2026-비정기-ARR-디지털자산-699

Vol.1

DeepChain
Cross-Chain Breaking Point

(디지털자산 크로스 체인 보안 위협 분석)

2026. 06.

금융보안원
FINANCIAL SECURITY INSTITUTE

C/O/N/T/E/N/T/S

Ⅰ

서 론

1. 배경

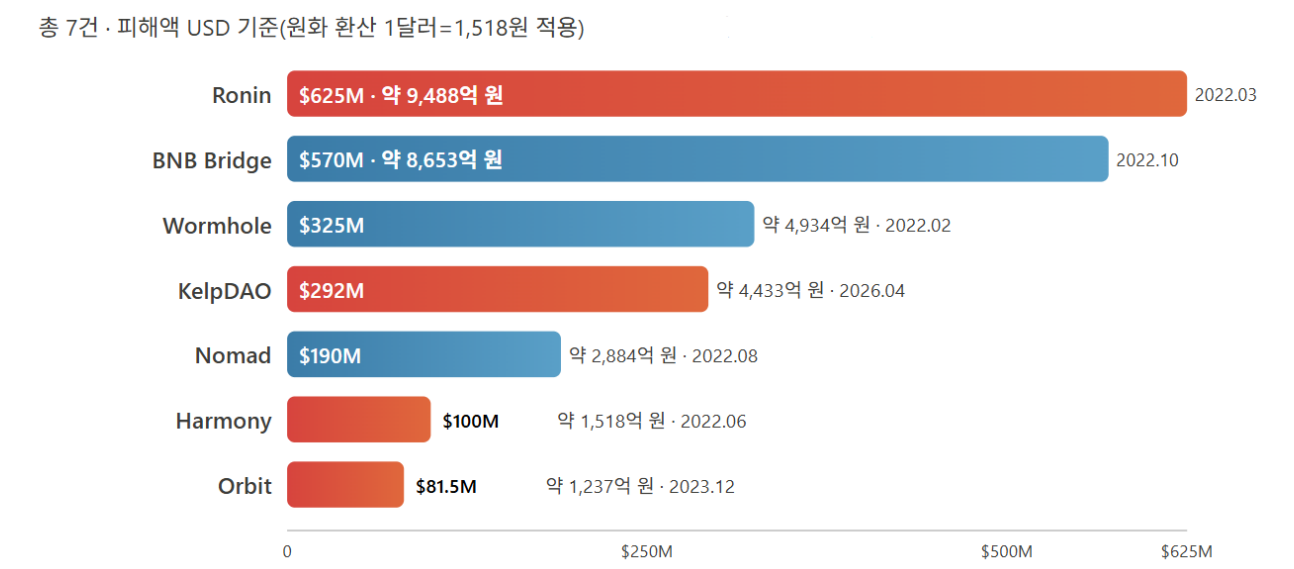
최근 금융산업에서 스테이블코인[1], 토큰증권[2] 등 디지털자산의 활용 및 가상자산 2단계 입법 등 제도화가 확대되면서 신한·하나·KB·우리 등 국내 주요 금융지주는 스테이블코인·토큰화자산 (RWA) 인프라 구축을 신년 핵심 과제로 제시함에 따라, 이를 통해 발생할 수 있는 사이버보안 위협이 한층 높아지고 있다.

이러한 환경에서 디지털자산의 업무 범위가 확대될수록 블록체인 간 네트워크를 이어주는 '크로스 체인 연계(Cross-Chain Interoperability)'가 필수적으로 고려되고 있으나, 크로스 체인 연계 관련 보안 사고는 대부분 대규모의 자산탈취로 이어지고 있다.

이에, 국내 금융권에서 향후 스테이블코인 등을 활용한 디지털자산 업무가 본격화되는 시점에 브릿지 등 크로스 체인 연계를 활용하는 경우, 대규모 자산 탈취 위험에 노출될 수 있으므로 크로스 체인에 대한 보안을 충분히 고려하여야 한다.

1) 스테이블코인(StableCoin) : 미국 달러나 원화 등 법정화폐나 특정 자산에 가치를 연동(페깅)하여 가격 변동을 최소화하도록 설계된 암호화폐
2) 토큰증권(Security Token) : 주식 및 채권, 부동산 등 실물 및 금융자산에 대한 권리를 블록체인 기반 토큰형태로 발행한 것으로, 자본시장법상 증권으로 규율되는 디지털자산의 형태

크로스 체인 주요 사고 피해 금액



2. 국내 디지털자산 현황

지난 3년간 국내·외 디지털자산 관련 제도는 빠르게 변화하였다. 주요국의 디지털자산 규제는 '개별 지침'에서 '통합 법제'의 형태로 변화하였고, 그 예시로 EU는 MiCA를 통한 암호자산 프레임워크 구현 및 미국은 GENIUS Act로 스테이블코인 법제를 제정하였다.

국내에서는 「가상자산 이용자 보호 등에 관한 법률」이 시행되어 그간 자금세탁방지 중심이었던 특정금융정보법 체계의 한계를 보완하고 이용자 자산 보호와 불공정거래 규제를 위한 기본틀을 마련하였다. 또한, 관련 법안의 2단계 입법이 추진되고 있으며 해당 법안에는 스테이블코인 규율 체계 및 사고 시 입증책임에 관한 규정 등이 포함될 예정이다.

추가로 토큰증권의 경우 자본시장법·전자증권법 개정안이 통과되어 2027년 2월 시행을 앞두고 있다.

3. 크로스 체인 연계 사례

국내 금융권이 준비 중인 디지털자산 관련 업무 확대를 위해서는 개별 사업 영역에 그치지 않고 서로 유기적인 연결이 필요하다. 이때, 단일 블록체인 내 활용이 아닌 복수의 블록체인 간 통신을 가능하게 하는 크로스 체인 연계 기술이 필요하게 된다. 이에, 크로스 체인 연계를 통한 대표적 활용 예시를 설명하고자 한다.

3.1. 해외 송금 및 무역 결제

스테이블코인을 활용한 해외 송금·무역 결제는 크로스 체인 연계 기술의 대표적 활용 사례가 될 것으로 판단된다. 기존 SWIFT 기반 송금은 평균 1~3 영업일과 비싼 수수료가 발생되는 반면 스테이블코인을 이용하면 짧은 시간(ex. 수분내) 낮은 수수료를 통한 거래가 성사될 수 있다.

이때, 무역 대금에 대해 송금하는 측과 받는 측이 동일한 블록체인을 사용한다는 보장이 없기에 거래에 사용될 스테이블코인을 상대방이 사용하는 다른 체인으로 이전하기 위해서는 체인 간 연계 즉, 크로스 체인 연계 기술이 필요하게 된다.

3.2. 법정화폐간 스테이블코인 전환

Circle과 같은 달러 기반의 글로벌 스테이블코인 사업자는 각국의 통화 화폐를 USDC로 전환하는 것에 대한 관심을 가지고 있다. 그 예시로 국내 5대 원화 거래소 내 원화 마켓에는 이미 원화를 이용하여 스테이블코인(USDC 등)을 구매할 수 있고 향후 원화 스테이블코인이 발행되면 원화 스테이블코인과 USDC 전환 같은 스테이블코인 간 전환이 가능해진다.

이때, 스테이블코인간 전환을 위해서는 크로스 체인 연계 기술이 필요하게 된다. 예를 들어, 원화 스테이블코인이 특정 체인(예: 국내 특정 사업자에서 개발한 체인 기반)에서 발행되고 USDC가 주로 이더리움(Ethereum, ETH)이나 솔라나(Solana) 등에서 유통된다면 두 자산 간 전환을 위해서는 크로스 체인 연계 기술을 통한 자산 이전이 필요하기 때문이다.

3.3. 기관 수탁 서비스

가상자산 이용자 보호법에 따라 이용자 자산의 분리·보관이 의무화되면서 기관수탁 서비스(Custody) 시장이 형성되고 있다. 수탁사는 단일 체인의 자산만 보관하는 것이 아니라 보관을 요청하는 고객의 다양한 체인(이더리움, 솔라나, 아발란체 등)의 자산을 관리해야 하고 필요시 체인 간 이동을 중개해야 하기에 크로스 체인 연계 기술이 필요해진다.

3.4. 실물자산 토큰화(RWA) 결제망

실물자산 토큰화(RWA)[3]가 활성화 되는 경우 스테이블코인을 통한 체인 간 결제가 발생할 수 있다. 예를 들어, 이더리움 기반으로 발행된 부동산 관련 토큰증권을 아발란체(Avalanche) 기반의 원화 스테이블코인으로 결제하는 경우다.

이때, 서로 다른 체인 간 자산의 소유권 이전과 대금 지급 등이 안전하게 처리되려면 크로스 체인 연계를 통한 자산 이전 등이 필요해진다.

3) 실물 자산 토큰화(RWA) : 부동산, 채권, 예술품 등 실물 자산의 소유권이나 가치를 블록체인 기반 토큰으로 발행하여, 분할 거래 및 이전이 가능하도록 디지털화

4. 위협환경의 구조적 전환

앞서 말한 바와 같이 디지털자산 확대에서 크로스 체인 연계는 필수적으로 고려하여야 하는 요소이나, 해당 기술은 기존 금융권이 고려하고 있었던 기술과는 전혀 다른 위협 구조를 가지고 있다.

기존 금융거래는 은행 및 중앙예탁기관 등 법적 책임을 지는 신뢰된 중앙 기관이 거래를 보증하고 거래 내역을 내부 원장에서 통제 및 관리함에 따라, 오류나 부정 거래가 발생하더라도 사후에 거래를 정정 및 취소하거나 동결할 수 있으며 분쟁 발생 시 책임 주체가 명확해진다.

반면, 크로스 체인 연계는 신뢰가 중앙 기관이 아니라 분산된 다수의 기술적 구성 요소에 의존하게 된다. 거래의 유효성은 검증자 노드, 서명 키, 외부 데이터(오라클), 메시지 전달자 등 여러 주체가 협력하여 보증하며, 블록체인상의 거래는 한번 체결되면 되돌릴 수 없는 비가역성을 가지므로 부정 출금이 일단 승인되면 사후 정정이나 회수가 사실상 불가능하다. 그 결과 단 한 번의 위·변조된 거래만으로도 대규모 자산이 유출될 수 있다.

이에, 크로스 체인 연계 기술을 활용하고자 하는 경우 단일 체인의 스마트 컨트랙트 보안 검증 등 일부 영역에 국한된 기술로는 충분하지 않고 검증자(Validator) 노드의 구성, 서명시스템 키 관리, 외부 데이터(오라클) 참조, 외부 메시지 전달자(Relayer) 등 다양한 신뢰 구성 요소가 동시에 보호되어야 한다.

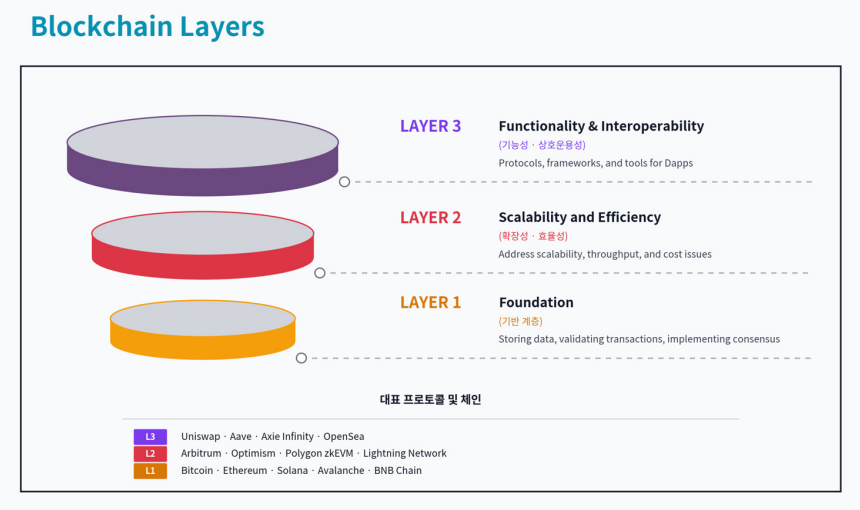
Ⅱ

크로스 체인 연계 인프라 구조

1. 블록체인 레이어

본 장에서는 크로스 체인 연계 기술을 정확히 이해하기 위한 기본 개념인 블록체인 레이어(Layer)에 대해 설명한다.

블록체인 '레이어'와 '크로스 체인 연계'에서 정확한 이해가 필요한 이유는 디지털자산 생태계에서 같은 계층이지만 다른 블록체인(L1 ↔ L1) 또는 다른 계층이면서 다른 블록체인(L1 ↔ L2) 등 서로 다른 계층 및 블록체인 간 디지털자산의 교환이 필요하고 이를 크로스 체인 연계 기술로 해결할 수 있기 때문이다.



1.1. 레이어(Layer) 1

블록체인의 생태계의 기반 레이어로 자체 합의 메커니즘(Consensus Mechanism)과 검증자 네트워크 등을 갖추고 있어 트랜잭션의 유효성을 검증하고 블록을 생성하는 등 체인 내 최종 정산(Final Settlement)을 담당한다. 다만, 네트워크 규모가 커질수록 시스템 성능 저하 및 비싼 수수료가 발생하게 되며 비트코인, 이더리움 등이 대표적 활용 코인이다.

핵심 특징

가. (탈중앙화) 단일 중앙 기관이라는 개념 없이 각지에 분포된 수천에서 수만개의 노드가 동등한 권한으로 네트워크를 운영하는 방식으로, 모든 노드가 거래 내역(트랜잭션 기록)을 보유하고 있고, 신규 블록을 추가하고자 하는 경우 다수 노드의 합의를 거쳐야 하는 특성이다.

나. (데이터 불변성) 한번 블록체인에 기록된 트랜잭션은 이론적으로 되돌릴 수 없는 특성이다. 각 블록이 이전 블록의 해시값을 포함하고 있고, 만일 위조를 시도하는 경우엔 해당 체인 노드 중 과반수 이상의 노드를 장악하여야 한다.

