정 보 보 호 학 회 지
제 32 권 제3호, 2022. 6

스트림 암호 기반 랜섬웨어에 대한 기술적 분석 동향

이 영 주*

요 약

최근까지 랜섬웨어 공격은 끊임없이 발생하고 있으며 공격자의 암호기술에 대한 이해가 향상되면서 다양한 암호 알고리즘을 사용하여 피해자의 중요한 데이터를 암호화하고 있다. 블록 암호만을 사용하던 초창기 랜섬웨어와 달리 최근 몇 년 전부터 스트림 암호를 사용하여 데이터를 암호화하는 랜섬웨어가 계속해서 발견되고 있다. 따라서, 본 논문에서는 스트림 암호 기반 랜섬웨어의 동작과정, 암호화 과정을 기술적으로 분석하여 어떠한 형태로 악성행위를 수행하는 지 알아보고자 한다.

Ⅰ. 서 론

최근 국내·외에서 해킹으로 피해자의 데이터를 암호화하고, 이를 풀어주는 대가로 돈을 요구하는 랜섬웨어 피해가 증가하고 있다. 랜섬웨어 공격은 서비스형 랜섬웨어(Ransomware as a Service)의 활성화로 범죄형태가 분업화·조직화되고 있으며, 지속적인 취약점 공격(APT)을 수행하는 등 갈수록 지능화되고 있다.

랜섬웨어 공격 초창기에는 AES 암호 알고리즘과 같은 블록 암호만 사용해서 파일을 암호화하는 공격 형태가 많았다. 하지만, 블록 암호만 사용하면 암호키가 유출 됐을 경우 보안 전문가에 의해 쉽게 분석된다는 취약점이 존재하였다. 이에 랜섬웨어 공격자들은 RSA 암호 알고리즘과 같은 공개키 암호를 이용하여 파일 암호화에 사용한 암호키를 암호화하여 보관하는 방식을 도입하여 분석이 되더라도 복구가 불가능하게 만들었다.

또한, 랜섬웨어 공격자들은 더 많은 대상에게 랜섬웨어 공격을 수행하고 금전적 이득을 취하기 위해 파일 암호화 속도 향상의 필요성은 인지하였다. 그래서, 블록 암호보다 안전성은 상대적으로 낮지만 암호화 속도가 빠른 스트림 암호를 사용하여 파일 암호화를 시도하였다. 이러한 시도가 성공하면서 스트림 암호 기반의 랜섬웨어가 매년 유포되고 있으며 영향력이 커지고 있는 상황이다. 따라서 이 논문에서는 스트림 암호 기반의 랜섬웨어가 어떠한 방법을 이용하여 동작하고 파일 암호화를 수행하는지 분석하여 기술적 동향을 파악하고자 한다.

Ⅱ. 스트림 암호

스트림 암호(Stream Cipher)는 대칭키 암호 구조 중 하나로 유사난수를 연속적(스트림)으로 생성하여 암호화하려는 데이터와 결합하는 구조를 가진다. 일반적인 스트림 암호는 유사난수를 1비트 단위로 생성하고, 생성된 값과 암호화하려는 각 값을 XOR하여 1비트의 암호화된 자료를 얻는다[1].

최근에 랜섬웨어가 주로 사용하는 스트림 암호는 Salsa20, ChaCha 알고리즘이다. Salsa20 알고리즘은 Daniel J. Bernstein이 2005년에 개발하였으며, 이를 수정하여 2008년에 개발된 알고리즘이 ChaCha 알고리즘이다. ChaCha 알고리즘은 여러 아키텍처에서 성능을 향상시키고 확산(Diffusion)을 증가시킨 새로운 라운드 함수를 사용한다.

2.1. Salsa20

Salsa20 알고리즘의 내부상태(Internal State)는 4bytes 크기의 16개의 값으로 구성된 4×4 행렬의 구조이다. 초기상태(Initial State)는 32bytes(4bytes × 8) 크기의 Key 값, 8bytes(4bytes × 2) 크기의 Position 값과 Nonce 값, 그리고 16bytes(4bytes × 4) 크기의 고정된 문자열로 구성되어 있다. 고정된 문자열은 expa, nd 3, 2-by, te k로 나누어져 있다.

