**s**



**实验报告**



**题目： 缓冲区溢出攻击实验**

**班 级： 2023211311**

**学 号： 2023211198**

**姓 名： 杜昊阳**

**学 院： 计算机学院**

**2024年 11 月 30 日**

**2**

一、实验目的

1. C语言程序的机器级表示。

2. 掌握GDB调试器的用法。

3. C编译器生成的x86-64机器代码，理解不同控制结构生成的基本指令模式，过程的实现。

4. 掌握两种缓冲区攻击方法，进一步理解软件漏洞的危害。

1. 实验环境
2. 服务器：10.112.61.68（关机时间：2024年12月5日00：00）
3. Linux
4. Objdump命令反汇编
5. GDB调试工具
6. 积分榜：

9-12班：<http://10.120.11.13:19330/scoreboard>

13-21班：[http://10.120.11.13:19340/scoreboard](http://10.120.11.13:19330/scoreboard)

说明：由于服务器及OS的差异，存在如下问题：

ctarget的第2、3关，在攻击成功后，显示Failed信息如下：

Valid solution for level 2/3 with target ctarget

Ouch!: You caused a segmentation fault!

Better luck next time

FAILED

**请同学们关注记分牌信息，得分后表示攻击成功**

三、实验内容

登录bupt1服务器，在home目录下可以找到一个targetn.tar文件，解压后得到如下文件：

README.txt；

ctarget；

rtarget；

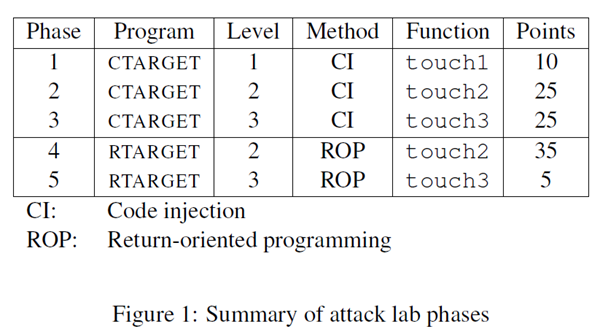
cookie.txt；

farm.c；

hex2raw。

ctarget和rtarget运行时从标准输入读入字符串，这两个程序都存在缓冲区溢出漏洞。通过代码注入的方法实现对ctarget程序的攻击，共有3关，输入一个特定字符串，可成功调用touch1，或touch2，或touch3就通关，并向计分服务器提交得分信息；通过ROP方法实现对rtarget程序的攻击，共有2关，在指定区域找到所需要的小工具，进行拼接完成指定功能，再输入一个特定字符串，实现成功调用touch2或touch3就通关，并向计分服务器提交得分信息；否则失败，但不扣分。因此，本实验需要通过反汇编和逆向工程对ctraget和rtarget执行文件进行分析，找到保存返回地址在堆栈中的位置以及所需要的小工具机器码。实验的具体内容见实验说明，尤其需要认真阅读各阶段的Some Advice提示。