1.2. 레이어(Layer) 2

Layer 1의 트랜잭션 처리 속도와 확장성을 개선하는 보조 계층으로 Layer 2를 활용 시 Layer 1 대비 처리량이 수십에서 수백배까지 향상되고, 수수료는 수십분의 일 수준으로 낮출 수 있다.

주요 유형

가. (옵티미스틱 롤업) Optimistic Rollup으로 트랜잭션은 기본적으로 유효하다고 가정하는 방식이다. 일반적으로 출금을 요청하는 경우 7일의 검증(이의 제기) 기간 후 출금이 자동으로 확정되는 방식이다.

나. (영지식 롤업) 영지식 증명(zk-SNARK 등)을 통해 트랜잭션의 유효성을 암호화적으로 증명하고, 증명 결과를 Layer 1에 기록하는 방식이다. 출금에 대한 지연이 없고, 수학적으로 유효성을 검증하여 보안적으로 우수하다고 알려져 있다. 다만, 암호화적 증명 방식을 이용함에 따라 연산 난이도가 높다.

1.3. 레이어(Layer) 3

Layer 2 위에서 실행되는 애플리케이션 특화 계층으로 특정 dApp·서비스·산업 등 특성화된 환경에 따른 요구사항에 맞춘 블록체인 환경을 제공하므로 Layer 2보다 성능·확장성 등이 강화된 계층이다. 다만, 별도 환경을 고려하여 만들어진 애플리케이션으로 신규 취약점이 발생될 수 있으며, 기술적 복잡성이 증가할 수 있다.

주요 활용 방안

가. (dApp) 특정 요구사항에 맞추어 수수료, 속도, 프라이버시 기능 등을 튜닝하여 제공할 수 있다.(게임 내 결제, NFT 민팅 최적화 등)

나. (앱 특화 체인) 게임 전용 체인, DeFi 전용 체인, RWA 토큰화 체인 등 특정 목적 또는 서비스를 위해 최적화된 독립 체인으로 활용할 수 있다.

2. 크로스 체인 연계 방식

크로스 체인 연계 방식 대표적 분류는 'Lock & Mint', 'Burn & Mint', 'Hybrid Burn & Mint' 총 3가지로 분류할 수 있다.

Lock & Mint 방식

정의	❑ 원본 체인에서 자산을 락업(Lock)한 후 대상 체인에서 이를 대체하는 '랩드 토큰(Wrapped Token)'을 신규로 발행(Mint)하는 방식
연계 흐름	 1. A 체인에서 원본 토큰을 잠금(Lock)한다. 2. Bridge Contract로 검증을 요청하고 검증이 완료된 발행 승인 메시지를 B체인으로 전달한다. 3. B체인에서는 메시지 확인 후 랩드 토큰을 발행(Mint)한다.
특징	❑ 다양한 체인에 적용이 가능한 방식으로 구현이 용이하다.
주요 예시	❑ WBTC, DAI, Wrapped ETH 등
고려 사항	❑ 브릿지 스마트 컨트랙트 또는 검증자 키가 침해될 경우 해당 브릿지를 통해 발행된 모든 랩드 토큰의 자산 가치가 무너질 수 있다.

Burn & Mint 방식

정의	❑ 원본 체인에서 토큰을 소각(Burn)하고 대상 체인에서 동일 수량의 토큰을 새로 발행(Mint)하는 방식
연계 흐름	 1. A 체인에서 원본 토큰 소각(Burn)하고, 소각 이벤트를 발행사로 전달한다. 2. 발행사에서는 소각 이벤트에 대해 암호학적 증명을 활용하여 검증(Attestation)을 수행한다. 3. 발행사에서는 검증의 결과를 확인 후 발행(Mint)을 승인한다. 4. 발행 승인 메시지(증명서)를 전달 받은 B 체인에서는 동일한 네이티브 토큰을 생성한다.
특징	❑ 토큰 총 공급량이 일정하게 유지되며, 랩드 토큰없이 네이티브 토큰으로 이전 가능 ❑ 발행 통제권 및 서명 루트가 발행사 서명으로 보안성이 높으며, 책임소재가 명확
주요 예시	❑ USDC(CCTP, Cross-Chain Transfer Protocol), USDT(일부 체인) 등
고려 사항	❑ 발행사 시스템 장애 발생 시 전체 시스템 중단 위험이 존재한다. ❑ 발행사에서 검증 및 서명하는 구조에서 서명이 위·변조 되는 경우 무담보 발행이 가능하다.

Hybrid Burn & Mint 방식

정의	❑ 기본적인 방식은 Burn & Mint 방식과 동일하나, 소각 및 발행 이벤트 메시지에 대한 검증을 제3자 검증 그룹인 DVN 검증자 그룹이 수행하는 방식
연계 흐름	 1. A 체인 컨트랙트에서 원본 토큰을 소각(Burn)한다. 2. 토큰 소각이 완료되는 경우, 크로스 체인 메시지를 생성한다. – Secure Chain ID, Destination Chain ID, Amount 등을 기록 3. DVN[4] 검증을 수행하고, 검증이 완료되면 DVN은 Attestation(암호학적 서명 증명서)를 제출한다. – 각 DVN은 자신이 운용하는 RPC 노드[5]를 통해 A 체인 상태를 조회하고, 해당 소각 트랜잭션이 실제로 확정되어 있는지 확인 4. '3' 과정이 완료되면 Executor가 실행되면서 B 체인의 컨트랙트에 메시지를 전달한다. 5. B체인 컨트랙트에서 '4' 과정을 통해 전달된 메시지의 진위 여부를 검증한다. 6. 검증이 완료된 후 B 체인에서 네이티브 토큰을 발행(Mint)한다.
특징	❑ 메시지 기반 전송 방식으로 확장성 및 유연성이 높으며, 다수 검증자 기반(DVN) 방식으로 메시지 위·변조에 대응
주요 예시	❑ USDT0(LayerZero)
고려 사항	❑ DVN 침해 시 메시지 위변조 가능

4) DVN(Decentralized Verifier Network) : 크로스 체인 메시지가 실제로 발생했는지를 독립적으로 검증하고 암호학적 서명을 제출하는 검증자 집합

5) RPC(Remote Procedure Call) 노드 : 블록체인 데이터를 외부에서 조회 및 요청할 수 있도록 응답해주는 역할을 하는 서버(EX. DVN 등이 특정 트랜잭션의 실제 발생 여부를 확인할 때 참조하는 역할 등)

3. 스마트 컨트랙트 구조 분석

본 장에서는 앞 장에서 다룬 크로스 체인 연계 방식 중 활용 사례가 가장 많은 'Lock & Mint' 방식을 예시로 스마트 컨트랙트 구조를 분석하여 설명한다.

Lock & Mint 방식에서의 크로스 체인은 기술적으로 3가지의 구성 요소가 상호 작용하는 구조로 구성되며, 구성 요소는 체인 A(원본 자산 체인), 체인 B(랩드 토큰 발행 체인)와 그리고 전달자(Relayer)와 검증자(Validator) 그룹이다.

스마트 컨트랙트 구조(Lock & Mint)

2.1. 스마트 컨트랙트 함수 분석(Lock & Mint 기준)

1. deposit()

고객 자산 예치 및 이벤트를 발행하는 함수로 사용자가 체인 A의 'Vault 컨트랙트'에 있는 "deposit()" 함수를 호출하면서 이전하려는 토큰과 수량, 도착지 체인 및 수신자 주소 정보를 전달한다. 컨트랙트는 해당 토큰을 내부 잠금 풀(Locking Pool)에 보관하고 이벤트 로그(Event log)를 온체인에 기록한다.

이 단계에서 고객 자산은 Vault 컨트랙트가 통제하는 주소로 이전되어 잠기며 유효한 출금 서명이 없는 한 컨트랙트 외부로 이동할 수 없다.

2. 전달자(Relayer) 및 검증자(Validator)

오프체인에서 동작하는 전달자(Relayer)는 체인 A의 온체인 이벤트 로그를 실시간으로 모니터링을 수행하고 'deposit()' 이벤트가 감지되면 해당 트랜잭션의 유효성을 확인하고 검증자(Validator)로 전달하여 서명을 요청하게 된다.

예를 들어, 브릿지가 N-of-M 멀티시그 서명 구조에서 5-of-9라면 9개의 검증자 중 5개의 검증자가 서명을 해야 해당 메시지가 유효하다고 판단하게 된다.

3. mint()

서명이 완료되면 전달자(Relayer)는 체인 B의 'Mint 컨트랙트'에 "mint()" 함수를 호출하는 트랜잭션을 전달하고 'Mint 컨트랙트'는 제출된 서명의 수학적 유효성과 임계값 충족 여부 등을 검증한 후 체인 A에서 잠김 자산과 동일한 수량의 랩드 토큰(Wrapped Token)을 수신자의 지갑 주소로 발행한다. 예를 들어, 체인 A에서 1개의 이더리움이 잠기면, 체인 B에서는 1개의 랩핑 이더리움을 수신자에게 발행하게 된다.

여기서, 랩드 토큰은 체인 A의 잠금 풀에 잠겨있는 원본 자산의 가치를 그대로 반영하여야 한다.

위까지의 과정이 'Lock & Mint'에서 실제로 자산을 발행하는 과정이었으며 발행한 자산을 소각하는 경우에 발생하는 과정도 추가로 알아보겠다.

4. burn()

사용자가 원래 체인 A의 자산을 돌려받고자 하는 경우 체인 B의 'Mint 컨트랙트'에서 "burn()" 함수를 호출하여 사용자가 보유한 랩드 토큰을 소각(파기)하고 소각된 수량과 체인 A에서의 수신자 주소 정보를 담은 출금 요청 이벤트를 온체인에 기록한다.

5. 전달자(Relayer) 및 검증자(Validator)

전달자(Relayer)는 체인 B에서 발생한 “burn()“ 함수 이벤트를 감지하고 동일하게 검증자(Validator)를 통한 서명 과정을 거쳐 체인 A의 ‘Vault 컨트랙트’에게 출금 승인 메시지를 전달한다.

6. withdraw()

체인 A의 ‘Vault 컨트랙트’에서 서명 과정을 거친 출금 요청을 수신하게 되면 출금 요청 메시지에 담긴 내용을 검증 후 잠금 풀에서 해당 수량의 원본 토큰을 사용자 체인 A 주소로 반환하게 된다.

Ⅲ

크로스 체인 보안

1. 크로스 체인 위협 요소

지금까지 크로스 체인 연계 방식과 스마트 컨트랙트 호출 구조 등 크로스 체인 공격을 쉽게 이해하기 위한 배경 지식을 설명하였으며, 이번 장에서는 크로스 체인에서 노출될 수 있는 공통 위협 요소에 대해 안전하게 유지되는 조건(전제 조건)과 공격자가 주로 이용할 수 있는 공격 벡터를 설명한다.

또한, 크로스 체인 연계 방식 중 일부 방식에서는 위험이 가중될 수 있는 요소를 추가로 설명한다.

1.1. 공통 위협 요소

1 스마트 컨트랙트

전제 조건

첫 번째 요소는 핵심 로직이 구현된 스마트 컨트랙트이다. 해당 요소를 안전하게 유지하기 위해서는 “스마트 컨트랙트 코드가 개발자(발행사 등)가 의도한 대로 정확하게 실행된다.”라는 조건이 지켜져야 한다. 따라서, ‘deposit()’, ‘withdraw()’, ‘mint()’, ‘burn()’과 같이 실제 자산과 직접적으로 연관된 중요 함수들이 모두 의도한 대로 실행되고 어떠한 비정상적인 경로로 호출이 되지 않아야 한다.

공격 벡터

공격자는 해당 요소를 공격하기 위해 스마트 컨트랙트 코드 결함을 주로 이용한다. 비즈니스 로직 관련 코드 오류, 재진입 취약점 등이 대표적 공격 벡터이다.

2 키 관리

전제 조건

두 번째 요소는 크로스 체인 연계 시 필수적으로 사용되는 암호화 키의 관리 체계이다. 출금을 승인하는 서명키, 발행사의 검증 관련 서명키 등 크로스 체인 연계에서 승인 및 서명 등에 활용되는 모든 키가 해당 요소에 포함이 된다. 해당 요소를 안전하게 유지하기 위해서는 “암호화 키가 인가된 사용자(발행사, DVN 등)만 접근 가능하고 안전하게 보관되고 있다.”라는 조건이 지켜져야 한다.

공격 벡터

크로스 체인 연계 관련 보안 사고의 경우 대부분 해당 요소가 안전하게 지켜지지 못해 발생하였다. 스마트 컨트랙트 코드 자체에는 결함이 없었으나 서명 키가 탈취되면서 발생한 유효한 위조 서명을 통해 발생한 사례 등 공격자는 주로 자산출금 및 발행에 이용되는 서명 키를 탈취하려고 시도한다.

3 검증자 집합

전제 조건

세 번째 요소는 서로 다른 블록체인 간 연결 시 이벤트를 감지하고 메시지를 전달·검증하는 운영 주체에 대한 부분이다. 해당 요소가 안전하게 유지되기 위해서는 “검증자 집합(전달자 및 검증자)이 정직하게 동작하여야 하고 서로 악의적 목적으로 담합하지 않아야 한다.”라는 조건이 지켜져야 한다.

예를 들어, N-of-M 멀티시그 구조에서 M명의 검증자 중 과반이 동시에 침해되거나 또는 악의적 목적으로 담합하는 경우 무분별한 발행 및 출금 등이 발생할 수 있다.

공격 벡터

이에, 공격자는 검증자의 권한을 탈취하려고 한다. 예를 들어, N-of-M 구조에서 M에 해당하는 검증자 노드 및 검증자 서명키를 침해 및 탈취하는 방식이다.

4 외부 데이터(오라클)

전제 조건

네 번째 요소는 브릿지에서 자산 가격이나 체인의 상태 데이터를 외부 오라클에 의존하는 경우에 해당된다. 해당 요소를 안전하게 유지하기 위해서는 “외부 데이터(오라클)이 제공하는 데이터가 실제 시장 상태 및 체인의 상태를 정확하게 반영하고 조작되지 않아야 한다.”라는 조건이 지켜져야 한다.

