TRACEABLE

CAN SECURITY KEEP UP WITH THE PACE OF CHANGE?

NEW TECHNOLOGIES ARE REVOLUTIONIZING SOFTWARE DEVELOPMENT, BUT THEY POTENTIALLY COME WITH THEIR OWN VULNERABILITIES. IS MODERN APPLICATION DEVELOPMENT LEAVING YOU OPEN TO ATTACK?

Doc ver: 2021-01-20-01

“THE ONLY WAY TO MAKE SENSE OUT OF CHANGE IS TO PLUNGE INTO IT, MOVE WITH IT, AND JOIN THE DANCE.” — ALAN WATTS

If you’ve spent any length of time in application development, you’re familiar with change. It’s the only constant.

And along with how we build applications come changes in the techniques used to keep them secure.

Securing modern applications requires more diligence than ever before.

New technologies and techniques such as GraphQL and cloud-native platforms promise outstanding benefits for developers and organizations. But these improvements come with a potentially devastating price to pay: it's not always clear how to secure them, nor what vulnerabilities they may introduce.

If organizations are slow to learn the differences between new and old technologies from a security perspective, they’ll be left open to attack.

Let’s look at how modern application development has changed the security and threat landscape and how companies can overcome these challenges.

Traditional Web Applications: Ordering From a Menu

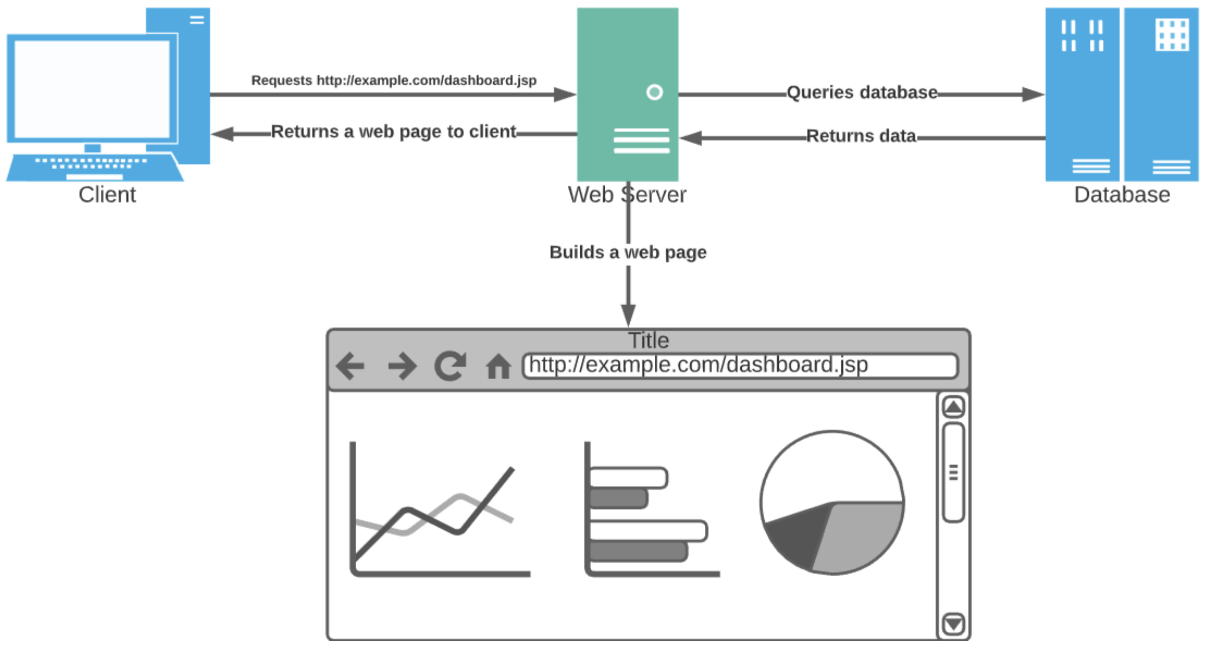
The traditional web application operates like ordering food from a restaurant menu. From a list of dishes available you ask for one. The chef decides which ingredients are necessary and how the plate looks when it arrives.

