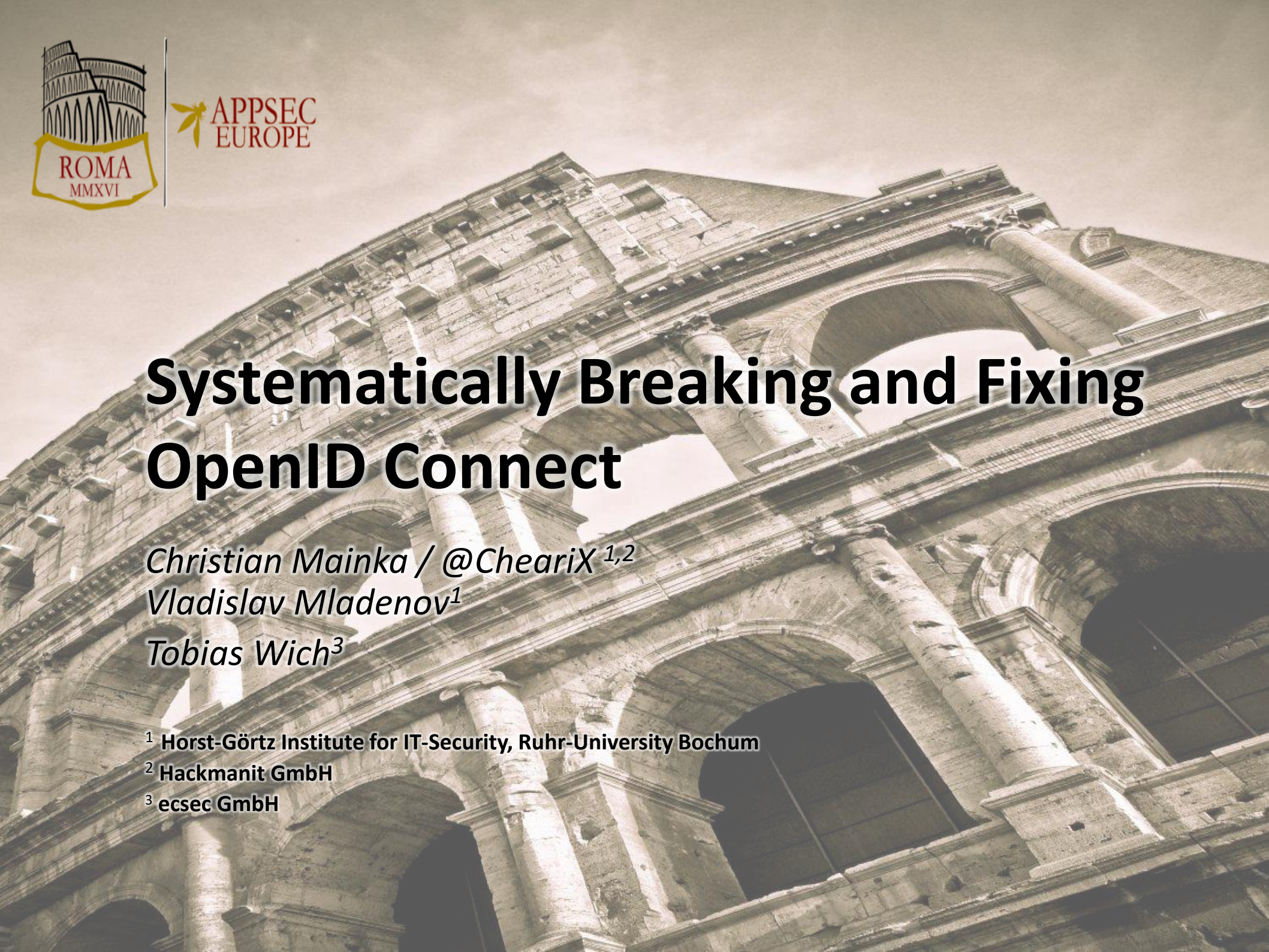


Systematically Breaking and Fixing OpenID Connect

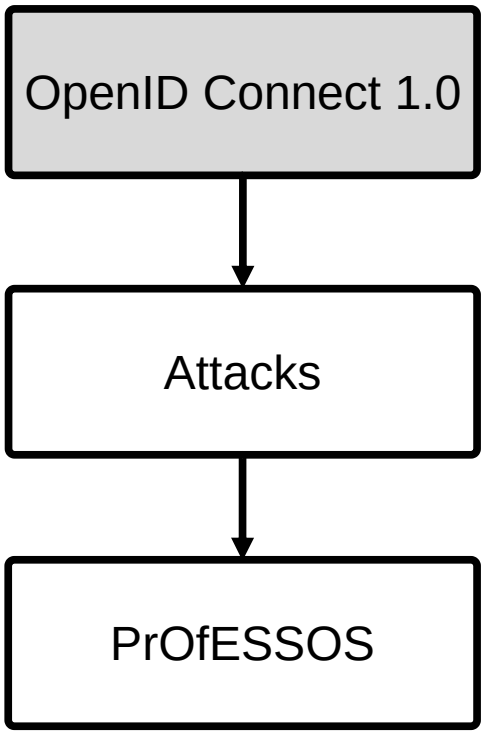
Christian Mainka / @CheariX [1,2]
Vladislav Mladenov[1]
Tobias Wich[3]

[1] Horst-Görtz Institute for IT-Security, Ruhr-University Bochum
[2] Hackmanit GmbH
[3] ecsec GmbH

Research Questions

- (Q1) Are old/known attacks addressed in OIDC?
- (Q2) How secure are officially referenced (certified) libraries?
- (Q3) How can the development of SSO libraries be brought closer to published state-of-the-art regarding security?

