

A red circle containing a white warning triangle with an exclamation mark inside.A 3D cube with a green top face and a blue center on the top face.A 3D cube with a blue top face and a green center on the top face.

**THE
EVOLUTION OF
APPLICATION
SECURITY (AND
WHERE WE GO
NEXT)**

APPLICATION DEVELOPMENT IS GOING THROUGH A RENAISSANCE. ARE YOUR SECURITY TOOLS PREPARED FOR THE FUTURE?

THE APPLICATION RENAISSANCE HAS BEGUN

It's been growing over the past several years and now has grown large enough that it can't be ignored. The world of APIs is upon us.

An Application Programming Interface, or API, is a set of functions allowing software programs to access and interact with the data and features of an application.

APIs break application functionality apart into discrete units that can be called independently of each other.

For example, a shopping cart can call an API to gather tax information for a specific state. Another API call is used to calculate postage or apply a discount. The calling application can choose which features to use and in what order to use them.

But along with changing application development standards come changes in how to secure them. In this paper, we'll delve into how the changing face of software development leads to sweeping changes in the application security landscape. We'll also discuss what application security looks like in the new world of APIs.



THE CURRENT STATE OF APPLICATION DEVELOPMENT

Software development standards have changed dramatically over the last decade. Long delivery cycles and large applications are no longer the norm.

Traditional web applications once consisted of large, monolithic chunks of code. They were deployed all at once after a long development cycle. Project management at the time started with all of the requirements upfront, spent several months (or even years) building the application to match said requirements, and then deploying everything at once.

However, those in the industry began to notice that software always seemed behind. What was delivered was likely going to be wrong in some major way. Sometimes entire applications or large pieces of them turned out to be expensive mistakes.

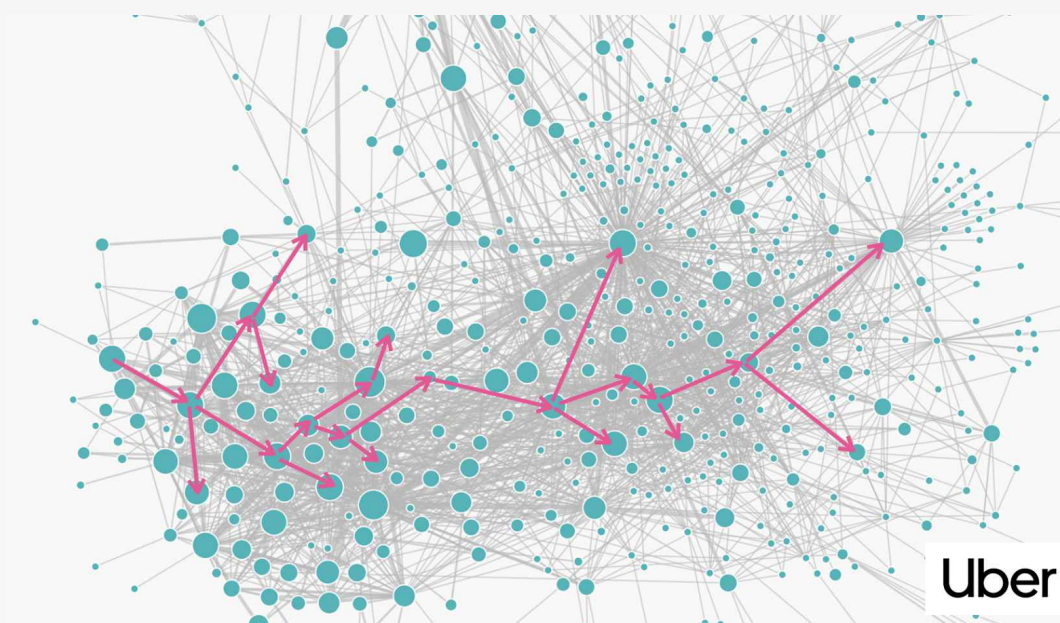
This realization gave way to more agile methods of software development, such as Kanban and Scrum. These methods focused on completing and delivering small pieces of software more frequently. Then users can give feedback and the development team can respond to the feedback and update the code. Instead of assuming you know all requirements upfront, you discover the requirements through use.

Lean methodologies take this philosophy a step further. When building new applications, you develop the minimum viable product, or MVP, and deliver it. It's enough to be valuable but doesn't have everything a user may want on day one. Over time, the application is expanded and changed based on feedback from users. This approach validates the software while it is built.

Changes in thinking about the right way to deliver software ran into an obstacle with existing development practices. Most application frameworks were built with the monolithic architecture in mind. Making a change to one piece of the code by itself was much too expensive to do frequently. This challenge led to companies trying to batch as many changes as possible into each release because each release was so difficult and expensive.