Let's say you want to see a dashboard of marketing metrics, and thus your browser retrieves example.com/dashboard.jsp. The client doesn't control what's on the dashboard; it just knows to ask for it.

The web server decides what data goes into the dashboard and where the data goes. The client simply asks for a resource and displays what the server returns.

The web server processes the request in three steps:

1. **The web server pulls data from the database.**
2. **The server builds an HTML page with the data placed inside.**
3. **The server returns the rendered HTML page to the client.**


Requests http://example.com/dashboard.jsp
Returns a web page to client
Client
Web Server
Queries database
Returns data
Database
Builds a web page
Title
http://example.com/dashboard.jsp

In a traditional model, the client typically has very little code. It requests a URL from the server and displays what comes back. All rendering and data processing happens on the server.

Modern Web Applications: Buffet Style

Modern web applications put the client in control. It’s more like a buffet than a sit-down restaurant.

The client is often a Single-Page Application (SPA), or an application that dynamically updates the same page instead of always requesting a new one from the server.

In a buffet, you take your plate and pick and choose the foods you want. Each dish can have multiple configurations based on your needs at the time.

SPAs work like those plates. Instead of a single server creating an entire page and returning it to the client, the client knows how to make the page and asks only for the data required to do it.

Typically, Application Programming Interfaces or APIs provide this data to the client. Business logic is exposed to clients using APIs that return only what the client needs. Maybe the client only wants the last 20 notifications for an account or the ten most recent emails.

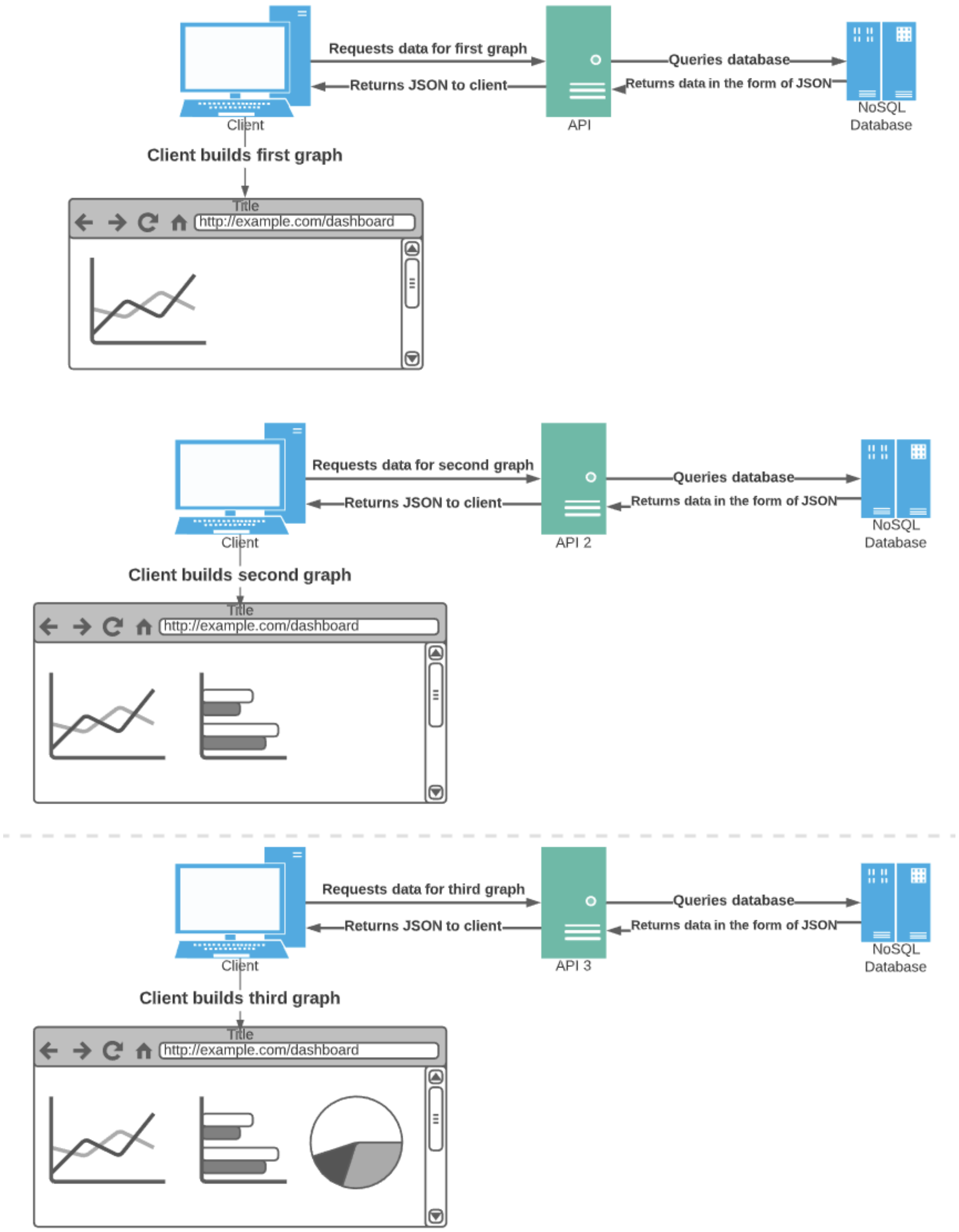
The use of APIs for business logic and data retrieval opens up the world of application development. You no longer need to build a web application. You can use a mobile application instead. You can use any technology or form factor to access the data, from mobile to Internet of Things (IoT) or other developers building applications of their own .JSON, a lightweight data format, is one of the most useful enabling technologies for modern applications. It allows all of the APIs, the client, and the database to store and transmit data using the same format. There are no data transformations necessary, which helps keep the application responsive.

