Security Analysis of Mattermost

By Changze Cui & Weihao Dong
What is Mattermost
What is Mattermost
What is Mattermost
Key Difference between Mattermost and Slack

<table>
<thead>
<tr>
<th></th>
<th>Mattermost</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Self-Hosting</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Open Source</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Mattermost envisions itself as an open source Slack alternative
Motivations
Phase 1

Read Documentation
Extracted features

Phase 2

Came up with Threat Model

Phase 3

Acquired and deployed a trial of Enterprise Edition of Mattermost

Phase 4

Conducted behavior analysis and source code analysis
Network Capture, Static Analysis, Manual Analysis, POC generation
Architecture of Mattermost

Internet

Proxy

Notification Service

Email Service

Mattermost Server

Database
(MySQL, Postgres)

File Store
(Amazon S3)

HTTPS
Secure Web Sockets

PC Web Experience
Mobile APP Experience
Mobile Web Experience
Email Client

https://docs.mattermost.com/deployment/deployment.html
Features of Mattermost

- Notification Service
- Proxy
- Email Service

Mattermost Server
- Integrations
  - Bot
  - Webhook
  - Plugins
  - Slash Command
  - Send/Reply/Edit/Delete/Mark
  - Interactive Messages
  - Formatting Text
  - Search History
  - Link Preview
  - File Share/Preview
- User Management
- User Team
- Channel Permission
- Policy Compliance
- Data Retention
- Compliance Export
- GDPR/U.S. Export Compliance
- Scalability Support
- Data Management
- RESTful API
- OAuth2.0

Database
- (MySQL, Postgres)
- Password Hashing
- Encryption-at-rest

File Store
- (Amazon S3)

Internet
- HTTPS
- Secure Web Sockets

PC Web Experience
- Mobile APP Experience
- Mobile Web Experience
- Email Client
Project Timeline

Phase 1: Read Documentation
- Extract features

Phase 2: Came up with Threat Model

Phase 3: Acquired and deployed a trial of Enterprise Edition of Mattermost

Phase 4: Conducted behavior analysis and source code analysis
- Network Capture, Static Analysis, Manual Analysis, POC generation
How We Come Up With Our Threat Model

- Internet
- Notification Service
- Proxy
- Email Service

**Mattermost Server**
- Integrations
  - Bot
  - Webhook
  - Plugins
  - Slash Command
  - Send/Reply/Edit/Delete/Mark
  - Interactive Messages
  - Formatting Text
  - Search History
  - Link Preview
  - File Share/Preview

- User Management
  - Users
  - Team
  - Channel
  - Permission

- Policy Compliance
  - Data Retention
  - Compliance Export
  - GDPR/U.S. Export Compliance

- User Authentication
  - Email
  - SAML 2.0
  - Team Invitation
  - Login and password reset page
  - MFA
  - AD/LDAP
  - Guest Account

- Site Management
  - Logs & Statistics
  - Performance Monitoring
  - Rate Limiting

- Scalability Support
  - RESTful API
  - OAuth2.0

**Database** (MySQL, Postgres)
- Password Hashing
- Encryption-at-rest

**File Store** (Amazon S3)

---

https://docs.mattermost.com/deployment/deployment.html
Attackers in Server & Database

Internet

Notification Service

Proxy

Email Service

Mattermost Server

Integrations
- Bot
- Webhook
- Plugins
- Slash Command
- Send/Reply/Edit/Delete/Mark
- Interactive Messages

Messaging
- Formatting Text
- Search History
- Link Preview
- File Share/Preview

User Authentication
- Email
- SAML 2.0
- Team Invitation
- Login and password reset page
- MFA
- AD/LDAP
- Guest Account

Site Management
- Logs & Statistics
- Performance Monitoring
- Rate Limiting

User Management
- Users
- Team
- Channel
- Permission

User Management

Policy Compliance
- Data Retention
- Compliance Export
- GDPR/U.S. Export Compliance

Password Hashing
- Encryption-at-rest

Database
- (MySQL, Postgres)

File Store
- (Amazon S3)

