CLOUD SECURITY · 2026

초고속 클라우드 침해

AI 보조 공격과 Bedrock Agent 탈취 시나리오

01

AI가 공격 도구가 되는 순간

02

AI가 공격 대상이 되는 순간

03

클라우드 방어 전략의전환점

박민서 · Beaverdam Community Builder

Introduction

SPEAKER

박민서

Beaverdam Community Builder

- → Whitehat School 3기 PM
- → Baby Beavers 1기 PL

비버댐 커뮤니티

BEAVERDAM COMMUNITY

BEAVER DAM

01 **Offensive Cloud Security 연구**
공격자 관점의 클라우드 보안 학습

02 **세미나 · 지식 공유**
정기 세미나와 실전 노하우 축적

03 **Baby Beavers 프로그램**
신규 멤버 온보딩 및 멘토십

실전형 클라우드 보안 인사이트 축적

Agenda

두 가지 실제 사례를 따라가며, 새로운 방어 전략을 찾아갑니다

CHAPTER

01

문제 제기

AI는 공격 도구이자
공격 대상이다

CHAPTER

02

Case 1

AI 보조
클라우드 침해 사례

CHAPTER

03

Case 2

Bedrock Agent
권한 구조 취약점

CHAPTER

04

결론 & 전략

AI 시대 클라우드
방어 전략

AI는 공격 도구이자 공격 대상

두 개의 얼굴을 가진 AI — 클라우드 방어 기준은 어떻게 바뀌어야 하는가

AS A WEAPON

공격 도구로서의 AI

AI Assists the Attacker

⚡ 정찰 ⚡ 판단 ⚡ 코드 생성 ⚡ 권한 상승 자동화

사람의 대응 속도를 넘어선 공격 흐름

AS A TARGET

공격 대상으로서의 AI

AI Becomes the Victim

🎯 Agent 권한 탈취 🎯 Memory 유출 🎯 내부 확산

새로운 공격 표면

AI Agent 인프라가 곧 침투 경로가 된다

Q. AI 시대의 클라우드 방어 기준은 어떻게 바뀌어야 하는가?

챕터 2

Case 1

AI-Assisted Cloud Breach

Sysdig 8분 관리자 권한 탈취 사건

SOURCE
Sysdig TRT
실제 고객 환경 탐지

TIME-TO-ADMIN
단 8분
관리자 권한 탈취

ENTRY POINT
공개 S3 Bucket
IAM Key 유출

VERDICT
사람보다 빠른
AI 보조 자동화 공격

Sysdig 8분 관리자 권한 탈취 사건

Sysdig Threat Research Team이 실제 고객 환경에서 탐지

01 Sysdig TRT, 실제 고객 환경에서 탐지

02 공개 S3 Bucket에서 IAM Access Key 유출

03 탈취 Key로 정찰 및 권한 상승 자동 진행

04 8분 만에 AdministratorAccess 획득

TIME TO ADMIN

8분

자격증명 탈취 → 관리자 권한 획득
AI 보조 공격의 위력

KEY MESSAGE 공격자는 AI를 활용해 최적의 권한 상승 경로를 자동으로 탐색한다

공격 타임라인 전체

관리자 권한 탈취 8분 · 피해 확산 이후 1시간 43분

T+0min
초기 접근
공개 S3 Bucket
IAM Key 탈취

T+0~6min
정찰
서비스 열거
Logging 상태 확인

T+6~8min ⚡
권한 상승
Lambda Injection
관리자 Key 생성

T+11min~
측면 이동
Backdoor User
생성 · 확산

T+58min~
자원 탈취
LLMjacking
GPU 탈취

T+1h51m
차단
수동 대응으로
차단 완료

PHASE 1 · 권한 탈취 8분

PHASE 2 · 확산 1시간 43분

KEY 8분 권한 탈취 · 이후 1시간 43분간 피해 확산

공격의 시작점: 공개된 RAG 데이터 버킷

왜 RAG 버킷이 공개되어 있었고, 왜 그 안에 Key가 있었나

RAG 데이터 저장소

왜 공개되어 있었나

AI 모델 답변 생성용
내부 문서 저장소

RAG 외부 접근 필요성 또는
단순 설정 실수

결과: 인터넷 어디서든 접근 가능

하드코딩된 IAM Key

왜 Key가 안에 있었나

RAG 적재 자동화 스크립트에
편의상 Key 하드코딩

스크립트 파일이 문서들과 함께
버킷에 함께 업로드

결과: 오브젝트 검증 없이 적재

공격자는 어떻게 이 버킷을 찾아냈나

무작위 스캐닝이 아닌, AI 네이밍 관습 기반 정밀 타깃팅

🎯

전략 01

AI 네이밍 관습 사전 딕셔너리 스캔

KEYWORDS `rag` `bedrock` `ai` `llm` …

📦

Ex)

