情報保護學會論文誌
第 21 卷 第 2 號, 2011. 4

# 윈도우즈 Crypto API를 이용한 악성코드 무력화 방안 연구 및 도구 구현

송 정 환,† 황 인 태‡
한양대학교

# A study on neutralization malicious code using Windows Crypto API and an implementation of Crypto API hooking tool

Jung-hwan Song,† In-tae Hwang‡
Hanyang University

요 약

암호기술의 발전으로 통신의 비밀이나 정보보호가 강화되었지만, 암호기술을 역이용하는 Cryptovirus가 등장하였고 윈도우즈 CAPI(Crypto API)를 사용하는 악성코드도 누구나 쉽게 제작하여 배포할 수 있는 환경이 되었다. CAPI를 사용하는 악성코드는 윈도우즈에서 제공하는 정상적인 API를 사용하기 때문에 IPS(Intrusion Prevention System) 등 정보보호시스템은 물론 백신프로그램에서도 탐지 및 분석이 쉽지 않다. 본 논문에서는 Cryptovirus를 비롯하여 윈도우즈 CAPI를 사용하는 악성코드 무력화 방안 연구 및 이와 관련된 Hooking 도구 구현결과를 제시하고자 한다.

ABSTRACT

Advances in encryption technology to secret communication and information security has been strengthened. Cryptovirus is the advent of encryption technology to exploit. Also, anyone can build and deploy malicious code using windows CAPI. Cryptovirus and malicious code using windows CAPI use the normal windows API. So vaccine software and security system are difficult to detect and analyze them. This paper examines and make hooking tool against Crytovirus and malicious code using windows CAPI.

**Keywords**: API Hooking, Crypto API, Cryptovirus

## I. 서 론

일반적으로 컴퓨터 바이러스(Computer Virus)는 컴퓨터의 정상적인 사용환경을 방해하는 악성코드의 일종이다. 인터넷의 급성장과 더불어 컴퓨터 통신 이용률의 폭발적인 증가를 통해 이러한 컴퓨터 바이러스가 네트워크를 통해 감염되는 웜의 특징이 결합되어 웜바이러스의 형태로 발전되었다. 안철수연구소가 발간한 악성코드 연간 동향보고서에 따르면 악성코드 탐지 건수가 2005년도에 2만여건에서 2010년도에는 무려 2900만건으로 분석되었다[1]. 인터넷의 발달로 인한 악성코드의 전파력 증가 및 백신프로그램의 악성코드 탐지 성능 향상에 기인했다고 볼 수 있다.

악성코드수의 증가와 함께 최근에는 중요자료 및 통신데이터 보호 등의 목적으로 사용되는 암호기술이 발전하면서 Cryptovirus와 같이 암호기술을 역이용하는 악성코드가 등장하였다. 특히, 윈도우즈에서 사용중인 정상적인 CAPI(Crypto API)를 사용하여

접수일(2011년 1월 24일), 수정일(2011년 3월 23일), 게재 확정일(2011년 4월 13일)
† 주저자, camp123@hanyang.ac.kr
‡ 교신저자, fornewt@gmail.com

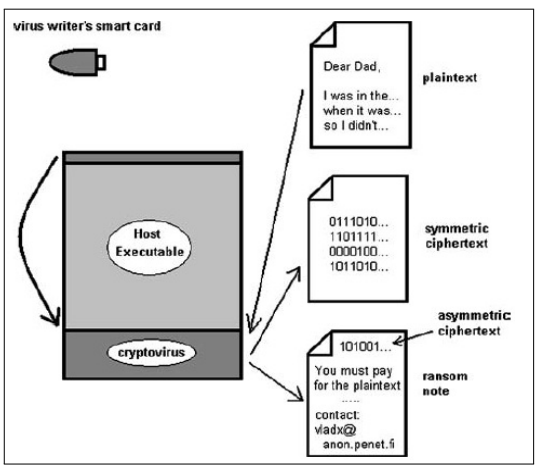
악성코드의 행위를 암호화시켜 놓거나 감염된 PC에 보관된 개인파일 등을 암호화하기 때문에 백신프로그램에서는 악성코드로 분류하여 탐지하는데 한계가 있다. 게다가 암호화된 파일을 네트워크를 통해서 전송시키는 경우에는 IPS(Intrusion Prevention System) 등 정보보호시스템마저도 무용지물이 되고 만다.

