OAuth 2.0

Muhammad **Rizwan** Asghar
The University of Auckland
September 17, 2015

For template of slides,
thanks to kingsoftstore.com

Overview of OAuth 2.0

- An open standard for authorisation
- Evolved from OAuth
- Not backward compatible
- Created in late 2006
- OAuth 2.0 was published as RFC 6749 in October 2012

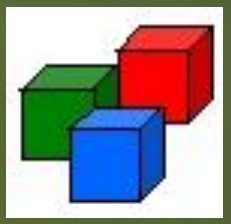
Why OAuth 2.0

- OAuth 2.0 is better than OAuth 1.0 due to
  - Clear separation of roles
  - Simplicity
  - Support of a variety of use cases
  - Addressing native applications

Basic Purpose

- Enabling third-party applications
  - To obtain limited access
  - To protected resources
  - On behalf of a Resource Owner
  - Or on its own behalf

Roles in OAuth 2.0



- Resource Owner
  - Grants access to protected resources
- Resource Server
  - Hosts protected resources
- Client
  - Requests access to protected resources
- Authorisation Server
  - Issues Access Tokens to the Client

Roles in Detail

- The Authorisation Server may be the same server as the Resource Server
- A single Authorisation Server may issue Access Tokens accepted by multiple Resource Servers

OAuth 2.0 Flow

Client

1 – Authorisation Request

2 – Authorisation Grant

3 – Authorisation Grant

4 – Access Token

5 – Access Token

6 – Protected Resource

Resource Owner

Google

Authorisation Server

Picasa

Resource Server

Authorisation Grant

- A credential representing authorisation by the Resource Owner
- To access protected resources
- Used by the Client to obtain an Access Token

Access Token

- Credentials used to access protected resources
- Tokens represent specific scopes and durations of access
  - Granted by the Resource Owner
  - Enforced by the Resource Server and Authorisation Server

Refresh Token

- Credentials used to obtain Access Tokens
- Used when the current Access Token expires
- It is optional

OAuth 2.0 Flow: Refresh Token

Client

1 – Authorisation Grant

2 – Access Token & Refresh Token

3 – Access Token

4 – Protected Resource

5 – Access Token

6 – Invalid Token Error

Picasa

Resource Server

7 – Refresh Token

8 – Access Token & Optional Refresh Token

Google

Authorisation Server

Security Requirements

- It requires the Transport Layer Security (TLS) mechanism for ensuring
  - Confidentiality
  - Integrity
  - Prevention of replay attack

Client Registration

- First, the Client registers with the Authorisation Server
- When registering a Client, a Client Developer specifies
  - Client Type
  - …

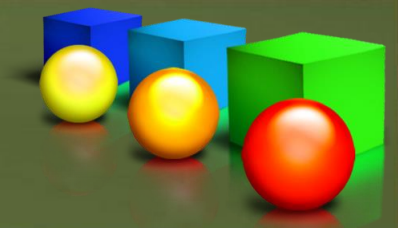
Client Credentials

- Client Identifier
  - It is unique but not secret
- Client Secret
  - Password, private/public key pair
  - Only for Confidential Clients (see next slide)

Client Types

- Confidential
  - Clients maintaining the confidentiality of their credentials
  - Capable of secure client authentication
- Public
  - Clients incapable of maintaining the confidentiality of their credentials
  - Incapable of secure client authentication

Client Profiles



- Web application
  - Confidential Client
- User-agent-based application
  - Public Client
- Native application
  - Public Client

