SGX를 이용한 엣지 서버로의 안전한 컴퓨팅 오프로드 방법

A Secure Computation Offloading Scheme based on Trusted-Execution Environment

우병수, 김명수, 김종윤, 김종석, 서의성*)

Byeong-Su Woo, Myeong-Su Kim, Jong-Yoon Kim, Jong-Seok Kim, Eui-Seong Seo

(16419) 수원시 장안구 서부로, 성균관대학교 소프트웨어대학 소프트웨어학과

wbs79@skku.edu, kimmstop@daum.net, kimjy0126@skku.edu, ks77sj@gmail.com, euiseong@skku.edu

요 약

컴퓨팅 오프로드(Computation offloading)를 사용하여 모바일 디바이스(mobile device)가 가지고 있는 컴퓨팅 능력의 한계를 해결하기 위한 여러 연구들이 진행되어왔다. 엣지 컴퓨팅(edge computing)을 사용하면 통신 딜레이를 줄일 수 있어 오프로드의 효율성이 더욱 향상된다. 그러나 엣지 서버(edge server)로의 안전한 컴퓨팅 오프로드를 위해서는 등록해야 하는 함수가 제 3자에게 유출되지 않고 안전하게 엣지 서버로 전달되어야 할 뿐만 아니라, 엣지 서버 내에서는 전달받은 함수의 무결성이 확인되어야 한다. 또한, 함수의 코드가 엣지 서버 내에서 유출되지 않아야 하고, 함수가 실행될 동안에는 클라이언트의 정보가 누출이 되지 않도록 보안이 이루어져야 한다. 본 연구는 위에 나열된 안전한 엣지 서버로의 컴퓨팅 오프로드의 조건을 충족시키기 위해 엣지 서버 측에 SGX를 결합하여 클라이언트의 Remote Procedure Call(RPC)요청을 안전하게 처리할 수 있는 client-vendor-edge server 모델을 제시한다.

키워드: SGX, 컴퓨팅 오프로드, PCL, RPC

1. 서론

CPU의 성능, 메모리의 크기 등이 증가하면서, 모바일 디바이스의 성능은 지난 수십년간 증가해왔다. 그러나 크기, 발열, 전력 소모와 같은 모바일 디바이스의 근본적인 문제로 인하여 성능은 여전히 제한적이다. 이러한 한계점에도 불구하고 어플리케이션이 요구하는 컴퓨팅자원은 증가하고 있다. 이러한 문제를 해결하기 위해 컴퓨팅 오프로드 모델이 제안되었다.

어플리케이션을 개발하고 해당 어플리케이션 서비스를 운영하는 vendor들이 Remote Procedure Call(RPC)과 같은 기법을 이용하여 자신들의 어플리케이션 부분 중 연산량이 많은 함수의 처리 위치를 모바일 디바이스에서 서버로 옮김으로써 함수 처리 속도가 증가했으나 이는 통신 딜레이의 발생을 야기했다. 그리하여 컴퓨팅 오프로드로 발생하는 통신 딜레이를 줄이기 위한 방안 중 하나로 클라이언트와 서버 사이의 물리적 거리를 좁힌 엣지 컴퓨팅(edge computing)이 제시되었고[1], 이를 모바일 디바이스에도 적용하는 연구가 진행되고 있다.

엣지 서버의 자원은 여러 사용자들에 의해 공유되는 자원이기 때문에 보안문제가 발생할 수 있다. 엣지 서버로 오프로드 된 vendor의 코드(code)가 공격자에 의해 탈취되어 vendor의 기술이 누출될 수 있고, 오프로드 된 함수를 사용하는 클라이언트의 데이터 역시 누출될 수 있다. 또한, 오프로드 된 함수가 변조되지 않았는지 해당 함수의 무결성 역시 확인해야한다. 따라서, 본 논문에서 우리는 Intel Software Guard Extension(SGX)을 이용하여 엣지

* 교신저자

서버로 오프로드 되는 코드의 무결성과 기밀성을 유지하기 위한 방법을 제시한다.

