Magic Login – Security Guide

Magic Login Pro · https://handyplugins.co/docs/magic-login-security/

Magic Login is designed with both convenience and security in mind. This guide explains how the plugin handles key areas such as encryption, token security, expiration, salts, and recommended hardening practices for production sites.

Token Security

Magic Login generates **single-use**, **time-limited** login tokens by default. You can adjust both the token validity window and TTL from the settings page. We recommend configuring these options carefully and enabling the IP checking feature where appropriate.

Each token is:

- Generated with a cryptographically secure random value
- Hashed using `hash_hmac('sha256', $token, wp_salt())` before being stored
- Always stored together with a **hashed version of the client's IP address**
- Stored in user meta and associated with a timestamp

The raw token is only shown to the user (in the link or code); the plugin only stores a hashed version for verification.

Token Formats by Channel

Depending on how you deliver the login, Magic Login uses different token formats:

- **Email link (default):** A long random token suitable for links
- **SMS link:** A shortened token to keep the login URL size reasonable (many SMS providers have ~300 character limits)
- **SMS code:** A 6-digit numeric PIN that users enter manually
- **Email code:** A 10-character, uppercase alphanumeric code (excluding ambiguous characters)

This keeps tokens usable in different channels while still being hard to guess.

Encryption & Decryption

Magic Login resolves its encryption key and salt in the following order:

1) Custom Plugin Constants (Recommended)

Define these in `wp-config.php`:

```
define( 'MAGIC_LOGIN_ENCRYPTION_KEY',  'your-long-random-secret-key' );
define( 'MAGIC_LOGIN_ENCRYPTION_SALT', 'your-long-random-secret-salt' );
```

If these are defined, they take priority.

2) WordPress Authentication Constants

If the plugin-specific constants are not defined, Magic Login falls back to:

```
LOGGED_IN_KEY
LOGGED_IN_SALT
```

This is safer than storing keys inside the database because WordPress expects these values to be defined in the configuration file.

3) Fallback Values (NOT SAFE FOR PRODUCTION)

If none of the above are defined, the plugin **must** fall back to:

```
this-is-not-a-secret-key
this-is-not-a-secret-salt
```

These are **placeholders**, intended for local development environments where encryption is still needed for functionality.

Why You Should Define Your Own Keys/Salts

Storing secrets inside the database is insecure.
Defining them in `wp-config.php` ensures:

- They are **not stored in the same place as encrypted data**
- They are harder for attackers to obtain
- They are consistent across deployments (if you're using version control)
- They are compatible with multi-server environments

If you run a cluster/multi-server setup, make sure all nodes share the same values.

How to Generate Secure Keys

You can generate strong keys using:

Option 1 — WordPress Secret Key Generator

https://api.wordpress.org/secret-key/1.1/salt

Use any of the generated values for the plugin constants.

Option 2 — Command Line

```
openssl rand -base64 48
```

Example Recommended Setup (wp-config.php)

```
define( 'MAGIC_LOGIN_ENCRYPTION_KEY',  'hTDe2J39fS8H...random...94ksE' );
define( 'MAGIC_LOGIN_ENCRYPTION_SALT', 'pppew83hdns...random...W29saa' );
```

API Rate Limit

Magic Login PRO includes built-in **API rate limiting** starting from version **2.6.2**. If you plan to expose or consume Magic Login's API endpoints, we strongly recommend enabling rate limiting.

Why Rate Limiting Matters

Without rate limiting, an attacker could attempt:

- **Enumeration attacks** (e.g., checking which email addresses or phone numbers exist)

- **Automated token generation abuse**
- **Flooding API endpoints** to degrade performance

Rate limiting helps mitigate these risks by restricting how many requests a specific client (e.g., IP address) can make within a certain time window.

How It Works

When rate limiting is enabled:

- API requests are tracked using a hashed version of the client's IP address for privacy
- The plugin counts the number of attempts within the configured interval
- Once the threshold is reached, further requests are **temporarily blocked**
- The block resets automatically after the cooldown period

No raw IP addresses are stored — only hashed identifiers, ensuring GDPR-friendly logging.

Server-Level Security Expectations

Magic Login does **not** configure HTTP security headers.
Headers such as:

- `Content-Security-Policy`
- `X-Frame-Options`
- `Strict-Transport-Security`

should be handled at the **server level** (nginx, Apache, CDN, Cloudflare).

This is the recommended approach because:

- These policies must apply site-wide
- They vary between hosting setups
- They often require per-server customization
- Plugin-based header injection can conflict with your host's security rules