API call restrictions

The API key method is simplest for internal scripts, while JWT is better if you need token expiration and more security.

1. API Key Authentication (Simple & Effective

```
<?php
// api/config.php
define('VALID_API_KEYS', [
  'abc123xyz789' => 'analytics_script',
  'def456uvw012' => 'reporting_script',
  'ghi789rst345' => 'backup_script'
]);

// api/auth.php
function validateApiKey() {
  $headers = getallheaders();
  $apiKey = $headers['X-API-Key'] ?? $_GET['api_key'] ?? null;

  if (!$apiKey || !array_key_exists($apiKey, VALID_API_KEYS)) {
  http_response_code(401);
  echo json_encode(['error' => 'Unauthorized - Invalid API key']);
  exit;
  }

  return VALID_API_KEYS[$apiKey];
}

// api/endpoint.php
require_once 'config.php';
require_once 'auth.php';

$scriptName = validateApiKey();

// Your API logic here
echo json_encode([
  'status' => 'success',
  'data' => 'Your data here',
  'accessed_by' => $scriptName
]);
```

Client script usage:

```
<?php
$apiKey = 'abc123xyz789';
$url = 'https://your-domain.com/api/endpoint.php';
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, [
    'X-API-Key: ' . $apiKey
]);
$response = curl_exec($ch);
curl_close($ch);
$data = json_decode($response, true);
print_r($data);
```

2. JWT Token Authentication (More Secure)

```
<?php
// Install: composer require firebase/php-jwt
use Firebase\JWT\JWT;
use Firebase\JWT\Key;
// api/jwt_auth.php
class JWTAuth {
    private static $secret = 'your-secret-key-change-
this'; // Store in environment variable

    public static function generateToken($clientId, $expiresIn = 3600) {
        $payload = [
            'iss' => 'your-domain.com',
            'aud' => 'your-api',
            'iat' => time(),
            'exp' => time() + $expiresIn,
            'client_id' => $clientId
        ];

        return JWT::encode($payload, self::$secret, 'HS256');
    }
    public static function validateToken() {
        $headers = getallheaders();
        $authHeader = $headers['Authorization'] ?? '';

        if (!preg_match('/Bearer\s+(.*)$/i', $authHeader, $matches)) {
            http_response_code(401);
            echo json_encode(['error' => 'No token provided']);
            exit;
        }

        $token = $matches[1];

        try {
            $decoded = JWT::decode($token, new Key(self::$secret, 'HS256'));
            return $decoded->client_id;
        } catch (Exception $e) {
            http_response_code(401);
            echo json_encode(['error' => 'Invalid token: ' . $e->getMessage()]);
            exit;
        }
    }
}
// api/get_token.php - Script calls this first to get token
require_once 'jwt_auth.php';
$clientId = $_POST['client_id'] ?? '';
$clientSecret = $_POST['client_secret'] ?? '';
// Validate client credentials
$validClients = [
    'script1' => 'secret123',
    'script2' => 'secret456'
];
if (!isset($validClients[$clientId]) || $validClients[$clientId] !== $clientSecret)
{
    http_response_code(401);
    echo json_encode(['error' => 'Invalid credentials']);
    exit;
}
```

```
$token = JWTAuth::generateToken($clientId);
echo json_encode([
    'access_token' => $token,
    'token_type' => 'Bearer',
    'expires_in' => 3600
]);
// api/protected_endpoint.php
require_once 'jwt_auth.php';
$clientId = JWTAuth::validateToken();
echo json_encode([
    'status' => 'success',
    'data' => 'Your protected data',
    'client' => $clientId
]);
```

3. IP Whitelisting

```
<?php
// api/ip_whitelist.php
function validateIP() {
    $allowedIPs = [
        '192.168.1.100',
        '10.0.0.5',
        '203.0.113.45'
    ];

    $clientIP = $_SERVER['REMOTE_ADDR'];

    // If behind proxy, get real IP
    if (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
        $clientIP = explode(',', $_SERVER['HTTP_X_FORWARDED_FOR'])[0];
    }

    if (!in_array($clientIP, $allowedIPs)) {
        http_response_code(403);
        echo json_encode(['error' => 'Access denied from IP: ' . $clientIP]);
        exit;
    }
}
// api/endpoint.php
require_once 'ip_whitelist.php';
validateIP();
// Your API logic
```

4. Combined Approach (Recommended)

```
<?php
// api/secure_auth.php
class SecureAuth {
    private static $apiKeys = [
        'abc123xyz789' => [
            'name' => 'analytics_script',
            'allowed_ips' => ['192.168.1.100'],
            'rate_limit' => 100 // requests per hour
        ]
```

```
];

public static function validate() {
    // Check IP
    self::validateIP();

    // Check API Key
    $apiKey = self::getApiKey();

    // Check rate limit
    self::checkRateLimit($apiKey);

    return self::$apiKeys[$apiKey]['name'];
}

private static function getApiKey() {
    $headers = getallheaders();
    $apiKey = $headers['X-API-Key'] ?? null;

    if (!$apiKey || !isset(self::$apiKeys[$apiKey])) {
        http_response_code(401);
        echo json_encode(['error' => 'Invalid API key']);
        exit;
    }

    return $apiKey;
}

private static function validateIP() {
    $clientIP = $_SERVER['REMOTE_ADDR'];
    $apiKey = getallheaders()['X-API-Key'] ?? null;

    if ($apiKey && isset(self::$apiKeys[$apiKey])) {
        $allowedIPs = self::$apiKeys[$apiKey]['allowed_ips'];
        if (!in_array($clientIP, $allowedIPs)) {
            http_response_code(403);
            echo json_encode(['error' => 'IP not authorized']);
            exit;
        }
    }
}

private static function checkRateLimit($apiKey) {
    session_start();
    $key = 'rate_limit_' . $apiKey;
    $limit = self::$apiKeys[$apiKey]['rate_limit'];
    $window = 3600; // 1 hour

    if (!isset($_SESSION[$key])) {
        $_SESSION[$key] = ['count' => 0, 'start' => time()];
    }

    $data = $_SESSION[$key];

    // Reset if window expired
    if (time() - $data['start'] > $window) {
        $_SESSION[$key] = ['count' => 1, 'start' => time()];
        return;
    }
```

```
        // Check limit
        if ($data['count'] >= $limit) {
            http_response_code(429);
            echo json_encode(['error' => 'Rate limit exceeded']);
            exit;
        }

        $_SESSION[$key]['count']++;
    }
}
```

5. Store Keys Securely

```
<?php
// .env file (never commit to git)
// API_KEY_SCRIPT1=abc123xyz789
// API_KEY_SCRIPT2=def456uvw012

// config.php
function getApiKeys() {
  return [
  getenv('API_KEY_SCRIPT1') => 'analytics_script',
  getenv('API_KEY_SCRIPT2') => 'reporting_script'
  ];
}
```

6. Complete Example with .htaccess Protection

```
# .htaccess - Force HTTPS
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

# Deny access to sensitive files
<Files "config.php">
  Require all denied
</Files>
```

Unique solution ID: #1203
Author: n/a
Last update: 2026-01-09 21:28