2. 배경 및 관련 연구

2.1. Intel SGX

SGX는 인텔 6세대 CPU부터 추가된 보안 명령어 집합(set)이다. 보호하고 싶은 코드를 라이브러리(library) 형태의 so파일로 만든 후 사인(sign)하여 signed.so파일을 만든다. signed.so파일을 이용해 enclave 인스턴스를 격리된 메모리 공간인 EPC에 생성하여 enclave 내의 코드를 enclave 외의 영역으로부터 보호한다. 또한 만들어진 enclave가 의도된 enclave인지 확인하는 attestation 기능과 enclave로 되기 이전의 signed.so파일을 역공학(Reverse engineering)이 불가능하도록 암호화하는 Protected Code Loader(PCL)기능을 지원한다.

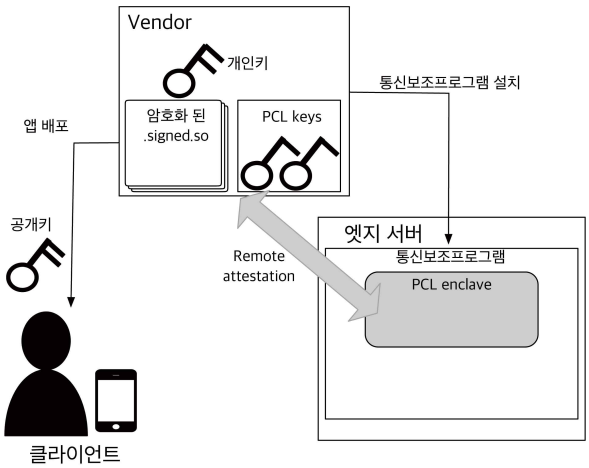
2.2. 관련연구

SGX를 이용해 클라우드(cloud), 엣지 서버와 같은 공용 컴퓨팅 자원에서 클라이언트의 데이터의 누출 없이 안전하게 처리하고, 오프로드 된 프로그램이 SGX가 제공하는 보안성을 유지하며 추가적으로 필요한 기능을 사용할 수 있도록 기능을 추가하는 [2], [3], [4]와 같은 연구들이 진행되었다. 본 연구는 위 연구들처럼 클라이언트의 데이터가 엣지 서버에서 노출이 되지 않도록 기밀성을 보장하는 것에서 더 나아가, enclave 인스턴스가 되기 전의 오프로드 되는 vendor의 함수 코드 또한 노출이 되지 않게 하여 vendor의 기술 역시 누출이 되지 않도록 하는 client-vendor-edge server 모델을 제시한다.

3. 보안성을 확보한 컴퓨팅 오프로드 구조

3.1 제안 방법

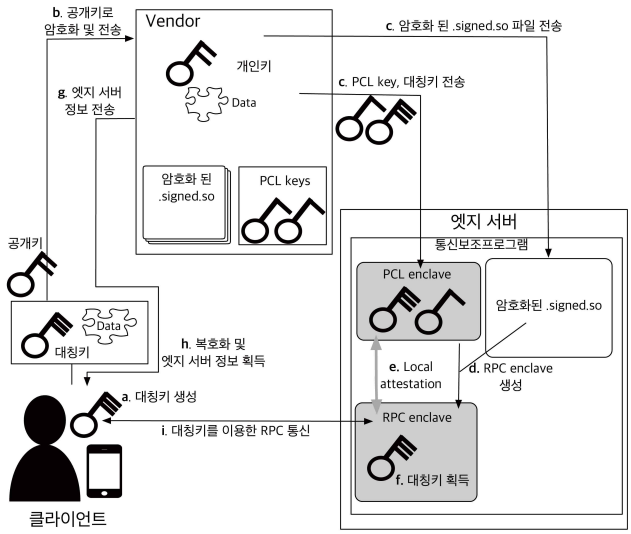
Vendor의 서비스 시작 전, vendor는 엣지 서버로 오프로드 되어 동작될 함수(이하 엣지 함수)들이 서버의 enclave 내부에서 동작할 수 있도록 만들기 위해 후술된 과정을 거쳐 그림 1과 같은 모습으로 만든다. 먼저 엣지 함수들을 이용하여 so파일을 만든 후, PCL을 이용하여 암호화 하고, 마지막으로 사인하여 signed.so형태로 만든 후, PCL키와 함께 vendor의 백엔드(back end) 서버에 저장한다. Vendor는 PCL enclave가 있는 통신보조프로그램을 엣지 서버에 설치하여, 추후 signed.so를 이용해 RPC enclave를 안전하게 생성할 수 있도록 한다. 그 뒤, remote attestation을 통해 vendor와 PCL enclave사이에 안전한 통신 채널(channel)을 만든다. 이 통신 채널은 추후에 Vendor가 엣지 서버로 PCL key와 클라이언트의 대칭키를 전달하는데 사용된다. Vendor는 비대칭키 쌍을 만들고 어플리케이션에 공개키를 내장 시킨 후 배포를 하며 서비스를 시작한다.