PC Web Experience
Mobile APP Experience
Mobile Web Experience
Email Client

HTTPS
Secure Web Sockets

https://docs.mattermost.com/deployment/deployment.html
Attacks on Mattermost as Non-Admin Users

Internet

Proxy

Email Service

Notification Service

Mattermost Server

- Integrations
  - Bot
  - Webhook
  - Plugins
  - Slash Command
  - Send/Reply/Edit/Delete/Mark
  - Interactive Messages
  - Formatting Text
  - Search History
  - Link Preview
  - File Share/Preview

User Management

- Users
  - Team
  - Channel
  - Permission

Policy Compliance

- Data Retention
- Compliance Export
- GDPR/U.S. Export Compliance

Data Management

- RESTful API
- OAuth2.0

Scalability Support

- Log & Statistics
- Performance Monitoring
- Rate Limiting

Password Hashing

- Encryption-at-rest

Database

- (MySQL, Postgres)

File Store

- (Amazon S3)

HTTPS

Secure Web Sockets

PC Web Experience

Mobile APP Experience

Mobile Web Experience

Email Client

https://docs.mattermost.com/deployment/deployment.html
Attackers as Non-Users

Mattermost Server

Integrations
- Bot
- Webhook
- Plugins
- Slash Command

Messaging
- Send/Reply/Edit/Delete/Mark
- Interactive Messages
- Formatting Text
- Search History
- Link Preview
- File Share/Preview

User Authentication
- Email
- SAML 2.0
- Team Invitation
- Login and password reset page
- MFA
- AD/LDAP
- Guest Account

Site Management
- Logs & Statistics
- Performance Monitoring
- Rate Limiting

User Management
- Users
- Team
- Channel
- Permission

Policy Compliance
- Data Retention
- Compliance Export
- GDPR/U.S. Export Compliance

Scalability Support
- RESTful API
- OAuth 2.0

Data Management
- Password Hashing
- Encryption-at-rest

Database
- (MySQL, Postgres)

File Store
- (Amazon S3)

Internet
- HTTPS
- Secure Web Sockets

Proxy
- Email Service

PC Web Experience
Mobile APP Experience
Mobile Web Experience
Email Client

https://docs.mattermost.com/deployment/deployment.html
Project Timeline

- **Phase 1**: Read Documentation, Extract features
- **Phase 2**: Came up with Threat Model
- **Phase 3**: Acquired and deployed a trial of Enterprise Edition of Mattermost
- **Phase 4**: Conducted behavior analysis and source code analysis
  - Network Capture, Static Analysis, Manual Analysis, POC generation
Attackers in Server & Database

Mattermost Server

Integrations
- Bot
- Webhook
- Plugins
- Slash Command

Messaging
- Send/Reply/Edit/Delete/Mark
- Interactive Messages
- Formatting Text
- Search History
- Link Preview
- File Share/Preview

User Authentication
- Email
- SAML 2.0
- Team Invitation
- Login and password reset page
- MFA
- AD/LDAP
- Guest Account

Site Management
- Logs & Statistics
- Performance Monitoring
- Rate Limiting

User Management
- Users
- Team
- Channel Permission

Policy Compliance
- Data Retention
- Compliance Export
- GDPR/U.S. Export Compliance

Scalability Support
- Data Management
- RESTful API
- OAuth2.0

Database
- (MySQL, Postgres)

File Store
- (Amazon S3)

Internet
- HTTPS
- Secure Web Sockets

Proxy

Email Service

PC Web Experience
Mobile APP Experience
Mobile Web Experience
Email Client

https://docs.mattermost.com/deployment/deployment.html
Attackers in Server & Database

Everything is readable. But is it the end of the nightmare?