공격 벡터

이에, 공격자는 주로 플래시 론(Flash Loan)을 이용해 단일 블록 내에서 오라클이 참조하는 가격을 인위적으로 급등시킨 후 왜곡된 가격 기준으로 실제 가치보다 과도한 랩드 토큰 발행을 유도하는 방식으로 공격한다.

5 거버넌스·관리 체계

전제 조건

마지막 다섯 번째 요소는 거버넌스 및 운영 체계이다. 해당 요소는 크로스 체인 연계뿐만 아니라 IT 관련 업무 시 반드시 고려하여야 하는 요소이다.

해당 요소를 크로스 체인 연계 관점에서 안전하게 유지하기 위해서는 “스마트 컨트랙트 업그레이드, 키 관리 체계 변경, 블록체인 네트워크 인프라 운영 등의 행위가 안전하고 투명한 절차를 통해 수행된다.”라는 조건이 지켜져야 한다.

공격 벡터

이에, 공격자는 취약한 거버넌스 운영 지점(인프라 운영 실수, 관리자 권한 탈취 등)을 통해 자산을 탈취하려고 한다.

1.2. 특정 방식에서의 위협 요소

1 Burn & Mint

발행자 검증 침해

Burn & Mint 방식의 경우 발행자가 발행 및 검증을 모두 수행하고 있다고 설명하였다. 이는, 제3자 검증자 또는 브릿지 사업자의 해킹을 통한 자산 유출 위험이 적어졌다고 설명할 수 있으나 발행자가 발행 및 검증을 모두 수행함에 따라 발행자의 검증 체계가 침해당할 경우 랩드 토큰이 아닌 원본 자산 자체가 유출될 가능성이 존재한다.

발행자 인프라 장애

발행자가 발행 및 검증을 모두 책임지고 있는 Burn & Mint 방식에서 발행자 인프라의 장애가 발생하는 경우 해당 블록체인 네트워크가 모두 중단될 수 있는 위험이 존재한다. 이는 직접적인 자산 피해로 바로 이어지지는 않지만 이용하고 있는 고객 피해 등을 통해 이미지 손실이 발생할 수 있다.

2 Hybrid Burn & Mint

DVN 구성 미흡(Hybrid Burn & Mint)

Hybrid Burn & Mint 방식은 위에 설명한 바와 같이, Burn & Mint 방식에서 발행자 및 검증자가 모두 단일 업체로 의존적인 부분을 보완하고자 별도의 검증 그룹을 두어 자산을 연계시키는 방식이다.

다만, 해당 검증을 담당하는 그룹(DVN)을 단일로 구성하거나 또는 다수로 구성된 DVN 일지라도 임계값이 1-of-3 등으로 낮게 설정된 경우 공격자에 의해 단일 DVN이 침해당하게 되면 위조 메시지에 대한 정상 검증이 가능해진다. 즉, 공격자가 DVN 서명키를 탈취하여 실제 소각 이벤트가 발생하지 않더라도 임의의 메시지에 유효한 서명을 발생시켜 목적지 체인에서 네이티브 토큰 무단 발행이 가능해진다는 개념이다.

또한, 검증자 그룹(DVN)과 실행 그룹(Executor)의 역할이 분리되어 있지 않은 경우, 공격자는 Executor의 권한을 탈취하여 검증이 완료된 메시지의 실행 지연 및 거부 등 DoS[6] 형태의 공격을 수행할 수 있다.

6) DoS(Denial of Service, 서비스 거부) : 시스템 및 서비스를 정상 이용자가 이용하지 못하도록 마비시키는 공격

2. 크로스 체인 신규 위협 요소

지금까지 크로스 체인의 위협 요소를 설명하였다. 하지만, 자산을 안전하게 보관 및 이동하기 위해서는 해당 위협 요소뿐만 아니라, 공격 기술 발전으로 인한 신규 위협 요소를 항시 모니터링하고 파악하여야 한다. 따라서, 이번 장에서는 신규 위협 요소 2가지에 대해 설명하겠다.

AI 기반 취약점 탐색

최근 미토스(Mythos)와 같이 취약점 탐색에 특화된 생성형 모델(LLM)이 다수 개발되어 제공되고 있다.

이와 같은 상황에서, 스마트 컨트랙트가 퍼블릭 블록체인(온체인)에 올라가는 순간 스마트 컨트랙트는 누구나 볼 수 있도록 공개되고 공격자는 AI를 통해 수백·수천개의 스마트 컨트랙트를 단기간에 분석하고 취약점이 존재하는 경우 자동 공격 코드 생성을 통한 자산 유출까지 이어질 수 있다.

또한, 공격자의 자동화 및 분석 속도가 크게 증가하므로 현재보다 스마트 컨트랙트 취약점을 통한 보안 사고 발생 빈도가 증가할 것으로 예상되며 이는 곧 고객의 자산 유출 피해가 증가할 수 있음을 시사한다.

양자컴퓨팅의 발전

현재 블록체인 서명 체계로 대부분 사용하는 ECDSA(타원곡선 암호화)는 충분한 성능을 가진 양자컴퓨팅 파워를 통해 개인키 역산이 이론적으로 가능하다고 알려져 있다.

다만, 양자컴퓨팅은 현재까지는 특정 공격자(개인)가 활용하기엔 기술적 구현이 어렵고 비용이 높아 단기 위협은 아니나, 북한 해킹 그룹과 같이 국가 단위의 지원을 받는 해킹 그룹을 통한 공격이 발생할 수 있는 신규 위협이다.

Ⅳ

크로스 체인 사고 사례

1. 크로스 체인 사고 개요

본 장에서는 실제 사고 사례를 통한 공격 및 자금 세탁 기법을 설명한다.

먼저, 크로스 체인이 원인으로 발생한 침해사고 중 단일 역대 사고는 Ronin Bridge 사고로 피해액이 약 9,500억(원화)에 달하며 현재(2026년 4월)까지 조사 중인 사고를 포함해 한화 약 2조 6천억(원화)의 피해가 발생하고 있다. 이는, 크로스 체인 연계 관련 사고가 대규모의 자산 피해로 이어질 수 있음을 시사한다.

중요한 점은 크로스 체인 해킹 사고 중 북한 해킹 그룹(이하, Lazarus 그룹) 소행으로 확정 또는 추정되는 사고로 발생한 피해액이 전체 피해액의 약 63%에 해당하는 높은 비중을 나타내고 있다는 점이다.

이에, 크로스 체인 사고 중 대부분을 차지하고 있는 Lazarus 그룹 관련(확정 또는 추정) 사고 사례를 통해 공격 및 자금 세탁 기법을 분석하고 설명한다.

2. 크로스 체인 대표 사고 사례 분석

Lazarus 그룹의 소행으로 확정 또는 추정되는 4개의 사고 사례(Ronin Bridge, Harmony Horizon, Orbit Chain, KelpDAO)에 대해서 사고 개요, 공격 흐름 등을 설명한다. 다만, Lazarus 그룹 소행으로 확정되지 않은 'Orbit Chain'에 대해서는 Lazarus 그룹이 아닌 '공격자'라는 표현을 활용하였다.

Case 1. Ronin Bridge

1 사고 개요

크로스 체인을 대상으로 Lazarus 그룹이 최초로 해킹한 사례이며 2022년 4월에 발생하였다. 블록체인 게임 업체인 Axie Infinity의 사이드체인 연결 브릿지인 Ronin Bridge에서 침해사고가 발생하였으며 검증자 노드 중 과반수 이상 노드의 서명키를 탈취하여 총 6억 2천 5백만 달러 규모의 자산을 탈취하였고, 2022년 4월 FBI와 OFAC(미국 재무부)은 해당 공격을 북한 Lazarus의 소행으로 판단하였다.

2 공격 흐름

공격 흐름은 크게 5가지로 사회공학 → 악성파일 실행 → 내부 침투(노드 장악 등) → 위조 출금(자산 탈취) → 자금 세탁의 단계로 이루어졌다.

1단계 : 사회 공학

Lazarus 그룹에서 채용 담당자를 위장하여 채용 플랫폼(LinkedIn 등)을 통해 공격 타켓(Sky Mavis 인프라 엔지니어)에게 고연봉을 미끼로 접근한 뒤 “채용 절차”를 가장하여 악성 파일을 전달한다.

2단계 : 악성 파일 실행

타켓이 된 담당자는 정상 채용 관련 파일로 위장한 파일을 실행하게 됨으로써, 담당자 단말에서 악성코드 파일(원격 제어 등) 다운로드 및 C2 서버 접속 등을 수행하게 된다.

3단계 : 내부 침투

Lazarus 그룹은 장악된 단말기를 통해 내부 인프라로 침투하여 Sky Mavis에서 보유하고 있는 4개의 검증자 노드의 서명키를 확보하게 된다. 하지만, Ronin의 경우 9개 검증자 노드 중 5개 이상의 검증자 노드 서명이 발생해야 출금이 승인되는 구조를 지니고 있어 필요한 서명 권한을 충족하지 못하였다.

여기서, 결정적 원인이 발생하게 된다. Sky Mavis에서 보유하고 있는 4개의 검증자 노드 외 1개의 검증자 노드 운영을 Axie Infinity 커뮤니티 조직인 Axie DAO에서 담당하고 있었는데 Sky Mavis에서 서명 시 발생하는 트랜잭션 부하를 해소하기 위해 Axie DAO에게 임시로 위임받은 서명 권한을 회수하지 않아, 결국 Lazarus 그룹은 Sky Mavis 인프라 침투를

통해 5개의 검증 노드 서명 권한을 확보하게 된다.

그렇다면, Lazarus 그룹은 확보한 검증 노드를 이용하여 어떻게 과반수 이상의 서명을 발생시킬 수 있었는지 실제 코드 중 일부를 발췌하여 설명하겠다.

MainchainGatewayManager.sol · Validator.sol — 실제 배포 코드

```
MainchainGatewayManager.sol
function verifySignatures(bytes32 _hash, bytes memory _signatures)
    public view returns (bool)
{
    uint256 _signatureCount = _signatures.length.div(66);
    Validator _validator =
        Validator(registry.getContract(registry.VALIDATOR()));
    uint256 _validatorCount = 0;
    address _lastSigner = address(0);

    for (uint256 i = 0; i < _signatureCount; i++) {
        address _signer = _hash.recover(_signatures, i.mul(66));

        if (_validator.isValidator(_signer)) {
            _validatorCount++;
        }
        require(_signer > _lastSigner);
        _lastSigner = _signer;
    }
    return _validator.checkThreshold(_validatorCount);
}

Validator.sol — Ronin num/denom = 5/9

function checkThreshold(uint256 _voteCount) public view returns (bool) {
    return _voteCount.mul(denom) >= num.mul(validatorCount);
}
```

※ (참고) Ronin은 9개 검증자 노드 서명 중 5개 이상의 서명이 필요하므로 기준값을 “num=5, validatorCount=9”로 설정

1단계	14~15번 라인에서 유효한 검증자 노드(isValidator()로 체크) 서명이 있는 경우 for문을 통해_validatorCount 값을 1씩 증가
2단계	20번 라인에서 for문을 수행한 _validatorCount 값을 checkThreshold() 함수 인자값으로 전달
3단계	25~26번 라인에서 전달된 _validatorCount 값을 _voteCount에 반영하여, 9개 검증자 노드 중 5개 이상의 검증자 노드가 서명에 참여하였는지 확인
4단계	5개 이상의 검증자 노드가 서명에 참여한 경우 return으로 ‘true’를 반환하여, 이후 비즈니스 로직을 정상적으로 실행

위 코드 설명과 같이, 스마트 컨트랙트에서는 유효한 검증자 노드들에서 설정한 임계치 값 이상 서명을 발생하는 경우 출금 등 비즈니스 로직을 처리하도록 구현하였으나 Lazarus 그룹은 획득한 5개의 서명 권한을 통해 위조 출금 요청을 승인하도록 만들었다.

4단계 : 위조 출금

Lazarus 그룹은 확보한 검증 노드를 통해 과반수 이상의 서명으로 173,600개의 이더리움 및 2,550만개의 USDC 출금을 무단으로 승인하여 탈취에 성공하게 된다.

※ (참고) 해당 사고는 약 6일 뒤 고객의 출금 실패를 통해서 알게 되었다.

5단계 : 자금 세탁

추적이 쉬운 USDC를 이더리움으로 환전해 세탁 대상을 이더리움으로 통일하였고 가상자산 거래소들이 수사 협조를 수행하자, Tornado Cash 믹서 서비스[7]를 활용하여 환전한 173,600개의 이더리움을 소액으로 반복 세탁하여 자금 추적을 힘들게 만들었다.

추후, 미국이 Tornado Cash를 제재한 뒤에는 체인 호핑[8] 기법을 활용하여 다른 블록체인 내 자산으로 교환 후 현금화를 시도하게 된다.

Case 2. Harmony Horizon

1 사고 개요

Harmony 블록체인의 자체 크로스 체인 브릿지인 Horizon Bridge에서 2022년 6월 발생한 침해 사고로 출금 승인에 쓰이는 멀티시그(다중서명) 검증자의 개인키가 탈취되어 약 1억 달러 규모의 자산이 탈취되었다.

7) 믹서 서비스(믹서, Mixer) : 여러 사용자의 암호화폐를 모아 뒤섞은 뒤 입금한 만큼 값으로 다시 돌려주는 서비스로 입금주소와 출금주소 간 추적을 어렵게 만드는 서비스

8) 체인 호핑(Chain Hopping) : 자금을 특정 블록체인에서 타 블록체인으로 이동시켜 추적을 어렵게 만드는 기법으로 크로스 체인 브릿지나 탈중앙화 금융 서비스(DeFi) 등을 활용

핵심 원인은 5명의 검증자 서명 중 2개의 서명만으로 출금 승인이 발생하는 낮은 임계치 설정으로 Lazarus 그룹은 해킹을 통해 획득한 2개의 키로 악의적 목적의 출금을 승인하여 자산을 탈취한다. 2023년 1월 FBI는 Lazarus 그룹을 사고 배후로 판단하였다.

2 공격 흐름

공격 흐름은 크게 5가지로 사회공학 → 악성파일 실행 → 내부 침투(검증자 키 탈취) → 위조 출금(자산 탈취) → 자금 세탁의 단계로 이루어졌다.

