블록체인 브릿지를 통해 이동한 가상자산의 추적 및 검증

[1]하동현, [2*]손태식

Tracking of cryptocurrency moved through blockchain Bridge

[1]Donghyun Ha, [2*]Taeshik Shon

요 약

블록체인 브릿지(이하 '브릿지'이라 한다.)는 블록체인간 자산 이동을 가능하게 해주는 서비스를 말한다. 브릿지는 사용자에게 가상 자산을 입금 받고 다른 블록체인의 사용자에게 동일한 가상 자산을 전달하는 역할을 한다. 블록체인 환경은 각각 독립적이기 때문에 일반적인 방식으로 다른 블록체인으로 자산을 옮길 수 없기 때문에 사용자는 브릿지를 이용한다. 따라서 브릿지를 이용한 자산 이동은 일반적인 방식으로 추적할 수 없다. 만약 악성 행위자가 브릿지를 통해 자금 이동을 하였다면 기존의 자산 추적 도구로는 추적에 한계가 있다. 따라서 본 논문에서는 브릿지의 구조를 파악하고 브릿지 요청에 대한 이벤트 로그를 분석하여 브릿지 이용 정보를 획득하는 방법에 대하여 제안한다. 우선 브릿지의 구조를 파악하기 위해 Ethereum Virtual Machine(EVM) 기반 블록체인에서 작동하는 브릿지를 대상으로 분석을 진행하였다. 분석한 내용을 바탕으로 임의의 브릿지 이벤트에 대하여 적용해보았다. 나아가 실제 추적에 사용될 수 있도록 브릿지 이용정보를 지속적으로 수집하여 저장하는 자동화 도구를 제작하였다. 실제 브릿지 이용 후 도구를 통해 이용 정보를 추출하여 송신 블록체인, 수신 블록체인, 전달받는 지갑 주소, 전송한 토큰의 종류, 수량과 같은 추적에 중요한 정보들을 확인할 수 있었다. 이를 통해 블록체인 브릿지를 이용한 자산 이동 추적의 한계를 극복할 수 있음을 보여주었다.

Abstract

A blockchain bridge (hereinafter referred to as "bridge") is a service that enables the transfer of assets between blockchains. A bridge accepts virtual assets from users and delivers the same virtual assets to users on other blockchains. Users use bridges because they cannot transfer assets to other blockchains in the usual way because each blockchain environment is independent. Therefore, the movement of assets through bridges is not traceable in the usual way. If a malicious actor moves funds through a bridge, existing asset tracking tools are limited in their ability to trace it. Therefore, this paper proposes a method to obtain information on bridge usage by identifying the structure of the bridge and analyzing the event logs of bridge requests. First, to understand the structure of bridges, we analyzed bridges operating on Ethereum Virtual Machine(EVM) based blockchains. Based on the analysis, we applied the method to arbitrary bridge events. Furthermore, we created an automated tool that continuously collects and stores bridge usage information so that it can be used for actual tracking. We also validated the automated tool and tracking method based on an asset transfer scenario. By extracting the usage information through the tool after using the bridge, we were able to check important information for tracking, such as the sending blockchain, the receiving blockchain, the receiving wallet address, and the type and quantity of tokens transferred. This showed that it is possible to overcome the limitations of tracking asset movements using blockchain bridges.

Keywords: *Blockchain, Crypto Bridge, Cryptocurrency, Tracking Cryptocurrency, Money Tracking*

[1] *아주대학교 사이버보안학과 학사과정(jsw5258@ajou.ac.kr)*
[2*]*교신저자 아주대학교 사이버보안학과 교수(tsshon@ajou.ac.kr)*

Received: Apr. 28, 2023, Revised: Jun. 13, 2023, Accepted: Jun.22, 2023

I. 서론

블록체인은 탈중앙화된 분산 시스템으로 여러 노드들이 트랜잭션을 검증하고 블록을 생성하며 유지된다. 이 과정에서 각 노드들은 동일한 프로토콜과 암호화 방식을 사용하고, 분산된 데이터를 공유한다. 각각의 노드들은 블록체인의 고유한 프로토콜을 따른다. 예를 들어 암호화 방식, 블록 생성 주기, 채굴자 보상 등이 있다. 그렇기 때문에 한 블록체인에서 생성된 블록은 다른 블록체인에서 사용할 수 없으며 이러한 특성에 의해 블록체인은 각각 독립적이다. 따라서 블록체인간 메세지 교환 또는 자산 이동을 위해서는 블록체인 브릿지와 같은 기술이 필요하다.

예를 들어 블록체인 브릿지를 이용하면 클레이튼 블록체인에 있는 자산을 이더리움 블록체인으로 옮길 수 있게 된다. 이러한 브릿지를 통해 블록체인간의 상호 운용성을 높일 수 있으며, 다양한 분야에서 블록체인 기술을 활용할 수 있게 된다. 여러 브릿지들의 정보를 모아 보여주는 Chainspot[1]에 따르면 2023 년 1 월 기준 브릿지를 통해 이동한 자산의 가치가 55 억 달러라고 추정하였다. 블록체인 업계가 발전함에 따라 수많은 블록체인이 새로 생겨나고 있으며, 블록체인 생태계가 파편화 됨에 따라 점점 더 브릿지를 이용하는 사람이 많아지게 되고 브릿지를 통한 자산 추적의 수요가 늘어날 것이다.

현재 상용화 되어있는 Ethereum Virtual Machine (EVM) 기반 블록체인 지갑 추적툴은 타겟 블록체인에서 일어난 트랜잭션을 분석하여 특정 지갑에서 자금이 어떻게 빠져나갔는지 보여준다. 블록체인의 특성상 타 체인의 트랜잭션의 정보를 담고 있지 않기 때문에 브릿지를 통하여 자산을 옮겼을 경우 기존 툴에서는 자금의 이동을 알아내지 못한다.

