checkin:输入`*ctf{this_1s_flA9_f0rm4t}`

re:反汇编后，发现 main 读入一个字符串，并与 0x601058 处字符串翻转比较。查得 0x601058 处字符串为"}yawyna_lahc_esrever_a_si_siht_wonk_uoy{ftc*"，翻转后为`*ctf{you_know_this_is_a_reverse_chal_anyway}`

crypto:扔到 python 跑一遍即可得到`*ctf{itworks!}`

blog: flag1:博客主人名为 shlr0kawa，到 github 搜索这个人，在个人简介中找到`*ctf{W3_@rE_GitHubbers!}`

login:得到 flag 的要求是 isLogin 为 true 以及 user 为 admin。观察代码可以发现，defaultValues 的赋值方式是若不存在则声明并赋值。因此，手动在链接中加入 isLogin=true 就可以阻止初始化。同时，在 cookie 中加入一条 passwd，值为任意值，避免 isLogin 被赋值为 false。得到`*ctf{eXtrAct_15_NoT_S@fe}`

pyc:反编译 check.pyc，内容为：

```
def do_check(s):
    if s[:5] != '*ctf{':
        return False
    if s[-1] != '}':
        return False
    res = [
     42, 46, 61, 60, 32, 62, 60, 40, 16, 46, 121, 107, 83, 112, 114,
118, 33, 43, 60, 50, 57, 59, 56, 0, 14, 61, 39]
    if len(s) > len(res):
        return False
    accu = 0
    for a, b in zip(s, res):
        accu ^= ord(a)
        if accu != b:
            return False
        accu ^= 103

    return True
```

简单的顺序异或，逆推一下即可得到`*ctf{yes_Y0u_Dec0mpiled_iT}`

diop:将 f 对 a 取模，由于 a 及其巨大，x^2 远远小于 a，即有 f % a == x^2。然后再求出 y 即可。得到`*ctf{Did_you_try_sympy_or_solve_by_hand}`

www:反编译后发现 main 函数读取信息后将给定位置的四个字节改为指定的内容。查看汇编指令时，发现 main 函数上方，0x804851D 处有一个未被调用的后门函数，其内容为调用参数为"/bin/sh"的 system 函数。因此，只需要想办法让进程执行到这段函数即可。由于栈地址随机，难以通过更改 ret 地址的方式。观察到修改完后调用系统调用 puts 输出"bye"，

调用方式为调用函数_puts，它会 jmp 到 0x804A00C 所存储的地址处。而 0x804A00C 的内存是可写的，所以只要将这个地址改成后门函数的入口地址即可。有 bash 后访问/home/pwd1/flag 得到`*ctf{writable_got_is_helpful}`

dontcheat:选出所有格子中颜色和其他不一样的格子。由于颜色差距很大，将显示器亮度和对比度调到最高然后手玩 32 层即可。得到 `CTF{Can_Y0u_5ee_m2}`

CompCipher:反编译后，发现得到 key 后，flag 会被子程序 0x4008AD 加密然后输出。观察子程序的代码，其中有各种混淆。整理后发现实际上子程序的加密方法如下：首先将读入的整数和 256 取模置于加密用数组 0x6020E0[0]位置（之后简称为数组 X），接下来令 X[i]的值为 X[0] + i。然后利用 X[i]的数字对输入字符串（之后简称 s）的第 i 位加密。加密方法为：循环 30 次（j=0~29），若 j 为奇数，将 s 和 X 异或；否则 s 加上 X 的值。然后令 X=X*97+7。因此解密也十分简单，任意得到某个整数下的加密结果，可以构造出数组 X，暴力枚举每一个字符在每一位的加密结果并验证。得到`*ctf{WOW_i5_this_c1pher_rea1ly_complicated??!}`

sh:运行程序后发现这个程序几乎是一个 sh，但是反编译检查代码后发现其输出当前目录地址的函数 0x401193 在当绝对目录名第 2 个字母是 p 时会调用进程 0x400FA2。这个程序检查了是否第 3 4 5 个字母分别为 roc，也就是说这段代码在当前目录处于/proc/时会调用。然后进程获取 proc 的下一层目录并转成十进制数字 d。当满足 d^752593%3138295297==2101783863 时，输出*ctf{str(x)}。所以只需要解出方程即可。可以发现模的这个数等于 50891*61667，显然可以得到 d^752593%50891==2101783863%50891，d^752593%61667==2101783863%61667。枚举 d 解出两个方程在 d<50891 和 d<61667 的解，就可以知道 d%50891=9122 和 d%61667=27567，解得 d=314159265，得到`*ctf{314159265}`

