Mix Up Attacks in MCP

Emily Lauber

Sr. Product Manager in Identity @ Microsoft

Facilitator for Auth Mix Up Attack Prevention WG

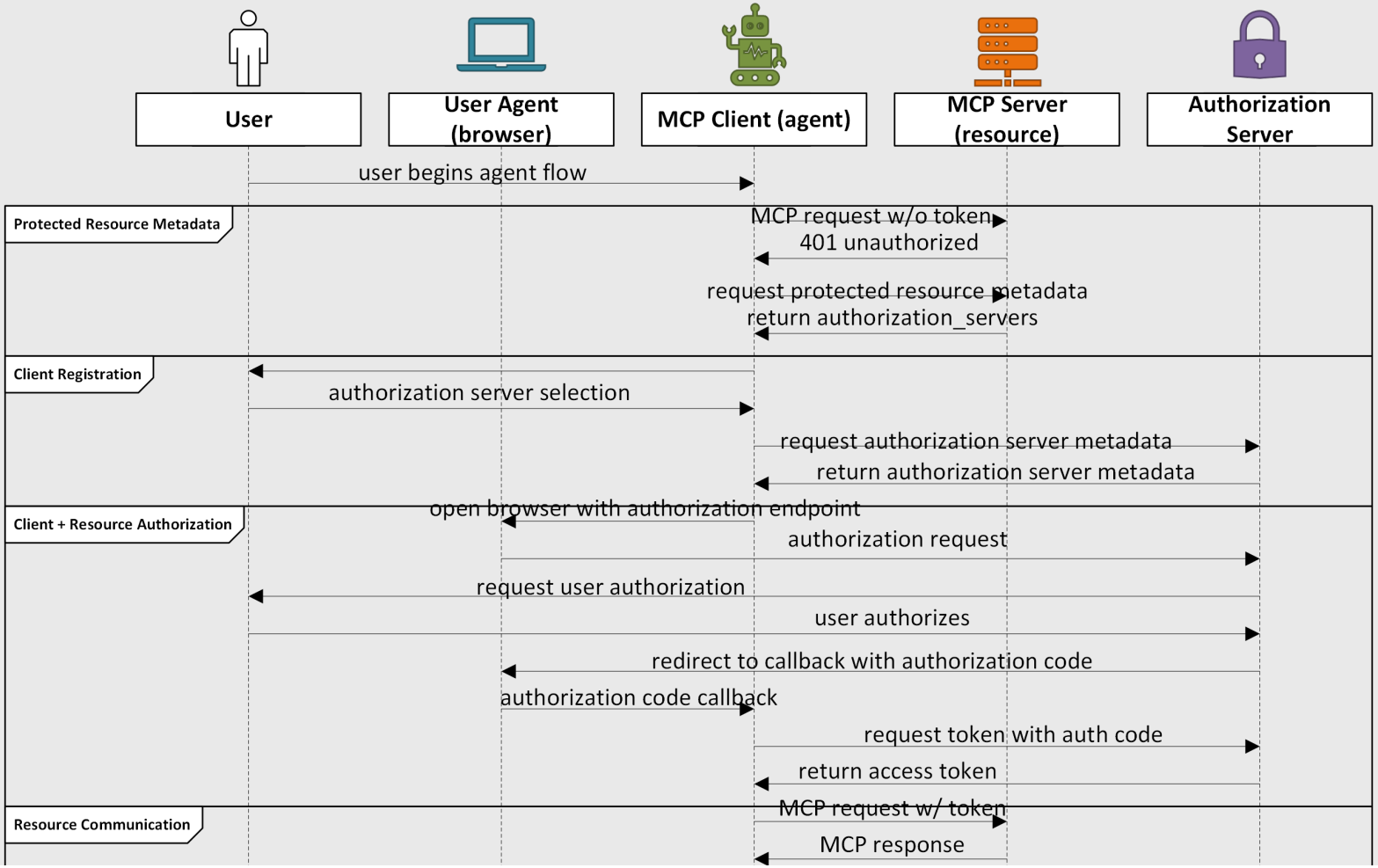
Mix Up Attack Prevention WG

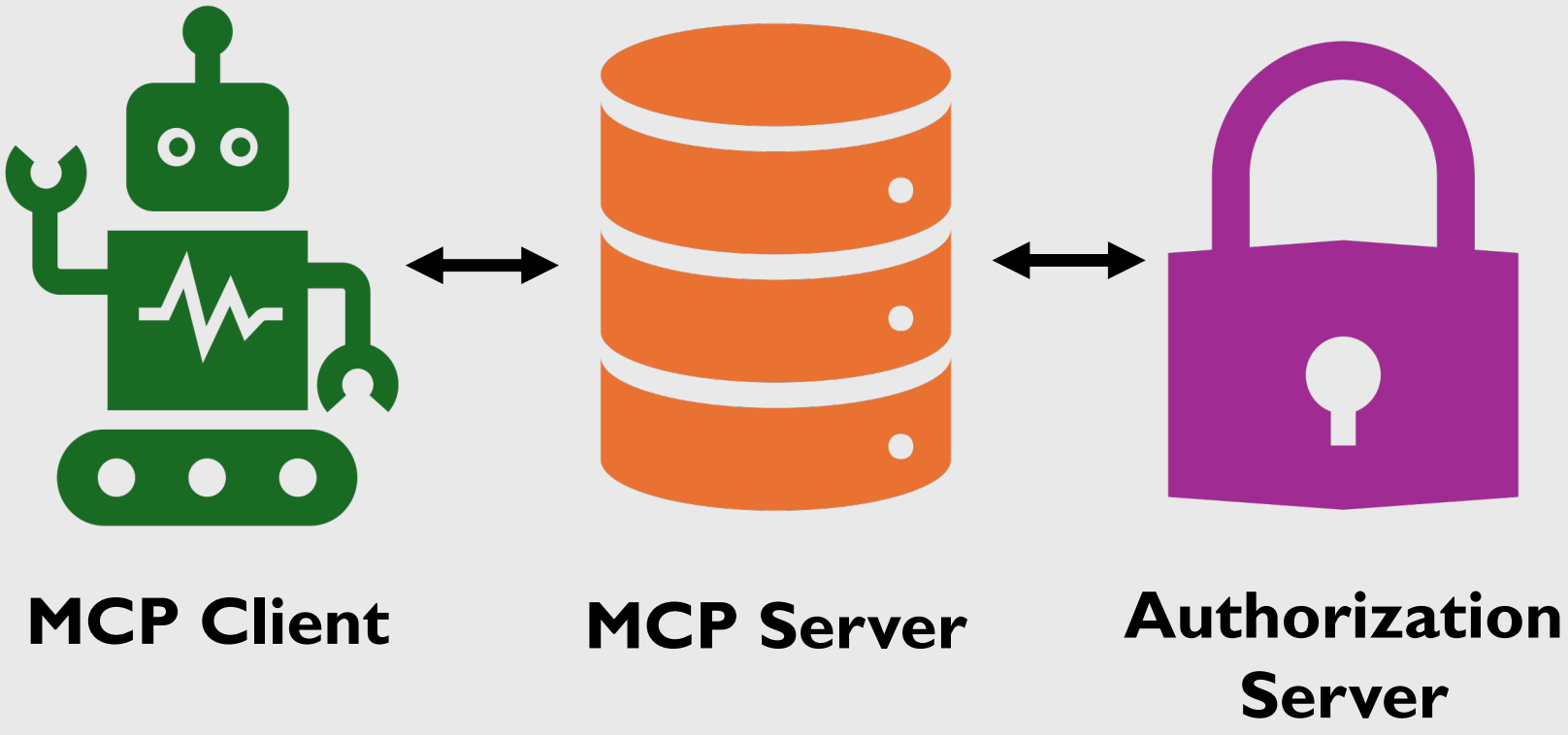
- Attack is subtle and hard to detect
- MCP architecture lends itself to Mix Up attacks
- MCP specification and ecosystem can change to address it