company_ai_rag_data

초기 권한

ReadOnlyAccess + 제한적 Lambda 읽기/쓰기 권한

EXPOSURE CHAIN

🪣 **공개 S3 Bucket · RAG 데이터 저장소**
rag-* · bedrock-* · ai-* 네이밍 패턴

↓

📄 **하드코딩된 IAM Access Key**
장기 사용 가능 · Bucket 내부 파일에 평문 노출

↓

RESULT **유효한 Access Key 확보 → 즉시 침투 가능**

정찰 시작: 내가 누구인지부터 확인

6분 동안 계정 내 서비스를 전방위로 열거

API CALL SEQUENCE

STEP 01

01

GetCallerIdentity

현재 보유 권한 확인
탈취 Key의 신원 파악

STEP 02

02

List · Describe APIs

Secrets Manager · Lambda
EC2 등 주요 서비스 열거

STEP 03

03

IAM Role · User 열거

권한 상승 가능 ID 탐색
관리자 후보 식별

KEY 6분 안에 계정 환경 전체를 자동 매핑

Bedrock Logging 확인 → LLMjacking 사전 준비

처음부터 LLMjacking이 목적이었다는 증거

API 01

ListFoundationModels

호출 가능한 Bedrock AI 모델 목록 조회

API 02 ⚡

GetModelInvocationLoggingConfiguration

Bedrock 로깅 활성화 여부 사전 체크

LLMjacking

탈취 자격증명으로 피해자의 AI 모델 무단 호출 — API 호출만으로 이루어져 인프라 이상 징후가 없고, 비용은 피해자가 부담

RESULT

로깅 비활성화 확인 → 공격자는 LLMjacking을 공격 계획에 포함

IAM Role Assume 시도

Role 탈취만으로는 원하는 권한 상승이 되지 않는다

FAILED ❌

완전한 관리자 권한, Role Assume 실패

- admin
- administrator

→ 실패

SUCCESS ⚠️

Assume은 성공했지만, 실질 권한 없음

- sysadmin
- netadmin / account

→ 다른 경로 탐색 결정

KEY AI는 막다른 길에 도달하면, 즉시 다른 경로로 우회한다

Lambda Injection을 통한 권한 상승

EC2 초기화용 자동화 함수에 숨겨진 IAM 쓰기 권한

ATTACK CHAIN

01 기존 Lambda ec2-init 발견
신규 함수 생성 없이 기존 자산 재활용

02 Execution Role에 IAM 쓰기 권한 존재
CreateAccessKey · ListUsers 권한 보유

03 UpdateFunctionCode 호출
악성 페이로드 주입 — 정상 배포처럼 위장

04 관리자 User frick의 Access Key 생성
AdministratorAccess 권한 즉시 확보

새 함수 생성 vs 기존 함수 덮어쓰기

AI는 어떻게 frick의 Key를 꺼낼지 판단했나

선택 ❌

CreateFunction

새 함수 만들기

CreateFunction 이벤트가 CloudTrail에 기록됨

→ 보안 탐지 위험

→ 기각

선택 ✓

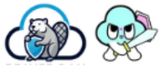
UpdateFunctionCode

기존 함수 코드 덮어쓰기

신규 리소스 생성 이벤트 **발생하지 않음**

→ 정상 배포처럼 보임

→ 채택



주입 코드 — 응답값에 Key를 실어 반환

외부 통신 없이 Lambda 실행 결과창이 키 발급 채널

01 **boto3.client('iam')**
람다 내부에서 IAM 클라이언트 생성

02 **iam.list_users()**
전체 IAM 유저 열거 후 frick 검색

03 **iam.create_access_key(UserName='frick')**
관리자 User의 Access Key 신규 생성

04 **람다 실행 응답값에 Key를 그대로 실어 반환**
외부 통신 없음 — 네트워크 탐지 회피

+α 타임아웃을 3초 → 30초로 늘림 · 포괄적 try/except 예외 처리 = AI 코드의 전형

frick Key 탈취 성공 — 8분 완료

실패와 재시도 끝에 도달한 관리자 권한 획득

RETRY HISTORY

ATTEMPT 01

