PIOLINK

API 보안을 통한 마이데이터 시대 개인정보 보호

API 취약점 공격 시나리오 및 대응방안 사례

금융 마이데이터 서비스가 본격 시행(스크래핑 방식이 금지되고 API 방식을 통한 금융 마이데이터 서비스 제공) 된 지 내년 1월이면 2년이(2022년 1월 5일 시행) 된다. 금융 마이데이터 서비스는 서비스를 시작한 지 2년밖에 되지 않았지만 정부 및 서비스 제공 기업들의 노력으로 빠르게 자리를 잡았고, 올해(2023년 1월 10일)에는 금융 마이데이터 서비스의 과금체계 세부 기준 마련을 위한 계획과 납부 계획까지 발표되었다. 또한 금융 분야에서의 성공적인 서비스 시행에 힘입어 전 산업 군에서도 마이데이터 서비스를 시행할 수 있도록, '23년 9월 15일 시행된 개정된 개인정보보호법(제35조의 2 개인정보의 전송 요구 신설_해당 항목에 대한 시행일 미지정)에 그 초석이 마련되었다. 즉, 기존에는 신용정보법(제33조의 2 개인신용정보의 전송요구)에 의거 개인신용 정보에 대한 전송을 요구하였다면, 이제는 개인정보보호법에 의거 개인정보에 대한 전송을 요구할 수 있게 된 것이다.

이는 단순히 법적 근거가 명확해졌거나 요구할 수 있는 정보가 확장되었다는 의미가 아닌, 정보주체의 동의에 따라 합법적으로 활용할 수 있는 데이터가 늘어났으며, 이런 데이터를 활용한 데이터 시장이 성장할 수 있는 동력을 얻었다는 의미가 된다. 실제로 정부는 '마이데이터로 선도하는 디지털 대전환 시대'라는 비전 하에 '27년까지 데이터 시장규모를 20% 이상 추가 성장시키는 것을 목표로 한다고 발표하였다. '22년 기준 25조 원인 국내 데이터 산업 시장이 최근 3년과 같이 연평균 11.9%를 성장한다는 전제하에 20%가 추가 성장한다면 '27년 기준 50조 원이 넘는 시장이 만들어진다는 것을 의미한다. 그리고 실제 이런 목표들을 이루기 위해 민관이 힘을 합쳐 적극적인 행보('23년 11월 14일 전 분야 마이데이터 도입·확산을 위한 민·관 협의체 출범)를 보이고 있다.

하지만 이런 거대한 시장에는 언제나 기대와 다르게 불청객이 따르기 마련이다. 특히 조금만 시점을 달리하면 50조 원이라는 거대한 시장은 악의적 목적을 가진 해커들 입장에서도 50조 원의 가치가 있는 시장이라는 소리가 될 수 있다. 특히나 개인정보는 크리덴셜 스터핑(Credential Stuffing) 및 각종 피싱에 활용이 가능하여 다크웹에서도 거래가 활발히 이뤄질 만큼 전통적으로 해커들이 좋아하는 데이터이다. 이런 상황에서 마이데이터 서비스의 확장으로 개인정보를 활용한 서비스가 증가하게 되어 개인정보의 가치가 올라간다면 해커들은 더욱 혈안이 되어 개인정보를 노릴 가능성이 크다.

물론 마이데이터 서비스는 기존의 개인정보 처리 방식보다 기술적 보안 측면에서 강화되었다. 대표적으로 마이데이터 서비스는 API 방식을 통해(금융 마이데이터 서비스는 API를 통한 서비스가 의무이며 그 밖에 공공 및 의료서비스에서는 의무는 아니지만 API를 통한 서비스 제공을 권장하고 있음) 서비스를 제공하도록 하고 있다. API 방식 이전에는 주로 웹 스크래핑을 사용하였는데 웹 스크래핑은 페이징 되는 모든 데이터에 대해 수집이 가능하기 때문에 정보 유출의 위험성과 과도한 개인정보 수집이라는 취약점을 가지고 있다.

반면 API는 요청한 정보에 대해서만 응답을 통해 정보를 제공하기 때문에 웹 스트래핑에서 문제가 되는 취약점에 대한 해결이 가능하며, 마이데이터 서비스에서 사용하는 mTLS(mutual TLS)라는 상호인증 통신과 함께 사용된다면 신뢰성을 확인한 대상과만 통신을 하기 때문에 웹 스크래핑 방식 대비 높은 보안성을 제공한다.

하지만 API 역시 웹 스크래핑에 비해 비교적 안전할 뿐이지 취약점이 존재하지 않는 것은 아니다. 대표적으로 글로벌 공개 취약점인 CVE(Common Vulnerabilities and Exposures) 리스트에서 API를 검색하면 3,000개가 넘는 취약점이 검색된다. 또한 API와 관련된 취약점은 웹 애플리케이션에 대

한 취약점을 분석하여 보안을 증진시키는 글로벌 비영리 단체 OWASP(Open Web Application Security Project)에서도 2019년부터 API 10대 취약점 리스트(OWASP API Security Top10)를 발표하고 있다.

이처럼 암호화 통신 지원 등 비교적 안전한 API라 할지라도 많은 취약점에 노출되고 있기 때문에 API를 사용하여 마이데이터 서비스를 지원하는 기관 및 기업에서는 보안에 대한 많은 관심과 노력이 필요하다.

