클라우드 환경에서의 크리덴셜 스터핑 방지를 위한 인증 프로토콜 개선 방안

조정석*, 곽 진**
*아주대학교 컴퓨터공학과 정보보호응용및보증연구실,
**아주대학교 사이버보안학과
*zzi0083@ajou.ac.kr, **secuity@ajou.ac.kr

Improvement of Authentication Protocol for Preventrion of Credential Stuffing in Cloud Environment

Jeong-Seok Jo*, Jin Kwak**
*ISAA Lab., Department of Computer Engineering, Ajou University.
**Department of Cyber Security, Ajou University.

요 약

다양한 서비스 제공자들은 서비스 제공과 비용의 효율성을 위해 클라우드 환경을 활용하여 서비스를 제공하고자 한다. 하지만 국내의 경우 많은 서비스 제공자들이 클라우드 환경에 대한 의심으로 클라우드 환경 적용에 대한 확신을 가지지 못하고 있다. 대표적으로 서비스 제공자들은 클라우드 환경에서의 정보 유출이 발생할 경우, 기존의 서비스 제공 환경보다 더 큰 피해가 발생할 것이라고 생각하고 있다. 클라우드 환경의 취약점 중 하나는 인증이다. 클라우드 환경은 가용성을 위해 높은 수준의 인증을 수행하지 않는다. 그에 대한 예로 새로운 사용자가 추가되었을 경우 사용자의 정확한 목적을 파악하지 못한 채 서비스를 제공해준다. 또한 기존의 네트워크 환경에서 클라우드 환경으로 변화하면서 새로운 취약점이 등장하고 있으며, 그에 대한 대응책이 요구된다. 따라서 해당 논문에서는 클라우드 환경에서의 인증 개선 방안을 제시하고자 한다.

1. 서론

클라우드 서비스가 등장한 뒤 클라우드 서비스는 점차 확대되고 있으며, 매년 클라우드 시장의 성장이 예측된다. Gartner는 2021년 3천억 달러 상당의 클라우드 시장 성장을 예측했다[1].

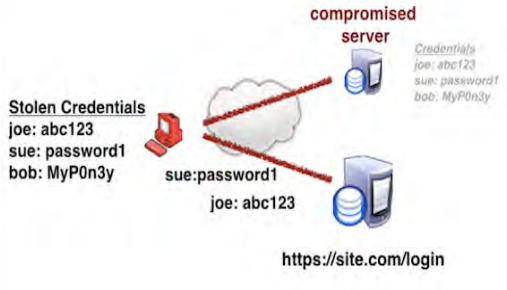
클라우드 서비스 성장함에 따라 다양한 논문이 클라우드 보안 요구사항(기밀성, 프라이버시, 무결성, 인증 등)을 제시하였고[2] 본 논문에서는 인증의 측면에서의 취약점을 분석하고 그에 대한 대응 방안을 제시하고자 한다.

본 논문에서는 2장에서 Lexus가 제시한 인증 프로토콜을 제시하고 해당 프로토콜의 취약점을 제시하였으며, 3장에서 개선된 인증 프로토콜을 제시하고, 4장에서 제시한 인증 프로토콜 안전성을 분석하였다.

2. 관련 연구

2.1 Credential Stuffing Attack

최근 클라우드 환경에서의 Credential Stuffing Attack이 발생하고 있다. Credential Stuffing Attack은 계정에 접근하기 위해 ID/PW 쌍을 자동으로 주입하는 공격이다[3]. 이를 이용해 공격자들은 Client의 ID/PW 쌍을 확보한다. 공격 방식은 (그림 1)과 같다.



(그림 1) Credential Stuffing Attack

Credential stuffing Attack 사례로 2017년 9월 알툴즈에서 다수의 ID/PW, 알패스가 유출되는 사건이 발생하였다. 약 13만 사용자들의 ID/PW가 유출되었으며, 지난 몇 년간 발생한 개인정보 침해사고에서 유출된 개인정보 무작위적 대입을 통한 공격방식으로, ID/PW 유출이 발생하였다[4].

Credential Stuffing Attack 방지를 위한 방안으로 MFA(Multi-Factor Authentication), Multi-Step Login Process, IP Blacklists 등이 있다. 본 논문에서는 MFA를

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학 ICT연구센터육성 지원사업의 연구결과로 수행되었음(IITP-2018-2015-0-00403)

활용한 인증 프로토콜을 제안한다[5].

