imperva

WHITEPAPER

How the Evolution of Agents has Been Essential for Modern Database Security

In today's data driven world, every organization's most important asset is their data. Accordingly and similarly to other protected components like applications, web and pherpirals gateways, databases require a dedicated security solution as well.

An essential database security solution must provide fundamental features such as: reporting, asset discovery, vulnerability management, and most importantly, database auditing capabilities.

Auditing is the most comprehensive component of a database security solution, it is a key element. Auditing implementation requires exceptional development and configuration for it to function flawlessly.

As of today, there are two main methods to deploy a database security solution with an audit trail:

1. Agentless -Extract the trail from the asset's native audit trail
2. Agent based -Extract the trail by parsing the asset network traffic

Each method has its pros and cons and each mainly covers different use cases. Generally, native logs are better suited for forensics and compliance use cases, whereas the latter is better suited to security and enterprise-grade deployments.

This whitepaper outlines the technological evolution of data security agents over the last two decades. Rather than focusing on an actual product development and chronological chain of events, we'll simply discuss the evolution of the technologies in use, emphasizing the reasoning and limitations of each technology and outline the key points that led to adoption of newer solutions that may address security gaps that have evolved.

Let's start at the beginning

As we continue with a global digital transformation, more valuable digital data is being created. Yet, since the early days of the digital journey, a dedicated solution for data protection in the digital space was required.

In order to maintain regulatory compliance and detect sensitive data breaches in time to stop them, visibility and control are required to be major components of this solution.

Database Activity Monitoring (DAM) is the holistic solution that has evolved over time. And our story of agents' evolution begins exactly in those early days.

In the early internet days, network operators were not too aware about cyber risk so the security posture was not too sophisticated. To generate an audit trail, one could simply sniff the bits and bytes flowing through the database's wire and pass them through a protocol-specific parser.

It was that easy: one data line, no extra bytes, all you have to do is "listen" to that line and gain full visibility into the data, and accordingly have a full audit trail. Similarly, it was also possible to configure TCP mirroring for the relevant network adapter to send traffic to the parser.

As you can see in Figure 1, simple packet sniffing is enough to make the solution work.

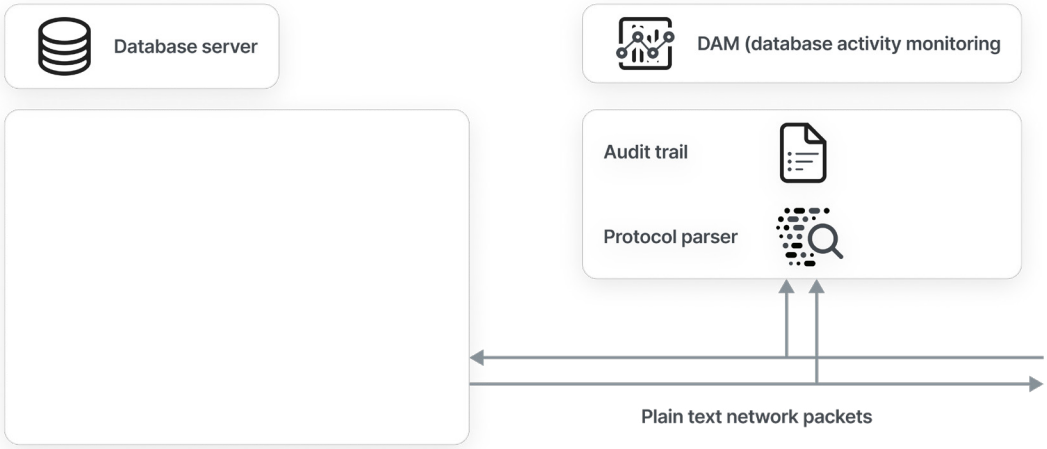

Figure 1

This simplistic approach was sufficient as encryption methodologies and layers protection approaches were not widely spread. Over time, encryption gained popularity and was widely embedded across the internet and organizations, making it harder for this naive eavesdropper to inspect your database traffic.

As encryption became a standard, the ability to decrypt the cipher blobs was added to cover this gap. Without getting too technical about how encryption works, let's just say that if you have a decryption key, then, ultimately, you have what it takes to decrypt the traffic. The problem in this scenario would be, how one gets the decryption key. Some key-exchange-protocols allow you to extract the decryption key from the tcp stream if you have the server's secret.