New technologies and architectures were needed to match the fast pace of business change and new agile methodologies. The introduction of API-first, cloud-native applications changed how application developers created code forever. Small microservices loosely coupled together gives developers ultimate flexibility.

For example, the below diagram shows a map of microservices at Uber. The pink arrows show a possible path a request to the application may take to gather the necessary information for the user.



This style of application architecture is not only used by trendy Silicon Valley “unicorns.” Government institutions, financial services companies, legacy retailers and non-profit organizations are beginning to develop using microservices. Many are migrating existing applications into microservices over time to take advantage of the many benefits.

Benefits of microservices include:

Scalability. New containers and service instances can be created dynamically during peak times and then taken down when the extra muscle is no longer necessary.

Technology agnostic. Loose coupling and standard interfaces between microservices mean each team builds their service using the best language for the use case.

Speed of development. Small code bases and a focused “do one thing well” philosophy means changes are fast and efficient. Many companies deploy to production several times a day.

Agile and DevOps movements have driven applications to focus on small pieces and APIs. But along with new ways of developing applications have come new security challenges.

What has security done to keep up with developers?



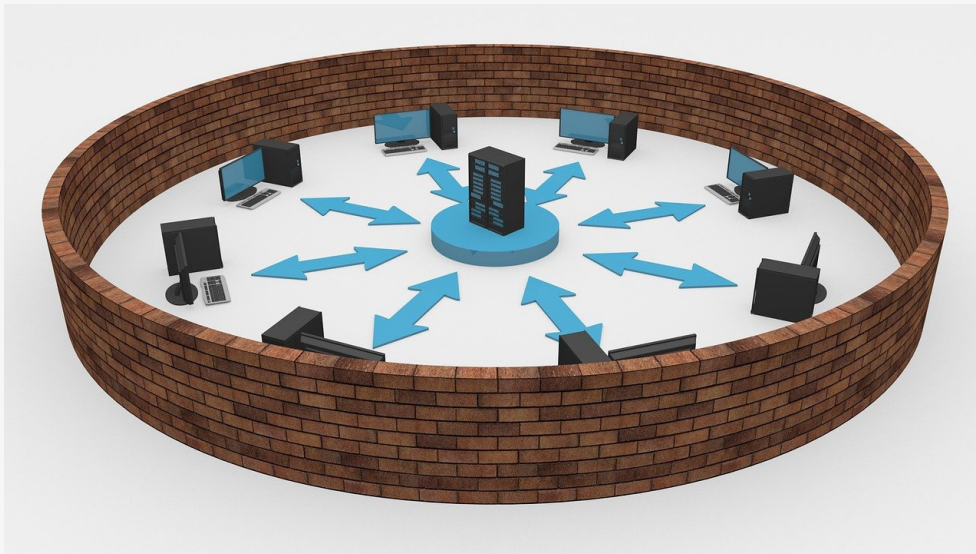
THE EVOLUTION OF APPLICATION SECURITY

The traditional security model is a “moat around the castle” approach. Build barriers around your network and keep the bad guys out.

Hardware firewalls and physical security measures are put in place to protect servers. Physical access to servers is closely monitored. Traffic coming into the network is vetted before being allowed to connect.

But physical servers aren't the only infrastructure under attack. Your software is under attack, too. Firewalls stop traffic based on IP addresses, so an attack that is sent through HTTP port 80 (or HTTPS port 443) will be allowed through. Without these ports open, you might as well close up shop, because legitimate traffic won't be allowed through either.

Something else was needed to protect web applications when attacks began to be launched through legitimate channels.



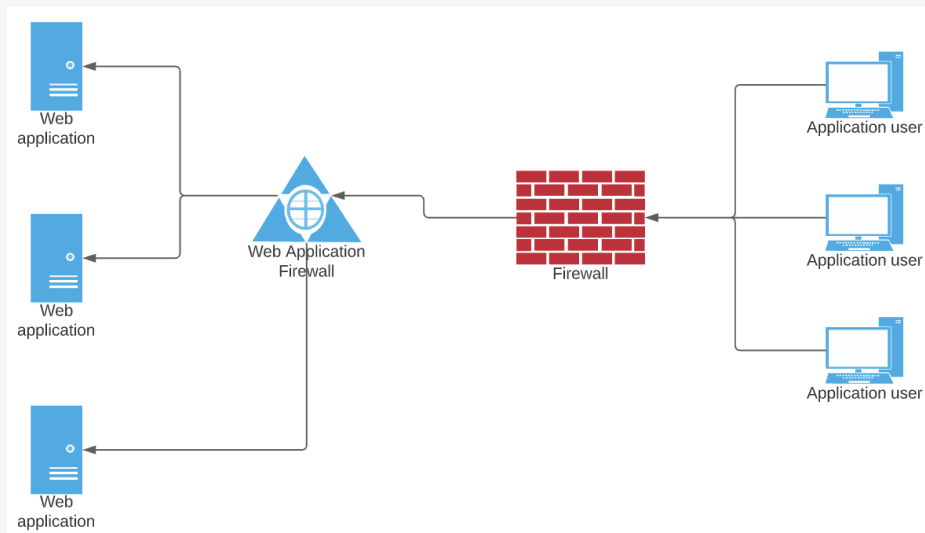
WAFS - BUT STATIC RULES FALL SHORT

As the world became increasingly dependent on web applications for doing business, attacks against web applications became the focus of application security.

