TOKEN BINDING

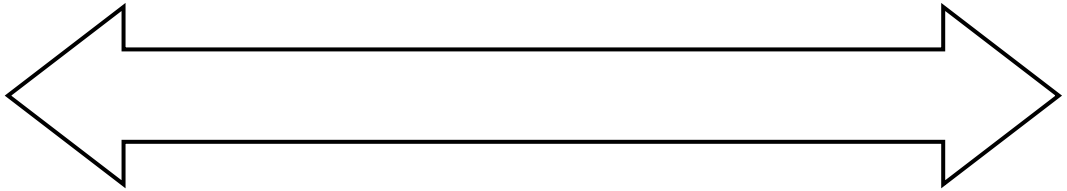
over HTTP

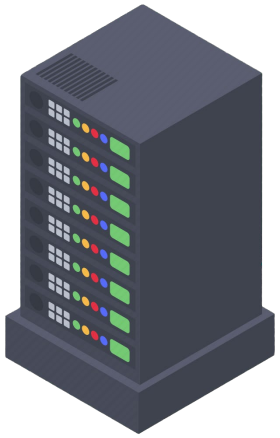
[RFC 8473]

What are tokens?

Token use cases – First-party Scenario

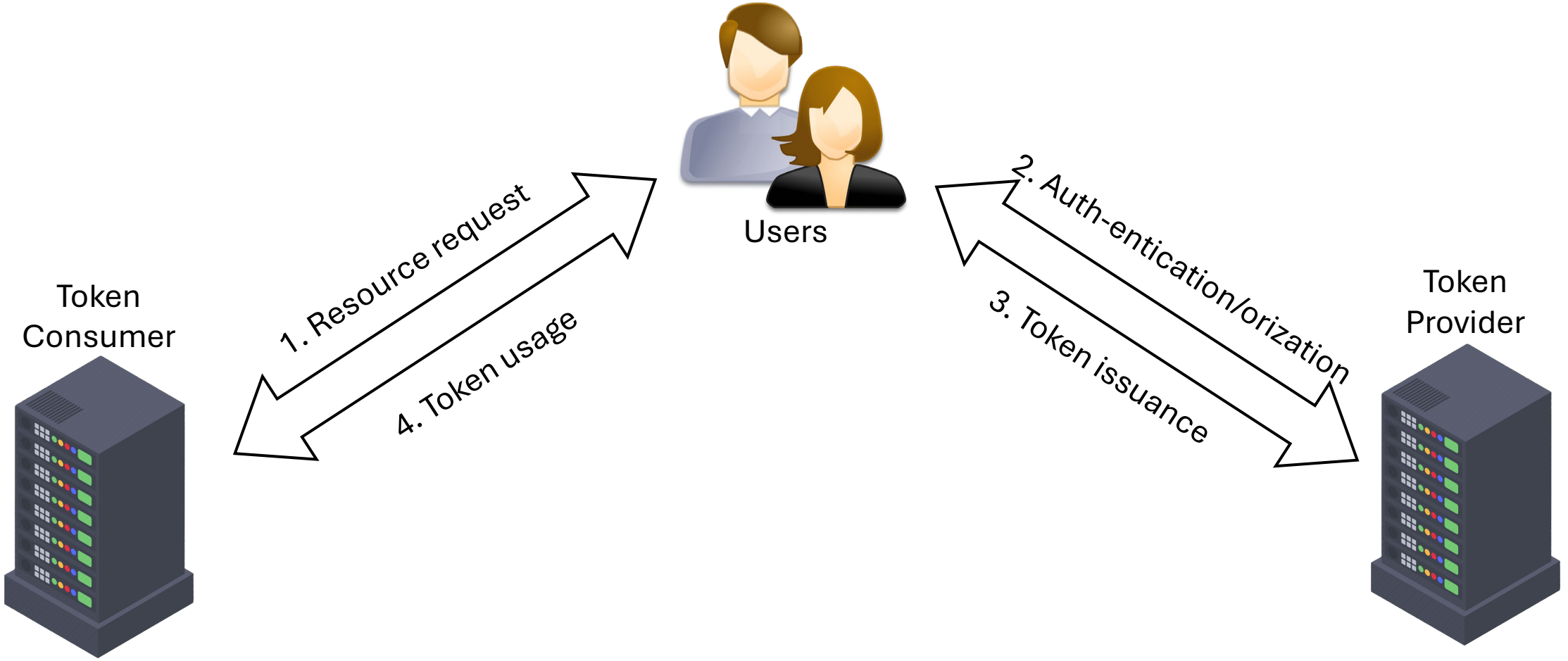
Users





Web Application

Token use cases – Federated Scenario



Token Binding Principles

The proposed solution is to bind the token to the TLS connection between Client and Server

