

CONTI

CONTI 랜섬웨어 분석하기

TRUEBIRD

Blog

[HTTPS://R-EVER-SCP.TISTORY.COM/](https://r-ever-scp.tistory.com/)

Agenda

01

01

Conti 랜섬웨어란 무엇인가

02

Conti 랜섬웨어 분석 (혼자)

03

Conti 랜섬웨어 분석(보고서)

04

왜 내 프로젝트는 끝이 아닌가

05

Q & A

Q & A

02



<https://www.slido.com/>

Code : 3783040

01.

WHAT IS CONTI

01

SMALL

2019년 부터 시작되어 현재 까지 가장 공격
적인 랜섬웨어

02

MEDIUM

-
- 스피어 피싱
 - 같은 네트워크 사용시 SMB 취약점을
이용하여 전파

03

LARGE

-
- 자료유출
 - 볼륨 웨도우 복사본 삭제
 - 로컬 시스템의 복구 방해
 - 파일의 암호화

02.

CONTI DETAILS

01

API

- CreateFileW: 파일이나 장치를 열고 핸들을 반환하는 함수
- VirtualAlloc: 프로세스의 가상 주소 공간에서 메모리 할당 하는 함수
- FindFirstFileExW: 디렉터리에서 첫 번째 파일 또는 하위 디렉터리를 찾는 함수
- FindNextFileW: FindFirstFile 함수를 호출하여 검색한 파일 검색 핸들에서 다음 파일 또는 하위 디렉터리를 검색하는 함수

```

77 HeapAlloc: 프로세스의 힙(메모리 관리 단위)에 메모리를 할당하는 함수
78
79 HeapFree: 프로세스의 힙에서 메모리를 해제하는 함수
80
81 FindClose: FindFirstFile 함수로 찾은 파일 검색 핸들을 닫는 함수
82
83 FindFirstFileExW: 디렉터리에서 첫 번째 파일 또는 하위 디렉터리를 찾는 함수
84
85 FindNextFileW: FindFirstFile 함수를 호출하여 검색한 파일 검색 핸들에서 다음 파일 또는 하위 디렉터리를 검색하는 함수
86
87 InvalidCodePage: 유효한 코드 페이지 인자 여부를 확인하는 함수
88
89 GetACP: 현재 ANSI 코드 페이지 식별자를 반환하는 함수
90
91 GetOEMCP: 현재 OEM 코드 페이지 식별자를 반환하는 함수
92
93 GetCPInfo: 지정된 코드 페이지의 정보를 반환하는 함수
94
95 GetCommandLine: ANSI 문자열로 현재 프로세스의 명령줄 인수를 반환하는 함수
96
97 GetCommandLineW: 유니코드 문자열로 현재 프로세스의 명령줄 인수를 반환하는 함수
98
99 MultiByteToWideChar: 멀티바이트 문자열을 유니코드 문자열로 변환하는 함수
100
101 WideCharToMultiByte: 유니코드 문자열을 멀티바이트 문자열로 변환하는 함수
102
103 GetEnvironmentStrings: 현재 프로세스의 환경 블록에 대한 포인터를 반환하는 함수
104
105 FreeEnvironmentStrings: 환경 블록을 해제하는 함수
106
107 LocalizedString: 지정된 로컬을 사용하여 문자열 매핑을 수행하는 함수
108
109 GetProcessHeap: 현재 프로세스의 힙 핸들을 반환하는 함수
110
111 GetFileType: 파일 핸들의 유형을 반환하는 함수
112
113 SetStdHandle: 표준 입력, 출력 또는 오류
114
115 GetStringTypeW: 문자열의 문자 다음을 결정하는 데 사용됩니다. 유니코드 문자열의 경우, 문자는 16비트로 표시되고, ANSI 문자열의 경우, 문자는 8비트로 표시됩니다.
116
117 HeapSize: 지정된 힙 블록의 크기를 검색합니다.
118
119 HeapReAlloc: 힙의 메모리 블록 크기를 변경합니다.
120
121 FlushFileBuffers: 파일 버퍼의 내용을 디스크에 쓰기 위해 파일 핸들에 대한 캐시 된 데이터를 강제로 버립니다.
122
123 GetConsoleCP: 현재 콘솔의 입력 코드 페이지를 반환합니다.
124
125 GetConsoleMode: 지정된 핸들이 참조하는 콘솔 입력 모드 또는 콘솔 출력 모드를 검색합니다.
126
127 DecodePointer: 인코딩된 포인터 값을 원래 포인터 값으로 디코딩합니다.
128
129 wprintf: 유니코드 문자열 형식 지정 함수입니다. 문자열에 값을 삽입하여 새 문자열을 생성할 수 있습니다.
130
131