이에 본 논문에서는 브릿지를 통해 다른 블록체인으로 넘어간 가상 자산에 대하여 추적하는 방법을 제시하고자 한다. 기존의 가상 자산 추적툴[2][3][4]은 독립된 하나의 블록체인의 데이터에서 추적을 하기에 브릿지를 통해 다른 블록체인으로 넘어간 자산에 대해선 추적할 수 없을 뿐더러 어떤 블록체인으로 이동했는지, 얼만큼의 자산이 이동했는지 정보조차 알 수 없다. 본 논문에서 제안한 방법을 통해 이런 단점을 극복할 수 있으며, 추적이 필요한 사건이 발생하였을 때 기존 방식보다 빠르게 자금을 추적할 수 있게 된다. 또한 기존 가상자산 추적 프로그램을 응용하여 다른 블록체인으로 넘어간 자산까지 추적할 수 있도록 확장 시킬 수 있다. 본 논문의 구성은 다음과 같다. 먼저 2 장에서 브릿지와 가상 자산 추적에 대한 선행연구를 살펴본다. 3 장에서는 다른 블록체인으로 자산을 옮기는 방법들에 대해 설명한다. 그리고 4 장에서는 브릿지를 통해 이동한 자산에 대한 분석방법을 제시한다. 5 장에서는 대상 브릿지를 선정하고 실제 브릿지의 소스코드를 분석하여 이벤트 데이터를 분석한다. 6 장에서는 5 장에서 분석한 정보를 자동으로 파싱하는 도구를 제작하여 검증한다. 마지막으로 7 장을 통해 결론을 맺는다.

II. 관련연구

본 장에서는 블록체인 자산 추적에 대한 기존 연구와 해당연구의 차별성에 대해 기술한다. 현재 브릿지를 통한 자산 추적에 대한 연구는 많이 이루어지지 않았다.

백경민 등[5]은 이더리움 기반의 가상자산 추적방안에 대한 연구를 진행하였다. 이 연구에서는 EVM 기반의 블록체인에서 트랜잭션을 읽는 방법을 설명하였다. 또한 트랜잭션의 ID 를 통해 자금의 이동을 추적하는 방법에 대해 설명하였다. 이 연구는 블록체인 스캐너를 통해 직접 데이터를 분석하는 방법을 제시하였으며 실제 추적을 위해선 자동화 도구가 필요하기 때문에 자동화 방법에 대한 추가 연구가 필요하다.

Dan Lin 등[6]은 이더리움 블록체인에서 자금을 추적할 때 랜덤 워크 기반 방식이 유효하다는 것을 보여주고 있다. 일반적인 자금 추적 시나리오에서 유의미한 연구결과를 보여주었으나 브릿지를 통해 이동한 자산에 대해서는 추가 연구가 필요하다.

Rafael Belchior 등[7]은 브릿지 해킹을 미리 막을 수 있도록 브릿지 이용정보를 모니터링하는 방식을 제안했다. 블록체인과 브릿지 사이에 모니터링 시스템을 만들어 이용정보를 감지하고 판별하는 과정을 거쳐 해킹 시도를 차단한다.

Ryan Zarick 등[8]은 다른 블록체인으로 가상 자산을 옮기는 방식에 대하여 설명하고, 기존 브릿지의 방식이 아닌 탈중앙화 메세징 프로토콜을 이용한 방식을 제안했다. 이는 일부 중앙화 되어있는 기존 브릿지의 문제를 해결하고 브릿지의 결과로 유저에게 지급되는 warp 토큰을 거치는 과정을 없앴다. 자산 이동 뿐만 아니라 다른 블록체인의 스마트 컨트랙트를 호출할 수 있도록 설계되었다.

지금까지 블록체인 내 자산 추적에 대한 연구는 이루어지고 있으며 여러가지 추적 방식을 적용한 상용 프로그램 또한 출시되고 있다. 하지만 브릿지를 통해 이동한 자산에 대한 추적에 대한 연구는 충분히 진행되지 않았다.

본 연구에서는 브릿지에 대한 설명과 추적 방법에 대해 제시한다. 또한 제시한 추적 방법을 적용한 자동화 도구를 만들어 추적 방법을 검증한다.

III. 브릿지를 통해 이동한 가상자산 추적 방법

본 장에서는 브릿지의 종류와 브릿지를 통해 이동한 가상자산 추적 방법에 대해서 설명한다.

3.1 다른 체인으로 가상자산을 옮기는 방법

가상자산을 다른 블록체인으로 이동하는 방법은 여러가지가 있으며 각각의 장단점이 존재한다. 사용자의 요구사항과 상황에 따라 적절한 방법을 선택하는 것이 중요하다. [그림 1]은 다른 체인으로 가상자산을 옮기는 대표적인 3 가지 방법이다.

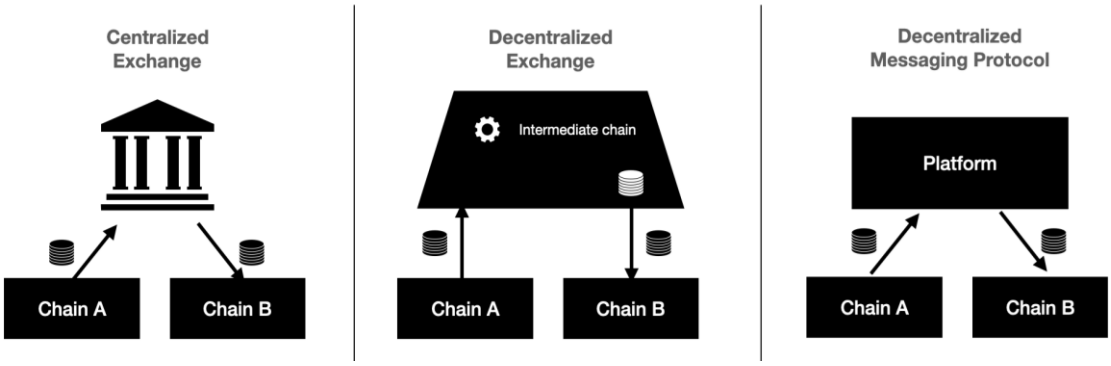


Figure 1. The process of transferring cryptocurrency to another blockchain.
그림 1. 다른 체인으로 가상자산을 옮기는 방식

먼저 간편하게 사용할 수 있는 중앙화된 거래소를 이용하는 방법이 있다. 해당 방법은 블록체인에 대한 이해 없이도 쉽게 사용할 수 있다.거래소를 통해 가상자산을 블록체인을 손쉽게 이동할 수 있다는 장점이 있지만, 거래소를 신뢰해야 한다는 단점이 있다. 거래소가 해킹을 당하거나 유동성이 부족한 경우 자산 이동중에 문제가 발생할 수 있다. 따라서 사용자는 이런 점을 고려하여 이용할 거래소를 잘 선택해야한다. 또한 주요 거래소는 대부분 실명제를 택하고 있어 이 방식을 이용하면 거래소에 신분이 노출된다는 단점도 존재한다.

