

MIPS架构的CTF逆向题--SUCTFbabyre题目writeup

原创

iqiqiya 于 2018-10-18 15:35:14 发布 2522 收藏 6

分类专栏: [我的逆向之路](#) [我的CTF之路](#) [我的CTF进阶之路](#) 文章标签: [MIPS架构的CTF逆向题--SUCTFbabyre题目writeup](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/xiangshangbashaonian/article/details/83146678>

版权



[我的逆向之路](#) 同时被 3 个专栏收录

108 篇文章 10 订阅

订阅专栏



[我的CTF之路](#)

92 篇文章 5 订阅

订阅专栏

[我的CTF进阶之路](#)

108 篇文章 18 订阅

订阅专栏

一, 这道题刚拿到的时候我是直接载入IDA 结果分析的一团糟

比赛结束之后 看了好多师傅wp 才知道mips架构的专用工具怎么用, 比如retdec,jeb-mips

开始是打算用retdec的, 结果在线版的<https://retdec.com/decompilation/>不能用了 感觉windows下不好装 有时间补坑

就选择了jeb-mips(试用版)

扯一点怎样下载 就是<https://www.pnfsoftware.com/jeb/demomips>这个链接 填写好个人信息 就会发下载地址到邮箱

```
iqiqiya@521:~/Desktop$ file babyre
babyre: ELF 32-bit MSB executable, MIPS, MIPS-II version 1, dynamically linked
(uses shared libs), for GNU/Linux 2.6.26, BuildID[sha1]=8a245bbb837d4679d7c20df7
ed3cbe1c24b86399, stripped
```

二, 如果想要在linux里面运行 就必须安装qemu了 我记得好像是直接 `sudo apt-get install qemu` 就好了

正常的话直接终端执行 `qemu-mips babyre`

但是这个文件是32bit 小端序 那么就得用 `qemu-mips babyre` 可是提示我文件损坏。。

```
iqiqiya@521:~/Desktop$ qemu-mipsel ./babyre
./babyre: Invalid ELF image for this architecture
```

三, 既然运行不了 就直接载入jeb-mips算了

可以看到关键字字符串flag is 等等 双击任意一个进入

babyre
 babyre
 metadata
 mips image

过滤器: 输入 "Enter" 来确认

mips image/层级

Name	Address	Size
• _libc_csu_fini	4013C0h	8h
• _libc_csu_init	4013C8h	B0
• _libc_start_m	4014E0h	10
• _ptr_close	400640h	10
• _ptr_malloc	400600h	10
• _ptr_memset	400630h	10
• _ptr_open	400620h	10
• _ptr_printf	4005F0h	10
• _ptr_puts	400610h	10
• _ptr_read	4005E0h	10
• finalizer_0	401500h	38
• initializer_0	400544h	68
• main	400EDCh	4E
• start	400650h	Ch
• sub_4005C0	4005C0h	20
• sub_40065C	40065Ch	54
• sub_4006B0	4006B0h	9C

Address	Name	Value	Comment
40003Ah	a4_4_4	4@4@4	
400154h	a_lib_ld_so_1	/lib/ld.so.1	
400421h	a_libc_csu_init	_libc_csu_init	
400431h	a_libc_csu_fini	_libc_csu_fini	
400441h	a_libc_start_ma	_libc_start_main	
400453h	a_gmon_start__	_gmon_start__	
400462h	aLibc_so_6	libc.so.6	
40046Ch	a_DYNAMIC_LINKIN	_DYNAMIC_LINKING	
40047Dh	a_RLD_MAP	_RLD_MAP	
400487h	aMemset	memset	
40048Eh	aOpen	open	
400493h	aPrintf	printf	
40049Ah	aClose	close	
4004A0h	aRead	read	
4004A5h	aPuts	puts	
4004AAh	aMalloc	malloc	
4004B1h	a_IO_stdin_used	_IO_stdin_used	
4004C0h	aGLIBC_2_0	GLIBC_2.0	
401550h	aUsage__s_flag_	usage: %s flag.txt	
401564h	aWrong_	wrong!	
40156Ch	aFlag_is__s_	flag is : %s	

描述 十六进制格式 Assembly Graph **字符串** 类型 导入 导出 Referenced Met

日志 Terminal

Relocation: MIPS: Only R_MIPS_REL32 relocation types are supported for non-obje

四，接着按大写X 查看交叉引用 可以看到都在main函数 双击任意一个

再按tab键反编译 可以看到sub_400780()就是关键 但是试用版是不支持反编译它的

根据师傅的博客 就是一个替换了base64编码表的加密函数

下面那个if判断可以抠出来要解密的eQ4y46+VufZzdFNFDx0zudsa+yY0+J2m


```

f read .plt
f printf .plt
f malloc .plt
f puts .plt
f open .plt
f memset .plt
f close .plt
f start .text
f sub_40065C .text
f sub_4006E0 .text
f sub_40074C .text
f sub_400780 .text
f main .text
f __libc_csu_fini .text
f __libc_csu_init .text
f sub_401480 .text
f __libc_start_main .MIPS
f _term_proc .fini
f sub_401520 .fini
f __iarp_read .exte
f __iarp_printf .exte
f __iarp_malloc .exte
f __iarp_puts .exte
f __iarp_open .exte

```

```

• text:00400B4C
• text:00400B50
• text:00400B54
• text:00400B58
• text:00400B5C
• text:00400B60
• text:00400B64
• text:00400B68
• text:00400B70
• text:00400B74
• text:00400B78
• text:00400B7C
• text:00400B80
• text:00400B84
• text:00400B88
• text:00400B8C
• text:00400B90
• text:00400B94
• text:00400B98
• text:00400B9C
• text:00400BA0
• text:00400BA4

```

```

addiu $v0, 0x3A
li $v1, "c"
sb $v1, 0($v0)
lw $v0, 0x30+var_10($fp)
addiu $v0, 0x3B
li $v1, "v"
sb $v1, 0($v0)
lw $v0, 0x30+var_10($fp)
addiu $v0, 0x3C
li $v1, "3"
sb $v1, 0($v0)
lw $v0, 0x30+var_10($fp)
addiu $v0, 0x3D
li $v1, "m"
sb $v1, 0($v0)
lw $v0, 0x30+var_10($fp)
addiu $v0, 0x3E
li $v1, "8"
sb $v1, 0($v0)
lw $v0, 0x30+var_10($fp)
addiu $v0, 0x3F
li $v1, "U"
sb $v1, 0($v0)

```

<https://blog.csdn.net/wangshangbashaonian>

最后写个脚本 本来自己写了有 但是发现有师傅<https://wangyx-max.github.io/MIPS%E5%90%88%E9%9B%86/>有更简单的

```

import string
import base64
my_base64table = "R9Ly6NoJvsIPnWhETyTHe4Sd1+MbGujaZpk102wKCr7/0Dg5zXAFqQfxBicV3m8U"
std_base64table = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
s = "eQ4y46+VuFZzdFNFDx0zudsA+yY0+J2m"
s = s.translate(string.maketrans(my_base64table, std_base64table))
print base64.b64decode(s)
#SUCTF{wh0_1s_{0ur_d41dy}

```

参考链接:

<https://wangyx-max.github.io/MIPS%E5%90%88%E9%9B%86/>

<https://github.com/Mem2019/Mem2019.github.io/tree/master/writeups/suctf2018#babyre>

<https://www.cnblogs.com/nww-570/p/8948371.html>