(그림 1) vendor의 서비스 시작 전

그림 2는 vendor의 서비스 시작 후, 클라이언트가 RPC통신을 할 엣지 서버를 할당받는 흐름을 나타낸다. 클라이언트가 앱을 실행하면 대칭키가 생성된다. 그 후, 클라이언트 정보와 생성된 대칭키가 공개키로 암호화가 되어 vendor로 전달된다. Vendor는 클라이언트의 정보를 파악하고 엣지 함수를 오프로드 할 엣지 서버를 결정한 뒤, 클라이언트의 대칭키와 PCL key를 PCL enclave에 전달한다. PCL로 암호화가 된 signed.so역시 엣지 서버로 전달한다. PCL enclave는 전달받은 PCL key로 암호화된 signed.so를 복호화 하여 RPC enclave를 만든다. 생성된 RPC enclave와 PCL enclave는 local attestation을 진행하여 서로의 무결성을 확인하고 안전한 통신채널을 만든 후 PCL enclave에 있던 대칭키를 RPC enclave로 전달한다. 성공적으로 대칭키가 RPC enclave에 전달되면 클라이언트와 RPC enclave가 각자 대칭키를 가지고 있는 상태가 된다. vendor는 엣지 서버의 주소를 클라이언트의 대칭키로 암호화를 한 뒤 client로 전달한다. 결과적으로 클라이언트는 RPC 요청을 처리할 엣지

서버를 할당 받고 클라이언트와 RPC enclave는 대칭키를 공유하고 있는 상태가 된다. 그 후 RPC enclave는 클라이언트의 RPC요청을 수행하게 된다. 클라이언트가 앱을 종료하면 enclave 제거가 요청되어 엣지 서버의 RPC enclave가 제거되고 통신이 종료된다.



(그림 2) vendor의 서비스 시작 후

3.2 제안한 기법의 보안성 분석

제안한 기법이 서론에서 제시한 보안문제를 어떻게 해결할 수 있는지 분석하고자 한다. 오프로드하는 함수의 무결성과 기밀성을 보장할 수 있는가, 오프로드된 함수를 호출하여 사용하는 클라이언트의 데이터의 기밀성을 보장할 수 있는가에 대해 분석해보고자 한다. 엣지 서버로 함수를 안전하게 오프로드하고, 오프로드된 함수를 사용하는 클라이언트의 데이터가 엣지 서버영역에서 누출이 되지 않는 것이 주제이므로, 클라이언트와 vendor의 보안은 논외로 한다. 엣지 서버 영역에서는 SGX로 보호받고있는 영역인 enclave내부만 신뢰할 수 있는 영역이고, 그 외의 영역은 신뢰할 수 없는 영역으로 정의한다. 마지막으로, SGX는 하드웨어 레벨의 보안을 제공하는 것이기 때문에 하드웨어기반의 부채널공격(side-channel attack)은 논외로 한다.

