

lab1: Rev基础

3250101697冯梓航

实验环境: Ubuntu 24.04 (aarch64), GCC 13.3.0, Clang 18.1.3

1. Task 1: rev__begin (57%)

1.1 Compilation (24%)

1.1.1 编译工具

```
● parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ gcc --version
gcc-13 (Ubuntu 13.3.0-6ubuntu2~24.04.1) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

● parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ clang --version
Ubuntu clang version 18.1.3 (1)
Target: aarch64-unknown-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
```

Image 1

安装过程中曾遇到 libllvm18/libclang 版本冲突 (已安装 `-libubuntu1` 版本而仓库中为 `-1` 版本), 通过 `--allow-downgrades` 降级后成功安装。

1.1.2 使用 g++ 对代码进行预处理 (宏展开)

```
lab1_rev > rev_begin_preprocessed.cpp > ...
50097 int main()
50101     cin >> input;
50102
50103     if (input.length() != 26)
50104     {
50105         cout << "Length Wrong!" << endl;
50106         return 0;
50107     }
50108
50109     unsigned char target[] = {
50110         0xFC, 0xFC, 0xF4, 0xD6, 0xF7, 0xAA, 0xEF, 0xEA, 0xEF, 0xF0,
50111         0xB6, 0xCC, 0x93, 0x8E, 0xB2, 0x12, 0xBB, 0x0C, 0x9E, 0xFC,
50112         0x42, 0xE9, 0x26, 0xD9, 0xDB, 0x08};
50113
50114     bool ok = true;
50115
50116     for (int i : views::iota(0, 26))
50117     {
50118         cmp(static_cast<unsigned char>(input[i]), target[i], i, ok);
50119     }
50120
50121     if (ok)
50122     {
50123         cout << "Flag Correct!" << endl;
50124     }
50125     else
50126     {
50127         cout << "Flag Wrong!" << endl;
50128     }
50129
50130     return 0;
50131 }
50132
问题 4/9 输出 调试控制台 终端 窗口
● parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ g++ -std=c++20 -E rev_begin.cpp -o rev_begin_preprocessed.cpp
● parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$
```

Image 2

展开所有 `#include` 头文件、宏定义，去除注释
展开后文件末尾保留原始代码逻辑

1.1.3 使用 g++ 将代码编译为汇编代码

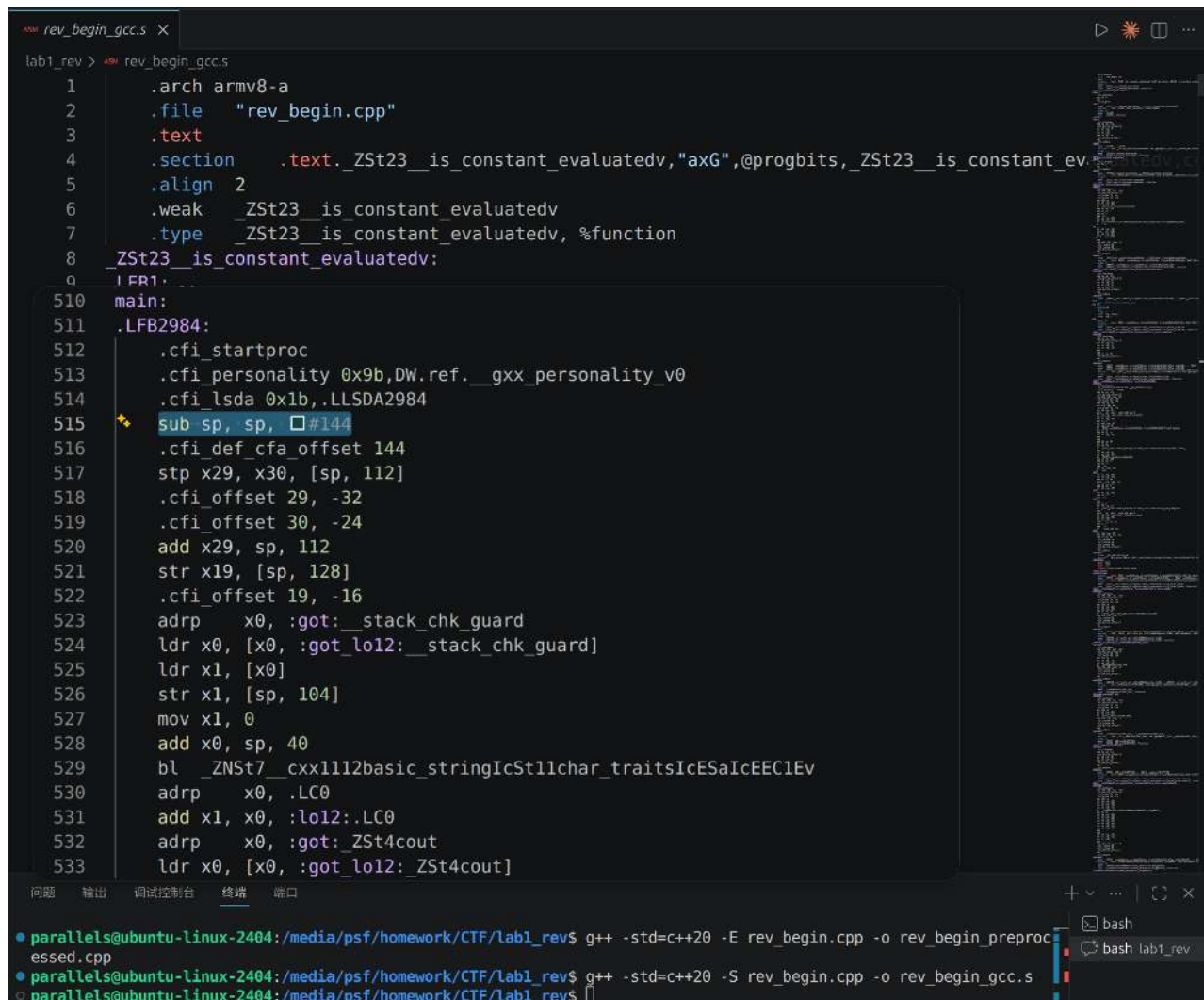


Image 3

将 C++ 代码编译为 ARM aarch64 汇编代码 (电脑是 arm 架构的)

汇编文件结构:

```
1      .arch armv8-a
2      .file    "rev_begin.cpp"
3      .text
4      // ... 各种 symbol 定义 ...
5
6  main:
7      .LFB2984:
8          .cfi_startproc
9          sub     sp, sp, #144          ; 分配栈帧 144 字节
10         stp     x29, x30, [sp, 112] ; 保存帧指针和返回地址
11         add     x29, sp, 112
12         // ... 调用 cout << "Please enter the flag: "
13         // ... 调用 cin >> input
```

14 // ... 长度检查、循环变换验证等

1.1.4 使用 g++ 将代码汇编（不链接）为目标文件

```
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ g++ -std=c++20 -c rev_begin.cpp -o rev_begin_gcc.o
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ file rev_begin_gcc.o
rev_begin_gcc.o: ELF 64-bit LSB relocatable, ARM aarch64, version 1 (SYSV), not stripped
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$
```

Image 4

将汇编代码翻译为机器码（目标文件），但不进行链接。

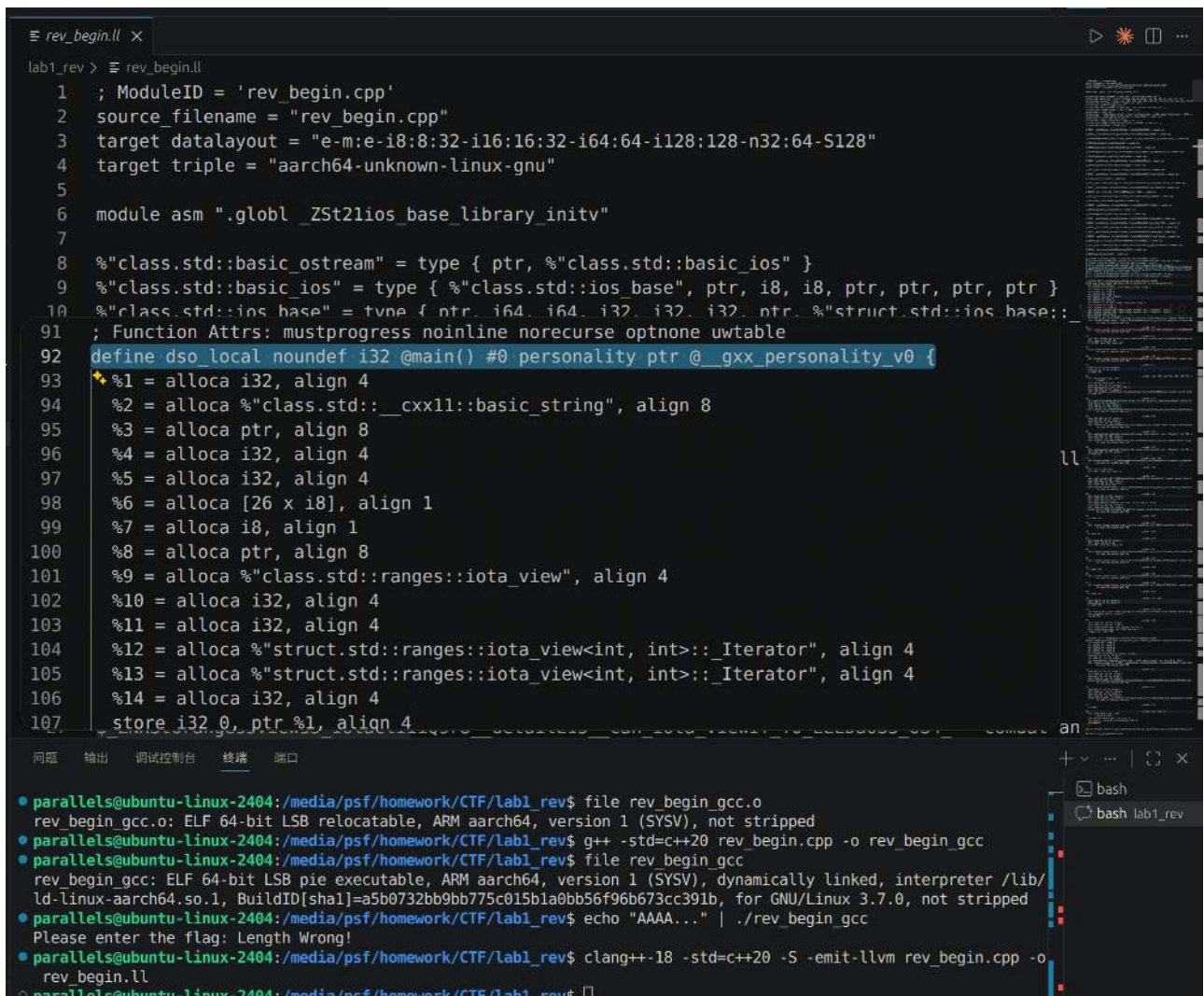
可以看到它是 `relocatable`（可重定位）文件，尚未链接。

1.1.5 使用 g++ 进行链接，生成可执行文件，并尝试运行

```
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ g++ -std=c++20 rev_begin.cpp -o rev_begin_gcc
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ file rev_begin_gcc
rev_begin_gcc: ELF 64-bit LSB pie executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=a5b0732bb9bb775c015b1a0bb56f96b673cc391b, for GNU/Linux 3.7.0, not stripped
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ echo "AAAA..." | ./rev_begin_gcc
Please enter the flag: Length Wrong!
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$
```

