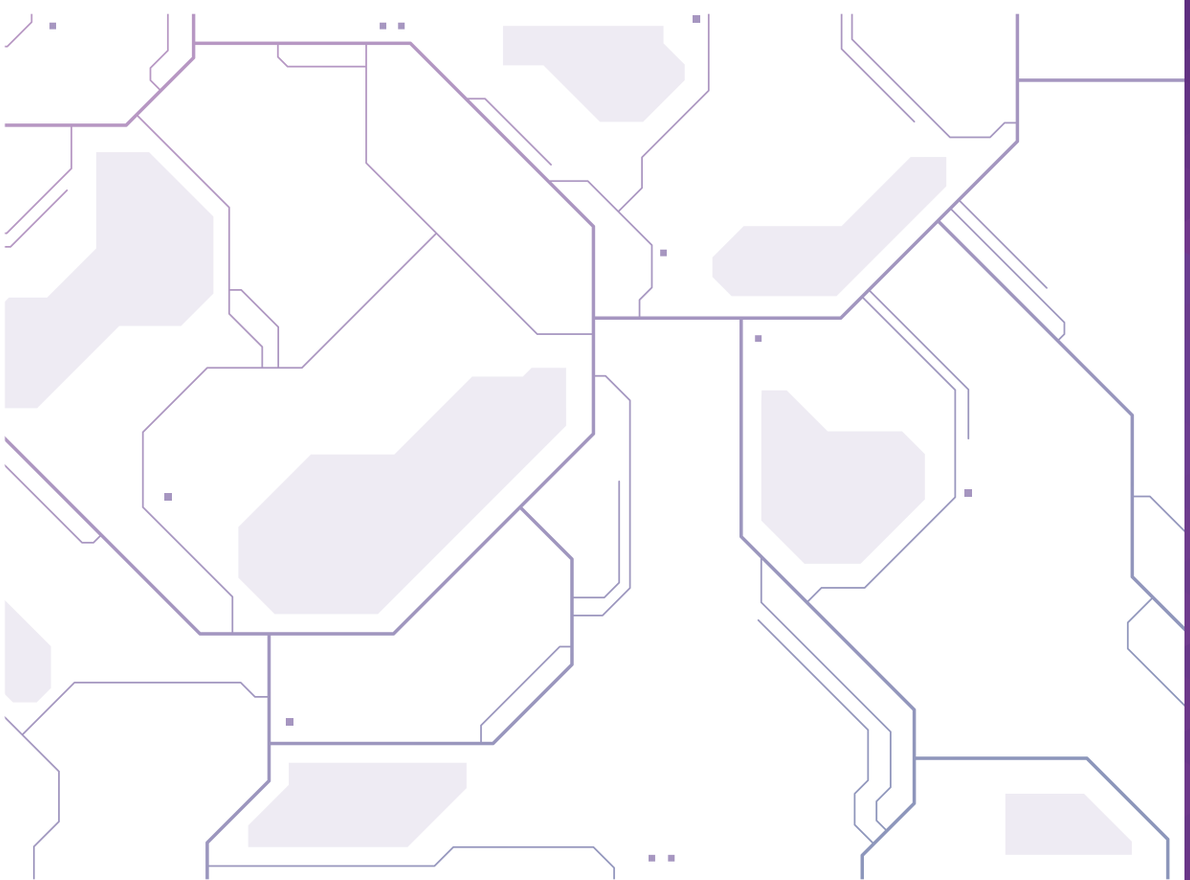
Building a Future-Proof API Inventory

How to Create a Living API Inventory and avoid vendor lock-in



Executive Summary

Modern organizations increasingly rely on APIs to deliver digital services, integrate systems, and expose resources to partners and customers. An API is not a single endpoint, but a collection of related endpoints built around a business need or application. As the number of APIs grows, managing them becomes increasingly complex. Without clear visibility and governance, APIs can proliferate uncontrollably, creating operational, security, and compliance risks. A living, governed API inventory ensures every API is documented, versioned, and maintained alongside development, providing a single source of truth that is always accurate, up to date, and fully aligned with the APIs that are exposed.