2.2 Lexus Proposed Protocol

Lexus Jun Hong Sim, Shu Qin Ren 등[6]는 클라우드 환경에서의 인증 방안을 새롭게 제안하였다. 제안한 프로토콜은 (그림 2)와 같다.

본 절에서는 Lexus가 제안한 프로토콜을 분석하고, Credential Stuffing 공격을 통한 취약점을 분석한다.

2.2.1 프로토콜에 사용되는 파라미터

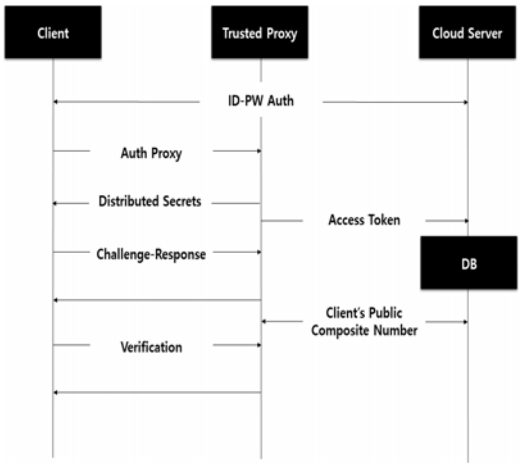
Lexus Jun hong Sim, Shu Qin Ren 등[6]가 제안한 프로토콜에서 사용되는 파라미터는 <표 1>과 같다.

P_{i_i}, P_{i_j}, S 3가지의 값들은 Client's Secrets에 속하며, Client가 안전히 보관하여야 한다.

I_i 는 S의 검증을 위해 사용되며, N_i는 사용자 구분을 위해 사용된다.

<표 1> Lexus Protocol Parameters

Parameter	Contents
P_{i_i}, P_{i_j}	- Randomly Generated Large Prime Number at least 1024-bits
N_i	- Product of the Primes $P_{i_i} \times P_{i_j}$
S	- Randomly Generated Number($1<S<N_i$)
I_i	- $I_i = S^2 \bmod N_i$



(그림 2) Lexus Proposed Protocol

2.2.2 프로토콜 동작 순서

Lexus Jun Hong Sim, Shu Qin Ren 등[6]가 제안한 프로토콜 동작 순서는 아래와 같다.

Step 1. ID-PW Auth(Client→Server)

Client가 Server에게 ID-PW Authentication Protocol을 이용하여 인증을 수행하며, Access Token 요청

Step 2. Auth Proxy(Client→Proxy)

앞서 수신한 Access Token을 활용하여 Proxy 인증을 수행

Step 3. Distributed Secrets(Proxy→Client)

Trusted Proxy는 생성한 P_{i_i}, P_{i_j}, S 3가지의 Client's Secrets 전송

Step 4. Access Token(Proxy→Server)

Client와 Server가 Proxy를 통과하고 Access Token을 이용해 Proxy를 인증

Step 5. Challenge-Response(Client→Server)

Client와 Proxy 간의 Challenge-Response 수행

Step 6. Client's Public Composite Number(Proxy→DB)

Client 구분을 위해 사용되는 값인 N_i를 DB에 저장

2.2.3 Credential Stuffing Attack 기반 프로토콜 취약점

Lexus가 제안한 프로토콜은 ID-PW를 통해 인증을 수행한 후 Trusted Proxy가 Client에게 3가지의 Client's Secrets을 전송한다.

그에 따라 공격자가 2.1에서 분석한 Credential Stuffing Attack을 활용해 공격자가 Client의 ID-PW를 확보했을 경우, 공격 진행 과정은 (그림 3)과 같으며 구체적인 내용은 아래와 같다.

Step 1. ID-PW Auth(Attacker→Server)

Attacker가 Server에게 확보한 ID/PW를 이용하여 ID-PW Authentication Protocol을 이용하여 인증을 수행하며, Access Token 요청

Step 2. Auth Proxy(Attacker→Proxy)

앞서 수신한 Access Token을 활용하여 Proxy 인증을 수행

Step 3. Distributed Secrets(Proxy→Attacker)

Trusted Proxy는 생성한 P_{i_i}, P_{i_j}, S 3가지의 Client's Secrets 전송

Step 4. Access Token(Proxy→Server)

Client와 Server가 Proxy를 통과하고 Access Token을 이용해 Proxy를 인증

Step 5. Challenge-Response(Attacker→Server)

Attacker와 Proxy 간의 Challenge-Response 수행

Step 6. Client's Public Composite Number(Proxy→DB)

