SK 쉴더스

Threat Intelligence Report

EQST INSIGHT

EQST(이큐스트)는 'Experts, Qualified Security Team' 이라는 뜻으로
사이버 위협 분석 및 연구 분야에서 검증된 최고 수준의 보안 전문가 그룹입니다.

2025
04

SK 쉴더스

Contents

Headline

의료기관을 겨냥한 사이버 공격 증가와 보안 대응 전략

EQST 사업그룹 EQST Lab 팀 김세용 수석

■ 개요

디지털 기술의 발전은 의료 서비스의 질을 향상시키는 한편, 새로운 형태의 보안 위협도 함께 초래하고 있다. 병원은 방대한 양의 개인정보와 진료 데이터를 보유하고 있으며, 다양한 의료기기들이 네트워크로 연결된 복합적인 IT 환경에서 운영 중이다. 특히 24 시간 작동되는 시스템의 경우, 시스템 중단이 환자의 생명과 직결될 수 있어 점검이나 유지보수가 자유롭지 않다. 이와 같은 특수성은 공격자에게 표적이 되며, 이를 노린 해킹, 랜섬웨어, DDoS 와 같은 시스템 마비형 공격이 지속적으로 시도되고 있다.

구분	계	발생유형		
		DDoS 공격	악성코드 감염·유포 (랜섬웨어)	시스템해킹
2020 년	5	0	5(5)	0
2021 년	5	2	1(1)	2
2022 년	23	1	5(5)	17
2023 년	39	1	7(6)	31
2024 년	57	0	4(4)	53
계	129	4	22(21)	103

* 출처 : 보안뉴스(자료 : 한국인터넷진흥원, 단위 : 건)

표 1. 의료기관 침해사고

사고 증가 추세

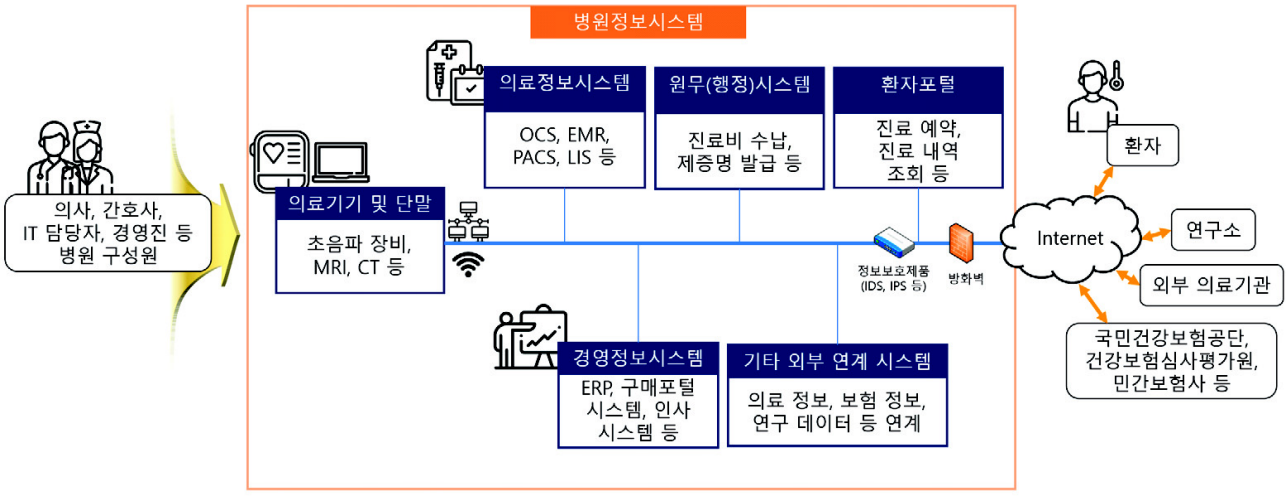
최근 몇 년간 병원을 대상으로 한 사이버 공격은 국내외에서 꾸준히 증가하고 있다. 해외에서는 대형 병원 네트워크가 랜섬웨어에 감염되어 수술 일정이 취소되거나 환자 기록 접근이 차단되는 사례가 있었으며, 국내에서는 환자 정보 유출 사고가 발생했다.

이러한 사건들은 의료기관의 보안 수준에 대한 우려를 키우고 있으며, 병원 내 정보자산과 시스템 보호에 대한 필요성을 더욱 부각시키고 있다.

시스템 복잡성

병원의 정보기술 인프라는 전자의무기록(EMR), 병원 내 네트워크, 의료장비 시스템 등으로 구성되어 있으며, 진료 편의성과 업무 효율성을 위해 지속적으로 디지털화되어 왔다.

하지만 그 이면에는 여러 보안 취약점이 존재한다. 예를 들어, 외부 협력업체 시스템과의 연계, 인터넷을 통한 장비 접근, 구 버전 소프트웨어의 지속적 사용 등이 해커의 주요 침투 경로로 악용될 수 있다.



* 출처 : 국가정보원 병원정보시스템 보안가이드라인(2025.4)

그림 1. 병원정보시스템

정책 대응 현황

사이버 위협이 현실화되면서 정부 차원의 대응도 강화되고 있다. 보건복지부, 국가정보원, 한국인터넷진흥원(KISA) 등 관계 기관은 의료기관의 보안 역량 강화를 위한 정책적 가이드라인을 마련하고 있으며, 그 중에서도 2025 년 4 월 발표된 '병원정보시스템 보안 가이드라인'은 현장의 실무자가 참고할 수 있는 구체적인 기준을 제시하고 있다. 이 가이드는 병원이 보안 체계를 점검하고, 보유 자산을 식별하며, 다양한 위협에 대응할 수 있도록 기반을 마련하는 데 목적이 있다.

■ 병원 보안의 구조적 취약성

운영 환경의 제약

의료기관은 생명과 직결된 진료를 수행하는 조직 특성상, 연속적인 시스템 운영이 절대적으로 요구된다. 그로 인해 병원은 일반 기업과 달리 시스템 점검, 네트워크 변경, 보안 패치 적용 등을 자유롭게 시행하기 어렵다. 또한, 각 진료과와 외부 협력 조직이 병원 시스템에 접근하는 구조는 보안 정책을 일관되게 적용하기 어렵게 하는 요인이 된다.

기술 인프라의 노후화

일부 병원에서 사용 중인 정보 시스템은 개발된 지 오래되어 운영체제가 구형이거나, 보안 패치가 더 이상 제공되지 않는 상태로 운용되고 있다. 전자의무기록(EMR), 처방전달시스템(OCS), 의료영상저장전송시스템(PACS) 등은 서로 독립적으로 구축되어 있어 통합 관리가 어렵고, 이들 간 연동을 위한 인터페이스는 새로운 보안 취약점의 발생 가능성을 높이고 있다.

의료기기의 보안 한계

병원 내 다양한 의료기기는 진단, 치료, 감시 목적으로 사용되며 네트워크로 연결되는 경우가 많다. 그러나 상당수 기기는 개발 당시부터 보안을 고려해 설계되지 않았고, 구형 운영체제 또는 변경되지 않은 기본 계정·비밀번호 상태로 운용되는 경우가 있다. 또한 펌웨어가 업데이트되지 않아 취약점이 장기간 방치되는 상황도 발생한다. 의료기기는 고가의 장비인 만큼 교체가 현실적으로 어려우며, 감염 시 내부망을 통해 병원 전체 시스템에 영향을 줄 수 있는 위험 요소가 된다.

외부 연계 시스템의 위험

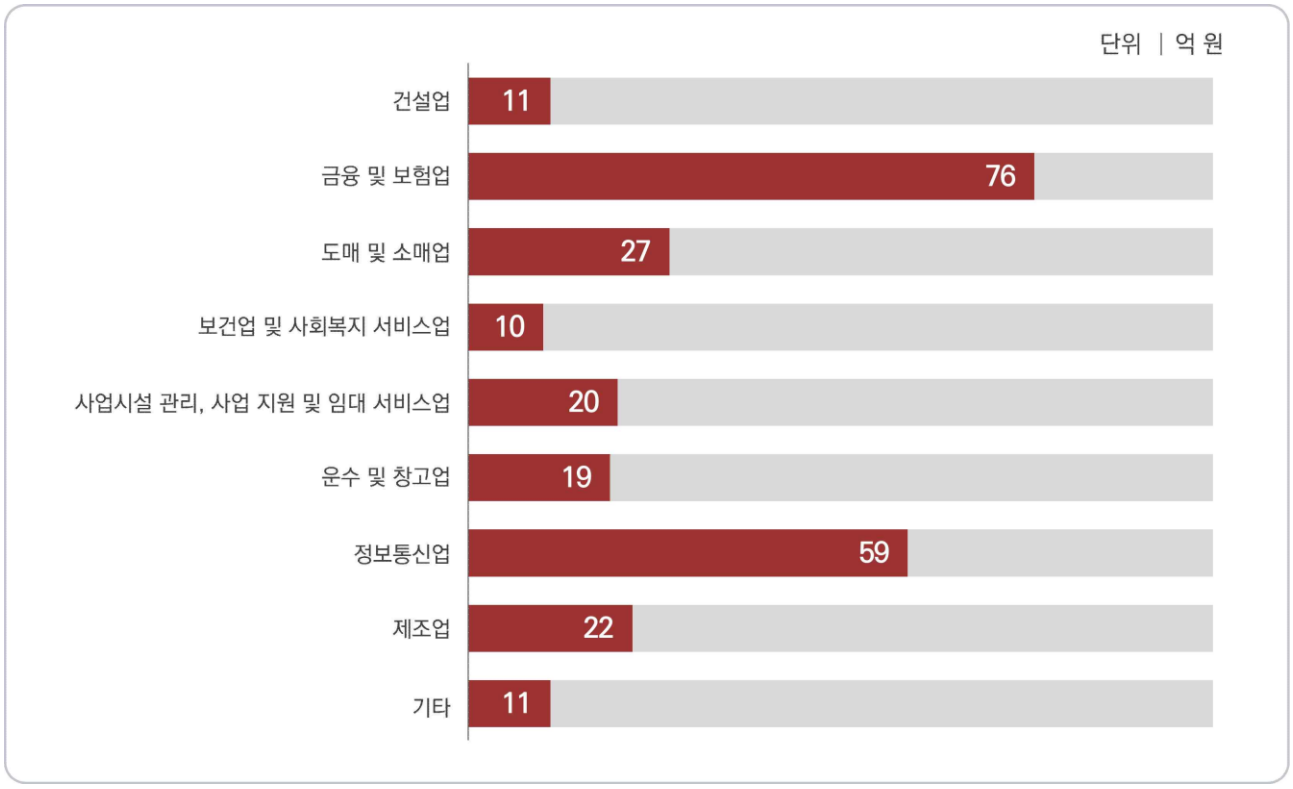
원격진료, 클라우드 기반 플랫폼, 원내외 협력 시스템 등은 병원 네트워크의 경계를 점점 확장 시키고 있으며, 이 과정에서 새로운 공격 지점이 형성된다. 외부에서 사용하는 VPN, API 연동 포인트, 진료정보 교류시스템 등은 인증 부족, 암호화 미적용, 접근통제 부재와 같은 문제로 인해 해킹에 취약한 환경이 될 수 있다.

보안 책임 체계의 부재

병원 내 정보보안에 대한 인식은 상대적으로 낮은 편이다. 또한, 전문적이고 독립적인 보안 조직이 부재하고 전담 예산이 확보되지 않는 경우가 많다. 특히 중소형 병원의 경우, 위기 상황이 발생해도 신속하게 대응하기 어려우며, 사고 분석이나 재발 방지를 위한 체계적인 대응도 쉽지 않다. 이러한 조직적 한계는 기술적 취약점보다도 더 본질적인 보안 위협이 될 수 있다.

보안 투자와 인식의 한계

정보보호의 중요성이 점차 부각되고 있음에도 불구하고, 실제 의료기관의 보안 투자는 여전히 낮은 수준에 머물러 있다. 한국인터넷진흥원(KISA)이 2024 년에 12 월에 발표한 조사에 따르면, 의료 업종의 정보보호 투자액은 8 개 주요 산업군 중 가장 낮은 수준으로 나타났다. 대한병원협회도 중소병원을 중심으로 보안 예산 편성과 전문 인력 확보가 어려운 현실을 지적하고 있다. 외부 감사나 보안 인증 획득을 위한 목적으로 일시적으로 투자를 진행하지만, 일상적 운영에서 보안을 전략적으로 관리하는 조직 문화는 부재하다고 평한다. 보안을 선택 가능한 항목이 아닌 필수 인프라로 바라보는 시각의 전환이 필요하다.



* 출처: 2024 정보보호 공시 현황 분석 보고서

그림 2. 업종별 평균 정보보호 투자액

■ 법적 책임과 보상 구조

병원 보안에 대한 투자와 관리 체계가 충분히 자리 잡지 못한 또 다른 원인 중 하나는, 정보 유출 사고에 따른 법적·도의적 책임이 미약하다는 점이다. 의료정보는 가장 민감한 개인정보 중 하나임에도 불구하고, 사고 발생 시 피해자에 대한 실질적인 보상이 이뤄지지 않거나, 기관이 감수해야 할 책임 범위가 상대적으로 낮은 편이다.

국내 사례

국내에서는 사고 발생 시 과태료 부과나 재발 방지 권고에 그치는 경우가 많다. 2018 년부터 2020 년까지 17 개 종합병원에서 환자 정보 유출 사고가 발생했지만, 개인정보보호위원회는 이 중 16 개 병원에 과태료를 부과하고 시정 권고를 내렸을 뿐, 피해자에 대한 직접적인 보상은 이루어지지 않았다.

해외 사례

반면 해외에서는 환자 개인정보 유출에 대해 손해 배상이 실제로 이루어지고 있다. 미국의 A 병원은 2023 년 해킹 사고로 환자 정보가 유출된 사건에서 약 65 만 달러(약 9 억 원)를 피해자 집단에 지급하기로 합의하고, 추가로 2 년간 무료 신용 모니터링 서비스를 제공했다.

또 다른 사례로, 미국 보건의료 기업 B 사는 같은 해 개인정보 유출 사고에 대해 각 피해자에게 최대 10,000 달러(약 1,400 만 원)의 보상을 포함한 합의 절차를 진행 중이다.

■ 보안 사고 분석

의료기관을 대상으로 한 사이버 공격은 시스템 마비, 진료 중단, 환자정보 유출 등 다양한 형태로 나타나고 있으며, 피해는 단순한 금전적 손실을 넘어 환자의 생명과 직결되는 상황으로 이어질 수 있다. 국내외 병원에서 발생한 대표적인 사례들은 의료기관이 직면한 보안 문제를 보여준다. 특히 최근에는 북한 해킹 조직이 병원을 표적으로 삼고 있다는 정황이 지속적으로 포착되고 있으며, 국가정보원은 북한이 보건의료기관을 주요 타깃으로 삼아 다양한 침투 시도를 하고 있다고 발표했다.

국내 사례

2021 년 C 병원은 북한 해킹 조직의 공격을 받아 내부망이 침해되는 대규모 보안 사고를 겪었다. 경찰청에 따르면, 공격자는 약 두 달간 병원 네트워크에 잠입해 환자 약 81 만 명과 전·현직 직원 1 만 7 천여 명의 개인정보를 유출했다. 공격은 국내외 외부 서버를 경유하고 병원 시스템 취약점을 이용하여 이루어졌으며, 경찰청은 공격 근원지의 IP 주소, IP 주소 세탁 기법, 시스템 침입·관리 수법, 북한 어휘 사용 등을 근거로 북한 해킹 조직의 소행으로 판단하였다.

2023 년 7 월에는 17 개 종합병원에서 총 18 만 5 천여 건의 환자 개인정보가 유출된 사건이 발생했다. 경찰 수사에 따르면, 병원 또는 제약사 직원이 특정 처방 정보를 확인하고 이를 이메일이나 USB 등을 통해 외부로 유출한 것으로 확인되었으며, 이는 의약품 판매질서 위반 관련 수사 과정에서 드러났다.

해외 사례

2020 년 9 월, 독일 D 병원은 외부 해킹 공격으로 전산망이 마비되었고, 이로 인해 응급 진료와 예약 시스템이 중단되었다. 응급환자 한 명이 다른 병원으로 이송 도중 사망하는 안타까운 사례까지 발생했으며, 원인은 Citrix VPN 장비의 취약점을 통한 침입으로 밝혀졌다.