1. Credential breach
2. Session token stealing
3. Integrated applications can be compromised
Attackers in Server & DB - Credential Breach

Password

- Plain text passwords can be used for credential stuffing
- Mattermost only stores bcrypt hashed passwords into database
- bcrypt is probably not the best choice with the evolution of parallel hardware
- scrypt and argon2 provide better defense against offline parallel cracking

Password

- Plain text passwords can be used for credential stuffing

Password Hashing Competition

and our recommendation for hashing passwords: Argon2

Attackers in Server & DB - Session Stealing

- Session token generated by Google UUID (based on RFC 4122 and DCE 1.1: Authentication and Security Services)
- Valid for 180 days
- Stored in database as plain text
- Can be stolen and used to impersonate any user

```
mysql> select * from Sessions;

<table>
<thead>
<tr>
<th>Id</th>
<th>Token</th>
<th>CreateAt</th>
<th>ExpiresAt</th>
<th>LastActivityAt</th>
<th>UserId</th>
</tr>
</thead>
<tbody>
<tr>
<td>1rz45jcGspyxnxp3q7a8e7ra</td>
<td>3pqzoni6mtnzdxxnu7opnr6he</td>
<td>1587621044790</td>
<td>1603173044790</td>
<td>1587621044790</td>
<td>fg3amosfirt67czezkxgi57eger:“Chrome/81.0.4044”, &quot;csrf&quot;: &quot;wftjtaqpo378489f9ruok3msdd7o&quot;, &quot;is_guest&quot;: &quot;false&quot;, &quot;os&quot;: &quot;Mac OS&quot;, &quot;platform&quot;: &quot;Macintosh&quot;</td>
</tr>
</tbody>
</table>
```
Attackers in Server & DB - Integrated Applications

GitHub Plugin
Subscribe to repositories, stay up-to-date with reviews, assignments and more
by Mattermost on August 9, 2018
Last updated on April 13, 2020
Programmed in Go
License: Apache 2.0
Attackers in Server & DB - Integrated Applications

Realtime notification

Show TODO list
Attackers in Server & DB - Integrated Applications

Mattermost GitHub Plugin by

Notifications
Read access
This application will be able to read your notifications (no code access).
③ Learn more

Repositories
Public repositories
This application will be able to read and write all public repository data. This includes the following:
• Code
• Issues
• Pull requests
• Wikis
• Settings
• Webhooks and services
• Deploy keys
③ Learn more
Attackers in Server & DB - Integrated Applications

Note: Currently, you can't scope source code access to read-only.

Enable Code Previews

- true
- false

(Optional) Allow the plugin to expand permalinks to github files with an actual preview of the linked file.
Attackers in Server & DB - Integrated Applications

- GitHub token is AES-256 encrypted and stored in database
- Encryption key is randomly generated and stored in a json file
- Still safe when the database is dumped, but not safe when fully compromised
Attackers in Server & DB - Integrated Applications

Mattermost Configuration

Step 1: Register an OAuth Application

1. Go to https://gitlab.com/profile/apps
2. Set the following values:
   - Name: Mattermost GitLab Plugin - (your company name)
     replacing https://your-mattermost-url.com with your Mattermost
3. Select **api and read_user** in Scopes

**Scopes**

- **api**
  Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.

- **read_user**
  Grants read-only access to the authenticated user’s profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.

- **read_api**
  Grants read access to the API, including all groups and projects, the container registry, and the package registry.
Attackers as Non-Admin Mattermost Users
Attackers as Non-Admin Mattermost Users

A solid step. But what’s next?

Target

- Intellectual property and trade secret (Posts), as the threat model suggests

Approach

- Become Mattermost’s system admin (XSS attack, password cracking, SQL injection etc.)
- Trick other users (Phishing)
- Dump the database (SQL injection)
Attackers as Non-Admin Users - XSS attack

Session token is stored in cookies

- Steal token by XSS attack - document.cookie; Token is set to be `httponly`

RESTful API of role management

- Send `<script>XMLHttpRequest.open()</script>` to the admin - `<>` will be escaped to `&gt; &lt;`
- Send `[click me](JavaScript:alert();)` to the admin - URL will be checked before rendered
Attackers as Non-Admin Users - XSS attack

Session token is stored in cookies

Steal token by XSS attack – document.cookie; Token is set to be httponly

Send `<script>XMLHttpRequest.open()</script>` to the admin

URL will be checked before rendered

<> will be escaped to &gt; &lt;
Attackers as Non-Admin Users - XSS attack

Session token is stored in cookies

- Steal token by XSS attack: `document.cookie ; Token is set to be `httponly`

RESTful API of role management

- Send `<script>XMLHttpRequest.open();</script>` to the admin: `<>` will be escaped to `&gt; &lt;`
- Send `[click me] (JavaScript:alert();)` to the admin: URL will be checked before rendered
Attackers as Non-Admin Users - XSS attack

