



THE VIBE CODE STORE · WHITEPAPER

# VCS: A Safe Place to Vibe-Code

Letting every employee build with AI, without the security risk,  
and the Solana token that powers the network

VCS Core Contributors

vcs.app · Version 0.5 (draft) · October 2026

**ABSTRACT.** Companies that do not adopt AI will fall behind, yet the usual routes are slow, expensive or unsafe: waiting for IT, building in-house, or letting staff paste company data into any AI tool. Employees who vibe-code their own tools are more productive, but every unmanaged app opens a path for data leaks and attacks. VCS, the Vibe Code Store, gives each company its own safe place to vibe-code. Apps written by any model run in isolated sandboxes with their own database, reach company systems only through connectors the company approves, sit behind single sign-on, and are scanned and scored before a teammate approves them; the approval is anchored on Solana. VCS runs inside a company's own systems or is set up and managed for them in Singapore. We describe the problem, the product, security and deployment models, the market (financial services and crypto first, starting in Singapore, then every company), the business model and VCS, an SPL token that teams can pay with, that burns with every payment, and that is minted only to the builders, security reviewers and bug hunters who make the network safer and more useful. For the token we prove a hard supply cap, derive the price at which net issuance is zero, show convergence of supply to a unique fixed point, and give closed-form conditions that make wash usage and dishonest reviewing unprofitable.

**KEYWORDS:** vibe coding; shadow AI; security scoring; single sign-on; Solana; burn-and-mint tokenomics.

## 1. INTRODUCTION

Every company now faces the same choice. Teams that use AI to build their own tools move faster; teams that do not, fall behind. Language models let a non-engineer describe a tool in plain words and receive working code in minutes, a practice known as *vibe coding*. The demand for small internal tools, from dashboards to approval forms to payout trackers, was always far larger than engineering capacity, and that demand can now be met at the point of need.

The obstacle is no longer writing the code. It is running it safely. When employees build with whatever AI tool is at hand, company data is pasted into public services, apps run on personal accounts, code goes unreviewed, keys end up inside source files, and nobody knows what exists or who owns it. Each of those apps is a new path for a leak or an attack. Security teams respond by saying no, and the productivity gain is lost.

**Thesis.** The bottleneck for AI-built software has moved from *writing* code to *trusting* it. A company that can give every employee a safe place to build, where security is measured instead of assumed, captures the productivity of vibe coding without its risk.

VCS is that place. The first half of this paper covers the business: the problem (Section 2), the product (Section 3), how it works and how it is secured (Section 4), the market (Section 5), competition (Section 6), the business model (Section 7) and go-to-market (Sec-

tion 8). The second half covers the token: its model and proofs (Section 9), incentive security (Section 10), governance (Section 11), launch, funding, risks and legal considerations.

## 2. THE PROBLEM

### 2.1 Use AI, or fall behind

A company that wants AI-built tools has four options, and three of them fail on speed, cost or safety. Waiting

TABLE 1. The options a company faces.

| Option               | Fast | Affordable | Safe |
|----------------------|------|------------|------|
| Wait for IT          | ×    | ×          | ✓    |
| Build it yourself    | ×    | ×          | ✓    |
| Let staff use any AI | ✓    | ✓          | ×    |
| <b>Use VCS</b>       | ✓    | ✓          | ✓    |

is not free: tools sit in a queue for months while the work stays manual and competitors pull ahead. Building in-house means hiring engineers and a security team before the first tool ships.

### 2.2 Shadow AI

Employees are already building with AI, usually without guardrails. Six risks recur:

1. company data pasted into public AI tools;
2. apps running on personal accounts;

3. code nobody reviewed;
4. passwords and API keys inside the code;
5. no record of who built what, or who can use it;
6. open doors for attackers through unmanaged apps.

### 2.3 The trust gap

An AI-written tool can safely reach production only when five questions have answers: *where does it run, what can it touch, who approved it, is this the approved version, and what happens when the builder leaves?* Each unanswered question sends the decision back to engineering or security, which restores the queue. We call the distance between “AI can write it” and “we can safely use it” the trust gap.

### 2.4 Who feels it first

The gap is widest where the cost of a leak is highest. Financial services firms (banks, fintechs, insurers) and crypto companies (exchanges, protocols, DAOs) hold sensitive data and money, answer to regulators or token holders, and still need tools faster than their engineering teams can deliver. We start with them, in Singapore, where trust in regulated technology is high; if VCS is safe enough for them, it is safe for everyone.

## 3. THE IDEA: THE VIBE CODE STORE

### 3.1 Product thesis

VCS gives each company its own safe place to vibe-code. Employees with no security background describe a tool, any AI model writes it, and VCS takes care of the rest: it runs the app in a sealed box, connects it only to approved systems, checks and scores it, routes it to a teammate for approval, and records the approval on Solana. Mobile app stores gave phone software distribution, review and payment; VCS gives company software a safe runtime, a measured review and a proof.

