

SECURITY AUDIT REPORT

acashcat.com Meme Token

SPL Token Deployment Script (Solana / Node.js)

Function-by-Function Security Review

Report date: October 10, 2026

Prepared by: Claude (Anthropic) - AI-assisted security review

Table of Contents

1. Executive Summary 3

2. Scope and Methodology 4

3. Findings Overview 5

4. Security Checklist 6

5. Function-by-Function Audit 7

6. Detailed Findings 10

F-01 Mutable metadata and retained update authority . . . 10

F-02 Non-atomic multi-transaction deployment . . . 10

F-03 Errors swallowed; process exits with success code . . . 10

F-04 Entire supply minted to a single deployer wallet . . . 10

F-05 Silent default to mainnet-beta . . . 11

F-06 Private key handled as plain environment string . . . 11

F-07 Freeze authority created, then revoked later . . . 11

F-08 Placeholder values and third-party metadata host . . . 11

F-09 No input validation . . . 11

F-10 Floating-point exponent in supply calculation . . . 12

F-11 No post-deployment on-chain verification . . . 12

F-12 No balance pre-check, priority fee or retry . . . 12

I-01 Token standard naming: SPL vs SLP . . . 12

I-02 Fixed 4-second sleep is redundant . . . 12

I-03 Dependency version and API compatibility . . . 12

7. Positive Observations 13

8. Recommended Remediation Order 13

Appendix A - Bangla to English Translation Log 14

Appendix B - Disclaimer 14

1. Executive Summary

This report presents a security review of the token deployment script supplied for the **acashcat.com meme token**. The script is a Node.js program that creates an SPL token on Solana, registers its Metaplex metadata, mints the full supply to the deployer wallet and then revokes the mint and freeze authorities. Every function and execution step was reviewed, and all Bangla comments were translated to English before the review (Appendix A). Program logic was not changed.

Item	Details
Project	acashcat.com Meme Token (placeholder name in code: "MY TOKEN NAME", symbol "TOKEN")
Artifact	Single file, 163 lines: token deployment script (JavaScript, Node.js)
Token standard	SPL Token (Solana Program Library, legacy Token program), 9 decimals, supply 1,000,000,000
Target network	Solana mainnet-beta (default if SOLANA_RPC is not set)
Review type	Manual static review, threat modelling and best-practice comparison. Syntax check: passed. Dependencies were not installed and the script was not executed on-chain.
Overall risk	**MEDIUM** - suitable for mainnet only after the High and Medium findings are addressed

Findings Summary

Critical	High	Medium	Low	Informational	Total
0	1	5	6	3	15

Key Takeaways

- **Good:** the script revokes mint authority and freeze authority after minting, so the supply is fixed and holders cannot be frozen once all steps complete. SDK call signatures are used correctly and no secret is printed.
- **Main concern (High):** token metadata is created as **mutable** and the deployer keeps the update authority, so name, symbol and image link can be changed after launch. This contradicts the "Rug-Proof" label in the script comments.
- **Operational concerns (Medium):** deployment is split across several transactions with no rollback, errors are swallowed (exit code 0), the full supply sits in one wallet, the network defaults to mainnet silently, and the private key is handled as a plain environment string.
- Revoking authorities does not by itself prevent a price collapse: a single wallet holding 100% of supply can still sell. Distribution and liquidity-lock practices matter as much as these on-chain flags.

2. Scope and Methodology

2.1 In Scope

File	Language	Lines	Description
deploy_token (supplied script; English-translated copy delivered with this report)	JavaScript (Node.js)	163	Creates SPL mint, Metaplex metadata, mints supply, revokes mint and freeze authorities

2.2 Out of Scope

- On-chain programs invoked by the script: the SPL Token program and the Metaplex Token Metadata program (these are separate, widely used programs and were not audited here).
- Third-party libraries (@solana/web3.js, @solana/spl-token, mpl-token-metadata, bs58, dotenv) and their supply chain.
- Wallet and device security of the deployer, the metadata JSON file and image hosting, the website, liquidity pool creation, tokenomics and legal or regulatory matters.

