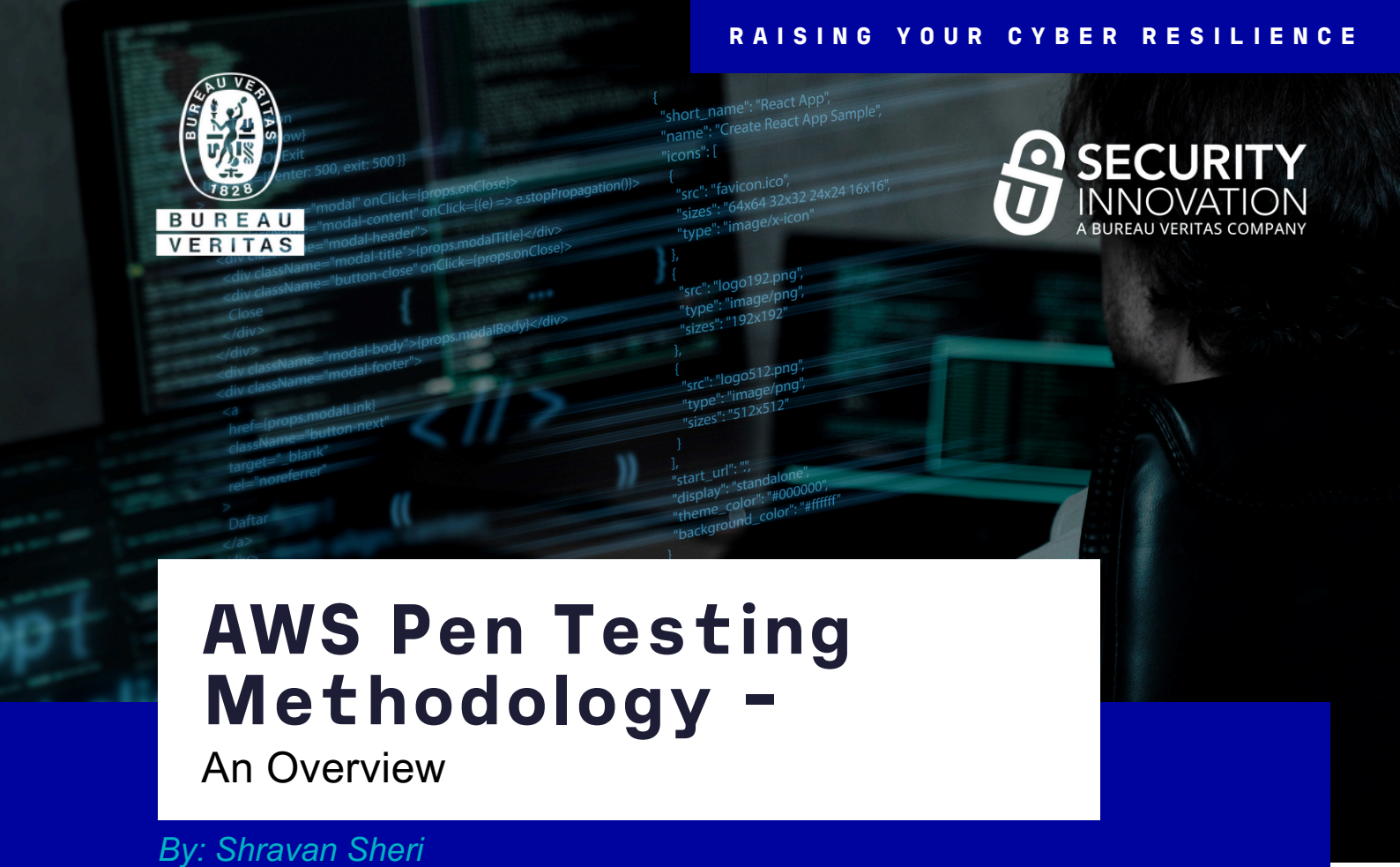

BUREAU VERITAS

SECURITY INNOVATION
A BUREAU VERITAS COMPANY

AWS Pen Testing Methodology -
An Overview

By: Shravan Sheri

The main goal of this guide is to provide a structured approach to black-box penetration testing for AWS, delivering a clear and practical learning experience.

