Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.2, Apr. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.2.339

Erebus 랜섬웨어에 대한 암호학적 분석 연구*

김 소 람,[1†] 김 지 훈,[1] 박 명 서,[1] 김 대 운,[3] 김 종 성[1,2‡]
[1]국민대학교 금융정보보안학과, [2]국민대학교 정보보안암호수학과, [3]한국인터넷진흥원

Study on Cryptographic Analysis of Erebus Ransomware*

Soram Kim,[1†] Jihun Kim,[1] Myungseo Park,[1] Daeun Kim,[3] Jongsung Kim[1,2‡]
[1]Dept. of Financial Information Security, Kookmin University,
[2]Dept. of Information Security, Cryptology, and Mathematics, Kookmin University
[3]Korea Internet & Security Agency

요 약

랜섬웨어는 데이터를 암호화하여 금전을 요구하는 악성프로그램이다. 전 세계적으로 랜섬웨어에 대한 피해가 증가하고 있으며 기업, 공공기관, 병원을 대상으로 하는 타깃형 공격이 늘어나고 있다. 또한 랜섬웨어가 서비스화 되어 유포되기 때문에 하나의 랜섬웨어가 발견되면 이후에 해당 랜섬웨어와 유사한 변종이 많이 발견된다. 이에 따라 랜섬웨어에 대한 정확한 분석이 해당 랜섬웨어뿐만 아니라 변종 랜섬웨어 복호화 방안 모색에 기반이 될 수 있다. 본 논문에서는 2017년 6월에 발견된 Erebus 랜섬웨어에 대해 암호학적 요소와 암호화 과정을 분석하였으며, 해당 결과를 기반으로 암호학적 취약점 및 메모리 분석 연구를 진행하였다.

ABSTRACT

Ransomware is a malicious program that requires money by encrypting data. The damage to ransomware is increasing worldwide, and targeted attacks for corporations, public institutions and hospitals are increasing. As a ransomware is serviced and distributed, its various usually emerge. Therefore, the accurate analysis of ransomware can be a decryption solution not only for that ransomware but also for its variants. In this paper, we analyze a cryptographic elements and encryption process for Erebus found in June, 2017, and investigate its cryptographic vulnerability and memory analysis.

Keywords: Ransomware, Erebus, Vulnerability

I. 서 론

랜섬웨어는 ransom(몸 값)과 software(소프트웨어)의 합성어로 시스템을 잠그거나 데이터를 암호화해 사용할 수 없게 한 뒤, 이를 인질로 삼아 금전을 요구하는 악성 프로그램이다. 락커 랜섬웨어(Locker Ransomware)가 시스템을 잠그는 방식이며, 크립토 랜섬웨어(Crypto Ransomware)는 데이터를 암호화하는 형태이다. 크게 두 가지 종류가 존재하며 현재는 크립토 랜섬웨어가 대부분의 비율을 차지하고 있다.

2015년 4월 국내 유명 커뮤니티 사이트가 CryptoLocker에 감염된 것을 시작으로 다양한 랜섬웨어가 국내에 유포되기 시작하였다. 한국랜섬웨어침해대응센터에 따르면 국내 기준으로 2015년에는

Received(11. 14. 2017), Modified(1st:02. 20. 2018, 2nd: 03. 08. 2018), Accepted(03. 08. 2018)
* 이 논문은 2017년도 방송통신발전기금(암호기술)의 재원으로 한국인터넷진흥원의 지원을 받아(부분적으로) 수행된 연구 사업임(KISA과제-2017-04)
* 이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아(부분적으로) 수행된 연구임(No. NRF-2016R1D 1A1A09919726)
† 주저자, kimsr2040@kookmin.ac.kr
‡ 교신저자, jskim@kookmin.ac.kr(Corresponding author)

약 1,090억 원 정도의 피해를 입었으며, 2016년에는 약 3,000억 원, 2017년에는 3,000억 원 이상의 피해를 입을 것이라고 예측하였다[1]. 이와 같이 랜섬웨어는 해마다 피해 량이 증가하고 있으며, 개인뿐만 아니라 기업, 병원, 공공기관으로 표적 대상을 넓혀가고 있다.