2.3 Methodology

- **Translation:** Bangla comments converted to English first; no logic changed (Appendix A).
- **Manual line-by-line review** of every function and execution step (Section 5).
- **Threat modelling** of the deployer, authorities, metadata, key handling and failure modes.
- **Best-practice comparison** against common SPL token launch practice (authority hygiene, atomicity, validation, verification).
- **Syntax check** with Node.js (passed). Dynamic testing on devnet was not performed.

2.4 Severity Classification

Severity	Meaning
Critical	Directly causes loss of funds or total loss of control; must be fixed before any use.
High	Significantly undermines the security or trust guarantees of the token; should be fixed before launch.
Medium	Increases risk under realistic conditions or contradicts best practice; fix recommended before mainnet.
Low	Minor weakness or robustness issue with limited impact.
Informational	No direct risk; clarity, maintainability or good-practice suggestions.

3. Findings Overview

ID	Title	Severity	Location	Status
F-01	Mutable metadata and retained update authority	**High**	L87, L101	Open
F-02	Non-atomic multi-transaction deployment	**Medium**	L55-L158	Open
F-03	Errors swallowed; process exits with success code	**Medium**	L163	Open
F-04	Entire supply minted to a single deployer wallet	**Medium**	L112-L132	Open
F-05	Silent default to mainnet-beta	**Medium**	L33, L50	Open
F-06	Private key handled as plain environment string	**Medium**	L34, L46-L51	Open
F-07	Freeze authority created, then revoked later	**Low**	L56-L65, L147-L158	Open
F-08	Placeholder values and third-party metadata host	**Low**	L26-L27, L30	Open
F-09	No input validation (length limits, u64 overflow)	**Low**	L26-L30, L128	Open
F-10	Floating-point exponent in supply calculation	**Low**	L128	Open
F-11	No post-deployment on-chain verification	**Low**	L160	Open
F-12	No balance pre-check, priority fee or retry	**Low**	L50, L110	Open
I-01	Token standard naming: SPL vs SLP	**Informational**	General	Open
I-02	Fixed 4-second sleep is redundant	**Informational**	L66-L68	Open
I-03	Dependency version and API compatibility	**Informational**	L1-L22	Open

Line numbers refer to the English-translated copy of the script delivered with this report.

4. Security Checklist

Check	Result	Reference / note
Mint authority revoked after minting	Pass	Step 5 (L134-L145); see F-02 for ordering risk
Freeze authority revoked	Pass	Step 6 (L147-L158); see F-07
Supply fixed and defined in one constant	Pass	TOTAL_SUPPLY, minted once
Metadata update authority renounced / immutable	Fail	F-01
Deployment is atomic or recoverable	Fail	F-02, F-03
Supply distribution / decentralisation	Warning	F-04 (100% to one wallet)
Network selection is explicit	Fail	F-05
Secrets not hard-coded or logged	Pass	Read from .env, never printed; see F-06 for handling
Inputs validated	Fail	F-09, F-10
On-chain state verified after deployment	Fail	F-11
SDK argument order and types	Pass	createMint, mintTo, setAuthority, ATA calls are correct
Metaplex program ID constant	Pass	Matches the standard Token Metadata program address
Dependencies pinned and reviewed	Warning	I-03

5. Function-by-Function Audit

Each function or execution step of the script is reviewed below. "Result" shows whether the unit is clean or which findings apply.

FA-1 Imports and dependencies

Location	L1-L22
Purpose	Loads Solana web3, SPL Token, Metaplex Token Metadata, bs58 and dotenv.

- All imported symbols are used; there are no unused imports.
- `createCreateMetadataAccountV3Instruction` exists in mpl-token-metadata v2.x; newer major versions use a different (Umi) API, so the version must be pinned (I-03).
- `require('dotenv').config()` loads secrets from .env; the file must never be committed (F-06).

Result: Pass with notes - I-03

FA-2 Configuration constants

Location	L24-L31
Purpose	Defines token name, symbol, total supply, decimals and metadata URI.

- Values are still placeholders ("MY TOKEN NAME", "TOKEN", "YOUR_JSON_LINK"). Because metadata is later written on-chain, deploying placeholders would publish them (F-08).
- No length or range validation: Metaplex limits name to 32, symbol to 10 and URI to 200 bytes; supply times 10^decimals must fit in u64 (F-09).
- Current numbers are valid: 1,000,000,000 x 10^9 = 1e18, below the u64 maximum of about 1.84e19.
- The metadata URI points to a third-party JSON host (F-08).

```
const TOTAL_SUPPLY = 1000000000;
const DECIMALS = 9;
const METADATA_URI = "https://jsonkeeper.com/b/YOUR_JSON_LINK";
```

Result: Findings F-08, F-09

FA-3 RPC endpoint and key configuration

Location	L33-L34
Purpose	Chooses the RPC URL and reads the payer private key from the environment.

