brink

Engineering Impact Report 2025

Table of Contents

Letter from the Executive Director

Bitcoin Core is the most widely used software for participating in the Bitcoin network. It validates every transaction, enforces every consensus rule, and connects tens of thousands of nodes into a resilient peer-to-peer system that secures trillions of dollars of value. There is no company behind it and no one is required to maintain it. The engineers funded by Brink are among the small number of people in the world who choose to do this work full time, and 2025 was one of their strongest years yet.

Brink funded eight Bitcoin Core engineers during 2025. They work in public on open source software, so anyone can see all that they've accomplished, but in the following pages we summarize their major achievements and share what they're excited about working on in 2026.

We lead this report with the projects our engineers contributed to, but we want to be clear that we focus on the quantity and quality of each engineer's code review and testing in addition to other contributions they made. While headline projects matter, the quiet, careful review and maintenance work that keeps Bitcoin Core secure is what makes those projects possible. As an organization, Brink always puts review, testing, security, and stability first and foremost. This work is the bedrock of Bitcoin’s monetary network and we've tried to reflect that in this report.

Mike Schmidt
Executive Director
Brink

The Work that Keeps Bitcoin Secure

The sections that follow describe each engineer's individual contributions in detail. But first, we want to highlight the broader categories of work that Brink engineers performed in 2025, work that is unquestionably good for Bitcoin as sound, reliable digital money.

Code Review. Every change to Bitcoin Core must be carefully reviewed by multiple qualified developers before it can be merged. Review is the project's primary defense against bugs, vulnerabilities, and unintended behavior reaching the software that tens of thousands of nodes rely on. An overlooked flaw in consensus code could cause a chain split. A missed vulnerability in networking code could allow an attacker to crash nodes or steal funds from connected Lightning wallets. In 2025, Brink engineers collectively left over 6,000 review comments on other developers' pull requests. Several of our engineers are among the most active reviewers in the entire project. Every one of those comments represents a line of defense between a code change and the Bitcoin network.

Testing and Bug Fixes. Brink engineers found and fixed security vulnerabilities in 2025, both in Bitcoin Core and in other Bitcoin software. Engineers independently discovered denial-of-service vulnerabilities, used fuzz testing to catch bugs on pull requests before they could be merged, and maintained infrastructure that continuously tests Bitcoin implementations around the clock. When bugs are found before they reach users, the network stays safe and node operators never have to worry. When vulnerabilities are found in production code, they are responsibly disclosed and fixed before they can be exploited. This is the kind of work that, when done well, is invisible to everyone except potential attackers.

Fuzzamoto. One of our engineers built and released Fuzzamoto, a novel framework for testing Bitcoin full nodes by running an actual node inside a virtual machine and sending it random sequences of network messages. Traditional fuzz testing tests isolated functions, while Fuzzamoto tests the software the way a real attacker would interact with it by using the network. Any bug it finds is a real bug reachable by a remote adversary. Fuzzamoto has already discovered serious bugs that no existing test would have caught, and the Quarkslab auditors (see below) called it "likely the most valuable path to pursue in order to trigger deeper and more complex bugs." For node operators and users, this means that Bitcoin Core is being tested against attack scenarios that were previously out of reach.

The First Independent Security Audit of Bitcoin Core. Brink sponsored and coordinated the first-ever third-party security audit of Bitcoin Core, conducted by Quarkslab over four months. Three security engineers reviewed Bitcoin Core's most critical components: the peer-to-peer networking layer, mempool, chain management, and consensus logic. The audit found no critical, high, or medium-severity issues, and produced improvements to Bitcoin Core's testing infrastructure. For the Bitcoin community, this audit serves as an important signpost that the code review and testing culture that Bitcoin Core developers have built over the years is working and independent experts have helped confirm it.

Release Management. A Brink engineer led the release process for every Bitcoin Core release in 2025 including seven releases across five maintained branches. He also merged 56% of all

pull requests to the project. Release management is unglamorous but critical as it is the process by which carefully reviewed code actually reaches the tens of thousands of nodes running on the Bitcoin network. Maintaining multiple older release branches ensures that users who haven't yet upgraded to the latest major version can still receive critical security/bug fixes through backported patches.

Build System. Bitcoin Core runs on every major platform and produces deterministically reproducible binaries, meaning anyone can compile the release and verify they get the exact same result, which minimizes trust in any individual developer. Brink engineers led the ongoing work to modernize and maintain the build toolchain, including shipping the first releases built with CMake, completing the migration to Qt 6, improving Windows support, and working to reduce Bitcoin Core's dependency on external libraries. Each external dependency is a potential attack surface and removing them makes Bitcoin more secure. This means that Bitcoin Core continues to be available and trustworthy on the hardware and operating system users choose to run.

Libsecp256k1. The libsecp256k1 library provides the cryptographic operations that secure every Bitcoin transaction including signatures, key generation, and related functions. Multiple Brink engineers contributed to this library in 2025, including taking over stewardship of the silent payments implementation, co-creating a shared Python library (secp256k1lab) for prototyping cryptographic proposals (already adopted by the FROST threshold signature and ChillDKG key generation BIPs), and elevating the library's CMake build system to official status. A libsecp256k1 maintainer described one Brink engineer as "a key contributor to the project" whose reviews "carry significant weight." Strengthening libsecp256k1 strengthens the cryptographic foundation that every Bitcoiner depends on.

Bitcoin Kernel. The Bitcoin Kernel (previously known as libbitcoinkernel) project is extracting Bitcoin Core's consensus validation logic, the code that actually enforces Bitcoin's rules, into a standalone library with a public API. This will make it possible for other Bitcoin software to use the exact same consensus code as Bitcoin Core, reducing the risk of accidental chain splits between different implementations. It also opens the door to a range of new tools including alternative node implementations, Electrum-style index builders, silent payment scanners, block analysis utilities, and script validation accelerators can all be built on top of the kernel library without needing to embed all of Bitcoin Core. Brink engineers became more central to this effort in 2025, contributing review, shaping the C API design, building Python bindings, and hosting workshops to grow the developer community around the project.