- The client generates a long-lived key pair
- The Client proves possession of the private key by signing the EKM of the TLS connection with the server
- The token contains a reference to the public key used to verify the signature of the EKM

Token Binding

```
Sec-Token-Binding: AIkAAgBBQFzK4_bhAqLDwRQxqJWte33d7hZ0hZWHwk-miKPg4E9fcgs \
                   7gBPoz-9RfuDfN9WCw6keHEw1ZPQMGs9CxpuHm-YAQM_jlaOwwej6a- \
                   cQBGU7CJpUHOvXG4VvjNq8jDsvta9Y8_bPEPj25GgmKiPjhJEtZA6mJ \
                                                      _9SNifLvVBTi7fR9wSAAAA
```

```
struct {
  TokenBindingType tokenbinding_type;
  TokenBindingID tokenbindingid;
  opaque signature<64..2^16-1>;
  TB_Extension extensions<0..2^16-1>;
} TokenBinding;
```

Source: Popov, A., Nystroem, M., Balfanz, D., Ed., Harper, N., and J. Hodges, "Token Binding over HTTP", RFC 8473, DOI 10.17487/RFC8473, October 2018, https://www.rfc-editor.org/info/rfc8473

Bound token

```
{
   "iss": "https://server.example.com",
   "sub": "0f6LkoE3KsPyxQ",
   "aud": "0d8f597e-bc45-46b2-97cf-043c88aa5ecc",
   "iat": 1467151051,
   "exp": 1467151651,
   "nonce": "1KjVsFnQRd4V2XC6",
   "cnf":{
     "tbh": "l1X0aVlpikNqDhaH92VwGgrFdAY0tSackYis1r_-fPo"
    }
}
```

Source: https://openid.net/specs/openid-connect-token-bound-authentication-1_0-03.html#Representation

Usage - First Party

```
GET / HTTP/1.1
Sec-Token-Binding: AIkAAgBBQFzK4_A...
. . .
Username=alice&passwd=whynot
```

```
200 OK
Set-Cookie:
SID=rZcd7FAtOIgJgLGxILM2FrbglFbyh...
. . .
. . .
```

Usage – Federated

```
GET /login HTTP/1.1
Sec-Token-Binding: AIkAAgBBQFzK4_A...
```

```
302 Moved temporarily
Location: https://token.provider/login
Include-Referer-Token-Binding-Id: true
```

Token consumer

```
HTTP/1.1 GET /login
Host: token.provider

headers : {Sec_Token_Binding =
ETBMSG[
[{EKM_CTP, TBID_CTP, provided}KsCTP],
[{EKM_CTP, TBID_CTC, referred}KsCTC]
]}
```

Token provider

Security Considerations

- Bound tokens prevent an attacker from exporting and replaying them
  - Can be replayed though by malware present in User Agent
  - Private keys are needed to export bound tokens
- The bound token needs to be integrity protected
  - Attacker could just remove the binding and reuse it
- Does not prevent collaborating client from sharing bound tokens
  - Either exporting the private key or signing the request for other clients

Security Considerations

- A client should not be tricked into sending a Token Binding Message signed with a key he doesn’t control
  - Possible if the Attacker has knowledge of the EKM
  - A can trick C into logging into A account on S

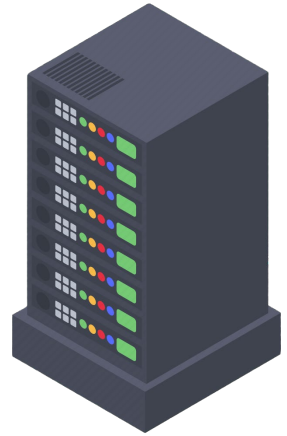
- The Sec-Token-Binding header field should be Browser-controlled
  - Ensures only client-owned keys are included, blocking tampering and impersonation.
  - The "Sec-" prefix prevents cross-origin modification, safeguarding token binding integrity.

Securing Federated Scenario – Assumptions

- The client has an authentication token with the Token Provider
  - bound to the client's Token Binding ID used with that Token Provider.
- The client requested access to some resource to Token Consumer
- A man-in-the-middle is allowed to intercept the connection between the client and the Token Consumer or between the client and the Token Provider (or both).

