These instructions detail how to create IAM roles for your Lambda function and API Gateway, emphasizing security best practices. **Creating custom policies with least privilege is strongly recommended over using the full-access managed policies.**

I. Creating the IAM Role for the Lambda Function:

1. **Open the IAM Console:** Log in to the AWS Management Console and navigate to the IAM service.
2. **Create a New Role:** Click "Create role."
3. **Select Trusted Entity Type:** Choose "AWS service" as the trusted entity type.
4. **Select AWS Service:** Select "Lambda" as the service that will use this role.
5. **Attach Policies (Least Privilege Approach):** Instead of attaching the full-access managed policies, create a custom policy with only the necessary permissions. This is crucial for enhanced security:
    - **Create Custom Policy:** Click "Create policy," then "JSON."
    - **Define Permissions (Example):** Paste the following JSON, adjusting resource ARNs to match your specific S3 buckets, DynamoDB table, and Lambda function:

```
 {
"Version": "2012-10-17",
"Statement": [
   {
      "Effect": "Allow",
      "Action": [
         "logs:CreateLogGroup",
         "logs:CreateLogStream",
         "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
   },
   {
      "Effect": "Allow",
      "Action": [
         "s3:GetObject",
         "s3:DeleteObject"
      ],
      "Resource": [
         "arn:aws:s3:::your-resume-uploads-bucket/*",  // Replace with your bucket ARN
         "arn:aws:s3:::your-processed-data-bucket/*"   //Replace with your bucket ARN
      ]
   },
   {
      "Effect": "Allow",
```

```
            "Action": "textract:AnalyzeDocument",
            "Resource": "*"  //You might want to restrict this further, but this depends on your setup.
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:GetItem"
            ],
            "Resource": "arn:aws:dynamodb:*:*:table/ParsedResumes" //Replace with your table
ARN
        }
    ]
}
```

6. content_copy Use code with caution.Json
    - **Review and Create Policy:** Review the policy carefully, ensuring it only grants the necessary permissions. Give it a name (e.g., LambdaResumeParserPolicy) and click "Create policy."
7.
8. **Attach Policy to Role:** Go back to your role creation process and attach the custom policy you just created.
9. **Review and Create Role:** Review the permissions and create the role. **Note the role's ARN.** You'll need this when configuring your Lambda function.

II. Creating the IAM Role for API Gateway:

1. **Create a New Role:** Follow steps 1-3 from the Lambda role creation process.
2. **Select AWS Service:** Select "API Gateway" as the service that will use this role.
3. **Attach Policies (Least Privilege):** Create a custom policy granting only the necessary permissions:
    - **Create Custom Policy (JSON):** Similar to the Lambda role, create a custom policy.
    - **Define Permissions (Example):** This policy only needs to invoke the Lambda function:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "arn:aws:lambda:*:*:function:your-lambda-function-name" // Replace with
your Lambda function ARN
```

```
    }
  ]
}
```

4. content_copy Use code with caution.Json
    - **Review and Create Policy:** Review the policy carefully and create it (e.g., name it APIGatewayInvokeLambdaPolicy).
5.
6. **Attach Policy to Role:** Attach this custom policy to your API Gateway role.
7. **Review and Create Role:** Review the permissions and create the role.

By using custom policies, Yanga significantly enhances the security of his application. Always follow the principle of least privilege when assigning permissions. Remember to replace the placeholder ARNs with the actual ARNs of your resources. Test these roles thoroughly to ensure they only provide the necessary permissions before deploying your application.