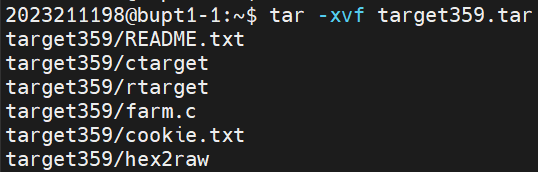
本实验包含了5个阶段（或关卡），难度逐级递增。各阶段分数如下所示：



四、实验步骤及实验分析

## （一）准备工作

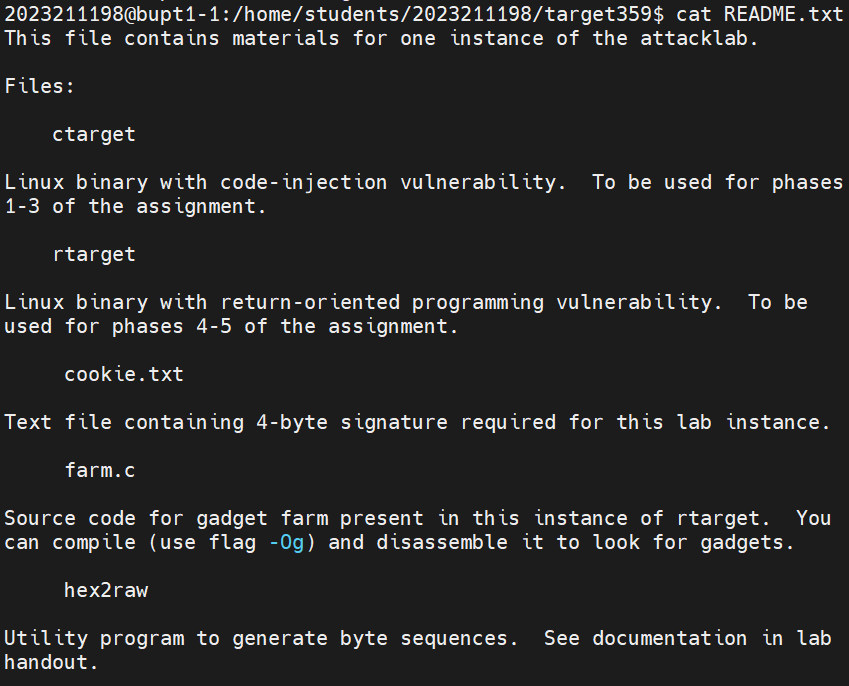
1.解压文件，得到六个文件



2.打开target359文件夹

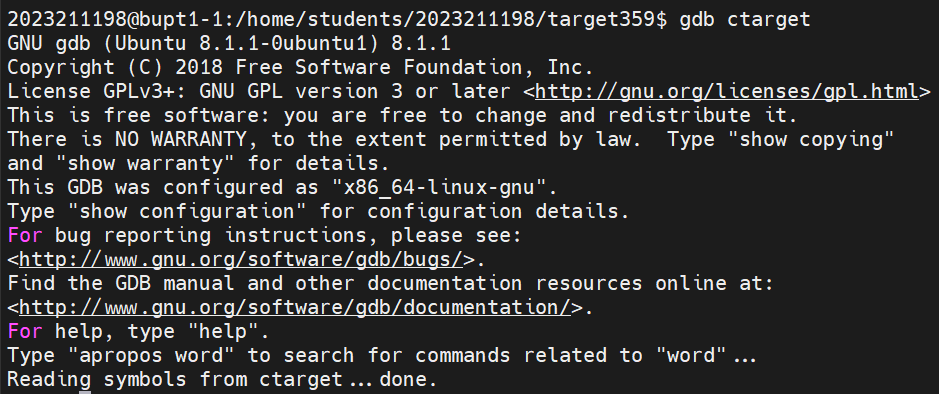


3.查看README.txt，确认各个文件用途

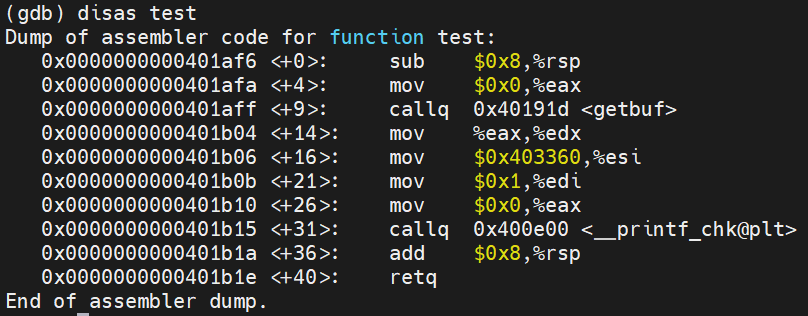


## （二）阶段一：phase1

1.启动gdb调试ctarget



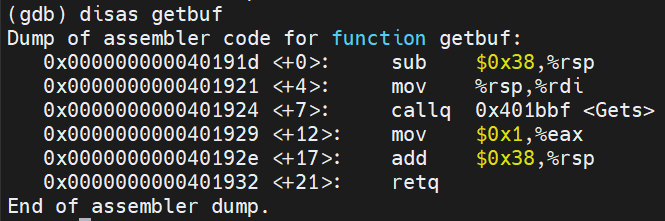
2.输入disas test，查看汇编代码



发现调用getbuf函数



3.输入disas getbuf，查看汇编代码

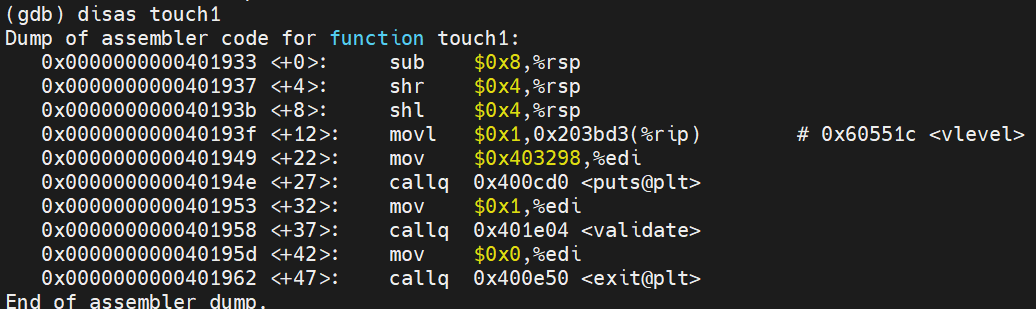


发现栈指针减去了0x38



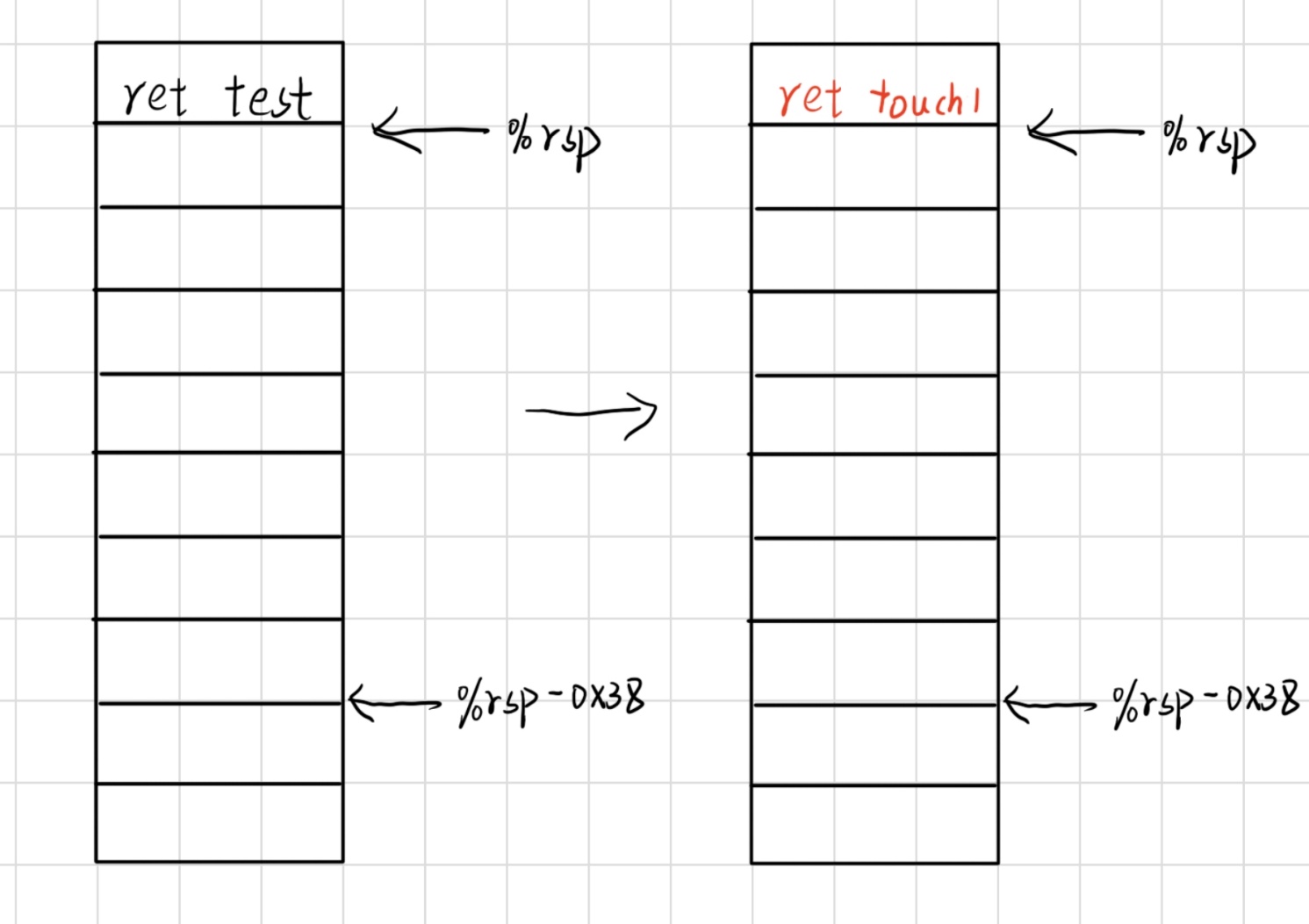
可得缓冲区大小为56字节。

4.输入disas touch1，查看汇编代码



未发现传入参数，说明成功调用touch1函数就算成功

5.栈分析



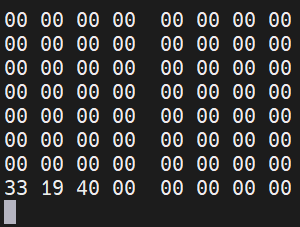
6.退出gdb



7.建立attack1文件



内容如下：



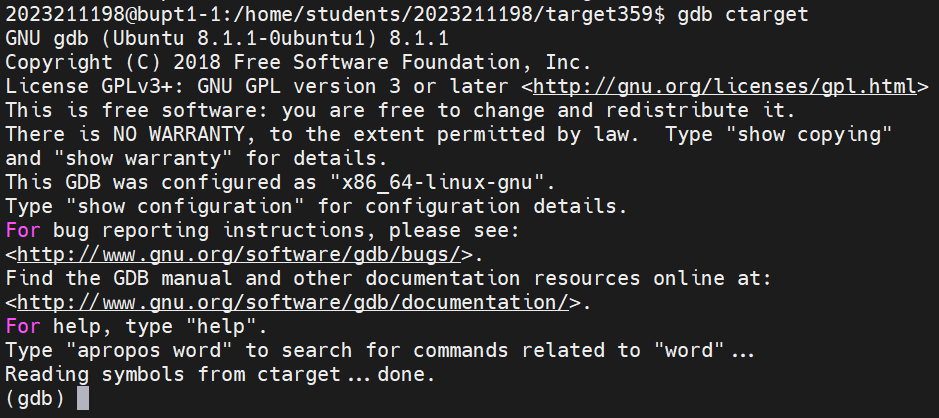
保存并退出



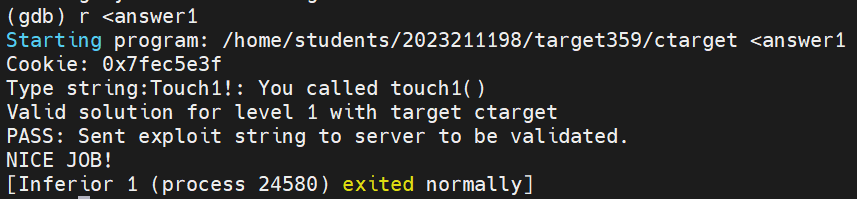
8.将attack1通过hex2raw文件转化为answer1



9.启动gdb调试ctarget

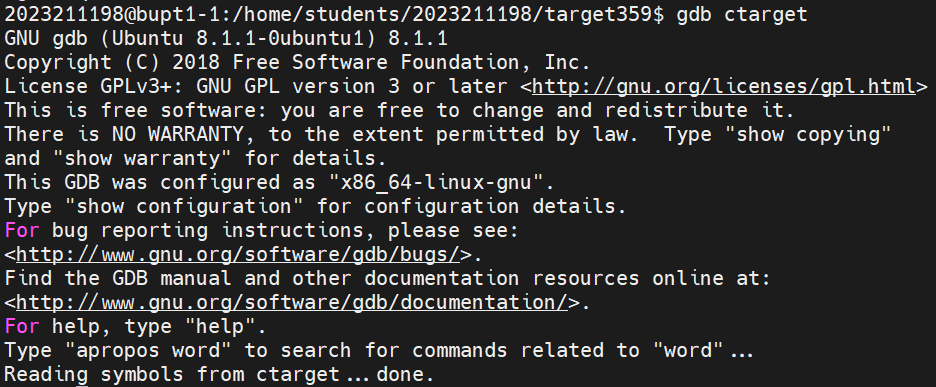


10.用answer1运行，成功调用touch1，阶段一成功