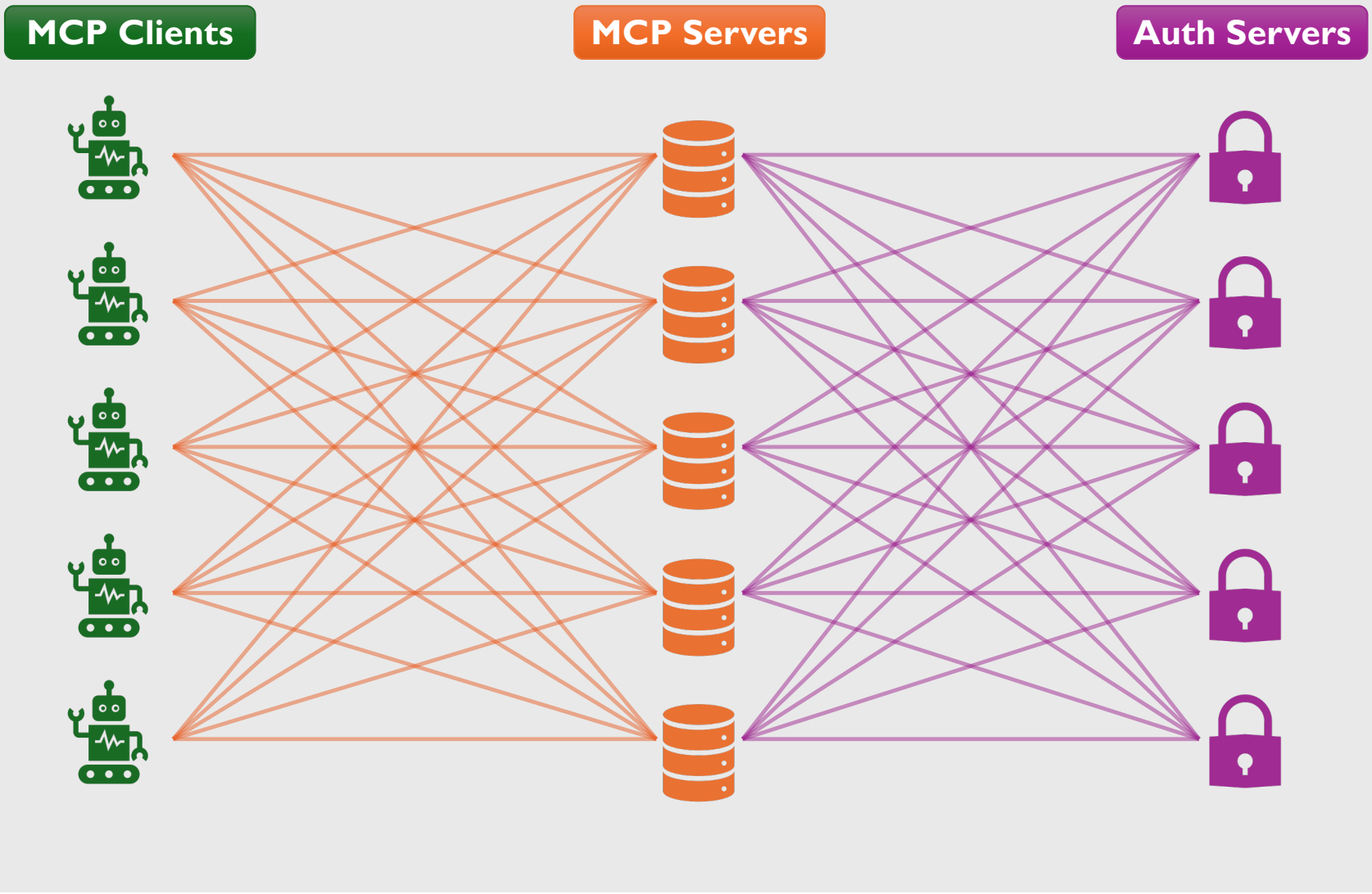
MCP Authorization Flow

Overview

MCP Authorization

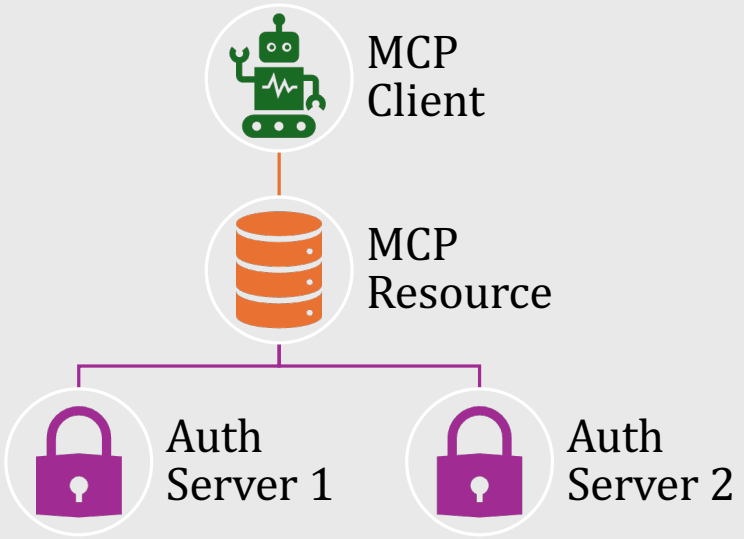



MCP Client
MCP Server
Authorization Server


MCP Clients
MCP Servers
Auth Servers

Mix-Up Attack Requirements

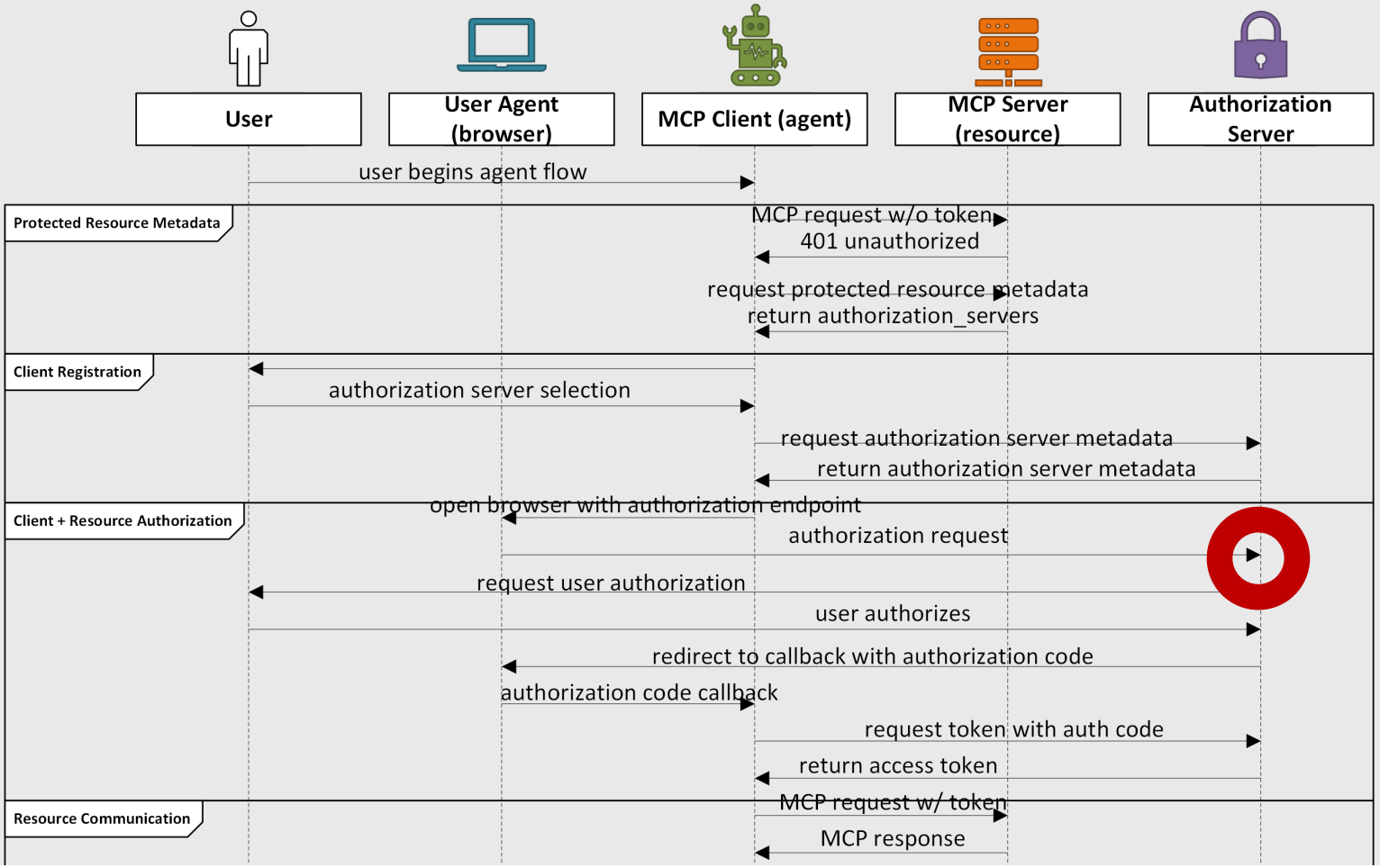
Client interacts with multiple authorization servers



