Challenge:-

University Pwn

Me and most of my friends got failed even though we wrote well..
So I decided to take matters in my own hands.
It's Revenge Time.

Author:-Itz_me_strikerr

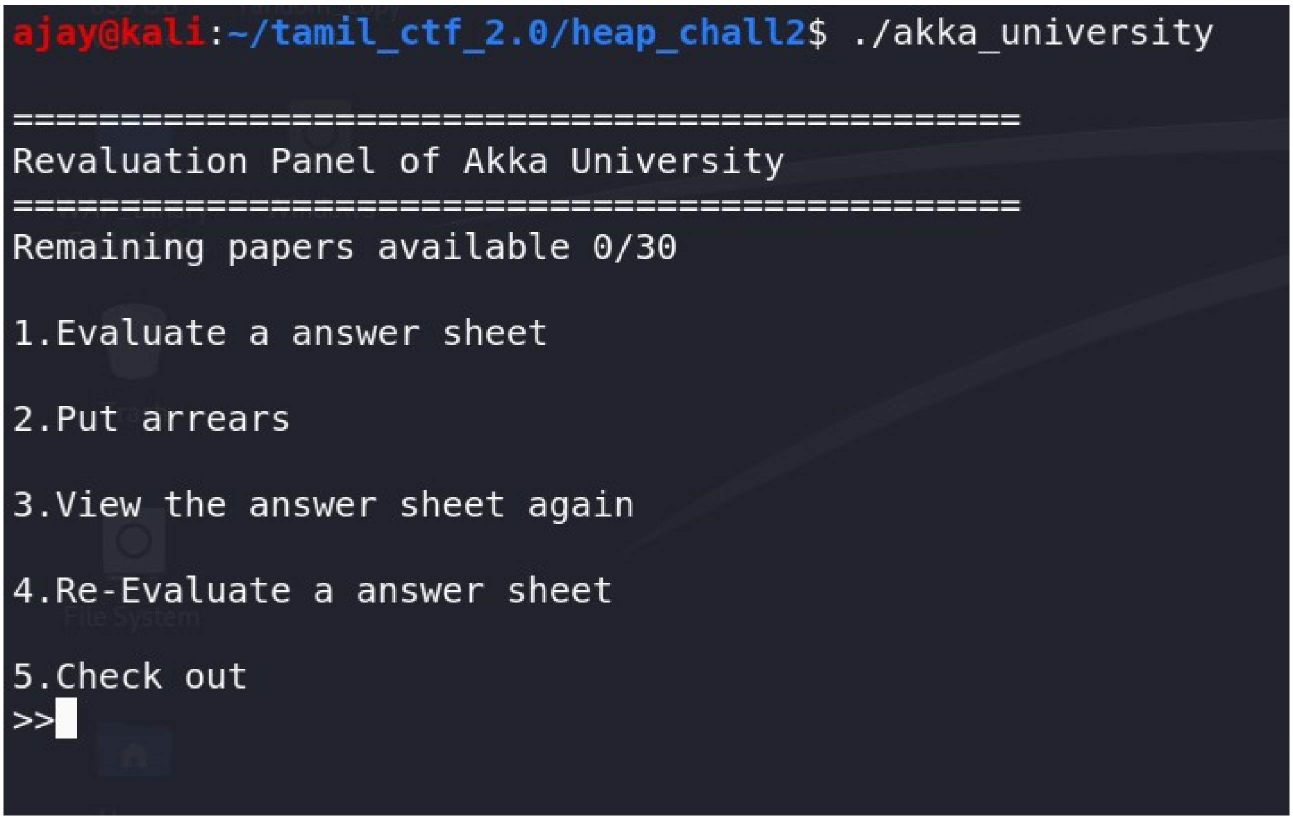
TL-DR:-

Overflow in remarks variable during Re-evaluate a answer sheet function caused by faulty length calculation by strlen allows us to control the address variable to get a use after free primitve and escalate that into a standard Tcache Poisoning to yield a shell.