### 3.2 What the company controls

- *Its own database* for every app, inside the company’s environment.
- *Approved connectors only*: the company chooses which systems apps may reach.
- *Single sign-on* on every app the company creates, with roles from its identity provider.
- *A security score* for every app; below the bar, it does not go live.
- *An audit log and a kill switch* for every app.

### 3.3 Two ways to run it

Companies with an IT team run VCS *in their own system*: their cloud or servers, their database, their single sign-on; company data never leaves. Companies without a tech team let VCS *set it up for them*: a private environment just for that company, with its own database and single sign-on, hosted in Singapore and run by VCS.

### 3.4 Who uses it

TABLE 2. Personas and what each gets from VCS.

| Persona         | Job and value                                                                                                                |
|-----------------|------------------------------------------------------------------------------------------------------------------------------|
| Builder         | Any employee. Describes a tool, gets a working app with real data, without waiting for engineers or knowing security.        |
| Reviewer        | Team lead or security officer. Sees the app’s score and what it asks for, approves with a signature, can revoke at any time. |
| Admin           | IT or operations owner. Sets approved connectors, single sign-on and approval rules; one catalogue of every app.             |
| Template author | Open-network builder. Publishes reusable apps and connectors, earns VCS when organisations use them.                         |
| Auditor         | Community member or investor. Verifies on chain what was approved, by whom, and when.                                        |

### 3.5 The user journey

#### JOURNEY 1: FROM IDEA TO LIVE APP

1. **Sign in.** With company single sign-on, or a Solana wallet (Sign In With Solana [4]).
2. **Describe.** “A bounty board where contributors claim tasks for VCS.”
3. **Build.** Any language model writes the app.
4. **Box and score.** VCS compiles it in an isolated sandbox, fingerprints it and scores it (Section 4.5).
5. **Permissions.** The app lists the data it needs; optional access can be switched off.
6. **Approve.** A reviewer checks the score and the permissions, and signs.
7. **Prove.** The approval is anchored on Solana.
8. **Live.** The team uses the app; every update repeats steps 4–7.

### 3.6 What people build first

Table 3 lists typical first applications for finance and crypto teams; the same patterns apply to any company. Each maps to an existing pain point and a small, explicit set of data permissions.

## 4. HOW IT WORKS

### 4.1 Applications as chains of builds

**DEFINITION 1 (Build).** A build is a tuple  $b = (h_s, h_c, S_{\text{req}}, \sigma)$  where  $h_s = H(\text{source})$ ,  $h_c = H(\text{config})$ ,  $S_{\text{req}}$  is the set of requested connector scopes, and  $\sigma$  is the builder’s signature over  $(h_s, h_c, S_{\text{req}}, h_{b-})$ , with  $h_{b-}$  the hash of the preceding build.

The platform, not the builder, compiles and runs builds; only a build whose hash matches a signed record may execute. This closes question 4 of the trust gap: the running version is provably the approved version.

TABLE 3. Typical first applications.

| App                   | What it replaces                | Permissions                       | Why it matters                                      |
|-----------------------|---------------------------------|-----------------------------------|-----------------------------------------------------|
| Treasury dashboard    | Spreadsheets of wallet balances | wallet balances (read)            | One view of every wallet; read-only by construction |
| Contributor payroll   | Manual monthly transfers        | team list (read), payouts (write) | Payments tied to an approved, versioned app         |
| Bounty board          | Forum threads and chats         | bounties (write), payouts (write) | Rewards paid automatically when work is accepted    |
| Token vesting tracker | Ad hoc vesting sheets           | vesting schedule (read)           | Transparent unlocks for team and investors          |
| Grant tracker         | Grant forum posts               | grants (read/write)               | Milestones and releases in one place                |
| Expense claims        | Receipts in chat                | ledger (write), team list (read)  | Reimbursement with a clear approval trail           |
| Governance tracker    | Bookmarks of proposals          | proposals (read)                  | Deadlines, quorum and outcomes for contributors     |

## 4.2 Sandboxed execution

Each build runs in an isolated runtime with its own datastore, no network egress except through connectors, and per-application limits on compute, storage and requests. Isolation means the system does not need to trust that model-written code is correct; only the boundary must hold.

## 4.3 Connectors and the access rule

A connector is the only path from a sandbox to an organisational system, such as a wallet, a ledger, a contributor list or a governance forum. Applications never hold credentials.