Image 5

1.1.6 使用 clang++ 生成文本格式的 LLVM IR



```
1 ; ModuleID = 'rev_begin.cpp'
2 source_filename = "rev_begin.cpp"
3 target datalayout = "e-m:e-i8:8:32-i16:16:32-i64:64-i128:128-n32:64-S128"
4 target triple = "aarch64-unknown-linux-gnu"
5
6 module asm ".globl _ZSt21ios_base_library_initv"
7
8 "%class.std::basic_ostream" = type { ptr, "%class.std::basic_ios" }
9 "%class.std::basic_ios" = type { "%class.std::ios_base", ptr, i8, i8, ptr, ptr, ptr, ptr }
10 "%class.std::ios_base" = type { ptr, i64, i64, i32, i32, i32, ptr, "%struct.std::ios_base::..." }
91 ; Function Attrs: mustprogress noinline norecurse optnone uwtable
92 define dso_local noundef i32 @main() #0 personality ptr @ _gxx_personality_v0 {
93     %1 = alloca i32, align 4
94     %2 = alloca "%class.std::basic_string", align 8
95     %3 = alloca ptr, align 8
96     %4 = alloca i32, align 4
97     %5 = alloca i32, align 4
98     %6 = alloca [26 x i8], align 1
99     %7 = alloca i8, align 1
100    %8 = alloca ptr, align 8
101    %9 = alloca "%class.std::ranges::iota_view", align 4
102    %10 = alloca i32, align 4
103    %11 = alloca i32, align 4
104    %12 = alloca "%struct.std::ranges::iota_view<int, int>::Iterator", align 4
105    %13 = alloca "%struct.std::ranges::iota_view<int, int>::Iterator", align 4
106    %14 = alloca i32, align 4
107    store i32 0, ptr %1, align 4
108    ...
109    ...
110    ...
111    ...
112    ...
113    ...
114    ...
115    ...
116    ...
117    ...
118    ...
119    ...
120    ...
121    ...
122    ...
123    ...
124    ...
125    ...
126    ...
127    ...
128    ...
129    ...
130    ...
131    ...
132    ...
133    ...
134    ...
135    ...
136    ...
137    ...
138    ...
139    ...
140    ...
141    ...
142    ...
143    ...
144    ...
145    ...
146    ...
147    ...
148    ...
149    ...
150    ...
151    ...
152    ...
153    ...
154    ...
155    ...
156    ...
157    ...
158    ...
159    ...
160    ...
161    ...
162    ...
163    ...
164    ...
165    ...
166    ...
167    ...
168    ...
169    ...
170    ...
171    ...
172    ...
173    ...
174    ...
175    ...
176    ...
177    ...
178    ...
179    ...
180    ...
181    ...
182    ...
183    ...
184    ...
185    ...
186    ...
187    ...
188    ...
189    ...
190    ...
191    ...
192    ...
193    ...
194    ...
195    ...
196    ...
197    ...
198    ...
199    ...
200    ...
201    ...
202    ...
203    ...
204    ...
205    ...
206    ...
207    ...
208    ...
209    ...
210    ...
211    ...
212    ...
213    ...
214    ...
215    ...
216    ...
217    ...
218    ...
219    ...
220    ...
221    ...
222    ...
223    ...
224    ...
225    ...
226    ...
227    ...
228    ...
229    ...
230    ...
231    ...
232    ...
233    ...
234    ...
235    ...
236    ...
237    ...
238    ...
239    ...
240    ...
241    ...
242    ...
243    ...
244    ...
245    ...
246    ...
247    ...
248    ...
249    ...
250    ...
251    ...
252    ...
253    ...
254    ...
255    ...
256    ...
257    ...
258    ...
259    ...
260    ...
261    ...
262    ...
263    ...
264    ...
265    ...
266    ...
267    ...
268    ...
269    ...
270    ...
271    ...
272    ...
273    ...
274    ...
275    ...
276    ...
277    ...
278    ...
279    ...
280    ...
281    ...
282    ...
283    ...
284    ...
285    ...
286    ...
287    ...
288    ...
289    ...
290    ...
291    ...
292    ...
293    ...
294    ...
295    ...
296    ...
297    ...
298    ...
299    ...
300    ...
301    ...
302    ...
303    ...
304    ...
305    ...
306    ...
307    ...
308    ...
309    ...
310    ...
311    ...
312    ...
313    ...
314    ...
315    ...
316    ...
317    ...
318    ...
319    ...
320    ...
321    ...
322    ...
323    ...
324    ...
325    ...
326    ...
327    ...
328    ...
329    ...
330    ...
331    ...
332    ...
333    ...
334    ...
335    ...
336    ...
337    ...
338    ...
339    ...
340    ...
341    ...
342    ...
343    ...
344    ...
345    ...
346    ...
347    ...
348    ...
349    ...
350    ...
351    ...
352    ...
353    ...
354    ...
355    ...
356    ...
357    ...
358    ...
359    ...
360    ...
361    ...
362    ...
363    ...
364    ...
365    ...
366    ...
367    ...
368    ...
369    ...
370    ...
371    ...
372    ...
373    ...
374    ...
375    ...
376    ...
377    ...
378    ...
379    ...
380    ...
381    ...
382    ...
383    ...
384    ...
385    ...
386    ...
387    ...
388    ...
389    ...
390    ...
391    ...
392    ...
393    ...
394    ...
395    ...
396    ...
397    ...
398    ...
399    ...
400    ...
401    ...
402    ...
403    ...
404    ...
405    ...
406    ...
407    ...
408    ...
409    ...
410    ...
411    ...
412    ...
413    ...
414    ...
415    ...
416    ...
417    ...
418    ...
419    ...
420    ...
421    ...
422    ...
423    ...
424    ...
425    ...
426    ...
427    ...
428    ...
429    ...
430    ...
431    ...
432    ...
433    ...
434    ...
435    ...
436    ...
437    ...
438    ...
439    ...
440    ...
441    ...
442    ...
443    ...
444    ...
445    ...
446    ...
447    ...
448    ...
449    ...
450    ...
451    ...
452    ...
453    ...
454    ...
455    ...
456    ...
457    ...
458    ...
459    ...
460    ...
461    ...
462    ...
463    ...
464    ...
465    ...
466    ...
467    ...
468    ...
469    ...
470    ...
471    ...
472    ...
473    ...
474    ...
475    ...
476    ...
477    ...
478    ...
479    ...
480    ...
481    ...
482    ...
483    ...
484    ...
485    ...
486    ...
487    ...
488    ...
489    ...
490    ...
491    ...
492    ...
493    ...
494    ...
495    ...
496    ...
497    ...
498    ...
499    ...
500    ...
501    ...
502    ...
503    ...
504    ...
505    ...
506    ...
507    ...
508    ...
509    ...
510    ...
511    ...
512    ...
513    ...
514    ...
515    ...
516    ...
517    ...
518    ...
519    ...
520    ...
521    ...
522    ...
523    ...
524    ...
525    ...
526    ...
527    ...
528    ...
529    ...
530    ...
531    ...
532    ...
533    ...
534    ...
535    ...
536    ...
537    ...
538    ...
539    ...
540    ...
541    ...
542    ...
543    ...
544    ...
545    ...
546    ...
547    ...
548    ...
549    ...
550    ...
551    ...
552    ...
553    ...
554    ...
555    ...
556    ...
557    ...
558    ...
559    ...
560    ...
561    ...
562    ...
563    ...
564    ...
565    ...
566    ...
567    ...
568    ...
569    ...
570    ...
571    ...
572    ...
573    ...
574    ...
575    ...
576    ...
577    ...
578    ...
579    ...
580    ...
581    ...
582    ...
583    ...
584    ...
585    ...
586    ...
587    ...
588    ...
589    ...
590    ...
591    ...
592    ...
593    ...
594    ...
595    ...
596    ...
597    ...
598    ...
599    ...
600    ...
601    ...
602    ...
603    ...
604    ...
605    ...
606    ...
607    ...
608    ...
609    ...
610    ...
611    ...
612    ...
613    ...
614    ...
615    ...
616    ...
617    ...
618    ...
619    ...
620    ...
621    ...
622    ...
623    ...
624    ...
625    ...
626    ...
627    ...
628    ...
629    ...
630    ...
631    ...
632    ...
633    ...
634    ...
635    ...
636    ...
637    ...
638    ...
639    ...
640    ...
641    ...
642    ...
643    ...
644    ...
645    ...
646    ...
647    ...
648    ...
649    ...
650    ...
651    ...
652    ...
653    ...
654    ...
655    ...
656    ...
657    ...
658    ...
659    ...
660    ...
661    ...
662    ...
663    ...
664    ...
665    ...
666    ...
667    ...
668    ...
669    ...
670    ...
671    ...
672    ...
673    ...
674    ...
675    ...
676    ...
677    ...
678    ...
679    ...
680    ...
681    ...
682    ...
683    ...
684    ...
685    ...
686    ...
687    ...
688    ...
689    ...
690    ...
691    ...
692    ...
693    ...
694    ...
695    ...
696    ...
697    ...
698    ...
699    ...
700    ...
701    ...
702    ...
703    ...
704    ...
705    ...
706    ...
707    ...
708    ...
709    ...
710    ...
711    ...
712    ...
713    ...
714    ...
715    ...
716    ...
717    ...
718    ...
719    ...
720    ...
721    ...
722    ...
723    ...
724    ...
725    ...
726    ...
727    ...
728    ...
729    ...
730    ...
731    ...
732    ...
733    ...
734    ...
735    ...
736    ...
737    ...
738    ...
739    ...
740    ...
741    ...
742    ...
743    ...
744    ...
745    ...
746    ...
747    ...
748    ...
749    ...
750    ...
751    ...
752    ...
753    ...
754    ...
755    ...
756    ...
757    ...
758    ...
759    ...
760    ...
761    ...
762    ...
763    ...
764    ...
765    ...
766    ...
767    ...
768    ...
769    ...
770    ...
771    ...
772    ...
773    ...
774    ...
775    ...
776    ...
777    ...
778    ...
779    ...
780    ...
781    ...
782    ...
783    ...
784    ...
785    ...
786    ...
787    ...
788    ...
789    ...
790    ...
791    ...
792    ...
793    ...
794    ...
795    ...
796    ...
797    ...
798    ...
799    ...
800    ...
801    ...
802    ...
803    ...
804    ...
805    ...
806    ...
807    ...
808    ...
809    ...
810    ...
811    ...
812    ...
813    ...
814    ...
815    ...
816    ...
817    ...
818    ...
819    ...
820    ...
821    ...
822    ...
823    ...
824    ...
825    ...
826    ...
827    ...
828    ...
829    ...
830    ...
831    ...
832    ...
833    ...
834    ...
835    ...
836    ...
837    ...
838    ...
839    ...
840    ...
841    ...
842    ...
843    ...
844    ...
845    ...
846    ...
847    ...
848    ...
849    ...
850    ...
851    ...
852    ...
853    ...
854    ...
855    ...
856    ...
857    ...
858    ...
859    ...
860    ...
861    ...
862    ...
863    ...
864    ...
865    ...
866    ...
867    ...
868    ...
869    ...
870    ...
871    ...
872    ...
873    ...
874    ...
875    ...
876    ...
877    ...
878    ...
879    ...
880    ...
881    ...
882    ...
883    ...
884    ...
885    ...
886    ...
887    ...
888    ...
889    ...
890    ...
891    ...