Federated Scenario - MITM between C and TP

User/Client



Token Provider

TLS_CA

TLS_ATP

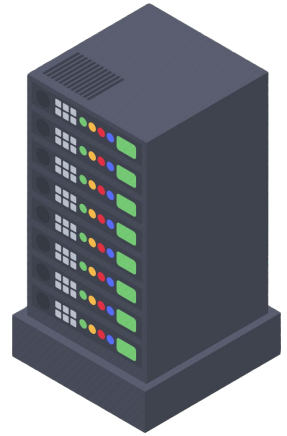
Attacker

```
headers :
{Sec_Token_Binding =
ETBMSG[[{EKM_CA}KsC,
TBID_CTP]]}

token: { tbh :
hash(TBID_CTP)}
```

```
headers :
{Sec_Token_Binding =
ETBMSG[[{EKM_CA}KsC,
TBID_CTP]]}

token: { tbh :
hash(TBID_CTP)}
```

Federated Scenario - MITM between C and TP

User/Client



Token Provider

TLS_CA

TLS_ATP

Attacker

```
headers :
{Sec_Token_Binding =
ETBMSG[[{EKM_CA}KsC,
TBID_CTP]]}

token: { tbh :
hash(TBID_CTP)}
```

```
headers :
{Sec_Token_Binding =
ETBMSG[[{EKM_ATP}KsA,
TBID_ATP]]}

token: {tbh :
hash(TBID_CTP)}
```

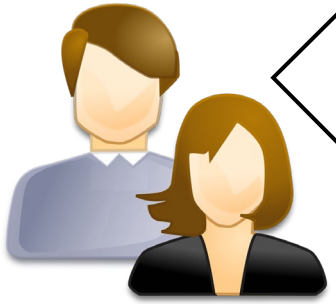
Federated Scenario - MITM between C and TC

Token Consumer

```
302 Moved temporarily
Location:
https://token.provider/author
ize?state=<base64url-encoded-
ClientTokenBindingMessage>
```

