CTF-攻防世界 Reverse新手练习解析

原创

 F1ash000
 ● 10457
 ● 收藏 48

 分类专栏:
 # CTF 信息安全 文章标签:
 CTF Reverse 新手小白

 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/weixin_42621117/article/details/99768988

 版权



CTF 同时被 2 个专栏收录

2 篇文章 1 订阅 订阅专栏



3 篇文章 0 订阅 订阅专栏

博主也是CTF小白,入门ing。。。方向是RE+PWN。文章可能多有纰漏,但会持续更新更正。希望大家多多指出不足之处。

0x1. re1

这道题很简单:

打开.exe随便输点东西进去,发现不对,退出。

用IDA打开,点到十六进制视图

点IDA视图--打开子视图--字符串(英文版IDA应该就是view这种常见的单词)。或者直接按shift+F12。然后在一大堆东西中找到 这个: flag get

1	IDA V	/iew-A	×	's'	字符串窗口	×	O	十六进制视图-1	×	A	结构体
地	址	长度	类型	字符串							
's'	.rdata:0…	000000C	С	MessageE	BoxW						
's'	.rdata:0…	00000010	С	GetActiv	veWindow						
's'	.rdata:0…	0000013	С	GetLastA	ctivePopup						
's'	.rdata:0…	000001A	С	GetUserC	bjectInformation	W					
's'	.rdata:0…	0000018	С	GetProce	ssWindowStation						
's'	.rdata:0…	0000005	С	\a\b\t\n	ı\v						
's'	.rdata:0…	00000061	С	!\"#\$%&	a' ()*+, /0123456	789:;<=	=>?@ABC	DEFGHIJKLMNOPQRSTU	JVWXYZ[\\	\]^ …	
's'	.rdata:0…	0000005	С	\a\b\t\n	ı\v						
's'	.rdata:0…	00000060	С	!\"#\$%&	a' ()*+, /0123456	789:;<=	=>?@abc	defghijklmnopqrstu	ıvwxyz[\\	\]^ …	
's'	.rdata:0…	0000005	С	\a\b\t\n	ı\v						
's'	.rdata:0…	00000060	С	!\″#\$%&	a' ()*+, /0123456	789:;<=	=>?@ABC	DEFGHIJKLMNOPQRSTU	JVWXYZ[\\	\]^ …	
's'	.rdata:0…	0000006	С	e+000							
's'	.rdata:0…	0000007	С	1#SNAN							
's'	.rdata:0…	0000006	С	1#IND							
's'	.rdata:0…	0000006	С	1#INF							
\mathbf{s}	.rdata:0…	0000007	С	1#QNAN							
's'	.rdata:0…	00000012	С	欢迎来到	DUTCTF呦\n						
\mathbf{s}	.rdata:0…	0000020	С	这是一道	很可爱很简单的逆	向题呦\	n				
s'	.rdata:0…	000000C	С	输入flag	;吧:						
\mathbf{s}^{\prime}	.rdata:0…	80000008	С	flag get	;						
's'	.rdata:0…	000000C	С	flag不太	对呦						
's'	.rdata:0…	0000006	С	油呦∖n							
's'	.rdata:0…	0000006	С	pause							
's'	.rdata:0…	000000D	С	KERNEL32	2. d11						

英文应该都能看懂吧……这就是关键 双击,跳到十六进制视图窗口就可轻松获得flag: DUTCTF{We1c0met0DUTCTF}

0x2.game

n是灯的序列号,m是灯的状态 如果第N个灯的m为1,则它打开,如果不是,则关闭 起初所有的灯都关闭了 现在您可以输入n来更改其状态 但是你应该注意一件事,如果改变第N盏灯的状态,第(N-1)和第(N+1)的状态也会改变 当所有灯都亮起时,将出现标志 现在,输入n (来自谷歌翻译……QAQ) 解析: 依旧什么都不用管, 直接拖到IDA打开 shift+F12 Alt+T(搜索字符串),搜索: flag 直接跳出来: done!!!the flag is 双击,跳到IDA View-A(这里说一下,字符串窗口双击跳转的窗口是打开字符串窗口时停留的窗口。也就是说,当你页面停在 IDA View-A时,你打开了字符串窗口,那在字符串窗口双击,就跳转到IDA View-A) Ctrl+X (交叉引用) F5(生成伪代码) 加下图