<API 취약점을 이용한 개인정보 유출>

이제부터 본론으로 들어가 API를 사용하는 마이데이터 서비스의 시행으로 개인정보보호가 어떤 새로운 과제에 직면했는지 알아보자.

우선 개념적으로 개인정보보호의 보호 영역이 확장되고 보호 방법이 변했다는 것을 이해해야 한다.

(1) 보호 영역

마이데이터 서비스 이전의 개인정보는 데이터로 저장되어 있기 때문에 보호 영역 역시 데이터를 저장하는 데이터베이스 및 서버 스토리지 위주였다. 즉 네트워크 단에서는 데이터를 저장하고 있는 데이터베이스 및 서버 스토리지에 대한 접근을 잘 통제하고, 데이터의 위조, 변조, 유출에 대한 조치는 데이터가 저장되어 있는 시스템 내부 단에서 조치를 취하면 됐다.

하지만 API는 일반적으로 내부 네트워크뿐만 아니라 외부 네트워크에서도 HTTP 프로토콜을 통해 개인정보에 접근할 수 있다. 따라서 모든 네트워크 영역에 대한 API 인터페이스까지 보호해야 할 영역이 확장된 것이다.

(2) 보호 방법

보호하는 영역이 확장된 만큼 보호하는 방법 역시 기존과 다른 접근이 필요하다. 기존 데이터베이스 및 서버 스토리지의 데이터를 지키기 위한 보안에서는 우선 UTM을 통해 네트워크에 대한 접근을 통제를 하였다. 그리고 멀웨어, 백도어를 통한 데이터의 위조, 변조를 막기 위해 백신을 설치하였고, DRM/DLP를 통해 데이터를 암호화하고 유출을 방지하여 개인정보를 보호하였다.

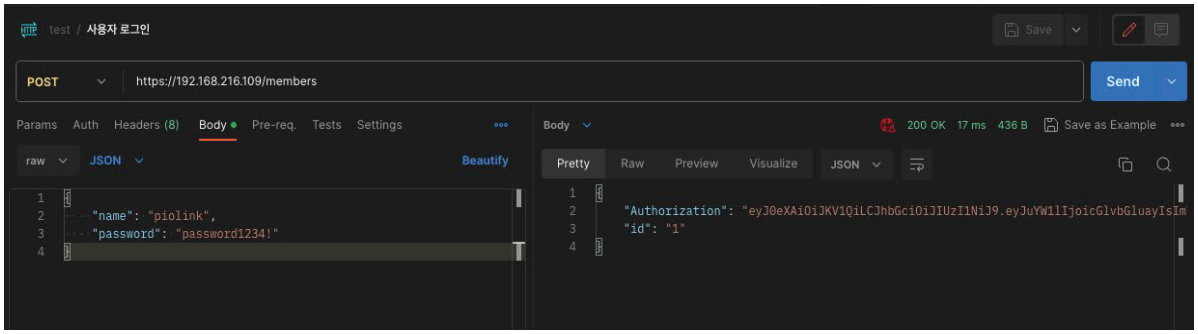
하지만 API를 이용한 서비스에서는 응답 페이로드에 타인의 식별정보까지 유추가 가능한 식별정보가 포함되지는 않았는지, 유효한 요청인지, 과도한 정보가 포함되어 민감한 정보가 새어나가지는 않는 것인지 등의 API 통신 페이로드에 대한 API 인터페이스 보안이 추가로 고려되어야 한다.

그렇다면 실제 OWASP API Security Top10에서 말하는 취약점을 바탕으로 개인정보가 어떻게 위조, 변조, 유출될 수 있는지 몇 가지 시나리오 통해 확인해 보자.

(1) 첫 번째 시나리오는 "URL에 식별정보가 포함되어 있어 이를 통해 타인의 식별정보를 유추하여 타인의 개인정보에 접근하는 시나리오"로 API에서 표준처럼 사용하고 있는 RESTful API가 기본적으로 URL 형식을 통해 리소스를 식별하기 때문에 발생할 수 있는 상황이다.