```

02

DECRYPTION

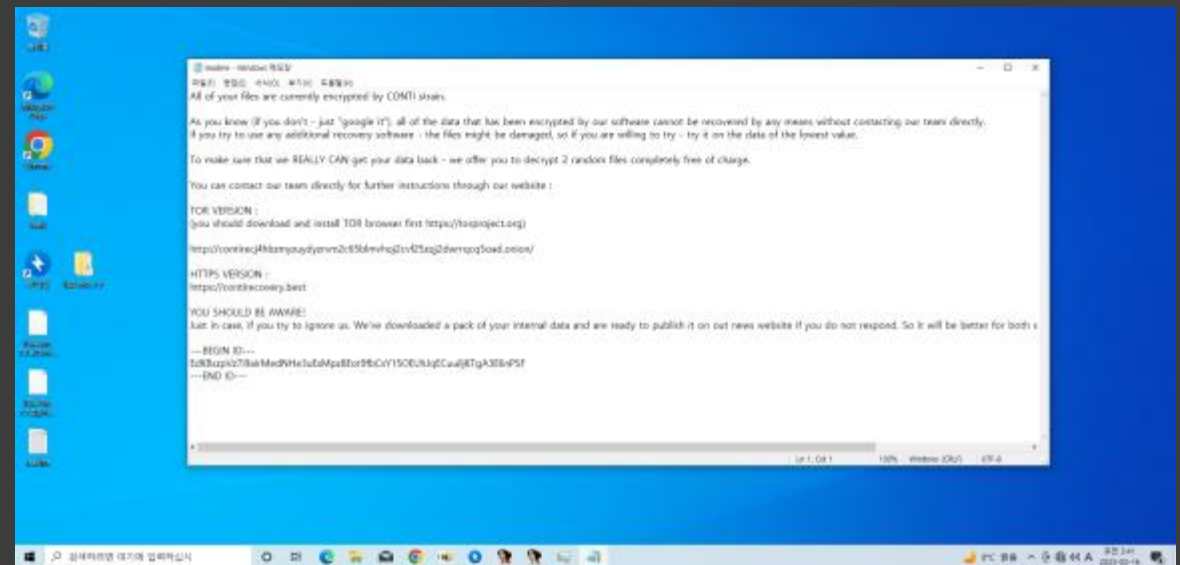
- 문자열 복호화 : 대부분의 문자열을 복호화 한다.
- API 주소 복호화 : API를 사용 할 때마다 API주소를 동적으로 복호화 하여 사용 (Call 할때마다 BP 걸어놓고 해야함 == 노가다)

```
19 v7[7] = 1;  
20 v7[8] = 67;  
21 v7[9] = 1;  
22 v7[10] = 59;  
23 v7[11] = 1;  
24 v7[12] = 106;  
25 v7[13] = 1;  
26 v7[14] = 89;  
27 v7[15] = 1;  
28 v7[16] = 21;  
29 v7[17] = 1;  
30 v7[18] = 50;  
31 v7[19] = 1;  
32 v7[20] = 59;  
33 v7[21] = 1;  
34 v7[22] = 59;  
35 v7[23] = 1;  
36 v7[24] = 1;  
37 v7[25] = 1;  
38 for ( i = 0; i < 0x1A; ++i )  
39     v7[i] = ((15 * (unsigned __int8)v7[i] - 15) % 127 + 127) % 127;  
40 Ldr = NtCurrentPeb()->Ldr;  
41 for ( j = (int)&Ldr[62331].InInitializationOrderModuleList.Blink + 1; !(j % 4); ++j )  
42     ;  
43 Blink = Ldr->InLoadOrderModuleList.Blink;
```


03

RANSOMNOTE

- TOR 버전을 이용한 것과 HTTPS 버전으로 접속하는 방법이 나뉨
- 현재 HTTPS 방식은 막힘



03.