Vendor가 엣지 서버로 전송하는 signed.so파일을 공격자가 탈취하더라도, 공격자는 PCL키를 알지 못하므로 암호화되어있는 vendor의 코드를 확인할 수 없다. 공격자에 의해 signed.so파일이 변조되거나 바뀌치기 당했다고 하더라도, 공격자가 PCL키를 알지 못하는 한 변조된 signed.so파일은 RPC enclave를 만드는 과정 중 복호화에 실패하게 되어 enclave 인스턴스가 만들어지지 않는다. 설령 enclave가 만들어지더라도 PCL enclave가 클라이언트의 대칭키를 RPC enclave에 전달하기 전 local attestation을 통해 의도한 RPC enclave인지 확인을 하기 때문에 의도되지 않은 enclave일 경우 통신을 중단할 수 있다. 혹, 공격자가 vendor의 signed.so파일의 내용을 탈취하기 위해 클라이언트가 되어 RPC요청을 하더라도, signed.so파일은 클라이언트 쪽이 아닌, vendor와 엣지 서버에만 존재하게 되므로, 공격자는 RPC요청을 통한 Black box fuzzing형태로밖에 분석할 수 없다.

클라이언트의 정보는 엣지 서버에서 RPC enclave 내부에서 복호화 및 암호화가 되므로 엣지 서버에서 enclave 외부에 노출되지 않는다. 공격자가 엣지 서버를 탈취하여 엣지 서버의 운영체제가 공격자에게 넘어갔다고 하더라도, SGX는 하드웨어레벨의 보안을 제공하기 때문에 공격자는 enclave내부에 있는 클라이언트의 데이터를 볼 수 없다.

4. 실험

3.1에서 제안한 기법으로 3.2에서 제시한 보안성을 얻을 수 있지만, 그로 인한 성능저하가 발생한다. SGX를 활용하여 보안성을 강화할 경우 발생하는 성능저하가 어느정도인지 파악하기 위하여, 제안한 기법과, 제안한 기법에서 SGX를 적용하지 않은 모델을 비교하는 실험을 진행했다.

4.1 실험 환경

<표 1> Vendor, 엣지 서버 프로그램 실행 환경

CPU	Intel i7-9700k
메인보드	ASRock Z390 EXTREME4
RAM	DDR4 PC4-21300 128GB
SSD	Samsung 970 PRO 512GB
OS	Ubuntu 18.04

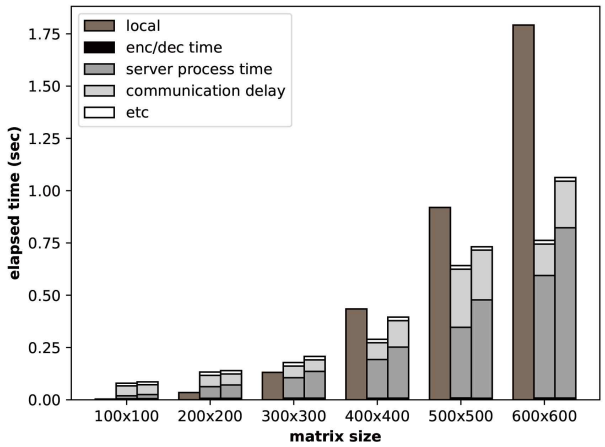
실험을 진행한 서버는 CPU와 메인보드 모두 SGX 지원하는 모델로 자세한 사양은 표 1과 같으며, 클라이언트 프로그램을 실행한 모바일 디바이스는 갤럭시 SM-G965N 64GB이다.

