논문 18-43-12-13 The Journal of Korean Institute of Communications and Information Sciences '18-12 Vol.43 No.12
https://doi.org/10.7840/kics.2018.43.12.2081

이더리움 노드 탐색 프로토콜 분석

명 세 인◆, 이 종 혁°

Analysis of the Ethereum Node Discovery Protocol

Sein Myung◆, Jong-Hyouk Lee°

요 약

신뢰하는 중앙기관 없이 노드간 합의를 통해 신뢰성 있는 데이터 공유를 지원하는 블록체인은 비트코인을 시작으로 다양한 분야에서 활용되고 있다. 다양한 블록체인 플랫폼 중 스마트 컨트랙트를 지원하는 이더리움은 다양한 블록체인 서비스 구축에 활발히 활용되고 있지만 아직까지 이더리움 네트워크의 동작 방식이나 특징을 분석한 연구는 미비하다. 본 논문에서는 이더리움 네트워크 구성시 반드시 필요한 노드 탐색 과정을 분석한다. 이더리움 네트워크의 특징과 주요 프로토콜 및 노드 탐색 과정을 설명하고, Go언어로 작성된 이더리움 구현물인 geth의 네트워크 동작에 따른 함수 콜 그래프 분석 결과를 제공한다.

Key Words : Ethereum, RLPx, Eclipse Attack, Geth

ABSTRACT

A blockchain that supports reliable data sharing through an agreement between nodes without a trusted central authority is being used in various fields, starting with Bitcoin. Ethereum, which supports smart contract among various blockchain platforms, is actively used to construct various blockchain services. However, there is not yet much research to analyze operation method and characteristics of an Ethereum network. In this paper, we analyze the node discovery process which is essential for an Ethereum network configuration. We explain the characteristics of the Ethereum network, the major protocols and node discovery process, and present the function call graph analysis result according to the network operation of geth, an Ethernet implementation written in Go language.

Ⅰ. 서 론

암호화폐 거래를 가능하게 하는 비트코인[1] 플랫폼의 핵심 기술로서 등장한 블록체인은 스마트 컨트랙트 기술과 융합하여 다양한 분산 애플리케이션의 실행 환경을 제공하는 플랫폼으로 발전하고 있다. 제2세대 블록체인 기술의 핵심인 스마트 컨트랙트를 지원하는 대표적인 블록체인 플랫폼으로는 이더리움[2] 등이 있다.

이더리움을 기반으로 신뢰하는 중앙기관 없이 노드간 합의 알고리즘을 통해 데이터에 대한 합의를 하고 그 결과로 다양한 서비스를 제공하고자 하는 노력이 진행 중이다. 대표적으로 암호화폐가 있겠지만, 인증 서비스에 활용되거나, 사물인터넷기기들에 대한 펌웨어 검증, 헬스케어 시스템으로의 적용 등 기존 구축 방식의 단점을 블록체인에서 제공하는 분산성과, 데이

※ 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터지원사업의 연구결과로 수행되었음 (IITP-2018-2018-0-01799).
◆ First Author : (ORCID:0000-0002-4384-3644)Dept. of Software, Sangmyung University, sein@pel.smuc.ac.kr, 학생회원
° Corresponding Author : (ORCID:0000-0002-1753-1284)Dept. of Software, Sangmyung University, jonghyouk@pel.smuc.ac.kr, 종신회원
논문번호 : 201806-B-054-SE, Received May 15, 2018; Revised September 3, 2018; Accepted September 6, 2018

터에 대한 비가역성, 무결성 등으로 해결하고 있다. 다음은 대표적인 블록체인 서비스를 보여준다.

1.1 암호화폐[2]

이더리움의 암호화폐(ETC)를 통해서 결제, 송금, 투자, 대출이 가능하다. 이때 스마트 컨트랙트 기능을 활용하여 보다 효율적인 거래를 할 수 있다. 하지만 이더리움의 구현상 취약한 요소를 검증하지 않고 활용한다면, 공격자가 쉽게 악의적인 송금 시도 및 의도하지 않은 컨트랙트 기능들이 동작할 수 있다. 그에 따라 플랫폼 이용자에게 직접적인 금전 피해를 줄 수 있다.

1.2 BIDaaS(Blockchain ID as a Service)[3]

사용자들의 가상 ID를 블록체인 네트워크에 등록하여 기기들 간의 인증이 가능하다. 이때 기존의 서버-클라이언트 구조에서의 인증 보다 안전한 인증이 가능하다. 하지만 구현상의 취약한 요소를 검증하지 않는다면 허가되지 않은 인증을 발생시킬 수 있다.

