DApp을 활용한 취약점 분산 관리 솔루션 제안

조민주*, 임재원*, 김병욱*‡
*동국대학교 경주캠퍼스 컴퓨터공학과
e-mail: jmxx.98@gmail.com, jeawon139@gmail.com, bwkim@dongguk.ac.kr

A Proposition distributed vulnerability management solution using DApp system

Min-Ju Jo*, Jae-Won Lim*, Byoungwook Kim*‡
*Dept of Computer Engineering, Dongguk University-Gyeongju

요 약

최근 다양한 보안 사고들이 발생함에 따라 보안의 중요성이 대두되고 분야를 가리지 않고 일어나는 보안 이슈에 대응하기 위한 개발이 다양화되고 있다. 이 중 블록체인 기반 암호화폐의 일종인 이더리움(Ethereum)의 탈중앙화 어플리케이션(DApp) 시스템이 이슈화되었다. DApp을 이용하여 개발된 취약점 분산 관리 솔루션은 DApp의 기능과 장점을 포함한다. 취약점 분산관리 솔루션은 현 시대의 데이터 관리 구조의 문제점을 해결하기 위한 대책으로 제시되며 필요성을 드러낸다. 또한, 기존의 중앙 집중형 시스템에서 벗어나 솔루션의 이점을 활용하여 사용 영역을 확장할 수 있다.

1. 서론

이더리움은 2015년 Vitalik Buterin이 공개한 암호화폐이며 플랫폼이다. 하지만 이더리움의 진정한 가치는 매우 다양한 스마트 컨트랙트(Smart contract)[1]와 이를 바탕으로 한 탈중앙화된 어플리케이션(DApp)을 만들 수 있는 플랫폼이라는 점에 있다. 이더리움은 블록체인 개발 환경을 통합적이고 상호 유기적으로 발전시킬 수 있도록 개발되었다. 특수한 기능 구현을 위한 별도의 체인을 만들 필요 없이 이더리움 체인을 기반으로 한 공통의 개발환경을 사용한다. 즉, 이더리움은 기존에 비해 효율적으로 다양한 종류의 탈중앙화된 어플리케이션(DApp)을 만들 수 있도록 해주는 베이스 플랫폼이라고 할 수 있다. 따라서 현 시대의 중앙 집중형으로 구성된 데이터 관리 구조의 문제점을 해결하고자 DApp을 활용한 취약점 분산 관리 솔루션을 제안한다.

2. DApp 소개

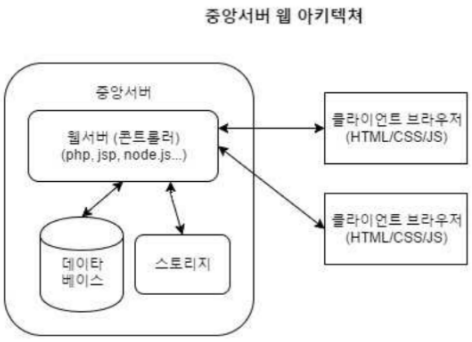
2.1 탈중앙화 어플리케이션(DApp)

탈중앙화 어플리케이션(DApp)이란 이더리움을 사용할 수 있는 어플리케이션을 보다 효율적으로 만들기 위한 시스템이라 할 수 있다. DApp 사용의 최대 장점은 블록체인을 이용해 어플리케이션을 구현하려고 하는 개발자들이 쉽게 체인 자체를 개발하고 운영하는 수고를 대폭 줄어준다는 점이다. 비유를 하자면 기존 웹사이트 하나를 만들기 위해서 웹서버 자체를 개발자 본인이 전부 코딩해서 만드는 것이 아닌 기존에 있던 서버 구현 노하우가 있는 상태라 할 수 있다. 이러한 노하우가 있다면 웹서버 로직 자체를 알지 못해도 웹 사이트를 만드는 일에 별 장애가 되지 않는 것과 같다. 이와 마찬가지로 DApp을 돌아가게 하는 모든 내부 매커니즘과 코드를 일일이 다 들추어서 보여주지 않더라도, DApp의 기본적인 구성과 작동 매커니즘을 보여줄 수 있으면 간편하게 개발이 가능하다. 프로그래밍의 수준이 전문 개발자들에 비해 미흡한 수준이어도, DApp의 개발이 가능한 것은 바로 이러한 이더리움 플랫폼이 가진 장점이라 할 수 있다. 그만큼 탈중앙화 어플리케이션 개발에 참여할 수 있는 문턱이 낮아졌다는 것을 의미하기도 한다.