DEFINITION 2 (Effective access). *For build  $b$  and user  $u$ ,*

$$A(b, u) = S_{\text{req}}(b) \cap S_{\text{appr}}(b) \cap R(u), \quad (1)$$

*where  $S_{\text{appr}}(b)$  is approved by the reviewer and  $R(u)$  is what  $u$ 's role permits.*

PROPOSITION 1 (Least privilege).  *$A(b, u) \subseteq R(u)$  and  $A(b, u) \subseteq S_{\text{appr}}(b)$ , and  $A$  cannot grow when any of the three sets shrinks.*

*Proof.* An intersection is contained in each operand and is monotone in each.  $\square$

## 4.4 Attestation ledger

DEFINITION 3 (Attestation).  $a_k = H(a_{k-1} \parallel h_b \parallel H(S_{\text{appr}}) \parallel \text{sig}_r \parallel \tau_k)$ , *anchored on Solana, where  $H$  is SHA-256,  $\text{sig}_r$  is the reviewer's Ed25519 wallet signature and  $\tau_k$  the slot of inclusion.*

PROPOSITION 2 (Tamper evidence). *If  $H$  is collision resistant, altering any field of attestation  $j < k$  changes  $a_k$ , except with negligible probability.*

*Proof sketch.* Induction on  $k-j$ : each altered preimage changes its hash, which is part of the next preimage.  $\square$

## 4.5 Security score

Every build is checked before a human sees it. Let  $k = 1, \dots, K$  index the checks (secrets scan, dependency vulnerabilities, network egress, permission scope, unsafe code patterns, company policy), with weights  $w_k \geq 0$ ,  $\sum_k w_k = 100$ , and results  $c_k(b) \in [0, 1]$ .

DEFINITION 4 (Security score).  $\sigma(b) = \sum_k w_k c_k(b)$ . *A build may go live only if  $\sigma(b) \geq \tau$  and every critical check passes ( $c_k(b) = 1$  for  $k \in \mathcal{C}$ ). The default bar is  $\tau = 80$ ; companies may raise it.*

The score is recomputed for every build, so an approved app that is later edited must be scored and approved again. Security reviewers who certify reusable templates and connectors stake on them (Section 10), so a certified component carries an economic guarantee as well as a score.

## 4.6 Identity

Users sign in through the company's identity provider (SAML or OIDC) or, for crypto teams, a Solana wallet. Role sets  $R(u)$  in the access rule (1) come from the identity provider's groups, so removing someone from a group removes their access to every app at once.

## 4.7 Privacy

Organisational data never leaves its tenant; Solana holds only hashes and signatures. The public learns that an approval occurred, not what it approved. An organisation can reveal a preimage to an auditor, who checks it against the anchored hash.

## 4.8 Implementation on Solana

VCS runs on Solana [5]. Four properties drive the choice. *Cost*: anchoring an attestation is one transaction, and Solana's base fee is 5,000 lamports per signature (0.000005 SOL), so a proof per app version costs a fraction of a cent. *Speed*: confirmations arrive in seconds, so approval feels instant to the reviewer. *Users*: crypto teams already hold Solana wallets such as Phantom, Solflare and Backpack and approve spend-

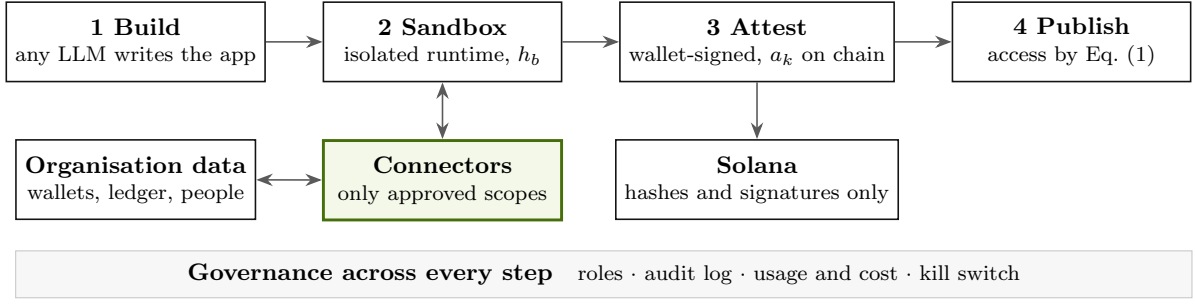


FIGURE 1. System architecture. Every application reaches organisational data only through connectors; only proofs reach Solana.

ing with Squads multisigs, while companies without wallets sign in with single sign-on and never touch the chain directly. *Standards*: the SPL token programs provide minting, burning and non-transferable balances without custom token code.

TABLE 4. Protocol components on Solana.

| Component   | Implementation                                                                                                              |
|-------------|-----------------------------------------------------------------------------------------------------------------------------|
| Sign-in     | Sign In With Solana message signed by the wallet (Ed25519).                                                                 |
| Reviewer    | Wallet signature; organisations may require a Squads multisig threshold.                                                    |
| Attestation | VCS program writes $a_k$ to a program account and checks that it extends $a_{k-1}$ .                                        |
| VCS token   | SPL token; mint authority held by the emission program, which enforces Eq. (4) and the cap.                                 |
| Burn        | Protocol market buys followed by the SPL burn instruction; burns are public and auditable.                                  |
| Credits     | Non-transferable balances, e.g. the Token-2022 non-transferable extension, or an off-chain ledger with on-chain settlement. |

## 5. MARKET

### 5.1 Segments and sequencing

We enter in three stages, each chosen so that the previous stage produces what the next one needs. Stage 1

TABLE 5. Market sequence.

| Stage    | Who and why                                                                                                                                                                   |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. Now   | Financial services and crypto, starting in Singapore: banks, fintechs, insurers, exchanges, protocols and DAOs. The strictest security needs and the clearest cost of a leak. |
| 2. Next  | Regulated and data-sensitive enterprises: healthcare, logistics, large platforms.                                                                                             |
| 3. Later | Every company, of any size, through the managed option.                                                                                                                       |

builds the reference customers, the connector library and the security track record that later stages rely on.