Building on Bitcoin's Existing Capabilities. Brink engineers worked on multiple projects that extended the benefits of the schnorr/taproot upgrade to real users. MuSig2 wallet support, which multiple Brink engineers helped review into Bitcoin Core, enables more efficient and private multisignature transactions, allowing groups of signers to produce a single compact signature that looks identical to an ordinary payment on the blockchain. Silent payments, a privacy-enhancing reusable address format being implemented in libsecp256k1 by a Brink engineer, will allow recipients to publish a single address that generates unique on-chain outputs for each payment, with no visible link between them. SwiftSync, prototyped by a Brink engineer, demonstrated a 5x speedup of initial block download, potentially reducing the time for a new user to sync a full node from days to hours. Work on FROST threshold signatures will allow for flexible multisignature schemes that go

beyond what MuSig2 offers, enabling setups like 3-of-5 key arrangements. CISA could enable multiple signatures within a single Bitcoin transaction to be aggregated, reducing transaction sizes and fees. And the secp256k1lab library is streamlining how new cryptographic proposals are developed and reviewed, reducing friction for anyone building on Bitcoin's cryptographic primitives. All of this work makes Bitcoin more useful, more private, and more accessible for everyday users.

Bitcoin Core is a collaborative open source project, and every achievement described in this report was made possible in part by the contributions of developers outside of Brink including fellow reviewers, co-authors, maintainers, researchers, and the many contributors who give their time to strengthen Bitcoin. We are grateful to all of them.

2025 Developer Highlights

We've included a selection of highlighted contributions for each engineer we fund below. For details about each developer's work, see later in this report. Engineers are listed in alphabetical order by last name.

Marco De Leon completed his transition from Brink fellow to full-time grantee, establishing himself as a capable contributor across multiple areas of the codebase. He drove the txid type safety project to completion across six pull requests, successfully removed Bitcoin Core's legacy checkpoints, left nearly 250 review comments, and contributed to fuzz testing infrastructure and P2P code review.

Sebastian Falbesoner upgraded to full-time work in 2025 and had a prolific year across both Bitcoin Core and libsecp256k1. He took over stewardship of the silent payments implementation in libsecp256k1, left over 570 combined review comments across both repositories, co-created the secp256k1lab Python library for cryptographic BIP prototyping, implemented the first proof-of-concept for SwiftSync (achieving a 5x speedup of initial block download), reviewed MuSig2 wallet support through to its merge, and contributed to the FROST threshold signature BIP.

Michael Ford continued his role as one of Bitcoin Core's most senior maintainers, merging 56% of all changes to the project, helping to cut seven new releases, and leaving over 1,600 review comments. He led the release process for versions 28.1 through 30.0 and continued driving the effort to modernize Bitcoin Core's build toolchain and reduce dependencies, directly supporting deterministic builds and the project's long-term security.

Niklas Gögge built and released Fuzzamoto, a novel framework for fuzz testing Bitcoin full nodes through their external interfaces, which has already discovered serious bugs that existing tests would not have caught. He open-sourced the continuous fuzzing infrastructure, assisted Quarkslab in conducting the first-ever independent security audit of Bitcoin Core, mentored two interns on security-focused projects, and continued his work on the Bitcoin Core security group processing advisories and triaging vulnerability reports.

Fabian Jahr continued championing the embedded ASMap project, which improves Bitcoin Core's resistance to network-level attacks by ensuring nodes connect to a diverse set of internet providers, moving it toward likely inclusion in a future release. He advanced cross-input signature aggregation (CISA) research significantly, implementing a full aggregation signature scheme in libsecp256k1. He contributed reviews across Bitcoin Core, libsecp256k1, and the ASMap toolchain, and was recognized as a subject matter expert for several areas of the codebase.

Eugene Siegel joined Brink as a grantee in April 2025 having already made an impact by independently discovering and helping fix a disk-filling vulnerability in Bitcoin Core's logging system. His log rate-limiting fix, shipped in Bitcoin Core v30.0, prevents a class of denial-of-service attack that could have allowed an attacker to fill a node's disk by sending invalid blocks. He also contributed fuzz testing harnesses and scenario work for Fuzzamoto.

Hennadii Stepanov left over 1,800 review comments and continued his wide-ranging work on Bitcoin Core's build system, platform support, GUI maintenance, and translations. He completed the migration to Qt 6 (shipped in Bitcoin Core v30.0), re-introduced native Windows CI testing for cross-compiled binaries, elevated libsecp256k1's CMake build system from experimental to official status, and maintained the bitcoin-core-nightly CI repository that catches portability regressions across BSD, illumos, and Windows platforms.

Stéphan Vuylsteke left over 700 review comments and became a central contributor to the Bitcoin Kernel project, the effort to extract Bitcoin Core's consensus logic into a standalone, reusable library. He built and maintained the Python bindings (py-bitcoinkernel), reviewed and shaped the C API design, contributed to validation logic and codebase maintainability improvements, and continued hosting London BitDevs and contributing to Bitcoin Optech.

Marco De Leon

@marcofleon

Marco joined Brink in August 2024 as a fellow, mentored by Niklas Gögge, after completing the Chaincode Seminars and Bitcoin FOSS program. His fellowship focused on fuzz testing including expanding coverage, adding useful oracles, and identifying vulnerabilities. By the end of 2025 he had completed his fellowship and transitioned to a full-time Brink grantee, having demonstrated both the technical skill and the collaborative instincts that the project needs from its contributors.

In 2025, Marco opened 11 pull requests (all merged), left nearly 250 review comments, and contributed across refactoring, testing, fuzzing, and P2P areas of the codebase.

Review and Testing

> *"I can provide more value to my peers in the project by tackling some of the harder to review PRs. I've tried to shift to this mindset more in the last few weeks or so and I can already see the benefits for myself and hopefully for the project as well." - Marco*

Marco's reviews covered P2P code, fuzzing infrastructure, and validation logic. Among the larger pull requests he reviewed were the TxOrphanage revamp, a significant improvement to how Bitcoin Core handles orphaned transactions to protect against denial-of-service attacks, and Pieter Wuille's spanning forest linearization algorithm for cluster mempool, one of the most technically demanding changes to Bitcoin Core's mempool in recent years. Colleagues noted that his reviews on both pull requests were valuable and helped move them toward being merged.

His growth as a reviewer was visible throughout the year. As he described it: "It was some time in September that something clicked and I found it much easier to maintain my mental model of the network and the codebase and how it all connects. The most recent CoreDev was the first one where I didn't feel lost at all."