2.2 DApp 아키텍처 구조

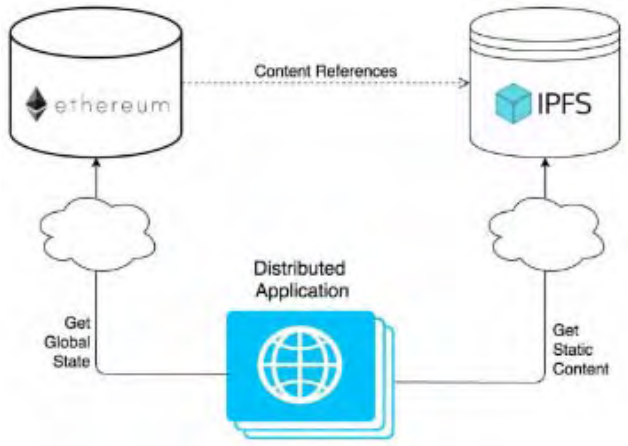
중앙서버는 한 회사나 조직에 의해 소유, 관리, 통제된다. 이 중앙서버에는 웹서버라는 것이 있다. 이 웹서버는 외부의 클라이언트들이 보낸 요청을 받아서, 이를 가공해 데이터베이스에 저장하거나, 또는 클라이언트가 요청한 정보를 데이터베이스로부터 검색, 가공해 다시 클라이언트로 보내주는 역할을 한다.

1) 계약에 필요한 요소들을 자동화하고 대체하는 컴퓨터 프로토콜



<그림 1> 중앙서버 웹 아키텍처

그러나 많은 사용자들의 정보가 중앙서버에 전부 모이는 모델은 늘 내·외부 집단에 의한 해킹과 데이터 유출에 대한 위험성을 포함해 정부와 소수 독점기업에 의한 개인 사생활에 대한 침해, 감시에 대한 우려도 높아지는 등 다양한 문제점이 지적되고 있다. 탈중앙화된 어플리케이션은 중앙화된 모델에서 파생하는 문제들에 대한 대안으로써 부각되기 시작했다.



<그림 2> DAapp 웹 아키텍처

[그림 1]의 중앙화된 모델과 비교해서 가장 큰 차이점은 중앙 서버가 없다는 점이다. 각각의 사용자가 모두 독립노드가 되어서 똑같은 역할을 하게 된다. 데이터를 중앙서버가 가지는 것이 아니라, 각 노드들이 동일한 복제본을 각각 보관하는 것이다. 각 노드들이 가진 데이터를 지속적인 싱크를 통해서 일치시켜 나감으로써 데이터의 정합성을 유지하는 모델이라 할 수 있다.

2.3 DApp 정의

응용 프로그램이 DApp으로 간주되기 위해서는 4가지 기준을 충족해야 한다. 첫째로 완전히 오픈소스여야 하고 자율적으로 작동해야 하며 토큰의 대부분을 제어하는 엔티티가 존재하지 않아야 한다. 둘째로 프로그램의 데이터와 작업 기록은 중앙 집중식 실패 원인을 피하기 위해 공개 분산된 블록체인에 암호로 저장되어야 한다. 셋째로 프로그램은 프로그램에 액세스하는데 필요한 암호화 토큰(이더리움의 고유 토큰)을 사용해야 하며 프로그램의 토큰에서 기여도를 인정받아 보상 받아야 한다. 분산처리 구조가 분산처리에 참여하는 개인의 리소스를 사용하는 구조이므로 사용하는 리소스에 대한 보상이 필요하기 때문이다. 또한 제작한 프로그램은 노드가 프로그램에 기여하는 값을 증명하는 표준 crytptographic 알고리즘에 따라 토큰을 생성해야 한다.

