
ORACLE®

Oracle Digital Assistant
The Complete Training

Oracle Digital Assistant Security Overview

Image courtesy of pixabay.com

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Topic agenda

1. Concerns
2. Security architecture
3. Channel security
4. OCI Security
5. Application security
6. Coding practices

Topic agenda

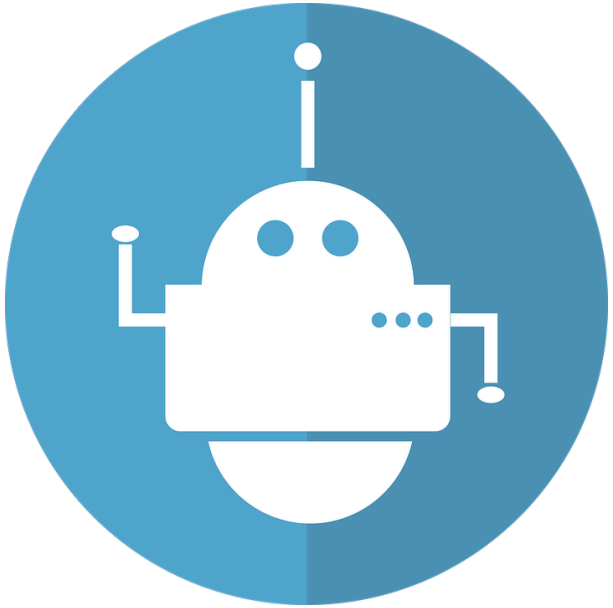
1. Concerns
2. Security architecture
3. Channel security
4. OCI Security
5. Application security
6. Coding practices

Security concerns with social media messengers

- Data privacy is at the discretion of the provider
  - Social media companies make money with targeted advertising
  - Protection on data transit, but not clear who else has access
- Identity theft and impersonation

Security concerns with chatbots

- Trusting messenger applications
  - Channel implementation may involve 3rd party messaging servers
- Sensitive data stored in logs
- Denial-of-service attacks
  - How to throttle number of bot requests
- Insufficient incident monitoring



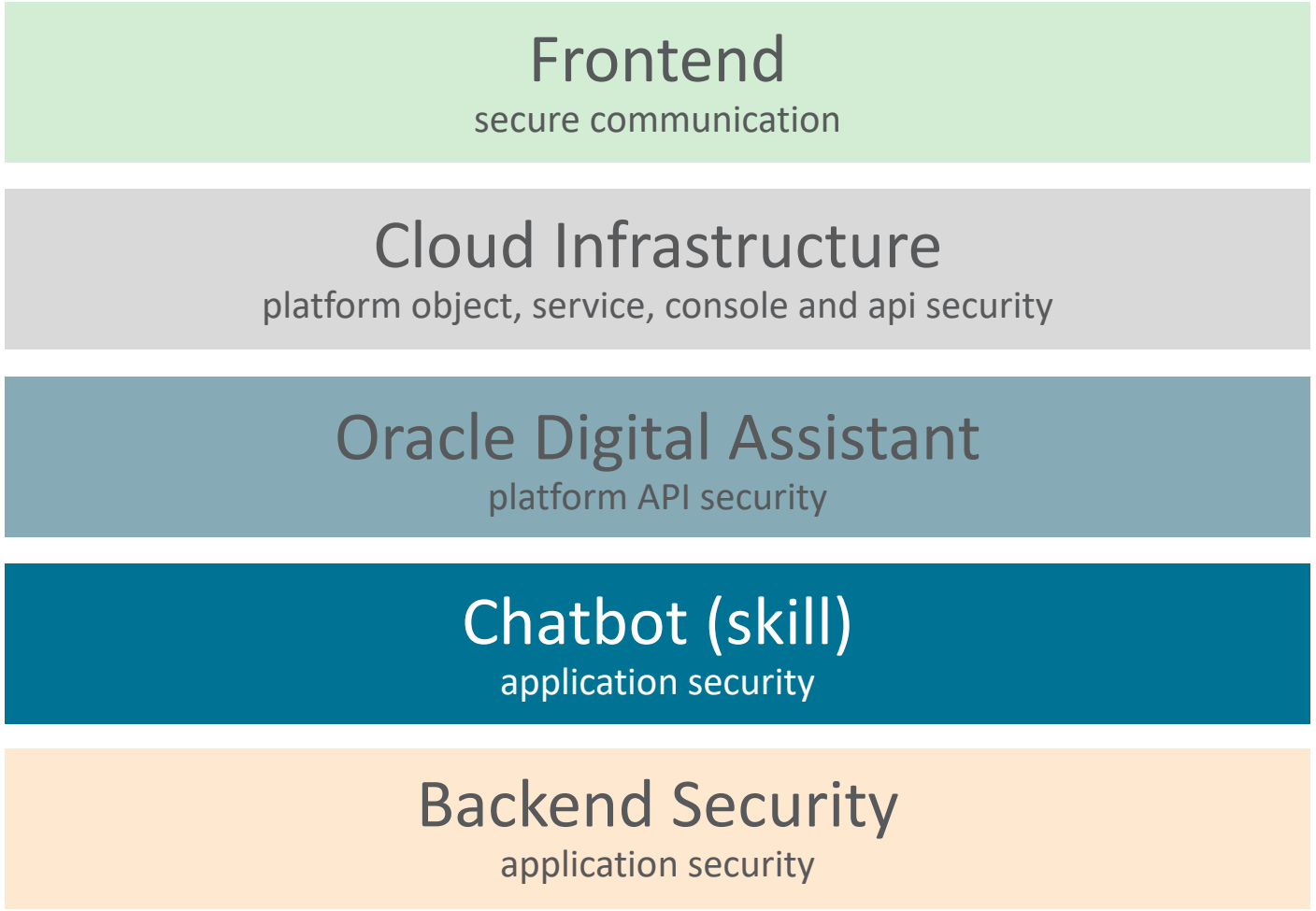
Web application threats that apply to chatbots