Transaction Identifier (txid) Type Safety

Marco's most visible project in 2025 was driving the txid type safety effort to completion. Bitcoin transactions have two identifiers: a txid and a wtxid (witness transaction ID). These serve different purposes in the codebase, but historically both were stored as the same generic `uint256` type, meaning the compiler couldn't catch cases where code accidentally used a txid where a wtxid was expected, or vice versa. This kind of type confusion can lead to subtle bugs, and Marco's work proved that it already had. Stronger type safety makes Bitcoin Core's codebase more resistant to this class of error going forward, benefiting every user who depends on the software's correctness.

Across six pull requests touching many files throughout the codebase, Marco converted generic `uint256` usage to distinct `Txid` and `Wtxid` types. In the process, he found and fixed an actual bug in the ephemeral spends checking logic where a txid was being used where a

wtxid was required. The project was a systematic, careful effort that gave Marco a wide overview of how Bitcoin Core's different subsystems interact.

Removing Checkpoints

Marco authored the pull request to remove Bitcoin Core's legacy checkpoints, hardcoded block hashes that the software historically used to prevent nodes from being fed fraudulent chains during initial sync. This removal was made possible by the headers presync logic introduced in earlier versions, which provides a more robust and dynamic defense against low-work chain attacks.

While conceptually straightforward, the change required Marco to deeply understand the headers sync mechanism and verify through testing that it provided adequate protection without checkpoints. The removal was also a small philosophical win as hardcoded checkpoints had long been criticized as an unnecessary trust assumption in Bitcoin Core, and their removal makes the security model cleaner and more transparent for node operators.

Press:

- Bitcoin Optech Newsletter #392 Podcast

Fuzzing and Security Work

Marco contributed to fuzz testing throughout the year, including work on improving fuzzing stability and determinism. He opened a fuzz harness for the coins database and contributed to LevelDB differential fuzzing, testing Bitcoin Core's database layer against a reference implementation to look for discrepancies.

He also became a contributor to Niklas Gögge's Fuzzamoto project (described in detail in Niklas's section of this report), adding support for compact block scenarios and helping extend the framework's input generation capabilities.

Plans for 2026

Marco plans to take on maintenance of Brink's continuous fuzzing infrastructure and focus on network connection management, specifically, helping increase the number of outbound connections a Bitcoin Core node can make to improve network robustness. He'll continue contributing to Fuzzamoto and deepening his expertise in Bitcoin Core's P2P layer.

Sebastian Falbesoner

@theStack

Brink has been funding Sebastian since 2021, initially for part-time review-focused work. In 2025, he upgraded to full-time, a transition that has proven to be enormously productive. Sebastian is one of those rare contributors who is equally effective working on Bitcoin Core's Python test framework, reviewing wallet code, and diving into the low-level elliptic curve mathematics of libsecp256k1. In 2025, he left 350 review comments in Bitcoin Core and 222 in libsecp256k1, opened 35 pull requests in Bitcoin Core (all merged) and 17 in libsecp256k1 (14 merged).

On behalf of our sponsors, Brink is pleased to continue funding Sebastian's work across both Bitcoin Core and libsecp256k1.

Review and Testing

Sebastian's reviews spanned an unusual breadth. In Bitcoin Core, he provided thorough review of the MuSig2 wallet implementation including the set of pull requests that brought multisignature spending support using schnorr signatures into Bitcoin Core's wallet for the first time. His familiarity with the underlying libsecp256k1 musig2 module allowed him to verify that the wallet code correctly implemented the signing workflow and, critically, avoided nonce reuse that could leak secret key material. MuSig2 wallet support was merged in late October 2025 and will be available in the upcoming v31.0 release.

He also provided an in-depth review of the TxOrphanage improvements, which redesigned how Bitcoin Core limits the resources consumed by orphaned transactions. His peer noted that his review was "thorough and helpful" and that his quick re-reviews helped the pull request maintain momentum through to its merge.

Jonas Nick, a libsecp256k1 maintainer, described Sebastian's impact on that project:

> *"@theStack's impact on libsecp256k1 has been significant and he has become a key contributor to the project. He consistently demonstrates the attention to detail required for cryptographic library development and his code is high-quality. Therefore, his reviews of other pull requests carry significant weight. He works effectively with very little direction. When he encounters design questions, he proactively writes code or benchmarks to help resolve them."*

Silent Payments

One of Sebastian's most significant responsibilities in 2025 was taking over stewardship of the silent payments implementation in libsecp256k1 after the original author stepped away in October. silent payments (BIP352) is a new type of Bitcoin address that can be reused for multiple payments without creating any on-chain link between those payments, a significant privacy improvement over most existing address types.

Implementing silent payments in libsecp256k1 is essential because the library's high security standards (including resistance to side-channel attacks) make it a trusted foundation that wallet developers can build on with confidence. However, the effort found a serious design challenge with a "worst-case scanning attack" was discovered that could stall a node for several minutes when processing pathological transactions designed to maximize the computational cost of scanning for silent payment outputs.

> *"I'm glad that this issue was discovered before the code was merged and adopted by wallets. We don't want to release code that potentially opens a denial-of-service vulnerability." - Sebastian*

Sebastian and other reviewers spent significant time researching and benchmarking two alternative scanning approaches with different trade-offs for different use cases. He opened

pull requests for both approaches and expects to bring the silent payments module to completion in 2026.

Press:

- Bitcoin Optech Newsletter #385: 2025 Year-in-Review Special
- Bitcoin Optech Newsletter #392 Podcast

SwiftSync

One of 2025's most exciting developments for Bitcoin Core usability originated from a conversation at a CoreDev meeting, where Ruben Somsen proposed a technique to dramatically speed up initial block download (IBD), the process a new node goes through to validate the entire blockchain from scratch. Sebastian heard the proposal, understood its elegance, and simply went and built it.

The result was the first proof-of-concept implementation of SwiftSync, which achieved a 5.28x speedup of IBD, reducing the initial sync time from roughly 41 hours to about 8 hours. SwiftSync works by using a pre-generated "hints file" that tells a syncing node which transaction outputs will still be unspent at a target block height. Instead of writing millions of entries to the UTXO database only to delete them later, the node skips those intermediate entries entirely and uses a cryptographic accumulator to verify correctness at the end.

SwiftSync has generated significant excitement in the community and a Rust implementation was announced later in the year. A pull request to Bitcoin Core is now open for review.