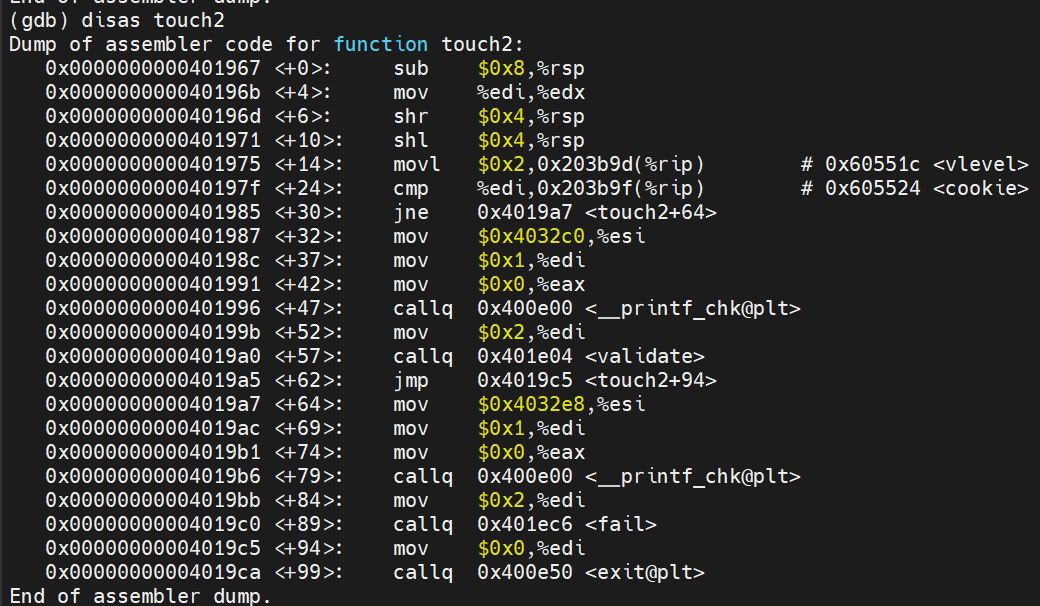


## （三）阶段二：phase2

1. 启动gdb调试ctarget



2.输入disas touch2，查看汇编代码

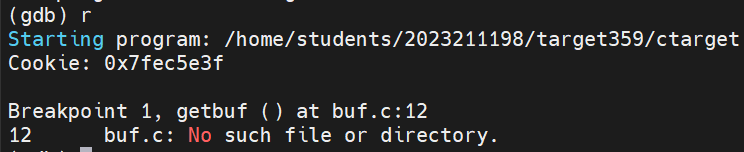


分析可知touch2传入一个参数与cookie比较，如果相等则成功

3.在getbuf上设置断点



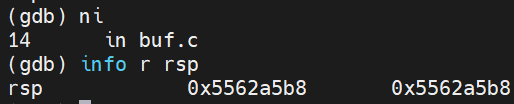
4.运行至断点



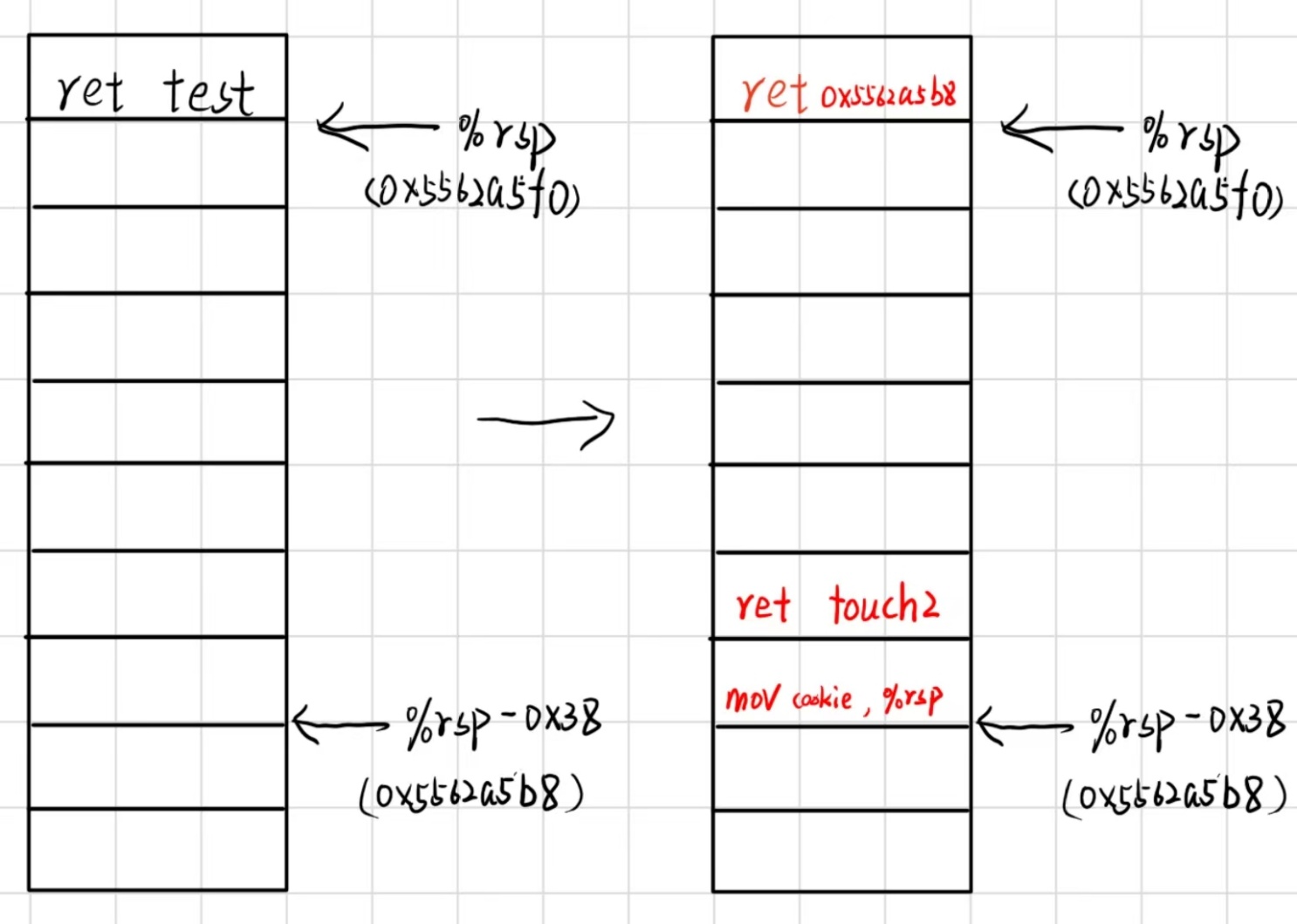
5.查看rsp指向的位置



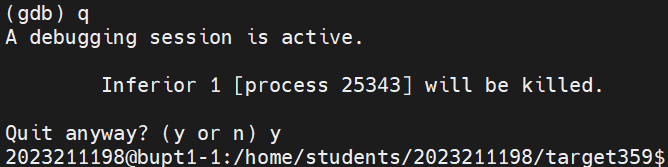
6.单步执行后查看rsp新位置



7.栈分析



8.退出gdb



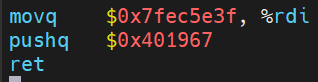
9.查看cookie的值



10.新建p2.s文件



编写汇编代码：



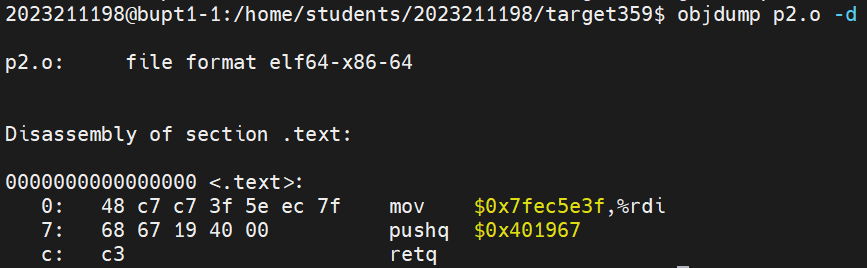
保存并退出



11.将p2.s编译为p2.o文件



12.通过objdump查看p2.o反汇编文件，找到十六进制表示

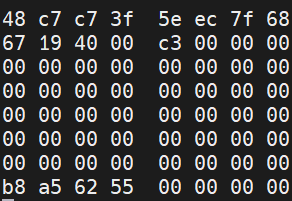


可知执行指令的十六进制表示为48 c7 c7 3f 5e ec 7f 68 67 19 40 00 c3

13.建立attack2文件



内容如下：



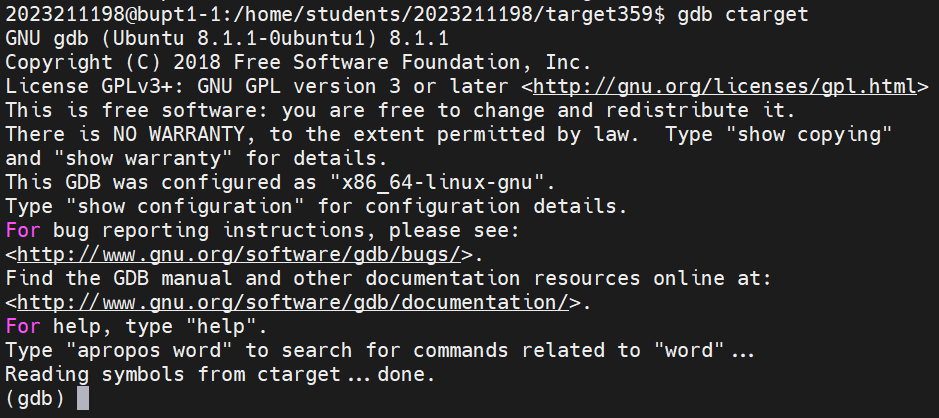
保存并退出



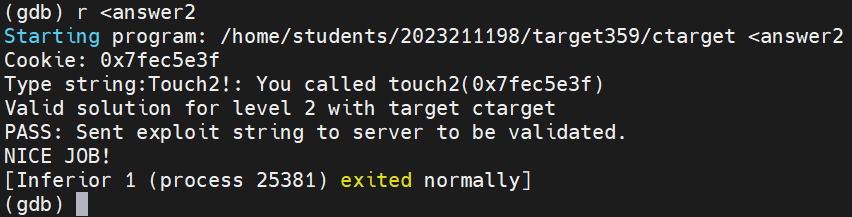
14.将attack2通过hex2raw文件转化为answer2



15.启动gdb调试ctarget

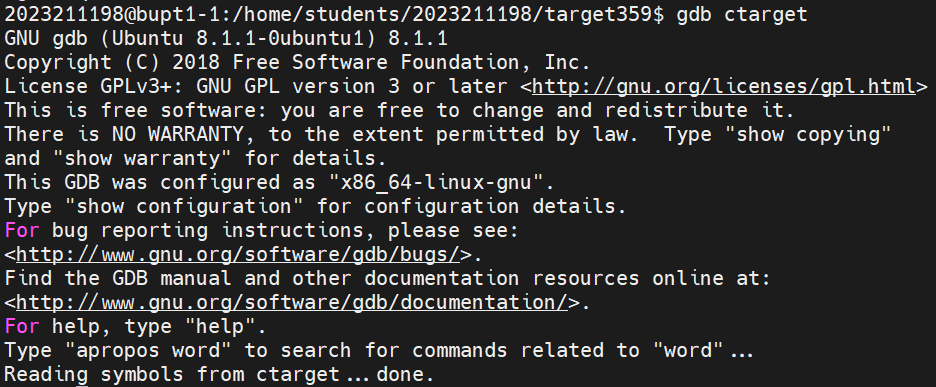


