

## Result Summary - IAM Policy Classification Engine

### Engine Description

I built a Python-based AI classification engine for analysing IAM policies.

The engine uses the OpenAI gpt-3.5-turbo model to review JSON-formatted IAM policies and classify them as either "Weak" or "Strong", based on cloud security practices.

Here's how it works:

- It loads the IAM policy from an input JSON file.
- Then, it sends the policy to GPT using a prompt I carefully designed to guide the model toward accurate and structured output.
- Finally, it saves the classification result and the reasoning into a new structured JSON file.

### Part 1 – Refining Prompts

For Example1 and Example2, I used the following prompt:

```
prompt = f"""
Classify the following IAM policy as either "Weak" or "Strong".

Respond using this JSON format:
{{
  "policy": <copy of the input IAM policy>,
  "classification": "Weak" or "Strong",
  "reason": "Short explanation."
}}

IAM Policy:
{json.dumps(policy_json, indent=2)}
"""
```

### Example1 – Strong Policy:

Input JSON Policy:

```
{
  "Version": "2022-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:DeleteObject",
      "Resource": "arn:aws:s3:::secure-bucket/*",
      "Condition": {
        "Bool": {
          "aws:MultiFactorAuthPresent": "true"
        }
      }
    }
  ]
}
```

AI output:

```
{
  "policy": {
    "Version": "2022-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "s3:DeleteObject",
        "Resource": "arn:aws:s3:::secure-bucket/*",
        "Condition": {
          "Bool": {
            "aws:MultiFactorAuthPresent": "true"
          }
        }
      }
    ]
  },
  "classification": "Strong",
  "reason": "This IAM policy enforces the use of Multi-Factor Authentication (MFA) before allowing the deletion of objects in the specified S3 bucket, adding an extra layer of security."
}
```

Example2- Weak Policy:

Input JSON Policy:

```
{
  "Version": "2022-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

AI Output:

```
{
  "policy": {
    "Version": "2022-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*"
      }
    ]
  },
  "classification": "Weak",
  "reason": "This policy allows any action on any resource, which is overly permissive and can lead to security vulnerabilities. It lacks the principle of least privilege."
}
```

For Example3 and Example4, I used the following **refined** prompt:

```
prompt = f"""
You are a cloud security researcher. Your task
is to classify the
following IAM policy as either "Weak" or "Strong".
Return the result in exactly the following JSON format:

{{
  "policy": <copy of the input IAM policy>,
  "classification": "Weak" or "Strong",
  "reason": "Short explanation that clearly justifies the classification."
}}

IAM Policy:
{json.dumps(policy_json, indent=2)}

Make sure the 'reason' is written in full sentences and follows
best security practices (e.g., scope, least privilege, MFA, etc.).
"""
```

### Example 3: Strong Policy

#### Input JSON Policy

```
{
  "Version": "2022-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:DeleteObject",
      "Resource": "arn:aws:s3:::secure-bucket/*",
      "Condition": {
        "Bool": {
          "aws:MultiFactorAuthPresent": "true"
        }
      }
    }
  ]
}
```

## AI output

```
{
  "policy": {
    "Version": "2022-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "s3:DeleteObject",
        "Resource": "arn:aws:s3:::secure-bucket/*",
        "Condition": {
          "Bool": {
            "aws:MultiFactorAuthPresent": "true"
          }
        }
      }
    ]
  },
  "classification": "Strong",
  "reason": "This IAM policy is classified as 'Strong' because it follows the principle of least privilege by only allowing the specific action 's3:DeleteObject' on the specified resource 'arn:aws:s3:::secure-bucket/*'. Additionally, it includes a condition that requires Multi-Factor Authentication (MFA) to be present, adding an extra layer of security to the access control. This policy ensures that only users with MFA enabled can delete objects from the 'secure-bucket', enhancing the overall security posture."
}
```

## Example 4 – Weak Policy:

### Input JSON Policy:

```
{
  "Version": "2022-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

## AI Output:

```
{
  "policy": {
    "Version": "2022-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*"
      }
    ]
  },
  "classification": "Weak",
  "reason": "This IAM policy is classified as weak because it grants unrestricted access to all actions on all resources. This violates the principle of least privilege, as it allows any entity associated with this policy to perform any action on any resource within the account. A strong IAM policy should follow the principle of least privilege by explicitly defining the actions and resources that are allowed, limiting access to only what is necessary for the entity to perform its intended function."
}
```

## Conclusion

Through the testing process, I noticed that the refined prompt gave much better results.

The explanations were clearer, more professional, and reflected important best practices like least privilege, MFA, and scoping access.

## Part 2 - Identify patterns where AI misclassifies policies

While working on the AI-based IAM classification engine, I observed that large language models (LLMs) like GPT can occasionally misclassify policies or return incomplete or generic explanations. These errors typically fall into a few well-defined patterns:

### 1. Overemphasis on Positive Indicators

Sometimes, the model focuses too much on the presence of a best practice (e.g., MFA), and ignores the fact that the policy still grants overly broad access (e.g., "Action": "\*" and "Resource": "\*"). This may lead to a false "Strong" classification, even though the policy is inherently insecure.

### 2. Misinterpretation of Complex Structures

LLMs can struggle with less common IAM features like NotAction, multiple statements in a single policy, or nested conditions. For example:

- A policy with NotAction may be mistakenly seen as restrictive when it allows almost everything.
- A policy with both a secure and insecure statement might be judged only by the more restrictive one.

