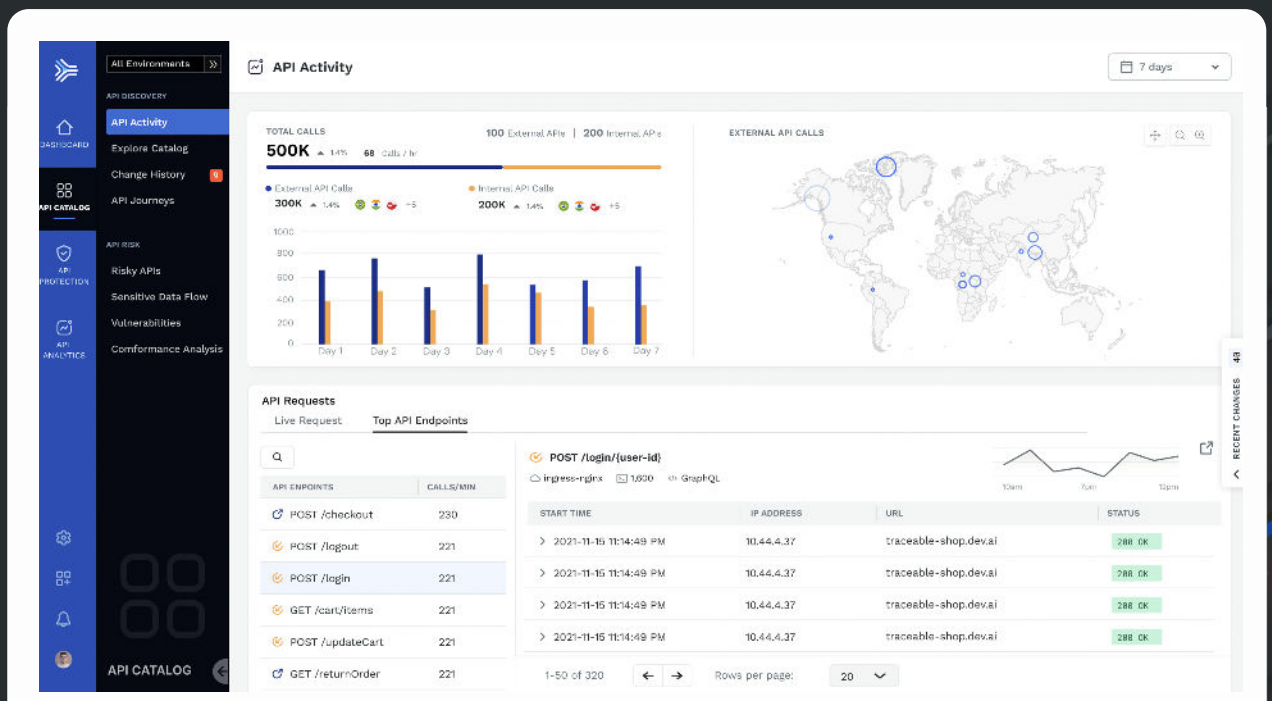
TRACEABLE

Security Observability: Why Tracing?



TRACEABLE

Executive summary

Detecting a cyber attack can take more than 200 days. During this time, attackers might be exploiting your system while you are completely unaware. Cyber attacks can cost affected organizations as much as $13 million per year.

What if we could reduce the time to detect a cyber attack all the way down to zero? How much money could you save by detecting and stopping an attack while it is still happening?

And what if we could do even better than that? What if we could detect security vulnerabilities before an attack even occurred?

Security observability is critical to achieving these goals. Let's take a look at what security observability means and how distributed tracing, which is one part of observability, can help to defend your business from cyber security threats.

What is Security Observability?

Knowing what is happening inside software is the best way to learn how to defend it. Observability refers to the ability to see the state of a system at any time. It's like opening up a watch and watching the gears turn.

Applications and services constructed using APIs and microservices have an advantage over other systems because you can more easily observe what's happening inside of them. You add hooks to the APIs and microservices and use additional tools to collect the details.