* 한국인터넷진흥원 (선임연구원, lyj5607@kisa.or.kr)

〔그림 1〕 Salsa20 알고리즘의 초기상태

Salsa20 알고리즘의 핵심은 4개의 값을 입력 받아 4개의 값을 출력하는 쿼터 라운드 $QR = (a, b, c, d)$이다.

$$b \hat{}= (a + d) \lll 7;$$
$$c \hat{}= (b + a) \lll 9;$$
$$d \hat{}= (c + b) \lll 13;$$
$$a \hat{}= (d + c) \lll 18;$$

홀수 라운드 시에는 $QR = (a, b, c, d)$를 4×4 행렬의 4개 열에 적용하고, 짝수 라운드 시에는 4개 행에 적용한다. 2개의 연속적인 라운드를 더블 라운드라고 부른다.

$QR(0, 4, 8, 12)$ // 열1
$QR(0, 4, 8, 12)$ // 열2
$QR(0, 4, 8, 12)$ // 열3
$QR(0, 4, 8, 12)$ // 열4

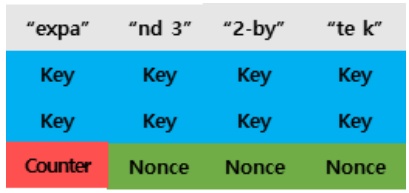
$QR(0, 4, 8, 12)$ // 행1
$QR(0, 4, 8, 12)$ // 행2
$QR(0, 4, 8, 12)$ // 행3
$QR(0, 4, 8, 12)$ // 행4

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

〔그림 2〕 4×4 행렬 구조의 Index

2.2. ChaCha

ChaCha 알고리즘의 내부 상태는 Salsa20 알고리즘과 마찬가지로 4bytes 크기의 16개의 값으로 구성된 4×4 행렬의 구조이다. 초기상태(Initial State)는 Salsa20과 유사해 보이지만 약간은 다르다. 32bytes(4bytes × 8) 크기의 Key 값, 4bytes(4bytes × 1) 크기의 Counter 값, 12bytes(4bytes × 3) 크기의 Nonce 값 그리고 16bytes(4bytes × 4) 크기의 고정된 문자열로 구성되어 있다. 그리고 행렬 요소의 위치를 재배치하여 Salsa20 알고리즘의 행렬 요소 위치와 다르다.



〔그림 3〕 ChaCha 알고리즘의 초기상태

ChaCha 알고리즘은 Salsa20 알고리즘의 쿼터 라운드 $QR = (a, b, c, d)$를 다른 형태의 쿼터 라운드로 변경하였다.

$$a \ += \ b;\ d \hat{}= \ a;\ d \lll= 16;$$
$$c \ += \ d;\ b \hat{}= \ c;\ b \lll= 12;$$
$$a \ += \ b;\ d \hat{}= \ a;\ d \lll= 8;$$
$$c \ += \ d;\ b \hat{}= \ c;\ b \lll= 7;$$

ChaCha 알고리즘은 해당 쿼터 라운드를 통해 확산 속도를 향상시켰다. 1개의 입력 비트를 변경하였을 경우 평균적으로 Salsa20 알고리즘은 8개의 출력 비트, ChaCha 알고리즘은 12.5개의 출력 비트를 변경한다. ChaCha 알고리즘은 Salsa20 알고리즘과 마찬가지로 더블 라운드를 수행하는데 홀수 라운드 시 쿼터 라운드를 열에 적용한다. 하지만, 짝수 라운드 시 대각선 열에 적용하는 점은 행에 적용하는 Salsa20 알고리즘과 다르다.

$QR(0, 4, 8, 12)$ // 열1
$QR(1, 5, 9, 13)$ // 열2
$QR(2, 6, 10, 14)$ // 열3
$QR(3, 7, 11, 15)$ // 열4

$QR(0, 5, 10, 15)$ // 대각선1

$QR(1,6,11,12)$ // 대각선2
$QR(2,7,8,13)$ // 대각선3
$QR(3,4,9,14)$ // 대각선4

Ⅲ. 스트림 암호 기반 랜섬웨어

본 장에서는 스트림 암호를 기반으로 파일을 암호화하는 랜섬웨어의 동작과정과 암호화 과정을 분석한 결과를 설명하고자 한다.