Client uses a single redirect_URI

https://example.client.com/auth/redirect

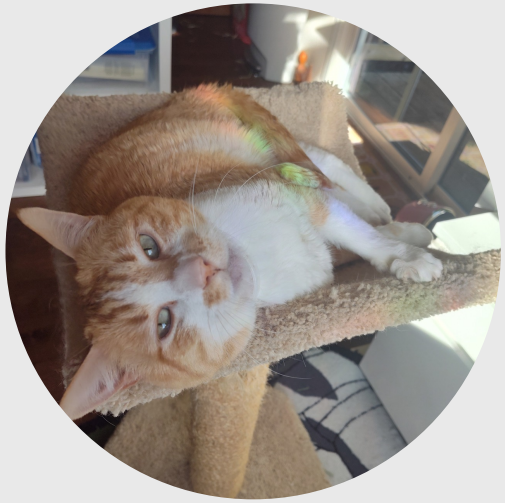
Attack Points



Pasta Bois

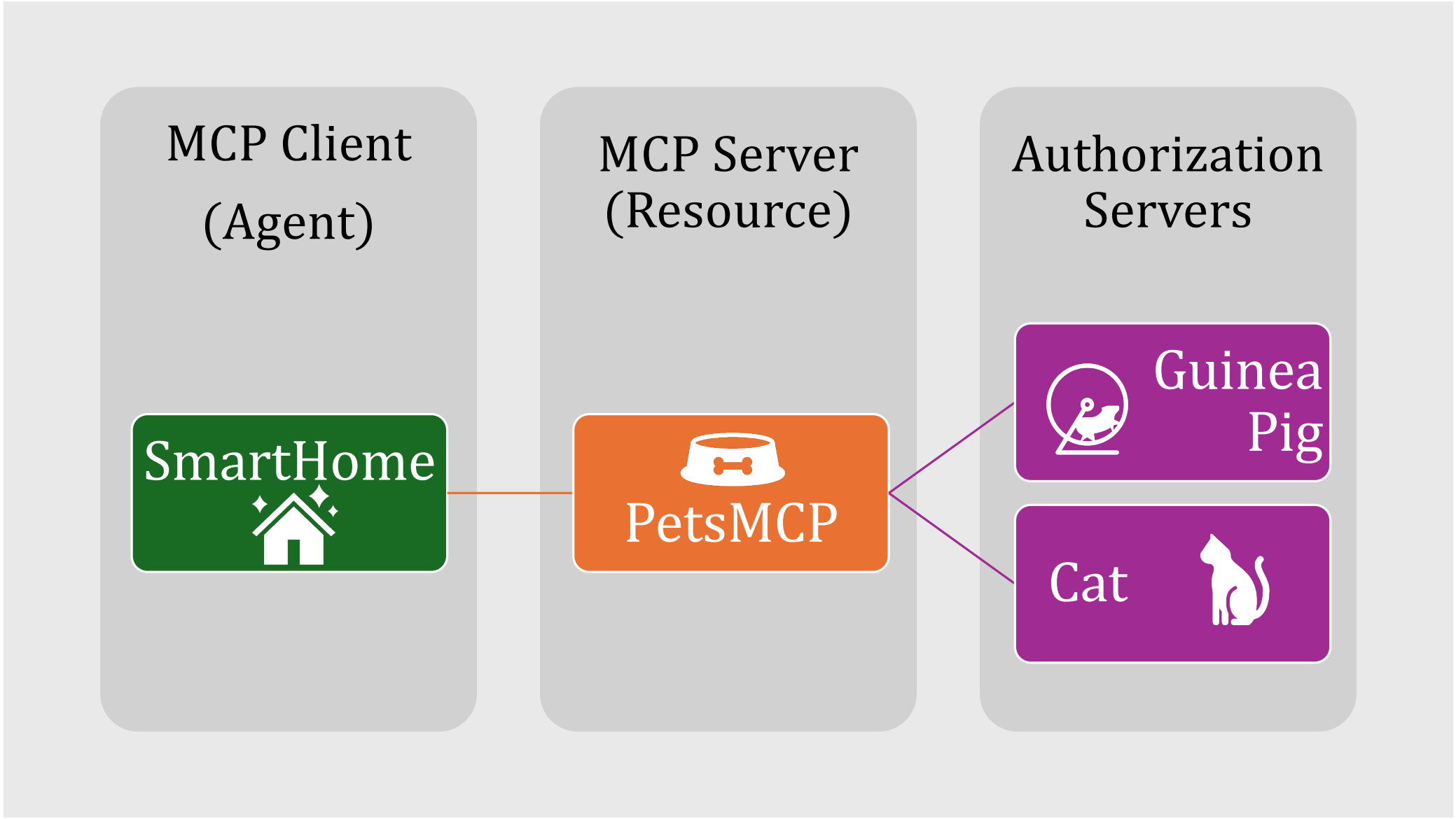
Rigatoni
Guinea Pig

Orzo
Guinea Pig

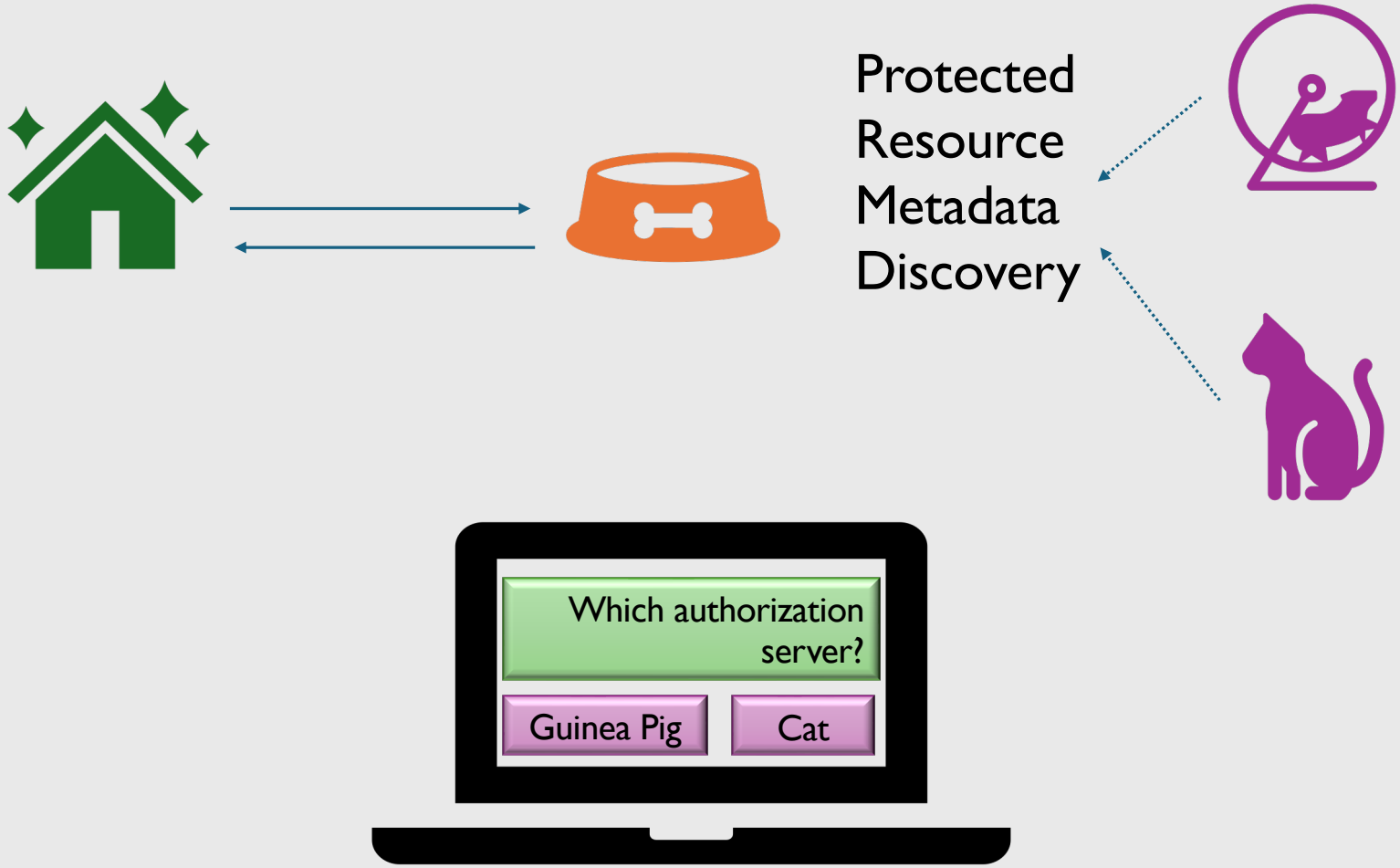


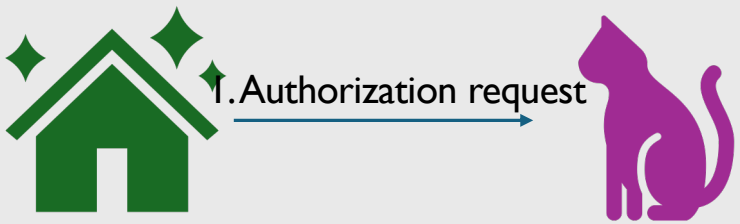
Fuseli
Cat

Doesn’t the cat attack the guinea pigs?


MCP Client
(Agent)
MCP Server
(Resource)
Authorization
Servers
SmartHome
PetsMCP
Guinea
Pig
Cat