Web Application Firewalls (WAFs) take the idea of network firewalls and wrap it around a web application. Network firewalls keep out known bad traffic and only let in safe traffic. WAFs prevent attacks by looking for signatures of known attacks and only allowing safe traffic through to the application.

WAFs changed the game by focusing on application vulnerabilities. A legitimate call to port 443 of a web application could contain an SQL injection attack. A hardware firewall would let it pass through. The WAF will stop it before it reaches the application and causes a data breach.

WAFs are an important part of application security. But there are limitations to static rule-based protection. Rule-based security is inherently behind current state attacks. An attack has to happen for a rule or signature to be created. If a novel attack occurs, someone has to pay the price and get hit before that attack can be stopped the next time. WAFs stop well-known attacks but keep you playing catch up. The next step in application security technology tried to fix the static rule problem.



A typical architecture with a firewall for network security and a WAF for application security

THE PROBLEMS THAT NG-WAF WON'T SOLVE

Next Generation WAFs (NG-WAFs) do the same work as traditional WAFs but with extra features added to help secure modern applications. Machine learning and behavior analysis are added to the signatures to provide a more complete defense. NG-WAFs promise to detect anomalous behavior and stop it before the attack succeeds.

An increased cost to attackers.

Attackers try to take over anything with an internet connection (DVRs, camera, smart light bulbs, etc) and use them as an army of “bots” to attack systems and cover their tracks. NG-WAFs increase the cost of these attacks and deter script kiddies by correlating these attacks with known patterns and stopping them.

Layered protection. NG-WAFs embrace the “Defense In-depth” philosophy and protect at multiple levels of your infrastructure. They protect the perimeter and embed themselves alongside your applications.

Automated Policy Learning. NG-WAFs use machine learning and behavioral analytics to understand attackers from a broader perspective. They can automatically disable signatures that would otherwise trigger false positives and update application policies automatically.

Virtual Patching. Not all vulnerabilities will be fixed quickly. Some may not be fixed at all. Some NG-WAFs support virtual patching to prevent vulnerability exploitation until a code fix is made available.

Cloud-Native Support. NG-WAFs support deployment in a public, private, or hybrid cloud. They can use containers to scale and distribute nodes around the world to reduce latency.