### 5.2 Sizing method

We size the opportunity bottom-up. For segment  $i$  with  $N_i$  organisations, adoption share  $a_i$  and average revenue per account per month  $ARPA_i$ ,

$$ARR = 12 \sum_i N_i a_i ARPA_i. \quad (2)$$

Table 6 applies (2) to *illustrative* inputs. The organisation counts are working assumptions to be replaced with sourced figures; they are not market data.

TABLE 6. Illustrative bottom-up sizing,  $ARPA = \$1,500/\text{month}$ .

| Segment              | $N_i$ (assumed) | $a_i$ | Accts. | ARR    |
|----------------------|-----------------|-------|--------|--------|
| Finance & crypto     | 10,000          | 10%   | 1,000  | \$18M  |
| Regulated enterprise | 20,000          | 5%    | 1,000  | \$18M  |
| Every company        | 1,000,000       | 1%    | 10,000 | \$180M |

### 5.3 Buyer and budget

The economic buyer is the head of technology, operations or security, who is already asked both to speed up internal tools and to stop shadow AI. VCS replaces internal-tool subscriptions, contractor hours for one-off tools, and the hidden cost of manual work in spreadsheets and chat, while reducing the risk of unmanaged AI use.

## 6. COMPETITIVE LANDSCAPE

### 6.1 Where the moat comes from

Code generation is becoming a commodity; the defensible assets are speed and the ones that accumulate:

- *Speed*. We ship to the first segment before larger platforms adapt, and every customer adds to the assets below.
- *Connector library*. Each connector written for one organisation serves every organisation on the same system. The library grows with usage and is costly to replicate.
- *Approved catalogue*. Once an organisation's apps, permissions and approval history live in VCS, moving them means re-approving everything.

TABLE 7. Positioning against adjacent categories. Examples are illustrative of each category.

| Category                    | Examples                     | Strength                                                      | Gap VCS fills                                                             |
|-----------------------------|------------------------------|---------------------------------------------------------------|---------------------------------------------------------------------------|
| AI app builders             | Lovable, Bolt, Replit, v0    | Fastest path from prompt to working code                      | No organisational runtime, permissions or approval binding                |
| Internal-tool platforms     | Retool, Power Apps, Appsmith | Governance, enterprise connectors                             | Built for engineers or trained builders; no public proof of approval      |
| Multisig and treasury tools | Squads and similar           | On-chain approval of transactions                             | Approve money, not software; no app runtime                               |
| Generic cloud hosting       | Major clouds and PaaS        | Scale and reliability                                         | Governance is left to each team to assemble                               |
| <b>VCS</b>                  |                              | Anyone builds; every app boxed, scored, approved and provable | AI-native, secure by default, runs in the company's own system or managed |

- *Attestation history.* A public, tamper-evident record of what was approved becomes a trust asset in itself, especially for organisations accountable to token holders.
- *Builder network.* Token rewards (Section 9) pay outside builders to extend the library, so supply grows faster than any single team could build it.

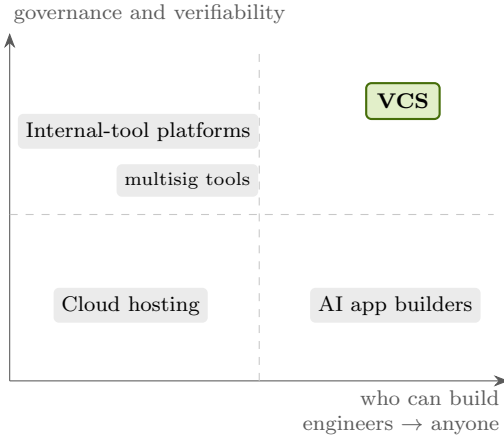


FIGURE 2. Positioning map. VCS targets the empty quadrant: anyone can build, and every app is governed and provable.

## 7. BUSINESS MODEL

### 7.1 Revenue streams

Everything a company buys from VCS (its plan, usage credits and security reviews) is priced in dollars and can be paid in USD, USDC or vcs. The business does not depend on the token price.

### 7.2 Pricing principles

Pricing is set from pilot data; three principles are fixed. First, organisations always see dollar prices; credits never change price. Second, building is cheap and running is metered, so experimentation is encouraged and value scales with use. Third, read-only apps cost

TABLE 8. Revenue streams.

| Stream           | Basis                                 | Retained |
|------------------|---------------------------------------|----------|
| Subscription     | per active builder or user, per month | 100%     |
| Build Credits    | prepaid usage at \$0.01 per credit    | 70%      |
| Security reviews | per review or certification           | 70%      |
| Implementation   | one-time, custom connectors           | 100%     |
| Marketplace      | share of template sales               | fee      |

TABLE 9. How a payment flows.

| Paid in         | What happens                                                                                          |
|-----------------|-------------------------------------------------------------------------------------------------------|
| USD or USDC     | 70% kept as revenue; 30% buys vcs on the market and burns it                                          |
| vcs             | 15% off the dollar price; half the vcs is burned, half goes to the treasury                           |
| 12 months ahead | The team keeps today's rate for the year. VCS prepayments sit in an on-chain escrow, released monthly |

less than apps that write data or move funds, because review and risk are lower.