16.用answer2运行，成功调用touch2，并正确传入cookie，阶段二成功

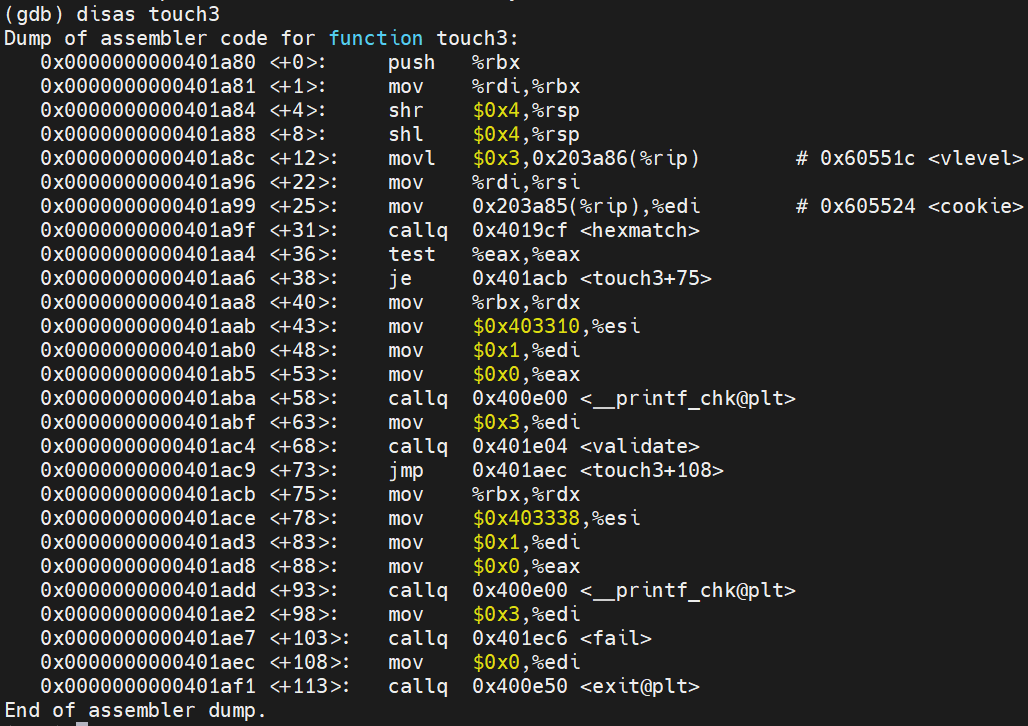


## （四）阶段三：phase3

1. 启动gdb调试ctarget



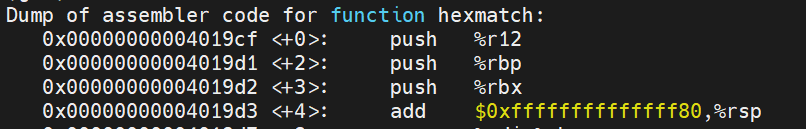
2.输入disas touch3，查看汇编代码



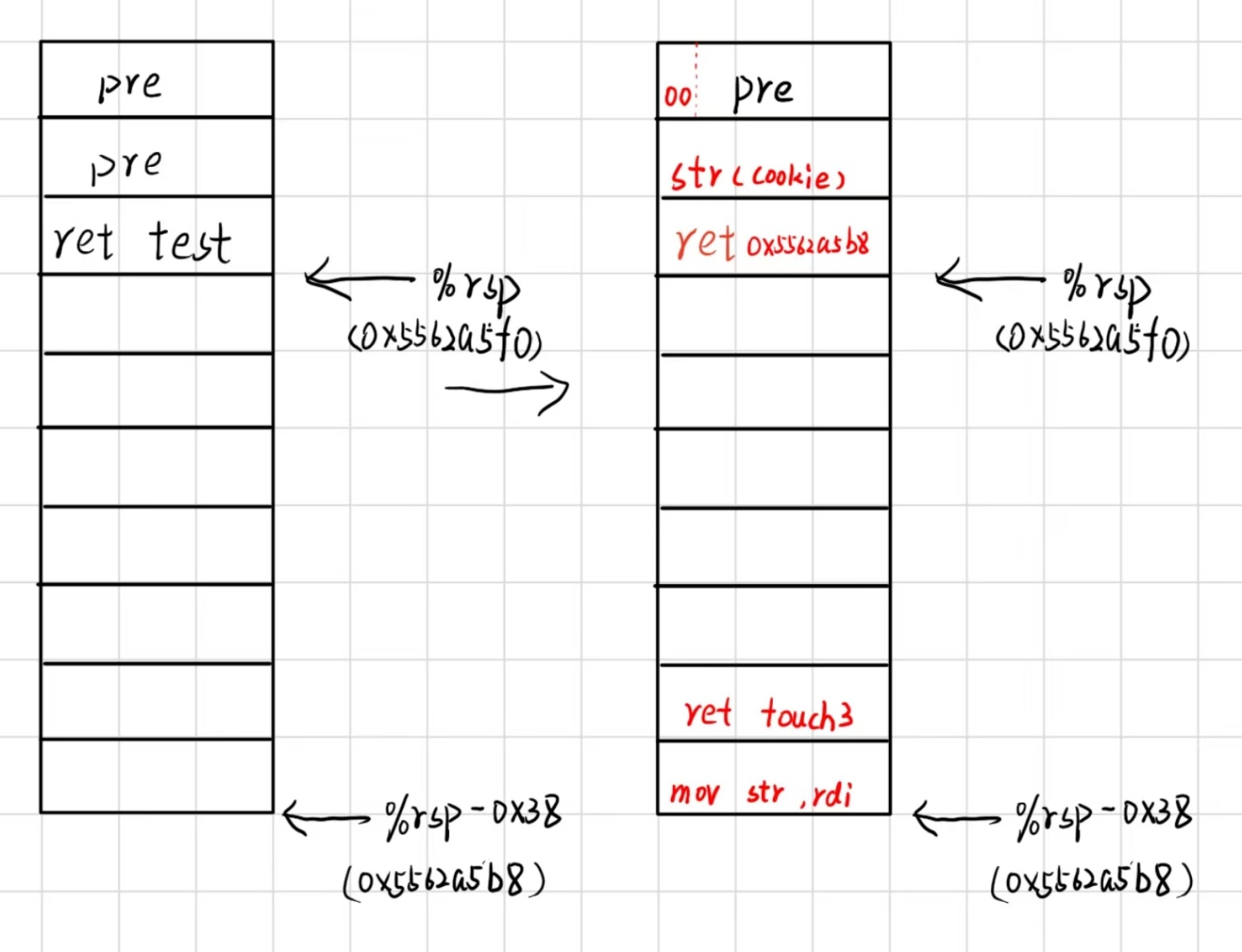
分析可知touch3传入一个字符串与cookie的字符串比较，如果相等则成功

同时发现调用hexmatch函数

3.输入disas hexmatch，发现调用starcmp函数，可能造成栈的覆盖



4.栈分析



因为调用touch3时，test并未返回，所以将cookie存在test函数的栈帧

5. 退出gdb

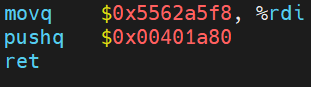


6.查ASCII表，得到cookie的ASCII码十六进制表示为：37 66 65 63 35 65 33 66 00

7. 新建p3.s文件



编写汇编代码：



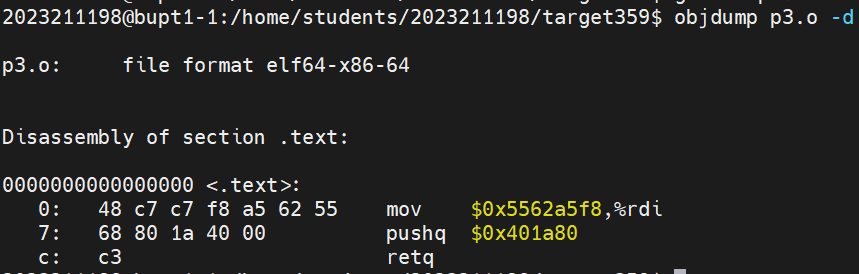
保存并退出



8.将p3.s编译为p3.o文件



9.通过objdump查看p3.o反汇编文件，找到十六进制表示

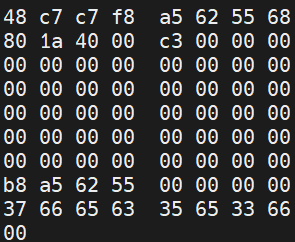


可知执行指令的十六进制表示为48 c7 c7 f8 a5 62 55 68 80 1a 40 00 c3

10.建立attack3文件



内容如下：



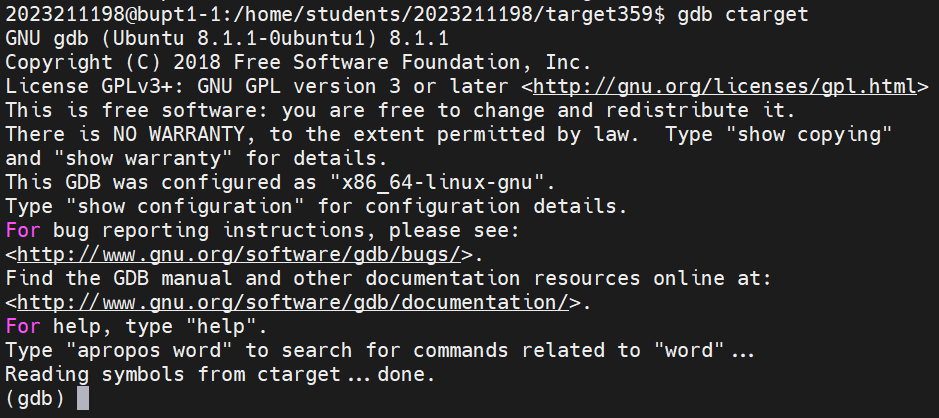
保存并退出



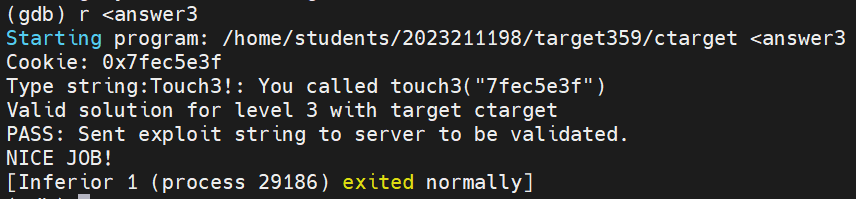
11.将attack2通过hex2raw文件转化为answer2



12.启动gdb调试ctarget

