Use Custom Events to hide sensitive function arguments

Issue

Some functions in PHP take arguments that contain sensitive information that should not be seen by programmers looking into script errors. However, the Zend Server Event for a function error will show all arguments in the clear. Values for PHP super-globals (POST, GET, SESSION etc.) can be concealed by a Blacklist, but function arguments cannot. The values will be shown in the Function Data tab in the event detail.

This article shows an example of db2_connect() showing a password in the Function Data. In this example, we remove db2_connect() from the Database Error rule, and instead use a Custom Event to show the error while hiding the password.

Environment

Zend Server for IBM i on any supported version of IBM i OS. This example uses Zend Server 8.5, but the technique is generally applicable to any version.

> **Note: Custom Events are a premium feature.**
>
> Custom Events are not available in the Basic version of Zend Server for IBM i.

Resolution

If the db2_connect() function fails, the default Database Error rule will produce an Event. On the Function Data tab of the Event, there will be some information displayed similar to this:

PHP Error [select all]

Function information [select all]

db2_connect (
'*LOCAL',
'PHPUSER',
'BLACKHAWK'
)
)

We can clearly see the password 'BLACKHAWK' for user 'PHPUSER'. This is what we want to avoid.

In this example, we take the db2_connect() function out of the Database Error rule, and instead set up a custom rule in our script to capture just the arguments we want to see visible.

To remove the db2_connect() function from the Database Error rule, go into the Zend Server UI and navigate to Monitoring -> Event Rules. Click on the Global Rules category to expand the list of rules. Find Database Error on the list, and click it to see the details for the rule. On the detail screen, you will see a list of functions to the right. Scroll through the list to find db2_connect, and click the x next to it to remove it. Then, click the Save button on the lower left to save your change. You will then need to perform a restart, which will take your website offline for a couple of minutes, so please do this activity when it will be OK to have some down time.

Next, we add a custom event to our script. This needs to be done for each script that makes a DB2 connection. We use the zend_monitor_custom_event() function to create the custom event, which will generate an Event that can be seen in the Events table in the UI. We can include an array in the custom event that will have any variables and other values we choose, and that array will be displayed as part of the event.

```
$conn = db2_connect($database, $user, $password);
if (!$conn) {
     echo 'connection failed';
     // Connection failed. Set up the Event Data and create the Custom Event.
     $event_data = array('Database'=>$database, 'User'=>$user, 'Password' => '********',
                'SQLSTATE value' => db2_conn_error(), 'SQL Error Message' => db2_conn_errormsg());
     zend_monitor_custom_event('Connection Error', 'DB2 Connection Failed', $event_data);
 }
```

When you view the Custom Event in the Events, you will see something like this under the Custom Variables tab on the details page:

User Data [select all]

- array (
  'Database' => '*LOCAL',
  'User' => 'PHPUSER',
  'Password' => '********',

```
  'SQLSTATE value' => '08001',
  'SQL Error Message' => 'Authorization failure on distributed database connection attempt. SQLCODE=-30082',
)
```

We just hard coded the asterisks into the Password array element, which just lets the person viewing the page know this is protected data. We also grabbed some error information from the connection error functions, which can be helpful to the programmer.

Please be aware that the Event will also show the Backtrace, which is the code for the script, positioned to the line that generated the event. If the password is hard coded in the script, it will be plainly visible in the Backtrace. Then again, if it is hard coded in the script, it will be plainly visible to the programmer when they review the script. Typically the user and password are collected in a Post variable and stored in a Session variable. These super-global values can be Blacklisted to prevent their display in the Event.

Note: Learn how to Blacklist super-globals here:

Monitor Security Blacklist