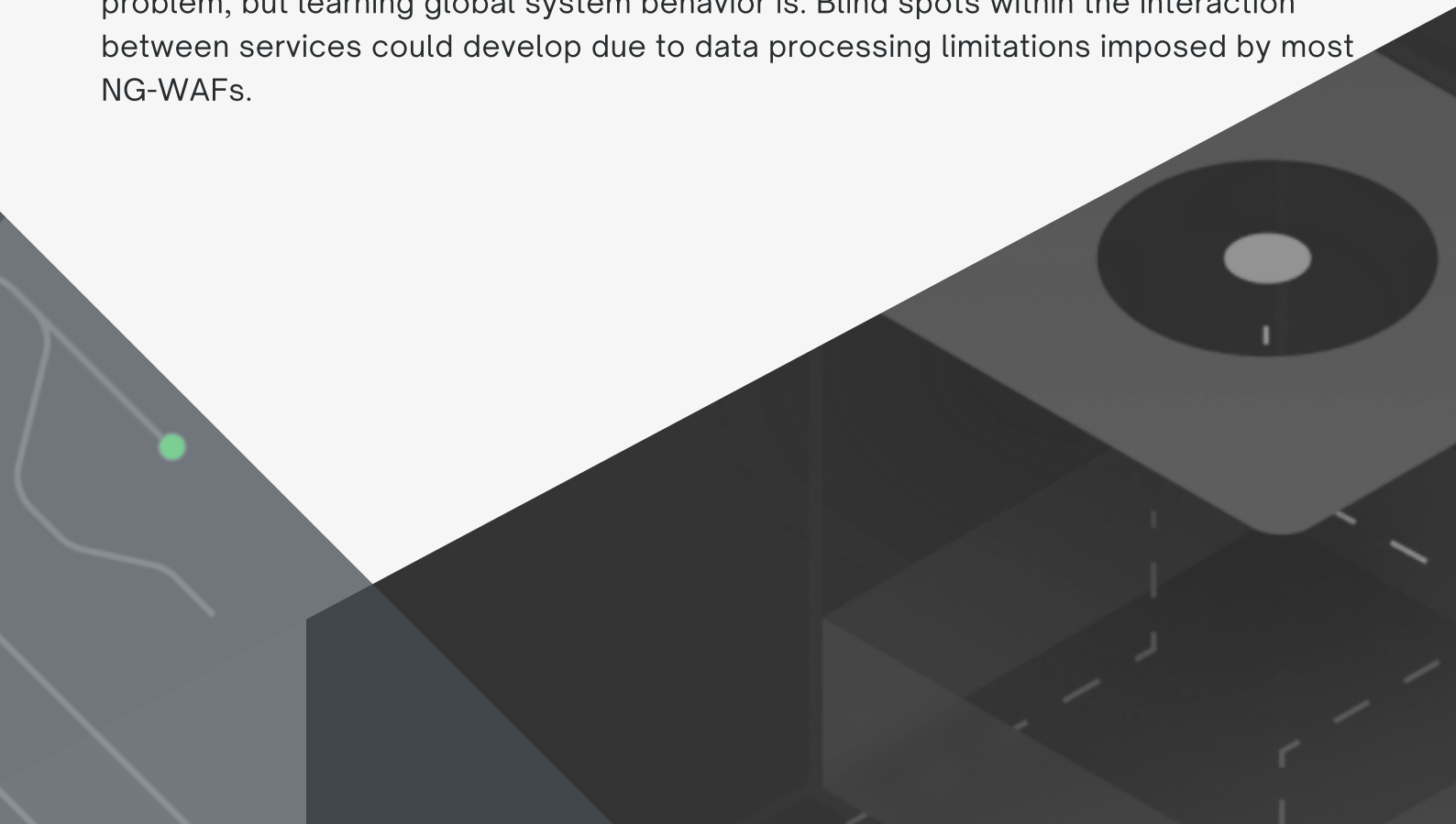
These are all valuable advances in application security. But NG-WAFs have failed to solve the following problems.

API sprawl can lead to holes. As applications grow in complexity and the number of APIs exposed, the attack surface grows, too. When the application becomes complex, it may require you to disable defenses that produce a higher rate of false positives. Determined attackers can find these holes and exploit them.

Use case exploitation. Application context varies greatly. Harmful requests for one application could be normal traffic for another. NG-WAFs can't detect the business use case. The attacks remain hidden within normal traffic patterns. Attackers may be able to tune themselves to the business traffic to bypass the NG-WAFs and remain undetected for long periods of time. Unfortunately, turning up detection then increases false positives and false negatives.

Focus on external traffic leaves internal actors undetected. NG-WAFs started as traditional WAFs and have inherited the “perimeter defense” model. They focus on attacks originating from outside the network. Insider attacks may go undetected.

Monitoring blind spots. As applications become more distributed, you no longer have visibility into what is happening across the system. Monolithic applications stored their logs in one place. On the other hand, each microservice has its own monitoring and logging capabilities. Monitoring each service individually isn't a problem, but learning global system behavior is. Blind spots within the interaction between services could develop due to data processing limitations imposed by most NG-WAFs.



RASP - BUT PATTERNS ARE MISLEADING

The next big step in application security is Real-Time Application Self-Protection, or RASP. With RASPs, security moves into the code by linking into the application and/or the runtime environment. They can control execution and detect and prevent real-time attacks.

RASPs attempt to fill the gaps left by perimeter-focused solutions like WAFs. They watch the application execute and detect attacks as they happen within the executing code. RASPs have several advantages over other security solutions:

Watches rather than predicts. RASP watches the application execute and alerts when it sees actions performed. If a RASP alerts you that an unexpected shell command was executed, it means a real attack occurred.

Better SDLC integration. RASP tools integrate well with the software development lifecycle. Feedback from attacks observed in production can be used by the development team to fix the vulnerabilities attackers are trying to exploit. This model fits well with the “rapid feedback” philosophy of agile development and DevOps.

Deployment agnostic. RASP solutions implant sensors in the existing application code, essentially becoming part of the application. They may not even require network calls to work. This design makes them agnostic to deployment architecture, so they’ll work in any combination of cloud, on-prem, and container-based architectures.

Virtual patching buys you time. RASP protects your application from attacks in real-time. It features the same virtual patching capabilities as NG-WAFs. Due to this protection, development teams have time to develop sound fixes to vulnerabilities without going into crisis mode.

RASP has many benefits. However, there are areas where RASP falls short.

Confined context. No application is self-contained. RASP is tied directly to the code it's protecting. RASP's visibility is limited when it has to be deployed with small microservices at different endpoints. Attacks against that endpoint will be stopped, but can it detect actions across hundreds of services at once?