### 7.3 Unit economics

Lifetime value and payback follow the standard definitions

$$\text{LTV} = \frac{\text{ARPA} \cdot g}{c}, \quad \text{payback} = \frac{\text{CAC}}{\text{ARPA} \cdot g}, \quad (3)$$

with gross margin  $g$  and monthly churn  $c$ . Table 10 gives *illustrative* values to be replaced by pilot measurements.

*Cost drivers.* Compute for sandboxes, Solana fees for anchoring (fractions of a cent per proof), connector maintenance, and review tooling. Gross margin depends mainly on sandbox density and on how much connector work is reusable rather than custom.

TABLE 10. Illustrative unit economics.

| Input or output           | Value    |
|---------------------------|----------|
| ARPA (per month)          | \$1,500  |
| Gross margin $g$          | 75%      |
| Monthly churn $c$         | 2%       |
| Customer acquisition cost | \$6,000  |
| LTV                       | \$56,250 |
| LTV / CAC                 | 9.4      |
| Payback (months)          | 5.3      |

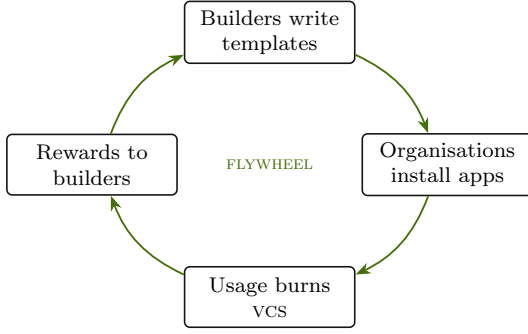


FIGURE 3. The network flywheel: more templates attract more organisations, whose usage funds rewards that attract more builders.

## 8. GO TO MARKET

### 8.1 Ideal first customer

A Singapore-based financial services or crypto company with 50–2,000 staff, compliance obligations, a small internal tools team, and employees already experimenting with AI.

### 8.2 Motion: pilot, then expand

Sales follow a standard enterprise path: an introduction by form or email; scoping and a security review with NDA; a signed pilot agreement and data processing addendum; workspace set-up with single sign-on and approved connectors; a 30–60 day pilot against agreed success metrics; and an order form for the full plan, invoiced in USD or USDC or paid in VCS. Inside the company, adoption starts with one low-risk app and expands as connectors and reviewers already exist.

### 8.3 Channels

- *Direct sales in Singapore* to financial services and crypto companies.
- *Partners*: system integrators and security consultancies who deploy VCS for clients.
- *The Solana ecosystem*: grant programmes, hackathons and builder bounties paid in VCS, which seed the template library.
- *The token launch* as a community moment, with bounties and builder quests live on day one.

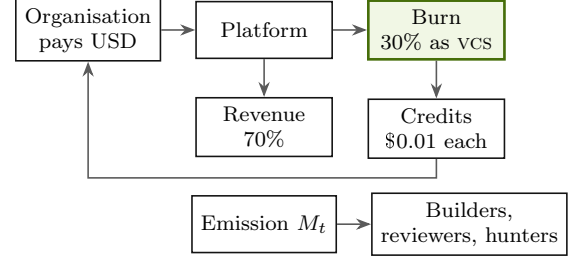


FIGURE 4. The credit loop. Usage burns VCS; new VCS reaches only builders, security reviewers and bug hunters whose work is used.

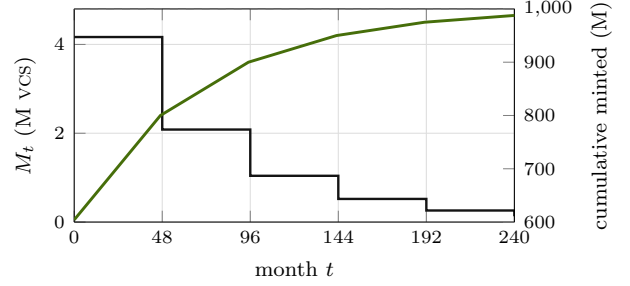


FIGURE 5. Emission (black) and cumulative minted supply (green) approaching the cap (dashed).

## 9. TOKEN MODEL

### 9.1 Units

VCS is a transferable SPL token on Solana with market price  $P_t$  (USD). A Build Credit (BC) is non-transferable and fixed at  $p_c = \$0.01$ ; it is a prepaid service balance, not an asset. Table 11 lists notation.