This presents us with an opportunity to better secure applications and services against cyber attacks by allowing us to see exactly what is happening inside the system.

The Pillars of Observability

Three types of data are needed to make a system observable. These are commonly referred to as the "Three Pillars of Observability:

Logs

Raw data, recording every activity that happens in the system

Metrics

Aggregated performance metrics, such as CPU load and memory usage, collected as min, max, average, number of counts, and so on

Distributed Traces

The complete path a request follows as it flows through the microservices

Data Analysis

Data analysis/visualization tools (sometimes referred to as the "fourth" pillar).

Yes, that makes 4, but this item isn't data. The above data is too complex to digest directly, making these tools paramount to achieving observability.

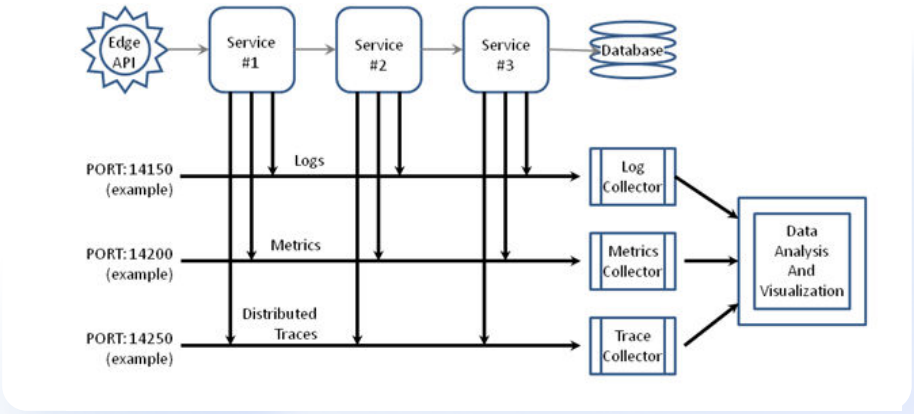


Figure 1. Sending observability data from the microservices to their respective collectors.

The three pillars are shown in the diagram in Figure 1. Each microservice in the diagram contains additional code that sends logs, metrics, and traces to their respective collectors. These collectors gather the data, store it, filter it, and pass it forward to analysis and visualization tools that make the data usable by humans.

Traditional security tools, such as SIEMs, make use of logs and metrics to secure a company's assets. However, these tools fall far short of protecting applications.

Limitations of Logs and Metrics for Protecting Apps

Logs and metrics begin to paint a picture of what is happening inside your APIs/microservices. But the picture is incomplete. Logs and metrics can help developers ensure that the system is working properly and they can help with solving problems during debugging. But they do not provide enough information to protect the system from cyber attacks.

Logs: Just the Facts

Every time a microservice is called or a request is generated, a message describing that activity is sent to the log collector.

This message contains information about what has happened, but it is in a very raw form. You can think of a log as a fact. It doesn't provide any real understanding of what happened in the system.

If this GET request was the first step in an attack, we can't tell that from looking at the logs alone.

```
127.0.0.1 - ServiceA [10/Dec/2020:13:55:36 -0700] "GET /apache_pb.gif
HTTP/1.0" 200 2326
```

Figure 2. Metrics example, showing the number of calls received for the Login service. What do the spikes reveal? Are they attacks, or are they normal user behavior?

Logs: Just the Facts

Metrics are aggregated data collected from microservice activity. Metrics may include things like CPU usage, memory usage, execution time, number of calls or requests per unit of time, and so on.

Each metric is associated with the microservice that recorded it.

Figure 2. shows activity over time for a login service. The login activity is usually uniform and consistent, but there are very large spikes that occur occasionally.

The spikes show that there are bursts of login attempts that are occurring occasionally on a system. They could indicate that a hacker is sending a brute force attack, attempting to guess usernames and passwords for administrators in the system. Or they could indicate a hacker is trying to overload the server with a Denial of Service (DoS) attack.