892    ...
893    ...
894    ...
895    ...
896    ...
897    ...
898    ...
899    ...
900    ...
901    ...
902    ...
903    ...
904    ...
905    ...
906    ...
907    ...
908    ...
909    ...
910    ...
911    ...
912    ...
913    ...
914    ...
915    ...
916    ...
917    ...
918    ...
919    ...
920    ...
921    ...
922    ...
923    ...
924    ...
925    ...
926    ...
927    ...
928    ...
929    ...
930    ...
931    ...
932    ...
933    ...
934    ...
935    ...
936    ...
937    ...
938    ...
939    ...
940    ...
941    ...
942    ...
943    ...
944    ...
945    ...
946    ...
947    ...
948    ...
949    ...
950    ...
951    ...
952    ...
953    ...
954    ...
955    ...
956    ...
957    ...
958    ...
959    ...
960    ...
961    ...
962    ...
963    ...
964    ...
965    ...
966    ...
967    ...
968    ...
969    ...
970    ...
971    ...
972    ...
973    ...
974    ...
975    ...
976    ...
977    ...
978    ...
979    ...
980    ...
981    ...
982    ...
983    ...
984    ...
985    ...
986    ...
987    ...
988    ...
989    ...
990    ...
991    ...
992    ...
993    ...
994    ...
995    ...
996    ...
997    ...
998    ...
999    ...
1000   ...
1001   ...
1002   ...
1003   ...
1004   ...
1005   ...
1006   ...
1007   ...
1008   ...
1009   ...
1010   ...
1011   ...
1012   ...
1013   ...
1014   ...
1015   ...
1016   ...
1017   ...
1018   ...
1019   ...
1020   ...
1021   ...
1022   ...
1023   ...
1024   ...
1025   ...
1026   ...
1027   ...
1028   ...
1029   ...
1030   ...
1031   ...
1032   ...
1033   ...
1034   ...
1035   ...
1036   ...
1037   ...
1038   ...
1039   ...
1040   ...
1041   ...
1042   ...
1043   ...
1044   ...
1045   ...
1046   ...
1047   ...
1048   ...
1049   ...
1050   ...
1051   ...
1052   ...
1053   ...
1054   ...
1055   ...
1056   ...
1057   ...
1058   ...
1059   ...
1060   ...
1061   ...
1062   ...
1063   ...
1064   ...
1065   ...
1066   ...
1067   ...
1068   ...
1069   ...
1070   ...
1071   ...
1072   ...
1073   ...
1074   ...
1075   ...
1076   ...
1077   ...
1078   ...
1079   ...
1080   ...
1081   ...
1082   ...
1083   ...
1084   ...
1085   ...
1086   ...
1087   ...
1088   ...
1089   ...
1090   ...
1091   ...
1092   ...
1093   ...
1094   ...
1095   ...
1096   ...
1097   ...
1098   ...
1099   ...
1100   ...
1101   ...
1102   ...
1103   ...
1104   ...
1105   ...
1106   ...
1107   ...
1108   ...
1109   ...
1110   ...
1111   ...
1112   ...
1113   ...
1114   ...
1115   ...
1116   ...
1117   ...
1118   ...
1119   ...
1120   ...
1121   ...
1122   ...
1123   ...
1124   ...
1125   ...
1126   ...
1127   ...
1128   ...
1129   ...
1130   ...
1131   ...
1132   ...
1133   ...
1134   ...
1135   ...
1136   ...
1137   ...
1138   ...
1139   ...
1140   ...
1141   ...
1142   ...
1143   ...
1144   ...
1145   ...
1146   ...
1147   ...
1148   ...
1149   ...
1150   ...
1151   ...
1152   ...
1153   ...
1154   ...
1155   ...
1156   ...
1157   ...
1158   ...
1159   ...
1160   ...
1161   ...
1162   ...
1163   ...
1164   ...
1165   ...
1166   ...
1167   ...
1168   ...
1169   ...
1170   ...
1171   ...
1172   ...
1173   ...
1174   ...
1175   ...
1176   ...
1177   ...
1178   ...
1179   ...
1180   ...
1181   ...
1182   ...
1183   ...
1184   ...
1185   ...
1186   ...
1187   ...
1188   ...
1189   ...
1190   ...
1191   ...
1192   ...
1193   ...
1194   ...
1195   ...
1196   ...
1197   ...
1198   ...
1199   ...
1200   ...
1201   ...
1202   ...
1203   ...
1204   ...
1205   ...
1206   ...
1207   ...
1208   ...
1209   ...
1210   ...
1211   ...
1212   ...
1213   ...
1214   ...
1215   ...
1216   ...
1217   ...
1218   ...
1219   ...
1220   ...
1221   ...
1222   ...
1223   ...
1224   ...
1225   ...
1226   ...
1227   ...
1228   ...
1229   ...
1230   ...
1231   ...
1232   ...
1233   ...
1234   ...
1235   ...
1236   ...
1237   ...
1238   ...
1239   ...
1240   ...
1241   ...
1242   ...
1243   ...
1244   ...
1245   ...
1246   ...
1247   ...
1248   ...
1249   ...
1250   ...
1251   ...
1252   ...
1253   ...
1254   ...
1255   ...
1256   ...
1257   ...
1258   ...
1259   ...
1260   ...
1261   ...
1262   ...
1263   ...
1264   ...
1265   ...
1266   ...
1267   ...
1268   ...
1269   ...
1270   ...
1271   ...
1272   ...
1273   ...
1274   ...
1275   ...
1276   ...
1277   ...
1278   ...
1279   ...
1280   ...
1281   ...
1282   ...
1283   ...
1284   ...
1285   ...
1286   ...
1287   ...
1288   ...
1289   ...
1290   ...
1291   ...
1292   ...
1293   ...
1294   ...
1295   ...
1296   ...
1297   ...
1298   ...
1299   ...
1300   ...
1301   ...
1302   ...
1303   ...
1304   ...
1305   ...
1306   ...
1307   ...
1308   ...
1309   ...
1310   ...
1311   ...
1312   ...
1313   ...
1314   ...
1315   ...
1316   ...
1317   ...
1318   ...
1319   ...
1320   ...
1321   ...
1322   ...
1323   ...
1324   ...
1325   ...
1326   ...
1327   ...
1328   ...
1329   ...
1330   ...
1331   ...
1332   ...
1333   ...
1334   ...
1335   ...
1336   ...
1337   ...
1338   ...
1339   ...
1340   ...
1341   ...
1342   ...
1343   ...
1344   ...
1345   ...
1346   ...
1347   ...
1348   ...
1349   ...
1350   ...
1351   ...
1352   ...
1353   ...
1354   ...
1355   ...
1356   ...
1357   ...
1358   ...
1359   ...
1360   ...
1361   ...
1362   ...
1363   ...
1364   ...
1365   ...
1366   ...
1367   ...
1368   ...
1369   ...
1370   ...
1371   ...
1372   ...
1373   ...
1374   ...
1375   ...
1376   ...
1377   ...
1378   ...
1379   ...
1380   ...
1381   ...
1382   ...
1383   ...
1384   ...
1385   ...
1386   ...
1387   ...
1388   ...
1389   ...
1390   ...
1391   ...
1392   ...
1393   ...
1394   ...
1395   ...
1396   ...
1397   ...
1398   ...
1399   ...
1400   ...
1401   ...
1402   ...
1403   ...
1404   ...
1405   ...
1406   ...
1407   ...
1408   ...
1409   ...
1410   ...
1411   ...
1412   ...
1413   ...
1414   ...
1415   ...
1416   ...
1417   ...
1418   ...
1419   ...
1420   ...
1421   ...
1422   ...
1423   ...
1424   ...
1425   ...
1426   ...
1427   ...
1428   ...
1429   ...
1430   ...
1431   ...
1432   ...
1433   ...
1434   ...
1435   ...
1436   ...
1437   ...
1438   ...
1439   ...
1440   ...
1441   ...
1442   ...
1443   ...
1444   ...
1445   ...
1446   ...
1447   ...
1448   ...
1449   ...
1450   ...
1451   ...
1452   ...
1453   ...
1454   ...
1455   ...
1456   ...
1457   ...
1458   ...
1459   ...
1460   ...
1461   ...
1462   ...
1463   ...
1464   ...
1465   ...
1466   ...
1467   ...
1468   ...
1469   ...
1470   ...
1471   ...
1472   ...
1473   ...
1474   ...
1475   ...
1476   ...
1477   ...
1478   ...
1479   ...
1480   ...
1481   ...
1482   ...
1483   ...
1484   ...
1485   ...
1486   ...
1487   ...
1488   ...
1489   ...
1490   ...
1491   ...
1492   ...
1493   ...
1494   ...
1495   ...
1496   ...
1497   ...
1498   ...
1499   ...
1500   ...
1501   ...
1502   ...
1503   ...
1504   ...
1505   ...
1506   ...
1507   ...
1508   ...
1509   ...
1510   ...
1511   ...
1512   ...
1513   ...
1514   ...
1515   ...
1516   ...
1517   ...
1518   ...
1519   ...
1520   ...
1521   ...
1522   ...
1523   ...
1524   ...
1525   ...
1526   ...
1527   ...
1528   ...
1529   ...
1530   ...
1531   ...
1532   ...
1533   ...
1534   ...
1535   ...
1536   ...
1537   ...
1538   ...
1539   ...
1540   ...
1541   ...
1542   ...
1543   ...
1544   ...
1545   ...
1546   ...
1547   ...
1548   ...
1549   ...
1550   ...
1551   ...
1552   ...
1553   ...
1554   ...
1555   ...
1556   ...
1557   ...
1558   ...
1559   ...
1560   ...
1561   ...
1562   ...
1563   ...
1564   ...
1565   ...
1566   ...
1567   ...
1568   ...
1569   ...
1570   ...
1571   ...
1572   ...
1573   ...
1574   ...
1575   ...
1576   ...
1577   ...
1578   ...
1579   ...
1580   ...
1581   ...
1582   ...
1583   ...
1584   ...
1585   ...
1586   ...
1587   ...
1588   ...
1589   ...
1590   ...
1591   ...
1592   ...
1593   ...
1594   ...
1595   ...
1596   ...
1597   ...
1598   ...
1599   ...
1600   ...
1601   ...
1602   ...
1603   ...
1604   ...
1605   ...
1606   ...
1607   ...
1608   ...
1609   ...
1610   ...
1611   ...
1612   ...
1613   ...
1614   ...
1615   ...
1616   ...
1617   ...
1618   ...
1619   ...
1620   ...
1621   ...
1622   ...
1623   ...
1624   ...
1625   ...
1626   ...
1627   ...
1628   ...
1629   ...
1630   ...
1631   ...
1632   ...
1633   ...
1634   ...
1635   ...
1636   ...
1637   ...
1638   ...
1639   ...
1640   ...
1641   ...
1642   ...
1643   ...
1644   ...
1645   ...
1646   ...
1647   ...
1648   ...
1649   ...
1650   ...
1651   ...
1652   ...
1653   ...
1654   ...
1655   ...
1656   ...
1657   ...
1658   ...
1659   ...
1660   ...
1661   ...
1662   ...
1663   ...
1664   ...
1665   ...
1666   ...
1667   ...
1668   ...
1669   ...
1670   ...
1671   ...
1672   ...
1673   ...
1674   ...
1675   ...
1676   ...
1677   ...
1678   ...
1679   ...
1680   ...
1681   ...
1682   ...
1683   ...
1684   ...
1685   ...
1686   ...
1687   ...
1688   ...
1689   ...
1690   ...
1691   ...
1692   ...
1693   ...
1694   ...
1695   ...
1696   ...
1697   ...
1698   ...
1699   ...
1700   ...
1701   ...
1702   ...
1703   ...
1704   ...
1705   ...
1706   ...
1707   ...
1708   ...
1709   ...
1710   ...
1711   ...
1712   ...
1713   ...
1714   ...
1715   ...
1716   ...
1717   ...
1718   ...
1719   ...
1720   ...
1721   ...
1722   ...
1723   ...
1724   ...
1725   ...
1726   ...
1727   ...
1728   ...
1729   ...
1730   ...
1731   ...
1732   ...
1733   ...
1734   ...
1735   ...
1736   ...
1737   ...
1738   ...
1739   ...
1740   ...
1741   ...
1742   ...
1743   ...
1744   ...
1745   ...
1746   ...
1747   ...
1748   ...
1749   ...
1750   ...
1751   ...
1752   ...
1753   ...
1754   ...
1755   ...
1756   ...
1757   ...
1758   ...
1759   ...
1760   ...
1761   ...
1762   ...
1763   ...
1764   ...
1765   ...
1766   ...
1767   ...
1768   ...
1769   ...
1770   ...
1771   ...
1772   ...
1773   ...
1774   ...
1775   ...
1776   ...
1777   ...
1778   ...
1779   ...
1780   ...
1781   ...
1782   ...
1783   ...
1784   ...
1785   ...
1786   ...
1787   ...
1788   ...
1789   ...
1790   ...
1791   ...
1792   ...
1793   ...
1794   ...
1795   ...
1796   ...
1797   ...
179
```

