HGAME-Week4-WriteUp

啊，hgame结束了，四周过的真快啊（并不，这周想做web的，但是好多没学的，不知道怎么下手🥺，服务器也过期了，而Re都是C，我借着点C基础转战Re，还好学长提点终于 ~~混~~ 做完了。

Re

1. Secret

> It's a secret!

emmm，这道题好坑啊，杀了我虚拟机好多次QAQ，导致我后面熟练的点击恢复快照，然后看其他的页面等待。~~(WGestures快速切换挺爽的)~~

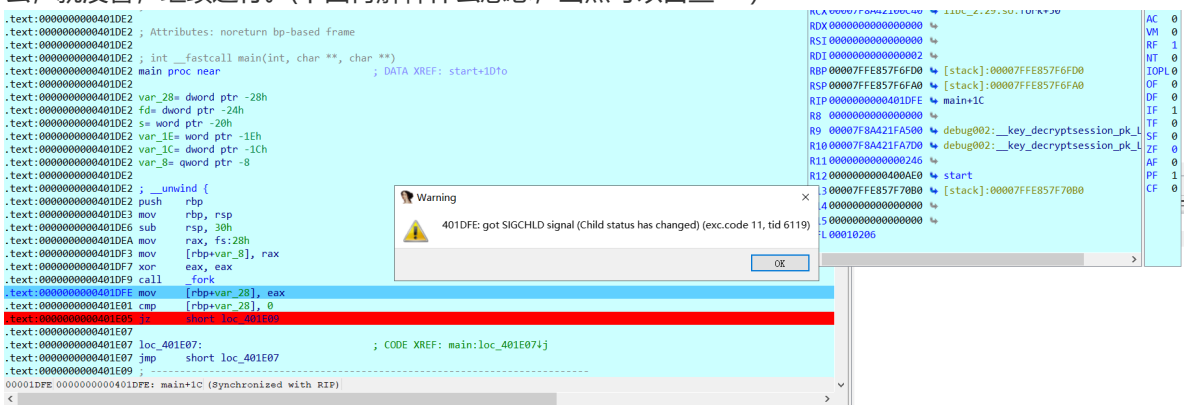
嗯，下面慢慢分析，会把整个做题的思路全理一下（应该不会有什么截图XD）。~~以显示我真的菜~~

一开始先运行，然后搜索语句，到函数里看看，看半天看不出什么，就开始动态调试。

首先我是直接在main函数那里进行调试，毫无疑问秒退，一开始以为我IDA远程调试坏了，扔gdb在main那里直接下断点调试也秒退，说明肯定有反调试。

接下来找入口点，欸，看这个start长得挺别致的，就决定是你了，breakpoint！嗯，接下来一路F8，在哪里终止就再来亿遍F7进入。（我感觉我的Fn键寿命大减）

从start进去的话，会发现有个跳回去再执行一遍就会退出程序的地方，手动改下ZF，让它继续顺利进行，接下来会到main函数里面，call _fork后会立马弹窗，但我pass掉好像就过了，查了下也没查出什么，就没管，继续运行。(下面再解释什么意思，当然可以自查XD)



> 象征性的放张图

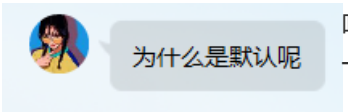
在图中我下断点的地方，还需要改下ZF，不然就跳进下面那个无限循环中了（当然跳进去了改RIP出来）

然后就是连接，读取服务器的内容存入本地的指定区域上，这个就是真正的核心代码，加密的本钱，坑爹的典型（雾，智慧的结晶，我们稍后分析。

稍后到了，接着走就到了核心代码区，先获得进程号，然后发送信号。

可我每次都在这里一发送信号就程序终止，然后虚拟机也没了（黑脸。然后我去查sigqueue函数，也没查出什么东西，就去问幼稚园学长。然后学长让我多了解一下信号相关的东西，我就去学习了一下。