Use case abuse. Business logic attacks remain some of the most insidious attacks against modern applications. Broken Object Level Authentication (BOLA) is an example of a business logic attack RASP is likely to miss. BOLA occurs when an attacker substitutes the resource ID in an API call with the resource ID of another user.

For example, an API that returns medical records uses a number sequence for IDs. An attacker looks at the ID in the URI and increments it by one (i.e. 123435 becomes 123436).

The lack of proper authorization checks allows the attacker to see someone else's medical record. Since the input is consistent with what the application expects, RASP will let it pass through. An understanding of the larger business context is required to detect this type of vulnerability or attack.

Perimeter and global attacks missed. RASP only sees the current application context. Large-scale assaults such as Distributed Denial of Service attacks by armies of bots cannot be effectively solved by RASP. You'll still require a perimeter defense solution.

Deployment challenges. RASP solutions have to be deployed along with the application. When dealing with microservices, you'll have to deploy the RASP along with each instance of the service. Some organizations may view this deployment model as intrusive and may not feel comfortable deploying it for critical applications. Add to that the dedicated effort required to deploy, manage, and update RASP components at every endpoint, and deployment becomes expensive.

Unintended consequences. The way RASP works could lead to unintended consequences. For example, defeating an attack could require the rewriting of your software's execution at the last minute. Incident response and support could be affected as the application might change its behavior on the fly.

A second unintended consequence is unexpected downtime due to interference with the running code and false positives. RASP doesn't learn the business context, so it may predict that a valid use case is an attack and stop it. Guess what? Your security solution just performed a Denial of Service attack against the application it's trying to protect.

Application security has seen several new approaches appear on the scene. Traceable believes it has discovered the way to take the best features of these solutions and upgrade them for use on modern cloud-native applications.

HOW TO PROTECT MODERN APPLICATION ARCHITECTURES

A holistic approach to protecting microservices, APIs, and cloud-native architectures requires you to address three essential areas: API discovery and risk management, application protection, and data privacy and compliance. Let's break these down one at a time and discuss how Traceable solves each concern.

API Discovery and Risk Management

The Problem:

API proliferation is a common occurrence in microservice architecture. Changes to the APIs, as well as the connections between them, are uncontrollable, unpredictable, and unplanned leading to unexpected risks and vulnerabilities. Many security departments struggle to proactively identify and address risks within their microservice applications.

Constant changes to APIs create a risk of falling out of compliance and becoming subject to fines or other adverse consequences. It's also difficult to keep tabs on sensitive data flowing between microservices in API calls, exposing security leaders to further regulatory risk.

The Solution:

Security teams need to understand all of their API risks, where vulnerabilities are hiding, and where they might be out of compliance. They also need to learn about all changes to APIs and the potentially sensitive data that flows between them. Tracking data flow shows where sensitive data is at risk.



As part of a solution, the requirements are as follows:

- Continuous API Inventory and Change Detection
 - Dynamically discover all external and internal APIs. No more shadow APIs.
 - Detect and get alerted on new and changed APIs.
- API Risk Scoring in production & pre-production
 - Score each API by risk factors – authentication, authorization, sensitive data etc.
 - Integrate with your CI/CD pipelines to understand risk of API changes before they go into production.
- Spec Conformance Analysis
 - Flag APIs that don't match developer specifications (OpenAPI/Swagger)
- API User Analytics
 - Monitor API users – call volume, error rates, geographies, user profiles, etc.

Organizations spend lots of time and manual effort to keep up with the changes in their applications. Microservices pop up or are changed daily with little visibility into changes as a whole. Companies feel like they're flying blind.

Organizations need to efficiently and effectively assess and manage the security posture of their applications and APIs. They need visibility into their APIs so that they can proactively address security concerns, quickly respond to compliance requests, have complete information about the location of key sensitive information in their applications, and automatically produce API documentation that developers and security engineers can rely on.

How Traceable AI Helps:

Traceable AI gives your teams and you insight into your current Application and API Security Posture discovering and then continuously observing the "DNA" of your application and associated APIs.

Every application has a unique Application DNA, which describes the unique characteristics and markers of the application and it's behaviors. At the application level the DNA is the unique combination of services and associated data that the application manages. At the microservice level, the microservice DNA defines how the microservice interacts with other services. At the communications level the API DNA describes the behavior of the communications between the various components of the application. Through understanding the different DNAs of the application, traceable is able to give teams real-time visibility and insight into the security posture and health of their application.

