랜섬웨어 암호기능 분석 보고서

- Magniber -

2019. 07

차세대암호인증팀

KISA 한국인터넷진흥원
KOREA INTERNET & SECURITY AGENCY

Contents

1. 개요

한국어를 사용하는 윈도우 운영체제 만을 공격 대상으로 한 Magniber는 2017년 10월 15일경부터 Magnitude 익스플로잇 킷을 통해 유포되었다. 파일 암호화 후 '.kgpvwnr' 확장명이 추가되고 주요 정보가 암호화되었으니 코인을 지불하라는 텍스트 파일이 생성된다.

◆ 랜섬웨어 명 : Magniber
◆ 랜섬웨어 공격 대상 : Windows 운영체제
◆ 감염 확장자 : .kgpvwnr
◆ 해시값 :
md5 : e76eca2f7d0450c84417a8ac242b424c
sha1 : abdb8a43a6d0bf9c60d9cd4223da787c33b341bb
sha256 : 2f40011df85d75556816ac944d805b6313da44c73c80778af62be5727c005811

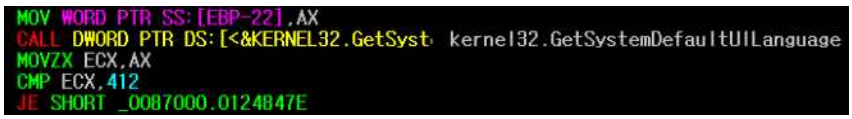
2. 랜섬웨어 분석

2.1 실행 과정

매그니베르 랜섬웨어의 동작 과정은 [그림 1]과 같다.

[그림 1] Magniber 랜섬웨어 암호화 과정

먼저 [그림 2]와 같이 동작하는 PC의 운영체제를 파악한다. GetSystemDefaultUILanguage를 통해 운영체제에서 한국어를 사용하는지 확인한다. 한국어를 사용할 경우에만 암호화를 진행한다.



[그림 2] GetSystemDefaultUILanguage 함수 호출 부분

그리고, 암호화 제외 경로를 확인한다. Magniber는 [표 1]과 같이 부팅과 관련 있거나, 시스템 운영과 관련 있는 폴더들은 암호화하지 않는다.

\documents and settings\all users, \default user, \localservice, \networkservice, \appdata, \local settings, \public, \ tor browser, \recycle.bin, \windows, \boot, \intel, \perflogs, \program files, \programdata, \recovery, \recycled, \recycler, \system volume information,

[표 1] 암호화 제외 폴더

[표 2]와 같이 1,215개의 암호화 대상 확장자를 포함하고 있으며, 랜섬웨어가 가지고 있는 리스트에 포함된 확장자를 가진 파일들을 모두 암호화한다.

.doc, .docx, .xls, .xlsx, .ppt, .pptx, .pst, .ost, .msg, .em, .vsd, .vsdx, .csv, .rtf, .123, .wks, .wk1, .pdf, .dwg, .onetoc2, .snt, .docb (생략)

[표 2] 암호화 대상 확장자

그 뒤 윈도우에서 제공하는 CryptoAPI 함수를 사용하여, AES 알고리즘으로 파일들을 암호화한다. 파일을 암호화시키고 파일의 확장자를 .kgpvwnr로 변경한다. 암호화가 끝난 후 메모리 내에 키 관련 정보들을 모두 제로화하고 랜섬 노트를 띄운다.