2015년 국립암센터를 비롯해 국립대 병원 임상시험센터, 서울 소재 사립대 병원, 지방 중소병원 등이 랜섬웨어에 감염되었으며, 2016년 6월에는 부산 관공서, 2017년에는 전 세계적으로 가장 큰 피해를 입힌 WannaCry가 등장하여 영화관 광고상영시스템, 종합병원 전산시스템 일부, 제조업체의 제조공정 서버 등이 피해를 입었다.

병원, 기업과 같은 기관들은 랜섬웨어에 감염되면 빠른 시간 내에 시스템을 복구해야 하므로 복호화 키를 받는 쪽을 택하게 되는 경우가 많다. 그러나 돈을 지불하고 복호화 키를 얻고자 하는 사용자들이 증가하게 되면 공격자 입장에서는 계속해서 공격을 시도하게 될 것이다. 따라서 이러한 악순환을 반복하지 않기 위해 근본적인 랜섬웨어 복호화 방안에 대한 연구가 필요하다.

현재 암호화된 데이터 복호화를 위해 네덜란드 경찰청의 National High Tech Crime Unit, 유로폴(Europol)의 European Cybercrime Centre, 그리고 보안업체인 카스퍼스키랩(Kaspersky Lab)과 맥아피(McAfee, 전 Intel Security)에서 NOMORERANSOM!이라는 프로젝트를 기획하였으며, 웹사이트를 운영하고 있다. 위의 기관들 이외에도 TRENDMICRO, Avast 등 여러 업체에서 복호화 툴을 해당 프로젝트에 제공하고 있다[2].

본 논문에서는 2017년 6월에 발견된 Erebus에 대해 암호학적 요소와 암호화 과정을 밝혀내며, 해당 결과를 기반으로 암호학적 취약점에 대해 분석하였다. 기존에는 주로 구조적인 취약점을 찾아내 대응 방법을 모색하였으나, 본 논문에서는 근본적인 암호키 생성을 중점적으로 복호화 방안에 대해 연구하였다. 또한, Erebus는 암호화 과정 등 암호학적 요소에 대해 상세한 분석이 이루어지지 않았기 때문에 리버스엔지니어링을 통해 암호화 과정을 밝혀냈으며, 해당 결과를 기반으로 암호학적 취약점과 메모리 분석을 진행하였다. Erebus 이전에 유포되었던 WannaCry의 복구 가능 요인과 비교하여 파일 복호화를 위한 요소를 알아보고 메모리 분석을 통해 해당 값 획득 및 RSA 개인키 재현 가능 여부에 대해 연구하였다.

본 논문의 구성은 다음과 같다. 2장에서 Erebus 랜섬웨어에 대해 소개하고, 3장에서는 Erebus가 사용하는 암호 알고리즘 등 암호학적 요소와 암호화 과정을 밝혀낸다. 4장에서 Erebus 랜섬웨어의 암호학적 취약점 분석 결과에 대해 설명하고 5장에서는 랜섬웨어 감염 시 대응 방법과 복호화를 위해 논의 되어야 할 부분들에 대해 언급한다. 마지막 6장에서 결론으로 마무리한다.

II. Erebus 개요

Erebus는 2016년 9월에 처음 등장하였으며, 발견될 당시 버전은 윈도우(Windows)를 대상으로 한다. RSA-2048 암호 알고리즘을 사용해 암호화를 진행하며 완료 시 파일들을 .encrypt 확장자로 변경한다. 암호화는 .ppt, .docx, .sql을 포함해 423개의 확장자를 대상으로 한다. 그 후, 2017년 2월 UAC(User Account Control)[1]라는 윈도우 보안 기능을 우회한 변종이 유포되었다[3].

2017년 6월에는 윈도우가 아닌 리눅스(Linux)로 목표를 변경하여 한국의 웹 호스팅 회사인 '인터넷나야나'를 감염시켰다. 이로 인해 리눅스 서버 153대가 감염되어 고객사 홈페이지 3,400여개가 마비되었다. Erebus에 감염된 화면은 Fig. 1과 같으며, 감염 시 _DECRYPT_FILE.html, _DECRYPT_FILE.txt, index.html 파일을 생성한다.