What to expect

- The content is broken down into phases, explained in the Methodology section, so you can follow along step by step.
- The focus is on real-world issues, especially user misconfiguration vulnerabilities, which are the most common and impactful in practice.
- The Tools section simplifies access, enabling quick retrieval of necessary resources.
- The Certifications and Resources section is included to further explore.

Methodology

Information Gathering/Enumeration

The Information Gathering phase in AWS black-box penetration testing focuses on external reconnaissance to identify publicly exposed resources, misconfigurations, and vulnerabilities without having direct access to the target environment. Below are the detailed steps for some interesting and commonly targeted resources with useful options and flags to enhance testing efficiency

1. Enumerating S3 Buckets

Objective: Discover publicly accessible S3 buckets and test for lax permissions or sensitive file exposure.
Steps:

Use **s3scanner** to identify open buckets and test access permissions.

```
python3 s3scanner.py --bucket <bucket_name>
```

2. Searching for Exposed Git Logs

Objective: Identify exposed .git directories and analyze logs for sensitive information.
Steps:

Use **git-dumper.py** to retrieve an exposed .git directory:

```
./git_dumper.py http://website.com/.git ~/website
```

Analyze the config and logs files for AWS credentials or API keys using the **git command line**, you can learn more on how to do using this **article**.

3. Scanning Websites and APIs

Objective: Identify publicly exposed endpoints and vulnerabilities in web applications.
Steps:

Use **Amass** or **Sublist3r** for subdomain enumeration.

```
amass enum -d <target-domain>
    sublist3r -d <target-domain>
```

Perform port scanning with **Nmap** to identify open services.

```
nmap -Pn -p- <target-ip>
```

Use a brute-forcing tool like **Gobuster** to search for directories and files.

```
gobuster dir -u http://<target-domain> -w /path/to/wordlist.txt
```

4. Identifying Web Application Vulnerabilities

Objective: Exploit web vulnerabilities to gain access to AWS resources like EC2 metadata and SQS.
Steps:

Test for SSRF vulnerabilities that query EC2 metadata:

```
curl -X GET "http://<vulnerable-endpoint>/?url=http://169.254.169.254/latest/meta-data/"
```

The Analysis of unauthenticated API endpoints for sensitive information using Burp Suite is explained below.

5. Endpoint Detection Tools for AWS Reconnaissance

Endpoint Detection tools streamline AWS black-box testing, saving time and ensuring accuracy in large scopes. Tools like Shodan, and Censys quickly identify public resources, while httprobe pinpoint misconfigurations and live services. These tools accelerate discovery, improve accuracy, and allow testers to focus on analyzing critical findings, making the process faster and more efficient for complex environments.

Shodan

Objective: Identify publicly exposed AWS resources, such as EC2 instances, RDS databases, and ElasticSearch services by scanning IPs and domains.

Steps:

Search for exposed AWS-hosted services using Shodan:

```
"aws" OR "Amazon" site:shodan.io
```

Refine your search query with specific service details:

- ElasticSearch: product:ElasticSearch region:aws
- S3 Buckets: "s3.amazonaws.com"
- RDS Databases: port:3306 "aws"

Analyze the discovered services for misconfigurations or weaknesses explained in the Identify Misconfiguration or Weakness phase.

Censys

Objective: Enumerate AWS-hosted services and assets through IP and TLS metadata analysis.

Steps:

Use Censys to search for exposed AWS assets:

```
services.metadata.product:"Amazon AWS"
```

Filter results for specific services:

- Example for S3: "s3.amazonaws.com" {region can be inferred from the endpoint patterns like s3.us-east-1.amazonaws.com}
- Example for Elastic Load Balancers: "*.elb.amazonaws.com"

Investigate for publicly accessible configurations or mismanaged certificates

- Check service for metadata information in Censys for:
  - Exposed endpoints s3.amazonaws.com or *.elb.amazonaws.com.
  - TLS/SSL certificates for validity, expiration, or misconfiguration.
- Perform further manual validation of these endpoints using tools like curl or AWS CLI to verify permissions and accessibility.

Httprobe

Objective: Probe live subdomains for AWS-hosted services or endpoints.

Steps:

Use a subdomain enumeration tool (e.g., Amass) to generate a list of potential targets.