On the Security of OpenID Connect



OAuth 2.0 vs. OpenID Connect 1.0

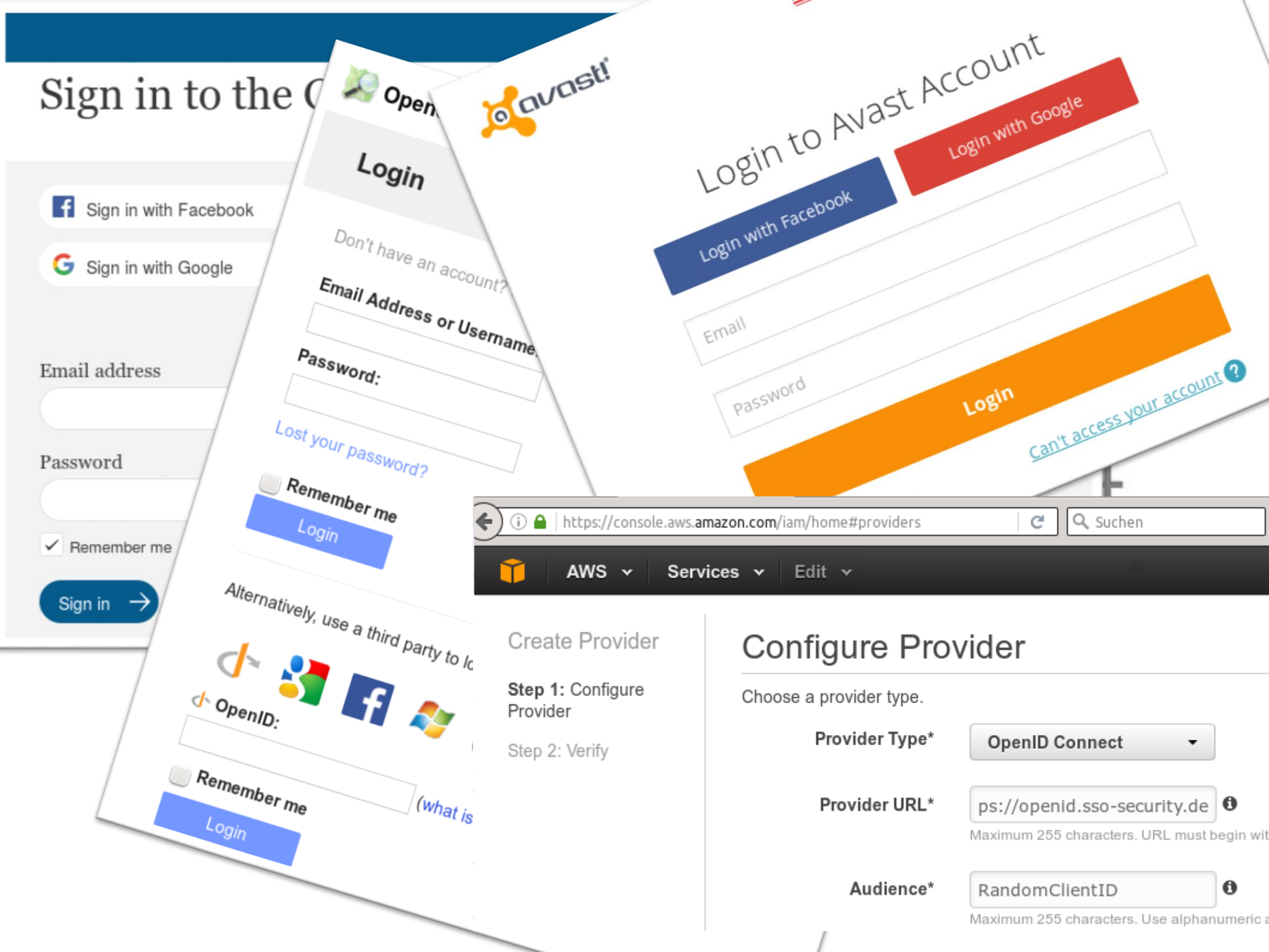
OAuth 2.0

Client
Application

OAuth 2.0 vs. OpenID Connect 1.0

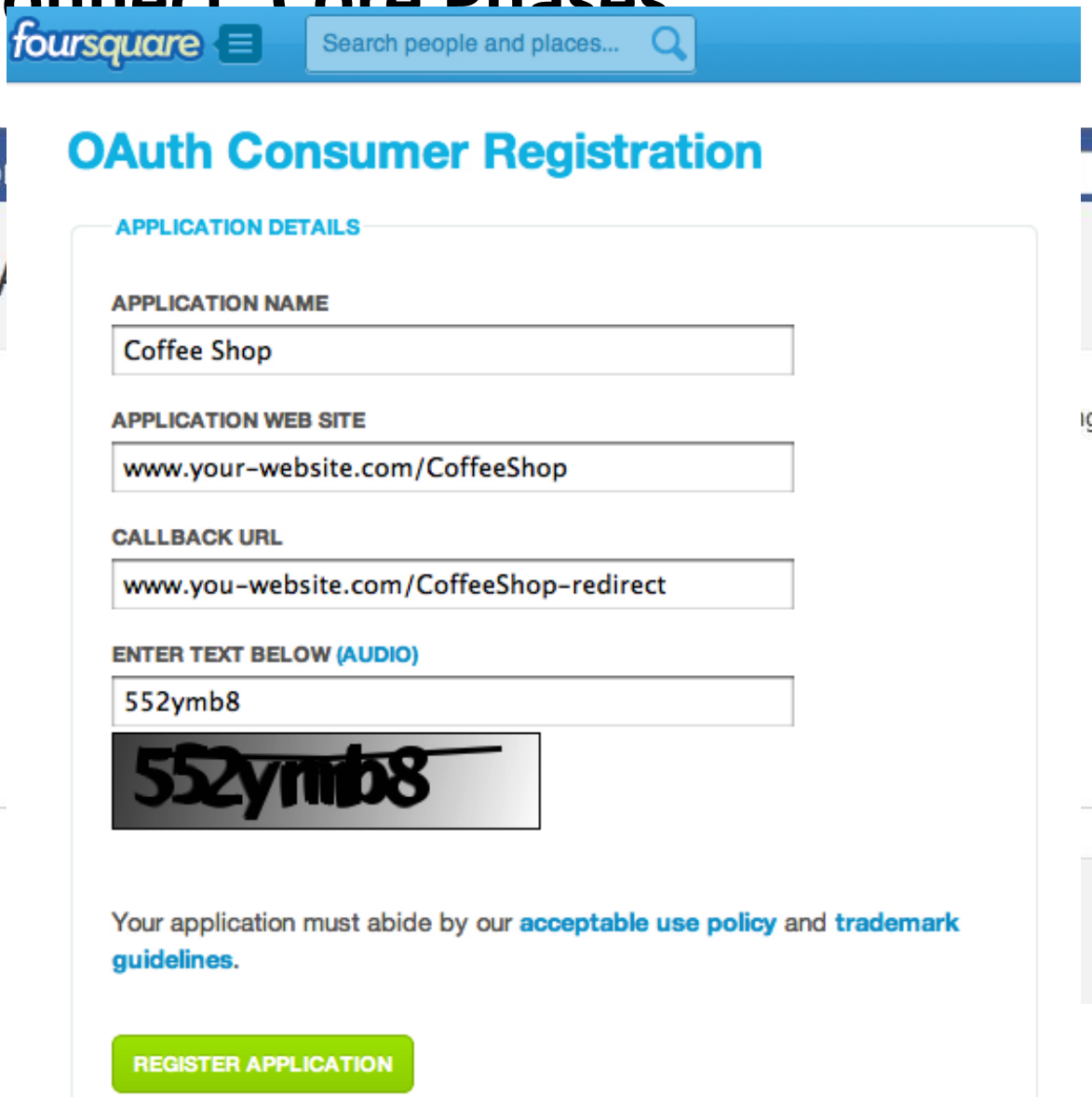
OpenID Connect 1.0

Client
Application

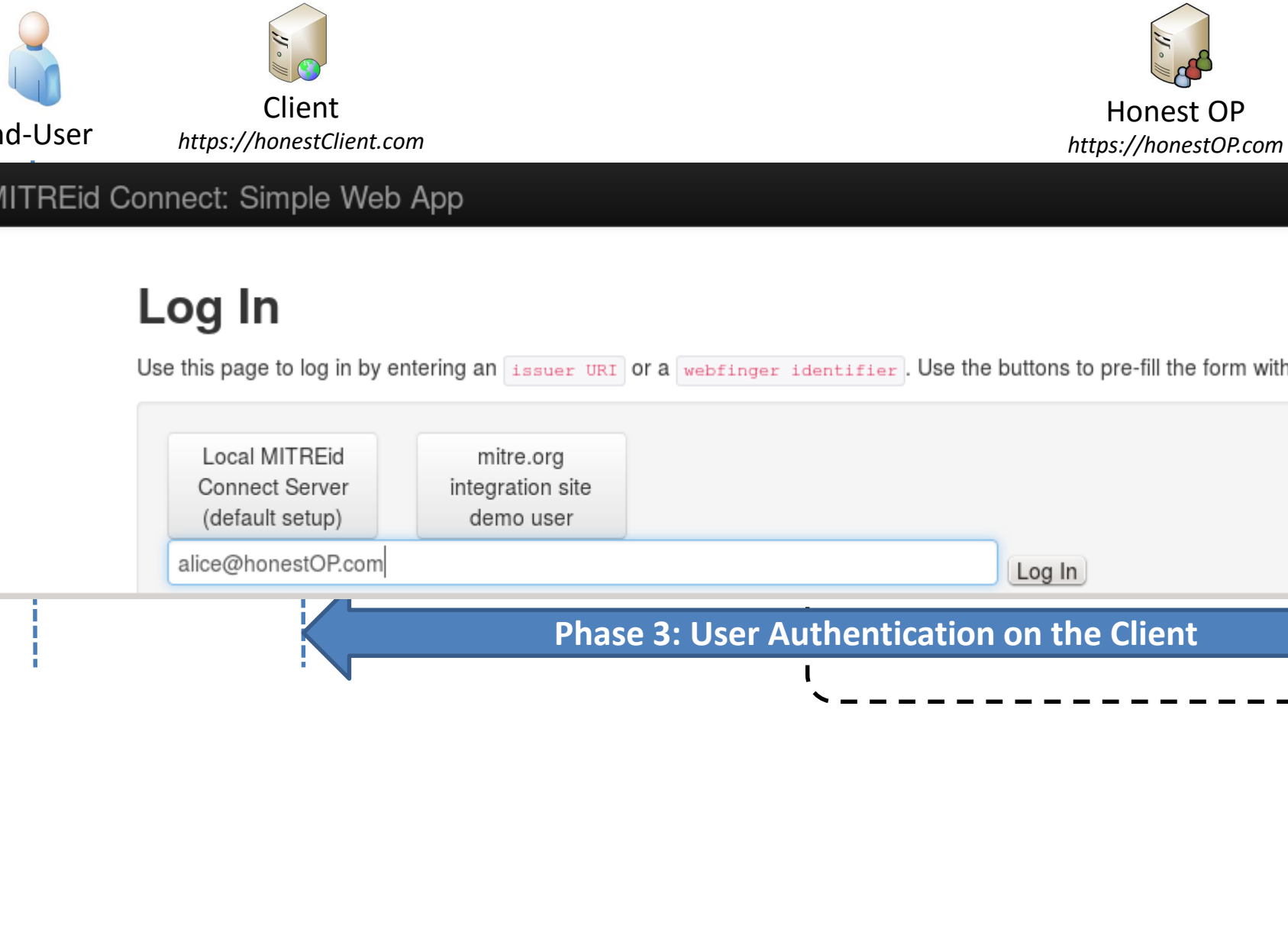

Sign in to the
Sign in with Facebook
Sign in with Google
Email address
Password
Remember me
Sign in
Login
Don't have an account?
Email Address or Username
Password:
Lost your password?
Remember me
Login
Alternatively, use a third party to lo
OpenID:
Remember me
Login
(what is
avast!
Login to Avast Account
Login with Facebook
Login with Google
Email
Password
Login
Can't access your account
https://console.aws.amazon.com/iam/home#providers
Suchen
AWS
Services
Edit
Create Provider
Step 1: Configure Provider
Step 2: Verify
Configure Provider
Choose a provider type.
Provider Type*
OpenID Connect
Provider URL*
ps://openid.sso-security.de
Maximum 255 characters. URL must begin wit
Audience*
RandomClientID
Maximum 255 characters. Use alphanumeric a

OpenID Connect: Core Phases

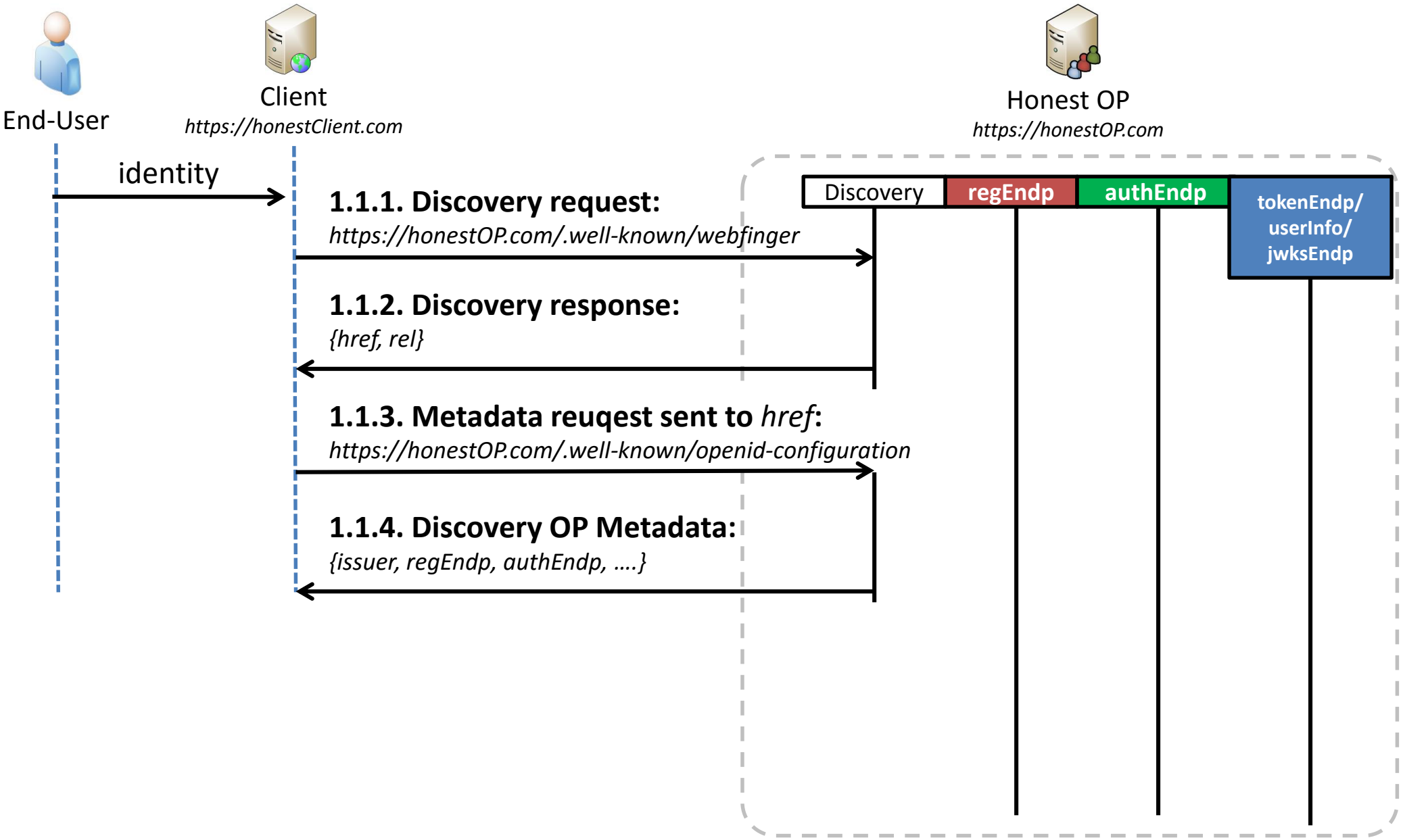
E



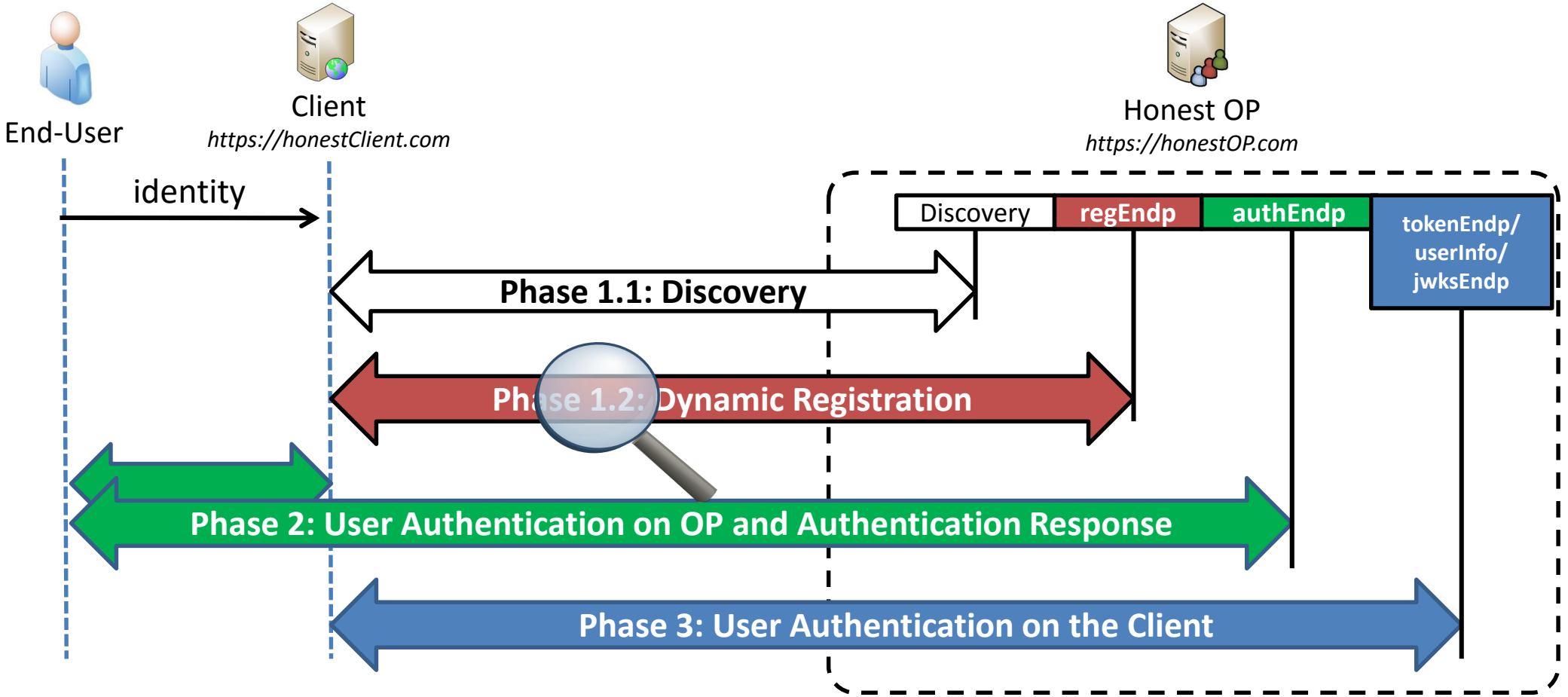
OpenID Connect: Phases with Discovery and Dynamic Registration