3.1. Conti 랜섬웨어

Conti 랜섬웨어는 2018년에 발견된 Ryuk 랜섬웨어의 리브랜딩 버전이자 서비스형 랜섬웨어이다[2]. 공격 대상의 시스템에 존재하는 파일들을 암호화한 후, 복구하는 대가로 금전을 요구한다. 뿐만 아니라 민감한 데이터를 탈취한 후 공개하겠다고 협박하는 방식으로 추가적인 금전을 요구한다[3].

3.1.1. 동작과정

Conti 랜섬웨어가 실행되면 파일 암호화가 중복되는 것을 방지하기 위해 'lslaif8aisuuugnzxbvmdjk'라는 이름을 가진 뮤텍스를 생성한다. 그다음, 시스템에서 인식되는 모든 논리적 드라이브를 기준으로 암호화 대상을 목록화한다. [표 1]과 같이 Windows 운영체제 및 랜섬웨어 실행과 관련된 확장자, 파일, 폴더가 제외된다.

Conti 랜섬웨어는 모든 경로에 랜섬노트(R3ADM3.txt)를 생성한 후, 멀티 스레드를 이용하여 파일을 빠른 속도로 암호화한다[4].

〔표 1〕 암호화 제외 대상

확장자	lnk, exe, msi, dll, sys
파일	readme.txt, CONTI_LOG.txt
폴더	tmp, winnt, temp, thumb, Windows, Boot, $Recycle.Bin, Trend Micro, System Volume Information,

3.1.2. 암호화 과정

Conti 랜섬웨어는 Microsoft에서 제공하는 CryptoAPI의 함수를 사용하여 파일 암호화를 수행한다. 먼저, CryptGenRandom 함수를 사용하여 암호화 시 사용할 암호키와 nonce 값을 생성한다. 크기는 각각 32bytes와 8bytes이다.

그다음, 생성한 암호키와 nonce 값을 사용하여 공격자가 하드코딩한 ChaCha8 스트림 암호로 파일을 암호화하는데 [표 2]와 같이 파일의 크기에 따라 다른 암호화 방식이 적용된다. 그리고 적용된 암호화 방식을 파악하기 위해 암호화된 파일의 끝에 플래그를 추가한다.

Conti 랜섬웨어는 파일 암호화 시 사용한 암호키와 nonce 값을 공격자의 공개키로 암호화하며, RSA-4096 암호 알고리즘을 사용한다. 해당 공개키는 하드코딩되어 있으며 CryptImportKey 함수를 사용하여 값을 불러온다. 암호화된 파일 암호키와 nonce 값은 암호화된 파일의 끝 부분에 저장된다[5].

〔표 2〕 암호화 방식

구분	암호화 방식	플래그
1MB 이하	파일 전체	0x24
1MB~5MB	파일 앞부분 1MB	0x26
5MB 이상	0xFFFFA bytes 단위로 블록을 나눈 후 홀수 번째만	0x25

〔그림 4〕 암호화된 파일 구조

3.2. Ragnar 랜섬웨어

Ragnar 랜섬웨어는 2019년 말에 발견된 랜섬웨어

로 주로 기업 네트워크를 노린다. 피해자의 파일을 암호화하기 위해 윈도우 XP 가상 머신에 배치되어 호스트에 설치된 보안 소프트웨어의 탐지를 우회한다[5]. 금전적 이득을 극대화하기 위해 주로 기업을 대상으로 공격한다.

3.2.1. 동작과정

Ragnar 랜섬웨어는 Windows에서 제공하는 GetLocalInfo 함수를 사용하여 특정 언어를 사용하는 시스템을 제외한다. 해당 언어는 Azerbaijani, Armenian, Belorussian, Kazakh, Kyrgyz, Moldavian, Tajik, Russian, Turkmen, Uzbek, Ukrainian, Georgian 총 12개이다.

Ragnar 랜섬웨어는 [표 3]과 같이 파일 암호화 수행에 있어 방해가 되는 특정 서비스를 중지시킨다. 중지 대상 목록은 랜섬웨어 실행파일 내부에 암호화되어 있으며 RC4 암호 알고리즘을 사용하여 복호화한다. 이때 사용되는 키 값은 0x40bytes 크기로 고정된 값이며, 실행파일 내부에 하드코딩되어 있다.

