



# Project Whitepaper

V2.0

e.kaya@onchainwin.com

September 2025

## Abstract

OnChainWin is a Web3 technology provider that delivers outsourceable, verifiably fair raffle infrastructure on EVM-compatible networks. The live interface at <https://onchainwin.com> operates as a proof of concept, while the core offering enables third-party integration of modular smart contracts for transparent giveaways, ticketed sales, community rewards, and NFT utilities.

Chainlink Verifiable Random Function (VRF) supplies cryptographically provable randomness across the entire raffle lifecycle. Ticketing, winner selection, and settlement occur on-chain, producing an auditable record that strengthens user trust and removes reliance on opaque processes.

Unlike consumer-facing raffle sites, OnChainWin functions as infrastructure that projects can embed into their own products and communities. This approach reduces engineering overhead, avoids marketplace commission leakage, and supports scalable engagement mechanics that are observable and verifiable.

A security-first architecture, a clear commercialization model, and developer-oriented integration paths position OnChainWin as a foundational layer for decentralized incentive mechanisms in the Web3 ecosystem.

<b>Vision and Positioning</b> .....	3
<b>Problem Statement</b> .....	4
<b>Solution Overview</b> .....	5
<b>System Architecture and Chainlink VRF</b> .....	7
<b>Protocol Mechanics: Raffle Lifecycle</b> .....	8
<b>Security, Fairness and Auditability</b> .....	10
<b>Business Model and Economics</b> .....	11
<b>Developer Integration and Developer Experience</b> .....	12
<b>Use Cases and Case Studies</b> .....	13
<b>Ecosystem, Partnerships and Compliance</b> .....	14
<b>Roadmap and Contact</b> .....	15

## 2. Vision and Positioning

OnChainWin is built to deliver a trustless winner selection process that eliminates manipulation and establishes fairness as a verifiable standard. Each raffle outcome is anchored in Chainlink Verifiable Random Function (VRF), turning transparency from a claim into a cryptographic guarantee.

The live interface at <https://onchainwin.com> serves as proof of concept and a community hub. It demonstrates a complete on-chain lifecycle for raffles, including ticketing, randomness requests, winner selection, and settlement. This public reference allows participants and developers to observe and validate the mechanics in real time.

Third-party projects can already integrate OnChainWin contracts to power giveaways, ticketed sales, community rewards, and NFT utilities. NFT collections, gaming ecosystems, creator communities, and decentralized applications can adopt OnChainWin to reduce engineering overhead, avoid marketplace commissions, and strengthen trust through verifiable outcomes.

Across the broader ecosystem, opacity in giveaways and reward distribution remains a significant barrier to community confidence. Centralized tools and manually run campaigns leave outcomes open to doubt, undermining engagement and slowing adoption.

OnChainWin addresses this gap by embedding verifiable fairness directly into on-chain execution, setting a higher baseline for transparency in incentive mechanisms.

For the long-term vision, OnChainWin aims to develop a proprietary Verifiable Random Function. Building an in-house VRF will further decentralize and harden the randomness stack, increase reliability, and provide a domain-specific solution for transparent incentive distribution at scale.

**WHY FULLY AUTOMATED?**

### Zero Human Intervention Required

Once deployed, the entire raffle lifecycle operates autonomously. Every step is triggered automatically by smart contract logic and Chainlink oracles.

Self-Executing Triggers	Chainlink Automation	Verifiable Randomness	Batch Processing
<p>When the raffle expiry timer runs out or the maximum participant cap is reached, the smart contract automatically triggers Chainlink VRF. No manual intervention required — the entire process is managed by code.</p>	<p>The Chainlink Keeper network continuously monitors on-chain conditions. When <code>checkUpkeep()</code> returns true, <code>performUpkeep()</code> executes automatically. Powered by decentralized node operators.</p>	<p>Chainlink VRF 2.5 generates cryptographically provable random numbers. No party — including the oracle — can predict or manipulate the outcome. Transparent and verifiable.</p>	<p>To avoid hitting gas limits, winners are processed in batches of 25. Chainlink Automation continues the loop until all winners are selected. Gas-efficient and scalable.</p>
Trigger <b>2 CONDITIONS</b>	Network <b>DECENTRALIZED</b>	Security <b>TAMPER-PROOF</b>	Batch Size <b>25 / BATCH</b>

### **3. Problem Statement**

Raffles, giveaways, and ticketed sales have become common tools for Web3 projects seeking to attract and retain communities. Despite their importance, most implementations remain opaque, fragmented, and vulnerable to manipulation. Outcomes are often determined by centralized actors or private scripts, leaving participants uncertain whether results are fair. This lack of verifiable transparency undermines trust, reduces engagement, and creates friction in community growth.