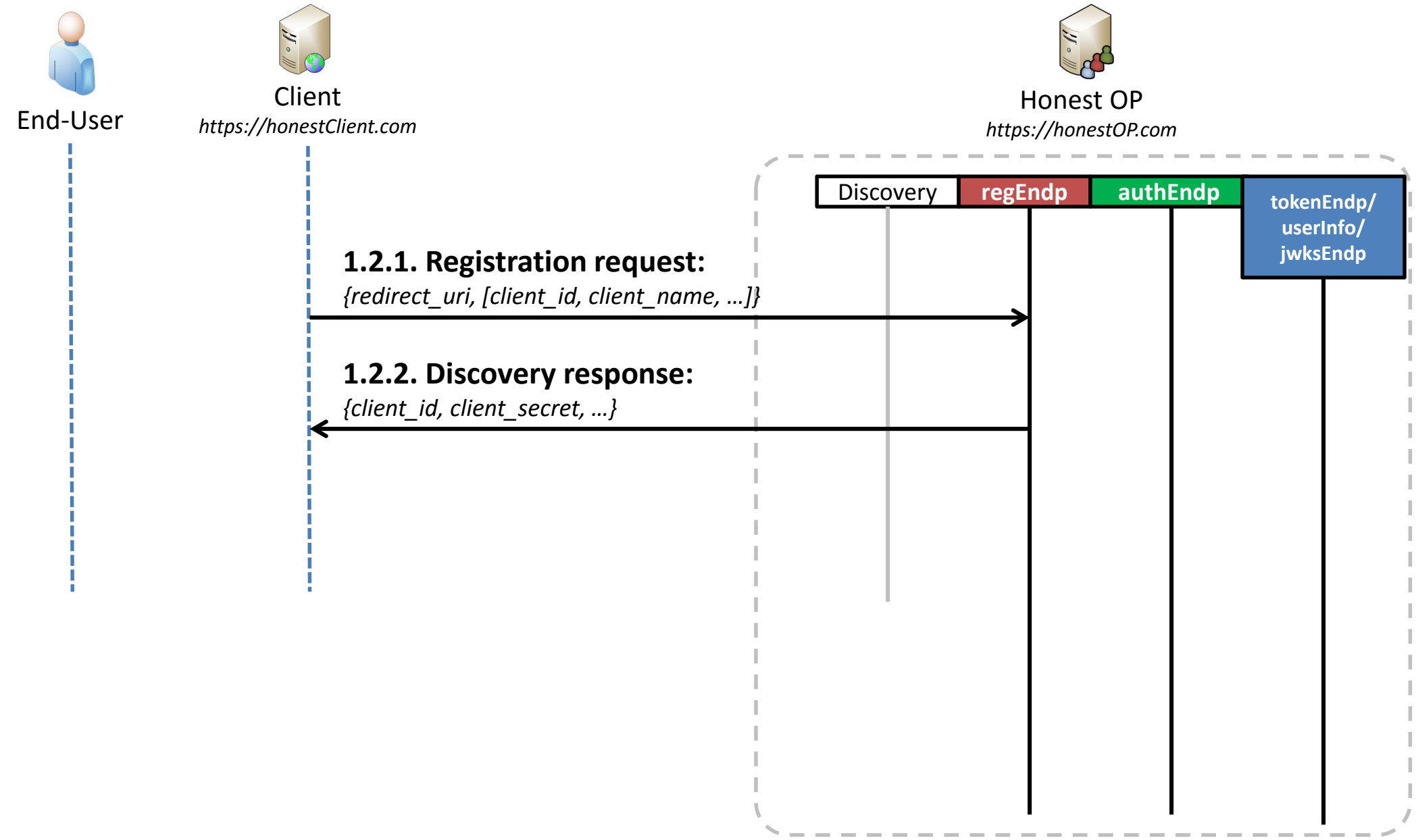
OpenID Connect: Discovery



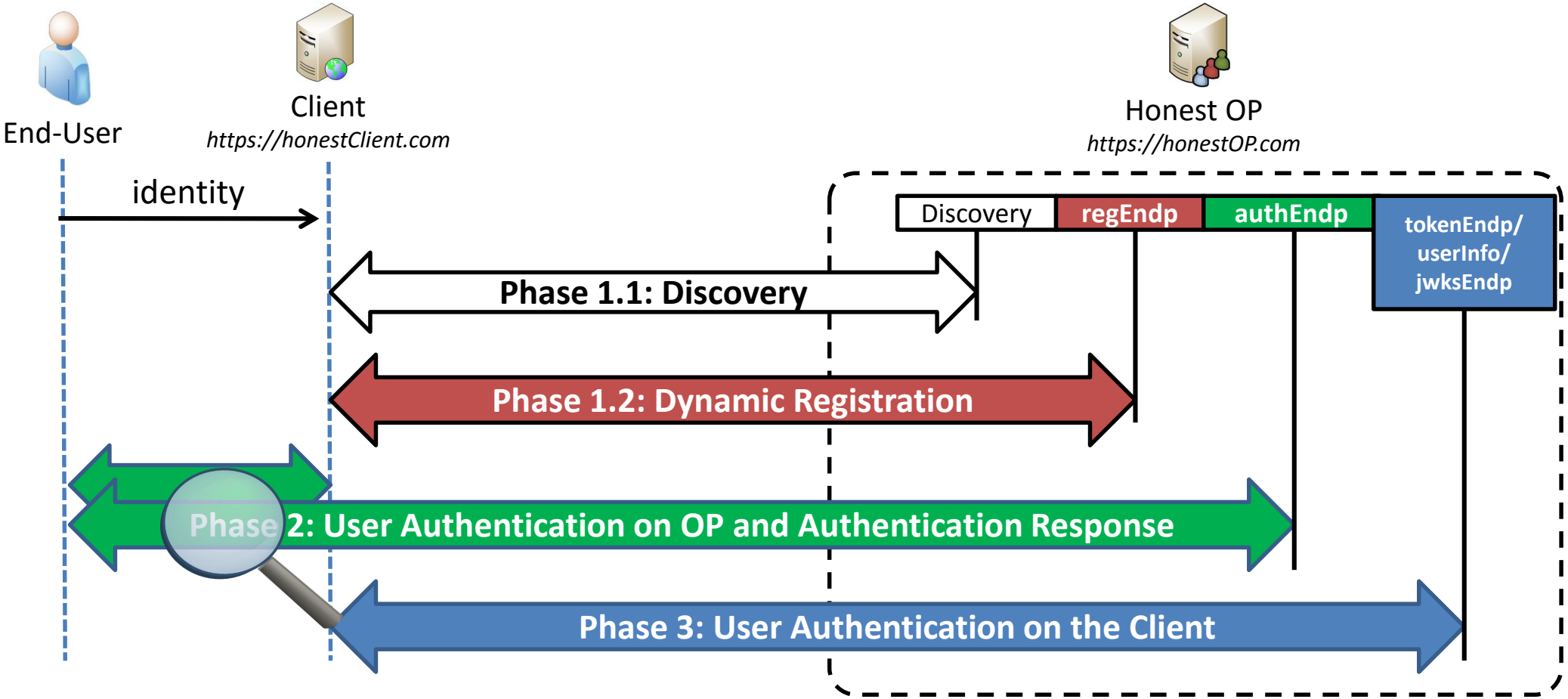
OpenID Connect: Phases with Discovery and Dynamic Registration



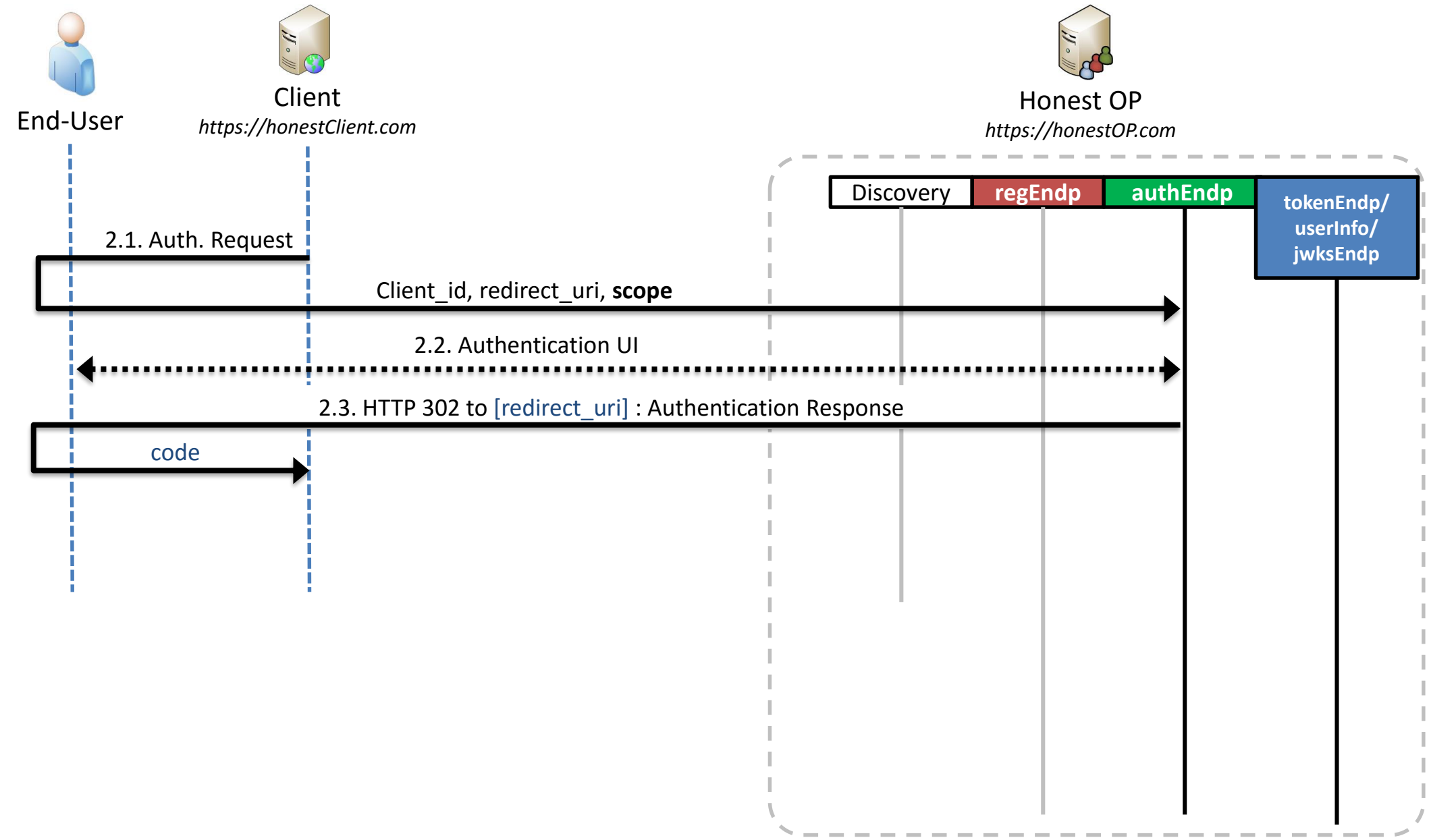
OpenID Connect: Dynamic Registration



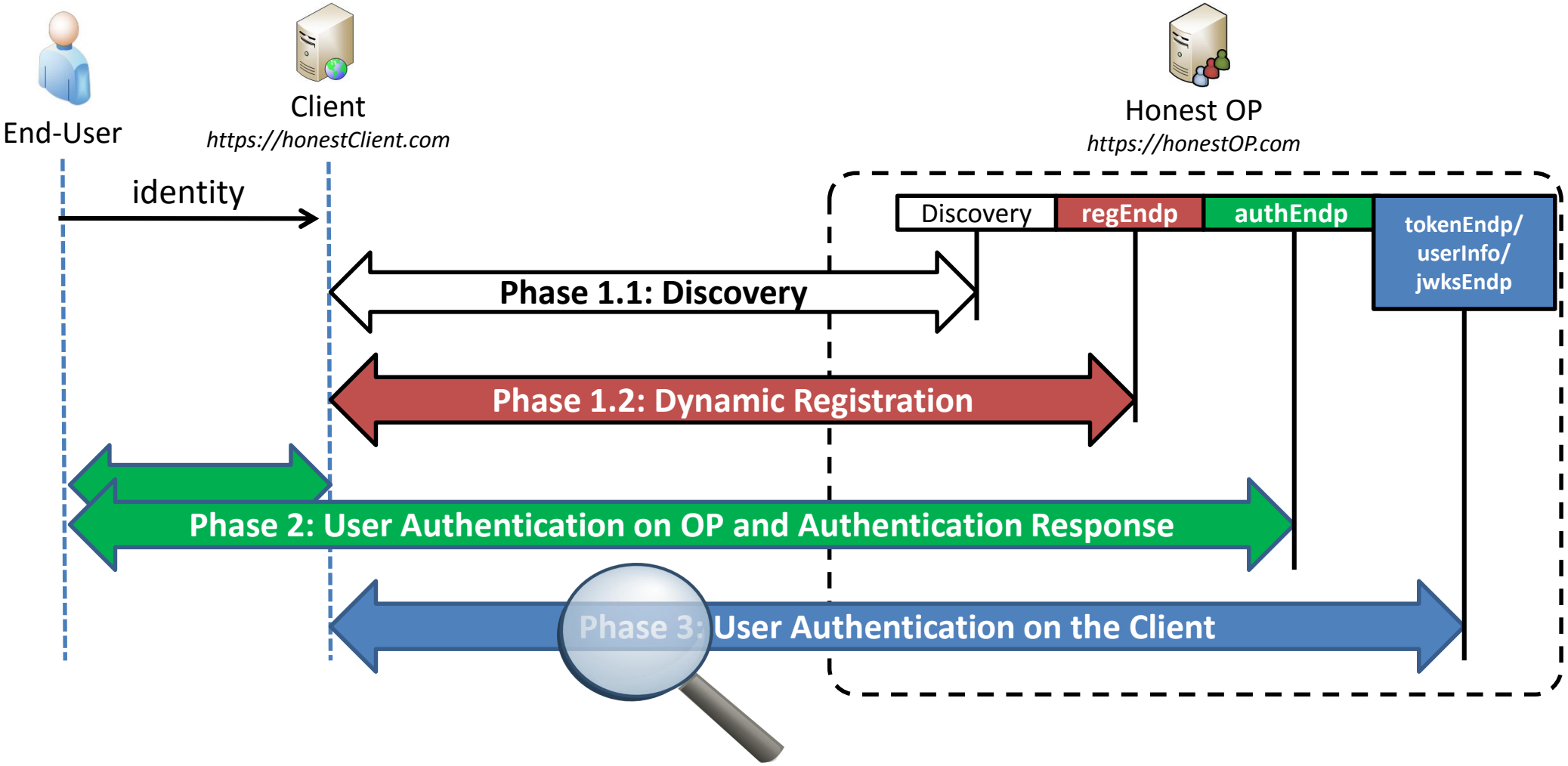
OpenID Connect: Phases with Discovery and Dynamic Registration