Backend sends unescaped text file to user for file preview

- Upload a malicious HTML file and trick the admin to preview it

File preview won’t be triggered for a link to the file

- Bypass the frontend escape by posting a link to the HTML file

Frontend escapes it

Content-Disposition in the header is set to attachment
Attackers as Non-Admin Users

XSS Attack

Backend sends unescaped text file to user for file preview

Upload a malicious HTML file and trick the admin to preview it

File preview won't be triggered for a link to the file

Bypass the frontend escape by posting a link to the HTML file

In the header is set to attachment
Attackers as Non-Admin Users

XSS attack

Backend sends unescaped text file to user for file preview

Upload a malicious HTML file and trick the admin to preview it

File preview won’t be triggered for a link to the file

Bypass the frontend escape by posting a link to the HTML file
Attackers as Non-Admin Users

- XSS attack

- Backend sends unescaped text file to user for file preview
- Upload a malicious HTML file and trick the admin to preview it
- File preview won't be triggered for a link to the file
- Bypass the frontend escape by posting a link to the HTML file

Frontend escapes it

Content-Disposition in the header is set to attachment
Attackers as Non-Admin Users - Phishing

1. Post a link
2. Send a GET request
3. Return full content
4. Summarize the content and display

User-Agent: Mattermost 5.22
Attackers as Non-Admin Users - Phishing

Aha! It's Mattermost making link preview

Twitter handle: PebblesPuss2014/status/1251476543538331648

Memes on Twitter
#Caturday everyone 😻
Attackers as Non-Admin Users - Phishing

Aha! It's a normal user, gonna trick him

User-Agent: Mattermost 5.22
Cute cat
User-Agent: Chrome/81.0.4044
Session Expire, Please log in
Attackers as Non-Admin Users - Phishing

User-Agent: Matter...

Session expired, please log in again

https://twitter.com/PebblesPuss2014/status/1251476543538331648

"Have a safe #Caturday everyone 😻"
Attackers as Non-Admin Users - SQL Injection

```go
func query(e SqlExecutor, query string, args ...interface{}) (*sql.Rows, error) {
    executor, ctx := extractExecutorAndContext(e)

    if ctx != nil {
        return executor.QueryContext(ctx, query, args...)
    }

    return executor.Query(query, args...)
}
```
Attckers as Non-Users
Attackers as Non-Users - User Onboard

- Email
- SAML
- Team Invitation
User Onboard - Email-based Registration

Signup

Enable Account Creation: true false
When false, the ability to create accounts is disabled. The create account button displays an error when pressed.

Restrict account creation to specified email domains: berkeley.edu, gmail.com, example.com
User accounts can only be created from a specific domain (e.g. "mattermost.org") or list of comma-separated domains (e.g. "corp.mattermost.com, mattermost.org"). This setting only affects email login for users. For Guest users, please add domains under Signup > Guest Access.

Enable Open Server: true false
When true, anyone can signup for a user account on this server without the need to be invited.

Enable Email Invitations: true false
When true users can invite others to the system using email.

Invalidate pending email invites
User Onboard - SAML-based Registration

- A XML-based protocol for exchanging identities between Identity Provider and Service Provider
- Used for Authentication
- Commonly used in Single-sign On application
- Fundamentally different from OAuth 2.0

How SAML 2.0 Works

1. Exchange certificate
2. Generate and sign the auth request
3. Verify auth request
4. Generate auth response
5. User logs in
6. User logs in the service
7. Redirect user to service provider along with auth response
8. Redirect user to identity provider along with the auth request

User logs in

Service Provider

Identity Provider

User

Generate and sign the auth request
Redirect user to identity provider along with the auth request
Verify auth request and
Generate auth response
User logs in
Redirect user to service provider along with auth response
SAML 2.0 - Sample Auth Request