본 논문에서는 Cryptovirus 뿐만 아니라 윈도우즈 CAPI를 이용한 악성코드의 행위를 감시할 수 있는 방안에 대해서 연구하고 이와 관련된 Hooking 도구를 직접 구현하여 테스트한 결과를 제시한다.

## II. CAPI 사용 악성코드

### 2.1 Cryptovirus

1996년 Yung과 Young에 의해서 처음 소개된 개념으로 불법적인 접근을 방지하기 위한 방어개념의 암호기술을 악의적으로 바이러스에 이용하여 공격에도 활용될 수 있음을 제시하였다[2, 3]. 이와같이 암호기술을 이용한 바이러스를 일반적으로 Cryptovirus라 부르며 이에 대한 연구를 Cryptovirology라 한다. Cryptovirus의 구조는 다음 [그림 1]과 같다.

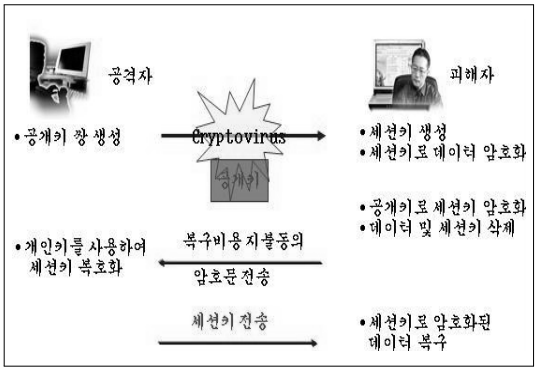



[그림 1] Cryptovirus 구조

Cryptovirus의 공격과정은 다음과 같이 4단계로 나타낼 수 있다[8].

- 1단계: 공격자는 공개키 암호 알고리즘을 사용하여 공개키와 개인키를 생성한 후, 공개키가 포함된 Cryptovirus를 배포한다.
- 2단계: 배포된 Cryptovirus가 피해자 PC에서 실행시 윈도우즈 CAPI를 이용하여 세션키(대칭키)를 생성한 후, 피해자 PC에 보관된 데이터를 세션키(대칭키)로 암호화한다. 1단계에서 생성한 Cryptovirus내 공개키를 이용하여 세션키(대칭키)를 암호화하고, 결과값을 공격자에게 전송한다. 이후 암호화된 데이터를 제외하고 피해자의 원본 데이터와 세션키(대칭키)는 삭제한다.
- 3단계: 공격자는 세션키(대칭키) 값을 알려주는 대가를 피해자에게 요구하고, 피해자가 이에 응할 경우 공격자가 1단계에서 생성한 개인키로 세션키(대칭키)를 복호화한다.
- 4단계: 피해자는 공격자가 전송해 준 세션키(대칭키)를 사용하여 암호화된 데이터를 복구한다.

[그림 2]는 Cryptovirus 단계별 공격과정을 도식화하여 나타낸 것이다.