공격 흐름 Harmony Horizon

1단계 : 사회 공학

Lazarus 그룹은 Ronin과 동일하게 채용 담당자 등을 위장한 사회공학으로 브릿지 운영·키 관리 권한에 접근할 수 있는 내부 인력을 표적으로 초기 접근을 수행하였다. 참고로, 사고 발생 이전부터 Horizon Bridge는 검증자 키 보관 등이 중앙화되어 있어 사회 공학 기법 등에 취약하다는 우려가 있었다.

2단계 : 악성 파일 실행

표적이 된 담당자는 Ronin과 동일한 패턴으로 정상적인 채용 관련 문서 및 소프트웨어로 위장한 악성 파일을 실행하게 된다.

3단계 : 내부 침투

Lazarus 그룹은 감염된 담당자 단말을 통해 내부 시스템에서 출금에 필요한 서명키 또는 권한을 획득하게 된다.

참고로, Horizon의 서명키는 패스프레이즈[9]와 키 관리 서비스로 관리되고 있었으나 Lazarus 그룹은 장악한 단말기 및 시스템을 통해 계속해서 내부 침투를 수행하여 핫월렛이 운영되고 있던 서버를 침투하여 2개의 서명키를 확보하게 된다.

4단계 : 위조 출금

Horizon의 멀티시그 정책은 5개의 서명키 중 2개의 서명키가 서명하면 트랜잭션에 대한 유효한 서명이 발생하는 낮은 임계치를 가지고 있었고 Lazarus 그룹은 확보한 2개의 키만으로 유효한 서명을 발생시켜 자신이 통제하는 지갑 주소로 출금을 성공하게 된다.

아래는 Horizon Bridge에서 실제로 활용했던 코드의 일부이며 해당 코드를 통해 Lazarus 그룹이 서명 및 출금에 성공하게 되는지 설명한다.

1단계	내부 침투를 통해 멀티시그 서명키 중 2개의 서명키(owner 권한)을 획득

2단계	`function submitTransaction(address destination, uint256 value, bytes memory data)`
	`public returns (uint256 transactionId)`
	`{`
	`transactionId = addTransaction(destination, value, data);`
	`confirmTransaction(transactionId);  // ← 제출과 동시에 1차 확인`
	`}`
	
	(2-1) 첫 번째 owner 개인키로 submitTransaction() 함수를 호출하고, 해당 함수는 내부적으로 addTransaction()를 호출하여 transactionId 값을 지정한다.
	
	(2-2) 그 후, 인자값으로 전달된 transactionId()에 대한 confirm(확인)을 수행하는 함수인 confirmTransaction() 함수를 호출하여, Lazarus 그룹이 요청한 악의적 트랜잭션의 서명 확인 수가 +1이 된다.

3단계	2단계와 동일하게 두 번째 owner 개인키로 submitTransaction() 함수를 호출하여, transactionId() 값으로 전달된 트랜잭션의 서명 확인 수를 총 2로 만들고, executeTransaction() 함수를 호출한다.

9) 패스프레이즈(Passphrase) : 비밀번호보다 길고 복잡하게 여러 단어나 문장을 조합해 만든 인증 및 암호화 용도의 문자열로 주로 개인키 등 중요한 자격증명을 한번 더 보호하는데 사용

4단계

```
MultiSigWallet · executeTransaction
function executeTransaction(uint txId)
    public ownerExists(msg.sender)
    confirmed(txId, msg.sender)
    notExecuted(txId)
{
    if (isConfirmed(txId)) {
        Transaction storage txn =
            transactions[txId];
        txn.executed = true;

        if (external_call(txn.destination,
              txn.value, txn.data.length,
              txn.data))
            emit Execution(txId);
    }
}
```

```
MultiSigWallet · isConfirmed
function isConfirmed(uint txId)
    public view returns (bool)
{
    uint count = 0;
    for (uint i = 0;
         i < owners.length; i++) {
        if (confirmations[txId][owners[i]])
            count += 1;

        if (count == required)
            return true;
    }
}
```

(4-1) executeTransaction() 함수는 내부적으로 isConfirmed() 함수를 실행하는데, 해당 함수는 트랜잭션 서명이 완료된 횟수를 읽어, 해당 횟수가 'count == required' 즉, 임계값과 맞는 경우에 'true'를 반환한다.

- 여기서, Harmony의 경우 required 값이 "2" 였기에, Lazarus가 탈취한 2개의 서명키로 임계값을 충족시키게 된다.

(4-2) 임계값을 충족한 경우, 공격자의 지갑 주소 등이 저장된 'destination, data' 등의 변수를 전달하여 external_call() 함수를 실행한다.

5단계

```
EthManager · unlockEth
modifier onlyWallet {
    require(msg.sender == wallet,
        "not-authorized"); _;
}
function unlockEth(uint amount,
    address payable recipient,
    bytes32 receiptId)

  public onlyWallet
{
    require(!usedEvents_[receiptId], …);
    usedEvents_[receiptId] = true;

  recipient.transfer(amount);
    emit Unlocked(…);
}
```

(5-1) 최종적으로, unlockEth() 함수가 실행되어 이더리움 자산의 잠김 처리가 풀리고, 'recipient.transfer(amount)' 코드를 통해 잠김 처리가 풀린 이더리움이 Lazarus 그룹의 지갑 주소로 전송되게 된다.

Lazarus 그룹은 위 공격 패턴을 원본 체인 내 잠김 처리하고 있던 자산별 컨트랙트에게 적용하여 이더리움 외 USDT, USDC 등을 추가로 탈취하여 총 약 1억 달러 규모의 자산을 획득한다.

※ (참고) Ronin 사고와 달리 발생 당일 오전에 운영팀이 인지하면서 발각되었다.

◆ 5단계 : 자금 세탁

공격자는 BUSD·USDC·WBTC 형태의 자산을 모두 이더리움으로 스왑[10]하여 세탁 대상을 통일한 뒤 탈취 자산의 98% 이상을 Tornado Cash 믹서로 보내 자금 추적을 어렵게 만든다.

이후, 자금 추적을 회피하기 위해 약 7개월간 보유만 하다가 2023년 1월 13일 피해 금액 중 절반에 해당하는 금액을(약 6천만 달러) 이더리움 프라이버시 서비스인 Railgun[11]을 통해 세탁하고, 해당 자금은 여러 단계를 거쳐 Binance·Huobi·OKX 등의 거래소 입금 계좌로 입금되어 상당수가 비트코인으로 인출되게 된다.

Case 3. Orbit Chain

1 사고 개요

Orbit Chain은 이더리움과 다른 체인을 잇는 크로스 체인 브릿지로 2023년 12월에 공격자가 10개의 멀티시그 서명자 중 7개에 대한 접근 권한을 확보하여 약 8천 1백만 달러 규모의 자산이 탈취된 사고이다.

다만, 앞선 Ronin과 Harmony 사고와 달리 FBI 등에서 Lazarus 그룹 소행이라고 공식 판단하지는 않았으나 공격 패턴이나 자금 세탁 기법이 해당 그룹과 유사하다는 관련 업계의 판단으로 금번 보고서에 내용을 반영하였다.

10) 스왑(Swap) : 암호화폐를 다른 암호화폐로 교환하는 방식
11) Railgun : 이더리움 기반의 프라이버시 프로토콜의 한 종류로 영지식 증명(zk-SNARK)을 활용하여 거래내역(송수신자, 금액 등)을 숨기는 서비스

2 공격 흐름

현재까지 공식 침투 경로가 발표되지는 않았다. 다만, 자금 세탁까지의 기법이나 실제 출금을 승인하게 하는 기법이 Lazarus 그룹과 매우 유사하여 아래와 같은 공격 흐름으로 자산을 탈취하였을 것으로 예상된다.

공격 흐름 Orbit Chain

1단계 : 사회 공학

명확한 침투 경로가 공식 공개되지는 않았으나 내부 침투를 위해 Lazarus 그룹이 주로 이용하는 사회공학 기법을 통해 브릿지 운영·키 관리 권한을 가진 내부 인력에게 접근하여 악성파일을 실행하게 유도했을 가능성이 높다.

2단계 : 악성 파일 실행

표적이 된 담당자가 채용 관련 정상 파일로 위장한 악성파일을 실행하여 담당자 단말이 감염 및 장악되는 Lazarus 그룹의 전형적인 패턴을 활용했을 것으로 추정된다.

3단계 : 내부 침투

공격자는 감염된 단말을 통해 내부 시스템으로 접근하여 10개의 멀티시그 서명키 중 7개의 키를 확보한 것으로 추정된다.

해당 사고의 경우, 7개의 개인키가 한번에 탈취당했다는 점에서 서명키가 모두 또는 7개 이상이 동일한 시스템 또는 네트워크 영역에 저장 및 표적이 된 담당자 또는 특정 직원이 해당 키들에 대한 접근 권한을 대부분 보유하고 있어 멀티시그가 사실상 무력화 되었다고 판단된다.

또한, Orbit 사고의 경우 전직 직원이 방화벽을 일부러 취약하게 설정하였다는 의혹이 있어 공격자 또는 공격 그룹이 사회 공학 기법과 더불어 중요 권한을 보유한 내부 인력을 매수했다는 의혹도 있다.

4단계 : 위조 출금

공격자는 확보한 7개 키로 멀티시그 승인 요건을 충족시켜 자신이 통제하는 주소로 출금을 승인하게 된다. 탈취된 자산은 이더리움 등 다양한 자산으로 교환된 후 다수의 지갑 주소로 분산하여 전송된다.

※ (참고) Orbit 측은 사고 인지 후 즉시, 서비스를 중단하였지만 이미 탈취된 자산이 전체 예치 자산의 절반 이상이었다고 알려져 있다.

5단계 : 자금 세탁

공격자는 Tornado Cash 믹서를 통해 수수료(가스비)부터 세탁하였고 탈취 직후에는 거래소 또는 금융당국 등에서 추적에 집중하므로 상당 기간 자산을 움직이지 않은 채(unmoved) 다수 주소에 분산하여 보관하는 등 추적이 집중되지 않도록 하였다.

다만, 현재까지 Orbit에서 탈취된 자산에 대한 현금화 기법에 대한 공식적인 결과는 존재하지 않는다.

Case 4. KelpDAO

1 사고 개요

2026년 4월에 발생한 보안 사고로 'Lock & Mint' 방식 등에 비해 안전하다고 알려져 있던 'Hybrid Burn & Mint' 방식을 활용하던 브릿지에서 발생하였다. 피해 업체는 KelpDAO이며 KelpDAO는 고객이 맡긴 이더리움을 재스테이킹해 수익을 발생시키고 그 수익으로 rsETH를 발행하여 전달하였다. 여기서, rsETH를 타 체인에서 사용하게 하기 위해 'Hybrid Burn & Mint' 방식의 LayerZero 브릿지를 활용하였다.

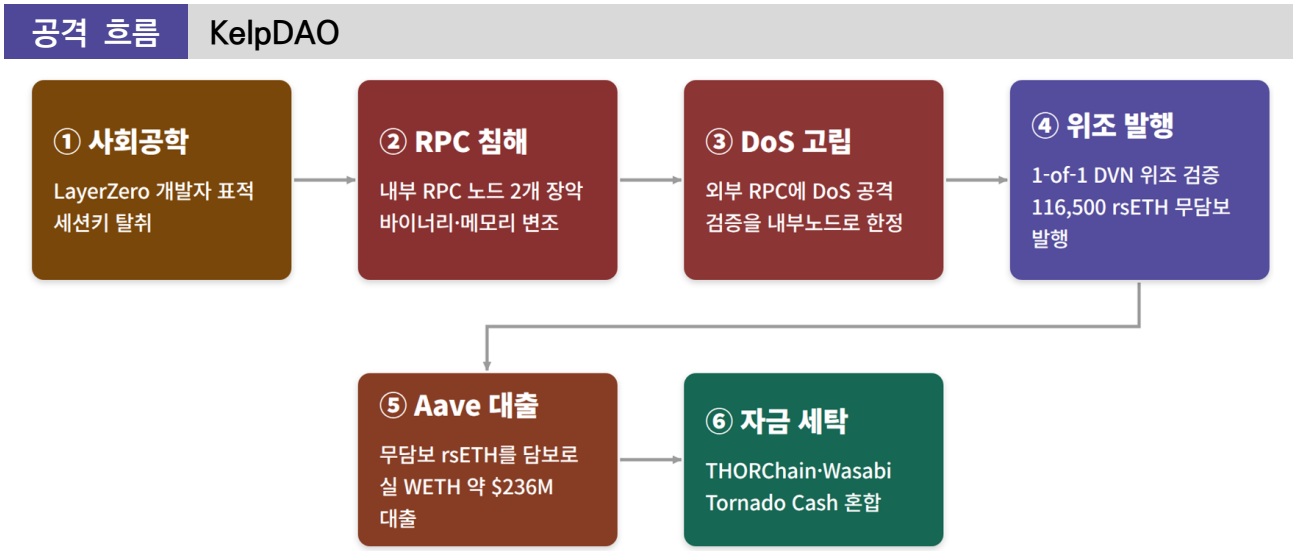
여기서, 위 사고들과의 차이점은 검증자 노드 및 검증용 서명키가 탈취되어 발생된 것이 아닌 'Hybrid Burn & Mint' 방식에서 검증에 이용되는 메시지를 위조하기 위해 RPC 노드를 침해하여 자산을 탈취하였다는 점이다. 이를 통해, 116,500개의 rsETH(약 2억 9천 2백만 달러)를 무단 발행하여 탈취하게 된다.

참고로, 사고의 원인인 RPC 노드에 높은 권한(예. root)으로 어떻게 접근했는지에 대해서는 공개된 바 없지만 일부 분석가들은 보고되지 않은 LayerZero 측의 취약점을 통해 이미 침해가 당했던 상태이거나 내부자 공모 등의 원인을 제기하고 있다.

※ (참고) KelpDAO에서는 사고의 직접적인 원인을 LayerZero의 안전하지 않은 DVN 구성(단일 구성)으로 판단하였으며, LayerZero 측은 사고 이후 DVN을 다중 검증자 구성으로 변경하는 등의 조치를 수행하였다.