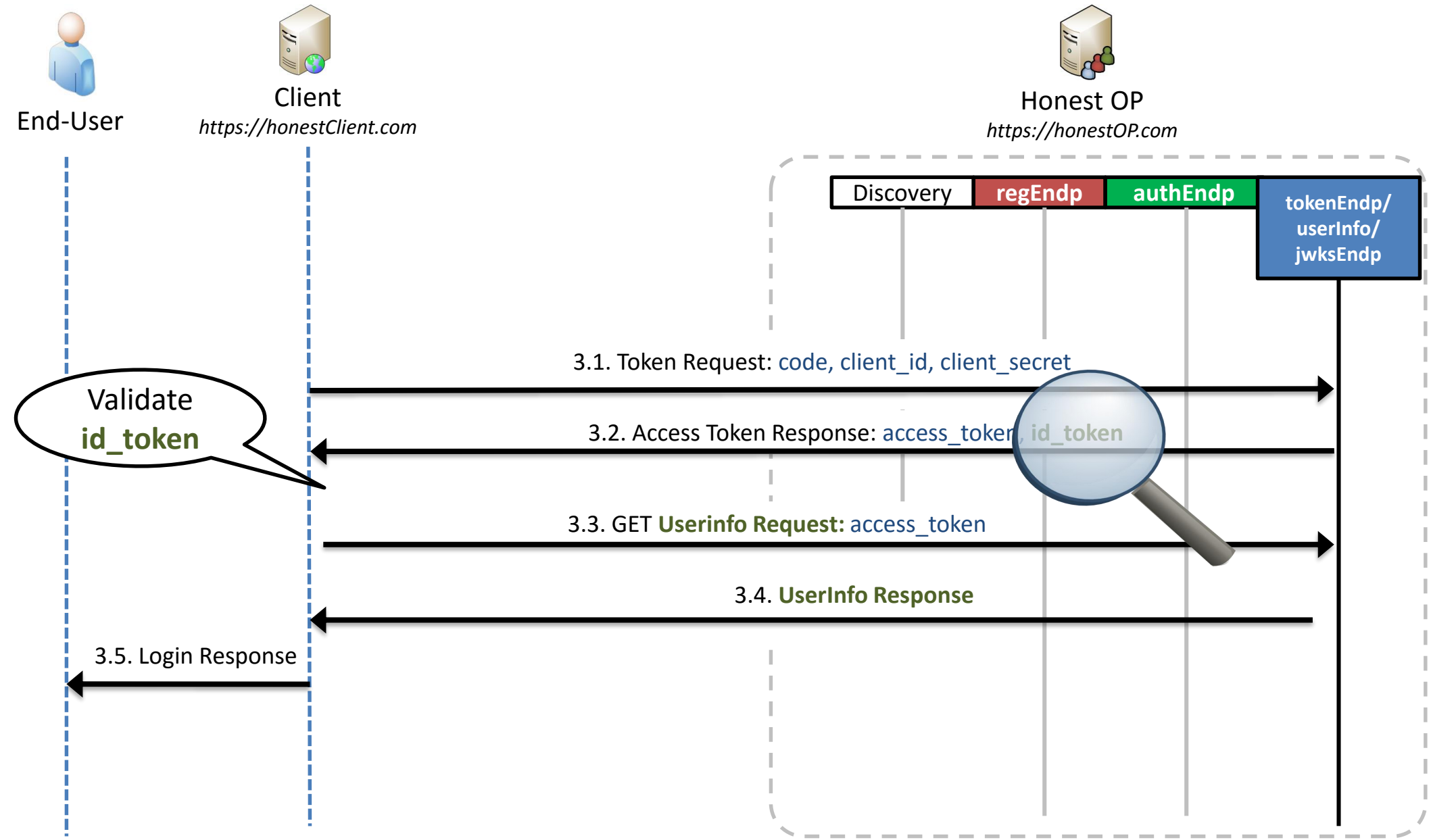
OpenID Connect: User Authentication on OP (Code Flow)



OpenID Connect: Phases with Discovery and Dynamic Registration



OpenID Connect: User Authentication on OP (Code Flow)



OpenID Connect: ID Token


```
eyJhbGciOiJIUzI1NiIsInR
5cCI6IkpXVCJ9.eyJleHAiO
jEzODY4OTkxMzEsImlzcyI6
ImppcmE6MTU0ODk1OTUiLCJ
xc2giOiI4MDYzZmY0Y2ExZT
QxZGY3YmM5MGM4YWI2ZDBmN
jIwN2Q0OTFjZjZkYWQ3YzY2
ZWE3OTdiNDYxNGI3MTkyMmU
5IiwiaWF0IjoxMzg2ODk4OT
UxfQ.uKqU9dTB6gKwG6jQCu
XYAiMNdfNRw98Hw_IWuA5Ma
Mo
```


Header


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```


Body


```
{
  "iss": "https://honestOP.com/",
  "sub": "user1",
  "exp": 1444148908,
  "iat": 1444148308,
  "nonce": "40c6b33b9a2e",
  "aud": "honestClientId",
}
```


Signature

Verify: valid/invalid?

ID Token: Summary

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Body

```
{
  "iss": "https://honestOP.com/",
  "sub": "user1",
  "exp": 1444148908,
  "iat": 1444148308,
  "nonce": "40c6b3...b9a2e",
  "aud": "honestCl...tId",
}
```

Signature

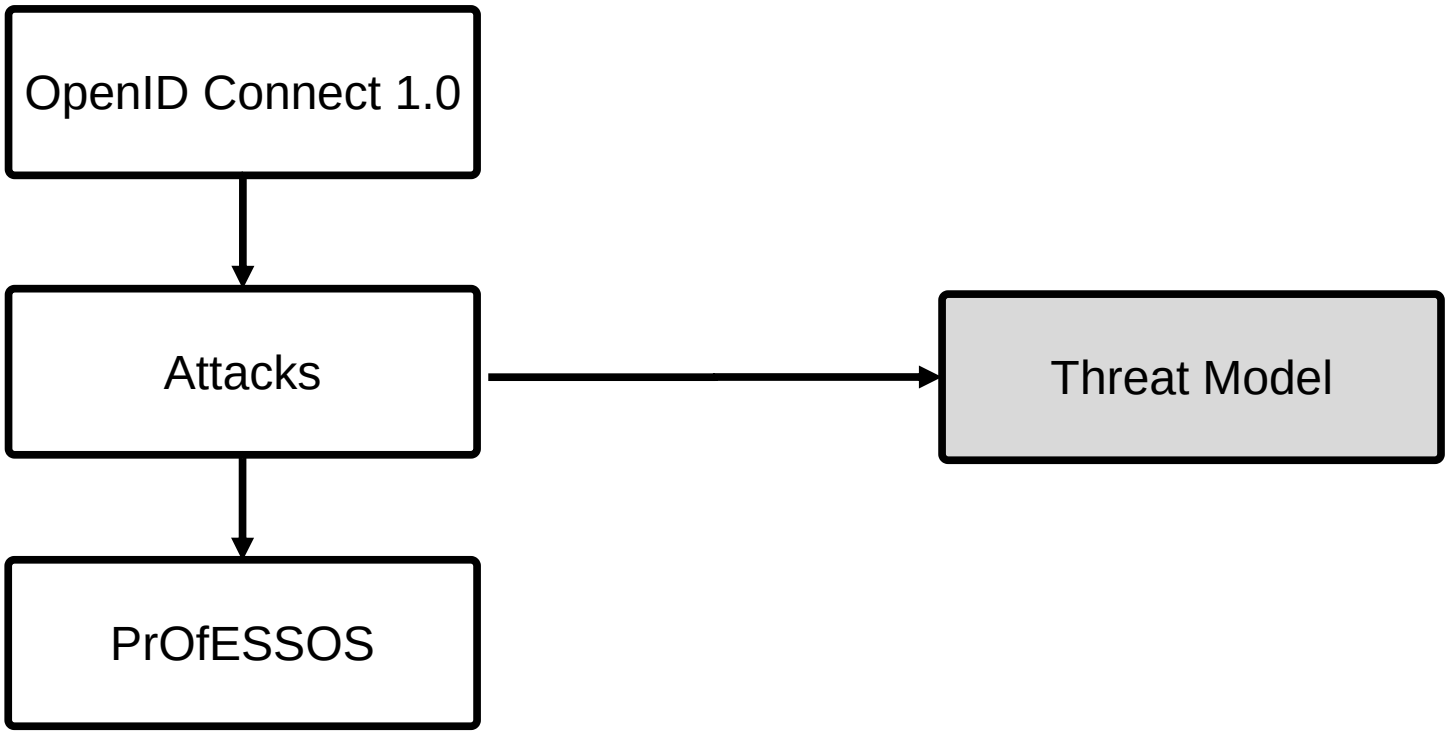
Verify: valid/invalid?

= iss sub

= iat exp nonce

= aud

On the Security of OpenID Connect

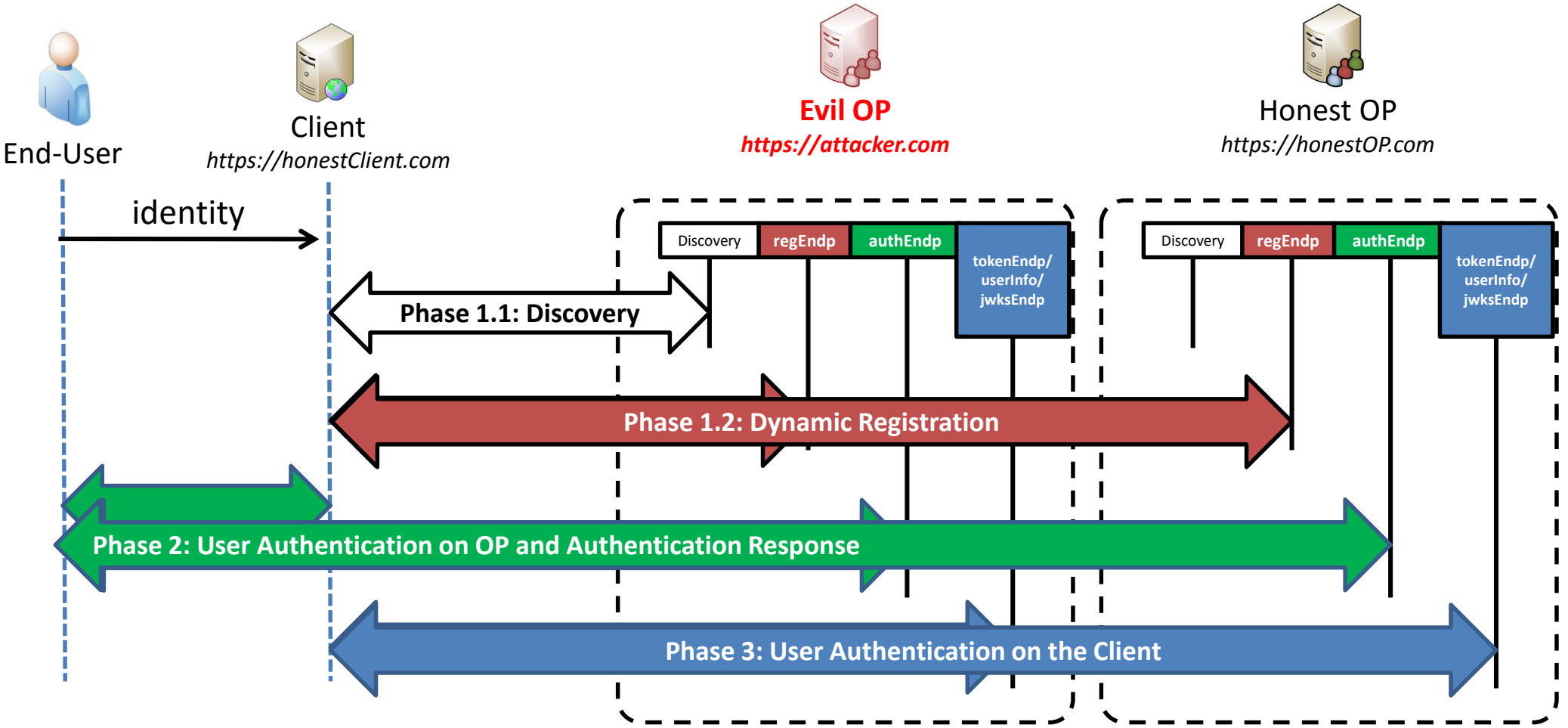


Threat Model

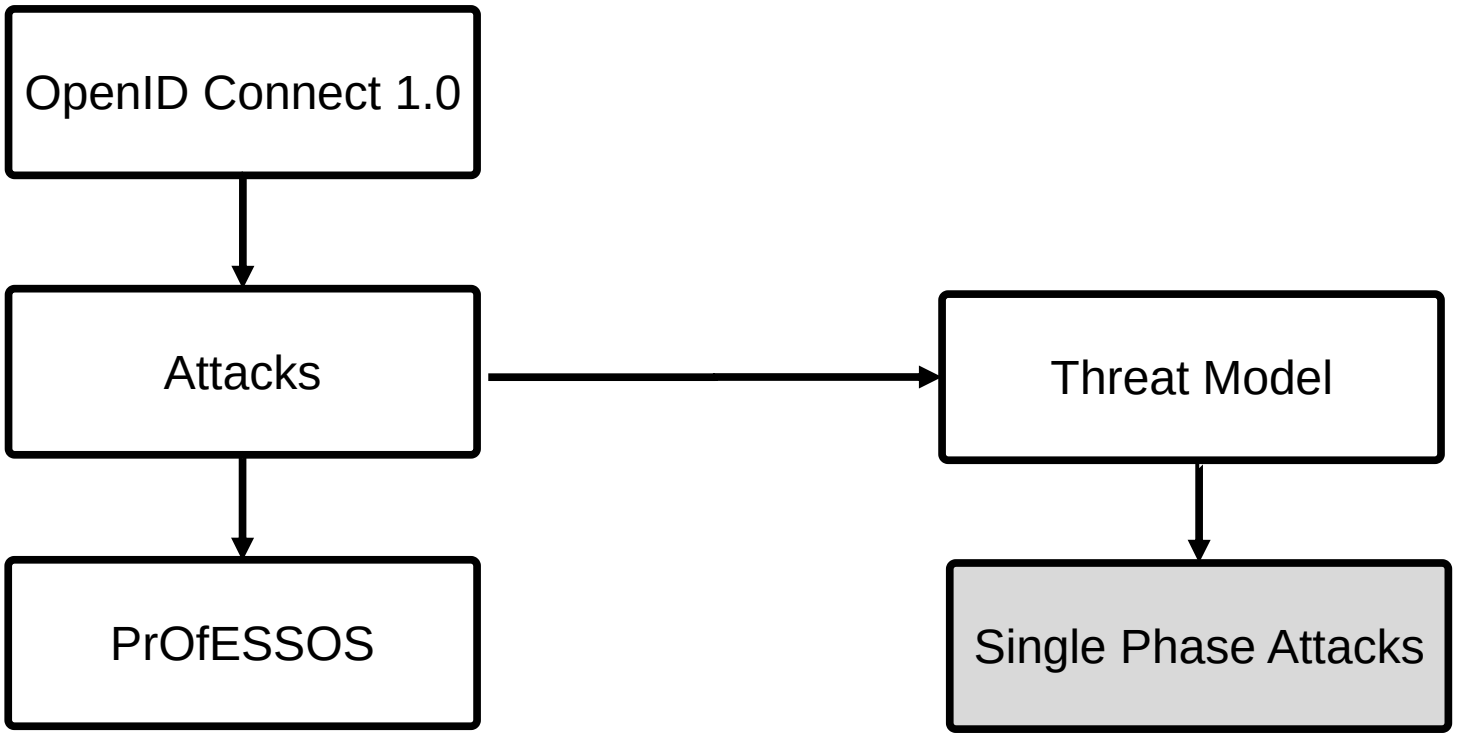
- Web attacker model

- Two Attack Categories
  - **Category A** with interaction of the victim
  - **Category B** no interaction at all