[그림 2] Cryptovirus 공격 전체개요도

### 2.2 GPCODE

2005년 5월경 윈도우즈 탐색기의 취약점을 통해 감염되면 사용자 PC의 문서(XLS, DOC 등)나 그림파일(JPG 등)을 암호화하는 Virus.Win32.Gpcode.b 악성코드가 발견되었는데, 제작자는 악성코드와 함께 해독 프로그램을 구할려면 200달러를 요구하는 메시지가 담긴 ATTENTION!!!.txt[그림 3] 파일을 동시에 배포하였다[4]. 국내에서는 '인질범 바이러스'로 불리기도 하였다. 해외 특정사이트를 방문하면 감염이 되며, 감염시 증상은 XLS, TXT, ZIP 등을 암호화시켜 피해자는 자신의 PC에 보관된 문서파일 실행이 불가능하게 된다. 이후 ATTENTION!!!.txt 파일이 피해자 PC 곳곳에 자동으로 생성된다.

Some files are coded.
To buy decoder mail: n{removed}@yahoo.com
with subject: PGPcoder 000000000032

〔그림 3〕 ATTENTION!!!.txt 파일 내용

### 2.3 Trojan.Zbot.B!inf

2010년 10월 1일 최초 보고된 이 악성코드는 Trojan.Zbot 악성코드의 업데이트 버전으로 확인되었으며, 이 악성코드는 윈도우즈 CAPI를 사용하여 악성코드를 다운로드할 URL을 생성하고 악성코드를 검증하는 용도로 사용되었다[5].

이외에도 윈도우즈 CAPI를 사용하면 목적에 따라 중요자료 절취, 피해 PC 시스템파일 손상 등 다양한 방법으로 활용이 가능하다.

## III. CAPI 사용 악성코드 Hooking

CAPI 사용 악성코드는 문자열이나 파일 등을 암호화 또는 복호화하기 위해서 윈도우즈에서 제공하는 암호알고리즘을 사용해야 한다. 본 절에서는 API Hooking 기법을 통해 악성코드가 CAPI에서 제공하는 암호알고리즘, 암호키 등을 호출하는 시점을 감시하여 그 내용을 가로챌 수 있는 방안을 제시하기로 한다.

### 3.1 악성코드 여부 판별

윈도우즈 CAPI는 암호화 또는 복호화를 위해 각종 프로그램에서 많이 사용되고 있다. 예를 들어 인터넷 뱅킹 키보드 입력정보 암호화나 온라인 게임 아이템정보 암호화 등 인터넷통신이나 중요자료 보호 목적으로 사용되고 있다. 따라서 CAPI를 사용하는 프로그램에 대한 악성코드 여부 판별에 대한 방법은 본 논문에서 제외하기로 하고, CAPI를 사용하는 악성코드로 범위를 제한하여 API Hooking 방안을 제시한다.

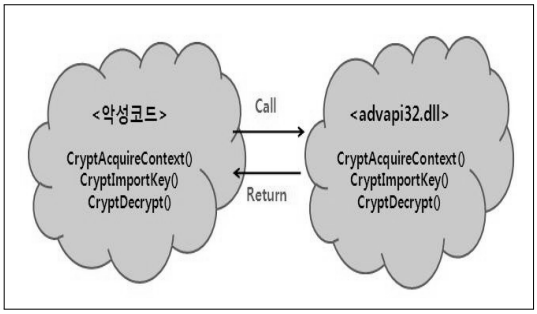
### 3.2 API Hooking

윈도우즈 운영체제에서는 사용자 어플리케이션이 메모리, 파일, 네트워크, 비디오, 사운드 등 시스템 자원을 사용하고 싶을 때 직접적인 접근이 불가능하다. 왜냐하면 윈도우즈 운영체제에서 직접 관리를 하며 안정성, 보안, 효율성 등의 이유로 인해 사용자 어플리케이션의 직접적인 접근을 차단해 놓았기 때문이다. 그래서 사용자 어플리케이션은 Win32 API를 호출하여 시스템 커널에게 시스템 자원 사용을 요청한다.

API Hooking이란 사용자 어플리케이션이 요청한 Win32 API 호출을 중간에 가로채서 제어권을 얻어내는 기법이다[9]. 본 논문에서 소개한 CAPI를 사용하는 악성코드도 사용자 어플리케이션의 일부이기 때문에 API Hooking이 가능하다.