Erebus 분석은 IDA(Interactive DisAssembler)와 GDB(GNU Debugger)[2]를 사용하였다. IDA는 정적 분석을 위해 윈도우 7에서 이용하였으며, GDB

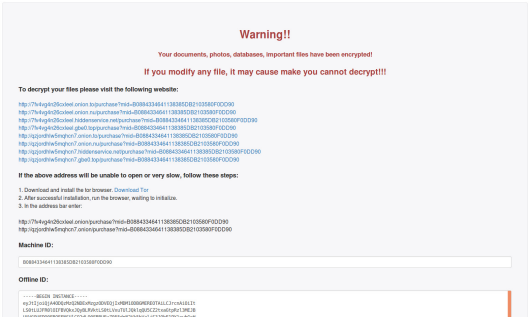

Fig. 1. Erebus ransom note

1) 사용자 계정을 통제하는 윈도우 보안 기술.
2) 유닉스 기반의 시스템을 위한 기본 디버거.

를 이용한 동적 분석은 우분투(Ubuntu) 15.10이 설치되어 있는 가상머신에서 진행하였다. 먼저 정적 분석을 통해 랜섬웨어에서 사용한 라이브러리 및 구조 파악을 한 후, 동적 분석 도구를 이용하여 랜섬웨어에서 사용되는 키 생성 및 암호화 함수의 실제 입력 값을 확인하며 암호화 과정을 분석하였다.

III. Erebus 암호화 구조

본 논문에서는 2017년 6월에 유포된 랜섬웨어 Erebus를 대상으로 분석한다. 사용하는 알고리즘 및 키 길이 등 암호학적 요소는 Table 1과 같다.

Table 1. Cryptographic specifications of Erebus

Items		Details
Crypto Algorithm		RSA-2048 AES-256 ARC4-256
Public Key	e	0x10001
	p(prime)	1024-bit
	q(prime)	1024-bit
Private Key	d	2048-bit
Key Encryption		RSA-2048 AES-256
File Encryption		ARC4-256

3.1 암호화 과정

Erebus 암호화 과정은 Fig. 2, Fig. 3과 같다.

