Federated Identity

Opportunities & Risks

thinktecture

TROOPERS10

IT-Security Conference & Workshops

Dominick Baier

- **Former ERNW employee**
- **Security consultant at thinktecture**
  - application security in distributed systems
  - identity management
  - mostly Windows & .NET

- **http://www.leastprivilege.com**
- **dominick.baier@thinktecture.com**

Objectives

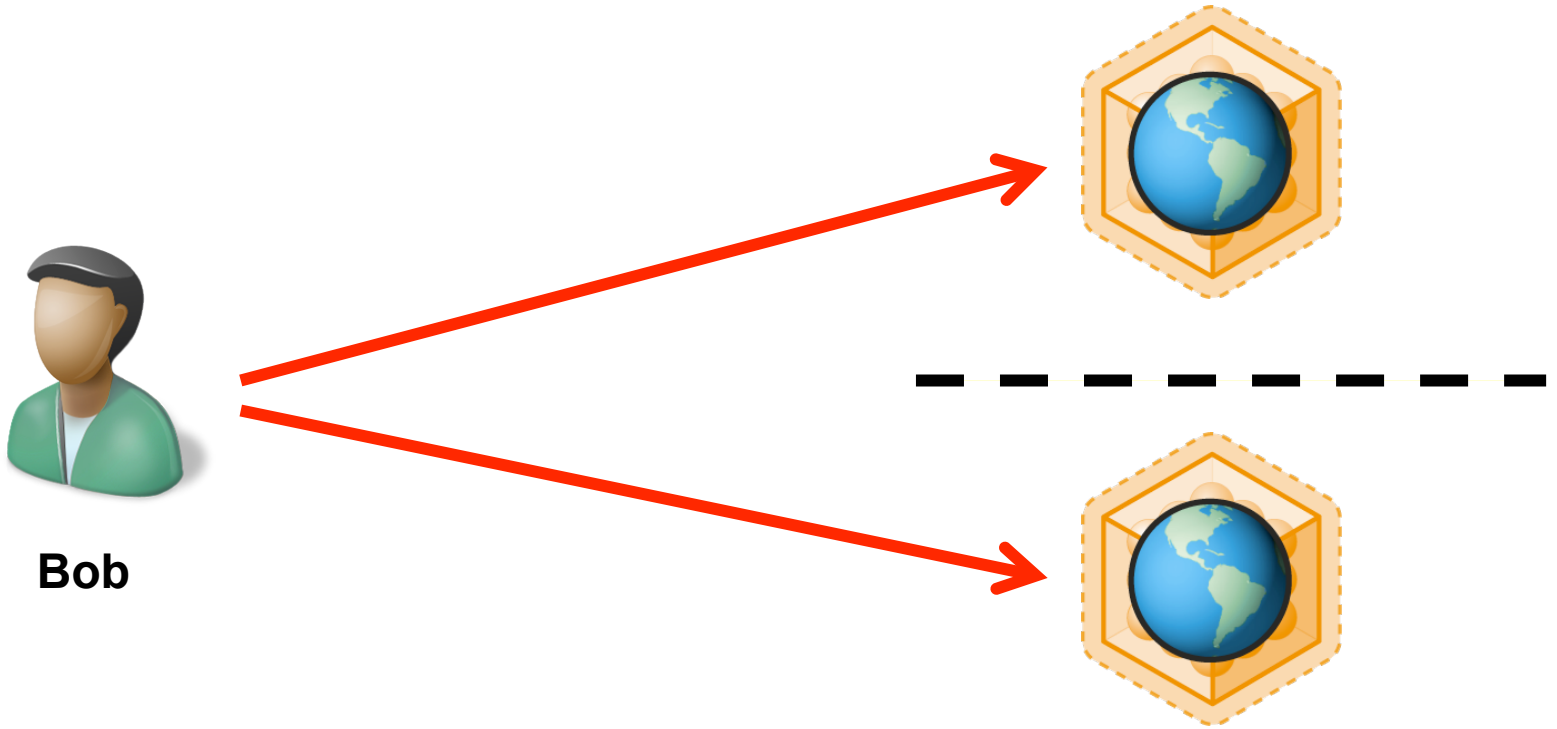
- What is federated identity?
- Why would I care?
- Anatomy of federated identity
- Enterprise & consumer usage
- Security considerations

What is identity?

