Nautilus_4_CTF

故事大概是这样子的 某 rosayxy 想搞一个面向 ctf 的 fuzzer，然后从上学期中一直想到了现在 找到了一个不错的solution~

思路

- 我们首先需要知道自己需要什么，为什么我们需要一个 fuzzer： 说到底 我们用的手动找洞通常主要还是静态的，然后动态找洞和静态找洞相比还是挺有优势的~ 而且你可能遇到除了C语言以外的逆向，C++ 和 rust 啥的，笔者逆向比较菜，所以单逆向可能就要花费比较久。。（逃）
- 特征：和正儿八经的工程项目相比，ctf 遇到的二进制文件通常没有源码（或者源码不全），需要黑盒工具，然后文件结构，输入模式会比较结构化和简单，所以这种情况为什么用AFL++ 的qemu mode 效果没有那么好，AFL++ 的种子变异模式没有那么有结构性，而且大多是二进制层面的变异，很难保持基本的输入结构 就像是菜单堆题这种的，所以就 emm 了
- 笔者上学期有两门专业课是形式语言与自动机和编译原理，然后就注意到**大部分我们ctf题中遇到的输入格式都可以用EBNF文法或者稍微强一点的形式表示！**加上之前了解过面向编译器的 fuzzer，这些都需要输入比较有结构性，所以就想到了找一个grammar-based fuzzer
- 但是 和正常面向编译器或者解释器的fuzzer相比，我们还有一个需求，就是我们的漏洞一般都在堆上。而且漏洞模式一般还是在how2heap 这样子的范围内。所以 **可以挂个sanitizer上去**
- 然后就到了更具体的找轮子环节！
    - 首先想到的是自己实现一个 EBNF 的parser（或者找个轮子）然后替换seed generation 和 mutation 的环节
    - 现有的fuzzer 一般都是mutation-based 思考换成**generation-based fuzzer**，找到了一个轮子叫做peach（https://github.com/MozillaSecurity/peach） 但是看上去就非常古老。。而且它的文法用xml格式，有点繁琐，懒得搓（逃
    - 然后和师兄讨论发现**LibAFL**，（特别感谢黄院士www）读文档的时候发现有很好的模块化，然后也接入了nautilus 的功能，去读了nautilus的原文，（https://www.ndss-symposium.org/ndss-paper/nautilus-fishing-for-deep-bugs-with-grammars/） 发现它generator 和mutator 设计很神奇，就是应该有的样子（不太清楚该怎么夸了hhh）但是论文里说它不支持blackbox fuzzing，那咱就用LibAFL 接一个fuzzer就行了叭hhh (快乐.jpg)
    - 2月6号看了将近一天的文档和源码，基本找全了接口，开始欢乐的splice它例子里的 baby_fuzzer_nautilus 和 qemu_coverage，然后顺手更新了一下rust-nightly,没想到build 的时候就 gg 了，看了一下是最新的 nightly 在一个库 AHash上面有feature stdsimd 的问题（https://github.com/tkaitchuck/aHash/issues/200）。。ggggg
    - 顺手点到了 nautilus 的 github 仓库上，发现人家**已经支持 AFL-Qemu 了?!**而且AFL++ 的QASAN的使用是用环境变量设置的，好家伙，咱直接就可以冲一波hh

轮子

- Nautilus（version 2.0）
    - Nautilus 是个非常经典的grammar-based fuzzer,最开始的用法是 fuzz 编译器这种，然后文法描述是用**EBNF 范式和 python 的 lambda 表达式~** 所以理论上它确实可以描述图灵机级别的输入结

构，但是还是希望开发者升级到可以用函数描述（逃
    - 题外话，Nautilus 真的很帅，它是把输入的结构生成一棵AST,之后变异也是在树的基础上去splice,havoc 啥的

- AFL++ + QASAN：
    - AFL++ 的 Qemu mode 非常经典，之前见过队里同是 pwn 方向的学长用过~
    - nautilus 用 AFL++ 的 Qemu 是用`cargo run /path/to/AFLplusplus/afl-qemu-trace -- ./test_bin @@` 的命令，而我们知道，QASAN 的使用是设环境变量，所以就冲就得了

开fuzz!

- 我们切到 rust 的 stable version，然后开写EBNF
- em 稍微吐槽一下，虽然说理论上是有图灵机的描述水平，但是实测可能实战中**就比 EBNF 强一点点**，主要瓶颈是lambda表达式表达能力非常有限。。
- 比如说是经典的先读一个size,再配空间，再`read(0,buf,size)` 这种情况，实测不如枚举0x10-0x500范围内的`str(16*i)+'a'*16*i`这种方法。。
- 测了6个文件，发现检测 UAF 和堆溢出还是比较给力的，但是像是house of orange 这样覆盖top chunk 的情况需要稍微用点心思，把输入字符增加 `str(16*i)+'\x00'*16*i` 这种情况。
- **off-by-one还是比较难 fuzz 出来的**，原因是感觉 QASAN sanitize 的精度可能没那么高，看了原论文，也感觉有点语焉不详。。
- nautilus**吞吐率很高**。测的时候一般都能到500-800+exec/sec，还是开了QASAN的情况下，就很不错(嗷如果测的 exec 一直比较低新路径还比较少，可能是off-by-one 这种情况或者你grammar写错了)，测的6个文件，有4个可以在5分钟之内出crash,还有一个是off-by-one, 一个 rust 语言的越界访问

发现的限制条件

1. 像上文所说，发现off-by-one 不太行。。
2. 需要程序可以正常终止，所以如果遇到有 abort 的情况要注意避开！不然会false positive ~ 还有那些无限死循环就不退出的就不太行。。
3. （ref 师兄）有些 ctf 题会要求 mmap 特定地址上的内存，Qemu 这个可能搞不定

不算总结

- 搜索 QASAN 的过程中发现了有友友有相同的想法的博客，是 https://kiprey.github.io/2021/09/protobuf_ctf_fuzz/，想法是protobuf + afl++ 的qemu-mode + QASAN ~ (杭哥tql!) 对文法的表述不是 EBNF 范式hhh
- 其实本来想手造一个轮子的 甚至名字都起好了www (大家可以现不急用 PrimAFL 这个名字么www，以后有机会想用www)
- 要新年了~大家龙年快乐喵 ~