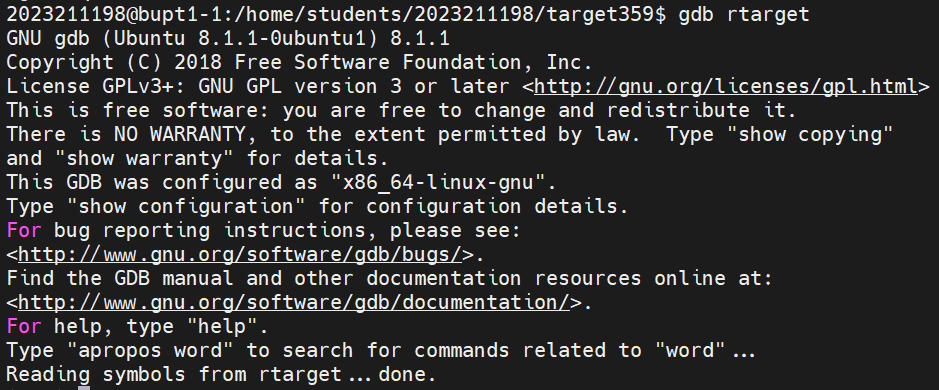


13.用answer3运行，成功调用touch3，并正确传入cookie字符串，阶段三成功

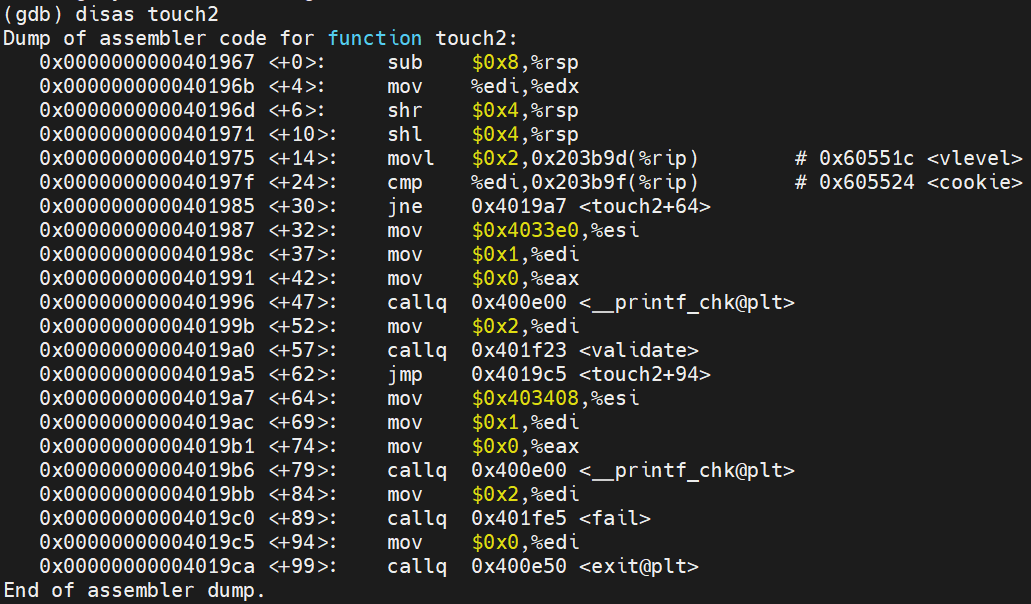


## （五）阶段四：phase4

1. 启动gdb调试rtarget



2.输入disas touch2，查看汇编代码



分析可知touch2传入一个参数与cookie比较，如果相等则成功

3.分析得出所需代码段为

pop %rax

mov %rax,%rdi

ret

4.查表，得汇编代码对应的机器码

|  |  |
| --- | --- |
| 汇编代码 | 机器码 |
| pop %rax | 58 c3 |
| mov %rax,%rdi | 48 89 c7 c3 |

5.将farm.c采用-Og编译



6.将编译生成的farm.o反汇编生成farm.s

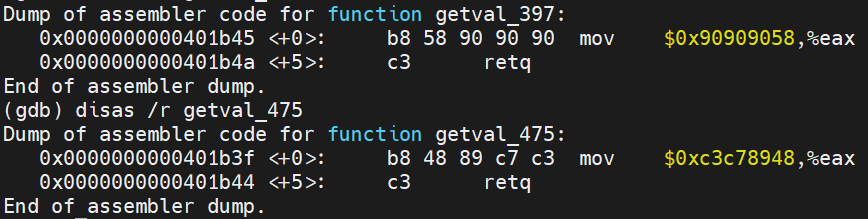


7.在farm.s中寻找符合要求的gadgaet序列

经寻找，getval\_397和getval\_475分别符合两个函数的机器码，对应如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 汇编代码 | 机器码 | farm.c中函数 | 函数对应机器码 |
| pop %rax | 58 c3 | getval\_397 | 0000000000000026 <getval\_397>:  26: b8 58 90 90 90 mov $0x90909058,%eax  2b: c3 |
| mov %rax,%rdi | 48 89 c7 c3 | getval\_475 | 0000000000000020 <getval\_475>:  20: b8 48 89 c7 c3 mov $0xc3c78948,%eax  25: c3 retq |