1.3 블록체인 기반 IoT 기기의 펌웨어 인증[4]

수 많은 IoT 기기의 펌웨어를 안전하게 검증하고 업데이트하기 위해서 블록체인 기술을 적용하여 분산된 검증 시스템을 구성할 수 있다. 이러한 검증시스템을 구현하기 위해 이더리움을 활용할 수 있으며, 이더리움의 구현상의 취약한 요소를 검증하지 않고 활용한다면, 공격자가 악의적인 펌웨어를 IoT 기기에 삽입하여 봇넷(Botnet)을 구성하고 추가공격이 발생할 수 있다.

1.4 헬스 케어[5]

환자 의료정보를 블록체인에 저장함으로써 안전하게 보관하고 제공할 수 있다. 이때 구현상의 취약한 요소가 발생하여 공격자가 악용한다면, 환자의 의료정보를 위조할 수 있고, 의사는 위조된 데이터를 통해서 환자의 직접적인 생명에 위협을 줄 수 있게 된다.

본 논문에서는 이더리움을 통해 블록체인 플랫폼을 구축 할 때 반드시 이해해야 할 네트워크 기능을 분석한다. 특히, 이더리움 네트워크 구성을 위해 필요한 노드 탐색 과정에 대해서 자세히 살펴본다. 본 논문의 2장에서는 이더리움을 네트워크관점에서 공격하는 관련연구를 살펴본다. 3장에서는 이더리움 합의 알고리즘과 스마트 컨트랙트에 대해 살펴본다. 4장에서 이더리움 네트워크의 특징과 주요 프로토콜 및 노드 탐색 과정을 설명한다. 5장에서는 Go언어로 작성된 이더리움 구현물인 geth의 네트워크 동작에 따른 함수 콜 그래프를 분석한다. 6장에서 결론을 서술한다.

Ⅱ. 관련 연구

이더리움 블록체인 플랫폼은 분산 데이터베이스기술에 합의알고리즘을 적용하여 스마트컨트랙트 기반의 신뢰할 수 있는 분산 애플리케이션을 실행할 수 있도록 한다. 따라서 이더리움을 기반으로 다양한 형태의 서비스[6]가 개발되고 있다. 그에 따라 공격자는 다양한 기술이 적용된 블록체인을 분석하고 악의적인 공격[7]을 실행하고 있다. 본 논문에서는 분산된 노드를 구성하는 블록체인 네트워크에서 이웃 노드가 될 피어를 찾고 선정하는 과정을 분석하고, 네트워크 수준의 취약점을 보호할 수 있도록 한다. 다음은 이더리움에서 네트워크의 취약점을 통해 특정 노드를 블록체인에서 격리하거나 공격하는 사례이다[8].

2.1 시빌 공격(Sybil Attack)

이더리움에서 노드를 구별하기 위해 사용하는 ID는 공개키 쌍 알고리즘을 통해 생성하므로 단일 컴퓨터에서 다수의 노드 ID를 생성할 수 있다. 이때 노드 아이디가 정당한 노드인지 확인 할 수 있는 방법이 없다. 따라서 공격자는 하나의 머신을 여러 개의 노드인 것처럼 동작시키는 시빌 공격이 가능하다.

2.2 연결 독점 이클립스 공격(Eclipse by Monopolizing Connections)

이 공격은 공격자가 희생자를 고립시키는 공격으로 이더리움 노드가 동시에 연결하는 피어의 수가 제한되어 있는 취약점을 악용한다. 공격자는 희생자에게 DoS(Denial of Service)를 통해 희생 노드가 재부팅되도록 한다. 희생 노드는 재부팅 이후 다시 이더리움 네트워크에 참여하기 위해 노드를 연결해야한다. 이때 공격자가 희생 노드에게 최대 연결 수만큼 연결을 요청하면 희생 노드는 공격자에게만 연결된다. 따라서 공격자가 제공하는 이더리움 정보만을 수신 하게 되어 정당한 이더리움 네트워크로부터 고립되게 된다.

2.3 테이블 오염 이클립스 공격(Eclipse by Table Poisoning)

이 공격은 희생 노드가 피어의 정보를 저장하고 있는 데이터베이스와 테이블을 오염시켜 정당한 이더리움 네트워크로부터 고립시키는 공격이다. 공격자는 시빌 공격을 수행하여 다수의 노드를 생성하고, 희생 노드가 노드 탐색을 수행할 시 공격자가 생성한 노드 ID

로 데이터베이스와 테이블을 채우도록 한다. 이후 희생자 노드가 재부팅되면 공격자가 생성한 노드 ID로 이더리움 연결 요청을 수행하게 되어 정당한 이더리움 네트워크로부터 고립되게 된다.