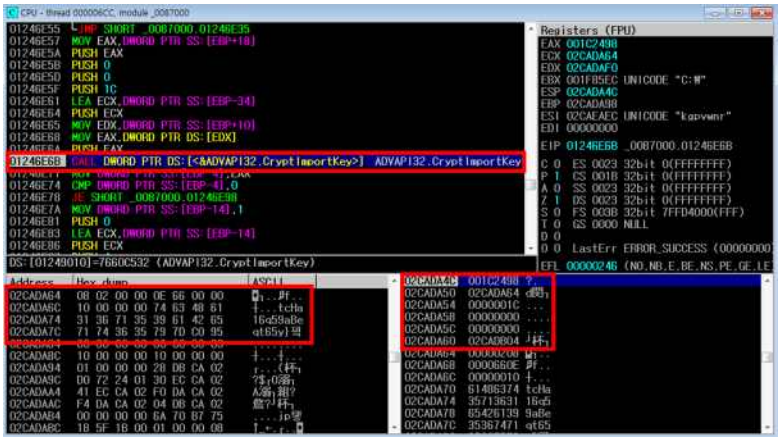
2.2 암호화 키 생성 과정

Magniber 랜섬웨어는 윈도우가 제공하는 CryptoAPI를 사용하여 암호화를 진행한다. 키 생성 과정 없이 바로 CryptImportKey를 사용한다. 이는 어딘가에 저장되어 있는 키를 가져온다는 의미인데 분석 결과 PLAINTEXTKEY BLOB이 랜섬웨어 내부에 하드코딩되어 있음을 확인했다. CryptImportKey 함수 호출을 위해 필요한 매개변수는 [그림 3]과 같다.

```
BOOL WINAPI CryptImportKey(
  _in_  HCRYPTPROV hProv,          // CSP 핸들의 포인터
  _in_  BYTE       *pbData,        // Key BLOB 포인터
  _in_  DWORD      dwDataLen,      // Key BLOB 길이
  _in_  HCRYPTKEY  hPubKey,        // 키 핸들
  _in_  DWORD      dwFlags,        // 플래그 값
  _Out_ HCRYPTKEY  *phKey          // HCRYPTKEY 포인터
);
```

[그림 3] CryptImportKey 함수 매개변수

두 번째 매개변수인 pbData는 키 블랍 데이터의 주소를 가지고 있다. pbData에 있는 해당 주소로 이동하면 키 블랍 정보를 얻을 수 있다. [그림 4]의 왼쪽 하단에서 키 블랍 데이터를 확인하였다.



[그림 4] CryptImportKey 함수

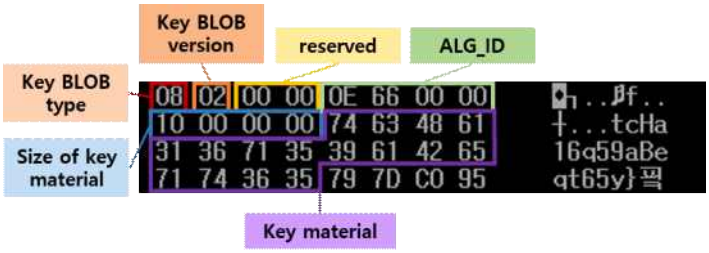
KEY BLOB의 구조는 [표 3]와 같다. KEY BLOB 구조체 중 BLOB HEADER의 구조는 모든 종류의 키 블랍이 동일하며, bType 값을 통해 키 블랍의 종류를 확인할 수 있다.

구조		크기	의미
BLOB HEADER	bType	BYTE	키 블랍 종류
	bVersion	BYTE	키 블랍 버전
	reserved	WORD	0
	aiKeyAlg	DWORD	알고리즘 ID
PLAINTEXTKEY BLOB	dwKeySize	DWORD	키 길이
	rgbKeyData	가변	키

[표 3] KEY BLOB 구조

[그림 5]는 키 블랍 데이터를 캡처한 화면이다. 0x00 위치의 bType을 보면 키 블랍 타입이 0x08이다. 0x08 값을 통해 해당 키 블랍이 PLAINTEXTKEY BLOB 임을 알 수 있다. 그리고, aiKeyAlg 위치인 0x04

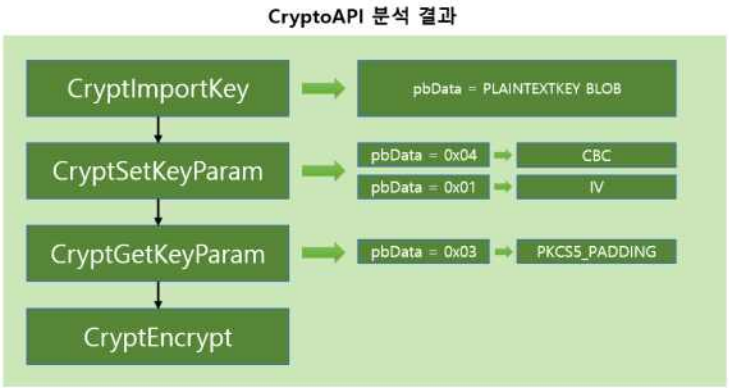
위치에 0x660E 값이 적혀있다. 해당 위치의 값을 통해 알고리즘 종류를 파악할 수 있다. 0x660E은 AES-128을 나타낸다. 마찬가지로 dwKeySize(0x20)는 0x10 값이 있고, 그 뒤, 16바이트 값이 키 정보라는 것을 알 수 있다. PLAINTEXTKEY BLOB은 키 정보를 평문으로 가지고 있다. 따라서 dwKeySize에 이어 나오는 16바이트의 값이 키 정보이다.