42Crunch helps organizations build this inventory using open, standardized formats such as OpenAPI, stored directly in source control alongside the API code. This approach avoids vendor lock-in, enables cross-team collaboration, and allows the inventory to be leveraged across development, testing, security, and operations. By embedding API governance into the development workflow, teams can enforce security and compliance continuously, maintain runtime protections, and adopt a true DevSecOps approach for scalable, secure API operations.

Table of Contents

Guidelines for Building an API Inventory

The primary purpose of an API inventory is to bring APIs under a managed, repeatable process for security, compliance, and API governance. Statically listing APIs is not enough, as APIs are constantly added, updated, and retired. To be reliable and actionable, an inventory must accurately track API changes at the pace of modern API development and release cycles.

A reliable API inventory achieves this by following three core principles:

1. **Alignment with the API Development**
   The inventory must be integrated with the API development process, ensuring it always reflects the APIs that are exposed and tracks development changes as they occur. This keeps the inventory accurate, and up to date for security, compliance, and operational purposes.

2. **Use of open, universal formats**
   API records should be documented using open formats, such as OpenAPI. This allows the inventory to act as a source of truth that all teams can understand and rely on, while enabling integration with a wide range of tools and platforms for API development, testing, security, and runtime monitoring.

3. **Storage alongside the API source code**
   API records should be maintained in the source code repository alongside the API code. This keeps the inventory aligned with the same versioning, workflows, and change-management processes as the code. By anchoring the inventory in the repository rather than a proprietary platform, teams avoid vendor lock-in and retain full control of their API assets.

By adhering to these principles, organizations create a living API inventory that evolves in lockstep with the APIs themselves, providing a foundation for secure, well-governed, and fully manageable APIs.

Inventory Aligned with API Development

Modern APIs are continuously added, updated, and retired. An inventory that does not track these changes will quickly become incomplete or inaccurate, creating API drift and security gaps.

42Crunch makes the API inventory a natural product of the development process. Developers work in their preferred IDEs (VSCode, IntelliJ, Eclipse), documenting APIs alongside their code so that the inventory reflects accurate API records from the start. Each of these records serves as an API contract, defining precisely how the API is expected to behave. 42Crunch API Audit validates these contracts against organizational standards, ensuring every addition to the inventory meets requirements for security, quality, and completeness. Meanwhile, 42Crunch API Scan dynamically tests running APIs against their contracts to detect and correct drift, keeping the inventory continuously aligned with actual API behavior.

To enforce consistency across teams at scale, API Audit and API Scan are also fully integrated into CI/CD pipelines such as Azure Pipelines, Jenkins, and GitHub Actions. Automated checks maintain accuracy and consistency as APIs are added or updated, creating a living API inventory across the organization that evolves with development, and supports secure and well-governed API lifecycles.

Open Standards for a Shareable API Inventory

Documenting APIs in open, standardized formats such as OpenAPI ensures the inventory remains interoperable, flexible, and accessible to every team and tool across the lifecycle.

API contracts act as a common language that different stakeholders and platforms can rely on. Some tools use them to generate documentation, while others create mocks or test collections for functional and security testing. Comprehensive platforms like 42Crunch build on the same contracts to enforce governance policies and apply runtime protections that expose only documented endpoints.

By adopting open standards, organizations create a single source of truth that supports cross-team collaboration, ensures consistency across tools, and strengthens the foundation for secure, well-governed APIs.

Source Code Repositories as the Single Source of Truth

A resilient API inventory is best managed alongside the API code in the source code repository. Keeping API contracts in the repository ensures the inventory evolves with development, providing a living, accurate record that other teams and processes can rely on without additional overhead.

By contrast, some API inventory solutions force organizations to maintain records on proprietary platforms, effectively siloed from the API code, and limiting integration with existing toolchains.