다음으로 탈중앙화 거래소를 이용하는 방식은 중앙화된 거래소를 이용하는 방식과는 달리 거래소에 예치한 자산은 해당 거래소에서 자체적으로 발행된 wrap 토큰으로 대체되어 이동하게 된다. 이 방식은 이동된 자산이 거래소에 종속된 토큰이므로 실제 유저가 원하는 토큰으로 스왑하기 위한 유동성 풀이 제공되어야 한다는 단점이 있다.

마지막으로 가장 최근에 제시된 탈중앙화 메세징 프로토콜을 이용하는 방법이 있다. 단순 자산 이동 뿐만 아니라 두 블록체인간 메세지를 전달하는 역할을 한다.. 탈중앙화 거래소와는 달리 중간에 wrap 토큰이 필요하지 않고 토큰의 전송이 가능하다는 장점이 있다. 이 방식은 블록체인

의 스마트 컨트랙트를 이용하여 자동화된 메세지 전달과 처리가 이루어지기 때문에 보다 안전하고 빠르게 처리될 수 있다는 장점이 있다. 하지만 이를 구현하기 위해선 상호간에 이해관계가 없는 독립적인 주체들의 협력이 필요하다.

3.2 브릿지 이벤트 로그 분석

3.1 에서 제시한 두가지 탈중앙화 방식은 근본적으로 동일한 방식을 취하고 있다. 먼저 자산을 이동할 블록체인에서 자산이동을 요청하는 스마트컨트랙트 호출이 일어난다. 블록체인 외부에서 동작하는 프로그램에 의해 해당 호출을 감지하고 호출이 유효한지 검증하는 과정을 거친다. 유효성 검사가 완료되면 자산이 이동될 블록체인의 스마트컨트랙트를 호출하여 유저에게 자산을 전송하게 된다.

이때 유저의 자산이동 요청을 감지하는 방식이 대부분 스마트컨트랙트의 이벤트 기능을 이용하여 구현하고 있다. 자산이동 요청이 발생하면 스마트컨트랙트는 이벤트를 발생시키고 이는 블록체인을 통해 전파된다. [그림 2]은 실제 브릿지의 자산이동 요청에 대한 이벤트로그이다. 이러한 이벤트 로그는 브릿지 플랫폼 뿐만 아니라 누구에게나 공개되어 있기 때문에 해당 이벤트를 감지하여 대상 블록체인, 전달 받을 주소, 전달될 가상자산의 종류, 전달될 수량 등 여러가지 정보를 파악할 수 있다.

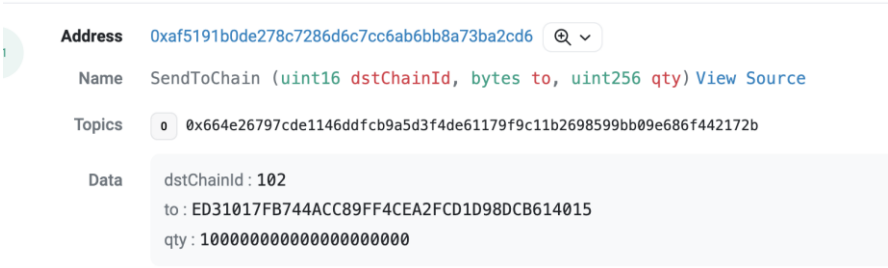


Figure 2. Stargate bridge event log
그림 2. Stargate 브릿지 이벤트 로그

이벤트 로그 분석은 정확한 추적이 가능하지만 각 브릿지의 이벤트 형식이 서로 다르기 때문에 각 브릿지마다 이벤트를 분석하여 구조를 파악해야 하는 단점이 있다.

3.3 자금의 입출금 분석

브릿지는 자금 이동 수단 중 하나로, 입금과 출금이 순차적으로 일어난다. 대부분의 브릿지에서는 입금한 금액에서 수수료를 제외한 금액이 하나의 계좌로 출금된다. 이를 이용하여 입출금을 추적할 수 있다.

예를 들어, 1%의 수수료가 존재하는 브릿지에서 100 개의 토큰을 브릿지 컨트랙트에 전송한 경우, 비슷한 시간대에 99 개의 토큰이 브릿지에서 출금되는 기록을 특정하여 해당 입금에 대한 출금임을 예측할 수 있다.

그러나 이러한 방식은 브릿지 이용자가 많을 경우 정확도가 높지 않을 수 있으며, 출금이 일어날 블록체인을 예측하기 어렵다는 단점이 있다

IV. 제안하는 브릿지 기반 가상자산 추적

3 장에서는 브릿지를 통해 이동한 자산의 추적방법 두가지를 제시하였다. 두가지 방법 중 이벤트 로그 분석을 하는 방법이 추적의 정확도도 높으며, 새로운 브릿지가 등장하는 경우 해당 브릿지에 맞는 이벤트 필터를 제작하여 플러그인 형식으로 업데이트할 수 있다. 따라서 4 장에서는 이벤트 로그를 통한 추적에 대하여 다룬다.

4.1 브릿지 선정

이벤트 로그는 브릿지의 스펙에 따라 각기 다르기 때문에 어떤 브릿지를 대상으로 추적할지 선정해야 한다. 여러 브릿지들 중 가장 최신의 기술인 3 장에서 언급했던 탈중앙화 메세징 프로토콜을 적용한 브릿지 Stargate 를 선정하였다.

4.2 브릿지 코드 분석

브릿지의 경우 컨트랙트에 대한 신뢰성이 확보되어야 하기 때문에 대부분 코드를 공개한다. Stargate 역시 마찬가지로 GitHub 를 통해서 코드를 공개하고 있다. 해당 레포지토리의 가장 최신 커밋인 c647a3[9]을 기준으로 분석하였다.