But these spikes might indicate normal user activity. Perhaps many users are running an app simultaneously because they just saw a compelling television commercial that was aired during the Super Bowl halftime. Or maybe everybody is contacting Uber to request a ride at the same time because a Rolling Stones concert just ended.

Another possibility is that these bursts of activity indicate a bug in the system. Maybe there is some sort of unforeseen infinite loop that is causing an unexpected burst of activity.

We just don't know what is causing these bursts. To gain the insight we need to secure APIs, we need to know what happened before this metric was generated. We need the ability to work backward.

Distributed Tracing: Context and Causality – Why Things Happen

Distributed tracing can help us understand the reason things happen. When we know why things happen, we can determine if an attack is underway.

Distributed tracing is a form of tracing that is specially adapted to microservice architectures. It allows us to observe each request traveling through the system from its beginning to its end. We can use traces to improve performance and understand what typical behavior looks like.

But traces also give us the power to identify attacks on our microservices and stop them.

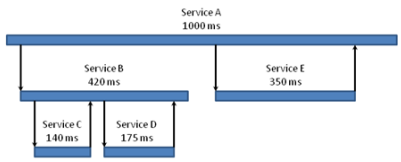


Figure3. A "span", or trace, showing the path a request follows through the API±microservices system.

Traces show the entire path a request followed through the edge API and microservices. They can tell us:

- If the request went through all of the required services, including authentication and authorization checks
- If the request called any unexpected services or granted access to unauthorized data
- Where the request entered the system. (If a request entered without going through an edge API, also known as an API gateway, there may be a problem)
- The IP address that originated the request
- And if they are working especially well, traces might even help identify an attacker.

Data Analysis and Visualization Tools: Seeing What Humans Can't

In a real system, there may be hundreds of different paths requests can take, just like the one shown in figure 3. And each request could touch tens or even hundreds of services. While distributed tracing gives us the ability to observe all of these, the complexity makes it impossible for humans to gain much insight.

Machine learning and data visualization tools can identify patterns in the data, in order to identify normal behavior and abnormal behavior, and can even create alerts when abnormal behavior is detected.

These patterns of normal and abnormal behavior help us to identify security vulnerabilities and attacks within APIs and microservices.

Tracing to Reveal Vulnerabilities

The diagram in Figure 4 shows the combination of traces through a system. We can use this diagram to identify possible security vulnerabilities.

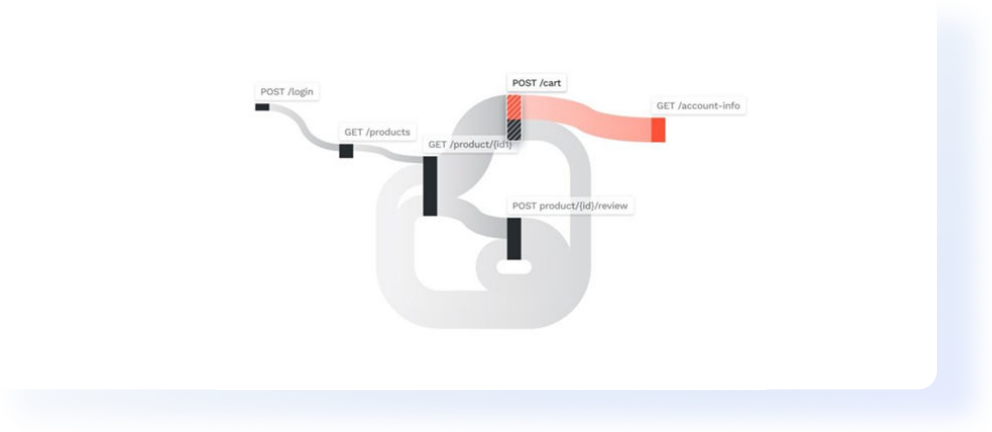


Figure 4. Possible security vulnerability, indicated by the red-colored trace. The diagram shows that this group of requests may inadvertently grant unauthorized access to the user's account info.

