웹 애플리케이션 데이터의 익명성 모니터링 기법 제안

김효경[1], 이병영[2]
[1]서울대학교 전기정보공학부 석사과정
[2]서울대학교 전기정보공학부 교수

hyokyung0808@snu.ac.kr, byoungyoung@snu.ac.kr

Proposal of a method to monitor the anonymity of web application data

Hyokyung Kim[1], Byoungyoung Lee[2]
[1]Dept. of Electrical and Computer Engineering, Seoul National University
[2]Dept. of Electrical and Computer Engineering, Seoul National University

요 약

사용자의 데이터를 수집하여 분석하는 서비스들이 증가함에 따라 데이터의 익명성 여부를 파악하는 것이 더욱 더 화제로 떠오르고 있다. 본 연구는 애플리케이션 데이터의 익명성 여부를 모니터링할 수 있는 기법을 제안한다. 기법은 두단계로, 정적 분석기와 런타임 미들웨어를 활용하여 사용자 데이터의 익명성 여부를 실시간으로 확인하여 사용자에게 전달한다. 본 연구는 제안한 기법의 프로토타입 구현과 함께 런타임 미들웨어의 오버헤드도 측정하여 제안하는 기법의 가능성을 선보인다.

1. 서론

최근 사용자의 데이터를 수집하여 분석하는 서비스들이 증가함에 따라 데이터의 익명성 여부가 더욱 더 화제로 떠오르고 있다. 이는 캘리포니아의 CCPA, 유럽의 GDPR 등 전세계적으로 개인정보보호법이 강화 혹은 새롭게 시행되는 경향에서도 볼 수 있다[1]. 더 나아가 최근 App Store 에 도입된 '프라이버시 라벨'은 개발자가 사용자의 데이터를 신원에 연결된 형태로 서버에 저장하는지 명시해야 한다. 이에 따라 데이터의 익명성 여부를 판단하는 것은 더욱 더 중요한 문제로 떠오르고 있다고 볼 수 있으며, 이를 위한 다양한 시도가 있어왔다 [2][3].

그러나 웹 애플리케이션 데이터의 익명성 여부를 판별하는 것은 쉬운 문제가 아니다. 웹 애플리케이션은 사용자의 데이터를 고유의 애플리케이션 로직에 따라 처리하여 데이터베이스에 저장한다. 따라서 데이터베이스에 저장된 데이터의 익명성 여부를 판별하려면 데이터만 살펴봐서는 안되며 애플리케이션 로직 또한 살펴봐야 한다. 따라서 본 연구는 애플리케이션 로직을 고려함으로써 애플리케이션 데이터의 익명성 여부를 판별하고 감시하는 기법을 제안한다.

2. 배경과 위협 모델

본 연구는 한번에 한 사용자의 요청을 받아, 애플리케이션 로직에 따라 데이터를 처리하여 데이터베이스에 저장한 후, 처리된 데이터의 일부를 다시 사용자에게 응답하는 웹 애플리케이션 모델을 고려한다. 사용자는 서비스를 이용하면서 개발자의 서버로 다양한 민감도의 정보를 보낸다. 이 중 어떤 데이터는 단독으로 개인에게 연결 가능한 개인식별정보며, 어떤 데이터는 그렇지 않은 개인정보며, 어떤 데이터는 웹 애플리케이션 응답으로 받은 애플리케이션 데이터다. 사용자는 어떤 개인정보는 익명으로, 즉 자신에게 연결되지 않는 형태로 처리가 되었으면 하는 바람이 있으며 개발자는 이러한 바람대로 특정 정보는 익명으로 처리하여 데이터베이스에 저장하겠다고 약속한다.

본 연구는 웹 애플리케이션 사용자와 개발자 간의 이해관계 충돌에서 발생하는 위협 모델을 고려한다. 개발자는 서비스의 사용자들에 대한 데이터를 최대한 많이 수집하고 신원으로 연결하여 이익을 창출하고자 한다. 따라서 개발자는 사용자와의 약속을 어기며, 웹 애플리케이션 로직에 특정 로직을 추가하여 익명으로 처리하기로 약속한 데이터를 신원에 연결한 형태로 데이터베이스에 저장한다. 동시에 사용자는 자신의 데이터가 약속대로 익명으로 처리되길 원하지만, 개

발자를 완전히 믿지 않는다. 그러나 사용자는 웹 애플리케이션 서비스를 이용하기 위해 어쩔 수 없이 본인의 데이터를 서버로 전송한다.

본 연구는 웹 애플리케이션 로직으로 데이터베이스에 데이터의 신원을 유출하는 공격만 고려한다. 애플리케이션 로직 외 애플리케이션을 배포한 서버의 하드웨어/소프트웨어 스택 구성 요소를 활용한 공격은 공간 제약의 이유로 본 연구에서 다루지 않는다. 또한 부채널, 코버트 채널 공격도 고려하지 않는다.