Pipe the results into httprobe to test for live endpoints:

```
cat subdomains.txt | httprobe
```

Analyze live domains to identify AWS-hosted services such as S3, Lambda, and API Gateway.

Identify Misconfigurations or Weaknesses

After information gathering, identifying misconfigurations and weaknesses in AWS is the next step in a black box penetration test. Misconfigurations are unintended settings or permissions that deviate from security best practices, leading to vulnerabilities. Weaknesses, on the other hand, are flaws that arise from improper implementation or lack of security controls.

Why is it important?

- Misconfigurations are among the most common causes of cloud breaches.
- AWS environments often expose resources due to overly permissive policies, poor Identity and Access Management (IAM) configurations, or lack of monitoring.
- Testing for these issues helps uncover potential entry points, lateral movement paths, and privilege escalation opportunities.

The transition from Information Gathering to this phase involves leveraging the collected data to directly test for exploitable flaws. The following are a few important ways you can identify weaknesses.

1. Publicly Accessible S3 Buckets

Why It Matters: S3 buckets with public read/write permissions can lead to data leaks or unauthorized tampering.

How to Identify Weaknesses:

Test Read Permissions: Use bucket names discovered during information gathering to check for public read access:

```
aws s3 ls s3://<bucket-name> --no-sign-request
```

Success: Indicates public read access.

Test Write Permissions: Attempt to upload a file to the bucket:

```
echo "Shravan Kumar Sheri owns this file" > test.txt
aws s3 cp test.txt s3://<bucket-name>/test-upload.txt --no-sign-request
```

Success: Confirms public write access, enabling tampering or malicious uploads.

Note: Cloudbreach has developed a script to automate these tests.

2. Exposed Identity and Access Management (IAM) Roles via EC2 Metadata

Why It Matters: Exposed EC2 metadata can leak temporary IAM credentials, enabling privilege escalation or lateral movement.

How to Identify Weaknesses:

Validate Retrieved Credentials: Use IAM credentials retrieved during information gathering (via SSRF or other methods) to test for permissions:

```
aws sts get-caller-identity --profile=hackedBySSK // retrieve username from this
```

Test Accessible Resources: Leverage the credentials to access AWS services:

```
aws s3 ls
aws ec2 describe-instances
```

Note: Cloudbreach developed a script to automate testing userdata attribute across multiple instances.

3. Open Security Groups

Why It Matters: Overly permissive security groups expose sensitive services, enabling unauthorized access to critical infrastructure (we can also combine information found from SSH keys found during directory brute-forcing or other exposed sensitive file).

How to Identify Weaknesses:

Verify Open Ports: Use open ports identified during information gathering to confirm accessible services:

```
nmap -Pn -p22,3389 <public-ip>
```

Test Accessible Services:

SSH

```
ssh <public-ip>
```

RDP

```
rdesktop <public-ip>
```

Success: Confirms that security group rules allow unrestricted access.

4. CloudFront Misconfigurations

Why It Matters: Misconfigured CloudFront distributions can expose backend services (e.g., S3 buckets or APIs), bypassing access controls.

How to Identify Weaknesses:

Analyze CloudFront Endpoints: Test known CloudFront endpoints for unrestricted access to origins:

```
curl -X GET https://<cloudfront-endpoint>
```

Inspect Backend Responses:

Direct access to S3 buckets

- If the CloudFront distribution is improperly configured then a curl requests to the URL might be forwarded directly to s3 Origin. For example, if below curl method returns a file stored in the s3 bucket, it indicates that the origin is exposed and accessible through CloudFront.

```
curl https://<cloudfront-endpoint>/file.txt
```

Exposed API responses or restricted content

- CloudFront API response might reveal information about the origin server, such as S3 URLS, through HTTP headers or error messages.
- Request

```
curl -I https://<cloudfront-endpoint>
```

- Response

```
x-amz-id-2
x-amz-request-id
```

- Error Messages Indicating Misconfigurations
  - CloudFront misconfigurations can reveal sensitive backend details through error messages in response headers and bodies. Use curl -I https://<cloudfront-endpoint> to inspect responses and identify potential leaks. Following are some examples
    API Gateway Misconfiguration (502 Bad Gateway):
  - HTTP/1.1 502 Bad Gateway
  - X-Cache: Error from cloudfront
    // Suggests backend connectivity issues or incorrect routing.
  - Leaked S3 Bucket Name (Response Body): Exposes bucket name, aiding targeted attacks.
  - <Error>
    <Code>AccessDenied</Code>
    <Message>Access Denied</Message>
    <HostId>example-bucket.s3.amazonaws.com</HostId>
    </Error>
  - Common CloudFront Error Codes:
    - 403 Forbidden: Misconfigured S3 bucket permissions.
    - 404 Not Found: Incorrect object path.
    - 502 Bad Gateway: Backend API issues.