Going back to the above dashboard example, an SPA wouldn’t ask the server for the entire dashboard at once. Instead, it would ask the appropriate APIs for the specific data it needs to build the dashboard itself.

The desire for APIs and for ultimate flexibility and speed of development gave way to the creation of microservices. These services are intentionally small to be updated independently of each other and more frequently.

SPAs emphasize the client, leading to more complex code in the clients and the risk of security vulnerabilities in the clients (more on that later). But this change has opened up transformational technologies like IoT devices.

Modern Web Applications: Buffet Style



Requests data for first graph
Returns JSON to client
Queries database
Returns data in the form of JSON
Client
API
NoSQL Database
Client builds first graph
Title
http://example.com/dashboard
Requests data for second graph
Returns JSON to client
Queries database
Returns data in the form of JSON
Client
API 2
NoSQL Database
Client builds second graph
Title
http://example.com/dashboard
Requests data for third graph
Returns JSON to client
Queries database
Returns data in the form of JSON
Client
API 3
NoSQL Database
Client builds third graph
Title
http://example.com/dashboard

In a modern application, the client asks for the data it requires to build the page itself. Small, self-contained services do one thing well.

TRACEABLE

TRACEABLE

FROM MONOLITHS TO MICROSERVICES – A RESPONSE TO THE ACCELERATING PACE OF CHANGE

A common theme in the evolution of application development has been toward breaking functionality and features into smaller and more granular components. Just as with the SPA example above, the functionality of the web page is in isolated components. The same trend is true when you look at application architecture in general.

Where a legacy application might have been designed with separate modules and functions, each internally making calls to each other, modern design now breaks out separate functions into discrete services.

Consider an ‘address verification’ service that can be used by many other applications. The functionality of validating an address is decoupled from the ‘customers’ that use the service, which introduces value into the delivery lifecycle. This approach enables development teams to focus on smaller, more granular changes and therefore shorten their cycle time and accelerate delivery.