[그림 5] PLAINTEXTKEY BLOB 데이터

2.3 암호화 과정

[그림 6]은 Magniber 랜섬웨어가 호출하는 CryptoAPI의 호출 순서이다. CryptImportKey → CryptSetKeyParam → CryptGetKeyParam → CryptEncrypt 순으로 호출된다.



[그림 6] 암호화 과정

암호화하는데 사용할 운영 모드와 IV는 CryptSetKeyParam 함수의 파라미터를 통해 알 수 있다. CryptSetKeyParam 함수에 사용되는 매개변수는 [그림 7]과 같다. dwParam 값이 0x04인 경우는 운영 모드를 설정하는데 사용하겠다고 알려주는 것이고, 0x01인 경우에는 IV 값을 설정하는데 사용하겠다고 알려주는 것이다.

```
BOOL WINAPI CryptSetKeyParam(
  _In_      HCRYPTKEY hKey,          // 키 핸들
  _In_      DWORD     dwParam,       // 타입 값
  _In_ const BYTE     *pbData,       // 버퍼 포인터
  _In_      DWORD     dwFlags        // 플래그 값
);
```

[그림 7] CryptSetKeyParam 매개변수

[표 4]는 CryptSetKeyParam 함수 실행 부분이며, dwParam 값이 0x04 이기 때문에 KP_MODE이다. 다음 매개변수인 pbData의 값은 1이며 이것은 CBC(Cipher Block Chaining) 모드를 의미한다.

타입	크기	값	의미
HCRYPTKEY	hKey	0x1E2AE0	-
DWORD	dwParam	0x04	KP_MODE
const BYTE	*pbData	0X01	CBC
DWORD	dwFlags	0x00	-

[표 4] 운용모드 확인

IV에 대한 정보는 [표 5]에서 확인 가능하다. CryptSetKeyParam 함수의 dwParam이 1인 경우에 알 수 있으며, *pbData가 가리키는 것이 IV 값이다. 확인한 IV 값은 0x(4E, 33, 69, 33, 4E, 65, 39, 4F, 31, 30, 56, 43, 4C, 55)이다.

타입	크기	값	의미
HCRYPTKEY	hKey	0x1E2AE0	-
DWORD	dwParam	0x01	KP_IV
const BYTE	*pbData	0x{4E, 33 69, 69, 33 4E, 65, 39, 4f, 31, 30, 56, 43, 35, 4C, 55}	N3ii3Ne9010VC5LU
DWORD	dwFlags	0x00	-

[표 5] IV 확인

CryptGetKeyParam 함수의 매개변수는 [그림 8]과 같다.

```
BOOL WINAPI CryptGetKeyParam(
  _In_    HCRYPTKEY hKey,          // 키 핸들
  _In_    DWORD     dwParam,       // 타입 값
  _Out_   BYTE      *pbData,       // 버퍼 포인터
  _Inout_ DWORD     *pdwDataLen,   // 버퍼 길이
  _In_    DWORD     dwFlags        // 플래그 값
);
```

[그림 8] CryptGetKeyParam 매개변수

dwParam 값이 0x3인 경우 KD_PADDING을 의미하며, [표 6]에서 확인 가능하다. *pdData가 가리키는 값을 보면 패딩 종류를 확인할 수 있는데, 해당 값이 1이므로 PKCS5_PADDING을 사용한다는 것을 알 수 있다.