- If SOLANA_RPC is unset the script silently targets mainnet-beta public RPC, which is rate-limited and has real-money consequences (F-05).
- The private key is a Base58 string in an environment variable; acceptable for testing but weak for a mainnet deployer (F-06).

```
const RPC_ENDPOINT = process.env.SOLANA_RPC || clusterApiUrl("mainnet-beta");
```

Result: Findings F-05, F-06

FA-4 TOKEN_METADATA_PROGRAM_ID

Location	L36-L38
Purpose	Declares the Metaplex Token Metadata program address used to derive the metadata PDA.

- The address is the standard, well-known Token Metadata program ID. Hard-coding a program ID is correct practice here.

Result: Pass

FA-5 sleep(ms)

Location	L40
Purpose	Promise-based delay helper.

- Functionally correct. Its only use is a fixed 4-second wait after minting, which is unnecessary because createMint already waited for finalized commitment (I-02).

Result: Pass - I-02

FA-6 main() - key loading and connection

Location	L42-L53
Purpose	Validates the key variable, decodes Base58, builds the Connection and Keypair.

- Fails fast with a clear message if the key is missing.
- Handles both CommonJS and default-export shapes of bs58 (good compatibility handling).
- `Keypair.fromSecretKey` throws on malformed keys, so no extra length check is required.
- No SOL balance check before starting; running out of SOL midway leaves a half-configured token (F-12).
- The wallet public address is logged, the secret is not (good).

Result: Findings F-06, F-12

FA-7 Step 1 - createMint()

Location	L55-L66
Purpose	Creates the SPL mint with the deployer as both mint authority and freeze authority.

- Argument order (connection, payer, mintAuthority, freezeAuthority, decimals, keypair, confirmOptions) is correct; `undefined` lets the SDK generate the mint keypair.
- Freeze authority is set to the payer and removed only at the end; passing null here would remove the exposure window completely (F-07).
- Uses finalized commitment, which is safe but slow.

```
const mint = await createMint(connection, payer,
  payer.publicKey,   // mint authority
  payer.publicKey,   // freeze authority  <-- could be null
  DECIMALS, undefined, { commitment: "finalized" });
```

Result: Finding F-07 (and F-02 for sequencing)

FA-8 Step 2 - Metadata PDA derivation

Location	L70-L78
Purpose	Derives the metadata account address from the seeds "metadata", program ID and mint.

- Seed layout matches the Token Metadata standard. Uses the synchronous, non-throwing findProgramAddressSync correctly (bump is not needed here).

Result: Pass

FA-9 Step 3 - Create metadata and send transaction

Location	L80-L110
Purpose	Builds CreateMetadataAccountV3 and sends it in its own transaction.

- Account list is complete for V3 (metadata, mint, mintAuthority, payer, updateAuthority, systemProgram).
- **isMutable: true** with **updateAuthority = payer** keeps the deployer able to rewrite name, symbol and URI forever (F-01).
- sellerFeeBasisPoints 0 and creators null are appropriate for a fungible token.
- Sent as a separate transaction from the mint creation and minting (F-02).

```
updateAuthority: payer.publicKey,
...
isMutable: true,   // metadata can be changed after launch
```

Result: Findings F-01, F-02

FA-10 Step 4 - Associated token account and mintTo()

Location	L112-L132
Purpose	Creates (or fetches) the deployer's token account and mints the entire supply to it.

- Argument order for getOrCreateAssociatedTokenAccount and mintTo is correct; amount is a BigInt as required.
- The whole supply goes to one wallet (F-04).
- `BigInt(10 ** DECIMALS)` relies on floating-point exponentiation; safe for 9 decimals but fragile (F-10).
- mintTo is the last privileged action; if the script crashes after it, mint authority is still active (F-02, F-03).

```
BigInt(TOTAL_SUPPLY) * BigInt(10 ** DECIMALS)   // prefer: 10n ** BigInt(DECIMALS)
```

Result: Findings F-02, F-04, F-10

FA-11 Step 5 - Revoke mint authority

Location	L134-L145
Purpose	Sets the mint authority to null, permanently fixing the supply.