Business leaders everywhere are demanding faster delivery from their technology teams. As a result, teams have adopted new ways of working and new ways of architecting their solutions. Enter Agile, CI/CD, and DevOps tools. Similarly, application development and architectures have evolved to increase velocity.

We are implementing features and products and using technology that were not invented 18 months ago. No longer can we afford these large monolithic programs that go on for two to three years. We know that what we set out to do at the beginning of that time is not what we will finish out doing. So, we are focusing on very rapid delivery cycles, asking ourselves: How do we mobilize a project very quickly? How do we use the right delivery techniques to work through it quickly? How do we get product into market or to customers or into the business?”

-- Bronwyn Clere, Executive Director for Capital Planning & Delivery, at Telstra Corporation, PMI Pulse of the Profession Report 2017

A Tour of Cloud-Native Architecture

The increasing pace of change would put significant stress on any IT infrastructure. But cloud-native architecture has fundamentally changed how applications can be built and scaled.

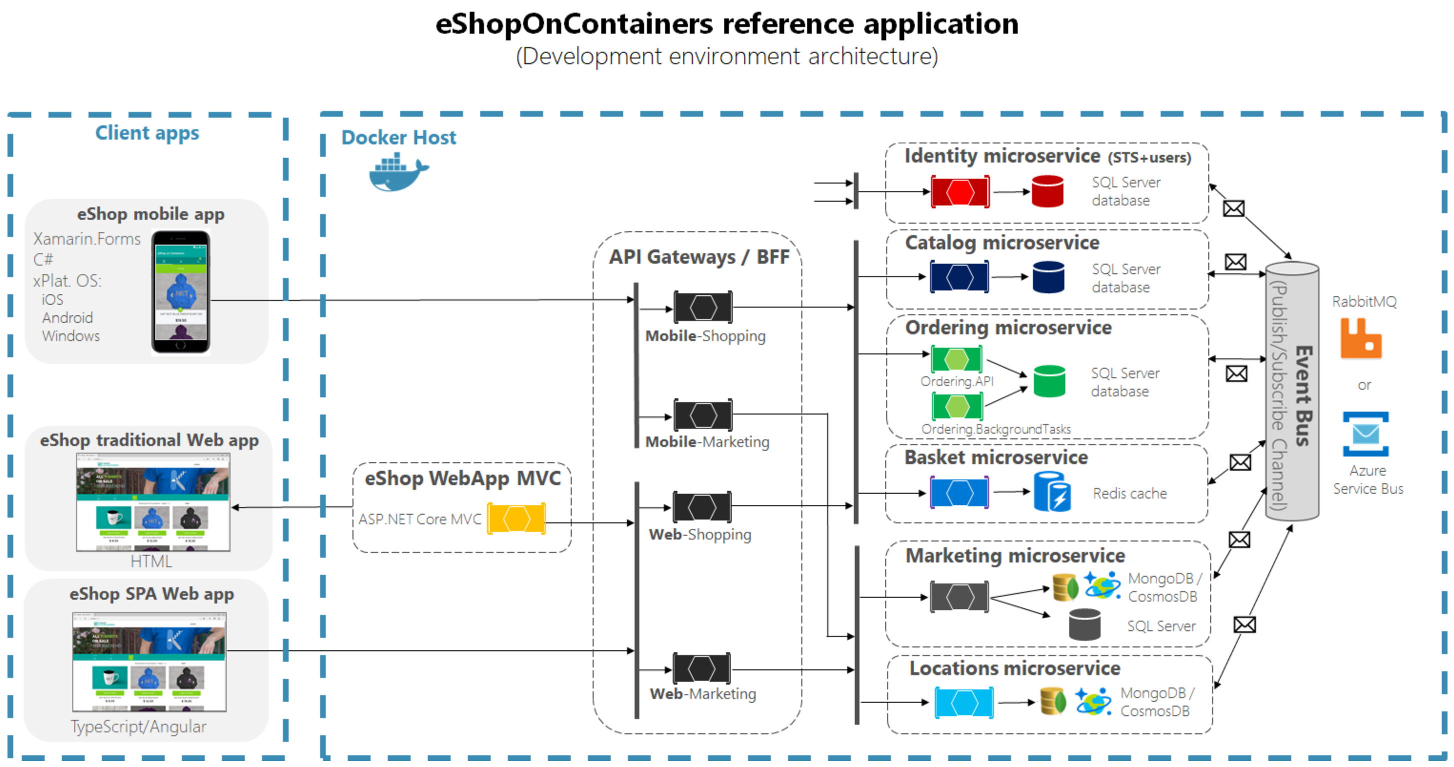
Cloud-native technologies, such as containers, service meshes, microservices, immutable infrastructure, and declarative APIs, empower developers to build and run scalable applications on public, private, and hybrid clouds.