이클립스 공격이란 P2P(Peer-to-Peer) 네트워크에서 공격의 대상이 되는 노드를 악의적인 노드에게만 연결되게 하여, P2P 네트워크를 구성할 수 없도록 하는 공격이다. 이러한 공격을 성공하기 위해서는 공격자만의 P2P 네트워크를 구성하기 위해 다수의 노드가 필요하다. 이더리움에서는 노드를 생성하기위해 공개키-개인키 쌍만 생성하면 되므로 한명의 공격자가 다수의 노드를 생성하는 시빌 공격에 제약이 없으므로 연결 독점 이클립스 공격 또는 테이블 오염 이클립스 공격과 같이 구현상의 취약점을 통해 이클립스 공격이 가능하다. 이러한 공격으로부터 노드를 보호하기 위해 본 연구에서 이더리움 블록체인의 P2P네트워크를 구성하기위한 프로토콜의 구현 코드를 분석한다.

본 연구의 코드 분석을 기반으로 악의적인 행동을 하는 노드의 연결요청 거부 기능 또는 노드 탐색 시 완전한 P2P네트워크 구성을 위한 기법 적용에 활용될 수 있다. 예를 들어 퍼블릭(public) 블록체인의 경우 개별 노드를 공개키 ID로만 식별하지 않고, IP 주소와 노드 ID를 매핑하여 단일 머신에서 다수의 노드가 동작할 수 없도록 할 수 있다. 따라서 공격자로 하여금 시빌 공격을 어렵게 할 수 있다.

프라이벳(private) 블록체인의 경우 해당 블록체인에서 신뢰할 수 있는 노드 그룹을 지정하고, 개별 노드는 해당 그룹 중 최소 하나의 노드와 필수적으로 연결되게 하는 방안을 통해 이클립스 공격을 보호할 수 있다. 이러한 P2P 네트워크의 특징을 보호하기 위해 노드탐색 프로토콜과, 네트워크 공격에 대한 분석이 필요하다. 이클립스 공격이 성공하여 희생자(victim) 노드가 이더리움 네트워크로부터 고립되어 정당한 데이터를 받지 못하는 상황은 그림 1과 같다.

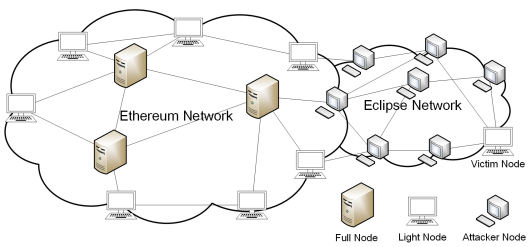

그림 1. 이클립스 공격
Fig. 1. Eclipse Attack

Ⅲ.이더리움 블록체인 플랫폼

이더리움 블록체인 플랫폼은 스마트 컨트랙트(Smart Contract)를 지원하는 2세대 블록체인이다. 합의 알고리즘을 기반으로 검증 가능한 실행코드를 배포하여 다양한 서비스를 구현할 수 있다. 이더리움은 프론티어(Frontier), 홈스테드(Homestead), 메트로폴리스(Metropolis), 세레니티(Serenity) 4단계의 발전 계획을 가지며 최종 단계인 세레니티에서 완전한 Casper 합의알고리즘 도입을 통해 가용성과 안정성 및 높은 처리율을 보장하는 분산 애플리케이션 플랫폼 제공을 목표로 하고 있다. 본 장의 1절 합의 알고리즘에서는 PoW(Proof-of-Work) 알고리즘인 Ethash와 PoS(Proof-of-Stake) 알고리즘인 Casper에 대해서 설명한다. 2절 스마트 컨트랙트에서는 이더리움 스마트 컨트랙트의 특징과 실행 환경인 EVM(Ethereum Virtual Machine)을 설명한다.

3.1 합의 알고리즘

이더리움의 합의 알고리즘은 세레니티 단계에서 기존의 Ethash에서 Casper로 완전한 합의 알고리즘 변경이 이루어진다. Ethash는 블록체인 합의 알고리즘의 가장 기본적인 PoW방식이다. 블록바디를 입력으로 하는 무작위 연산을 수행하여 이더리움 네트워크에서 합의중인 난이도에 맞는 출력을 찾을 때 까지 연산한다. 이 과정에서 DAG (Directed Acyclic Graph)를 활용하여 주문형 반도체(ASIC, Application Specific Integrated Circuit)를 통한 합의 알고리즘 가속화를 방지한다.

이더리움의 Casper알고리즘에서는 PoW를 통해 블록을 생성하며, 합의알고리즘을 진행하면서 PoS의 검증자가 일정 주기(Epoch)마다 어떤 분기를 이더리움의 메인 블록체인으로 진행할지에 대해 지분을 투자한다. 전체 지분의 2/3이상이 투표한 분기에 투표하면 보상을 받으며, 반대의 경우에는 검증자 자격을 박탈하고 예치금을 몰수한다. 이더리움의 PoS인 Casper를 사용하면 네트워크 참여자의 지분에 따라 합의가 진행 되며, 지분 투표가 성공적으로 진행된 블록체인을 되돌릴 수 없는 블록 확정(Finality) 개념이 추가된다.