3. 데이터를 식별하는 공격

공격자는 사용자가 전송한 데이터를 신원에 연결시키기 위해, 그 데이터를 식별 가능한 데이터와 연결된 형태로 저장한다. 단독으로 특정 개인을 알아볼 수 없는 데이터라도, 식별가능한 데이터와 연결이 생길 경우 개인을 알아볼 수 있는 데이터가 된다[5]. 한번 신원에 연결된 데이터는 다른 데이터와 연결될 시 그 다른 데이터를 신원에 연결된 데이터로 만든다. 공격자는 애플리케이션 로직을 활용하여 이러한 연결지점들을 생성하여 공격을 한다.

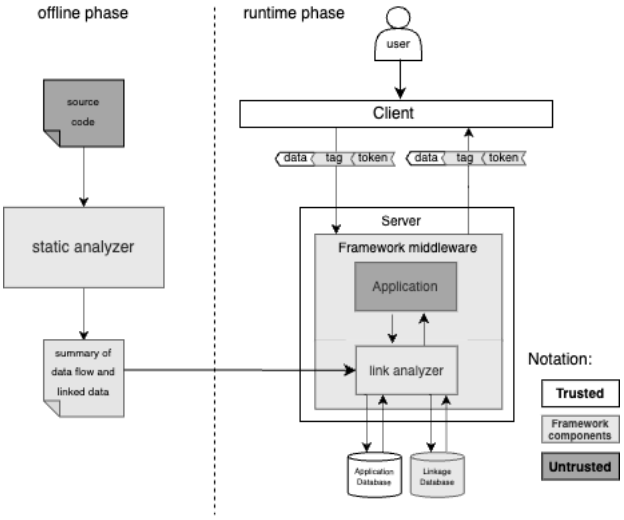
공격자가 사용자에게 상품 후기 정보를 익명으로 처리한다고 약속한 경우를 생각해보자. 공격자는 사용자로부터 상품 후기 데이터를 받는다. 그리고 이 데이터를 신원으로 연결시키기 위해 사용자 요청에 함께 들어있던 다른 데이터 중 신원에 연결된 정보를 상품 후기와 함께 처리한다. 이후 신원에 연결된 데이터와, 그 개인이 보낸 후기 데이터가 함께 처리가 되었다는 사실을 어떠한 형태로든 데이터베이스에 기록을 한다. 이후 공격자는 데이터베이스의 데이터만 봐도 어떤 사용자가 상품후기를 작성했는지 식별해낼 수 있게 된다.

사용자가 정말로 자신의 상품 후기를 익명으로 처리하길 원하면, 신원에 연결된 정보를 함께 안보내면 되지 않느냐는 의문이 들 수도 있다. 그러나 앞서 설명했듯이, 사용자의 요청에는 이전에 애플리케이션 응답에서 전송 받았던 애플리케이션 데이터도 포함된다. 사용자는 이러한 애플리케이션 데이터가 이전 요청에서 처리될 때 자신의 신원에 연결되었는지 알 방법이 없다. 그렇기에 이러한 애플리케이션 데이터가 공격에 사용될 수 있는 신원에 연결된 데이터인지, 아니면 정말로 요청을 수행하기 위한 데이터인지, 판단할 수 없다. 그렇기 때문에 사용자는 서비스를 사용하는 데에 있어서 웹 애플리케이션이 요구하는 대로 데이터를 보낼 수 밖에 없으며 이로 인해 공격자는 공격이 가능하게 된다.

4. 디자인

본 연구가 제안하는 익명성 모니터링 기법의 아키텍처는 그림 1 과 같으며 총 두가지 단계로 나뉜다.

첫번째 단계는 오프라인 단계로, 애플리케이션 코드를 정적 분석하여 사용자의 데이터가 데이터베이스에 저장되기 까지의 데이터 흐름을 분석한다. 공격자의 웹 애플리케이션이 사용자의 데이터를 처리하여 데이터베이스에 저장할 때, 처리과정에서 사용자의 신원에 연결된 데이터와 겹치는 데이터 흐름이 발생하였다면, 데이터 간에 연결이 생겼다고 볼 수 있으며, 결과적으로 신원에 연결되었다고 볼 수 있다. 그렇기에 정적분석기를 돌림으로써 데이터베이스에 저장되는 애플리케이션 데이터 간 연결지점들을 파악한다. 정적 분석기는 이러한 연결지점들에 대한 요약을 파일로 추출하여 저장한다.



(그림 1) 익명성 모니터링 프레임워크.