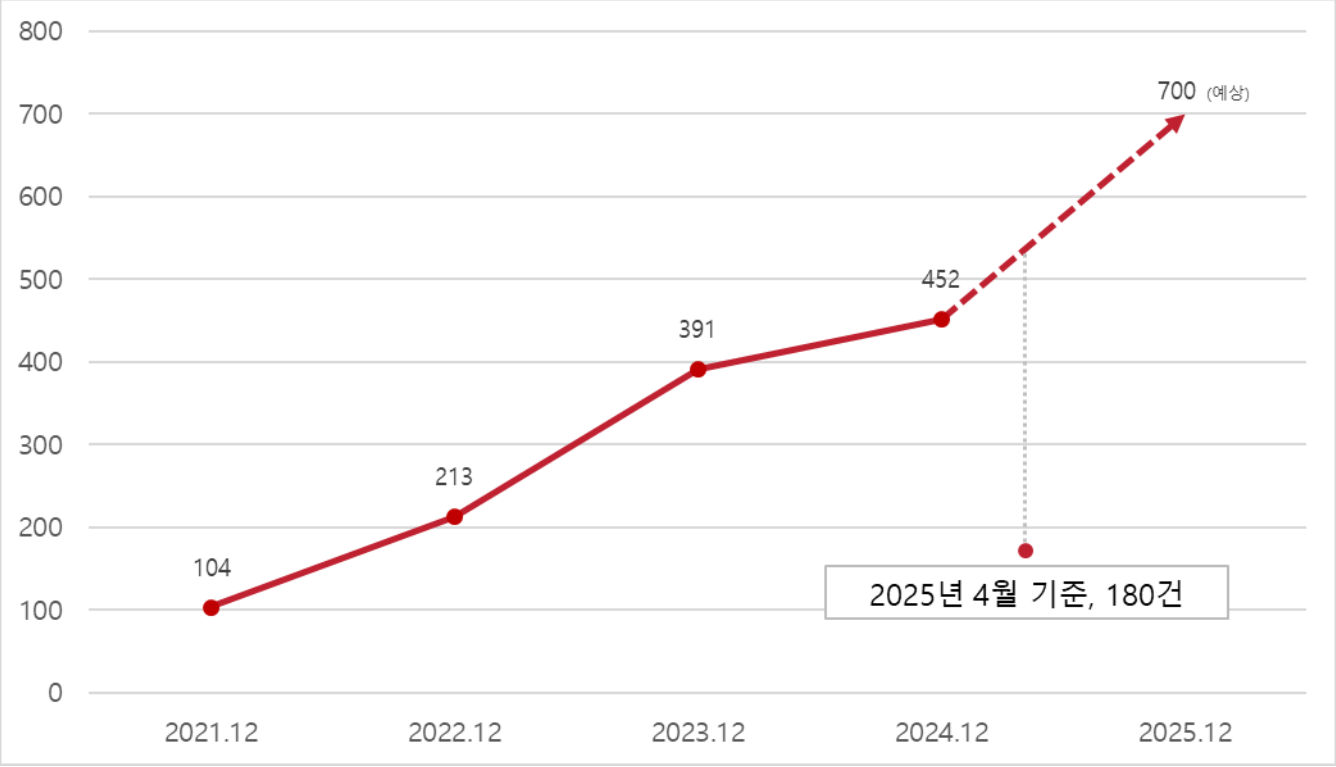
또한 2022 년 10 월, 미국의 비영리 의료조직인 E 사는 랜섬웨어 공격을 받아 21 개 주의 164 개 병원 및 진료소의 시스템이 동시에 마비되었고, 약 62 만 명의 환자 정보가 유출되었다. 공격은 외부 협력업체와의 시스템 연계 경로를 통해 이루어진 것으로 추정되고 있다.

■ 환자 개인정보 유출의 영향

병원이 다루는 정보는 단순한 이름이나 연락처 수준을 넘어, 환자의 진단명, 병력, 정신건강기록, 수술 이력, 감염병 내역 등 고도로 민감한 의료 기록을 포함한다. 이러한 정보가 유출될 경우, 그 피해는 단기간에 그치지 않으며, 취업·보험·사회적 낙인 등 다양한 측면에서 환자의 삶에 장기적인 영향을 미칠 수 있다.

최근 다크웹에서는 민감한 의료정보가 지속적으로 유통되고 있으며, 이들 데이터는 금융 범죄나 사회공학적 공격의 재료로 활용될 가능성이 높다. 유출된 의료정보는 한 번 퍼지면 사실상 삭제나 회수가 불가능하고, 어디까지 퍼졌는지 추적하기도 어렵다. 그 결과 피해자는 오랜 기간 2 차 피해에 노출되며, 이 피해는 시간이 지날수록 누적되고 심화되는 양상을 보인다.

실제로 다크웹에 게시된 의료 데이터 건 수는 2021 년 104 건에서 2024 년 452 건으로 약 4 배 증가했으며, 2025 년 4 월 현재까지 180 건이 확인되었다. 이 같은 증가 추세가 이어질 경우, 2025 년 연말까지 의료정보 게시 건 수는 700 건을 넘어설 것으로 예상된다.



* 출처 : SK 쉴더스

그림 3. 의료 데이터 다크웹 게시 건 수

■ 병원 정보보안 및 개인정보보호 컴플라이언스 의무사항

의료법에 따른 환자 정보보호 의무

의료법은 의료기관이 환자의 진료기록과 개인정보를 철저히 보호할 책임이 있음을 법적으로 명시하고 있다. 의료법 제19조에서는 의료인이 진료나 조제 과정 등 업무 중 알게 된 환자의 비밀을 누설하거나 발표해서는 안 된다고 규정하며, 이를 위반할 경우 형사처벌의 대상이 된다. 또한 제21조는 환자 본인이나 정당한 법적 권한을 가진 자에 한해 진료기록 열람을 허용하고 있으며, 제22조는 진료기록의 보존과 전자의무기록(EMR)의 안전한 관리 조치를 의무화하고 있다. 이러한 조항들은 환자 정보를 보호하기 위한 기본적인 법적 기반을 제공하며, 위반 시 과태료 또는 행정처분 등의 법적 제재가 따르게 된다.

개인정보 보호법의 병원 적용

병원은 개인정보보호법상 '개인정보처리자'로 분류되며, 이에 따라 환자의 개인정보를 수집, 저장, 활용, 파기하는 전 과정에서 법적 책임을 진다. 개인정보보호법 제29조 및 시행령 제30조 등에 따라 의료기관은 개인정보를 안전하게 보호하기 위한 관리적·기술적·물리적 보호조치를 반드시 마련해야 하며, 제34조에 따라 개인정보 유출이 발생하면 즉시 관계기관에 신고하고 피해 확산을 방지해야 한다. 위반 시 과징금, 과태료 부과 및 형사처벌까지 가능하므로 병원의 정보보안 체계는 이 법의 요구사항을 충족해야 한다.

ISMS, ISMS-P 인증 의무화

ISMS(정보보호 관리체계) 인증, ISMS-P(정보보호 및 개인정보보호 관리체계) 인증은 과학기술정보통신부와 개인정보보호위원회가 공동으로 시행하는 국가 인증제도다. 정보통신망법 제47조 2항에 따라 전년도 기준으로 총 매출액 또는 세입이 1,500억 원 이상이며 의료법 제3조의4에 따른 상급 종합병원이거나, 정보통신서비스 매출이 100억 원 이상인 경우, 혹은 하루 평균 100만 명 이상의 이용자를 보유한 의료기관은 인증을 의무적으로 받아야 한다. 인증 기준에는 관리체계 수립 및 운영, 보호대책 요구사항, 개인정보 처리단계별 요구사항 등 3개 부문의 등 정보보호 전 영역이 포함된다.

전자서명법에 따른 전자의무기록(EMR) 보안 요건

전자의무기록(EMR)은 환자의 진료 정보가 집약된 민감한 데이터로서, 정보보안의 핵심 대상 중 하나다. 의료기관이 전자의무기록(EMR) 시스템을 운영할 경우, 해당 정보의 법적 효력 보장을 위해 전자서명 기술의 적용이 필수적이다. 의료법 제23조는 의료인이 진료기록부 등을 전자문서로 작성하는 경우, 전자서명법에 따른 전자서명을 포함해야 함을 명시하고 있으며, 의료법 시행규칙 제16조는 전자의무기록의 생성·저장, 전자서명의 검증이 가능한 장비를 갖출 것을 요구한다. 이러한 법적 요건은 전자의무기록(EMR) 시스템이 정보보안 측면에서 위·변조 방지, 무결성 등을 충족해야 함을 의미하며, 병원은 이를 위해 전자서명 기술을 포함한 보안 체계를 반드시 구축해야 한다.

■ 기술적 대응 전략

애플리케이션 보안 점검과 취약점 진단

병원에서 운영하는 외부 연계 시스템은 공격자에게 침투 지점이 될 수 있다. 이러한 위협에 대비하려면 정기적인 보안 점검과 취약점 진단이 필요하며, 점검 이력은 체계적으로 기록·관리되어야 한다. 특히 외부 보안 전문가를 활용한 시나리오 기반 진단은 내부 보안 점검 체계를 보완하는 데 효과적이다. 이러한 활동은 단발성으로 끝나는 것이 아니라, 시스템 변경이나 서비스 확장 시마다 반복적으로 수행되어야 할 보안 검증 루틴으로 정착되어야 한다.

네트워크 구조 보안

병원은 진료망, 행정망, 의료기기망, 외부망 등 다양한 네트워크가 혼재된 복합 환경을 운영한다. 이러한 구조는 보안 위협이 확산되기 쉬운 특성을 가지므로, 병원 내 네트워크는 논리적 또는 물리적으로 분리되어야 하며, 필요 시 VLAN 및 전용망을 통해 업무별 통신 영역을 분리해야 한다.

또한 외부 접속 시에는 VPN 기반의 보안 통신을 기본으로 하고, 장비 인증, 접속 시간·위치 제한 등 조건 기반 접근 정책이 병행되어야 한다. 이는 무분별한 원격 접속을 차단하고, 시스템의 불필요한 노출을 최소화하기 위한 핵심 방어 수단이다.

시스템 보안 및 패치 관리

병원의 정보시스템은 운영체제(OS), 데이터베이스(DB), 전자의무기록(EMR), 의료영상저장전송시스템(PACS), 웹 서버 등으로 구성되어 있으며, 각 구성요소마다 독립적 보안 설정과 정기적인 패치 적용이 필요하다. 패치가 어려운 시스템은 네트워크 접근 차단, 애플리케이션 제어, 화이트리스트 기반 실행 제한 등 대체 기술을 활용해야 한다.

의료기기 보안

의료기기는 주로 폐쇄형 운영체제(OS) 또는 펌웨어 기반으로 운영되며, 제조사 또는 유지보수 업체를 통해서만 변경이 가능한 구조가 많다. 이로 인해 보안 패치나 점검이 원활하지 않고, 구형 시스템이 장기간 방치되는 경우가 많다. 보안 대책으로는 전용 네트워크 분리, 보안 게이트웨이 설치, 접근 제어, 로그 기록, 권한 분리 등이 있으며, 특히 무선 연결 또는 원격 진단 기능이 있는 장비는 반드시 암호화 통신 및 인증 체계가 적용되어야 한다.

계정 관리 및 접근 통제

병원 전산 시스템은 다양한 직군이 사용하는 환경이며, 교대근무나 공용 계정 사용으로 인해 사용자별 활동 추적이 어렵다. 이를 해결하기 위해 역할 기반 접근 통제(RBAC)를 적용하고, 관리자 계정에는 이중 인증(MFA)을 반드시 설정해야 한다.

또한 외부 협력사나 파견 인력에게 부여하는 권한은 시간, 경로, 명령어 등 조건에 따라 제한하고, 모든 접근 기록은 감사 로그로 남겨 정기적으로 점검되어야 한다.

로그 분석 및 이상행위 탐지

보안 로그는 단순히 보관하는 데 그치지 않고, 이상행위를 조기에 감지하는 데 활용되어야 한다. 시스템 로그와 사용자 행위 데이터를 기반으로 이상 접근을 실시간 탐지할 수 있는 모니터링 체계를 갖춰야 하며, 접속 이력, 계정 사용 내역, 시스템 변경 사항 등을 주기적으로 분석하는 체계를 마련해야 한다. 이러한 체계는 내부자 위협, 랜섬웨어 감염 등 보안 위협을 조기에 차단하는 효과적인 수단이 된다.

■ 관리적 대응 전략

보안 인력과 내부 역량

병원 정보보안의 핵심은 전담 인력의 안정적인 확보에 있다. 2024년 기준, 병원급 의료기관 중 다수는 5명 미만의 보안 인력으로 운영되고 있으며, 전체 보건의료기관의 평균 정보보호 전담 인력 수는 2.8명에 불과하다. 특히 중소병원의 경우, 상근 보안 인력을 두지 못해 보안 업무 전반을 외부 IT유지보수 업체에 위탁하는 경우가 일반적이다. 이러한 구조에서는 침해 사고 발생 시 즉각적인 대응이 어려울 뿐 아니라, 보안 운영의 지속 가능성도 낮아질 수 있다. 일정 규모 이상의 병원은 자체 보안 전담 인력을 확보하고, 네트워크·시스템·보안관제 등 분야를 분리해 기능별 역할을 수행할 수 있는 내부 구조를 갖추는 것이 바람직하다.

보안 교육과 인식 제고

병원은 다양한 직종이 공존하는 복합 조직이다. 단순한 교육 자료 배포만으로는 보안 인식이 충분히 전파되기 어려우며, 각 직무에 특화된 맞춤형 교육이 요구된다. 예를 들어, 의료진에게는 피싱 메일 탐지 훈련, 행정직에는 계정 관리 위험 교육, 장비 담당자에게는 의료기기 보안 절차 안내 등 업무 기반의 분리 교육이 필요하다. 이러한 교육은 연 1회 이상의 정기 교육과 모의 훈련을 병행하는 방식으로 구성되며, 지속적인 실천을 통해 병원 전반에 보안 문화를 정착시켜야 한다.

외주 및 공급망 보안

병원은 다양한 외부 업체와 연결된 구조를 갖고 있다. 협력사는 병원 시스템에 일정 수준 이상의 접근 권한을 보유하기 때문에, 해킹의 우회 경로로 악용될 수 있는 보안 취약점이 되기도 한다. 따라서 외주 계약 시에는 정보보호 관련 조항을 명시하고, 정기적인 보안 점검 및 준수 평가를 병행해야 한다. 또한 외부 접속은 시간, 권한, 세션 기록 등을 기반으로 통제하고, 이를 통해 외부 인력의 활동을 투명하게 관리할 수 있는 구조를 마련해야 한다.

대응 훈련과 사고 대응 체계 운영

2024년 기준, 전체 의료기관의 82.1%가 랜섬웨어나 해킹에 대비한 대응 훈련을 수행하고 있으며, 보건업의 대응 훈련 이행률은 96.2%로 가장 높은 수준을 기록했다. 이는 정기적 훈련이 보안 역량 강화에 실질적인 효과를 발휘함을 시사한다. 병원은 단순한 해킹 이메일 탐지 시나리오를 넘어서, 의료기기 오작동, 전자의무기록(EMR) 차단, 백업 시스템 이탈 등 복합 상황을 포함한 시나리오형 통합 훈련을 기획해야 한다. 이러한 훈련 결과는 보안 인식 향상은 물론, 전사적인 위기 대응 체계 구축의 출발점이 될 수 있다.

■ 병원정보시스템 보안가이드라인

2025 년 4 월, 국가정보원은 보건복지부, 한국인터넷진흥원(KISA)과 협력해 전국 의료기관을 대상으로 새로운 병원정보시스템 보안 가이드를 발표했다. 이는 최근 고도화된 병원 대상으로 지속되고 있는 사이버 공격, 특히 의료기기와 외부연계 시스템 등 새로운 형태의 IT 환경이 병원 내로 확산되면서, 기존 보안 체계만으로는 대응에 한계가 있다는 인식에서 비롯되었다.

병원 시스템이 마비될 경우, 단순한 서비스 중단을 넘어 환자의 생명에 영향을 줄 수 있다. 때문에, 다양한 위협 환경에 대응하기 위해 병원이 자율적으로 구축해야 할 보안 기준과 실무 지침이 마련되어야 있다.

가이드 구성

이번 가이드는 네트워크, 시스템 및 애플리케이션, 관리적 보안대책의 세 영역으로 구성되어 있다. 기술적 조치와 관리적 조치가 병원 환경에 맞게 조화를 이루도록 설계되었으며, 병원 규모에 따라 탄력적으로 적용할 수 있는 유연한 구조를 갖추고 있다.

항목	내용
네트워크 보안대책	병원 내부의 다양한 네트워크(진료망, 의료기기망, 외주망 등)를 구간별로 논리·물리적으로 분리하고, 외부 접속은 DMZ 구성을 통해 내부망을 보호하며, VPN·API 등 연계 통신은 암호화와 인증을 필수화 하는 등 병원 전반의 네트워크 환경에 대한 보호 체계를 명확히 규정함.
시스템 및 애플리케이션 보안대책	병원정보시스템을 구성하는 OS, DB, EMR, PACS, 웹 서버 등의 구성요소에 대해 보안 설정과 주기적 패치 관리를 요구하고, 의료기기 보안 항목을 별도로 마련하여 인터넷 차단, 접근 통제, 패치 확인을 강조하며, 원격진료 시스템과 클라우드 환경에서 발생할 수 있는 취약점 대응 방안도 포함함.
관리적 보안대책	병원 내 정보보호 책임자 지정, 조직 내 역할과 책임 분장, 계정·권한 관리 기준 수립, 외주 협력업체의 접근 통제, 보안 교육 시행, 보안 로그 보관 및 주기적 점검 절차 등 관리적인 통제를 통해 병원 전반의 보안 수준을 유지·운영할 수 있도록 요구함.

표2. 병원정보시스템 보안가이드 라인 개요

■ 맺음말

병원을 겨냥한 사이버 위협은 단순한 기술적 문제가 아니다. 이는 환자의 안전, 진료의 연속성, 그리고 공공의 신뢰와 직결되는 중대한 사안이다.

최근 국내외에서 발생한 병원 보안 사고들은 의료기관의 구조적 취약성과 운영상 허점을 정확히 파고들며, 단 한 번의 침입이 전체 시스템 마비로 이어지고, 때로는 환자의 생명에까지 영향을 미치는 상황을 초래하고 있다.

의료기관은 진료가 최우선인 환경 특성상, 시스템 점검이나 보안 강화 조치에 제약이 많은 것이 현실이다. 이 같은 제약은 공격자에게 '공격 성공률이 높은 표적'이라는 인식을 심어주며, 실제로 병원은 다른 산업에 비해 사고 대응이 늦고 피해 범위도 넓은 편이다.

의료 서비스는 앞으로 점점 더 디지털화되고, 환자정보의 흐름은 병원 내부를 넘어 연계 시스템으로 계속 확장될 것이다. 이런 환경에서 병원의 운영 안정성과 사회적 신뢰를 지키기 위해, 보안은 선택이 아닌 필수적 조건으로 인식되어야 한다.

Keep up with Ransomware

빠르게 진화하는 Vanhelsing 랜섬웨어

■ 개요

2025 년 3 월 랜섬웨어 피해 사례 수는 지난 2 월(1067 건)에 비해 약 28% 감소한 773 건을 기록했다. 3 월에 피해 사례가 감소한 이유는 Clop 그룹이 Cleo 의 파일 전송 솔루션 취약점을 악용하여 발생시킨 피해자를 2 월에 모두 공개했기 때문이다. 지난달에 비해 감소했지만 773 건은 평균적으로 높은 수치이며, 이러한 이유는 다수의 신규 랜섬웨어 그룹이 등장하고 활동하기 때문이다.