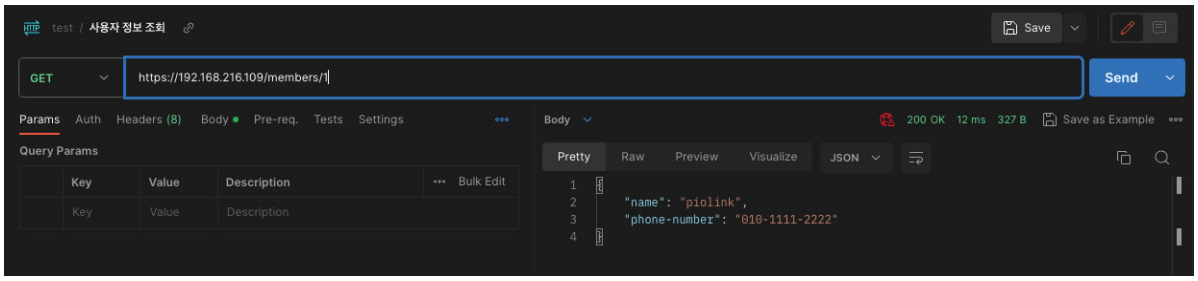
① 정상적인 로그인 시도

② 로그인 요청에 대한 응답 페이로드에서 사용자 식별정보가 1이라는 것을 확인



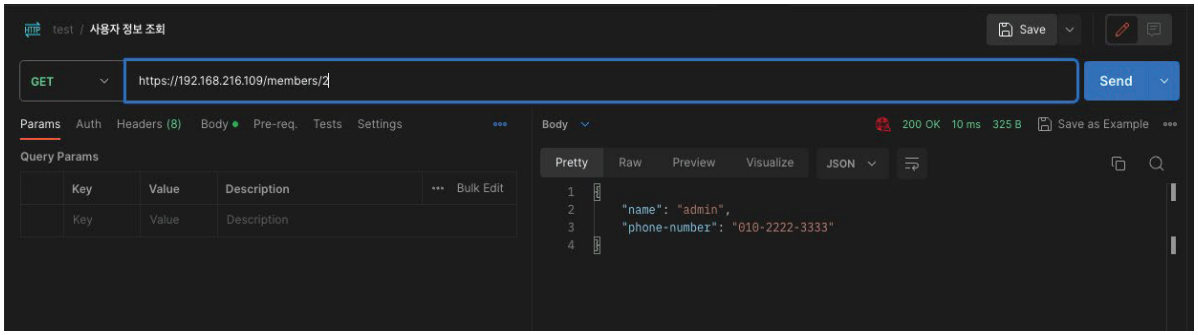
<그림 1> 사용자 로그인을 통해 식별정보를 확인

③ 본인에 대한 사용자 식별정보가 1이라는 점을 통해 다른 사용자의 식별정보도 정수로 되어 있음을 추측



<그림 2> 확인된 식별정보를 사용하여 개인정보에 접근

④ 추측한 사용자 식별정보를 이용하여 타인의 정보에 접근



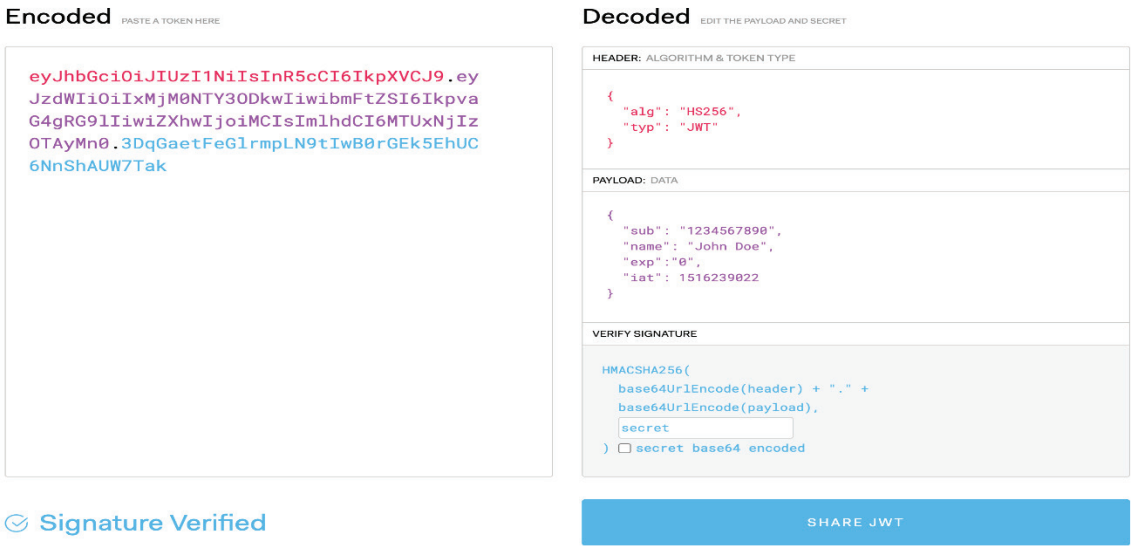
<그림 3> 유추한 식별정보를 사용하여 타인의 개인정보에 접근

이 같은 시나리오에서는 서버에 따라 추측한 식별정보를 이용하여 타인의 개인정보 획득은 물론 정보에 대한 위조·변조도 가능하다.

(2) 두 번째 시나리오는 "API에서 주로 사용하는 JSON 포맷(금융 마이데이터 서비스에서 표준 API로 사용)의 JWT(JSON Web Token) 무결성이 손상되어 개인정보에 접근하는 시나리오"이다. 두 번째 시나리오는 API서버가 JWT에 대한 무결성 검사를 고려하지 않았을 때 발생하는 경우로 만료된 JWT를 사용하였는데도 정상 요청으로 받아들이는 시나리오와 JWT가 하이재킹 되어 인증 정보가 유출되었다는 가정하에 해당 인증 정보를 통해 다른 IP에서 접속을 시도하였는데도 정상 요청으로 받아들여지는 시나리오, 총 2가지 경우의 상황을 보여준다.