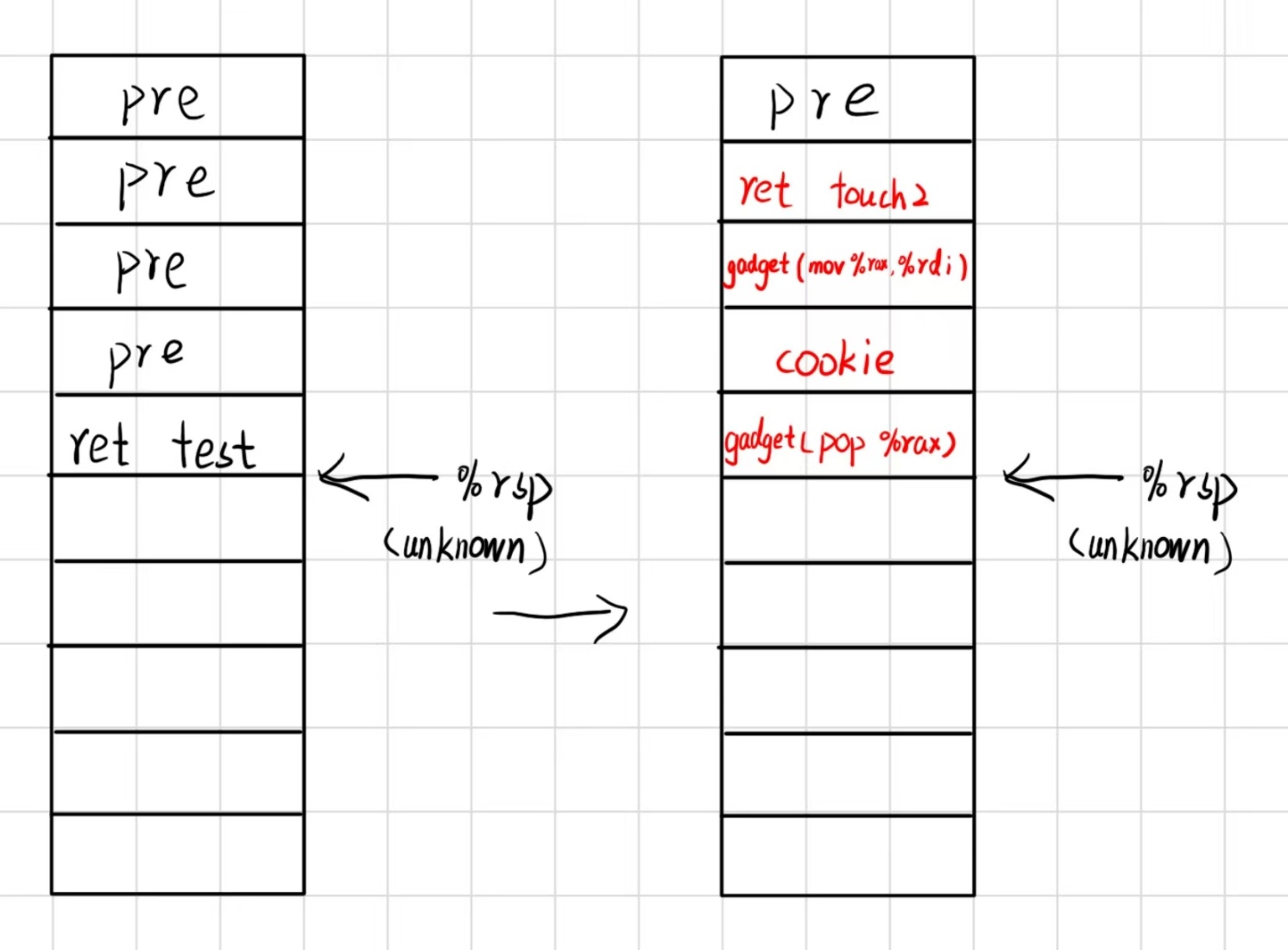
8.在gdb rtarget中反汇编查看代码



9.推算出所需机器码对应地址

|  |  |  |  |
| --- | --- | --- | --- |
| 汇编代码 | 机器码 | farm.c中函数 | 机器码对应位置 |
| pop %rax | 58 c3 | getval\_397 | 0x00401b46 |
| mov %rax,%rdi | 48 89 c7 c3 | getval\_475 | 0x00401b40 |