嗯，当时查到的表上都是非实时信号的，上面图里的SIGCHID就代表着子进程退出，fork创建了子进程后，跑完了，就退出了（幼稚园如是说），但关于实时信号我还是毫无头绪，去问问学长，实时信号的默认行为都是杀死进程，然后学长回复：



哎呀，妈呀，天哪（感叹三连，我是被什么蒙蔽了心灵，怎么会没想到这一点，开始看各个函数时发现sigaction呀，赶紧回过去仔细看了下：

```
.text:00000000004015D2  push    rbp
...//省略自己去看XD
.text:00000000004017BB  retn
```

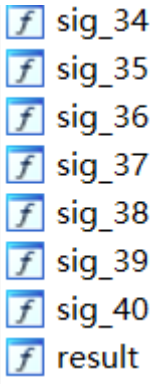
这里将实时信号的默认行为都定义好了。

那么接下来就研究核心代码到底干了些什么就行了。（这里核心代码只能看汇编，反正我f5没反应

后面我就在read之后，服务器读取到的数据到本地后，把所有数据弄下来，然而IDA不知道怎么全选，于是**手动**选择（全部）复制（真的拖了好久，共1477580行，导致我的记事本都无法调整窗口大小，一调就卡。sublime就香一些，为什么想到弄下来呢，因为~~照应开头~~后面的sigqueue太多了，一调错就崩。

对了这里还得插句话。一开始我已sigqueue就崩，f7进入后是在syscall那里崩的，也就是一发送信号和数据就坏掉，自己想不通，问学长，学长提示我去看它发给谁了。不是发给改进程吗？？我开始还转了一下16进制呀。这回我仔细转了一下getppid得到的（**注意，这里是ppid**）16进制，不是当前进程的进程号，我一开始一直看成了getpid，那么就明了了，因为我IDA远程调试的，父进程不是改被调试的程序，是调试被调试程序的程序（有点绕，实际就是linux_server64），所以发信号发错一个了，马匹拍给牛看，还终止掉我的虚拟机（黑脸。这里有个小细节，如果ps -a的话，只会看到linux_server64和该程序，若ps -ef可以看到开始fork出的子程序。

好像插了不止一句（嘿嘿，我把所有数据弄下来后，就~~慢慢看~~硬肝呗，此时还剩下一天不到，幼稚园学长告诉我时间充足，还举了大佬7小时搞定的栗子（扑通跪下了TwT.



我把函数重新命名，发送什么信号执行对应哪个函数的行为。

在读取数据前，执行了34号1次，35号4次，我视为对数据进行初始化，然后读取长为8个字节的输入，这里参数fd是0，也就是是标准输入，所以read可以读取输入的数据。读完之后，就36号处理一下，目的是将数据放在某处待用，也就是放置输入的数据。

接着进入一个循环，依次执行37，38，39，而这里是我很头疼的地方，一堆位运算，想当初我协会的位运算作业还是爆破的（笑，~~当时还看不懂youzhiyuan的ctf写法去问了学长~~），我想着我一步步调，观察数据总行吧。

跳到下面判断处（笑容渐渐消失，循环36次，还得每次修改getppid得到的进程号，这反调试绝了。幼稚园学长好坑啊。无奈了，硬搞加密算法得了，想着我C语言复现一个应该不难，那么首先要搞清楚各个变量里的数据。然而不知道为什么我read到的数据36处理后和前面34，35和后面37，38，39等的数据都不见了（后面知道要选择yes而不是no），连变量都没摸清楚我怎么复现啊，气的把键盘砸了（并没有，都没带键盘）。而且后面我调试，服务器读下的数据IDA没有把它变成正常的汇编代码，导致执行不下去，或者干脆就是16进制数据，uc也无效。无果，夜也深了，锁屏睡觉。（中途挣扎起床再搞一次还是不行0-0

早起问学长（学长睡得早），学长给了引导，让我查查魔数。说起魔数就让我想起前面有个函数里赋了一堆很奇怪的值，当时查了知道这是sha加密用到的数（然而和解题无关），Wiki上对魔数解释的很清楚了，好像也没有什么出奇的地方，那么突破口在哪呢?