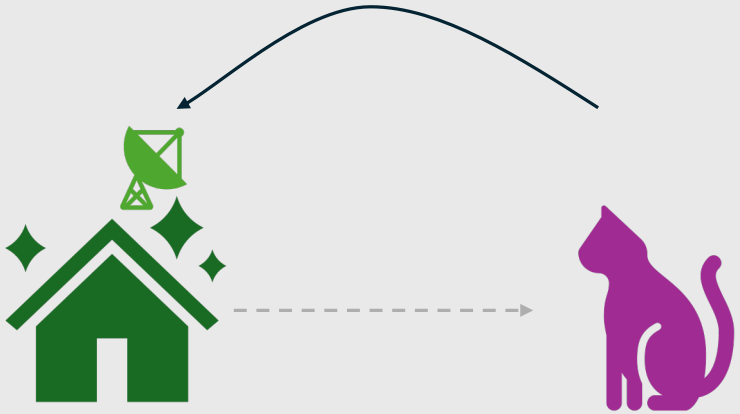

How can I help?
Please feed my pets

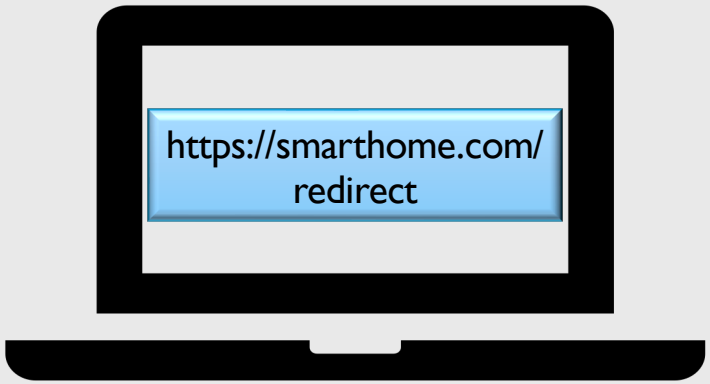

Protected
Resource
Metadata
Discovery
Which authorization server?
Guinea Pig
Cat


1. Authorization request

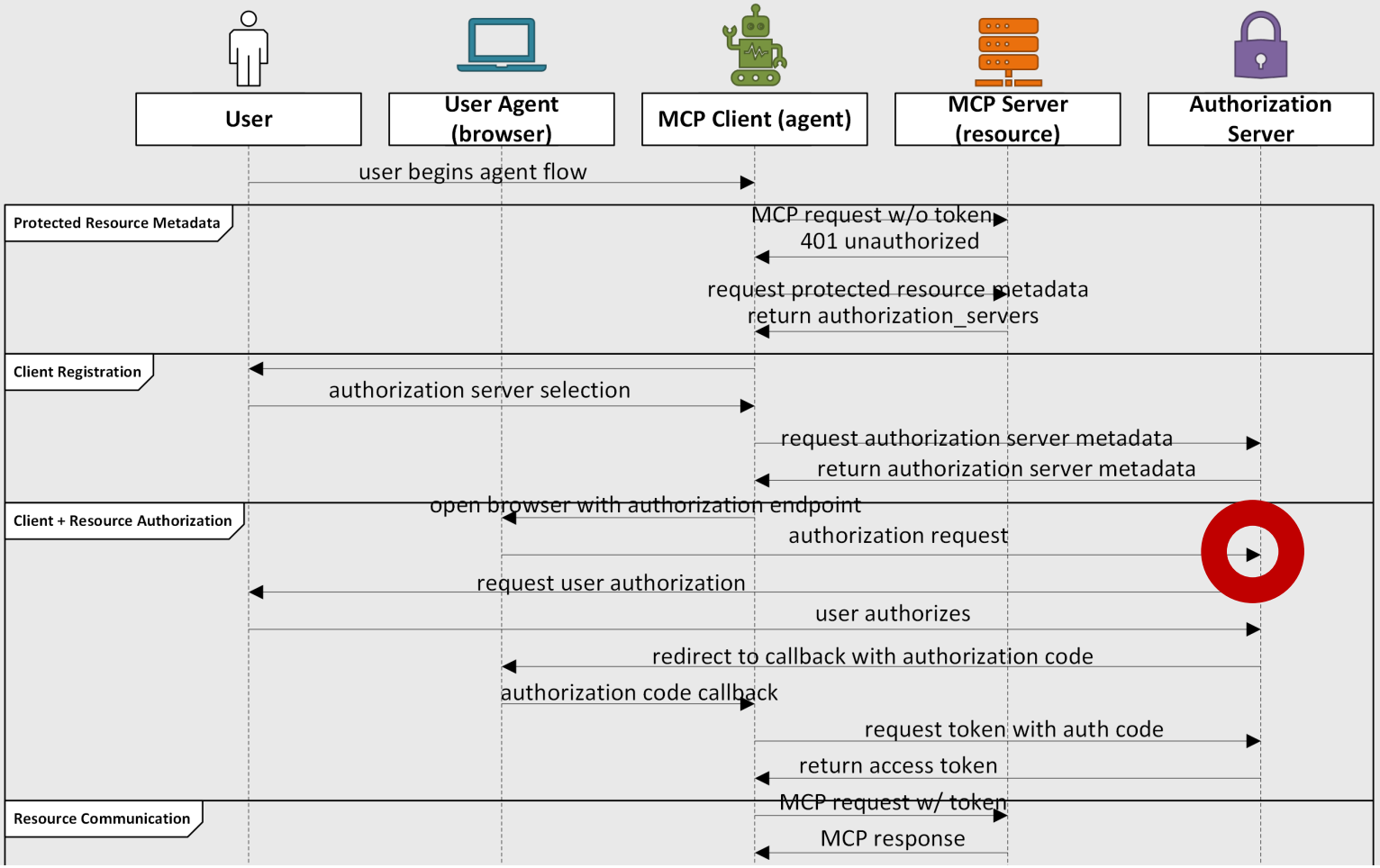

Authorize SmartHome
to PetMCP?