3 월에도 국내 랜섬웨어 위협이 확인됐다. NightSpire 그룹은 3 월에 새로 등장한 그룹으로 3 월에만 15 건의 피해자를 게시했으며, 그 중에는 국내 영상 콘텐츠 제작 업체가 포함되어 있었다. 공개한 샘플 데이터는 해당 업체에서 제작한 드라마의 대본 1 회분이었으며, 공개 예정일이 지났음에도 전체 데이터는 공개되지 않은 상태다. 4 월 기준으로 다크웹 유출 사이트는 비활성화 됐다.

BlackLock 을 운영하는 그룹이 Mamona 라는 신규 랜섬웨어 서비스를 발표했다. BlackLock 은 23 년 9 월에 LostTrust 로 처음 등장했으며, 이후 24 년 6 월에 ElDorado, 24 년 9 월에는 BlackLock 으로 리브랜딩한 그룹이다. 이들은 BlackLock 과 별개로 Mamona RIP 이라는 신규 그룹을 만들어 러시아 해킹 포럼에 홍보하기 시작했으나, Mamona 서버의 보안 설정이 미흡하여 Mamona 의 다크웹 유출 사이트가 해킹되고 BlackLock 의 유출 사이트는 비활성화되는 일이 발생했다. 현재는 BlackLock 의 유출 사이트만 접속이 가능한 상태이다.

RansomHub 랜섬웨어가 최근 SocGholish(FakeUpdates)라는 멀웨어 서비스 프레임워크를 통해서 배포된 정황이 확인됐다. SocGholish 는 2018 년에 등장했으며, 정상적인 웹사이트에 악성 스크립트를 주입한 뒤 사용자의 트래픽을 하이재킹하는 방식을 사용한다. 사용자가 해당 웹사이트를 방문하면 브라우저 업데이트 알람으로 위장한 가짜 페이지에 접속하게 되고, ZIP 형태로 압축된 SocGholish 스크립트 파일을 다운로드 하도록 유도한다. 사용자가 해당 스크립트 파일을 실행하게 되면 공격자는 데이터 탈취나 원격 명령 실행이 가능해지며, 이를 통해서 RansomHub 랜섬웨어를 배포한 것으로 확인됐다.

Akira 그룹이 최근 웹캠을 활용해 랜섬웨어 공격에 성공한 사례가 확인됐다. Akira 그룹은 자신들이 주로 사용하는 전략인 취약한 원격 접속 솔루션에 노출된 자격 증명을 이용하거나 무차별 대입을 통해서 시스템에 침입한 다음 랜섬웨어 페이로드 배포를 시도했다. 하지만 랜섬웨어 배포 행위가 EDR[1] 솔루션에 의해 차단되어 EDR 솔루션 우회를 위해 접근 가능한 시스템 중에서 취약한 웹캠을 식별했고, 웹캠에는 EDR 솔루션이 존재하지 않는다는 점을 이용해 원격 쉘 접근을 통해 랜섬웨어를 배포했다.

[1] EDR (Endpoint Detection and Response): 컴퓨터와 모바일, 서버 등 단말기에서 발생하는 악성 행위를 실시간으로 감지하고 분석 및 대응하여 피해 확산을 막는 솔루션

■ 랜섬웨어 뉴스

RansomHub 그룹, SocGholish 멀웨어 서비스를 이용한 랜섬웨어 배포

- SocGholish는 정상 웹사이트에 스크립트를 삽입하여 사용자의 트래픽을 가로채는 방식
- 감염된 웹사이트에 방문하면, 브라우저 업데이트로 위장한 SocGholish 스크립트를 다운 및 실행하도록 유도
- 파일 실행 시 공격자는 데이터 탈취나 원격 명령 실행이 가능

Akira 그룹, 웹캠을 활용해 랜섬웨어 공격 성공

- EDR 솔루션을 우회하기 위해 초기 침투 이후 네트워크에 연결된 웹캠을 통해서 랜섬웨어 페이로드 배포
- 웹캠으로 원격 쉘 접근이 가능하며, EDR 솔루션이 존재하지 않는 점을 이용

북한 위협 그룹 Moonstone Sleet, Qilin 랜섬웨어 배포 정황 발견

- Moonstone Sleet은 자체 제작 랜섬웨어인 FakePenny 랜섬웨어를 배포한 이력이 있는 그룹
- 25년 2월 말부터 Qilin 랜섬웨어를 배포하기 시작

Mamona 랜섬웨어 서비스, DragonForce 그룹에 의해 해킹

- Mamona는 BlackLock으로 알려진 그룹이 새로 공개한 랜섬웨어 서비스
- 미흡한 보안 설정으로 인해 관리자 페이지 및 패널이 노출
- Mamona의 다크웹 유출 사이트는 DragonForce에 의해 해킹당했으며, BlackLock DLS는 비활성화
- 해킹 이후 약 2주 뒤, BlackLock DLS 복구

신규 Vanhelsing 그룹 파트너 모집

- 러시아 해킹 포럼 Ramp 포럼에서 RaaS 파트너를 모집하는 글 게시
- 출시 3주만에 공격 대상 플랫폼 확대
- 약 한달간 총 8명의 피해자 게시


신규 NightSpire 그룹, 국내 영상 콘텐츠 제작 업체 공격
3월에만 15건의 피해자를 게시했으며, 그 중에는 국내 영상 콘텐츠 제작 업체 포함
샘플 데이터로 업체에서 제작한 드라마의 대본 1회분 공개
데이터 공개 기한이 지났음에도 전체 데이터 공개되지 않은 상태
Mimic 랜섬웨어 v.10 업데이트 공개
RAMP 포럼을 통해서 업데이트 내용 공개 및 파트너 모집 시작
랜섬웨어 서비스를 이용할 파트너 외에도, 초기 침투 전문가와 광고 담당자 등 함께 일할 구성원도 모집
신규 랜섬웨어 Oxthief, skira, Crazyhunter 그룹 등장
Oxtheif 그룹, 피해자 1건 게시 후 3월 중순부터 비활성화
Skira 그룹, 피해자 5건 게시 후 3월 말부터 비활성화
Crazyhunter 그룹, 피해자 10건 게시 후 4월부터 비활성화

그림 1. 랜섬웨어 동향

■ 랜섬웨어 위협

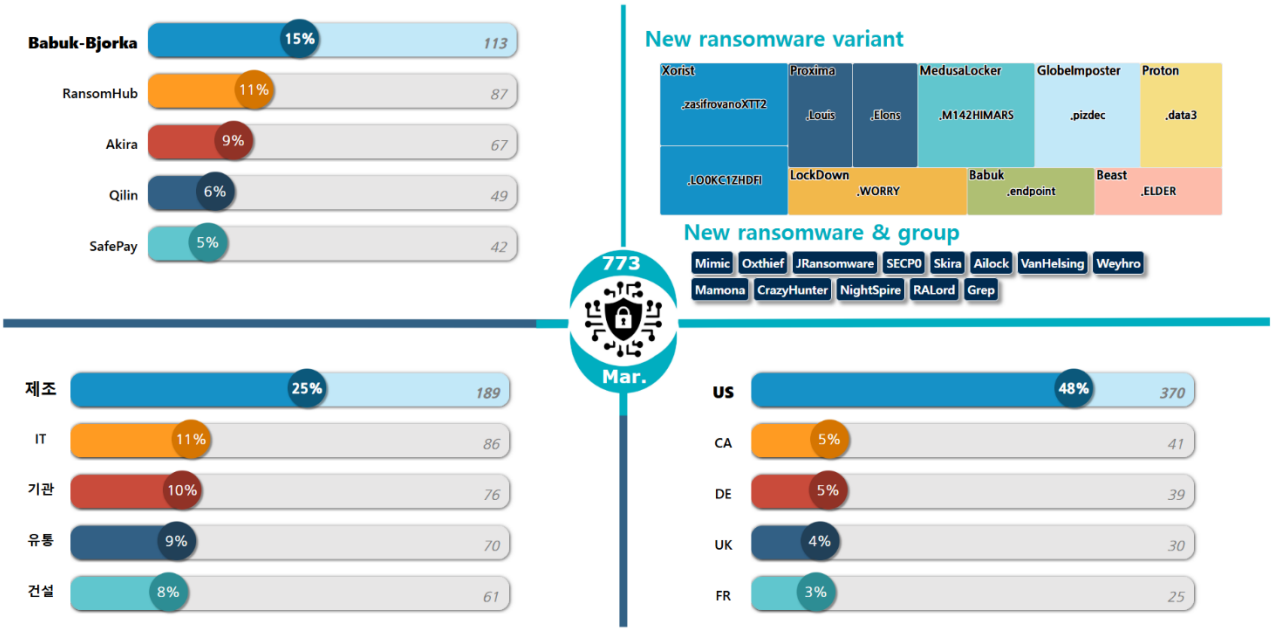


그림 2. 2025 년 3 월 랜섬웨어 위협 현황

새로운 위협

3 월에는 기존 랜섬웨어 그룹의 업데이트 소식도 확인됐으며 다수의 신규 랜섬웨어 그룹이 발견됐다. 총 6 개의 신규 랜섬웨어 그룹이 확인됐으며, 그 중 3 개의 그룹 Oxthief, Skira, CrazyHunter 는 4 월을 기준으로 현재 접근이 불가능한 상태이다. Oxtheif 그룹과 Skira 그룹의 경우 3 월 초 등장해 각각 1 건, 5 건의 피해자를 업로드했으나, 3 월부터 다크웹 유출 사이트가 비활성화 됐으며, CrazyHunter 의 경우 4 월에 다크웹 유출 사이트가 비활성화 됐다.

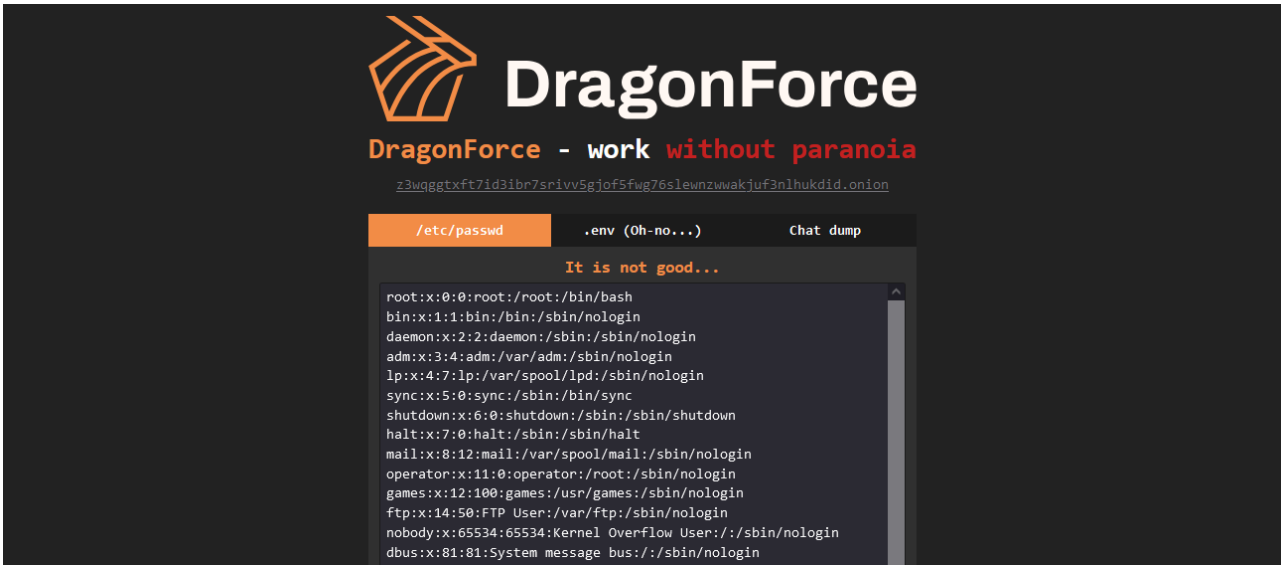


그림 3. DragonForce 에 의해 해킹된 Mamona 다크웹 유출 사이트

BlackLock 을 운영하는 그룹이 공개한 Mamona 라는 신규 랜섬웨어 서비스의 보안 설정이 미흡하여 Mamona 관리자 페이지는 물론 관리 패널이 노출됐다. 러시아 해킹 포럼에서 해당 이슈가 공유되어 포럼 유저들이 서비스에 무단으로 접근하거나 데이터를 조회할 수 있었으며, BlackLock 의 유출 사이트는 비활성화 되고 DragonForce 그룹이 Mamona 의 다크웹 유출 사이트를 변조하기도 했다. 약 2 주 뒤 BlackLock 은 비활성화된 다크웹 유출 사이트를 복구했으며, Mamona 는 더 이상 운영되지 않고 있다.

러시아 해킹 포럼에서 자신들의 랜섬웨어 서비스를 홍보하는 신규 그룹도 1 개 확인됐다. Vanhelsing 그룹은 3 월 초 러시아 해킹 포럼에서 자신들의 RaaS[2] 를 이용할 파트너를 모집하기 시작했으며, 출시한지 3 주만에 공격 대상 플랫폼을 추가했다. 또한 이들은 한 달간 8 명의 피해자를 게시했다.

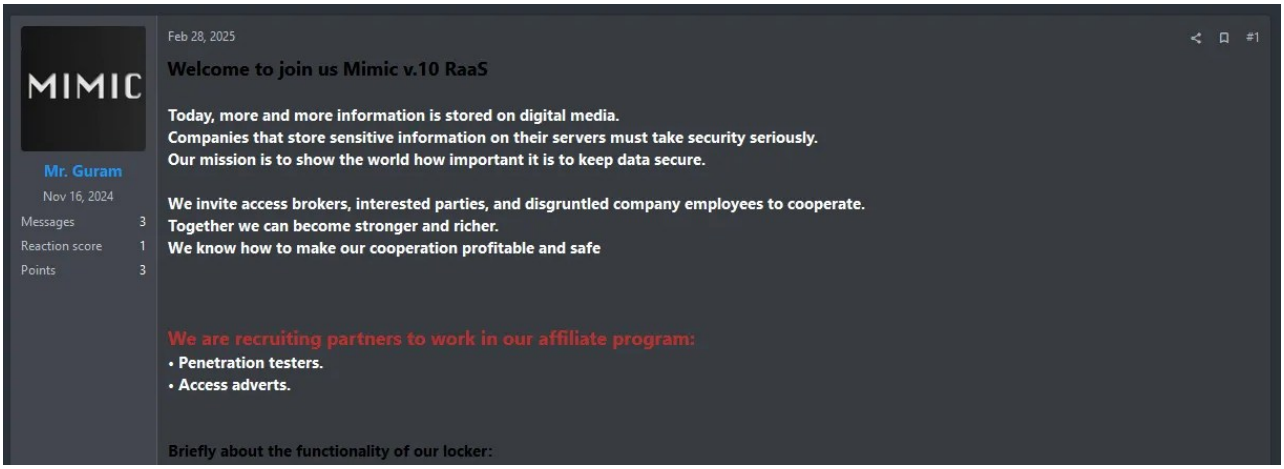


그림 4. Mimic v.10 홍보글

22 년 6 월부터 활동한 것으로 알려진 Mimic 랜섬웨어가 v.10 버전을 출시했으며, 러시아 해킹 포럼에서 함께 일할 초기 침투 전문가와 접근 권한 광고 담당자를 모집하고 있다. 이들이 제공하는 랜섬웨어는 Windows, ESXi[3] ,NAS[4] ,FreeBSD[5] 와 같은 다양한 운영체제를 지원한다. 랜섬웨어 제공뿐만 아니라 피해자에게 협박 전화를 하는 서비스나, 각종 작업에 필요한 소프트웨어도 제공한다.

랜섬웨어 그룹이지만, 별도의 데이터 유출 사이트는 운영하지 않고 몸값 협상을 위한 채팅 페이지만 운영하는 그룹이 2 개 발견됐다. JRansomware 와 Ailock 그룹이 이에 해당한다. 이들은 랜섬노트에 채팅 페이지 주소와 로그인에 필요한 세션 ID 를 제공해 각 피해자 별로 채팅 페이지를 관리하고 있다.