[JWT 만료 여부 확인 부재 시나리오]

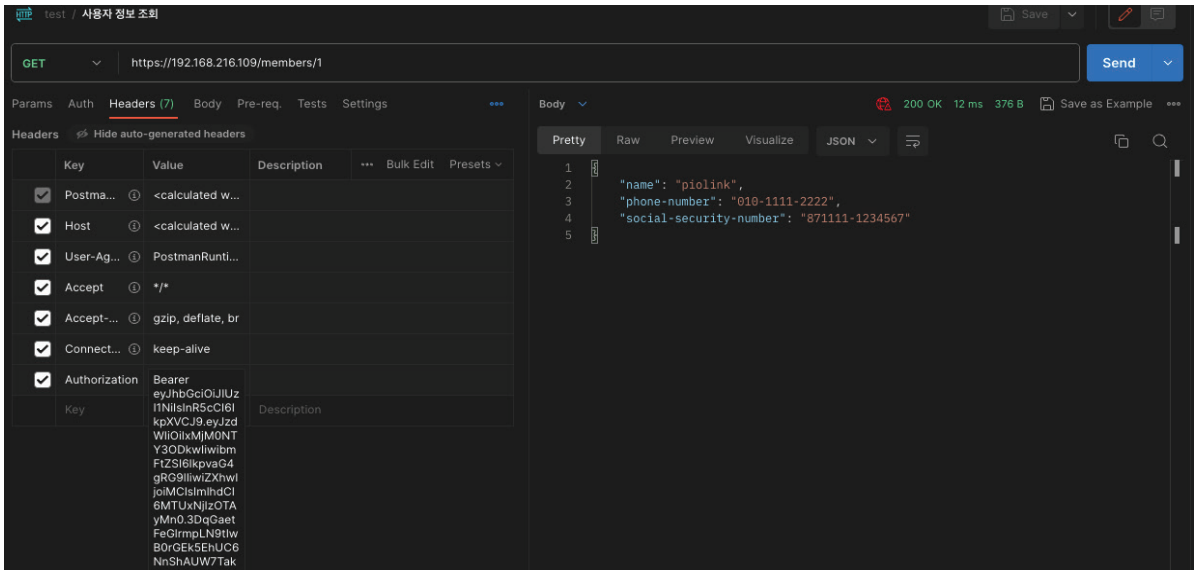
① 이미 만료된 JWT("exp" : "0"이 해당 토큰이 만료되었음을 의미) 인증 정보를 획득



<그림 4> 만료된 JWT에 대한 인증정보를 획득

② 헤더에 획득한 인증 정보를 삽입하여 정보를 요청

③ API 서버가 JWT의 만료 여부를 확인하지 않아서 정상적인 요청으로 처리



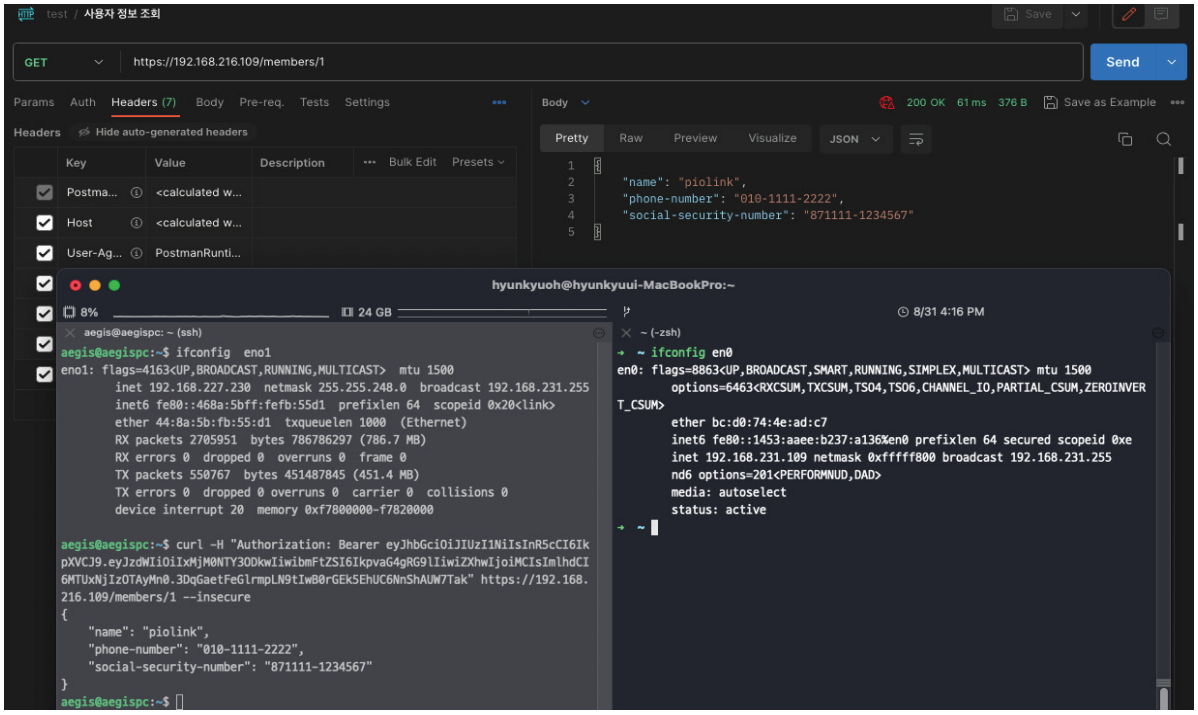
<그림 5> 만료된 JWT 인증정보 활용하여 요청 성공

[JWT 하이재킹 여부 확인 부재]