生成人类可读的 LLVM 中间表示 (IR)，这是 LLVM 编译器的核心中间语言。

LLVM IR 使用 SSA (静态单赋值) 形式，以 `%` 编号的虚拟寄存器承载所有中间值。

1.1.7 使用 clang++ 生成二进制形式的 LLVM Bitcode

```
parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ clang++-18 -std=c++20 -c -emit-llvm rev_begin.cpp -o rev_begin.bc
parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ file rev_begin.bc
rev_begin.bc: LLVM IR bytecode
parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$
```

Image 7

生成二进制格式的 LLVM Bitcode，可供后续 LTO (链接时优化) 等使用。

1.1.8 使用 clang++ 生成可执行文件，并尝试运行

```
parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ clang++-18 -std=c++20 rev_begin.cpp -o rev_begin_clang
parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ file rev_begin_clang
rev_begin_clang: ELF 64-bit LSB pie executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=bc5fc59fe1bd57d566df3ef397f2647a936f18af, for GNU/Linux 3.7.0, not stripped
parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ echo "AAAAAAAAAAAAAAAAAAAAAAAAAAAA" | ./rev_begin_clang
Please enter the flag: Flag Wrong!
parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$
```

Image 8

使用 clang++ 完整编译生成可执行文件。

运行结果与 g++ 编译版本行为一致。

1.2 Analyze and Tool Use (33%)

1.2.1 1. 逆推 Flag

根据源码中的变换逻辑逆推 flag。 `cmp` lambda 对每个字符的变换为：

```
1 | c = a ^ 0x42          // XOR 0x42
2 | c = c + i             // 加下标
3 | c = ~c               // 按位取反
4 | c = c ^ ((i*7) & 0xFF) // XOR (i*7) 低8位
```

逆向推导 (逆序执行逆操作)：

```
1 | target = [0xFC, 0xFC, 0xF4, 0xD6, 0xF7, 0xAA, 0xEF, 0xEA, 0xEF, 0xF0,
2 |           0xB6, 0xCC, 0x93, 0x8E, 0xB2, 0x12, 0xBB, 0x0C, 0x9E, 0xFC,
3 |           0x42, 0xE9, 0x26, 0xD9, 0xDB, 0x08]
4 |
5 | for i, b in enumerate(target):
6 |     c = b ^ ((i * 7) & 0xFF)    # 逆 XOR
7 |     c = (~c) & 0xFF            # 逆 NOT (注意 8-bit 掩码)
8 |     c = (c - i) & 0xFF         # 逆加法
9 |     a = c ^ 0x42              # 逆 XOR
10 |    print(chr(a), end='')
```

得到 flag: `AAA{R3v_beG1n_c7f101_2o26}`

验证:

```
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ echo "AAA{R3v_beGln_c7f101_2o26}" | ./rev_begin_gcc && echo "AAA{R3v_beGln_c7f101_2o26}" | ./rev_begin_clang
Please enter the flag: Flag Correct!
Please enter the flag: Flag Correct!
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$
```

Image 9

1.2.2 2. 静态分析

使用 IDA Pro 分别对 GCC 和 Clang 编译的两个可执行文件进行静态反编译分析。

1.2.2.1 2.1 `std::views::iota` 的处理对比

GCC — 直接在 main 中构造 `iota_view{0, 26}`，无中间层：

```
if ( std::string::length(v13) == 26 )
{
    *(_OWORD *)v14 = xmmword_1DD0;
    *(_OWORD *)&v14[10] = *(__int128 *)((char *)&xmmword_1DD0 + 10);
    v7 = 1;
    v11 = 0x1A00000000LL;
    v12 = &v11;
    v8 = std::ranges::iota_view<int,int>::begin(&v11);
    v9 = std::ranges::iota_view<int,int>::end(v12);
    while ( (((unsigned __int8)std::ranges::operator==(v8, &v9) ^ 1) & 1) != 0 )
    {
        v10 = std::ranges::iota_view<int,int>::Iterator::operator*(v8);
        v4 = (unsigned __int8 *)std::string::operator[](v13, (int)v10);
        ZNK3cmpMULTyT_50_iRbE_c1hEEDa50_50_i51(&cmp, *v4, (unsigned __int8)v14[v10], v10, &v7);
        std::ranges::iota_view<int,int>::Iterator::operator++(&v8);
    }
    if ( (v7 & 1) != 0 )
        v5 = std::operator<<<std::char_traits<char>>(&std::cout, "Flag Correct!");
    else
        v5 = std::operator<<<std::char_traits<char>>(&std::cout, "Flag Wrong!");
    std::ostream::operator<<(&v5, &std::endl<char,std::char_traits<char>>);
}
```

Image 10

- `0x1A00000000LL`：将两个 `int`（0 和 26）合并为一次 64-bit 写入
- 直接调用 `iota_view::begin()` / `end()`，走标准迭代器模型

Clang — 多了 `views::_Iota` CPO 这一层：

```
24 if ( std::string::length(v18) == 26 )
25 {
26     *(_OWORD *)v16 = xmmword_1A7C;
27     *(_OWORD *)&v16[10] = *(__int128 *)((char *)&xmmword_1A7C + 10);
28     v15 = 1;
29     v12 = 0;
30     v11 = 26;
31     can_iota_viewIT_T0_EEEDa0S3_054 = ZNKSt6ranges5views5_iotaclIiiQsr8__detailE15_can_iota_viewIT_T0_EEEDa0S3_054_(
32         &std::ranges::views::iota,
33         &v12,
34         &v11);
35     p_can_iota_viewIT_T0_EEEDa0S3_054 = &can_iota_viewIT_T0_EEEDa0S3_054;
36     v10 = std::ranges::iota_view<int,int>::begin(&can_iota_viewIT_T0_EEEDa0S3_054);
37     v9 = std::ranges::iota_view<int,int>::end(p_can_iota_viewIT_T0_EEEDa0S3_054);
38     while ( (ZNKSt6ranges9iota_viewIiiE9_IteratorFeqERKS2_S4_Q19equality_comparableIT_E(&v10, &v9) & 1) == 0 )
39     {
40         v8 = std::ranges::iota_view<int,int>::Iterator::operator*(v10);
41         v6 = (unsigned __int8 *)std::string::operator[](v18, (int)v8);
42         $0:operator()<unsigned char>(&cmp, *v6, (unsigned __int8)v16[v8], v8, &v15);
43         std::ranges::iota_view<int,int>::Iterator::operator++(&v10);
44     }
45     if ( (v15 & 1) != 0 )
46     {
47         v5 = std::operator<<<std::char_traits<char>>(&std::cout, "Flag Correct!");
48         std::ostream::operator<<(&v5, &std::endl<char,std::char_traits<char>>);
49     }
50     else
```