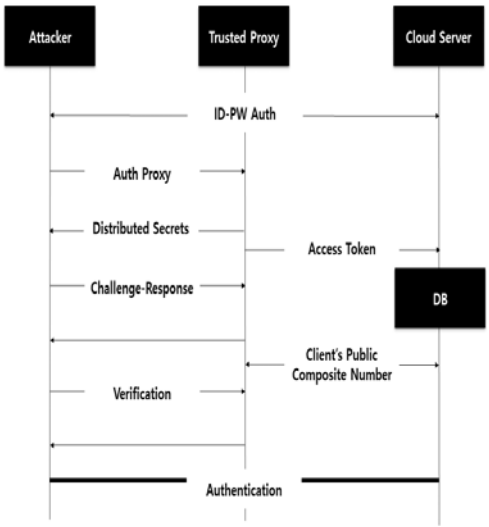
Client 구분을 위해 사용되는 값인 N_i를 DB에 저장

Step 7. Authentication(Attacker)

공격자는 성공적으로 Cloud Server와의 인증 수행하고, Client의 자원에 접근 가능하다.

결과적으로, 공격자가 ID-PW 인증 프로토콜을 성공적으로 수행할 경우 클라우드 서버에 접근 가능하다. 그리고 클라우드 플랫폼의 확장성으로 인해 더 큰 피해가 발생 가능하다.

이러한 취약점으로 인해, Client에 대한 추가적인 인증 요소가 요구된다.



(그림 3) Lexus Attack Model

3. 제안 사항

3.1 개선된 인증 방안 제안

2.2.3에서 제시한 취약점을 개선한 제안사항은 (그림 4)와 같다.

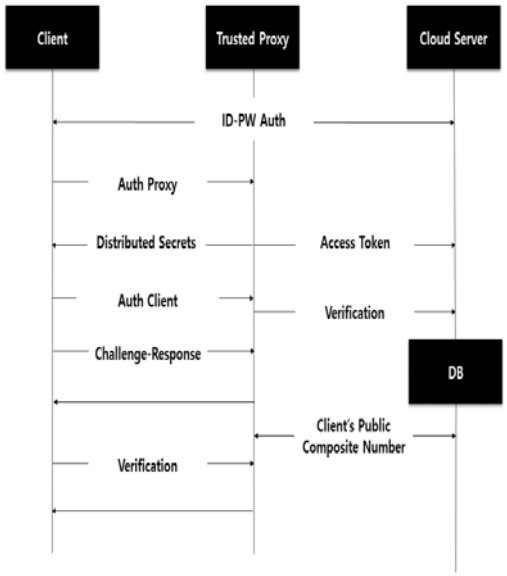
본 절에서 제안하는 프로토콜은 Client의 IP Address, MAC Address와 S를 제공받은 소수를 이용하여 공개키 암호화 방식 통해 생성한 값을 보내어 Client의 ID-PW뿐만이 아닌 추가적인 요소에 대한 검증을 수행한다. 또한 수신한 값들에 대한 검증을 수행한다.

3.1.1 제안 방안에서의 사용되는 파라미터

제안 프로토콜에서 사용되는 파라미터는 <표 2>와 같다

<표 2> Proposed Protocol Parameters

Parameter	Contents
P_{i_i}, P_{i_j}	- Randomly Generated Large Prime Number at least 1024-bits - Using as public key, private key
N_i	- Product of the Primes $P_{i_i} \times P_{i_j}$
S	- Randomly Generated Number($1<S<N_i$)
I_i	- $I_i = S^2 \bmod N_i$
IP Addr	- Client's IP Address
MAC Addr	- Client's MAC Address



(그림 4) Proposed Protocol

3.1.2 제안 방안에서의 프로토콜 동작 순서

본 논문에서 제안한 프로토콜의 동작 순서는 아래와 같다.

Step 1. ~ ***Step 4.***는 Lexus가 제안한 프로토콜과 동일하다.

Step 5. Auth Client(Client→Proxy)

Client의 $E_{p_{i_i}}(IP\,Addr, MAC\,Addr, S)$ 값 송신을 통한 추가적인 인증 수행

Step 6. Verification(Proxy→DB)

Client로부터 수신한 값을 DB와의 통신으로 검증한다.

P_{i_j}를 이용한 복호화를 통해 P_{i_i}를 검증하고, IP Address 와 MAC Address를 확인한다.

