MongoDB: Capabilities for Use in a Zero Trust Environment

MongoDB

What Is Zero Trust?

It wasn't that long ago that security professionals protected their IT in much the same way that medieval guards protected a walled city: They made it as difficult as possible to get inside. But once someone was past the perimeter, they had generous access to the riches contained within. That mindset has changed as users increasingly access networks and applications from any geography, on any device, on platforms hosted in the cloud. Reliance on perimeter security has become increasingly insufficient.

Zero trust presents a new paradigm for cybersecurity. In a zero trust environment, the perimeter is assumed to have been breached. There are no trusted users, and no user nor device gains trust simply because of its physical or network location. Every user, device, and connection must be continually verified and audited. The creator of zero trust, security expert John Kindervag, summed up the approach: "Never trust, always verify."

Because databases contain so much of an organization's sensitive (and regulated) information, as well as data that may not be sensitive but is critical to keeping the organization running, it's imperative that your database is ready and able to work in a zero trust environment.

That means the database must be secure by default, and it needs to limit users' opportunities to make it less secure. It must support a wide range of tools to allow users to authenticate themselves to the database. Fine-grained, role-based access controls must govern what a user is allowed to do – or not do – once they've been authenticated. And the database should have auditing capabilities to ensure that administrators can track suspicious or unexpected behavior by event, user, or role.

Additional technologies, notably encryption, are necessary to support the successful implementation of zero trust. The National Institute of Standards and Technology (NIST) wrote a 2020 paper on Zero Trust Architecture, which was cited in President Joe Biden's executive order on cybersecurity; it specifically mentions data protections as a component of zero trust architecture.

This paper will demonstrate MongoDB's ability to be integrated into a zero trust environment. We'll also explain some of the security measures that protect our cloud-based database, Atlas, and the rules governing employee access. And for our government customers, MongoDB Atlas for Government is FedRAMP Ready.

There are several crucial aspects of performance in a zero trust environment: security by default; limited and controlled connectivity to the internet; robust authentication for all users; similar control over user authorizations; and encryption and redaction capabilities that can strictly limit access to certain data.

MongoDB and Security by Default

A database that is secure by default, such as MongoDB, will be secure out of the box. This takes some of the responsibility for security out of the hands of users. The highest levels of security are in place from the start, without requiring attention – or even knowledge – from users or administrators. To allow access, users and administrators must proactively make changes. No one is automatically granted access because of rank or role.

In the case of MongoDB, secure by default means, in part, that Atlas clusters do not allow for any connectivity to the internet when they're first spun up. Each dedicated Atlas cluster is deployed in a unique virtual private cloud (VPC) configured to prohibit inbound access. (Free and shared Atlas clusters do not support VPCs.) The only way to access these clusters is through the Atlas interface. Users can configure IP access lists to allow certain addresses to attempt to authenticate to the database. Without being included on such a list, application servers are unable to access the database. Even the person who sets up the clusters needs to add their IP address to the access list.

Atlas also allows customers to set up temporary access lists with predetermined expiration dates. This can be helpful when a team member needs access from a temporary work location.

Data centers and physical storage

MongoDB Atlas is built to work equally well with any of the three largest cloud providers: Amazon Web Services (AWS), Google Cloud, and Microsoft Azure. No matter which of these platforms a customer chooses, data stored in clusters of level M10 and above is stored in single-tenant dedicated virtual servers created solely for that Atlas customer. These virtual servers are isolated in their own dedicated virtual private cloud and do not share logical data storage or processing with other customers.

Zero Trust Authentication With MongoDB

An IT organization may use any number of methods to allow users to authenticate themselves to a database, including a username and password. MongoDB enables a variety of other ways to authenticate a user into a system. MongoDB supports LDAP proxy authentication as well as Kerberos authentication.

All forms of MongoDB support transport layer security (TLS) and SCRAM authentication. They are turned on by default and cannot be disabled. Traffic from clients to Atlas is authenticated and encrypted in transit, and traffic between a customer's internally managed MongoDB nodes is also authenticated and encrypted in transit using TLS.

MongoDB uses TLS 1.2 by default. Customers can use TLS 1.1 or 1.0 if needed, but MongoDB 4.0 and later does not support TLS 1.0 if a more recent version is available. Starting with MongoDB 4.4, TLS 1.3 is supported when used with a compatible OpenSSL library; TLS 1.3 support in Atlas is coming soon. The MongoDB security team monitors the status of transport protocols and continually updates requirements to ensure that weak ciphers are deprecated.

For passwordless authentication to MongoDB, MongoDB offers two different options to support the use of X.509 certificates. The first option, called "easy," auto-generates the certificates needed to authenticate database users. The "advanced" option is for organizations already using X.509 certificates, and which already have a certificate management infrastructure. These organizations can upload their CA certificate to MongoDB Atlas and continue to use their in-house X.509 certificates for authentication. The advanced option can be combined with LDAPS for authorization. Atlas will send out automated alerts when certificates issued by the Atlas certificate authority, or appearing on the certificate revocation list, are close to expiration.

For dedicated clusters (M10 and above), Atlas provides an easy-to-read log of database authentication events, including both successes and failures. These logs include the database user who attempted to authenticate, source IP address, and time stamp. The logs are available either within the Atlas user interface or via an API.

Access infrastructure can only be reached via bastion hosts and by users whom senior management has approved for backend access. These hosts require multifactor authentication and are configured to require SSH keys, not passwords.

Zero Trust Authorization with MongoDB

Authentication only verifies that someone is who they say they are. Authorization determines what a user is allowed to do. In a zero trust environment, each user has a specific level of allowed access or activity.