CONTI DETAILS

01

DECRYPTION

- 사용할 DLL 찾은 후 해당 라이브러리의 모듈의 핸들을 얻어온다.

```
switch ( dll_index )
{
    case 15:
        break;
    case 16:
        dll = Advapi32_dll;
        break;
    case 17:
        dll = Netapi32_dll;
        break;
    case 18:
        dll = Iphlpapi_dll;
        break;
    case 19:
        dll = Rstrtmgr_dll;
        break;
    case 20:
        dll = User32_dll;
        break;
    case 21:
        dll = ws2_32_dll;
        break;
    case 22:
        dll = Shlwapi_dll;
        break;
    case 23:
        dll = Shell32_dll;
        break;
    case 24:
        dll = Ole32_dll;
        break;
    case 25:
        dll = OleAut32_dll;
        break;
    case 26:
        dll = ntdll_dll;
        break;
    default:
        return 0;
}
dll_handle = LoadLibraryA_dword_430BA0(dll);
```

01

DECRYPTION

- 원하는 API를 불러온다. 불러오는 과정에서 API의 HASH, 모듈 핸들, API 주소를 저장할 버퍼를 인자로 넘긴다.

```
api_addr = get_api_4033D0(api_hash, dll_handle, &api_addr);
kernel32_dll_ = (api_addr + 2991921);
if ( (api_addr + 2991921) % 4 )
    return api_addr;
do
    ++kernel32_dll_;
while ( !(kernel32_dll_ % 4) );
return api_addr;
```

01

DECRYPTION

- 수동으로 EAT 위치 찾기
- EAT를 순회하며 원하는 API 찾기
- 찾은 API의 주소를 찾아서 반환
- EAT?

라이브러리 파일에서 제공하는 함수를 다른 프로그램에서 쓸수 있도록 해주는 메커니즘

```
api_hash_2 = api_hash;
e_lfanew = *(lib_module_handle + 0x3C);
export_dir = (lib_module_handle + *(e_lfanew + lib_module_handle + 0x78));
addr_of_import_dir = *(e_lfanew + lib_module_handle + 0x7C); // DB88
export_dir_1 = export_dir; // DF1A
if ( HIWORD(api_hash) && (v7 = 0, v8 = (v4 + export_dir[8]), addr_of_ordinals = (v4 + export_dir[9]), export_dir[6]
{
    while ( 1 ) // EAT순회
    {
        api_name = (v4 + *v8);
        api_name_length = 0;
        if ( *api_name )
        {
            do
                ++api_name_length;
            while ( *(api_name + api_name_length) );
        }
        if ( api_hashing(api_name, api_name_length) == api_hash_2 ) // API 해싱 -> 얻고자 하는 API와 일치하는지 확인
            break;
        ++v7;
        v4 = lib_module_handle;
        ++v8;
        ++addr_of_ordinals;
        if ( v7 >= *(export_dir_1 + 24) )
            goto LABEL_7;
    }
    v13 = find_api_addr_402FB0(export_dir_1, lib_module_handle, *addr_of_ordinals); // 해당하는 API 주소 찾기
    if ( find_api_addr_402CB0(export_dir_1, v13, addr_of_import_dir) )
        v13 = sub_401C90(v13);
    API_addr = v13;
}
```

02

THREAD

- 빠른 암호화 속도를 구현하기 위해 멀티 쓰레드
- 프로세서의 갯수 == 쓰레드의 갯수

```
GetNativeSystemInfo v12 = get_api 405250(0xF, 0xDF1AF05E, 0x13);  
(GetNativeSystemInfo_v12)(SYSTEM_INFO); // 0x9 == x64(Intel or AMD)  
j = SYSTEM_INFO[0] + 0x2DA731;  
v12 = (j & 0x80000003) == 0;  
if ( j < 0 )  
    v12 = (((LOBYTE(SYSTEM_INFO[0]) + 0x31) & 3) - 1) | 0xFFFFFFFFC == -1;  
if ( v12 )  
{  
    do  
        ++j;  
    while ( !(j % 4) );  
}  
// 생성할 쓰레드의 갯수를 설정  
if ( opt_code_42E880 == 0xA )  
    num_of_cpu_thread = num_of_processors;  
else  
    num_of_cpu_thread = 2 * num_of_processors;
```

02

THREAD

- 멀티 쓰레드를 효율적으로 관리하기 위한 구조체와 버퍼를 생성한다

```
if ( opt_code_42E880 == 0xA || opt_code_42E880 == 0xB )// default / -m local
{
    if ( !init_thread_struct_418760(0, num_of_cpu_thread) )
        return 1;
    for ( j = num_of_cpu_thread + 2991921; !(j % 4); ++j )
        ;
    if ( !launch_enc_thread_4187E0(0) ) // local
        return 1;
    for ( j = num_of_cpu_thread + 2991921; !(j % 4); ++j )
        ;
}
```