Traceable Provides

- Complete visibility and risk management of all APIs
 - Automated discovery, inventory and classification of all APIs used by your applications including shadow APIs
 - How many APIs do you have?
 - Which are Internal vs. External
 - What applications do you have and what services and APIs implement those applications?
 - Which of those applications & APIs are critical to your internal and customer-facing business functions or can significantly disrupt the business if compromised?
 - Up to date state of your API usage, and changes across all your apps
 - Where are these APIs called from?
 - Who invokes these APIs, locations, protocols?
 - What is the frequency of these calls?
 - Do the call patterns align with the developer's expectations?
 - Do the actual APIs and functions match in specification with what has been documented, reviewed and approved?
 - When an API changes, are the changes appropriately noted, reviewed, documented and risks mitigated?
 - Identify your API vulnerabilities
 - Which ones have potential security issues?
 - Which ones are more likely to get attacked?
 - Which ones have sensitive data flowing through them?
 - Abuse of APIs identified and blocked before it can cause damage to your business.

Application Protection

The Problem:

Protecting modern applications is challenging. These new applications are increasingly complex, built in the cloud from dozens of microservices, and connecting to users on the web and mobile devices. Each level of complexity is a new threat vector.

Additionally, attacks are becoming more sophisticated and even harder to mitigate. Traditional web/application firewalls are no longer able to detect and prevent new threats. Companies need to be able to detect and prevent threats in modern applications, but the current tools and processes simply can't keep up with the challenge.

The Solution:

- Protection at the edge similar to an NG-WAF
- An intelligent and adaptable system where threats are detected and blocked, similar to a RASP
- Protection between microservices where threats hide between your microservices.
- The ability to understand and detect business logic attacks
 - Fraud Prevention
 - BOLA prevention and other vulnerabilities from the OWASP API Top 10
 - Authentication and authorization vulnerabilities
- Protection from future attacks through insight and research (user behavior)
 - Root cause analysis
 - Understand user behavior throughout the application
 - Threat hunting
- The ability to meet PCI compliance requirements around app security

These abilities increase confidence that threats to modern applications are detected and blocked. A complete understanding of how modern applications function and how to defend them keeps your company out of the headlines. Defense at the perimeter and within the application protects against every avenue an attacker may use to steal valuable data or shut down your APIs.

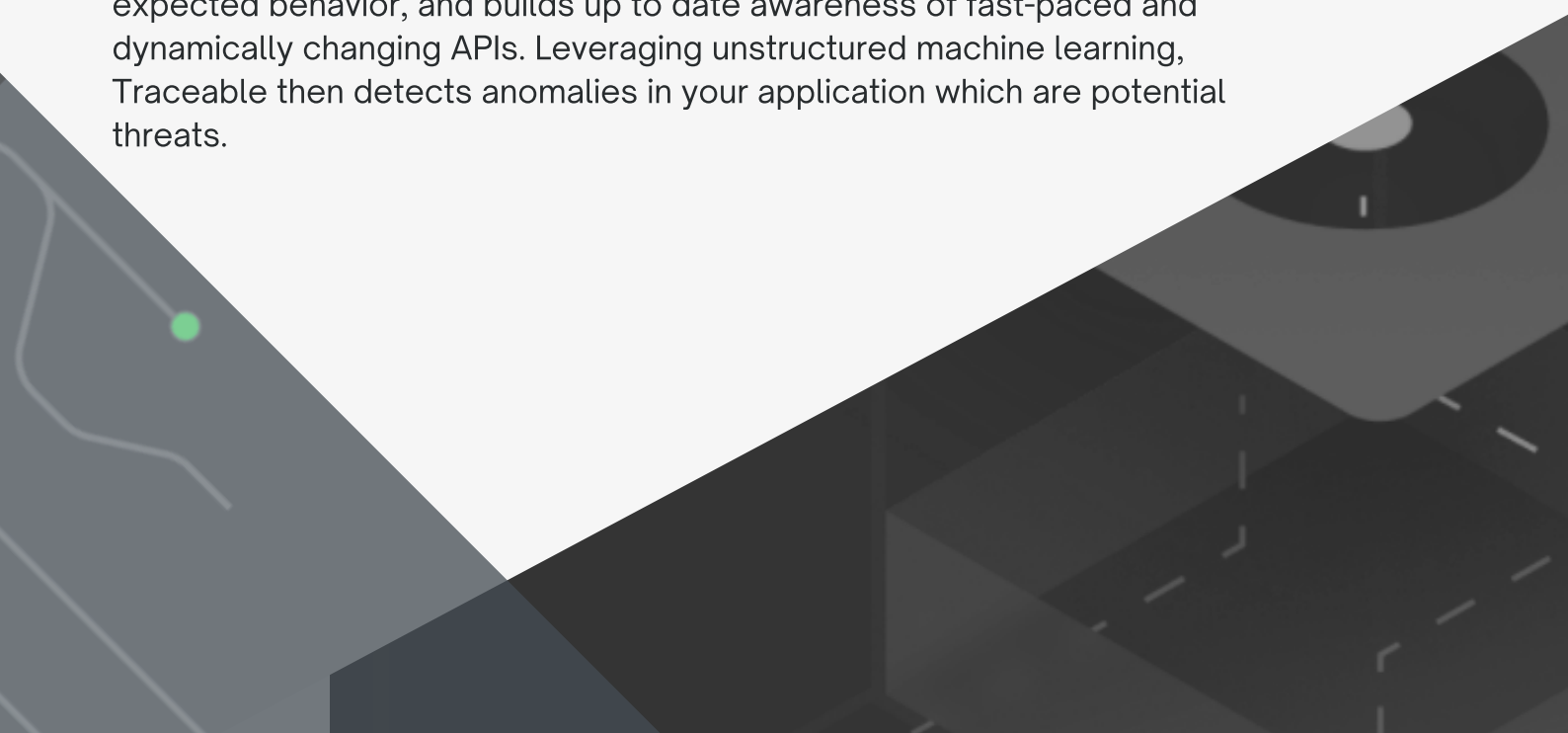
In addition, costs are reduced by deploying a single solution that gives you the protection of NG-WAFs and RASP while adding a holistic view and understanding of your business logic. More attacks are stopped before they cause damage and the right vulnerabilities can be fixed quickly.