Image 11

- 先调 `views::_Iota::operator()`（CPO），内部校验类型后才构造 `iota_view`

- `end()` 返回 `sentinel` 类型，循环用 `IteratorFeq(Iterator, Sentinel)` 比较

方面	GCC	Clang
iota 构造	直接实例化	通过 CPO 间接创建
调用层级	1 层	2 层（多 CPO）
结束判断	<code>operator== sentinel IteratorFeq</code>	

Table 1

1.2.2.2 2.2 Lambda 表达式的实现对比

GCC — 整个 lambda 融合为单行表达式，`i*7` 做强度削减（具体见后文动态调试的反汇编）：

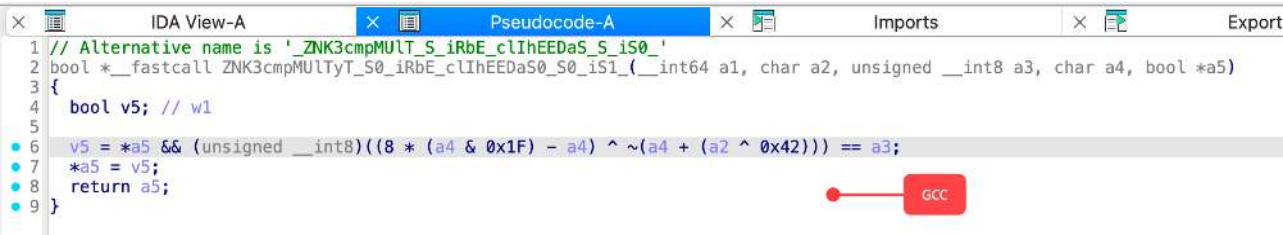


Image 12

```

1 | v5 = *a5 && (unsigned __int8)((8 * (a4 & 0x1F) - a4) ^ ~(a4 + (a2 ^
  | 0x42))) == a3;

```

- `8 * (a4 & 0x1F) - a4` = `i*8 - i` = `i*7`（移位替代乘法）
- `*a5 && (...)` 表达式内短路

Clang — 用 `if` 分支保持结构清晰，`i*7` 直接乘：

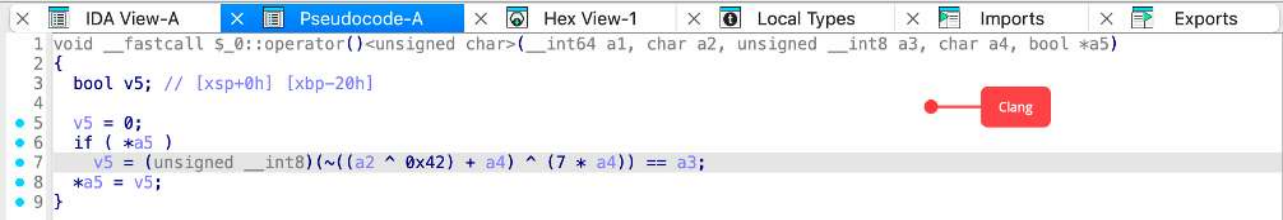


Image 13

```

1 | v5 = 0;
2 | if ( *a5 )
3 |     v5 = (unsigned __int8)(~((a2 ^ 0x42) + a4) ^ (7 * a4)) == a3;
4 | *a5 = v5;

```

- `7 * a4` 直接 `mul`，未做强度削减
- `if (*a5)` 控制流短路；闭包类型命名 `$_0`（GCC 用 `cmp`）

	GCC	Clang
结构	单行融合表达式	<code>if</code> 分支
<code>i*7</code>	<code>i*8-i</code> （强度削减）	<code>7*i</code> （直乘）
命名	<code>cmp:::{lambda}... \$_0::operator()...</code>	

Table 2

两者核心逻辑等价，运行结果一致。

GCC 偏重算术融合与强度削减，Clang 保留更多源码结构。

1.2.3 动态调试 (GDB)

1.2.3.1 进入 Lambda 内部

在 main 的 lambda 调用处 (`main+308`) 设断点，`stepi` 进入：

```
1 (gdb) break *main+308
2 (gdb) continue
3 (gdb) stepi
4 0x0000aaaaaaaa17e8 in cmp::{lambda}::operator()<unsigned char>()
```

`disassemble` 查看 lambda 指令：

```
1 0x...17e8 : sub    sp, sp, #0x30      ← 栈帧 48 字节
2 ...
3 0x...1840 : ubfiz  w0, w0, #3, #5        ← i << 3
4 0x...1844 : sub    w0, w0, w1             ← i*8 - i = i*7
```

GCC 将 `i*7` 优化为 `(i&0x1F)*8 - i`，用移位+减法替代乘法，即静态分析中观察到的强度削减。

```
(gdb) break *main+308
Note: breakpoint 1 also set at pc 0xaaaaaaaa140c.
Breakpoint 2 at 0xaaaaaaaa140c
(gdb) c
Continuing.

Breakpoint 1, 0x0000aaaaaaaa140c in main ()
(gdb) stepi
0x0000aaaaaaaa17e8 in auto cmp::{lambda<typename $T0>($T0, $T0, int, bool&)#1}::operator()<unsigned
char>(unsigned char, unsigned char, int, bool&) const ()
(gdb) disassemble
Dump of assembler code for function ZNK3cmpMULTyT_S0_iRbE_cLIhEEDaS0_S0_iS1_:
=> 0x0000aaaaaaaa17e8 <+0>:      sub    sp, sp, #0x30
0x0000aaaaaaaa17ec <+4>:      str     x0, [sp, #24]
0x0000aaaaaaaa17f0 <+8>:      strb   w1, [sp, #23]
0x0000aaaaaaaa17f4 <+12>:     strb   w2, [sp, #22]
0x0000aaaaaaaa17f8 <+16>:     str     w3, [sp, #16]
0x0000aaaaaaaa17fc <+20>:     str     x4, [sp, #8]
0x0000aaaaaaaa1800 <+24>:     ldrb   w1, [sp, #23]
0x0000aaaaaaaa1804 <+28>:     mov     w0, #0x42          // #66
0x0000aaaaaaaa1808 <+32>:     eor     w0, w1, w0
0x0000aaaaaaaa180c <+36>:     strb   w0, [sp, #47]
0x0000aaaaaaaa1810 <+40>:     ldr     w0, [sp, #16]
0x0000aaaaaaaa1814 <+44>:     and     w0, w0, #0xff
0x0000aaaaaaaa1818 <+48>:     ldrb   w1, [sp, #47]
0x0000aaaaaaaa181c <+52>:     add     w0, w0, w1
0x0000aaaaaaaa1820 <+56>:     strb   w0, [sp, #47]
0x0000aaaaaaaa1824 <+60>:     ldrb   w0, [sp, #47]
0x0000aaaaaaaa1828 <+64>:     mvn     w0, w0
0x0000aaaaaaaa182c <+68>:     strb   w0, [sp, #47]
0x0000aaaaaaaa1830 <+72>:     ldr     w0, [sp, #16]
0x0000aaaaaaaa1834 <+76>:     and     w0, w0, #0xff
0x0000aaaaaaaa1838 <+80>:     mov     w1, w0
0x0000aaaaaaaa183c <+84>:     mov     w0, w1
0x0000aaaaaaaa1840 <+88>:     ubfiz  w0, w0, #3, #5
0x0000aaaaaaaa1844 <+92>:     sub     w0, w0, w1
0x0000aaaaaaaa1848 <+96>:     and     w0, w0, #0xff
0x0000aaaaaaaa184c <+100>:    sxtb   w1, w0
0x0000aaaaaaaa1850 <+104>:    ldrsb  w0, [sp, #47]
```

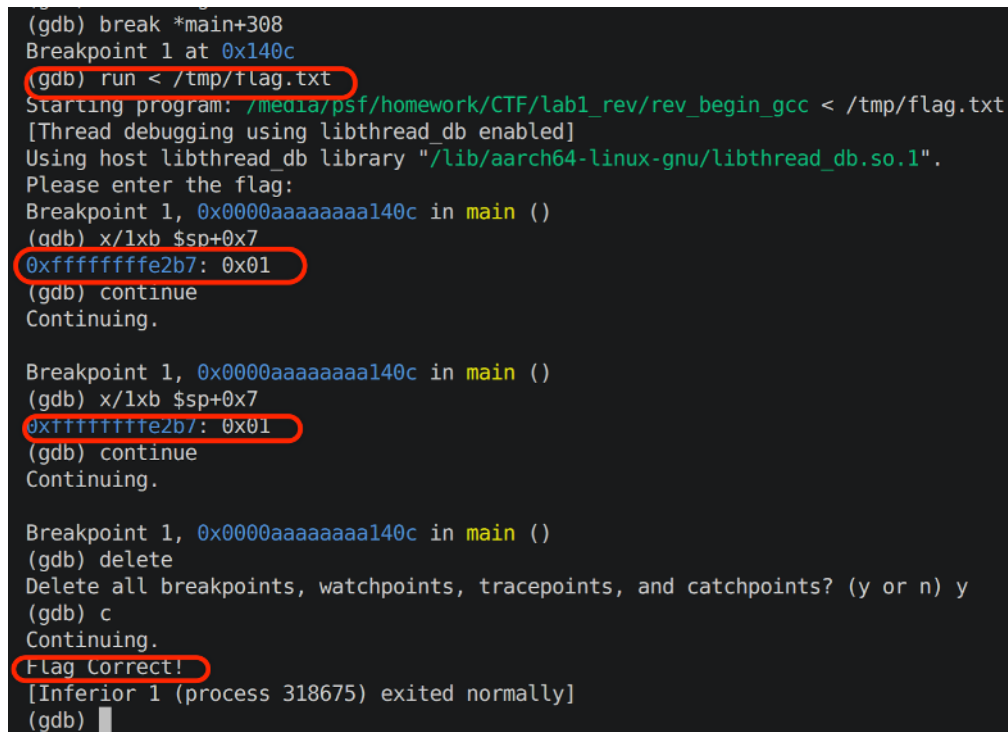
Image 14

1.2.3.2 3.3 正确 vs 错误输入对比

输入正确 flag: 任意循环迭代中查看 `ok` 标志:

```
1 | (gdb) x/1xb $sp+0x7
2 | 0xfffffffffe237: 0x01 ; ok 始终为 true
```

程序输出 `Flag Correct!`



The screenshot shows a GDB terminal session. The user sets a breakpoint at `*main+308` (address `0x140c`). They then run the program `/tmp/flag.txt`. The program prompts for a flag. The user enters `0xfffffffffe2b7: 0x01` (highlighted with a red circle). The program continues, and the user checks the memory at `$sp+0x7`, which shows `0xfffffffffe2b7: 0x01` (highlighted with a red circle). The user continues, and the program outputs `Flag Correct!` (highlighted with a red circle). The program then exits normally.