Cloud-native architecture focuses on creating loosely coupled services with high resiliency. Developers can make changes frequently without negatively impacting the entire system.

Below is a diagram of Microsoft's eShopOnContainers open-source application. It demonstrates what a real-world application looks like using cloud-native technologies.

In this example, the development team building the Marketing microservice can make changes as often as they want without affecting the ordering or cart functionality. Different technologies can be used within each service, and each service has a separate datastore.

Flexibility and speed are the hallmarks of modern development.



The bulk of cloud-native applications include many microservices deployed in containers. Each service is isolated from the others and can be destroyed and recreated easily without affecting other services.

3 Necessary Mindset Shifts For Cloud-Native Success

The pace of change affects more than technology. Processes, management styles, and even ways of thinking have to adjust to make modern application development successful within your company.

DELIVERY: THINK AGILE AND DEVOPS

Because of the intensive need to shorten delivery cycles and to automate delivery, cloud-native architectures embrace delivery practices that support short and responsive turnaround.

Developers can deploy new code in seconds, with some companies pushing new code to production hundreds of times a day.

PRODUCTION: THINK PRODUCTS, NOT PROJECTS

The days of ‘throw it over the wall’ and let Operations manage the application are over.

In order to successfully scale adoption of cloud-native applications, you need to explore and embrace “Product Management” over “Project Management” as a practice to manage the application's full lifecycle..

SECURITY: VALIDATE EVERYTHING

The traditional approach of building a fortress of defensive walls and moats around the application and then "trust" the activity inside is secure, is insufficient in a world where individual elements of the application are expected to communicate with other components.

The new security paradigm is one where each microservice must have security built-in, They need to embrace "zero trust" as a policy for how they operate. Validate every request before acting on it.

