Addressing Metadata in Salesforce Security Posture Management

Metadata security is an important factor in maintaining reliable functionality in your Salesforce environment. A comprehensive approach is essential to preserving the integrity of this important pool of data.

Here are 8 steps to addressing metadata in Salesforce security posture management:

1 Identify Your Types of Metadata

Understanding the differences between these types of metadata will help you put together a better plan for metadata security.

2 Perform a Risk Assessment

A risk assessment improves visibility into how your Salesforce metadata is used, while also highlighting which sets of data are more sensitive and need to be protected.

3 Analyze Permissions Settings

Team members should only be able to access the data they need to perform their daily duties. Overexposed metadata is much more likely to experience accidental deletions or costly corruptions.

4 Secure Access Points

Secure passwords and multi-factor authentication are necessities for every member of your organization with access to your Salesforce environment.

5 Utilize Version Control

Version control enables teams to monitor the accuracy of each change, which allows them to identify potential issues with their metadata and take the security precautions necessary to protect it.

6 Set a Schedule to Review Security Policies

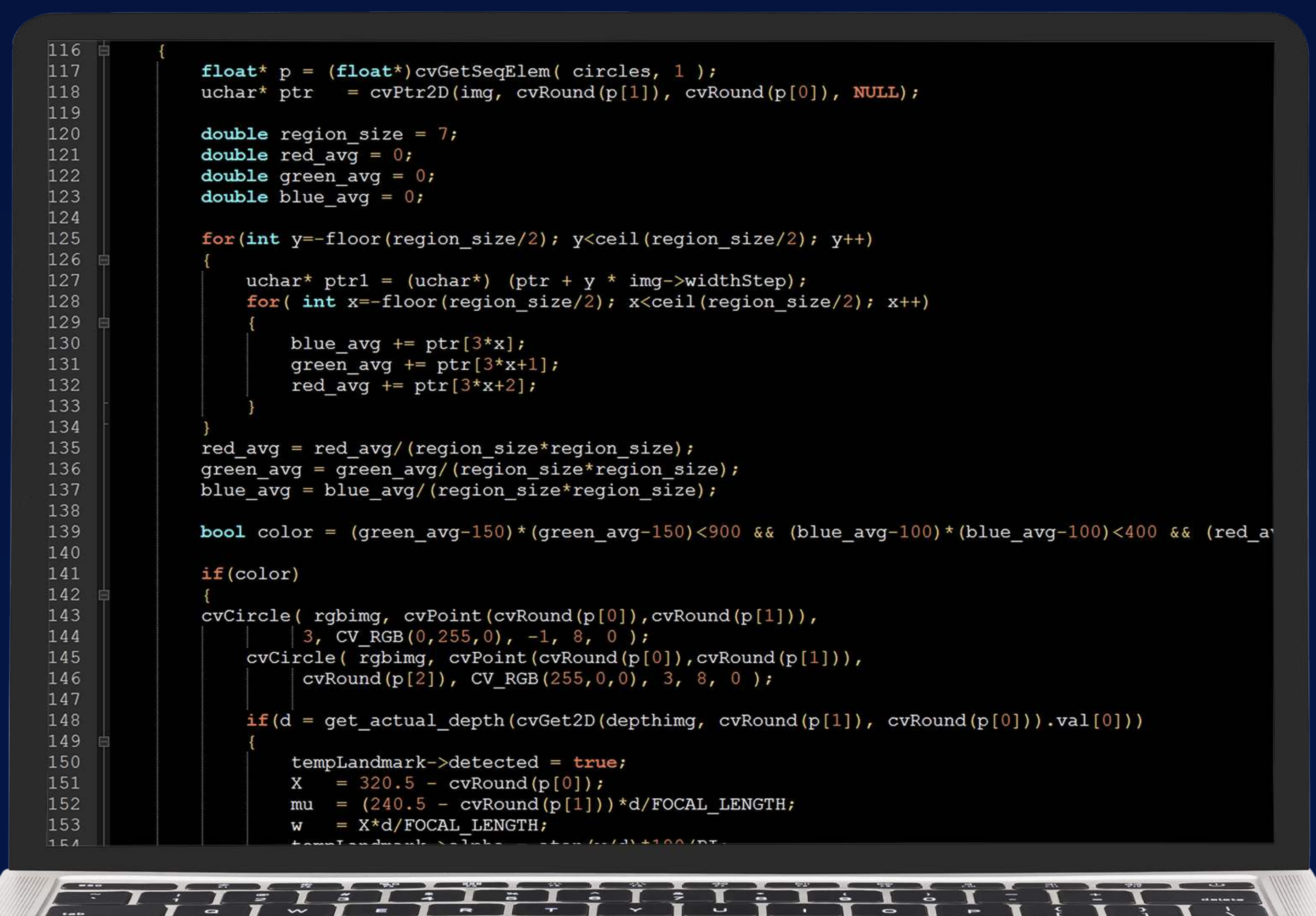
Metadata security is an essential aspect of Salesforce security posture management and needs to be continually addressed.

7 Provide Security Training to Team Members

A repeated cycle of cybersecurity training keeps best practices fresh in the minds of your team members and works to maintain the integrity of your Salesforce data and metadata.

8 Backup Everything

Frequent backups of a Salesforce environment will provide the coverage necessary to get your system back online quickly and efficiently to minimize downtime.



```
    {
        float* p = (float*)cvGetSeqElem( circles, 1 );
        uchar* ptr   = cvPtr2D(img, cvRound(p[1]), cvRound(p[0]), NULL);

        double region_size = 7;
        double red_avg = 0;
        double green_avg = 0;
        double blue_avg = 0;

        for(int y=-floor(region_size/2); y<ceil(region_size/2); y++)
        {
            uchar* ptr1 = (uchar*) (ptr + y * img->widthStep);
            for( int x=-floor(region_size/2); x<ceil(region_size/2); x++)
            {
                blue_avg += ptr[3*x];
                green_avg += ptr[3*x+1];
                red_avg += ptr[3*x+2];
            }
        }
        red_avg = red_avg/(region_size*region_size);
        green_avg = green_avg/(region_size*region_size);
        blue_avg = blue_avg/(region_size*region_size);

        bool color = (green_avg-150)*(green_avg-150)<900 && (blue_avg-100)*(blue_avg-100)<400 && (red_a

        if(color)
        {
        cvCircle( rgbimg, cvPoint(cvRound(p[0]),cvRound(p[1])),
                    3, CV_RGB(0,255,0), -1, 8, 0 );
            cvCircle( rgbimg, cvPoint(cvRound(p[0]),cvRound(p[1])),
                    cvRound(p[2]), CV_RGB(255,0,0), 3, 8, 0 );

            if(d = get_actual_depth(cvGet2D(depthimg, cvRound(p[1]), cvRound(p[0])).val[0]))
            {
                tempLandmark->detected = true;
                X    = 320.5 - cvRound(p[0]);
                mu   = (240.5 - cvRound(p[1]))*d/FOCAL_LENGTH;
                w    = X*d/FOCAL_LENGTH;
```