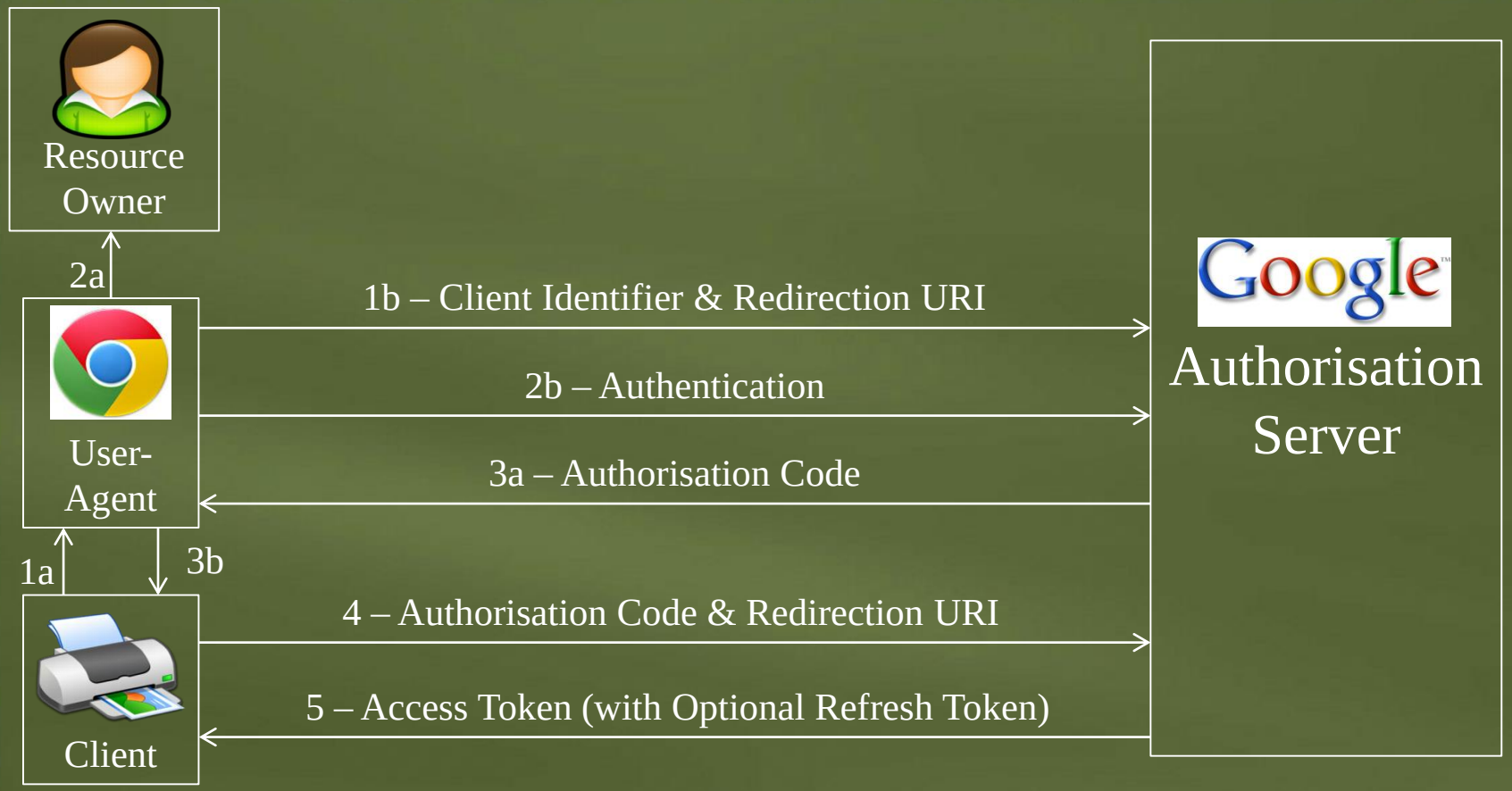
Grant Types

- Authorisation code
- Implicit
- Resource Owner password credentials
- Client credentials
- Extensibility mechanism for defining additional types

Authorisation Code

- Used to obtain both Access Tokens and Refresh Tokens
- Optimised for Confidential Clients
- Client interacts with User Agent of the Resource Owner

