**AWS Lambda Web Adapter HTTP/2 Header Sanitization**

**🧩 Overview**

This project modifies the AWS Lambda Web Adapter to ensure proper HTTP/2 compatibility by sanitizing HTTP/1.1 headers that are incompatible with HTTP/2. It solves a real-world issue where Application Load Balancers (ALBs) + Lambda does not automatically remove disallowed HTTP headers (like Connection and Keep-Alive) from responses when serving over HTTP/2.

**🔥 The Problem**

When using Lambda functions behind an ALB with HTTP/2 enabled:

1. Lambda functions with web frameworks like Next.js, Express, Flask, etc., often automatically include HTTP/1.1 headers in responses
2. These headers (like Connection: keep-alive) are explicitly prohibited in HTTP/2 per RFC 7540
3. Unlike with EC2/IP-based targets, ALBs do **not** sanitize these headers from Lambda responses
4. This causes HTTP/2 protocol errors and broken responses when using ALB + HTTP/2 + Lambda

**✅ The Solution**

Our solution adds header sanitization in two ways:

1. **Lambda Layer Approach**: Modify the AWS Lambda Web Adapter to sanitize prohibited headers
2. **Python Wrapper Approach**: Use a Python function wrapper that sanitizes headers before returning

The solution is non-invasive and requires no changes to your application code.

**🧪 Proof of Concept Results**

| **Target Type** | **Has HTTP/2-Forbidden Headers** | **Works with HTTP/2** | **Notes** |
| --- | --- | --- | --- |
| EC2/IP-based | Yes (initially) | ✅ | ALB automatically strips prohibited headers |
| Vanilla Lambda | Yes | ❌ | Fails with HTTP/2 PROTOCOL\_ERROR |
| Lambda with our solution | No | ✅ | Headers properly sanitized |

**🚀 Getting Started**

**Prerequisites**

* AWS CLI installed and configured
* Go 1.18+ (for building the Lambda Web Adapter)
* Python 3.8+ (for Lambda function testing)
* curl with HTTP/2 support (for testing)

**Quick Start**

1. Clone this repository:
2. git clone https://github.com/yourusername/aws-lambda-http2-headers.git
3. cd aws-lambda-http2-headers
4. Set up the development environment:
5. # Use the appropriate script for your platform
6. ./1-local-adapter-setup.sh # Linux/macOS
7. ./1-local-adapter-setup.ps1 # Windows
8. Build the custom Lambda Layer:
9. # Use the appropriate script for your platform
10. ./2-build-layer-zip.sh # Linux/macOS
11. ./2-build-layer-zip.ps1 # Windows
12. Publish the Layer to AWS Lambda:
13. ./5-publish-layer.sh # The ARN will be saved to layer-arn.txt
14. Apply the Layer to your Lambda function and add the environment variable:
15. AWS\_LAMBDA\_EXEC\_WRAPPER: /opt/extensions/bootstrap

**🔬 Testing**

This repository includes several scripts for testing the solution:

**Local Testing**

# Test the adapter locally

./3-test-local-adapter.sh

# Test with a Flask application to verify header sanitization

./4-test-adapter-with-flask.sh

**AWS Testing**

# Deploy the ALB + Lambda + EC2 testing stack

aws cloudformation deploy --template-file 5-alb-lambda-http2-header-sanitization-test.yaml --stack-name http2-headers-test --capabilities CAPABILITY\_IAM

# Test the deployed solution

./6-alb-test-http2-sanitization.sh your-alb-dns-name.region.elb.amazonaws.com

**🔍 Implementation Details**

**Disallowed HTTP/2 Headers**

The following headers are prohibited in HTTP/2 per RFC 7540 section 8.1.2.2:

* connection
* keep-alive
* proxy-connection
* transfer-encoding
* upgrade

Our implementation removes these headers from Lambda responses before they are sent back through the ALB.