[그림 3]은 Stargate 에서 브릿징을 하기 위한 엔트리 포인트는 swap 이라는 함수이다. 해당 함수의 인자로 목적지 블록체인의 id, 전송할 자산의 종류, 전송 받을 주소 등 핵심 정보들을 유저로부터 제공 받고 있다. 이후 해당 인자들을 바탕으로 payload 를 생성한다. 이 payload 는 메세징 프로토콜인 LayerZero 로 전달되며 UltraLightNode 에서 Packet 이라는 이벤트가 발생한다.

```
//--------------------------------------------------------------------------
// LOCAL CHAIN FUNCTIONS
function swap(
    uint16 _chainId,
    uint256 _srcPoolId,
    uint256 _dstPoolId,
    address payable _refundAddress,
    Pool.CreditObj memory _c,
    Pool.SwapObj memory _s,
    IStargateRouter.lzTxObj memory _lzTxParams,
    bytes calldata _to,
    bytes calldata _payload
) external payable onlyRouter {
    bytes memory payload = abi.encode(TYPE_SWAP_REMOTE, _srcPoolId, _dstPoolId, _lzTxParams.dstGasForCall, _c, _s, _to, _payload);
    _call(_chainId, TYPE_SWAP_REMOTE, _refundAddress, _lzTxParams, payload);
}
```

Figure 3. swap function of Bridge.sol
그림 3. Bridge.sol 의 swap 함수

[그림 4]은 최종적으로 Packet 이벤트가 발생하는 UltraLightNodeV2 의 send 함수이다. Packet 이벤트는 encodePacked 를 통해 여러 정보들을 압축하여 담고 있으며 각각의 자료형을 소스코드를 통해 알기 때문에 역으로 decode 를 할 수 있다.

```
// emit the data packet
bytes memory encodedPayload = abi.encodePacked(nonce, localChainId, ua, dstChainId, dstAddress, payload);
emit Packet(encodedPayload);
```

Figure 4. send function of UltraLightNodeV2.sol
그림 4. UltraLightNodeV2.sol 의 send 함수

4.3 실제 트랜잭션 분석

Polygon 체인의 Stargate 컨트랙트에서 swap 함수를 호출한 임의의 트랜잭션 (0x7d34e8dd1b0cf9f280237b4ee191212c8b791cf25ae123285304b03405da0a54)을 분석하여 어떤 자산을 이동시켰는지 확인하였다. [그림 5]는 실제 트랜잭션에서 발생한 Packet 이벤트 로그이다.

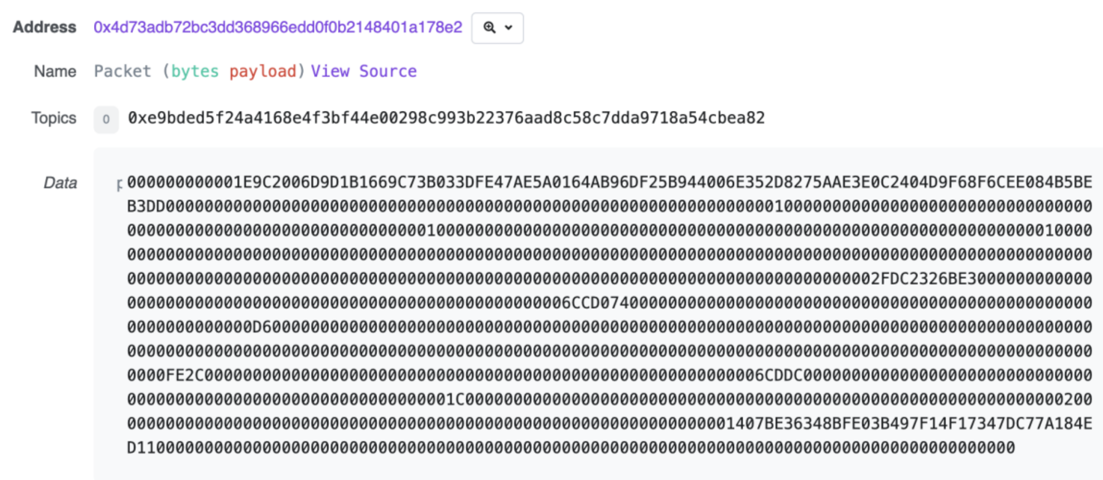


Figure 5. Packet event log
그림 5. Packet 이벤트 로그

4.2 에서 알아낸 Packet 의 구조는 uint64 nonce, uint16 localChainId, address ua, uint16 dstChainId, bytes dstAddress, bytes payload 이다. Application Binary Interface(ABI)[10] encoding 을 이용하여 압축 되어있으므로 자료형에 맞게 분리하면 [표 1]과 같은 데이터가 추출된다.

Table 1. Packet data
표 1. Packet 데이터

Datatype	Data
uint64 nonce	000000000001E9C2
uint16 localChainId	006D
address ua	9D1B1669C73B033DFE47AE5A0164AB96DF25B944
uint16 dstChainId	006E
bytes dstAddress	352D8275AAE3E0C2404D9F68F6CEE084B5BEB3DD
bytes payload	0000000000000000000000000000000000000000000000000000000000000001000000000000 0000000000000000000000000000000000000000000000000001000000000000000000000000 0000000000000000000000000000000000000001000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000000002FDC232 6BE30000000000000000000000000000000000000000000000000000000000006CCD07400000000 00000000000000000000000000000000000000000000000000000D6000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000000000000000 0000000000000000FE2C00000000000000000000000000000000000000000000000000000000 06CDDC0000000000000000000000000000000000000000000000000000000000000001C00000 0000000000000000000000000000000000000000000000000000000002000000000000000000 00000000000000000000000000000000000000000000001407BE36348BFE03B497F14F17347D C77A184ED1100000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000

이 데이터를 통해 localChainId 와 dstChainId 를 알 수 있고 각 체인에 해당하는 값은 LayerZero 의 chain id[11]를 참고하였다. 위 브릿지 트랜잭션은 polygon 에서 arbitrum 으로 자산이동이 일어났음을 알 수 있다. ua 와 dstAddress 는 전송을 요청한 주소, 전송 받을 주소이지만 현재 Stargate 라는 브릿지에서 요청한 것이기 때문에 브릿지의 주소가 담겨져 있다. 실제 전송 받을 계정을 알기 위해선 payload 에 있는 데이터를 봐야 하며, 보낸 계정은 트랜잭션의 from 란을 확인해야 한다.

위와 같은 방식으로 5.2 에서 확인한 구조를 통해 payload 를 분석하면 [표 2]와 같은 데이터가 추출된다.