Traditional marketplaces and social platforms add further limitations. Fees and commissions reduce the value flowing back to creators and communities. Manual processes and closed systems introduce inefficiency and make it difficult to audit or replicate results. As projects scale, these weaknesses compound into higher costs, reputational risks, and community fatigue.

Within the broader Web3 ecosystem, fairness and verifiability are increasingly recognized as essential features rather than optional enhancements. Token launches, NFT mints, gaming assets, and loyalty programs all rely on mechanisms that distribute value to participants. Without cryptographic guarantees, these activities expose communities to doubt and slow adoption of decentralized models.

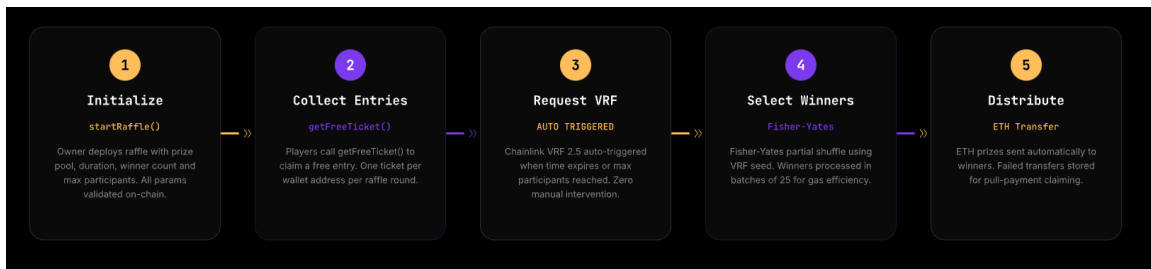
OnChainWin addresses this gap by introducing on-chain raffles powered by Chainlink VRF. Technology establishes fairness as a default condition, not a subjective claim, and makes outcomes independently auditable.

## 4. Solution Overview

OnChainWin delivers a modular raffle infrastructure designed for transparent, provably fair outcomes across Web3 ecosystems. The solution addresses the limitations of centralized giveaways and manual reward systems by combining on-chain execution, verifiable randomness, and outsourceable integration options.

### 4.1 Trustless Raffle Lifecycle

- Creation: Smart contracts define parameters such as ticket cost, prize pool, and closing time.
- Ticketing: Participants purchase entries on-chain, with transactions visible and auditable in real time.
- Randomness: Chainlink VRF supplies cryptographic randomness, making outcomes tamper-proof.
- Winner Selection: Contract logic automatically applies the randomness result to select winners.
- Settlement: Payouts or prize transfers are executed by the contract, closing the event without manual intervention.



### 4.2 Modular Contract Utilities

- Giveaway Contracts: Transparent prize draws for community engagement.
- Ticketed Sales Contracts: Cost-sharing models where multiple buyers contribute, and one receives the asset.
- Reward Distribution Contracts: Automatic distribution of tokens, NFTs, or digital assets to selected participants.
- NFT Utility Contracts: Integration of raffle mechanics into NFT collections to unlock new forms of utility.

### **4.3 Outsourcing Benefits for Projects**

- Reduced Development Overhead: No need to design, audit, and maintain bespoke raffle logic.
- Elimination of Hidden Costs: Marketplace commissions and middleman fees are avoided.
- Faster Deployment: Incentive mechanisms can be activated rapidly.
- Scalable Trust: Communities receive clear assurance that outcomes are fair and auditable.

### **4.4 Ecosystem Applications**

- NFT Mints and Launches.
- Gaming Economies.
- Creator Campaigns.
- Loyalty and Community Programs.

### **4.5 Proof of Concept and Community Layer**

The live platform at <https://onchainwin.com> demonstrates the system in action and serves as a community hub. It validates the operational model, gathers adoption feedback, and builds credibility for partners considering integration.

### **4.6 Long-Term Vision of Independence**

OnChainWin currently relies on Chainlink VRF to deliver verifiable randomness. In the long term, the roadmap includes the development of a proprietary VRF system to provide additional decentralization and a randomness stack tailored to incentive distribution.