Press:

- SwiftSync speedup for initial block download - Bitcoin Optech Newsletter #349
- Discussion with Sebastian Falbesoner and Ruben Somsen - Bitcoin Optech Podcast #349

secp256k1lab and FROST

Sebastian co-created secp256k1lab with Jonas Nick and Tim Ruffing, a Python library that provides a unified, reusable implementation of secp256k1 elliptic curve operations for use in BIP reference code and prototyping. Previously, every cryptographic BIP had to include its own custom Python implementation of these primitives, leading to subtle inconsistencies and making review unnecessarily difficult. The library was announced in April and has already been adopted by the FROST signing and ChillDKG BIP drafts.

Sebastian also contributed to the FROST threshold signature effort, which extends the MuSig2 work to enable t-of-n multisignatures (rather than just n-of-n). He contributed to the ChillDKG distributed key generation BIP's Python reference implementation and helped integrate secp256k1lab as a shared dependency across the FROST BIP repositories. The FROST signing BIP draft was formally opened in the BIPs repository during 2025.

Press:

- secp256k1lab announcement - Bitcoin Optech Newsletter #348

- Secp256k1lab: An INSECURE Python Library That Makes Bitcoin Safer - Bitcoin Magazine

Other Notable Work

Sebastian published a historical benchmark comparing ECDSA signature verification performance between OpenSSL and libsecp256k1 over the past decade, demonstrating that the performance gap has widened from 2.5–5.5x to over 8x, a testament to the ongoing optimization work by libsecp256k1 contributors. He also participated in the Crypto Camp event, an intensive program focused on provable cryptography and libsecp256k1 internals and authored the Libsecp256k1 article for Bitcoin Magazine's "The Core Issue".

Press:

- Comparing performance of ECDSA signature validation - Bitcoin Optech Newsletter #379
- The Core Issue: Libsecp256k1, Bitcoin's Cryptographic Heart - Bitcoin Magazine

Plans for 2026

Sebastian's top priority for 2026 is getting the silent payments module merged in libsecp256k1, followed by the DLEQ proof module that allows wallets to verify correct derivation of silent payment outputs. He also plans to review BIP54 (the "Consensus Cleanup" soft fork proposal), continue his work on FROST, and contribute review to the libevent removal effort.

Michael Ford

@fanquake

A Bitcoin Core contributor since 2012 and project maintainer since 2019, Michael is one of Brink's most senior developers. In 2025, he merged over half of all changes to the Bitcoin Core project, helped cut seven releases, and left over 1,600 review comments. He is often the first to comment on a new pull request and serves as the project's institutional memory. Colleagues have praised his encyclopedic knowledge of everything happening across the codebase.

On behalf of our sponsors, Brink is pleased to continue funding Michael's essential engineering work.

Review and Maintainer Duties

Michael merged 56% of all pull requests to Bitcoin Core in 2025 (continuing his track record from previous years). This involves maintaining a bird's-eye view of the entire project day-to-day, keeping tabs on what has been reviewed and by whom, what is ready to merge and what it may conflict with, and what the merge order of changes could be.

He helped lead the release process for every Bitcoin Core release in 2025: versions 28.1, 29.0, 28.2, 29.1, 30.0, 29.2, and 28.3. The two major releases shipped significant improvements for users and the network. Five maintenance releases (28.1, 28.2, 28.3, 29.1, 29.2) delivered backported bug fixes and security improvements to users on older versions. Michael

performed backporting and maintenance across five branches (master, 30.x, 29.x, 28.x, and 27.x), ensuring that users running older versions continue to receive fixes.

He also continued his work on Bitcoin Core's security efforts, engaging with CVE discussions and ensuring relevant fixes made it into releases. In 2025, this included coordinating the response to the disk-filling vulnerability (CVE-2025-54605), ensuring the log rate-limiting fix was included in Bitcoin Core v30.0.

Build System and Toolchain

All Bitcoin Core releases aim to be deterministically reproducible, meaning anyone who compiles a release version should be able to obtain identical executables. This is important to minimizing trust in the release manager or any particular developer. Michael's work on the build system, described below, directly supports this property.

Michael's most popular pull requests in 2025 were almost entirely build system and toolchain work, the kind of infrastructure that rarely makes headlines but is essential to the project's ability to produce secure, reproducible binaries that run across a wide variety of platforms. His focus areas included improving determinism in the Guix-based release build process (fixing issues where builds on different Linux distributions could produce slightly different outputs), upgrading the compiler toolchain from GCC 13 to GCC 14, and making progress toward fully static release binaries, which would reduce the project's dependency on system libraries and simplify deployment.

> *"A trustless, decentralised protocol for internet cash should exist, and Bitcoin remains our best shot at making it happen." - Michael*

The effort to reduce Bitcoin Core's external dependencies is a long-running theme of Michael's work and is directly motivated by security concerns. Every external library Bitcoin Core depends on represents a potential attack surface, whether through security vulnerabilities, supply-chain compromises, or reproducibility challenges. Brink published a blog post detailing the history and importance of this dependency reduction work, much of which has been led or supported by Michael.

Michael also contributed to other repositories beyond the main Bitcoin Core codebase, including bitcoincore.org, the Guix signatures repository, packaging scripts, maintainer tools, and downstream packaging for Homebrew and Docker.

Plans for 2026

Michael plans to continue maintaining Bitcoin Core and shipping releases. His primary technical goal is to deliver static binaries, modularizing the codebase, and working toward distinct executables that can be independently maintained and updated. This is closely related to the multiprocess project (see Stéphan Vuylsteke's section) and addresses a practical problem, for example when a bug is found in the wallet, the current monolithic binary means the entire release, including the node, may need to be rolled back, even if the node code is unaffected.

Niklas Gögge

@dergoegge

Niklas began contributing to Bitcoin Core before graduating in March 2022 and joined Brink full time shortly after. Since then, he's built a reputation as one of Bitcoin's most effective security engineers by finding vulnerabilities through testing, responsibly disclosing them, and then building infrastructure so that similar bugs can be caught automatically in the future. In 2025, he brought this work to a new level by releasing a novel fuzz testing framework, helping coordinate Bitcoin Core's first-ever third-party security audit, and mentoring the next generation of security-focused contributors.

In 2025, Niklas left over 200 review comments in Bitcoin Core and opened 3 pull requests (all merged), but these numbers understate his output as the majority of his work happened outside the main Bitcoin Core repository, in Fuzzamoto, the continuous fuzzing infrastructure, the security audit coordination, and mentorship.

On behalf of our sponsors, Brink is pleased to continue funding Niklas's security engineering work and his growing leadership role within the project.