<pre>107 char v105; // [esp+14Fh] [ebp-16h] 108 char v106; // [esp+15h] [ebp-15h] 109 char v108; // [esp+15h] [ebp-14h] 110 char v108; // [esp+15h] [ebp-13h] 111 char v110; // [esp+15h] [ebp-16h] 112 char v111; // [esp+15ch] [ebp-16h] 113 char v111; // [esp+15ch] [ebp-16h] 114 char v112; // [esp+15ch] [ebp-16h] 115 char v113; // [esp+15ch] [ebp-16h] 116 char v114; // [esp+15ch] [ebp-16h] 117 char v115; // [esp+15ch] [ebp-16h] 118 119 sub_45A7RE("done!!! the flag is "); 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 133 v72 = 12; 134 v72 = 4e: 235 v55 = 0; 234 for (i = 0; i < 56; ++i) 235 { *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 239 return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968) </pre>		IDA View-A	×	▣ 伪作	弋码	×	's'	字符串窗口		٥	十六进制
<pre>108 char v106; // [esp+14Fh] [ebp-13h] 109 char v107; // [esp+15h] [ebp-13h] 111 char v109; // [esp+15h] [ebp-12h] 112 char v110; // [esp+15h] [ebp-11h] 113 char v111; // [esp+15h] [ebp-16h] 114 char v112; // [esp+15h] [ebp-16h] 115 char v113; // [esp+15h] [ebp-ch] 116 char v113; // [esp+15h] [ebp-ch] 117 char v115; // [esp+15h] [ebp-ch] 118 119 <u>sub_45A78E("done!!! the flag is ");</u> 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v57 = 6; 129 v68 = 48; 130 v59 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 133 v72 = 12; 134 v73 = A8: 250 v55 = 117; 235 v56 = 117; 236 *(&v2 + i) ^= *(&v59 + i); 37 *(&v2 + i) ^= *(&v59 + i); 38 } 129 return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968) 00007D68 sub_45E940:211 (45E968)</pre>	107	char v105;	//	[esp+14Eh]	[ebp-	16h]					
<pre>109 char v100; // [esp+150h] [ebp-13h] 110 char v100; // [esp+151h] [ebp-13h] 111 char v110; // [esp+151h] [ebp-13h] 112 char v111; // [esp+151h] [ebp-10h] 113 char v111; // [esp+151h] [ebp-10h] 114 char v111; // [esp+151h] [ebp-0h] 117 char v115; // [esp+151h] [ebp-0h] 118 sub_45A78E("done!!! the flag is "); 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 133 v72 = 12; 133 v72 = 12; 133 v56 = 117; 134 v73 = 48: 145 v57 = 6; 155 v58 = 0; 157 v50 = 1; 158 v57 = 126; 159 v58 = 0; 159 v58 = 0; 150 v59 = 0; 150 v50 = 1; 150 v50 = 1; 151 v56 = 117; 152 v57 = 126; 153 v58 = 0; 153 v58 = 0; 154 for (i = 0; i < 56; ++i) 155 { 152 *(&v2 + i) ^= *(&v59 + i); 152 *(&v2 + i) ^= 0x13u; 153 } 159 return sub_45A7BE("%s\n"); 150 v50 v50 v50 v50 v50 v50 v50 v50 v50 v</pre>	108	char v106;	11	[esp+14Fh]	[ebp-	15h]					
<pre>110 char v108; // [esp+151] [ebp-131] 111 char v109; // [esp+152] [ebp-12h] 112 char v110; // [esp+153h] [ebp-10h] 113 char v111; // [esp+155h] [ebp-Fh] 114 char v112; // [esp+158h] [ebp-Ch] 115 char v113; // [esp+158h] [ebp-Ch] 116 char v114; // [esp+158h] [ebp-Ch] 117 char v115; // [esp+158h] [ebp-Ch] 118 119 sub_48A7EE("done!!! the flag is "); 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v59 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 134 v73 = 4e: 144 v73 = 4e: 155 the form (i = 0; i < 56; ++i) 125 { 126 *(&v2 + i) ^= *(&v59 + i); 127 *(&v2 + i) ^= *(&v59 + i); 128 } 129 return sub_45A7BE("%s\n"); 129 00007D68 sub_45E940:211 (45E968)</pre>	109	char v107;	11	[esp+150h]	[ebp-	14h]					
<pre>111 clian v109; // [esp+153h] [ebp-11h] 112 char v111; // [esp+153h] [ebp-11h] 113 char v111; // [esp+153h] [ebp-1h] 114 char v112; // [esp+153h] [ebp-1h] 115 char v113; // [esp+153h] [ebp-1h] 116 char v114; // [esp+153h] [ebp-1h] 117 char v115; // [esp+153h] [ebp-1h] 118 119 sub_45A78E("done!!! the flag is "); 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 134 v72 = 12; 135 v58 = 0; 134 v73 = 48. 135 v58 = 0; 135 v58 = 0; 136 *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 138 } 239 return sub_45A7BE("%s\n"); 240</pre>	110	char V108;		[esp+151n]	[ebp-	13N] 1261					
<pre>char vilis; // [esp+15sh] [ebp-18h] 113 char vili; // [esp+15sh] [ebp-18h] 114 char vili; // [esp+15sh] [ebp-18h] 115 char vili; // [esp+15sh] [ebp-0h] 117 char vili; // [esp+15sh] [ebp-0h] 118 119 sub_45A7RE("done!!! the flag is "); 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 122 v62 = 5; 124 v63 = 2; 125 v64 = 44; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; v56 = 117; 231 v56 = 117; 232 v57 = 126; 233 v58 = 0; 134 for (i = 0; i < 56; ++i) 135 { *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 138 } 129 v68 sub_45E940:211 (45E968)</pre>	112	char $v109$,	11	$\left[e_{sp+152h}\right]$	[ebp-	120) 116]					
<pre>114 char v112; // [esp+155h] [ebp-Fh] 115 char v113; // [esp+156h] [ebp-Fh] 116 char v114; // [esp+157h] [ebp-Ch] 117 char v115; // [esp+158h] [ebp-Ch] 118 119 sub_45A7BE("done!!! the flag is "); 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v59 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; v56 = 117; 134 v73 = 40: 235 v57 = 126; 233 v58 = 0; 234 for (i = 0; i < 56; ++i) 235 { *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 238 } 239 return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968)</pre>	113	char v111:	11	[esp+154h]	[ebp-	10h]					
<pre>115 char v113; // [esp+156h] [ebp-Eh] 116 char v114; // [esp+157h] [ebp-Dh] 117 char v115; // [esp+158h] [ebp-Ch] 118 119 sub_45A7BE("done!!! the flag is "); 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 133 v72 = 12; 134 v73 = 48: v56 = 117; 235 v56 = 117; 236 v58 = 0; 234 for (i = 0; i < 56; ++i) 235 { *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 238 } return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968)</pre>	114	char v112;	11	[esp+155h]	[ebp-	Fh]					