## 5. System Architecture and Chainlink VRF

OnChainWin operates on EVM-compatible infrastructure with an architecture designed for transparency, modularity, and verifiable randomness. The production deployment is live on Scroll and Base networks, and the contracts can be integrated on any EVM-compatible chain.

### 5.1 Architectural Layers

- Smart Contract Layer: Core contracts manage raffle creation, ticket purchases, escrow of funds, randomness requests, winner selection, and settlement across EVM chains.
- Randomness Layer: Chainlink Verifiable Random Function delivers cryptographically secure randomness. Requests and responses are recorded on-chain.
- Application Layer: User interfaces, dashboards, and partner-facing endpoints provide access for participants and integrators.
- Integration Layer: SDKs, ABIs, and partner tooling enable third-party platforms to embed raffle utilities directly into existing products.

### 5.2 Randomness Flow with Chainlink VRF

- 1) A raffle contract triggers a VRF request when a draw closes.
- 2) The VRF oracle returns a random value with a cryptographic proof of validity.
- 3) The contract verifies the proof on-chain and applies the result.
- 4) The outcome and settlement are finalized on-chain.

### 5.3 Security and Auditability

- Automated Execution: Outcome determination and settlement execute through contract logic.
- Immutable Records: Ticket purchases, VRF requests, responses, and settlements are persisted on-chain.
- Fail-Safe Mechanics: Refund and expiration paths are included for edge cases such as unmet ticket thresholds or timeouts.

### 5.4 Developer Integration Paths

- Direct Contract Calls using ABIs on Scroll, Base, and other EVM networks.
- SDK and Helper Packages for common flows.
- API and Webhook Options for off-chain UX and analytics.
- Test Environments: A guided onboarding kit for Base network is available at <https://onchainwin.com/docs>.

### 5.5 EVM Compatibility and Multi-Chain Support

- Live Networks: Scroll and Base.
- EVM Compatibility: Contracts are portable to other EVM chains with minimal configuration.
- Extensibility: Additional modules and features can be deployed chain-by-chain.

### 5.6 Road to Independence in Randomness

Chainlink VRF currently supplies verifiable randomness. The long-term vision includes development of a proprietary VRF to further decentralize the randomness stack.

## 6. Protocol Mechanics: Raffle Lifecycle

OnChainWin defines a complete, auditable lifecycle for raffles across EVM-compatible networks. The lifecycle guarantees fairness from creation to settlement and addresses edge cases that are common in centralized or manual systems.

### 6.1 Creation

- A raffle contract is deployed with parameters such as prize asset, ticket cost, total ticket supply, closing time, and settlement rules.
- Parameters become fixed at deployment to prevent condition changes after ticket sales begin.
- Raffle modes include paid entry and free entry. Free entry requires participants to cover network gas fees.

#### PHASE 1 Raffle Initialization

The owner or authorized starter calls `startRaffle()` to begin a new raffle round. All parameters are validated on-chain before the raffle becomes active.

REQUIRED PARAMETERS	ON-CHAIN VALIDATIONS
→ <code>_prizeAmount</code> : Total ETH prize pool. Must be evenly divisible by winner count. Sent in Wei.	✓ No active raffle — only one raffle can run at a time
→ <code>_duration</code> : Raffle duration in minutes. Timer starts immediately when <code>startRaffle()</code> is called. Based on <code>block.timestamp</code> .	✓ Prize amount > 0 and evenly divisible by winner count
→ <code>_numberOfWinners</code> : Number of winners to select. Must be $\geq 1$ . Processed in batches of 25 via batch processing.	✓ Contract balance $\geq$ prize + previous unclaimed prizes
→ <code>_maxParticipants</code> : Player capacity. Must be greater than <code>numberOfWinners</code> and $\leq 10,000$ .	✓ Max participants > winners and $\leq 10,000$ limit
	✓ Caller must be owner or authorized starter address

### 6.2 Funding and Ticketing

- Participants purchase or claim tickets on-chain using supported tokens or native currency.
- All purchases or claims are recorded on-chain as verifiable receipts.
- For paid entry raffles, funds are escrowed in the contract until settlement.

#### PHASE 2 Player Entry & Automatic Triggers

Players call `getFreeTicket()` to claim their free entry. The contract automatically monitors two conditions and triggers VRF when either is met.