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="pfx41d8ef22-e612-8c50-9968-1b16f15741b3"
    Version="2.0"
    ProviderName="SP_test"
    IssueInstant="2014-07-16T23:52:45Z"
    Destination="http://idp.example.com/SSOService.php"
    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    AssertionConsumerServiceURL="http://sp.example.com/dem01/index.php?acs">
    <samlp:Issuer>http://sp.example.com/dem01/metadata.php</samlp:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        ...
    </ds:Signature>
    <samlp:NameIDPolicy Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress" AllowCreate="true"/>
    <samlp:RequestedAuthnContext Comparison="exact">
    </samlp:RequestedAuthnContext>
</samlp:AuthnRequest>
```

https://developers.onelogin.com/saml/examples/authnrequest
SAML 2.0 - Sample Auth Response

```xml
<saml:Response xmlns:saml="urn:oasis:names:tc:SAML:2.0:protocol"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               ID="4d17e9ad988a0b437c3873c7ee05f60f6d72f1e6"
               Version="2.0"
               IssueInstant="2014-07-17T01:01:48Z"
               Destination="http://sp.example.com/demol/index.php?acs"
               InResponseTo="ONELOGIN_4fee3b046395c4e751011e977f8900b5273d56685">
    <saml:Assertion
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        ID="pfx61086f73-67d9-9633-17c5-f36349c000c8"
        Version="2.0"
        IssueInstant="2014-07-17T01:01:48Z">
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            ...
        </ds:Signature>
        <saml:AttributeStatement>
            <saml:Attribute Name="uid" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                    test
                </saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute Name="mail" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                    test@example.com
                </saml:AttributeValue>
            </saml:Attribute>
        </saml:AttributeStatement>
    </saml:Assertion>
</saml:Response>
```

https://developers.onelogin.com/saml/examples/response
SAML Signature Algorithm is not Safe

SHA-1 is a Shambles
First Chosen-Prefix Collision on SHA-1
and Application to the PGP Web of Trust

Gaëtan Leurent\textsuperscript{1} and Thomas Peyrin\textsuperscript{2,3}

\textsuperscript{1} Inria, France
\textsuperscript{2} Nanyang Technological University, Singapore
\textsuperscript{3} Temasek Laboratories, Singapore

The first collision for full SHA-1

\textsuperscript{1}, Elie Bursztein\textsuperscript{2}, Pierre Karpman\textsuperscript{1}, Ange Albertini\textsuperscript{2}, Ya

\textsuperscript{1} CWI Amsterdam
\textsuperscript{2} Google Research
info@shattered.io
https://shattered.io
If attackers could forge the Signature

1) visit

2) Auth request

3) Eavesdropping

Mattermost

4) Sign the fake Auth Response

Identity Provider

Hard but possible
## User Onboard - Team Invitation

### Signup

<table>
<thead>
<tr>
<th>Feature</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Account Creation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restrict account creation to specified email domains:</td>
<td>berkeley.edu, gmail.com, example.com</td>
<td></td>
</tr>
<tr>
<td>Enable Open Server:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enable Email Invitations:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Enable Account Creation:**
  - Default: True
  - Description: When false, the ability to create accounts is disabled. The create account button displays error when pressed.

- **Restrict account creation to specified email domains:**
  - User accounts can only be created from a specific domain (e.g. "mattermost.org") or list of comma-separated domains (e.g. "corp.mattermost.com, mattermost.org"). This setting only affects email login for users. For Guest users, please add domains under Signup > Guest Access.

- **Enable Open Server:**
  - Default: False
  - Description: When true, anyone can signup for a user account on this server without the need to be invited.

- **Enable Email Invitations:**
  - Depends on the setting of Enable Open Server.
  - When true users can invite others to the system using email.
User Onboard - Team Invitation

Invite Members to 261test

Share This Link

http://34.94.128.164/signup_user_complete/?id=pqqbcx9irprqpmjhrf

Share this link to invite people to this team.
The `uuid` package generates and inspects UUIDs based on RFC 4122 and DCE 1.1: Authentication and Security Services. This package is based on the github.com/pborman/uuid package (previously named code.google.com/p/go-uuid). It differs from these earlier packages in that a UUID is a 16 byte array rather than a byte slice. One loss due to this change is the ability to represent an invalid UUID (vs a NIL UUID).

Install