How Traceable Helps:

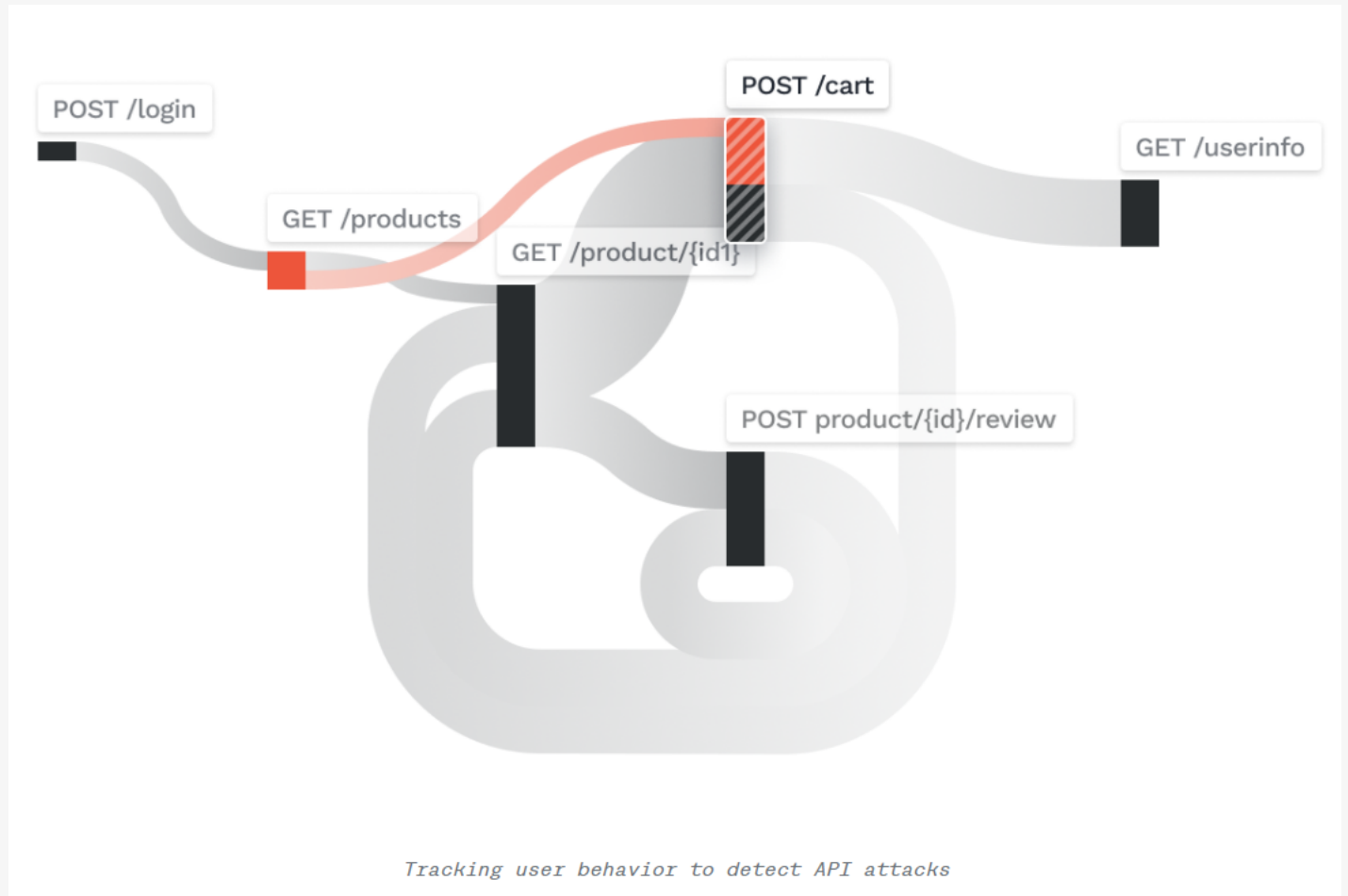
Rather than depending only on fragile rules to protect an application, Traceable understands your application's DNA to detect and prevent threats originating both inside and outside your application. Traceable understands the API level interactions between microservices both public and private and is able to discover anomalies that can be tracked and blocked.