Anchoring the inventory in the repository avoids this lock-in, keeps it tightly coupled to the API lifecycle, and ensures it remains the definitive and reliable source of truth.

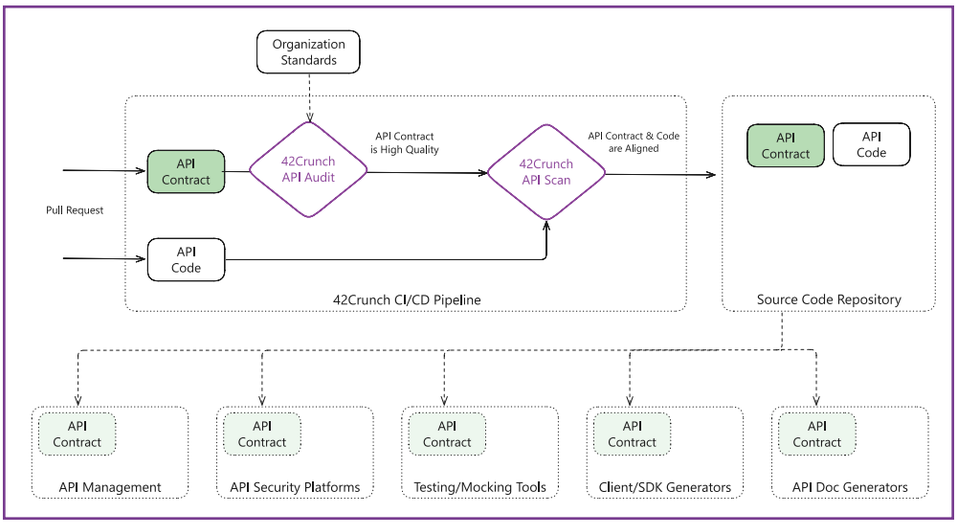


Figure 1 - Governed API Inventory

Inventory-Driven API Protection

Used intelligently, an API inventory becomes far more than a record of API endpoints. With 42Crunch it can also serve as a precise whitelist for runtime protection. Built from API contracts, the inventory provides an authoritative map of the endpoints and operations each API supports. This precision matters, as attackers frequently probe or fuzz APIs in search of undocumented or hidden entry points.

42Crunch API Firewall enforces this living inventory at runtime, exposing only what is explicitly defined in the contracts and blocking all other requests by default. Protections update as contracts evolve with API development, ensuring runtime protection is always accurate, and shields backend API servers from the risk and performance impact of inspecting and blocking invalid requests in real-time.

This is a true DevSecOps model for API security. Development teams maintain accurate API contracts as part of their normal workflow, operations teams enforce those contracts in production, and security is engineered into the process from the start. The result is automated collaboration across functions, with security at the core.