Review

Of Niklas's 200+ review comments, the majority focused on fuzzing-related pull requests and security-sensitive areas of the codebase. His reviews frequently use fuzzing as a review tool. Rather than just reading code and reasoning about its correctness, he also runs it through Fuzzamoto or other fuzzing infrastructure to test whether a proposed change introduces unexpected behavior. This approach caught bugs on the cluster mempool and Erlay pull requests during 2025. He also reviewed work across a range of related projects including Fuzzamoto and the qa-assets fuzzing corpus repository.

Fuzzamoto

Niklas's most significant achievement in 2025 was building and releasing Fuzzamoto, a framework for fuzz testing Bitcoin full nodes through their external interfaces (P2P networking and RPC) rather than testing isolated internal components.

Traditional fuzz testing in Bitcoin Core works by taking individual functions or classes, feeding them quasi-random inputs, and checking for crashes or unexpected behavior. This is effective but can be limited in that it can only test components that have been carefully extracted and isolated, and many of Bitcoin Core's most security-critical code paths involve complex interactions between multiple subsystems that are difficult to test in isolation.

Fuzzamoto takes a different approach. It runs an actual Bitcoin Core node inside a virtual machine, sends the node sequences of P2P messages through the networking port, and uses snapshot fuzzing to reset the node's state between iterations. This means any bug it finds is a real bug reachable by a remote attacker, not an artifact of the testing setup.

> *“It tests production interfaces, so any bugs that surface are likely to be actual bugs reachable by a remote adversary. Nothing needs to be merged or maintained in Bitcoin Core for this to work. Any Bitcoin protocol implementation can easily be tested using the same approach.” - Niklas*

Fuzzamoto has already justified its existence by finding bugs that none of Bitcoin Core's existing tests would have caught, including a cluster mempool bug discovered on a pull request under review.

The framework is not limited to Bitcoin Core. Because it tests through external interfaces, it can be used to fuzz any Bitcoin protocol implementation and developers have already begun adapting the techniques for fuzz testing Lightning Network implementations.

Press:

- Fuzz testing tool for Bitcoin nodes - Bitcoin Optech Newsletter #350
- Discussion with Niklas Gögge on Fuzzamoto - Bitcoin Optech Podcast #350
- Quarkslab Bitcoin Core audit report: "Fuzzamoto is likely the most valuable path to pursue in order to trigger deeper and more complex bugs" - Quarkslab Blog

Security Audit

In 2025, Brink sponsored the first-ever independent third-party security audit of the Bitcoin Core codebase, conducted by Quarkslab and coordinated with the Open Source Technology Improvement Fund (OSTIF). Niklas served as the primary technical liaison from Brink, working directly with the Quarkslab engineers, including hosting them for a week in Brink's London office at the start of the engagement to familiarize them with the codebase and development practices.

Over a four-month period from May through September, three Quarkslab security engineers conducted a review focused on Bitcoin Core's most security-critical components: the peer-to-peer networking layer, mempool, chain management, and consensus logic. The auditors reported no critical, high, or medium-severity issues which is a strong validation of the code review and testing culture that Bitcoin Core developers have built over the years.

The audit also produced practical improvements to Bitcoin Core's testing infrastructure, including new fuzzing harnesses for block connection and chain reorganization scenarios and a virtual filesystem utility to speed up fuzz testing. Some of these contributions may be prepared for integration into the main Bitcoin Core repository.

Press:

- An Independent Security Audit of Bitcoin Core - Brink Blog
- Brink Funds First Third Party Security Audit of Bitcoin Core - Bitcoin Magazine
- Quarkslab Bitcoin Core Audit - Quarkslab Blog
- Bitcoin Core Audit Complete! - OSTIF Blog

Continuous Fuzzing Infrastructure and Security Group Work

Niklas open-sourced his continuous fuzzing infrastructure, Fuzzor, in 2025, after rewriting it from Python to Rust in 2024. This system runs fuzz tests 24 hours a day against Bitcoin Core, LND, Core Lightning, and LDK, automatically reporting crashes and generating coverage reports. Adding snapshot fuzzing support was a key milestone, though he notes this currently remains somewhat unstable for continuous automated use.

He continued his work on Bitcoin Core's security group, helping triage vulnerability reports, develop fixes, and publish advisories. Five security advisories were published in 2025 by the group. Niklas also improved the public-facing advisories page by adding point-of-contact details and more descriptive severity classifications.

Mentorship

Niklas mentored two interns in 2025 who worked on extending Fuzzamoto's capabilities adding support for address relay, BIP37, taproot scripts, and compact blocks in the framework's input representation. Both interns progressed faster than expected, and their work has directly expanded Fuzzamoto's ability to test more of Bitcoin Core's P2P surface area. One intern also began working on using Fuzzamoto to differentially fuzz libbitcoin, which could serve as a long-term testing oracle for Bitcoin Core's consensus behavior.

This builds on Niklas's track record as a mentor, previously guiding Marco De Leon through his fellowship year, and colleagues have noted that his mentorship skills are something Brink should continue to encourage.

Plans for 2026

Niklas plans to make Fuzzamoto more accessible to other Bitcoin Core contributors, including simplifying test composition using container-based workflows. He's exploring the development of a deterministic hypervisor, inspired by Antithesis's approach, that would enable fully deterministic execution for the fuzzer and reliable reproduction of even the rarest bugs. He'll also be writing tests for the Antithesis testing platform, which Brink has begun evaluating, and has already found 9 bugs in the first weeks on the platform.

Fabian Jahr

@fjahr

After a long history of volunteer part-time contribution to Bitcoin, Brink began funding Fabian full time in April of 2023. He has since established himself as someone who can drive complex, multi-year projects across both Bitcoin Core and libsecp256k1, from the practical (getting AssumeUTXO deployed to mainnet) to the research-oriented (advancing cross-input signature aggregation). In 2025, he left 665 review comments in Bitcoin Core and over 100 in libsecp256k1, opened 22 pull requests in Bitcoin Core (10 merged), and made significant contributions to the ASMap toolchain and CISA research.

On behalf of our sponsors, Brink is pleased to continue funding Fabian's work on Bitcoin protocol research and development.

Review

Fabian's review activity in 2025 extended across three codebases: Bitcoin Core, libsecp256k1, and the ASMap ecosystem. In Bitcoin Core, he was recognized by his fellow contributors as a "subject matter expert" on testnets, indexes, and addrman/ASMap, a designation that carries the expectation that he will pay close attention to pull requests touching those areas and provide timely, informed review.