Based on distributed tracing and observability using micro-agents, Traceable has the ability to trace an individual application request from the user at the edge, to the data source and back, across multiple external APIs, internal APIs, and microservices.

Traceable then automatically builds an API inventory, specs, learns expected behavior, and builds up to date awareness of fast-paced and dynamically changing APIs. Leveraging unstructured machine learning, Traceable then detects anomalies in your application which are potential threats.



The below diagram shows a user behavior mapping for an API endpoint. The red services and path show where an attack took place. This attack was prevented, but now analysts know from where it originated and what services to inspect for code vulnerabilities. This level of visibility doesn't exist in RASP or NG-WAF solutions. You'll understand your application better than ever before and be able to find weak spots and fix them.



Traceable Provides

- Detect and block web application attacks similar to a traditional WAF
 - But without the requirement to tune and maintain rules
 - Detect and block OWASP Top 10 attacks, such as
 - Injection
 - XML external entities (XXE)
 - Cross-site scripting (XSS)
 - Fewer false positives through continuous unstructured machine learning
- Detect and block API and business logic based attacks that traditional WAFs cannot
 - By understanding the flow of transactions through the application from edge to data and back.
 - Advanced attacks stealing data by manipulating business logic (focused on fraud)
 - OWASP API Top 10 attacks such as
 - Mass Assignment
 - Broken Function Level Authorization
 - Broken Object Level Authorization
 - Excessive Data Exposure
- Detect and block unknown/unexpected attacks by understanding the baseline API and user behavior and continuously watching for and flagging anomalies.
- Map the current state of application DNA understanding the inventory and structure of microservices and associated APIs to understand App Security posture.
- Easy to leverage data captured through Traceable to help meet PCI compliance requirements

Data Privacy and Compliance

The Problem:

The proliferation of cloud services and cloud-native applications has created unique data privacy and compliance challenges. Critical business data increasingly flows between microservices where APIs may or may not consistently secure and protect the data. This presents very real challenges for IT leaders who are responsible to ensure their systems are compliant and that their controls effectively manage and protect business and privacy related data.

Leaders face exposure to fines and sanctions if they are unable to secure and protect their customer's data. Furthermore, they need to be able to keep up with the rapidly accelerating pace of change in their applications and ensure that their processes and practices are compliant with their controls.

The Solution:

The solution is a system that understands the nature of the data flowing between your microservices. You need to track changes that may impact customer or business-sensitive data and know where sensitive data is potentially at risk.




Data Privacy and Compliance

How Traceable Helps:

Traceable offers API Sensitive Data Detection. It automatically detects incoming and outgoing sensitive data in all API calls. It then creates a downstream data flow trace for all outgoing sensitive data from your application to any backend or third-party system. It then creates data flow maps to show how data flows between microservices.

Traceable helps you visualize data in motion. You'll see where sensitive data is stored, how it's being used, and how it moves throughout the application. You'll see what services touch sensitive data throughout the lifecycle of a request so you can make sure those services are compliant with standards and regulations.

Data classification becomes easier as Traceable tags sensitive data as it sees it move through the application. Any classifications you missed in the planning stages of the API can be found and corrected. The data flow created also assists in auditing and fraud detection by building a comprehensive map of all data moving through the application and analysis of previous requests.



Step Into the Future of Application Security with Traceable

Security tools must continue to change along with applications they protect. The tools we discussed here such as NG-WAFs and RASP are not “wrong” or “bad” tools. They developed in response to shifts in how applications were made.

However, a new shift has occurred within the industry, one that hasn't been addressed fully by the tools available today. Cloud-native, distributed, and API-based applications create new risks and new opportunities for attackers to steal data and attack your system.

Traceable is the next step in application security. It meets the challenges met by cloud-native applications today. Increased visibility, the ability to stop business logic attacks (internally and at the perimeter) in their tracks, and a data flow map to protect against compliance risk all meet the needs of modern application architecture.

If you're interested in seeing what Traceable can do for you, [check out a demo](#) of how it works on Traceable's website.