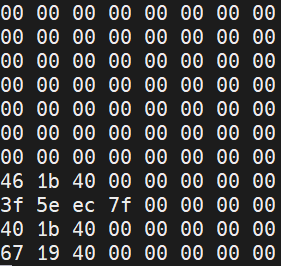
10.栈分析



11. 建立attack4文件



内容如下：



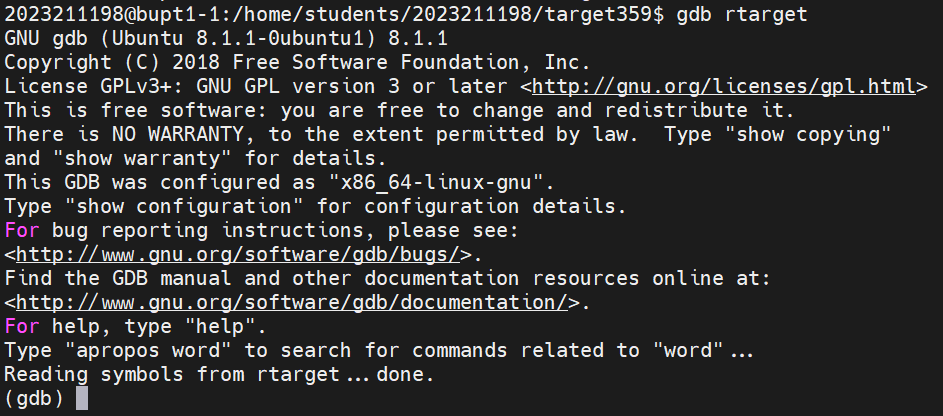
保存并退出



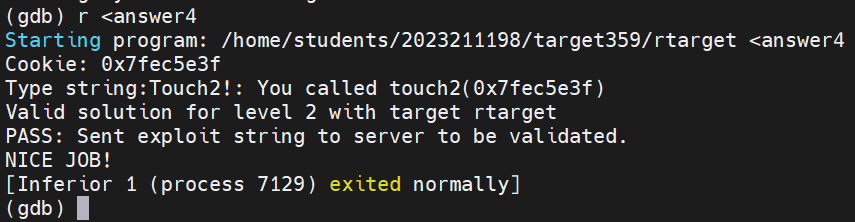
12.将attack4通过hex2raw文件转化为answer4



13.启动gdb调试rtarget

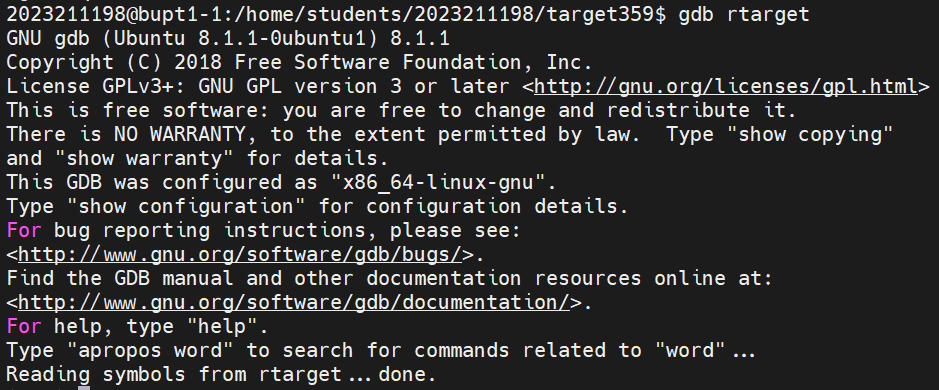


14.用answer4运行，成功调用touch4，并正确传入cookie，阶段四成功

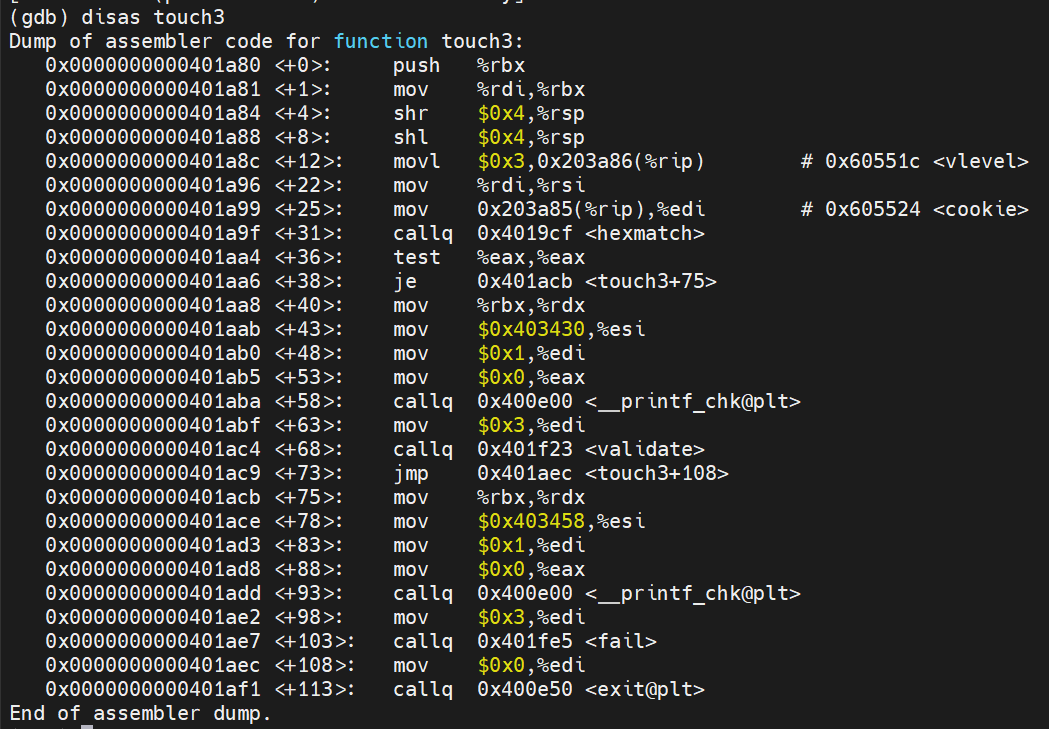


## （六）阶段五：phase5

1. 启动gdb调试rtarget



2.输入disas touch3，查看汇编代码



分析可知touch3传入一个字符串与cookie的字符串比较，如果相等则成功

3.分析得出所需代码段为

pop %rax

mov %rax,%rdi

ret

4.查表，得汇编代码对应的机器码

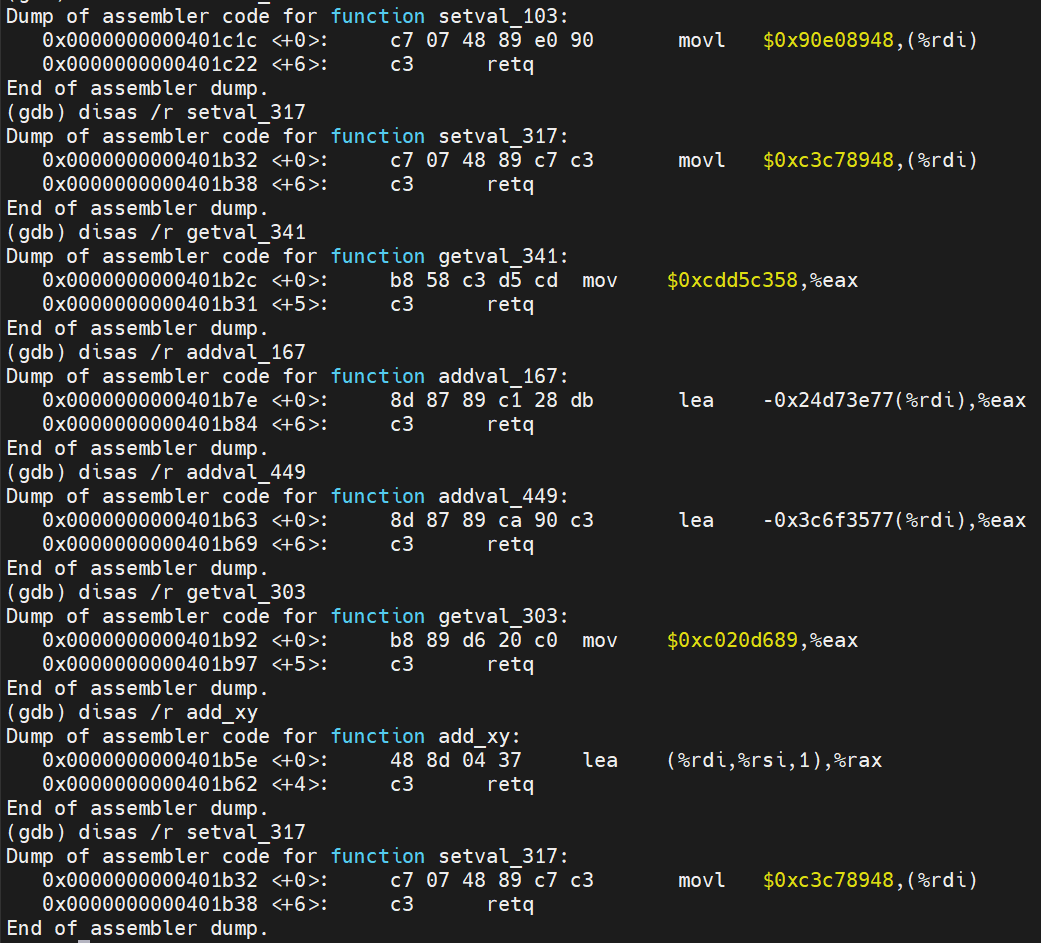
|  |  |
| --- | --- |
| 汇编代码 | 机器码 |
| mov %rsp,%rax | 48 89 e0 c3 |
| mov %rax,%rdi | 48 89 c7 c3 |
| pop %rax | 58 c3 |
| mov %eax,%ecx | 89 c1 XX XX c3 |
| mov %ecx,%edx | 89 ca XX XX c3 |
| mov %edx,%esi | 89 d6 c3 |
| lea (%rdi,%rsi,1),%rax | 48 8d 04 37 c3 |
| mov %rax,%rdi | 48 89 c7 c3 |

5.在farm.s中寻找符合要求的gadgaet序列

经寻找，找到分别符合两函数的机器码，对应如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 汇编代码 | 机器码 | farm.c中函数 | 函数对应机器码 |
| mov %rsp,%rax | 48 89 e0 c3 | setval\_103 | 00000000000000fd <setval\_103>:  fd: c7 07 48 89 e0 90 movl $0x90e08948,(%rdi)  103: c3 retq |
| mov %rax,%rdi | 48 89 c7 c3 | setval\_317 | 0000000000000013 <setval\_317>:  13: c7 07 48 89 c7 c3 movl $0xc3c78948,(%rdi)  19: c3 retq |
| pop %rax | 58 c3 | getval\_341 | 000000000000000d <getval\_341>:  d: b8 58 c3 d5 cd mov $0xcdd5c358,%eax  12: c3 retq |
| mov %eax,%ecx | 89 c1 XX XX c3 | addval\_167 | 000000000000005f <addval\_167>:  5f: 8d 87 89 c1 28 db lea -0x24d73e77(%rdi),%eax  65: c3 retq |
| mov %ecx,%edx | 89 ca XX XX c3 | addval\_449 | 0000000000000044 <addval\_449>:  44: 8d 87 89 ca 90 c3 lea -0x3c6f3577(%rdi),%eax  4a: c3 retq |
| mov %edx,%esi | 89 d6 c3 | getval\_303 | 0000000000000073 <getval\_303>:  73: b8 89 d6 20 c0 mov $0xc020d689,%eax  78: c3 retq |
| lea (%rdi,%rsi,1),%rax | 48 8d 04 37 c3 | add\_xy | 000000000000003f <add\_xy>:  3f: 48 8d 04 37 lea (%rdi,%rsi,1),%rax  43: c3 retq |
| mov %rax,%rdi | 48 89 c7 c3 | setval\_317 | 0000000000000013 <setval\_317>:  13: c7 07 48 89 c7 c3 movl $0xc3c78948,(%rdi)  19: c3 retq |