Fortunately, back then, the most common and used key-exchange-protocols allowed it. Thus, asking the customer to simply upload his server's private key into the DAM service, allowed the parser to decrypt the cipher blobs before parsing them (Figure 2).

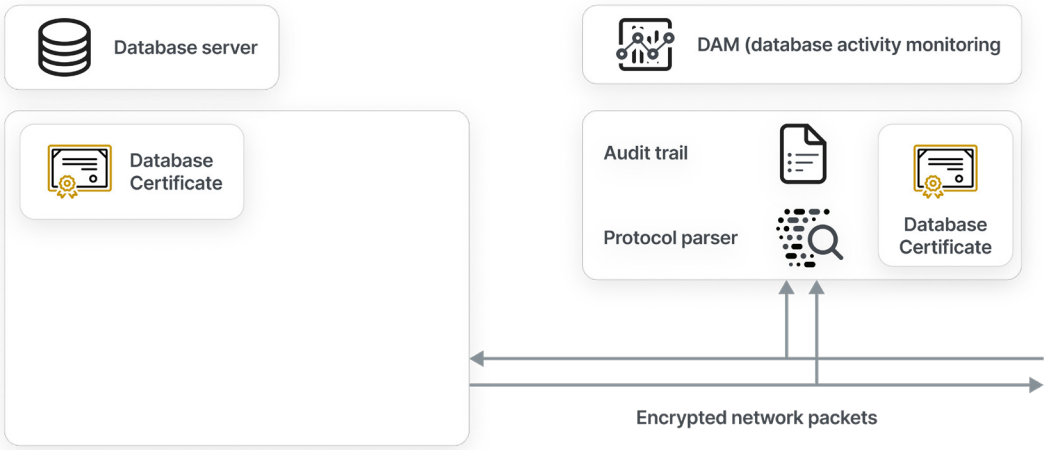

Figure 2

The main issues with both these two early life solutions, is that even though it sounds feasible, it was really hard to deploy it into a data center. Unlike WAF (Web Application Firewall) which usually sniffs traffic from the data center gateway - a single point of inspection, DAM must sniff traffic from every data asset and there is no single point of inspection. This problem is exaggerated with every new database you need to protect, which makes it really not scalable when we deal with large amounts of data and database servers.

Agents are born

Exponential increase in digital data as well as a new requirement to inspect local data, making the previous methods not scalable and irrelevant to the new times. This led to the rise of agents inside the database server machines. The fact that a client agent is installed on a database allows visibility into the data and great granularity for the type of data we need to extract. Those agents were quite simple when they were young. All they needed to do was to follow the same flow as described above - sniff the database network traffic and send it to the DAM service. One might even say that all they were in this stage is just a fancy implementation of TCP mirroring with fine-grained abilities.

Sniffing is a relatively simple task that is widely used and achieved by using the same interface that other network capturing tools use. If you are familiar with Wireshark or tcpdump then you probably guessed it yourself already (Figure 3).

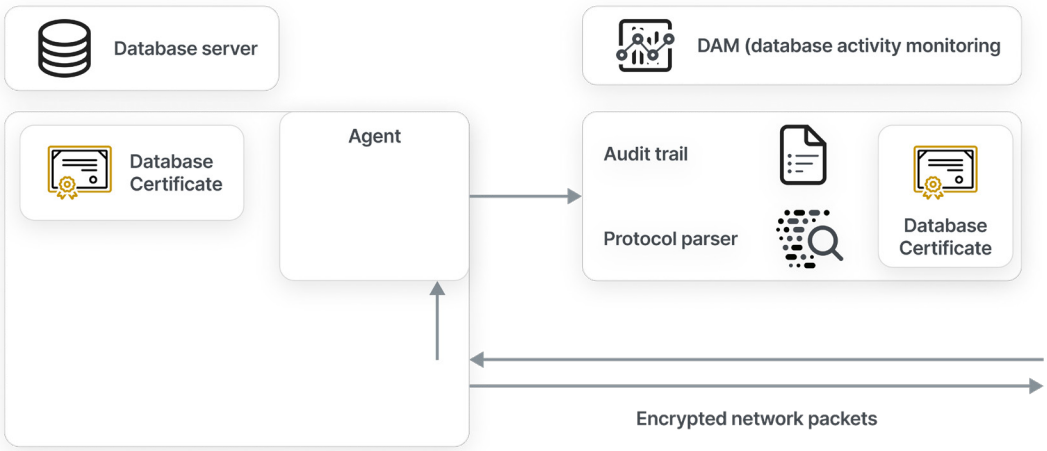

Figure 3

Having a foothold in the database machine allowed the agents to collect additional information without changing the deployment procedure. They didn't want to settle for collecting only TCP packets, they wanted more than that. Accordingly, after being deployed on the servers, evolving into the next phase was somewhat inevitable.