```go
go get github.com/google/uuid
```
Attackers as Non-Users - Team Invitation

Invite Members to 261test

Share This Link

http://34.94.128.164/signup_user_complete/id=pqqbcx9iprqmjhfr

Share this link to invite people to this team.

Invite Members to 261test

Share This Link

http://34.94.128.164/signup_user_complete/id=pqqbcx9iprqmjhfr

Copy Link
Attackers as Non-Users - Team Invitation
Attackers as Non-Users - Login

⚠ Enter a valid email or username and/or password, or sign in using another method.

admin

......

Sign in

Password Reset

If the account exists, a password reset email will be sent to: admin@test.com

Please check your inbox.
Attackers as Non-Users - Login

⚠️ Your account is locked because of too many failed password attempts. Please reset your password.

changze

...
Attackers as Non-Users - Login

⚠️ Your account is locked because of too many failed password attempts. Please reset your password.

- changze

- ...

Sign in

⚠️ Enter a valid email or username and/or password, or sign in using another method.

- changze

- ...

Sign in
Attackers as Non-Users - Phishing with webhook

Application A

Exposed URL

Application B
Attackers as Non-Users - Phishing with webhook

1. Create an incoming webhook
2. Returns a secret link
3. Send a request to the secret link when a trading curb is triggered
4. HTTP POST: {"text": "alarm!"}
5. Post "alarm!" to a channel
6. Steal admin’s account
7. HTTP POST: {"text": "phishing!"}
8. Post "phishing!" to a channel
Attackers as Non-Users - Phishing with webhook

- A BOT tag is attached to messages posted by webhook

  weihao  BOT  4:51 PM
  Changze is our new team leader now, please add changze@evil.com to the github repo.

- Allows username and icon override

  Robinhood  BOT  4:51 PM
  Your robinhood account is logged out, please login again.

Enable integrations to override usernames:

When true, webhooks, slash commands and other integrations, such as Zapier, will be allowed to change the username they are posting as. Note: Combined with allowing integrations to override profile picture icons, users may be able to perform phishing attacks by attempting to impersonate other users.
Attacks in Supply Chain Vulnerabilities

Internet

HTTP Secure Web Sockets

PC Web Experience
Mobile APP Experience
Mobile Web Experience
Email Client

Proxy

Notification Service

Email Service

Mattermost Server

Integrations
- Bot
- Webhook
- Plugins
- Slash Command
- Send/Reply/Edit/Delete/Mark
- Interactive Messages
- Formatting Text
- Search History
- Link Preview
- File Share/Preview

User Management
- Users
- Team
- Channel Permission

Policy Compliance
- Data Retention
- Compliance Export
- GDPR/U.S. Export Compliance

Scalability Support

Password Hashing

Encryption-at-rest

Database
- (MySQL, Postgres)

File Store
- (Amazon S3)

User Authentication
- SAML 2.0
- Team Invitation
- Login and password reset page
- MFA
- AD/LDAP
- Guest Account

Site Management
- Logs & Statistics
- Performance Monitoring
- Rate Limiting

RESTful API
- OAuth2.0

https://docs.mattermost.com/deployment/deployment.html
Attackers in Supply Chain Vulnerabilities

49 personal projects

Mattermost Server

271 open source projects in total

<table>
<thead>
<tr>
<th>Project</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masterminds / squirrel</td>
<td>Code, Issues 7, Pull requests 7</td>
</tr>
<tr>
<td>nytimes / gziphandler</td>
<td>Code, Issues 12, Pull requests 7</td>
</tr>
<tr>
<td>armon / go-metrics</td>
<td>Code, Issues 20, Pull requests 16</td>
</tr>
<tr>
<td>beevik / etree</td>
<td>Code, Issues 4, Pull requests 4</td>
</tr>
<tr>
<td>miekg / dns</td>
<td>Code, Issues 1, Pull requests 1</td>
</tr>
<tr>
<td>bing / semver</td>
<td>Code, Issues 58, Pull requests 7, Active</td>
</tr>
<tr>
<td>daintegration / imaging</td>
<td>Code, Issues 6, Pull requests 8, Active</td>
</tr>
<tr>
<td>fsnotify / fsnotify</td>
<td>Code, Issues 120, Pull requests 16, Active</td>
</tr>
<tr>
<td>biotravel / asnt-ber</td>
<td>Code, Issues 2, Pull requests 4, Active</td>
</tr>
<tr>
<td>go-sol-driver / mysql</td>
<td>Code, Issues 48, Pull requests 4, Active</td>
</tr>
<tr>
<td>go-asnt-ber / asnt-ber</td>
<td>Code, Issues 2, Pull requests 4, Active</td>
</tr>
<tr>
<td>go-sol-driver / mysql</td>
<td>Code, Issues 48, Pull requests 4, Active</td>
</tr>
<tr>
<td>dasky / naxa</td>
<td>Code, Issues 2, Pull requests 7, Active</td>
</tr>
<tr>
<td>hako / durafmt</td>
<td>Code, Issues 11, Pull requests 7</td>
</tr>
<tr>
<td>hashicorp / go-immutable-radix</td>
<td>Code, Issues 2, Pull requests 2, Active</td>
</tr>
<tr>
<td>hashicorp / go-plugin</td>
<td>Code, Issues 11, Pull</td>
</tr>
<tr>
<td>sirup / logrus</td>
<td>Code, Issues 80, Pull requests 4, Active</td>
</tr>
<tr>
<td>dgryski / dgogauth</td>
<td>Code, Issues 2</td>
</tr>
<tr>
<td>jaytaylor / html2text</td>
<td>Code, Issues 4</td>
</tr>
<tr>
<td>jmoirion / sqlx</td>
<td>Code, Issues 178, Pull requests 53, Active</td>
</tr>
<tr>
<td>muesli / smartcrop</td>
<td>Code, Issues 4, Pull requests 3, Actions</td>
</tr>
<tr>
<td>sean- / seed</td>
<td>Code, Issues 6, Pull requests 9</td>
</tr>
</tbody>
</table>

Fluent SQL generation for golang
Go middleware to gzip HTTP responses
A Golang library for exporting performance metrics
DNS library in Go
Semantic Versioning (semver) library written in golang
Imaging is a simple image processing package for Go
Cross-platform file system notifications for Go
ASN1 BER Encoding/Decoding Library for Go
Go MySQL Driver is a MySQL driver
A powerful HTTP router and URL matcher
Better time duration formatting in Go!
An immutable radix tree implementation in Golang
Golang plugin system over RPC
Google Authenticator for Go
Golang HTML to plaintext conversion library
general purpose extensions to golang’s database/sql
smartcrop finds good image crops for arbitrary crop sizes
Securely seed Go’s random number generator
Attackers in Supply Chain Vulnerabilities

Golang HTML to plaintext conversion library  https://jaytaylor.com/html2text

