Security is Everyone's Responsibility

It's your job to make sure leadership considers security to be important and foster policies to protect your church data.
It's your leadership's job to enforce those policies with your staff.
It's your staff's responsibility to do their work with diligence and report anything suspicious or strange.

Organizational Roles

Only 1 or 2 people at your organization should have the RSR - Rock Administration role in Rock.
    Separate duties and create your own more granular roles where needed to empower your team.
Everyone is responsible for data integrity, but only a few people should be in charge of it.
    Use formal processes for your ministry staff to report errors in a person's data (i.e. this person moved) but rather than letting all staff be able to edit any person record, have them fill out a person data error form and let your technical team vet the information and be the ones to make updates or merge records.
    This will ensure that data is consistent and more accurate.
Principal of Least Privilege
    If someone doesn't absolutely need to be able to do something in order to do their job, don't give them the ability to do it.
Data Cleaning should happen every week
    Stay on top of merging duplicate records and fixing data issues.

Policies

You should have policies dictating how staff, volunteers, and group leaders are allowed to use and share church data.
    Security via Obscurity is not security at all. Sharing anything with a "anyone with the link" can view permission is putting that data out there for anyone to see and exploit.
You should have ways to monitor that those policies are being followed so you can enforce them.

Merge Block Lava App

Designed to display login and security note information on the merge block to help prevent account hijacking from an accidental merge.

Lava App Settings

Name: Merge Security Info
Slug: merge-security-info
Security: RSR Admin, Data Integrity (or whoever does your merges), and Lava App Admin Only people who can use/view

Endpoint Settings

User Login

Slug: merge-security-logins
Security Mode: Application View
Method: GET
Cache: No-Store
Enabled Commands: SQL
Code Template:

HTML

```
{% assign pid = QueryString.PersonId %}
{% sql id:'{{pid}}' %}
    SELECT UserName FROM UserLogin WHERE PersonId = @id;
{% endsql %}

{{ results | Select:'UserName' | Join:', ' }}
```

User Security Notes

Slug: merge-security-notes
Security Mode: Application View
Method: GET
Cache: No-Store
Enabled Commands: Rock Entity
Code Template:

HTML

```
{% assign pid = QueryString.PersonId %}
{% note where:'EntityId == {{pid}} && IsAlert == true' %}
    {% for note in noteItems %}
        <div class="alert alert-danger">
            <div>
                {{ note.Text }}
            </div>
            <hr/>
            <div>
                {{ note.CreatedByPersonName }} {{ note.CreatedDateTime | Date:'MM/dd/yyyy' }}
            </div>
        </div>
    {% endfor %}
{% endnote %}
```

Lava Application Content Block

Add block below Merge Block, select the lava application

HTML

```
<script>
    let existingIds = null
    function checkForNewRecords() {
        let mergeProspects = []
        document.querySelectorAll('[data-person-id].merge-header-summary').forEach(el => {
            if(mergeProspects.indexOf(el.dataset.personId) < 0) {
                mergeProspects.push(el.dataset.personId)
            }
        })
        console.log(mergeProspects)
        if(JSON.stringify(mergeProspects) != existingIds) {
            console.log('Update Security Info')
            existingIds = JSON.stringify(mergeProspects)
            displaySecurityInfo(mergeProspects)
        }
    }
    function displaySecurityInfo(ids) {
        $('.merge-block .panel-body .grid tfoot').children().remove()
        let loginRow = '<tr><td class="grid-row-header">Logins</td>'
        let noteRow = '<tr><td class="grid-row-header">Security Notes</td>'
        ids.forEach(id => {
            loginRow += '<td class="js-merge-field-cell merge-field-cell"><div id="htxUserLogin_'
+ id + '" hx-get="^/merge-security-info/merge-security-logins?PersonId=' + id + '"
hx-trigger="load"></div></td>'
            noteRow += '<td class="js-merge-field-cell merge-field-cell"><div id="htxUserNotes_' +
id + '" hx-get="^/merge-security-info/merge-security-notes?PersonId=' + id + '"
hx-trigger="load"></div></td>'
        })
        loginRow += '</tr>'
        noteRow += '</tr>'
        $('.merge-block .panel-body .grid tfoot').append(loginRow)
        $('.merge-block .panel-body .grid tfoot').append(noteRow)
        ids.forEach(id => {
            htmx.process(document.getElementById('htxUserLogin_' + id))
            htmx.process(document.getElementById('htxUserNotes_' + id))
        })
    }
    setInterval(() => checkForNewRecords(), 5000);
</script>

<style>
    .grid-table > tfoot > tr > td.grid-row-header {
        font-weight: var(--font-weight-bold);
    }
    .grid-table > tfoot > tr > td.merge-field-cell {
        padding: var(--spacing-medium) var(--spacing-medium) var(--spacing-medium) 12px;
    }
</style>
```

Reports for Monitoring Staff Accounts

Things we have data views for to watch that are regularly reviewed for irregularities:
Recent memberships to security roles
People with High/Extreme Protection Profiles
Family Members of Staff (or just people with Extreme Profile Protection Levels)
Lock all these reports to only be visible to specific staff accounts like RSR Data Integrity and RSR Admin