3.2 스마트 컨트랙트

스마트 컨트랙트란 서면으로 작성되어 있는 계약서를 디지털 명령어로 작성하면 명확한 조건에 따라 계약이 성사되는 기술을 의미한다. 이 개념은 신뢰할 수 있는 분산 애플리케이션 실행 환경을 제공하는 블록

체인 기술을 활용하여 디지털 명령어로 작성된 스마트 컨트랙트가 실행됨을 보장할 수 있다. 이더리움 플랫폼에서 솔리디티(Solidity)언어로 스마트 컨트랙트를 작성하여 컴파일하면 EVM이 실행할 수 있는 바이트코드를 출력할 수 있다. 스마트 컨트랙트를 실행하기 위해 바이트코드를 포함하는 트랜잭션을 작성하고 블록체인에 포함되면 스마트 컨트랙트가 실행된다. 스마트 컨트랙트를 이더리움 네트워크에 배포하고 실행하는 과정을 그림 2.에 나타내었다.

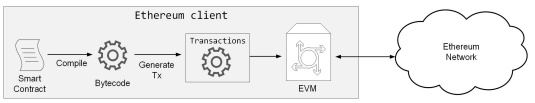

그림 2. 스마트 컨트랙트 배포
Fig. 2. Deploymemt of Smart Contract

Ⅳ. 이더리움 네트워크 동작 분석

이더리움 네트워크는 트랜잭션으로 작성된 데이터를 P2P네트워크에 분산 저장하고, 이러한 데이터를 기반으로 분산 애플리케이션을 실행하는 플랫폼이다. 중앙 신뢰 기관이 없는 P2P네트워크에서 동작하기 위해 노드 탐색 과정이 필요하며, 각각의 노드는 특정 노드에 의존하거나 네트워크 분리가 일어나지 않도록 구성되어야 한다. 본 장의 1절 네트워크 특징 에서는 이더리움 네트워크에 참여하는 노드와 공유하는 데이터를 설명한다. 2절 주요 프로토콜에서는 이더리움에서 정의하는 프로토콜에 대해서 설명한다. 3절 노드 탐색 과정에서는 노드가 이더리움 네트워크에 참여하는 과정을 설명한다.

4.1 네트워크 특징

이더리움 네트워크는 신뢰하는 중앙기관 없는 탈중앙화 환경에서 개별 노드간 합의 알고리즘 기반의 데이터 공유 네트워크로 볼 수 있다. 그림 3.은 풀 노드와 라이트 노드를 비교하여 나타내었다. 일반적으로 라이트노드는 분산 애플리케이션 즉 스마트 컨트랙트를 실행하고 실행 결과에 대한 내용을 확인하며 수수료를 지분하는 노드로 구분된다. 하지만, 풀 노드의 경우 라이트 노드의 기능에 더해 합의알고리즘 수행과 전체 블록데이터 검증을 수행하며 이더리움 네트워크를 진행시키는 역할을 맡는다.

즉, 라이트 노드는 트랜잭션 생성과 검증 및 합의알고리즘 결과인 블록헤더의 검증 정도만 수행하고, 이더리움 네트워크 진행을 담당하는 풀 노드는 트랜잭션 생성과 검증 기능을 기본으로 블록 바디생성과 합의알고리즘 수행 및 블록헤더 검증을 수행한다.

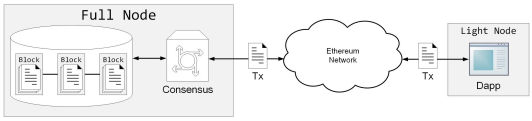

그림 3. 이더리움 노드의 역할
Fig. 3. Roles of Ethereum Node

4.2 주요 프로토콜

이더리움 네트워크를 구성하는 주요 프로토콜로는 RLPx[9]와 Wire[10] 프로토콜이 있다. RLPx 프로토콜은 P2P네트워크를 구성하고 애플리케이션 계층에 인터페이스를 제공하는 프로토콜로써 노드 탐색, 핸드셰이크 등을 담당한다. Wire프로토콜은 RLPx의 보조 프로토콜로서 노드간 전송해야하는 블록체인 데이터 통신을 정의한다. RLPx의 주요 기능은 아래와 같다.

1) 노드 탐색: Kademlia알고리즘 기반의 UDP 구현
2) 암호화된 핸드셰이크: AES-256 CTR을 사용하기 위해 ECC기반의 ECDHE키 교환을 수행
3) 암호화된 전송: 프레임단위 암호화된 세션 통신
4) 흐름제어: 동작중인 프로토콜 수 제한 및 프로토콜 별 윈도우 크기 제한을 두어 네트워크 과부하 방지

본 논문에서는 노드 탐색을 중점적으로 분석하고, 분석한 내용을 기반으로 퍼블릭, 프라이빗 유형의 블록체인 플랫폼을 채택하여 새로운 서비스를 구축할 시 네트워크 계층에서 발생할 수 있는 공격의 보완 방향에 대해서 설명한다.