His libsecp256k1 review work grew substantially in 2025, driven by the Crypto Camp he attended in the summer, an intensive program on provable cryptography and libsecp256k1 internals. He describes this as giving him "a huge jump in my understanding of the library," and the results were visible in his increasing engagement with the project. He contributed reviews to the silent payments module, the batch verification pull request, and several other efforts, attended a dedicated secp256k1 contributor meeting, and implemented pull requests aiding batch verification and CISA.

He also reviewed MuSig2-related pull requests and opened a follow-up that expanded test coverage contributing to the effort that brought MuSig2 wallet support into Bitcoin Core.

Cross-Input Signature Aggregation (CISA)

CISA remains Fabian's most ambitious long-term project. The goal is to enable multiple signatures within a single Bitcoin transaction to be aggregated into fewer (or even one) signature, reducing transaction sizes and fees. This matters for privacy because it creates a direct economic incentive for techniques like coinJoin and payjoin that combine multiple users' transactions together. Today these privacy-enhancing techniques cost extra in fees, but with CISA they would actually be cheaper than standard transactions.

Two forms of aggregation are being developed. Half aggregation, which Fabian has been championing through a BIP proposal and libsecp256k1 implementation, would reduce transaction sizes by roughly 8% in typical cases. Full aggregation would reduce them by roughly 16% in typical cases (or up to 40% in the best case), but requires a more complex signature scheme.

The major breakthrough of 2025 was the publication of the DahLIAS paper in April, the first interactive aggregate signature scheme compatible with Bitcoin's existing cryptographic primitives. Fabian reviewed the paper in depth, implemented it in Python, and then built a full implementation as a libsecp256k1 module, an achievement he describes as the one he is most proud of this year.

> *"I got a huge jump in my understanding of the library in the 6 weeks preparation phase as well as at the in-person event. The primary result is my secp256k1 implementation of full-agg based on DahLIAS, probably the achievement that I am most proud of this year." - Fabian*

He also discussed two potential versions of a transaction-wide CISA consensus change BIP at the October CoreDev meeting and received encouraging feedback. A collaboration with the payjoin project to build a proof-of-concept demonstrating CISA's privacy benefits is planned for early 2026.

Press:

- Cypher Tank non profit competition
- Discussion with Jonas Nick on DahLIAS - Brink Engineering Call

ASMap

ASMap is a project Fabian has championed since joining Brink, aimed at improving Bitcoin Core's resistance against network-level attacks. When a node connects to the Bitcoin network, an attacker who controls IP addresses across many different subnets could potentially trick the node into connecting only to the attacker's nodes, an "eclipse attack" that could be used to censor transactions or feed the node false information about the blockchain.

ASMap works by mapping IP addresses to the autonomous systems (ASes) that control them, allowing Bitcoin Core to ensure it connects to peers across a diverse set of network providers. The challenge has been creating these maps in a trustless way. Fabian developed tools and processes for multiple contributors to independently generate maps and verify they match, without trusting any single individual.

In 2025, the ASMap toolchain (Kartograf) that Fabian created and maintains saw 52 pull requests merged with over 150 commits and significant performance improvements. The asmap-data repository completed 5 successful collaborative map generation runs. Most importantly, the pull request to embed ASMap data directly into Bitcoin Core has progressed through extensive refactoring and review, and Fabian is cautiously optimistic it will be included in a future release.

> *I continue to be the sole 'maintainer' of the asmap-data and kartograf projects. While most PRs were not opened by me anymore this year, I have been reviewing and merging everything. - Fabian*

Press:

- Improved reproducible ASMap creation process - Bitcoin Optech Newsletter #290

Other Activities

Fabian continued hosting the Berlin BitDevs meetup on a monthly basis. He spoke at conferences including Bitcoin++ Privacy Edition in Riga and Bitcoin MENA, and helped organize the Frankfurt CoreDev meeting. He's proud that several regular BitDevs attendees have gone on to apply for developer grants or the Chaincode BOSS program.
He also continued to maintain a GitLab-based backup of the Bitcoin Core repository, a contingency that would allow the project to resume development within days if GitHub were ever to delist the project.

Plans for 2026

Fabian's top priority is completing a working proof-of-concept of CISA with payjoin and publishing all associated BIPs and implementation code for public review. He'll continue championing the embedded ASMap project through to its inclusion in a Bitcoin Core release. He also plans to contribute significantly to the libevent removal effort, review the silent payments implementation, and to deepen his contributions to libsecp256k1.

Eugene Siegel

@Crypt-iQ

Eugene joined Brink as a grantee in April 2025, bringing experience in Bitcoin Core's P2P and networking code. His grant focuses on fuzz testing, particularly improving coverage of Bitcoin Core's most security-sensitive message processing and validation code. In his first nine months, he made an outsized impact by independently discovering a security vulnerability and building the fix that shipped in Bitcoin Core v30.0.

In 2025, Eugene opened 7 pull requests in Bitcoin Core (6 merged), left over 300 review comments, and contributed 7 merged pull requests to Fuzzamoto.

Review

Eugene's reviews focused on areas adjacent to his fuzzing work: P2P networking, block validation, and fuzz testing infrastructure. Notable reviews included work on the block index tree fuzz target, compact block relay logic, and witness stripping detection. He also reviewed several pull requests related to making existing fuzz tests more deterministic, an important property for reproducibility and debugging.

Log Rate Limiting

Eugene's most significant contribution in 2025 was his work on mitigating disk-filling attacks against Bitcoin Core nodes. He independently discovered that an attacker could cause a victim node to fill up its disk space by repeatedly sending invalid blocks that triggered unconditional log messages which was the same vulnerability that Niklas Gögge had also found through different means (CVE-2025-54605).

Rather than just fixing the specific bug, Eugene designed and implemented a comprehensive log rate-limiting system that prevents this entire class of attack. His pull request (#32604) received 291 review comments, reflecting both the importance of the fix and the care required to get it right. A follow-up pull request (#33011) with 104 comments refined the approach further.

The fix was included in Bitcoin Core v30.0 and prevents future issues of the same type from occurring, not just the specific attack vector that was discovered. For node operators, this means one less category of attack to worry about. Rather than playing whack-a-mole with individual bugs, the log rate-limiting system closes off an entire class of potential vulnerabilities.