타입	크기	값	의미
HCRYPTKEY	hKey	0x01E2AE0	-
DWORD	dwParam	0X03	KD_PADDING
BYTE	*pbData	0x01	PKCS5_PADDING
DWORD	*pdwDataLen	-	-
DWORD	dwFlags	0x00	-

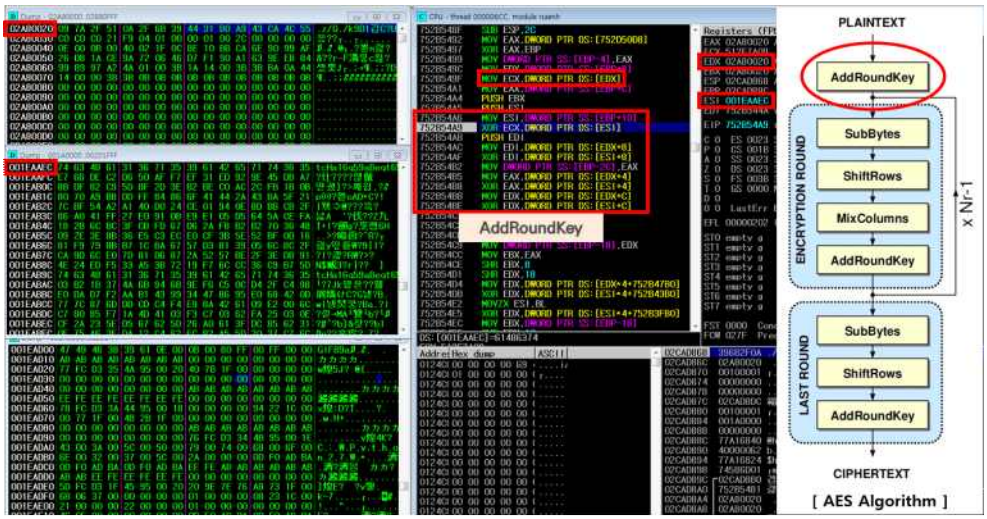
[표 6] 패딩기법 확인

[그림 9]는 Magniber에 감염된 파일의 바이너리이다. Magniber에 감염된 파일의 제일 처음 16바이트는 IV 값이며, 그 뒤로는 암호화된 원본 데이터가 기록되어 있는 것을 확인하였다.

[그림 9] Magniber 감염 파일

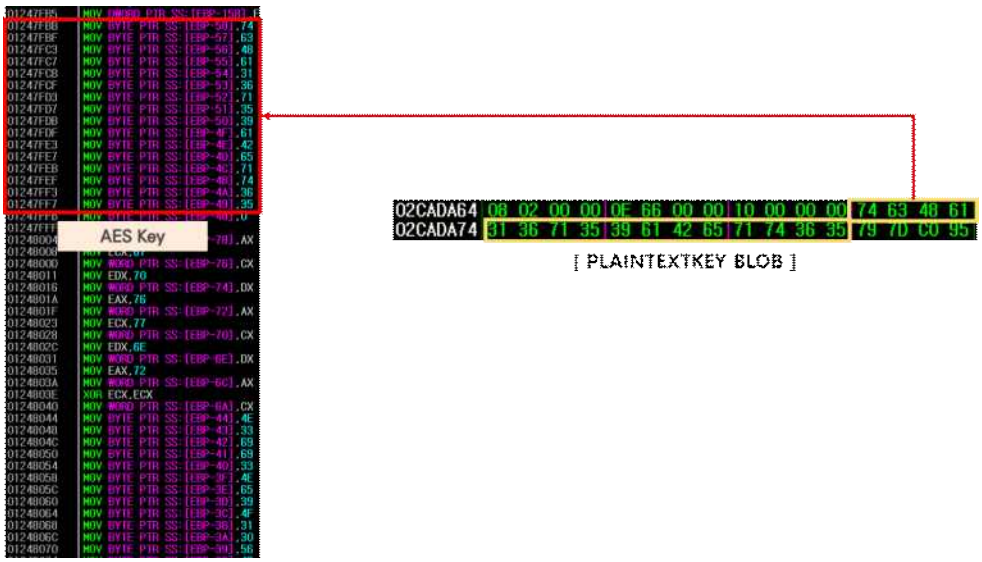
3. 복원 기술

PLAINTEXTKEY BLOB에 저장된 키 정보가 실제로 쓰이는지 확인하기 위해 CryptEncrypt 함수를 디버깅하였다[그림 10]. 실제 AES 암호화가 진행되는 부분을 살펴본 결과, 첫 번째 AddRoundKey에서 키 블랍의 키 데이터 값이 AES 키로 그대로 사용된다.