TABLE 11. Notation and baseline parameters.

| Symbol           | Meaning                          | Baseline        |
|------------------|----------------------------------|-----------------|
| $\bar{S}$        | hard cap on minted VCS           | $10^9$          |
| $G$              | genesis allocation               | $6 \times 10^8$ |
| $M_t$            | emission in month $t$            | Eq. (4)         |
| $U_t$            | credit sales in month $t$ (USD)  | scenario        |
| $\theta$         | burn share of credit sales       | 0.30            |
| $\alpha, \gamma$ | builder, reviewer share of $M_t$ | 0.70, 0.20      |
| $\phi$           | burn share of VCS payments       | 0.50            |
| $\delta$         | discount for paying in VCS       | 0.15            |
| $P_t$            | VCS price (USD)                  | market          |
| $V$              | annual velocity of VCS           | 4–12            |

### 9.2 Emission and the hard cap

$$M_t = M_0 2^{-\lfloor t/48 \rfloor}, \quad M_0 = \frac{\bar{S} - G}{96} \approx 4.17 \times 10^6. \quad (4)$$

PROPOSITION 3 (Hard cap).  $G + \sum_{t \geq 0} M_t = \bar{S}$ .

*Proof.* Era  $e$  contributes  $48M_0 2^{-e}$ ; summing,  $\sum_t M_t = 96M_0 = \bar{S} - G$ .  $\square$

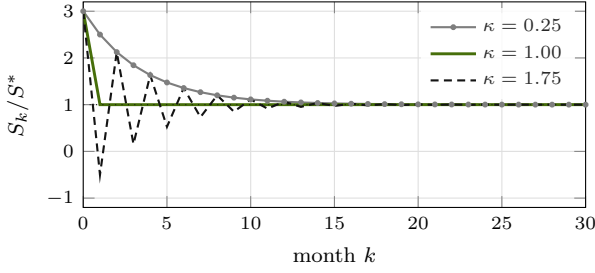


FIGURE 6. Theorem 1: supply normalised by  $S^*$  from  $S_0 = 3S^*$ .

### 9.3 Burn, supply and break-even price

Monthly burn and supply are

$$B_t = \frac{\theta U_t}{P_t} + D_t, \quad S_{t+1} = S_t + M_t - B_t, \quad (5)$$

where  $U_t$  counts payments made in dollars and  $D_t = \phi V_t$  burns a share of the  $V_t$  VCS paid directly by teams. We set  $D_t = 0$  below; direct payments only strengthen the results. VCS held in 12-month escrow also leaves the float for its term.

DEFINITION 5 (Break-even price).  $P_t^* = \theta U_t / M_t$ .

PROPOSITION 4 (Sign of net issuance).  $S_{t+1} < S_t$  if and only if  $P_t < P_t^*$ .

*Proof.*  $S_{t+1} - S_t = M_t - \theta U_t / P_t < 0 \iff P_t < \theta U_t / M_t$ .  $\square$

In business terms: whenever the market values VCS below the usage it settles, supply shrinks. Break-even rises with adoption and doubles at each halving.

### 9.4 Convergence under quantity-theory demand

By the equation of exchange [2], settling annual usage  $12U$  at velocity  $V$  requires float value  $D = 12U/V$ . If the market clears at  $P_t = D/S_t$ , then  $B_t = \kappa S_t$  with  $\kappa = \theta V / 12$ .

THEOREM 1 (Geometric convergence). *For constant  $U, V, M$  and  $0 < \kappa < 2$ , supply has the unique fixed point  $S^* = M/\kappa$  and  $|S_t - S^*| = |1 - \kappa|^t |S_0 - S^*|$ .*

*Proof.*  $S_{t+1} = (1 - \kappa)S_t + M$ , so  $S_{t+1} - S^* = (1 - \kappa)(S_t - S^*)$ ; iterate, with  $|1 - \kappa| < 1$  iff  $0 < \kappa < 2$ .  $\square$

COROLLARY 1 (Price consistency). *At the fixed point  $P = D/S^* = \theta U/M = P^*$ .*

Since  $\kappa < 2$  iff  $V < 80$ , the condition holds for any realistic velocity. At  $V = 8$ , deviations halve roughly every three months.

### 9.5 Scenarios

Paying organisations follow a logistic diffusion [3],  $N_t = K/(1 + e^{-r(t-t_0)})$ , with  $r = 0.12$ ,  $t_0 = 36$  and ARPA = \$1,500, so  $U_t = 1,500 N_t$ . Scenarios differ in saturation  $K$ .