<pre>116 char v114; // [esp+157h] [ebp-Dh] 117 char v115; // [esp+158h] [ebp-Ch] 118 119 sub_45A78E("done!!! the flag is "); 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 134 v72 = 40: 135 v56 = 117; 136 v56 = 117; 137 v56 = 117; 138 v58 = 0; 139 v58 = 0; 139 v58 = 0; 139 v58 = 0; 139 v58 = 0; 131 v70 (i = 0; i < 56; ++i) 135 { 139 *(&v2 + i) ^= *(&v59 + i); 130 *(&v2 + i) ^= 0x13u; 139 return sub_45A7BE("%s\n"); 130 v50 sub_45E940:211 (45E968) 00007D68 sub_45E940:211 (45E968)</pre>	115	char v113;	11	[esp+156h]	[ebp-	Eh]					
<pre>117 char v115; // [esp+158h] [ebp-Ch] 118 119 sub_45A7BE("done!!! the flag is "); 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 133 v72 = 12; 134 v73 - 48: 129 v55 = 1; 124 v55 = 117; 123 v55 = 117; 123 v58 = 0; 123 v58 = 0; 124 v58 = 0; 125 { 125 { 126 v55 = 126; 127 v56 = 117; 128 v57 = 126; 129 v58 = 0; 129 v58 = 0; 129 v58 = 0; 120 v58 = 0; 120 v58 = 0; 121 v56 = 117; 123 v58 = 0; 123 v58 = 0; 124 v57 = 126; 125 { 125 v57 = 126; 126 v56 ; 127 v56 = 117; 127 v56 = 117; 128 v57 = 126; 129 v58 = 0; 129 v58 = 0; 129 v58 = 0; 120 v58 = 0; 120 v58 = 0; 120 v58 = 0; 121 v56 = 117; 122 v57 = 126; 123 v58 = 0; 123 v58 = 0; 124 v58 = 0; 125 v54 v56 v56 v56 v56 v56 v56 v56 v56 v56 v56</pre>	116	char v114;	//	[esp+157h]	[ebp-	Dh]					
<pre>118 119 sub_45A7BE("done!!! the flag is "); 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 133 v72 = 12; 134 v73 = 48: 235 v56 = 117; 232 v57 = 126; 233 v58 = 0; 234 for (i = 0; i < 56; ++i) 235 { 236 *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 238 } 239 return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968)</pre>	117	char v115;	//	[esp+158h]	[ebp-	Ch]					
<pre>119 Sub_asyrbit done:: the fing is f; 120 v59 = 18; 121 v60 = 64; 122 v61 = 98; 123 v62 = 5; 124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 133 v72 = 12; 134 v73 = 48: 234 v56 = 117; 235 v58 = 0; 234 for (i = 0; i < 56; ++i) 235 { 236 *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 238 } 239 return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968)</pre>	118		<u>("</u>			- "\.					
<pre>120</pre>	119	$\frac{SUD_{45A/BE}}{18}$	<mark>(</mark> ao	ne!!! the -	riag i	s);					
<pre>1122 V61 = 98; 1122 V62 = 5; 1124 V63 = 2; 1125 V64 = 4; 1126 V65 = 6; 1127 V66 = 3; 1128 V67 = 6; 1129 V68 = 48; 1130 V69 = 49; 1131 V70 = 65; 1132 V71 = 32; 1133 V72 = 12; 1134 V73 = 48: 1134 V73 = 48: 1136 V56 = 117; 1237 V56 = 117; 1238 V58 = 0; 1234 for (i = 0; i < 56; ++i) 235 { 1236 *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 1237 *(&v2 + i) ^= 0x13u; 1238 } 1239 return sub_45A7BE("%s\n"); 1240 } 00007D68 sub_45E940:211 (45E968)</pre>	120121	$v_{59} = 10$, $v_{60} = 64$:									
123 $\sqrt{62} = 5;$ 124 $\sqrt{63} = 2;$ 125 $\sqrt{64} = 4;$ 126 $\sqrt{65} = 6;$ 127 $\sqrt{66} = 3;$ 128 $\sqrt{67} = 6;$ 129 $\sqrt{68} = 48;$ 130 $\sqrt{69} = 49;$ 131 $\sqrt{70} = 65;$ 132 $\sqrt{71} = 32;$ 133 $\sqrt{72} = 12;$ 133 $\sqrt{72} = 12;$ 134 $\sqrt{73} = 40;$ 134 $\sqrt{56} = 117;$ 135 $\sqrt{56} = 117;$ 136 $\sqrt{56} = 117;$ 137 $\sqrt{56} = 0;$ 138 $\sqrt{58} = 0;$ 139 $\sqrt{58} = 0;$ 130 $\sqrt{58} = 0;$ 131 $\sqrt{56} = 117;$ 132 $\sqrt{57} = 126;$ 133 $\sqrt{58} = 0;$ 134 $\sqrt{57} = 126;$ 135 $\sqrt{58} = 0;$ 135 $\sqrt{58} = 0;$ 136 $\sqrt{58} = 0;$ 137 $\sqrt{6} (\sqrt{59} + 1);$ 138 $\sqrt{58} = 0;$ 139 $\sqrt{58} = 0;$ 139 $\sqrt{58} = 0;$ 130 $\sqrt{58} = 0;$ 130 $\sqrt{58} = 0;$ 131 $\sqrt{56} = 117;$ 132 $\sqrt{57} = 126;$ 133 $\sqrt{58} = 0;$ 134 $\sqrt{56} = 117;$ 135 $\sqrt{56} = 117;$ 135 $\sqrt{56} = 117;$ 136 $\sqrt{56} = 117;$ 137 $\sqrt{56} = 117;$ 138 $\sqrt{56} = 117;$ 139 $\sqrt{56} = 117;$ 130 $\sqrt{56} = 117;$ 130 $\sqrt{56} = 117;$ 131 $\sqrt{56} = 117;$ 132 $\sqrt{57} = 126;$ 133 $\sqrt{58} = 0;$ 134 $\sqrt{56} = 117;$ 135 $\sqrt{56} = 117;$ 136 $\sqrt{56} = 117;$ 137 $\sqrt{56} = 117;$ 138 $\sqrt{56} = 117;$ 139 $\sqrt{56} = 117;$ 130 $\sqrt{56} = 117;$ 130 $\sqrt{56} = 117;$ 131 $\sqrt{56} = 117;$ 132 $\sqrt{57} = 126;$ 133 $\sqrt{58} = 0;$ 134 $\sqrt{56} = 117;$ 135 $\sqrt{56} = 117;$ 135 $\sqrt{56} = 117;$ 136 $\sqrt{56} = 117;$ 137 $\sqrt{56} = 117;$ 138 $\sqrt{56} = 117;$ 139 $\sqrt{56} = 117;$ 130 $\sqrt{56} = 117;$ 130 $\sqrt{56} = 117;$ 131 $\sqrt{56} = 117;$ 132 $\sqrt{57} = 126;$ 133 $\sqrt{56} = 117;$ 134 $\sqrt{56} = 117;$ 135 $\sqrt{56} = 117;$ 136 $\sqrt{56} = 117;$ 137 $\sqrt{56} = 117;$ 138 $\sqrt{56} = 117;$ 139 $\sqrt{56} = 117;$ 130 $\sqrt{56} = 117;$ 130 $\sqrt{56} = 117;$ 131 $\sqrt{56} = 117;$ 132 $\sqrt{57} = 126;$ 133 $\sqrt{56} = 117;$ 134 $\sqrt{56} = 117;$ 135 $\sqrt{56} = 117;$ 137 $\sqrt{56} = 117;$ 138 $\sqrt{56} = 117;$ 139 $\sqrt{56} = 117;$ 130 $\sqrt{56} = 117;$ 131 $\sqrt{56} = 117;$ 131 $\sqrt{56} = 117;$ 132 $\sqrt{56} = 117;$ 133 $\sqrt{56} = 117;$ 134 $\sqrt{56} = 117;$ 135 $\sqrt{56} = 117;$ 135 $\sqrt{56} = 117;$ 136 $\sqrt{56} = 117;$ 137 $\sqrt{56} = 117;$ 138 $\sqrt{56} = 117;$ 138	122	v61 = 98;									
<pre>124 v63 = 2; 125 v64 = 4; 126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 134 v73 = 40: 235 v56 = 117; 236 v56 = 117; 237 v56 = 0; 234 for (i = 0; i < 56; ++i) { 235 { 236 *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 238 } 239 return sub_45A7BE("%s\n"); 240}</pre>	• 123	v62 = 5;									
125 $\sqrt{64} = 4;$ 126 $\sqrt{65} = 6;$ 127 $\sqrt{66} = 3;$ 128 $\sqrt{67} = 6;$ 129 $\sqrt{68} = 48;$ 130 $\sqrt{69} = 49;$ 131 $\sqrt{70} = 65;$ 132 $\sqrt{71} = 32;$ 133 $\sqrt{72} = 12;$ 134 $\sqrt{72} = 4e$. 231 $\sqrt{56} = 117;$ 232 $\sqrt{57} = 126;$ 233 $\sqrt{58} = 0;$ 234 for (i = 0; i < 56; ++i) 235 { 236 $*(\&v2 + i) \land = *(\&v59 + i);$ $*(\&v2 + i) \land = 0x13u;$ 237 $return sub_45A7BE("%s\n");$ 240 } 00007D68 sub_45E940:211 (45E968)	• 124	v63 = 2;									