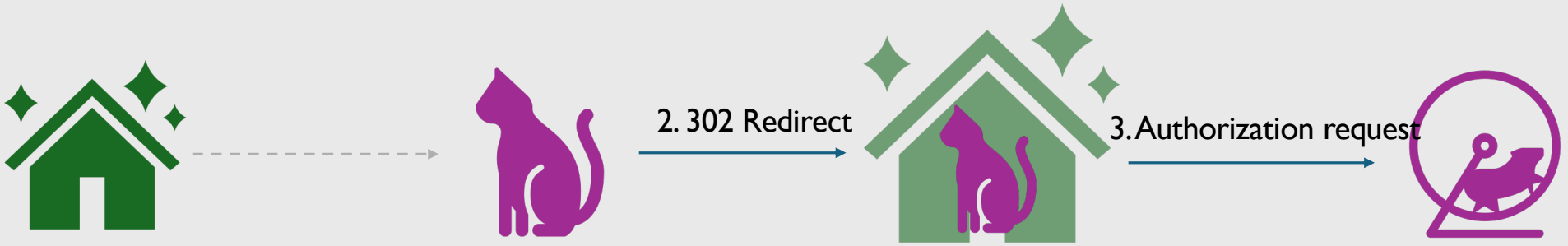
2. Auth code to client redirect_URI





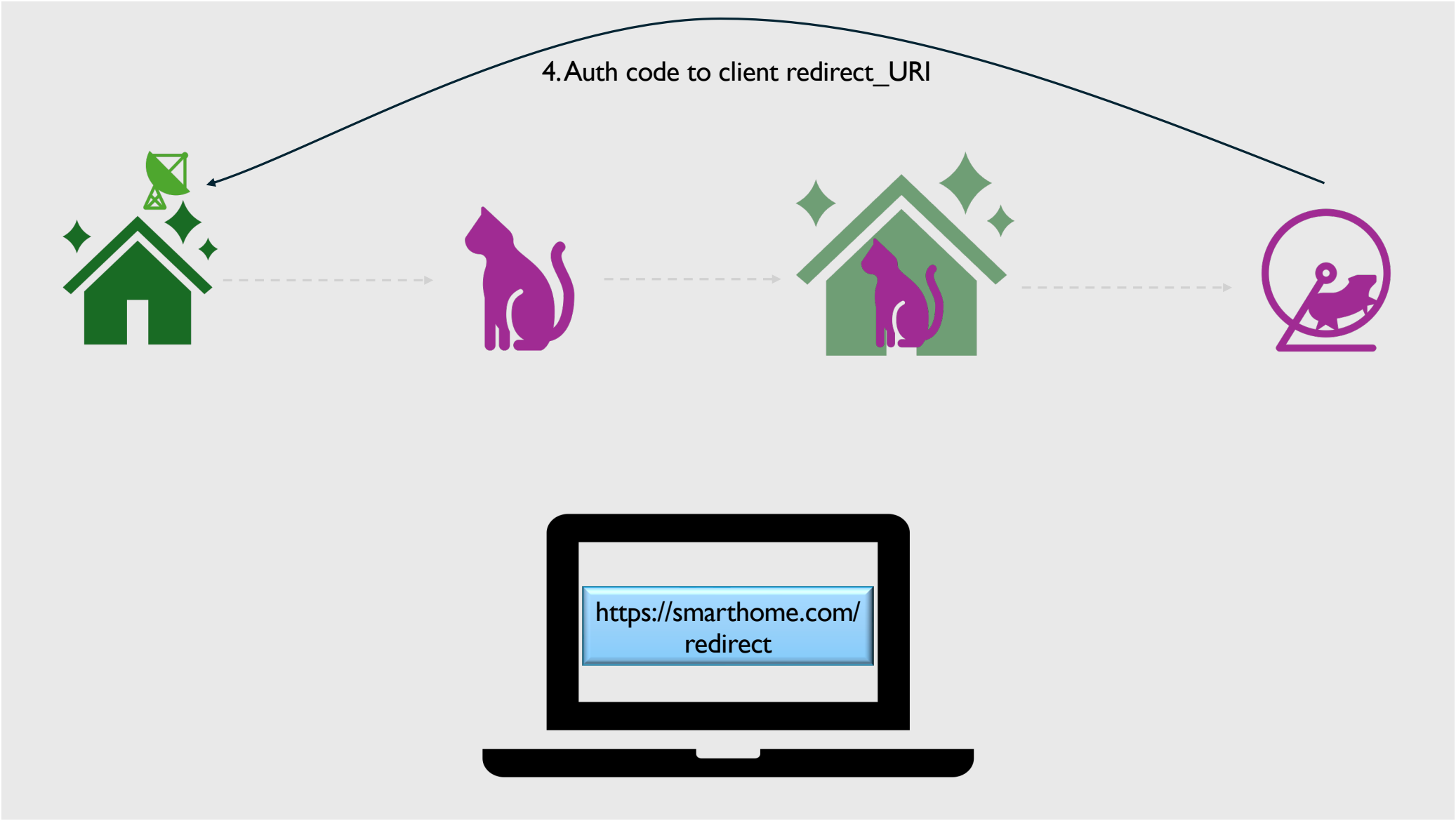
Malicious Auth Server Redirects

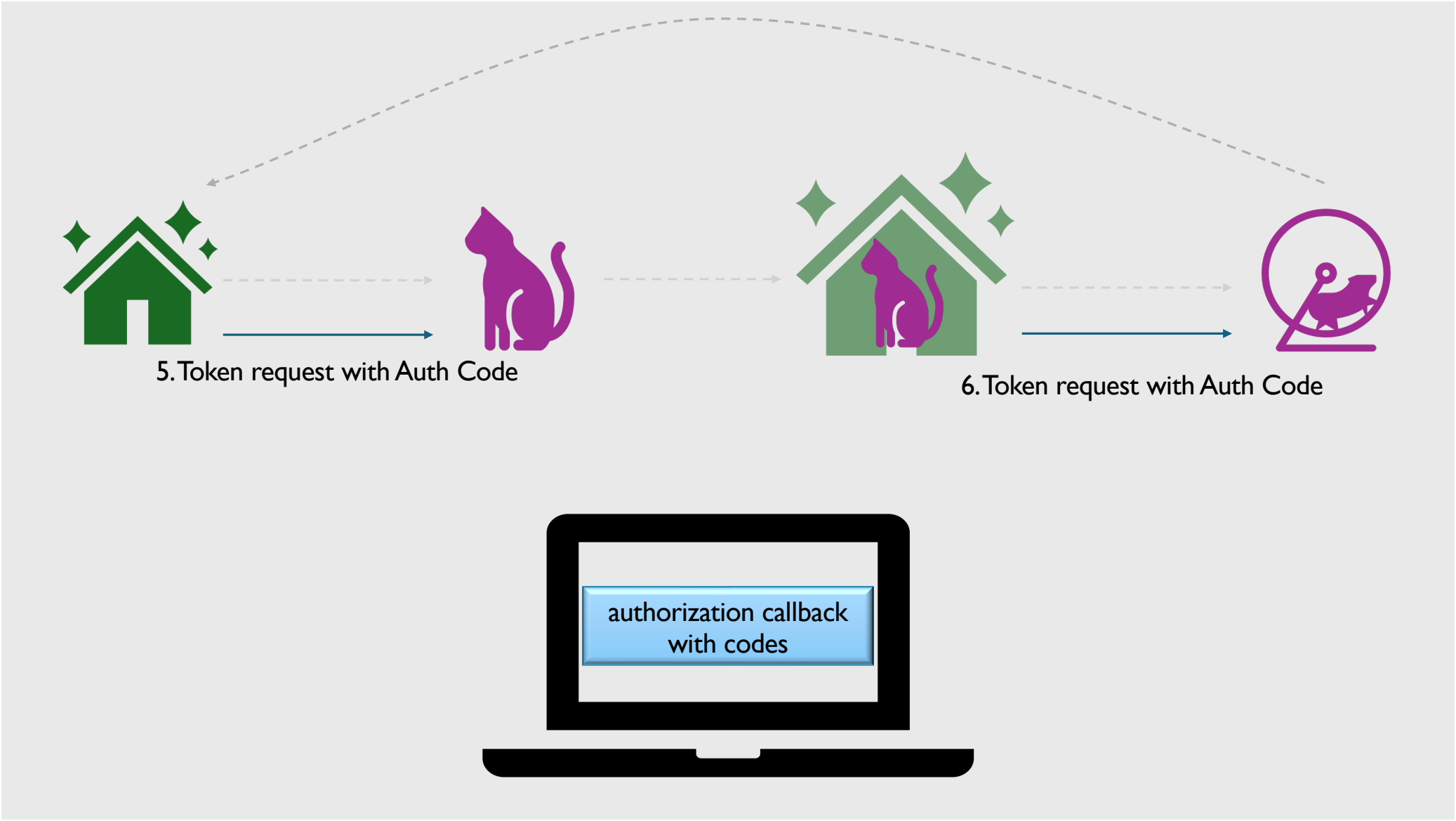


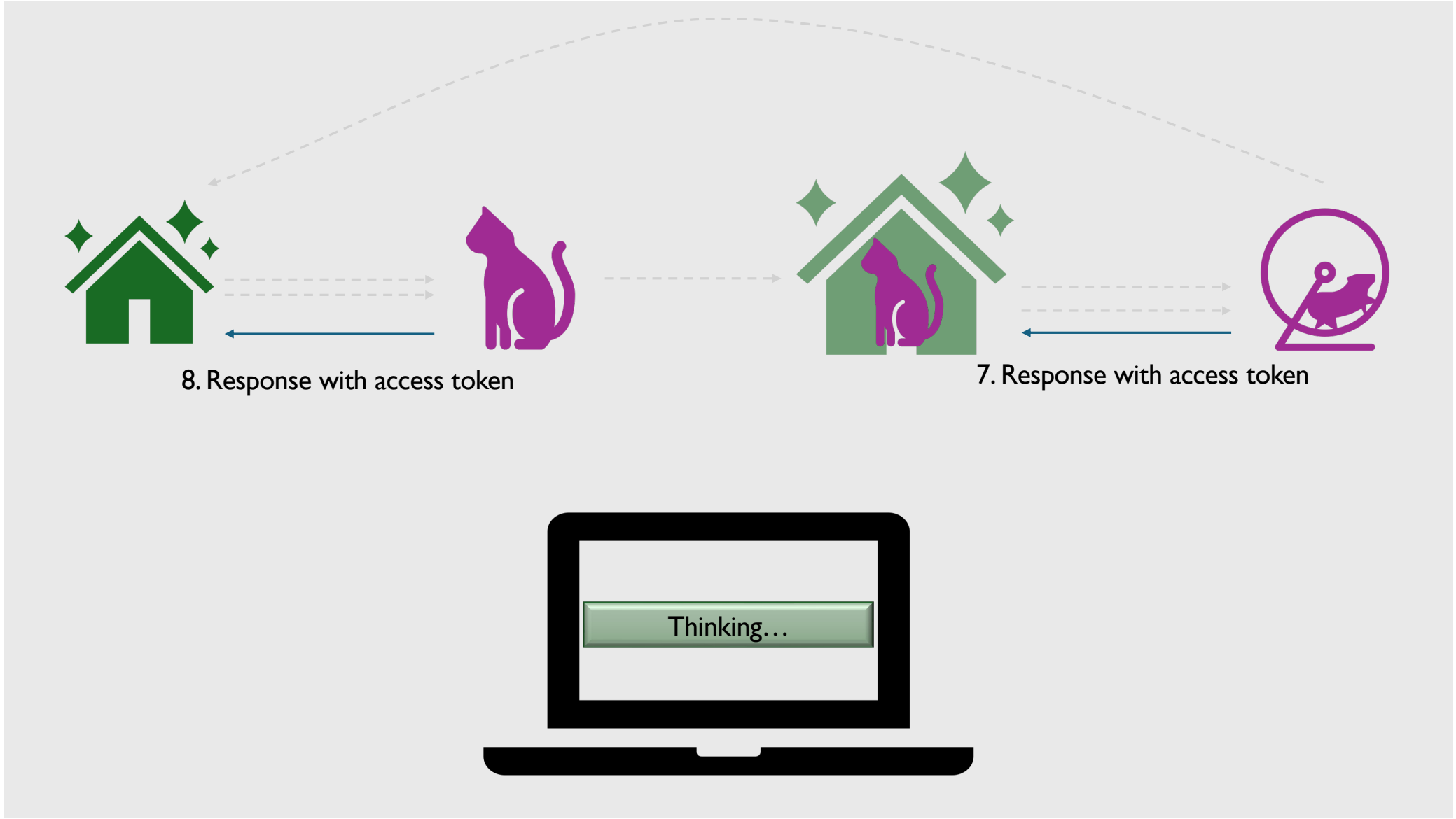


User sees a Guinea Pig authorization

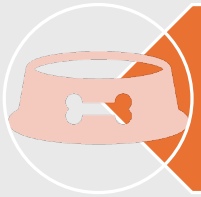
Guinea Pig gets a request that looks like it's coming from SmartHome