Matured agents

To provide a superior auditing trail, agents then were loaded with their code into the server's operating system. Simply put, the operating system manages pretty much any resource on the server, so if you load your code there, you can do practically whatever you want. In other words, they loaded their kernel extension into the kernel (underlying operating system).

This code made it possible to intercept not only TCP connections but all other connection types as well. Among which their new connections were UNIX domain sockets, pipes, named pipes, and shared memory.

It also allows the agents to give an additional context to the new connections - which organization user created this connection, what commands are run, with what arguments, etc. The audit trail was never richer both for quality and quantity (Figure 4).

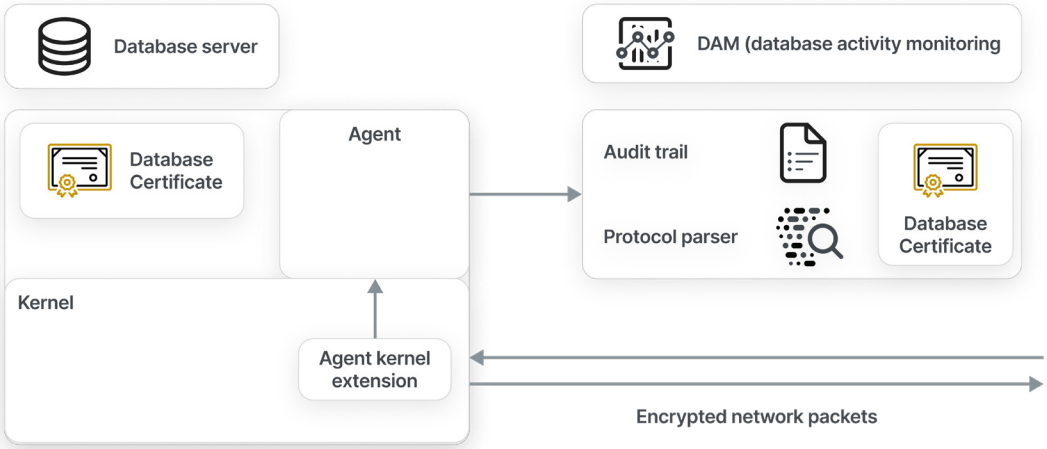

Figure 4

Do you remember we said earlier that "Some" key-exchange-protocols allow you to decrypt the traffic? Guess what happened next?

That's right, traffic encryption fought back and denied DAM from decrypting the cipher blobs. There was a change in cipher suites standardization which started to shift toward "Diffie-Hellman" based key-exchange algorithms which prevented DAM from decrypting the cipher blobs. Having the server private key wasn't enough, a new way was needed.

This change forced the agents to make an evolutionary leap.

The rise of database application, in-memory agents

Following the mass expansion and implementation of encryption technologies there were two available options:

1. Extract the actual decryption keys from the database process memory, which later be used to decrypt the streams
2. Extract the actual plaintext traffic from the database process memory - Outbound traffic before the encryption and inbound traffic after the decryption

Both options are valid. The second option is considered to be more stable over the years and compatible with multiple and new types of cipher suites.

Accordingly, new tools were developed to allow the agent to load its own code into a foreign process memory space (the database's process in this case) and instrument specific internal functions to extract the actual plaintext stream. This tech is highly dependent on the underlying operating system, runtime environment, and server processor architecture. In addition, picking the specific database

application's functions is tricky and requires extensive preliminary research. Addressing those development challenges is costly in terms of development resources, yet doable (Figure 5).