Figure 4 shows a pattern that contains an "Excessive Data Exposure" vulnerability, which is vulnerability #3 in the OWASP API Security Top 10. The pattern shows that when a user views a shopping cart, the API will also return the user's account information. This unnecessary information may include personal information, account information, or financial information a hacker can exploit.

These insights help developers recognize potential security vulnerabilities and correct the code to eliminate them.

Tracing to Reveal Vulnerabilities

Tracing can monitor each request and compare it to established norms. The system can generate an alert if a trace deviates from this expected behavior.

Data Analysis and Visualization Tools: Seeing What Humans Can't

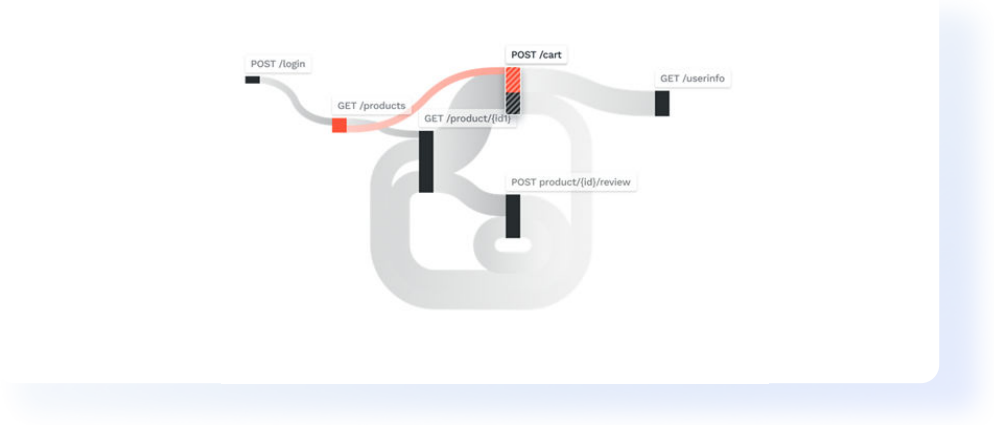


Figure 5. Possible attack in progress. The diagram shows that a possible attacker may have found a way to bypass a required authentication or authorization step.

Figure 5 illustrates a pattern that could indicate either a "Broken Object Level Authorization" attack or a "Broken User Authentication" attack, which are #1 and #2 in the OWASP API Security Top 10 vulnerabilities.

This figure indicates that an attacker has found a way to enter another user's shopping cart and possibly obtain information for that user, by bypassing either an authorization or an authentication check.

Typically, tracing allows us to go back in time to find attacks such as the one above. Can tracing help to stop attacks in real-time?

Yes, tracing provides the information we need to detect and stop attacks in real-time. However, more is needed. A sophisticated AI on top of a tracing mechanism would be able to notice abnormalities humans may miss and can stop abnormal behavior before any damage is done.

Tracing plus AI gives us real-time security observability. In a future blog post, we'll discuss how AI takes what tracing gives us and uses it to create real-time protection against attacks.

Using Distributed Tracing to Defend Your APIs

Cybercrime is expensive, and it can take a long time to detect an attack. Attackers are creative, and older techniques using logging and monitoring do not give enough insight to recognize an attack when it happens.

Security observability uses internal states to thoroughly comprehend what is happening inside your API/microservices system with the goal of protecting them from attack.

Observing your software, using logs, metrics, and most importantly, distributed traces gives you the opportunity to identify attack patterns, eliminate security vulnerabilities, and block attacks before they can cause serious damage. View a recorded Traceable demo to see for yourself.

About us

Traceable AI was founded by third-time entrepreneur Jyoti Bansal and Sanjay Nagaraj. Bansal and Nagaraj saw the massive adoption of cloud-native architectures firsthand during their time at AppDynamics and founded Traceable AI as a result to protect applications from next-generation attacks.

Traceable AI applies the power of machine learning and distributed tracing to understand the DNA of the application, how it is changing, and where there are anomalies in order to detect and block threats, making businesses more secure and resilient.

www.traceable.ai