How YADA Keeps Your Data Secure

Whether your organization depends on data for R&D, sales and marketing, operations management, or a host of other business needs, three scenarios are likely: you are regularly adding new data sources; your teams are frequently adopting new tools for analyzing and visualizing that data; and you are continually fielding requests for access from new users.

Each scenario requires developers to secure, and re-secure, sensitive data. Just as YADA eliminates time-consuming and repetitive work when providing data access, the YADA Security API enables developers to enforce and expand the data protection policies that your organization has in place. Developers and YADA administrators can build access rules once into every stored query in YADA, keeping data secure regardless of where it is stored or where the request for it originates.

Why YADA Security?

The YADA Security API addresses data protection requirements that arise when access to data becomes more widely distributed.

Most organizations want to restrict access to data in a variety of ways. Some data may be accessed only by employees, or may be limited further to people working on specific teams or projects. Other data may be governed by regulations or corporate policies that define how it may be used, or for what purposes.

Typically, these rules are configured at the data source. Only people or applications with credentials are authorized to access the data. When data sources are configured to accept YADA queries, YADA is provided with access credentials as well. Because YADA, just like any other intermediary, sits between the user and the data, in theory, anyone with a URL to a query stored in YADA could access the underlying data, whether or not they have been assigned credentials that allow them to have it.

The YADA Security API solves this problem by enabling developers to apply the necessary security parameters to any YADA query. The parameters are stored with the query and applied every time the query is executed.

Tying security to each query makes YADA technology-independent. YADA queries can be written to comply with whatever security policies are in place for each data source, whether at the database, server, web service or user level. This approach can enhance existing security practices. Users accessing data through a new (to the data source) application can do so even if security has not been configured for the application, because the data-protection configuration

is stored with the query. Likewise, role-based access can be maintained across applications and data sources.

The YADA Security Approach

YADA delivers data security using a four-tiered model that encompasses URL pattern matching, token authentication, execution policies and content policies. (See Figure 1: YADA Security API Flowchart, below). Each tier can be defined using a default parameter stored alongside a protected query that can be configured to prevent any user from overriding the security settings. YADA applies the security policies described in the default parameter when it receives a request for data access, before executing the query.

For definitions and details about the features of the YADA Security API, see the YADA Security Guide. To learn how to implement security using the YADA Security Wizard, see the Admin Guide.

The Four Tiers of YADA Security

Tier 1: Is the Request Targeting An Authorized Domain?
Developers and YADA administrators can restrict YADA queries to be executed only when the request is targeting a URL in an authorized domain or set of domains. If, for example, a company uses a security gateway to control access to the corporate network, YADA can be configured to reject any query that does not refer to an address inside the secured network domain.

Tier 2: Is the User Verified?
Some organizations use a token authentication service to verify users' identities with something other than a password, such as cookies, encrypted keys or biometric identifiers. (Facebook Connect is an example of a token authentication service: when someone logs into a site using their Facebook credentials, the site compares them with information held by Facebook to confirm that the submitted credentials match those of the same person on Facebook).

YADA can be configured to implement whatever token authentication method an organization uses so that any query from an unverified user is rejected.