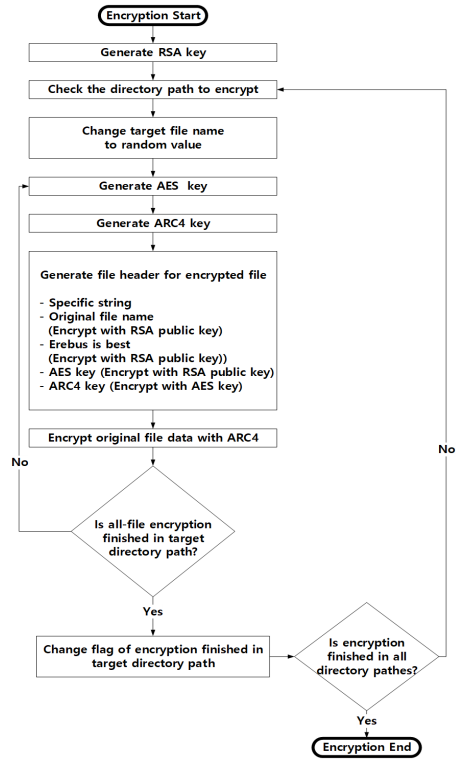

Fig. 3. Encryption process of Erebus

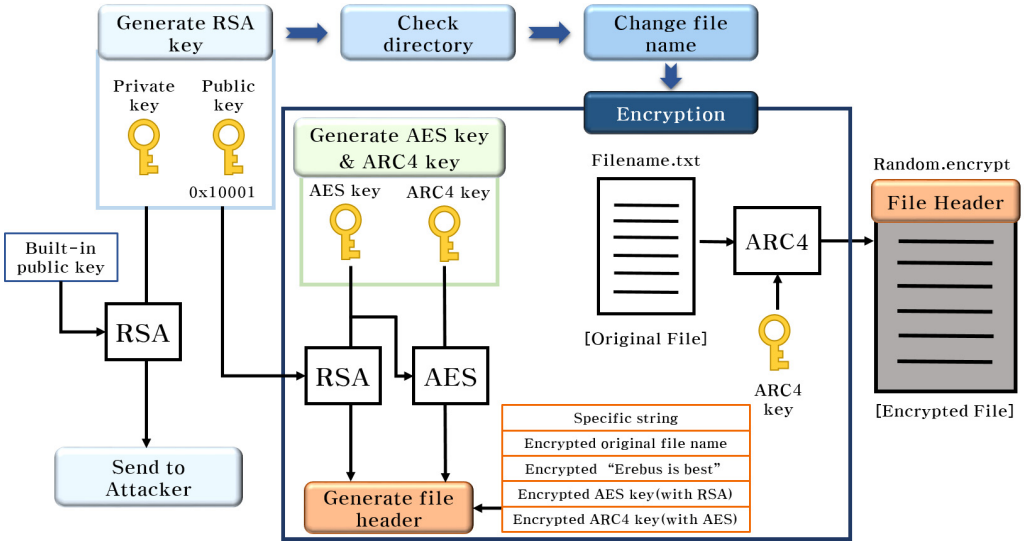

Fig. 2. Encryption structure of Erebus

3.1.1 RSA 공개키/개인키 쌍 생성

RSA의 공개키는 0x10001을 사용하며, 이에 대응하는 개인키를 mbedtls_rsa_gen_key 함수를 이용하여 생성한다. 생성한 개인키는 랜섬웨어 실행파일 내부에 내장되어 있던 공격자의 공개키로 암호화하여 C&C(Command and Control) 서버[3]로 전송한다.

사용자 PC에서 다시 한 번 RSA 키 쌍을 생성하는 이유는 각 PC마다 다른 키를 사용해서 관리하기 위함이다. 만약 모든 PC가 내장되어 있던 공격자의 공개키로 대칭키를 암호화한다면 모두 같은 개인키로 복호화를 할 수 있기 때문이다.

3.1.2 암호화할 디렉터리를 확인

시스템 파일등을 암호화 대상에서 제외하기 위하여 암호화 대상 디렉터리를 지정해 둔다. Erebus가 검색하는 시스템 테이블은 Table 2와 같으며, 웹사이트 파일을 저장하는 곳인 /var/www/와 MySQL에서 사용하는 데이터베이스 파일인 ibdata를 포함하며, 시스템 영역 디렉터리들은 제외한다.

감염 PC에 해당하는 상세한 디렉터리 목록은 /var/tmp//.57FCBAE7175B4A8A1F14600DF124E073.res 파일 내에 저장되어 있으며 상세 내용은 Fig. 4와 같다.

Table 2. System table that Erebus searches

Included directories	/var/www/
Included files	ibdata0, ibdata1, ibdata2, ibdata3, ibdata4, ibdata5, ibdata6, ibdata7, ibdata8, ibdata9, ib_logfile0, ib_logfile1, ib_logfile2, ib_logfile3, ib_logfile4, ib_logfile5
Excluded directories	$/bin/, $/boot/, $/dev/, $/etc/, $/lib/, $/lib64/, $/proc/, $/run/, $/sbin/, $/srv/, $/sys/, $/tmp/, $/usr/, $/var/, /.gem/, /.bundle/, /.nvm/, /npm/
Excluded files	-

3) 원격지에서 명령을 내리거나 악성코드를 제어하는 서버.

```
0/var/www/
0/mnt/
0/opt/
0/root/
0/lost+found/
0/home/ubuntu/Desktop/
0/home/ubuntu/Documents/
0/home/ubuntu/Downloads/
0/home/ubuntu/.config/upstart/
0/home/ubuntu/.config/gnome-session/saved-session/
0/home/ubuntu/.config/gnome-session/
0/home/ubuntu/.config/libaccounts-glib/
0/home/ubuntu/.config/nautilus/
0/home/ubuntu/.config/libreoffice/4/user/basic/Standard/
0/home/ubuntu/.config/libreoffice/4/user/basic/
0/home/ubuntu/.config/libreoffice/4/user/psprint/
```

Fig. 4. Part of target directory path for encryption

3.1.3 파일 이름을 랜덤 값으로 변경

암호화 대상 파일 확장자임을 확인 후에 파일명을 "16-byte 랜덤 값.encrypt" 형식으로 변경한다.