TRACEABLE

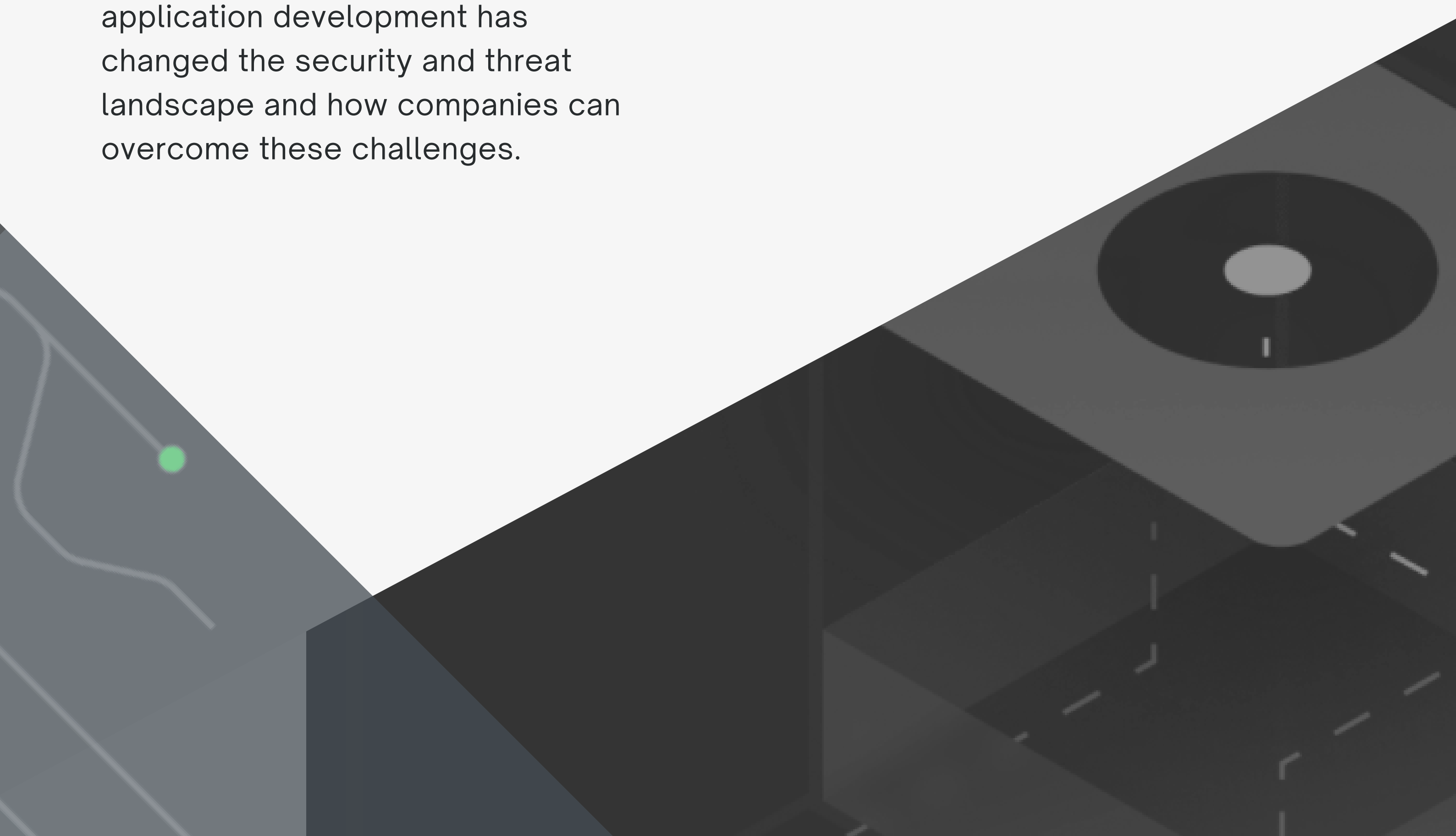
PROTECTING AGAINST THE HIDDEN THREATS OF NEW TECHNOLOGIES

New technologies and techniques such as GraphQL and cloud-native platforms promise outstanding benefits for developers and organizations. But these improvements come with a potentially devastating price to pay: it's not always clear how to secure them, nor what vulnerabilities they may introduce.

If organizations are slow to learn the differences between new and old technologies from a security perspective, they'll be left open to attack.

Let's look at how modern application development has changed the security and threat landscape and how companies can overcome these challenges.

New technologies are revolutionizing software development, but they potentially come with their own vulnerabilities. Is modern application development leaving you open to attack?



High Rate and Cost of Hacking Attempts Requires Action

39
seconds

43
percent

3.9
million

INCREASED ATTACKS

A hacker attacks every 39 seconds

SMALL BUSINESSES TARGETED

43% of cyber attacks target small businesses

EXPENSIVE DATA BREACHES

The average cost of a data breach is $3.9 million

NEW TECHNOLOGIES, NEW THREATS

Fearmongering isn't our style, but it's dangerous to ignore the reality of the cyberworld. Hackers have found the new oil: data. And they'll go to great lengths to take it from those who have it.

6
months

FREE REIGN

The most devastating statistic may be this: It typically takes about six months for companies to discover a data breach. How much damage could an attacker do with six months of free playtime in your network?

TRACEABLE

INCREASED DATA PRIVACY LEGISLATION FORCES COMPANIES TO SECURE DATA

Data privacy legislation has increased in response to growing concerns from consumers.