4. Auth code to client redirect_URI
https://smarthome.com/
redirect


5. Token request with Auth Code
6. Token request with Auth Code
authorization callback
with codes

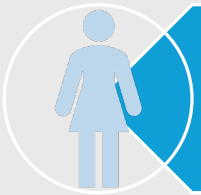

8. Response with access token
7. Response with access token
Thinking…


9. MCP Request with Token to PetMCP
10. MCP Server Response



Grants access with valid guinea pig tokens

Completes the only authorization flow it was listening for



Sees guinea pig data they are expecting

Uses the trusted redirectURI and PKCE for the requesting client

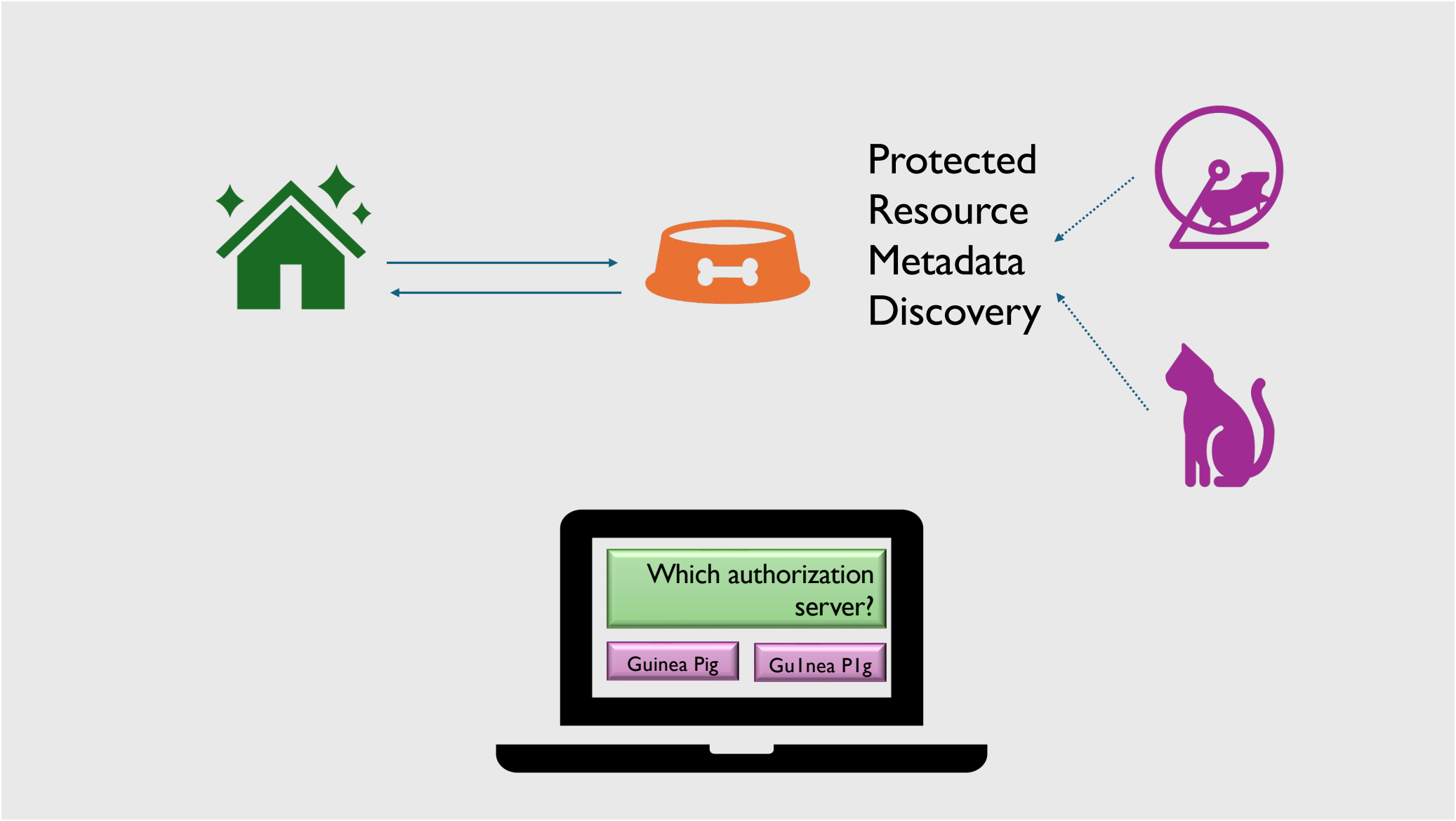
No One Notices The Mix Up

The cat now has:

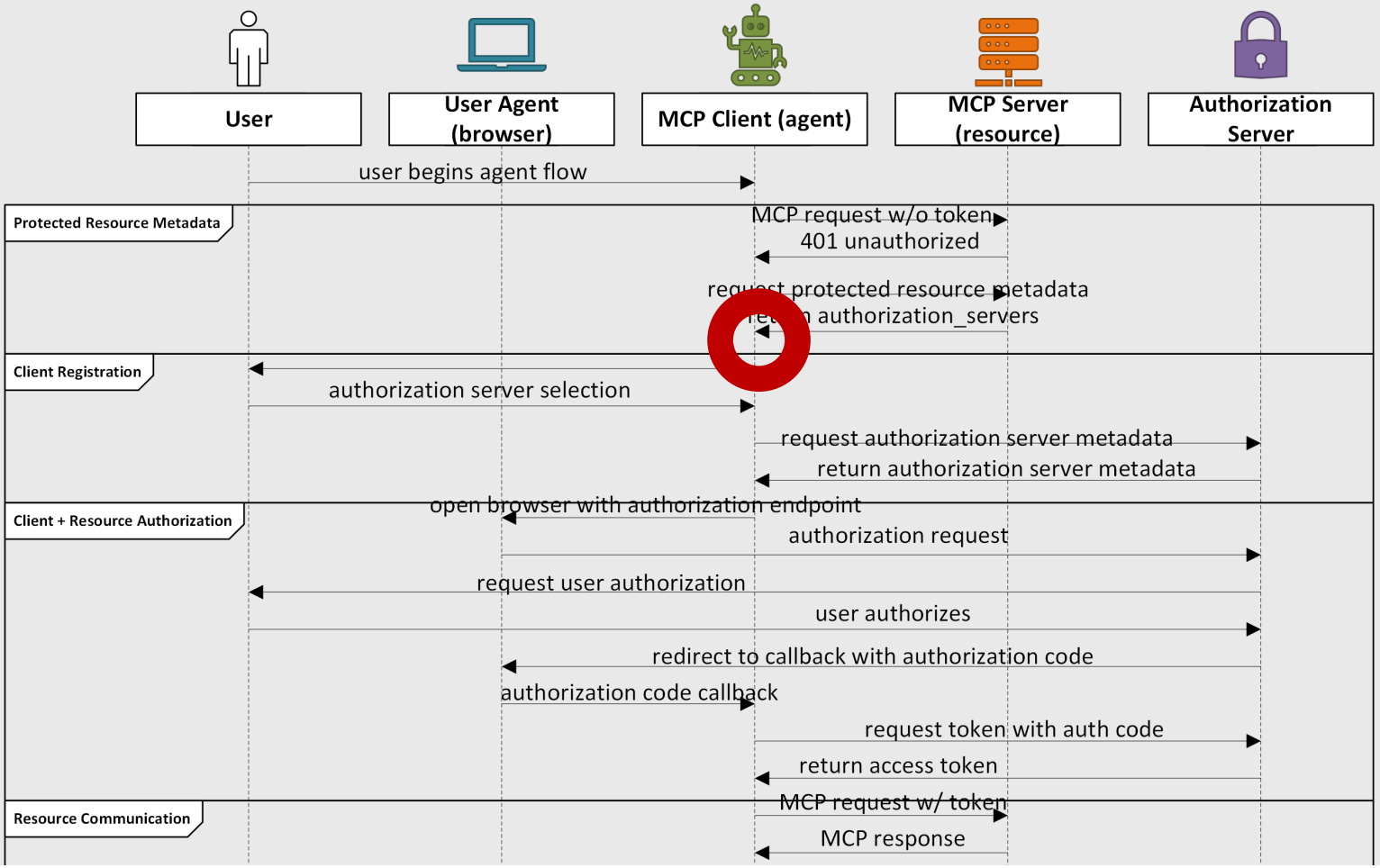
- Guinea Pig Auth Code
- Guinea Pig Access Token

How was a malicious auth server in the trust domain?

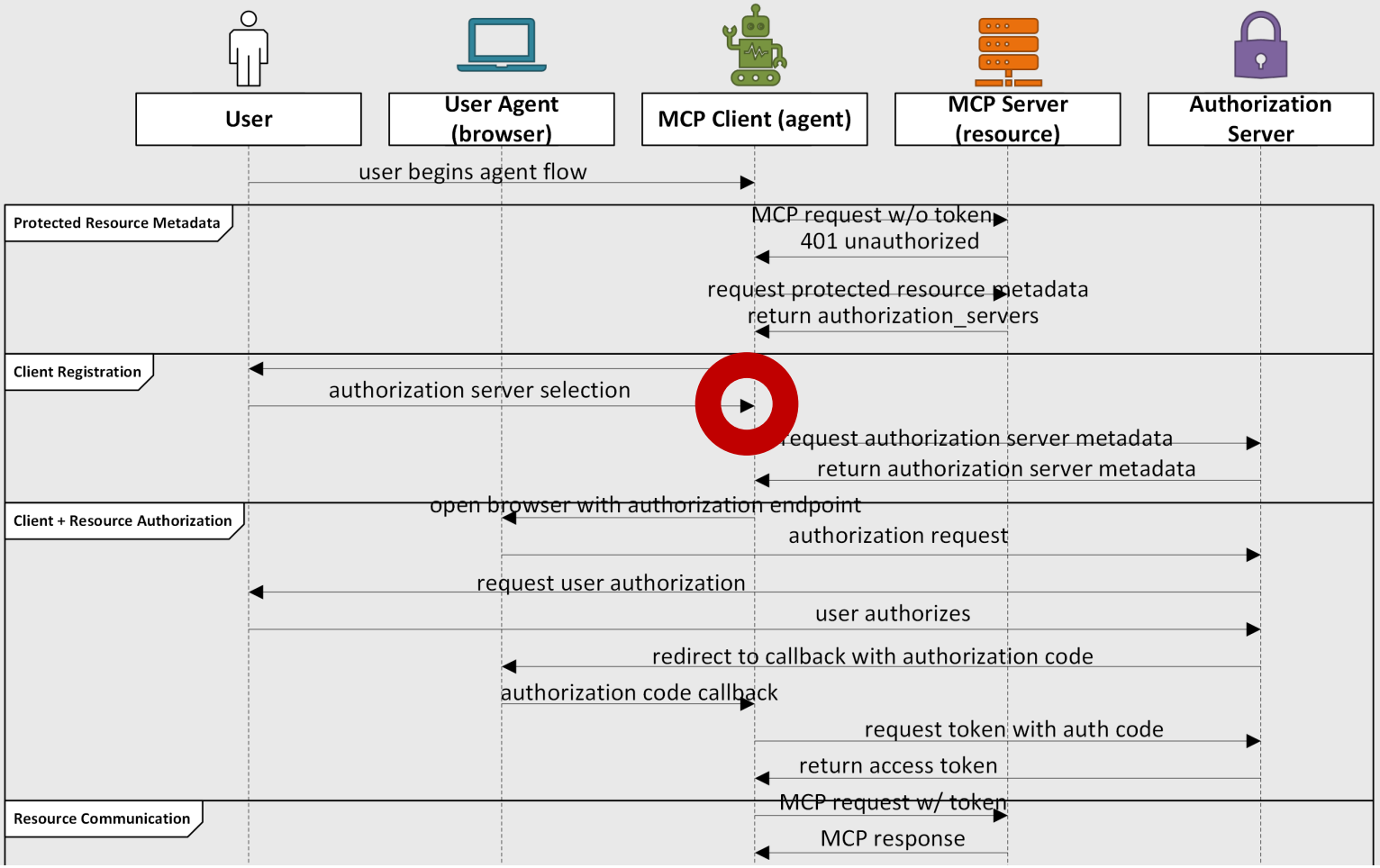
Someone taught the cat how to vibe code


Protected
Resource
Metadata
Discovery
Which authorization server?
Guinea Pig
Guinea Pig