4.3 노드 탐색 과정

이더리움의 노드 탐색은 RLPx프로토콜의 역할로서 Kademlia[11]를 기반으로 구현되었다. 본 절에서는 RLPx를 기준으로 노드 탐색 과정을 설명하기 위해 노드 간 거리 계산 방법, 노드 탐색에 사용되는 패킷 유형, 탐색 과정에 사용되는 파라미터 정의, 노드 탐색 프로세스를 설명하고 특정 노드를 탐색하는 시나리오를 설명한다.

완전한 분산 네트워크를 구성하기 위해서는 모든 노드는 네트워크에 참여하고 있는 모든 노드의 정보를 알아야한다. 하지만 항상 온라인을 보장하지 못하며 무수히 많은 노드가 자율적으로 참여하는 이더리움 블록체인에서는 모든 노드의 정보를 저장할 수 없

다. 따라서 임의의 노드를 선택할 수 있도록 논리적 거리 개념을 도입하여 오버레이(Overlay) 네트워크를 구성해야한다. 이더리움 노드는 512bit 길이를 갖는 공개키를 노드 ID로 갖는다. 따라서 노드 ID가 해당 노드의 논리적 위치를 나타낸다. 두 노드의 거리를 계산하는 식은 (1)과 같으며, 특정 노드를 탐색하는 요청이 오면 자신의 버켓에서 대상 노드와 가장 근접한 노드를 계산하여 응답한다. 식 (1)의 bit단위의 XOR 연산은 식 (2), (3), (4), (5)을 만족함으로써 거리 계산에 사용될 수 있다.

$$distance(Node_A, Node_B) = A_{ID} \veebar B_{ID} \quad (1)$$

$$distance(x,x) = 0 \quad (2)$$

$$distance(x,y) > 0, if\ x \neq y \quad (3)$$

$$distance(x,y) = distance(y,x) \quad (4)$$

$$distance(x,z) \leq distance(x,y) + distance(y,z) \quad (5)$$

RLPx 프로토콜의 노드 탐색 과정에 사용되는 패킷은 아래와 같이 정의된다.

1) Ping: 탐색을 완료하여 저장된 노드가 온라인인지 확인하기 위한 요청메시지로 사용된다.
2) Pong: Ping 메시지에 대한 응답 메시지로 사용된다.

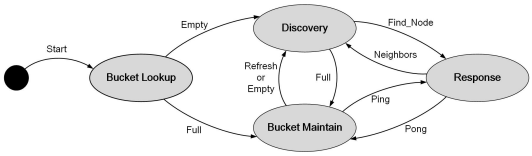

그림 4. 노드 탐색 상태 변화
Fig. 4. Statement of Node Discovery

3) Find_Node: 특정 노드 ID를 전송하여 ID와 가장 가까운 거리를 갖는 노드 정보를 요청한다.
4) Neighbors: Find_Node의 응답으로 요청받은 노드 ID와 가장 가까운 거리를 갖는 노드 ID를 전송한다.

이더리움 노드는 노드 탐색 시 동시에 α=3개(alpha)의 노드에게 특정 ID를 요청한다. 요청에 대한 응답으로 새로운 노드ID를 얻게 되면 버켓(Bucket)에 저장하고 버켓을 채울 때 까지 노드 탐색을 반복한다. 각각의 버켓은 16개(k)의 노드 ID를 저장하며, 거리별로 다수의 버켓을 유지한다. 노드를 충분히 탐색하여 버켓을 채우면 일정주기(Refresh Time, 1시간)에 따라 버켓을 정리(Refresh)한다. 그림 4는 노드가 이더리움 네트워크에 참여하기 위해 진행하는 노드 탐색 과정을 보여준다.

노드가 처음으로 이더리움 네트워크에 참여할 때 수행하는 노드 탐색을 아래 그림 5에 나타내었다. 참여하는 노드 P는 동시에 2개의 노드 탐색을 수행하며, 최초로 이더리움 네트워크에 참여하므로 부트스트랩