[2] RaaS (Ransomware-as-a-Service): 랜섬웨어를 서비스 형태로 제공해서 누구나 쉽게 랜섬웨어를 만들고 공격할 수 있도록 하는 비즈니스 모델

[3] ESXi: VMware 에서 개발한 호스트 컴퓨터에서 다수의 운영체제를 동시에 실행시킬 수 있는 UNIX 기반의 논리적 플랫폼

[4] NAS: 네트워크를 통해 접근할 수 있는 다수의 저장장치가 연결된 스토리지

[5] FreeBSD: 유닉스 계열의 오픈 소스 운영체제

Top5 랜섬웨어

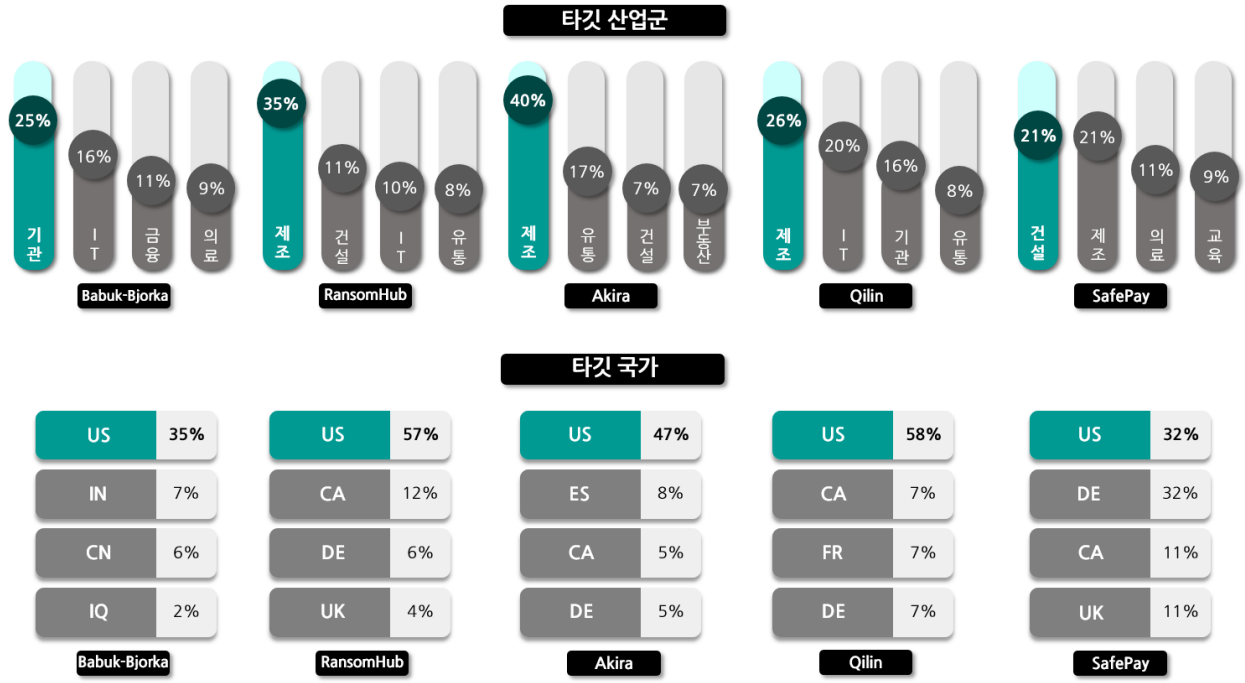


그림 5. 산업/국가별 주요 랜섬웨어 공격 현황

Babuk-Bjorka 그룹은 자신들이 Babuk2 라고 주장하는 그룹으로, 1 월부터 활동을 시작했다. 이들은 3 월에만 113 건의 피해자를 게시하며 가장 많은 활동을 보였는데, 다수의 피해자가 과거 RansomHub, Meow, Everest, Babuk, Funksec, LockBit 등 다른 그룹에 의해 데이터가 공개된 이력이 확인됐다. 따라서 이들이 실제로 해당 기업을 공격해 데이터를 탈취한 것인지, 아니면 이미 공개된 데이터를 재활용해 몸값을 요구하는 것인지 지켜볼 필요가 있다.

RansomHub 그룹은 말레이시아의 엔지니어링 서비스 기업 HexcoSys Group 을 공격해 계약서, 청사진, 소스 코드, 제품 개발 데이터가 포함된 336GB 데이터를 탈취했다. 또한 일본의 자동차 부품 제조 업체 Japan Rebuilt 를 공격해 생산 데이터와 재무 정보, 결제 세부 사항, 고객 기록이 포함된 200GB 의 데이터를 유출했다.

Akira 그룹은 3 월에 벨기에의 산업 기계 제조 업체 CS Plastics 를 공격해 감사 보고서, 재무 보고서와 같은 재무 데이터와 직원 및 고객의 개인정보가 포함된 민감한 데이터를 유출했다. 또한 최근에는 웹캠을 이용해서 EDR 솔루션의 탐지를 우회하는 등 새로운 공격 전략을 사용하는 모습도 확인됐다. Akira 그룹의 세부 공격 전략과 대응방안은 SK 쉴더스 KARA 랜섬웨어 동향 보고서 2024 4Q 에서 자세하게 확인할 수 있다.

Qilin 그룹이 최근 북한의 위협 그룹 Moonstone Sleet 과 연관이 있는 것으로 밝혀졌다. Moonstone Sleet 의 경우 이전에 자체 제작한 FakePenny 랜섬웨어를 배포한 이력이 있는 위협 그룹으로, 25 년 2 월 말부터 Qilin 랜섬웨어 페이로드를 배포한 정황이 확인됐다.

SafePay 그룹은 3 월 30 일에만 31 개의 피해자를 일괄적으로 게시했다. 영국의 하이 와이컴 소재의 중등학교 Sir Willian Ramsay School 는 이번 공격으로 183GB 에 달하는 데이터가 유출되었으며, 호주의 건설 계약 업체 Brighton Australia 는 재무 제표, 회계 기록, 인사 및 고객 파일이 포함된 160GB 데이터가 유출됐다.

■ 랜섬웨어 집중 포커스

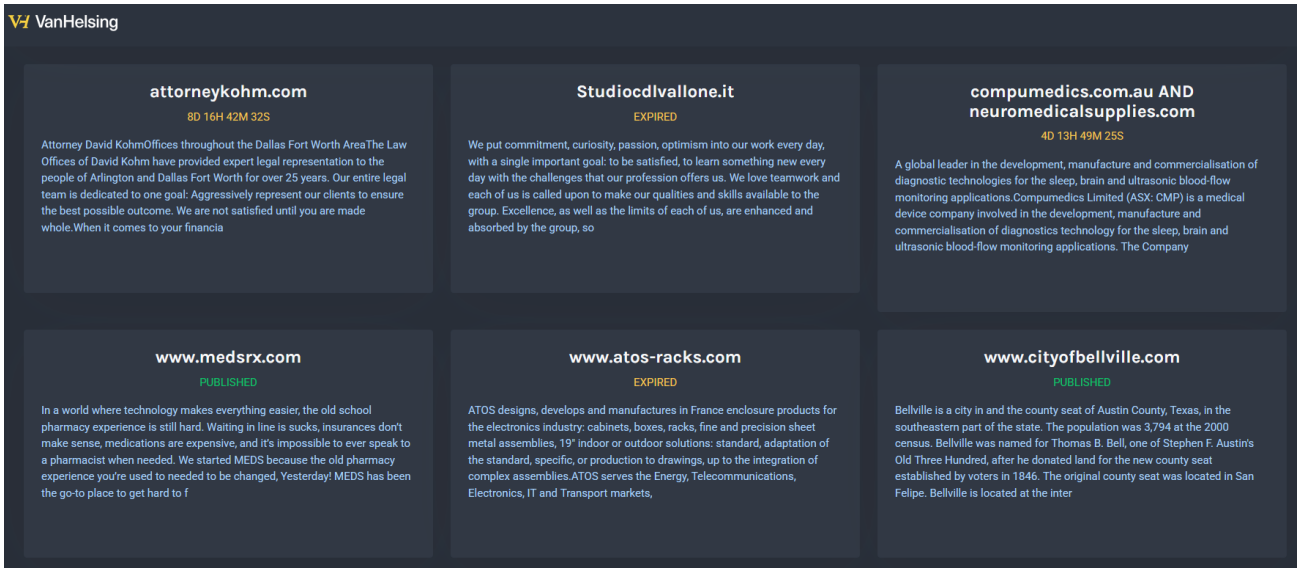


그림 6. Vanhesling 다크웹 유출 사이트

Vanhelsing 그룹은 3 월 7 일 등장한 랜섬웨어 그룹으로 러시아 해킹 포럼인 RAMP 포럼에서 계열사 모집을 시작했다. 이들은 어느정도 평판이 있는 사람은 별도의 가입 비용을 받지 않으며, 그 외에는 5,000 달러(한화 약 730 만원)에 해당하는 보증금을 지불해야 가입할 수 있는 형태이다. 포럼 홍보글에 따르면 이들은 CIS 국가에 대한 공격을 금지하고 있으며, 탈취한 몸값의 20%만 수수료로 요구하고 있다.

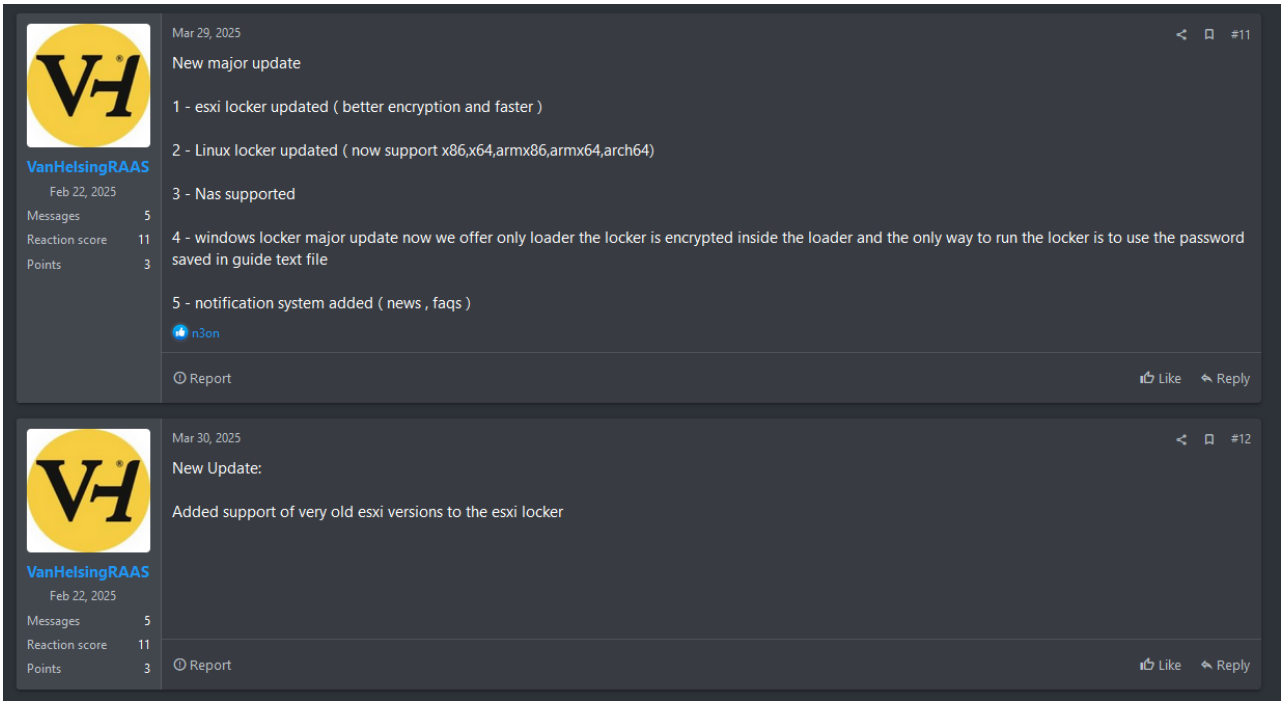


그림 7. Vanhelsing 기능 업데이트

출시 초기부터 이들은 Windows, Linux, BSD[6] , ARM[7] , ESXi 를 포함한 다양한 플랫폼을 대상으로 암호화를 지원한다고 소개했다. 3 월 말 운영자가 게시한 업데이트 내역에 따르면, Linux 와 ESXi 의 경우 더 다양한 버전과 아키텍처를 공격할 수 있으며, NAS 또한 공격 지원 대상에 추가됐다.

현재까지는 vanhelsing, vanlocker 라는 암호화 확장자를 사용하는 Windows 버전의 랜섬웨어만 확인됐다. 확인된 두 버전의 랜섬웨어는 약 5 일 간격으로 제작됐으며, 최신 버전에서는 기존에 없던 내부 네트워크 전파 기능이 추가됐다. 이 외에도 아직 기능이 구현되지는 않았지만, vCenter[8] 전파와 관련된 인자도 확인됐다. 랜섬웨어 기능이 빠른 속도로 업데이트되고 있을 뿐만 아니라, 랜섬웨어 홍보글에 따르면 향후 더 다양한 플랫폼을 대상으로 한 공격도 가능할 것으로 보인다. 이에 따라, 다가올 위협에 대비하기 위해 우선적으로 Windows 버전 랜섬웨어에 대한 분석 내용을 살펴보고자 한다.

[6] BSD: UC 버클리에서 개발한 유닉스 계열의 운영체제

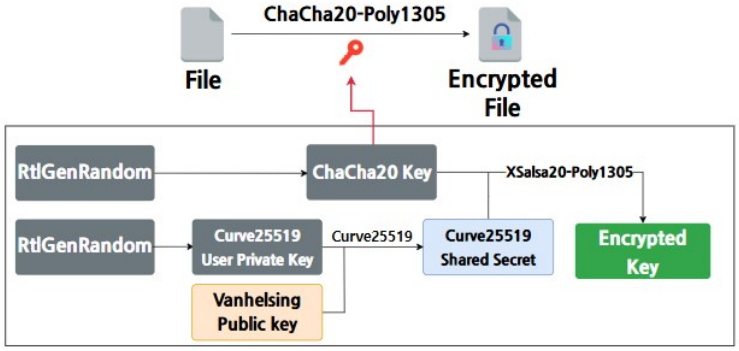
[7] ARM: 모바일 기기, IoT 장비 등에 주로 사용되는 저전력 고효율 프로세서 아키텍처

[8] vCenter: VMware 에서 개발한 중앙 집중식 관리형 유틸리티로 여러 ESXi 호스트 및 종속 요소를 관리하기 위해 사용

Vanhelsing Ransomware

암호화 키

ChaCha20-Poly1305 알고리즘으로 암호화를 하고, 해당 키를 Curve25519 공유 비밀로 보호



암호화 방식

1GB 이하: 전체 암호화
1GB 초과: 전체 크기의 30%만 암호화

특징

- 부분 암호화
- 백업 복사본 삭제
- 내부 전파
- 네트워크 공유 폴더 암호화
- ChaCha20/Xsalsa20 암호화
- 프로세스 우선순위 변경
- 바탕화면 및 아이콘 변경

랜섬노트

README.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
--= No news is a good news ! =--

Your network has been breached and all your  files Personal data, financial reports and important documents  has been stolen , encrypted

if you willing to continue your bussines and make more money and keep bussines secret safe you need to restore your files first, And to re
don't bother your self and wast your time or make it more harder on your bussines , we developed a locker that can't be decrypted using

making your self geek and trying to restore the files with third part decrypter this will leads to lose all your date ! and then the even you

to chat with us :

1 - Download tor browser https://www.torproject.org/download/
2 - go to one of these links above
        http://vanhelcbxqt4tqie6fuevfng2bsdtxgc7xslo2yo7nitaacdfrlpxnqd.onion
        http://vanhelqmjstkvlhrjwzgjzpq422iku6wlggiz5y5r3rmfdeiaj3ljaid.onion
        http://vanhelsokskrlaacilyfmtuqqa5haikubsjaokw47f3pt3uoivh6cgad.onion
        http://vanheltarnbfjhuvggbncniap56dscnzz5yf6yjmxqivqmb5r2gmllad.onion

3 - you will be asked for your ticket id to enter the chat this for you : TICKET ID [redacted]

usefull links :
#OUR TOR BLOG :
http://vanhelvuuo4k3xsiq626zkqvp6kobc2abry5wowxqysibmqs5yjh4uqd.onion
http://vanhelwmbf2bwzw7gmseg36qqm4ekc5uuhqbsew4eihzcahyq7sukzad.onion
http://vanhelxjo52qr2ixcmtjayqqrcodkuh36n7uq7q7xj23ggotyr3y72yd.onion
```

README.txt

변경 확장자

.vanhelsing, .vanlocker

제작 언어

C/C++

그림 8. Vanhelsing 랜섬웨어 개요

Vanhelsing 랜섬웨어 전략

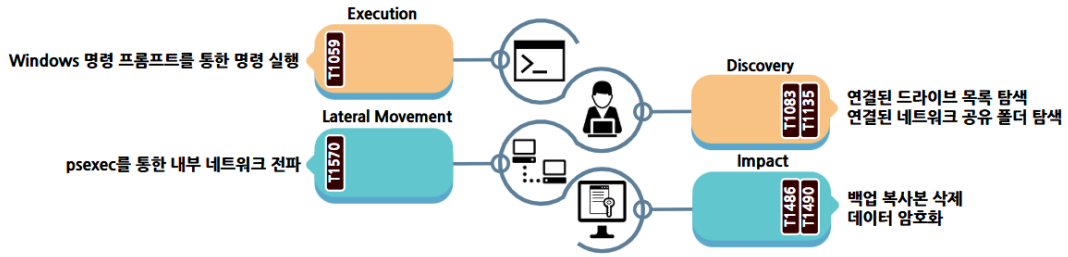


그림 9. Vanhelsing 랜섬웨어 공격 전략

Vanhelsing 랜섬웨어는 다양한 실행 인자를 사용해 암호화 대상이나 방식을 설정할 수 있으며, 바탕화면 변경이나 백업 복사본 삭제와 같은 기능의 사용 여부도 결정할 수 있다. 다만, 랜섬웨어에서 보여주는 도움말 메시지와 실제 인자가 다르며, 일부 인자는 확인만 하고 사용하지 않거나 기능이 구현되지 않은 경우도 있다. 실제 확인하는 인자와 기능은 아래 표와 같다.

인자	설명
-h	실행 인자 도움말 출력
-v	로그 출력
--skipshadow	백업 복사본 삭제 생략
--Driver <driver>	지정한 드라이버만 암호화
--Directory <directory>	지정한 폴더만 암호화
--File <file>	지정한 파일만 암호화
--Force	랜섬웨어 중복 실행 허용
--no-priority	랜섬웨어 우선순위 설정 생략
--no-wallpaper	바탕화면 변경 생략
--no-local	로컬 파일 암호화 생략
--no-mounted	고정된 로컬 드라이브만 암호화
--no-network	네트워크 공유 폴더 암호화 생략
--spread-smb	내부 네트워크 전파
--no-logs	로그 미출력
--no-admin	관리자 권한 여부 상관없이 실행
--Silent	모든 대상 파일 암호화 후 확장자 일괄 변환
--system	기능 미구현
--no-autostart	기능 미구현
--spread-vcenter	기능 미구현

표 1. Vanhelsing 랜섬웨어 실행 인자

실행 인자를 확인한 이후 암호화 중 오류 방지를 위해 몇 가지 작업을 수행한다. 바탕화면, 아이콘 변경과 네트워크 공유 폴더 접근 등 관리자 권한이 필요한 경우가 있으므로, 현재 관리자 권한으로 실행 중인지 확인한다. 만약 관리자 권한으로 실행 중이지 않으면 랜섬웨어를 종료하지만, "--no-admin" 인자를 사용하면 관리자 권한이 없더라도 랜섬웨어를 종료하지 않는다. 또한 랜섬웨어가 중복으로 실행되는 것을 방지하기 위해 "Golbal\\VanHelsing" 문자열로 뮤텍스 [9] 를 생성하고, 암호화 속도를 높이기 위해 랜섬웨어 프로세스의 우선순위를 가장 높은 우선순위로 설정한다. 이 두 기능은 각각 "--Force", "--no-priority" 인자로 비활성화가 가능하다.

또한 암호화된 파일을 사용자가 임의로 복구하지 못하도록 백업 복사본을 삭제한다. 마찬가지로 "--skipshadow" 인자를 사용하면 백업 복사본을 삭제하지 않는다. 백업 복사본을 삭제하기 위해 사용하는 명령어는 아래와 같다.