- "Shoulder surfing”
- Script injection
- Broken access control
- Sensitive data exposure
- Security misconfiguration
- Message replaying
- Insufficient logging

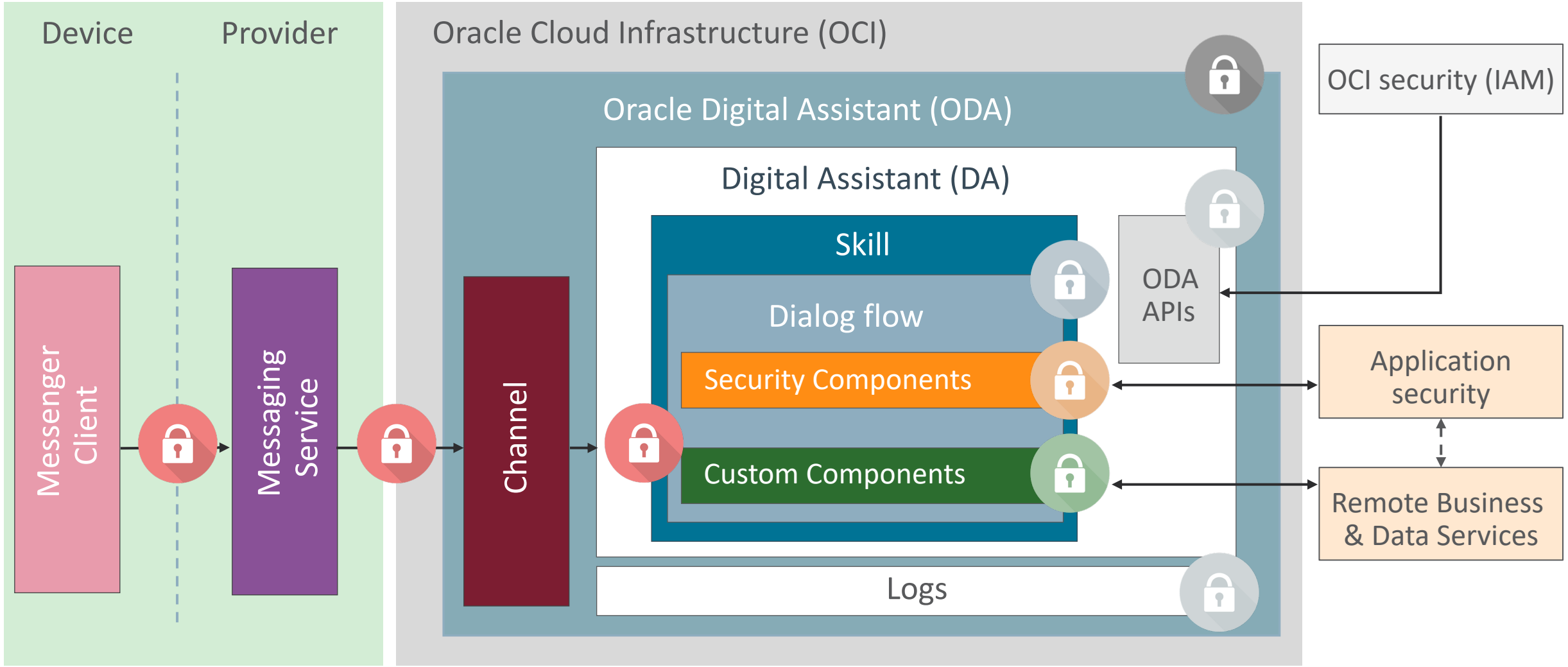
Topic agenda

1. Concerns
2. Security architecture
3. Channel security
4. OCI Security
5. Application security
6. Coding practices

Multi layer chatbot security



Oracle Digital Assistant multi layer security architecture

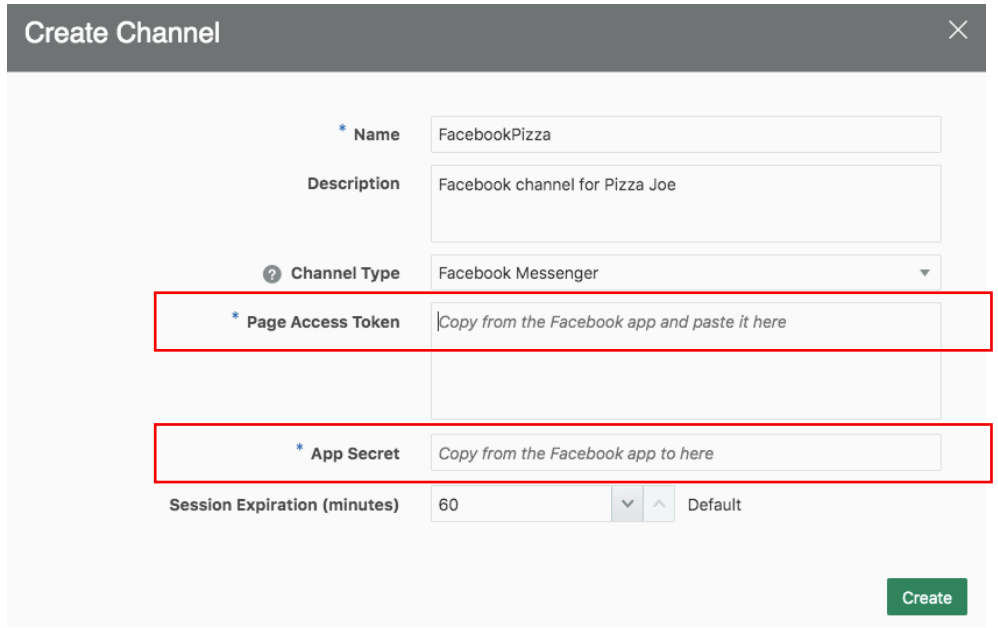


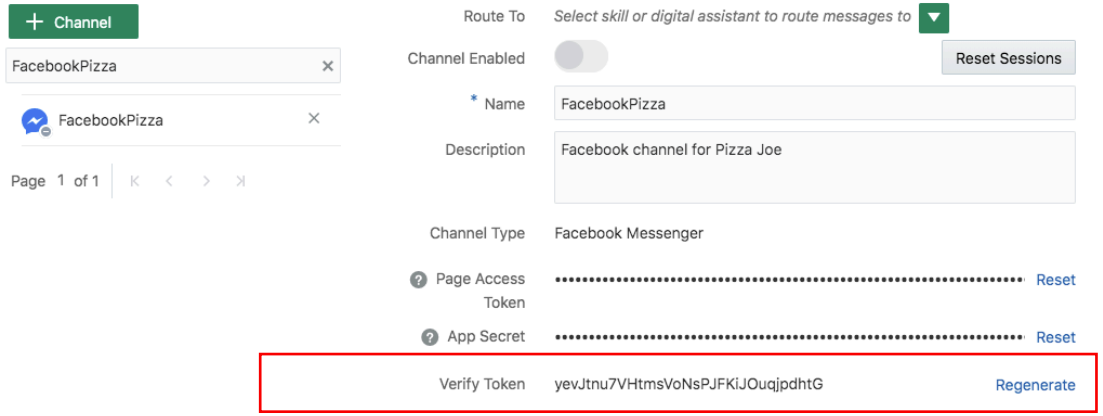
Topic agenda

1. Concerns
2. Security architecture
3. Channel security
4. OCI Security
5. Application security
6. Coding practices

Common channel security

- HTTPS for protecting messages in transit
- SHA 256 signature prevents messages to be changed on transit
- Different messengers have different security configuration
  - Channels in Oracle Digital assistant assist you in creating messenger specific secrets





The security between the user device and the message server is the responsibility of the message channel provider