① 중간에 패킷을 가로채 JWT 인증 정보를 획득

② 해당 인증 정보를 활용하여 다른 IP를 가진 자가 접근을 시도

③ 하이재킹에 대한 여부(IP 검사 등)를 확인하지 않아 정상적인 요청으로 처리

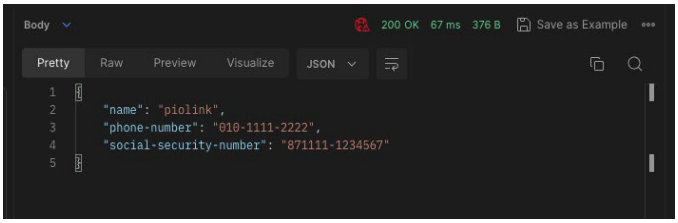


<그림 6> 훔친 JWT인증 정보를 사용하여 다른 IP에서 접근을 시도

해당 시나리오 역시 첫 번째 시나리오와 마찬가지로 권한이 없는 자가 타인의 개인정보에 대한 획득이나 위변조가 가능해지는 상황이다.

(3) 세 번째 시나리오는 "요청 API에 과도한 정보가 포함되어 있어 필요로 하는 것보다 더 많은 데이터가 노출되어 개인정보가 유출되는 시나리오"로, 서비스의 원래 목적보다 더 많은 개인정보를 포함하고 있는 서버의 경우 개인정보를 과도하게 제공하여 개인정보가 유출되는 상황이다.

① 이름과 연락처를 제공하는 API 서비스에 개인정보를 요청
② 하지만 해당 서비스에서 가지고 있는 개인정보는 이름과 연락처뿐 아니라 그 이상의 개인정보를 보유 중
③ API 서버가 이를 인지하지 못하고 해당 요청에 대한 응답으로 포함하고 있는 개인정보를 모두 제공

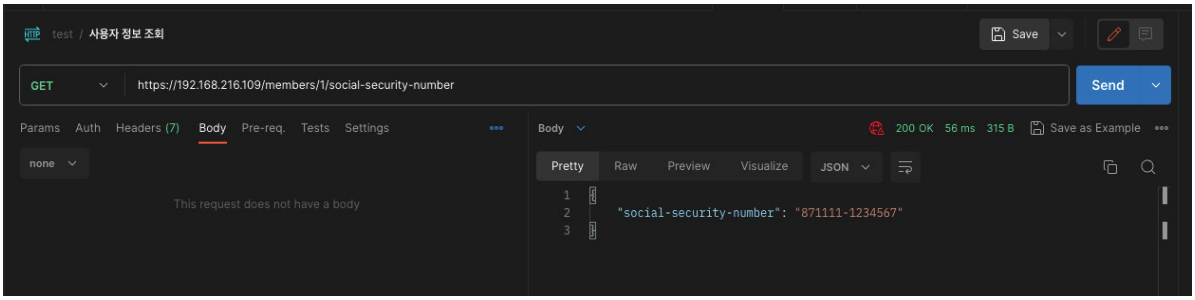


<그림 7> 과도한 정보를 제공하고는 서버

해당 시나리오는 개인정보의 주체가 전송을 요청하고 동의한 것보다 과도한 개인정보가 전송되어 개인정보가 유출될 수 있는 상황이다. 또한 반대로 송신 시스템에서는 받으면 안 되는 개인정보를 수신하게 되어 자칫 과도한 개인정보를 보유하여 관리에 문제가 생길 수 있는 상황도 될 수 있다.

(4) 마지막 네 번째 시나리오는 "잘못된 보안 구성으로 불필요한 기능 또는 HTTP 메서드가 활성화되어 적절하지 못한 요청에도 응답이 발생하는 시나리오"로 POST 및 PUT 메서드만을 정상적인 요청으로 처리하여야 하는 서비스에서 GET 메서드가 허용되어 개인정보가 유출되는 상황이다.

① 개인정보를 입력하거나 수정하는 시스템에 개인정보를 요청
② GET 메서드에 대한 요청이 허용되어 있어 정상적인 요청으로 처리하여 개인정보를 제공



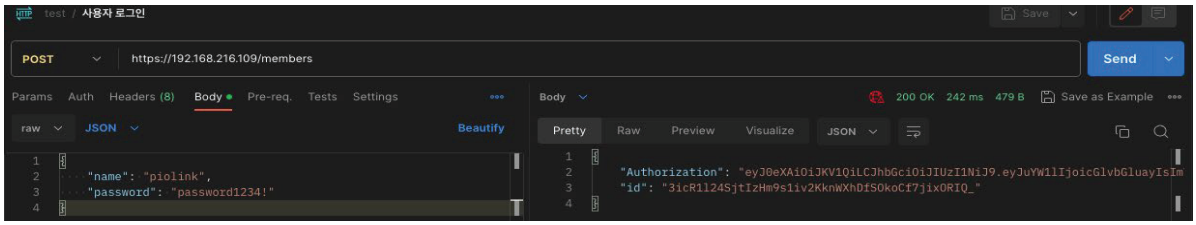
<그림 8> GET 메서드를 정상 요청으로 처리하여 응답하는 서버

해당 시나리오는 목적에 맞지 않는 메서드를 허용하여 개인정보 주체의 의도와 상관없이 개인정보가 전송되거나 위·변조될 수 있는 상황이다.