Writeup:-

We are given a docker file, shared libraries and the challenge binary.

Let’s do some basic enumerations in this challenge binary.

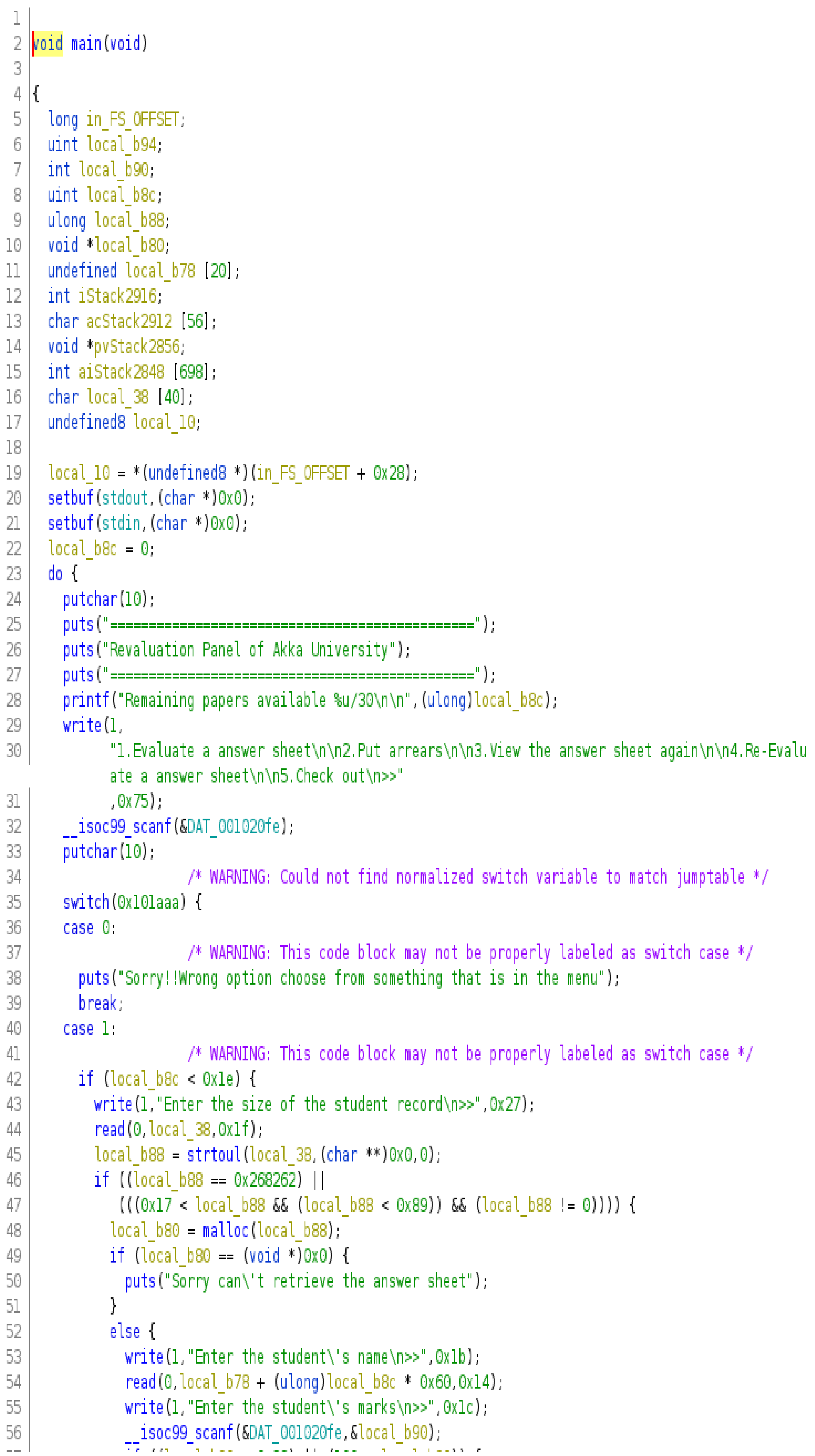


```
ajay@kali:~/tamil_ctf_2.0/heap_chall2$ ./akka_university

===============================================
Revaluation Panel of Akka University
===============================================
Remaining papers available 0/30

1.Evaluate a answer sheet

2.Put arrears

3.View the answer sheet again

4.Re-Evaluate a answer sheet

5.Check out
>>
```