2 공격 흐름

실제로 Lazarus 그룹은 위 사고 사례에 비해 고도화된 공격 기법을 활용하였다. 초기 사회공학 기법을 통한 내부 인프라 접근 권한 획득은 동일하나 이후, RPC 인프라 침해 후 바이너리 변조, DoS 공격 등 이전에 수행하지 않던 방식의 공격 기법을 추가로 활용하였다.



1단계 : 사회 공학

최초 공격은 이전 사고와 동일하게 사회 공학 기법을 통해 LayerZero Labs 개발자가 보유한 클라우드 인프라 접근 권한 세션 키를 획득하면서 시작되었다. 특히, Lazarus 그룹은 해당 크로스 체인 연계 방식의 특성을 파악하여 브릿지나 스마트 컨트랙트를 직접 운용 및 관리하는 직원이 아닌 검증용 인프라(DVN)를 운영하는 회사의 개발자를 표적으로 삼았다.

2단계 : RPC 침해

Lazarus 그룹은 사회 공학 기법으로 사전에 탈취한 접근 권한(세션 키)을 이용하여 LayerZero Labs의 클라우드 인프라에 접근 후 DVN 검증에 직접적으로 활용되는 내부 RPC 노드 목록을 파악하고 해당 노드에서 실행되고 있는 바이너리 변조를 시도한다.

여기서, 변조 대상이 된 바이너리는 RPC 노드에서 블록체인 상태(트랜잭션, 로그 등)를 읽고 응답하는 'op-geth' 바이너리로 Lazarus 그룹은 총 2개의 RPC 노드에 침투하여 'op-geth' 바이너리 파일을 변조된 바이너리로 교체한다. 변조된 바이너리는 DVN 검증 노드에게만 변조된 응답 값을 전달하게 하고 나머지 서버(모니터링 서버 등)에 대해서는 이전과 같이 정상 응답을 보내게 만든다.

다만, 변조된 바이너리 파일은 'self-destruct'[12] 기법을 통해 목적을 달성한 후 삭제되어 찾을 수 없지만 아래의 예시 코드와 같이 바이너리를 변조하지 않았을까 하는 추정을 할 수 있다.

바이너리 변조 코드(예시)

```
on rpc_request(method, params, caller):
    if caller is LayerZero_DVN and method in RECEIPT_METHODS:
        return FORGED_RECEIPT    # 가짜 burn 이벤트
    else:
        return REAL_RESPONSE     # 정상 데이터 (탐지 회피)
```

위 코드의 경우 RPC 노드로 요청을 주는 주체(caller)가 'LayerZero_DVN'인 경우에는 실제 발생하지 않은 위조된 Burn 이벤트를 반환하고 그 외 응답 요청에 대해서는 정상 데이터를 반환하도록 변조한 코드의 예시이다.

3단계 : DoS 공격

2단계를 통해 Lazarus 그룹은 DVN 검증 노드가 내부에서 활용하는 RPC 노드의 바이너리는 변조하였으나 DVN 검증 노드의 경우 내·외부의 RPC 노드를 통해 검증을 수행하도록 구성되어 있었다.

이에, DVN 검증 노드가 외부 RPC 노드와 통신 후 검증하지 못하도록 Lazarus 그룹은 DVN 노드가 참조하고 있는 외부 RPC 노드 제공자에 DoS 공격을 수행하여 통신 자체가 발생하지 않도록 하였고 이로 인해 DVN 검증 노드는 바이너리가 변조된 내부 RPC 노드로만 통신을 수행하게 된다.

12) 자가 삭제(self-destruct) : 악성 파일이 목적을 달성 후 수사기관의 분석을 피하기 위해 의도적으로 스스로 실행 파일, 관련 레지스트리 항목 및 로그 등을 삭제하는 기법

4단계 : 위조 발행

위 단계를 통해 이미 조작된 환경에서 위조된 크로스 체인 메시지에 대해 유효한 서명(attestation)이 생성되었고 단일 DVN 검증 체계로 인해 목적지 스마트 컨트랙트에서 rsETH를 발행하게 된다. 결과적으로 고객(사용자)이 실제 예치한 ETH등을 담보로 하지 않는 무담보의 rsETH(116,500개)가 이더리움 메인넷에서 발행되게 된다.

5단계 : 무담보 대출(실자산 인출)

Lazarus 그룹은 무한 발행한 rsETH를 바로 현금화하지 않고 탈중앙 대출 프로토콜인 Aave에 해당 rsETH를 담보로 예치하고 예치한 수량만큼 WETH(이더리움 랩핑 토큰)를 대출하게 된다.

이를 통해, Lazarus 그룹은 Aave[13]에서 52,834개의 WETH 및 추가로 Arbitrum[14]에서 29,782개의 WETH 등을 대출해 총 약 2억 3천 6백만 달러 상당의 실 자산을 획득하게 된다.

※ (참고) 탈취된 자산을 통한 대출로 인해 Aave는 약 1억 9천만 달러 상당의 부실채권이 쌓이게 된다.

6단계 : 자금 세탁

해당 사고를 Lazarus 그룹의 소행이라고 판단하는 이유가 자금 세탁 단계에서도 드러나게 된다. Orbit 사고와 동일하게 공격 전 Tornado Cash 믹싱 서비스를 통해 수수료를 세탁하고 이후 탈취된 자산을 THORChain[15]과 Umbra[16] 서비스 및 Wasabi, Tornado Cash 등의 믹서 서비스를 활용하게 된다. 해당 과정을 통해 탈취된 자산 중 대부분인 약 2억 2천만 달러 규모를 세탁하고 실제 회수가 가능한 자산은 7천 1백만 달러만 남게 된다.

13) Aave : 이더리움 등 블록체인 위에서 동작하는 대표적인 탈중앙화 대출 프로토콜
14) Arbitrum : 이더리움 기반의 레이어2 확장 솔루션 중 하나
15) THORChain : KYC를 수행하지 않고 서로 다른 체인 간 자산을 직접 교환해주는 탈중앙화 크로스 체인 스왑 프로토콜
16) Umbra : 받는 지갑 주소(수신자)를 감추는 프라이버시 프로토콜의 종류

3. 공격 기법 상세 분석

Lazarus 그룹의 소행이 확실 또는 추정되는 4가지의 사고 사례를 통해 주로 수행하는 공격 및 자금 세탁 기법을 간단히 파악할 수 있었다.

이번 장에서는 Lazarus 그룹이 수행한 공격 기법에 대해 상세 분석하여 설명하고 MITRE ATT&CK 매트릭스[17] 형태로 정리하였다.

핵심부터 설명하면 Lazarus 그룹의 경우 크로스 체인 서비스 인프라 담당자와 같이 사람을 먼저 표적으로 정하여 "출금 승인 등에 관련된 권한을 탈취 및 획득" 하고 "해당 권한을 활용하여 정상 거래로 위장한 악의적 목적의 자산 탈취를 수행" 후 이에 대한 추적을 막기 위해 "믹서 서비스를 활용한 자금 세탁"이 핵심이다.

3.1. 사회 공학

참고 MITRE ATT&CK 매트릭스(사회 공학)

Reconnaissance 정찰	Resource Development 자원 개발	Initial Access 초기 침투
T1589 **피해자 신원 정보 수집** 직무·연락처·계정 파악	T1585 **가짜 계정 생성** 사칭을 위한 프로필 제작	T1566 **피싱 (스피어피싱)** 채용 미끼 파일·링크 전달
T1591 **피해자 조직 정보 수집** 채용 관련 플랫폼(LinkedIn 등)에서 표적 엔지니어 물색	T1586 **계정 탈취·사칭** 실제 채용담당자 계정 도용	
	T1587 / T1588 **악성 도구 제작·확보** 트로이목마 SW·RAT 준비	

17) MITRE ATT&CK 매트릭스 : 실제 공격자들이 사용하는 공격 목적 및 기법을 체계적으로 분류하여 정리한 프레임워크

4가지 사고 사례를 통해 Lazarus 그룹이 공격의 첫 단계로 활용하는 기법임을 알 수 있다. Lazarus 그룹은 해당 업체에 재직 중인 직원들에 대해 불특정하게 공격하는 게 아닌 업무 영역 등을 사전에 면밀히 파악하여 실제적으로 접근 권한을 가질만한 직원을 표적으로 삼는다. 주로 검증 인프라 접근 및 키 관리 권한을 가진 엔지니어나 운영자가 표적이 되는 이유이다.

Lazarus 그룹은 사회 공학 기법 중 주로 채용 위장(Operation Dream Job) 기법을 사용하였다. 높은 연봉 및 직급 등을 제시하며 화상 면접의 형태로 유도한다.

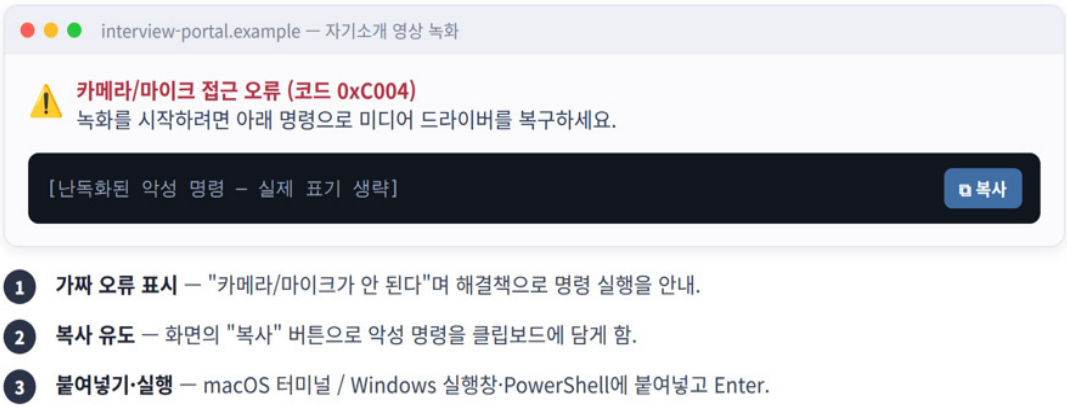
악성 파일 실행 유도

화상 면접 유도	화상 면접을 가장하여 엔지니어에게 악성 행위를 심어놓은 가짜 화상회의 애플리케이션 파일을 전달하고 실행하게 한다.

하지만 위 기법 같은 경우에는 피해자 즉, 엔지니어가 직접 파일을 실행하여야 하는 단점이 존재하였다. 여기서 말하는 단점은 예를 들어, 파일을 전달받은 엔지니어가 실행하지 않고 백신 프로그램 등에서 먼저 탐지하는 경우 공격이 실패할 수 있게 때문이다.

이에, Lazarus 그룹은 'ClickFix' 기법을 활용하게 된다. 'ClickFix' 기법은 첨부파일을 다운로드하거나 실행하는 구조가 아닌 사용자가 직접 명령어를 실행하게 하는 구조로 악성 파일을 감염시키는 기법이다.

예시 ClickFix 기법



이와 같이, Lazarus 그룹에서 사회 공학 기법을 공격의 첫 단계로 활용하는 이유는 명확하다. 인간이 지니고 있는 더 많은 연봉 및 권력 등의 욕구를 노리고 신뢰를 쌓아나가면서 수행하는 방식은 내부에서 기술적으로 구현한 보안 솔루션 등을 통해서는 막기 어렵기 때문이다.

3.2. 단말기 장악

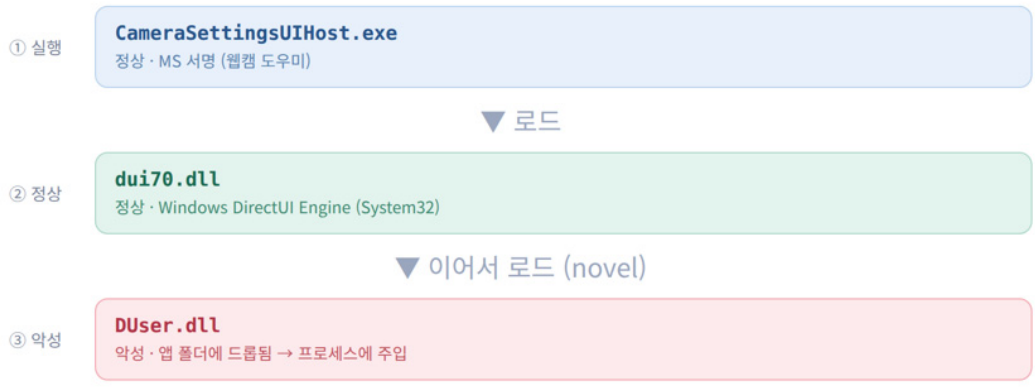
참고 MITRE ATT&CK 매트릭스(단말기 장악)

Lazarus 그룹은 내부 시스템 침투를 위해 주로 정상 채용 파일로 위장한 PDF 형태의 악성 파일을 활용하였고 해당 악성 파일을 통해 최초 침투한 엔지니어 등 담당자의 단말기를 장악하였다.

악성 DLL 로드(DLL Search-order Hijacking)

Lazarus 그룹은 단말기 장악을 위해 악성 DLL[18]을 로드하는 'DLL Search-order hijacking'[19] 기법을 활용한다. 해당 기법은 디지털 서명까지 완료된 정상 실행파일을 전달하고 해당 실행 파일이 악성 DLL을 로드하는 방식이다. 이렇게 활용하는 이유는 디지털 서명까지 완료된 실행파일의 경우 일반 악성 파일에 비해 백신 및 EDR의 탐지를 피할 가능성이 높기 때문이다.

이해를 돕기 위해, Lazarus 그룹이 가상자산 탈취를 목적으로 제작하여 활용하였던 악성 파일을 대상으로 설명한다.