🔔 TRIGGER A: Max Participants	🕒 TRIGGER B: Time Expiry
When the last available slot is filled, the contract automatically sends a randomness request to Chainlink VRF within the same transaction. If <code>maxParticipants</code> is reached when a player calls <code>getFreeTicket()</code> , <code>_requestRandomWords()</code> is triggered. <pre>if (players.length <math>\geq</math> maxParticipants) → _requestRandomWords()</pre>	When the raffle duration expires and enough players have joined ( $\geq$ winners + 1), Chainlink Automation detects the condition via <code>checkUpkeep()</code> . Once met, <code>performUpkeep()</code> automatically sends the VRF request. <pre>if (timeUp <math>\&amp;\&amp;</math> enoughPlayers) → performUpkeep(ACTION_REQUEST_VRF)</pre>

## 6.3 Randomness Request

- At closure, the contract initiates a randomness request to Chainlink VRF.
- The request and related details are visible on-chain.
- Chainlink VRF returns a random value with a cryptographic proof, which is verified by the contract.

**PHASE 3 Chainlink VRF 2.5 Integration**

The contract requests cryptographically secure randomness from Chainlink VRF 2.5. No party — including the oracle — can predict or manipulate the outcome.

- 1 Request Random Words**

The contract calls `requestRandomWords()` with subscription ID, key hash, and callback parameters. The request is logged on-chain with a unique ID. 3 block confirmations are required.

  - Confirmations: 3 blocks
  - Callback Gas: 500,000
  - Random Words: 1
- 2 Fulfill Random Words**

The Chainlink oracle generates and delivers the random word via callback. The `fulfillRandomWords()` function stores the result. Lightweight callback — winner processing happens separately for gas optimization.
- 3 Seed Expansion**

A single VRF word is expanded via keccak256 hashing — generating a unique random value for each winner. Deterministic: the same seed always produces the same result. Verifiable.

```
keccak256(abi.encodePacked(
  randomSeed, currentIndex
))
```

## 6.4 Winner Selection

- The verified random value is deterministically applied to the participant pool.
- Winners are selected according to the configured rules.

**PHASE 4 Winner Selection — Fisher-Yates Shuffle**

Once randomness is available, Chainlink Automation calls `performUpkeep()` to process winners. The Fisher-Yates partial shuffle ensures fair, unbiased selection with  $O(1)$  gas per winner.

ALGORITHM STEPS	BATCH PROCESSING
<b>01</b> Start from current winner index position	To stay within gas limits, winners are processed in batches of 25. Chainlink Automation keeps calling <code>performUpkeep()</code> until all winners are selected. Each batch is a separate transaction.
<b>02</b> Generate unique random via <code>keccak256(seed, index)</code>	Max per batch: <b>25 winners</b>
<b>03</b> Pick random index from remaining players	Max participants: <b>10,000</b>
<b>04</b> Swap selected player to winner position	Processing: <b>Fully Automated</b>
<b>05</b> Record winner and auto-transfer prize	

## 6.5 Settlement

- The contract transfers prizes or pooled funds to winner addresses automatically.
- Settlement produces on-chain transaction hashes as delivery proof.

**PHASE 5 Prize Distribution & Pull Payment**

Prizes are automatically transferred to winners as they're selected. If a transfer fails, the prize is stored for manual claiming — ensuring no funds are ever lost.

PUSH PATTERN	PULL PATTERN (FALLBACK)	RESCUE MECHANISM
Each time a winner is selected, the contract automatically transfers ETH via <code>winner.call(value: prizePerWinner)()</code> . Successful transfers happen instantly and emit the <code>WinnerSelected</code> event. <pre>winner.call{value: prizePerWinner}() + success: emit WinnerSelected + fail: unclaimedPrizes[winner] += amount</pre>	If a transfer fails (contract wallet, gas issues, etc.), the prize is stored as unclaimed. The winner can withdraw at any time via <code>claimPrize()</code> . Follows the Checks-Effects-Interactions pattern. <pre>claimPrize() + unclaimedPrizes[winner] = 0 + ETH transfer to winner</pre>	Broken contract addresses cannot receive ETH or claim. The owner can rescue these funds via <code>rescueUnclaimedPrize()</code> — no funds are ever permanently locked. Safe fund management. <pre>rescueUnclaimedPrize(winner) + ETH rescued to owner</pre>

## **6.6 Transparency and Historical Data**

- Each stage is visible on-chain, including ticketing, VRF proofs, and settlements.
- Code and processes are fully auditable by partners and trusted Web3 cybersecurity firms through controlled assessments.
- Historical outcomes are available at <https://onchainwin.com/historicaldata>.