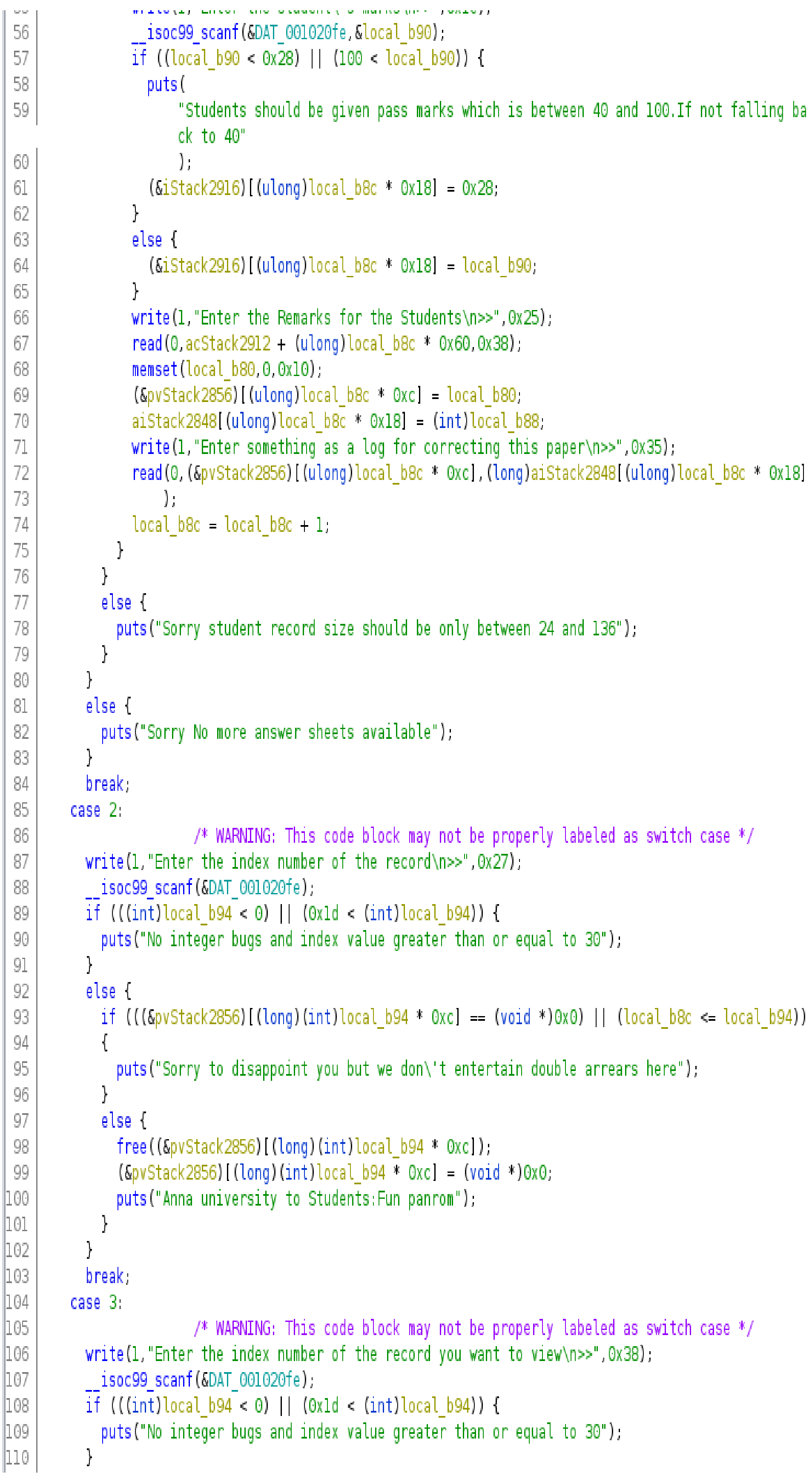
Looks like a normal heap note challenge.

