cHaOSneT

Executive overview

The inability of projects and protocols to test their code, economics, or predict unforeseen events in production-like settings is the main cause of lost funds in the Web3 ecosystem today. cHaOSneT is a new tool that unlocks the ability for development teams to create a private testnet environment that mirrors the state and activity of multiple mainnet chains at once, replicating the dynamics of production better than other existing tools.

Our tool makes it possible for teams to fully integrate their applications with the tools and dependencies they expect to utilize in production, and obtain real-world insights of how their project will respond under those conditions. cHaOSneT lets teams have as close to “testing-in-prod” as you can get without risking any user funds in the process.

What is cHaOSneT?

cHaOSneT is a testnet-as-a-service platform that provides private testnets which host a configurable ecosystem of autonomous agents, or "bots", each intricately designed to mimic real-world behaviors of production applications on various chains within the network's environment. In traditional local blockchain testing, all behavior is user-driven, providing no network activity out of the box or any way to simulate user interaction. Publicly accessible testnets have activity from a multitude of users, but since that activity is not economically incentivized, it tends to be random and not very representative of the production environment. In contrast, Chaosnets are characterized by a level of activity that closely mirrors the live production environment by replicating behavior in the models that our agents use. This realism allows developers to stress-test their applications and deployments against a wide array of scenarios very easily without an enormous amount of setup, leading to robust and resilient testing environments for all types of Web3 applications and use cases.

Who is it for?

Developers or Security Auditors looking to design, test, and refine smart contract systems, application frontends, monitoring/support infrastructure, response/upgrade procedures, or any other components of their production deployments can benefit from realistic, production-like environments.

Using cHaOSneT as a simulation environment, they can gain valuable insights into how their protocols and infrastructure might perform under different market conditions, potential black swan events, or complex user behaviors, helping to create more robust and efficient applications.

Why did ApeWorX make it?

Traditional testnets offer a valuable testing ground, but they often fall short in replicating the complexity and unpredictability of real-world network conditions. Static simulations provide only a limited range of test conditions, leaving applications potentially unprepared for the full range of scenarios they could encounter in a live network. With cHaOSneT, ApeWorX aimed to bridge this gap. The AI-powered, autonomous agents within our chaosnets emulate the behaviors of real-world network participants. They learn, adapt, and execute actions that mirror the unpredictability of a live network, providing a dynamic and diverse range of testing conditions while also not risking any funds in the process.

Use Cases for cHaOSneTs

Use Case 1: DeFi Protocols

Defi protocols aim to provide secure and efficient services to their users. Over the years it has become quite obvious that there is not enough testing for teams to be confident their contracts are safe to use. By adding the ability to recreate mainnet activity they can test for advanced attacks like Oracle Manipulation, Flash Loan Exploits, and Collateral Valuation failures which will provide more confidence for their users and themselves in their code.

DeFi applications such as Yearn are often highly-connected with other DeFi protocols and chains, and can utilize cHaOSneT to reduce the testing environment setup that's needed to test the full integration of their application with other applications and actors that interact with their system.

Use Case 2: Bridges and Cross-chain Applications

When testing cross-chain applications today using protocols such as Connext, many developers have to rely on extensive mock testing in order to have a local testnet environment suitable for validating the application logic they are developing. However once ready for a full-scale integration test, their only option is to switch to testing in a production setting since many public testnets either don’t provide the same deployments that they require, or are under-supported in practice for the use cases they need to test against.

cHaOSneT allows those developers to provision a private environment set up for testing multi-chain interactions, including the bridging infrastructure that relays messages between chains, in similar fashion as they would experience during production use. The availability of this type of environment helps teams avoid issues in production settings, and can be used as a canary deployment that they can continuously leverage throughout the production lifecycle of their application.

Use Case 3: Evaluating Security Detection/Response Procedures

Developing security detection infrastructure and threat response to procedures, hacks and other protocol issues is a necessity for nearly all production teams.

However, without an environment that enables teams to test these monitoring tools and response procedures, it's impossible to tell if there are issues with them until it is too late.