3. 취약점 분산 관리 솔루션

3.1 솔루션 소개

본 솔루션은 취약점 분산 관리 플랫폼의 이름으로 IT 공급업체들을 위한 분권화된 네트워크에서 돌아가는 취약점 피드 관리를 목적으로 DApp 시스템을 이용해 만들어졌다. 현재의 취약점 관리 플랫폼들은 전부 중앙 집중형의 저장소를 생성하여 IT제품에서 식별된 취약점들을 파악해 문제를 해결하고 있다. 편리함 때문에 많은 기업이 이용하고 있지만 데이터의 조작과 도난이 쉽고 서버 손상으로 인해 모든 서비스가 종료할 가능성이 있는 등, 취약점과 같은 민감한 데이터를 다루기에 적합하지 않은 모델이다. 이를테면 NIST나 CERT가 관리하는 데이터베이스는 전부 중앙 집중형이라 할 수 있다. 따라서 본 솔루션은 이전 중앙 집중형의 불안전한 관리 환경에서 벗어나 분산 관리 형태로 여러 노드를 통해 안전하게 데이터를 관리할 수 있다. 본 솔루션은 취약점에 대한 정보를 투명하게 공유 가능하고 안정성과 신뢰도 향상에 기여할 수 있는 취약점 관리 솔루션으로써 적합함을 알 수 있다. 솔루션의 구조를 이루고 있는 파일은 GitHub에서 배포중이며 (https://github.com/JMingMing2/KIPS_-Using-DApp-) web3콘솔을 이용하여 상호작용 및 유지 스크립트를 사용하고 있다.

3.2 솔루션 장점

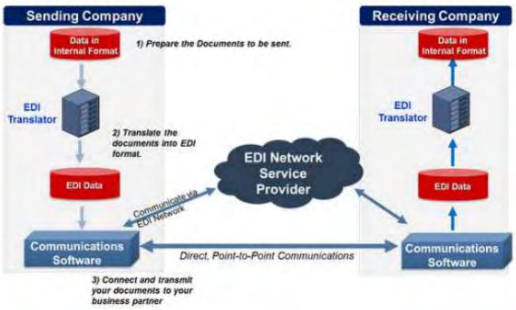
본 솔루션의 장점은 아래와 같다.

3.2.1 경제적 비용의 절감

DApp을 이용한 본 솔루션을 구축하여 분산 관리 솔루션을 이용하게 된다면 현재 쓰이고 있는 중앙 집중식 시스템에 들어가는 관리비용보다 훨씬 적은 비용으로 운용이 가능해진다. 이는 이미 준비가 되어있는 시스템을 이용함으로써 기회비용과 실질적 관리비의 절감이 가능하다는 것이다. 분산처리 시스템의 장점을 설명하며 나오는 Grosh법칙에 근거하여 저가의 CPU를 통해 고가의 단일 CPU보다 우수한 성능의 시스템을 제공할 수도 있다는 것을 설명할 수도 있다. CPU의 계산능력은 그 값의 제곱에 비례하기 때문에 2배의 값을 지불함으로써 성능은 4배의 효과를 얻을 수 있다. 그러나 현재는 마이크로 프로세스의 발전으로 2배 가격의 CPU가 4배의 성능을 제공하지 않으므로 저가격이면서 다수의 CPU로 상호 동작하는 시스템이 훨씬 효과적인 비용 해결책이라는 것이다.