두번째 단계는 런타임 단계로, 본 연구가 제안하는 미들웨어가 애플리케이션 동작을 실시간으로 모니터링하여 애플리케이션이 사용자의 데이터를 실질적으로 신원에 연결시키는지 실시간으로 모니터링한다. 사용자는 자신의 데이터를 공격자의 서버로 보낼 때, tag 와 token 정보도 함께 보낸다. Tag 정보는 사용자가 전송하는 데이터의 민감도를 나타내는 정보로, 데이터가 개인식별정보인지, 신원에 연결된 데이터인지, 익명으로 처리되어야 하는 데이터인지 나타낸다. Token 은 과거 미들웨어에게서 전송 받았던 정보로, 현재 사용자가 보내는 데이터가 예전에 이미 애플리케이션 데이터베이스에 저장되었던 데이터인지 나타낸다. 미들웨어는 이 tag, token 정보와 오프라인 단계에서 추출된 연결지점 요약, 총 세가지를 참고하여, 애플리케이션이 데이터베이스로 쿼리를 전송할 때마다 실시간으로 읽고 쓰여지는 데이터의 익명성 여부를 모니터링한다. 이러한 익명성 여부는 업데이트된 tag, token 형태로 웹 애플리케이션 응답에 추가되어, 사용자에게 다시 전송된다. 사용자는 업데이트된 tag 를 받아 본래 보냈던 데이터의 익명성 여부를 확인할 수 있다.

런타임에서 익명성 여부를 실시간으로 확인해야 하는 이유는 크게 두가지인데, 첫번째는 실시간으로 사용자에게 데이터의 익명성 여부를 전달하기 위해, 두번째로는 데이터의 익명성이 데이터베이스의 다른 데이터와의 관계로 인하여 실시간으로 바뀌기 때문이다. 데이터 간 연결지점이 생겼다 하여 무조건적으로 신

원에 연결된 건 아니며, 실시간으로 데이터베이스에 저장된 데이터의 양을 살펴보며 이 데이터가 실질적으로 하나의 개인으로 연결될 수 있는지 살펴보는 것이 중요하다. 따라서 k-익명성을 적용하여 데이터베이스의 데이터가 신원에게 연결되었는지 런타임에 검증하는 단계가 필요하다. k-익명성의 경우 동일 레코드 값이 k 개 이상 존재할 경우 그 데이터는 익명 데이터로 간주된다. 이에 따라, 런타임에서 데이터 간 연결지점들을 살펴보며 k 개 이상의 동일한 레코드가 존재하는지 확인해야 한다. 만일 k 개 이상의 레코드가 존재한다면 연결지점은 없는 것으로 간주하여 연결되었던 데이터는 신원에 연결된 정보에서 익명으로 분류가 바뀌는 것이 적절하다.

5. 실험 구현 및 성능 평가

본 연구는 프로토타입으로 파이썬 정적분석기인 pyt[4]를 수정하여 정적분석을 수행하였으며, Django 프레임워크[6]를 수정하여 k-익명성 확인을 제외한 그림 1 의 런타임 모니터링 미들웨어를 구현하였다. 예시 웹 애플리케이션으로는 django e-commerce 애플리케이션[7] 을 수정하여 실험을 하였다. wrk http 벤치마킹 툴[8]로 간단한 데이터베이스 읽기와 쓰기 동작에 대해서 성능을 측정한 결과, 수정된 런타임은 기존의 런타임에 비해 1.45 배 정도의 오버헤드를 가지는 것으로 측정되었다.

6. 결론

본 연구에서는 웹 애플리케이션 데이터의 익명성 모니터링 기법을 제안하였다. 오프라인 단계에서 정적분석을 통해 사용자 데이터의 흐름을 파악하여 가능한 공격 경로를 파악하고, 이를 런타임에 모니터링하여 실시간으로 익명성 여부를 확인하는 두 단계 아키텍처를 제안하였다. 추후 정적분석기의 성능을 개선시켜 더 다양한 웹 애플리케이션 사례에 대해서 정확도와 성능을 측정해볼 예정이다.

Acknowledgement

본 논문은 2024 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. RS-2023-00209093)

참고문헌

[1] Nalneesh Gaur, “This is what increasing data protection laws mean for your company”, World Economic Forum, 02/15/2023, https://www.weforum.org/agenda/2023/02/data-protection-laws-global-patchwork/

[2] Cohen, Aloni, and Kobbi Nissim. "Towards formalizing the GDPR’s notion of singling out." *Proceedings of the National Academy of Sciences* 117.15, 8344-8352, 2020

[3] Article 29 Data Protection Working Party, Opinion 05/2014 on anonymisation techniques. https://iapp.org/media/pdf/resource_center/wp136_concept-of-personal-data_06-2007.pdf

[4] 개인정보보호법, https://www.law.go.kr/lsInfoP.do?lsiSeq=111327#0000

[5] pyt, https://github.com/python-security/pyt

[6] Django, https://www.djangoproject.com/

[7] Django e-commerce, https://github.com/justdjango/django-ecommerce

[8] wrk, https://github.com/wg/wrk