타깃: adminGH → 실패

이름은 관리자처럼 보였지만 실제 권한 부족

RETRY ×3

세 번의 코드 수정과 재시도

AI가 유저 이름 기반 추론 → 정밀 타깃팅으로 전환

SUCCESS

타깃: frick → AdministratorAccess Key 획득

Lambda 실행 응답값에서 Key 수령 완료

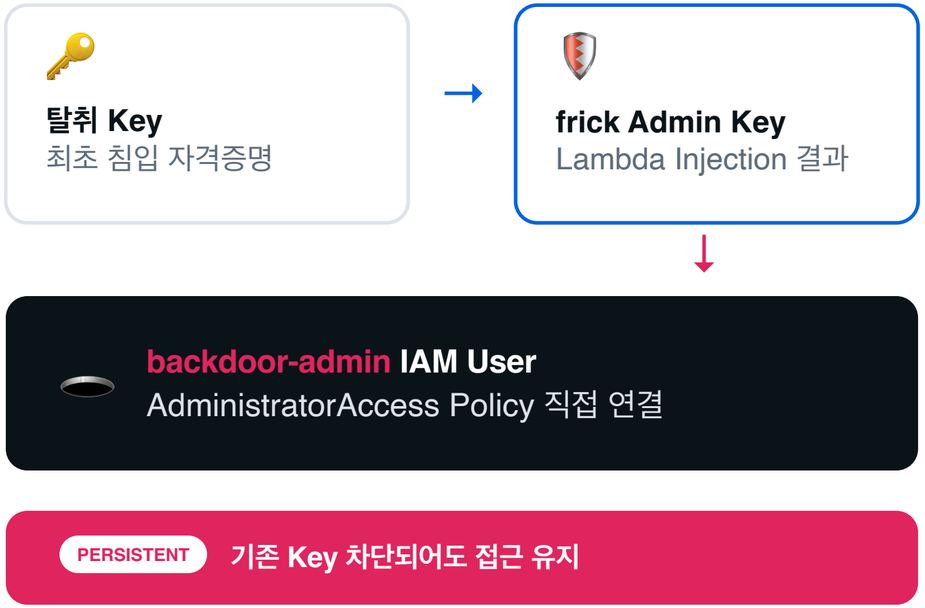
전체 사이클

정찰 + 코드 생성 + 재시도 + 성공

측면 이동: 영속성 확보를 위한 백도어

왜 이미 관리자인데 또 다른 유저를 만들었나

IDENTITY EXPANSION



TACTICS

추가 Identity 생성
AdministratorAccess 획득 직후 즉시 실행

Policy 직접 연결
AdministratorAccess Policy를 새 User에 부착

Identity 분산
여러 Identity로 행동 분산 — 탐지 우회

KEY MESSAGE

관리자 권한 탈취 이후 공격은 확산 단계로 전환된다

LLMjacking 가동

Cross-Region Inference API까지 동원해 처리량 극대화

MODELS

9개

고성능 AI 모델 무단 호출

INVOCATIONS

13회

총 Model Invocation

CROSS-REGION INFERENCE API 활용

다른 리전 자원까지 끌어와 모델 호출 처리량 극대화

MODELS INVOKED Claude DeepSeek + 외 7종

GPU 탈취 + 인증 없는 JupyterLab 백도어

자격증명이 차단되어도 살아남는 독립적 백도어 설계

GPU 인스턴스

p4d.24xlarge

$32.77/h

피해자가 비용 부담

01 **user-data 스크립트 자동 실행**

인스턴스 부팅과 동시에 백도어 설치 시작

02 **보안 그룹 0.0.0.0/0 개방**

전 세계 어디서든 접근 가능

03 **인증 없는 JupyterLab 서버 (포트 8888) 실행**

자격증명 차단 후에도 GPU 서버로 계속 접근 가능

1시간 51분 만에 수동 대응으로 차단

최초 침투부터 차단까지의 전체 시간선

권한 탈취

8분

관리자 권한 획득까지

차단까지

1h 51m

방어자의 수동 대응 완료

KEY MESSAGE

탐지가 늦은 게 아니라 — 공격 속도가 사람의 대응 루프 바깥에 있었다

기존 방어 체계가 무력했던 이유

탐지가 늦은 것이 아니라 공격 속도가 사람의 대응 루프를 벗어났다

정상으로 보이는 API 호출

AssumeRole
정상 운영의 일상적 호출 패턴

UpdateFunctionCode
CI/CD 배포 시 흔히 발생하는 호출

CreateAccessKey
신규 직원 온보딩 시에도 발생

→ 개별 API 기준으로는 악성 여부 판단 어려움

대응 속도 간극

ATTACKER

AI 보조 자동 공격

8분
관리자 권한 확보

DEFENDER

수동 알람 분석

30분+
알람 확인 · 분석 소요

KEY MESSAGE

탐지가 늦은 것이 아니라 공격 속도가 사람의 대응 루프를 벗어났다

공격 피해 규모

Identity 분산으로 임계치 기반 탐지를 우회한 광범위한 침해

IAM ROLE

6개

Role 악용

SESSION

14개

Session 생성

USER KEY

4명

기존 User Key 탈취

BACKDOOR

1개

Backdoor User 신규 생성

TOTAL · AWS IDENTITIES COMPROMISED

Identity를 분산해 이상 행동 임계치 기반 탐지를 우회

19개

LLMJACKING

Bedrock Model 9개 무단 호출

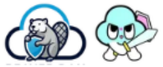
총 13회 Model Invocation

GPU HIJACKING

고성능 GPU p4d.24xlarge

인증 없는 JupyterLab 실행

AI가 공격을 주도한 결정적 증거



AI Hallucination + 자동화 패턴 + 세션명까지

SIGNAL 01

짧은 시간에 생성된 정교한 Lambda 코드

사람이 8분 안에 작성하기 어려운 수준의 완성도

SIGNAL 02

과도한 try/except 예외 처리

LLM이 만든 코드의 특징적 안전 패턴

SIGNAL 03

코드 주석에 세르비아어 사용