## **6.7 Edge Cases and Fail-Safe Mechanics**

- Insufficient ticket sales: If the minimum threshold is not met, automatic refunds are executed. If needed, refunds can be configured according to predefined policy for that raffle.
- Expiration without draw: If a raffle reaches the expiration condition, participant refunds are triggered. If needed, expiration handling follows the predefined policy for that raffle.
- Network or oracle disruption: Fallback logic prevents indefinite locking of funds and enables orderly resolution. If needed, emergency execution paths remain governed by on-chain controls.
- Threshold-free execution option: A customer may elect winner selection without any ticket threshold, subject to predefined policy and on-chain controls.

## **6.8 Multi-Chain Functionality**

- Lifecycle logic is consistent across Scroll, Base, and other EVM-compatible networks.

## **7. Security, Fairness and Auditability**

OnChainWin is designed to ensure that fairness is a verifiable outcome. Security principles, cryptographic randomness, and continuous auditability establish confidence for both participants and partner projects.

### **7.1 Fairness Through Chainlink VRF**

- Randomness for raffles is provided by Chainlink Verifiable Random Function.
- Each request produces a random value along with a cryptographic proof.
- Proofs are verified on-chain before being accepted by the raffle contract.

### **7.2 On-Chain Transparency**

- Ticket sales, randomness requests, randomness proofs, winner selection, and settlements are recorded on-chain.
- Historical data is published at <https://onchainwin.com/historicaldata>.

### **7.3 Security Controls**

- Automated Execution through contracts.
- Immutable State for deployed parameters.
- Fail-Safe Mechanics with refunds, expiration handling, and disruption fallbacks. If needed, refunds or emergency execution paths follow predefined policies and preserve on-chain auditability.
- Threshold-Free Execution configurable per raffle policy.

### **7.4 Auditability**

- Contracts and operational processes are fully auditable by integration partners and independent, reputable Web3 cybersecurity companies through controlled assessments.

### **7.5 Ongoing Monitoring**

- Real-time monitoring tools and dashboards allow continuous tracking of raffles in progress.

## **8. Business Model and Economics**

OnChainWin operates as a technology provider offering outsourceable raffle infrastructure for third-party projects. The model is transparent, scalable, and aligned with partner incentives.

### **8.1 Monetization Components**

- Subscription Access: Annual service plans.
- Transaction Service Fee: Percentage fee applied to ticket sales in paid entry raffles.
- Prize Commission: Commission applied to the prize amount at settlement for both free entry and paid entry raffles.
- Direct Utility Option Revenue Share: For partners that run raffles as a direct utility inside their own product, partners recognize primary revenue while OnChainWin participates via a negotiated revenue share.

### **8.2 Free Entry Raffles**

- Participants only cover gas fees.
- Prize commission and, where applicable, Direct Utility Option revenue share provide sustainable economics.

### **8.3 Partner-Oriented Economics**

- Flexible mix of subscription, transaction fee, prize commission, and revenue share.
- Cost avoidance compared to building in-house.
- Faster time to market through modules and onboarding resources.

### **8.4 On-Chain Transparency of Flows**

- Fees, prize commissions, and revenue shares can be calculated and distributed on-chain.

### **8.5 Long-Term Sustainability**

- Diversified income streams support development, security reviews, and multi-chain expansion.
- Roadmap to an in-house VRF strengthens the economic moat.

## 9. Developer Integration and Developer Experience

OnChainWin is designed for seamless adoption across EVM-compatible networks. The developer experience emphasizes flexibility, fast onboarding, and integration options that scale from basic use cases to fully customized implementations.

### 9.1 Direct Contract Interaction

- Contracts deployed on Scroll and Base, portable across EVM-compatible chains.
- Direct ABI interaction for configuration and lifecycle operations.

### 9.2 SDKs and Helper Packages

- Pre-packaged SDKs for common workflows such as ticketing, randomness requests, and settlement.

### 9.3 API and Webhook Options

- APIs and webhooks for off-chain UX, notifications, and analytics.

### 9.4 Guided Onboarding for Base Network

- Onboarding kit at <https://onchainwin.com/docs> for safe testing before mainnet.

### 9.5 Ease of Use for Non-Technical Partners

- Managed integration pathways and pre-built modules for operators without smart contract expertise.

### 9.6 Extensibility

- Additional modules such as subscription raffles, recurring rewards, and loyalty incentives.

## 10. Usecases and Case Studies

OnChainWin contracts are in active use by third-party projects. The examples below illustrate how transparent raffles and reward mechanics are embedded into external ecosystems.