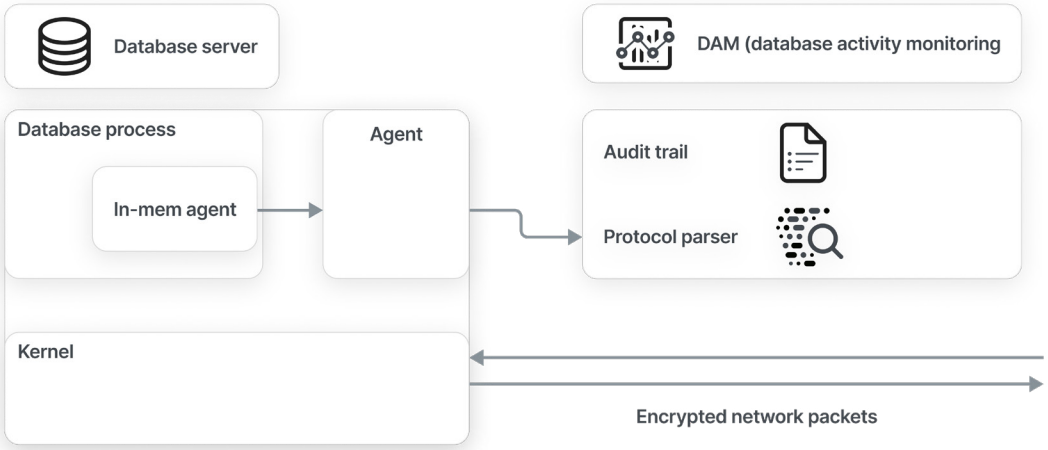

Figure 5

Having reached this point, it seemed as if the agent could do whatever it wanted wherever it wanted. It possessed the core technology to sniff, monitor, and affect everything on the server. Whether it requires something from a specific process memory space or from the kernel. The core technology existed and only minor adaptations were needed to close yet another coverage gap.

It is true that in some cases there were some exceptions that required additional significant development efforts, but the true focus revolved around what adaptations are needed to fully utilize the current technology and address those gaps. It felt like the famous quote that is usually attributed to Lord Kelvin (1824-1907):

"There is nothing new to be discovered in physics now. All that remains is more and more precise measurement."

Along came the cloud...

Within the adoption of cloud technologies the rules of data protection agents' game have been changed. When someone refers to his "database in the cloud", he means one of the following:

1. DBaaS (Database as a service) - a database that is managed by the cloud provider. It moves the management responsibility from the owner to the cloud provider, preventing him (and us) from installing our agent.
2. Database containerization - running a database on some sort of containerization environment - Kubernetes, OpenShift, etc.
3. *Database on virtual server - This type of cloud deployment leaves the management responsibility to the owner and makes it the same as an old-fashioned "on-premise" database.*

When we say on-premise we don't necessarily mean it literally. The important part from our perspective is that installing an agent and giving it super user permissions is possible. And it doesn't matter if it resides on AWS EC2, GCP Compute engine, Azure virtual machines or any other cloud provider IaaS solution. They all keep abiding to the same basic rules which allows the server owner full control over it.

DBaaS and containerization pose new challenges to the agent as we know it today, while the "on-prem" deployment can be solved without significant technology modifications.

Monitoring DBaaS with a server-side agent is impossible. If customers want to move to a managed database service, they need to understand that they must rely on a database native audit trail which is enough for many use cases. (By the way, this explains why AWS

released custom RDS which can be considered on-prem for that matter. If it allows you access to the underlying EC2 then it is an EC2). As humans typically access databases from jump servers, it is possible to install client-side agents on them, making it a hybrid solution - native audit logs for applications and client-side agents for humans.

It is also worth noting that a brand new solution can be created for this use case - a database reverse proxy - it works the same way most WAFs work. The ironic part is that the customer is using a managed database because he wanted to ease his pain and such a solution would require him to manage a new service. Simply put, It moves the pain of managing a database into managing a scalable cluster of reverse-proxies per database instance.

Containers are challenging. Containers allow a user to run his application (or database for that matter) within an off-the-shelf virtualized operating system. Since containers are launched and terminated quite often, it is impractical to install an agent for every instance.

You may think embedding an agent into the actual database image would be a realistic option. Unfortunately, your customer will hate you for interfering with his CI/CD process. This leads to 2 different approaches:

1. Ignoring the containerization paradigm by forcing an agent upon the database containers
2. Leveraging eBpf with UProbes to allow the agent to harness the power of in-memory agents while maintaining a containerized approach and keeping the database container sterilized. It is worth mentioning that eBpf is gaining a lot of popularity in containerization environments these days

Summary

In this paper, I described the evolution of the agent's technology to the point when cloud databases emerged. There is no denying that cloud deployments present a significant challenge to agent-based DAM solutions. More and more solutions that try to solve this problem emerge, and in order to achieve a proper solution, both vendors and customers will need to accept the fact that the cloud denies full control of what happens to the database.

In the future, all data security solutions will need to embrace a hybrid approach to data collection. They will need to understand they can't come up with a single method to create the desired audit trail they must have.