Lava for Displaying Login Info

HTML

```
<div>
    {% userlogin where:'PersonId == {{Id}}' securityenabled:'false' %}
        {% for l in userloginItems %}
            {% entitytype id:'{{l.EntityTypeId}}' securityenabled:'false' %}
                {% assign textColor = "color: green;" %}
                {% if l.IsConfirmed == false or l.IsLockedOut == true %}
                    {% assign textColor = "color: red;" %}
                {% elseif entitytype.FriendlyName != 'Database With 2FA' %} //- Replace with Your
Provider
                    {% assign textColor = "color: orange;" %}
                {% endif %}
                <div style="{{textColor}}">
                    {% if l.IsLockedOut == true %}
                        <i class="fa fa-lock mr-2"></i>
                    {% else %}
                        <i class="fa fa-lock-open mr-1"></i>
                    {% endif %}
                    {% if l.IsConfirmed == true %}
                        <i class="fa fa-check-circle mr-1"></i>
                    {% else %}
                        <i class="fa fa-circle mr-1"></i>
                    {% endif %}
                    {{l.UserName}} - {{entitytype.FriendlyName}}
                </div>
            {% endentitytype %}
        {% endfor %}
    {% enduserlogin %}
</div>
```

Lava for Displaying Number of Logins

```HTML
{% person id:'{{Id}}' %}
    {% userlogin where:'PersonId == {{Id}}' securityenabled:'false' %}
        {% assign noAcct = userloginItems | Size %}
    {% enduserlogin %}
    {% assign textColor = "color: orange;" %}
    {% if noAcct == 1 %}
        {% assign textColor = "color: green;" %}
    {% elseif noAcct > 1 %}
        {% assign textColor = "color: red;" %}
    {% endif %}
    <div style="font-weight: bold; {{textColor}}">{{noAcct}}</div>
{% endperson %}
```

Useful Lava for Impersonation Tokens

Wrap the use of impersonation tokens around making sure it is valid for that user and won't result in a TokenProhibted and logout action

```PHP
{% if Person.AccountProtectionProfile != 'Extreme' %}{% endif %}
```

Useful SQL

Convert Comma Separate List to Individual Rows

Example Context:
Page with a page parameter filter to filter a list of sermon content channel items down by tags and sermon title displayed by a dynamic data block that has Tags in the query parameters field

```SQL
SELECT Tag.Name AS Tag, ContentChannelItem.Id, ContentChannelItem.Title
FROM Tag
    INNER JOIN (
        SELECT value AS Tag
        FROM STRING_SPLIT(@Tags, ',')
    ) AS SearchTags ON Tag = Tag.Name
    INNER JOIN TaggedItem ON Tag.Id = TaggedItem.TagId
    INNER JOIN ContentChannelItem ON EntityGuid = ContentChannelItem.Guid
WHERE ContentChannelTypeId = 21
  AND Status = 1
```

Use `STRING_SPLIT` to convert "Community,Serve,Volunteer" to rows that can be joined on our Tag table to filter the results.
Could also store the results of the STRING_SPLIT in a Temporary Table or Table Variable if needed to optimize performance of the SQL.

Conditional SQL

Wrap WHERE statements in parentheses for conditional filtering similar to building conditional LINQ statements in C#
Example Context:
Querying Group Scheduling Attendance Records
Page parameter “LimitToScheduled” that contains the value we want to filter the “ScheduledToAttend” field on
i.e. the page parameter block has a filter like “Limit to People Scheduled to Attend” that can be Blank, True, or False saved to “LimitToScheduled”
Then the query is on attendance records for group scheduling and we either include all results or filter to results where the “ScheduledToAttend” (i.e. ScheduledToAttend) is true

```SQL
WHERE
(
    (@LimitToScheduled IS NULL OR @LimitToScheduled = '' OR @LimitToScheduled = 'False')
    OR
    (@LimitToScheduled = 'True' AND ScheduledToAttend = 1)
)
```

Use SQL IF blocks to change what SQL is executed conditionally
Example Context:
Page parameter “StartDate” that is a date but could also be a “Current Date” +/- X Minutes

```SQL
    DECLARE @StartDateKey INT = NULL;
    IF @StartDate IS NOT NULL AND @StartDate LIKE 'CURRENT%'
        BEGIN
            DECLARE @StartModifier INT = SUBSTRING(@StartDate, 9, LEN(@StartDate) - 8);
            SET @StartDateKey = FORMAT(DATEADD(DAY, @StartModifier, @currentDate), 'yyyyMMdd');
        END
    ELSE
        IF @StartDate IS NOT NULL
            BEGIN
                SET @StartDateKey = FORMAT(CAST(@StartDate AS DATE), 'yyyyMMdd');
            END
```

Sanitizing SQL

Casting

Rock Documentation for type casting (https://community.rockrms.com/lava/filters/other-filters#asboolean)

```
HTML
{% assign userInput = 'Global' | PageParameter:'Search' %}

//- true/false
{{ userInput | AsBoolean }}
//- '0', 'false' > false '1','true' > true

//- integer (whole number)
{{ userInput | AsInteger }}
//- '1', '1.5' > 1

//- decimal (number with decimal places)
{{ userInput | AsDecimal }}
//- '5.00000000000000009' > 5.00000000000000009

//- double (number with decimal places, but it is less precisce than a decimal)
{{ userInput | AsDouble }}
//- '5.00000000000000009' > 5

//- guid (a database-wide unique value for a row)
{{ userInput | AsGuid }}
//- '628C51A8-4613-43ED-A18D-4A6FB999273E' > 628c51a8-4613-43ed-a18d-4a6fb999273e
//- '628C51A8-4613-43ED-A18D-4A6FB999273E0000' > null (Invalid guid becomes null)
```

SanitizeSql Filter

```
HTML
{% assign userInput = 'Global' | PageParameter:'Search' %}

//- escapes single quotes for SQL
{{ userInput | SanitizeSql }}
//- "'; SELECT * FROM Person; SELECT'" > "''; SELECT * FROM Person; SELECT ''"
```