THUCTF 2020: Writeup

计科 50 赵梓硕 (ID: quq)

总体感想

其实我的研究方向（无论是之前的 CV 还是现在的 OR）和网络安全毫不相关，之前对 CTF 这个比赛也一无所知，就是在开赛前一个小时看到朋友圈有人发 THUCTF 的消息，就突然心血来潮想打着玩玩。我毕业之前曾经在设计女生节横幅时出过一些 Crypto 的题目，当时就有同学说，对这种风格的题感兴趣的话，要不要了解一下 CTF？不过当时实在是太忙，如今毕业之后能够花几天时间打一打早就听闻的这个比赛，也算是让自己本科生活少一项遗憾吧。

刚参加的时候其实做出了 baby-xorrrr、RSA-checkin 和 CantHearYou 三题之后，就感觉其他题目涉及的知识就很陌生了。但想到比赛时间还有这么长，加上看到 decaf 就想到自己曾经被编译原理支配的恐惧（wiku 对计算机底层的那些东西真的是好不擅长啊）... 于是觉得，既然当时做编译 PA 的时候就是根据已有的框架去顺藤摸瓜填上一些 feature，那是不是用同样的方法也可以试着做个一两题？然后就看到 Ant 其实就是个语法树，既然编译原理还没忘光，稍稍研究下也就做出来了。

紧接着就看这个 minidecaf。我最初的想法是直接手动模拟 ir 的那些操作，于是比较顺利地解开了 check1。但从 check2 开始就要面对那些不知所云的 mcfx 函数，于是就先看看 mcfx0 吧，手推了几步发现 1 是右移 3 是左移，0 就是异或，然后发现 0、2、4 其实挺像，稍稍对比一下也就明白了 2 是与、4 是或。在这个过程中我发现 ir 中的操作似乎有些固定的套路，于是尝试写了几个等价的 decaf 语句，发现翻译成 ir 之后几乎完全一样。所以后面的分析就不再从底层模拟 ir 的操作了，而是根据 ir 手工反编译 decaf，居然还比较管用。

感觉这次 CTF 让人感觉最有意思的就是 minidecaf 和 lfsr。lfsr 的话，那个输出 majority 的非线性设定让人有些迷茫，百思不得其解后，突然想到任何一个串和 majority 重合的概率是大于 50% 的，再加上知道初值，所以可以穷举 mask 然后统计重合率。而第二部分最初不知异或怎么处理，但意识到异或就是 GF(2) 里的加，然后 mask5 有多种可能性的现象也让我意识到如果把两个特征函数乘起来得到的递推式一定两个都符合，于是输出的异或就满足一个 384 阶的递推式，最后解方程求出这个递推式，再因式分解就算出来了。

然后最后在提示下做出了一道 web（虽然自己对 web 一无所知，全靠现学）。听说 babypwn 很简单但是有点累了，放假还是好好休息吧 orz（毕竟休息两天后实习的活得多做点了... 前几天有点摸鱼...）

（最快乐的还是在 deadline 前和同学出去浪了半天，没有人比我更**不**懂 deadline！[手动滑稽]）

1 Crypto

1.1 baby-xorrrr

注意到 flag 的前 6 字节为 “THUCTF”，将其与 secret、输出的前 6 字节异或即可得到 key 的值。再将输出与 cycle(key)、cycle(secret) 异或即可得到 flag。

flag：THUCTF{eaeffc2d-0f47-49ca-be26-c9fee3a3d53e}

1.2 RSA-checkin

设 $A = 2 * d + e * phi * 2020 + 2020$ 为已知值，那么令 $d' = \frac{A-2020}{2} = d + e * phi * 1010$，则明文 $m \equiv c^d \equiv c^{d'} (\bmod n)$。用快速幂求得 m，再转换为 ASCII 即可。

flag：THUCTF{99049c76-3447-46d0-b52e-299e25755a48}

1.3 lfsr

Step 1

根据 task.py 我们得知，输出文件中 20000 位的 0-1 串是由两部分 A, B 穿插组成。由于 s3、s4、s5 的初值已知，我们应当首先据此得到 mask(3-5)。

由于 B 的每一位是 b(3-5) 中出现较多的数字，对于任何 $i \in \{3, 4, 5\}$，若将 b(3-5) 中的另两个视作均匀随机，那么 $\Pr[B[i] = b[i]] = 0.75$。

据此，可以将 mask(3-5) 分别进行爆破，$\forall i \in \{3, 4, 5\}$，若 mask(i) 猜测正确，则得到的 B 会有大约 75% 与 s(i) 重合，反之则只有 50%。总时间为 $3 \cdot 2^{21} \cdot 10000$，可以在数分钟内完成。