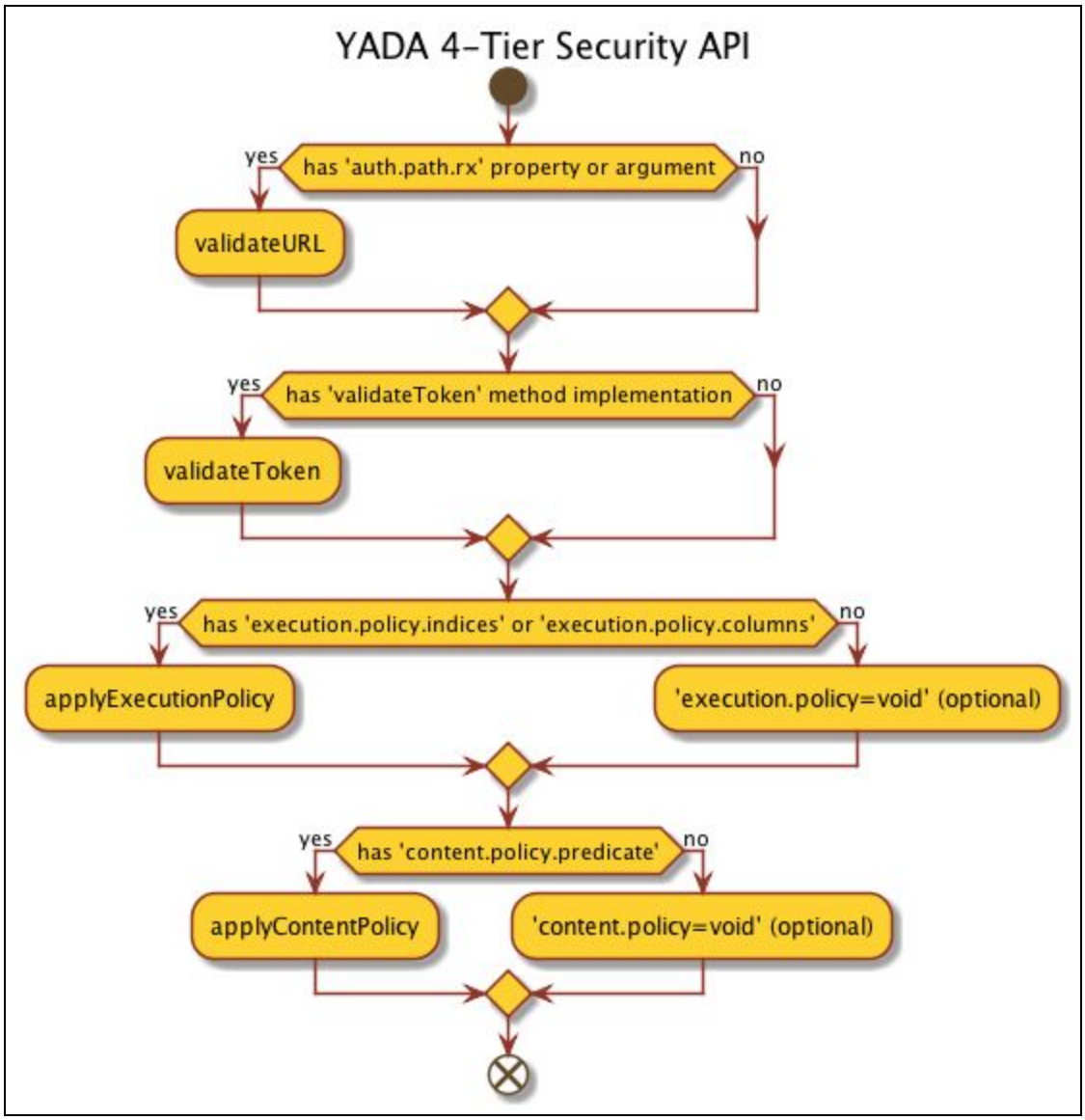
Tier 3: Is the User Allowed to Execute the Query?
A user may be verified and logged in to the appropriate domain, but that doesn't mean they are allowed to ask for data or make changes to it. Developers and YADA administrators can map existing whitelist or blacklist policies to any query to define which individuals or teams may—or may not—execute the query.

Tier 4: Is the User Allowed to View the Data?

In many cases, a user may be allowed to execute a query because they are authorized to view some of the data, but not all of it. As a simple example, a salesperson could be allowed to query a database that contains customer account records but may be restricted to viewing only their own accounts or those of people on their team. The regional head of sales, on the other hand, could be allowed to view the account records for every salesperson in the region, but may not be allowed to view customers' contact information.

YADA enables developers and administrators to apply content policies at the query level that will filter the results based on role, identity or any other criteria. YADA achieves this by amending a query *before* execution, rather than filtering results afterwards as is often the only option. This feature can be enabled for any JDBC data source.

Figure 1: YADA Security API Flowchart



YADA Enhances Security and Saves Work

Many companies build data warehouses with elaborate security layers to protect data regardless of where a request comes from, or who makes it. This common practice works, but it has limits:

- New users must have their credentials mapped to the data warehouse before they can access the data, which creates a bottleneck in gaining data access
- Even worse, users may share their own credentials, or generic credentials, amongst themselves.
- Any new application, such as a visualization tool, must be authorized to call data from the data warehouse.
- Though there are exceptions, most databases do not currently support content filtering policies as a default query parameter. Instead, content policies must be hard-coded into each implementation and rewritten every time a query is repurposed for a new set of users.
- When bringing new data sources online, security has to be configured at the source before it can be made available to users.
- Not every data source ends up in a protected data warehouse. Security may not be properly configured, creating vulnerabilities.

All these limitations of traditional data warehouse security add up to work—and rework. With the YADA Security API, developers and administrators can code security rules once, using the YADA Security Wizard, and be sure that data is properly protected no matter who is requesting it. They won't have to configure security continually for new groups of users or new access points. It's easy to update security rules too, when necessary, by updating the default parameter, without having to reimplement the associated query.

Like the rest of the YADA framework, the YADA Security API is designed to save developer and administrator time, speed data access, and help data users keep working.

Have questions about YADA or YADA security? Read the documentation or email info@yadadata.com