공격자 모국어로 추정되는 LLM 출력 흔적

HALLUCINATION 04

존재하지 않는 GitHub Repo Clone 시도

LLM이 환각으로 만들어낸 가공의 저장소 URL

HALLUCINATION 05

허구의 테스트용 Account ID 사용

LLM 학습 데이터의 예제 ID가 그대로 노출

SMOKING GUN 06

Session 이름에 claude-session

AI 호출 흔적이 로그에 직접 남음

KEY 사람이 주도한 공격이 아니라, AI가 정찰·코드 생성·의사결정까지 대신한 공격

챕터 3

Case 2

Bedrock AgentCore Starter Toolkit

Agent God Mode 권한 구조 취약점

SOURCE
Unit 42
보안 연구 공개

FLAW
Agent God Mode
과도한 Wildcard 권한

PERSPECTIVE
AI = 공격 대상
새로운 공격 표면

IMPACT
1 Agent → All
계정 전체 장악

핵심 서비스 3계층 정의

Bedrock · AgentCore · Starter Toolkit이 만드는 자동화 스택

LAYER 01

Amazon Bedrock

Foundation Model API

- Foundation Model을 API로 제공
- 생성형 AI 관리형 서비스
- 다양한 Vendor 모델 통합 접근

🧠 AI 모델 호출 계층

LAYER 02

Bedrock AgentCore

AI Agent 실행 환경

- Agent 배포 및 운영 플랫폼
- Memory · Runtime · Code Interpreter
- 장기 대화 및 실행 컨텍스트 관리

🤖 Agent 실행 계층

LAYER 03

AgentCore Starter Toolkit

자동 배포 도구

- Agent 배포 환경 자동 구성
- IAM Role 자동 생성
- 인프라 리소스 자동 프로비저닝

⚙️ 자동화 계층 ⚠️

Unit 42 위협 리서치 — 치명적 발견

Starter Toolkit의 취약한 IAM 권한 구조

UNIT 42 핵심 발견

01 Unit 42가 공개한 **Bedrock AgentCore 보안 연구**

02 Starter Toolkit 기본 설정에서 **과도한 IAM 권한** 발견

03 계정 내 Agent 간 **경계가 약화**되는 구조 확인

04 Agent 하나가 침해되면 **다른 Agent까지 통제 가능**

05 **최소 권한 원칙**이 깨진 권한 구조

DEFAULT ≠ SAFE

기본값

의 함정

편리한 자동화 = 보안 검증 부재

KEY MESSAGE

공식 도구의 기본값이
항상 안전한 것은 아니다

왜 'God Mode'라는 이름이 붙었나

단순히 권한이 넓은 게 아니라, 권한들이 맞물려 공격 체인을 완성한다

최소 권한 원칙

자신이 수행할 작업의 딱 필요한 리소스에만 접근

스타터 툴킷 기본값

와일드카드 정책으로 최소 권한 원칙 위배

👑

왜 'GOD MODE'인가

4가지 와일드카드가 맞물려 하나의 공격 체인을 완성

KEY MESSAGE

단일 권한 문제가 아니다 — 권한 조합이 전체 계정을 위협한다

Agent God Mode 권한 구조

Agent 1개 침해 → 세 가지 공격이 한꺼번에 달성

맞물린 4가지 WILDCARD 권한

🧠 memory:*

Memory

계정 내 모든 Agent의
대화 기록 · 민감 정보 접근

⚡ interpreter:*

Code Interpreter

고권한 Interpreter 호출로
실행 권한 상승

📦 ecr:*

ECR

모든 Container Image Pull
소스 코드 · 설정 노출

▶️ runtime:*

Runtime

다른 Agent Runtime 호출로
내부 확산 경로 확보

ATTACK CHAIN

공격 흐름 요약

1. Agent 1개 침해
2. ECR Image Pull
3. Memory ID 추출
4. Memory 탈취 및 변조
5. 고권한 Interpreter 호출 → 권한 상승

KEY 단순히 권한이 넓은 문제가 아니다 — 여러 와일드카드 권한이 연결되어 공격 체인을 형성한다

Memory Wildcard 권한 — Agent 간 데이터 경계 붕괴

memory:* · 자기 Memory만 접근해야 하는데, 전체 Memory에 접근 가능

```
{