실제 사례	BloxHolder(2022.06. Lazarus 그룹 소행)
설명	먼저, 가상자산 자동매매 플랫폼을 위장한 사이트에서 윈도우 설치파일(MSI)를 다운로드하고 실행하게 만든다. 피해자가 해당 설치 파일을 실행하는 경우 디지털 서명이 완료된 정상 실행 파일(CameraSettingsUIHost.exe)과 악성 DLL(DUser.dll)을 같은 폴더 내 드랍하게 된다. 실행된 정상 파일은 윈도우 System32 폴더 내 정상 DLL(dui70.dll)을 로드하는데 해당 DLL이 추가로 필요한 DLL 호출 시 같은 폴더에 있는 악성 DLL을 먼저 호출하게 하는 방식이다. (참고) Windows의 경우 일반적으로 DLL을 찾을 때 정해진 검색 순서가 있는데 보통은 실행 파일이 있는 폴더를 System32보다 먼저 검색을 하게 된다. 따라서, 정상 DLL이 악성 DLL을 호출하는 과정에서 정상 DLL이 실행 파일이 있는 폴더 내 정상 DLL 파일과 이름만 같은 악성 DLL을 먼저 호출해서 로드하게 되는 것이다. 

18) DLL(Dynamic Link Library) : 프로그램이 사용하는 기능 및 코드 등을 구현한 파일로, 윈도우 실행 파일(exe)이 구동될 때 로드되어 활용

19) DLL Search-order hijacking : 정상 DLL과 동일한 이름의 악성 DLL을 먼저 검색되는 폴더에 다운로드하여, 정상 실행 파일이 악성 DLL을 대신 로드하게 만드는 기법

메모리 로드 및 암호화

이렇게 로드된 악성 DLL은 실제 C2 서버에게 단말기 정보를 전달하고 추가 악성 파일을 다운로드하게 된다. 해당 과정을 위해 암호화된 페이로드를 활용하는데 이때 암·복호화키를 메모리상에서만 노출하여 활용함으로써 추후 분석을 어렵게 만든다.

이해를 돕기 위해, 실제로 Lazarus 그룹이 활용하였던 악성 파일을 기준으로 예시 코드를 작성하였다.

예시 코드 **암·복호화 관련 코드**

설명

```
void DllMain(args):
    key = args[0]
    blob = embedded_resource(mi.dll)

    payload = AES128_CBC_decrypt(blob, key, iv)

    run_in_memory(payload)
    wipe(key); wipe(payload)
```

(1) 복호화 키를 실행 인자값으로만 호출 (key=args[0])
(2) 암호화된 페이로드를 복호화 키를 통해 메모리에서만 복호화 수행
(payload = AES_128_CBC_decrypt(blob, key, iv))
(3) 복호화된 페이로드를 디스크에 쓰지 않고 메모리에서만 로드
(run_in_memory(payload))
(4) 복호화에 사용되었던 키 및 페이로드 값을 삭제 (wipe(key); ,wipe(payload))

악성파일 다운로드

악성 DLL의 동작으로 담당자의 단말기에서 C2 서버에 접근하여 악성 파일 다운로드를 시도하는데 해당 단계에서 네트워크 트래픽 탐지를 피하기 위해, Lazarus 그룹은 GitHub, Microsoft OneDrive와 같이 일반적으로는 정상적으로 활용되는 클라우드 서비스를 주로 활용하였다. 이는 일부 네트워크 보안 장비에서는 GitHub이나 Microsoft에 접속하는 트래픽을 정상으로 판단하여 차단하지 않기 때문이다.

단말기 장악

악성 파일의 유형이 다양하나 Lazarus 그룹은 주로 원격 제어를 할 수 있는 악성 파일을 활용하였다. 다운로드 된 악성파일은 피해자가 직접 실행하지 않더라도 자동으로 실행되어 Lazarus 그룹이 해당 단말을 원격으로 조종 가능한 상태로 만든다.

이를 통해, 명령 실행, 파일 업로드·다운로드, 키로깅, 화면 캡처, 자격증명·세션키 탈취, 추가 도구 설치 등 다양한 공격이 가능해진다.

또한, Lazarus 그룹은 담당자 단말기가 재시작되어 메모리에 로드된 악성파일이 동작하지 않게 되는 경우를 고려하여 추가적으로 레지스트리 자동 실행 영역에 키값을 추가하여 단말기가 재시작되어도 계속해서 악성 파일이 실행되게끔 하였다.

참고 **악성파일 실행을 통한 단말기 장악 흐름도**

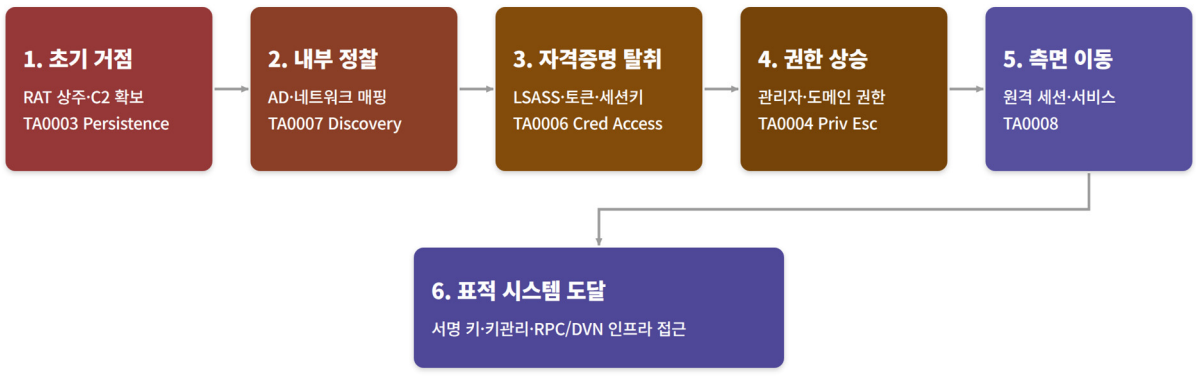
3.3. 내부 침투

참고 MITRE ATT&CK 매트릭스(내부 침투)

Credential Access 자격증명 접근	Discovery 내부 정찰	Privilege Escalation 권한 상승	Lateral Movement 측면 이동
T1003 **OS 자격증명 덤프** 인증정보 탈취	T1087 **계정 탐색** 권한 보유 계정 식별	T1068 **취약점 악용 권한상승** 관리·콘솔 권한 확보	T1021 **원격 서비스 이용** 정상 원격 경로로 이동
T1528 **앱 액세스 토큰 탈취** 클라우드 토큰 확보	T1018 **원격 시스템 탐색** 관리 서버·노드 위치 파악	T1078.004 **클라우드 유효 계정** IAM 역할·토큰 악용	T1078 **유효 계정 재사용** 정상 로그인으로 시도
T1539 **웹 세션 쿠키 탈취** MFA 우회	T1482 **도메인 신뢰 탐색** AD 구조·신뢰관계 매핑		→ 최종 도달 **검증 키·노드 권한 확보** 서명키 또는 검증 인프라 장악
T1552 **보호 안 된 자격증명** 중요 키에 대한 분산 저장 미흡			

해당 공격 기법의 목적은 실제 자금 인출 및 발행에 관여하는 서명 키 획득 및 관련 인프라를 장악하기 위함으로 Lazarus 그룹은 'Lateral Movement'[20](측면 이동) 침투 기법을 주로 활용했을 것으로 추정된다.

참고 Lazarus 그룹의 내부 침투 방식



20) Lateral Movement(측면 이동) : 공격자가 최초로 침투한 시스템에서 머무르지 않고, 다른 시스템으로 계속해서 옮겨가며 최종 목표에 도달해가는 공격 기법

초기 거점 확보(Foothold)

Lazarus 그룹의 특징은 초기 악성파일 감염 등을 통한 거점을 확보한 후에는 악성 파일 실행을 최소화하고 운영체제에 내장되어 있는 정상 소프트웨어를 악의적으로 활용해 침투 범위를 확대하는 'LotL(Living off the Land)' 기법을 활용한다는 점이다.

이러한 방식을 Lazarus 그룹이 활용하는 이유는 디지털 서명까지 완료된 정상 도구를 공격 목적으로 활용하는 경우 악성 파일 대비 내부 보안솔루션(백신 등)에서 탐지될 가능성이 낮아지기 때문이다.

내부 정찰(Discovery)

장악한 단말을 통해 Lazarus 그룹은 주로 도메인/액티브 디렉터리 구조 열람, 네트워크·호스트·공유 폴더 식별, 권한 있는 계정과 관리 서버 위치 파악 등을 수행한다. 이 과정에서도, LotL 기법을 활용함으로써 탐지를 어렵게 만든다.

자격증명 확보(Credential Access)

내부 정찰의 주된 목적은 주요 자격 증명 저장소 접근 및 탈취에 있다. 이에, Lazarus 그룹은 메모리 영역에 남아 있는 인증 관련 데이터, 브라우저 및 캐시 폴더 등에 저장된 토큰 및 세션 쿠키, 키체인 및 자격 증명 저장소 접근 등을 시도한다. 대표적 예시로는 KelpDAO 사고 사례에서 'LayerZero 개발자의 세션 키' 탈취가 있다.

예시 **주요 자격 증명**

자격 증명	공격 사유
메모리 내 인증 데이터	로그인 시 평문 또는 해시값의 형태로 메모리에 남아 있는 인증 데이터로 공격자가 추가 인증 없이 계정 탈취를 시도할 때 활용
세션 쿠키 / 토큰	이미 인증이 완료된 상태의 데이터로 공격자가 비밀번호 및 추가 인증 없이 세션 토큰 재사용 공격 등을 통해 로그인을 시도할 때 활용
키체인 / 자격 증명 저장소	Mac OS의 Keychain 이나 Windows Credential Manager 등 운영체제에서 비밀번호 또는 인증값 등을 모아 보관하는 저장소로, 공격자가 한 번의 접근으로 다수의 접근 권한을 획득하고자 할 때 활용하는 저장소

자격 증명	공격 사유
SSH 키 등 원격 접속 자격	중요 시스템(서버 등)에 접근하여 로그인 할 수 있는 원격 접속 인증 파일로 공격자가 측면 이동(Lateral Movement) 기법 활용
개발자 서명키	스마트 컨트랙트 검증, 배포, 서명 등 직접적인 자산 이동과 관련된 파일로 공격자가 위조 트랜잭션 서명 등의 최종 목표를 위해 활용

권한 상승(Privilege Escalation)

장악된 내부 시스템에서 관리자 권한과 같이 중요 서버 또는 데이터에 추가적으로 접근하기 위해 권한 상승을 시도한다. Lazarus 그룹은 탈취한 관리자 계정 재사용, 미패치 취약점 악용, 잘못 설정된 권한 및 서비스를 통한 계정 탈취 등의 공격을 수행한다.

클라우드 환경에서는 주로 과도하게 부여된 IAM 역할 및 서비스 계정을 확보해 최종적으로 인프라 콘솔 접근 권한을 획득하려고 시도한다.

참고 **권한 상승 기법(실제 사례)**

공격 기법	내용
제로데이 공격	Lazarus 그룹이 2024년에 가상자산 개발 또는 거래소 관련 엔지니어를 표적으로 삼아 활용했던 기법으로 최초 일반 사용자 권한 획득 후, 'AFD.sys'라는 윈도우 드라이버 파일의 제로데이 취약점(CVE-2024-38193)을 통해 단말기 내 SYSTEM 권한을 획득

측면 이동(Lateral Movement)

Lazarus 그룹은 위 단계를 통해 확보한 자격 증명으로 계속해서 시스템을 이동한다. 이때 주로 활용하는 방식은 시스템 내 취약점을 직접적으로 활용하기보다 이미 획득한 자격 증명으로 "정상적인 원격 접속 및 관리 경로를 그대로 쓴다"라는 부분이다.

즉, 내부 직원이 정상적인 경로를 통해 로그인하는 것처럼 위장해서 시스템으로 침투하게 되는 것이다. 따라서 내부 보안 솔루션 및 정보보안팀에서는 "정상 트래픽"으로 판단하여 탐지가 어렵게 된다.

참고 **측면 이동 기법(Lazarus 그룹)**

공격 기법	내용
탈취한 자격 증명 활용	Lazarus 그룹은 공격 목표를 달성하기 위해, 이미 확보한 자격증명을 활용하여 윈도우에서 제공하는 정상 서비스인 원격 데스크톱 서비스(RDP)를 통해 다른 윈도우 단말기 또는 서버 등으로 측면 이동하며 침투 영역을 확대하는데 활용한다.

표적 시스템 도달(Objective)

브릿지 시스템과 같이 대규모의 인프라를 운용하는 회사의 네트워크는 매우 복잡하고 위 과정을 1번만 수행해서 서명키나 검증에 활용되는 노드 서버를 바로 파악하고 탈취하는 것은 불가능에 가깝다.

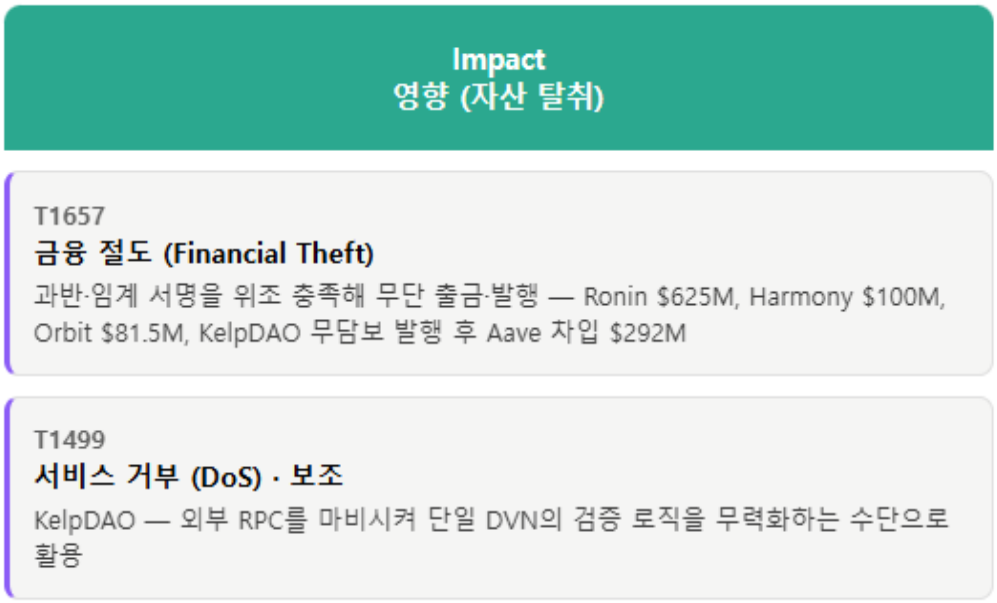
이에, Lazarus 그룹은 내부 정찰, 자격증명 탈취, 권한 상승, 측면 이동을 계속해서 반복하여 최종 목적지인 검증에 활용되는 서명키가 보관된 저장소 또는 서버 등에 접근하게 된다.