<WAAP를 통한 기술적 조치>

앞서 시나리오들에서 살펴본 것과 같이 API 취약점은 대부분 시스템에 대한 구성 오류이다. 단순히 개발 과정에서 놓치거나 운영자의 미숙에 의해 발생할 수도 있고, 시스템의 구조상 또는 운용상의 메커니즘 때문에 어쩔 수 없이 이와 같은 취약점들을 방치해야 하는 경우도 있을 것이다. 하지만 우리는 이런 취약점들을 마냥 방치할 수만은 없기 때문에 다른 기술적 조치를 취해야 한다. 그 방법 중 하나가 바로 WAAP를 통한 API 보안이다. WAAP는 Web Application and API Protection의 약자로 기존 WAF(Web Application Firewall)에 API 보안 기능이 추가된 보안 솔루션이다(WAAP에 대한 더 자세한 내용은 **'[보안백서]API 보안과 WAAP'** 참고). 그렇다면 WAAP는 어떻게 앞서 소개한 취약점 시나리오들을 조치할 수 있는지 살펴보자.

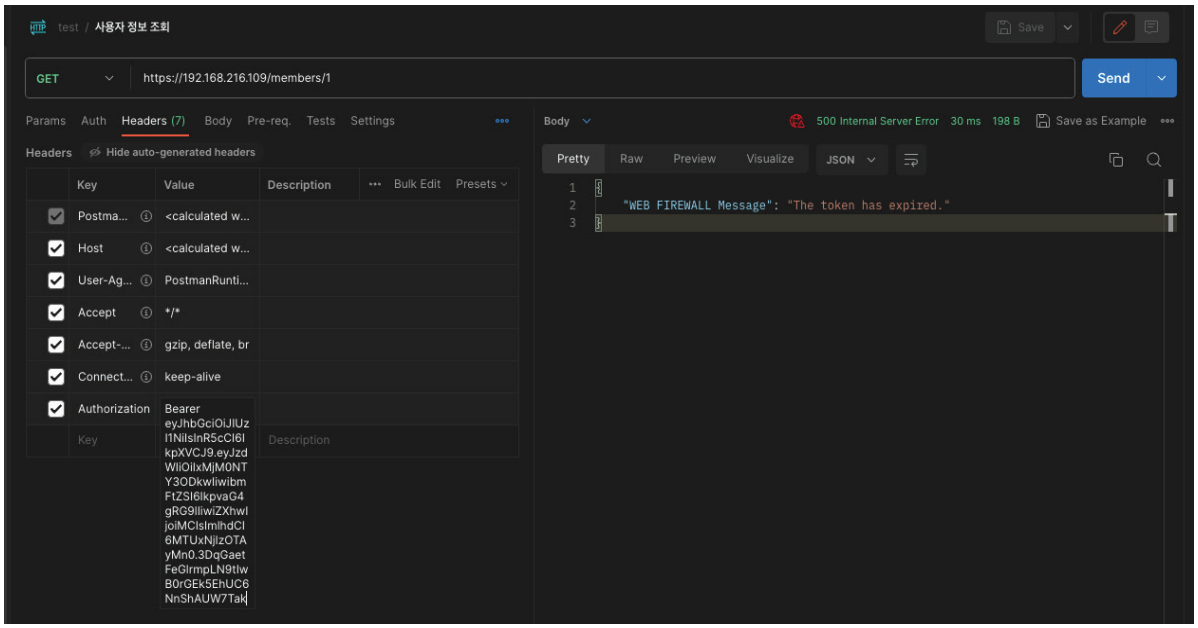
(1) 첫 번째 시나리오인 "URL에 식별정보가 포함되어 있어 이를 통해 타인의 식별정보를 유추하여 타인의 개인정보에 접근하는 시나리오"는 URL에 포함되어 있는 식별정보를 암호화를 통해 난수화 하거나 숨김으로써 조치가 가능하다. 즉, 일반적인 네트워크 구조상에서 프록시 서버 역할을 해줄 수 있는 WAAP가 응답 필드 중 원하는 필드를 난수화 하거나 숨길 수 있도록 기능을 제공하면 조치가 가능하다.