〔표 3〕 서비스 중지 대상

vss	sql	memtas
mepocs	sophos	veeam
backup	pulseway	logme
logmein	connectwise	splashtop

〔표 4〕 암호화 제외 대상

확장자	db, msi, sys, drv, dll, exe, lnk
파일	autorun.inf, boot.ini, bootfont.bin, bootsect.bak, desktop.ini, ntldr, iconcache.db, ntuser.dat, ntuser.dat.log, ntuser.ini, thumbs.db, 랜섬노트
폴더	Windows, Windows.old, Tor browser, Internet Explorer, Google, Opera, Opera Software, $Recycle.Bin, Mozilla, Mozilla Firefox, ProgramData, All Users

Ragnar 랜섬웨어는 [표 4]와 같이 특정한 확장자, 파일, 폴더를 암호화에서 제외한다.

Ragnar 랜섬웨어는 암호화가 완료되면 CreateProcessAsUser 함수를 사용하여 랜섬노트를 메모장으로 실행시킨 후, ExitProcess 함수를 사용하여 랜섬웨어 실행파일을 종료한다[6].

3.2.2. 암호화 과정

Ragnar 랜섬웨어는 Microsoft에서 제공하는 CryptoAPI의 CryptGenRandom 함수를 사용하여 난수를 생성한다. 생성된 난수에 SHA-512를 적용한 해시값의 일부분을 seed 값으로 사용한다.

첫 번째 seed 값은 a값의 상위 0x28bytes이다.

$$rn_1 = CryptoGenRandom()$$
$$a = SHA512(rn_1)$$

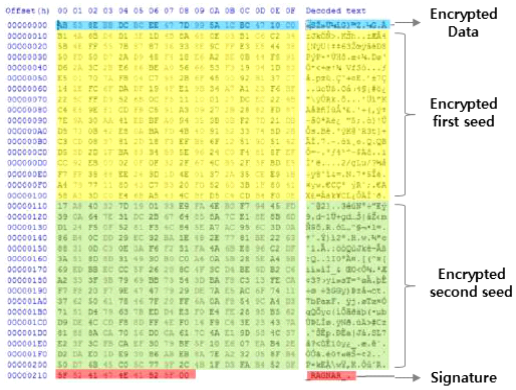
두 번째 seed 값은 b값의 상위 0x20bytes이다.

$$rn_2 = CryptoGenRandom()$$
$$b = SHA512(rn_2)$$

2개의 seed 값이 생성되면 해당 값을 사용하여 Salsa20 스트림 암호의 initial state를 생성한다. initial state는 4bytes 크기의 8개의 값을 가진 Key, 4bytes 크기의 2개의 값을 가진 Pos와 Nonce, 4bytes 크기의 4개의 값을 가진 고정된 문자열로 구성된다. 이때 Pos는 항상 0이다.

Ragnar 랜섬웨어는 앞서 생성한 첫 번째 seed 값의 상위 32bytes 크기의 값을 Key로, 하위 8bytes 크기의 값을 Nonce로 사용한다. 그리고, 일반적으로 사용되는 Salsa20 스트림 암호와 달리 2개의 seed 값을 무작위로 연산한 값(시프트 연산, 산술 연산 등)을 고정된 문자열로 사용한다. initial state 생성이 완료되면 Salsa20 스트림 암호로 파일을 암호화한다.

파일 암호화에 사용된 2개의 seed 값은 공격자의 공개키로 암호화되며, 이때 사용되는 암호 알고리즘은 RSA-2048이다.공격자의 공개키는 랜섬웨어 실행파일 내부에 암호화된 상태로 저장되어 있으며, RC4 스트림 암호로 복호화된다. 암호화된 파일은 암호화된 데



〔그림 5〕 암호화된 파일 구조

이터, 암호화된 seed 값, Signature 순으로 구성되어 있다. Signature는 '_RAGNAR_' 문자열과 Null(0x00) 값으로 구성되어 있다[7].

3.3. BlackMatter 랜섬웨어

BlackMatter 랜섬웨어는 2021년 7월 말에 발견된 서비스형 랜섬웨어이다. 타 사이버 범죄 조직을 통해 공격 대상 시스템의 접근 권한을 구매한 후 시스템에 침투하는 방식으로 랜섬웨어 실행파일을 유포한다[8].