- Correct use of setAuthority with AuthorityType.MintTokens and null. This is irreversible, which is the intended behaviour.
- Executes only after minting, which is the right order.

Result: Pass

FA-12 Step 6 - Revoke freeze authority

Location	L147-L158
Purpose	Sets the freeze authority to null.

- Correct and irreversible. Becomes redundant if freeze authority is null from creation (F-07).
- The code comments call this "Rug-Proof"; this overstates the guarantee while metadata is mutable and supply is concentrated (F-01, F-04).

Result: Pass - F-07

FA-13 Entry point and error handling

Location	L160-L163
Purpose	Prints success and runs main().

- `main().catch(console.error)` only logs the error; the process exits with code 0, so scripts or CI treat a failed deployment as success (F-03).
- No recovery information is saved (mint address is only printed), and no final state verification is performed (F-11).

```
main().catch(console.error);
```

Result: Findings F-03, F-11

6. Detailed Findings

F-01 Mutable metadata and retained update authority

High **Category:** Centralization / Trust **Location:** L87, L101 **Status:** Open

Description. The metadata account is created with isMutable set to true and the deployer wallet as update authority. The deployer can later change the token name, symbol and metadata URI (and therefore the image and description) at any time.
Impact. A compromised or malicious deployer key can rebrand the token, point it to phishing content or impersonate another project. Holders, explorers and rug-check tools treat mutable metadata as a risk flag, and it contradicts the "Rug-Proof" comment.
Recommendation. After testing on devnet and finalizing name, symbol and URI, create the metadata with isMutable false. If future edits are needed, keep it mutable only briefly, then renounce it and publish the transaction as proof.

```
isMutable: false,   // metadata becomes permanent
```

F-02 Non-atomic multi-transaction deployment

Medium **Category:** Operational / Logic **Location:** L55-L158 **Status:** Open

Description. Mint creation, metadata, token account creation, minting and both authority revocations are separate transactions executed one after another, each waiting for finalization.
Impact. If the network, RPC or script fails between steps, the token is left in a partial state: for example, supply minted while mint authority is still active. Retrying blindly can create a second, different mint.
Recommendation. Where transaction size allows, combine create-mint (with null freeze authority), metadata, ATA creation, mintTo and revoke-mint-authority into one transaction so they succeed or fail together. If it must stay split, persist the mint address to disk right after creation and add a resume routine that checks which steps are already done.

```
const tx = new Transaction().add(
  SystemProgram.createAccount({...}),
  createInitializeMintInstruction(mint, DECIMALS, payer, null),
  createMetadataIx,
  createAssociatedTokenAccountInstruction(...),
  createMintToInstruction(...),
  createSetAuthorityInstruction(mint, payer, AuthorityType.MintTokens, null));
```

F-03 Errors swallowed; process exits with success code

Medium **Category:** Error handling **Location:** L163 **Status:** Open

Description. main().catch(console.error) prints the error but does not set a failing exit code, and no state is saved.
Impact. A failed run can look successful to wrappers or CI, and the operator may assume authorities were revoked when they were not.
Recommendation. Exit non-zero on failure, log which step failed, and record progress.

```
main().catch((e) => {
  console.error("Deployment failed:", e);
  process.exit(1);
});
```

F-04 Entire supply minted to a single deployer wallet

Medium **Category:** Centralization **Location:** L112-L132 **Status:** Open

Description. All 1,000,000,000 tokens are minted to the deployer's token account.
Impact. Even with mint and freeze authority revoked, one wallet holding 100% of supply can sell and collapse the price. Fixed supply does not equal fair distribution.
Recommendation. Define and publish the allocation (liquidity, team with vesting or lock, community, marketing). Lock or burn LP tokens after creating the pool, and consider a multisig for any team wallet.