3.1.4 AES 키와 ARC4 키 생성

AES(Advanced Encryption Standard)와 ARC4(Alleged RC4(Rivest Cipher 4)) 키는 고정 값을 사용하는 것이 아니라 ctr_drbg 난수발생기를 사용해 생성된 난수를 그대로 사용한다. 난수발생기는 mbedtls_ctr_drbg_seed와 mbedtls_ctr_drbg_random 함수를 이용한다. 키 길이는 AES와 ARC4 모두 256-bit이며, 각 파일마다 다른 키를 사용한다.

3.1.5 암호화 파일 헤더 생성

랜섬웨어 감염 시 파일 암호화가 완료되면, 해당 파일은 암호화된 원본 데이터 앞쪽에 0x838만큼의 헤더를 덧붙인다. 헤더 맨 앞부분에는 0x438만큼의 고정 문자열이 기록되어 있으며, 경고 문구, 파일이

Table 3. Encrypted file header format

Offset	Size	Data
0x0	0x438	Specific string
0x438	0x100	Original file name encrypted with RSA public key
0x538	0x100	"Erebus is best" encrypted with RSA public key
0x638	0x100	AES key encrypted with RSA public key
0x738	0x100	ARC4 key encrypted with AES key

암호화되었다는 메시지, 비트 코인 지불 주소 등을 포함하고 있다. 이후에는 "Erebus is best"라는 문자열, 파일 이름과 AES 키를 각각 RSA 공개키로 암호화해서 0x100-byte씩 저장한다. 마지막 0x100-byte만큼은 AES 키로 암호화한 ARC4 키가 저장된다(Table 3).

3.1.6 원본 파일 암호화

파일 암호화는 ARC4 스트림 암호를 이용해 진행하며 0x80000-byte 크기씩 암호화한다(Fig. 5). 이보다 작은 크기의 파일은 파일 크기만큼 한 번에 암호화를 진행한다.

```
fseek(s, 0, 0);
fread(v20, 1u, *(_DWORD *)(a1 + 2144), s);
mbedtls_md5_update((int)&v23, v20, *(_DWORD *)(a1 + 2144));
fseek(s, *(_DWORD *)(a1 + 2144), 0);
while ( v11 )
{
  mbedtls_arc4_crypt((int)&v22, v11, (int)ptr, (int)v20);
  mbedtls_md5_update((int)&v24, ptr, v11);
  mbedtls_md5_update((int)&v23, v20, v11);
  memcpy(ptr, src, n);
```

Fig. 5. File encryption code

3.1.7 암호화 완료 시 대상 경로의 플래그 변환

암호화 대상인 디렉터리의 모든 파일이 암호화 완료되면 /var/tmp//.57FCBAE7175B4A8A1F14600DF124E073.res 파일에서 해당 디렉터리의 플래그 값을 0에서 1로 바꾼다(Fig. 6).

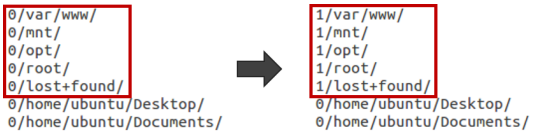

Fig. 6. Altered flags of encryption target path

IV. Erebus에 대한 암호학적 분석

Erebus는 RSA, AES, ARC4 세 가지 알고리즘을 사용한다. 파일 암호화에 사용되는 ARC4 키는 AES 키로 암호화되고, 해당 AES 키는 RSA 공개키로 암호화하여 관리한다.

$$Encryption_{ARC4_{256}}(File\,data),$$

$$Encryption_{AES_{256}}(ARC4_{256}\,key),$$

$$Encryption_{RSA_{pub}}(AES_{256}\,key).$$

따라서 특정 AES, ARC4 키를 찾는다 하더라도 모든 파일을 복구하는 것은 불가능하다. 모든 파일을 복구하기 위해서는 RSA 개인키를 알아내서 각 파일에 해당하는 AES, ARC4 키를 복구해 내는 것이 가장 효율적인 방법이다. 따라서 개인키 생성 시 필요한 정보들이 메모리에 남아 있는지, 개인키 생성에 사용된 요소들을 추적하여 개인키 재현이 가능한지 알아본다.