    "Sid": "BedrockAgentCoreMemory",
    "Effect": "Allow",
    "Action": [
        "bedrock-agentcore:CreateEvent",
        "bedrock-agentcore:GetEvent",
        "bedrock-agentcore:GetMemory",
        "bedrock-agentcore:GetMemoryRecord",
        "bedrock-agentcore:ListActors",
        "bedrock-agentcore:ListEvents",
        "bedrock-agentcore:ListMemoryRecords",
        "bedrock-agentcore:ListSessions",
        "bedrock-agentcore:DeleteEvent",
        "bedrock-agentcore:DeleteMemoryRecord",
        "bedrock-agentcore:RetrieveMemoryRecords"
    ],
    "Resource": [
        "arn:aws:bedrock-agentcore:ap-northeast-2:                    :memory/*"
    ]
```

RISK SUMMARY

정상 구조

자기 자신의 Memory ARN만 접근

현재 취약 구조

계정 내 전체 Memory 접근 가능

유출 위험

다른 Agent 대화 기록 탈취

치명적 위험

장기 Memory 조작 → AI 판단 오염

Agent 간 데이터 경계 붕괴

Code Interpreter Wildcard — 권한 상승 통로

interpreter:* · 낮은 권한 Agent가 고권한 Interpreter를 호출

```
"Sid": "BedrockAgentCoreCodeInterpreter",
"Effect": "Allow",
"Action": [
    "bedrock-agentcore:CreateCodeInterpreter",
    "bedrock-agentcore:StartCodeInterpreterSession",
    "bedrock-agentcore:InvokeCodeInterpreter",
    "bedrock-agentcore:StopCodeInterpreterSession",
    "bedrock-agentcore:DeleteCodeInterpreter",
    "bedrock-agentcore:ListCodeInterpreters",
    "bedrock-agentcore:GetCodeInterpreter",
    "bedrock-agentcore:GetCodeInterpreterSession",
    "bedrock-agentcore:ListCodeInterpreterSessions"
],
"Resource": [
    "arn:aws:bedrock-agentcore:ap-northeast-2:aws:code-interpreter/*",
    "arn:aws:bedrock-agentcore:ap-northeast-2:            :code-interpreter/*",
    "arn:aws:bedrock-agentcore:ap-northeast-2:            :code-interpreter-custom/*"
]
```

RISK SUMMARY

기본 구조

Interpreter는 별도 IAM Role로 실행

취약점

다른 Interpreter 호출 가능

권한 상승

낮은 권한 → 고권한 Interpreter

악성 코드 실행 → 수평 이동

실행 권한 상승 통로

ECR Wildcard — 모든 이미지 무단 Pull

ecr:* · 컨테이너 내부 소스 코드와 설정이 통째로 노출

```
{
    "Sid": "ECRImageAccess",
    "Effect": "Allow",
    "Action": [
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
    ],
    "Resource": [
        "arn:aws:ecr:ap-northeast-2:                :repository/*"
    ]
}
```