Table 2. Payload data
표 2. payload 데이터

Datatype	Data
uint8 type	0000000000000000000000000000000000000000000000000000000000000001
uint256 _srcPoolId	0000000000000000000000000000000000000000000000000000000000000002
uint256 _dstPoolId	0000000000000000000000000000000000000000000000000000000000000002
uint256 dstGasForCall	0000000000000000000000000000000000000000000000000000000000000000
uint256 CreditObj.credits	0000000000000000000000000000000000000000000000000000000000000000
uint256 CreditObj.idealBalance	0000000000000000000000000000000000000000000000002FDC2326BE3
uint256 SwapObj.amount	0000000000000000000000000000000000000000000000000000000006CCD074
uint256 SwapObj.eqFee	0000000000000000000000000000000000000000000000000000000000000D60
uint256 SwapObj.eqReward	0000000000000000000000000000000000000000000000000000000000000000
uint256 SwapObj.lpFee	0000000000000000000000000000000000000000000000000000000000000000
uint256 SwapObj.protocolFee	0000000000000000000000000000000000000000000000000000000000FE2C
uint256 SwapObj.lkbRemove	0000000000000000000000000000000000000000000000000000000006CDDC00
bytes _to	00000000000000000000000007BE36348BFE03B497F14F17347DC77A184ED110
bytes _payload	0000000000000000000000000000000000000000000000000000000000000000

payload 데이터에서는 _srcPoolId 와 _dstPoolId 를 통해 어떤 자산을 브릿지 요청하였는지 알 수 있다. 각 자산에 대한 값은 stargate docs[12]를 통해 확인하였다. pool Id 가 2 이므로 USDT 를 이동하였음을 알 수 있다. 그 수량은 SwapObj.amount 를 통해 114.086004 임을 확인할 수 있다. 또한 이전 Packet 데이터에선 알 수 없었던 전달받을 주소 또한 _to 데이터를 통해 확인할 수 있다.

결과적으로 해당 트랜잭션의 from 은 0x07be36348bfe03b497f14f17347dc77a184ed110 이므로 Polygon 의 0x07be36348bfe03b497f14f17347dc77a184ed110 는 USDT 114.086004 만큼을 Arbitrum 체인의 0x07be36348bfe03b497f14f17347dc77a184ed110 에게 전송하였음을 확인할 수 있다.

V. 브릿지를 통해 이동한 가상자산 추적 도구 개발 및 검증

추적 대상으로 설정한 Stargate 브릿지에서 Arbitrum One 에서 Polygon 체인으로 USDT 토큰 0.5 개를 이동을 수행하도록 요청하였다. 해당 요청 정보를 바탕으로 자동화 도구의 추출 결과를 비교하여 검증한다.

5.1 브릿지 이용 추적 자동화 도구

4 장에서 소스코드 분석을 통해 이벤트의 구성요소를 알아내었고 실제 이벤트 로그를 바탕으로 유저의 브릿지 이용 정보를 알아낼 수 있었다. 분석한 내용을 바탕으로 자금 추적이 필요한 경우 빠르게 파악할 수 있도록 자동화 도구를 제작하였다.

자동화 도구는 블록체인상에서 발생하는 Packet 이벤트를 받아온 뒤 브릿지 이용정보로 파싱하여 저장하고 보여주는 역할을 한다. 그렇기 위해서 Packet 이벤트를 받아야 한다. [그림 6]과 같이 web3.js 의 getPastEvents 를 이용하여 발생한 Packet 이벤트를 받아올 수 있었다.

```
await contract.getPastEvents('Packet', filter, async function(error, events) {
```

Figure 6. web3.js getPastEvents
그림 6. web3.js getPastEvents

받아온 Packet 이벤트는 [그림 7]과 같이 raw data 형태이다. 해당 데이터를 분석 내용을 토대로 decode 하는 함수를 작성하여 정보를 추출해야 한다.


```
{
  address: '0x4D73AdB72bC3DD368966edD0f0b2148401A178E2',
  blockHash: '0xba29af8d7821fd2948ce3a00cf2b65a525c2f5e50faf05ebb12c49334c5ac196',
  blockNumber: 81950000,
  logIndex: 45,
  removed: false,
  transactionHash: '0x06d39327e5c6ad326dad4c25f4c94463d710c24a76eefab13d031dc0ce5023fd',
  transactionIndex: 3,
  id: 'log_d76d9719',
  returnValues: Result {
    '0': '0x00000000000cf085006e352d8275aae3e0c2404d9f68f6cee084b5beb3dd006f701a95707a0290ac8b90b3719e8ee5b210360
883000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000000000000000
0000000000000000d0000000000000000000000000000000000000000000000000000000000000000d0000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000002c68af65360b3800000000000000000
0000000000000000000000000000038357d52b1ae896c5f0000000000000000000000000000000000000000000000000a3f55e573786e0
0000000000000000000000000000000000000000000001428c8b3d3000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000017f06e558b2000000000000000000000000000000000000000000000000a40e9152184f3000000000000000000
0000000000000000000000000000000000000001c0000000000000000000000000000000000000000000000000000000000000020000000
0000000000000000000000000000000000000000000001427283f3e4fd177af44dd2858bf927124b9a984e000000000000000
0000000000000000000000000000000000000000000000000000000000000000',
```


Figure 7. event Packet
그림 7. Packet 이벤트

[그림 8]은 4 장에서 분석한 이벤트의 구성요소를 web3.js 의 abi decode 기능을 사용하여 파싱하는 함수이다. 해당 함수의 인자로 이벤트의 raw data 를 전달하면 브릿지 요청 정보 구조체로 변환되어 리턴 된다.

```
function decodePayload(bytesData) {
  return web3.eth.abi.decodeParameters(
      [
          'uint8', 'uint256', 'uint256', 'uint256', {
          "CreditObj": {
              "credits": 'uint256',
              "idealBalance": 'uint256'
            }},{
            "SwapObj": {
              "amount": 'uint256',
              "eqFee": 'uint256',
              "eqReward": 'uint256',
              "lpFee": 'uint256',
              "protocolFee": 'uint256',
              "lkbRemove": 'uint256'
            }}, 'bytes', 'bytes'
      ], bytesData);
}
```

Figure 8. decodePayload function that decode data
그림 8. 데이터를 파싱하는 decodePayload 함수

파싱이 완료된 데이터는 데이터를 그래프의 형태로 볼 수 있도록 해주는 graph db 인 neo4j 에 저장하였다. 최종적으로 파싱된 데이터의 보낸 사람, 받는 사람, 보낸 수량, 보낸 블록체인 등을 그래프 형태로 보여준다. [그림 9]은 브릿지의 이벤트 로그를 수집하여 브릿지 이용정보를 시각적으로 표현한 것이다. 하나의 화살표는 한번의 브릿지 이용정보를 나타내며, 일반적인 브릿지를 이용하는 사용자는 다른 블록체인의 동일한 자신의 주소로 자산을 이동시키기 때문에 [그림 9]와 같이 자기 자신을 가리키는 경우가 많았다.