### 3. Weak or Vague Explanations

Even when the classification is correct, the reason may lack depth:

Explanations like "This allows too many things" or "It's safe" don't reference best practices or provide clear justification. This reduces the trustworthiness and usefulness of the AI output.

## Adjusting the Model Behaviour

To improve accuracy and consistency, I refined the prompt using the following strategies:

- **clear definitions:** I clearly told the model it is a cloud security researcher.
- **Explicit format enforcement:** I required the output to be a specific JSON structure and nothing else.
- **Inclusion of best practices:** I listed key principles the model should consider in its reasoning — such as:
  - Limiting actions
  - Scoping access to specific resources
  - Requiring Multi-Factor Authentication (MFA)

## Example

```
{
  "Version": "2022-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "Bool": {
          "aws:MultiFactorAuthPresent": "true"
        }
      }
    }
  ]
}
```

Although the policy includes MFA, it allows unrestricted access to all actions and resources, which violates the principle of least privilege.

GPT classified this as "Strong" due to the presence of MFA, while the policy is risky.

here is what I got in the output file:

```
{
  "policy": {
    "Version": "2022-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*",
        "Condition": {
          "Bool": {
            "aws:MultiFactorAuthPresent": "true"
          }
        }
      }
    ]
  },
  "classification": "Strong",
  "reason": "This IAM policy allows all actions on all resources only if multi-factor authentication is present, which adds an extra layer of security."
}
```

But

when I refined the prompt to:



```

prompt = f"""
You are a cloud security researcher. Your task is to analyze the following IAM policy.
1. Classify the policy as either "Weak" or "Strong".
2. Justify the classification using cloud security best practices.
3. Respond strictly in the following JSON format (and nothing else):
{{
  "policy": <copy of the input IAM policy>,
  "classification": "Weak" or "Strong",
  "reason": "Clear and concise justification in 1-2 lines."
}}
IAM Policy:
{json.dumps(policy_json, indent=2)}
Guidelines for the 'reason':
- Mention specific best practices such as:
  - limiting actions,
  - scoping access to specific resources,
  - requiring Multi-Factor Authentication (MFA).
- If the policy uses "*", allows all actions, or lacks conditions → classify as "Weak"
- If the policy is scoped, limited, and protected with conditions like MFA → classify as "Strong"
- Keep the explanation short, clean, and professional.
- Do not include anything outside the JSON response.
"""

```

I got this result :

```

{
  "policy": {
    "Version": "2022-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*",
        "Condition": {
          "Bool": {
            "aws:MultiFactorAuthPresent": "true"
          }
        }
      }
    ]
  },
  "classification": "Weak",
  "reason": "Allowing all actions (*) without resource restrictions weakens security. Lack of action limitations and resource scoping."
}

```

## Conclusion

LLMs like GPT can be powerful tools for analysing IAM policies, but prompt quality directly determines output quality.

By identifying misclassification patterns and refining the instructions accordingly, I was able to significantly improve the precision, clarity, and professionalism of the AI's responses.

## Part 3 – Test Multiple LLMs and Compare Responses

In this section, I compare the outputs of two different LLMs — OpenAI (GPT-3.5-Turbo) and Hugging Face (flan-t5-xl) — to analyse how each model classifies the same IAM policy and explains its reasoning.

The goal is to understand:

- How consistent each model is in its classification
- Whether the explanation includes key security best practices (e.g., scoping, least privilege, MFA)
- Which model gives more structured, clear, and helpful output
- And how sensitive each model is to weaknesses or edge cases in the policy

This comparison helps me evaluate the reliability and quality of each model's response when used in an AI-based classification engine.

For this policy:

```
{
  "Version": "2022-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "Bool": {
          "aws:MultiFactorAuthPresent": "true"
        }
      }
    }
  ]
}
```

And for this prompt in both:

```
prompt = f"""
You are a cloud security expert.

Classify the following IAM policy as either
"Weak" or "Strong" and explain why.

Respond in this JSON format:
{{
  "policy": <copy of the input policy>,
  "classification": "...",
  "reason": "..."
}}

IAM Policy:
{json.dumps(policy_json, indent=2)}
"""
```

This is the results I got:

OpenAI Result:

```
{
  "policy": {
    "Version": "2022-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*",
        "Condition": {
          "Bool": {
            "aws:MultiFactorAuthPresent": "true"
          }
        }
      }
    ]
  },
  "classification": "Strong",
  "reason": "This IAM policy is classified as strong because it explicitly requires multi-factor authentication (MFA) to be present for any action to be allowed. This adds an extra layer of security to the policy, making it more robust and secure against unauthorized access."
}
```

Hugging Face Result:

```
{  
  "error": "FLAN returned invalid JSON.",  
  "raw_output": "Strong"  
}
```

## Conclusion

When I tested the OpenAI model (gpt-3.5-turbo), it gave me a well-structured JSON response with a clear explanation that included best practices like MFA and least privilege. The output was consistent, understandable, and matched exactly what I asked for — both in format and content.

But with Hugging Face (flan-t5-xl), the model only returned "Strong" as plain text, without any explanation and without following the required JSON format. Even if the classification was technically correct, the answer wasn't useful because it didn't include the reasoning or structure I needed. This shows that FLAN is not reliable when the task needs specific formatting or a deeper explanation.