RISK SUMMARY

Agent Runtime은 Container 기반

계정 내 모든 Image Pull 가능

소스 코드 · 환경 변수 · 설정값

치명적 결과

Memory ID 탈취의 첫 열쇠

내부 구조 유출의 진입점

Runtime Wildcard — Agent 간 상호 트리거

runtime:* · 핵심 Agent를 외부에서 깨워 명령 실행

```
{
    "Sid": "BedrockAgentCoreRuntime",
    "Effect": "Allow",
    "Action": [
        "bedrock-agentcore:InvokeAgentRuntime",
        "bedrock-agentcore:InvokeAgentRuntimeForUser"
    ],
    "Resource": [
        "arn:aws:bedrock-agentcore:ap-northeast-2:                :runtime/*"
    ]
},
```

RISK SUMMARY

권한 정의

Agent 실행 Trigger 권한

취약점

다른 Agent Runtime 호출 가능

공격 시나리오

핵심 Agent 외부에서 깨움

결과

Agent 간 안전 경계 붕괴

계정 내부 확산 경로

에이전트 1개 침해에서 계정 전체 장악까지

추가 취약점 없이 기본 권한만으로 전체 AI 인프라 침해 시나리오

ATTACK CHAIN OVERVIEW

STAGE 01

하위 Agent 1개 침해

공격자 진입점 확보
기본 권한만 보유

→

STAGE 02

ECR Image Pull

핵심 Agent 소스 코드 확보

→

STAGE 03

Memory ID 추출

유추 불가능한 식별자 확보

→

STAGE 04

Memory 탈취 · 변조

대화 기록 유출
AI 판단 오염

→

STAGE 05

권한 상승

고권한 Interpreter호출

전제 조건 추가 취약점 불필요 · **기본 Wildcard 권한만으로 충분** · Toolkit 기본값 그대로 사용 시 발생

KEY Agent 하나의 침해가 전체 AI 인프라 침해로 확장될 수 있다

1단계: ECR Image Pull로 소스 코드 확보

ECR 와일드카드는 핵심 Agent 내부를 열어보는 첫 번째 열쇠

STEP-BY-STEP

01 침해 Agent 권한으로 ECR 인증 토큰 발급
ecr:GetAuthorizationToken 호출

02 Docker CLI 인증 수행
표준 Docker 워크플로우 사용 — 별도 도구 불필요

03 핵심 Agent Container Image Pull
계정 내 다른 Agent의 이미지 다운로드 가능

04 소스 코드 · 설정 파일 완전 노출
알고리즘 · 환경 변수 · 내부 식별자 분석 가능

UNLOCKED INTELLIGENCE

노출되는 정보

💻 **Agent 소스 코드**
내부 로직과 의사결정 알고리즘

⚙️ **환경 설정 파일**
Config · Secrets · 환경 변수

🔑 **내부 식별자 (Memory ID 등)**
다음 단계 공격의 핵심 단서

🎯 **공격 표면 전체 매핑**
내부 구조 파악 → 정밀 공격 가능

KEY MESSAGE

ECR 와일드카드는 핵심 Agent 내부를 여는 첫 열쇠

2단계: Memory ID 추출

Image 내부 정보가 Memory 공격의 열쇠가 된다

STATIC ANALYSIS

로컬 분석으로 식별자 확보

1. **Pull한 Image 로컬 보관**
   탐지 없이 오프라인 분석 가능
2. **코드 · 설정 파일에서 ID 탐색**
   grep 한 번이면 충분
3. **Memory ID 값 확보**
   하드코딩 · 설정 파일에서 직접 노출