leakage:运行代码，提示你程序运行空间经过了随机化。之后会有输入名字，并询问 main 函数所在的地址。输入名字发现，除了输入的信息在其后会跟着一些乱码。反编译源码后发现使用了 read 进行读取，并不会在后面添加\0，同时栈空间有着一些残留信息，为之前在栈上出现的内容。通过这个信息，我们可以控制名字输入长度来获取从 buffer 地址开始至多 144 个字节的信息。虽然地址经过了随机化，但是在执行代码时，非库函数间代码的相对位置是不变的。同时，同一个程序在栈上的行为也是确定的，因此在本地运行 leakage，通过 pmap 检查程序地址空间，调整输入名字的长度来找到残留在栈空间的某一个非库函数的地址，然后计算偏移量算出 main 函数的地址。得到`*ctf{ithinkyoudbetterinitalizeyourstack}`

ha?:I'm not happy with haskell. 经过猜测，发现需要输入一个大整数 D，满足 d%arr[i]=t[i]。使用中国剩余定理求解，得到解为 17242228295552655727242023973399694343830519588224915405718223882489671415372938627005895013790960495753716622184866372119715675156623414496027593008015438556748389939762531169892617839476325003544286235300268279229320163519462706219995120818836713316584735184918377640259892154900207720777790022367205087 4，调用系统指令得到 kiBlTGpe/vj3b/4dRSJW/T2XQzHABA4V18ge2VztkO8=，与密文异或得到`*ctf{84e4b84cc4e38df2d96267ffe35241af}`

c++11 and lambda:首先将下划线们替换成 func1-func6，l1-l6，e1-e7。求值时传入的匿

名函数 return -(~i)即为 i+1。经过理解发现 l1-l6 的函数是返回一个将传入函数连续执行 i 次的函数。例如：l4(func f) -> return { f'(x) return f(f(f(x)))) }。然后没有研究 e 的内容，开始暴力枚举，发现 e4-e7 的值和 a，b 线性相关。得到`*ctf{cxx11_i5_fUn(19260817)}`

blog: flag2:由于 shlr0kawa 说这是在学习 git 和 github 创建的博客，所以这个博客很可能是用 git 进行版本控制的。首先搜索 hexo 包含的文件，尝试访问/blog/_config.yml，发现可以获取文件，说明外部访问权限不止 public 文件夹，但是访问目录则会 403forbidden。尝试后发现/.git 出现 403forbidden，并且可以访问其下的文件。访问/.git/config，在其中发现 url = https://shlr0kawa:e6d33e42109f91917a50d4c1e44926accbbeb7b6@github.com/starctf/starctf_blog.git。将这个仓库 clone 下来，发现是一个私人仓库，包含了这个博客的所有信息。查找 History，发现有一个 blog & flag deployed 的版本，在 flag.php 中得到`*ctf{.git_Sh0U1d_N0T_8E_3Xpo5eD}`

daoke:查看/root/.ash_history，发现黑客安装了 python，更换了 apk 的源，设置 DNS 服务器，进入了一个被隐藏的目录，将 ctf.fudan.edu.cn:80 的内容导入到 secret 文件中。查找发现 secret 文件位于/var/tmp/下，内容为一个请求网址无法获取的错误网页。显示访问的地址是 http://202.120.224.114/backdoor。将 ip 换成域名后，发现跳转至百度，检查标头，发现状态码 302 跳往百度。在标头中得到`*ctf{d0ck3r_i5_v3ry_1n7er3Stin9}`

sql:装了 sqlite 执行 explain 和 substr 了解输出内容，然后发现每次调用 substr 都为一个字符，且前一次调用的第二个参数代表该字符在字符串的位置。将字符连接起来后得到`*ctf{S1MpLe_sqlIte_bYtec0de}`

blog: flag3:在 stage2 中成功 clone 了一个 private 的仓库，因此使用此 token 是有权利 clone 该用户能够访问的所有仓库的，只需要得到仓库的名称就可以 clone。谷歌搜索以后发现 github 提供了查询用户所有 repository 的 API，通过 https://api.github.com/user/repos 查询到仓库名称/starctf/starctf_blog_stage3_d6c75c60cc80e1615e3f.git，将其 clone 下来，在 README.md 中得到`*ctf{U_kN0W_g1Thu8_V3ry_W3!!}`