With 5 options, Seems like first option is used to evaluate a answer sheet (malloc) and the second option is used to put arrears(free) and the third option is used to view the answer sheet again(read) will be useful to get memory leaks and the fourth option is used to Re-evaluate a answer sheet(edit contents)

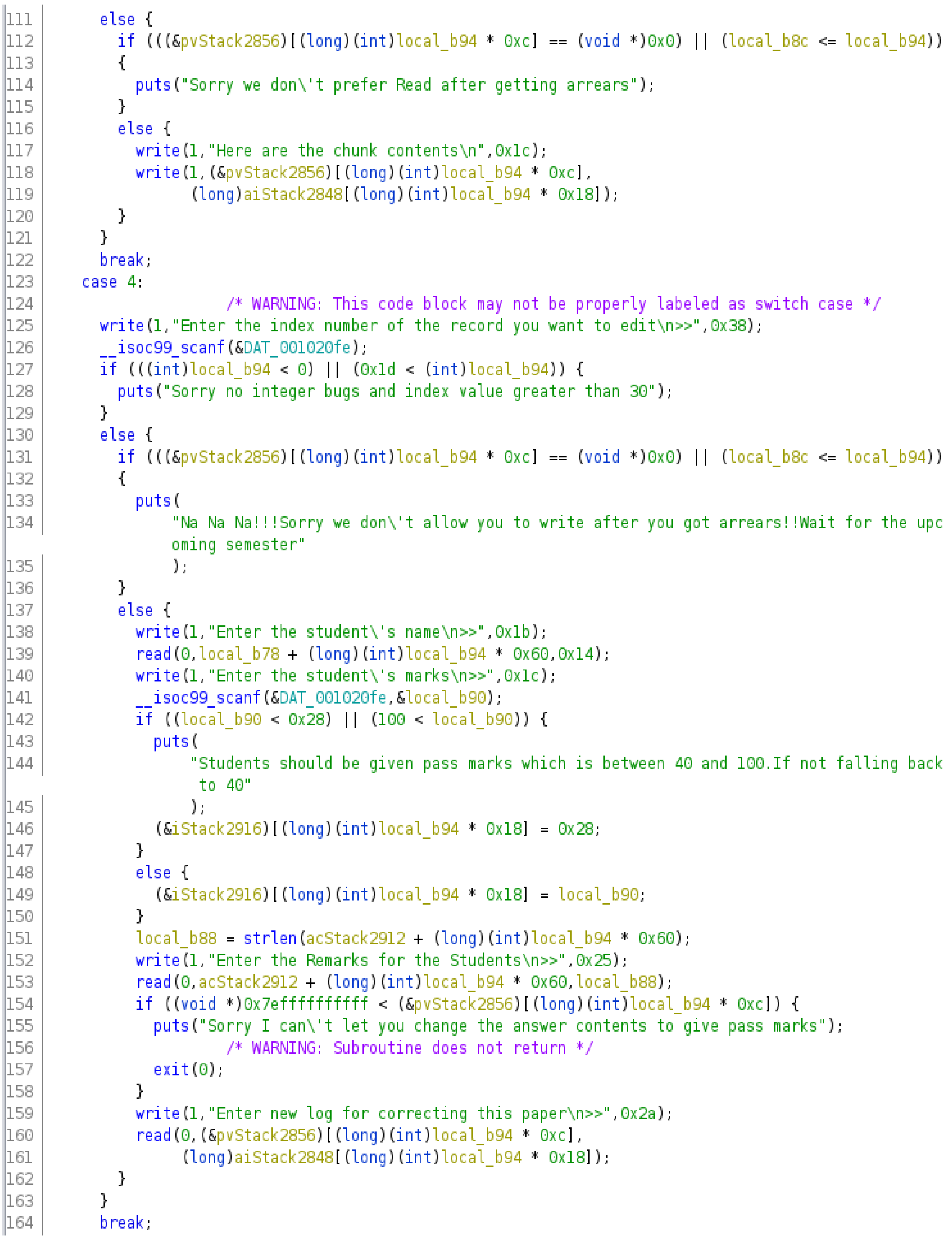
Rather than some assumptions let’s view this binary in ghidra to get a decompiled view for a better understanding..