4.1 암호학적 요소 분석

Erebus 키 생성 함수들과 난수발생기는 Table 4와 같다.

Table 4. Cryptographic elements of Erebus

Items	Functions
Function of RSA public/private key pair generation	mbedtls_rsa_gen_key
Function of AES key & ARC4 key generation	mbedtls_ctr_drbg_random
Random bit generator	mbedtls_ctr_drbg_seed
Entropy of random bit	/dev/urandom

그 중 RSA 키 생성 함수는 mbedtls_rsa_gen_key이며 동작 과정은 다음과 같다(Fig. 7)[4].

(1) mbedtls_rsa_gen_key
  (1.1) mbedtls_mpi_gen_prime(소수 p 생성)
  (1.2) mbedtls_mpi_gen_prime(소수 q 생성)

mbedtls_rsa_gen_key 함수 내부에서는 mbedtls_mpi_gen_prime이라는 함수를 사용해서 소수를 생성하며, 각각 한 번씩 사용하여 p, q를 생성한다.

(2) mbedtls_mpi_gen_prime
  (2.1) mbedtls_mpi_fill_random
  (2.2) mbedtls_mpi_is_prime
  (2.3) mpi_miller_rabin

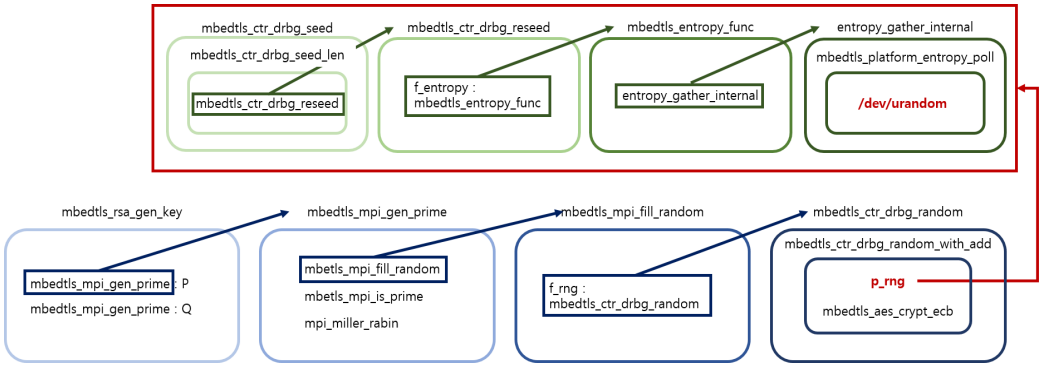

Fig. 7. RSA public key/private key pair generation process

먼저 2를 제외한 모든 소수는 홀수이기 때문에 난수 생성한 후 최하위 1-bit를 1로 고정시킨다. 이후에 Miller-Rabin 소수 판별법을 통해 소수인지 체크를 하게 되며, 소수인 경우에는 해당 값을 그대로 사용하고 아닌 경우에는 2를 더해준 뒤 다시 소수임을 확인하는 과정을 거치게 된다.

(3) mbedtls_mpi_fill_random
  (3.1) mbedtls_ctr_drbg_random

임의의 값을 선택하는 것은 mbedtls_ctr_drbg_random이라는 난수발생기를 사용한다.

(4) mbedtls_ctr_drbg_random
  (4.1) mbedtls_ctr_drbg_random_with_add
  (4.2) PRNG(Pseudo Random Number Generator)
  (4.3) mbedtls_aes_crypt_ecb

PRNG는 mbedtls_ctr_drbg_seed 함수를 사용하며, /dev/urandom의 값을 seed로 사용한다. 결국 RSA 개인키를 재현한다는 것은 /dev/urandom을 동일하게 생성할 수 있어야 한다. 그러나 /dev/urandom은 리눅스 내부에서 기본적으로 제공되는 난수발생기이며, 입력장치, 디스크 등의 인터럽트 이벤트로부터 수집한 엔트로피로 난수를 생성한다. 따라서 /dev/urandom의 엔트로피를 재현하는 것은 암호학적으로 키에 대한 전수조사량과 유사한 복잡도를 가진다.