3.3.1. 동작과정

BlackMatter 랜섬웨어는 IAT(Import Address Table)에 자신이 사용하는 함수를 숨기기 위해 라이브러리의 이름을 암호화한다. 해당 랜섬웨어는 각 라이브러리에 대한 암호화된 해시 블록을 가지고 있다. 해당 값은 키(0x22065FED)를 이용해 복호화할 수 있다.

BlackMatter 랜섬웨어는 PE 구조 상 '.rsrc' 섹션에 환경설정 파일이 저장되어 있다. 해당 파일은 LZ(Lempel-Ziv) 기반 압축 라이브러리이자 패킹에 주로 사용되는 알고리즘인 aPlib 알고리즘으로 압축이 해제된다. 환경설정 파일에는 C&C(Command & Control) 서버 호스트 이름, C&C용 AES-128 암호키, 중지할 서비스 및 프로세스 목록, 암호화 제외 대상 확장자, 파일, 폴더 목록, 랜섬노트가 저장되어 있다.

BlackMatter 랜섬웨어는 [표 5]와 같이 파일 암호화 수행에 있어 방해가 되는 특정 서비스와 프로세스를 중지시킨다.

〔표 5〕 서비스 및 프로세스 중지 대상

서비스	mepocs, memtas, veeam, svc$, backup, sql, vss, msexchange
프로세스	ensvc, thebat, mydesktopqos, xfssvccon, firefox, infopath, winword, steam, synctime, notepad, ocomm, onenote, mspub, thunderbird, agensvc, sql, excel, powerpnt, outlook, wordpad, dbeng50, visio isqlplussvc, sqbcoreservice, oracle, ocautoupds, dbsnmp, msaccess, tbirdconfig, ocssd, mydesktopservice

특정 서비스와 프로세스가 중지된 후 백업을 통해 파일 복구가 가능한 볼륨 섀도 복사본이 삭제된다. 그리고 파일 암호화를 수행하는데 [표 6]과 같이 특정한 확장자, 파일, 폴더는 암호화에서 제외된다. 암호화가 완료된 후 모든 폴더마다 랜섬노트가 생성된다[9].

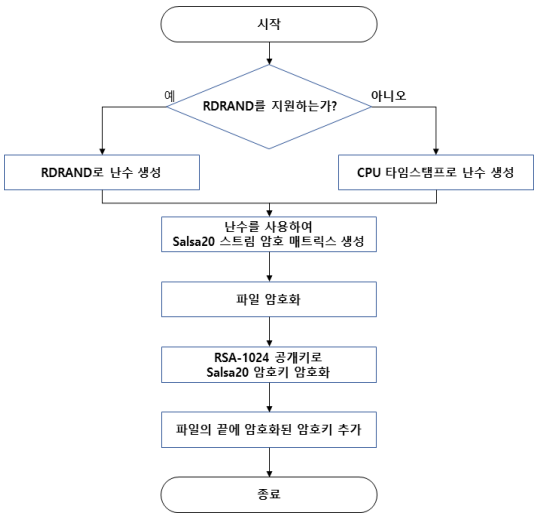
〔표 6〕 암호화 제외 대상

확장자	themepack, nls, diagpkg, msi, lnk, exe, cab, scr, bat, drv, rtp, msp, prf, msc, ico, key, ocx, diagcab, diagcfg, pdb, wpx, hlp, icns, rom, dll, msstyles, mod, ps1, ics, hta, bin, cmd, ani, 386, lock, cur, idx, sys, com, desktheme, pack, shs, ldf, theme, mpa, nomedia, spl, cpl, adv, icl, msu
파일	desktop.ini, autorun.inf, ntldr, bootsect.bak, thumb.db, boot.ini, ntuser.dat, iconcache.db, bootfont.bin, ntuser.ini, ntuser.dat.log
폴더	system volume information, intel, $windows.~ws, application data, $recycle.bin, mozilla, program files (x86), program files, $windows.~bt, public, msocache, windows, default, all users, tor browser, programdata, boot, config.msi, google, perflogs, appdata, windows.old

3.3.2. 암호화 과정

BlackMatter 랜섬웨어는 Salsa20 스트림 암호를 사용하여 파일을 암호화하고 RSA-1024 공개키 암호를 사용하여 파일 암호화에 사용한 암호키를 암호화한다. 이때, 파일의 크기가 1MiB 이하면 파일 전체를 암호화하고 1MiB를 초과하는 경우 파일의 첫 1MiB만 암호화한다.