126 v65 = 6; 127 v66 = 3; 128 v67 = 6; 129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 134 v73 = 48: v56 = 117; v56 = 117; v57 = 126; v58 = 0; 234 for (i = 0; i < 56; ++i) 235 { *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 238 } return sub_45A7BE("%s\n"); 240}	0 125	v64 = 4;									
$127 V66 = 3;$ $128 v67 = 6;$ $129 v68 = 48;$ $130 v69 = 49;$ $131 v70 = 65;$ $132 v71 = 32;$ $133 v72 = 12;$ $133 v72 = 12;$ $134 v73 = 48;$ $V55 = 117;$ $232 v57 = 126;$ $v57 = 126;$ $v58 = 0;$ $234 for (i = 0; i < 56; ++i)$ $235 \{$ $*(&v2 + i) \ ^{=} *(&v59 + i);$ $237 *(&v2 + i) \ ^{=} 0x13u;$ $238 \}$ $return sub_45A7BE("%s\n");$ $240 \}$ $00007D68 sub_45E940:211 (45E968)$	• 126	v65 = 6;									
<pre>129 v68 = 48; 130 v69 = 49; 131 v70 = 65; 132 v71 = 32; 133 v72 = 12; 134 v73 = 48: 231 v56 = 117; 232 v57 = 126; 233 v58 = 0; 234 for (i = 0; i < 56; ++i) 235 { 236 *(&v2 + i) ^= *(&v59 + i); 237 *(&v2 + i) ^= 0x13u; 238 } 239 return sub_45A7BE("%s\n"); 240}</pre>	12/	V66 = 3;									
130 130 131 131 132 132 132 133 132 133 133	120	$v_{68} = 48$									
131 $\sqrt{70} = 65;$ 132 $\sqrt{71} = 32;$ 133 $\sqrt{72} = 12;$ 134 $\sqrt{72} = 48.$ 135 $\sqrt{72} = 48.$ 136 $\sqrt{73} = 48.$ 137 $\sqrt{56} = 117;$ 138 $\sqrt{56} = 117;$ 139 $\sqrt{56} = 117;$ 139 $\sqrt{57} = 126;$ 139 $\sqrt{58} = 0;$ 139 $\sqrt{58} = 0;$ 139 $\sqrt{58} = 0;$ 130 $\sqrt{58} = 0;$ 130 $\sqrt{58} = 0;$ 131 $\sqrt{58} = 0;$ 131 $\sqrt{56} = 117;$ 132 $\sqrt{57} = 126;$ 133 $\sqrt{58} = 0;$ 133 $\sqrt{58} = 0;$ 134 for (i = 0; i < 56; ++i) 135 { 136 $\sqrt{60} + 1;$ 137 $\sqrt{60} + 1;$ 138 $\sqrt{60} + 1;$ 138 $\sqrt{60} + 1;$ 139 return sub_45A7BE("%s\n"); 139 $\sqrt{60}$ 130 $\sqrt{272} + 1;$ 130 $\sqrt{60} + 1;$ 130 $\sqrt{60} + 1;$ 131 $\sqrt{60} + 1;$ 131 $\sqrt{60} + 1;$ 132 $\sqrt{60} + 1;$ 133 $\sqrt{72} + 1;$ 134 $\sqrt{60} + 1;$ 135 $\sqrt{60} + 1;$ 136 $\sqrt{60} + 1;$ 137 $\sqrt{60} + 1;$ 138 $\sqrt{60} + 1;$ 139 $\sqrt{60} + 1;$ 139 $\sqrt{60} + 1;$ 130 $\sqrt{60} + 1;$ 130 $\sqrt{60} + 1;$ 130 $\sqrt{60} + 1;$ 131 $\sqrt{60} + 1;$ 131 $\sqrt{60} + 1;$ 132 $\sqrt{60} + 1;$ 133 $\sqrt{60} + 1;$ 134 $\sqrt{60} + 1;$ 135 $\sqrt{60} + 1;$ 135 $\sqrt{60} + 1;$ 136 $\sqrt{60} + 1;$ 137 $\sqrt{60} + 1;$ 138 $\sqrt{60} + 1;$ 138 $\sqrt{60} + 1;$ 139 $\sqrt{60} + 1;$ 139 $\sqrt{60} + 1;$ 130 $\sqrt{60} + 1;$ 130 $\sqrt{60} + 1;$ 130 $\sqrt{60} + 1;$ 131 $\sqrt{60} + 1;$ 131 $\sqrt{60} + 1;$ 132 $\sqrt{60} + 1;$ 133 $\sqrt{60} + 1;$ 134 $\sqrt{60} + 1;$ 135 $\sqrt{60} + 1;$ 135 $\sqrt{60} + 1;$ 136 $\sqrt{60} + 1;$ 137 $\sqrt{60} + 1;$ 138 $\sqrt{60} + 1;$ 139 $\sqrt{60} + 1;$ 130 $\sqrt{60} + 1;$	120130	v69 = 49;									
132 133 $\sqrt{72} = 32;$ 133 $\sqrt{72} = 12;$ 134 $\sqrt{72} = 48:$ 134 $\sqrt{72} = 48:$ 135 $\sqrt{56} = 117;$ 232 $\sqrt{56} = 117;$ 232 $\sqrt{57} = 126;$ 233 $\sqrt{58} = 0;$ 234 for (i = 0; i < 56; ++i) 235 { 236 *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 237 *(&v2 + i) ^= 0x13u; 238 } return sub_45A7BE("%s\n"); 240 }	• 131	v70 = 65;									
133 $\sqrt{72} = 12;$ $\sqrt{73} = 48$ 231 $\sqrt{56} = 117;$ 232 $\sqrt{57} = 126;$ 233 $\sqrt{58} = 0;$ 234 for (i = 0; i < 56; ++i) 235 { 236 $*(\&v2 + i) ^{=} *(\&v59 + i);$ $*(\&v2 + i) ^{=} 0x13u;$ 237 $*(\&v2 + i) ^{=} 0x13u;$ 238 } 00007D68 sub_45E940:211 (45E968); $\sqrt{6}$	• 132	v71 = 32;									
<pre>134</pre>	• 133	v72 = 12;									
<pre>230 v55 = 1; 231 v56 = 117; 232 v57 = 126; 233 v58 = 0; 234 for (i = 0; i < 56; ++i) 235 { 236 *(&v2 + i) ^= *(&v59 + i); *(&v2 + i) ^= 0x13u; 238 } 239 return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968)</pre>	▲ 1 ⊋ /	1/72 - ΛQ·									
231 $v56 = 117;$ 232 $v57 = 126;$ 233 $v58 = 0;$ 234 for (i = 0; i < 56; ++i) 235 { 236 $*(\&v2 + i) \land = *(\&v59 + i);$ 237 $*(\&v2 + i) \land = 0x13u;$ 238 } 239 return sub_45A7BE("%s\n"); 240}	1 1 2 2 2	V55 = 13	-								
<pre>231 v30 = 117; 232 v57 = 126; 233 v58 = 0; 234 for (i = 0; i < 56; ++i) 235 { 236 *(&v2 + i) ^= *(&v59 + i); 237 *(&v2 + i) ^= 0x13u; 238 } 239 return sub_45A7BE("%s\n"); 240}</pre>	230	$v_{56} - 11$, 17.								
<pre>232 v57 = 120, v58 = 0; 234 for (i = 0; i < 56; ++i) { 235 { 236 *(&v2 + i) ^= *(&v59 + i); 237 *(&v2 + i) ^= 0x13u; 238 } 239 return sub_45A7BE("%s\n"); 240}</pre>	221	$v_{50} = 1$, , ,								
<pre>> 233</pre>		$\sqrt{57} = 12$	20 ,								
<pre>234 for (1 = 0; 1 < 56; ++1) 235 { 236</pre>	233	V58 = 03	;								
235 { 236 $*(\&v2 + i) ^{=} *(\&v59 + i);$ 237 $*(\&v2 + i) ^{=} 0x13u;$ 238 } 239 return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968)	234	for (1	= 6); 1 < 56	; ++	1)					
<pre>236 *(&v2 + i) ^= *(&v59 + i); 237 *(&v2 + i) ^= 0x13u; 238 } 239 return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968) </pre>	235	{									
<pre>237 *(&v2 + i) ^= 0x13u; 238 } 239 return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968)</pre>	236	* (& v2	+ i	i) ^= *(8	v59 ·	+ i);					
<pre>238 } 239 return sub_45A7BE("%s\n"); 240 } 00007D68 sub_45E940:211 (45E968) </pre>	237	* (& v2	+ i	i) ^= 0x1	.3u;						
<pre>239 return sub_45A7BE("%s\n"); 240 00007D68 sub_45E940:211 (45E968) </pre>	238	}									
240 } 00007D68 sub_45E940:211 (45E968)	239	return s	sub	45A7BE("	%s∖n	"):					
00007D68 sub_45E940:211 (45E968)	240	}	-								
00007D68 sub_45E940:211 (45E968)											
		00007D68 sub	_45E	940:211 (4	5E968)	hitpe://blo	chasalaraa	əi/wəixin_426	21117		