TABLE 12. Break-even price  $P_t^*$  (USD) by scenario.

| Scenario              | $t = 12$ | $t = 24$ | $t = 60$ | $t = 120$ |
|-----------------------|----------|----------|----------|-----------|
| Bear ( $K = 1,000$ )  | 0.006    | 0.021    | 0.205    | 0.432     |
| Base ( $K = 5,000$ )  | 0.029    | 0.103    | 1.023    | 2.160     |
| Bull ( $K = 20,000$ ) | 0.115    | 0.414    | 4.090    | 8.640     |

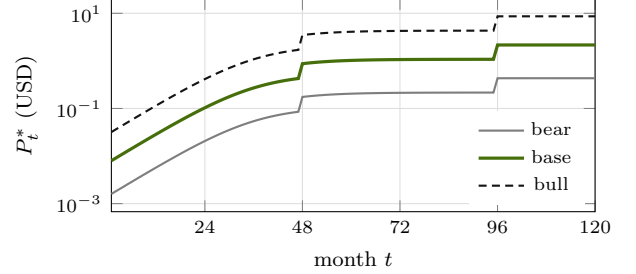


FIGURE 7. Break-even price under logistic adoption; the jump at  $t = 48$  is the first halving.

LEMMA 1 (Price independence for users). *An organisation's cost for a fixed workload does not depend on  $P_t$ .*

*Proof.* Workloads are priced in BC at constant  $p_c$ ;  $P_t$  affects only the quantity the protocol buys and burns.  $\square$

## 10. INCENTIVE SECURITY

### 10.1 Builder rewards

Each month  $\alpha M_t$  is paid to builders in proportion to verified paid usage of their components,  $r_i = \alpha M_t c_i / \sum_j c_j$ . Only credits bought with dollars or burned VCS by verified tenants count, and a builder's own tenant is excluded.

### 10.2 Wash usage

An attacker spending  $x$  USD against legitimate usage  $X$  earns

$$\pi(x) = \alpha M_t P_t \frac{x}{X + x} - x. \quad (6)$$

PROPOSITION 5 (Wash resistance).  $\pi(x) \leq 0$  for all  $x > 0$  iff  $\alpha M_t P_t \leq X$ ; otherwise the optimal spend is  $x^\dagger = \sqrt{\alpha M_t P_t X} - X$ .

*Proof.*  $\pi$  is concave,  $\pi(0) = 0$ ,  $\pi'(0) = \alpha M_t P_t / X - 1$ . If  $\pi'(0) > 0$ ,  $\pi'(x) = 0$  gives  $x^\dagger$ .  $\square$

COROLLARY 2. *At  $P_t = P_t^*$  and  $X = U_t$  the condition is  $\alpha \theta = 0.21 \leq 1$ ; attacks pay only above  $P_t^* / (\alpha \theta) \approx 4.76 P_t^*$ .*

### 10.3 Security reviewers

A security reviewer stakes  $s$  on an app or connector they certify and earns rate  $\rho$  while it is in good standing. A faulty component is detected each month with probability  $q$ , and a fraction  $\lambda$  of stake is slashed.

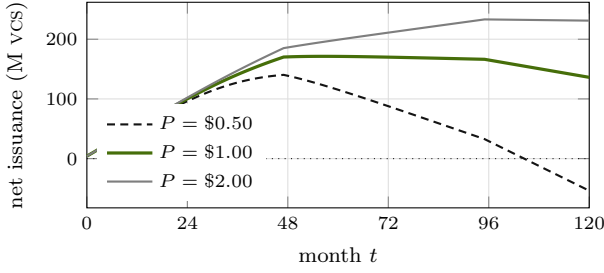


FIGURE 8. Cumulative net issuance, base scenario, at three constant prices.

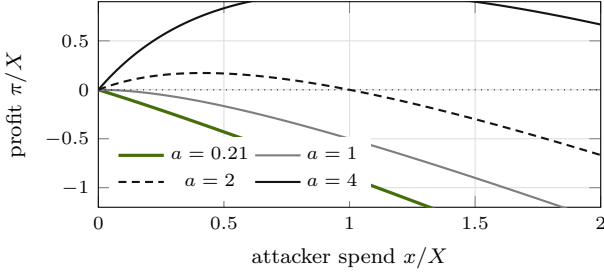


FIGURE 9. Wash-attack profit,  $a = \alpha M_t P_t / X$ . At equilibrium ( $a = 0.21$ ) every attack loses money.

PROPOSITION 6 (Honest reviewing). *Vouching for a faulty component loses money in expectation iff  $q\lambda > \rho$ ; the protocol sets  $\lambda \geq 2\rho/q_{\min}$ .*

*Proof.* Expected monthly payoff is  $s(\rho - q\lambda)$ .  $\square$

## 11. GOVERNANCE AND ECOSYSTEM

Responsibilities are split so that customers get a dependable product and token holders steer the open network. The bounty and grants pool (10% of emission)

TABLE 13. Who decides what.

| Decided by        | Scope                                                                                                        |
|-------------------|--------------------------------------------------------------------------------------------------------------|
| VCS company       | Product, security, pricing in dollars, customer contracts, sandbox and connector reliability.                |
| Token holders     | Which public connectors the treasury funds, staking parameters ( $\lambda$ , $q_{\min}$ ), grant programmes. |
| Each organisation | Its reviewers, roles, approved apps and revocations.                                                         |

pays bug hunters and funds connectors and templates that no single customer would pay for but many will use. A builder programme pairs grants with usage-based rewards, so early builders are paid for shipping and later builders for adoption.