注意到 mask5 有多个可行解，但得到的 b5 完全相同。

Step 2

此节中所有 0-1 串均在 $GF[2]$ 的意义下讨论，因此，运算 “+” 等同于 “异或”。

根据 mask(3-5)，我们即可得到 b3 和 b4 的序列，从而得到 $b^* := (b1 + b2)$。注意到 b1、b2 分别满足 $GF[2]$ 一个 256 阶和 128 阶的递推式。设递推式

$$\sum_{i=0}^{m} c[i]t[n-i] = 0$$

对应的特征多项式为

$$f_t(x) = \sum_{i=0}^{m} c[i]x^i$$

那么 $F(x) = f_{b1}(x)f_{b2}(x)$ 是 $b1 + b2$ 的一个特征多项式。设

$$F(x) = \sum_{i=0}^{384} w[i]x^i$$

则

$$\sum_{i=0}^{384} w[i]b^*[n-i] = 0$$

对任意 $n \geq 384$ 成立。

于是，由于已知 b^* 的值，我们可以通过解 $GF[2]$ 的线性方程组来得出 w（即 F），并对 $F(x)$ 在 $GF[2]$ 上进行因式分解，得到了次数分别为 3、25、28、30、41、42、95、120 的 8 个因式。

由于 $\deg(f_{b1}) = 256$，$\deg(f_{b2}) = 128$，我们枚举全部 256 种情况，这些因式取一部分相乘得到 256 次的情况只有 $120 + 95 + 41 = 256$ 和 $120 + 42 + 41 + 28 + 25 = 256$ 两种，可分别得到相应的 mask。根据 flag 的 sha256 值可排除后者，因此前者为正确答案。

flag：THUCTF{58b2cc4350854e922ad5add4a123dbd767187c5855a52ee82add4a094f45ec48}

2 Web

2.1 showmeflag

根据提供的 python 代码可知，该网站带有上传文件功能，但会在上传的文件的开头和末尾各添加一个 10 字节的随机字符串。而显示 flag 需要通过访问该服务器本地的 URL 返回一个内容为 “Show me flag !!!!!!!!” 的响应。

因此，我们需要使用 HTTP 请求的 Range Request 特性，仅返回该已上传文件中的第 10-30 字节即可。

flag：THUCTF{206_partial_c0ntent_mayb3_useful1}

3 Misc

3.1 CantHearYou

用 foremost 提取出该音频中的一个 zip 文件，根据 zip 内提示密码为纯数字，进行爆破即可解压出 flag.txt，其内容即为 flag。

flag：THUCTF{G00D!You_are_EN3R6ET1C!}

4 Reverse

4.1 Ant

经过反编译发现该程序的核心是对 flag 进行语法检查。首先根据 Lexer 部分知道了该语法中的基本 Token 仅有 THUCTF、{、}、@、0、1。紧接着根据 Parser 知道了 flag 的基本格式为 THUCTF{A_A_A_A}，其中 A 包含 8 个字符，且满足以下语法：

A -> $@^n$ A’

A’ -> 0^n1^n

A’ -> 1^n0^n

A’ -> 回文

于是每个 A 可能的情况大大减少了（仅数十种）。枚举所有可能情况并核对 Hash 值，得到答案。

flag：THUCTF{@@000111_11011011_@@@@1100_@1001001}

4.2 minidecaf

根据 check.ir 中的 size=160 首先得出了 flag 的格式为 THUCTF{.{32}}，之后分析 main 函数知整体上是将中间 32 字节每 8 字节分别用 check1、check2、check3、check4 来检查正确性。

解题过程中，我首先模拟了 ir 语言所对应的操作，之后尝试着通过 decaf->ir 的编译器，找到了 decaf 中各种语句所对应的 ir，并据此手工“还原”出了生成该 ir 的 decaf 源码，并理解了 mcfx(0-4) 所对应的操作：

mcfx0：异或

mcfx1：右移

mcfx2：与

mcfx3：左移

mcfx4：或

在理解 check(1-4) 的检查逻辑之后，得知 check1、check2 是简单的按字节比对，只需要算出各个字节的值即可通过。check3 则是对两个 32 位整型值分别检查它们的和以及进行取反、与、或、异或后的值是否满足要求。在这里只需要首先假设它们和相同，于是便可根据一个求得另一个，再遍历检查它们进行其他运算的值是否也符合要求即可。2^{32} 种可能性对计算机而言也不算太多。

check4 的操作相对复杂，每一轮都是将输入值与当前状态进行一系列迭代操作得到输出，之后判断输出是否符合要求。于是我们可以每轮都尝试输入字节的 256 种可能情况并检查该轮输出是否符合要求，若符合要求再进入下一轮，如此顺藤摸瓜，即可得出正确的输入。

注意到相比其他部分，check3 所需要时间稍长（大约数秒）。

flag：THUCTF{Wri+1n9_y0ur_OwN_cOmPil3r_1$_fuN}

总结

第一次参赛感觉还是开了不少眼界的，以及发现自己前几年最讨厌的编译原理和计组没白学，真的感觉有些成就感！有点遗憾的就是因为不太熟悉这个比赛，有几道其实很简单的 misc 没做出来（deepblue 就在图上，像素点那题是觉得像什么地图但没想到是紫荆地图 orz）

不过作为一个纯萌新... 能够有这样的结果还是很有成就感了。至少可以让自己多一点自信，相信自己未来就算真的要面对完全陌生的领域，也不用过于害怕吧。

（以及这个 writeup 上也藏了个小 flag >_< 虽然并不会有人在意吧 233333