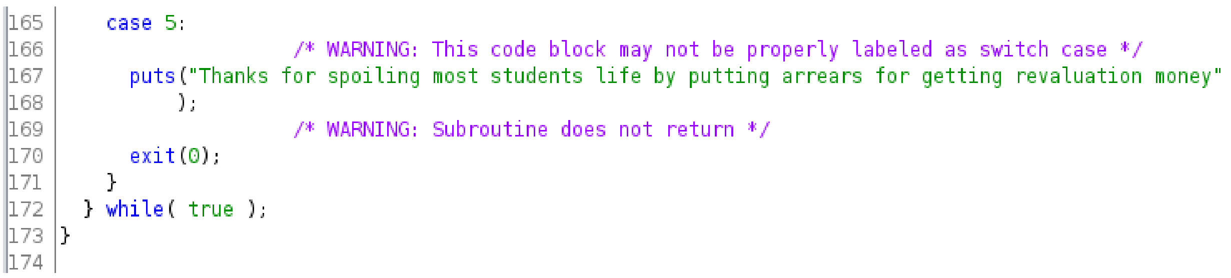
```c

void main(void)

{
  long in_FS_OFFSET;
  uint local_b94;
  int local_b90;
  uint local_b8c;
  ulong local_b88;
  void *local_b80;
  undefined local_b78 [20];
  int iStack2916;
  char acStack2912 [56];
  void *pvStack2856;
  int aiStack2848 [698];
  char local_38 [40];
  undefined8 local_10;

  local_10 = *(undefined8 *)(in_FS_OFFSET + 0x28);
  setbuf(stdout,(char *)0x0);
  setbuf(stdin,(char *)0x0);
  local_b8c = 0;
  do {
    putchar(10);
    puts("===============================================");
    puts("Revaluation Panel of Akka University");
    puts("===============================================");
    printf("Remaining papers available %u/30\n\n",(ulong)local_b8c);
    write(1,
          "1.Evaluate a answer sheet\n\n2.Put arrears\n\n3.View the answer sheet again\n\n4.Re-Evaluate a answer sheet\n\n5.Check out\n>>"
          ,0x75);
    __isoc99_scanf(&DAT_001020fe);
    putchar(10);
                    /* WARNING: Could not find normalized switch variable to match jumptable */
    switch(0x101aaa) {
    case 0:
                    /* WARNING: This code block may not be properly labeled as switch case */
      puts("Sorry!!Wrong option choose from something that is in the menu");
      break;
    case 1:
                    /* WARNING: This code block may not be properly labeled as switch case */
      if (local_b8c < 0x1e) {
        write(1,"Enter the size of the student record\n>>",0x27);
        read(0,local_38,0x1f);
        local_b88 = strtoul(local_38,(char **)0x0,0);
        if ((local_b88 == 0x268262) ||
           (((0x17 < local_b88 && (local_b88 < 0x89)) && (local_b88 != 0)))) {
          local_b80 = malloc(local_b88);
          if (local_b80 == (void *)0x0) {
            puts("Sorry can\'t retrieve the answer sheet");
          }
          else {
            write(1,"Enter the student\'s name\n>>",0x1b);
            read(0,local_b78 + (ulong)local_b8c * 0x60,0x14);
            write(1,"Enter the student\'s marks\n>>",0x1c);
            __isoc99_scanf(&DAT_001020fe,&local_b90);
```



```
        __isoc99_scanf(&DAT_001020fe,&local_b90);
        if ((local_b90 < 0x28) || (100 < local_b90)) {
          puts(
              "Students should be given pass marks which is between 40 and 100.If not falling ba
              ck to 40"
              );
          (&iStack2916)[(ulong)local_b8c * 0x18] = 0x28;
        }
        else {
          (&iStack2916)[(ulong)local_b8c * 0x18] = local_b90;
        }
        write(1,"Enter the Remarks for the Students\n>>",0x25);
        read(0,acStack2912 + (ulong)local_b8c * 0x60,0x38);
        memset(local_b80,0,0x10);
        (&pvStack2856)[(ulong)local_b8c * 0xc] = local_b80;
        aiStack2848[(ulong)local_b8c * 0x18] = (int)local_b88;
        write(1,"Enter something as a log for correcting this paper\n>>",0x35);
        read(0,(&pvStack2856)[(ulong)local_b8c * 0xc],(long)aiStack2848[(ulong)local_b8c * 0x18]
            );
        local_b8c = local_b8c + 1;
      }
    }
    else {
      puts("Sorry student record size should be only between 24 and 136");
    }
  }
  else {
    puts("Sorry No more answer sheets available");
  }
  break;
case 2:
                /* WARNING: This code block may not be properly labeled as switch case */
  write(1,"Enter the index number of the record\n>>",0x27);
  __isoc99_scanf(&DAT_001020fe);
  if (((int)local_b94 < 0) || (0x1d < (int)local_b94)) {
    puts("No integer bugs and index value greater than or equal to 30");
  }
  else {