TABLE 14. Allocation of  $\bar{S} = 10^9$  vcs.

| Pool                 | Share | Release               |
|----------------------|-------|-----------------------|
| Network emission     | 40%   | Eq. (4)               |
| Team                 | 15%   | 1-yr cliff, 4-yr vest |
| Early investors      | 15%   | 1-yr cliff, 3-yr vest |
| Treasury             | 15%   | governed, multi-year  |
| Ecosystem & partners | 10%   | milestone grants      |
| Liquidity            | 5%    | at token launch       |

TABLE 15. Phases and the gate that opens the next one.

| Phase    | Focus                                                         | Gate                           |
|----------|---------------------------------------------------------------|--------------------------------|
| 1 Build  | Product, security scoring, first connectors; audits           | External security audit passed |
| 2 Pilot  | Paid pilots with finance and crypto teams in Singapore        | Paying customers               |
| 3 Launch | Token launch with its uses live; builder and reviewer network | Legal opinion on the token     |
| 4 Scale  | Regulated enterprises, then every company                     | —                              |

## 12. DISTRIBUTION, ROADMAP AND FUNDING

### 12.1 Allocation

### 12.2 Roadmap with gates

Each gate is measured by a small set of metrics: active builders per organisation, apps per organisation, 30-day app retention, review time, and credit usage per account.

### 12.3 Funding and token launch

The first capital is raised from investors, including crypto venture funds, as a SAFE or a simple agreement for future tokens, with rights to the investor allocation (Table 14) under the stated lock-ups. The token launches only when its uses are live: payment, escrowed annual plans, reviewer staking, bug bounties and builder rewards. The launch doubles as the network's main community moment.

TABLE 16. Illustrative use of a first round of about US\$1.2M over 18 months.

| Use                                       | US\$ |
|-------------------------------------------|------|
| Engineering team                          | 270k |
| Founder salary                            | 145k |
| Security audits                           | 120k |
| Legal (entity, token opinion, agreements) | 100k |
| Infrastructure and AI                     | 90k  |
| Go-to-market and community                | 150k |
| Buffer                                    | 120k |

TABLE 17. Principal risks.

| Risk        | Description                                                                     | Mitigation                                                                                                               |
|-------------|---------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| Platform    | Model and cloud providers bundle hosting and governance into existing products. | Move fast; be model-neutral; compete on security scoring, connectors, approval history and the builder network.          |
| Security    | A sandbox escape or faulty connector exposes data or funds.                     | Defence in depth, external audits, write-scope limits, kill switch, staged rollout of write connectors.                  |
| Integration | Bespoke systems turn sales into one-off projects.                               | Price custom work separately; fund reusable connectors from the treasury.                                                |
| Adoption    | Reviewers become a new bottleneck.                                              | Risk-tiered review: read-only apps fast-tracked; templates pre-reviewed.                                                 |
| Token       | Price may not track usage; velocity is endogenous; speculation dominates early. | Prices stay in dollars; burn tied to payments; launch only with uses live.                                               |
| Regulatory  | Token treatment and data rules differ by jurisdiction.                          | Legal opinion before launch; customer data stays in the company’s environment; Singapore hosting for the managed option. |
| Model       | Theorem 1 assumes constant $U, V, M$ within a horizon.                          | Treat results as qualitative; recalibrate parameters from live data.                                                     |

### 13. RISKS AND MITIGATIONS

### 14. LEGAL CONSIDERATIONS

This paper describes a design, not an offer. Whether VCS is treated as a security, commodity or other instrument depends on jurisdiction and on facts at launch; we will seek specialist advice before any distribution. Credit purchases are service prepayments. Customer data is processed inside each organisation’s tenant, and only hashes and signatures are published.

### 15. CONCLUSION

Companies have to use AI to keep up, but letting every employee build with any tool is a security risk they cannot accept. VCS gives each company a safe place to vibe-code: its own database, approved connectors, single sign-on, a security score on every app, and approvals anchored on Solana, running in the company’s own system or managed in Singapore. We begin with financial services and crypto, where the cost of a leak is highest, and expand to every company. The business prices in dollars; the token lets teams pay, burns with every payment, and rewards the builders, reviewers and bug hunters who make the network safer, with a capped supply and incentives that make cheating unprofitable.

### REFERENCES

- [1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
- [2] I. Fisher, *The Purchasing Power of Money*. New York: Macmillan, 1911.
- [3] F. M. Bass, “A new product growth for model consumer durables,” *Management Science*, 15(5):215–227, 1969.
- [4] Phantom, “Sign In With Solana (SIWS),” specification.
- [5] A. Yakovenko, “Solana: A new architecture for a high performance blockchain,” whitepaper, 2018.

- [6] Helium Network, “Data Credits and the burn-and-mint equilibrium,” network documentation.

*Notice. This paper is a draft for discussion and not an offer to sell tokens or securities in any jurisdiction. Market sizes, unit economics and token parameters are illustrative assumptions, not forecasts, and are subject to change and legal review.*