猛地一下想到后面34和35初始化里携带了一些奇怪的数据。因为可以携带数据的用了sigqueue，没有携带数据的37，38，39，40，41（41我命名为了result)用了kill，这么反差明显，果断搜索，一搜，34的**9E3779B9h**就自爆了~~（狼人自爆，直接进入黑夜）~~，啊这一个个网页明明白白显示了这个加密算法，TEA，啊不，XXTEA，啊呸，XTEA，（数据没显示哪一个算法，因为都用了，具体对照代码可以发现是XTEA)

当时一行行对照后，妈呀，真的是他是他就是它。

然后学长给了我肯定的答复，还说很标准的，没改什么东西（也就还是改了）

仔细对比了一下发现，主要就是交换加密的顺序，还有最后存放的顺序，下面上代码（网上找的改了一下）：

```
#include<stdio.h>
#include <stdint.h>

/* take 64 bits of data in v[0] and v[1] and 128 bits of key[0] - key[3] */

void encipher(unsigned int num_rounds, uint32_t v[2], uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0=v[0], v1=v[1], sum=0, delta=0x9E3779B9;
    for (i=0; i < num_rounds; i++) {
        v1 += (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[sum & 3]);
        sum += delta;
        v0 += (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[(sum>>11) & 3]);
    }
    v[0]=v0; v[1]=v1;
}

void decipher(unsigned int num_rounds, uint32_t v[2], uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0=v[0], v1=v[1], delta=0x9E3779B9, sum=delta*num_rounds;
    for (i=0; i < num_rounds; i++) {
        v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[(sum>>11) & 3]);
        sum -= delta;
        v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[sum & 3]);
    }
    v[0]=v0; v[1]=v1;
}

int main(){
    int v[2]={0x665E78C77,0x0605E304},k[4]=
{0x42655F29,0x9E822EFC,0x0DA278C92,0x4E355A62};
    decipher(36,v,k);
    printf("%02x %02x",v[0],v[1]);
}
```

改了后，加密的部分就和这个题目基本一样了，解密也很简单了。key就是35存进去的4个数，看汇编可以看出顺序没变化，密文从result函数里跳过去就行，上面这个脚本略糙（数据是最后两个的），得手动输入和手动对照解密，我当时赶着激动劲手撸出来flag了。

最后还有个坑点，存好的密文还有个问题，你得逆着搞。你看原本按顺序的话原数据是：04E30506 778CE765，对比代码的数据就明白处理方式了。

而且因为35的处理，它将明文4个一分成两组，本来在前面的放到后面了，本来在后面的放到前面了，不过对最终读取没太大影响（倒着读就行），就是密文放置时得注意一下就行

小结：这是我真正意义手撸出的有一点难度的题了QwQ，对于一个寒假刚开始看《汇编语言》的，当初汇编的培训作业还没写出来的我真的进步挺大的orz。感谢幼稚园学长的引导。

2. easyVM

> easyVM

这道题我一开始是非预期解的。我要是说我是猜出来的，然后就没了会不会被打死XD，真的巨好猜啊，两个断点，随便输一下就发现加密后的和密文前面一截一致，多试几下就出来了（大雾

为了不被打死（求生欲旺盛，我决定还是再回头研究一下

首先，它叫vm，那么肯定是有理由的，多半和虚拟机有点关系，关系是什么呢？我也不是很清楚，查了下相关的资料，感觉上就是自定义指令。。

翻到幼稚园学长的网站（感谢学长又打开了网站）：

> 看了十多分钟别人对VM的讲解，我大概知道了什么是VM，其实就是一种小型的解释器，将所进行的操作编程 OperationCode，然后程序运行的时候用解释器去解释OperationCode。那么还是需要硬头皮去理解每个Bytecode所代表的含义。