### 10.1 Partner: NFT Raffles Powered by OnChainWin

- Context: Partner runs community raffles for NFT holders and newcomers.
- Integration: Partner operates with OnChainWin for secure raffles, instant rewards, and on-chain fairness.
- Outcomes: Verifiable randomness, on-chain ticketing and settlement, and flexible support for both free-entry and paid-entry raffles.
- Link: <https://Partner.xyz>

### 10.2 Partner: Quest-Based Participant Rewards

- Context: Partner operates quest campaigns where users complete tasks and receive on-chain rewards.
- Integration: OnChainWin powers participant selection and reward distribution through a verifiable mechanism.
- Link: <https://app.Partner.com/quest/17>

### 10.3 Additional Ecosystem Applications

- Creator Campaigns
- Gaming Economies
- NFT Utilities
- Loyalty programs

### 10.4 Quantitative Metrics and Verification

- Publicly auditable lifecycle records at <https://onchainwin.com/historicaldata>
- Metrics categories: number of completed raffles by network, ticket volume and unique participant wallets, settlement success rate and median settlement time, share of free-entry vs paid-entry events, third-party integrations.

# 11. Ecosystem, Partnerships and Compliance

OnChainWin operates within a modular Web3 ecosystem that combines EVM-compatible networks, verifiable randomness providers, and partner platforms that embed raffle utilities into their products.

## 11.1 Network Alignment

- EVM compatibility with live deployments on Base and Scroll.
- Chain services are abstracted within integration tooling to support consistent behavior across networks.

## 11.2 Randomness and Infrastructure Partners

- Chainlink VRF supplies randomness. Long-term vision includes a proprietary VRF.

## 11.3 Partner Models

- Direct Utility Integration with partner primary revenue and OnChainWin revenue share.
- Subscription and service models with transaction service fees and prize commissions.

## 11.4 Developer and Community Enablement

- Onboarding kit for Base developers at <https://onchainwin.com/docs>
- Historical data at <https://onchainwin.com/historicaldata>
- ABIs, SDKs, APIs, and webhooks for varying technical needs.

## 11.5 Compliance and Responsible Use

- Raffle modules designed as technology utilities for engagement and transparent distribution.
- Partners remain responsible for complying with regional rules for promotions, prizes, eligibility, disclosures, and taxes.
- Access controls can support eligibility checks or geofencing where required.
- Full auditability supports internal governance and external review.

# 12. Roadmap and Contact

## 12.1 Roadmap

### 2024: Contract Foundation and Scroll Deployment

- Design and build of the raffle contract suite covering creation, ticketing, VRF request, winner selection, and settlement.
- Production deployment on Scroll network.
- Initial operational learnings from on-chain activity and early community usage.
- Partnership engagement with an additional Scroll-based project to validate integration paths.

### 2025: Base Migration, Outsource Model, and Live Integrations

- Expansion to Base network and operational migration where relevant.
- Introduction of the outsource integration model for partners running raffles as direct utilities.
- Launch of the Base Onboarding Kit for developers at <https://onchainwin.com/docs>.
- Live third-party integrations including Partner raffles and Partner quest-based rewards.
- Business model expansion to include subscription access, transaction service fees, prize commissions, and Direct Utility Option revenue share.
- Chainlink VRF refinements and configuration updates informed by partner feedback.
- Unofficial partner reviews and operational assessments for additional assurance.

### 2026: Governance, Audits, Multi-Network Scale, and Community Growth

- Execution of company legal framework.
- Formal audits by independent third-party Web3 cybersecurity firms.
- Broader availability on additional EVM-compatible networks.
- Growth of the OnChainWin community through structured programs, transparency reports, and collaborations.

### 2027: OCW Token and Proprietary VRF

- Introduction of the OnChainWin token aligned with transparent, utility-driven incentives.
- Release of a proprietary OnChainWin Verifiable Random Function to increase independence and optimize randomness for incentive distribution at scale.

## 12.2 Contact

- Website: <https://onchainwin.com>
- Documentation: <https://onchainwin.com/docs>
- Historical Data: <https://onchainwin.com/historicaldata>
- Email: [e.kaya@onchainwin.com](mailto:e.kaya@onchainwin.com)
- X: <https://x.com/onchainwinx>
- Medium: <https://medium.com/@onchainwin>
- Discord invitation: [discord.gg/SufjnfDkN6](https://discord.gg/SufjnfDkN6)