03

ENCRYPTION

- 디렉터리들을 순회하며 파일을 탐색한다.
- 파일을 찾았다면 . 와 .. 비교 , 심볼릭 링크인지 검사
- 디렉터리를 찾았다면 화이트 리스트에 포함되는지 검사한다. 화이트 리스트 목록에는 시스템 동작에 지장이 갈 수 있는 폴더들이 포함된다.
- RSA256로 암호화 합니다.

```
FindFirstFileW_v13 = get_api_405250(15, -491516027, v56);
search_handle = (FindFirstFileW_v13)(lpFileName, lpFindFileData);
```

```
qmemcpy(str_dot, "\aFFF", 4);
for ( j = 0; j < 4; ++j )
    str_dot[j] = (35 * (70 - str_dot[j]) % 127 + 127) % 127; // "."
Driver_Path_3 = Drive_Path_2;
lstrcpw v29 = get_api_405250(15, 964366815, 49);
if ( (lstrcpw v29)(v90, str_dot) ) // 유효한 파일인지 확인
{
    str_dot[7] = 0;
    qmemcpy(&str_dotdot, "n\n"]", 6);
    for ( k = 0; k < 6; ++k )
        *(&str_dotdot + k) = (55 * (*(&str_dotdot + k) - 93) % 127 + 127) % 127; // ".."
    lstrcpw v31 = get_api_405250(15, 964366815, 49);
    if ( (lstrcpw v31)(v90, &str_dotdot) ) / 유효한 파일인지 확인
    {
        if ( (lpFindFileData[0] & 0x400) == 0 ) / 0x400 : symbolic link
        {
```

```
if ( (lpFindFileData[0] & 0x10) != 0 && WhiteList_Folder_414EE0(v90) )
{
    v61 = 0;
    v62 = 7;
    LOWORD(Src) = 0;
    wide_mem_mov_414800(&Src, v90, wcslen(v90));
    mem_mov_416800(&v51, v82);
    join_str_4148D0(v51, v52, v53, v54, v55, v56, Src, str_slash, v59, v
    v31 = operator new(0x20u);
    v31[4] = 0;
    v31[5] = 7;
    *v31 = 0;
```


04

NETWORK

- ARP 캐시 조회를 통해 네트워크 스캔을 수행합니다.

```
GetIpNetTable_2 = get_api_405250(18, -1080542143, 60);  
if ( (GetIpNetTable_2)(nettable_buffer, &Size_ptr, 0) )// 성공 시 0  
{  
    GetLastError_2 = get_api_405250(15, 532396111, 16);  
    last_error_2 = GetLastError_2();  
    v39[0] = 92;  
    v39[1] = 73;  
    v39[2] = 41;  
    v39[3] = 73;  
    v39[4] = 79;  
    v39[5] = 73;  
    v39[6] = 114;  
    v39[7] = 73;  
    v39[8] = 35;  
    v39[9] = 73;  
    v39[10] = 42;  
    v39[11] = 73;  
    v39[12] = 41;  
    v39[13] = 73;
```

```
inet_ntoa_v16 = get_api_405250(21, -868764469, 109);  
dotted_decimal_IP = (inet_ntoa_v16)(nettable_ptr_2);// dotted_decimal_IP로 변환  
WSAGetLastError();
```