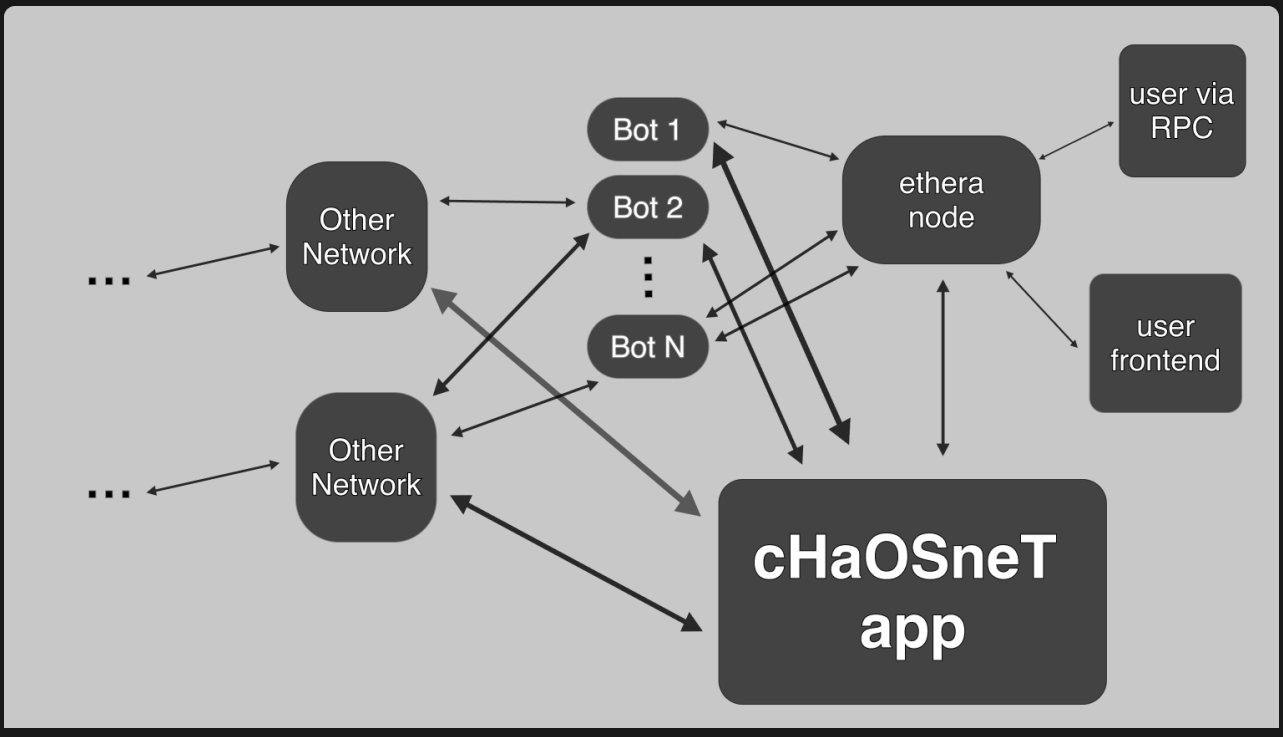
cHaOSneT unlocks the ability to provision canary deployments including all your security detection tools, which allows teams to conduct practice drills of responding to threats against their applications and evaluate their responses, without truly being at risk during the process.

Use Case 4: Testing Economic Validity of Protocols

With the collapse of Terra the need for testing the economic validity of a protocol has become an essential component of modern DeFi testing. However, tools for this type of testing are seriously lacking or otherwise too high-level to be used with production contracts. With cHaOSneT it is possible to create a complex agent-based simulation using the real contract system showing the robustness (or fragility) of the economics of a protocol and demonstrate possible attacks on the protocol.

A protocol team could also deploy a Chaosnet as a canary deployment of their application, prior to mainnet launch or in parallel to it. The community could then compete to build different bot implementations that interact with the protocol in different and unexpected ways, creating a sort of “economic bug bounty competition” that helps evaluate the robustness of the underlying protocol while also rewarding engagement with the project itself.