파일 암호화에 사용되는 Salsa20 암호키는 무작위로 생성되며, nonce 값은 0이다. 무작위 값은 CPU의 RDRAND 인스트럭션 지원 여부에 따라 달라진다. RDRAND 인스트럭션을 지원하는 경우에는 인스트럭션을 2번 실행하여 생성한 2개의 값을 연접하여 난수를 생성한다. 지원하지 않는 경우 2개의 CPU 타임스탬프 카운터 값의 하위 32bits를 연접하여 난수를 생성한다. Salsa20 암호키는 파일 암호화에 사용된 후 RSA-1024 공개키로 암호화되어 암호화된 파일의 끝부분에 저장된다. 파일의 끝엔 암호화된 파일 암호키와 원본 파일의 크기 등 파일 복구에 사용되는 정보가 저장된다[10].



〔그림 6〕 BlackMatter 랜섬웨어 암호화 과정

3.4. Babuk Locker 랜섬웨어

Babuk Locker 랜섬웨어는 2021년 1월에 발견된 서비스형 랜섬웨어이다[11]. 실행파일 형태로 이메일, 스피어피싱 등 다양한 경로를 통해 유포된다.

3.4.1. 동작과정

Babuk Locker 랜섬웨어는 Command Line 명령어를 사용하여 네트워크 드라이브와 로컬 드라이브의 암호화 순서를 결정한다. '-lanfirst' 명령어를 사용하면 네트워크 드라이브를 암호화한 다음 로컬 드라이브를 암호화한다. '-lansecond' 명령어를 사용하면 로컬 드라이브를 암호화한 후 네트워크 드라이브를 암호화한다. '-nolan' 명령어를 사용하면 로컬 드라이브만 암호화한다.

Babuk Locker 랜섬웨어는 [표 7]과 같이 파일 암호

〔표 7〕 서비스 및 프로세스 중지 대상

서비스	vss, sql, svc$, memtas, mepocs, VeeamTransportSvc, sophos, veeam, backup, BackupExecVSSProvider, GxVss, GxBlr, GxFWD, GxCVD, BackupExecAgentAccelerator, GxClMgr, DefWatch, ccEvtMgr, ccSetMgr, BackupExecAgentBrowser, SavRoam, RTVscan, QBFCService, QBIDPService, Intuit.QuickBooks.FCS, BackupExecDiveciMediaService, QBCFMonitorService, YooBackup, YooIT, BackupExecJobEngine, zhudongfangyu, VeeamTransportSvc, stc_raw_agent, VSNAPVSS, BackupExecManagementService, AcronisAgent, CASAD2DWebSvc, CAARCUpdateSvc
프로세스	sql.exe, oracle.exe, ocssd.exe, dbsnmp.exe, winword.exe, visio.exe, synctime.exe, agntsvc.exe, isqlplussvc.exe, xfssvccon.exe, wordpad.exe, mydesktopservice.exe, ocautoupds.exe, encsvc.exe, firefox.exe, notepad.exe, tbirdconfig.exe, mydesktopqos.exe, ocomm.exe, dbeng50.exe, steam.exe, excel.exe, sqbcoreservice.exe, infopath.exe, msaccess.exe, thebat.exe, mspub.exe, onenote.exe, outlook.exe, powerpnt.exe, thunderbird.exe

화 수행에 있어 방해가 되는 특정 서비스와 프로세스를 중지시킨다.

Babuk 랜섬웨어는 특정 서비스와 프로세스를 중지시킨 후 볼륨 섀도 복사본을 삭제하여 백업을 통해 시스템을 복구할 수 없게 한다. 파일 암호화가 완료되면 폴더마다 랜섬노트가 생성된다. 랜섬노트에는 유출된 파일의 샘플 URL과 랜섬웨어 공격자와 복구비용과 관련하여 협상할 수 있는 Tor 사이트로 연결되는 링크가 포함되어 있다[12].