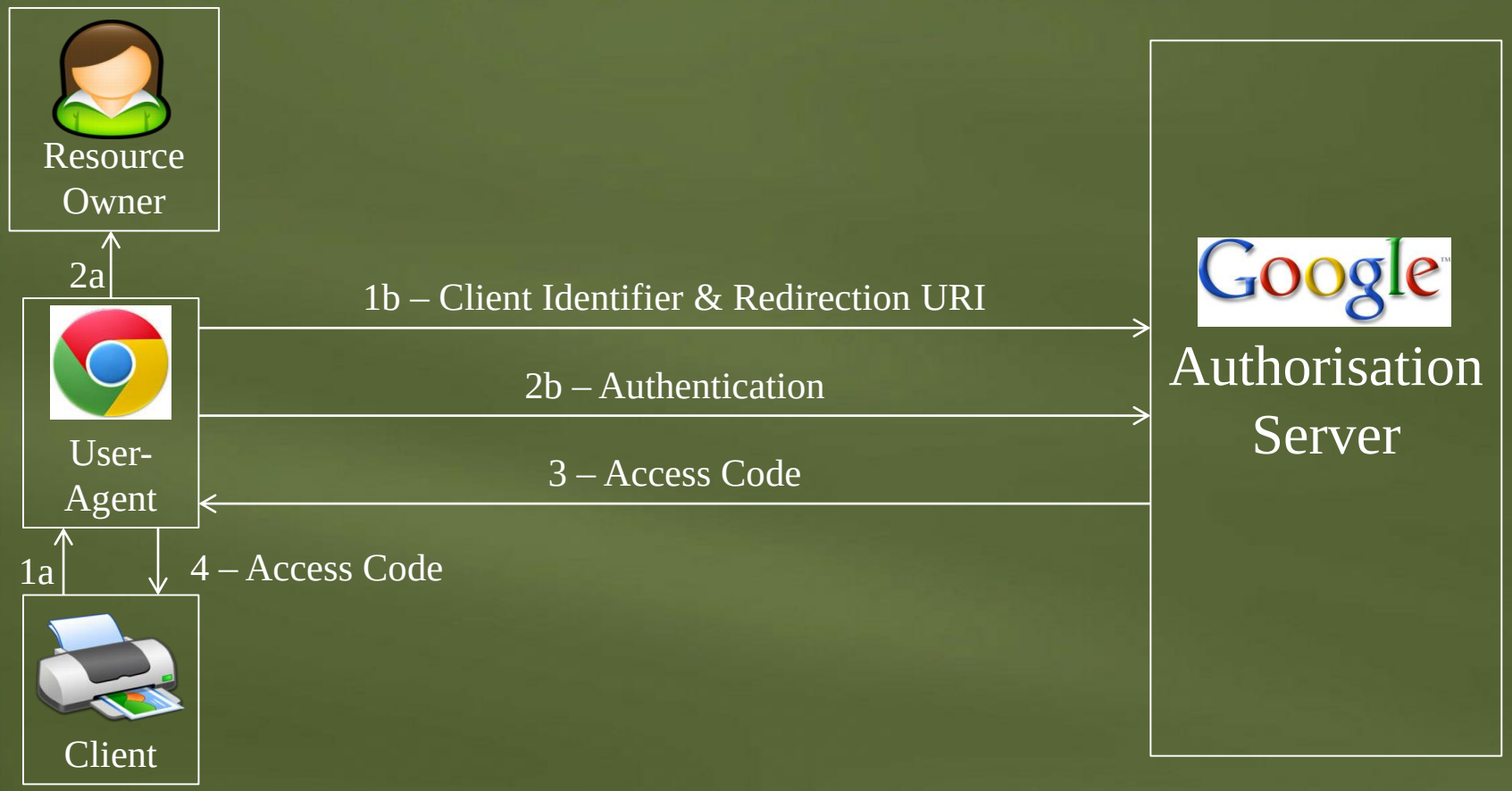
OAuth 2.0 Flow: Authorisation Code



Implicit

- Used to obtain Access Tokens
- It does not support the issuance of Refresh Tokens
- Optimised for Public Clients
- Client interacts with User Agent of the Resource Owner
- A Client receives the Access Token as the result of the Authorisation Request