To determine each user's privileges, MongoDB uses role-based access controls (RBAC). A user's defined role allows certain activities and denies others. It may allow someone to read data from a database, for example, but not insert data.

MongoDB offers several predefined roles covering the most commonly requested privileges and restrictions. It also allows you to define your own roles, constructing fine-grained access controls tailored to your organization, project, or database. These can reflect a specific functionality that a user needs in their job duties, or a customized role tied to your organizational structure. It also makes it possible to create a separation of duties among different parties accessing and managing data.

Database administrators can also create temporary MongoDB users, which Atlas will automatically delete after a specified period of time. This capability complements granular database auditing (described below). If a user needs temporary access to perform maintenance, for example, the assigned role and its actions can be comprehensively audited. Once the user is deleted, any client or application attempting to authenticate with that user will not have access to the database.

MongoDB also allows the construction of a multilevel security framework, such as "confidential," "secret," and "top secret," by using a $redact operator in conjunction with tags. Objects and users can be grouped into different security levels, and unauthorized users are prevented from accessing information at a higher security level than their authorization.

Database Access by MongoDB Employees

MongoDB uses a combination of technical and logical controls to limit and audit employees who have access to systems with sensitive data. Under normal circumstances, MongoDB engineers are not able to access customer data or the systems that store and process them. In certain "break glass" reliability situations, metadata or logs may be accessed by appropriate personnel to investigate and restore critical services.

Customers can also apply a control setting that will disable the ability of MongoDB support and technical services engineers to SSH into the backend of a customer's infrastructure. If, in trying to assist a customer, an Atlas support team member believes SSH access would be helpful, they will ask the customer to temporarily enable another setting, granting SSH access to a particular cluster.

Only MongoDB employees with preapproved operational roles can be granted access to MongoDB Atlas's underlying systems. These permissions are audited on a regular basis. Access to underlying hosts requires multifactor authentication and a bastion host.

Auditing to Support Zero Trust

MongoDB supports a wide variety of auditing strategies, making it easier to monitor your zero trust environment to ensure that it remains in force and encompasses your database. Administrators can configure MongoDB to log all actions or apply filters to capture only specific events, users, or roles.

MongoDB Enterprise Advanced's role-based auditing allows you to log and report activities by specific role, such as userAdmin or dbAdmin,

coupled with any roles inherited by each user, rather than having to extract activity for each individual administrator. This makes it easier for organizations to enforce end-to-end operational control and maintain the insight necessary for compliance and reporting.

The audit log can be written to multiple destinations in a variety of formats, such as to the console and syslog (in JSON) and to a file (JSON or BSON). It can then be loaded to MongoDB and analyzed to identify relevant events.

MongoDB and Encryption in a Zero Trust Environment

Encryption is a key technology in a zero trust environment. In essence, it's the last line of defense. If someone manages to convince the system that they are someone they're not, with authorizations they shouldn't have, encryption must still protect your data.

MongoDB allows you to encrypt your data in flight, at rest, or even, with field-level encryption, in use. For data in motion, all versions of MongoDB support TLS and SSL encryption. For data at rest, MongoDB supports AES-256 in both CBC mode and GCM mode. It can also be configured for FIPS compliance.

To encrypt data when it is in use, MongoDB offers an industry-leading capability called client-side field-level encryption. Client-side field-level encryption can be implemented to safeguard data even from database administrators and vendors who otherwise would have access to it.

Client-side field-level encryption is different from other database encryption approaches because the process of encrypting and decrypting is completely separate from the database server. Encryption and decryption are handled exclusively within the MongoDB drivers in the client, before sensitive data leaves the application. The database only ever uses it as ciphertext.

Client-side field-level encryption is highly flexible. You can selectively encrypt individual fields within a document, multiple fields within the document, or the entire document. Each field can be optionally secured with its own key and decrypted seamlessly on the client.

Securing data with client-side field-level encryption allows you to move to managed services in the cloud with greater confidence. The database only works with encrypted fields, and organizations control their own encryption keys, rather than having the database provider manage them. This additional layer of security enforces an even finer-grained separation of duties between those who use the database and those who administer and manage it.

Client-side field-level encryption also makes it easier to comply with so-called right to be forgotten mandates in modern privacy legislation such as GDPR and CCPA. If a user invokes their right to be forgotten, destruction of the associated field encryption key will render the user's personally identifiable information unreadable and irrecoverable to anyone.

Because the database server has no access to the encryption keys, certain query operations such as sorts, regexes, and range-based queries on encrypted fields are not possible server-side. Because of this, client-side field-level encryption is best applied to selectively protect just those fields containing highly sensitive personal information such as email addresses, phone numbers, credit card information, or social security numbers. (MongoDB's Queryable Encryption, introduced in June 2022, is shifting the encryption landscape. It allows you to encrypt sensitive data from the client side, store it as fully randomized encrypted data on the server side, and run expressive queries on the encrypted data.)

Conclusion

MongoDB is optimally suited for use within a zero trust environment. MongoDB is secure by default and has developed industry-leading capabilities in important areas such as access, authorization, and encryption. Used together, these features help protect the database from outside attackers and internal users who otherwise could gain an unauthorized level of access.

For our cloud-based database-as-a-service, Atlas, MongoDB has taken additional steps to ensure the database cannot be accessed through the underlying cloud-based infrastructure. These features directly address the most critical requirements for a modern database to function as an integrated component of a zero trust environment. For a more complete discussion of MongoDB's security capabilities, please consult our security documentation.

Contact MongoDB to learn more about using MongoDB in a zero trust environment.