MCP Server Returns Bad Actor

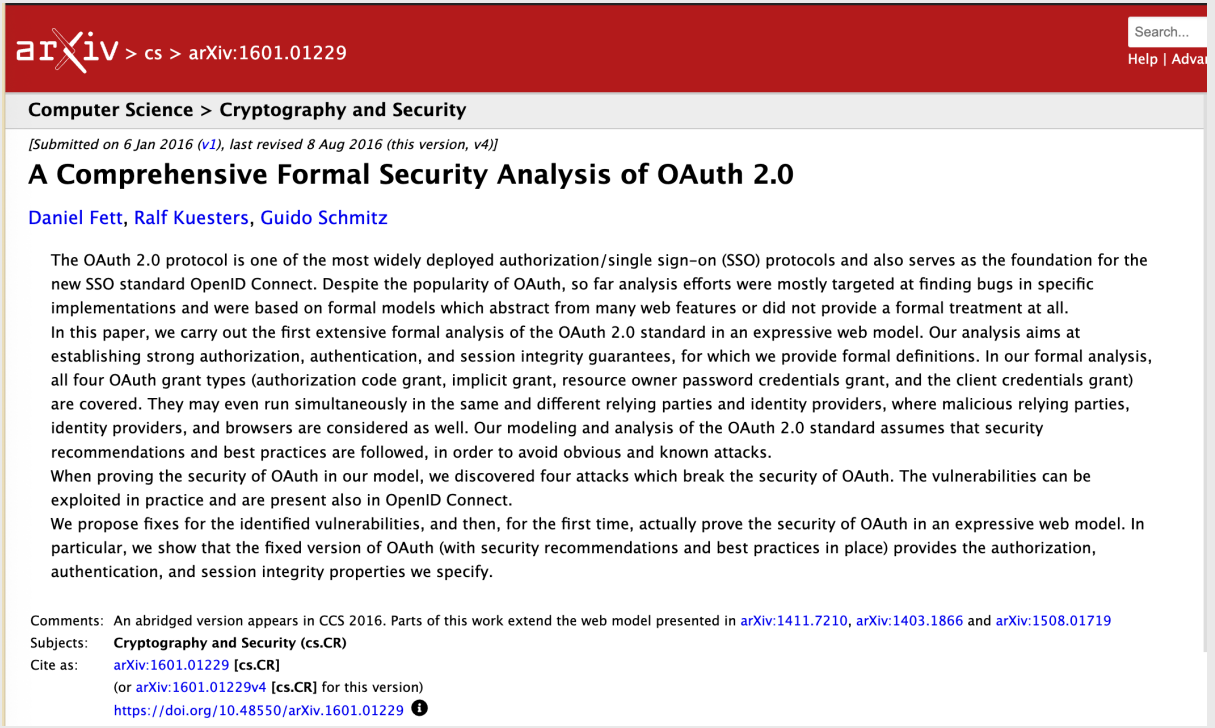


User Selects Malicious Auth Server



Mix Up Attacks in OAuth

Mix up attacks identified as a risk with OAuth 2.0 since 2016



arXiv > cs > arXiv:1601.01229

Search... Help | Adva

Computer Science > Cryptography and Security

[Submitted on 6 Jan 2016 (v1), last revised 8 Aug 2016 (this version, v4)]

A Comprehensive Formal Security Analysis of OAuth 2.0

Daniel Fett, Ralf Kuesters, Guido Schmitz

The OAuth 2.0 protocol is one of the most widely deployed authorization/single sign-on (SSO) protocols and also serves as the foundation for the new SSO standard OpenID Connect. Despite the popularity of OAuth, so far analysis efforts were mostly targeted at finding bugs in specific implementations and were based on formal models which abstract from many web features or did not provide a formal treatment at all.
In this paper, we carry out the first extensive formal analysis of the OAuth 2.0 standard in an expressive web model. Our analysis aims at establishing strong authorization, authentication, and session integrity guarantees, for which we provide formal definitions. In our formal analysis, all four OAuth grant types (authorization code grant, implicit grant, resource owner password credentials grant, and the client credentials grant) are covered. They may even run simultaneously in the same and different relying parties and identity providers, where malicious relying parties, identity providers, and browsers are considered as well. Our modeling and analysis of the OAuth 2.0 standard assumes that security recommendations and best practices are followed, in order to avoid obvious and known attacks.
When proving the security of OAuth in our model, we discovered four attacks which break the security of OAuth. The vulnerabilities can be exploited in practice and are present also in OpenID Connect.
We propose fixes for the identified vulnerabilities, and then, for the first time, actually prove the security of OAuth in an expressive web model. In particular, we show that the fixed version of OAuth (with security recommendations and best practices in place) provides the authorization, authentication, and session integrity properties we specify.

Comments: An abridged version appears in CCS 2016. Parts of this work extend the web model presented in arXiv:1411.7210, arXiv:1403.1866 and arXiv:1508.01719
Subjects: **Cryptography and Security (cs.CR)**
Cite as: arXiv:1601.01229 **[cs.CR]**
(or arXiv:1601.01229v4 **[cs.CR]** for this version)
https://doi.org/10.48550/arXiv.1601.01229

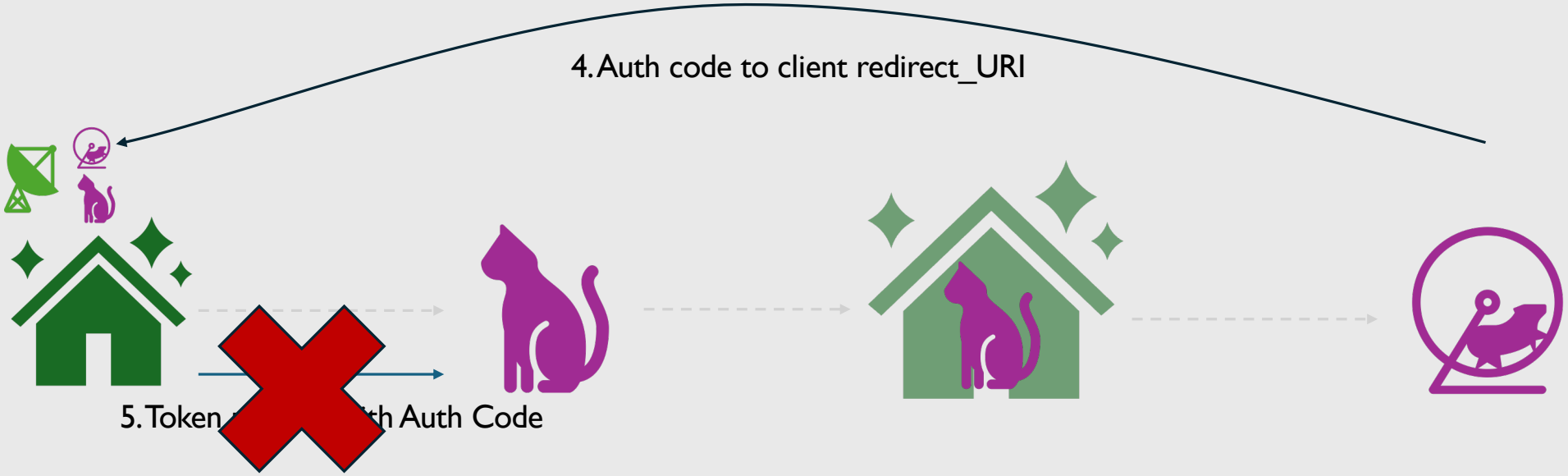

MCP
Mix Up Attacks

Why Mix Up Matters More for MCP

- Scale
  - More clients interacting with more authorization servers
- Vibe Coding
  - AI has increased phishing attempts
- Non-Deterministic
  - Clients and authorization servers often do not know ahead of time which they will interact with

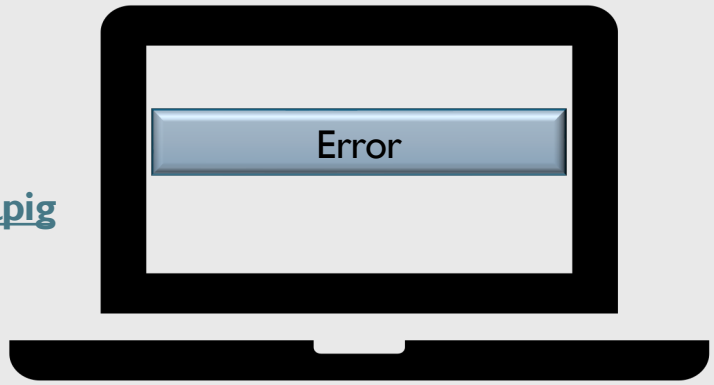
Unique redirect URIs per authorization server