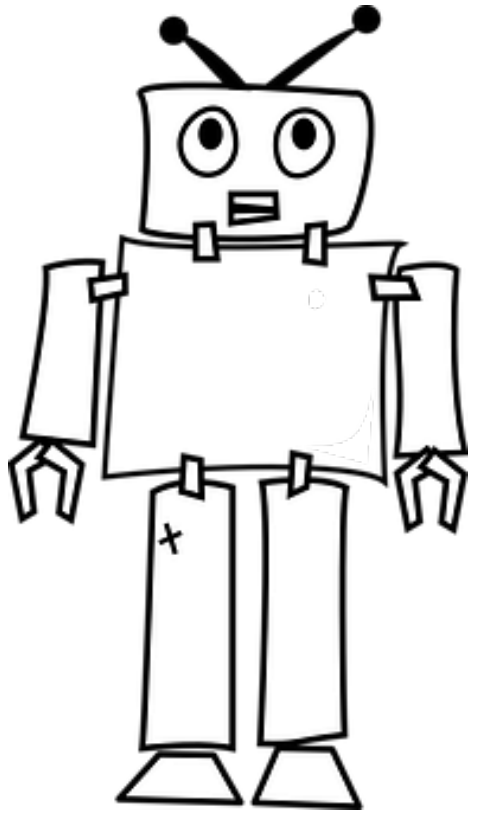
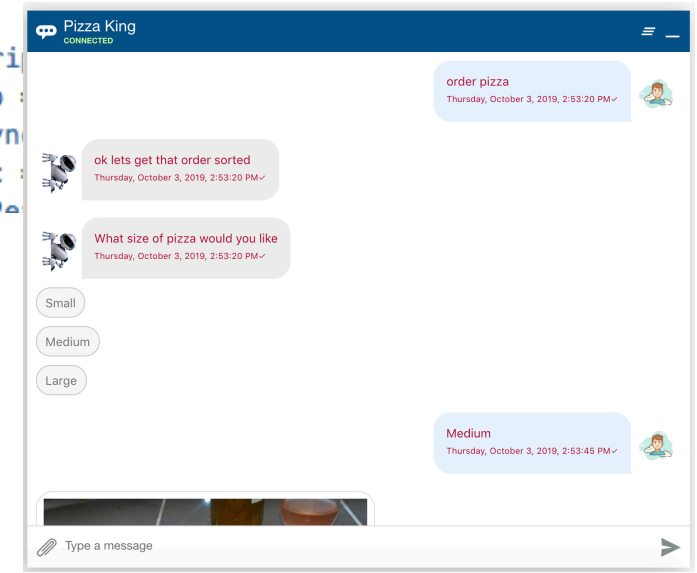


Image courtesy of pixabay.com

Oracle web SDK

- Oracle owned JavaScript library to add digital assistant to web applications and web pages
- Uses Oracle own chat server
  - No Smooch
- Highly customizable web messenger
- State of the art messenger features
  - timestamp display, typing indicator
  - Document and image upload
  - Security

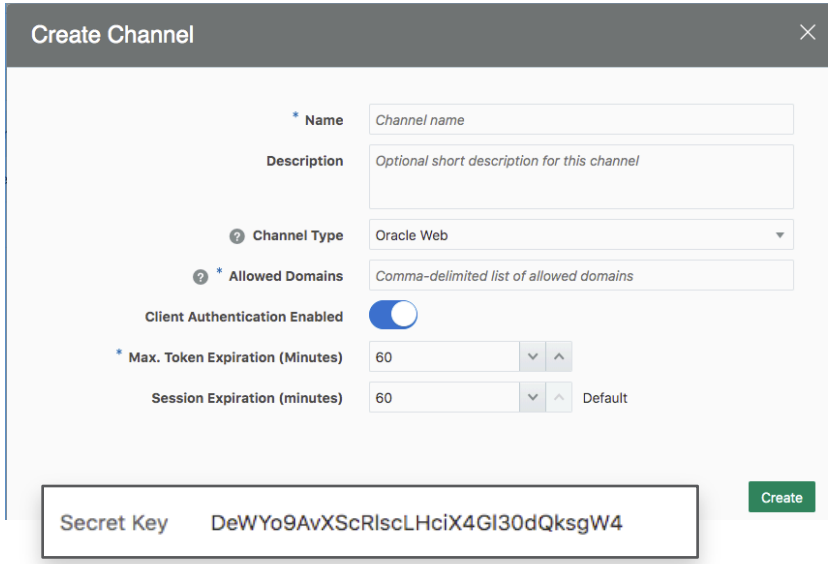
```
<script>
    var chatWidgetWebSettings = {
        URI: '<Chat server URI>',
        channelId: '<Channel ID>',
        userId: '<User ID>'
    };
    !function(globalObj, doc, library, name) {
        function initiateSDK() {
            setTimeout(function() {
                globalObj[name] = Object.assign(WebSDK);
                globalObj[name] = new globalObj[name](chatWidg
                globalObj[name].connect();  // Connect to chat
            }, 2000);
        }
        try {
            var scri
                lib :
            lib.asyn
            lib.src :
            lib.onRe
```