Image 15

输入错误字符串 `AAAA...`: 同样位置:

```
1 | (gdb) x/1xb $sp+0x7
2 | 0xfffffffffe237: 0x00 ; 第四个'A'时变为 false
```

程序输出 `Flag Wrong!`

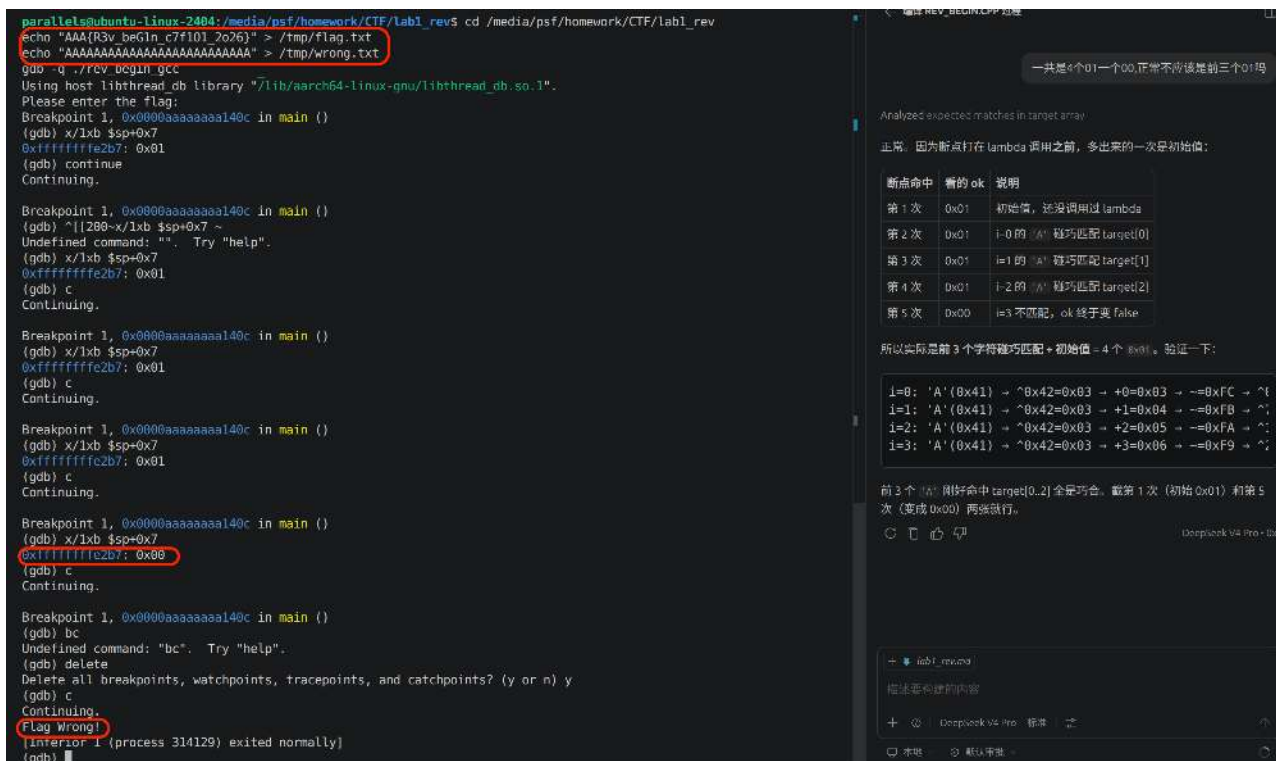


Image 16

ok 初始为 0x01, 一旦某次 cmp 比较失败, ok && (c==b) 短路后 ok 变为 0x00, 后续循环中即使其他字符匹配也无法恢复。

2. Task 2: rev_more (31%)

rev_more 是 Go 1.23 编译的 x86-64 静态链接可执行文件。

采用静态分析。

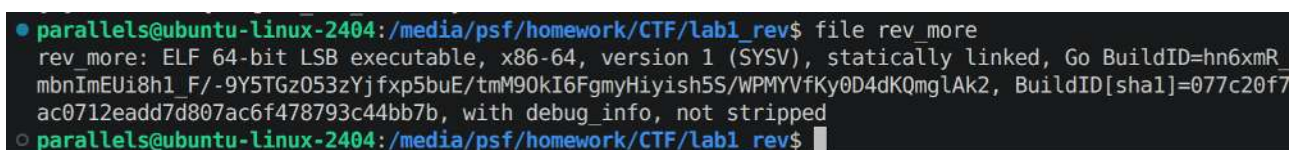


Image 17

2.1 逆向分析过程

2.1.1 提取字符串和符号

用 nm 看符号表, 发现用户代码只有 main.main 一个函数 (其余全是 Go 运行时):

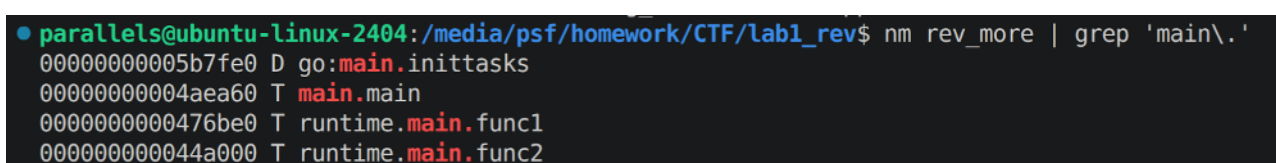


Image 18

用 strings 搜关键字字符串, 定位程序行为:

```

● parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ strings rev_more | grep -i flag
flag
Flag
flag
TFlag
fmtFlags
clearflags
*abi.TFlag
*fmt.fmtFlags
*reflect.flag
*abi.FuncFlag
(scan MB in pacer: % CPU ( zombiegc bits, j0 = head = ,errno=panic: GODEBUG nmsys= locks= dying= a
llocsrax rbx rcx rdx rdi rsi rbp rsp r8 r9 r10 r11 r12 r13
r14 r15 rip rflags cs fs gs signal Signal m->g0= pad1= pad2= minpc= valu
e= (scan) types : type TuesdayJanuaryOctober19531259765625fips140AES-CBCavx512fruntimeos/exect
ls3destlsshlnet/urlflag.txttoStringCTR_DRBGnil PoolscavengepollDescsyncsttesttraceBufdeadlockraceFini

```

Image 19

```

● parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ strings rev_more | grep cryptocustom
morebuf={pc:: no frame (sp=runtime: frame ts set in timertraceback stuckunexpected kindno su
ch processadvertise errornetwork is downno medium foundkey has expired,M3.2.0,M11.1.0476837158203125
invalid bitSizenot in a cgroupincomplete linejstmpmlitinterptarinsecurepathurlstrictcolonsx509keypai
rleafx509usepolicieszipinsecurepath0123456789abcde1cryptocustomrandinteger overflowdecoratemappingsg
cshrinkstackofftracefpunwindoffGC scavenger waitGC worker (idle)page trace flushselect (durable)SIGN0
NE: no trap__vdso_getrandomheap reservation/gc/gogc:percent, not a functiongc: unswept span KiB work
(bg), mheap.sweepden=runtime: nelems=workbuf is emptymSpanList.remove t.insertbad special k
indpage alloc hex编码 and summary dataruntime: addr = runtime: base = runtime: head = already; errno=
containermaxprocs=0,cryptocustomrand=1,decoratemappings=0,tlsscpmlkem=0,tlssha1=1,updatemaxprocs=0,
urlstrictcolons=0,x509sha256skid=0
build DefaultGODEBUG=containermaxprocs=0,cryptocustomrand=1,decoratemappings=0,tlsscpmlkem=0,tlss
ha1=1,updatemaxprocs=0,urlstrictcolons=0,x509sha256skid=0
build DefaultGODEBUG=containermaxprocs=0,cryptocustomrand=1,decoratemappings=0,tlsscpmlkem=0,tlss
ha1=1,updatemaxprocs=0,urlstrictcolons=0,x509sha256skid=0