Attacker IdP



On the Security of OpenID Connect



Attack: ID Spoofing

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Body

```
{
  "iss": "https://honestOP.com/",
  "sub": "user1",
  "exp": 1444148908,
  "iat": 1444148308,
  "nonce": "40c6b33b9a2e",
  "aud": "honestClientId",
}
```

Signature

```
Verify: valid/invalid?
```

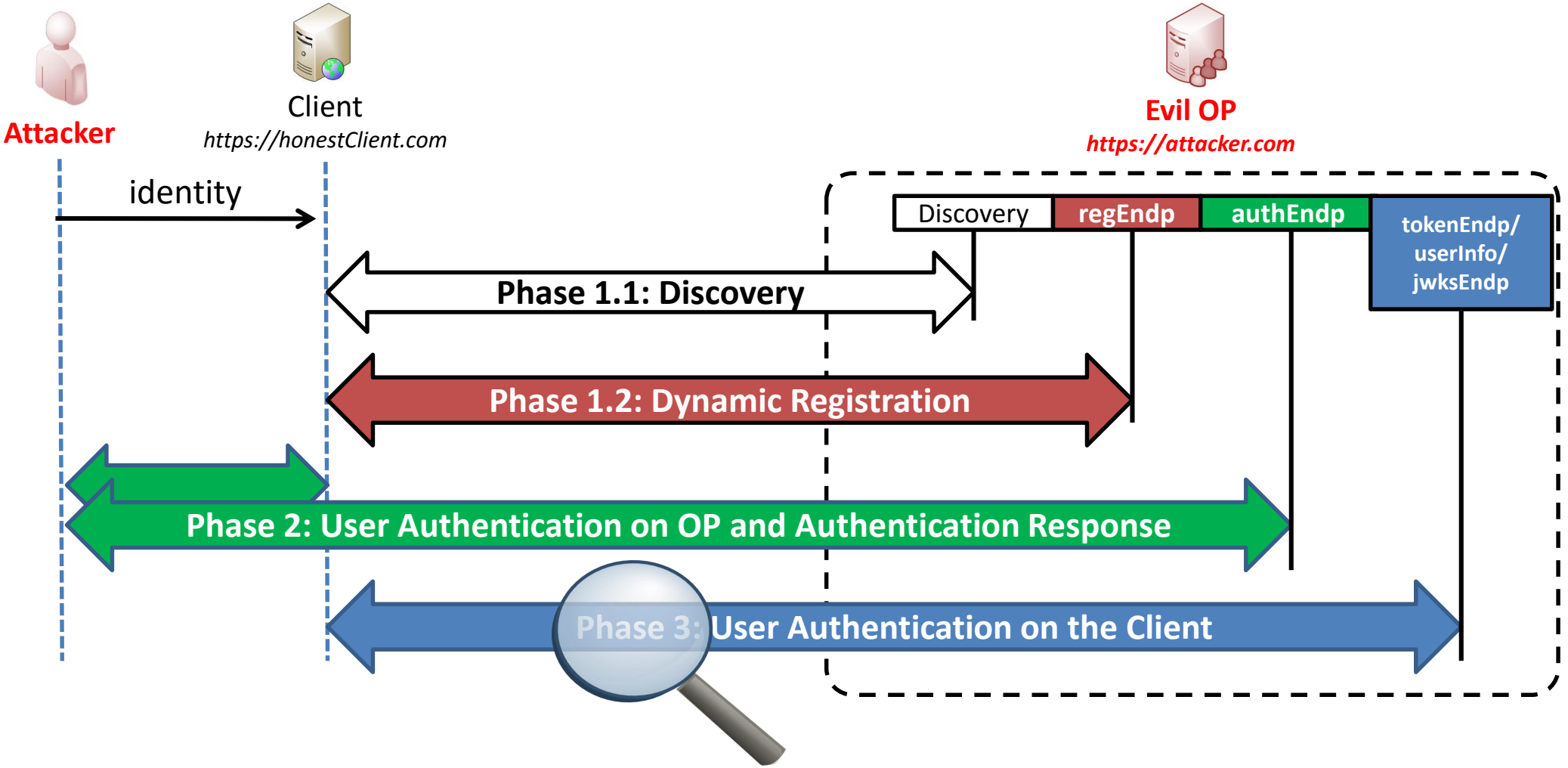
= iss sub

= iat exp nonce

= aud

Category B

Implementation flaws on the Client: ID Spoofing



Implementation flaws on the Client: ID Spoofing

= Issuer *http://google.com* + Subject *Alice*

Token created by OP: http://google.com

= Issuer *http://attacker.com* + Subject *Alice*

Token created by OP: http://attacker.com

= Issuer *http://google.com* + Subject *Alice*

Token created by OP: http://attacker.com

Attack: Wrong Recipient

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Body

```
{
  "iss": "https://honestOP.com/",
  "sub": "user1",
  "exp": 1444148908,
  "iat": 1444148308,
  "nonce": "40c6b33b9a2e",
  "aud": "honestClientId",
}
```

Signature

```
Verify: valid/invalid?
```

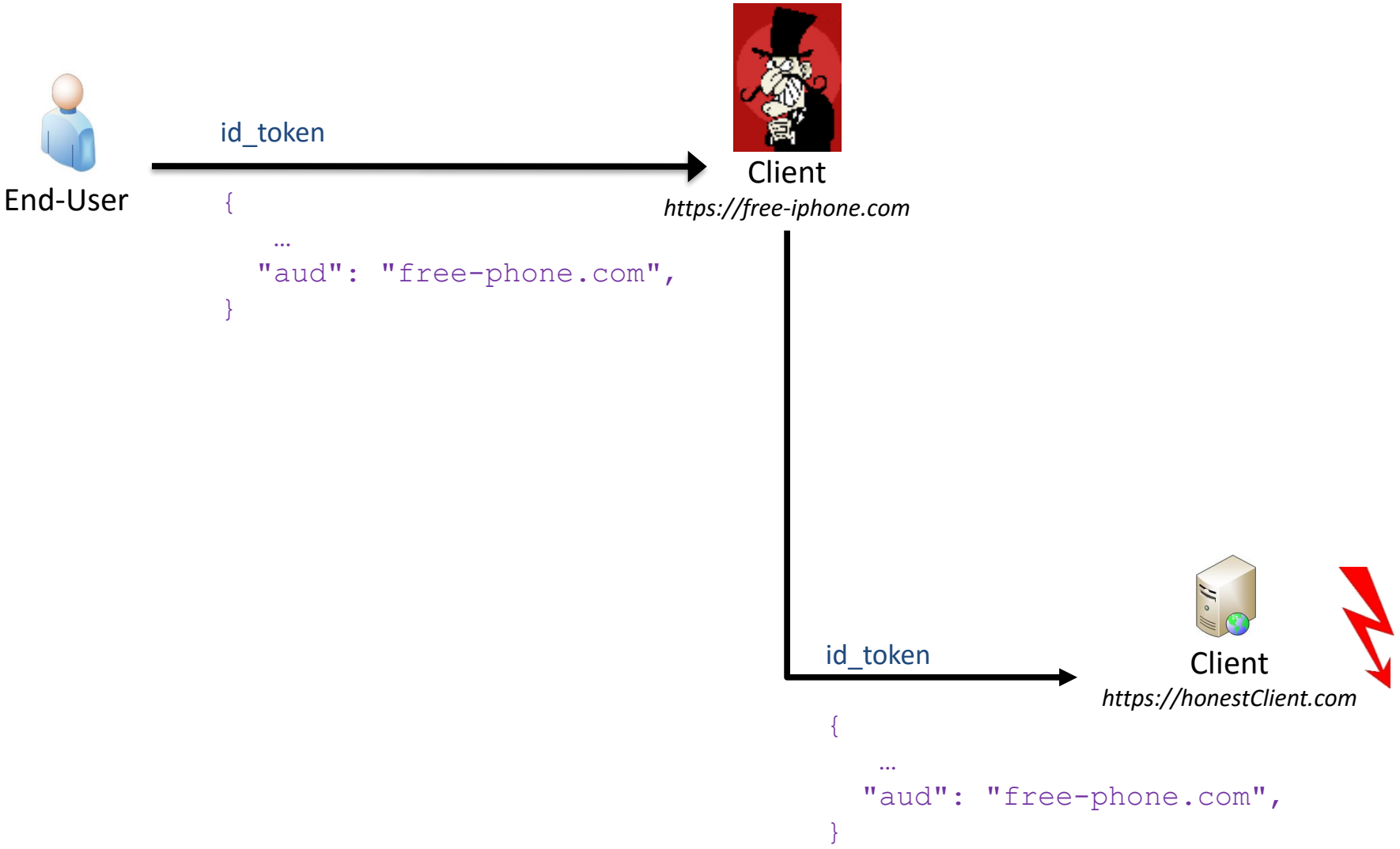
= iss sub

= iat exp nonce

= aud

Category A

Attack: Wrong Recipient



Attack: Replay

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Body

```
{
  "iss": "https://honestOP.com/",
  "sub": "user1",
  "exp": 1444148908,
  "iat": 1444148308,
  "nonce": "40c6b33b9a2e",
  "aud": "honestClientId",
}
```

Signature

```
Verify: valid/invalid?
```

= iss sub

= iat exp nonce

= aud

Category B

Attack: Signature Bypass

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Body

```
{
  "iss": "https://honestOP.com/",
  "sub": "user1",
  "exp": 1444148908,
  "iat": 1444148308,
  "nonce": "40c6b33b9a2e",
  "aud": "honestClientId",
}
```

Signature

```
Verify: valid/invalid?
```

= iss sub

= iat exp nonce

= aud

Category B

Attack: Signature Bypass

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Body

```
{
  "iss": "https://honestC...om/",
  "sub": "user1",
  "exp": 14...308,
  "iat": 1444148308,
  "nonce": "40c6b33b9a2e",
  "aud": "honestClientId",
}
```

Signature

```
Verify: valid/invalid?
```

= iss sub

= iat exp nonce

aud

Category B

Implementation flaws on the Client: Signature Verification

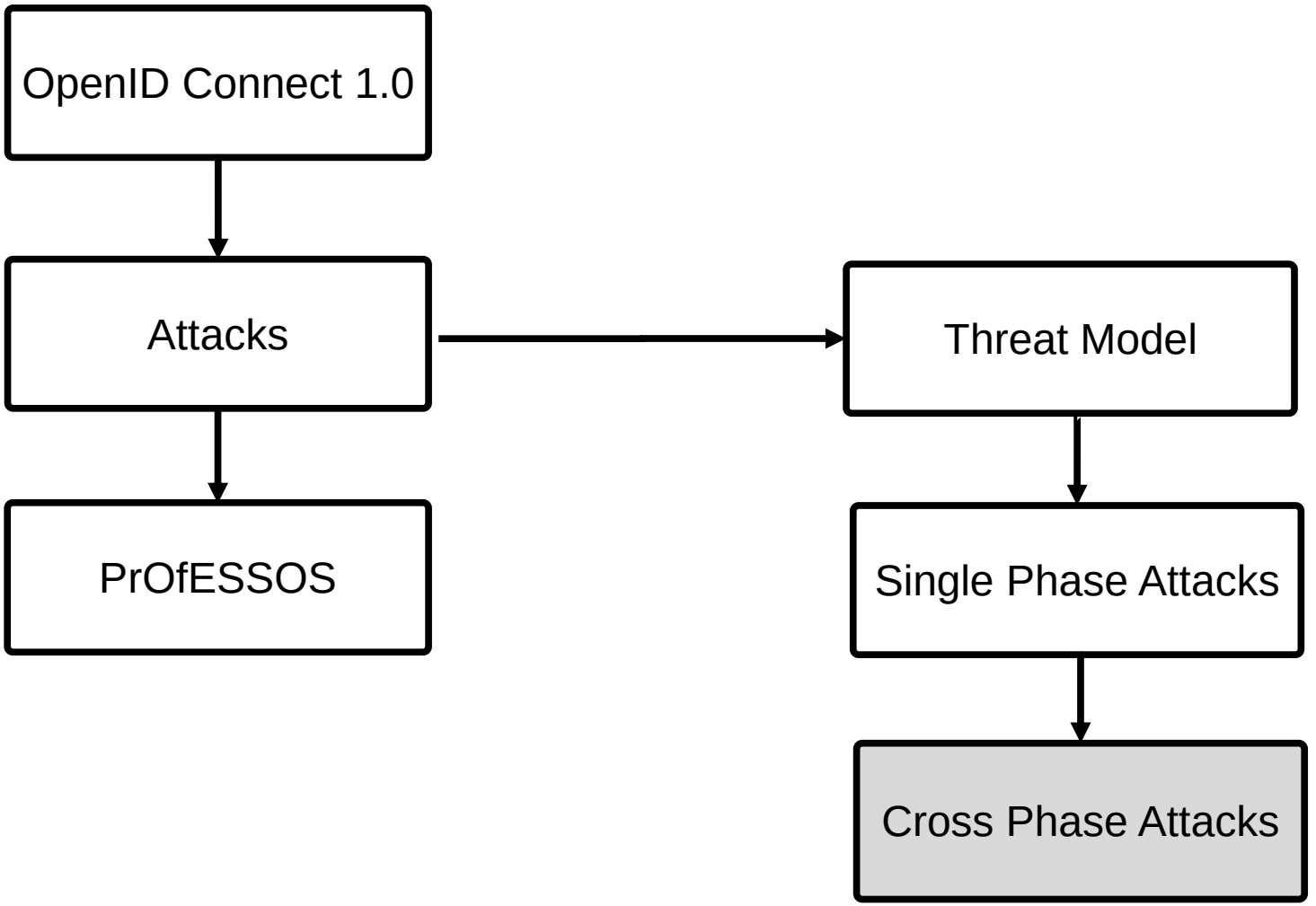
- `alg` defines Algorithm
- Supported values:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