这里我们就初步接触到了逆向的加解密,加解密其实也就是算法的使用。这里加密比较简单,甚至都不能称为加密。*(&v2 + i)的 值练起来就是flag的值

所以得到解密代码: (博主使用python, 其他语言均可)

```
#v2: 原代码v2-v58的值
v2 = [123,32,18,98,119,108,65,41,124,80,125,38,124,111,74,49,83,108,94,108,84,6,96,83,44,121,104,110,32,95,117,1
01,99,123,127,119,96,48,107,71,92,29,81,107,90,85,64,12,43,76,86,13,114,1,117,126,0]
#v59: 原代码v59-v115的值
v59 = [18,64,98,5,2,4,6,3,6,48,49,65,32,12,48,65,31,78,62,32,49,32,1,57,96,3,21,9,4,62,3,5,4,1,2,3,44,65,78,32,1
6,97,54,16,44,52,32,64,89,45,32,65,15,34,18,16,0]
s = ""
for i in range(57):
    v2[i] = v2[i] ^ v59[i]
    v2[i] = v2[i] ^ 19
    s += chr(v2[i])
print(s)
```

得到flag: zsctf{T9is_tOpic_1s_v5ry_int7resting_b6t_others_are_n0t}

0x3.Hello,CTF

解析: 老办法,遇到.exe直接打开看看是啥玩意儿。随便输,发现会弹出来wrong,输出巨长字符串后会直接退出 拖到IDA打开,shift+F12 发现和我们的程序中有一个东西是匹配的: "wrong!\n",关键点get到! 双击进IDA View-A,Ctrl+X,F5

```
伪代码
IDA View-A 🛛
                      3
                                        ×
                                            ◎ 十六进制视图-1 ◎ 🔄
                                                                        字符串窗口
   3
      signed int v3; // ebx
      char v4; // al
   4
      int result; // eax
   5
   6
      int v6; // [esp+0h] [ebp-70h]
      int v7; // [esp+0h] [ebp-70h]
   7
   8
      char v8; // [esp+12h] [ebp-5Eh]
      char v9[20]; // [esp+14h] [ebp-5Ch]
  9
      char v10; // [esp+28h] [ebp-48h]
  10
  11
      __int16 v11; // [esp+48h] [ebp-28h]
  12
      char v12; // [esp+4Ah] [ebp-26h]
      char v13; // [esp+4Ch] [ebp-24h]
  13
  14
• 15
      strcpy(&v13, "437261636b4d654a757374466f7246756e");
16
      while (1)
  17
      {
18
        memset(&v10, 0, 0x20u);
• 19
        v11 = 0;
20
        v12 = 0;
21
        sub_40134B(aPleaseInputYou, v6);
22
        scanf(aS, v9);
23
        if (strlen(v9) > 0x11)
          break;
24
0 25
        v3 = 0;
  26
        do
  27
        {
28
          v4 = v9[v3];
          if ( !v4 )
29 🔵
30
            break;
          sprintf(&v8, asc 408044, v4);
031
32
          strcat(&v10, &v8);
933
          ++v3;
  34
        }
35
        while (v_3 < 17);
        if ( !strcmp(&v10, &v13) )
36
37
          sub_40134B(aSuccess, v7);
  38
        else
0 39
          sub_40134B(aWrong, v7);
  40
      }
• 41
      sub_40134B(aWrong, v7);
• 42
      result = stru_408090._cnt-- - 1;
• 43
      if ( stru_408090._cnt < 0 )
• 44
        return _filbuf(&stru_408090);
• 45
      ++stru 408090. ptr;
      return result;
• 46
• 47 }
```

简单的逻辑推理: v9为我们的输入,长度≤0x11(10进制的17) v10储存的就是v9,和v13进行比较。相同就success 到这里我们就知道输入必须就是v13这个字符串相同。但是发现引号中字符数>17,所以判断这是个16进制数表示的字符串 (ASCI码),用网上16进制转字符串得到flag: CrackMeJustForFun

0x4.open-source

拿到源码了嘤嘤嘤,就直接IDE打开不解释! 源码如下图:

```
#include <stdio.h>
 #include <string.h>
]int main(int argc, char *argv[]) {
7
    if (argc != 4) {
        printf("what?\n");
    -}
    unsigned int first = atoi(argv[1]);
3
    if (first != 0xcafe) {
        printf("you are wrong, sorry.\n");
        exit(2);
     }
    unsigned int second = atoi(argv[2]);
3
    if (second % 5 == 3 || second % 17 != 8) {
        printf("ha, you won't get it!\n");
        exit(3);
    }
]
     if (strcmp("h4cky0u", argv[3])) {
        printf("so close, dude!\n");
         exit(4);
     }
    printf("Brr wrrr grr\n");
    unsigned int hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207;
    printf("Get your key: ");
    printf("%x\n", hash);
    return 0;
- }
```

解析: 这个题不用逆向也能做,纯源码分析就能得到答案。 逆向做法: 随便输入几个参数编译链接执行发现wrong 拖IDA, shift+F12,发现"Get your key:",双击,Ctrl+X+确定,F5 发现v3就是key,写出代码求得v3

v3 = 11 * (25 % 17) + 1628458542 + len("h4cky0u") - 1615810207 print(v3)

得到12648430,转16进制得到flag: cOffee

0x5.simple-unpack

从题目就知道需要脱壳,但是让我们假装不知道QAQ!依旧还是拖到IDA里面看看,果然! 什么都看不懂……那还是老步骤: shift+F12,发现了一个关键字: upx,说明他是upx压缩的文件,所以就需要upx解压

这里博主还是推荐大家装一个kali,双系统或者虚拟机都可以。如果原本就用的Ubuntu等Linux可以忽略这句话QWQ

upx -d filename脱壳

root@kali-Flash	000:~/CTF#	upx -d b7cf4629544f4e759d690100c3f96caa										
	υι	cimate Pa	cker for execu									
1 朱		copyrign	t (C) 1996 - 2	018								
UPX 3.95	Markus Ob	erhumer,	Laszlo Molnar 🤅	& John Reiser	Aug 26th 2018							
1收站					-							
File si	ze	Ratio	Format	Name								
他位 912808 <-	352624	38.63%	linux/amd64	b7cf4629544f4e	759d690100c3f96c							
a												
Unpacked 1 file					https://blog.csdn.net/weixin_42621117							

拖到IDA, shift+F12直接得到flag: flag{Upx_1s_n0t_a_d3liv3r_c0mp4ny}

0x6.logmein

日常拖IDA, shift+F12

第一次经验性进You entered the correct password!\nGreat job!\n,发现反编译出来的函数没啥用,所以第二次选择进输入点Enter your guess(类似于找OEP时先找PUSHAD和POPAD)

C BURD O

	IDA View-A		伪代码	×	's'	字符	串窗口	×	Ō	十六进制
1	voidfastca	allnore	turn main(_	_int64	a1,	char	**a2,	char	**a3)	
2	{									
3	size_t v3;	// rsi								
4	int i; // [_rsp+3Ch]	[rbp-54h]							
5	char s[36];	; // [rsp+4	40hj [rbp-5	0h]						
6	int v6; //	[rsp+64n]	[rbp-2Ch]	oh 1						
/	$_1$ 10164 V/;	// [rsp+	oðnj [rop-2 Zahl [mhn 2	an j						
0	int vo. //	[nen+8Ch]	/onj [rop-2 [nbn_4b]	נחסי						
10	IIIC V9, //	[1 sp+och]	[100-411]							
• 11	$v_{9} = 0$:									
12	strcpv(v8.	":\"AL RT	^L*.?+6/46"):						
• 13	v7 = 285371	1945736195	60LL;	//						
• 14	v6 = 7;									
• 15	printf("Wel	Lcome to t	he RC3 secu	ire pas	swor	d gues	sser.\ı	n", a2	, a3)	;
• 16	<pre>printf("To</pre>	continue,	you must e	nter t	he co	orrect	t passı	word.\	n");	
• 17	printf("Ent	er your g	uess: ");							
• 18	isoc99_sc	anf("%32s	", s);							
• 19	v3 = strler	1(s);								
20	if (v3 < s	<pre>strlen(v8)</pre>)							
21	sub_40070	20(v8);								
• 22	for (i = 0); i < str.	len(s); ++i)						
23	{	- t - 1 (- > >							
24	1† (1 >=	<pre>strien(Va (*)(vaid)</pre>	(8)	$\sim \sim$						
25)SUD_4007C0)(*((PVTE	*\9.7		2		; 1 \ \		
 20 27 	(void	(*)(void)) ($(())$	() () ()	τ ⊥ /	~ vo)	vo[.	1))		
27	3	()(voiu)	/sub_400/ce)();						
0 20	sub 4007F0 <i>(</i>	0:								
0 30	}	. / 3								
	د									