Mitigation 1

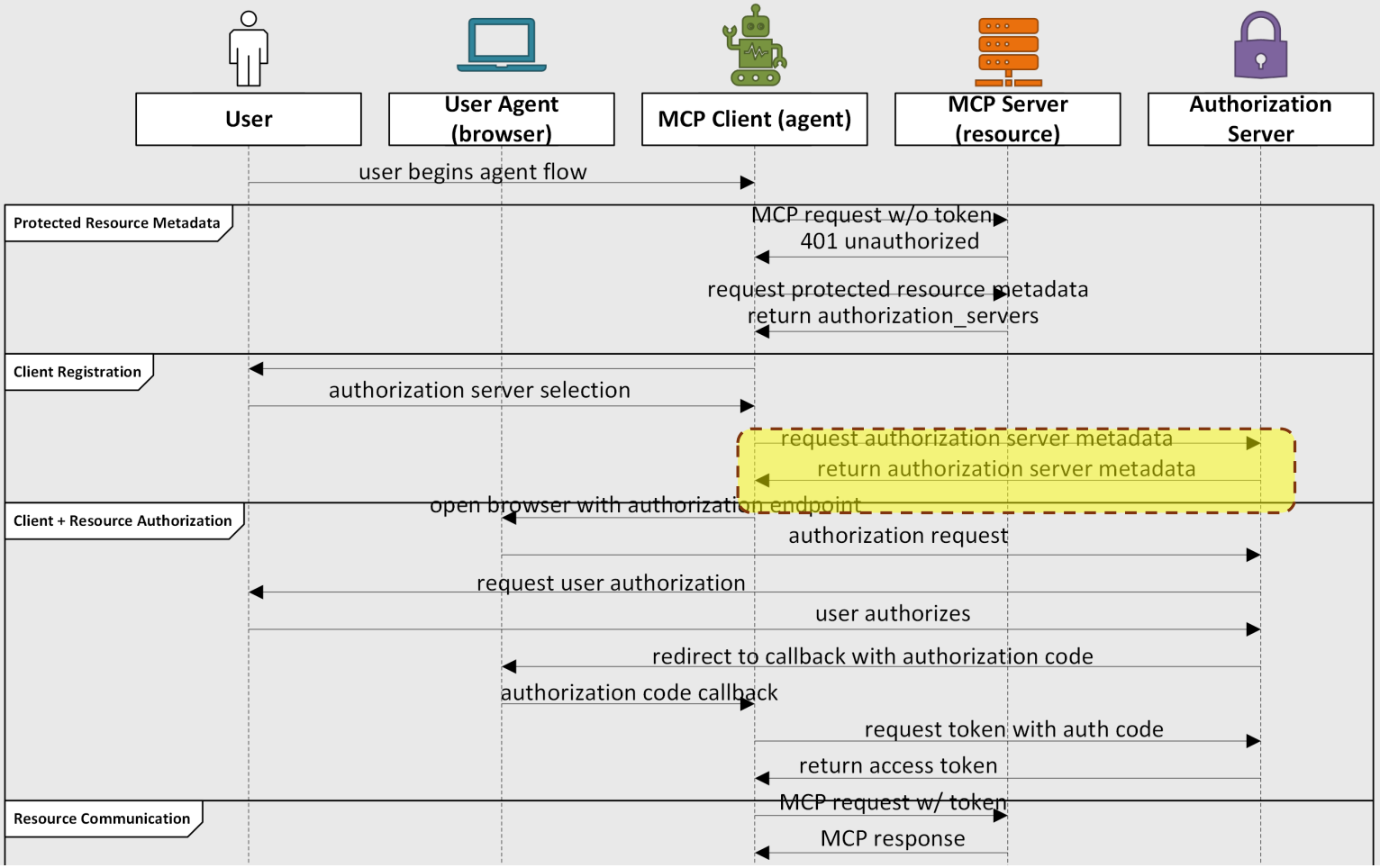


Listening on
https://smarthome.com/redirect_**cat**

Receives auth codes at
https://smarthome.com/redirect_**guineapig**



MCP Authorization



Client Registration in MCP

- Pre-Registration
  - Easy to set up unique redirect_URI
  - Does not allow for non-deterministic agentic connections
- Dynamic Client Registration
  - Could work, other issues with DCR registering client instances
  - Deprioritized in MCP specification to “MAY”
- CIMD
  - Defeats the purpose of having a single source of metadata if customized per auth server

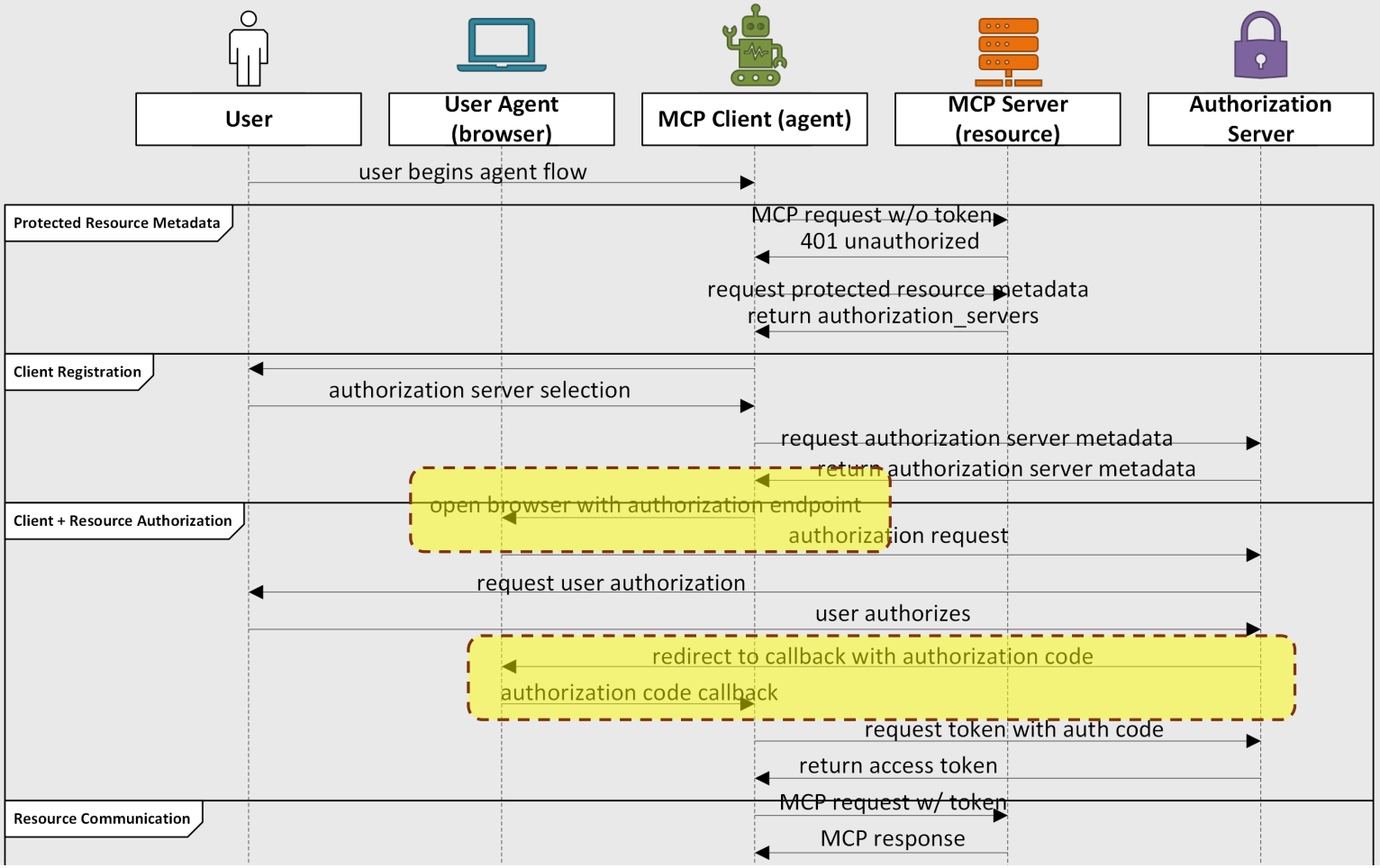
Validate the Issuer Parameter in Authorization Responses

Mitigation 2

How?

- Authorization servers define their issuer in their server metadata
- Each authorization response from the auth server includes the issuer
- Client validates the authorization response's issuer matches the issuer the client sent the authorization request to

MCP Authorization



Issuer (*iss*) Definition

- MUST be a URL with https:// scheme and contain NO query or fragments
  - Example:
    - https://auth.cat.com
    - https://auth.guineapig.com

- RFC8414: *iss* is a required field for OAuth 2.0 Authorization Server Metadata
- RFC9207: introduces *authorization_response_iss_parameter_supported* authorization server metadata parameter

iss not required by default in authorization responses

- RFC9207: OAuth 2.0 Authorization Server Issuer Identification
  - Requires *iss* specifically for Mix Up Attacks
- RFC9700: Best Current Practice for OAuth 2.0 Security
  - Describes the attack, but requires the auth server to know the client supports multiple auth server with a single redirectURI
- OAuth 2.1?
  - https://datatracker.ietf.org/doc/draft-ietf-oauth-v2-1/
  - Open issue and discussion to require *iss* in authorization responses

DRAFT

SEP-2468: Recommend Issuer (iss) in MCP Auth Responses

Draft change to the MCP specification

Thank you to @max-stytch, @0xbrainkid, @aaronpk @SamMorrowDrums

Changes for MCP Specification

MCP Clients MUST

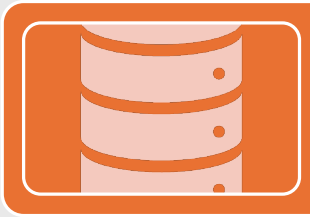
- When fetching auth server metadata, check *authorization_response_iss_parameter_supported*
- If TRUE,
  - Store the state of the issuer for authorization request
  - MUST validate the issuer parameter matches in authorization response

Auth Servers SHOULD

and eventually MUST

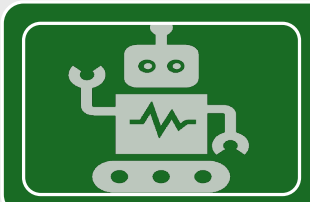
- Set *authorization_response_iss_parameter_supported* server metadata to TRUE
- Send *iss* parameter with authorization responses

Call to Action



MCP Resources

- Trust carefully your authorization servers



MCP Clients

- If you receive an issuer parameter, validate it

MCP SDKs

- Add support for easy adoption

Authorization Server

- Send the issuer parameter in auth responses

Protect the guinea pigs.

Use the issuer parameter.

Thank you!
Connect at /EmLauber