Product Design



How cHaOSneT helps with Local Testing and Public Testnets:

In traditional local testing, all behavior is user-driven, providing no network activity out of the box. Publicly accessible testnets have activity from multiple users, but since that activity is not economically incentivized, it tends to be random and not very representative of the production environment. In test driven development based in Web3, traditional testnets offer a valuable testing ground, but they often fall short in replicating the complexity and unpredictability of real-world network conditions. Static simulations provide only a limited range of test conditions, leaving applications potentially unprepared for the full range of scenarios they could encounter in a live network.

How cHaOSneT provides Dynamic Simulations

In its current state, a single cHaOSneT is able to perform specific tasks in response to chain state based on pre-programmed or trained behavior. However, many Chaosbots with differing behavior can be composed together so that they react to each other and your own on-chain stimuli, providing a Dynamic Simulation

environment. Users will be able to further augment this dynamic environment by developing and training their own bots to operate based on their own set of desired behaviors. The combination of all of these behaviors together create chaotic scenarios in the cHaOSneT environment, meaning users will get to expose themselves to unexpected situations and reveal unknown issues more easily in a safe testnet environment.

cHaOSneT and how it uses Ape Framework

Building the Chaosbots that ship with cHaOSneT would not be an easy task, without the assistance of Ape Framework. Ape makes it easy to design and test the individual bots' behaviors, as well as bundle them into discrete containerized agents which can be deployed at massive scale. Further, what's helped our engineers build Chaosbots isn't just the best-in-class execution experience of using Ape, but the ability to extract large amounts of on-chain data to do data science and machine learning tasks more efficiently than other tools.

How does ApePay work with cHaOSneT?

ApePay is a newly developed Web3 SaaS payment system we developed to not need to rely on centralized payment providers, and helped us create a "Pay-as-you-go" model for cHaOSneT. This assures that users can start and terminate deployments at any time, and customize the runtime length of the deployment without having to worry about being stuck to specific SaaS tiers.

How does Silverback work with cHaOSneT?

Silverback is the backbone of cHaOSneT and allows us to integrate its entire toolkit within cHaOSneT to create the right experience for monitoring and managing your deployments. For example, Silverback allows us to productionize the behavior of Chaosbots and track if they are performing as close to the mainnet user behavior as possible. When the Chaosbots encounter unexpected scenarios that cause failure of their expected behavior, Silverback ensures that recovery or mitigation from those faults are handled as smoothly as possible vs. running a traditional Ape-based script (or using another Framework). Silverback's world-class bot running platform ensures a reliable and robust experience for cHaOSneT users.

Custom Bot Creation

Chaosnet is the first big step to help reduce risk, enhance security and improve the overall user experience in the modern DeFi and Web3 ecosystem. cHaOSneT comes with an ever-growing taxonomy of bots for different protocols, blockchains, and periods of history; all built using the Silverback SDK.

Users can also leverage the Silverback SDK to create their own bots that can easily integrate with the same platform that all our chaosnets use for hosting the bots, or even self-execute those bots on their own infrastructure. In the future, we want to make it even easier for developers to create their own unique behaviors for their favorite protocols using the Silverback SDK, and offer a revenue sharing program to encourage their development.

Conclusion

cHaOSneT provides the perfect sandbox for rigorous testing before deploying on mainnet, thereby minimizing the risks of vulnerabilities or unforeseen events from impacting your applications. The Silverback SDK additionally helps users create custom bots that can be added to their own chaosnet deployment, or use them in production on our upcoming platform.

It is important for developers and security auditors to conduct thorough testing, security audits, and ongoing monitoring to identify and address potential vulnerabilities to ensure the safety and integrity of the ecosystem. We hope that cHaOSneT will prove to be an invaluable tool on the path to prod, capable of solving all your pre-launch needs and issues.

How can I learn more about cHaOSneT?

Website: apeworx.io
ApePay GitHub: https://github.com/ApeWorX/ApePay
Silverback GitHub: https://github.com/ApeWorX/silverback
Apeworx Discord: https://discord.com/ApeWorX
Apeworx Twitter: https://twitter.com/ApeFramework