逻辑分析:

v8是给定的字符串,v7是long long的数据类型

s是输入,v3是s的长度,v3必须≥v8的长度17,否则会进入提示输入错误的函数sub_4007C0()

重点是: (_BYTE *)&v7的意思是,把longlong型的v7强制转化为byte型的地址,简单的说,就是把它看成字符串(C语言字符串本质都是指针首地址+偏移)。

所以我们用先用v7的值10进制转16进制,然后16进制转文本得到: ebmarah

重点来了!为什么直接套这个字符串不对,根本原因是因为在机器虚拟化内存后,规定地址排列规则时使用了小端法(最低有效 字节在前面)。因此我们真正的解码文本应该是把上面的答案倒过来写:harambe

```
v8 = ":\"AL_RT^L*.?+6/46"
v7 = 'harambe'
for i in range(len(v8)):
     char = ord(v7[i % 7]) ^ ord(v8[i])
     print(chr(char),end='')
```

得到flag: RC3-2016-XORISGUD

当然个人感觉最简单的办法还是C++重现一遍。。。就不用考虑这么多

```
#include <iostream>
using namespace std;
int main(){
  long long v7 = 28537194573619560;
  char *p = (char*)&v7;
  char v8[] = ":\"AL_RT^L*.?+6/46";
  for(int i = 0;v8[i]!=0;i++){
   v8[i] = v8[i]^p[i%7];
  }
  cout<<v8<<endl;
  return 0;
}</pre>
```

0x7. insanity

解析:

这个真的不知道咋解析……至于为啥放这里,也许就和题目所言一样吧,希望大家身心愉悦继续肝吧…… 拖IDA,shift+F12直接拿到flag: 9447{This_is_a_flag}

0x8.no-strings-attached

这个题是真的有难度QAQ

正常步骤拖到IDA静态分析, shfit+F12, 发现第一行赫然出现: /lib/ld-linux.so.2。看见这个大家心里应该都有数了, 和linux有关 没跑了。同时也说明这是个ELF文件

字符串没有关键字,就从IDA左边函数列表找到main函数双击进去,F5反汇编,再进到authenticate函数看看(有的东西做多了就知道了),如下:



此时就真的看英语了……计算s2的函数decrypt正是非常专业的术语:解密。 粗略的看一下下面的伪码,得出:ws是输入,ws==s2时就是正确的flag 此时我们需要转变一下思维:之前我们都是各种找、各种逻辑推断正确输入。但是我们忽略了一件事,那个与输入的比较的正确 答案,一定是加载到内存里面之后,才与输入比较。要是我们能跟踪到这个正确答案储存在内存的位置然后把他拿出来,这不也 行嘛!!! (Reverse!)

思路有了,还需要实际的操作。这里就不能用静态分析了。这里插一句,我们逆向分析分为静态分析和动态分析,直接拖到IDA 反汇编看伪代码,逻辑推断等等都属于静态分析。换言之,在没有执行程序或程序是静态时的分析。 所以要用IDA动态调试ELF—IDA remote linux debugger

环境配置参考IDA动态调试ELF写的非常清楚



首先我们进入authenticate, F5,点左边设置断点,如下图(在s2刚被赋值完毕后停止,找s2的值)



08048708 848708 08048708 85 = dword ptr -800Ch 08048708 98048708 08048708 98048708 08048708 91000000000000000000000000000000000000
08048708 ws= dword ptr -800Ch EE 08048708 s2= dword ptr -0Ch EE 08048708 summind { EE 08048708 punwind { EE 08048708 push ebp 08048708 push ebp 08048709 mov ebp, esp 08048709 sub esp, 8028h 08048711 mov dword ptr [esp+4], offset dword_8048A90 ; wchar_t * 08048719 mov dword ptr [esp], offset s ; s 08048720 call decrypt 08048725 mov [esp+s2], eax 08048728 mov exp+s], eax ; stream 08048721 mov [esp+s2], eax 08048728 mov [esp+s4], 2000h ; n
<pre>08048708 s2= dword ptr -0Ch 08048708 08048708 08048708 ;unwind { 08048709 mov ebp, esp 08048709 mov ebp, esp 08048708 sub esp, 8028h 08048711 mov dword ptr [esp+4], offset dword_8048A90 ; wchar_t * 08048719 mov dword ptr [esp], offset s ; s 08048720 call decrypt 08048725 mov [ebp+s2], eax 08048725 mov [ebp+s2], eax 08048720 mov [esp+8], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n</pre>
<pre> 08048708 08048708;unwind { 08048708 push ebp 08048709 mov ebp, esp 08048708 sub esp, 8028h 08048708 sub esp, 8028h 08048711 mov dword ptr [esp+4], offset dword_8048A90 ; wchar_t * 08048719 mov dword ptr [esp], offset s ; s 08048720 call decrypt 08048725 mov [ebp+s2], eax 08048725 mov [esp+8], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n 0804874 080487</pre>
<pre> 08048708 ;unwind { 08048708 push ebp 08048709 mov ebp, esp 08048709 mov ebp, esp 08048708 sub esp, 8028h 08048711 mov dword ptr [esp+4], offset dword_8048A90 ; wchar_t * 08048719 mov dword ptr [esp], offset s ; s 08048720 call decrypt 08048725 mov [ebp+s2], eax 08048725 mov eax, ds:stdin@@CLIBC_2_0 08048720 mov [esp+8], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n </pre>
08048708 push ebp 08048709 mov ebp, esp 08048708 sub esp, 8028h 08048708 mov dword ptr [esp+4], offset dword_8048A90 ; wchar_t * 08048711 mov dword ptr [esp], offset s ; s 08048720 call decrypt 08048725 mov [ebp+s2], eax 08048725 mov [ebp+s2], eax 08048720 mov [esp+8], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n
08048709 mov ebp, esp 08048708 sub esp, 8028h 08048711 mov dword ptr [esp+4], offset dword_8048A90 ; wchar_t * 08048719 mov dword ptr [esp], offset s ; s 08048720 call decrypt 08048725 mov [ebp+s2], eax 08048728 mov eax, ds:stdin@@GLIBC_2_0 08048720 mov [esp+8], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n
0804870B sub esp, 8028h 08048711 mov dword ptr [esp+4], offset dword_8048A90 ; wchar_t * 08048719 mov dword ptr [esp], offset s ; s 08048720 call decrypt 08048725 mov [ebp+s2], eax 08048728 mov eax, ds:stdin@GLIBC_2_0 08048720 mov [esp+8], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n
08048711 mov dword ptr [esp+4], offset dword_8048A90 ; wchar_t * 08048719 mov dword ptr [esp], offset s ; s 08048720 call decrypt 08048725 mov [ebp+s2], eax 08048728 mov eax, ds:stdin@GLIBC_2_0 08048720 mov [esp+4], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n
08048719 mov dword ptr [esp], offset s ; s 08048720 call decrypt 08048725 mov [ebp+s2], eax 08048728 mov eax, ds:stdin@@GLIBC_2_0 08048720 mov [esp+s], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n
08048720 call decrypt 08048725 mov [ebp+s2], eax 08048728 mov eax, ds:stdin@@GLIBC_2_0 08048720 mov [esp+8], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n
08048725 mov [ebp+s2], eax 08048728 mov eax, ds:stdin@@GLIBC_2_0 0804872D mov [esp+8], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n
08048728 mov eax, ds:stdin@@GLIBC_2_0 0804872D mov [esp+8], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n [minimized]
0804872D mov [esp+8], eax ; stream 08048731 mov dword ptr [esp+4], 2000h ; n
08048731 mov dword ptr [esp+4], 2000h ; n
08048739 lea eax, [ebp+ws]
0804873F mov [esp], eax ; ws
08048742 call _fgetws
08048747 test eax, eax
08048/49 jz short loc_8048/9C
0804874B lea eax. [ebp+ws]
08048751 mov [esp], eax ; s
00.00% (-401.218) (680.299) 00000728 08048728: authenticate+20 (Synchronized with EIP)
1十六进制模图1
8048700 45 F4 83 C4 34 58 5D C3 55 89 F5 81 FC 28 80 00 F[](
8048710 00 C7 44 24 04 90 8A 04 08 C7 04 24 A8 8A 04 08\$\$
8048720 E8 33 FF FF FF 89 45 F4 A1 3C A0 04 08 89 44 24ED\$
8048730 08 C7 44 24 04 00 20 00 00 8D 85 F4 7F FF F8 9\$
8048740 04 24 E8 59 FD FF FF 85 C0 74 51 8D 85 F4 7F FF .\$Q
0000708 08048708: authenticate
1 输出窗口
nipeseting auto-networks. Alternative and a second a second