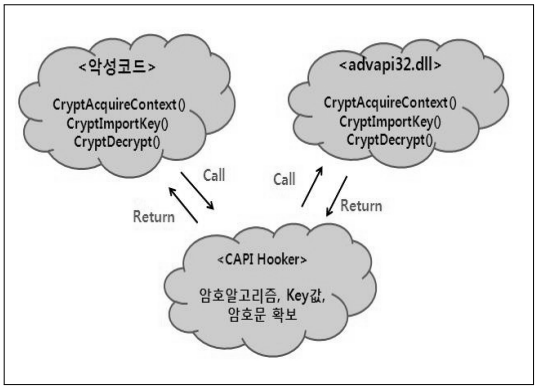
〔그림 4〕는 악성코드가 advapi32.dll에서 제공하는 각종함수를 정상적으로 호출하는 과정을 도식화한 것이다.

- CryptAcquireContext() : 암호알고리즘 호출 함수
- CryptImportKey() : Key 호출 함수
- CryptDecrypt() : 복호화 함수




〔그림 4〕 정상적인 API 호출

〔그림 5〕는 〔그림 4〕에서 설명한 정상적인 API 호출에 API Hooking 단계를 추가하여 악성코드가 사용하는 함수들을 CAPI Hooker가 중간에 가로채어 제어권을 확보한 후, 악성코드 세부행위를 파악할 수 있는 절차를 도식화한 것이다.




〔그림 5〕 API Hooking

이처럼 API Hooking 기법을 이용하면 CAPI를 사용하는 악성코드가 사용중인 암호알고리즘, 암호키, 암호문 또는 평문 등을 확인할 수 있다.

### 3.3 CAPI 사용 악성코드 Hooking

소스코드를 이용하여 컴파일된 실행파일들은 바이너리 형태로 존재하기 때문에 소스코드가 공개되지 않고서는 확인이 불가능하다. 본 논문에서 소개한 CAPI를 사용하는 악성코드도 실행파일 형태로 존재한다. 즉, 악성코드가 사용하는 암호알고리즘, 암호키 등을 확인하기 위해 소스코드를 참고하는 것은 불가능하다.

대신에 윈도우즈에서 제공하는 CAPI를 사용하기 때문에 악성코드는 문자열이나 파일을 암호화 또는 복호화 할 때 CAPI에 정의된 함수를 반드시 사용하여야 한다. 악성코드가 CAPI 함수를 호출하는 시점에 API Hooking을 적용하고, 이에 대한 결과값을 추출하면 암호알고리즘, 암호키 등을 확인할 수 있다. 다음에서는 이에 대한 세부방법과 절차들을 제시하기로 한다.

#### ① 암호알고리즘 Hooking

악성코드의 암호알고리즘 사용과정은 〔그림 6〕과 같이 암호알고리즘 등을 변수로 사용하여 CryptAquireContext() 함수를 호출한다. 호출시점에 API Hooking을 통해 이 함수의 제어권을 넘겨받아 실행시키면 악성코드가 사용하는 암호알고리즘을 확인할 수 있다.

```
CryptAcquireContext(HCRYPTPROV *phProv,
LPCTSTR pszContainer, LPCTSTR pszProvider,
DWORD dwProvType, DWORD dwFlags);
```

〔그림 6〕 CryptAcquireContext() 함수〔8〕

#### ② Key Hooking

악성코드가 암호화 및 복호화에 필요한 Key를 사용하기 위해서는 〔그림 7〕과 같이 CryptImportKey() 함수를 호출한다. 이때 암호알고리즘 Hooking과 동일한 방법을 사용하면 악성코드의 암호키를 확인할 수 있다.

```
CryptImportKey(HCRYPTPROV hProv, BYTE
*pbData, DWORD dwDataLen, HCRYPTKEY
hPubKey, DWORD dwFlags, HCRYPTKEY
*phKey);
```

