码块或多个代码块看成一个域，那么对目标点的遍历便等同于对域的遍历。本文选择一个针对测试程序的样本集T，从中取出测试用例 t ∈ T ，假若它没有进入可疑区域，则收集它执行的路径条件，并对路径条件进行操作分析，生成新的测试用例，使得下次执行路径能更接近目标域，然后按照这种方式逐步迭代，直到生成的测试用例能够覆盖目标，具体如图 8 所示。

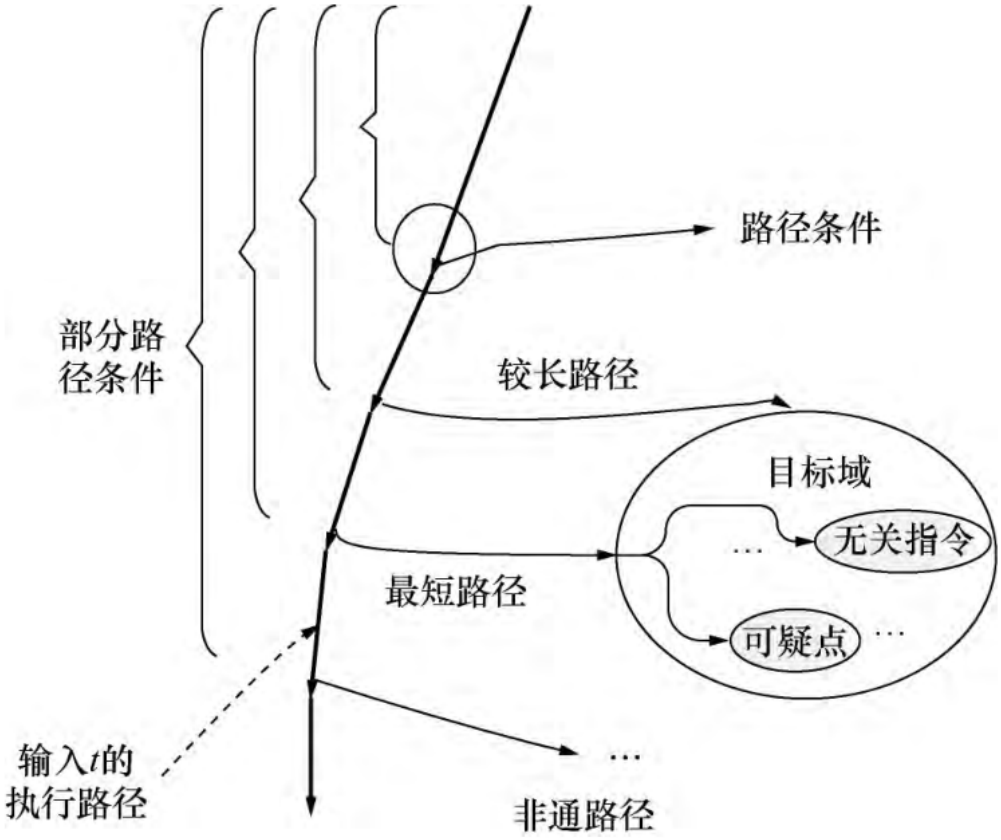

图 8　基于域收敛的路径遍历

图 8 中间的粗线条表示一条没有命中的执行路径，本文把没有到达目标点的程序路径称为部分路径条件（partial path conditions，PPC），然后利用PPC与目标域的距离来指导样本的优先构造顺序，从而去逼近或覆盖可疑指针。

假设一条执行路径没有进入程序指定的危险区域（danger），它对应一个测试输入t,t的路径条件是ψ1∧ψ2∧…∧ψm。如果寻找从 t 逼近danger的输入，则可以从 t 的任何一个分支进行逼近。如果一个输入从分支的 k 处开始，那么它必须满足：PPC=ψ1∧ψ2∧…∧ψk-1∧ψk．也就是说，这个新的输入，满足原始输入 t 的前（k-1）个分支条件，但是不满足第 k 个分支条件。因而可以设计一个满足上式的新输入，让程序执行不同的输入，使得程序可能执行到目标处。另外，一般来说，若 PPC与可疑目标点的距离较小，执行路径也较少，相应的测试效率也会更高。则本文取反的路径条件应该满足下面的条件：

$$PPC \wedge \min|danger - PPC|.$$

如上所述，算法设计思路如下：首先，根据控制流图，计算路径上每个节点到特定目标点距离。在定义距离权重时，分为 2 种情况进行定义：基本块与基本块边的权重为 1 ，辅助的边（如节点中的函数调用）的权重设为 0 ；若命中包含目标指令的基本块，却没有命中需要的指令时，在基本块内指令与指令间的距离也定义为 1 。其次，利用 Dijkstra算法 [15] 计算PPC与目标点的最短距离，指导条件取反，生成新的样本。例如，若程序 P 的一个路径分支为 b ，对应的路径条件为 ψ ，dist（ψ）为b 到特定目标的最短路径若dist（ψ）=0 ，则表示已进入特定目标；若dist（ψ）≠0 ，则利用算法2 ，根据 PPC的路径条件 φi（ψ1∧ψ2∧…∧ψi∧ψi+1），选取合适的条件 ψi 进行取反，根据约束表达式，通过求解约束表达式构造新的测试用例，不断逼近测试目标，继而遍历所有的可疑目标点，具体如图 9 所示。

==========

算法：基于域收敛的路径遍历算法

输入： P ， t ，unused（未使用的 PPC），S （所有的 PPC ）

输出：程序 P 测试集 Tp

==========

```
1  Set unused, S, T_P = φ
2  procedure Excute (P, T)
3     execute P with input T
4     let f = (ψ1 ∧ ψ2 ∧ ⋯ ∧ ψm) be the pt-condition
5     for all i from 1 to m do
6     φi =def= (ψ1 ∧ ψ2 ∧ ⋯ ∧ ψi ∧ ψi+1)
7     if dist(ψi+1) ≠ ∞ and dist(ψi+1) > 0 and φi ∉ S
8        S ∪= φi , unused ∪= φi
9     end if
10    end for
11 end procedure
12 procedure Produce(unused, danger)
13    While unused ≠ ∅ and not timeout do
14       Select φ ∈ unused with min-dist to danger
15       Remove φ from unused
16       Let φ = (ψ1 ∧ ψ2 ∧ ⋯ ∧ ψk−1 ∧ ψk)
17       Construct θ = (ψ1 ∧ ⋯ ∧ ψk−1 ∧ ¬ψk)
18       solve θ
19       if θ is satisfied then
20          let tθ be an input that satisfies θ
21        return tθ
22       else return null
23       end if
24    end while
25 end procedure
26 for all t ∈ T_P do
27    Execute(P, t)
28    t_new = Produce(unused, danger)
29    if t_new ≠ null then
30       T_P ∪= t_new
31    end if
32    Remove t from T_P
33 end for
34 return T_P
```

图 9　基于域收敛的路径遍历算法