Matrix Bot Workshop

Sumner Evans and Robby Zampino

March 1, 2022

A bit about us

- Sumner
  - Graduated from Mines in 2019 with a master's in CS.
  - Work at Beeper, a company that is building a Matrix-based chat app.
  - Teaching *CSCI 406 Algorithms* and previously *CSCI 400* and *CSCI 564*.
- Robby
  - Graduated from Mines in 2019 with a bachelor's in EE and minor in CS.
  - Founder and admin of ohea.xyz, an internet collective (ask me if you want an ohea.xyz Matrix account).

We became interested in Matrix when we were looking for an open source chat platform for ACM!

Overview

1. Introducing Matrix
2. A brief overview of how Matrix works
3. Writing a Matrix Bot

Introducing Matrix

What is Matrix?

Matrix is an **open** specification for **encrypted**, **decentralized** communication.

Matrix is an *open specification*

Open specifications and standards are all around you. They just make sense™.

Examples:

- Power plugs
- USB
- Wi-Fi
- Every crypto algorithm that's any good

Open protocols allow for *open development* and *clean-room implementations*, they *encourage competition*, and are *externally auditable*.

Matrix is an *open specification*

Open specifications and standards are all around you. They just make sense™.

Examples:

- Power plugs
- USB
- Wi-Fi
- Every crypto algorithm that's any good

Open protocols allow for *open development* and *clean-room implementations*, they *encourage competition*, and are *externally auditable*.

Matrix is an *open specification*

Open specifications and standards are all around you. They just make sense™.

Examples:

- Power plugs
- USB
- Wi-Fi
- Every crypto algorithm that's any good

Open protocols allow for *open development* and *clean-room implementations*, they *encourage competition*, and are *externally auditable*.

Matrix is *encrypted* by default*

Matrix has encryption built-in.

The core of the encryption is **Olm**, which is a clone of the Signal double-ratchet protocol.

- If a single key is compromised, the attacker cannot see past messages. This is called **forward secrecy**.
- Key exchanges happen often, and if an attacker misses a single key exchange, they are once again locked out. This is called **break-in recovery**.

You end up with 1:1 Olm ratchets between all participants in the room.

Those ratchets are used to share the key data for the group ratchet (called Megolm) which is used to encrypt messages.

Matrix is *encrypted* by default*

Matrix has encryption built-in.

The core of the encryption is **Olm**, which is a clone of the Signal double-ratchet protocol.

- If a single key is compromised, the attacker cannot see past messages. This is called **forward secrecy**.
- Key exchanges happen often, and if an attacker misses a single key exchange, they are once again locked out. This is called **break-in recovery**.

You end up with 1:1 Olm ratchets between all participants in the room.

Those ratchets are used to share the key data for the group ratchet (called Megolm) which is used to encrypt messages.

Matrix is *encrypted* by default*

Matrix has encryption built-in.

The core of the encryption is **Olm**, which is a clone of the Signal double-ratchet protocol.

- If a single key is compromised, the attacker cannot see past messages. This is called **forward secrecy**.
- Key exchanges happen often, and if an attacker misses a single key exchange, they are once again locked out. This is called **break-in recovery**.

You end up with 1:1 Olm ratchets between all participants in the room.

Those ratchets are used to share the key data for the group ratchet (called Megolm) which is used to encrypt messages.

Matrix is *decentralized*

The Matrix architecture is actually a *federated* architecture.

Individual devices communicate to a *homeserver* which anyone can host.

The homeserver communicates with other homeservers in the federation.

Think of it like email. You can email somebody using Outlook from Gmail.*

Every server in the federation gets a copy of a room, so no one entity controls the network.

This also means that the network is resilient to censorship, individual server outages, or even wider internet outages.

Matrix is *decentralized*

The Matrix architecture is actually a *federated* architecture.

Individual devices communicate to a *homeserver* which anyone can host.

The homeserver communicates with other homeservers in the federation.

Think of it like email. You can email somebody using Outlook from Gmail.*

Every server in the federation gets a copy of a room, so no one entity controls the network.

This also means that the network is resilient to censorship, individual server outages, or even wider internet outages.

Matrix is *decentralized*

The Matrix architecture is actually a *federated* architecture.

Individual devices communicate to a *homeserver* which anyone can host.

The homeserver communicates with other homeservers in the federation.

Think of it like email. You can email somebody using Outlook from Gmail.*

Every server in the federation gets a copy of a room, so no one entity controls the network.

This also means that the network is resilient to censorship, individual server outages, or even wider internet outages.

Matrix is *decentralized*

The Matrix architecture is actually a *federated* architecture.

Individual devices communicate to a *homeserver* which anyone can host.