참고 **표적 시스템 분류(예시)**

표적 시스템	내용
서명 권한 보유자 단말	멀티시그 서명자, 검증 노드 운영자, 핵심 개발자 등 서명 및 승인 권한을 직접 보유한 인원의 단말기로, 해당 단말을 장악할 경우 서명 및 승인 권한을 확보할 수 있어 가장 우선이 되는 표적이다.
서명키 보관소	개인키 또는 서명키가 실제로 저장 및 운용되는 핫월렛 서버, HSM(하드웨어 보안 모듈), 키 관리 시스템(KMS), 자격증명 저장소 등을 말하며 자산 이동에 필요한 키가 직접 보관된 서버 또는 저장소로서 탈취 시 서명 및 출금 권한 확보로 이어질 수 있어 '서명 권한 보유자 단말'만큼 우선이 되는 표적이다.
검증 노드 및 검증 인프라	트랜잭션의 유효성을 검증하는 검증자 노드, 외부 데이터를 호출하는 오라클, 트랜잭션의 결과 값을 송수신하는 RPC 노드 등 서명 및 승인에 직접 활용되지 않더라도 검증 결과나 그 입력 데이터를 조작함으로써 실제로 발생하지 않은 트랜잭션을 정상 승인되도록 유도할 수 있어 Lazarus 그룹에 주된 표적이 된다.
클라우드 및 인프라 관리 콘솔	과도하게 권한이 부여된 IAM 계정을 탈취하여 클라우드 관리 콘솔에 접근하는 경우, 인프라 전반을 확인 및 통제할 수 있어 서명키 보관소 또는 검증 노드에 접근하는 원인을 제공하므로 Lazarus 그룹이 초기 침투 목적으로 주로 접근하는 표적이다.

3.4. 위조 출금

참고 MITRE ATT&CK 매트릭스(위조 출금)



해당 공격 기법은 실제로 공격에 활용하는 기법이 아닌 이미 획득한 서명키 또는 검증에 활용되는 노드를 이용하여 악의적 목적의 출금을 승인하여 자산을 탈취하는 기법이다.

다만, KelpDAO 사고의 경우에는 앞선 설명한 바와 같이 서명키 자체를 탈취한 것이 아닌 검증 메시지를 송수신하는 RPC 노드를 해킹함으로써 위조 출금 등이 가능해진 사고이다.

4. 자금 세탁 기법 상세 분석

이번 장에서는 무단 발행 등을 통해 탈취가 완료된 자산의 추적을 피하기 위해 Lazarus 그룹이 주로 활용한 자금 세탁 기법을 설명한다.

참고 자금 세탁 단계(Lazarus 그룹)

❶ 사전 준비

Lazarus 그룹은 공격을 수행하기 이전부터 공격에 활용하기 위한 지갑의 수수료(가스비)를 자신의 지갑에서 직접 이동하여 처리하는 경우 추적의 근거가 될 수 있으므로 미리 믹서 서비스를 거친 출처가 분명하지 않은 자산으로 채워 지갑부터 익명화를 시도한다.

❷ 세탁 단위 통일

추적이나 동결이 쉬운 스테이블코인이나 또는 현금화가 어려운 일반 자산 등을 1개 또는 2개의 자산(이더리움 등)으로 환전해 세탁 단위를 통일하고 환전 시 KYC[21]가 없는 탈중앙 거래소(DEX)를 이용한다.

21) KYC(Know Your Customer) : 금융회사 또는 거래소가 고객의 신원(실명, 신분증 등 활용)을 확인해 본인임을 검증하는 절차로, 자금세탁 등의 범죄 방지 용도로 활용하는 제도

❸ 추적 회피

자금 세탁에 가장 중요한 단계이다. 블록체인의 특성상 모든 거래가 블록체인 내 영구적으로 기록되므로 이론상 탈취된 자산에 대해서도 지갑 주소 트랜잭션 기록만 계속 추적하면 범죄 주체를 찾을 수 있다.

따라서, Lazarus 그룹과 같은 공격자는 탈취 된 자산과 현금화하는 자산 간의 추적을 어렵게 만드는 것이 주요 목표가 된다. 이 단계에서 주로 이용하는 방법이 바로 믹서 서비스, 체인 호핑, 프라이버시 프로토콜[22] 활용이다.

믹서 서비스는 기본적으로 믹서 서비스에 입금한 코인을 타인의 코인과 섞어서 최종적으로 입금하였던 코인의 가치만큼 반환하는 서비스로 추적하는 입장에서는 공격자의 특정 지갑에서 믹서 서비스까지 이동한 기록은 추적이 쉬워도 믹서 서비스에서 어느 지갑으로 공격자가 탈취한 코인이 이동했는지에 대한 추적은 어려워진다.

참고로, Lazarus 그룹은 초기에는 Tornado Cash와 같은 믹서 서비스와 탈중앙화 금융 서비스(DeFi)[23] 및 크로스 체인 브릿지를 통한 체인 호핑을 주로 이용하였으나, 최근 발생한 KelpDAO에서는 비트코인 기반의 믹서 서비스인 Wasabi CoinJoin을 추가로 사용하는 등 이중 믹서 서비스 활용 및 프라이버시 프로토콜 서비스를 추가로 활용하는 등 자금 세탁 기법을 고도화하고 있다.

❹ 분산 및 은닉

Lazarus 그룹은 자산 탈취가 식별되는 경우 피해 업체 및 수사기관 등에서 초기에는 추적을 집중하나 시간이 지날수록 추적이 어려워짐을 이용하여 믹서 서비스 등을 거친 자산을 바로 현금화하려고 시도하지 않고 소액으로 분할 하여 다수의 지갑에 장기간 보관한다.

---

22) 프라이버시 프로토콜(Privacy Protocol) : 받는 사람(수신자)의 실제 지갑 주소를 은닉해주는 서비스

23) DeFi(Decentralized Finance, 탈중앙화 금융 서비스) : 은행 등 중앙 기관 없이, 스마트 컨트랙트가 자동으로 거래를 처리하는 금융서비스로, 자산 교환(DEX), 대출, 예치 등의 서비스를 제공하며 신원 확인(KYC) 절차가 없어 공격자가 주로 활용

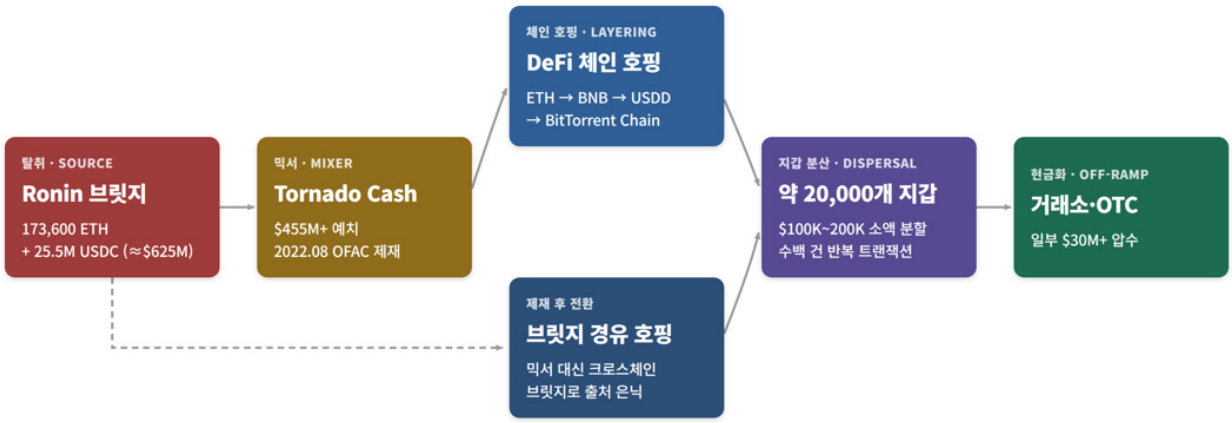
❺ 현금화

최종적으로 실제 Lazarus 그룹과 같은 공격자는 탈취한 자산에 대한 현금화를 목표로 한다. 즉, 탈취한 자산을 지갑에서 계속 보관만하는 것은 실제 현금화가 되지 않아 이득이 될 수 없으므로 탈취한 자산을 달러 등과 같은 법정화폐로 전환하려 한다.

이 과정에서의 핵심은 OTC[24]를 통한 현금화(off-ramp)[25]이다. 여기서, OTC는 원래 매수 또는 매도 금액이 큰 개인 간(또는 브로커를 통한) 거래를 위한 정상 서비스이나 거래소를 통하지 않는다는 부분에서 KYC 절차가 수행되지 않기에 Lazarus 그룹 등 공격자가 주로 활용한다. 특히, Lazarus 그룹은 탈취 자산을 소액 단위로 분할하여 여러 브로커를 통해 환전함으로써 추적을 더욱 어렵게 만들었다. 이렇게, OTC를 통해 현금화(off-ramp) 되는 경우 추적은 사실상 불가능해진다.

앞서 설명한 자금 세탁 기법을 보다 구체적으로 이해할 수 있도록, 실제 발생한 Ronin·Harmony 사고 사례를 분석하여 설명한다.

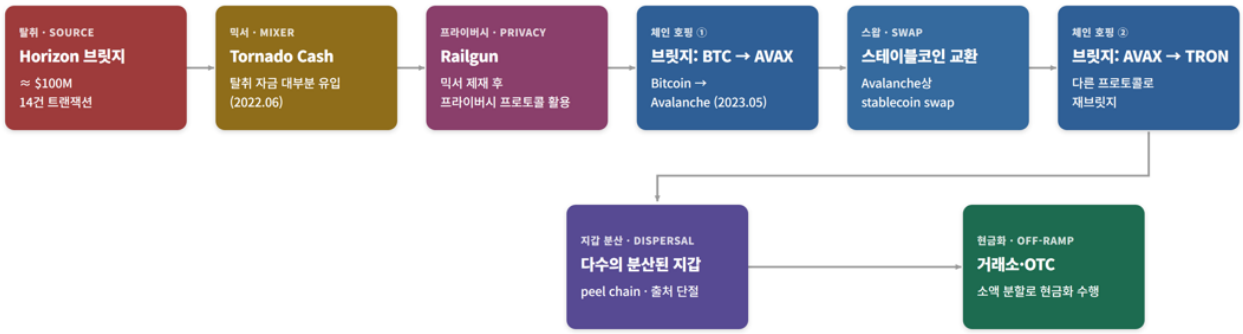
사례 분석 Ronin



24) OTC(Over-The-Counter, 장외거래) : 거래소의 거래 서비스를 활용하지 않고, 당사자끼리(혹은 브로커를 통해) 큰 금액으로 거래하는 방식으로, 대규모 구매자 또는 판매자가 시세에 영향을 최소화하여 거래할 때 쓰는 방식
25) off-ramp : 암호화폐를 법정화폐로 변환하는 방식

1단계 (믹싱)	Lazarus 그룹은 탈취 직후 자금의 상당 부분을 Tornado Cash 믹서로 보내 세탁한다. 이후, 미국은 Ronin에서 탈취된 대부분의 자산(약 4억 5,500만 달러 이상)이 Tornado Cash를 통해 세탁된 점을 통해 해당 믹서를 2022년 8월 제재 대상으로 지정한다.
2단계 (체인 호핑)	Tornado Cash 믹서의 제재 이후, Lazarus 그룹은 믹서를 통해 세탁된 자산을 DeFi 서비스 및 크로스 체인 브릿지를 활용한 체인 호핑을 통해 USDD(달러 기반 스테이블코인의 한 종류)로의 스왑 및 타 블록체인(BitTorrent 등)으로 옮기는 등 추적을 어렵게 만들었다.
3단계 (지갑 분산)	2단계를 통해 세탁된 자금은 다수의 지갑으로 분할되어 전달되었는데 그 숫자가 약 2만개에 달한다. Lazarus 그룹은 한 지갑당 10만~20만 달러 수준의 소액으로 반복된 트랜잭션을 발생시켰는데, 이를 peel-chain 기법이라 한다. 이러한 소액 전달은 수사기관 등이 대규모 자산이 이동하는 단일 트랜잭션에 대해 집중적으로 탐지하는 부분을 회피하기 위함으로 보인다.
4단계 (현금화)	위 단계를 통해 최종적으로 여러 지갑에 분산된 자금 중 일부는 거래소를 통한 현금화를 하였으나, 대부분은 OTC 네트워크를 통해 현금화(off-ramp) 한 것으로 분석되었다.

사례 분석 Harmony



1단계 (믹싱)	Ronin 사고와 마찬가지로, 탈취한 자금의 대부분을 Tornado Cash 믹서로 보내 세탁한다. 해당 믹서가 제재 대상으로 지정된게 22년 8월 이기에, Lazarus 그룹이 6월까지는 해당 믹서를 활용한 것으로 분석되었다. Tornado Cash 믹서 제재 후 에는 Railgun이라는 프라이버시 프로토콜을 활용하여 자금을 세탁한다.
2단계 (체인 호핑)	이후 크로스 체인 브릿지 체인 호핑을 통해 비트코인 블록체인에서 아발란체(Avalanche) 블록체인으로 이동시킨 후 아발란체 블록체인에 있는 스테이블코인으로 스왑한 뒤, 크로스 체인 브릿지 체인 호핑을 통해 다시 트론(TRON) 블록체인으로 이동시킨다.
3단계 (지갑 분산)	위 단계를 통해 세탁 및 이동된 자금은 Ronin과 마찬가지로 소액에 대해 반복적으로 트랜잭션을 일으켜 다수의 지갑으로 분산하여 보관한다.
4단계 (현금화)	3단계를 통해 최종적으로 분산된 자금은 거래소 및 OTC 네트워크를 통해 현금화(off-ramp)를 수행하며 Lazarus 그룹은 주로 아시아 OTC 네트워크를 통해 현금화하는 패턴을 보였다.