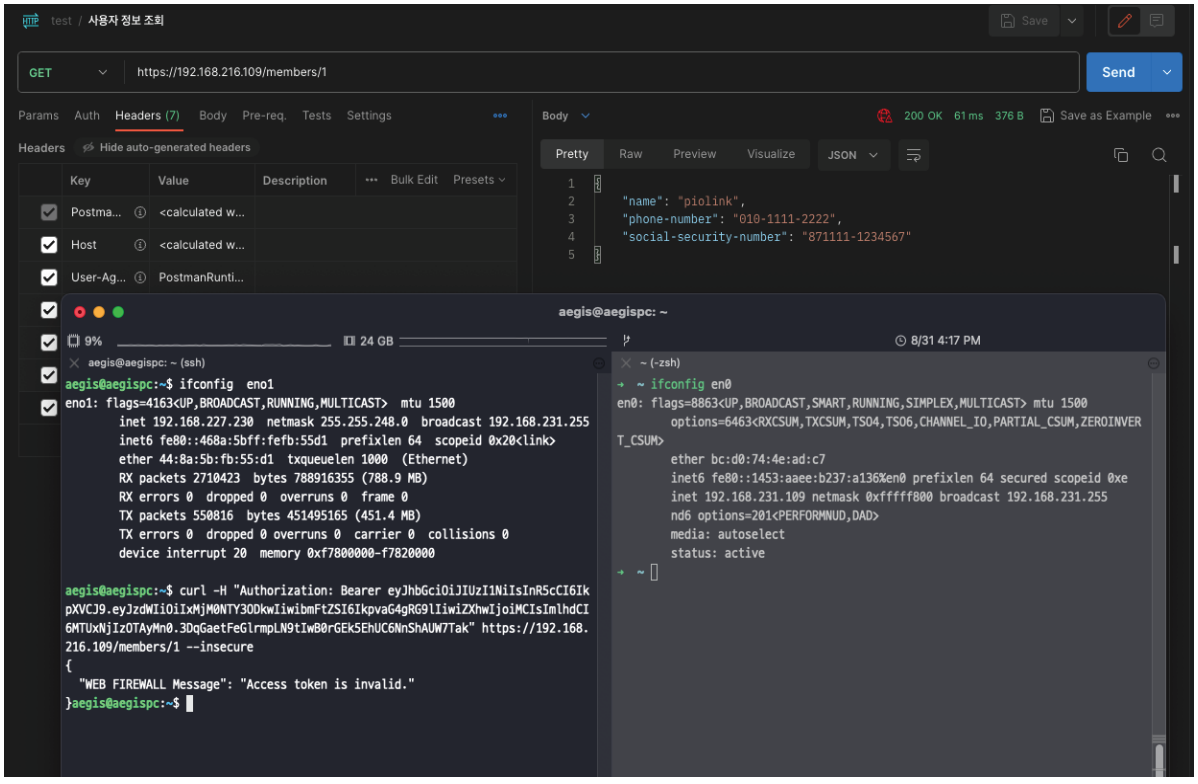
<그림 9> 응답 페이로드의 식별정보가 난수화 되어 유추가 불가능

(2) 두 번째 시나리오인 "JWT(JSON Web Token) 무결성이 손상되어 개인정보에 접근하는 시나리

오" 역시 WAAP가 프록시 서버로서 JWT에 대한 무결성 검사(Timestamp를 통한 토큰의 만료 상태 확인, 토큰의 IP를 매치시켜 정당성을 확인하는 기능 등)를 수행해 줌으로써 조치가 가능하다.