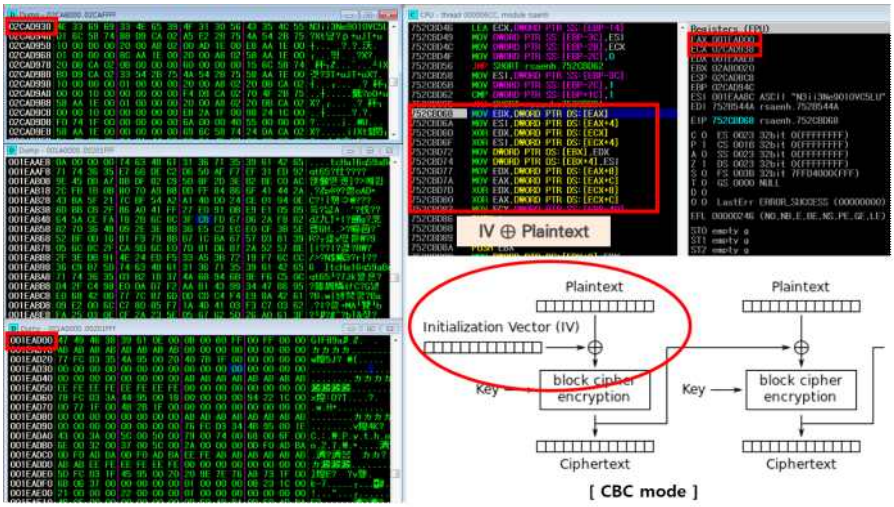
[그림 10] 마스터키값 확인

랜섬웨어 프로세스 메모리에서 확인한 결과 [그림 11]과 같이 PLAINTEXTKEY BLOB 부분의 키 데이터 값이 Magniber 실행 초반에 메모리에 로드된다.



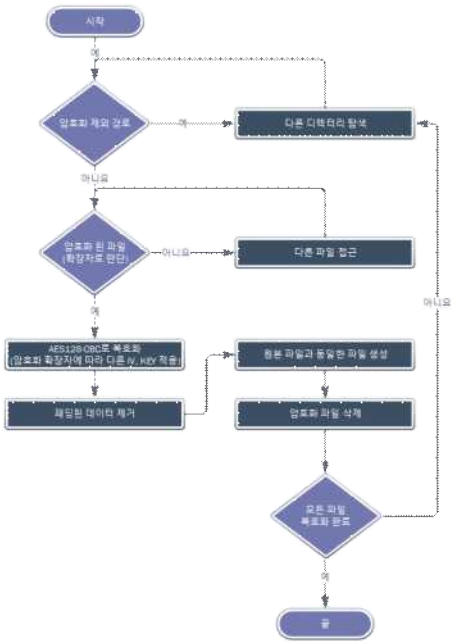
[그림 11] 암호화 시 고정키 사용 확인

키 정보와 마찬가지로 CryptEncrypt 함수 디버깅을 통해 [그림 12]의 화면에서 CryptSetKeyParam 함수 사용 시에 설정하였던 IV 값과 CBC 모드를 사용된다.



[그림 12] CBC 모드 사용 확인

분석 결과를 토대로 [그림 13]과 같은 복호화 과정을 구성하였다.