```
cmd.exe /c C:\\Windows\\System32\\wbem\\WMIC.exe shadowcopy where "ID='%s'" delete
```

표 2. 백업 복사본 삭제 명령어

"--spread-smb" 인자를 사용하면 내부 네트워크로 전파가 가능하다. 내부 네트워크에 연결된 PC 나 서버에서 랜섬웨어를 실행하기 위해 psexec[10] 를 사용한다. 해당 프로그램은 랜섬웨어에 함께 저장되어 있기 때문에, 해당 프로그램을 임시 폴더에 저장한 후 활용한다.

```
network_list = WSAStartup_sub_40CA90(pMemoryBlock: pMemoryBlock_1);
GetTempPathW(nBufferLength: 0x1F4u, lpBuffer: temp_path);
m_format_string_sub_40D890(Buffer: psexec_path, Format: L"%s\\psexec.exe", temp_path);
m_print_message_sub_4011D0(L"[*]\tpsexec_path : %s \n");
memset(buf: stream, value: 0, n0x20: 0xB0u);
m_wfsopen_sub_40BB30(buf: stream, Buffer: psexec_path, n48: 0x30, v43, Buffer: psexec_path);// open %TEMP%psexec.exe stream
m_write_stream_sub_40BD50(this: stream, &psexec_data, 0xAED90i64);
m_close_stream_sub_406660(buf: stream);
```

그림 10. psexec 저장

[9] 뮤텍스(Mutex): 하나의 자원에 여러 스레드 혹은 프로세스가 동시에 접근하지 못하도록 하는 동기화 매커니즘으로, 랜섬웨어에서는 흔히 중복 실행 방지를 위해 사용한다.

[10] psexec: Windows 시스템에서 프로세스를 원격으로 관리하고 실행할 수 있도록 하는 명령줄 도구

psexec 를 저장한 이후에는 현재 랜섬웨어가 실행중인 시스템의 IP 주소를 가져온 이후, 마지막 옥텟 [11] 에 해당하는 값을 1 부터 255 까지 변경하며 접근이 가능한 내부 네트워크가 존재하는지 확인한다. 접근 가능한 내부 네트워크 주소가 확인되면 해당 주소의 네트워크 폴더 중 쓰기 권한이 있는 폴더에 랜섬웨어를 "vanlocker.exe" 라는 이름으로 복사하며, psexec 를 활용해 내부 네트워크에 연결된 PC 나 서버에 랜섬웨어를 실행한다. 이 때 실행 인자로 "--no-mounted", "--no-network"를 사용하며, 사용하는 실행 명령어는 아래와 같다.

```
cmd.exe /c %TEMP%psexec.exe -accepteula \\${shared_folder} -c -f
${shared_folder}\vanlocker.exe -d --no-mounted --no-network < NUL
```

표 3. 내부 전파 명령어

내부 네트워크 전파 외에도 여러가지 실행 인자를 통해서 파일 암호화 대상 설정이 가능하며 크게 파일, 폴더, 로컬 드라이브, 네트워크 공유 폴더로 구분된다. "--File" 인자를 사용하면 지정한 1 개의 파일만 암호화하며, "--Directory" 인자를 사용하면 지정한 폴더와 그 하위 폴더를 암호화하고, "--Driver" 인자를 사용하면 지정한 드라이버 전체를 암호화한다. 만약 암호화와 관련된 실행 인자를 아무것도 사용하지 않으면 모든 드라이버와 네트워크 공유 폴더를 암호화하고, "--no-network" 인자를 사용해 네트워크 공유 폴더 암호화를 비활성화 하거나 "--no-local" 인자를 사용해 로컬 파일 암호화를 생략할 수 있으며, "--no-mounted" 인자를 사용해 고정된 로컬 드라이브만 암호화하는 것도 가능하다.

암호화 대상을 설정했으면, 각 디렉터리를 순회해 예외 항목에 해당하는지 확인한다. 우선적으로 대상 디렉터리가 예외 항목인지 확인하며, 디렉터리 확인이 끝났다면 각 디렉터리에 존재하는 파일이 예외 항목인지 확인한다. 확인하는 암호화 예외 대상은 아래 표와 같다.

폴더명	확장자 및 파일명
tmp, winnt, temp, thumb, $Recycle.Bin, $RECYCLE.BIN, System Volume Information, Boot, Windows, Trend Micro, program files, program files(x86), tor browser, windows, intel, all users, msocache, perflogs, default, microsoft	.vanlocker, .exe, .dll, .lnk, .sys, .msi, .bat, .bin, .com, .cmd, .386, .adv, .ani, .cab, .ico, .bod, .msstyles, .msu, .nomedia, .ps1, .rtp, .syss, .prf, .deskthemepack, .cur, .cpl, .diagcab, .diagcfg, .dll, .drv, .hlp, .pdb, .hta, .key, .lock, .ldf, .icns, .ics, .idx, .mod, .mpa, .msc, .msp, .nls, .rom, .scr, .shs,. spl, .theme, .themepack, .wpx, boot.ini, autorun.inf, bootfont.bin, bootsect.bak, desktop.ini, iconcache.db, ntldr, ntuser.dat, ntuser.dat.log, ntuser.ini, thumbs.db, GDIPFONTCACHEV1.DAT, d3d9caps.dat, LOGS.txt, .README.txt

표 4. 암호화 예외 대상

[11] 옥텟: 32 비트로 이루어진 IP 주소에서 8 비트씩 구분해 표현하는 단위

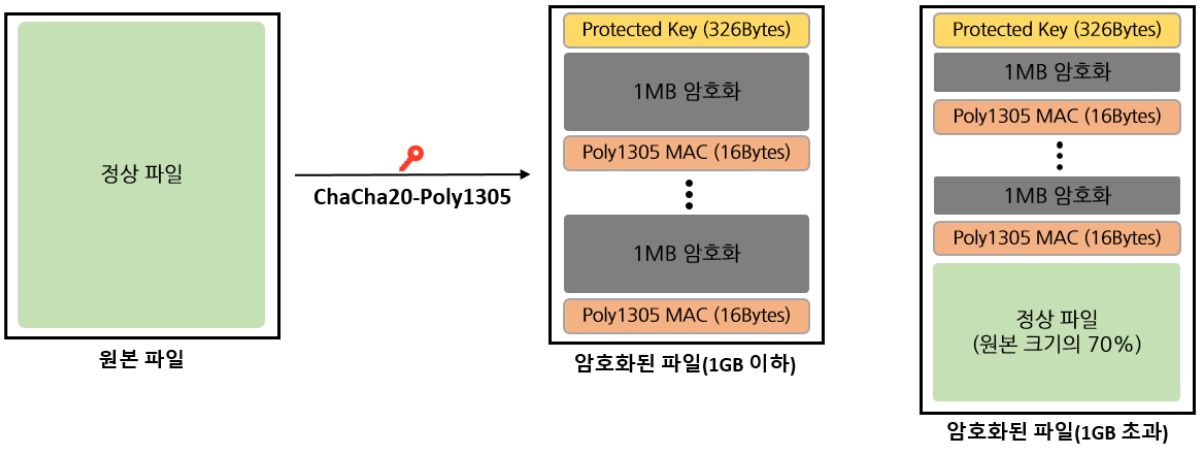
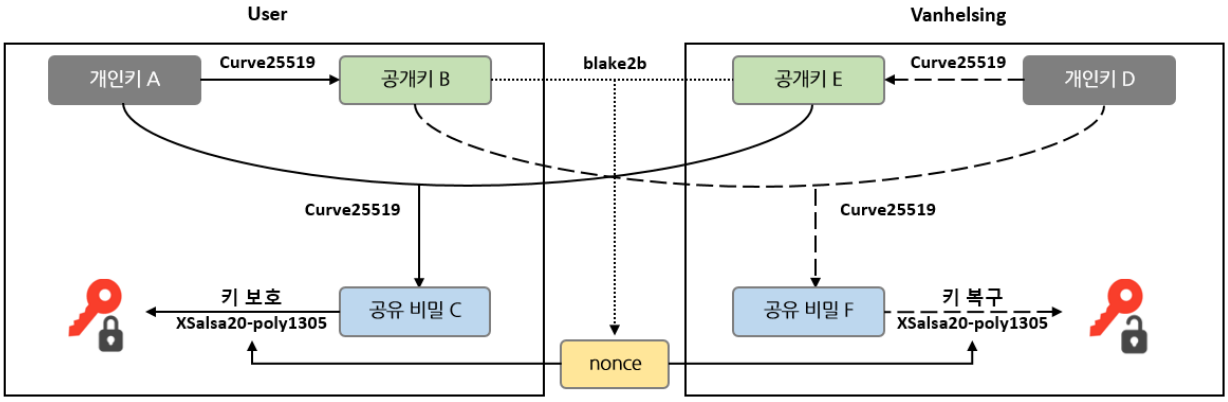


그림 11. 크기 별 파일 암호화 방식

파일 암호화는 1GB 이하의 파일은 전체를 암호화하고, 1GB 보다 큰 파일은 전체 크기의 30%만 암호화한다. 각 파일마다 랜덤한 32bytes 키와 12bytes nonce 를 만들고, ChaCha20-Poly1305 알고리즘을 사용해 파일을 암호화한다. 암호화는 1MB 단위로 진행하며, ChaCha20-Poly1305 의 특성상 데이터 무결성을 보장하기 위한 메시지 인증 코드(MAC)가 함께 생성되므로, 암호화된 파일에는 1MB 단위마다 16bytes 의 MAC 이 함께 저장된다.



공유 비밀 C = 공유 비밀 F

그림 12. 키 보호 방식

또한 암호화에 사용된 키를 보호해 파일의 맨 앞에 저장하는데, 이 때 Curve25519 를 통해서 생성한 공유 비밀로 암호화 키와 nonce 를 보호하고 키를 복구할 수 있는 사용자의 공개키를 함께 저장하는 방식을 사용한다. 파일 암호화 키와 별개로 각 파일마다 랜덤한 개인키를 하나 생성한 다음, 하드코딩된 Vanhelsing 의 공개키를 사용해 공유 비밀을 생성할 수 있다. 자신의 개인키와 상대방의 공개키로 만든 공유 비밀이 자신의 공개키와 상대방의 개인키로 만든 공유 비밀과 동일한 값을 가지는 Curve25519 의 특성을 이용한 것이다. 키 보호에는 XSalsa20-Poly1305 알고리즘을 사용하는데, 해당 알고리즘은 키 외에도 nonce 로 사용할 추가 데이터가 필요하다. nonce 는 사용자의 공개키와 Vanhelsing 의 공개키를 이어 붙인 다음 blake2b 알고리즘으로 12bytes 크기의 해시를 생성해 사용한다.

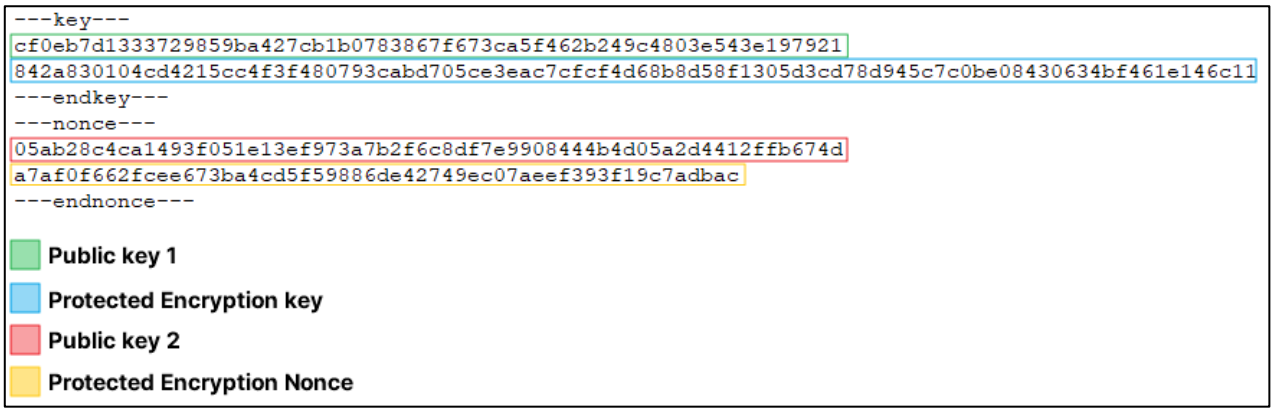


그림 13. 보호된 키 저장 방식

보호된 키는 복구를 위한 공개키와 함께 암호화된 파일의 맨 앞에 텍스트 형태로 저장된다. ---key---, ---nonce--- 와 같이 사용한 키와 nonce를 구분하며, 공개키와 보호된 키를 순차적으로 저장한다.

그림 14. 변경된 바탕화면

파일 암호화 이후에는 랜섬웨어에 저장된 이미지와 아이콘 파일로 바탕화면과 암호화된 파일의 아이콘을 변경한다. 배경화면은 “vhlocker.png”로 저장되며, 아이콘 이미지는 “vhlocker.ico”로 저장된다. --no-wallpaper” 인자를 사용하면 배경화면과 아이콘 이미지 변경은 생략한다.

Vanhelsing 랜섬웨어 대응방안

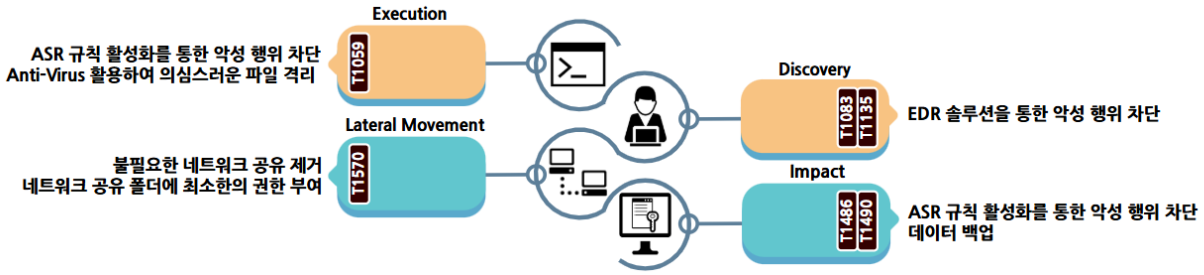


그림 15. Vanhelsing 랜섬웨어 대응방안

Vanhelsing 랜섬웨어는 Windows 명령 프롬프트를 활용해 백업 복사본 삭제와 내부 전파를 위한 명령어를 실행한다. 따라서 ASR[12] 규칙 활성화를 통해서 비정상적인 프로세스를 차단해 악성 행위를 막을 수 있다. 또한 랜섬웨어에 저장된 프로그램을 임시 폴더에 저장하거나 랜섬웨어 자체를 네트워크 공유 폴더에 복제하기 때문에 Anti-Virus 를 활용하여 의심스러운 파일을 격리할 수 있다.

내부 전파와 네트워크 공유 폴더 암호화를 위해서 현재 시스템의 내부 네트워크 대역을 탐색하고, 연결이 가능한 네트워크 주소와 공유 폴더에 접근을 시도한다. 뿐만 아니라 파일 암호화의 경우 모든 드라이브를 탐색한 뒤 실행 인자에 따라서 암호화할 드라이브를 구분하는데, EDR 솔루션을 통해 공격자의 악성 행위를 막을 수 있다.

또한 내부 전파는 랜섬웨어 쓰기 권한이 있는 공유 폴더에 복사한 뒤 실행하기 때문에, 별도의 접근 권한이 없다면 내부 전파 차단이 가능하다. 따라서 네트워크 공유 폴더 접근 권한을 최소한으로 부여하는 것도 하나의 방법이다. 그 외에 네트워크 공유 폴더와 같은 네트워크 서비스가 필요하지 않다면, 서비스 자체를 비활성화 하거나 SMB 포트(445)를 차단해 피해를 최소화해야 한다.

암호화된 파일을 사용자가 임의로 복구하는 것을 방지하기 위해 시스템에 존재하는 모든 백업 복사본을 삭제한 뒤 파일을 암호화한다. ASR 규칙 활성화를 통해서 백업 복사본을 삭제하는 프로세스와 파일을 암호화는 것을 차단할 수 있다. 뿐만 아니라 백업 복사본을 별도의 네트워크나 저장소에 소산 백업하여, 시스템이 암호화되더라도 복구할 수 있도록 조치해야 한다.

[12] ASR (Attack Surface Reduction): 공격자가 사용하는 특정 프로세스와 실행 가능한 프로세스를 차단하는 보호 기능

IoCs

Hash(SHA-256)
86d812544f8e250f1b52a4372aaab87565928d364471d115d669a8cc7ec50e17
99959c5141f62d4fbb60efdc05260b6e956651963d29c36845f435815062fd98

■ 참고 사이트

• BleepingComputer (https://www.bleepingcomputer.com/news/security/microsoft-north-korean-hackers-now-deploying-qilin-ransomware/)

• Cyber Daily (https://www.cyberdaily.au/security/11919-exclusive-contractor-brighton-australia-listed-on-safepay-s-ransomware-leak-site)

• S-rm (https://www.s-rminform.com/latest-thinking/camera-off-akira-deploys-ransomware-via-webcam)

• BleepingComputer (https://www.bleepingcomputer.com/news/security/ransomware-gang-encrypted-network-from-a-webcam-to-bypass-edr/)

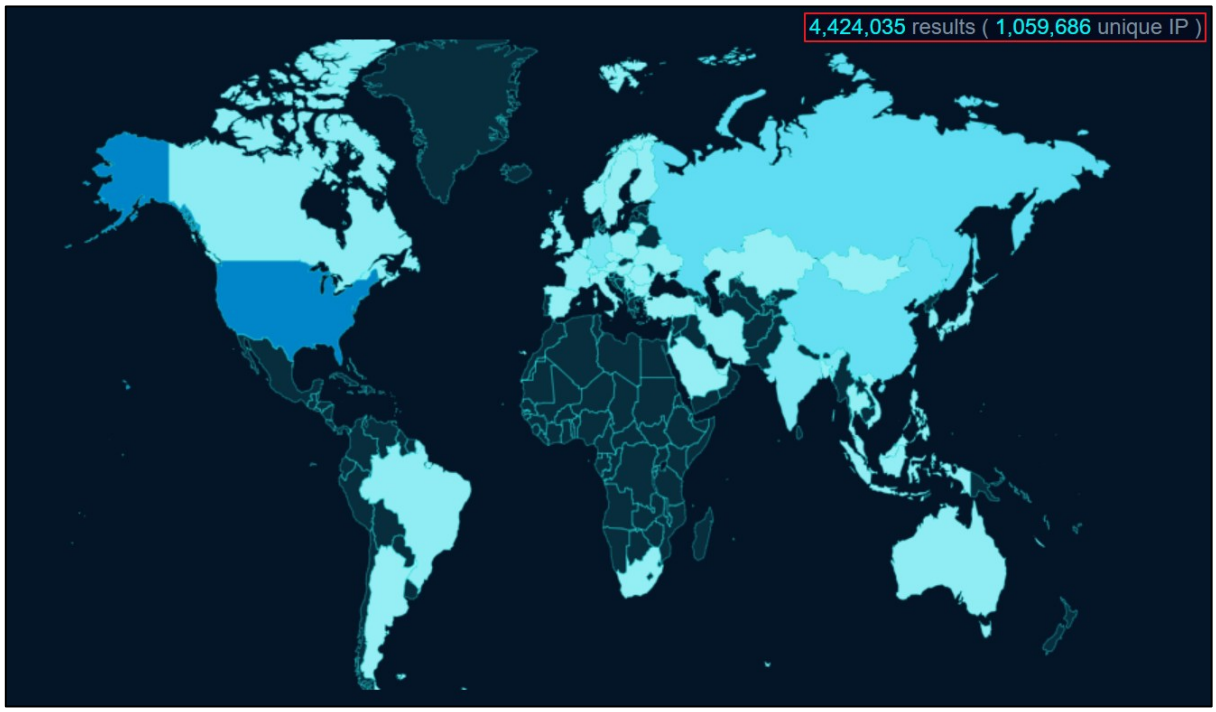
• Trend Micro (https://www.trendmicro.com/en_us/research/25/c/socgholishs-intrusion-techniques-facilitate-distribution-of-rans.html)

Research & Technique

JSONPath-Plus RCE 취약점(CVE-2025-1302)

■ 서론

Next.js 는 서버 사이드 렌더링(SSR)[1] 과 정적 웹 페이지 생성(SSG)[2] 을 지원하는 Node.js 기반의 오픈 소스 웹 프레임워크다. 전 세계적으로 널리 사용되는 자바스크립트 라이브러리인 React 는 공식 문서에서 권장하는 툴체인 중 하나로 Next.js 를 소개하고 있다.
OSINT 검색 엔진을 통해 인터넷에 공개된 Next.js 사용 현황을 조사한 결과, 2025 년 4 월 15 일 기준 미국, 러시아, 독일 등 수많은 국가의 총 442 만 개 사이트에서 Next.js 가 사용되고 있는 것으로 확인되었다.



출처: fofa.info

그림 1. Next.js 사용 통계

[1] 서버 사이드 렌더링 (Server Side Rendering): 서버에서 모든 데이터를 작성하여 클라이언트로 전송, 클라이언트는 해당 데이터를 해석해 웹사이트를 표시하는 웹 통신 방법

[2] 정적 웹 페이지 생성 (Static Site Generation): 페이지 HTML 을 미리 빌드 타임에 생성하여 각 요청마다 재사용하는 방법

2025 년 3 월 21 일, Next.js 의 middleware[3] 우회 취약점(CVE-2025-29927)이 공개됐다. 이 취약점은 특정 헤더 값을 이용해 middleware 가 이미 실행되었는지 여부를 확인하는 Next.js 의 내부 로직을 악용함으로써 발생한다. 공격자는 헤더 값을 조작해 middleware 를 우회할 수 있으며, 인증 절차가 middleware 를 통해 구현되어 있을 경우, 이를 우회해 제한된 페이지에 접근이 가능하다. Next.js 는 광범위하게 사용되는 웹 프레임워크인 만큼, 해당 취약점에 노출되어 있는지 반드시 점검해야 한다.

■ 공격 시나리오

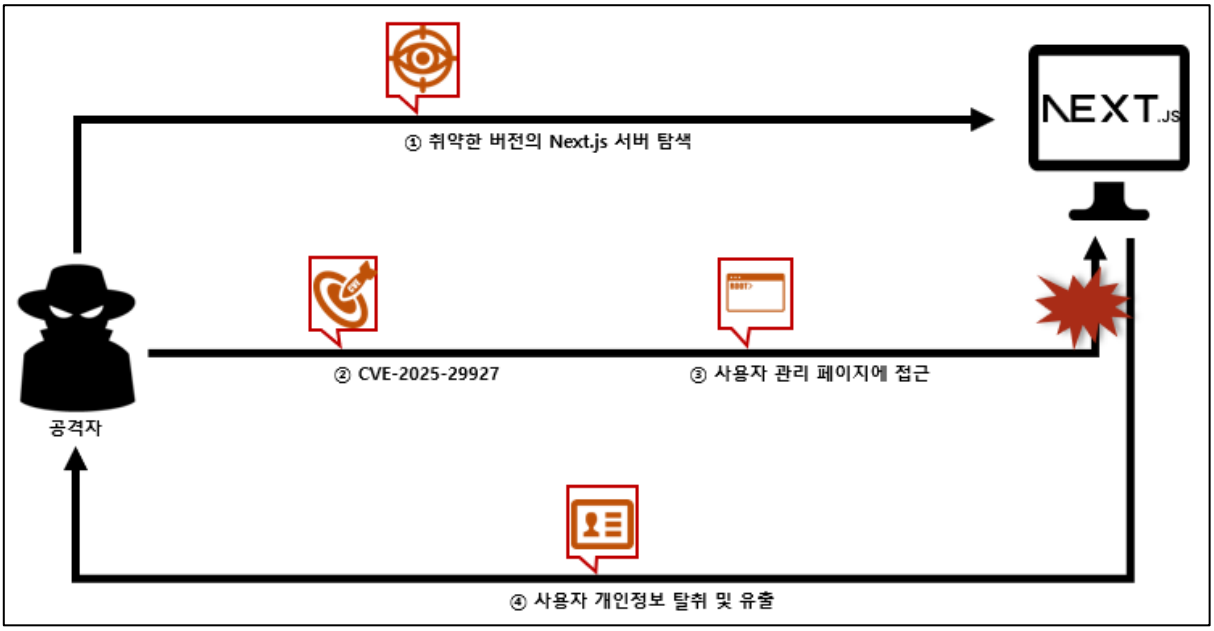


그림 2. CVE-2025-29927 공격 시나리오

① 공격자는 취약한 버전의 Next.js를 사용하는 서버를 탐색
② 이후 CVE-2025-29927 취약점을 이용하여 인증 우회 시도
③ 인증 우회에 성공한 공격자는 피해자의 사용자 관리 페이지에 접근
④ 사용자 관리 페이지에 접근한 공격자는 대량의 개인정보를 열람·탈취하여 외부로 유출

■ 영향받는 소프트웨어 버전

CVE-2025-29927 에 취약한 소프트웨어 버전은 다음과 같다.

S/W 구분	취약 버전
Next.js	v15.2.3 미만
	v14.2.25 미만
	v13.5.9 미만
	v12.3.5 미만
	v11 버전 전체

[3] 미들웨어 (middleware): 요청과 응답 주기에서 들어오는 요청이 도달하기 전에 처리하고 응답이 전송되기 전에 처리하는 개념

■ 테스트 환경 구성 정보

테스트 환경을 구축하여 CVE-2025-29927 의 동작 과정을 살펴본다.

이름	정보
피해자	Next.js v15.1.7 (192.168.0.3)
공격자	Kali Linux (192.168.216.133)

■ 취약점 테스트

Step 1. 환경 구성

피해자 PC 에 Next.js v15.1.7 환경을 설치하고, 취약점 재현을 위한 테스트 환경을 구성한다. 자세한 구성 방법은 아래 GitHub 저장소에서 확인 가능하다.

• URL: https://github.com/EQSTLab/CVE-2025-29927