```go
func SendMail(c smtpClient, mail mData, fileBackend filesstore.FileBackend, date time.Time, *model.AppError) {
    mlog.Debug("sending mail", mlog.String("to", mail.smtpTo), mlog.String("subject", mail.subject))

    htmlMessage := "\n\n<html><body>" + mail.htmlBody + "</body></html>"

    txtBody, err := html2text.FromString(mail.htmlBody)
    if err != nil {
        mlog.Warn("Unable to convert html body to text", mlog.Err(err))
        txtBody = ""
    }
}
```
Attackers Aiming at Plugins

- Allows installing plugin binaries from untrusted third party
- No permission control for plugins
- Plugins have the same permission as Mattermost itself

Security

Plugins are intentionally powerful and not artificially sandboxed in any way and effectively become part of the Mattermost server. Server plugins can execute arbitrary code alongside your server and webapp plugins can deploy arbitrary code in client browsers.

While this power enables deep customization and integration, it can be abused in the wrong hands. Plugins have full access to your server configuration and thus also to your Mattermost database. Plugins can read any message in any channel, or perform any action on behalf of any user in the webapp.

You should only install custom plugins from sources you trust to avoid compromising the security of your installation.
Future Work Beyond Our Project

● A closer look at Mattermost
  ○ Notification system
  ○ Client side software
  ○ Upgrading
  ○ License Management

● Apply our investigation method to similar applications (e.g., Slack)
Thanks