〔그림 7〕 CryptImportKey() 함수〔8〕

#### ③ 암호문 또는 평문 내용 Hooking

악성코드가 암호문, 평문을 복호화, 암호화 하기위해서는 〔그림 8〕과 같이 CryptEncrypt() 또는 CryptDecrypt() 함수를 호출한다. 이때 암호알고리즘 Hooking과 동일한 방법을 사용하면 암호문 또는 평문의 내용을 확인할 수 있다.

```
CryptEncrypt(HCRYPTKEY hKey,
HCRYPTHASH hHash, BOOL Final, DWORD
dwFlags,BYTE *pbData, DWORD *pdwData-
Len, DWORD dwBufLen);

CryptDecrypt(HCRYPTKEY hKey,
HCRYPTHASH hHash, BOOL Final, DWORD
dwFlags, BYTE *pbData, DWORD *pdwData-
Len);
```

〔그림 8〕 CryptEncrypt(), CryptDecrypt() 함수〔8〕

## IV. CAPI 사용 악성코드 Hooking 도구 구현

본 절에서는 앞서 설명한 CAPI 악성코드 행위에 대한 Hooking 기법을 직접 구현하여 테스트한 결과를 제시한다. 구현한 악성코드와 Hooking 도구는 Windows XP(Service Pack 3) 환경에서 Microsoft Visual C++ 6.0으로 제작하였다.

```
// Base64 인코딩된 암호문
CString g_strEncrypt = _T("nLNb4pFhw076u1XI1SyK57Vt0iGqpJ2GDLhTmTlOIGp7IaaZ2y7WV0kdoUts053uXP8N6P6hf

// 0x254 길이의 암호키
unsigned char Key[0x254] =
{
    0x07, 0x02, 0x00, 0x00, 0x00, 0xA4, 0x00, 0x00, 0x52, 0x53, 0x41, 0x32, 0x00, 0x04, 0x00, 0x00,
    0x01, 0x00, 0x01, 0x00, 0xED, 0x64, 0xE3, 0x72, 0x3F, 0x89, 0x09, 0xA4, 0x34, 0xC2, 0xB1, 0xD0,
    0x23, 0xCB, 0x67, 0x69, 0x81, 0x0E, 0x94, 0xC9, 0x8E, 0xE5, 0x69, 0x32, 0xD7, 0x23, 0x4F, 0x51,
    0x7E, 0xDD, 0x61, 0x99, 0x34, 0x2C, 0x38, 0x20, 0x1B, 0xC7, 0x54, 0x32, 0x4B, 0xB4, 0x16, 0x98,
    0xE8, 0xE8, 0xD8, 0xB1, 0xE8, 0x16, 0x38, 0x86, 0x42, 0x38, 0x70, 0xB2, 0x81, 0x26, 0x17, 0xE8,
    0x26, 0xC6, 0x22, 0x98, 0x39, 0x44, 0xE8, 0x51, 0x6D, 0x78, 0xEB, 0xC7, 0xEE, 0x91, 0x8F, 0xF2,
    0x50, 0xC6, 0xA1, 0xED, 0xB4, 0xE4, 0xF2, 0x2E, 0x64, 0x43, 0x1C, 0x77, 0xFD, 0xB6, 0x25, 0xC0,
    0x5D, 0x29, 0x7A, 0x22, 0xA5, 0x84, 0x1E, 0x35, 0xA5, 0xE1, 0xAC, 0x48, 0x5E, 0x41, 0x05, 0x39,
    0x33, 0x3E, 0x47, 0xB6, 0xCB, 0x89, 0xB0, 0xB7, 0xAB, 0xEB, 0xEA, 0x7D, 0x5D, 0x75, 0xD2, 0x8A,
    0x43, 0x3E, 0x6A, 0xB5, 0x65, 0x8C, 0x5E, 0x2F, 0x50, 0x93, 0xAB, 0xEF, 0x37, 0x88, 0x67, 0xC5,
    0x7D, 0xEE, 0x7B, 0xAB, 0xAD, 0x69, 0x4F, 0x78, 0x13, 0xD0, 0xD2, 0xEB, 0xE1, 0xF9, 0x00, 0x04,
    0x6E, 0xD7, 0x20, 0x71, 0x26, 0x7C, 0x6A, 0x6E, 0x92, 0x70, 0x7F, 0xDB, 0xD2, 0x25, 0x73, 0xE8,
    0x12, 0x65, 0xEE, 0x63, 0x49, 0x1D, 0xF1, 0x5C, 0x44, 0x60, 0xAA, 0x53, 0x4F, 0x90, 0xBA, 0x0B,
    0xBE, 0xF0, 0x4E, 0xDA, 0xE9, 0xD9, 0x44, 0xEE, 0xF2, 0x82, 0x93, 0xDF, 0xE2, 0x7E, 0x80, 0x84,
    0x36, 0x90, 0x76, 0x3A, 0x5D, 0x29, 0xBA, 0xC3, 0x31, 0x41, 0x88, 0xCC, 0x89, 0xAC, 0xB0, 0xEC,
    0xC8, 0x4F, 0x79, 0x55, 0x16, 0xDD, 0xF7, 0x90, 0x43, 0xC8, 0xAA, 0x09, 0xFC, 0x21, 0xFC, 0x41,
```

