

## CTF - RE secret.exe

Chunk secret.exe binary into Ghidra and find main:

```
if (DAT_00409020 != '\0') {
    __Z12DecodeParamsv();
}
local_11 = '\x01';
local_28.lpSecurityDescriptor = (LPVOID)0x0;
local_28.bInheritHandle = 0;
local_28.nLength = 0xc;
BVar2 = _CreateDirectoryA@8(&DAT_00409021, (LPSECURITY_ATTRIBUTES)&local_28);
if ((BVar2 == 0) && (DVar3 = _GetLastError@(), DVar3 == 0xb7)) {
    local_11 = '\0';
}
local_1c = (HANDLE)0xffffffff;
local_1c = _CreateFileA@28(&DAT_00409085, 0xc0000000, 1, (LPSECURITY_ATTRIBUTES)0x0, 2, 0x80,
    (HANDLE)0x0);
if (local_1c == (HANDLE)0xffffffff) {
    DVar3 = _GetLastError@();
}
else {
    local_2c = 0;
    BVar2 = _WriteFile@20(local_1c, &DAT_004090e9, 100, &local_2c, (LPOVERLAPPED)0x0);
    if ((BVar2 == 0) || (local_2c != 100)) {
        bVar1 = true;
    }
    else {
        bVar1 = false;
    }
}
```

Notices WriteFile trying to write 100 bytes data at 0x004090e9.

```
else {
    local_2c = 0;
    BVar2 = _WriteFile@20(local_1c, &DAT_004090e9, 100, &local_2c, (LPOVERLAPPED)0x0);
    if ((BVar2 == 0) || (local_2c != 100)) {
        bVar1 = true;
    }
    else {
        bVar1 = false;
    }
}
if (bVar1) {
```

Checkout 0x004090e9, I am guessing this is the flag

```

4090e0 73cf02bf 3491c2e2 aaf7ae83 cd78277b s...4.....x'{'
4090f0 9db22b0d 9d8d919e 540fe9fc 4e01799a ..+.....T...N.y.
409100 27503852 1baa5ddc 5bfeae87 d977081a 'P8R...].[....w..
409110 e7dd592a eed2f6ff 30688c88 3d5e0eff ..Y*....0h..=^..
409120 55356726 73cf02bf 3491c2e2 aa03751a U5g&s...4.....u.
409130 e7dd592a eed2f6ff 30688c88 3d5e0eff ..Y*....0h..=^..
409140 55356726 73cf02bf 3491c2e2 aac5efa0 U5g&s...4.....
409150 c56d1036 c78f3850 81a0d697 511bacfb .m.6...8P....Q...
409160 58307adf 2c5a1206 12ef71da 57e3a796 X0z.,Z....q.W...
409170 f56e1069 94bc3e4f cff2a297 591bace5 .n.i...>0....Y...
409180 582d7d9e 32504751 1aa36e9f 47f4ae84 X-}.2PGQ..n.G...
409190 8a671069 93af2c49 9af29f91 1e46a282 .q.i...I....F...

```

We found the encrypted flag, now here is how to decrypt

```

void __Z12DecodeParamsv(void)
{
    uint local_8;

    for (local_8 = 0; local_8 < 100; local_8 = local_8 + 1) {
        (&DAT_00409021)[local_8] = (&DAT_00409021)[local_8] ^ (&DAT_00409000)[local_8 & 0x1f];
        (&DAT_00409085)[local_8] = (&DAT_00409085)[local_8] ^ (&DAT_00409000)[local_8 & 0x1f];
        (&DAT_004090e9)[local_8] = (&DAT_004090e9)[local_8] ^ (&DAT_00409000)[local_8 & 0x1f];
        (&DAT_0040914d)[local_8] = (&DAT_0040914d)[local_8] ^ (&DAT_00409000)[local_8 & 0x1f];
        (&DAT_004091b1)[local_8] = (&DAT_004091b1)[local_8] ^ (&DAT_00409000)[local_8 & 0x1f];
    }
    DAT_00409020 = 0;
    return;
}

```

Now, a simple python script can simply decrypt this.