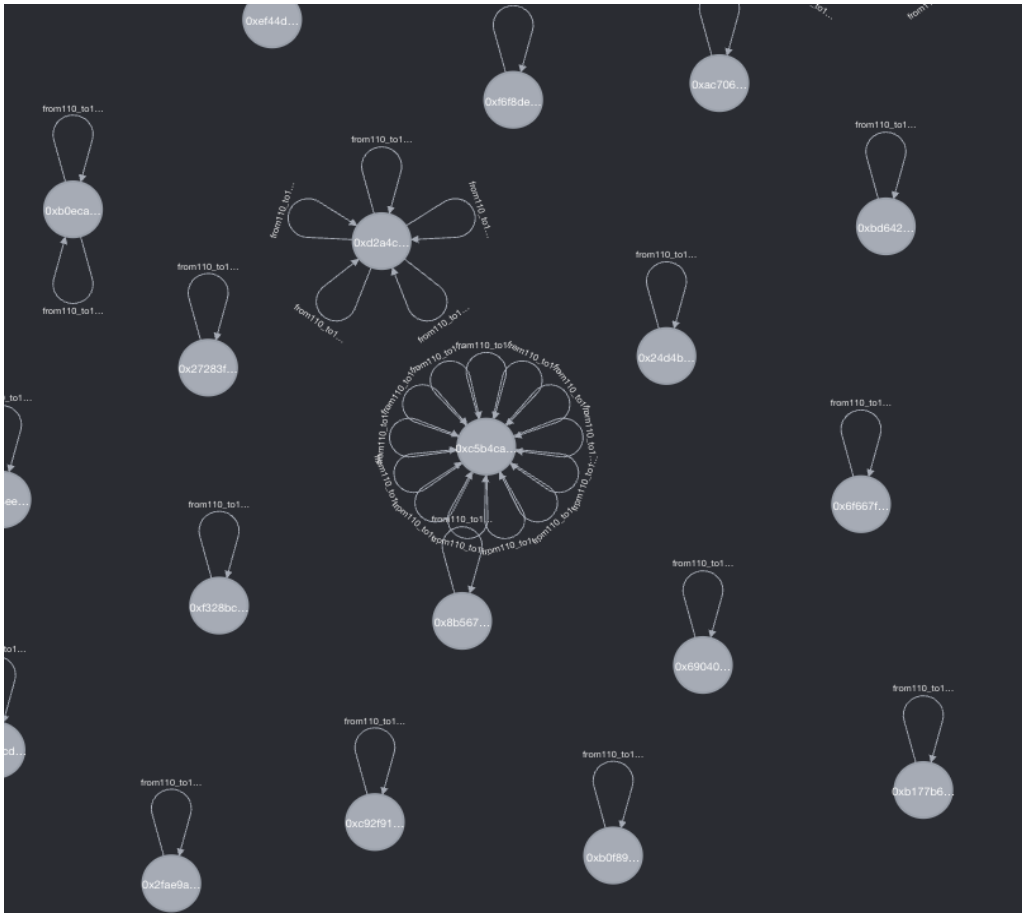


Figure 9. Result of tracking tool
그림 9. 자동화 도구 추출 결과

5.2 브릿지를 통한 가상자산 이동 요청

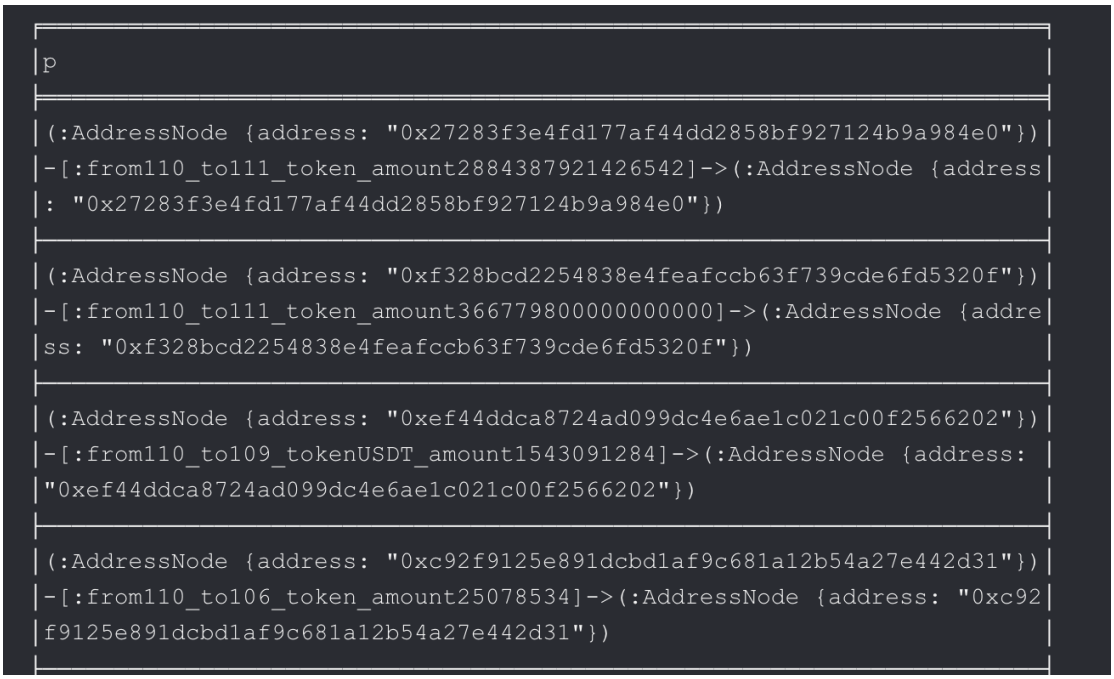
자동화 도구가 동일한 결과를 도출할 수 있는지 검증하기 위하여 실제 브릿지를 이용하여 자금 이동을 요청하였다. 검증하고자 하는 시나리오는 Arbitrum One 에서의 추적 대상지갑 0xe86d4e36124eCc84760CDC2Eb2e0E7C9728C4FF0 이 Stargate 브릿지를 이용한 것으로 Polygon 체인의 0x955916640F96eFCD66F4AeE595e0a44700A57868 에게 0.5 USDT 를 전송하였다. 자세한 요청 정보는 [표 3]과 같다.