The EU's Global Data Protection Regulation (GDPR) fundamentally changed how companies handle customer data. 88% of companies spent more than $1 million preparing to implement the rule.

The GDPR did more than increase data-protection requirements. Legislators saw for the first time the possibilities of holding companies accountable for how they use, handle, and protect customer data.

This game-changing regulation has led to several federal and state laws in the United States with similar goals.

Federal laws that impact the collection and storage of consumer data include:

- The Federal Trade Commission Act
- The Children's Online Privacy Protection Act (COPPA)
- The Health Insurance Portability and Accounting Act (HIPAA)
- The Gramm Leach Bliley Act
- The Fair Credit Reporting Act

Many states have also issued regulations that govern the use of their residents' personal information:

- California Consumer Privacy Act (CCPA)
- New York's Stop Hacks and Improve Electronic Data Security (SHIELD) Act

The trend is clear. Companies must take better care of their customers' information or face consequences that can ruin their brand, destroy customer loyalty, and lead to enormous fines and other legal penalties.

TRACEABLE

APIS REQUIRE NEW APPLICATION SECURITY TECHNIQUES

Web application security practices have changed along with application architecture. The OWASP Top 10 has been the go-to list of security vulnerabilities for some time. However, this list isn't as applicable to modern API-based applications.

As development frameworks have evolved, many have added protections out of the box against the most common vulnerabilities. Because of this, the proper configuration of development frameworks mitigates and eliminates many OWASP Top 10 vulnerabilities.

Cloud-native architectures are a different way of building applications. Much of security is context, and the context completely changes in a cloud-first environment.

Issue	Solved By
SQL Injection	ORMs
CSRF	Use of authorization header
XSS	Clients are responsible
Path Manipulation	Cloud-based storage
XXE	JSON

This table shows several "classic" web application vulnerabilities and the ways they are solved by default when using APIs.

Does this mean API developers can breathe easy because all of their security concerns are gone? Changing the context changes the attack vectors and vulnerabilities. The "old" vulnerabilities make way for new threats that developers have to learn how to mitigate.

TRACEABLE

API SECURITY IN THE CLOUD-NATIVE WORLD

APIs and microservices provide unprecedented flexibility and speed of development.