04

NETWORK

- ARP캐시로 얻은 IP의 목록이 네트워크 대역과 일치하는지 확인한다.

```
v20 = (StrStrIA_v19)(dotted_decimal_IP, IP_band_172); // IP scan: 172.(B Class)
v42[70] = 0;
Cclass_v43[0] = 114;
Cclass_v43[1] = 108;
Cclass_v43[2] = 18;
Cclass_v43[3] = 21;
Cclass_v43[4] = 114;
Cclass_v43[5] = 15;
Cclass_v43[6] = 77;
Cclass_v43[7] = 21;
Cclass_v43[8] = 119;
v35 = v20;
for ( l = 0; l < 9; ++l )
    Cclass_v43[l] = (41 * (Cclass_v43[l] - 119) % 127 + 127) % 127;
StrStrIA_v22 = get_api_405250(22, 1752676342, 73); // IP Scan: 192. (C Class)
v23 = (StrStrIA_v22)(dotted_decimal_IP, Cclass_v43);
IP_band_172[7] = 0;
v46 = 15;
qmemcpy(v47, "{Ud", sizeof(v47));
v34 = v23;
for ( m = 0; m < 4; ++m )
    *(&v46 + m) = (20 * (100 - *(&v46 + m)) % 127 + 127) % 127;
StrStrIA_v25 = get_api_405250(22, 1752676342, 73);
v26 = (StrStrIA_v25)(dotted_decimal_IP, &v46); // IP scan: 10.(A Class)
Cclass_v43[11] = 0;
v27 = v26;
qmemcpy(v44, "\\")`jC", 5);
for ( n = 0; n < 5; ++n )
    v44[n] = (37 * (v44[n] - 67) % 127 + 127) % 127; // 169.(APIPA 대역)
StrStrIA_v29 = get_api_405250(22, 1752676342, 73);
v30 = (StrStrIA_v29)(dotted_decimal_IP, v44);
```

04

NETWORK

- 공유 네트워크를 검색합니다.
- 공유 폴더 목록을 얻는다.

```
hThread = (CreateThread_v8)(0, 0, get_shared_resource_sub_419660, 0, 0, 0); // 공유 네트워크 탐색
if ( hThread == -1 )
{
    v26[0] = 0;
    v26[1] = 45;
    v26[2] = 22;
    v26[3] = 97;
    v26[4] = 22;
    v26[5] = 1;
    v26[6] = 22;

    while ( 1 )
    {
        NetShareEnum = get_api_405250(17, 375969649, 59);
        result = (NetShareEnum)(v2, 1, &bufptr, -1, &v31, &v29, &v30);
        if ( !result )
            break;
        if ( result != 0xEA )
            return result;
    }
}
```

04

NETWORK

- IP 주소 + 공유 폴더 이름을 조합하여 실제로 접근 가능한 주소로 변환한다.
- ADMIN\$면 windows 폴더를 의미하기 때문에 암호화 대상에서 제외한다.

```

if ( net_share_name )
{
    v23[35] = 0;
    str_ADMIN[0] = 63;
    str_ADMIN[1] = 62;
    str_ADMIN[2] = 65;
    str_ADMIN[3] = 62;
    str_ADMIN[4] = 71;
    str_ADMIN[5] = 62;
    str_ADMIN[6] = 26;
    qmemcpy(v25, ">V>>>", 7);
    for ( i = 0; i < 0xE; ++i )
        str_ADMIN[i] = (62 * (62 - str_ADMIN[i]) % 127 + 127) % 127; // ADMIN$
    bufptr_2 = *bufptr_1;
    lstrcpw = get_api_405250(15, -684828759, 28);
    if ( (lstrcpw)(bufptr_2, str_ADMIN) ) // if not ADMIN$
    {
        v25[8] = 0;
        v26[0] = 17;
        v26[1] = 1;
        v26[2] = 17;
        v26[3] = 1;
        v26[4] = 1;
        v26[5] = 1;
        for ( j = 0; j < 6; ++j )
            v26[j] = ((26 - 26 * v26[j]) % 127 + 127) % 127; // \\
        lstrcpyW = get_api_405250(15, 1301742288, 22);
        (lstrcpyW)(net_share_name, v26);
        lstrcatW = get_api_405250(15, 129639993, 17);
        (lstrcatW)(net_share_name, a1); // \\{IP}\{Share name}
        v26[7] = 0;
        qmemcpy(v27, "\"444\"", 4); // \
        for ( k = 0; k < 4; ++k )
            v27[k] = (9 * (v27[k] - 52) % 127 + 127) % 127;
        lstrcatW_1 = get_api_405250(15, 129639993, 17);
        (lstrcatW_1)(net_share_name, v27);
        v15 = *bufptr_1;
        lstrcatW_2 = get_api_405250(15, 129639993, 17);
        (lstrcatW_2)(net_share_name, v15);
    }
}

```

04.

FAIL?
NONFAIL?



01

NOT YET

- 다른 사람들과 하는 프로젝트라 아직 끝나지 않음
- 길게 잡고 한 프로젝트라 아직 초입
- 결과물은 아직 구체화 하지 못했지만 아마 복호화
툴 제작

Q & A

HAPPY OR UNHAPPY

TRUEBIRD

로그인 구독 바랍니다