- **Too many definitions**
  - what you say about your self
  - what others say about yourself
- **Technically speaking**
  - proving you are a valid directory entry

What is federated identity?

- **Again many definitions**
  - being able to use your identity in more than one security domain
  - often in single-sign-on style



Where is it used?

- **Enterprise space**
  - connect customers and partners to internal applications
  - connect employees to external applications
  - internal federation between branches/domains
- **Consumer space**
  - re-use accounts between various internet applications
  - more for leisure type of apps – less e-commerce

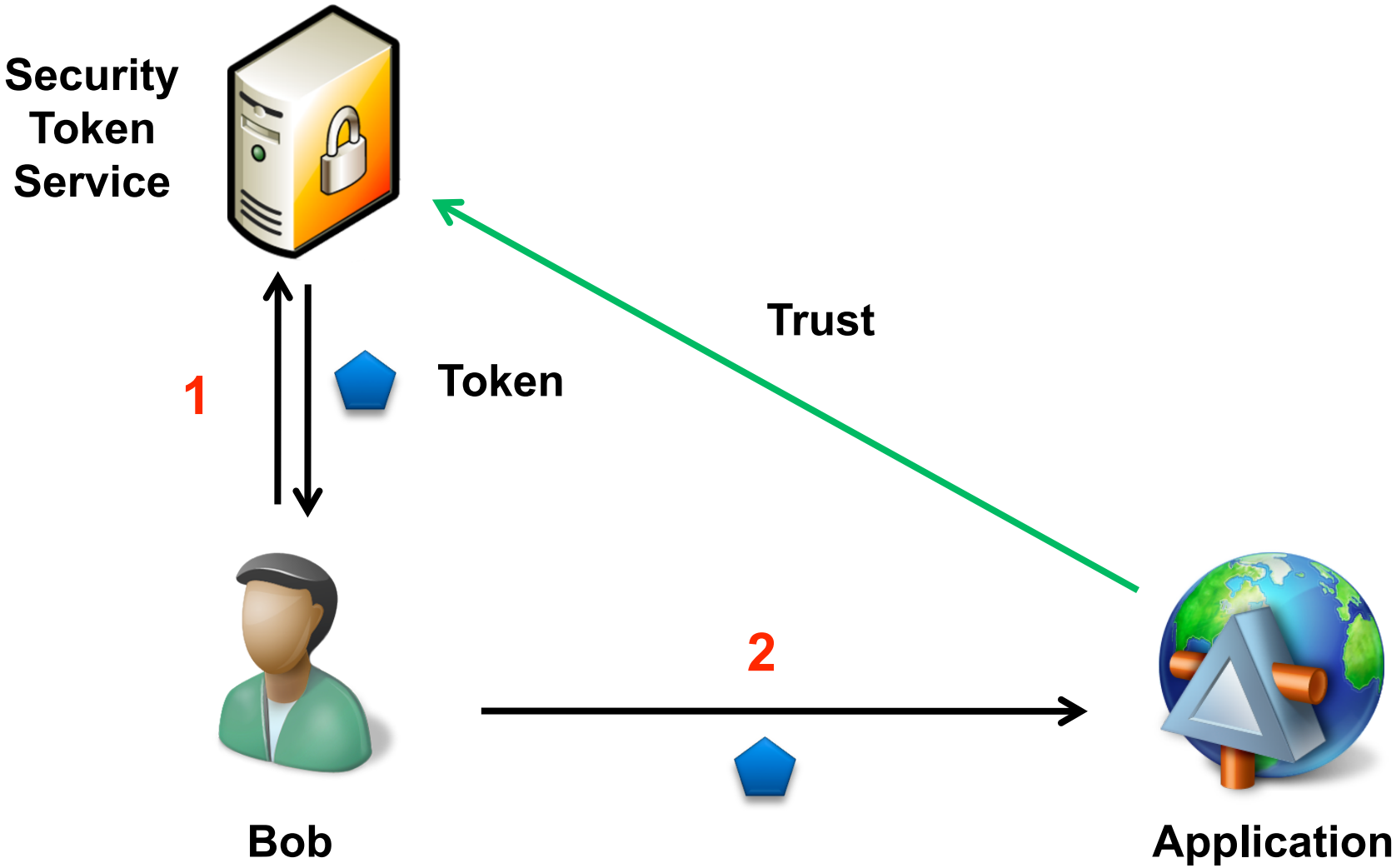
- **ISV space**
  - somewhere in-between
  - depends on to whom they want to sell their software to