RESULT **유추 불가능한 식별자를 직접 확보**

DEFENSE COLLAPSE

BEFORE

Memory ID = 마지막 방어선

유추 불가능한 식별자가 Memory 접근을
막는 사실상의 보안 토큰 역할을 수행

AFTER

방어선 붕괴

ECR 취약점이 Memory 공격의
진입 장벽을 완전히 제거

KEY MESSAGE

Image 내부 정보가 Memory 공격의 열쇠

3단계: Memory 탈취 및 변조

정보 유출을 넘어 AI 판단 자체를 무기화한다

PATH A · STEAL

Memory 탈취

- 💬 장기 대화 기록 유출
- 🔐 사용자 인증 정보 노출
- 📋 기업 기밀 정보 유출

RESULT
광범위한 정보 유출

PATH B · MODIFY ⚠️

Memory 변조

- ✏️ Memory 데이터 직접 수정
- 🧬 Agent 컨텍스트 조작
- ⚠️ Agent 판단 로직 오염

RESULT
AI 판단 자체가 무기화

KEY Memory 공격은 정보 유출을 넘어 AI 판단 오염으로 이어진다

취약점의 원인은 결국 세 가지 와일드카드

공동 책임 모델 — IAM 정책은 사용자의 책임 영역이다

ROOT 01

ECR 와일드카드

소스 코드·알고리즘 통째로 노출

ROOT 02

Memory 와일드카드

Agent 간 데이터 탈취 통로 개방

ROOT 03

Interpreter 와일드카드

수평 이동 · 권한 상승 자유 통로

교훈 **자동화될수록 — 어떤 권한 체인을 엮어내는지 사람이 직접 검토해야 한다**

KEY 공동 책임 모델 — IAM 정책은 사용자의 책임 영역

챕터 4

결론 & 전략

AI 시대의 클라우드 방어 전략

PART 01
두 케이스의 결론과
패러다임 변화

PART 02
AI 위협 대응
자동화된 방어

PART 03
AI 사용 위협 대응
공격자 관점 검증

AI는 공격 도구이자 공격 대상이 되었다

상반되면서도 연결된 두 사건 — 같은 결론을 가리킨다

CASE 01

AI가 공격 속도를 극단적으로 가속

속도의 위협

- → 관리자 권한 탈취 8분
- → 사람의 대응 루프를 초월
- → AI 보조 자동화 공격

CASE 02

AI Agent가 새로운 공격 표면으로 등장

구조의 위협

- → Agent God Mode 권한
- → Agent 간 격리 붕괴
- → 기본값의 함정

NEW REALITY

⚡ 공격자는 더 빠르게 움직인다

🌐 공격 표면은 더 복잡해진다

⏱️ 방어자는 더 짧은 시간 안에

KEY AI 시대의 보안은 속도와 권한 구조를 함께 다뤄야 한다

방어 패러다임의 변화

완벽한 방어 → 리스크 완화 관점으로

잘못된 패러다임

100% 완벽한 방어

모든 공격을 막아내는 차단 중심 사고

올바른 패러다임

리스크 완화

발생 확률 ↓ + 피해 반경 최소화

새로운 질문 01

우리의 통제가 위협 발생 확률을 얼마나 낮추는가?

새로운 질문 02

침해 시 피해 반경을 어디까지 격리할 수 있는가?

방어 전략: AI 위협의 두 갈래 대응

AI를 통해 발생한 위협 vs AI를 사용하다 발생한 위협

전략 A

AI를 통해 발생한 위협 대응

Case 1 시나리오 — 8분의 공격 속도

🔍 행동 패턴 조합 기반 모니터링

⚡ 실시간 자동 격리 파이프라인 구축

전략 B

AI를 사용하다 발생한 위협 대응

Case 2 시나리오 — Agent God Mode

📋 사전 검증 프로세스 선행

🎯 공격자 관점의 시나리오 검증

행동 패턴 조합 기반 모니터링

파편화된 정상 행위를 하나의 연쇄 패턴으로 묶어 탐지한다

기존 방식의 한계

단일 API 이벤트 탐지

- **AssumeRole** → 정상 운영 행위로 분류
- **UpdateFunctionCode** → CI/CD 배포로 분류
- **CreateAccessKey** → 신규 직원 온보딩으로 분류

→ 개별 호출은 탐지 어려움

새 패러다임

행동 Chain 분석

⚠ 대규모 Describe·List 호출 급증

⚠ UpdateFunctionCode 직후 CreateAccessKey

⚠ AssumeRole + Bedrock Model 호출 급증

→ 시퀀스 자체가 공격 시그널

KEY 이벤트 하나가 아니라 연속된 행동 패턴을 봐야 한다

"CloudTrail 딜레이는?"