Ⅴ

시사점 및 대응 방안

앞 장까지 크로스 체인 연계 방식과 인프라 구조, 보안 요소 및 실제 사고 사례를 통한 Lazarus 그룹의 공격 및 자금 세탁 기법을 설명하였다.

본 장에서는 위 결과를 토대로 크로스 체인 환경에서의 보안 시사점을 도출하고 국내 금융권이 크로스 체인을 활용한 디지털자산 업무 및 관련 서비스 제공을 추진하는 경우 안전한 이용 환경 조성을 위한 대응 방안을 제시한다.

1. 시사점

1.1. 크로스 체인 보안 수준의 결정

앞 장의 내용을 통해 분석 결과 주목하여야 할 점은, 결국 크로스 체인의 보안 수준은 ❑ 스마트 컨트랙트 ❑ 키 관리 ❑ 검증자 집합 ❑ 외부 데이터(오라클) ❑ 거버넌스·운영 등 다양한 요소 중 가장 취약한 요소에 의해 결정된다는 점이다.

실제로 사고 사례 대부분은 스마트 컨트랙트 코드 자체에는 결함이 없었으나 키 관리와 검증자 집합 요소의 보안 수준이 미흡하여 발생하였고, 최근의 KelpDAO 사고 역시 검증 인프라와 검증자 구성 요소의 보안 수준이 미흡하여 발생하였다.

따라서, 향후 금융권에서 크로스 체인 기술 도입 시 다양한 구성 요소들을 모두 고려하여 보호하는 것이 중요하다.

1.2. 인적 보안의 중요성

크로스 체인 사고의 출발점은 모두 '사람'이었다. 검증키 및 관련 인프라 운영 권한을 가진 엔지니어를 노린 사회 공학 공격에서 침투가 시작됐기 때문이다.

이는, 높은 기술적 보안 통제 수준을 갖추었더라도 이를 운영하는 사람이 공격에 당하는 순간 보안 수준이 무력화 될 수 있음을 시사한다. 따라서, 인적 보안을 우선적으로 고려하여야 하며 특히, 검증키 및 관련 인프라 운영 권한을 가진 핵심 인력에 대한 보안 교육과 통제가 중요하다.

1.3. 단일 실패 지점(SPOF)의 제거

사고 사례 분서 결과 결국 자산 탈취로 이어지는 직접적인 원인은 검증 키 및 관련 노드의 '잘못된 분산'이라는 단일 실패 지점(SPOF)[26]이 제거되지 않았기 때문이었다. 여기서 '잘못된 분산'이라 함은 키 및 노드의 개수 즉, '구성 요소'의 숫자만 고려하고 '구성 요소 간의 독립성'을 고려하지 않은 분산을 말한다.

이에, 크로스 체인에서는 독립된 담당자(조직) 및 업체를 통한 다수의 검증용 키 및 관련 노드의 구성을 통해 단일 실패 지점(SPOF)을 제거하는게 중요하다.

26) 단일 실패 지점(Single Point of Failure) : 시스템 전체의 정상 동작 여부가 특정 하나의 구성 요소에 의존하여, 해당 요소만 침해·장애가 발생하여도 전체 시스템의 침해·장애가 발생한다는 개념

예시 **단일 실패 지점 제거를 위한 분산**

1.4. 자금 세탁 대응

Lazarus 그룹은 탈취된 자산의 동결 및 회수를 피하기 위해 세탁 기법을 계속해서 고도화하고 있다. 예를 들어, 단일 믹서가 아닌 이중 믹서의 사용 및 단기간 보유 및 처리에서 수년에 걸친 장기 보유 및 분할 처리를 통한 추적 회피 등의 기법 활용이다.

이렇게, 세탁이 완료된 자산은 회수가 사실상 불가능에 가까워지고, 이는 결국 이용자(고객)에게 피해가 고스란히 전달된다.

이를, 최대한 저지하기 위해서는 2개의 요소를 우선적으로 고려하여야 한다. 첫째는 '자금이 믹서로 유입되는 시점의 탐지' 즉, 침해사고의 발생을 빠르게 탐지하고 취약점에 대한 조치 수행이 중요하고, 두 번째는 '세탁 자금이 거래소 입금 주소에 도달하는 시점의 동결' 즉, 거래소와 크로스 체인 업체 간 공조 및 협업이 중요하다.

2. 대응 방안

앞서 도출한 시사점을 바탕으로 크로스 체인 보안 요소별 대응 방안과 인적 보안 및 자금 세탁 차단 관점의 대응 방안을 설명한다. 핵심 사항은 “연계 방식과 무관하게 보안 수준을 지키기 위한 요소를 모든 방식에서 함께 고려한다”는 점이다.

2.1. 보안 요소별 대응 방안 수립

스마트 컨트랙트

스마트 컨트랙트 등을 통해 블록체인 내 비즈니스 로직 구현 시 중요 함수(deposit, withdraw, mint, burn 등)가 의도대로만 실행되도록 검증하여야 한다. 이를 위해, ❑ 컨트랙트 배포 전 제3자 보안 감사(Audit) 의무화 ❑ CI/CD 파이프라인 내 보안성 검증 자동화 ❑ 결함 발견 시 수정·대응을 위한 별도 정책 수립 등을 고려하여야 한다.

키 관리

크로스 체인 사고의 대부분이 키 침해에서 비롯된 만큼 중점적으로 관리하여야 한다. 이를 위해, ❑ HSM(하드웨어 보안 모듈)·MPC(다자간 연산) 기술을 통한 안전한 키 보관 ❑ 키 접근자에 대한 철저한 인증 및 감사 ❑ 다수의 조직 및 장소에 분산 보관 등 검증에 이용되는 중요키에 대해서는 탈취로 이어지지 않도록 안전한 설계를 고려하여야 한다.

검증자 집합

크로스 체인 내 검증자가 정직하게 운영되고 악의적 목적의 담합 또는 동시에 침해되지 않아야 한다. 이를 위해, ❑ 동일 조직(업체) 내 검증자 노드 및 서명키 집중을 금지하고 다수 조직(업체) 및 장소로의 분산 ❑ 승인 임계값을 충분한 수준으로 설정(2-of-5 지양 등)등을 고려하여야 한다.

특히, Hybrid Burn & Mint 방식에서는 단일 DVN 구성을 지양하고 복수 DVN(multi-DVN) 기반 교차 검증을 적용하여야 한다. 더불어, Ronin 사례와 같이 서명 위임과 같은 중요 권한은 회수 절차를 의무화하여 위임 기간 후 방치되지 않도록 하여야 한다.

◆ 외부 데이터(오라클)

오라클이 제공하는 데이터가 조작되지 않고 실제 시장·체인 상태를 정확히 반영하여야 한다. 이를 위해, ❑ 단일 오라클 의존성 회피 ❑ TWAP(시간 가중 평균 가격) 적용 ❑ 복수 오라클 참조 값 활용 등을 통해 플래시 론 등을 이용한 가격 조작 공격을 방지하여야 한다.

◆ 거버넌스

컨트랙트 업그레이드, 키 체계 변경, 인프라 운영 등 내부 거버넌스가 안전하고 투명한 절차로 수행되어야 한다. 이를 위해, ❑ 중요 작업 시 타임락[27]을 통한 충분한 검증 ❑ 다중 승인 기반 의사결정 ❑ 거버넌스·운영 체계에 대한 주기적 점검이 필요하다.

특히, 운영 권한이 특정 개인에게 집중되어 해당 개인이 침해를 당하거나 부재 시 대응이 불가능해지는 구조를 배제하여야 한다.

2.2. 인적 보안 수행

모든 사고의 출발점이 사회 공학 기법인 만큼, 기술적 통제와 별개로 인적 보안을 강화하여야 한다. 이를 위해, ❑ 키·검증·인프라 운영 권한을 가진 핵심 인력에 대한 주기적 보안 교육 ❑ 업무용 기기에서 출처가 불분명한 외부 파일(채용 관련 파일로 위장한 악성 파일) 실행 통제 ❑ 중요 단말에 대한 EDR[28] 및 비정상 인증 행위 탐지 적용 등을 고려하여야 한다.

27) 타임락(Timelock) : 스마트 컨트랙트의 중요 변경(업그레이드·키 변경 등)이 즉시 실행되지 않고, 미리 정해진 대기 시간이 지난 후에 실행되도록 하는 지연 방안

28) EDR(Endpoint Detection and Response) : PC 및 서버 등 엔드포인트(단말)의 행위를 실시간으로 감시하여, 악성코드 실행이나 비정상 행위를 탐지하고 대응하는 기술

2.3. 자금 세탁 대응 방안 수립

Lazarus 그룹과 같은 공격자의 최종 목표인 현금화를 차단하기 위해서는 자금 세탁이 완료되기 전 탐지 및 차단하는 것이 핵심이다. 이를 위해, ❑ 믹서(Tornado Cash 등)를 경유한 자산의 유입 탐지 ❑ 거래소 입금 단계에서의 의심 자산 신속 동결 ❑ 트래블 룰 및 AML[29](자금세탁방지) 도구를 활용한 크로스 체인 거래 모니터링이 필요하다.

2.4. 사후 대응 및 복원력 확보

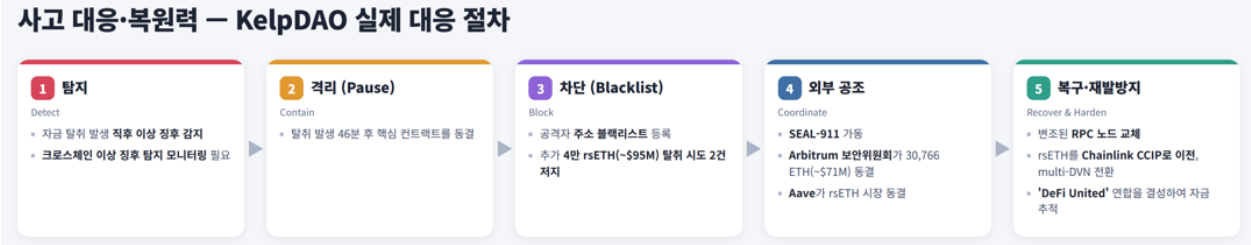
블록체인 내 디지털자산 거래는 한번 체결되면 되돌릴 수 없는 비가역성을 가지므로 부정 출금이 발생하면 사후 조치가 매우 어렵다. 이에, 사고 발생 시 피해를 최소화하고 신속히 조치하기 위한 사후 대응 및 복원력 체계를 설계하여야 한다.

사후 대응 및 복원력 체계를 갖추기 위해서는 ❑ 긴급 정지(서킷 브레이커) 권한 및 절차 마련 ❑ 공격자 주소 차단 ❑ 외부 기관(거래소, 금융당국 및 수사기관 등)과의 신속한 공조 ❑ 복구 및 재발 방지 대책 수립 등을 고려하여야 한다. 또한, 사고 대응 과정에서 책임소재에 관한 분쟁이 발생할 수 있으므로 사전에 사고 시 대응 주체와 책임 범위를 명확히 하여야 한다.

이러한 체계를 통해 적극적으로 대응한 사고 사례가 KelpDAO 사고이다. KelpDAO에서는 탐지 후 스마트 컨트랙트를 통한 자산 동결, 공격자 지갑 주소 식별 등을 통한 약 4만개의 추가 탈취 시도를 차단하고 외부 공조를 통해 추가로 공격자가 탈취한 자산으로 식별되는 약 3만개의 이더리움을 동결하였다. 또한, 변조된 RPC 노드의 교체 및 자금 추적을 위한 연합체를 만들어 대응하는 등의 사후 조치를 수행하였다.

29) AML(Anti-Money Laundering) : 범죄 수익의 출처를 숨기거나 합법 자금으로 위장하는 자금 세탁을 탐지 및 차단하기 위한 금융 규제 및 통제 체계

실제 사례 KelpDAO 사고에서의 사후 대응



2.5. 신규 위협에 대한 선제 대비

마지막으로, 공격 기술 발전에 따른 신규 위협을 모니터링하고 대비하여야 한다. 이를 위해, ❑ AI 기반 취약점 탐색에 대비한 배포 전 스마트 컨트랙트의 철저한 사전 검증과 취약점 점검 ❑ 양자컴퓨팅 발전에 대비한 양자내성암호(PQC)[30] 전환 로드맵의 사전 검토 등이 필요하다.

30) 양자내성암호(Post-Quantum Cryptography) : 양자컴퓨팅의 연산 능력으로도 풀기 어렵도록 설계되어, 양자컴퓨팅을 통한 공격에도 안전성을 유지할 수 있는 암호 기술

VI 결 론

크로스 체인 연계는 디지털자산 업무 확대에 필수적인 기술이며 보안 수준은 스마트 컨트랙트, 키 관리, 검증자 집합, 외부 데이터(오라클), 거버넌스·운영 체계라는 다양한 보안 요소 중 가장 취약한 요소에 의해 결정된다.

따라서, 크로스 체인 연계의 보안 핵심은 어떤 연계 방식을 선택하든 보안 요소를 모두 안전하게 유지하는데 있다.

특히, 스테이블코인 관련 제도화 논의 등 국내 금융권에서 크로스 체인을 통한 업무 확대가 예상되는 만큼 서비스 구상 단계부터 이러한 보안 요소에 대한 안전한 설계 및 지속적인 검토를 수행하여야 하고, AI 및 양자컴퓨팅 등 신규 위협까지 고려한 중·장기적인 대비책 마련이 필요한 시점이다.

DeepChain Vol.1

Cross-Chain Breaking Point

(디지털자산 크로스 체인 보안 위협 분석)

발 행 일 | 2026년 7월

발 행 인 | 박상원

작 성 자 | 금융보안원 디지털자산실 디지털자산기술팀(팀장: 김현민)
유희만, 장성찬

발 행 처 | 금융보안원

경기도 용인시 수지구 대지로 132

TEL: 02-3495-9000 FAX: 02-3495-9789


금융보안원
FINANCIAL SECURITY INSTITUTE