F-05 Silent default to mainnet-beta

Medium **Category:** Operational safety **Location:** L33, L50 **Status:** Open

Description. If SOLANA_RPC is missing, the script deploys to mainnet-beta using the public RPC with no confirmation prompt. A fresh typo or missing .env turns a test run into a real, irreversible deployment.
Impact. Accidental mainnet deployment, wasted SOL and permanent on-chain mistakes; the public endpoint is also rate-limited and can cause timeouts.
Recommendation. Require an explicit cluster setting, print cluster, wallet, balance and token parameters, and ask for confirmation before mainnet. Rehearse on devnet first and use a dedicated RPC provider for mainnet.

```
const CLUSTER = process.env.SOLANA_CLUSTER;   // 'devnet' | 'mainnet-beta'
if (!CLUSTER) throw new Error('Set SOLANA_CLUSTER explicitly');
```

F-06 Private key handled as plain environment string

Medium **Category:** Key management **Location:** L34, L46-L51 **Status:** Open

Description. The deployer secret key is stored as a Base58 string in .env and held in process memory. No secret is logged, which is good, but the file itself is a single point of compromise.
Impact. Leaked .env (git commit, backup, malware, screen share) gives full control of the deployer wallet and any assets in it.
Recommendation. Use a brand-new wallet funded with only the SOL needed for fees and rent. Keep .env in .gitignore, restrict file permissions, never paste the key into chats or tools, and move assets to a secure wallet (hardware or multisig) right after deployment, then retire the deployer key.

F-07 Freeze authority created, then revoked later

Low **Category:** Authority hygiene **Location:** L56-L65, L147-L158 **Status:** Open

Description. The mint is initialised with the payer as freeze authority and removed in Step 6.
Impact. Between creation and revocation (and permanently if the script crashes) the deployer could freeze holder accounts.
Recommendation. Pass null as the freeze authority in createMint and delete Step 6.

```
createMint(connection, payer, payer.publicKey, null, DECIMALS, undefined, opts)
```

F-08 Placeholder values and third-party metadata host

Low **Category:** Configuration **Location:** L26-L27, L30 **Status:** Open

Description. Name, symbol and URI are placeholders, and the URI points to jsonkeeper.com, a third-party JSON paste service.
Impact. Placeholders could be published permanently, especially if metadata is made immutable. If the host deletes or changes the file, the token loses its name, image and description in wallets and explorers.
Recommendation. Add a guard that aborts on placeholder values. Host metadata and the logo on permanent storage (Arweave or IPFS via a pinning service), confirm the JSON (name, symbol, description, image) loads before deployment.

```
if ([TOKEN_NAME, METADATA_URI].some(v => /MY TOKEN|YOUR_JSON/.test(v)))
  throw new Error('Replace placeholder values first');
```

F-09 No input validation

Low **Category:** Input validation **Location:** L26-L30, L128 **Status:** Open

Description. The constants are used without checks. Metaplex limits name to 32, symbol to 10 and URI to 200 bytes, and the minted raw amount must fit in an unsigned 64-bit integer.
Impact. Invalid values fail late in the flow, after the mint already exists. A larger supply or more decimals (for example 20 billion with 9 decimals) would overflow u64.
Recommendation. Validate lengths, integer-ness of supply and decimals, and the u64 bound before sending any transaction.

```
const raw = BigInt(TOTAL_SUPPLY) * 10n ** BigInt(DECIMALS);
if (raw > 18446744073709551615n) throw new Error('Supply exceeds u64');
```

F-10 Floating-point exponent in supply calculation

Low **Category:** Arithmetic **Location:** L128 **Status:** Open

Description. BigInt(10 ** DECIMALS) computes the power as a JavaScript Number first. It is exact for the current value but becomes imprecise for large exponents and throws for non-integer input.
Impact. Latent bug if decimals are changed later.
Recommendation. Use BigInt exponentiation (see F-09 snippet).

F-11 No post-deployment on-chain verification

Low **Category:** Verification **Location:** L160 **Status:** Open

Description. The script prints success without reading the chain to confirm the final state.
Impact. Silent failures or wrong parameters go unnoticed.
Recommendation. Fetch the mint and metadata after deployment and assert supply, decimals, both authorities null and metadata values; print explorer links.

```
const info = await getMint(connection, mint);
console.assert(info.mintAuthority === null && info.freezeAuthority === null);
console.assert(info.supply === raw);
```

F-12 No balance pre-check, priority fee or retry

Low **Category:** Reliability **Location:** L50, L110 **Status:** Open

Description. No SOL balance check, no compute-unit price and no retry or backoff for transient RPC errors.
Impact. Congestion or low balance can interrupt the multi-step flow (amplifies F-02).
Recommendation. Check balance up front, add a modest priority fee on mainnet, and retry idempotent steps with backoff.