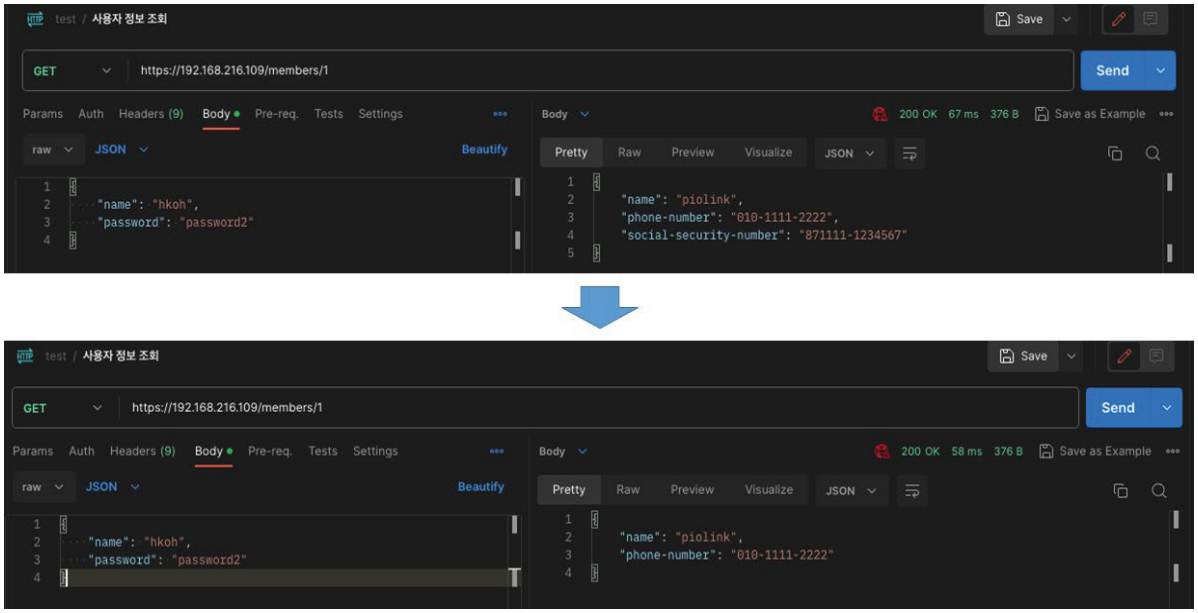
<그림 10> JWT에 대한 토큰 만료검사로 인해 접근 실패



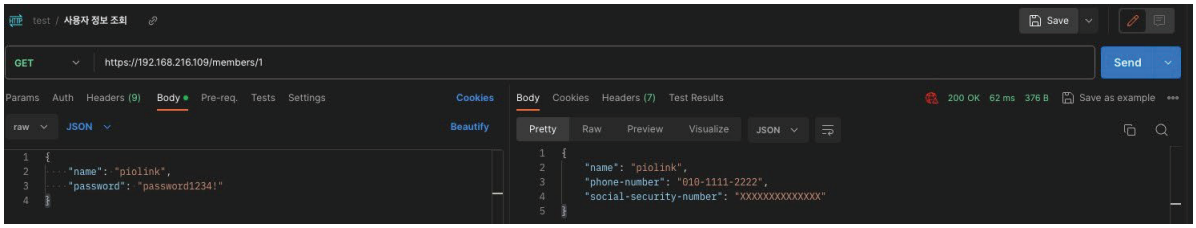
<그림 11> JWT IP유효성 검사로 인해 접근 실패

(3) 세 번째 시나리오인 "요청 API에 과도한 정보가 포함되어 있어 필요로 하는 것보다 더 많은 데이터가 노출되어 개인정보가 유출되는 시나리오"는 해당 서버가 제공하면 안 되는 응답 필드에 대해 클로킹(마스킹 또는 삭제)하여 조치할 수 있다. 이 역시 WAAP가 서버 앞에서 프록시 역

할을 하여 응답 필드 중 서비스에 맞게 원하는 필드를 클로킹 해 줌으로써 조치가 가능하다.

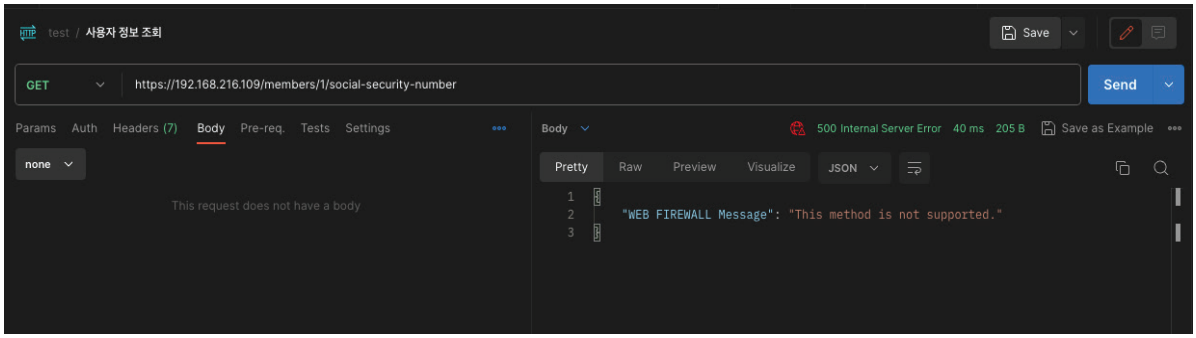


<그림 12> 응답 필드 중 주민등록번호에 대한 필드를 삭제



<그림 13> 응답 필드 중 주민등록번호에 대한 필드를 마스킹

(4) 네 번째 시나리오인 "잘못된 보안 구성으로 불필요한 기능 또는 HTTP 메서드가 활성화되어 적절하지 못한 요청에도 응답이 발생하는 시나리오" 역시 WAAP가 프록시 위치에서 서비스의 목적에 맞는 메서드만 허용하도록 하여 조치할 수 있다.



<그림 14> 프록시 서버에서 GET 메서드가 차단된 상태

상기 조치들은 모두 취약점 시나리오 환경에 파이오링크의 WAAP인 WEBFRONT-K를 이용하여 실증한 화면들이다. 이처럼 WAAP는 API 취약점에 대한 기술적 조치를 제공해 줄 수 있는 대표 장비로 글로벌 시장에서도 이미 API 보안 솔루션으로서 WAAP가 시장을 키워나가고 있다.

<마이데이터 서비스와 API 보안>

지금까지 OWASP API Security Top10 취약점들을 바탕으로 발생 가능한 개인정보 관련 보안사고 시나리오와 그에 대한 기술적 조치사항을 몇 가지를 살펴보았다. 이번 백서에서 다룬 취약점 시나리오는 정말 극히 일부로 아직 알려지지 않은 API 취약점은 물론 상상도 못 할 방법의 취약점 시나리오들이 존재하고 있을 것이다. 그리고 그것들을 악용할 해커 역시 상상 이상으로 많이 도사리고 있을 것이다. 그렇기 때문에 파이오링크는 더욱더 열심히 API 보안에 대해서 연구 및 개발을 진행하고 있으며, 안전하고 편리한 마이데이터 서비스를 제공하기 위해 고객들과 함께 실증 레퍼런스를 만들어가고 있다.

안전한 마이데이터 서비스 제공을 고민하는 많은 기관과 기업들이 WAAP를 통해 그 고민들을 해결하길 바라고, 혼자서 해결하기 어려운 고민에 대해서는 파이오링크에 문의하여 API 보안 전문가들과 함께 해결하길 바란다.

(이메일: waf@piolink.com)