4.2 메모리 분석

RSA 개인키 생성 시 필요한 정보들은 아래와 같으며, 이 중 최소 한 가지 정보를 확보한다면 RSA 개인키를 복구할 수 있다.

- RSA 개인키 생성 시 사용된 소수
- 소수 생성 시 사용된 난수
- 난수 생성 시 사용된 seed 값

4.2.1 RSA 개인키 생성 시 사용된 소수

RSA 키 쌍을 생성할 때 사용하는 mbedtls_rsa_gen_key 함수는 키 생성 후 소수가 저장된 메모리를 해제한다. 메모리 해제는 mbedtls_mpi_free라는 함수를 사용하며, 함수 내부의 mbedtls_mpi_zeroize 함수를 사용해 메모리를 0으로 초기화 하는 것을 알 수 있다.

4.2.2 소수 생성 시 사용된 난수

Erebus에서는 생성한 난수를 이용하여 소수를 생성한다. 따라서 처음 생성된 난수만 예측할 수 있다면 RSA 개인키 재현이 가능하다. 그러나 해당 값 역시 초기화 되는 것을 확인하였다.

4.2.3 난수 생성 시 사용된 seed 값

난수를 생성하기 위해 사용된 seed 값은 RSA 키가 생성된 이후에 메모리를 해제하며 0으로 초기화한다.

이로써 RSA 개인키 복구를 위한 모든 요소는 메모리 해제와 동시에 0으로 초기화 된다는 것을 확인하였다.

V. 논 의

Erebus 이전에 발견되었던 랜섬웨어들이 복호화 가능한 경우는 다음과 같다[5,6,7,8,9].

- 키 생성 요소(사용자 고유 ID, 임의로 생성한 값 등)를 별도의 파일로 저장.
- 리소스 영역의 고정된 값을 seed 값으로 사용.
- 시스템 시간을 seed 값으로 사용.
- 공개키/개인키 사용 시 키 생성에 사용된 소수가 메모리에 남아있음.
- 랜섬웨어 제작자가 암호화 키를 공개.

그러나 Erebus는 위의 경우에 포함되지 않는다. 이미 공개된 취약점은 보완해서 배포하기 때문에 이후에 발견되는 랜섬웨어 대부분도 위의 경우에 해당되지 않을 가능성이 높다.

먼저 랜섬웨어 감염 시 대처 방안은 다음과 같다. 로그 분석을 통해 '알 수 없는 자동 실행 파일 등록', '알 수 없는 프로세스 생성', '알 수 없는 파일이 다른 서버로 전송', '외부 서버 접속을 위한 네트워크 구성 변경' 사실을 감지해 초동 조치가 가능하다. 또한 랜섬웨어의 숙주 파일을 제거해 감염 확산을 중단 시킬 수 있다. 숙주 파일 생성에 가장 빈번하게 사용하는 경로는 'C:\Users\사용자\AppData\Roaming'나 'C:\Users\사용자\Documents' 이다[6].

랜섬웨어가 모든 파일 암호화 완료 시 암호학적 요소 및 메모리 분석을 진행하거나 새로운 방안을 모색해야 한다. 감염이 종료된 후에 키 생성 요소들을 얻을 수 없다면, 랜섬웨어 감염 시작과 종료 시점 사이에 해당 요소들을 획득하는 방법이 필요하다. 이에 앞서 랜섬웨어 감염 여부를 정확하게 판단하는 것이 중요하며, 랜섬웨어에서 자주 사용하는 API나 함수 등을 통해 판별하는 방법을 고려해 볼 수 있다.

그 후에 복호화 요소를 획득하는 방법은 다음과 같이 생각해 볼 수 있다. 공개키 알고리즘을 사용하며 대상 PC에서 키 생성을 한 번 하는 경우에는 메모리상의 데이터에서 소수로 판단되는 값들을 획득한 후 개인키 생성에 적용 가능할 것이다.