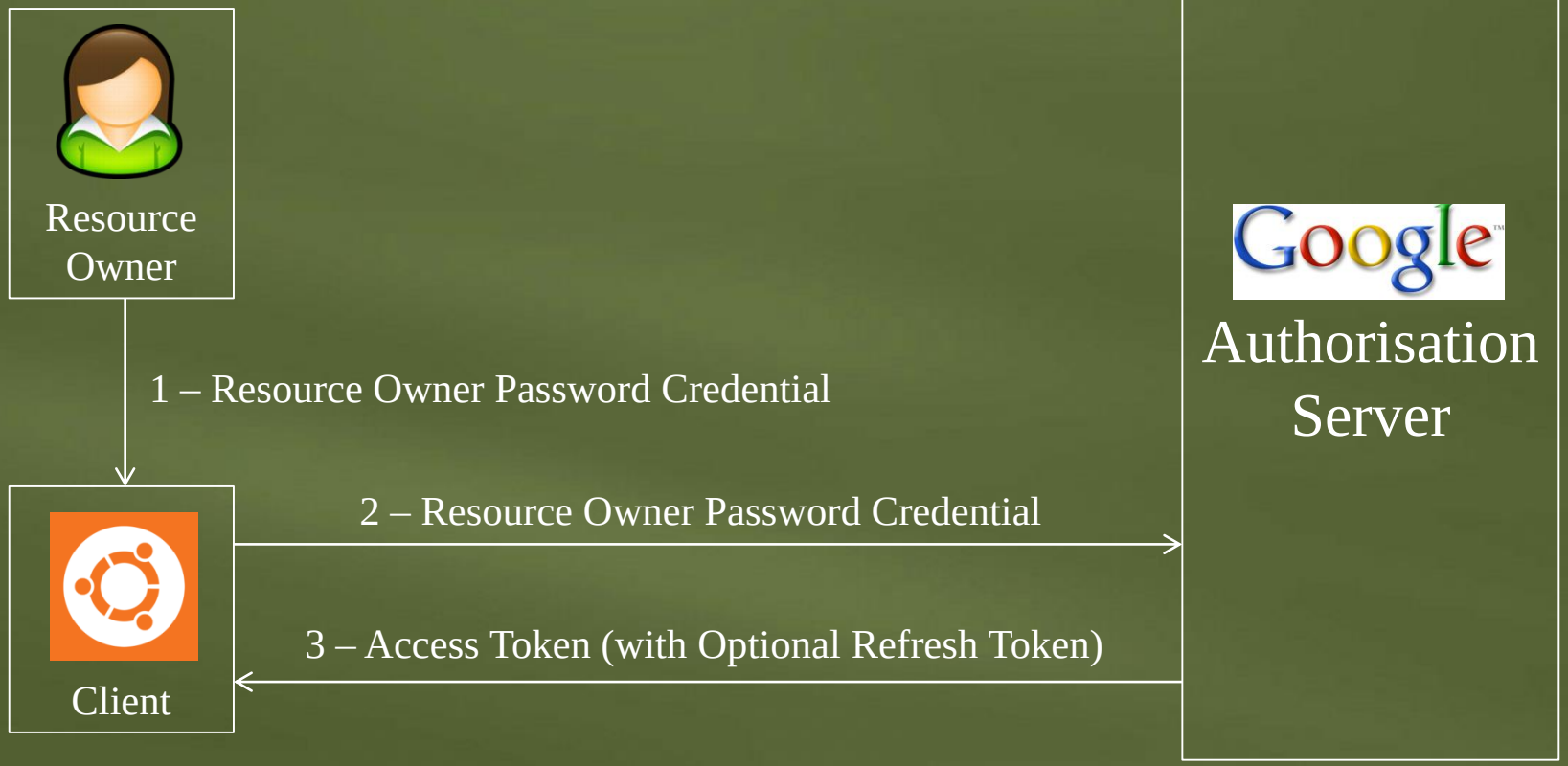
OAuth 2.0 Flow: Implicit



Resource Owner Password Credential

- Suitable when the Resource Owner has a trust relationship with the Client
- Examples
  - Operating system
  - Highly privileged application