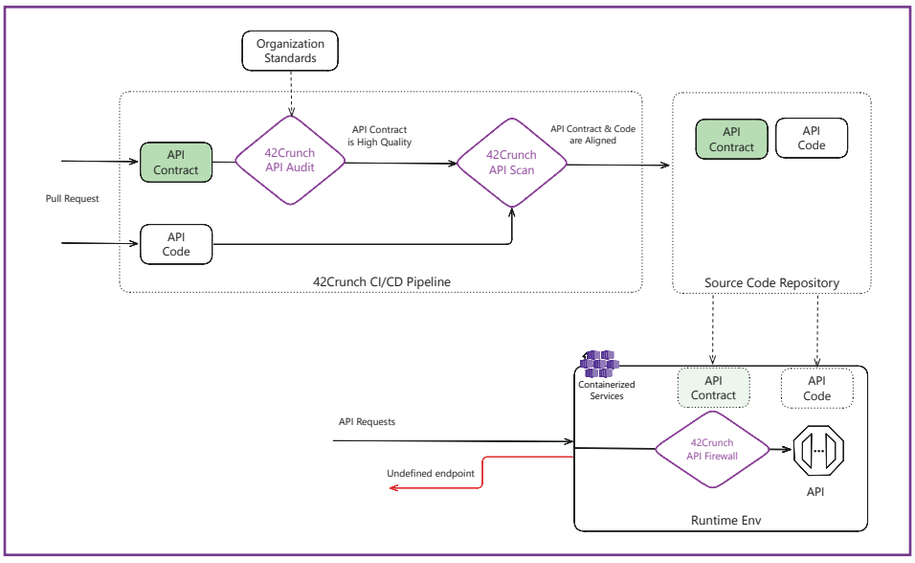


Figure 2. Only endpoints from the API inventory are accessible.

Teams can adopt this approach incrementally, starting with their most sensitive APIs to immediately block fuzzing attempts and anomalous behavior, then extending coverage across their portfolios. The outcome is runtime security that is deterministic, development-driven, automated, and scalable.

Getting Started

Getting started with 42Crunch is straightforward and designed to make doing the right thing the easiest thing to do. The popular IDE extension, 42Crunch OpenAPI Editor, downloaded over 2 million times by VSCode, IntelliJ, and Eclipse users, lets developers create precise API contracts

Figure 3. Defining allowed API endpoints and operations.

without leaving their IDE. Developers can work on API code and the contract side by side, while automation features help design-first teams quickly produce high-quality contracts that guide implementation.

Teams can begin with something as simple as defining the allowed API endpoints and operations.

As contracts are created, they are submitted to source control along with the API code, immediately becoming part of the organization's living API inventory. From there, the inventory grows organically as developers maintain contracts alongside code, ensuring it remains accurate, up to date, and fully integrated into the development workflow from day one.

For code-first developers, 42Crunch API Contract Generator accelerates the creation of API contracts from Postman Collections or web traffic logs. As teams explore or test their applications, API Contract Generator automatically generates contracts that can be committed to source control and integrated into automated pipelines for continuous development and improvement.

Whether contracts are crafted by design-first teams in the IDE or generated by code-first teams from collections and logs, once stored in source control they are automatically synchronized with 42Crunch API Security Platform. This creates a complete, continuously updated API inventory. From there, teams can apply consistent security policies and automated tests across their entire API portfolio, while the repository remains the single authoritative source of truth.

Clean-up Operations with API Discovery

A sustainable API security strategy begins with a complete and accurate API inventory. The most effective way to achieve this is by aligning the inventory with the API development process. When every API has a clear, up-to-date definition, the inventory accurately reflects the current state of published APIs. This governance process ensures that no API enters production without a corresponding definition, reducing the risk of unmanaged endpoints and unintended security gaps.

By contrast, API Discovery should be viewed primarily as a clean-up activity. Organizations that have not enforced API governance often lose visibility into their portfolios, leaving unknown APIs to operate outside of security and compliance controls. In these cases, discovery can help identify existing APIs in the environment so they can be documented, aligned with organizational standards, and brought under the same governance and testing regime as actively managed APIs.

API Discovery therefore serves as a short-term solution, but it is not a substitute for proper governance. The long-term solution is to embed API definition and inventory management into the development lifecycle, ensuring that visibility, security, and compliance are engineered in by design.

The Risks of Relying on Traffic-Based API Discovery

Traffic-based API discovery, using network monitoring and machine learning, creates major challenges and risks. Each new device, node, or service must be instrumented, and infrastructure changes require constant updates, generating ongoing operational overhead. Even with exhaustive effort, blind spots remain where traffic cannot be captured.

Machine learning results are non-deterministic, producing incomplete or incorrect findings that require manual review by security teams. This shifts responsibility from developers, who are best positioned to document APIs accurately, to security and operations teams.

Visibility is delayed: APIs are detected only after they are in production, leaving a window of exposure during which new or low-traffic endpoints remain unprotected, yet still exploitable. What should be a one-time cleanup can become a perpetual, costly cycle, including recurring license and maintenance expenses.

