ez_ollvm_M4yb3

这道题考点是ollvm+父进程保护+魔改rc4

ollvm用的是github上的一个魔改项目，解完混淆后的

源码如下：

```
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>
#include <sstream>
#include <cstdint>
#include <fstream>
#include <algorithm>
#include <csignal>
#include <cerrno>
#include <cctype>

#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/prctl.h>


static std::string to_lower(const std::string &s)
{
    std::string r = s;
    std::transform(r.begin(), r.end(), r.begin(),
                   [](unsigned char c)
                   { return std::tolower(c); });
    return r;
}

static std::string get_process_name(pid_t pid)
{
    std::string path = "/proc/" + std::to_string(pid) + "/comm";
    std::ifstream fin(path);
    std::string name;
    if (fin)
    {
        std::getline(fin, name);
    }
    return name;
}

static bool check_tracer_pid(pid_t pid)
{
    std::string path = "/proc/" + std::to_string(pid) + "/status";
    std::ifstream fin(path);
    if (!fin)
        return false;

    std::string line;
    while (std::getline(fin, line))
    {
```

```
        if (line.rfind("TracerPid:", 0) == 0)
        {
            std::string num_str = line.substr(std::string("TracerPid:").size());
            int tracer = 0;
            try
            {
                tracer = std::stoi(num_str);
            }
            catch (...)
            {
                tracer = 0;
            }
            return tracer != 0;
        }
    }
    return false;
}

class RC4_1024
{
public:
    RC4_1024(const std::vector<uint8_t> &key)
    {
        init(key);
    }

    void crypt(std::vector<uint8_t> &data)
    {
        uint16_t i = 0;
        uint16_t j = 0;

        for (size_t n = 0; n < data.size(); ++n)
        {
            i = static_cast<uint16_t>((i + 1) % N);
            j = static_cast<uint16_t>((j + S[i]) % N);
            std::swap(S[i], S[j]);
            uint16_t idx = static_cast<uint16_t>((S[i] + S[j]) % N);
            uint8_t k = static_cast<uint8_t>(S[idx] & 0xFF);
            data[n] ^= (k ^ 0x91) - 0x45;
        }
    }

private:
    static constexpr size_t N = 1024;
    uint16_t S[N];

    void init(const std::vector<uint8_t> &key)
    {
        for (size_t i = 0; i < N; ++i)
        {
            S[i] = static_cast<uint16_t>(i);
        }

        uint16_t j = 0;
        for (size_t i = 0; i < N; ++i)
        {
            j = static_cast<uint16_t>(
                (j + S[i] + key[i % key.size()]) % N);
            std::swap(S[i], S[j]);
```

```
        }
    }
};

static void real_work()
{
    if (check_tracer_pid(getpid()))
    {
        _exit(1);
    }

    std::vector<uint8_t> key = {'V', 'C', 'T', 'F'};

    const std::string TARGET_HEX =
        "f5c98cef5e185466a5014f6d655088211128c310c8b330e18722ee885506c51df5817343c6ee";

    std::string plainStr;
    std::cout << "plz input your flag" << std::endl;
    if (!std::getline(std::cin, plainStr))
    {
        std::cout << "wrong" << std::endl;
        return;
    }

    std::vector<uint8_t> data(plainStr.begin(), plainStr.end());

    RC4_1024 rc4(key);
    rc4.crypt(data);

    std::ostringstream oss;
    oss << std::hex << std::setfill('0') << std::nouppercase;
    for (size_t i = 0; i < data.size(); ++i)
    {
        oss << std::setw(2) << static_cast<int>(data[i]);
    }
    std::string hexCipher = oss.str();

    if (hexCipher == TARGET_HEX)
    {
        std::cout << "congratulate!" << std::endl;
    }
    else
    {
        std::cout << "wrong" << std::endl;
    }
}

int main()
{
    if (check_tracer_pid(getpid()))
    {
        return 1;
    }

    pid_t ppid = getppid();
    std::string parent_name = get_process_name(ppid);

    prctl(PR_SET_DUMPABLE, 0);
```

```
    pid_t child = fork();
    if (child < 0)
    {
        return 1;
    }

    if (child == 0)
    {
        real_work();
        return 0;
    }
    else
    {
        int status = 0;
        while (true)
        {
            pid_t r = waitpid(child, &status, WNOHANG);
            if (r == child)
            {
                break;
            }

            if (check_tracer_pid(child) || check_tracer_pid(getpid()))
            {
                kill(child, SIGKILL);
                break;
            }

            usleep(500 * 1000);
        }
    }

    return 0;
}
```

exp:

```
TARGET_HEX = "f5c98cef5e185466a5014f6d655088211128c310c8b330e18722ee885506c51df5817343c6ee"
KEY = b"VCTF"

N = 1024

def rc4_1024_ksa(key: bytes):
    """按题目中的 C++ 代码做 1024 长度的 KSA 初始化 S 数组"""
    S = list(range(N))
    j = 0
    key_len = len(key)
    for i in range(N):
        j = (j + S[i] + key[i % key_len]) % N
        S[i], S[j] = S[j], S[i]
    return S

def rc4_1024_crypt(data: bytes, key: bytes) -> bytes:
    S = rc4_1024_ksa(key)
    i = 0
    j = 0
    out = bytearray(data)
```

```
    for n in range(len(out)):
        i = (i + 1) % N
        j = (j + S[i]) % N
        S[i], S[j] = S[j], S[i]
        idx = (S[i] + S[j]) % N
        k = S[idx] & 0xFF

        # C++ 中是 uint8_t ^= ((k ^ 0x91) - 0x45) ，按 8 位截断
        mask = ((k ^ 0x91) - 0x45) & 0xFF
        out[n] ^= mask

    return bytes(out)

def main():
    cipher = bytes.fromhex(TARGET_HEX)
    key = KEY

    plain = rc4_1024_crypt(cipher, key)
    print("Plain bytes:", plain)
    try:
        print("Plain text:", plain.decode())
    except UnicodeDecodeError:
        print("Plain (utf-8 decode error), repr:", repr(plain))

if __name__ == "__main__":
    main()
```