5. Lambda Function Misconfigurations

Why It Matters: Misconfigured Lambda functions can expose sensitive data (e.g., environment variables) or allow unauthorized actions.
How to Identify Weaknesses

Probe API Endpoints: Test Lambda function endpoints identified during information gathering for sensitive information or execution issues:

curl -X GET https://<lambda-endpoint>

Analyze Responses:

- Environment variables or credentials in error messages.
- Signs of misconfigurations like verbose stack traces or execution errors.

Note: Lambda endpoints often follow a standardized syntax, similar to S3 buckets. By gathering enough information, you can construct these endpoints.

Example: If you know the AWS region and account ID, a Lambda endpoint typically looks like:

https://<lambda-function-name>.lambda.<region>.amazonaws.com/<DirName>

For instance, if the Lambda function name is my-function, the AWS region is us-east-1, and the stage is prod, the endpoint might be:

https://my-function.lambda.us-east-1.amazonaws.com/prod

Exploitation for Initial Access

This phase focuses on leveraging misconfigurations or weaknesses to gain access or establish a foothold in the target AWS environment. This phase combines techniques to exploit public resources, retrieve IAM roles or credentials, and escalate privileges, building on findings from the information gathering and identification phases.

1. Exploiting Publicly Accessible S3 Buckets

Objective: Leverage public read/write permissions to access sensitive files or tamper with data.
Steps

Download sensitive files:

```
curl -X GET https://<bucket-name>.s3.amazonaws.com/<file-name>
```

Inject malicious files:

```
curl -X PUT -T malicious.html https://<bucket-name>.s3.amazonaws.com/malicious.html
```

Note: This technique can achieve a full Remote Code Execution (RCE) on a company during its application security testing.

2. Exploiting API Misconfigurations

Objective: Test misconfigured AWS API Gateway endpoints for unauthenticated access or overly permissive Cross-Origin Resource Sharing (CORS) settings using Burp Suite.
Steps

Scenario 1: Access Unauthenticated Endpoints

- Open Burp Suite and configure the target endpoint in the Repeater tab.
- Set the HTTP method to POST and enter the AWS API Gateway endpoint:

https://<api-id>.execute-api.<region>.amazonaws.com/<stage>/<resource>

- In the Request Body, add a sample JSON payload:
- {
  "action": "create",
  "data": "test"
  }
- Send the request and inspect the response for any signs of successful data processing or creation.

Scenario 2: Test for Overly Permissive CORS Policies

- Open Burp Suite and configure the target endpoint in the Repeater tab.
- Set the HTTP method to OPTIONS and enter the AWS API Gateway endpoint.
- Add the following header to simulate a malicious origin:
- Key: Origin
  Value: evil.com

Send the request and inspect the response headers.

Note: If the response includes Access-Control-Allow-Origin: * or **evil.com**, it indicates overly permissive CORS settings and If the API processes the request without authentication, it indicates a misconfiguration.

How It Helps

- Misconfigured API Gateway: Identifies endpoints that allow sensitive operations like creating or updating data without authentication.
- CORS Policy Exploitation: Confirms if unauthorized cross-origin POST requests can interact with the API, enabling attackers to exfiltrate or manipulate data.

3. Obtaining IAM Roles or Credentials

Objective: Leverage exposed AWS resources or misconfigurations to obtain IAM roles or credentials.
Steps

Search for credentials in public repositories:

```
truffleHog https://github.com/<repository-url>
```

Retrieve public files from S3:

```
aws s3 cp s3://<bucket-name>/config.env ./
```

Search for access keys:

```
grep -E 'AKIA[0-9A-Z]{16}' config.env
```

Probe API Gateway endpoints invoking Lambda functions:

```
curl -X GET https://<api-endpoint>
```

Privilege Escalation/ Lateral Movement

With valid IAM credentials obtained during the previous phase, the process shifts into a hybrid stage, blending black-box and white-box techniques. These credentials allow direct interaction with AWS APIs and services, enabling deeper exploration of the environment while maintaining an attacker's perspective.