By contrast, maintaining API documentation as part of the development workflow ensures inventories are accurate, deterministic, and available before production. This proactive, standards-based approach eliminates gaps, blind spots, and operational overhead, providing a secure, reliable, and scalable foundation for API security.

API Discovery with 42Crunch

Building a complete, accurate API inventory often requires capturing evidence of API activity from multiple sources across the organization. But discovery on its own is not enough. As outlined earlier, the primary purpose of an API inventory is to bring APIs under a managed, repeatable process for security, compliance, and governance. Once APIs are discovered, teams need to address the root causes of why those APIs were unmanaged in the first place. That means documenting them in open formats like OpenAPI, storing them in enduring, non-proprietary platforms, and avoiding vendor lock-in that perpetuates a cycle of repeatedly rediscovering shadow APIs instead of governing them from the start.

42Crunch supports this by locating and converting evidence of API activity into open, standardized API contracts that can be fully integrated into a unified system for governance.

Connect to Source Code Repositories

Most API teams are already in the practice of creating OpenAPI files, since they are the easiest, and in some cases required, way to publish APIs on platforms like AWS API Gateway, Azure API Management, or Google Cloud API Gateway. Even when developers don't share details about published APIs with security or operations, the OpenAPI artifacts usually exist because they make self-publishing and external exposure much simpler. This long-standing practice means that API contracts are often present, even for unmanaged APIs.

Building on this, 42Crunch streamlines API discovery by automatically scanning source code repositories for API contracts (such as Swagger or OpenAPI files), importing them into the 42Crunch API Security Platform for audit and security analysis, and adding them to the managed inventory. Discovery can be performed on a scheduled basis or triggered manually at any time. 42Crunch also integrates discovery with the normal development cycle, where on each pull request by developers, the current repository is searched for evidence of new API contracts

Integrate with Native API Platforms

To extend discovery of API contracts beyond repositories, 42Crunch also integrates with popular platforms like Microsoft Azure to locate and collect API contracts that developers have already imported there. Once identified, these APIs and their contracts are automatically audited and scanned for security risks, ensuring they are brought under the same governance, security, and quality processes as managed APIs.

API Discovery from Postman and QA Collections

All APIs, whether managed or unmanaged, are typically exercised by development and QA teams during testing. Test artifacts such as Postman collections or website log files can therefore serve as a valuable source of insight into APIs that may never have been published through formal channels. Involving QA teams in the API discovery process helps enrich visibility and ensures that hidden or shadow APIs are not overlooked.

With 42Crunch, Postman collections and HAR (HTTP Archive) files can be directly ingested into API Contract Generator, which automatically generates OpenAPI contracts from the test assets. These contracts are then processed through API Audit, where organizational security and quality policies are applied, bringing every API into a consistent, governed inventory.

Uncovering Hidden APIs via Network and Operations Logs

Unmanaged APIs often leave traces in network and platform logs from web servers, load balancers, gateways, or firewalls. By working with networking and operations teams, organizations can uncover calls to undocumented endpoints that have not yet been governed.

Attempting to instrument every network device with agents or applying machine learning to monitor all traffic creates a costly, ongoing burden for operations teams. A simpler and safer approach is to collaborate with teams to retrieve relevant logs. These logs can be exported and shared locally, keeping sensitive traffic data internal. Tools like 42Crunch API Contract Generator can then parse the logs, filter for API traffic, and automatically generate API contracts for import into a governed API inventory, efficiently bringing unmanaged APIs under control.