Press:

- CVE-2025-54605: Disk filling from invalid blocks - Bitcoin Core Security Advisory

Fuzz Testing

Eugene's primary grant focus is on improving fuzz test coverage of Bitcoin Core's net_processing code, the layer that handles incoming P2P messages from other nodes on

the network. This is among the most security-critical code in Bitcoin Core, as it directly processes untrusted input from the internet.

He wrote a targeted fuzz harness for compact block processing (#33300, 81 review comments), which tests the complex state machine that handles how nodes efficiently relay blocks to each other using only the transactions they're missing. Compact blocks involve intricate interactions between multiple peers and multiple internal data structures, making them an ideal candidate for fuzz testing.

> *"I learned all the ins-and-outs of the compact blocks code and went back through the first PRs introducing compact blocks to really understand why each line of code was there." - Eugene*

Eugene also contributed substantially to Fuzzamoto, Niklas Gögge's framework for fuzz testing Bitcoin nodes through their external interfaces. His seven merged pull requests to the project included documentation improvements, bug fixes, and the addition of a coinbase transaction generator and compact block support to the framework's input representation. He also explored integrating snapshot fuzzing directly into Bitcoin Core's testing infrastructure.

Plans for 2026

Eugene plans to focus more deeply on the mocking effort that would allow cleaner separation between Bitcoin Core's net_processing and validation layers. This is work that would enable more effective fuzz testing of the most critical code paths. He'll continue contributing to Fuzzamoto, including work on incremental snapshots that could significantly speed up the framework's testing throughput. He also plans to write fuzz tests for newer Bitcoin Core features like BIP153 template sharing and BIP157 compact filters.

Hennadii Stepanov

@hebasto

Hennadii has been contributing to Bitcoin Core since 2018 and became the project's GUI maintainer in 2021. Brink began funding him in 2021. During 2025, he left over 1,800 review comments and continued his extraordinarily wide-ranging work across the build system, platform support, GUI, translations, CI infrastructure, and libsecp256k1. Two major Bitcoin Core releases (v29 and v30) shipped in 2025 as the first to use the CMake build system that Hennadii spent years helping to develop, and both went extremely smoothly.

On behalf of our sponsors, Brink is pleased to continue funding Hennadii's work on Bitcoin Core's build infrastructure and platform support.

Review

Hennadii reviewed every change to Bitcoin Core's main build system and depends build subsystem in 2025, occasionally catching issues even after they were merged. His nearly 1,000 review comments on other authors' pull requests covered build system changes, CI configuration, the Bitcoin Kernel project, and GUI improvements. Nearly every review from

Hennadii starts with him actually running the code, something other reviewers sometimes skip but which occasionally reveals bugs that are obvious to a human yet hidden from automated testing infrastructure.

Sedited, the lead developer of the Bitcoin Kernel project, noted Hennadii's contributions to that project: "You've been very helpful with some of the build snags and the cleanups we landed in CMake made this all much easier."

Build System

The CMake-based build system, that Hennadii championed, had hundreds of pull requests involving a dozen contributors and two years of review and testing. It reached a major milestone in 2025 when both the v29 and v30 releases were built using CMake, and the transition was seamless. This was the culmination of years of work and determination on Hennadii's part to replace Bitcoin Core's aging Autotools-based build system with a modern alternative that is easier to maintain, extends better to new platforms, and unlocks improvements that were previously impossible.

In 2025, Hennadii's build system work focused on adaptation (adjusting to changes in the source tree), maintenance (fixing issues and working around upstream CMake bugs), restructuring (introducing new target output and install layouts), and modernization (removing outdated modules and supporting new features like the kernel API). He also contributed portability fixes, ensuring Bitcoin Core compiles correctly across a growing range of platforms and configurations.

Hennadii also maintains the bitcoin-core-nightly CI repository, which runs tests against a wide array of platforms that aren't covered by Bitcoin Core's main CI including FreeBSD, OpenBSD, NetBSD, illumos-based systems, Alpine Linux, and various Windows configurations including cross-compilation with different toolchains. This repository serves as an early warning system for portability regressions and has been the source of multiple bug reports that improved code quality in the main repository.

Press:

- Bitcoin Core switch to CMake build system - Bitcoin Optech Newsletter #316
- Discussion about the CMake build system - Bitcoin Optech Newsletter #316 Recap Podcast

Qt 6 Migration

One of Hennadii's concrete goals for 2025 was completing the migration from Qt 5 to Qt 6 which was necessary because Qt 5 reached end-of-life. He accomplished this in a substantial pull request (#30997) that generated 274 review comments, making it one of the most extensively reviewed changes of the year. Bitcoin Core v30.0 was the first release to ship with a Qt 6-based GUI.

As part of this migration, Hennadii achieved a long-awaited goal of splitting Qt into native and target-specific packages in the depends build subsystem. This separation enables future security and performance improvements that weren't possible with the previous monolithic

Qt integration. He also contributed several follow-up bug fixes and improvements to ensure the transition was smooth for users.

Windows Support and CI

Hennadii is one of the Bitcoin Core developers most focused on ensuring the software remains fully functional on Windows, a platform with a large number of less-technical users who want to use Bitcoin without trusting any third party. In 2025, he re-introduced native Windows CI testing for cross-compiled binaries (#31176, 121 review comments), an improvement over the previous approach because the binaries are now tested on actual Windows systems rather than emulation layers.

He also introduced new CI jobs for cross-compiling and testing using the Universal C Runtime (UCRT), which represents the future direction for Windows support, and began work on enabling native Windows builds using Clang, a project that could bring significant performance improvements to Windows users.

Libsecp256k1

Hennadii accomplished a goal he had set for 2025 by elevating libsecp256k1's CMake-based build system from "experimental" to official status (secp256k1#1680). This was completed in a pull request that prompted the libsecp256k1 author to begin using CMake for his own presentations, a gratifying validation of the work. He also attended the Crypto Camp alongside fellow Brink engineers Sebastian and Fabian, deepening his understanding of the library's internals and beginning to contribute more actively to its review process.

GUI and Translations

Hennadii continued maintaining both the Bitcoin Core GUI repository and the QML GUI repository throughout 2025, triaging issues, reviewing pull requests, and often fixing bugs himself. He managed the full translation process for both major releases, coordinated with language translators, and recruited new translation coordinators for Polish, Czech, and Greek.