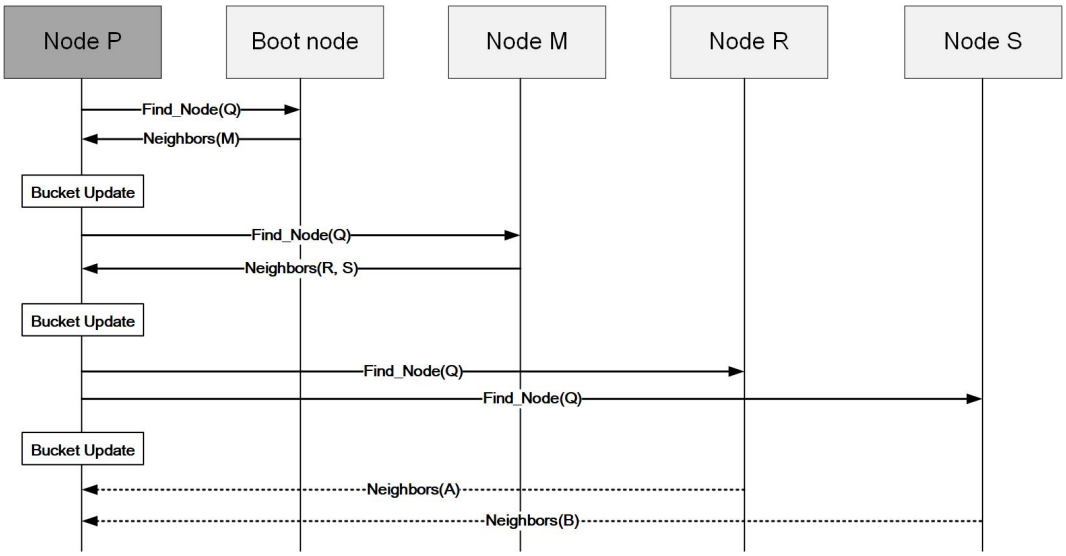

그림 5. 노드 탐색 과정
Fig. 5. Node Discovery

을 제공하는 노드에게 이더리움 네트워크 연결을 요청한다.

노드 P는 자신과 근접한 타겟 노드인 Q에 대해 노드 탐색을 요청하고, 부트 노드는 자신의 버켓에서 노드 Q와 가장 근접한 노드인 노드 M을 응답한다. 노드 P는 자신의 노드 버켓을 채우기 위해 노드 M에게 다시 노드 Q의 정보를 요청하고 노드 M은 노드 Q와 근접한 노드 R과 노드 S를 응답한다. 노드 P는 자신의 버켓을 채울 때 까지 이 과정을 반복하며 노드 탐색을 수행한다.

Ⅴ. geth 네트워크 동작 분석

이더리움 플랫폼은 오픈소스 프로젝트로 진행되고 있다. geth는 Go언어를 기반으로 구현중인 이더리움 노드 클라이언트이며 본 장에서 실제 소스코드의 함수 콜그래프를 설명한다. 1절 노드 초기화 과정에서 geth의 실행부터 이더리움 노드의 주요 기능을 수행하는 함수 호출 단계까지 초기화 과정으로 분류하고 설명한다. 2절 노드 탐색 과정은 geth의 이더리움 네트워크 참여에 관련된 주요 기능을 수행하는 함수 호출을 설명한다. 본 장에서 분석한 geth는 1.8.2 버전이다[12].

5.1 노드 초기화 과정

geth를 컴파일해서 실행하는 경우 노드의 설정을 위해 다양한 옵션을 인자로 전달한다. 그림 6.에서 geth함수가 호출되기까지 전달된 옵션에 따라 기본 설정을 진행한다. makeFullnode함수에서 노드의 역할을 설정한다. startNode함수에서 StartMining함수를 통해 합의 알고리즘 마이닝 프로세스를 진행한다. 네트워크 관련 호출은 StartNode함수를 통해 노드의 공개키 설정(NodeKey), 노드 이름 설정(NodeName), 고정된 노드를 탐색(StaticNodes), 기존에 탐색한 노드 확인(NodeDB)등을 수행하고 Start함수를 통해 네트워크 통신을 위한 서버 설정으로 넘어간다.

네트워크 통신 설정은 다른 노드의 메시지 수신과 요청 작업 처리를 담당하는 서버 실행(Open Node Server)함수와 디스커버리를 수행하기 위해 네트워크 작업을 실행하는 함수(Start Network Task)로 구분된다. 네트워크 시작과 관련해서 노드 탐색 과정은 다음 절에서 자세히 설명한다.

5.2 노드 탐색 과정

노드의 기본설정이 완료되어 이더리움 네트워크에 참여하기 위한 함수 호출은 그림 7과 같다. ListenUDP 함수를 통해 다른 노드의 요청을 처리하는 함수(readLoop)와 내부적으로 발생하는 요청 작업