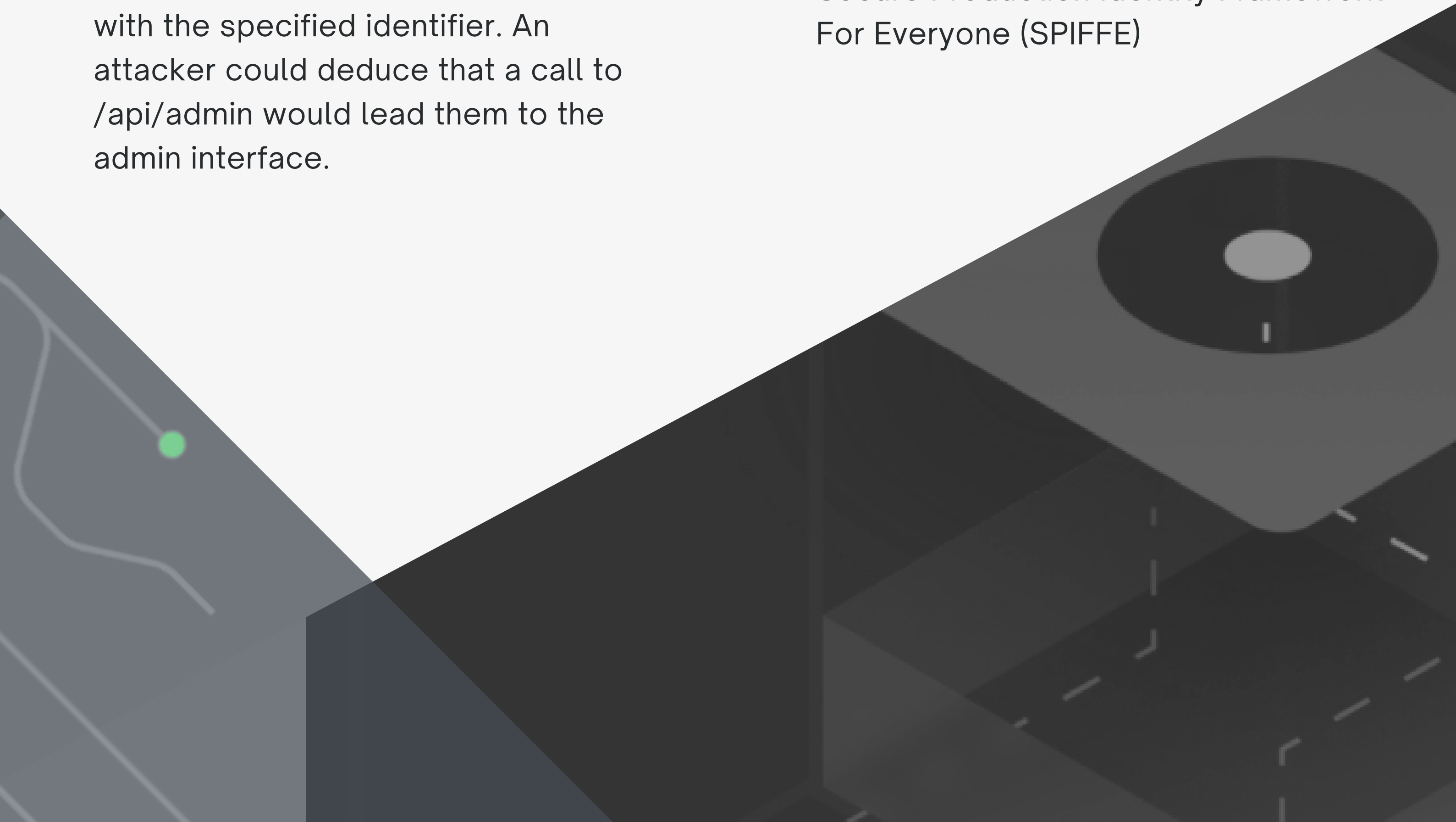
However, new technologies used to build APIs also give rise to new threats. For example, a set of APIs that each expose a small part of the business logic could make it easier for attackers to figure out your backend logic. They can watch network connections and see the URLs the client is calling and in what sequence.

APIs are predictable. Most follow Representational State Transfer architecture or some variation of it. Because of this, common patterns develop and attackers know how to exploit them. An API with the URI /api/users/{id} is searching for a user with the specified identifier. An attacker could deduce that a call to /api/admin would lead them to the admin interface.

Authorization is the most daunting challenge facing API developers. Monolithic applications log in the user once, store authorization information in the session, and never think about it again. APIs are loosely coupled, platform-agnostic, and built to be lightweight.

Solving authentication in APIs is a bigger topic than we have space to discuss, but here are ways to implement authentication and authorization between APIs:

- Core technologies
  - OAuth2
  - OpenID Connect
  - Mutual TLS (client and server certificates)
- Frameworks to help manage API security
  - API gateways
  - Service meshes
  - Secure Production Identity Framework For Everyone (SPIFFE)



THE CHANGING NEEDS OF MODERN APPLICATION SECURITY

Application development is undergoing a renaissance. Towering monoliths and slow delivery cycles have given way to agile microservices and daily production releases.

Along with these changes come new threats and new mitigation strategies. Traceable AI is built to detect, warn, and protect against these new threats to cloud-native applications.

View a demo of Traceable today. You'll see first hand how Traceable AI reduces breaches, false positives, and cost while securing your next-gen application.

“TIMES AND CONDITIONS CHANGE SO RAPIDLY THAT WE MUST KEEP OUR AIM CONSTANTLY FOCUSED ON THE FUTURE.” -- WALT DISNEY