```
# GitHub 저장소 클론
git clone https://github.com/EQSTLab/CVE-2025-29927

# 디렉토리 이동 후 도커 이미지 빌드 및 실행
cd next15
docker build -t nextjs .
docker run -p 3000:3000 --rm -it nextjs
```

웹 브라우저에서 피해자 주소에 접속하여 Next.js 서버가 정상 구동 중인지 확인한다.

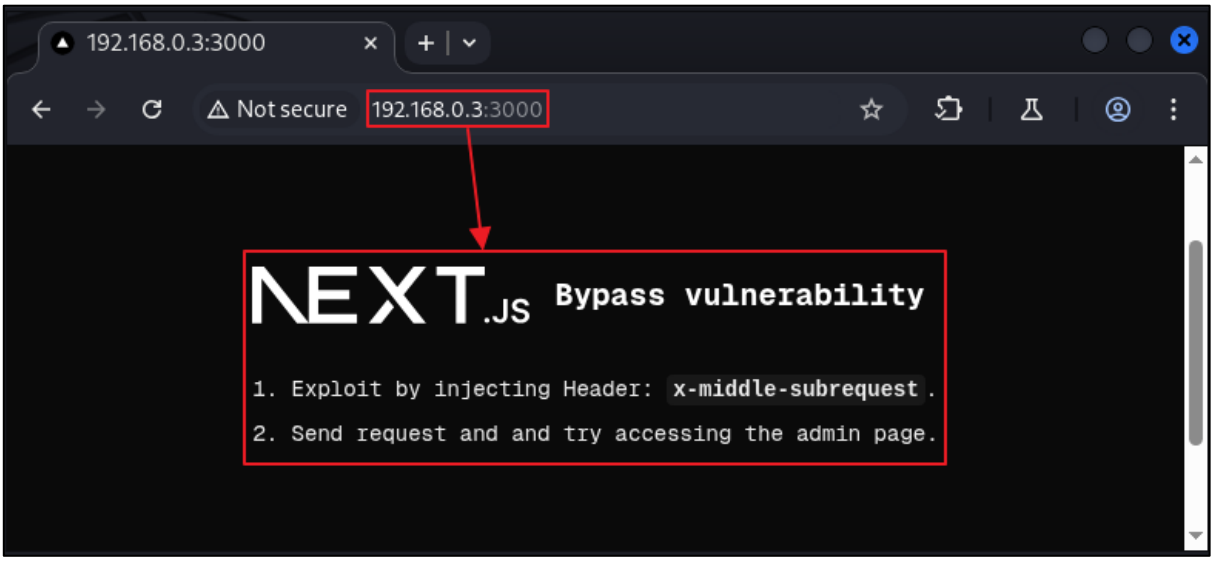


그림 3. 취약한 Next.js 환경 구축 확인

Step 2. 취약점 테스트

Next.js 서버는 /admin 엔드포인트에 대해 인증이 없는 접근을 차단하도록 구성되어 있다.

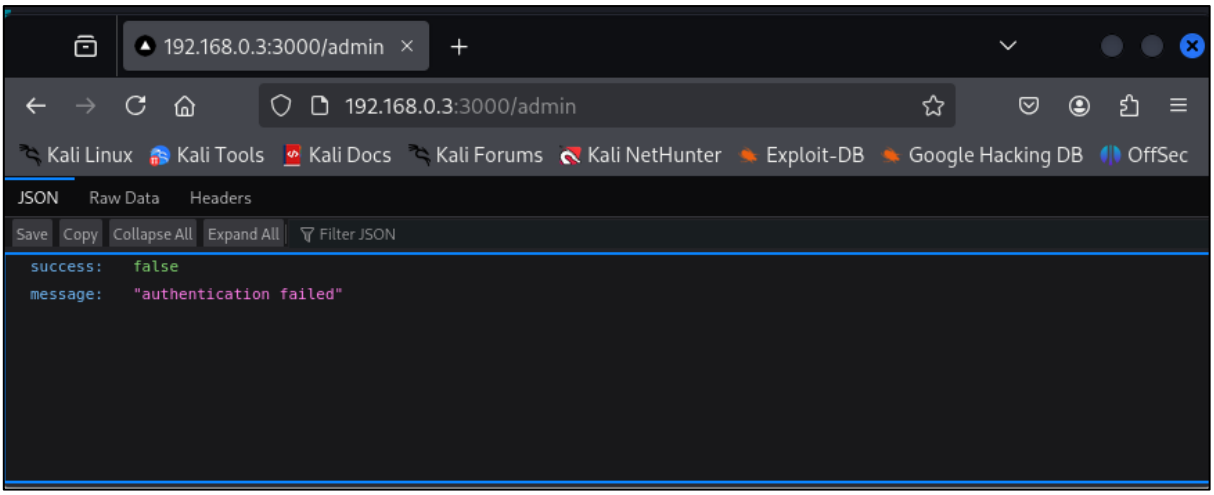


그림 4. /admin 엔드포인트 접근 차단

하지만, 다음과 같은 헤더를 요청에 추가하면 middleware 인증 절차가 우회된다.

```
x-middleware-subrequest: middleware: middleware: middleware: middleware: middleware
```

이 헤더가 포함된 요청을 전송하면, middleware 가 이미 실행된 것으로 처리되어 /admin 엔드포인트에 접근이 가능해진다.

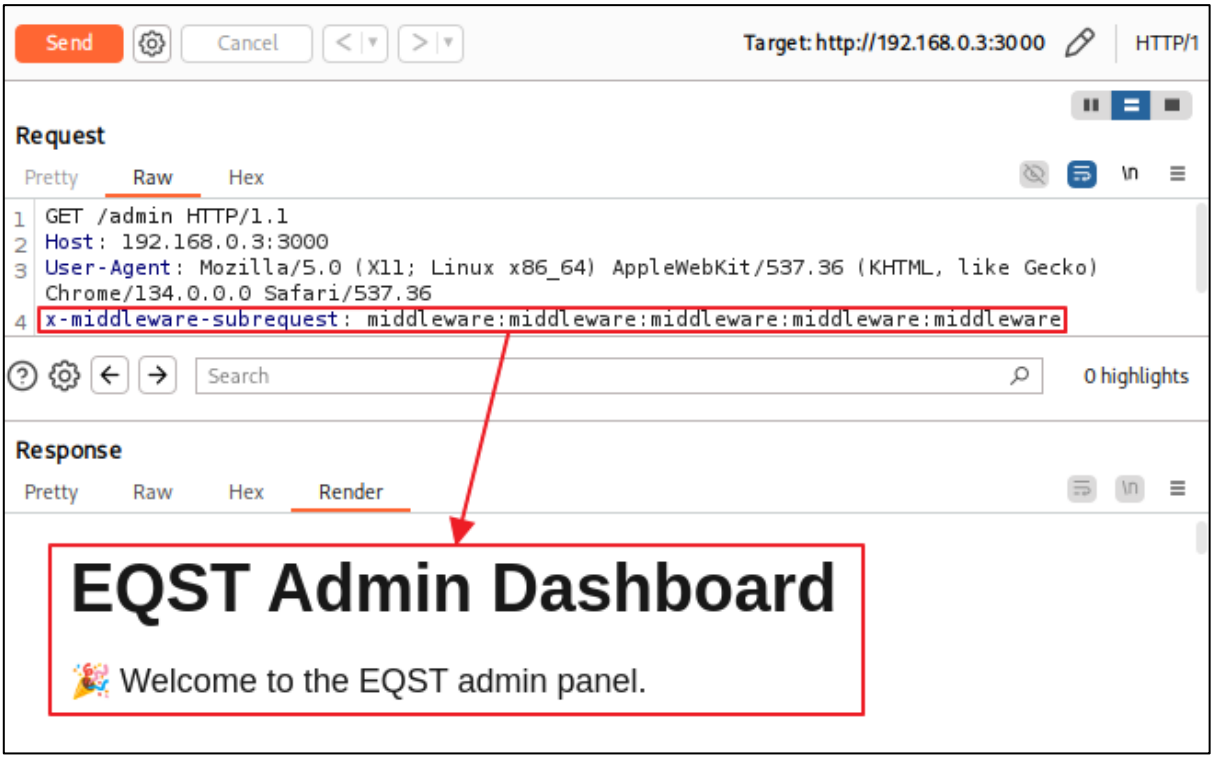


그림 5. middleware 인증 우회 확인

■ 취약점 상세 분석

취약점 상세 분석에서는 CVE-2025-29927 취약점의 발생 원리부터 인증 우회 과정까지 단계적으로 설명한다. Step 1 에서는 Next.js 의 middleware 개념과 버전별 특징을 소개한다. Step 2 에서는 각 버전의 middleware 구현에서 나타난 취약 지점과, 이를 활용한 인증 우회 기법을 설명한다.

Step 1. Next.js middleware

1) Next.js middleware 의 특징

Middleware 는 요청-응답 주기에서 클라이언트의 요청이 애플리케이션에 도달하기 전에 처리되는 중간 계층으로, 응답이 전송되기 전에도 추가적인 처리를 수행할 수 있다.

Next.js 에서는 이 기능을 통해 요청 전 검증, 헤더 재작성, 리디렉션 등을 구현할 수 있으며, middleware.ts 또는 middleware.js 파일을 통해 정의된다. Middleware 파일은 프로젝트 최상위 디렉토리에 app, pages 폴더와 나란히 위치하거나, src 디렉토리 내에 포함될 수 있다.

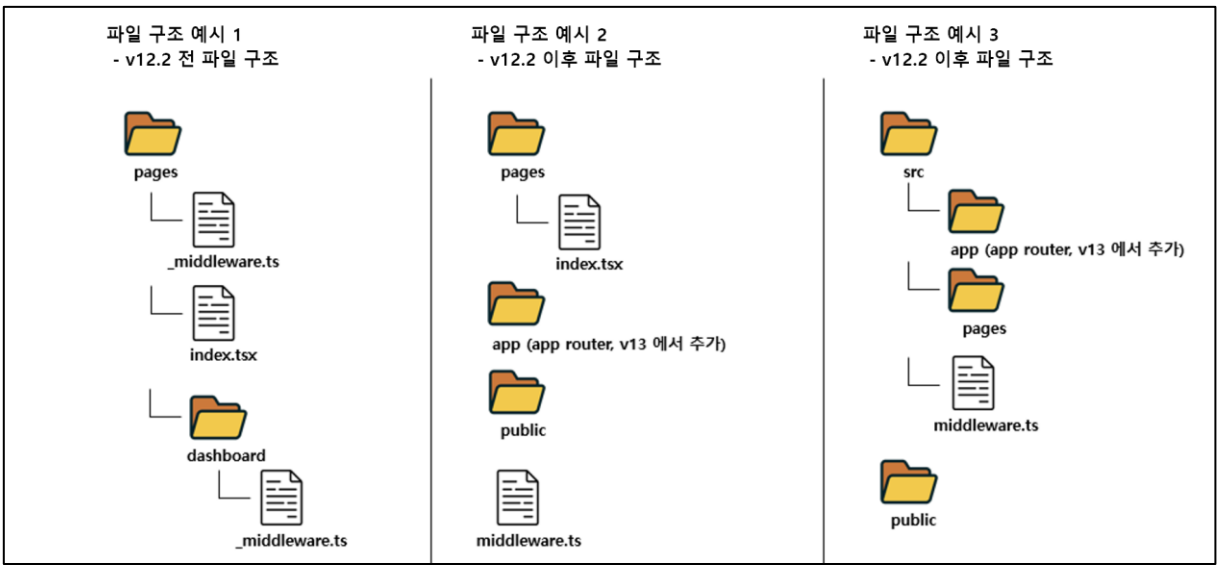


그림 6. 파일 구조별 middleware 위치 예시

2) Next.js 버전별 Middleware 특징

Next.js 는 v12.2 부터 middleware 를 업그레이드하고 패치하여, 이후 middleware 사용 방식에 중요한 변화가 있었다.

(1) v12.2 미만 버전

Next.js v12.2 이전 버전에서는 모든 디렉토리에서 middleware 를 정의할 수 있었다. 이 경우 서버는 요청이 들어오면 상위 디렉토리부터 하위 디렉토리 순으로 middleware 를 차례로 실행한다.

예를 들어, /pages 디렉토리에 middleware 가 정의되어 있으면, 그 index.tsx 와 about.tsx, teams.tsx 전체 요청에 해당 middleware 가 적용된다.