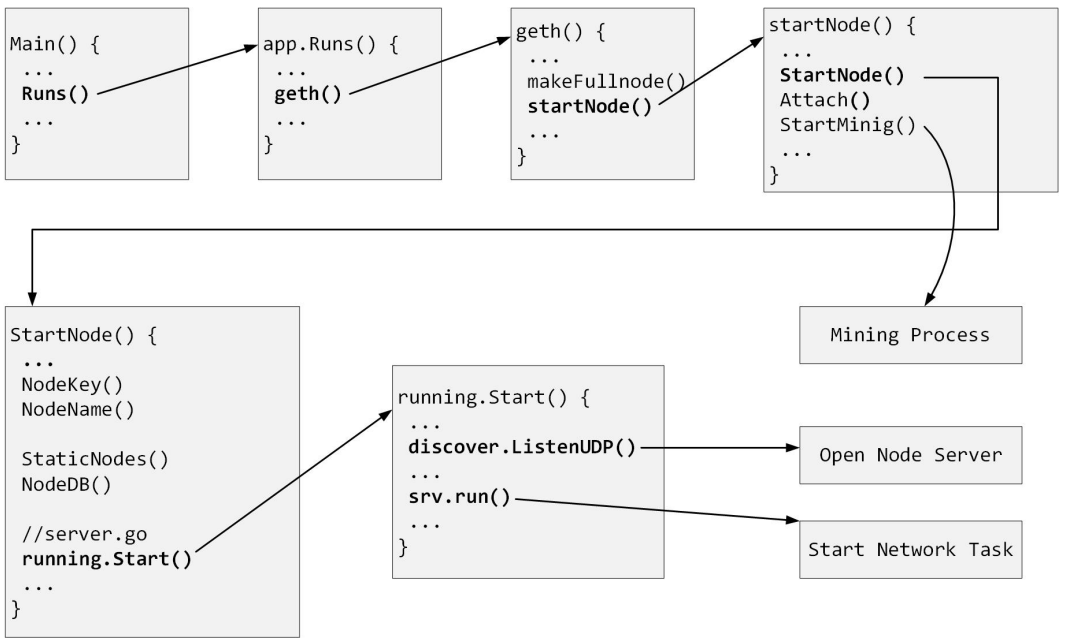

그림 6. Geth 클라이언트의 초기화
Fig. 6. Initialization of Geth 클라이언트

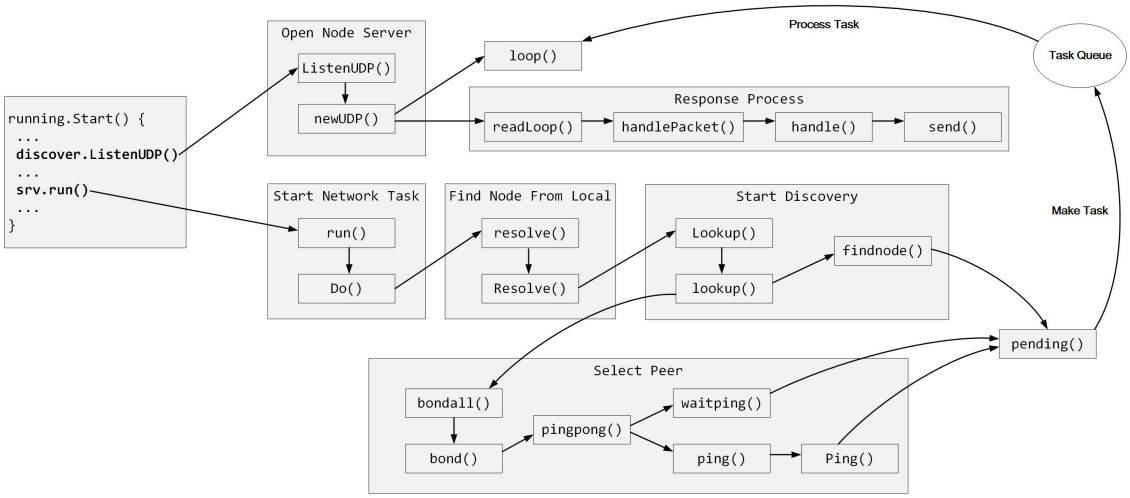

그림 7. Geth 클라이언트의 초기화
Fig. 7. Initialization of Geth 클라이언트

을 처리하는 노드 서버(loop)를 실행한다. run함수는 노드 탐색을 수행하기 위한 함수로 로컬에 저장되어 있는 노드의 정보를 읽어와 버켓 테이블을 구성하는 함수(resolve, Resolve)를 통해 노드 탐색을 시작한다. lookup 함수에서 버켓이 비어있는 경우 findnode 함수를 호출하여 노드 탐색 요청 작업을 생성한다.

또한 bondall 함수를 호출하여 로컬에서 읽어온 노드와 새로 탐색된 노드들이 온라인 상태인지 확인하는 pingpong 함수를 호출한다. 결과적으로 ping함수를 통해 다른 이더리움 노드의 상태를 확인하는 작업을 생성한다. 생성된 작업은 pending함수에서 작업 리스트를 구성하며, 노드 서버 실행 과정(Open Node Server)에서 실행한 loop함수가 작업 리스트에 맞는 네트워크 메시지를 전송하여 처리한다.