```

Image 20

2.1.2 安装 Go 工具链反汇编

Go 二进制保留了完整符号和行号信息，用 `go tool objdump` 可反汇编到源码行。

```

parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ /tmp/go/bin/go tool objdump -s 'main.main$' rev_more
main$ rev_more
TEXT main.main(SB) /rev_more.go
rev_more.go:25      0x4aea60      4c8da42460fffff LEAQ 0xfffff60(SP), R12
rev_more.go:25      0x4aea68      4d3b6610      CMPQ R12, 0x10(R14)
rev_more.go:25      0x4aea6c      0f86e3040000  JBE 0x4aef55
rev_more.go:25      0x4aea72      55           PUSHQ BP
rev_more.go:25      0x4aea73      4889e5      MOVQ SP, BP
rev_more.go:25      0x4aea76      4881ec18010000 SUBQ $0x118, SP
rev_more.go:26      0x4aea7d      488d05d4b20300 LEAQ 0x3b2d4(IP), AX
rev_more.go:26      0x4aea84      bb0800000000 MOVL $0x8, BX
rev_more.go:26      0x4aea89      e832defeff   CALL os.ReadFile(SB)
rev_more.go:27      0x4aea8e      4885ff      TESTQ DI, DI
rev_more.go:27      0x4aea91      0f85b1040000 JNE 0x4aef48
rev_more.go:27      0x4aea97      660f1f8400000000 NOPW 0(AX)(AX*1)
rev_more.go:30      0x4aea9a      4885db      TESTQ BX, BX
rev_more.go:30      0x4aea9d      0f848c040000 JE 0x4aef35
rev_more.go:30      0x4aea9f      48f7c30700000000 TESTQ $0x7, BX
rev_more.go:30      0x4eaa02      0f857f040000 JNE 0x4aef35
rev_more.go:30      0x4eaa05      48898c2490000000 MOVQ CX, 0x90(SP)
rev_more.go:30      0x4eaa08      48899c2488000000 MOVQ BX, 0x88(SP)
rev_more.go:30      0x4eaa0b      48898424e8000000 MOVQ AX, 0xe8(SP)
rev_more.go:34      0x4eaa0e      488d44243c   LEAQ 0x3c(SP), AX
rev_more.go:34      0x4eaa11      440f1138   MOVUPS X15, 0(AX)
rev_more.go:35      0x4eaa14      bb1000000000 MOVL $0x10, BX
rev_more.go:35      0x4eaa17      4889d9      MOVQ BX, CX
rev_more.go:35      0x4eaa1a      90         NOPL
rev_more.go:35      0x4eaa1d      e8dbfdffff   CALL crypto/rand.Read(SB)
rev_more.go:36      0x4eaa20      488d05d4b20300 LEAQ 0x3b2d4(IP), AX
rev_more.go:36      0x4eaa23      bb0800000000 MOVL $0x8, BX
rev_more.go:36      0x4eaa26      e832defeff   CALL os.ReadFile(SB)
rev_more.go:36      0x4eaa29      4885ff      TESTQ DI, DI
rev_more.go:36      0x4eaa2c      0f85b1040000 JNE 0x4aef48
rev_more.go:36      0x4eaa2f      660f1f8400000000 NOPW 0(AX)(AX*1)
rev_more.go:39      0x4eaa32      4885db      TESTQ BX, BX
rev_more.go:39      0x4eaa35      0f848c040000 JE 0x4aef35
rev_more.go:39      0x4eaa38      48f7c30700000000 TESTQ $0x7, BX
rev_more.go:39      0x4eaa3b      0f857f040000 JNE 0x4aef35
rev_more.go:39      0x4eaa3e      48898c2490000000 MOVQ CX, 0x90(SP)
rev_more.go:39      0x4eaa41      48899c2488000000 MOVQ BX, 0x88(SP)
rev_more.go:39      0x4eaa44      48898424e8000000 MOVQ AX, 0xe8(SP)
rev_more.go:42      0x4eaa47      488d44243c   LEAQ 0x3c(SP), AX
rev_more.go:42      0x4eaa4a      440f1138   MOVUPS X15, 0(AX)
rev_more.go:43      0x4eaa4d      bb1000000000 MOVL $0x10, BX
rev_more.go:43      0x4eaa4f      4889d9      MOVQ BX, CX
rev_more.go:43      0x4eaa52      90         NOPL
rev_more.go:43      0x4eaa55      e8dbfdffff   CALL crypto/rand.Read(SB)
rev_more.go:43      0x4eaa58      4885db      TESTQ BX, BX
rev_more.go:43      0x4eaa5b      0f853b040000 JNE 0x4aef29
hex.go:127          0x4eaa5e      488d9424c0000000 LEAQ 0xc0(SP), DX
hex.go:127          0x4eaa61      440f113a   MOVUPS X15, 0(DX)

```

Image 21

反汇编 main.main:

```
1 | /tmp/go/bin/go tool objdump -s 'main.main$' rev_more
```

从反汇编结果可以看出，程序先把 flag 用 XTEA 加密，再把密钥和密文 hex 输出。整个流程不比对密码，只需按加密的逆序就能解密。

flag 校验逻辑如下：

源码行	关键指令	逻辑
rev_more.go:26	CALL os.ReadFile	读取 flag.txt
rev_more.go:30	TESTQ \$0x7, BX	长度必须为 8 的倍数
rev_more.go:35	CALL crypto/rand.Read	生成 16 字节随机密钥
rev_more.go:38	CALL fmt.Fprintln	hex 输出密钥 (32 字符)
rev_more.go:41	BSWAP	密钥/flag 字节翻转
rev_more.go:17	CMPQ AX, \$0x6f	111 轮加密循环
rev_more.go:19	LEAL 0x4a537f29	delta=0x4A537F29
rev_more.go:59	CALL fmt.Fprintln	hex 输出密文

Table 3

完整的 flag 处理流程如上表。可以看出密钥是随机的，每次运行都不一样；但加密算法是固定的，把流程反过来就是解密。编写代码，连接题目远程环境并解出 flag (5%)：

```

lab1_rev > solve_rev_more.py > decrypt_block
1  #!/usr/bin/env python3
2  """
3  rev_more - Custom XTEA variant solver.
4
5  Algorithm (from Go binary analysis):
6  - Delta: 0x4a537f29 (custom, not standard XTEA 0x9E3779B9)
7  - Rounds: 111 (0x6f)
8  - Key: 4 x 32-bit words derived from random bytes (BSWAP applied)
9  - Block size: 8 bytes (2 x 32-bit words)
10 - Output: hex(key) + hex(encrypted flag)
11 """
12
13 import struct
14 import websocket
15
16 DELTA = 0x4a537f29
17 ROUNDS = 111
18
19 def bswap32(x):
20     return struct.unpack('>I', struct.pack('<I', x & 0xFFFFFFFF))[0]
21
22 def decrypt_block(v0, v1, key):
23     mask = 0xFFFFFFFF
24     s = (DELTA * ROUNDS) & mask
25     for _ in range(ROUNDS):
26         v1 = (v1 - (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (s + key[(s >> 11) & 3])) & mask
27         s = (s - DELTA) & mask
28         v0 = (v0 - (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (s + key[s & 3])) & mask
29     return v0, v1
30
31 URL = "wss://ctf.zjusec.net/api/proxy/019f9308-763d-7590-826c-01ba335186a6"
32
33 ws = websocket.create_connection(URL, timeout=10)

```

终端输出:

```

• parallels@ubuntu-linux-2404:/media/psf/homework/CTF$ /usr/bin/python3 /media/psf/homework/CTF/lab1_rev/solve_rev_more.py
AAA{xT3@SG0_r3v_1sHard!}
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF$