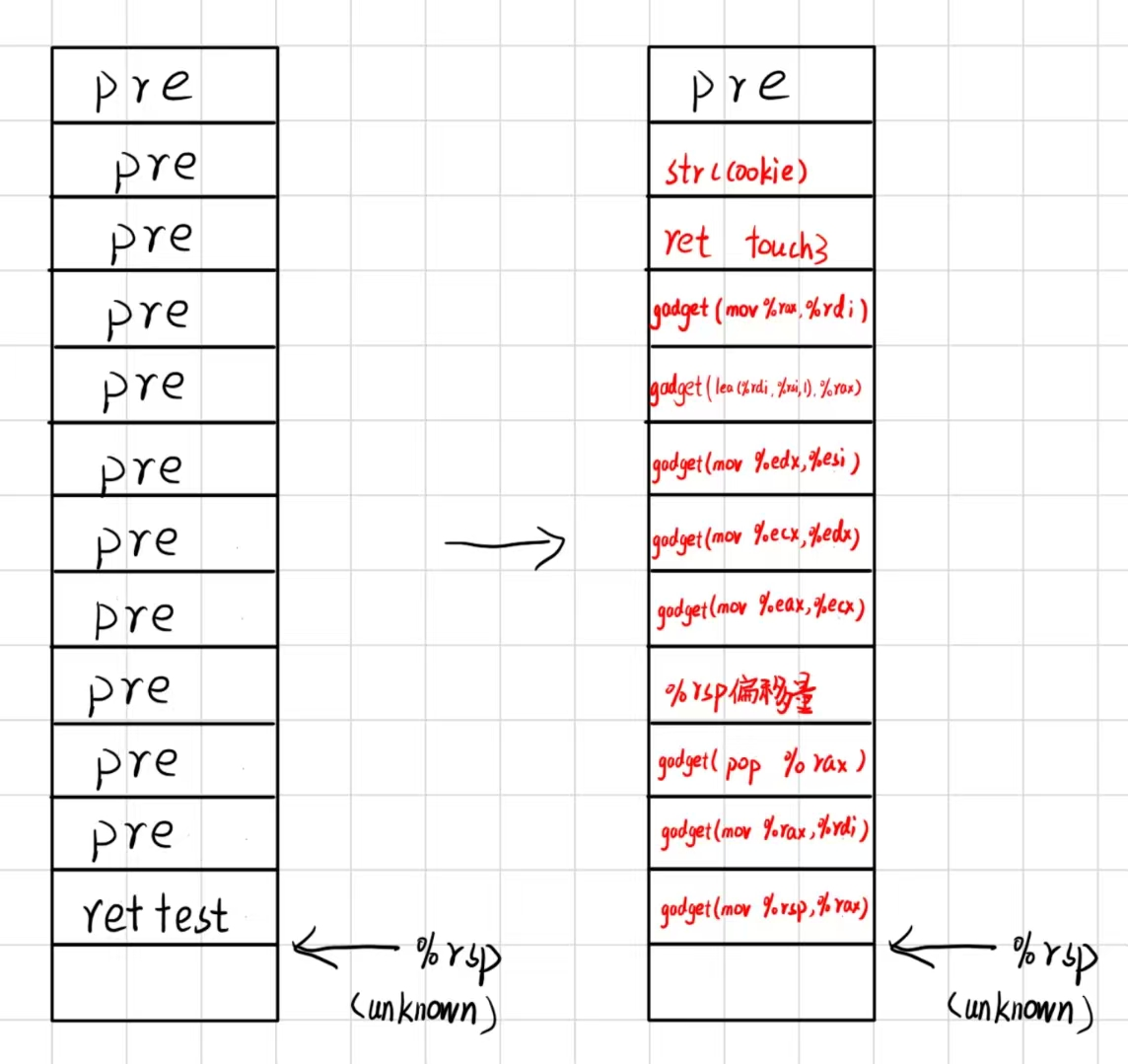
6.在gdb rtarget中反汇编查看代码



7.推算出所需机器码对应地址

|  |  |  |  |
| --- | --- | --- | --- |
| 汇编代码 | 机器码 | farm.c中函数 | 机器码对应位置 |
| mov %rsp,%rax | 48 89 e0 c3 | setval\_103 | 401c1e |
| mov %rax,%rdi | 48 89 c7 c3 | setval\_317 | 401b34 |
| pop %rax | 58 c3 | getval\_341 | 401b2d |
| mov %eax,%ecx | 89 c1 XX XX c3 | addval\_167 | 401b80 |
| mov %ecx,%edx | 89 ca XX XX c3 | addval\_449 | 401b65 |
| mov %edx,%esi | 89 d6 c3 | getval\_303 | 401b93 |
| lea (%rdi,%rsi,1),%rax | 48 8d 04 37 c3 | add\_xy | 401b5e |
| mov %rax,%rdi | 48 89 c7 c3 | setval\_317 | 401b34 |

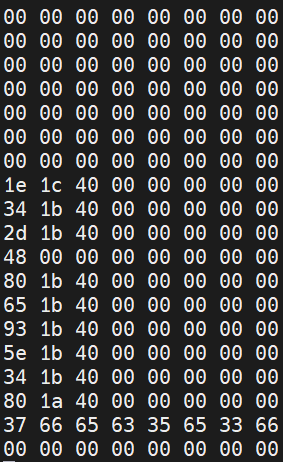
8.栈分析



9. 建立attack5文件



内容如下：



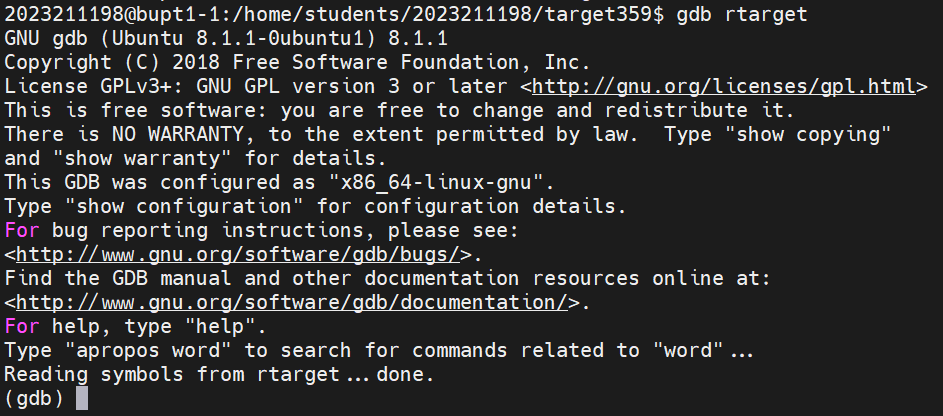
保存并退出

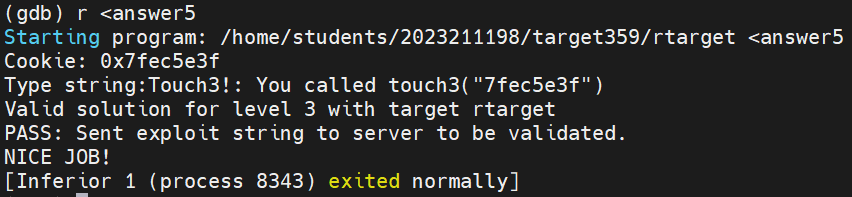


10.将attack5通过hex2raw文件转化为answer5



11.启动gdb调试rtarget



12.用answer5运行，成功调用touch5，并正确传入cookie，阶段五成功

五、总结体会

在第五阶段中，函数存在“1a”导致输入异常截止问题，通过询问以及上网查询，确认问题原因进行替换函数

六、诚信声明（不签扣10分）

需要填写如下声明，并在底部给出手写签名的电子版。

在完成本次实验过程中，我曾分别与以下各位同学就以下方面做过交流：

1.来自肖璨的建议，对第五阶段farm中存在异常的函数进行更换

此外，我还参考了以下资料：

1、CSDN(<https://blog.csdn.net/>)

(1) 【Linux C】GCC编译 && GDB调试 从入门到放弃 （gcc调试选项详解、gdb调试、条件断点、远程调试、脚本化调试）

(<https://blog.csdn.net/weixin_43764974/article/details/130787032>)

用于学习gcc和gdb调试指令

(2) 汇编指令和机器码的对应表

(<https://blog.csdn.net/xqhrs232/article/details/52084324>)

用于研究汇编指令的机器码表示

(3) 汇编基础-----mov与常见寻址方式

(<https://blog.csdn.net/a1309525802/article/details/137461445>)

用于研究汇编指令

3、Bilibili(<https://www.bilibili.com>)

速学150个Linux常用命令

(<https://www.bilibili.com/video/BV12L411a7Ne/?spm_id_from=333.337.search-card.all.click>)

用于学习Linux常用指令

在我提交的程序中，还在对应的位置以注释形式记录了具体的参考内容。

我独立完成了本次实验除以上方面之外的所有工作，包括分析、设计、编码、调试与测试。

我清楚地知道，从以上方面获得的信息在一定程度上降低了实验的难度，可能影响起评分。

我从未使用他人代码，不管是原封不动地复制，还是经过某些等价转换。

我未曾也不会向同一课程（包括此后各届）的同学复制或公开我这份程序的代码，我有义务妥善保管好它们。

我编写这个程序无意于破坏或妨碍任何计算机系统的正常运行。

我清楚地知道，以上情况均为本课程纪律所禁止，若违反，对应的实验成绩将按照0分计。

（签名）