4.2 실험 방법

오프로드를 하지 않고 모바일 디바이스에서 직접 함수를 동작 시킬 때와 엣지 서버에서 SGX를 사용하지 않고 오프로드 할 경우, 그리고 엣지 서버

에서 SGX를 사용한 경우 이렇게 세 가지 경우를 비교하였다. 엣지 서버 프로그램과 vendor 프로그램은 같은 컴퓨터에서 작동시켰으며, 클라이언트는 모바일 디바이스에서 작동시켰다. Vendor 프로그램과 엣지 서버 프로그램을 같은 컴퓨터에서 동작시키므로, 엣지 서버에서 동작 시킬 함수가 들어있는 signed.so파일은 전달이 되었다고 가정하고 진행했다. 실험에서는 두 정사각행렬의 곱을 수행하는 함수를 직접 구현하여 사용하였으며, 실험값은 1000번 측정한 값의 평균을 이용했다.

4.3 실험 결과



(그림 3) 행렬 곱 실험 결과

그림 3은 모바일 디바이스가 행렬의 크기를 서버에 보내면, 서버가 해당 크기의 행렬을 임의로 만들고 행렬곱을 진행하도록 하여 얻은 실험 결과이다. 각각의 세 막대에서 왼쪽 막대는 모바일 디바이스에서 직접 동작시킨 경우, 가운데 막대는 SGX를 사용하지 않고 오프로드를 진행한 경우, 오른쪽 막대는 SGX를 사용한 경우에 측정된 소요시간을 나타낸다. 행렬의 크기가 커짐에 따라 모바일 디바이스에서의 소요시간의 증가량이 서버에서의 소요시간의 증가량보다 커져 오프로드를 한 경우보다 소요시간이 커지는 것을 확인할 수 있다. 우리가 제안한 모델을 사용한 경우, SGX를 사용하지 않고 구현한 모델에 비하여 엣지 서버 측의 소요시간이 1.15배에서 1.43배까지 증가하는 모습을 보여준다.

5. 결론

본 연구에서는 모바일 디바이스가 엣지 서버를 대상으로 컴퓨팅 오프로드를 진행 할 때 정보의 유출 위험 없이 안전하게 진행하기 위한 새로운 client-vendor-edge server 모델을 제안하였다. 제안하는 모델은 엣지 서버에 SGX를 적용시킴으로써 RPC요청시 전달되는 클라이언트의 정보뿐만 아니라 vendor가 오프로드 하는 함수의 내용까지도 제 3자가 엣지 서버에서 변조, 탈취하지 못하게 하는 방법을 제시했다. 이와 같은 모델을 통하여 엣지 서버 측의 소요시간이 최대 약 1.43배 증가하지만, 엣지 서버 측에서의 정보 유출 위험이 없는 안전한 컴퓨팅 오프로딩을 달성할 수 있음을 확인하였다.

Acknowledgement

이 논문은 2020년도 정부 (과학기술정보통신부)의 재원으로 정보통신기획평가원의지원을 받아 수행된 연구임 (No.2014-3-00035, 매니코어 기반 초고성능 스케일러블 OS 기초연구)

참고문헌

[1] L. Lin, X. Liao, H. Jin, and P. Li, “Computation offloading toward edge computing,” Proceedings of the IEEE, vol. 107, no. 8, pp. 1584 - 1607, 2019.

[2] F. Alder, N. Asokan, A. Kurnikov, A. Paverd, and M. Steiner, “S-faas: Trustworthy and accountable function-as-a-service using intel sgx,” in Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop, ser. CCSW’19. New York, NY, USA: Association for Computing Machinery, p. 185 - 199, 2019

[3] B. Trach, O. Oleksenko, F. Gregor, P. Bhatotia, and C. Fetzer, “CLEMMYS: Towards Secure Remote Execution in FaaS, ”SYSTOR’19″ Proceedings of the 12th ACM International Conference on System and Storage, pp. 44 - 54, May 2019.

[4] K. Bhardwaj, M. Shih, P. Agarwal, A. Gavrilovska, T. Kim, and K. Schwan, “Fast, scalable and secure onloading of edge functions using airbox,” in 2016 IEEE/ACM Symposium on Edge Computing(SEC), pp. 14 - 27, 2016.