3.4.2. 암호화 과정

Babuk Locker 랜섬웨어는 다양한 버전으로 유포되고 있으며, 모든 버전에서 스트림 암호와 공개키 암호를 같이 사용하는 하이브리드 형태의 암호화 방식을 취하고 있다. 하지만 버전에 따라 암호화에 사용된 암호가 다르다. 버전 3의 경우 ChaCha20 스트림 암호를 사용하며 나머지 버전은 ChaCha8 스트림 암호를 사용한다.

파일 암호화에 사용되는 첫 번째 암호키와 nonce 값은 무작위로 생성된 값이다. 암호화가 완료되면 무작위로 공개키 쌍이 생성되고 ECDH를 기반으로 두번째 암호키와 nonce 값을 생성한다. 키 교환에 사용되는 공격자의 공개키는 랜섬웨어 실행파일 내부에 하드코딩되어 있다. 키 교환을 통해 생성된 비밀값에 SHA-256이 적용되고 그 결과 값은 두 번째 암호키로 사용된다. 두 번째 nonce 값은 앞서 생성된 비밀값의 상위 8bytes이다.

Babuk Locker 랜섬웨어는 파일 암호화 시 파일 크기에 따라 다른 암호화 방식을 적용한다. 파일 크기가 41MB 이하인 경우 버전에 따라 ChaCha20 또는 ChaCha8 스트림 암호로 파일 전체가 2번 암호화한다. 파일 크기가 41MB 초과인 경우 파일 전체를 세 영역으로 나눈 후 각 영역의 상위 10MB만 2번 암호화한다[13].

Ⅳ. 결 론

4종(Conti, Ragnar, BlackMatter, Babuk Locker)의 랜섬웨어 동작과정 및 암호기능 분석 사례를 통해 파일 암호화 시 스트림 암호가 자주 사용되고 있음을 확인하였다. 대부분 Winodws의 CryptoAPI에서 제공하는 함수를 사용하여 암호키 생성에 필요한 무작위 값들을 추출하고 파일 암호화를 수행한다.

최근 들어, 랜섬웨어 제작자의 기술 수준이 향상되면서 갈수록 랜섬웨어 공격이 정교해지고 있다. 블록암호뿐만 아니라 스트림 암호를 이용하여 파일을 암호화하는 사례도 계속해서 보고되고 있다. 이러한 추세에 맞춰 스트림 암호 기반 랜섬웨어를 심층적으로 분석하고 복구할 수 있는지 확인하는 연구를 진행하고분석 사례를 확보해야 할 필요가 있다.

참 고 문 헌

[1] “Wikipedia”, https://en.wikipedia.org/wiki/Salsa20

[2] “DailySecu”, https://www.dailysecu.com/news/articleView.html?idxno=118479

[3] 성윤기, 김기문, 이영주, “Conti 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 2-4, 2021

[4] 성윤기, 김기문, 이영주, “Conti 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 5-6, 2021

[5] “ESTsecurity, https://blog.alyac.co.kr/3009”

[6] 박창열, 김기문, 김대운, “Ragnar 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 2-8, 2020

[7] 박창열, 김기문, 김대운, “Ragnar 랜섬웨어 암호기능 분석 보고서”, 한국인터넷진흥원, pp. 9-12, 2020

[8] “ESTsecurity”, https://blog.alyac.co.kr/3960

[9] 성윤기, 김기문, 이영주, “2021년 3분기 랜섬웨어 동향 보고서”, 한국인터넷진흥원, pp. 29-31, 2021

[10] 성윤기, 김기문, 이영주, “2021년 3분기 랜섬웨어 동향 보고서”, 한국인터넷진흥원, pp. 27-28, 2021

[11] “Trend Micro”, https://www.trendmicro.com/en_us/research/21/b/new-in-ransomware.html

[12] 성윤기, 김기문, 이영주, “2021년 1분기 랜섬웨어 동향 보고서”, 한국인터넷진흥원, pp. 12-14, 2021

[13] 성윤기, 김기문, 이영주, “2021년 1분기 랜섬웨어 동향 보고서”, 한국인터넷진흥원, pp. 11-12, 2021

〈저 자 소 개〉

이 영 주 (Yeong Ju Lee)
정회원
2017년 2월 : 전남대학교 지리학과 졸업
2018년 6월~현재 : 한국인터넷진흥원 차세대암호융합팀 선임연구원
<관심분야> 랜섬웨어, 암호기술