Federated authentication

- **Toughest problem to solve**
  - authentication across security boundaries
  - without replicating accounts
- **Various requirements**
  - providing a stable (scoped) user identifier
  - provide additional information for authorization & personalization

- **Bunch of protocols out there**
  - WS-Federation, WS-Trust, SAML (Enterprise)
  - OpenID, OAuth/WRAP (Consumer)

Federated authentication



Enterprise space

- **SAML 2.0 Protocols (SUN, RSA, IBM)**
  - SAML 2.0 token type
  - various profiles (web apps & services)

- **WS-* and friends (Microsoft, IBM, VeriSign)**
  - WS-Federation Passive Profile (web applications)
  - WS-Trust, WS-Security (web services)
  - token agnostic, but typically SAML 1.1/2.0

- **Both rely on a batch of sub-specifications**
  - HTTP, XML Encryption, XML Signatures etc…

SAML Assertion

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
  <saml:AttributeStatement>
    <saml:Attribute AttributeName="userid"
                                AttributeNamespace="http://...">
      <saml:AttributeValue>42</saml:AttributeValue>
    </saml:Attribute>

    <saml:Attribute AttributeName="name"
                                AttributeNamespace="http://... ">
      <saml:AttributeValue>Dominick</saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute AttributeName="department"
                                AttributeNamespace="http://... ">
      <saml:AttributeValue>Research</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>

  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#" />
</saml:Assertion>
```

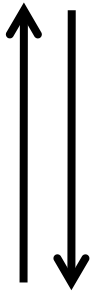
Passive token request (WS-Federation)

Identity Provider

login?wa=wsignin1.0&wtrealm=address_of_rp

```
<form method="POST" action="http://server/app/">
  <input name="wresult" value="<saml:Assertion…" />
  …
  <script >
      window.setTimeout('document.forms[0].submit()', 0);
  </script>
</form>
```

GET /login



Client

GET /app

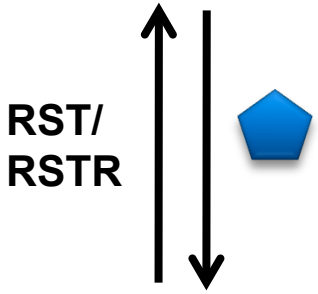
POST /app

Relying Party

SAML Bearer tokens

- **Token provided as-is**
- **Optionally encrypted**
- **Owner of token can authenticate**
  - either legitimate or eavesdropping etc..
- **Replay attack/transport protection important**

Active token request (WS-Trust)

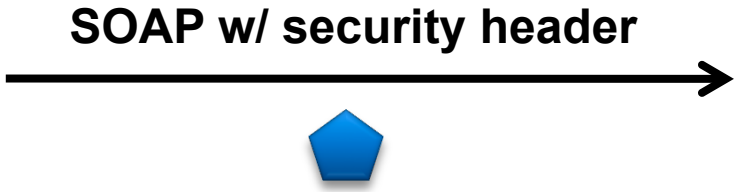



