Sample Code

PHP

This is a comprehensive example of a middleware to validate the Webhook source, with a verifier class that checks the Timestamp and the Signature.

```
<?php declare(strict_types=1);

namespace Nrsdb\Middleware;

use Laminas\Diactoros\Response\JsonResponse;
use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;
use Psr\Http\Server\MiddlewareInterface;
use Psr\Http\Server\RequestHandlerInterface;
use Nrsdb\Models\Audit;
use Nrsdb\Utils\Webhooks\WebhookTimestamp;

/**
 * validation middleware for webhook requests
 */
class WebhookAuthMiddleware implements MiddlewareInterface
{
    /**
     * check that the webhook request is from the expected source
     * @param ServerRequestInterface $request
     * @param RequestHandlerInterface $handler
     * @return ResponseInterface
     */
    public function process(ServerRequestInterface $request, RequestHandlerInterface
$handler): ResponseInterface
    {
        // Get raw body for signature verification
        $body = (string)$request->getBody();
```

```
        // Get headers
        $signature = $request->getHeaderLine('X-Webhook-Signature');
        $timestamp = (int)$request->getHeaderLine('X-Webhook-Timestamp');
        $webhookId = $request->getHeaderLine('X-Webhook-ID');

        // Validate required headers
        if (empty($signature) || empty($timestamp) || empty($webhookId)) {
            return new JsonResponse([
                'error' => 'Missing required webhook headers'
            ], 400);
        }

        // Get webhook secret (based on your auth system)
        $secret = $this->getWebhookSecret($request);
        if (!$secret) {
            return new JsonResponse([
                'error' => 'Invalid webhook configuration'
            ], 401);
        }

        // Verify signature
        $verification = WebhookTimestamp::verifyWebhookRequest(
            $body,
            $signature,
            $timestamp,
            $secret
        );

        if (!$verification['valid']) {
            return new JsonResponse([
                'error' => $verification['error']
            ], 401);
        }

        return $handler->handle($request->withAttribute('webhookId', $webhookId));
    }

    /**
     * Get webhook secret (implement based on your auth system)
     */
```

```
    private function getWebhookSecret(ServerRequestInterface $request): string
    {
        return $_ENV['WEBHOOK_SECRET'];
    }
}
```

```
<?php declare(strict-types=1);

namespace Nrsdb\Utils\Webhooks;

class WebhookTimestamp
{
    /**
     * Verify webhook with timestamp validation
     */
    public static function verifyWebhookRequest(
        string $payload,
        string $signature,
        int $timestamp,
        string $secret
    ): array {
        // Check timestamp to prevent replay attacks
        if (!self::isTimestampValid($timestamp)) {
            return [
                'valid' => false,
                'error' => 'Timestamp too old or in future (possible replay attack)'
            ];
        }

        // Verify signature
        $isValid = self::verifyWebhookWithTimestamp(
            $payload,
            $signature,
            $secret,
            $timestamp
        );

        if (!$isValid) {
            return [
                'valid' => false,
```

```
                'error' => 'Invalid signature'
            ];
        }

        return ['valid' => true];
    }

    /**
     * Validate that timestamp is recent (default within 5 minutes)
     */
    private static function isTimestampValid(int $timestamp, int $toleranceSeconds = 300):
bool
    {
        $now = time();
        $difference = abs($now - $timestamp);

        return $difference <= $toleranceSeconds;
    }

    /**
     * Verify signature with timestamp (for recipients)
     */
    public static function verifyWebhookWithTimestamp(
        string $payload,
        string $signature,
        string $secret,
        int $timestamp
    ): bool {
        $signedData = $payload . $timestamp;
        $expectedSignature = hash_hmac('sha256', $signedData, $secret);

        return hash_equals($expectedSignature, $signature);
    }
}
```

Java

```
public class WebhookSignatureValidator {
    public boolean validateSignature(String body,
                                     String signature,
                                     String secretKey,
                                     String timestamp) throws NoSuchAlgorithmException,
InvalidKeyException {

        // Create the message to be signed by concatenating the message body and the timestamp
        var message = body + timestamp;

        // Initialize the MAC routine with the secret key
        var mac = Mac.getInstance("HmacSHA256");
        var secret = new SecretKeySpec(secretKey.getBytes(StandardCharsets.UTF_8),
"HmacSHA256");
        mac.init(secret);

        // Create a digest of the bytes in the message
        byte[] digest = mac.doFinal(message.getBytes());

        // Print the digest as lower-case hex characters
        var enc = DatatypeConverter.printHexBinary(digest).toLowerCase();

        return signature.equals(enc);
    }
}
```

Revision #3
Created 23 August 2026 09:11:54 by Simon Gilhooly
Updated 23 August 2026 09:56:45 by Simon Gilhooly