VI. 결 론

본 논문에서는 2017년 6월에 배포된 Erebus에 대해 암호화 구조, 암호학적 취약점 분석 및 메모리 분석을 하였다. Erebus는 RSA-2048, AES-256, ARC4-256 세 가지 알고리즘을 사용하며, 파일 암호화에는 ARC4 스트림 암호를 이용한다. 각 ARC4 키는 AES 키로 암호화하고, AES 키는 RSA 공개키로 암호화하여 파일 헤더에 저장한다. 각 파일마다 서로 다른 암호화키를 사용하기 때문에 ARC4 키와 AES 키는 파일별로 생성된다. 따라서 모든 파일을 복호화하기 위한 가장 효율적인 방법은 RSA 개인키를 복구하는 것이다. RSA 개인키를 재현하기 위해서는 개인키 생성 시 사용된 소수, 소수 생성 시 사용된 난수, 난수 생성 시 사용된 seed 값 중 적어도 한 가지를 알아야한다. 그러나 메모리 분석 결과, 세 가지 모두 메모리 해제 후 0으로 초기화 되는 것을 확인하였다. 또한 소수 생성 시 사용된 난수는 /dev/random 난수 발생기를 이용하며, 해당 엔트로피를 재현하는 것은 암호학적으로 키에 대한 전수조사량과 유사한 복잡도를 가진다.

References

[1] RanCert, "2017 Ransomware Infringement Analysis Report," 2017-A-0201, Feb. 2017

[2] NOMORERANSOM, "https://www.nomoreransom.org/ko/partners.html"

[3] Trendmicro, "https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/erebus-linux-ransomware-impact-to-servers-and-countermeasures"

[4] armMBED, "https://tls.mbed.org/aes-source-code"

[5] AhnLab, "ASEC REPORT," pp. 1-21, Vol. 87, Apr. 2017

[6] AhnLab, "Recent Ransomware Trend Analysis," pp. 1-43, Jan. 2017

[7] ESTSecurity, "http://blog.alyac.co.kr/1204"

[8] ESTSecurity, "http://blog.alyac.co.kr/676"

[9] ESTSecurity, "http://blog.alyac.co.kr/1105"

〈저자소개〉

김 소 람 (Soram Kim) 학생회원
2016년 2월: 국민대학교 수학과 졸업
2018년 2월: 국민대학교 금융정보보안학과 석사
2018년 3월~현재: 국민대학교 금융정보보안학과 박사과정
〈관심분야〉 디지털 포렌식, 정보보호

김 지 훈 (Jihun Kim) 학생회원
2017년 2월: 국민대학교 수학과 졸업
2017년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 정보보호, 암호 알고리즘

박 명 서 (Myunseo Park) 학생회원
2013년 2월: 국민대학교 수학과 졸업
2015년 2월: 국민대학교 금융정보보안학과 석사
2014년 12월~2017년 2월: 국가보안기술연구소 연구원
2017년 3월~현재: 국민대학교 금융정보보안학과 박사과정
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식

김 대 운 (Daeun Kim) 종신회원
2015년 2월: 전남대학교 컴퓨터공학과 졸업
2017년 2월: 전남대학교 정보보안협동과정 석사
2017년 3월~현재: 한국인터넷진흥원
〈관심분야〉 암호학, 디지털 포렌식, 악성코드 분석

김 종 성 (Jongsung Kim) 종신회원
2000년 8월/2002년 8월: 고려대학교 수학 전공 학사/이학석사
2006년 11월: K.U.Leuven, ESAT/SCD-COSIC 정보보호 전공 공학박사
2007년 2월: 고려대학교 정보보호대학원 공학박사
2007년 3월~2009년 8월: 고려대학교 정보보호기술연구센터 연구교수
2009년 9월~2013년 2월: 경남대학교 e-비즈니스학과 조교수
2013년 3월~2017년 2월: 국민대학교 수학과 부교수
2014년 3월~현재: 국민대학교 일반대학원 금융정보보안학과 부교수
2017년 3월~현재: 국민대학교 정보보안암호수학과 부교수
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식