〔그림 9〕 악성코드 소스코드(암호문, Key값 정의)

### 4.1 CAPI 악성코드 구현

이 소스코드는 본 논문의 연구목적을 보여주기 위해 직접 제작하였기 때문에 확인이 가능하고, 실제로 악성코드는 실행파일로 존재하기 때문에 소스코드 확인은 불가능하다. [그림 9]는 CAPI를 사용하는 악성코드의 소스코드 일부로써 RSA 알고리즘을 사용하여 개인키로 암호화된 문자열이 g_strEncrypt 변수로 정의되어 있다. 또한 악성코드 해독에 필요한 공개키가 Key[0x254] 변수로 정의되어 있다.

또한 [그림 10]의 CAPI 악성코드의 소스코드에는 제 3절에서 설명한 CryptAcquireContext(), CryptImportKey(), CryptDecrypt() 함수가 정의되어 있다.

```
int KeyDecrypt(/*IN*/DWORD dwOutBufLength, /*IN*/BYTE* pbData)
{
    HCRYPTPROV  hProv = NULL;
    HCRYPTHASH  hHash = NULL;
    HCRYPTKEY   hKey  = NULL;

    // Wincrypt API 프로바이더 핸들 얻기
    if(!CryptAcquireContext(&hProv, 0, "Microsoft Base Cryptographic Provider v1.0",
                            PROV_RSA_FULL, 0))
    {
        if(!CryptAcquireContext(&hProv, 0, "Microsoft Base Cryptographic Provider v1.0",
                                PROV_RSA_FULL, 8u))
            return -1;
    }

    // 키 임포트
    if(!CryptImportKey(hProv, (BYTE*)Key, 0x254u, 0, 0, &hKey))
    {
        CryptReleaseContext(hProv, 0);
        return -1;
    }

    // 복호화, pbData 평문이 기록됨
    if(!CryptDecrypt(hKey, 0, 0, 0, (BYTE*)pbData, &dwOutBufLength) )
    {
        CryptDestroyKey(hKey);
        CryptReleaseContext(hProv, 0);
        return -1;
    }
```

[그림 10] 악성코드 소스코드(CAPI 제공함수 적용)