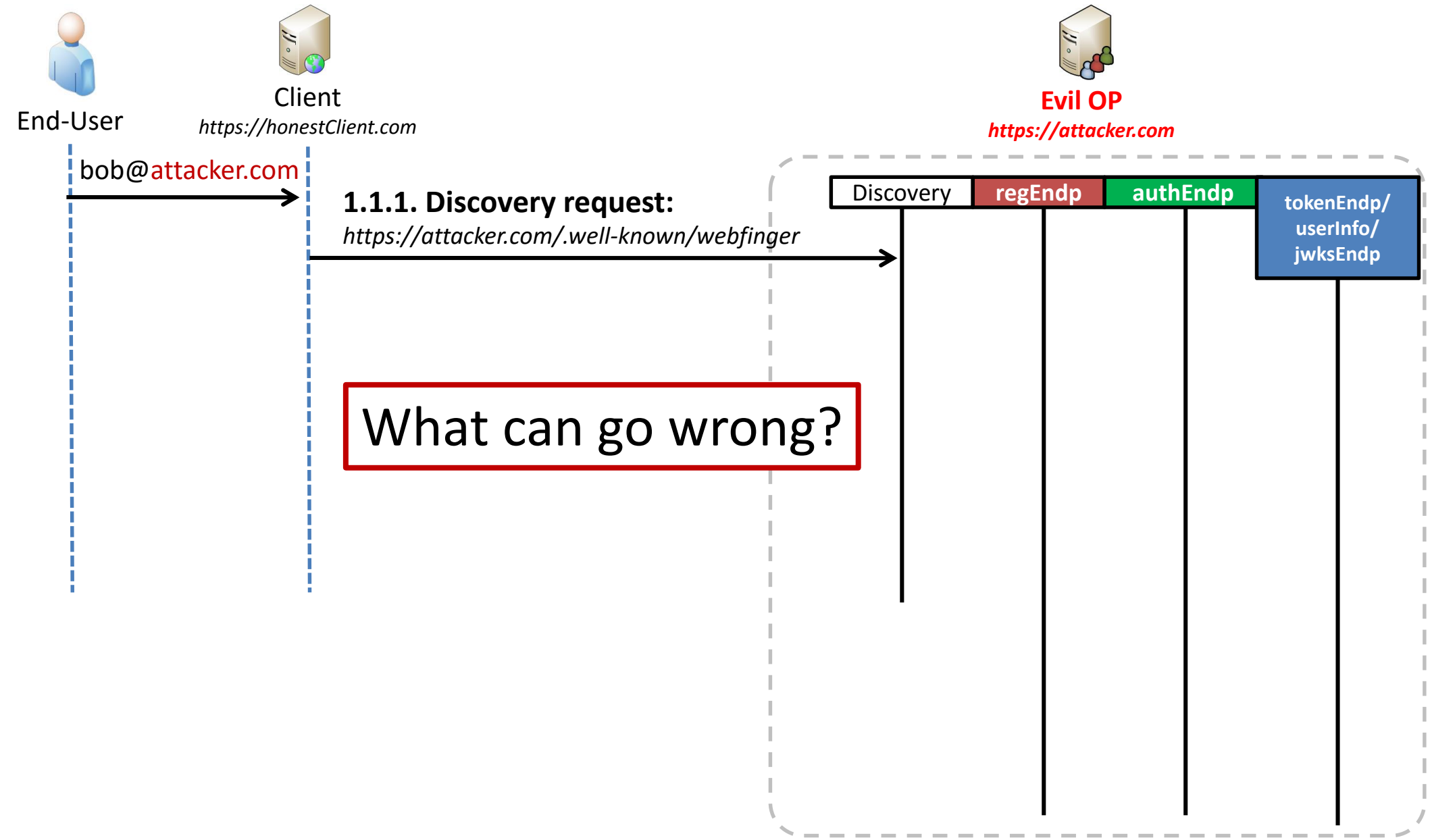
alg Parameter Value	Digital Signature or MAC Algorithm	Implementation Requirements
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS-v1_5 using SHA-512	Optional
ES256	ECDSA using P-256 and SHA-256	Recommended+
ES384	ECDSA using P-384 and SHA-384	Optional
ES512	ECDSA using P-521 and SHA-512	Optional
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC performed	Optional