```
- package.json
- /pages
    └ _middleware.ts       # Will run on all routes under /pages
    └ index.tsx
    └ about.tsx
    └ teams.tsx
```

그림 7. /pages 디렉토리 내 _middleware 파일

또한 /pages, /pages/about, /pages/about/teams 각 디렉토리에 middleware 가 정의되어 있는 경우, 요청이 들어오면 디렉토리의 계층 구조에 따라 상위부터 하위 순서로 middleware 가 실행된다.

```
- package.json
- /pages
    └ _middleware.ts              # will run first
    └ index.tsx
    └ /about
        └ _middleware.ts          # will run second
        └ about.tsx
        └ /teams
            └ _middleware.ts      # will run third
            └ teams.tsx
```

그림 8. 중첩된 _middleware 파일 실행 순서

(2) v12.2 이후 버전

Next.js v12.2 부터는 middleware 파일명에서 더 이상 underscore(_)를 사용하지 않으며, 파일명은 middleware.ts 또는 middleware.js 로 변경되었다. 또한, 디렉토리별로 middleware 를 중첩 정의하는 방식이 지원되지 않고, 프로젝트 루트나 pages 디렉토리와 병렬 구조에 단 하나의 middleware 파일만 둘 수 있다.

이처럼 디렉토리 단위의 middleware 적용이 불가능해지면서, 특정 경로에 대해 middleware 를 적용하려면 matcher 설정을 활용해야 한다. 예를 들어, /admin 으로 시작하는 경로에 middleware 를 적용하려면 아래와 같이 matcher 에 해당 패턴을 지정해야 한다.

```
1   import { NextResponse } from 'next/server'
2   import type { NextRequest } from 'next/server'
3
4   export function middleware(request: NextRequest) {
5       return NextResponse.rewrite(new URL('/about-2',  request.url))
6   }
7
8   // Supports both a single string value or an array of matchers
9   export const config = {
10      matcher: ['/about/:path', '/dashboard/:path'],
11  }
```

그림 9. matcher 설정 예시 코드

또는 아래와 같이 middleware 내부에 조건문을 작성해, 요청 경로에 따라 동작을 분기할 수도 있다. 이 방식은 matcher 설정만으로는 제어하기 어려운 복잡한 조건이 필요한 경우에 유용하게 활용된다.

```
1   import { NextResponse } from 'next/server'
2   import type { NextRequest } from 'next/server'
3
4   export function middleware(request: NextRequest) {
5       if (request.nextUrl.pathname.startsWith('/about')) {
6           return NextResponse.rewrite(new URL('/about-2', request.url))
7       }
8
9       if (request.nextUrl.pathname.startsWith('/dashboard')) {
10          return NextResponse.rewrite(new URL('/dashboard/user', request.url))
11      }
12  }
```

그림 10. 조건 분기 예시 코드

Step 2. Next.js middleware 인증 구현 및 버전별 우회 방법

1) middleware 를 활용한 인증 구현

Next.js 에서 제공하는 인증 구현 방식 중에는 middleware 를 활용하는 방법도 포함되어 있다. 예를 들어, 아래는 사용자 권한에 따라 특정 경로로 리다이렉션하는 코드의 예시이다. 이러한 방식은 UI 요소를 숨기거나, 권한 및 역할에 따라 접근을 제어하는 등의 빠른 처리가 필요한 상황에서 유용하게 사용된다.

```
1   import { NextRequest, NextResponse } from 'next/server'
2
3   // Protected and Public Routes
4   const protectedRoutes = ['/dashboard']
5   const publicRoutes = ['/login', '/signup', '/']
6
7   export default async function middleware(req: NextRequest) {
8     // Some authentication logic
9     return NextResponse.next()
10  }
11
12  // Routes Middleware should not run on
13  export const config = {
14    matcher: ['/(?!api|_next/static|_next/image|.*\\.png$).*'],
15  }
```

그림 11. middleware 를 활용한 Next.js 인증 구현 예시

middleware 를 통해 인증을 구현한 후, 인증 없이 /admin 엔드포인트에 접근을 시도하면, 아래와 같이 요청이 차단되는 것을 확인할 수 있다.

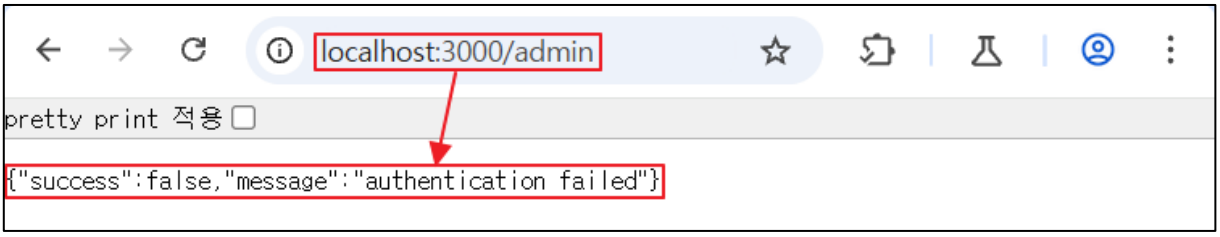


그림 12. x-middleware-request 검증 과정

2) v12.2 미만 버전

Next.js 는 내부적으로 x-middleware-subrequest 헤더를 활용하여 해당 요청에 대해 middleware 가 이미 실행되었는지를 판단한다. 이 로직은 v12.0.1 기준, /packages/next/server/next-server.ts 파일 내에서 확인할 수 있다. 코드의 주요 흐름은 다음과 같다.

```
630   const subreq = params.request.headers[`x-middleware-subrequest`]
631   const subrequests = typeof subreq === 'string' ? subreq.split(':') : []
632   const allHeaders = new Headers()
633   let result: FetchEventResult | null = null
634
...
650
651     if (subrequests.includes(middlewareInfo.name)) {
652       result = {
653         response: NextResponse.next(),
654         waitUntil: Promise.resolve(),
655       }
656       continue
657     }
```

그림 13. x-middleware-request 검증 과정

① x-middleware-subrequest 헤더를 통해 전달된 값을 colon(:) 기준으로 나눠서 배열로 구성한다.
② 구성된 배열에 middlewareInfo.name 변수의 값이 있는지 확인한다.
③ 만약 해당 값이 존재한다면 NextResponse.next() 함수를 통해 middleware 를 무시하고, 다음 처리 단계로 넘어가 요청을 보낸 경로의 응답을 받는다.

이때, middlewareInfo.name 변수의 값은 /packages/next/server/next-server.ts 파일 내 getMiddlewareInfo 함수를 통해 로드된다. 이 함수는 요청 경로에 대응하는 middleware 정보를 가져오는 역할을 하며, 해당 정보를 바탕으로 x-middleware-subrequest 헤더를 검사하여 mid-dleware 실행 여부를 판단하게 된다.

```
698       const middlewareInfo = getMiddlewareInfo({
699         dev: this.renderOpts.dev,
700         distDir: this.distDir,
701         page: middleware.page,
702         serverless: this._isLikeServerless,
703       })
```

그림 14. getMiddlewareInfo 로 불러오는 middlewareInfo

getMiddlewareInfo 함수는 /packages/next/server/require.ts 파일에 구현되어 있으며, 지정된 경로에 해당하는 middleware 정보를 객체 형태로 반환한다.

```
83  export function getMiddlewareInfo(params: {
84    dev?: boolean
85    distDir: string
86    page: string
87    serverless: boolean
88  }): { name: string; paths: string[] } {
89    const serverBuildPath = join(
90      params.distDir,
91      params.serverless && !params.dev ? SERVERLESS_DIRECTORY : SERVER_DIRECTORY
92    )
93
94    const middlewareManifest: MiddlewareManifest = require(join(
95      serverBuildPath,
96      MIDDLEWARE_MANIFEST
97    ))
...
```

그림 15. getMiddlewareInfo 내 MIDDLEWARE_MANIFEST 호출

이 로직은 Next.js 가 빌드된 후 생성되는 .next 디렉토리 내의 .next/server/middleware-manifest.json 파일을 참조하여 middleware 정보를 불러온다.

```
11    "middleware": {
12      "/": {
13        "env": [],
14        "wasm": [],
15        "files": [
16          "server/middleware-runtime.js",
17          "server/pages/_middleware.js"
18        ],
19        "name": "pages/_middleware",
20        "page": "/",
21        "regexp": "^/(?!_next).*$"
22      }
23    },
24    "version": 1
25  }
```

그림 16. middleware-manifest.json 내 middleware name 정보

middleware-manifest.json 파일의 name 키는 각 middleware 의 위치 경로를 나타낸다. 이에 따라 middlewareInfo.name 값은 최종적으로 pages/_middleware가 된다. 이는 디버깅 도구를 통해 실행 중인 값을 확인했을 때에도 동일하게 확인된다.