3.2.2 데이터에 대한 보편적이고 균일한 접근

수많은 데이터 소스에 걸쳐 연결성과 관리 용이성을 설명하는 개념이다. EII(Enterprise information integration)및 EDI(Electronic data interchange)와 같은 분야의 경우 , 상이한 데이터 유형 및 데이터 소스 분석과 관련하여 이 개념이 사용되기도 한다. EII(Enterprise information integration)는 기업 정보 통합이란 단어의 약자로, 이는 전체 조직에 대한 데이터 및 정보의 통합 된 뷰를 지원하는 기능이며, EDI(Electronic data interchange)는 전자 데이터 교환이란 단어의 약자로, 이는 모든 전자 수단을 통해 데이터를 교환하기 위한 표준을 제공하는 전자통신 메소드이다. 두 기능의 공통점은 크게 데이터 해석기능을 가지고 있다는 점에 있는데 EII의 경우 서로 다른 데이터 세트를 결합하는 기능에 사용하며 EDI는 내부 시스템과 송수신되는 EDI 형식 간의 인터페이스를 제공하는데 이때 파일의 구조를 체크할 때 데이터 해석 기능을 사용한다.



<그림 3> EDI 시스템 구조도

EDI 소프트웨어의 경우 서로 다른 데이터 세트 간에 데이터를 이동시킬 경우, 모든 단계에 대해 완벽한 감사(監査)를 실시한다. 감사를 통해 이동하는 데이터가 손실되지 않도록 하는 것이다. 이와 같이 데이터에 대한 보편적이고 균일한 접근은 다루는 데이터에 대해 안정성과 신뢰도를 부여하고 더욱 안전한 관리가 이루어질 수 있도록 할 수 있다.

3.3 Solidity 구조

Solidity는 이더리움의 내장 고유 언어로 어떠한 어플리케이션이든 개발 가능하며 다른 언어에서 할 수 있는 대부분을 구현 가능하다는 장점을 가진다. 취약점 자료를 업로드 하는데 쓰일 solidity 파일 구조는 아래와 같다.

```
// 제품 회사(vender 포함)
struct Company
{
    uint c_id;
    string c_name;
    string c_details;
    string c_link;
    address c_addr;
}

// 취약점 관련 제품
struct Product
{
    uint p_id;
    uint p_c_id;
    string p_name;
    string p_details;
    string p_version;
    string p_link;
}

// 제품 취약점
struct Vuln
{
    uint vl_id;
    string v_name;
    string v_link;
    uint v_p_id;
    uint v_c_id;
    uint v_cve_num; // CVE 번호 관리
    string v_poc;
}
```

<그림 4> 구조체 선언

위 구조체는 제품에 대한 회사, 그리고 회사의 제품과 제품에 대한 취약점을 입력받고 관리할 수 있도록 해놓았다. 해당 구조체의 변수들을 통해 추가적인 함수를 제작하여 각종 정보들을 추가할 수 있다. solidity의 struct는 Python이나 swift언어의 class와 그 역할이 거의 동일하다고 볼 수 있다.

```
37    mapping (uint => Company) Companys;
38    mapping (uint => Product) Products;
39    mapping (uint => Vuln) Vulns;
```

<그림 5> 구조체 매핑

이후 mapping 함수를 사용하여 정의한 구조체들에 대해 상태변수를 선언한다.

```
33  function register_vendor ( string c_name , string c_details , string c_link)
34  { ... }
35
36  function add_prod(string p_name , string p_details , string p_version , string p_link , uint p_c_id )
37  { ... }
38
39  function add_vuln(string v_name , string v_link , uint v_p_id , uint v_c_id ,uint v_cvss )
40  { ... }
41
42  function list_vendor_details(uint co_id)
43  { ... }
44
45  function list_products_details(uint p_id)
46  { ... }
47
48  function list_vuln_details(uint vl_id)
49  { ... }
```