Oracle web messenger client security

- Client authentication using signed JSON Web Tokens (JWT)
  - Generate signed JWT token on custom server
    - Web channel secret as signing key
  - Signed JWT token added as 'Bearer' token in Authorization header for each request
  - ODA validates information in token body
- Domain whitelisting
  - Restricts domains allowed to access channel
  - Works with and without client authentication

```
<script>
    var chatWidgetWebSettings = {
        URI: '<Chat server URI>',
        clientAuthEnabled: true
    };
    var generateToken = function() {
        return new Promise((resolve) => {
            fetch('https://yourbackend.com/endpointToGenerateJWTToken').then((token) => {
                resolve(token);
            })
        });
    }
    !function(globalObj, doc, library, name) {
        function initiateSDK() {
            setTimeout(function() {
                globalObj[name] = Object.assign(WebSDK);
                globalObj[name] = new globalObj[name](chatWidgetWebSettings, generateToken);
                globalObj[name].connect();  // Connect to chat server
            }, 2000);
        }
```



Hiding / exposing information based on channels

- Some channels may be more trusted than others
  - Customize ODA based on channel type and channel name
- Expression
  - `${system.message.channelConversation.type}`
  - `${system.message.channelConversation.channelName}`
- System Common Response Component
  - channel include | exclude property to determine visibility
- Custom component:
  - `conversation.channelType()`
    `Converation.request().message.channelConversation.channelName`

Topic agenda

1. Concerns
2. Security architecture
3. Channel security
4. OCI Security
5. Application security
6. Coding practices

HTTP Request Security

- TLS 1.2
  - Transport layer security
- Basic & Bearer authentication
  - HTTP endpoint protection
- OAuth2
  - Token based authorization
- Cryptographic signature
  - Message integrity protection

OCI Security

- Oracle cloud infrastructure security
  - Secures APIs, cloud objects (e.g. storage) and services
  - Perimeter security of data centers
- OCI Identity and Access Management (IAM)
  - Manages users, user groups, resource groups (compartments), policies
  - Security for network-, compute-, storage-, console- and APIs
  - Users are platform users like admins, developers, business analysts
- Authorization requires cryptographic signatures

OCI security protects cloud APIs and resources. It separates custom application security from cloud security.