> *“I am deeply and sincerely grateful to Brink for the opportunity to work on the project that lays the foundation for the future of humanity.” - Hennadii*

Other Contributions

Beyond the main Bitcoin Core repository, Hennadii contributed to libsecp256k1, libmultiprocess, the Bitcoin Core packaging repository, and the maintainer tools repository. He also stepped in to maintain the bitcoin-core and libsecp256k1 packages in the Guix package manager, and contributed upstream fixes and improvements to Boost, Cap'n Proto, Libevent, and Qt.

Plans for 2026

Hennadii plans to contribute more significantly to libsecp256k1 review, leveraging the cryptographic skills he developed at Crypto Camp. He'll continue maintaining the build system with a focus on improving the performance of Guix-based release builds (targeting up to a 20% speedup), transitioning Windows release binaries to the UCRT runtime, enabling

native Windows builds with Clang, updating Qt 6 to the latest version, and expanding IWYU (Include What You Use) checks across the codebase.

Stéphan Vuylsteke

@stickies-v

Stéphan joined Brink in 2022 and works full time on Bitcoin Core development. In 2025, he deepened his involvement with the Bitcoin Kernel project, the effort to extract Bitcoin Core's consensus logic into a standalone, reusable library, and it has become a major focus of his work. His contributions there earned praise from the project's lead developer, while he continued to provide reliable review across the broader Bitcoin Core codebase and to contribute to Bitcoin's developer community through events and educational work.

In 2025, Stéphan opened 14 pull requests, left over 700 review comments, and made significant contributions to py-bitcoinkernel.

On behalf of our sponsors, Brink is pleased to continue funding Stéphan's work on the kernel project and his contributions to the broader Bitcoin Core codebase.

Review

> *"Like previous years, I try to focus on going in-depth while providing quick re-review to prevent the PR from losing momentum".- Stéphan*

Stéphan continued to provide reliable, thorough review across the Bitcoin Core codebase. His most significant reviews outside of the kernel project included work on log rate limiting (84 review comments on the PR that mitigated disk-filling attacks), validation logic improvements, and codebase modernization efforts like improving compile-time type safety and converting `GenTxid` to use `std::variant`.

Colleagues continued to value his consistency. One peer noted in the 2023 report that "if he reviews something, he does re-reviews very quickly, and he sticks with it until it's merged" a quality that remained evident throughout 2025. He also made a point of providing conceptual feedback on pull requests he believed weren't worth the maintenance burden, a difficult but necessary contribution to the project's health.

Bitcoin Kernel

The Bitcoin Kernel project aims to separate Bitcoin Core's consensus validation logic from the rest of the codebase and expose it through a clean, stable API. This makes it easier to audit the code that actually enforces Bitcoin's rules. It allows other Bitcoin implementations to use the exact same consensus code as Bitcoin Core, reducing the risk of accidental chain splits. And it enables new use cases like alternative node implementations, block analysis tools, and indexers like silent payment scanners. Any application that needs to validate blocks and transactions can do so with Bitcoin Kernel without running a full Bitcoin Core node.

Stéphan became a central contributor to this effort in 2025. He spent extensive time reviewing and exploring alternatives for the C header API (#30595), including researching trade-offs around C versus C++ headers, ownership semantics, reference counting, logging, and thread safety. He participated actively in the weekly kernel working group meetings and contributed upstream improvements to Bitcoin Core to simplify the kernel's internal architecture.

His most visible contribution was building and maintaining py-bitcoinkernel, the Python bindings for the kernel library. Over the year, he significantly improved the build and CI system, added wheel releases for easy installation on most platforms, upgraded the interface to reflect improvements to the C API, added Python-native logging, improved documentation, and implemented the Bitcoin Kernel Binding Conformance Tests. The project is approaching the point where it can be taken out of alpha status, mirroring the progress of the underlying C API.

Sedited, the kernel project's lead developer, described Stéphan's contributions:

> *"Stephan has been instrumental over the past year to make significant progress on the kernel library. He authored Python bindings, re-implemented the API during review, and provided high-quality review comments that had an impact for realizing a coherent API. He also helped recruit other developers to the project."*

Notably, Stéphan also hosted a Bitcoin Kernel workshop at the BTC Prague dev/hack/day, bringing more awareness to the project and its potential. Fellow Brink engineer Sebastian Falbesoner used py-bitcoinkernel for his SwiftSync proof-of-concept, demonstrating the practical value of the Python bindings for prototyping.

Educational and Community Work

Stéphan hosted 8 London BitDevs socratic seminars in 2025, maintaining a regular cadence of focused technical discussion for the London Bitcoin developer community. He contributed to Bitcoin Optech, including authoring and reviewing content for the 2025 year-in-review special and joining as a guest on the Optech podcast. He also co-maintains the Bitcoin Core PR Review Club, an effort to cultivate the next generation of developers.

Plans for 2026

Stéphan's top priority is continuing to extend and improve the Bitcoin Kernel public interface and taking py-bitcoinkernel out of alpha status with robust documentation, tests, and a changelog. He plans to help build out the broader kernel ecosystem by making the library useful for developers beyond Bitcoin Core. He'll continue providing review across the codebase, contributing to Bitcoin Core's maintainability and robustness, and working on outreach through talks, workshops, and contributions to Optech.

Outlook

The security and reliability that Bitcoin users depend on every day is the product of careful, deliberate, daily work. Reading other people's code line by line, writing tests for edge cases, contributing to the underlying build system and cryptographic libraries, and quietly fixing vulnerabilities before anyone knows they exist. Brink engineers do this work because they believe Bitcoin matters, and because they know that open source software only stays strong when talented people are given the support and freedom to maintain it.

Brink exists to provide that support. We find exceptional engineers, fund them to work full time on Bitcoin's open source infrastructure, and give them the environment and resources they need to do their best work. We are proud of what our team accomplished in 2025, and we are excited about the work ahead in 2026.

None of it would be possible without our donors. Brink is 100% funded by donations from individuals and organizations who share our mission of strengthening Bitcoin. Every contribution, whether from a major sponsor or an individual supporter, goes toward ensuring that Bitcoin's codebase continues to be maintained, tested, reviewed, and improved by people with the skills and dedication to do it in a “Move slow, fix things” fashion. To everyone who has supported Brink, thank you. Your generosity funds the work that keeps Bitcoin secure for everyone.

If you or your organization is interested in supporting open source Bitcoin development, please visit brink.dev or email us at donate@brink.dev.