```
<RequestSecurityToken>
  <RequestType>Issue</RequestType>
  <TokenType>SAML#1.1</TokenType>
```

```
<RequestSecurityTokenResponse>
  <saml:Assertion>
    …
  </saml:Assertion>
<RequestSecurityTokenResponse>
```

```
    </EndpointReference>
  </AppliesTo>
<RequestSecurityToken>
```

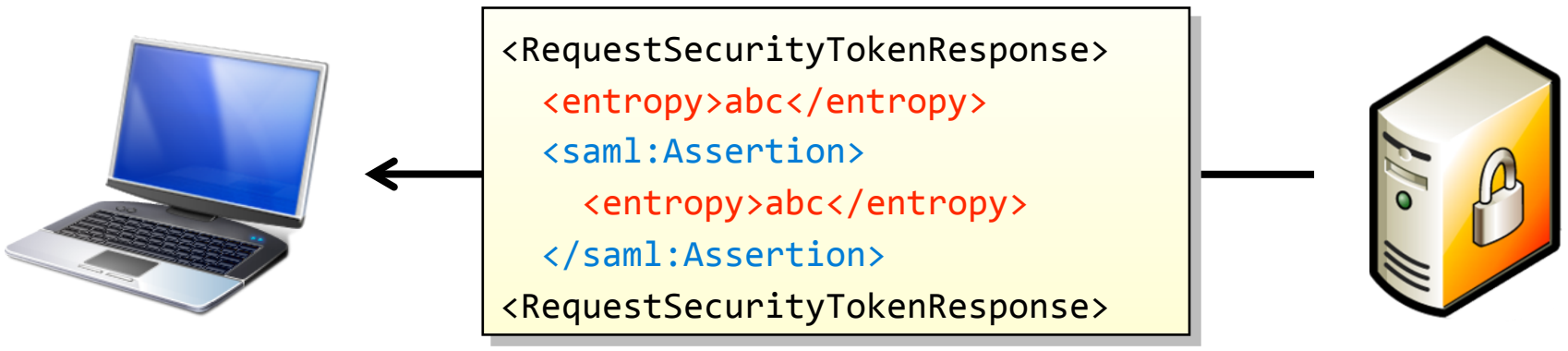
Client



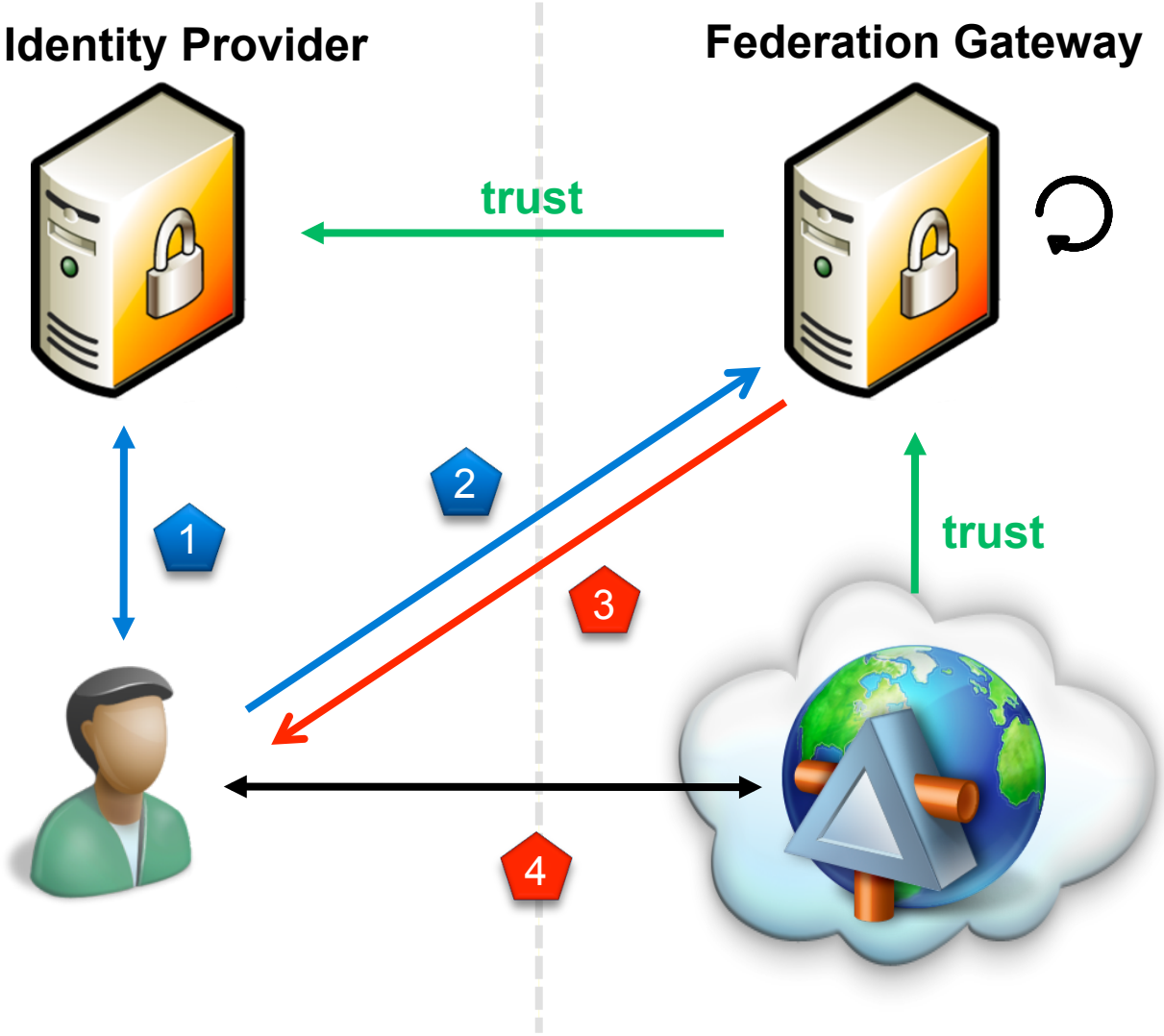
Relying Party

SAML Proof-of-Possession tokens

- **Similar to Kerberos service tickets**
- **Tokens must be encrypted**
- **(Symmetric) key material both embedded in token and in response message**
  - key used to sign message to relying party thus proving to be the original requester

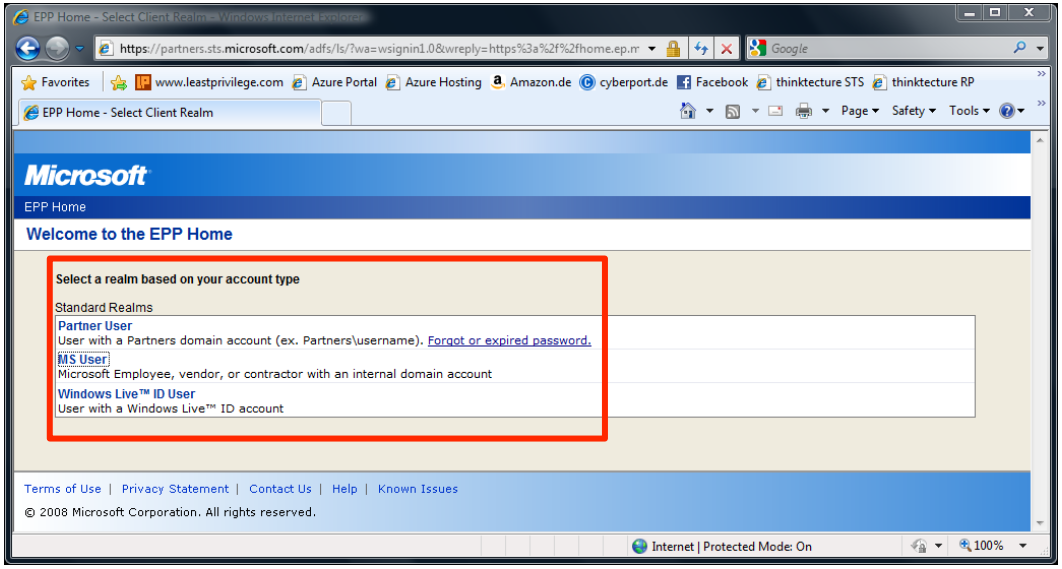


Common scenario



Home realm discovery

- **Common issue in web applications**