这个时候,我们看到了s2就储存在寄存器eax中,所以我们在下面的Hex View窗口中右键,synchronized with,选eax,就能看到 值啦,这就是flag,如下图

081FE7C0	5F	43	54	59	50	45	00	00	00	00	00	00	31	00	00	00	_CTYPE1
081FE7D0	A0	E7	1F	0 8	01	00	00	00	00	00	00	00	00	00	00	00	
081FE7E0	BØ	E6	1F	08	D0	E6	1F	<u> 8</u> 0	60	E6	1F	0 8	00	00	00	00	· · · · · · · · · · · · · · · · · · ·
081FE7F0	00	00	00	00	00	00	00	00	00	00	00	00	31	00	00	00	1
081FE800	39	00	00	00	34	00	00	00	34	00	00	00	37	00	00	00	9447
081FE810	7B	00	00	00	79	00	00	00	6F	00	00	00	75	00	00	00	{you
081FE820	5F	00	00	00	61	00	00	00	72	00	00	00	65	00	00	00	are
081FE830	5F	00	00	00	61	00	00	00	6E	00	00	00	5F	00	00	00	an
081FE840	69	00	00	00	6E	00	00	00	74	00	00	00	65	00	00	00	inte
081FE850	72	00	00	00	6E	00	00	00	61	00	00	00	74	00	00	00	rnat
081FE860	69	00	00	00	6F	00	00	00	6E	00	00	00	61	00	00	00	iona
081FE870	6C	00	00	00	5F	00	00	00	6D	00	00	00	79	00	00	00	1my
081FE880	73	00	00	00	74	00	00	00	65	00	00	00	72	00	00	00	ster
081FE890	79	00	00	00	7D	00	00	00	00	00	00	00	57	00	00	00	y}W
081FE8A0	01	00	00	00	E8	E1	9F	F7	EA	Ε1	9F	F7	EC	Ε1	9F	F7	
081FE8B0	EE	E1	9F	F7	FØ	E1	9F	F7	F2	E1	9F	F7	F4	E1	9F	F7	himu/kies sets set/usi/sis/ 3969447
AR1FERCA	E6	F1	QE	F7	F۶	F1	QE	F7	F۸	F1	QE	F7	Q1	aa	aa	aa	

至此拿到flag: 9447{you_are_an_international_mystery}

0x9.csaw2013reversing2

解析: 拖到IDA中分析发现有重要的函数IsDebuggerPresent(),这个函数目的就是反调试(检测是否处于调试环境中)。既然如此千方百计阻止我们调试,那就直接OD动态走起。

我们拖到OD中,ctrl+n找到lsDebuggerPresent(),确定他的位置之后下断点开始调试程序,发现底下有两个对话框的代码(能看 见注释那里有Flag,Text字样就ok),手动F8看一次,发现00C61000那里的函数没有执行。本着现在是"你不让干的事我偏要搞 一次"的思想,我们修改程序跳转代码,发现flag赫然出现!

由于这样的方法强行改汇编跳转也存在"试"的成分,所以直接给修改完成的代码(修改了4处),如下图:

00C6108A		75 ØA	<mark>jnz</mark> short 3f356420.00C61096		
00C6108C		90	nop	IsDebuggerPresent函数	
00C6108D		90	nop		
00C6108E		90	nop		
00C6108F		98	nop		
00061090		90	nop		
00C61091		98	nop		
00C61092		8500	test eax,eax		
00061094		98	пор		
00C61095		98	nop		
00C61096	>	41	inc ecx		
00C61097		41	inc ecx		
00C61098		41	inc ecx		
00C61099		41	inc ecx		
00C6109A		98	пор	inc 3 🔁	
00C6109B		8B55 F4	<pre>mov edx,dword ptr ss:[ebp-0xC]</pre>		
00C6109E		E8 5DFFFFFF	call 3f356420.00C61000		
00C610A3		EB 14	jmp_short_3f356420.00C610B9		
00C610A5		6A 02	push 0x2	rStyle = MB ABORTRETRYIGNORE/MB APPLMODAL	
00C610A7		68 20786600	push 3f356420.00C67820	Flag	
00C610AC		FF75 F4	push dword ptr ss:[ebp-0xC]	Text = ""	
00C610AF		6A 00	push 0x0	hOwner = NULL	
00C610B1		FF15 E460C60	call dword ptr ds:[<&USER32.MessageBoxA]	MessageBoxA	
00C610B7		EB 14	jmp short 3f356420.00C610CD		
00C610B9	>	6A 02	push 0x2	rStyle = MB_ABORTRETRYIGNORE MB_APPLMODAL	
00C610BB		68 20786600	push 3f356420.00C67820	Flaq	
00C610C0		8B45 F4	<pre>mov eax,dword ptr ss:[ebp-0xC]</pre>	-	
00C610C3		40	inc eax		
00C610C4		50	push eax	Text = "flag{reversing_is_not_that_hard!}"	
00C610C5		6A 00	push 0x0	hOwner = NULL	
00061007		FF15 E460C60	call dword ptr ds:[<&USER32.MessageBoxA)	HessageBoxA	
00C610CD	>	FF75 F4	<pre>push dword ptr ss:[ebp-0xC]</pre>	rpMemory = 031C05B8	
00C610D0		6A 00	push 0x0	Flags = 0	
00C610D2		FF75 FC	push dword ptr ss:[ebp-0x4]	hHeap = 031C0000	
00C610D5		FF15 0860C60	<pre>call dword ptr ds:[<&KERNEL32.HeapFree>]</pre>	LHeapFree	
00C610DB		8945 F8	<pre>mov dword ptr ss:[ebp-0x8],eax</pre>		
00C610DE		FF75 FC	<pre>push dword ptr ss:[ebp-0x4]</pre>	rhHeap = 031C0000	
00C610E1		FF15 0C60C60	<pre>call dword ptr ds:[<&KERNEL32.HeapDestri</pre>	LHeapDestroy	
00C610E7		6A 00	push 0x0	rExitCode = 0x0	
00C610E9		FF15 0060C60	<pre>call dword ptr ds:[<&KERNEL32.ExitProces</pre>	-ExitProcess	
00C610EF	>	6A FF	push -0x1	rExitCode = 0xFFFFFFF	
00C610F1		FF15 0060C60	<pre>call dword ptr ds:[<&KERNEL32.ExitProces</pre>	LExitProcess	
00C610F7	-	C9	leave	https://blog.csdn.net/weixin_426/	1 🖂