<그림 6> 정보 입력 및 관리 함수

이후 함수들이 정의된다. 제품 회사를 등록한 뒤 제품과 취약점을 추가하고 이후 필요하면 각 정보들을 리스트업 할 수 있도록 정의되어 있다. 그 중 취약점에 대한 정보를 입력받는 add_vuln 함수의 구조는 아래 <그림 7>와 같다.

```
// 취약점 등록
function add_vuln(string v_name, string v_link, uint v_p_id, uint v_c_id, uint v_cve_num, v_poc) returns (uint vl_id)
{
    if(Companys[v_c_id].c_addr==msg.sender)
    {
        if (Products[v_p_id].p_id==v_p_id)
        {
            vl_id = vl_no++;
            Vulns[vl_id] = Vuln(vl_id, v_name, v_link, v_p_id, v_c_id, v_cve_num, v_poc);
        }
    }
}
```

<그림 7> add_vuln 함수

add_vuln 함수는 정보수집에 필요한 요소들을 인자로 받고 vl_id값을 반환한다. 누적되는 취약점 정보들을 처리하기 위해 vl_id 값을 누적시키며 상태변수로 선언한 Vulns에 인자로 받은 정보들을 넘기는 것을 확인할 수 있다. 함수 내부에는 벤더사의 c_addr값과 msg.sender값을 대조하는 if문이 보이는데 msg.sender는 contract에 당사자의 정보를 요청하는 일종의 API와 같다.

3.4 주 이용 대상

최근 다양한 보안 사고들이 발생하며 분야를 가리지 않고 보안이슈에 대응할 방안을 마련하는 추세이다. 본 솔루션은 좀 더 나아가 제품 취약점 자체에 대한 데이터를 관리하는 시스템이기 때문에 IT와 관련되어 있는 모든 분야에 폭넓게 사용될 수 있다.

3.4.1 IT기업

최근 IoT 시장이 활발히 움직이며 빠른 성장세를 보였

으나 동시에 그 보안 위험성 또한 증가하는 모습을 확인할 수 있었다. 때문에 국내에서는 한국인터넷진흥원(KISA)에서 IoT 기기를 대상으로 한 버그바운티 제도 즉, 취약점 포상제를 운영하여 IoT 기기 취약점을 찾은 분석가에 대해 찾은 취약점에 대한 가격을 매기고 포상을 제공하는 제도를 운영 중이다.

따라서 국내의 이러한 분위기를 반영하여 타 기업에서도 위와 같이 제품에 대한 취약점을 신고 받고, 취약점 제보에 대해 포상하는 제도를 따라 운영하는 모습을 보이고 있다. 취약점에 대한 데이터를 안전하게 관리하기 위해서는 IT 제품을 판매하는 기업에서도 본 솔루션을 이용할 가치가 있다.

3.4.2 CERT(Computer Emergency Response Team)

CERT는 직역하자면 침해 사고 대응 팀으로, 보안 사고에 대응하기 위해 기업이나 기관의 업무관할 지역 내에서 침해서고의 접수 및 처리 지원을 비롯하여 침해사고 예방, 피해 복구 등의 업무를 수행하는 조직을 말한다. CERT에서는 실제로 보안 솔루션을 운영하기도 하고 주기적인 보안 취약성에 대해 점검하는 업무를 진행하기 때문에 업무 중 자연스럽게 취약점에 대한 정보를 담은 데이터들을 접할 수 밖에 없는 구조에 있다. 또한 국가기관에 속한 CERT도 존재하는 만큼, 보다 안전한 데이터 관리가 필수적이라 할 수 있다.