$D_{p_{i_j}}(IP\,Addr, MAC\,Addr, S)$

Step 7. Client's Public Composite Number(Proxy→DB)

Client 구분을 위해 사용되는 값인 N_i를 DB에 저장

4. 안전성 분석

2.2.3에서 수행한 공격 모델을 제안한 개선 방안에서 수행한 공격 과정은 (그림 5)와 같으며, 구체적인 내용은 아래와 같다.

Step 1. ID-PW Auth(Attacker→Server)

Attacker가 Server에게 확보한 ID/PW를 이용하여 ID-PW Authentication Protocol을 이용하여 인증을 수행하며, Access Token 요청

Step 2. Auth Proxy(Attacker→Proxy)

앞서 수신한 Access Token을 활용하여 Proxy 인증을 수행

Step 3. Distributed Secrets(Proxy→Attacker)

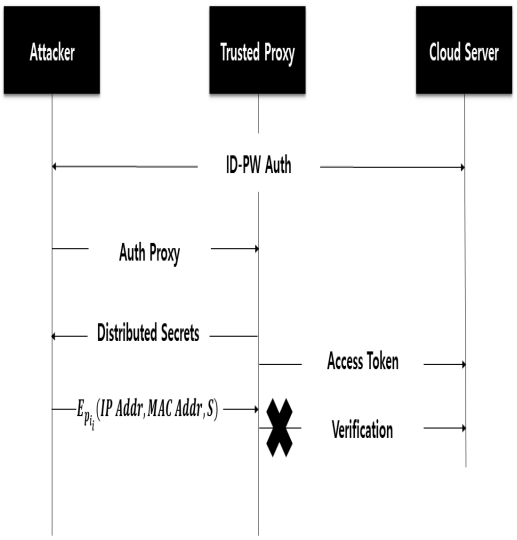
Trusted Proxy는 생성한 P_{i_i}, P_{i_j}, S 3가지의 Client's Secrets 전송

Step 4. Access Token(Proxy→Server)

Client와 Server가 Proxy를 통과하고 Access Token을 이용해 Proxy를 인증

Step 5. Auth Client(Attacker→Proxy)

Attacker는 정상적인 'Client Auth'값 생성이 불가능하므로, $E_{p_{i_i}}(IP Addr, MAC Addr, S)$ 값 전송 후 검증 단계를 통과할 수 없다.



(그림 5) Credential Stuffing Prevention

5. 결론

본 논문을 통해 클라우드 환경에서의 인증 프로토콜을 분석하고 취약점을 제시하여 클라우드 환경 인증 프로토콜 개선의 필요성을 강조하였다.

분석한 클라우드 인증 프로토콜의 추가적인 파라미터 검증 과정과 Client 검증을 위한 검증 요소 추가를 통한 개선된 클라우드 환경 인증 프로토콜을 제안하였다.

해당 개선사항은 Client의 ID-PW만을 활용하여 인증을 수행한 뒤 파라미터를 전송하고 순차적으로 인증 프로토콜을 수행하는 Lexus의 인증 프로토콜에 추가적으로 Client의 IP Address와 MAC Address 값을 활용하여 Client 인증을 위한 요소를 추가하였다. 그리고 수신한 S 값을 이용한 $E_{p_{i_i}}(IP Addr, MAC Addr, S)$ 값을 활용한 Client 인증 단계를 수행하여 Client 인증을 강화하였다.

또한 수신한 Client's Secret인 P_{i_i}, P_{i_j}, S를 활용하여 생성한 값($E_{p_{i_i}}(IP Addr, MAC Addr, S)$)을 이용하여 해당 값들에 대한 검증을 수행하는 단계를 추가하여 Client가 제대로 수신하였는가에 대한 검증을 수행한다.

향후 클라우드 환경에서의 인증 프로토콜의 전반적인 개선방안을 연구할 예정이다.

참고문헌

[1] Gartner,"Gartner Forecasts Worldwide Public Cloud Revenue to Grow 21.4 Percent in 2018", Apr. 2018.

[2] Sarang V. Hatwar,"Cloud Computing Security Aspects, Vulnerabilities and Countermeasures", Jun. 2015.

[3] "Credential Stuffing", OWASP, "https://www.owasp.org/index.php/Credential_stuffing"

[4] "사건 발생 경위 및 분석 진행 상황", 알툴즈, 2018년02월23일

[5] "Credential Stuffing Prevention Cheat Sheet", OWASP,"https://www.owasp.org/index.php/Credential_Stuffing_Prevention_Cheat_Sheet"

[6] Lexus Jun Hong Sim,"A Cloud Authentication Protocol using One-Time Pad", Nov. 2016.