```

Image 22

3. Task 3: rev_again (27%)

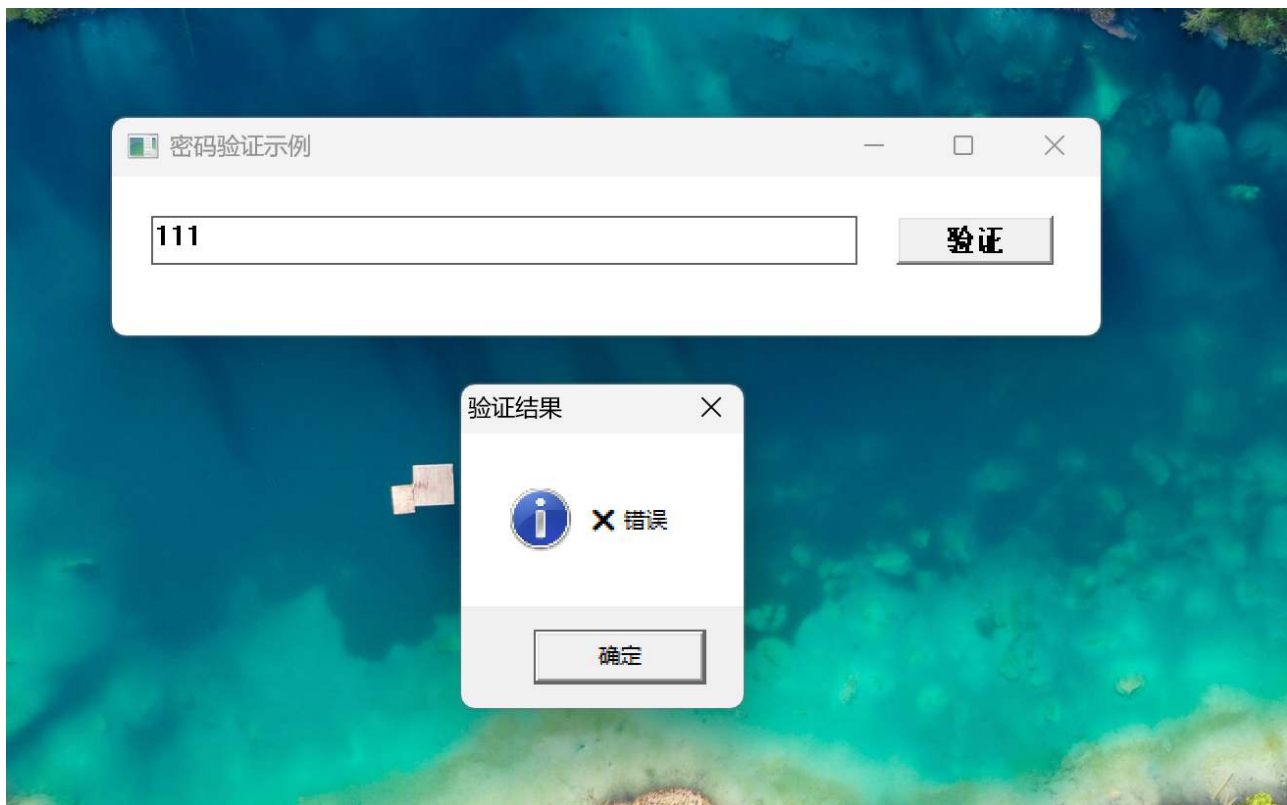


Image 23

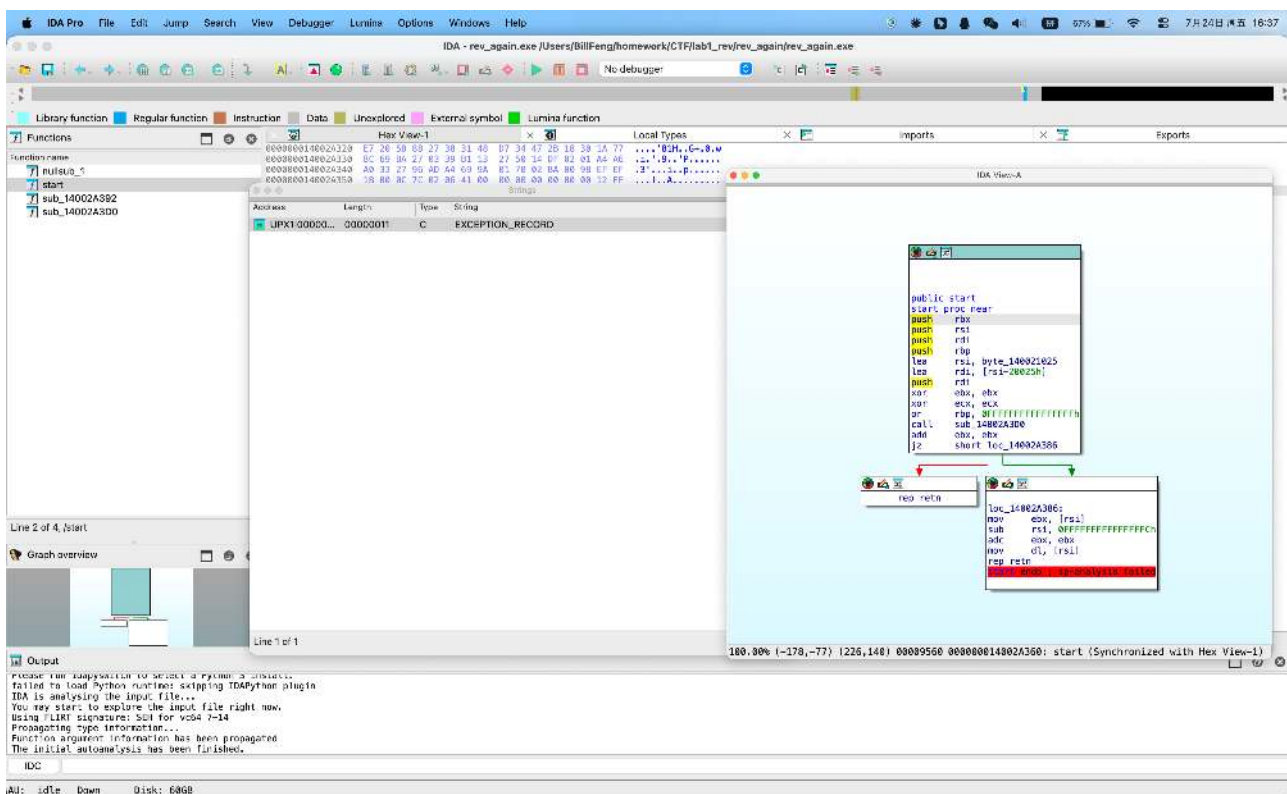


Image 24

IDA 打开后发现空空如也。判断是 UPX 壳，先脱壳：

```
/usr/bin/uxp
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev$ cd /media/psf/homework/CTF/lab1_rev/rev_again && upx -
d rev_again.exe -o rev_again_unpacked.exe 2>&1
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2024
UPX 4.2.2 Markus Oberhumer, Laszlo Molnar & John Reiser Jan 3rd 2024

File size      Ratio      Format      Name
-----
193181 <-    121501    62.89%    win64/pe    rev_again_unpacked.exe

Unpacked 1 file.
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF/lab1_rev/rev_again$
```

Image 25

脱壳后看到 WinMain 创建了一个窗口，核心逻辑在 WindowProc。

```
21  char v20; // [rsp+337h] [rbp+207h]
22  HWND hWnd; // [rsp+358h] [rbp+208h]
23
24  if ( a2 == 273 )
25  {
26      if ( (_DWORD)a3 == 1002 )
27      {
28          hWnd = GetDlgItem(hWndParent, 1001);
29          memset(String, 0, sizeof(String));
30          GetWindowText(hWnd, String, 256);
31          v22 = &v15;
32          std::wstring::basic_string<std::allocator<wchar_t>>(v8, String, &v15);
33          std::_new_allocator<wchar_t>::~__new_allocator(&v15);
34          wstring_to_utf8(v9, v8);
35          v21 = &v16;
36          std::string::basic_string<std::allocator<char>>(v10, "kEy3rd", &v16);
37          std::_new_allocator<char>::~__new_allocator(&v16);
38          encrypt_data(v14, v9, v10);
39          v20 = &v17;
40          v7[0] = &C_8_0;
41          v7[1] = 44;
42          std::vector<unsigned char>::vector(v13, v7, &v17);
43          std::_new_allocator<unsigned char>::~__new_allocator(&v17);
44          v23 = std::operator==<unsigned char, std::allocator<unsigned char>>(v14, v13);
45          v19 = &v18;
46          if ( v23 )
47              v4 = &unk_140006028;
48          else
49              v4 = &unk_140006032;
50          std::wstring::basic_string<std::allocator<wchar_t>>(v11, v4, &v18);
51          std::_new_allocator<wchar_t>::~__new_allocator(&v18);
52          v5 = (const WCHAR *)std::wstring::c_str(v11);
53          MessageBox(hWndParent, v5, &Caption, 0x40u);
54          std::wstring::~wstring(v11);
55          std::vector<unsigned char>::~vector(v13);
56          std::vector<unsigned char>::~vector(v14);
57          std::string::~string(v10);
58          std::string::~string(v9);
59          std::wstring::~wstring(v8);
60      }
61  }
62  else
63  {
64      if ( a2 > 0x111 )
```

Image 26

WindowProc 的流程：按钮点击 → 读输入 → `encrypt_data(input, key="kEy3rd")` → 与硬编码的 `C_8_0` 比较。

```
1  __int64 __fastcall encrypt_data(__int64 a1, __int64 a2, __int64 a3)
2  {
3      int v3; // ebx
4      const unsigned __int8 *v4; // rax
5      __int64 v5; // rax
6      unsigned __int8 *v6; // rbx
7      int v7; // esi
8      const unsigned __int8 *v8; // rax
9      unsigned __int8 v10[263]; // [rsp+20h] [rbp-60h] BYREF
10     char v11; // [rsp+127h] [rbp+A7h] BYREF
11     char *v12; // [rsp+128h] [rbp+A8h]
12
13     v3 = std::string::size(a3);
14     v4 = (const unsigned __int8 *)std::string::c_str(a3);
15     cipher_init(v10, v4, v3);
16     v12 = &v11;
17     v5 = std::string::size(a2);
18     std::vector<unsigned char>::vector(a1, v5, &v11);
19     std::_new_allocator<unsigned char>::~__new_allocator(&v11);
20     v6 = (unsigned __int8 *)std::vector<unsigned char>::data(a1);
21     v7 = std::string::size(a2);
22     v8 = (const unsigned __int8 *)std::string::c_str(a2);
23     cipher_crypt(v10, v8, v7, v6);
24     return a1;
25 }
```

Image 27

`encrypt_data` 内部调用了 `cipher_init` 和 `cipher_crypt` :

- `cipher_init` 是标准 RC4 KSA
- `cipher_crypt` 是 修改版 RC4 $j = (j + S[i] + 1) \& 0xFF$

```
1 unsigned __int8 *__fastcall cipher_init(unsigned __int8 *a1, const unsigned __int8 *a2, int a3)
2 {
3     unsigned __int8 *result; // rax
4     unsigned __int8 v4; // [rsp+3h] [rbp-Dh]
5     int j; // [rsp+4h] [rbp-Ch]
6     int v6; // [rsp+8h] [rbp-8h]
7     int i; // [rsp+Ch] [rbp-4h]
8
9     for ( i = 0; i <= 255; ++i )
10    {
11        result = &a1[i];
12        *result = i;
13    }
14    LOBYTE(v6) = 0;
15    for ( j = 0; j <= 255; ++j )
16    {
17        v6 = (unsigned __int8)(a1[j] + v6 + a2[j % a3]);
18        v4 = a1[j];
19        a1[j] = a1[v6];
20        result = (unsigned __int8 *)v4;
21        a1[v6] = v4;
22    }
23    return result;
24 }
```

Image 28

```
1 int64 __fastcall cipher_crypt(unsigned __int8 *a1, const unsigned __int8 *a2, int a3, unsigned __int8 *a4)
2 {
3     __int64 result; // rax
4     unsigned __int8 v5; // [rsp+3h] [rbp-Dh]
5     unsigned int i; // [rsp+4h] [rbp-Ch]
6     int v7; // [rsp+8h] [rbp-8h]
7     int v8; // [rsp+Ch] [rbp-4h]
8
9     LOBYTE(v8) = 0;
10    LOBYTE(v7) = 0;
11    for ( i = 0; ; ++i )
12    {
13        result = i;
14        if ( (int)i >= a3 )
15            break;
16        v8 = (unsigned __int8)(v8 + 1);
17        v7 = (unsigned __int8)(a1[v8] + v7 + 1);
18        v5 = a1[v8];
19        a1[v8] = a1[v7];
20        a1[v7] = v5;
21        a4[i] = a1[(unsigned __int8)(a1[v8] + a1[v7])] ^ a2[i];
22    }
23    return result;
24 }
```

Image 29

根据逆向结果编写解密脚本 `solve_rev_again.py` , 用修改版 RC4 解密 `C_8_0` :

```

lab1_rev.md solve_rev_again.py solve_rev_more.py
lab1_rev > solve_rev_again.py > ...
1  #!/usr/bin/env python3
2  """
3  rev_again - Modified RC4 solver.
4
5  Algorithm: RC4 variant with extra +1 in PRNG: j = (j + S[i] + 1) & 0xFF
6  Key: "kEy3rd"
7  Encrypted data: C_8_0 (44 bytes)
8  """
9
10 def decrypt(key, data):
11     S = list(range(256))
12     j = 0
13     for i in range(256):
14         j = (j + S[i] + key[i % len(key)]) & 0xFF
15         S[i], S[j] = S[j], S[i]
16     i = j = 0
17     out = []
18     for c in data:
19         i = (i + 1) & 0xFF
20         j = (j + S[i] + 1) & 0xFF
21         S[i], S[j] = S[j], S[i]
22         out.append(c ^ S[(S[i] + S[j]) & 0xFF])
23     return bytes(out)
24
25 enc = bytes([
26     0x50, 0xF7, 0x8C, 0xF9, 0x80, 0x59, 0xA8, 0x77,
27     0x85, 0xCA, 0xCB, 0x89, 0x6F, 0x56, 0x1A, 0x6F,
28     0x2F, 0x24, 0xFD, 0x84, 0x67, 0x8E, 0x8F, 0x36,
29     0xD4, 0x37, 0x7F, 0x36, 0xDB, 0xD0, 0xEC, 0xC9,
30     0xEC, 0xD9, 0x04, 0x03, 0x9C, 0xED, 0x53, 0x90,
31     0xB5, 0xFD, 0xE5, 0x12])
32
33 flag = decrypt(b"kEy3rd", enc)

```

问题 输出 调试控制台 终端 端口

```

• parallels@ubuntu-linux-2404:/media/psf/homework/CTF$ /usr/bin/python3 /media/psf/homework/CTF/lab1_rev/solve_rev_agai
n.py
AAA{#rd_ch@I 1N wln32aPi&Rc$ canyousolveit}
• parallels@ubuntu-linux-2404:/media/psf/homework/CTF$

```

bash
bash rev_again
Python

Image 30

[lab 0] asm tour 100 pts 66 次解出	[lab 0] crackme 100 pts 86 次解出	[lec 1] rev_again 100 pts 70 次解出
[lec 1] rev_begin 100 pts 84 次解出	[lec 1] rev_more 100 pts 70 次解出	[lec2] my_vm_example 60 pts 9 次解出

Image 31