Securing Your App: A Hacker's Perspective

Jan de Vries / Community Engagement Manager / Mendix
Dirk van Veen / Security Consultant / The S-Unit
$1.5 trillion
Security in a Mendix App

- Cross-site scripting
- SQL injection
- Code execution
Security in Mendix

Roles and access controls
“If you want full security, you need to explicitly give access to forms, entities and microflows before someone can access them.”

Mendix Documentation
How a hacker sees Mendix

UI vs. API
Bypassing the UI
Manually and through JS API
/XAS/ API

Backend JSON endpoint for Mendix apps

Used for all data retrieval and modification

- Basic CRUD operations
- Search or filter functionality
- Calling Microflows
- Authentication
- Get Model metadata

Undocumented
```javascript
return void 0 === t && (t = {}), void 0 === n && (n = {}), Object(a.post)(r, {
    params: {
        xpath: t,
        schema: t,
        count: n
    }
});

function d(e, t, n, l) {
    return Object(a.post)(r, {
        action: "retrieve_by_query",
        params: {
            queryId: e,
            constraint: n,
            context: t,
            options: l
        }
    });
}

function h(e, t, n, l, r) {
    return Object(a.post)(r, {
        action: "retrieve_by_Microflow",
        params: {
            queryId: e,
            params: t,
            paging: n,
            changes: l
        },
        objects: r
    });
}
```
Client API

JavaScript framework for Mendix apps

Abstraction layer on top of /xas/

Support most /xas/ functionalities

• Basic CRUD operations
• Search or filter functionality
• Calling Microflows
• Etc.

Documented!
Home

This is the Mendix client API documentation.
This documentation is primarily meant to assist the development of custom client-side extensions (widgets and third-party integration code, for the most part).

Foundations

The Mendix client uses the Dojo library as a basis, in particular the Dijit library which provides the widget infrastructure.
Basic styling is provided by the Twitter Bootstrap CSS framework.

Structure

The client consists of a number of subsystems and some object classes that implement the concepts used by these subsystems. All subsystems exist inside a single object mix, while the object classes are divided between the mxui and mendix namespaces.

You can load the client with some lines similar to the following in your HTML file:

```html
<link rel="stylesheet" href="path/to/mxui/ui/mxui.css">
<script>
DojoConfig = {
baseP1: "path/to/dojo/"
};
</script>
```
Exercise
Insecure Read Access

- Readable sensitive entities
- Readable sensitive attributes
- Missing XPath constraints on access rules
Go Do It

Go to:
[url]

Do the following:
1. Click on Exercise 1
2. Open developer console
3. Use `mx.data.get` to find the hidden `ModuleRead.Secret` objects.

Time: 3m
Exercise
Insecure Write Access

- Writeable sensitive attributes
- Missing XPath constraints on access rules
### Module Security Type 'Security'

**Note:** Changes you make here modify the domain model.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Module roles</th>
<th>Read</th>
<th>Write</th>
<th>Member access</th>
<th>XPath constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile</td>
<td>Admin</td>
<td>Yes</td>
<td>Yes</td>
<td>Full Read, Full Write</td>
<td></td>
</tr>
<tr>
<td>Company</td>
<td>Admin</td>
<td>Yes</td>
<td>Yes</td>
<td>Full Read, Full Write</td>
<td></td>
</tr>
<tr>
<td>Profile</td>
<td>User</td>
<td>No</td>
<td>No</td>
<td>Full Read, Limited Write</td>
<td>[MendixWorld.Profile.User == %CurrentUser%]</td>
</tr>
<tr>
<td>Profile</td>
<td>User</td>
<td>No</td>
<td>No</td>
<td>Limited Read, No Write</td>
<td>[MendixWorld.Profile.Company/MendixWorld.Company/MendixWorld.Pro...</td>
</tr>
<tr>
<td>Member</td>
<td>Access rights</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>---------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secrets (String (200))</td>
<td>Read, Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bio (String (200))</td>
<td>Read, Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nickname (String (200))</td>
<td>Read, Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsPublic (Boolean)</td>
<td>Read, Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MendixWorld.Profile_Company (MendixWorld.Company)</td>
<td>Read, Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MendixWorld.Profile_User (System.User)</td>
<td>Read</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Go Do It

Go to:
[url]

Do the following:
1. Click on Exercise 2
2. Use `mx.data.get` to retrieve your `ModuleWrite.User` object
3. Use `MxObject.set` and `mx.data.commit` to change your name to `Admin`
4. Find the `ModuleWrite.Secret` objects

Time: 3m
Exercise
Insecure Microflow Access

• Accessible sensitive microflows
• Accessible insecure microflows
Go to:
[url]

Do the following:
1. Click on **Exercise 3**
2. Use `mx.data.get` to find the hidden `ModuleFlow.Secret`
3. Use `mx.data.action` to call microflow `ModuleFlow.ShowSecret` with the hidden object and retrieve its secret

Time: 3m
The S-Unit & Mendix Apps

Mendix Quick Scan
Inventory your publicly exposed Mendix objects and scan them for sensitive information

Mendix Penetration Test
Black/grey box penetration test of your Mendix app without prior knowledge of your Mendix model

Mendix Model Review
Whitebox review of your Mendix model including roles, access controls, flows, Java actions, and more

www.the-s-unit.nl
Upgrade Your Security Knowledge

Mendix Evaluation Guide
Detailed explanation of platform, runtime and cloud security, including our security model.
www.mendix.com/evaluation-guide

Mendix Online Academy
Choose from a range of learning paths to learn everything you need to know about app security.
Learn.mendix.com

Mendix Blog
Read about the latest trends in security, app development and Mendix releases in the Mendix blog.
www.mendix.com/blogs
Questions