Table 3. Bridge request information
표 3. 브릿지 요청 정보

Type	Data
Request time	Apr-19-2023 03:10:05 AM +UTC
Source blockchain	Arbitrum One
Destination blockchain	Polygon
Sender wallet address	0xe86d4e36124eCc84760CDC2Eb2e0E7C9728C4FF0
Receiver wallet address	0x955916640F96eFCD66F4AeE595e0a44700A57868
Target cryptocurrency	USDT
Amount	0.5

5.3 자동화 도구를 통한 데이터 추출

가상자산을 요청한 시점을 포함하는 Arbitrum One 의 81950000 블럭부터 81956217 블럭까지 데이터를 추출하였으며 총 609 개의 브릿지 요청이 감지되었다. [그림 10]은 추출한 데이터의 일부를 테이블의 형태로 표시한 것이다. 추출된 데이터들은 neo4j 의 관계로 표현되며, 관계이름에 이용한 블록체인, 전송한 토큰, 수량 등이 표시된다.



```
p
(:AddressNode {address: "0x27283f3e4fd177af44dd2858bf927124b9a984e0"})
-[:from110_to111_token_amount2884387921426542]->(:AddressNode {address
: "0x27283f3e4fd177af44dd2858bf927124b9a984e0"})
(:AddressNode {address: "0xf328bcd2254838e4feafccb63f739cde6fd5320f"})
-[:from110_to111_token_amount366779800000000000]->(:AddressNode {addre
ss: "0xf328bcd2254838e4feafccb63f739cde6fd5320f"})
(:AddressNode {address: "0xef44ddca8724ad099dc4e6ae1c021c00f2566202"})
-[:from110_to109_tokenUSDT_amount1543091284]->(:AddressNode {address:
"0xef44ddca8724ad099dc4e6ae1c021c00f2566202"})
(:AddressNode {address: "0xc92f9125e891dcbd1af9c681a12b54a27e442d31"})
-[:from110_to106_token_amount25078534]->(:AddressNode {address: "0xc92
f9125e891dcbd1af9c681a12b54a27e442d31"})
```

Figure 10. Part of data from tracking tool
그림 10. 자동화 도구로 추출된 데이터 일부

5.4 추출된 데이터 검증

추적 대상 주소인 0xe86d4e36124eCc84760CDC2Eb2e0E7C9728C4FF0 로부터 요청된 브릿지 이용 정보를 찾기 위하여 [그림 11]의 Cypher 쿼리를 이용하였다. 그 결과 [그림 12]와 같은 결과를 도출할 수 있었다.

```
1 MATCH
  (a:AddressNode{address:"0xe86d4e36124ecc84760cdc2eb2e0e7c9728c4ff0"
  })-[r]→(b)
2 RETURN a,r,b
```

Figure 11. Cypher query to find data by specifying the sender
그림 11. 송신자를 특정하여 데이터를 찾는 Cypher 쿼리

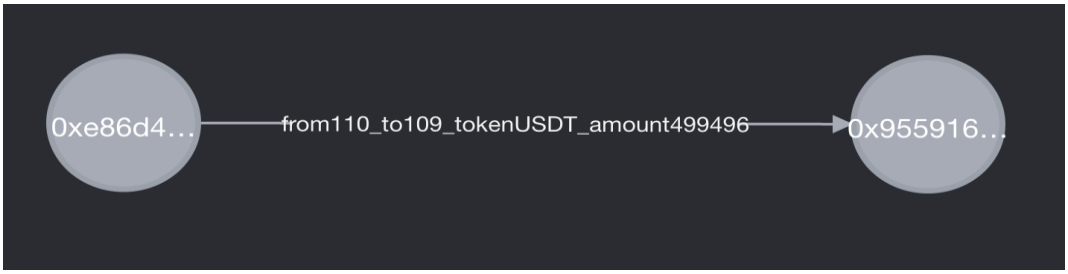


Figure 12. Extracted data
그림 12. 추출된 데이터

추출된 데이터의 관계에 정의된 데이터를 통해 브릿지 정보를 알 수 있다. 추출된 데이터는 [표 4]와 같다.

Table 4. Extracted data from tracking tool
표 4. 자동화 도구 추출 데이터

Type	Data
Request time	Block 81952581
Source blockchain	110 (Arbitrum One)
Destination blockchain	109 (Polygon)
Sender wallet address	0xe86d4e36124eCc84760CDC2Eb2e0E7C9728C4FF0
Receiver wallet address	0x955916640F96eFCD66F4AeE595e0a44700A57868
Target cryptocurrency	USDT
Amount	0.499496

결과적으로 [표 3]의 요청한 정보와 유사한 데이터를 얻을 수 있었다. 실제 Polygon 체인에서 수신 지갑 주소에서 받은 USDT 는 0.499496 개 였으며 수량의 경우 브릿지에서 수취하는 수수료를 제외한 실제 도착 블록체인에서 받는 양을 알 수 있었다.

VI. 논의

기존의 가상자산 추적 툴이 브릿지를 통해 이동한 자산에 대해선 추적하지 못한다는 한계가 있었다. 이러한 한계를 해결하기 위하여 브릿지가 어떻게 설계 되어있고 자산 이동요청에 대해 어떤 이벤트가 발생하는지 알아내기 위해 브릿지의 소스코드를 분석하였다. 분석한 내용을 바탕으로 이벤트로 전달되는 내용을 통해 브릿지 이용정보를 알아낼 수 있음을 발견하였다. 나아가 브릿지 이용정보 실시간 조회를 위해 브릿지 컨트랙트에서 발생한 이벤트를 자동으로 수집하여 브릿지 이용정보를 보관하는 데이터베이스를 구축하였다. 수집된 데이터와 자동화 도구를 통해 원하는 추적 대상의 브릿지 이용 정보에 대해서 알아낼 수 있었다.