  - how does the application know where the user is coming from?
- **Several ways to approach this problem**
  - Resource-STS provides UI
  - home realm encoded in URL
    - https://www.app.com/partner1



Products (excerpt)

- **Security Token Services / Identity Provider**
  - Microsoft Active Directory Federation Services 2.0
  - IBM Tivoli Federation Manager
  - Sun OpenSSO
  - CA SiteMinder
  - Novell Access Manager
- **Relying Party / Service Provider toolkits**
  - Microsoft Windows Identity Foundation (.NET)
  - Bandit (Java)
  - simpleSAML (PHP)

Consumer space

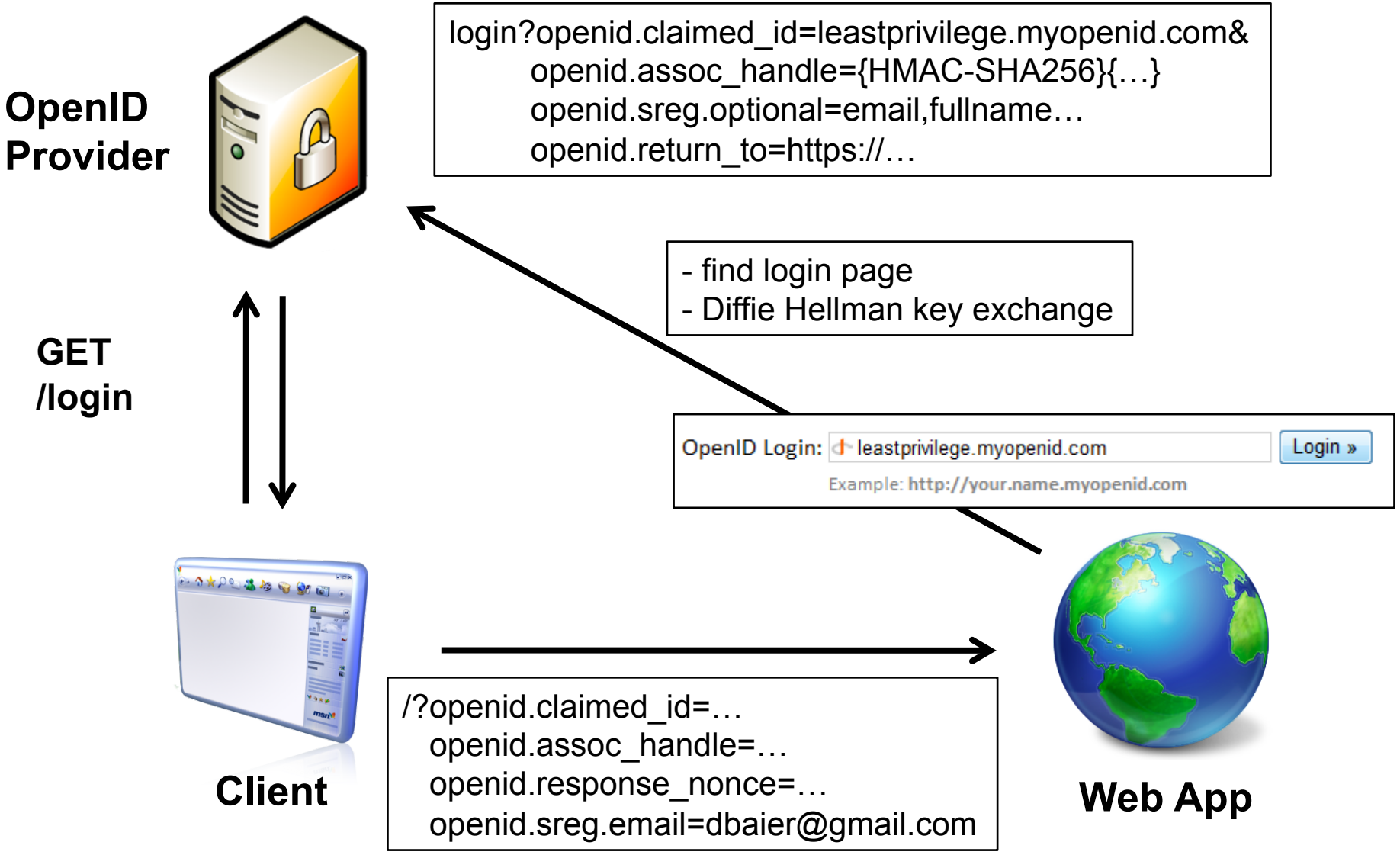
- **OpenID**
  - easy to implement authentication protocol
  - large backing in community
  - plurality of providers/applications by design
  - limited security features in standard profile
  - based on HTTP

- **OAuth/WRAP**