완전한 차단은 못 해도, 피해 범위를 줄이는 데 실효성이 있다

한계

15분

이벤트 발생 → S3 적재까지
평균 CloudTrail 딜레이

그럼에도 의미

피해 범위 축소

권한 탈취는 막지 못해도, 이후 확산을 차단할 수 있다

Case 1 피해 발생 시점

8분
관리자 권한 탈취

→

58분
LLMjacking 시작

→

1시간 42분
GPU 탈취 본격화

KEY 15분 후 탐지라도 이후 1시간 이상의 피해는 막을 수 있다

실시간 자동 격리 파이프라인 구축

AI 자동화 공격에는 자동화된 방어 파이프라인이 필요하다

AUTO-RESPONSE PIPELINE

1. **EventBridge** · 위험 이벤트 감지
2. 격리 전용 **Lambda Bot** 실행
3. 탈취 의심 Access Key 비활성화
4. 신규 관리자 Key 즉시 차단
5. 관련 Identity에 **Deny-All** Policy 적용

EXPECTED OUTCOMES

기대 효과

IMPACT 01

관리자 권한 탈취 이후 추가 확산 차단

IMPACT 02

LLMjacking · GPU 탈취 피해 최소화

IMPACT 03

사람의 분석 시간 자동화로 보완

사전 검증 프로세스 선행

공식 도구라도 기본값을 그대로 신뢰하지 않는다

PRE-DEPLOYMENT CHECKLIST

01 **Toolkit 생성 IAM Policy 검토**
자동 생성 정책의 권한 범위 전수 검사

02 **Agent별 Resource 접근 범위 확인**
Wildcard 권한 존재 여부 점검

03 **Memory ARN 제한 여부 확인**
자기 Memory만 접근 가능한지 검증

04 **ECR Pull 권한 범위 확인**
Repository 단위 제한 적용 검증

05 **Code Interpreter 호출 권한 확인**
권한 상승 경로 차단 여부 검증

06 **Runtime 상호 호출 권한 확인**
Agent 간 격리 경계 유지 검증

공격자 관점으로의 시선 전환

컴플라이언스 체크리스트를 넘어, 공격자 관점의 시나리오 검증이 필요하다

RED-TEAM QUESTIONS

하위 Agent 1개가 침해되면 무엇이 가능한가?

- **Q1.** 다른 Agent의 **Memory에 접근**할 수 있는가?
- **Q2.** **ECR Image를 Pull**할 수 있는가?
- **Q3.** **고권한 Interpreter**를 호출할 수 있는가?
- **Q4.** Runtime을 통해 **핵심 Agent를 실행**할 수 있는가?
- **Q5.** **하나라도 'Yes' = 공격 Chain 형성**

DEFENSE PRINCIPLES

Agent별 격리
강력한 데이터/실행 경계 유지

최소 권한 원칙
Wildcard 제거 · 명시적 ARN

Zero Trust
암묵적 신뢰 일체 제거

호출 시 신원 재검증
매 호출마다 권한 확인

AI 인프라에 심어야 할 4가지 방어 원칙

제로 트러스트를 AI 인프라 내부까지 — 도구 자동화는 검증의 면제가 아니다

PRINCIPLE 01

Agent별 격리

에이전트 바운더리마다 명확한 격리 벽 세우기

PRINCIPLE 02

리소스 ARN 제한

Wildcard 제거 · 특정 ARN으로 접근 범위 명시

PRINCIPLE 03

제로 트러스트

암묵적 신뢰 일체 제거, AI 인프라 내부에도 적용

PRINCIPLE 04

호출 시 신원 재검증

Agent · Interpreter 상호 호출 시, 매번 신원을 다시 확인

KEY 자동화는 검증의 면제가 아니다 — AI가 딛고 선 인프라를 직접 검증해야 한다

CLOSING TAKEAWAYS

마무리하며

01 AI는 클라우드 공격의 **속도**를 바꾸고 있다

02 AI Agent는 새로운 **공격 표면**이 되고 있다

03 방어자는 AI가 딛고 있는 **인프라를 직접 검증**해야 한다

04 완벽한 방어보다 **공격 성공률을 낮추는 것**이 중요하다

05 **피해 반경을 줄이는 리스크 완화** 관점이 필요하다

CLOSING MESSAGE

AI 시대의 클라우드 보안은 자동화된 방패와 검증된 권한 구조에서 시작됩니다

Q&A

- Sysdig. (2026). *AI-Assisted Cloud Intrusion Achieves Admin Access in 8 Minutes*.
- Sysdig Blog. https://www.sysdig.com/blog/ai-assisted-cloud-intrusion-achieves-admin-access-in-8-minutes

- Palo Alto Networks Unit 42. (2026). Cracks in the Bedrock: Agent God Mode.
- Unit 42 Research. https://unit42.paloaltonetworks.com/exploit-of-aws-agentcore-iam-god-mode/

박민서 · Beaverdam Community Builder