Setting alg to „none“ allows to remove the signature

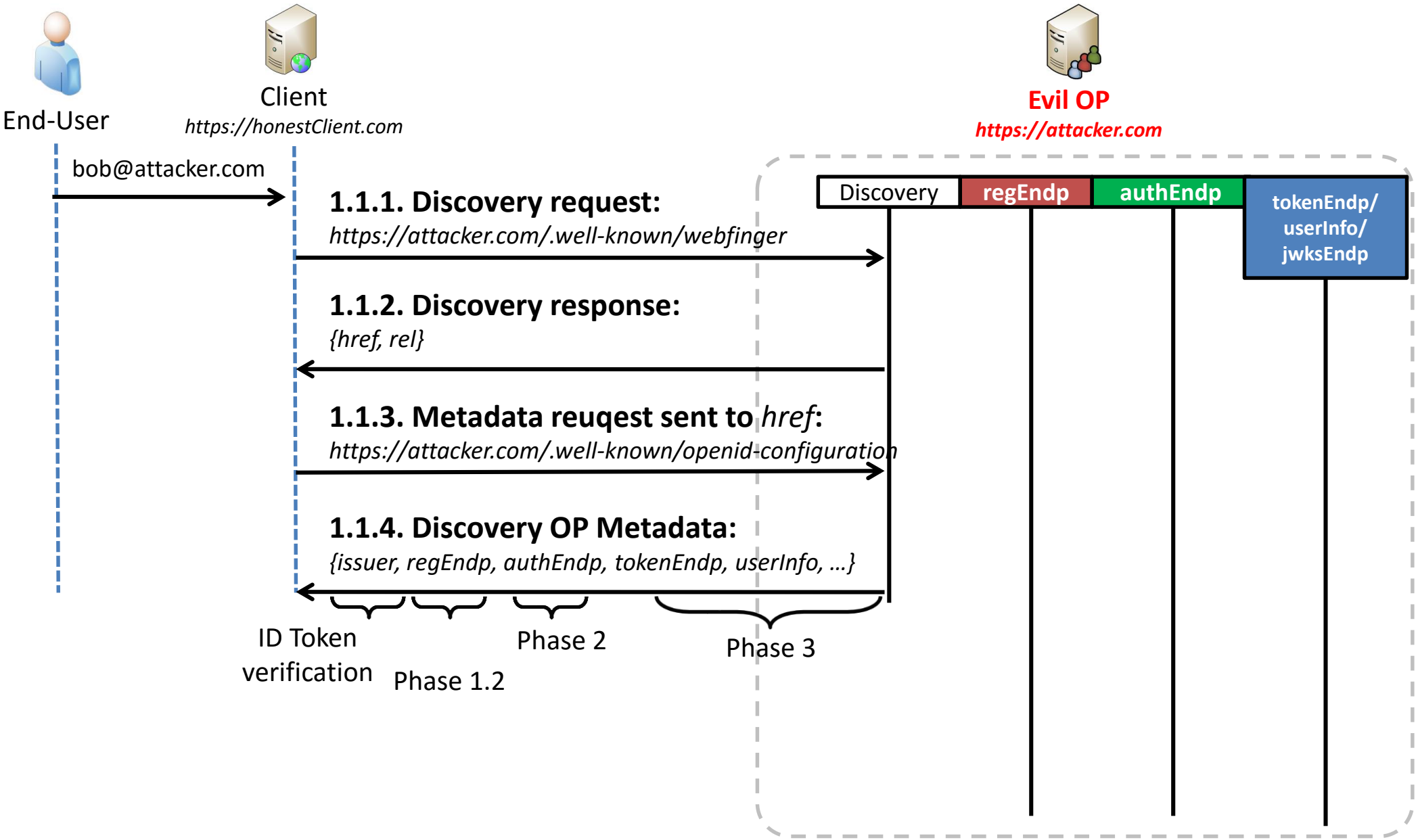
On the security of OpenID Connect



Cross Phase Attacks



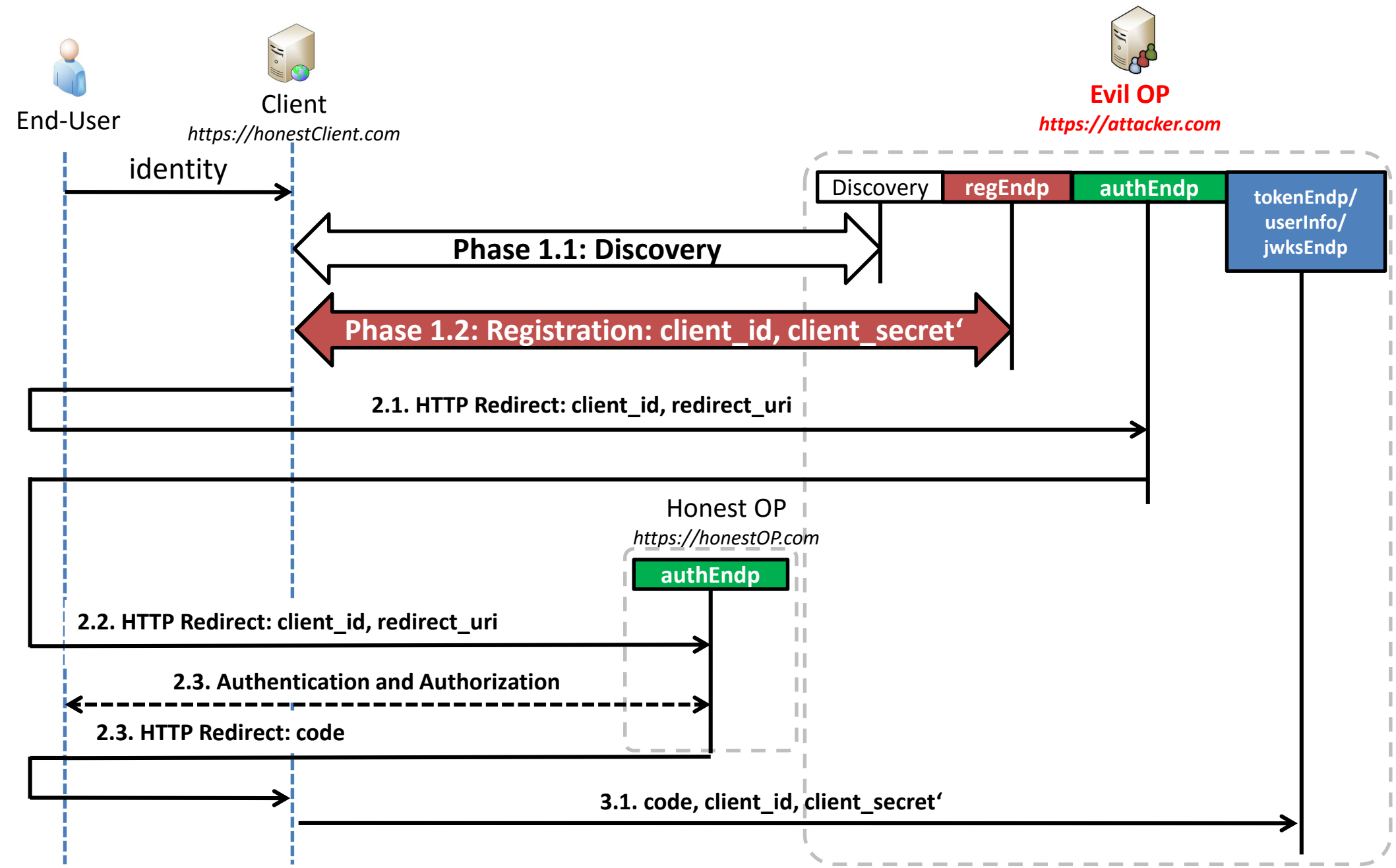
Cross Phase Attacks



Cross Phase Attacks

- Issuer Confusion
- IdP Confusion
- Malicious Endpoint Attacks

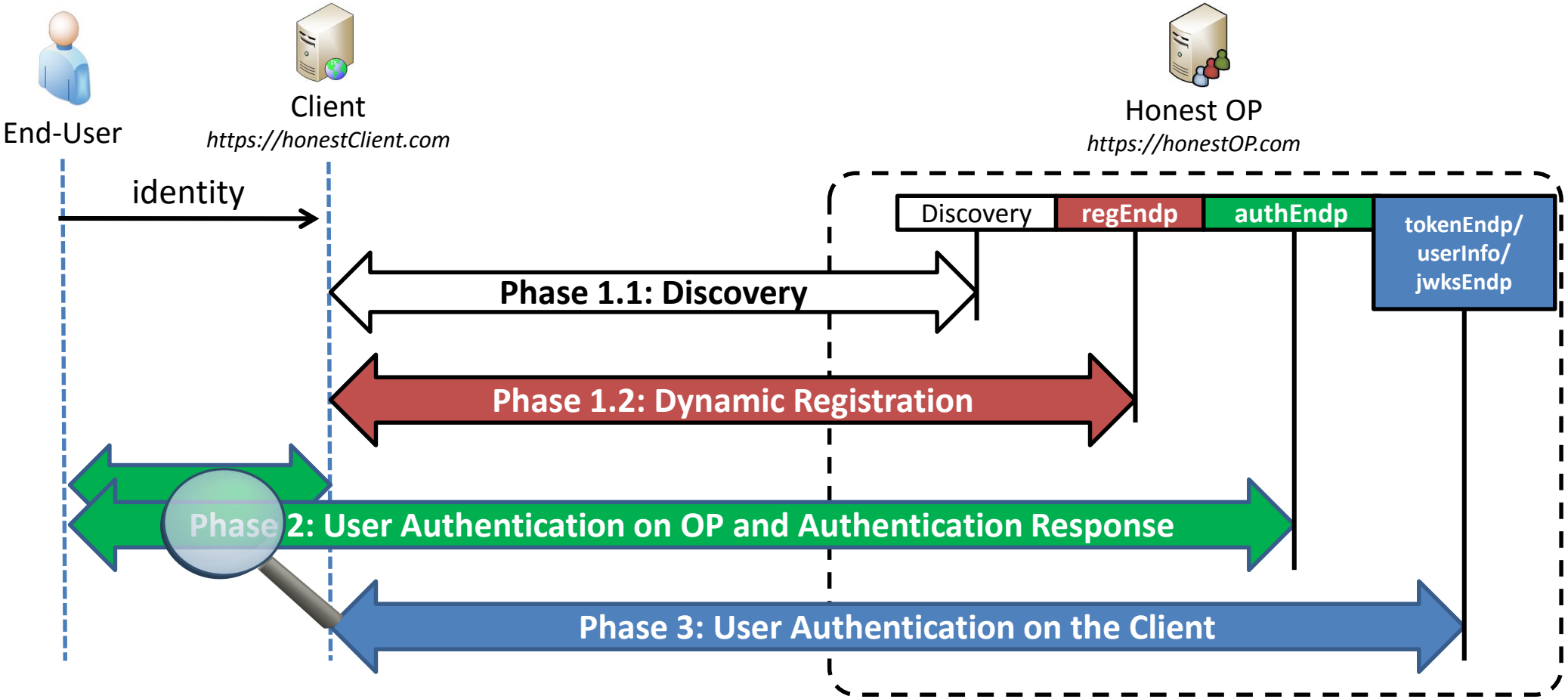
IdP Confusion



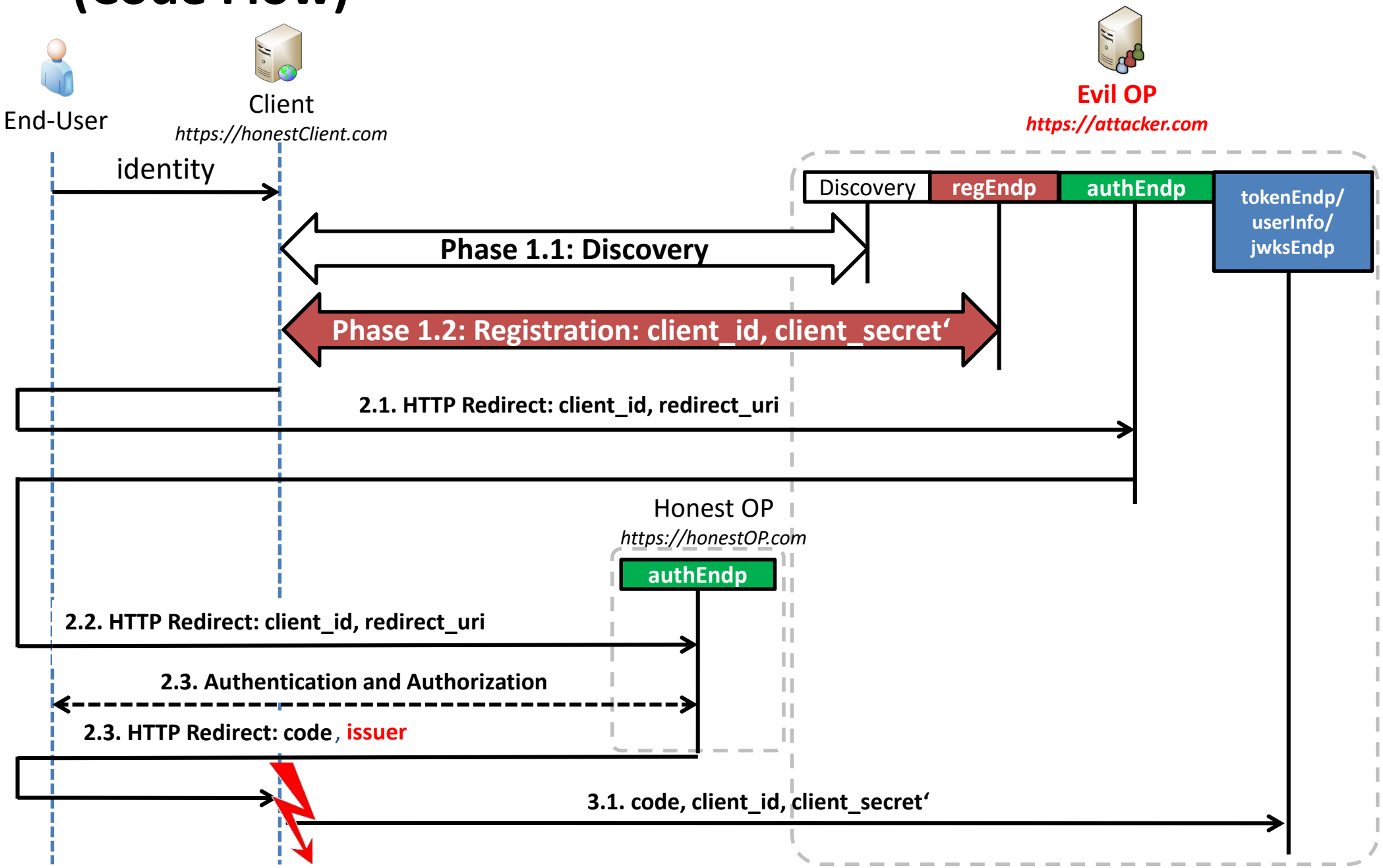
OpenID Connect: Countermeasures

- Attack concepts known since 2012
  - „Do not trust me: Using malicious IdPs for analyzing and attacking Single Sign-On“ (OpenID 2.0)
  - „Your Software at my Service” (SAML 2.0)
- Attacks reported in September 2014
  - Reaction in Oktober 2015
  - First mitigation draft in January 2016
- Changes in the OpenID Connect and OAuth specifications
  - https://tools.ietf.org/html/draft-jones-oauth-mix-up-mitigation-01

OpenID Connect: Countermeasures



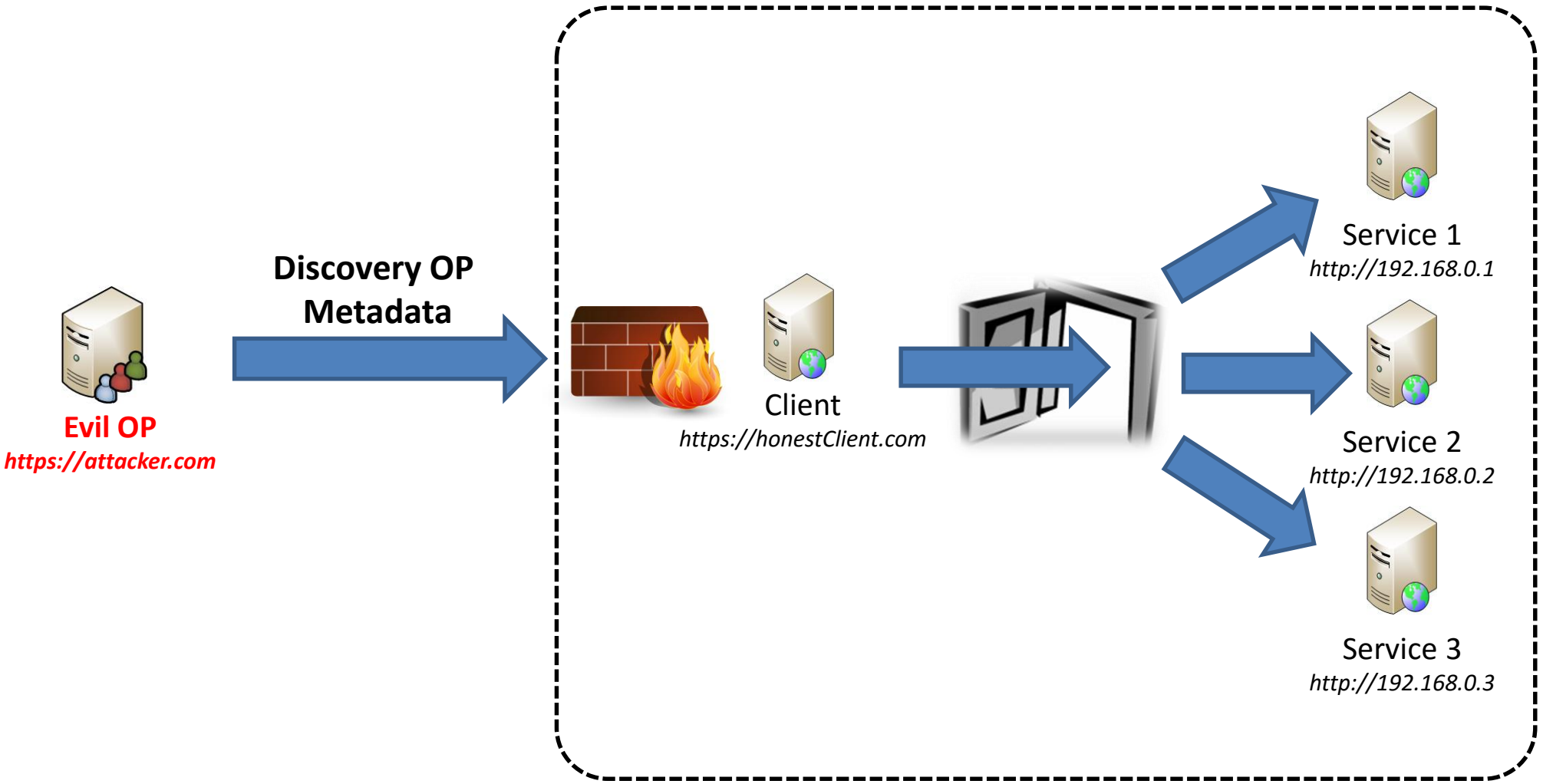
OpenID Connect: Countermeasures (Code Flow)



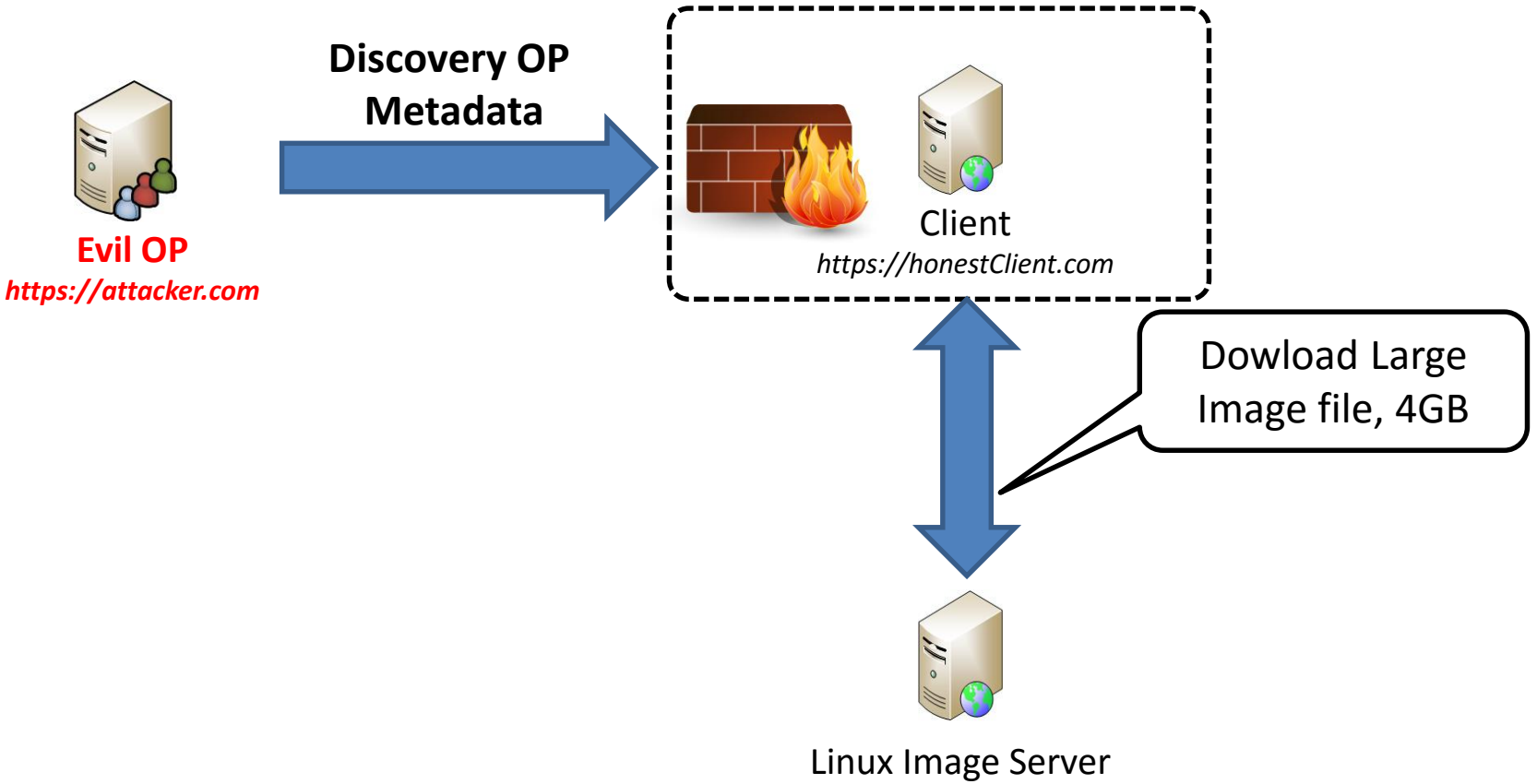
Malicious Endpoints Attacks: Idea

The *maliciously acting Discovery service* influences partially the protocol execution in **Phase 1.2, Phase 2 and Phase 3**