所以直接能拿flag啦: flag{reversing_is_not_that_hard!}

0xa.getit

解析:依旧老套路,拖IDA,shift+F12看字符串发现linux和一个很像flag形式的字符串"SharifCTF{???}",双击点进去,然后在左边的框找到主函数,反汇编成伪代码。如下图:

×	1111	IDA V	iew-A	×	111	伪代码	×	's'	字符	串窗口	×	Ō	十六进
ç	1	int	cdecl	main(in	t argc	, const	char **a	argv,	const	char	**envp)		
init	2	{											
olt	3	char	v3; /	/ al									
plt	4	in	t64 v5	;//[r	sp+0h]	[rbp-40	h]						
plt	5	int	i; //	[rsp+4h] [rbp	-3Ch]							
plt	6	FILE	*stre	am; //	[rsp+8l	n] [rbp-	38h]						
51t	7	' char	filer	ame[8];	// [rs	sp+10h]	[rbp-30	h]					
olt	8	unsi	gned _	_int64	v9; //	[rsp+28	h] [rbp	-18h]					
plt	9)											
blt	0 10	v9 =	rea	dfsqwor	d(0x28ı	ı);							
p1t	• 11	LODW	ORD(v5) = 0;									
plt	• 12	whil	e ((s	igned i	nt <mark>)v5</mark> ∙	< strlen	(s))						
text	13	{											
text	• 14	if	(v5	& 1)									
text	• 15		v3 = 1	;									
text	16	el	se										
text	• 17	,	v3 = -	1;									
text	• 18	*(&t + (signed	int <mark>)</mark> v5	+ 10) =	s[(sign	ned ir	nt)v5]	+ v3;			
text	• 19	LO	DWORD (v5) = v	5 + 1;								
fini	20	}			-								
xtei	0 21	strc	py(fil	ename,	"/tmp/	Flag.txt	");						
xter	22	stre	am = f	open(fi	lename	, "w");							
rter	23	fpri	ntf(st	ream, "	%s∖n",	u, v5);							
xter	24	for	(i =	0; i <	strlen	(&t); ++	i)						
xtei	25	{		-			·						
xtei	0 26	fs	eek(st	ream, p	[i], 0]);							
xtei	0 27	fp	utc(*(&t + p[i]), st	ream);							
xtei	28	fs	eek(st	ream, 0	LL, 0)								
	0 29	fp	rintf(stream,	"%s∖n'	', u);							
	30	}											
	• 31	fclo	se(str	eam);									
	• 32	remo	ve(fil	ename):									
	• 33	retu	rn 0;										
	• 34	}	· · · ·										
		-											

简单分析代码:(重点是11~20行)s长度限定,v5条件选择,v3偏移量,用参数操作s,t为最终存放数组,最后用流写入tmp文件夹下的flag.txt中。但是/tmp是linux主目录下一个存放临时文件的文件夹,程序return后写入的临时文件也一并丢弃。

这里额外说一下,这道题可以用在linux环境下运行,然后设置断点去/tmp文件夹下找,或者直接更改流写入的目标文件夹都是可以的。这里我们使用windows纯代码分析的方法。

```
通过分析我们发现v3,v5已知,需要知道s和t。我们在IDA的IDA View-A的窗口中找到s的值,如下图
:
.data:00000000006010A0
                                       public s
.data:00000000006010A0 ; char s[]
                                       db 'c61b68366edeb7bdce3c6820314b7498',0
.data:0000000006010A0 s
.data:00000000006010A0
                                                               ; DATA XREF: main+251o
.data:0000000006010A0
                                                               : main+3Fîr
找t的值,代码如下:(注意这里是题目有bug!!!t的值是SharifCTF{???},可在16进制视图窗口查看)
.data:00000000006010C1
                                    align 20h
.data:0000000006010E0
                                    public t
.data:00000000006010E0 ; char t
.data:00000000006010E0 t
                                   db 53h
                                                          ; DATA XREF: main+651w
.data:00000000006010E0
                                                          ; main+C91o ...
```

.data:00000000006010E1 aHarifctf .data:000000000060110C

.data:000000000601120



最后写出代码

v5 = 0
s = 'c61b68366edeb7bdce3c6820314b7498'
t = ['S','h','a','r','i','f','C','T','F','{','?','?','?','?','?','?','?','?','?
,'?','?','?','?','?','?','?','?','?','?
v3 = 0
1 = len(s)
while(v5 < 1):
if(v5 & 1):
v3 = 1
else:
v3 = -1
t[10+v5] = chr(ord(s[v5])+v3)
v5 += 1
flag = ''
for x in t:
flag+=x
print(flag)

得到flag: SharifCTF{b70c59275fcfa8aebf2d5911223c6589}

0xB.python-trade

下载完文件发现是一个.pyc文件,百度得知.pyc文件其实是PyCodeObject的一种持久化保存方式(感兴趣可自行搜索学习)。所以思路就比较清晰了:用python反编译在线工具反编译这个.pyc文件得到源码,如下图

请选择pyc文件进行解密。支持所有Python版本

选择文件未选择任何文件

```
#!/usr/bin/env python
# encoding: utf-8
# 如果觉得不错,可以推荐给你的朋友! http://tool.lu/pyc
import base64
def encode(message):
    s = ''
    for i in message:
       x = ord(i) ^ 32
       x = x + 16
       s += chr(x)
    return base64.b64encode(s)
correct = 'X]NkVmtUI1MgXWBZXCFeKY+AaXNt'
flag = ''
print 'Input flag:'
flag = raw_input()
if encode(flag) == correct:
   print 'correct'
else:
   print 'wrong'
```

```
关键点: encode(flag) == correct 所以就很容易写出逆向解码的代码:
```

<pre># encoding: utf-8</pre>
import base64
s = "XlNkVmtUI1MgXWBZXCFeKY+AaXNt"
flag = ""
#base64
b = base64.b64decode(s)# print(b)
#encode
for i in b:
i -= 16
i ^= 32
flag += chr(i)
print(flag)

拿到flag: nctf{d3c0mpil1n9_PyC}

0xC.maze

解析: ELF文件, 日常拖到IDA, 查找字符串, 交叉引用, F5大法好。

分析代码,s1储存输入对象,比较前5位是不是"nctf{",第25位最后一位是不是"}"。之后发现asc_601060中储存的是一个8*8的迷宫,迷宫如下:

***	***	**
* >	*	*
*** >	* *	**
** >	* >	**
** >	* *	**
* *:	ŧ	*
** **	**	*
**		*
****	**:	**

通过分析,发现v4是玩家输入的方向: 'O'-左, 'o'-右,'.'-上, '0'-下,由迷宫得到轨迹: 右下右右下下左下下下右右右右上上 左左

到这里,整个攻防世界Reverse的Exercise area就解答完毕了,希望大家能多多交【pi】流【ping】! RE真好玩~强颜欢笑.jpg