[그림 11]은 CAPI 악성코드를 Build한 후, 실행


[그림 11] CAPI 악성코드 실행화면

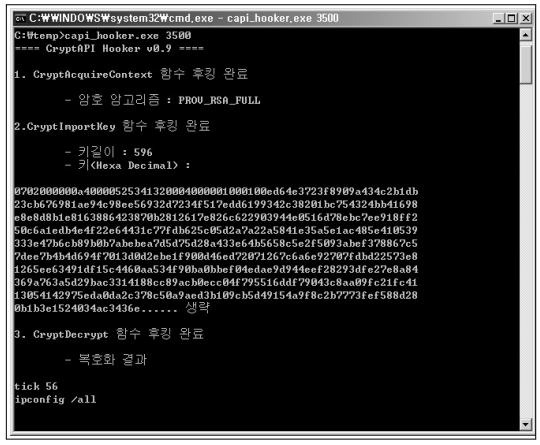
파일로 만들어 실행한 화면을 캡쳐하였다. 실행화면에 표시된 "Test it" 버튼을 클릭하면, 악성코드는 내부에 정의된 암호화된 문자열을 복호화하는 작업을 수행한다. 이 과정에서 윈도우즈 CAPI 함수들을 호출한다. 악성코드가 복호화 작업을 수행하는 동한 API Hooking 도구를 이용하여 악성코드가 사용하는 암호알고리즘, 암호키 등을 확인할 수 있다.

### 4.2 API Hooking

여기서는 API Hooking 도구를 직접 구현하여 앞서 제작한 악성코드가 동작하는 동안 CAPI를 호출하는 부분을 감시하는 과정을 보여주겠다. [그림 12]은 Hooking 도구의 소스코드 일부이다. CAPI 악성코드가 사용했던 암호알고리즘, 암호키, 암호문 등을 호

```
BOOL OnCreateProcessDebugEvent(LPDEBUG_EVENT pde)
{
    // Hooking 대상 API 주소 구하기
    HMODULE hAdvAPI32 = LoadLibrary("advapi32.dll");
    g_pfCryptDecrypt = GetProcAddress(hAdvAPI32, "CryptDecrypt");                  // 암호문 확보
    g_pfCryptImportKey = GetProcAddress(hAdvAPI32, "CryptImportKey");              // 키 확보
    g_pfCryptAcquireContext = GetProcAddress(hAdvAPI32, "CryptAcquireContextA");   // 암호 알고리즘

    // 후킹할 대상 함수의
    //   첫 번째 byte 를 0xCC (INT 3) 으로 변경
    //   (orginal byte 는 백업)
    g_cpdi = pde->u.CreateProcessInfo;      // CREATE_PROCESS_DEBUG_INFO

    // CryptDecrypt
    ReadProcessMemory(g_cpdi.hProcess, g_pfCryptDecrypt,
                      &g_chDecryptOrgByte, sizeof(BYTE), NULL);
    WriteProcessMemory(g_cpdi.hProcess, g_pfCryptDecrypt,
                       &g_chDecryptINT3, sizeof(BYTE), NULL);

    // CryptImportKey
    ReadProcessMemory(g_cpdi.hProcess, g_pfCryptImportKey,
        &g_chImportKeyOrgByte, sizeof(BYTE), NULL);
    WriteProcessMemory(g_cpdi.hProcess, g_pfCryptImportKey,
        &g_chImportKeyINT3, sizeof(BYTE), NULL);

    // CryptAcquireContext
    ReadProcessMemory(g_cpdi.hProcess, g_pfCryptAcquireContext,
        &g_chAcquireContextOrgByte, sizeof(BYTE), NULL);
    WriteProcessMemory(g_cpdi.hProcess, g_pfCryptAcquireContext,
        &g_chAcquireContextINT3, sizeof(BYTE), NULL);


    return TRUE;
}
```

[그림 12] CAPI 악성코드 Hooking 도구(소스코드)




[그림 13] CAPI 악성코드 Hooking 도구(실행결과)

출하는데 필요한 함수들을 마찬가지로 적용하여 함수 호출 시점에 Hooking을 하게 된다.