很明显，我进入前面我猜的时候认为的加密函数，里面就是典型的switch结构，每个case下的子函数就是进行操作。

那么首先搞清除数据的去向，一开始，将我输入的数据放在***[rsp+460h+var_440]***（前面两个函数可视为printf和scanf），也就是对应栈上的***Stack[00007DEC]:000000134B50FE00***，后面有这个操作

```
lea     rdx, [rsp+460h+var_440]     //rdx值为0x134B50FA00，其上存着0x134B50FE00
mov     [rbp+360h+var_260], 0Ch
lea     rcx, [rbp+360h+var_60]      //注意到一堆mov操作中有两个取地址操作
                                    //rcx值为0x134B50FDE0,其上存着
0501080501020500，不知道是                                    //否有用，先记录下来，
输入的值在这个地址后没多远
```

将输入的数据的地址扔到rdx寄存器中，接着一路进入我所谓的“加密函数”，进入之后先创建堆栈，看不太懂，，接着switch，更看不懂了orz，好像有些子函数都干着一样的事。（其实只是函数一样而已=-=）

暂时看不懂我就动态调试跟过去看看吧。第一个跟过去的case（5）下的子函数将输入的内容移到一块区域上，然后跳回去进行下一次判断，再接着的case（2）下可以看到把前面case存好的区域进行操作了，哦不，好像只是移位，继续跟，有换了个case（1）进行操作，好像还是数据的移动，还是没有操作，再跟，好像回到最开始的case（5）了（挠头，这次没看见我输入的数据了，接着换了一个新的case（8），w没看懂它干了啥。接着又到了case（1），嗯，这次把输入的数据弄出来了，感觉要有动作了，嗯，重复了前面case1做的事情QwQ。继续follow，又到了case(5),好像这里并没有操作我输入的数据，接着到了case（2），呀过了这里到了新case了，case（E），嗯移入数据 了，这里没有子函数欸。。。。好像只是做了个比较？？又回到case（5），又到新的case（C），又没了。。。但后面到case（4），这里调用的子函数移入了输入的数据（这个子函数和前面caseC的子函数一样），到case（3)了，在调用函数前就导入了输入的数据，咦，case3后还是case（3)?好像进行了一样的操作后，进入了case（6），这个case6调用了两个子函数（好像总共就2个子函数XD），按我的理解这两个子函数的作用就好像解释器一样？将数据转到自己设定的相当于寄存器的地方，所以显示的就是数据搬移，接着到case（5）

嗯，我感觉我可能知道怎么回事了，其他的指令都是混淆的（雾，真正的操作都还不在这里，观察一下有些长一点的case，那些应该才是加密方式。所以经过观察，case(A),case(7)一套操作，艾玛，大意了，真正的黑手在rbx，在case（7）已经直接替换了。并没有直接用输入的数据操作，绕了个弯，处理好了直接替换了。

（果然是在难为我，当成盲盒处理多香啊，我还能腾出时间去追个番，我炮姐更新都没看）

我过会就来继续。（咕

写在最后：

持续4周的hgame结束了，没想到四周就这样过去了，也没想到我这四周能学这么多东西：python，php，js，正则，汇编等等。（当然还只是个入门）这放在以前对我是很难想象的，一个长假期我要拖到最后一天完成作业，假期前踌躇满志，假期后咸鱼一条。我也比较怕这种历史再次上演，我在寒假开始之后的日历上添了很多待办事项，全天通知hgame，第一周隔一两天就有通知，就是怕我自己懈怠下去。后面发现是多余的，因为真的 ~~土头啊~~ 可以一直做的。

在做的时候，看历年学长们的wp，基本都是确定了方向做的，但我一直在web和re徘徊，week1在web，week2在Re，week3又做web，week4做的Re，我本是想选web的，但好像很多的基础知识我都没学，导致看见一个题目都不知道考什么，而Re虽难，但我有C语言的基础呀，所以可能导致了我摇摇摆摆一直没确定方向，而且我比较菜orz，寒假前python，php等语言看都没看，正则也不懂，汇编就看了《汇编语言》开头几章，我前两周基本每天是睡4-6个小时，偶尔还通个宵，后面熬不住了，还是收了点，不然我怎么可能做到进步这么大的（自己深知进步多大）。谢谢学长们给了一个很好的学习交流的机会，希望线下赛能好好签到（逃