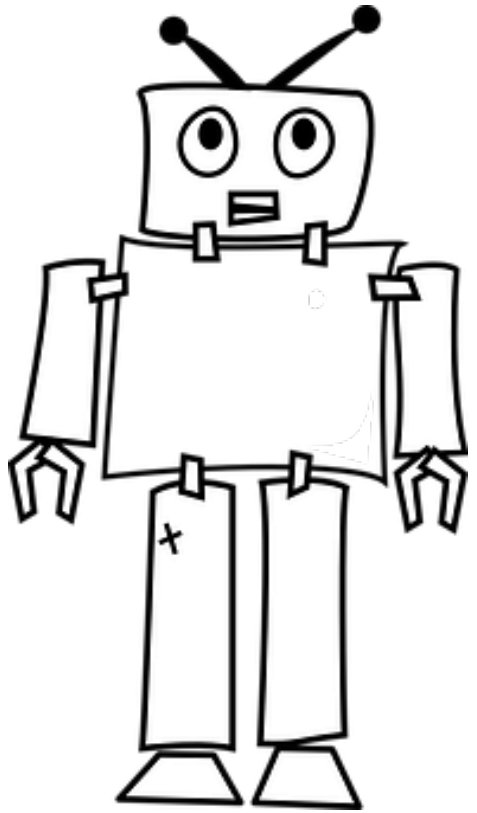
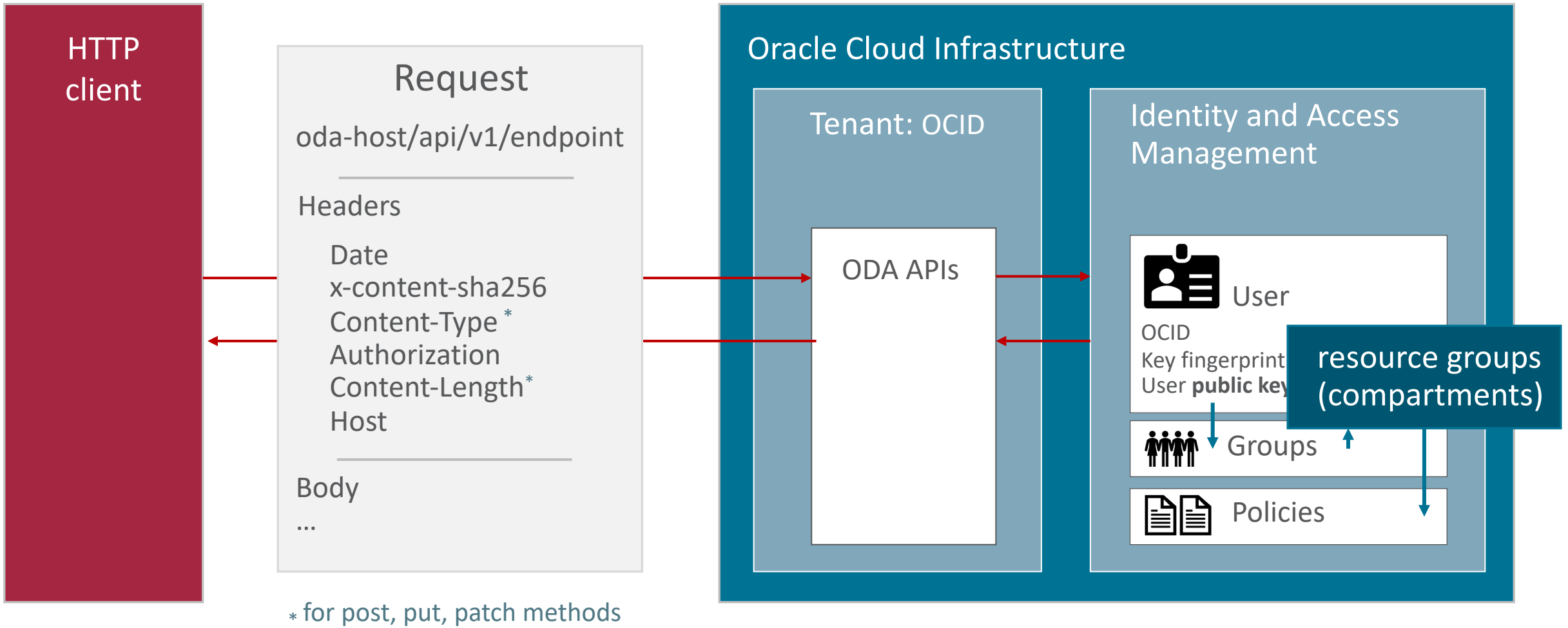


Image courtesy of pixabay.com

Anatomy of an API request in OCI



* for post, put, patch methods

Topic agenda

1. Concerns
2. Security architecture
3. Channel security
4. OCI Security
5. Application security
6. Coding practices

Application **security** today **is less about who the user is**. It's about whether a client is authorized to access a protected resource.