실제로 Hooking 도구를 실행한 상태에서 앞서 제작한 CAPI 악성코드를 실행한 후, "Test it !!" 버튼을 클릭하면 Hooking 도구에서는 〔그림 13〕과 같이 악성코드가 사용했던 암호알고리즘, 암호키, 암호화된 문자열의 복호화 결과값을 얻어 낼 수 있다.

## V. 결론 및 향후방향

지금까지 API Hooking 기법을 통해서 CAPI를 사용하는 악성코드의 동작을 감시하는 방안과 이와 관련된 도구를 직접 구현하여 테스트한 결과를 제시하였다.

본 논문에서는 비록 암호키나 암호문이 악성코드에 포함된 경우에 대해서 테스트를 했지만, 그 반대의 경우에도 윈도우즈 CAPI를 사용한다면 Key값과 암호문은 변수로 입력이 되기 때문에 API Hooking을 통해서 악성코드의 행위에 대해서 충분히 확인이 가능하다.

향후 이 도구를 응용하면 RSA 뿐만 아니라, DES 등 여타 암호 알고리즘에 대해서도 구현이 가능할 것이다. 또한 윈도우즈에서 실행중인 모든 프로세스에 대해서 Crypto API 사용여부 감시 등 전역적으로 확대하는 도구 제작도 가능하다. 다만, 모든 프로세스를 감시하는 방안을 적용할 경우 제 3절에서 소개한 악성코드 판별 여부에 대한 연구가 선행되어야 할 것이다.

또한 악성코드에 포함된 암호문 뿐만 아니라, XLS, DOC 등 문서파일 대상으로 수행되는 암호화에 대해서도 사전에 인지가 가능한 프로세스를 추가함으로써 개인자료 훼손 및 유출 등의 피해를 사전에 예방이 가능할 것으로 기대된다.

## 참고문헌


〔1〕 http://www.ahnlab.com/kr/site/securitycenter/asec/asecReportList.do

〔2〕 A. Young, "Cryptoviral extortion using Microsoft's Crypto API", International Journal of Information Security, pp. 2-3, 2006년.

〔3〕 염용진, 배병철, "악성 프로그램의 진화", 정보통신산업진흥원 주간기술동향, 1244호, pp. 32-33, 2006년 5월.

〔4〕 http://articles.yuikee.com.hk/newsletter/2005/05/f.html

〔5〕 http://www.symantec.com/connect/blogs/how-trojanzbotbinf-uses-crypto-api, 2010.

〔6〕 http://msdn.microsoft.com/en-us/library/aa380255(v=vs.85).aspx

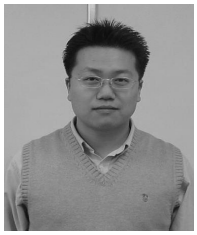
〔7〕 http://www.reversecore.com/65

〔8〕 A. Young, M. Yung, "An implementation of cryptoviral extortion using microsoft's crypto api", http://www.cryptovirology.com/cryptovfiles/newbook/Chapter2.pdf, pp. 9-16, 2005

〔9〕 Greg Hoglund, James Butler, 윤근용, 루트킷 윈도우 커널 조작의 미학, 제4장 전통적인 후킹기술, 에이콘, 2007.

## 〈著者紹介〉

송 정 환 (Junghwan Song)
1984년 2월: 한양대학교 수학과 졸업
1989년 5월: Syracuse University 수학과 석사
1993년 5월: Rensselaer Polytechnic Insitute 수학과 박사
1999년 3월~현재: 한양대학교 수학과 교수
〈관심분야〉 암호학, 정보보호, 수리계획법



황 인 태 (Intae Hwang)
2002년 7월: 한양대학교 수학과 졸업
2009년 2월: 한양대학교 수학과 석사
〈관심분야〉 정보보호, 암호