Why This Phase is Critical

- Enhanced Visibility: IAM credentials provide access to AWS resources, including details unavailable during the initial information gathering phase.
- Privilege Escalation: Identifying misconfigurations or unused permissions may reveal opportunities for gaining higher privileges.
- Lateral Movement: Mapping additional IAM users, roles, or resources enables attackers to move within the AWS environment.
- Stealth: Understanding CloudTrail and logging configurations helps minimize detection risks during further actions.

1. Verify IAM Credentials

Why It Matters: Validating credentials ensures they are functional and provides insights into the associated identity (e.g., account, user, or role).
Steps

Verify credentials:

```
aws sts get-caller-identity
```

How It Helps:

- Identifies the account ID for targeting further resources.
- Reveals whether the credentials are associated with a user or role, guiding enumeration efforts.
- Confirms that the obtained credentials are functional.

2. Enumerate IAM Permissions

Why It Matters: Understanding the scope of permissions assigned to the IAM credentials reveals the potential for privilege escalation and resource access.

Steps

List attached policies:

```
aws iam list-attached-user-policies --user-name <user-name>
```

Simulate policy permissions:

```
aws iam simulate-custom-policy --policy-input-list file://policy.json --action-names s3:ListBucket
```

Enumerate effective permissions:

```
aws iam list-user-policies --user-name <user-name>
```

How It Helps

- Determines if the credentials have overly permissive privileges.
- Guides targeted exploitation efforts (e.g., accessing S3 buckets, escalating roles).
- The primary objective of this enumeration is to assess current permissions and explore potential avenues for leveraging them to achieve lateral or vertical privilege escalation.

3. Enumerate S3 Buckets

Why It Matters: S3 buckets often store sensitive data, including configuration files, logs, and secrets that could reveal additional attack vectors.

Steps

List accessible buckets:

```
aws s3 ls
```

Check permissions:

```
aws s3api get-bucket-acl --bucket <bucket-name>
```

Access bucket content:

```
aws s3 cp s3://<bucket-name>/<file-name> ./
```

How It Helps

- Uncovers stored credentials, secrets, and sensitive files.
- Exploits overly permissive bucket policies for further access.

4. Enumerate EC2 Instances

Why It Matters: Identifying EC2 instances helps map compute resources, discover their configurations, and potentially identify exposed services.

Steps

List running instances:

```
aws ec2 describe-instances
```

Analyze security groups:

```
aws ec2 describe-security-groups
```

Identify public IPs:

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].PublicIpAddress'
```

How It Helps:

- Identifies targets for further exploitation (e.g., metadata service).
- Reveals potential misconfigurations in security groups.

5. Enumerate IAM Roles

Why It Matters: Discovering other IAM roles can reveal opportunities for privilege escalation and lateral movement.

Steps

List all roles:

```
aws iam list-roles
```

Inspect role permissions:

```
aws iam get-role --role-name <role-name>
```

6. Enumerate CloudTrail

Why It Matters: Understanding logging configurations help avoid detection and identify potential blind spots in monitoring.

Steps

List CloudTrails:

```
aws cloudtrail describe-trails
```

Check logging status:

```
aws cloudtrail get-trail-status --name <trail-name>
```

How It Helps

- Identifies whether actions are being logged.
- Helps plan stealthier exploitation strategies.

7. Enumerate Additional Services

Why It Matters: AWS services beyond the primary ones (S3, EC2, IAM) can expose critical resources or misconfigurations.

Services

- RDS: Databases may expose sensitive information if misconfigured.
- Lambda: Environment variables may store secrets or tokens.
- SNS/SQS: Can expose notifications, tasks, or disrupt workflows.

Steps

RDS