    if (((&pvStack2856)[(long)(int)local_b94 * 0xc] == (void *)0x0) || (local_b8c <= local_b94))
    {
      puts("Sorry to disappoint you but we don\'t entertain double arrears here");
    }
    else {
      free((&pvStack2856)[(long)(int)local_b94 * 0xc]);
      (&pvStack2856)[(long)(int)local_b94 * 0xc] = (void *)0x0;
      puts("Anna university to Students:Fun panrom");
    }
  }
  break;
case 3:
                /* WARNING: This code block may not be properly labeled as switch case */
  write(1,"Enter the index number of the record you want to view\n>>",0x38);
  __isoc99_scanf(&DAT_001020fe);
  if (((int)local_b94 < 0) || (0x1d < (int)local_b94)) {
    puts("No integer bugs and index value greater than or equal to 30");
  }
```



```
      else {
        if (((&pvStack2856)[(long)(int)local_b94 * 0xc] == (void *)0x0) || (local_b8c <= local_b94))
        {
          puts("Sorry we don\'t prefer Read after getting arrears");
        }
        else {
          write(1,"Here are the chunk contents\n",0x1c);
          write(1,(&pvStack2856)[(long)(int)local_b94 * 0xc],
                (long)aiStack2848[(long)(int)local_b94 * 0x18]);
        }
      }
      break;
    case 4:
                    /* WARNING: This code block may not be properly labeled as switch case */
      write(1,"Enter the index number of the record you want to edit\n>>",0x38);
      __isoc99_scanf(&DAT_001020fe);
      if (((int)local_b94 < 0) || (0x1d < (int)local_b94)) {
        puts("Sorry no integer bugs and index value greater than 30");
      }
      else {
        if (((&pvStack2856)[(long)(int)local_b94 * 0xc] == (void *)0x0) || (local_b8c <= local_b94))
        {
          puts(
              "Na Na Na!!!Sorry we don\'t allow you to write after you got arrears!!Wait for the upc
              oming semester"
              );
        }
        else {
          write(1,"Enter the student\'s name\n>>",0x1b);
          read(0,local_b78 + (long)(int)local_b94 * 0x60,0x14);
          write(1,"Enter the student\'s marks\n>>",0x1c);
          __isoc99_scanf(&DAT_001020fe,&local_b90);
          if ((local_b90 < 0x28) || (100 < local_b90)) {
            puts(
                "Students should be given pass marks which is between 40 and 100.If not falling back
                 to 40"
                );
            (&iStack2916)[(long)(int)local_b94 * 0x18] = 0x28;
          }
          else {
            (&iStack2916)[(long)(int)local_b94 * 0x18] = local_b90;
          }
          local_b88 = strlen(acStack2912 + (long)(int)local_b94 * 0x60);
          write(1,"Enter the Remarks for the Students\n>>",0x25);
          read(0,acStack2912 + (long)(int)local_b94 * 0x60,local_b88);
          if ((void *)0x7effffffffff < (&pvStack2856)[(long)(int)local_b94 * 0xc]) {
            puts("Sorry I can\'t let you change the answer contents to give pass marks");
                    /* WARNING: Subroutine does not return */
            exit(0);
          }
          write(1,"Enter new log for correcting this paper\n>>",0x2a);
          read(0,(&pvStack2856)[(long)(int)local_b94 * 0xc],
               (long)aiStack2848[(long)(int)local_b94 * 0x18]);
        }
      }
      break;
```



```
    case 5:
                    /* WARNING: This code block may not be properly labeled as switch case */
      puts("Thanks for spoiling most students life by putting arrears for getting revaluation money"
          );
                    /* WARNING: Subroutine does not return */
      exit(0);
    }
  } while( true );
}
```