```
RUN AND DEBUG
VARIABLES
Block: runMiddleware
middlewareInfo = {name: 'pages/_middleware', paths: Array(2), env: Array(0), wasm: Array(0)}
  env = (0) []
  name = 'pages/_middleware'
```

그림 17. 디버거를 통해 확인한 middlewareInfo.name 의 값

즉, x-middleware-subrequest 헤더 값이 pages/_middleware 로 설정되면, 해당 요청은 이미 middleware 를 통과한 것으로 처리된다. 이로 인해 middleware 가 실행되지 않으며, 내부에 구현된 인증 절차 역시 우회할 수 있어 취약점이 발생하게 된다.

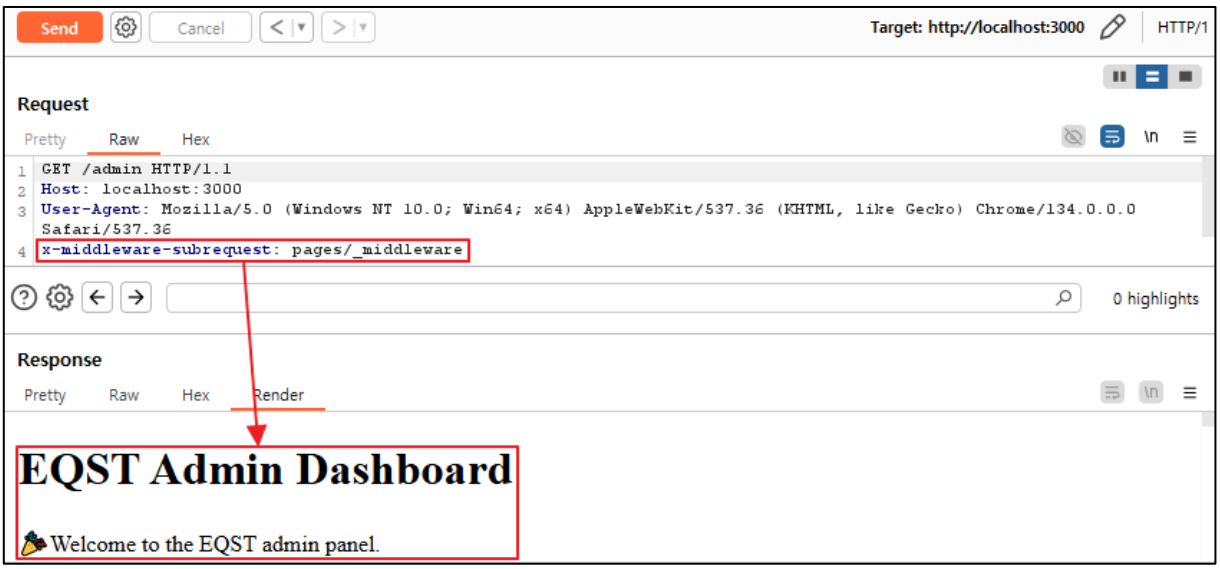


그림 18. middleware 인증 과정 우회 확인

앞서 Step 1 에서 설명했듯, v12.2 이전 버전에서는 디렉토리별로 middleware 를 개별 정의할 수 있다. 이 경우 pages 경로 아래에 정의된 각 middleware 경로를 x-middleware-subrequest 헤더 값에 입력하면 인증 절차를 우회할 수 있다. 그림 8 예시의 경우 pages/about/_middleware 또는 pages/about/teams/_middleware 를 헤더에 지정하면, 해당 middleware 를 우회하는 것이 가능하다.

3) v12.2 이상 v14.2 미만 버전

x-middleware-subrequest 에 대한 로직은 변경되지 않았기 때문에, 이전과 마찬가지로 해당 헤더에 middleware 경로를 설정하면 인증을 우회할 수 있다. v12.2 부터는 middleware 파일명이 middleware.ts 또는 middleware.js 로 변경되었고, 위치도 pages 디렉토리 내부가 아닌 그와 나란한 상위 경로로 통일되었다. 이에 따라 middleware 경로는 기존의 pages/_middleware 가 아닌, 단순히 middleware 로 바뀌었다.

이와 같은 변경 사항은 .next/server/middleware-manifest.json 파일에서 확인할 수 있으며, 이 파일을 통해 middleware 의 경로가 pages/_middleware 에서 middleware 로 변경된 것을 알 수 있다.

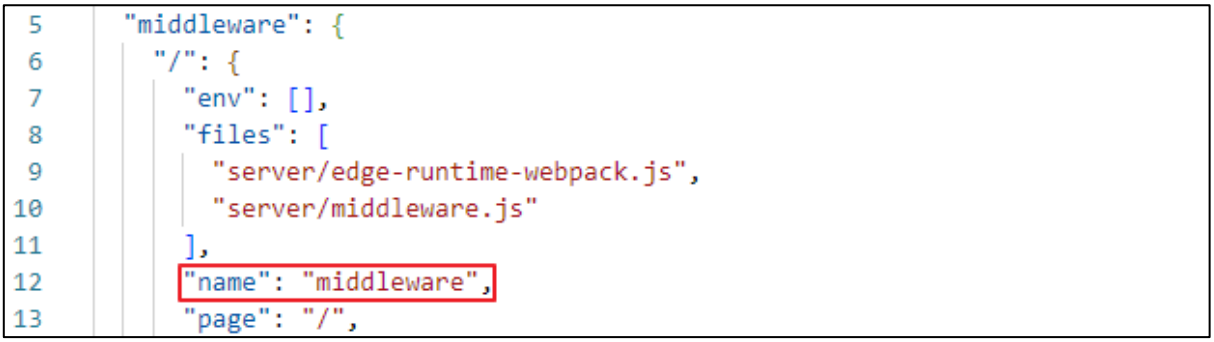


그림 19. middleware-manifest.json 내 middleware name 정보

위의 Next.js v12.2 사례와 마찬가지로, 디버거를 통해 실행 중인 값을 확인하면 해당 값이 middleware 임을 확인할 수 있다.

그림 20. 디버거를 통해 확인한 middlewareInfo.name 의 값

마찬가지로, x-middleware-subrequest 헤더의 값을 middleware 로 설정하면 인증 절차를 우회할 수 있다.

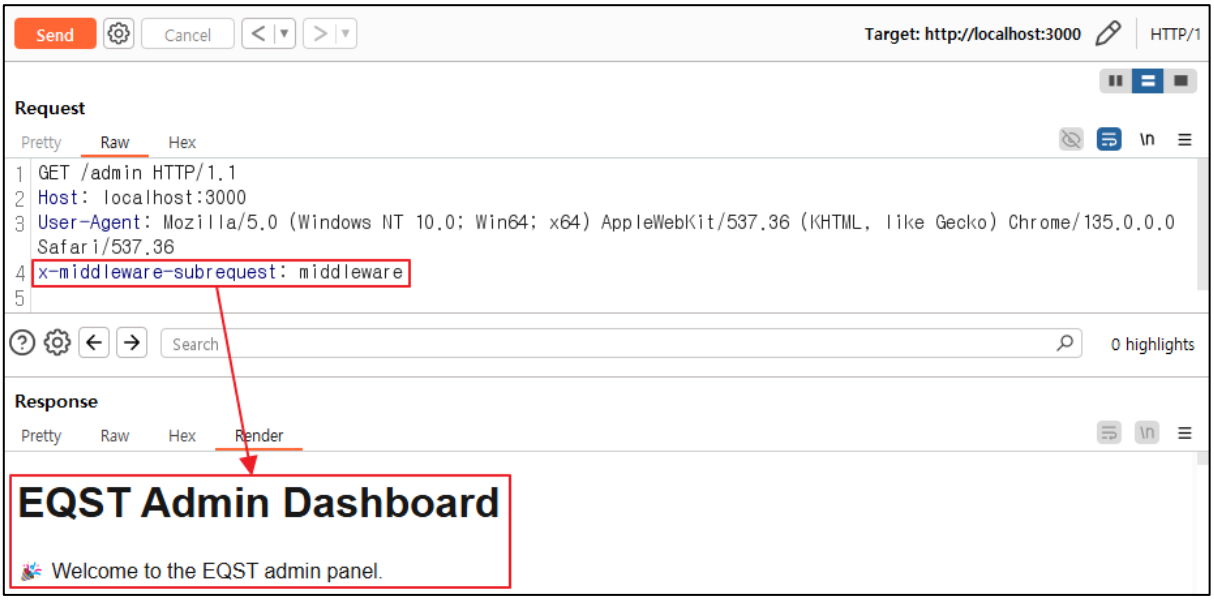


그림 21. middleware 인증 과정 우회 확인

4) v14.2 이후 버전

Next.js v14.2 부터는 middleware 의 x-middleware-subrequest 헤더의 값에 포함된 middleware 경로가 colon(:)을 기준으로 일정 횟수 이상 반복되면, 해당 요청이 middleware 를 우회하도록 코드가 변경되었다.

이는 /packages/next/src/server/web/sandbox/sandbox.ts 코드를 통해 확인할 수 있다.

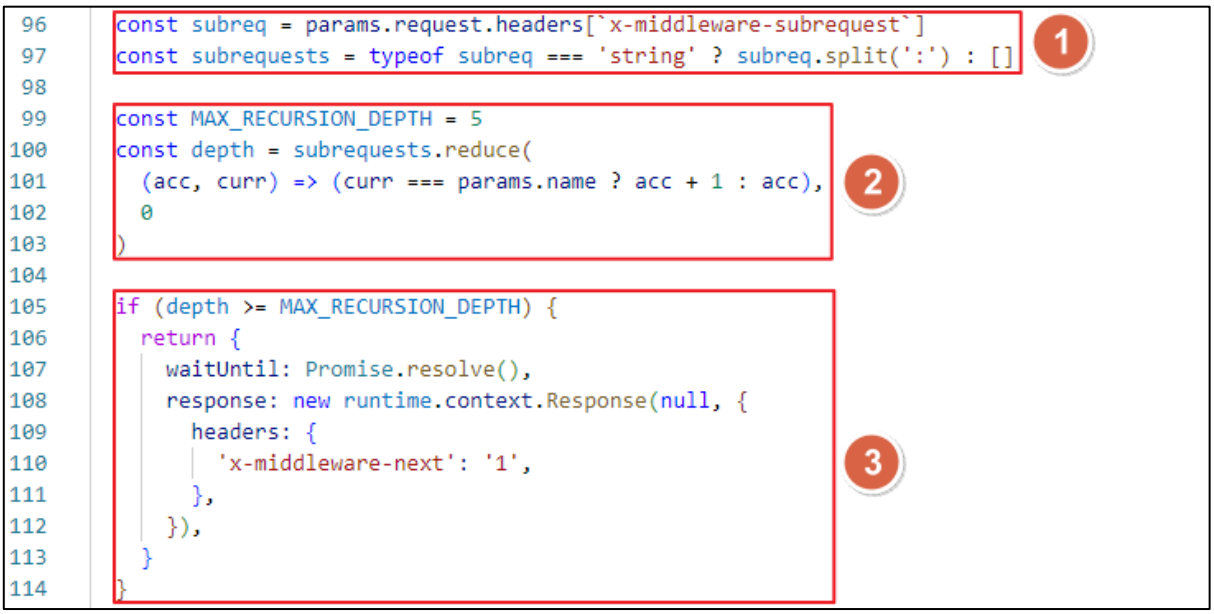


그림 22. 변경된 x-middleware-request 검증 과정

① x-middleware-subrequest 헤더를 통해 전달된 값을 colon(:) 기준으로 나눠서 배열로 구성한다.

② 구성된 배열에 params.name 변수의 값이 몇 개가 있는지 확인한다.

③ 만약, params.name 변수의 값이 5번 이상 반복이 된다면, middleware를 건너뛰고 요청을 보낸 경로의 응답을 받는다.

여기서 params.name 은 앞서 언급한 middleware.name 과 동일한 값으로, 이는 .next/server/middleware-manifest.json 파일과 디버깅을 통해 확인할 수 있다.

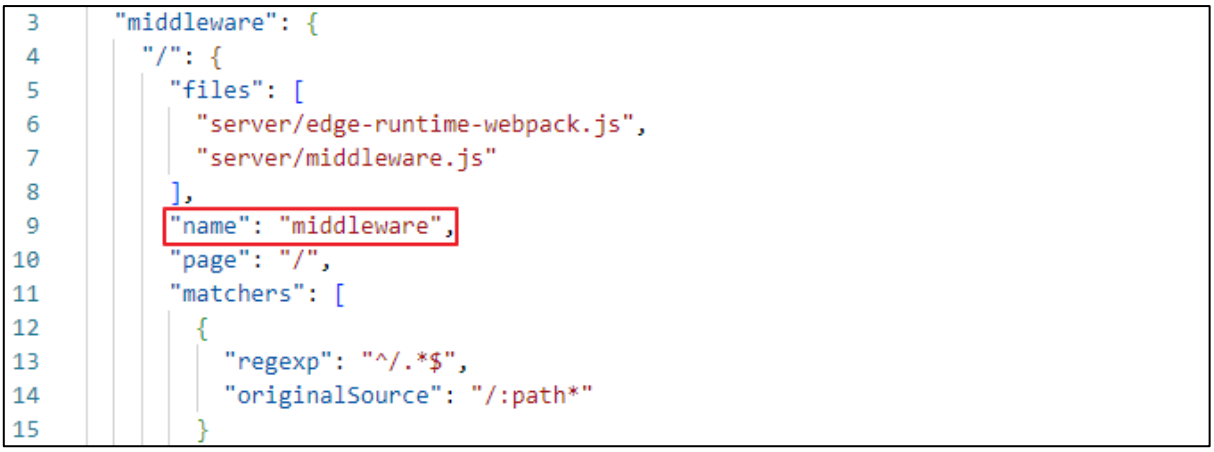


그림 23. middleware-manifest.json 내 middleware name 정보

그림 24. 디버거를 통해 확인한 params.name 의 값

따라서, v14.2 이후에는 colon(:)을 기준으로 middleware 경로가 5 번 이상 반복되어야 하므로, middleware:middleware:middleware:middleware:middleware 와 같은 값을 x-middleware-subrequest 헤더에 입력하면 middleware 검증을 우회할 수 있다.

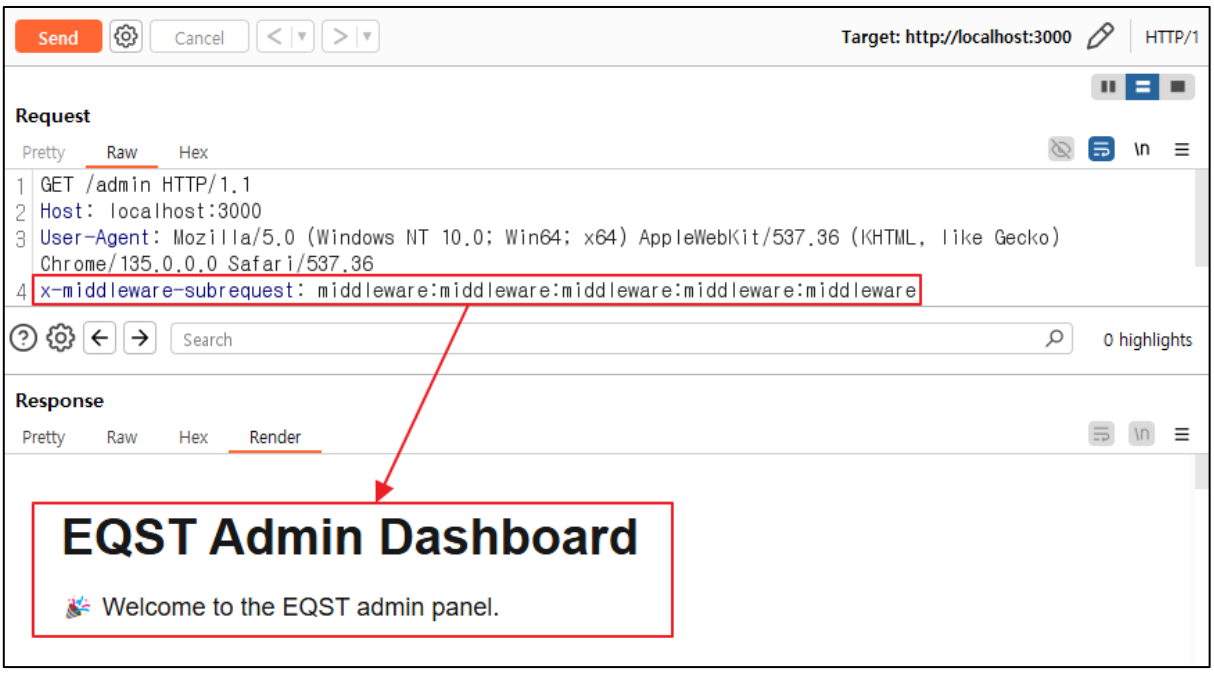


그림 25. middleware 인증 과정 우회 확인

5) 버전별 middleware 우회 payload

그림 6 과 같이, middleware 는 src 디렉토리 내에 위치할 수도 있다. 따라서, x-middleware-subrequest 헤더 값으로 입력할 수 있는 middleware 우회 payload 는 버전별로 아래와 같다.

버전	우회 payload 예시
v12.2 미만	pages/[SubDirectory]/_middleware 또는 ages/[subdirectory]/_middleware
v12.2 이상 v14.2 미만	middleware 또는 src/middleware
v14.2 이상	middleware:middleware:middleware:middleware:middleware 또는 src/middleware:src/middleware:src/middleware:src/middleware:src/middleware

■ 대응 방안

Next.js middleware 인증 우회 취약점인 CVE-2025-29927 은 취약점 연구원인 zhero;와 inzo_에 의해 발견되었으며, 2025 년 2 월 27 일에 처음으로 신고되었다.

•URL:https://zhero-web-sec.github.io/research-and-things/nextjs-and-the-corrupt-middleware#section-7

해당 취약점은 2025 년 3 월 18 일에 패치되었으며, 이에 대한 보안 권고는 3 월 21 일에 공개되었다.

•URL: https://github.com/vercel/next.js/security/advisories/GHSA-f82v-jwr5-mffw

v15.2.3 에서 패치된 내역을 살펴보면, 어떤 보안 조치가 취해졌는지 확인할 수 있다.

•URL: https://github.com/vercel/next.js/compare/v15.2.2...v15.2.3

우선, packages/next/src/server/lib/router-server.ts 에서 crypto.getRandomValues 함수를 사용하여 8 바이트의 랜덤 값을 생성한다. 이후 이 값을 @next/middleware-subrequest-id 에 대한 값으로 전역 변수[4] 에 저장한다.

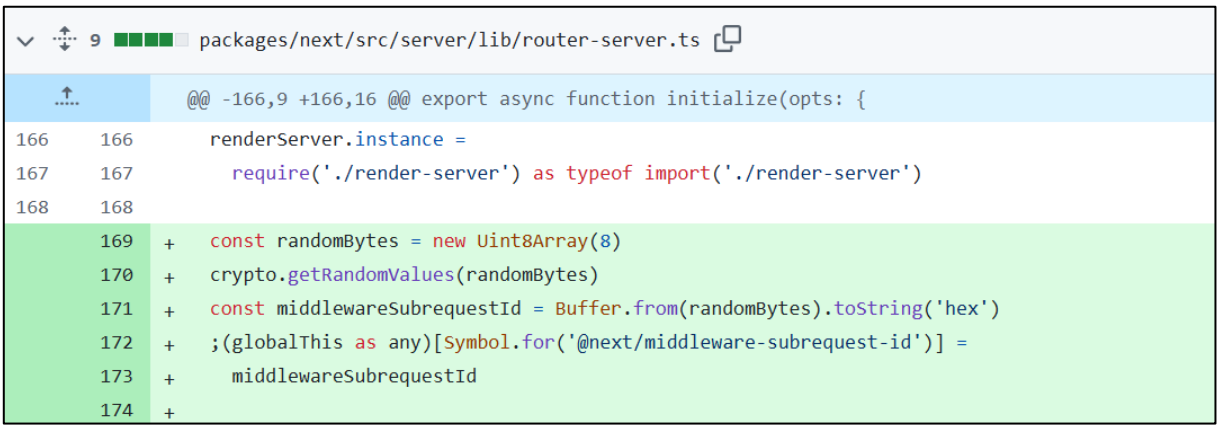


그림 26. packages/next/src/server/lib/router-server.ts 수정 사항

[4] 전역 변수(global variable): 함수의 외부에서 선언된 변수로, 프로그램 어디에서나 접근할 수 있는 변수

이후, packages/next/src/server/lib/server-ipc/utils.ts 에서 구현된 검증 과정은 x-middleware-subrequest 헤더와 x-middleware-subrequest-id 헤더 값이 존재하는지 확인하고, x-middleware-subrequest-id 헤더 값이 앞서 전역 변수에 저장된 @next/middleware-subrequest-id 와 일치하는지 검사한다. 만약 값이 일치하지 않으면, x-middleware-subrequest 헤더를 삭제하여 이를 활용할 수 없도록 막는다.

11 ■■■■■ packages/next/src/server/lib/server-ipc/utils.ts

```
@@ -57,5 +57,16 @@ export const filterInternalHeaders = (
57 57      if (INTERNAL_HEADERS.includes(header)) {
58 58        delete headers[header]
59 59      }
60 +
61 +    // If this request didn't origin from this session we filter
62 +    // out the "x-middleware-subrequest" header so we don't skip
63 +    // middleware incorrectly
64 +    if (
65 +      header === 'x-middleware-subrequest' &&
66 +      headers['x-middleware-subrequest-id'] !==
67 +        (globalThis as any)[Symbol.for('@next/middleware-subrequest-id')]
68 +    ) {
69 +      delete headers['x-middleware-subrequest']
70 +    }
```

그림 27. packages/next/src/server/lib/server-ipc/utils.ts 수정 사항

사용자는 내부적으로 저장되는 8 바이트 난수 값을 예측할 수 없으므로, 서버 측에서는 x-middleware-subrequest 헤더를 안전하게 활용하여 middleware 내부 실행 여부를 판별할 수 있다.

취약한 버전 사용 여부는 소스코드 내 package.json 파일을 통해 확인할 수 있다. 15.2.3 미만, 14.2.25 미만, 13.5.9 미만, 12.3.5 미만, 11.x 버전이 취약한 버전이므로, 해당 버전을 사용 중이라면 패치를 통해 x-middleware-subrequest 가 공격자에게 악용되지 않도록 차단하는 것이 중요하다.

```
{
  "name": "case4",
  "version": "0.1.0",
  "private": true,
  ▷Debug
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "react": "^19.0.0",
    "react-dom": "^19.0.0",
    "next": "15.2.2"
  },
  "devDependencies": {
    "typescript": "^5",
    "@types/node": "^20",
    "@types/react": "^19",
    "@types/react-dom": "^19",
    "postcss": "^8",
    "tailwindcss": "^3.4.1"
  }
}
```

그림 28. 취약한 next.js 버전 사용 예시

다만 11.x 버전은 이미 지원이 종료되어 더 이상 패치가 불가능하므로, 공식 지원이 지속되고 있는 15 버전으로의 마이그레이션을 권장한다. 만약 안전한 버전으로의 패치가 어려운 상황이라면, x-middleware-subrequest 헤더를 포함한 외부 사용자의 요청이 Next.js 애플리케이션에 도달하지 않도록 차단하는 것이 바람직하다.

■ 참고 사이트

• Wikipedia (Next.js) : https://en.wikipedia.org/wiki/Next.js
• Wikipedia (Static web page) : https://en.wikipedia.org/wiki/Static_web_page
• Wikipedia (Server-side rendering) : https://en.wikipedia.org/wiki/Server-side_scripting
• Project structure and organization (Next.js docs) : https://nextjs.org/docs/app/getting-started/project-structure
• middleware (Next.js docs) : https://nextjs.org/docs/app/building-your-application/routing/middleware
• Next.js and the corrupt middleware: the authorizing artifact (zhero_web_security) : https://zhero-web-sec.github.io/research-and-things/nextjs-and-the-corrupt-middleware
• Authorization Bypass in Next.js Middleware (GitHub Security Advisory) : https://github.com/vercel/next.js/security/advisories/GHSA-f82v-jwr5-mffw

EQST INSIGHT

2025.04

SK 쉴더스

SK쉴더스(주) 13486 경기도 성남시 분당구 판교로227번길 23, 4&5층
https://www.skshieldus.com

발행인 SK쉴더스 EQST사업그룹
제 작 SK쉴더스 마케팅그룹