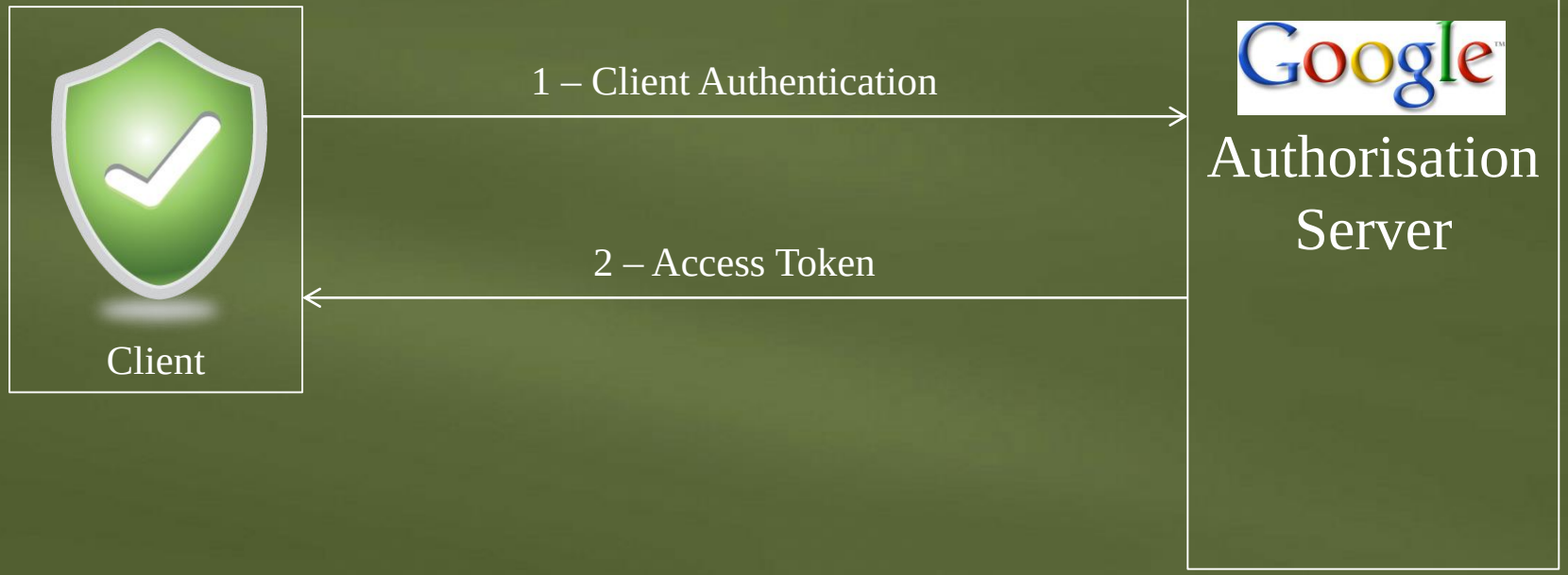
OAuth 2.0 Flow: Resource Owner Password Credential



Client Credentials

- A Client can request an Access Token using only Client Credentials
- Only used by Confidential Clients

OAuth 2.0 Flow: Client Credentials



Access Token Response

- Access Token
- Expiry
- Refresh Token
- Scope

Refreshing Access Tokens

- Client makes a refresh request
  - Grant type
    - Must be ‘Refresh Token’
  - Refresh Token
  - Scope

Accessing Resources

- Client interacts with the Resource Server
- Client accesses protected Resources by presenting Access Tokens
- The Resource Server validates
  - Validity of the Access Token
  - Scope

Attacks and Countermeasures

- Obtaining Client Secrets
  - Revoke Client Secrets
- Obtaining Refresh Tokens
  - Revoke Refresh Tokens
- Obtaining Access Tokens
  - Keep lifetime short

Limitations

- No backward compatibility
- It relies on SSL/TLS for ensuring
  - Confidentiality
  - Integrity
  - Prevention of replay attack
- Phishing attack

Limitations (2)

- Privacy issues
  - Servers will know more about Resource Owners and Clients
- Denial-of-Service (DoS) attack
  - Effect on Clients and Servers

OAuth Service Providers

- Flickr 

- Google App Engine 
- Netflix 

- Yahoo 
- …

OAuth 2.0 Service Providers

- Amazon
- AOL
- Facebook
- GitHub
- Google
- Microsoft
- Paypal

Service Providers Supporting Both

- Dropbox
- LinkedIn
- Twitter
- …

Summary

- OAuth 2.0 is evolved from OAuth
- Provides clear separation of roles
- A variety of use cases
  - Native applications
- Enterprises offer OAuth, OAuth 2.0 or both


Q&A

References

- OAuth 2.0, http://oauth.net/2/
- The OAuth 2.0 Authorization Framework, http://tools.ietf.org/html/rfc6749
- OAuth 2.0 Threat Model and Security Considerations, http://tools.ietf.org/html/rfc6819