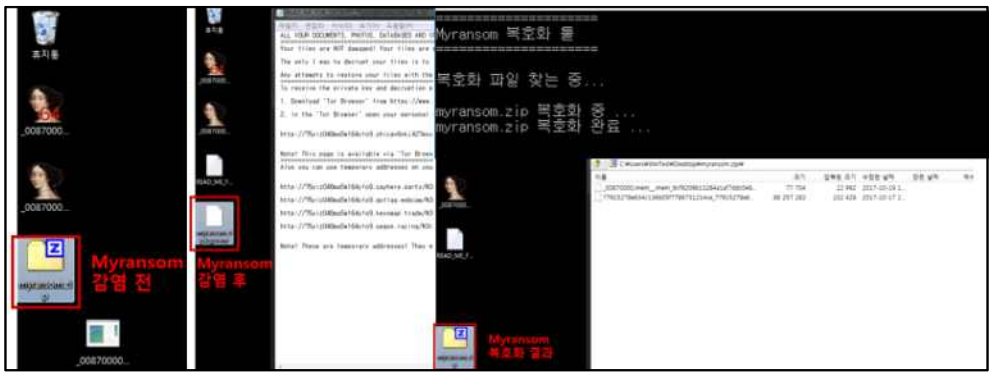


[그림 13] Magniber 복호화 과정

먼저, 암호화 제외 경로를 걸러낸다. 그리고, 파일 시스템을 탐색하여 .kpgvwnr로 확장자가 바뀐 암호화된 파일들을 판별한다. 분석을 통해 찾아

낸 IV와 키값을 적용해 AES-128-CBC로 복호화를 진행한다. 복호화 된 파일의 끝부분에서 패딩 된 데이터를 제거한다. 복호화가 완료된 암호화 파일은 제거하며, 모든 파일이 복호화 될 때까지 동일한 과정을 반복한다.

감염 후 암호화된 파일과 복호화 도구를 이용해 파일을 복호화 한 결과는 [그림 14]와 같다.



[그림 14] Magniber 복호화 도구 실행 후 결과 화면

위 결과를 활용한 복구 코드는 아래 표와 같다.

```
def Decrypt_Magniber(path_filename, filesize):
        readfile = open(path_filename,"rb")
        writefile = open(path_filename.split(".kgpvwnr")[0],"wb")
        #iv = 0x4e336969334e65394f31305643354c55
        iv = 'N3ii3Ne9O10VC5LU'
        #key = 0x7463486131367135396142657174363 5
        key = 'tcHa16q59aBeqt65'

        readfile.seek(16)
        cipher = readfile.read(filesize - 16)
        encryptor = AES.new(key, AES.MODE_CBC, iv)
        padding_plain = encryptor.decrypt(cipher) #복호화 된 평문(패딩 포함)
        size = filesize-16-ord(padding_plain[-1]) #패딩 전 원본 파일 크기
        plain = padding_plain[0:size] #복호화 된 평문(패딩 제외)
        writefile.write(plain)
        readfile.close()
        writefile.close()
```

[표 7] Magniber 랜섬웨어 복구 코드

4. 결론

Magniber 랜섬웨어의 동작 과정, 암호화 키 생성 방식, 파일 암호화 과정 등 암호학적 요소를 상세히 분석하고, 분석 결과를 바탕으로 Magniber 랜섬웨어의 복구 가능성을 분석하였다.

Magniber 랜섬웨어 내부에 하드코딩된 암호화 키를 통해 복구에 성공하였다.

상세 요소		분석 내용
비대칭키	알고리즘명	-
해시함수		-
대칭키	알고리즘명	AES-128
	암호화 키	0x{74,63,48,61,31,36,71,35,39,61,42,65,71,74,36,35}
	IV	0x{4E, 33, 69, 33, 4E, 65, 39, 4F, 31, 30, 56, 43, 4C, 55}
	패딩여부	PKCS5-PADDING
	운영모드	CBC
암호화 API		CryptAPI
공개키 생성 함수		-
대칭키 생성 함수		-
난수 생성기		-
엔트로피 수집		-
키 관리 방법	모든 파일 상이한 암호화 키	X
	모든 파일 동일한 암호화 키	O
	네트워크 연결에 따른 키 변화	X
할당 해제 수행 여부		X
제로화 수행 여부		X
운영체제 백업파일 삭제		O

랜섬웨어 암호기능 분석 보고서 작성 공헌자

구 분	소 속	직 위	성 명
책임자	KISA	팀장	박해룡
작성자	KISA	선임	김기문
		주임	김대운
		주임	이영주

랜섬웨어 암호기능 분석 보고서 - Magniber

2019년 7월 발행

발행처 KISA 한국인터넷진흥원
KOREA INTERNET & SECURITY AGENCY