Malicious Endpoints Attacks: SSRF

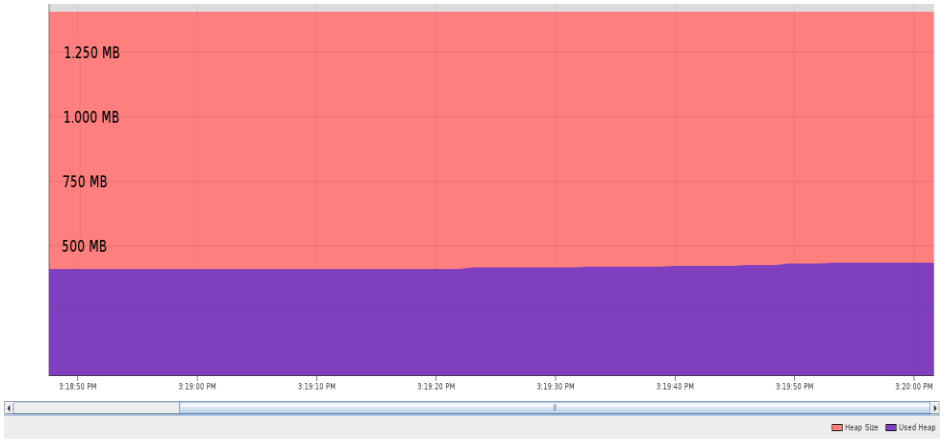


Malicious Endpoints Attacks: DoS

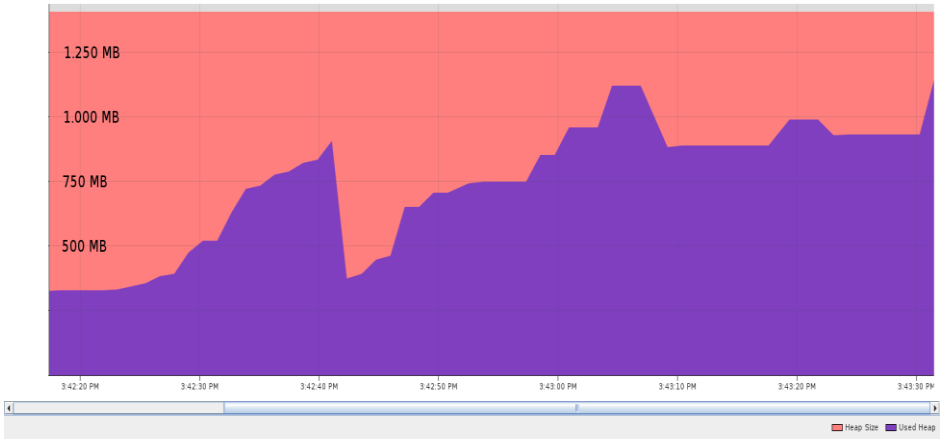


Malicious Endpoints Attacks: DoS

OpenID Connect with 5 parallel connections to an Honest OP



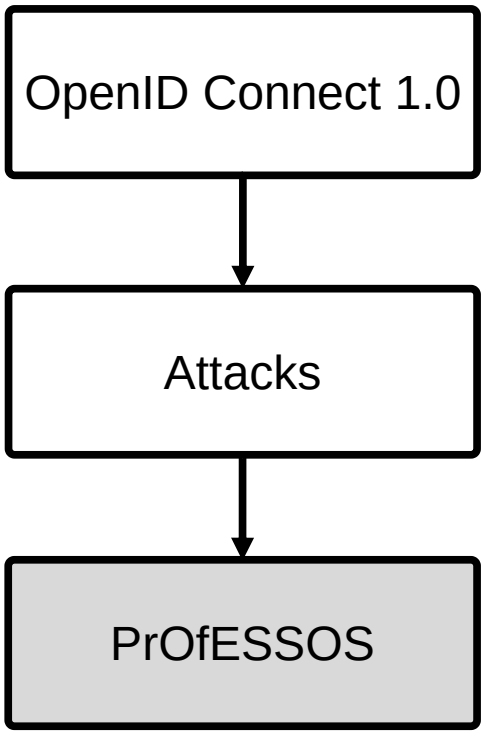
OpenID Connect with 5 parallel connections to a Malicious Discovery service



Evaluation Results

SPs / Libraries / Attack Category	Custom IdP	Dynamic Trust	Single-Phase Attacks				Cross-Phase Attacks	
			IDS Cat $\mathcal{B}$	Wrong Recipient Cat $\mathcal{A}$	Replay Cat $\mathcal{A}$	Signature Bypass Cat $\mathcal{B}$	Issuer Conf. Cat $\mathcal{B}$	Specification Flaws Cat $\mathcal{A}$
mod_auth_openidc	**Yes**	**Yes**	✓	✓	✓	**Vuln.**	✓	**Vuln.**
MITREid Connect	**Yes**	**Yes**	✓	✓	✓	✓	✓	**Vuln.**
oidc-client	**Yes**	**Yes**	**Vuln.**	**Vuln.**	**Vuln.**	✓	**Vuln.**	**Vuln.**
phpOIDC	**Yes**	**Yes**	**Vuln.**	**Vuln.**	**Vuln.**	**Vuln.**	**Vuln.**	**Vuln.**
DrupalOpenIDConnectd	**Yes**	*No*	**Vuln.**	**Vuln.**	**Vuln.**	**Vuln.**	**Vuln.**	**Vuln.**
pyoidc	**Yes**	**Yes**	**Vuln.**	**Vuln.**	**Vuln.**	**Vuln.**	✓	**Vuln.**
Ruby OpenIDConnect	**Yes**	**Yes**	✓	✓	✓	✓	✓	**Vuln.**
Apache Oltu	**Yes**	*No*	✓	✓	**Vuln.**	**Vuln.**	✓	**Vuln.**
Total Successful Attacks	8/8	6/8	4/8	4/8	5/8	5/8	3/8	8/8

On the Security of OpenID Connect



PrOfESSOS

Security Test Runner

https://sso-security.o

Stage 1:
Configu

Stage 3
Security Report

Pentester

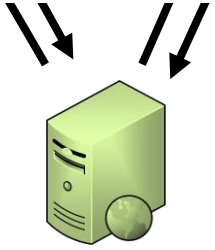
Stage 1: Setup - Client Parameters

OP Parameters

Test ID: bvw0KwJn8ZYnKYgcKCt8dA
Honest OP Identity: `http://idp.oidc.honest-sso.de/bvw0KwJn8ZYnKYgcKCt8dA`
Evil OP Identity: `http://idp.oidc.attack-sso.de/bvw0KwJn8ZYnKYgcKCt8dA`

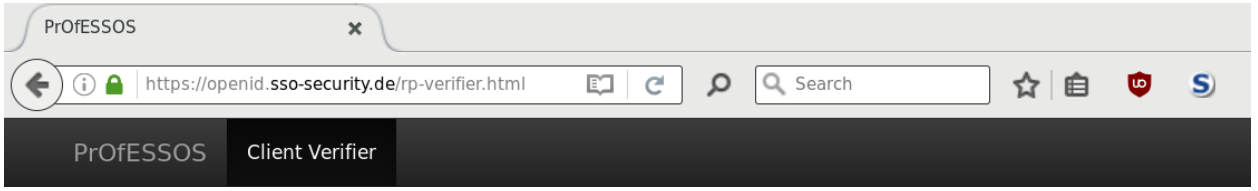
Client Parameters

Login-Site URL:	
Input-Field Name:	
Selenium Script:	
Success URL:	
Honest User Needle:	
Evil User Needle:	
User Profile URL:	



Service Provider

Identity Provider



Legend

- **Test not run**
- **Test passed**
- **Test failed (Attack succeeded)**
- **Test outcome undetermined**

Stage 1: Setup - Client Parameters

OP Parameters

Test ID: `Q0XdAuKdNZIOvv-5TF12SA`
Honest OP Identity: `http://idp.oidc.honest-sso.de/Q0XdAuKdNZIOvv-5TF12SA`
Evil OP Identity: `http://idp.oidc.attack-sso.de/Q0XdAuKdNZIOvv-5TF12SA`

Client Parameters

Login-Site URL: http://www.honestsp.de:8080/simple-web-app/login

Input-Field Name: identifier

Selenium Script:

```
var opUrl = document.querySelector("input[name='identifier']");
opUrl.value = "$step["browser.input.op_url"]";
opUrl.form.submit();
```

Success URL: http://www.honestsp.de:8080/simple-web-app/

Honest User Needle: {sub=honest-op-test-subject, iss=http://idp.oidc.honest-sso.de/Q0XdAuKdNZIOvv-5TF12SA}

Evil User Needle:

{sub=evil-op-test-subject, iss=http://idp.oidc.attack-sso.de/Q0XdAuKdNZIOvv-5TF12SA}

http://www.honestsp.de:8080/simple-web-app/user

Stage 2: Configuration Evaluation

Learn

Learning Log

Stage 3: Tests and Attacks

Run all Tests

PrOfESSOS

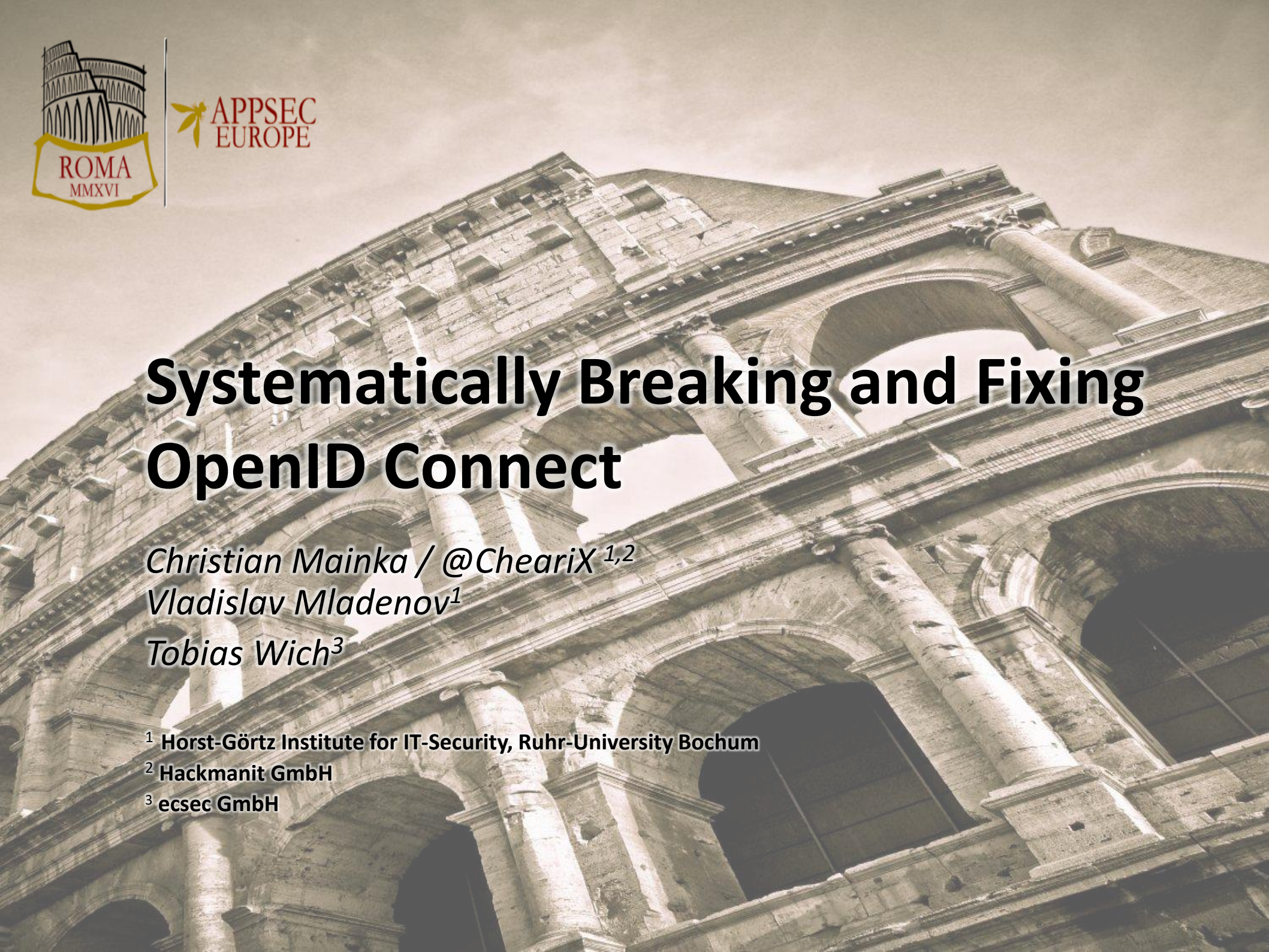
- Current status
  - ✓ Configuration and Learning stage
  - ✓ Security tests for Service Providers
    - ✓ 20 security tests implemented
    - ✗ More tests will be implemented
  - ✗ Countermeasure advices and improvements
  - ✗ Security evaluation of Identity Providers
  - ✗ OAuth 2.0

Conclusion

- OIDC Specification addresses Single Phase Attacks
- But *stupid* implementations flaws will always exist
  - Specifications are too complex to understand
- Security testing during development can help
  - PrOfESSOS

Sources

- http://ssoattacks.org/OIDC_MaliciousDiscoveryService/
- http://web-in-security.blogspot.de/
- „On the security of modern Single Sign-On Protocols: Second-Order Vulnerabilities in OpenID Connect”
  - http://arxiv.org/abs/1508.04324
- Mitigation
  - https://tools.ietf.org/html/draft-jones-oauth-mix-up-mitigation-01



Systematically Breaking and Fixing OpenID Connect

Christian Mainka / @CheariX [1,2]
Vladislav Mladenov[1]
Tobias Wich[3]

[1] Horst-Görtz Institute for IT-Security, Ruhr-University Bochum
[2] Hackmanit GmbH
[3] ecsec GmbH