  - mechanism to access protected resources/APIs
  - piggybacks on various authentication mechanisms
  - enables „simple delegation“ scenarios

OpenID

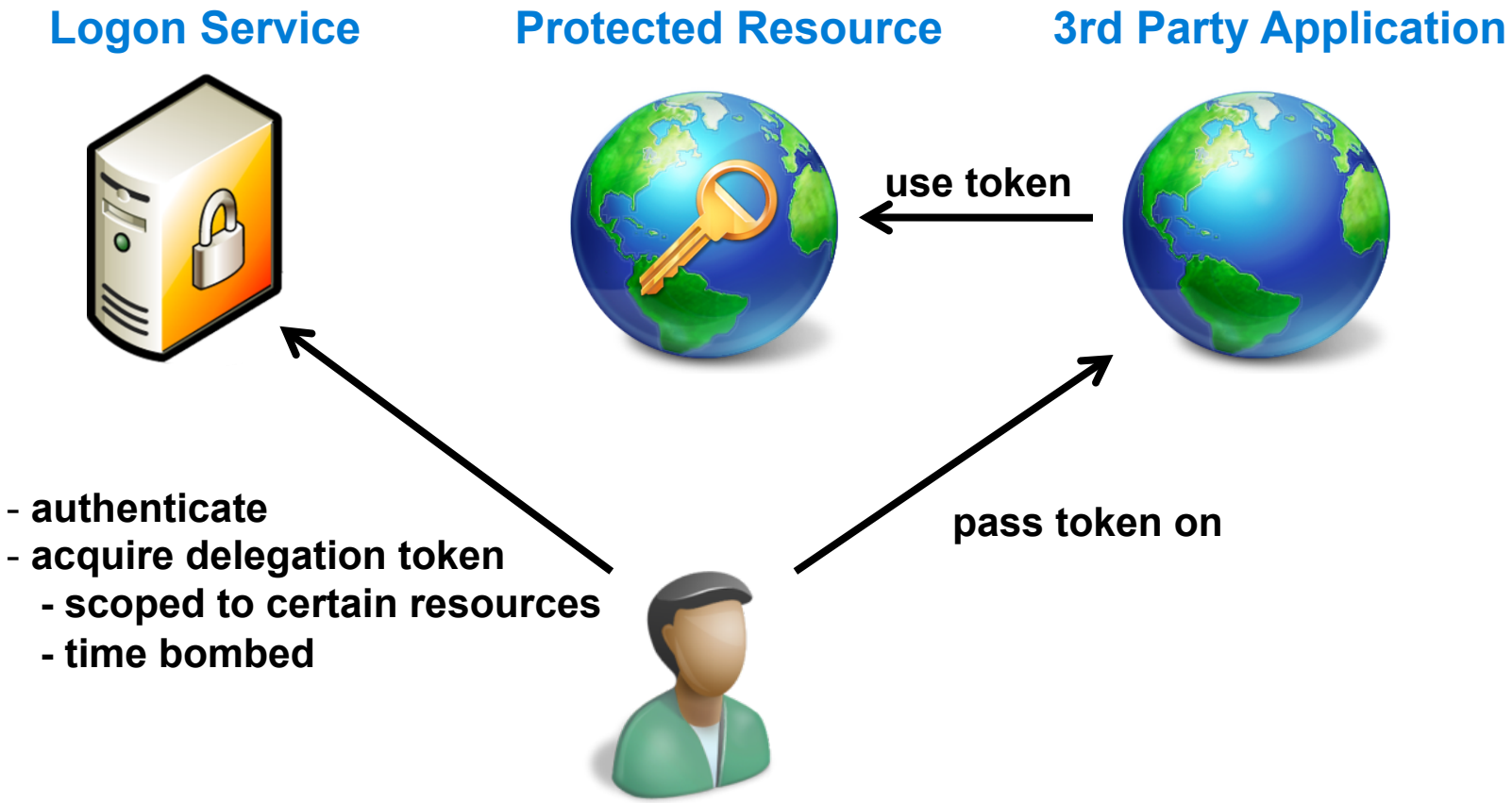
- **Most popular 3rd party authentication mechanism in the consumer space**
  - Google
  - Facebook
  - Yahoo
  - Twitter
  - Flickr
  - MySpace
  - AOL
  - Verisign
  - MyOpenID
- **Approx. one billion user accounts / 50K enabled web sites**

OpenID 2.0 authentication (in its simplest form)



„Simple delegation“

- **Grant access to protected resource „on behalf of“**



Toolkits (excerpt)

- **Plugins for various blog/CMS engines…**
  - Drupal, Wordpress, phpBB
- **DotNetOpenAuth (.NET)**
- **JOpenID (Java)**
- **PHP OpenID**
- **Ruby OpenID**
- **OpenID4Perl**
- **Google AppEngine OpenID (Python)**

Problems with federated identity