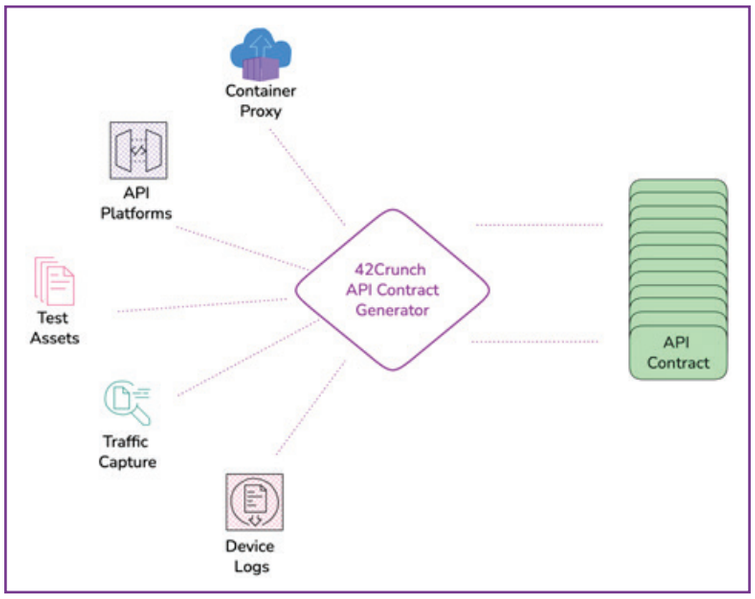


Figure 4. Standardize API Inventory using open formats

Discovering APIs by Proxy and Monitoring for Inventory Drift

Another effective approach to API discovery is by selectively proxying and monitoring live traffic. Done intelligently, this method relies on collaboration with networking, security, or operations teams to ensure coverage without creating unnecessary operational overhead. API discovery should be treated as a once-off cleanup operation, with the primary focus on establishing a system of good governance for API documentation and inventory-based runtime security.

42Crunch API Firewall can be deployed in monitoring mode to proxy container-based API traffic. It inspects responses from the server to detect and report unmanaged or undocumented routes. For environments beyond containers, 42Crunch also provides a lightweight proxy component that can be deployed on web servers or other nodes, capturing API routes that are not yet defined in the governed inventory.

This approach allows organizations to detect and report hidden APIs while maintaining control and minimizing risk. By combining selective live traffic monitoring with log-based discovery, teams can efficiently bring all APIs under governance without overburdening infrastructure or personnel.

By consolidating disparate API evidence into a single, standardized inventory, 42Crunch enables teams to enforce governance, and apply security policies consistently for all APIs.

Conclusion

A living, governed API inventory is the foundation of secure, well-managed, and scalable API operations. With 42Crunch, API contracts become a natural part of development, created, updated, and maintained alongside API code without disrupting workflows. Stored in source control, these contracts form a single source of truth that is accurate, shareable, and under the team's control, avoiding vendor lock-in while enabling collaboration across development, security, and operations.

API discovery should be treated as a short-term clean-up operation to bring existing unmanaged or undocumented APIs under governance. By keeping the process simple, talking to teams, retrieving evidence from sources such as API gateways, Postman collections, traffic logs, or device logs, and converting it into standardized API records, organizations can quickly capture and standardize legacy APIs without ongoing operational overhead.

Building the inventory on open, standardized formats ensures it can be leveraged by multiple tools and processes, including automated documentation, mock generation, functional and security testing, governance enforcement, and runtime protection. Storing contracts in version control alongside the API code maintains alignment with development, supports cross-team collaboration, and avoids the drawbacks of proprietary inventory products, such as lock-in, siloing, and limited integration.

By making the right practices the easiest practices, 42Crunch enables a true DevSecOps model for API security, where inventory, governance, and protection evolve seamlessly with development and continuously protect the organization's API ecosystem.

About 42Crunch

42Crunch enables a standardized approach to API security that automates the enforcement of security compliance across distributed development and security ecosystems. Our API security testing and runtime threat protection services are used by Fortune 500 enterprises, mid-sized firms and more than 1.8 million developers worldwide.

The 42Crunch API security platform empowers developers to build security from the IDE into the API pipeline and gives application security teams full governance from the CI/CD across the entire API lifecycle. This seamless DevSecOps approach to API security reduces governance costs and accelerates the delivery of secure APIs.

Contact Details

Email: **sales@42crunch.com**
Website: **42crunch.com**