I-01 Token standard naming: SPL vs SLP

Informational **Category:** Documentation **Location:** General **Status:** Open

Description. The script creates an **SPL** token (Solana Program Library). "SLP" (Simple Ledger Protocol) is a different standard on Bitcoin Cash.
Impact. Using the wrong term in docs or listings can confuse users and listing teams.
Recommendation. Call the asset an SPL token in all materials.

I-02 Fixed 4-second sleep is redundant

Informational **Category:** Code quality **Location:** L66-L68 **Status:** Open

Description. createMint already waited for finalized commitment, so the extra fixed delay adds no safety and is not a reliable synchronization method.
Impact. None; slows the script slightly.
Recommendation. Remove it, or poll getAccountInfo for the mint if an additional check is wanted.

I-03 Dependency version and API compatibility

Informational **Category:** Supply chain **Location:** L1-L22 **Status:** Open

Description. No versions are shown. createCreateMetadataAccountV3Instruction belongs to mpl-token-metadata v2.x, and bs58 v6 and later is ESM-only, which can break require() on some Node versions (the script already handles the default-export shape).
Impact. Unpinned installs may fail or pull a compromised or incompatible release.
Recommendation. Pin exact versions in package.json, commit the lockfile, run npm audit, and use a current LTS Node.js release.

7. Positive Observations

- Mint and freeze authorities are revoked using the correct SDK calls and in the right order relative to minting.
- Secrets come from the environment, are checked for presence and are never printed.
- All SDK calls use correct argument order and types; BigInt is used for the raw supply.
- Finalized commitment is used for every critical step, which avoids acting on unconfirmed state.
- Configuration is isolated in a clearly marked block, making review simple.
- Token parameters are sensible for a fixed-supply meme token (9 decimals, supply within u64).

8. Recommended Remediation Order

Priority	Action	Findings
1	Make metadata immutable (or renounce update authority) after final testing	F-01
2	Set freeze authority null at creation; make deployment atomic or resumable; fix exit code	F-07, F-02, F-03
3	Explicit cluster selection with confirmation; dedicated deployer wallet and RPC	F-05, F-06, F-12
4	Add input validation and replace placeholder metadata; host JSON and logo permanently	F-08, F-09, F-10
5	Add post-deploy verification; publish allocation plan and lock liquidity	F-11, F-04
6	Pin dependencies; rehearse the whole flow on devnet before mainnet; get a human auditor to re-check	I-01 to I-03

Appendix A - Bangla to English Translation Log

Only comments were translated. No code, string literal used at runtime or identifier was changed. The Bangla script itself is not embedded in this PDF; the original text is identified by line number and meaning.

Line	Original Bangla comment (meaning)	English translation used
24	Token details heading: "change only this part"	TOKEN DETAILS (change only this section)
26	Token's name	Token name
27	Token's symbol	Token symbol
28	Total supply (for example: 1 billion)	Total supply (e.g., 1 billion)
29	Solana standard	Solana standard
30	Your JSON metadata link	Your JSON metadata link

The translated script is delivered alongside this report as `deploy_token_translated.js`. It is a faithful translation only; the recommendations in this report have **not** been applied to it.

Appendix B - Disclaimer

This document is an AI-assisted security review produced from a static reading of the supplied script. It is not an audit by CertiK, Consult or any other firm, does not use their names or methodologies as certification, and provides no guarantee that the code is free of vulnerabilities. Findings reflect the supplied code at the time of review; later changes are not covered. Token launches carry financial, legal and market risks that are outside this report. Nothing here is investment, legal or financial advice. Obtain an independent review by qualified human auditors before deploying to mainnet.



END OF REPORT

This report is an AI-assisted static security review of the supplied script only.
It is not a CertiK, Consult or any other third-party firm's audit and carries no certification.
It does not guarantee the absence of vulnerabilities or protect against market, legal or operational risk. Have a qualified human auditor verify before mainnet deployment.
This is not financial or investment advice.