**Lambda Layer vs. Python Wrapper Approach**

We provide two implementation options:

1. **Lambda Layer (Go)**: Modifies the Lambda Web Adapter to strip headers
   * Advantages: Works with any runtime, no application changes needed
   * Limitations: Requires adding a layer and environment variable
2. **Python Wrapper (Python)**: Sanitizes headers in the Lambda handler
   * Advantages: No additional layer needed, simpler to understand
   * Limitations: Runtime-specific, requires wrapping response logic

def sanitize\_http2\_headers(response):

"""Sanitize HTTP/2 disallowed headers"""

# List of disallowed headers in HTTP/2

disallowed\_headers = [

"connection",

"keep-alive",

"proxy-connection",

"transfer-encoding",

"upgrade"

]

# Remove disallowed headers (case-insensitive)

if "headers" in response and response["headers"]:

sanitized\_headers = {}

for header\_name, header\_value in response["headers"].items():

if header\_name.lower() not in disallowed\_headers:

sanitized\_headers[header\_name] = header\_value

# Replace headers with sanitized version

response["headers"] = sanitized\_headers

return response

def handler(event, context):

# Original handler logic

response = {

"statusCode": 200,

"headers": {

"Content-Type": "text/plain",

"Connection": "keep-alive",

"Keep-Alive": "timeout=72"

},

"body": "Your response content here"

}

# Apply sanitization before returning

return sanitize\_http2\_headers(response)

**📦 Project Structure**

aws-lambda-http2-headers/

├── 1-local-adapter-setup.sh # Setup script (Linux/macOS)

├── 1-local-adapter-setup.ps1 # Setup script (Windows)

├── 2-build-layer-zip.sh # Build Lambda Layer script (Linux/macOS)

├── 2-build-layer-zip.ps1 # Build Lambda Layer script (Windows)

├── 3-test-local-adapter.sh # Local adapter test script (Linux/macOS)

├── 3-test-local-adapter.ps1 # Local adapter test script (Windows)

├── 4-test-adapter-with-flask.sh # Flask integration test script (Linux/macOS)

├── 4-test-adapter-with-flask.ps1 # Flask integration test script (Windows)

├── 5-publish-layer.sh # AWS Lambda Layer publishing script

├── 6-alb-test-http2-sanitization.sh # Test script for deployed solution

├── aws-lambda-web-adapter/ # Forked and modified web adapter code

│ ├── src/

│ │ ├── lib.rs # Added sanitization logic

│ │ └── adapter/hyper.rs # Modified to call sanitization

│ ├── bin/ # Compiled binaries

│ └── custom-lambda-layer/ # Generated Lambda Layer structure

├── ec2.py # Sample Flask app for testing

├── lambda.py # Sample Lambda function for testing

└── doc/ # Documentation

└── architecture.png # Architecture diagram

**🔄 ALB Behavior Comparison**

This project reveals an interesting inconsistency in how AWS ALB handles headers:

* For **EC2/IP-based targets**: ALB automatically strips HTTP/2-incompatible headers
* For **Lambda targets**: ALB does not strip these headers, causing HTTP/2 errors

Our solution brings Lambda behavior in line with EC2 behavior, ensuring HTTP/2 compatibility.

**🛡️ Security Considerations**

This solution:

* Does not modify or intercept request body content
* Only filters response headers based on a strict allowlist of known problematic headers
* Uses standard AWS Lambda Layers for deployment
* Runs with the same permissions as your Lambda function

**🤝 Contributing**

Contributions are welcome! If you find bugs or have suggestions:

1. Open an issue
2. Submit a pull request with your changes
3. Include tests for any new functionality

**🔮 Future Work**

Potential improvements:

* Support for additional frameworks and runtimes
* Automated testing with GitHub Actions
* CloudFormation/CDK templates for easier deployment
* Performance benchmarking and optimization

**📄 License**

This project is licensed under the MIT License - see the [LICENSE](https://claude.ai/chat/LICENSE) file for details.

**🙏 Acknowledgments**

* AWS Lambda Web Adapter team for the original adapter code
* HTTP/2 RFC 7540 for header specifications
* AWS documentation on ALB + Lambda integration