Issue - who‘s identity is it & who controls it?

- **Not much of a problem in enterprise space**
  - user‘s identity is owned by the employer anyway
  - typically very tight trust relationships
  - minimum disclosure policy typically already in the company‘s interest
- **Different story in consumer space**
  - federation relationships typically unclear to user
  - too much has happened already
  - users often prefer „manual“ solutions (and isolation)
  - all based on trust – and often there‘s not much of that

Technical issues

- **Protocols are complex**
  - shouldn‘t try implement yourself
  - go with a proven library/product
- **The federated identity is an attractive target**
  - gives access to many resources with a single credential
  - phishing
  - CSRF
- **In most cases, the browser is the driver of the protocol**
  - all known (and unknown) attacks against browsers (or their operators)
  - think SslStrip (additional encryption of token recommended)
  - web services typically don‘t have this issue due to stricter security handling

Summary

- **Federated identity has benefits**
  - reduction of (potentially poor) credentials
  - streamlining of login experience
  - removal of authentication code in applications
  - isolation of complex security related code
  - remove friction in B2B scenarios
  - enabler for the cloud
- **Federated identity has implications**
  - amplification of existing attacks
  - user credentials gain power – users need to be aware of that
  - poor application design may open up even more critical vulnerabilities
  - even when technically sound – users may reject it