```
aws rds describe-db-instances
```

Lambda

```
aws lambda list-functions
```

Tools Utilized

NAME	LINK	USAGE
aws cli	https//docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html#cliv2-linux-install	The AWS Command Line Interface (AWS CLI) is an open source tool that enables you to interact with AWS services.
S3Scanner	https://github.com/sa7mon/S3Scanner	A tool to find open S3 buckets in AWS.
Nmap	https://nmap.org/	A tool to perform network scanning.
gitdumper	https://github.com/arthaud/git-dumper	A tool to find exposed git repositories.
amass	https://github.com/owasp-amass/amass	A powerful tool for subdomain enumeration and mapping external assets.
sublist3r	https://github.com/aboul3la/Sublist3r	A fast and lightweight tool for enumerating subdomains using multiple search engines.
cURL	https://curl.se/download.html	A command-line tool for transferring data to and from a server, often used for interacting with web endpoints.
httprobe	https://github.com/tomnomnom/httprobe	A tool to quickly probe a list of URLs or domains for live HTTP and HTTPS endpoints.
shodan	https://www.shodan.io/	A search engine for identifying publicly accessible devices and services, including AWS resources, via IPs and domains.
censys	https://censys.com/	A search engine for discovering internet-connected devices and services through IP and TLS metadata analysis.
gobuster	https://github.com/OJ/gobuster	A directory and DNS brute-forcing tool used to discover hidden files, directories, and subdomains on web servers.
trufflehog	https://github.com/trufflesecurity/trufflehog	A tool for detecting and verifying secrets such as API keys, credentials, and tokens in source code repositories, filesystems, and cloud storage like S3 and GCS.

Certifications

NAME	DESCRIPTION
AWS Red Team Apprentice (ARTA)	The ARTA course by hacktricks, it teaches AWS security with black-box and white-box methods, service exploitation, and stealth techniques through videos and labs.
AWS Red Team Expert	The ARTE (Next Step to ARTA) course offers comprehensive AWS security training, covering basics to advanced services, white-box and black-box methodologies, detection bypass techniques, and hands-on labs to prepare for the htARTE certification.
Offensive AWS Security Professional	The Breaching AWS course equips you to perform enumeration, identify misconfigurations, exploit access gaps, simulate phishing, and conduct AWS penetration tests and cloud security audits.

Conclusion

Securing AWS environments requires a thorough understanding of common misconfigurations and weaknesses that attackers exploit. This guide has outlined a structured approach to AWS black-box penetration testing, covering key phases such as information gathering, identifying misconfigurations, exploiting vulnerabilities, privilege escalation, and data exfiltration. By leveraging publicly accessible tools and techniques, testers can simulate real-world attack scenarios and help organizations strengthen their cloud security posture.

However, it is important to recognize that every AWS environment is unique, with configurations tailored to specific organizational needs. As a result, the vulnerabilities and misconfigurations found in one environment may not necessarily exist in another. For further learning and hands-on practice, refer to the provided certifications, references, and lab resources to deepen your knowledge and stay updated with evolving AWS security trends.

SECURITY INNOVATION
A BUREAU VERITAS COMPANY

About Security Innovation/ Bureau Veritas

Security Innovation is a leader in software security, providing comprehensive assessment solutions to secure software from design to deployment, across all environments, including web, cloud, IoT, and mobile. Leveraging decades of expertise and as part of Bureau Veritas, a global leader in Testing, Inspection, and Certification, we seamlessly integrate world-class security into development processes, safeguarding the way companies build and deliver products.

Security Innovation is a Bureau Veritas company. Bureau Veritas (BV) is a publicly listed company specialized in testing, inspection and certification. BV was founded in 1828, has over 80,000 employees and is active in 140 countries.

Other Resources

- https://cloud.hacktricks.xyz/pentesting-cloud/aws-security
- https://github.com/kh4sh3i/cloud-penetration-testing
- https://github.com/CyberSecurityUP/Awesome-Cloud-PenTest
- https://github.com/lutzenfried/OffensiveCloud/blob/main/AWS/AWS%20Pentest%20Cloud%20-%20Resources.md
- https://github.com/redskycyber/Cloud-Security/blob/main/AWS-Security-Pentesting-Resources.md

BUREAU VERITAS

Interested?

Contact us today to start raising your cyber resilience.

sisales@securityinnovation.com
+1 877 839 7598
securityinnovation.com