170627 逆向-GeekerDoll

原创

奈沙夜影 于 2017-06-28 03:29:55 发布 925 收藏
分类专栏： CTF

本文链接：https://blog.csdn.net/whklhhhh/article/details/73825636

1625-5 王子昂 总结《2017年6月27日》【连续第268天总结】

A.全国大学生安全竞赛之GeekerDoll

B.之前试了很多方法，IDA中找不到源码，因为是64位的所以OD也跟不了，拖到linux虚拟机中用IDA单步跟，被计时器的消息丝

最后换了writeup看，发现了之前提到过但是没给连接的github上的haskell的反编译的程序

感谢大佬写wp的时候顺手把工具也给了让小白如我也能跟着走

hsdecomp-master: https://github.com/gereeter/hsdecomp

从github上把hsdecomp-master拖下来，装到python上

由于没有nmake和cmake又报了半天错，最后直接手动下了capstone才成功运行

装好可以import以后突然发现可以直接调用runner.py运行OTZ，虽然还是报错了

调试没明白怎么回事，以为是很复杂的结构问题……

找了同学一起后马上发现问题了：

metadata.py的循环中，str给了第二个参数'ascii'，是多余的；本身会报的错全部被try捕获然后pass掉了OTZ

对其使用encode即可

show.py中`settings.binary[parsed_offset]`本身就是一个char类型了，不需要再使用chr转换，以及循环条件把!=0改为!=''即可

都是类型匹配错误，估计大神写完没运行尝试就发布了

反编译出源码以后就很简单了：

```
('Main_main_closure', '=', '>>= $fMonadIO
    getArgs
    (\\sz_info_arg_0 ->
        case /= $fEqInt (length sz_info_arg_0) (I# 1) of
            <tag 1> -> case == ($fEq[] $fEqChar) (rkl_info (!! sz_info_arg_0 (I# 0))) (unpackCString# "bk_vefuhfuhfuha1n4shaqcz\x00\x00\x00\x00\x00\x00\x00\x00Nope
                False -> $ putStrLn (unpackCString# "Nope\x00\x00\x00\x00Congratz\x00\x00\x00\x00\x00\x00\x00\x00Usage:"),
                True -> putStrLn (unpackCString# "Congratz\x00\x00\x00\x00\x00\x00\x00\x00Usage:"),
            sPh_info_case_tag_DEFAULT_arg_0@_DEFAULT -> putStrLn (unpackCString# "Usage:")
    )')
()
('rkl_info', '=', '\\rkl_info_arg_0 ->
    case rkl_info_arg_0 of
        <tag 1> -> [],
        sA0_info_case_tag_DEFAULT_arg_0@_DEFAULT -> case sA0_info_case_tag_DEFAULT_arg_1 of
            <tag 1> -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7030896 of
                False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7030912 of
                    False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7030864 of
                        False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7030880 of
                            False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7030960 of
                                False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7030976 of
                                    False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7030928 of
                                        False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7030944 of
                                            False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7030848 of
                                                False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7030992 of
                                                    False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7031056 of
                                                        False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7031072 of
                                                            False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7031088 of
                                                                False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7031104 of
                                                                    False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7031008 of
                                                                        False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7031024 of
                                                                            False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7031200 of
                                                                                False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7031264 of
                                                                                    False -> case == $fEqChar sA0_info_case_tag_DEFAULT_arg_0 loc_7031296 of
```


虽然没看过haskell，不过大概可以猜出来loc_是变量，也就是说一堆case实质上就是循环判断试图变换的过程

将最下面的变量值替换到上面的case中，然后上下对应即可发现替换表

在头部是调用下面函数的地方，发现字符串“bk_vefuhfuhfuha1n4shaqcz“，下面True为Congratz，False为Nope，那么很明显这就是比对字符串了

也就是说输入的字符串进行替换后得到“bk_vefuhfuhfuha1n4shaqcz“即可

按替换表反做，得到flag：

flag{pwnpwnpwn_1e4rn_sh}

C.明日计划

CSICN-拯救世界（安卓脱壳）