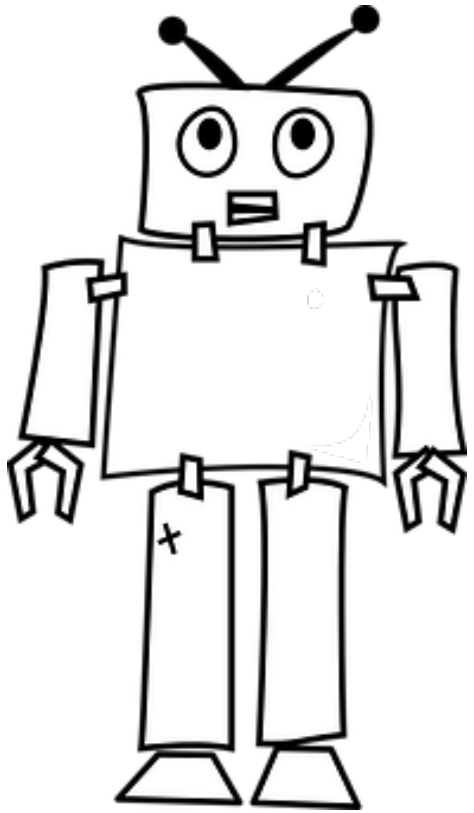


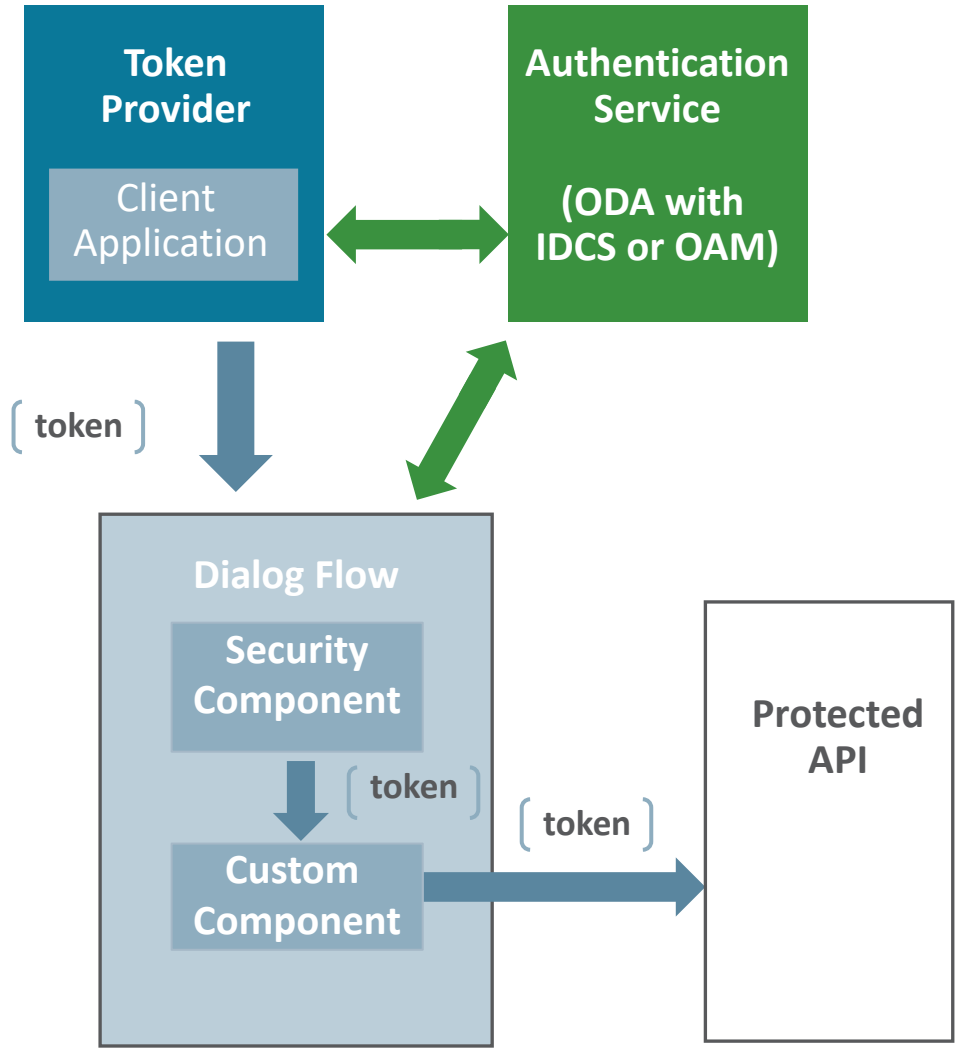
Image courtesy of pixabay.com

OAuth2 Authorization

- Industry standard for token based authorization
  - Widely used (including in social media)
- Commonly used authorization types
  - Authorization code flow
    - Client accesses resource API on behalf of a user
    - User authenticates to authorization provider to obtain access token
    - Client uses access token to access remote service APIs
  - Client credential flow
    - Resource API is accessed on behalf of a system
    - Uses shared client ID and client secret to obtain access token
    - Client uses access token to access remote service APIs

OAuth2 authorization in ODA skills

- ODA supports authorization code- and client credential grant flows
- Declarative support for OAuth2 with Oracle IDCS and OAM through
  - Authentication service
  - OAuth2 security components
- Support for 3rd party providers
  - 2-legged authorization
- Protected resources accessed from custom components



Built-in OAuth2 security components

- System.OAuth2AccountLink
  - 3-legged authorization
  - Obtains access token from Oracle Identity Cloud Service
  - Supports refresh tokens
- System.OAuthAccountLink
  - 2-legged authorization
  - Obtains authorization token from 3rd party OAuth2 authorization provider
  - Requires custom component to obtain access token
- System.OAuth2Client
  - Obtains access token based on client credentials