3.4.3 보안 취약점 분석 업체

보안 컨설팅 업체와 같이 보안 취약점 분석을 주 업무로 삼는 기업들이 존재한다. 이들은 취약점 리포트를 위해 데이터를 안전하게 관리할 수 있는 솔루션이 필요하다. 특히 현재는 보안 이슈에 대한 리포트를 공유하는 사이트들이 많이 존재한다. 본 솔루션을 이용하면 이들의 정보 공유에 있어서 안전성을 보장해 보안 리포트를 이용하는 고객들에게도 보다 섬세한 정보를 넘겨줄 수 있도록 발전할 수 있는 가능성을 열어준다.

3.4.4 CVE Number 발급

CVE(Common Vulnerabilities and Exposure)는 공개적으로 알려진 소프트웨어의 보안 취약점을 가리키는 고유 표기이다. CVE는 CVE를 관리하는 MITRE Corporation에서 논의를 거쳐 번호를 발급하는데 이 경우 논의가 진행될 때, 그리고 등록된 뒤 리스트에 들어가 많은 업체들에 의해 해당 데이터가 공유되는 순간들이 발생한다. 때문에 민감한 정보를 수시로 공유하게 되는 CVE 발급 작업 역시 본 솔루션이 활약할 수 있는 분야이다.

3.5 솔루션 사용 예

3.5.1 보안 취약점 관리 스타트업

본 솔루션을 이용하여 취약점 관리 플랫폼을 개설하게 된다면 이더리움의 블록체인을 이용해 피드를 관리한다. 각각의 공급업체들이 관리하는 제품들의 취약점, 버그에 대한 업데이트를 피드를 사용하여 알릴 수 있다. 그러기 위해서는 우선 제품 취약점을 관리할 관리자를 등록하고 제품을 보유한 IT 공급업체를 등록시킨다. 그리고 사용자가 만든 IT 제품을 등록하고 제품정보를 업데이트 한다. 그 뒤 취약점을 릴리즈 하고 피드를 통해 취약점 업데이트 사실을 알린다. 이런 식으로 한 회사에 국한되어 있지 않고 다른 회사들의 취약점과 버그를 관리해주는 스타트업에 대한 계획이 가능하다.

4. 결론

본 논문에서는 암호화폐의 일종인 이더리움의 DApp 시스템에 대해 알아보고 DApp 시스템을 이용한 취약점 분산 관리 솔루션과 그 필요성에 대해 다뤘다. 이러한 분산 관리 시스템은 점점 중요해져가는 정보자산에 대한 위협 대응으로써 적합하다 판단되며 향후 CVSS 유형 채점 매커니즘을 추가해 갖가지 보안위협의 클래스를 나눠 좀 더 편리한 대응이 가능해지도록 하고 또한 분산된 세계에 MAEC 등 보안 인텔리전스와 같은 다양한 방법을 추가하기로 하였다. 이 연구를 통해 미래의 보안은 분권화 되어 있는 네트워크에서 좀 더 안전하게 데이터를 관리할 수 있고 빠른 대응을 통해 IT제품들의 안정성과 신뢰도가 더욱 향상될 수 있으리라 기대할 수 있었다.

참고문헌

[1] Vitalik Buterin, A Next-Generation Smart Contract and Decentralized Application Platform, ethereum white paper, March, 2015.

[2] Lee-Je young, Moon-Hong Ju, Moon-Sang yong, Gwon-Wook Hyuen, Park-Ig Soo, Token-passing Bus Access Method on IEEE 802.3 Physical Layer for Control Networks of a Distributed Control System, The Korean Institute of Electrical Engineers Conference Proceedings, November, 1998.

[3] Kim-Byung Hee, Modeling Smart Contract by Timed Automata, Graduate School of Engineering, Korea University Thesis, 2017.

[4] Kim-Joo Hwan, Jo-Ju Young, Choi-Seung Hyeon, Lee-Jae Hwan, A Mesos-based distributed processing system using single-board computers for In-Storage Processing, 2016 Korea Computer Engineering Conference, July, 2016.

[5] “Decentralized Applications” https://due.com/blog/why-build-decentralized-applications-understanding-dapps/