이더리움 블록체인 플랫폼의 실제 구현물을 분석하여 노드 탐색과 관련된 함수의 호출을 그림 7과 같이 나타내었다. 본 장과 같이 노드 탐색과 관련된 실제 함수 부분을 알아내어 새로운 노드 및 노드 관리와 관련된 부분(Select Peer)에서 보다 엄격한 조건을 확인하는 과정을 추가하게 되면, 1장 2절에서 살펴본 이더리움의 네트워크를 대상으로 하는 다양한 공격을 완화 할 수 있다. 추가적으로 데이터 통신을 분석하여 네트워크 계층에서 노드의 행동을 모니터링하고 분석할 수 있으면 더욱 적극적으로 악의적인 노드를 탐지하여 격리할 수 있다.

Ⅵ. 결 론

다양한 서비스 구현의 기반기술로 블록체인 기술 활용이 증가하는 추세이다. 하지만 블록체인 플랫폼 기술에 대해 구현된 코드 수준에서 보안성을 고려하는 연구는 미비한 상태이다. 본 논문에서는 대표적인 블록체인 플랫폼인 이더리움에 대해서 네트워크 단계의 공격 사례를 살펴보고, Go언어로 구현된 코드의 네트워크 관련 기능에 대해서 안전한 서비스 구축을 위한 분석을 수행하였다. 따라서 이클립스 공격의 특징을 이해하고, 구현 시 이클립스 공격이 가능한 요소를 보완하여 위험성을 제거할 수 있다. 또한 코드수준 분석을 통해 블록체인 플랫폼을 구현상 취약점을 보완한다면, 블록체인 기술의 장점은 유지하고 공격 위험성은 완화된 블록체인 서비스를 구축할 수 있다. 본 연구를 기반으로 노드의 이더리움 네트워크 연결 후 데이터 송수신 과정을 분석하여 악의적인 행동을 하는 노드를 탐지하는 연구 및 블록체인 트랜잭션에 대한 침입탐지 연구로 발전할 수 있다.

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system(2018), Retrieved May, 1, 2018, from https://bitcoin.org/bitcoin.pdf.

[2] V. Buterin, A next-generation smart contract and decentralized application platform(2014), Retrived Apr, 1, 2018, from http://blockchainl

ab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf.

[3] J.-H. Lee, "BIDaaS: Blockchain based ID as a service," *IEEE Access*, vol. 6, pp. 2274-2278, Dec. 2017.

[4] C. McFarlane, et al., "Blockchain-based secure firmware update for embedded devices in an Internet of Things environment," *The J. Supercomputing*, vol. 73, no. 3, pp. 1152-1167, 2017.

[5] McFarlane, Chrissa, et al., Patientory: A healthcare peer-to-peer EMR storage network v1.(2017), Apr., 1, 2018, from https://www.interoperabilityshowcase.org/sites/interoperabilityshowcase/files/patientory_whitepaper_1.pdf

[6] Y. Lim, et al., "Blockchain Based Microgrid Power Trading System," Proceedings of Symposium of the Korean Institute of communications and Information Sciences, pp. 271~272, Kangwon, Korea, Jan, 2018.

[7] S. Lee, et al., "Security analysis of blockchain systems: Case study of cryptocurrencies," *Journal of the Korea Institute of Information Security & Cryptology*, vol. 28, no. 1, pp. 5-14, 2018.

[8] Y. Marcus, et al., Low-resource eclipse attacks on ethereum's peer-to-peer network(2018), Apr., 1, 2018, from http://www.cs.bu.edu/~goldbe/projects/eclipseEth.pdf

[9] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," International Workshop on Peer-to-Peer Systems, Berlin, Germany, Mar, 2002.

[10] Ethereum RLPx Protocol, Retrieved Apr., 1, 2018, from https://github.com/ethereum/devp2p/blob/master/rlpx.md.

[11] Ethereum Wire Protocol, Retrieved Apr., 1, 2018, from https://github.com/ethereum/wiki/wiki/%C3%90%CE%9EVp2p-Wire-Protocol.

[12] Go Ethereum, Retrieved Apr., 1, 2018, from https://github.com/ethereum/go-ethereum.

명 세 인 (Sein Myung)

2018년 2월 : 상명대학교 공학사
2018년 8월~현재 : 상명대학교 석사과정
<관심분야> 네트워크 보안, 블록체인

이 종 혁 (Jong-Hyouk Lee)

2010년 2월 : 성균관대학교 공학박사
2009년 6월~2012년 2월 : 프랑스 INRIA 연구원
2012년 3월~2013년 8월 : 프랑스 그랑제꼴 TELECOM Bretagne 조교수
2013년 9월~현재 : 상명대학교 소프트웨어학과 조교수
<관심분야> 정보보안, 프로토콜 분석, 블록체인