The homeserver communicates with other homeservers in the federation.

Think of it like email. You can email somebody using Outlook from Gmail.*

Every server in the federation gets a copy of a room, so no one entity controls the network.

This also means that the network is resilient to censorship, individual server outages, or even wider internet outages.

Matrix allows for *bridges* and *bots*

Bridges bring external chat networks into Matrix. (Sumner gave a talk about these earlier in the year.)

Bots allow for automated interactions and notifications. This is the focus of this workshop.

Matrix allows for *bridges* and *bots*

Bridges bring external chat networks into Matrix. (Sumner gave a talk about these earlier in the year.)

Bots allow for automated interactions and notifications. This is the focus of this workshop.

A brief overview of how Matrix works

Two APIs

The **Client-Server API** specifies how clients communicate with their homeserver.

This is the one we care about.

The **Server-Server API** or **Federation API** specifies how servers communicate with other servers to ensure that everyone has the same room state.

Two APIs

The **Client-Server API** specifies how clients communicate with their homeserver.

This is the one we care about.

The **Server-Server API** or **Federation API** specifies how servers communicate with other servers to ensure that everyone has the same room state.

Everything is an event

Everything* in Matrix is an event or a room. There are two main event types: **message events** and **state events**.

Message events:

These describe transient 'once-off' activities in a room such as an instant message, VoIP call setup, file transfer, etc. They generally describe communication activity.

State events:

These describe updates to a given piece of persistent information ('state') related to a room, such as the room's name, topic, membership, participating servers, etc. State is modelled as a lookup table of key/value pairs per room, with each key being a tuple of `state_key` and `event type`. Each state event updates the value of a given key.

See `https://matrix.org/docs/spec/#room-structure`

Everything is an event

Everything* in Matrix is an event or a room. There are two main event types: **message events** and **state events**.

Message events:
These describe transient 'once-off' activities in a room such as an instant message, VoIP call setup, file transfer, etc. They generally describe communication activity.

State events:
These describe updates to a given piece of persistent information ('state') related to a room, such as the room's name, topic, membership, participating servers, etc. State is modelled as a lookup table of key/value pairs per room, with each key being a tuple of `state_key` and `event type`. Each state event updates the value of a given key.

See `https://matrix.org/docs/spec/#room-structure`

Everything is an event

Everything* in Matrix is an event or a room. There are two main event types: **message events** and **state events**.

Message events:
These describe transient 'once-off' activities in a room such as an instant message, VoIP call setup, file transfer, etc. They generally describe communication activity.

State events:
These describe updates to a given piece of persistent information ('state') related to a room, such as the room's name, topic, membership, participating servers, etc. State is modelled as a lookup table of key/value pairs per room, with each key being a tuple of `state_key` and `event type`. Each state event updates the value of a given key.

See `https://matrix.org/docs/spec/#room-structure`

Writing a Matrix Bot

Maubot

We will be showing you how to use Maubot, a Matrix bot framework by Tulir (one of Sumner's coworkers).

Maubot is a Python bot framework. It provides nice utilities on top of the `mautrix-python` library.

Admin Panel: We have set up a maubot instance here:

`https://argon.ohea.xyz/_matrix/maubot/`

Username: `demo`
Password: `matrixiscool`

Maubot

We will be showing you how to use Maubot, a Matrix bot framework by Tulir (one of Sumner's coworkers).

Maubot is a Python bot framework. It provides nice utilities on top of the `mautrix-python` library.

Admin Panel: We have set up a maubot instance here:

`https://argon.ohea.xyz/_matrix/maubot/`

Username: `demo`
Password: `matrixiscool`

General bot-writing workflow

Documentation:

`https://docs.mau.fi/maubot/dev/getting-started.html`

Install maubot on your local machine:

```
pip3 install --user maubot
```

Writing your bot:

- Run **`mbc init`** for interactive plugin creation.
- Run **`mbc login`** to connect to the maubot instance.
- After you write your code, upload it to the server with:
  `mbc build --upload`
- Create a user for a client:
  `mbc auth --register --homeserver argon.ohea.xyz --server argon`
- Create a client and instance on the Maubot Manager.

Example: Echobot

Example: Firefighter bot

Bot Ideas

Now it's your turn! We recommend you start with something small (an existing bot for example), then build up from there.

- Try writing a cooler echobot or a better reaction bot.
- A bot that sends back ASCII-art of the given word or phrase.
- A bot that applies a filter to images sent in a room.
- A 20 questions bot.
- A dice roll bot
- Welcome bot that says hi to new users.
- Upvote/downvote tracker bot that tracks the number of thumbs up and thumbs down emojis that each user gets.
- Any other ideas?