Request authorization in ODA is not limited to OAuth2

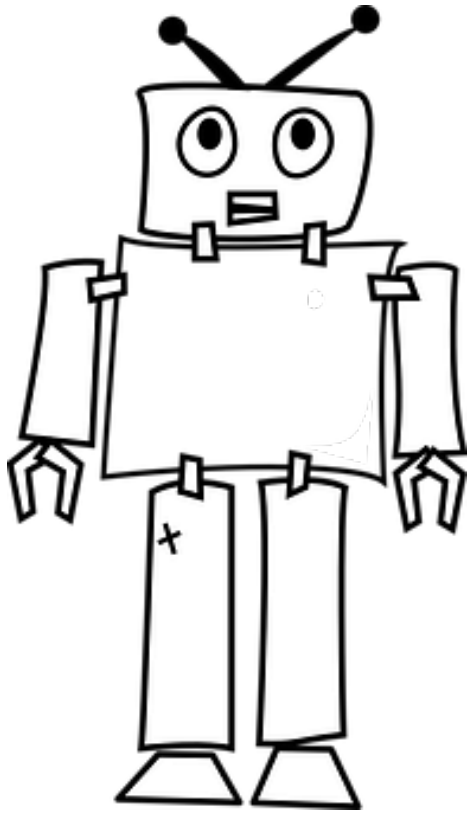


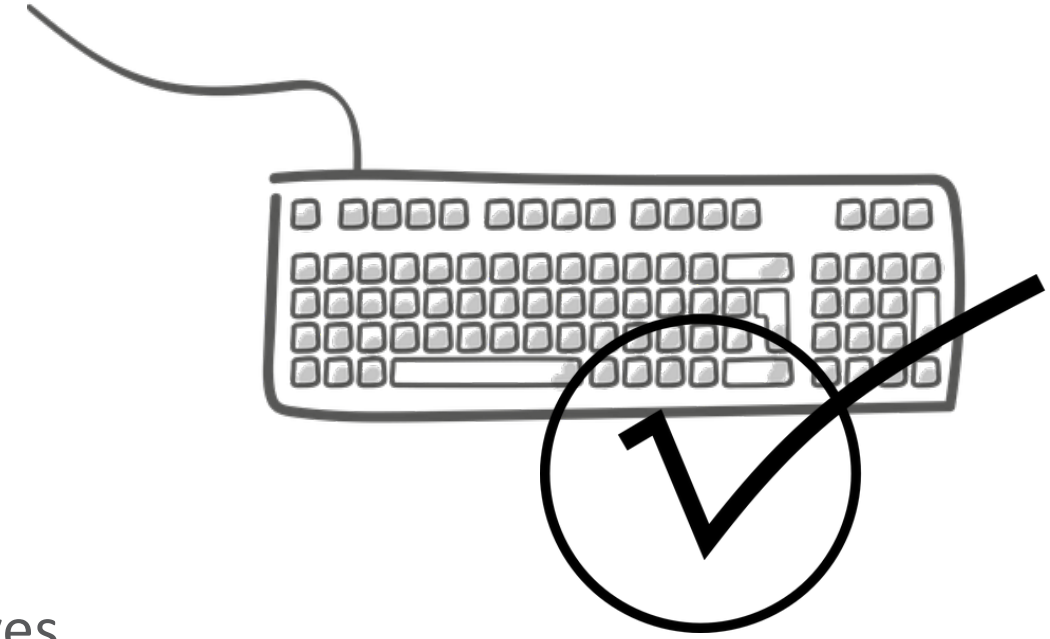
Image courtesy of pixabay.com

Topic agenda

1. Concerns
2. Security architecture
3. Channel security
4. OCI Security
5. Application security
6. Coding practices

Validate user input

- Dialog flow
  - Apache FreeMarker
  - Entity validation
    - System.MatchEntity
    - Using entity type variables on input components
- Custom Components
  - Use and validate input parameters
- Backend systems
  - Backends should know to protect themselves



Keep sensitive information out of the chat history

- Unless you fully control the messenger: Don't use it for any sensitive data input and display
  - Use System.Webview to provide information outside of the messenger
- Truncate or mask sensitive information like credit card numbers or license keys in bot responses
  - Use Apache FreeMarker expressions or custom components

Use out-of-order message handling

- Posting of old action message is referred to as "out-of-order message"
- Out-of-order message handling need to be evaluated for each usecase
  - Default implementation is to allow out-of-order messages
  - Use cases to consider
    - Allow and follow out-of-order messages
    - Suppress out of order messages
    - Ask user what to do
- Don't allow sensitive actions to be executed from the chat history
  - Configure out-of-order message handling to suppress the action or direct to a state informing the user that the requested option is no longer available

Consider restricting content based on the channel

- Social media messengers may be considered less trustworthy
- Optimize skill responses to exclude sensitive information on less trusted channels
- Conversation flows and custom components have access to
  - Channel type (e.g. facebook, web, webhook etc.)
    - ${system.message.channelConversation.channelType}
    - ${system.message.channelConversation.channelName}
    - conversation.channelType()
  - The channel name you defined when configuring a channel

Backend service security

- Remote business and data services must protect themselves
  - Enforce security
  - Do not make assumptions about the security enforced by the client
  - Validate and authorize requests

Integrated Cloud

Applications & Platform Services


ORACLE®