제안한 추적 방식을 통해 실제 브릿지 이용 정보를 얻을 수 있음을 검증하였다. 이를 통해 기존의 추적 툴에서 감지하지 못하는 브릿지 이용 정보에 대해서 알 수 있다. 기존의 추적 툴의 관점에서 브릿지의 이용을 바라보면 A 블록체인의 B 라는 주소에서 가상자산을 입금 받고, C 블록체인의 D 라는 주소에게 가상자산을 출금해주는 것으로 보일 것이다. A 블록체인에서 시작되었음은 알 수 있지만 C 블록체인으로 이동했다는 것, D 라는 주소는 알 수 없기 때문에 브릿지 추적이 유의미하다. 본 논문에서 제안한 추적방식을 적용하여 기존의 추적툴의 플러그인 형태로 적용시킨다면 추적이 끊어지는 부분을 이어줄 수 있을 것이다.

분석의 대상으로 선정한 Stargate 브릿지는 LayerZero 라는 탈중앙화 메세징 프로토콜을 이용하고있다. 하지만 해당 프로토콜은 브릿지만 사용하는 것이 아닌 크로스 체인 로직이 필요한 다른 Dapp 들도 사용한다. 따라서 분석대상으로 잡은 Packet 이벤트가 브릿지에서 온 것이 아닌 경우가 존재하며 이를 완벽하게 걸러내는 것에 한계가 있었다.

VII. 결론

최근 다양한 블록체인의 등장으로 인해 블록체인간 연결을 도와주는 브릿지의 활용도가 증가하고 있다. 따라서 브릿지를 통해 이동한 가상자산에 대해 추적에 대한 수요도 높아지고 있는 추

세이다. 브릿지를 통해 이동한 경우 일반적인 추적 도구를 통한 추적이 불가능 하다. 기존 추적 도구는 하나의 블록체인의 트랜잭션을 분석하여 자산의 입출금을 알아내는 방식을 사용하고 있기 때문이다. 일반적인 추적도구로 자금 추적을 할 경우 브릿징 된 자산에 대해 자산은 입금되었지만 출금 기록은 보이지 않을 것이다. 따라서 브릿지를 위한 추적방식이 연구되어야 하고 이를 기존 추적툴과 연계하여 사용할 수 있도록 설계하는 것이 중요하다.

본 논문에서는 브릿지를 통해 이동한 가상자산을 추적하는 방법을 제시하고 실제 브릿지 트랜잭션을 분석하였다. 분석한 내용을 바탕으로 추적도구를 제작하였다. 이를 통해 기존 추적 도구에서 확인하지 못했던 정보를 확인할 수 있었다. 브릿지를 이용한 뒤 해당 정보와 추적 도구에서 추출한 데이터를 비교하여 검증하였다. 실제 브릿지 요청 정보와 동일한 데이터를 얻을 수 있었다.

본 논문에서 제시한 방법은 브릿지의 이벤트로그를 파싱하여 정보를 얻어낼 수 있다는 점을 이용하였다. 아직은 자산이동에 대한 이벤트 형태의 표준이 정해지지 않았기 때문에 브릿지마다 이벤트의 형식이 다르다. 그렇기 때문에 브릿지마다 코드 분석을 통해 필터를 만들어야 한다. 이러한 방식은 브릿지의 종류가 많아질 수록 적용하기 어려워질 것이다. 향후 연구를 통해 브릿지 이벤트 표준을 제시하거나 모든 브릿지에 적용가능한 방법에 대한 방식을 제안하고 적용한다면 발전한 형태의 추적방식이 될 수 있을 것이다.

VIII. 참고문헌

[1] Chainspot Cross-Chain ecosystem review, “https://chainspot.io/report”
[2] Breadcrumbs, “Breadcrumbs wallet tracker”, 2023. [Online]. Available: “https://www.breadcrumbs.app/”
[3] Bloxy, “Bloxy wallet tracker”, 2023. [Online]. Available: “https://bloxy.info/”
[4] Ethtective, “Ethtective wallet tracker”, 2023. [Online]. Available: “https://ethtective.com/”
[5] Back KyoungMin, Yoon Cheolhee. A study on Ethereum-based virtual asset tracking method, Proceedings of KIIT Conference, 35-37, 2022.
[6] Dan Lin, Jiajing Wu, Qi Xuan, Chi K. Tse, Ethereum transaction tracking: Inferring evolution of transaction networks via link prediction, Physica A: Statistical Mechanics and its Applications
[7] Belchior, R., Somogyvari, P., Pfannschmid, J., Vasconcelos, A., & Correia, M. 2022, September 1. Hephaestus: Modelling, Analysis, and Performance Evaluation of Cross-Chain Transactions.
[8] LayerZero: Trustless Omnichain Interoperability Protocol, “https://layerzero.network/pdf/LayerZero_Whitepaper_Release.pdf”
[9] Github, “Stargate protocol”, 2023. [Online]. Available: “https://github.com/stargate-protocol/stargate/commit/c647a3a647fc693c38b16ef023c54e518b46e206”
[10] Soliditylang, “contract abi specificaion”, 2023. [Online]. Available: “https://docs.soliditylang.org/en/v0.8.19/abi-spec.html#contract-abi-specification”
[11] Layerzero, “supported chain ids”, 2023. [Online]. Available: “https://layerzero.gitbook.io/docs/technical-reference/mainnet/supported-chain-ids”
[12] Stargate protocol, “pool ids”, 2023. [Online]. Available: “https://stargateprotocol.gitbook.io/stargate/developers/pool-ids”

저자소개

하동현(Donghyun Ha)

2018 년 2 월 ~ 현재 : 아주대학교 재학

관심분야 : 블록체인 보안, 리버스 엔지니어링 등

손 태 식 (Taeshik Shon)

2005 년 ~ 2011 년 : 삼성전자 통신 · DMC 연구소 책임연구원
2017 년 ~ 2018 년 : Illinois Institute of Technology 방문교수
2011 년 ~ 현재 : 아주대학교 정보통신대학 사이버보안학과 교수

관심분야 : Digital Forensics, ICS/Automotive Security