I know it’s a lot to absorb from here..
So let me tell you what every option does here:-

1)Evaluate an answer sheet

It does size checks and see whether the user tries to pass some negative request size and the request size lies between 24 to 136.

Asks for some information and then finally asks for log for correcting the paper which will get placed in the heap.

And from the decompiled view there is something wrong..Slightly ghidra messed up this one.

if((size == system+0x16219a) || (size > 0x17) && (size < 0x89) && (size > 0))

This is given to reduce the burden of creating an unsorted bin attack near __free_hook to create a fake size field which will become more complex. So as to ease this it will also allocate if your request size matches with the /bin/sh memory location and using it with malloc hook which has system address overwritten on it will work as system(“/bin/sh”) when malloc hook written with system address is called.

And checks how may times the chunk are created.. The maximum is 30 chunks.

And after malloc checks the return type to see whether malloc function worked well..

2)Put arrears

Asks the index for the chunk and does some checks to prevent uaf or double free scenarios
If legitimate chunk, it frees it.

3)View the answer sheet again

Asks the index and after some checks to determine whether it is a legitimate chunk it shows the memory contents, if already freed it will not work to prevent read after free scenarios.

4)Re-evaluate a answer sheet

Asks the index and does some important checks to determine whether it is a legitimate chunk or freed chunk to prevent write after free scenarios.
And weirdly it checks for address is not bigger than 0x7f0000000000.

5)Checkout

It simply calls the exit.

<u>Bug:-</u>

On Re-evaluate a answer sheet, when it asks for remarks it uses strlen function to calculate the old remarks, If the length of old remarks is 56 because of the absence of null byte it causes to calculate the length of string beyond it bounds and it also includes the address causes the length to reach up to 62 chars. This gives an overflow primitive to control the address parameter which internally gives a read after and write after free primitive. Now you guys know why I placed the address check which should not be greater than 0x7f0000000000 to avoid a straight overwrite of malloc hook.

<u>Exploit plan:-</u>

1)Allocate 3 chunks of size 136 and free first 2 chunks and trigger the overflow by editing the third chunk to get heap leak to ready up some partial overwrite parameters.

2)Allocate some more 136 sized chunks and free them to fill up tcache and allocate a fastbin sized chunk to perform a guard and then allocate a 136 sized chunk and free to push them to unsorted bin.

3)Then trigger the overflow on the same third chunk overflow and a partial overwrite perform an another read after free to get libc leak from the unsorted bin chunk.

4)Then using the same overflow trick to perform one more partial overwrite to perform a write after free to edit a tcache bin fd pointer to perform a tcache poisoning attack and allocate some chunks to link a fake chunk to edit the malloc hook to system function and use that special allocation to execute system(”/bin/sh”) to yield a shell.

You can find the exploit for this binary in:-
[Will be uploaded in github and link will posted here during publish time]