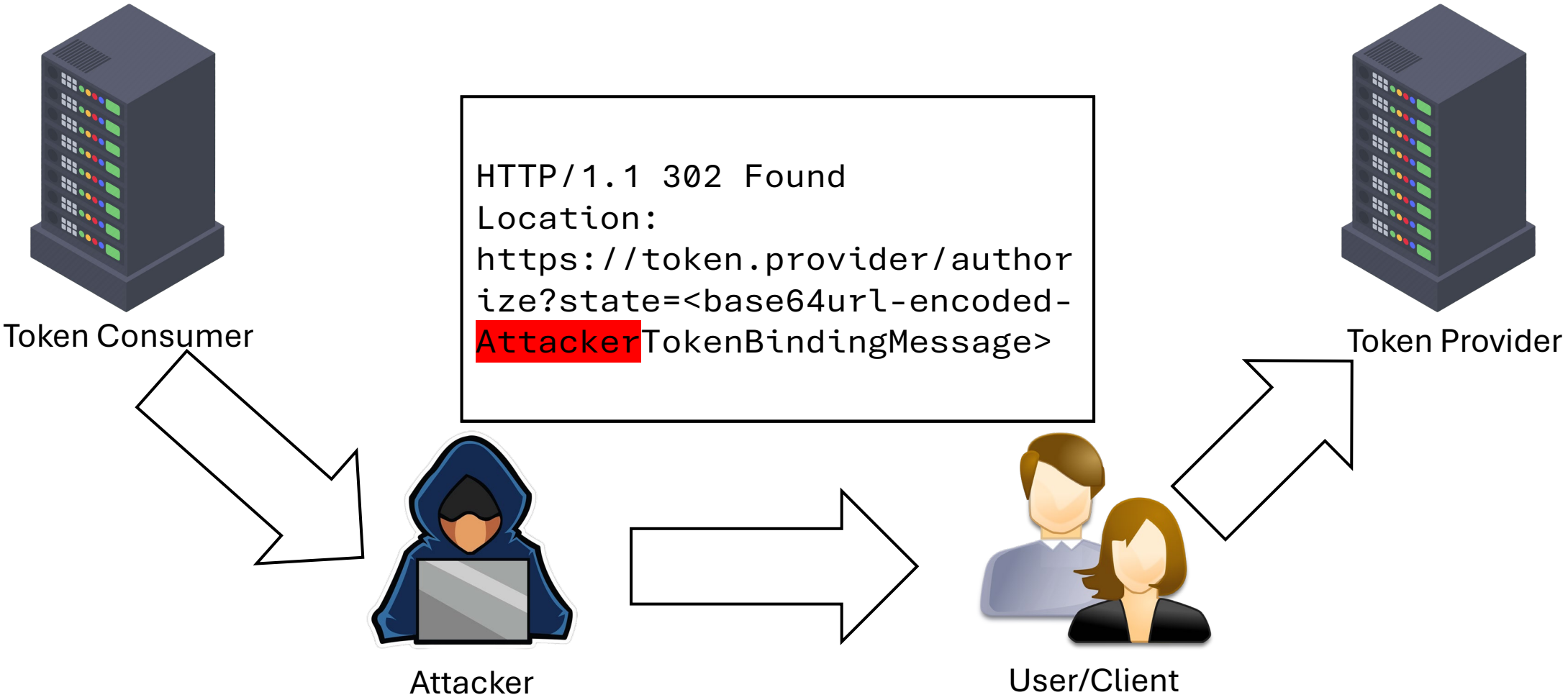
Token Provider

Attacker

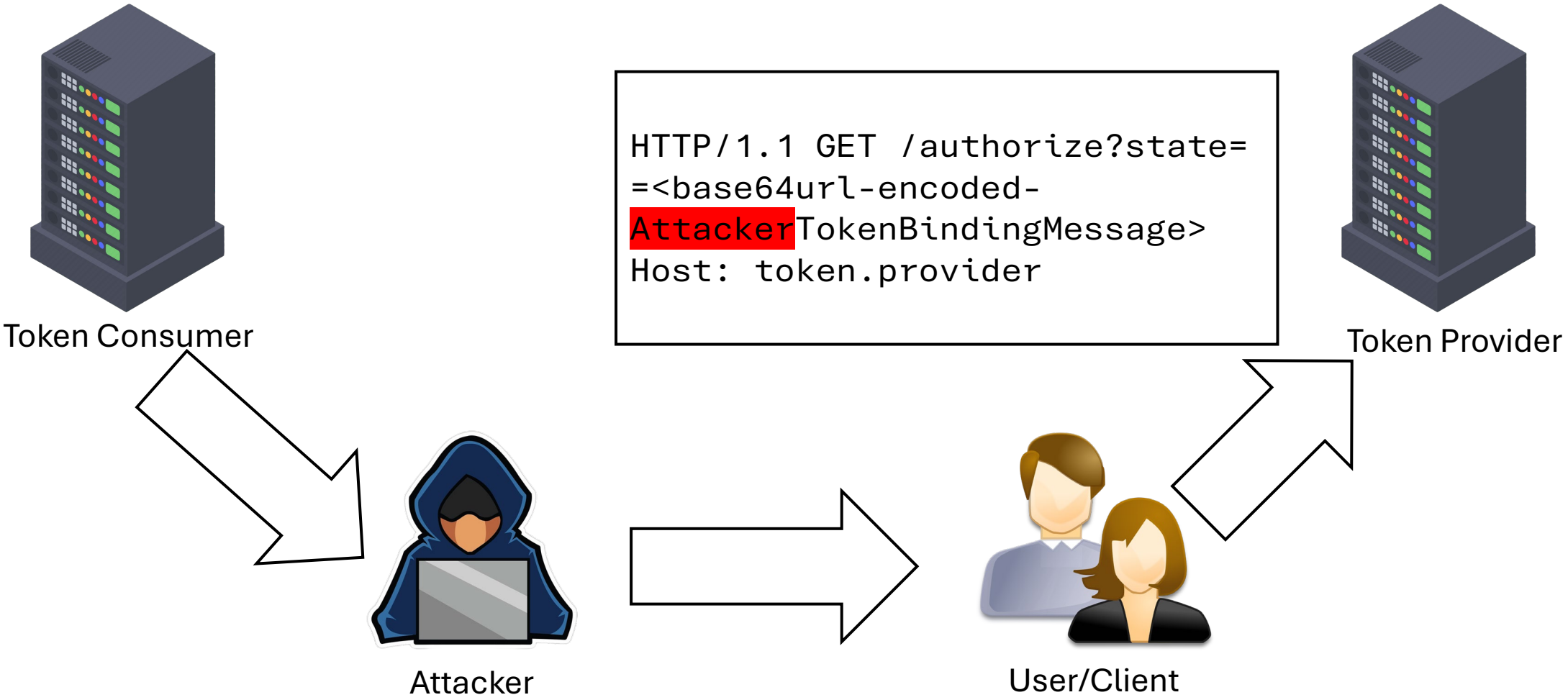


User/Client

Federated Scenario - MITM between C and TC



Federated Scenario - MITM between C and TC

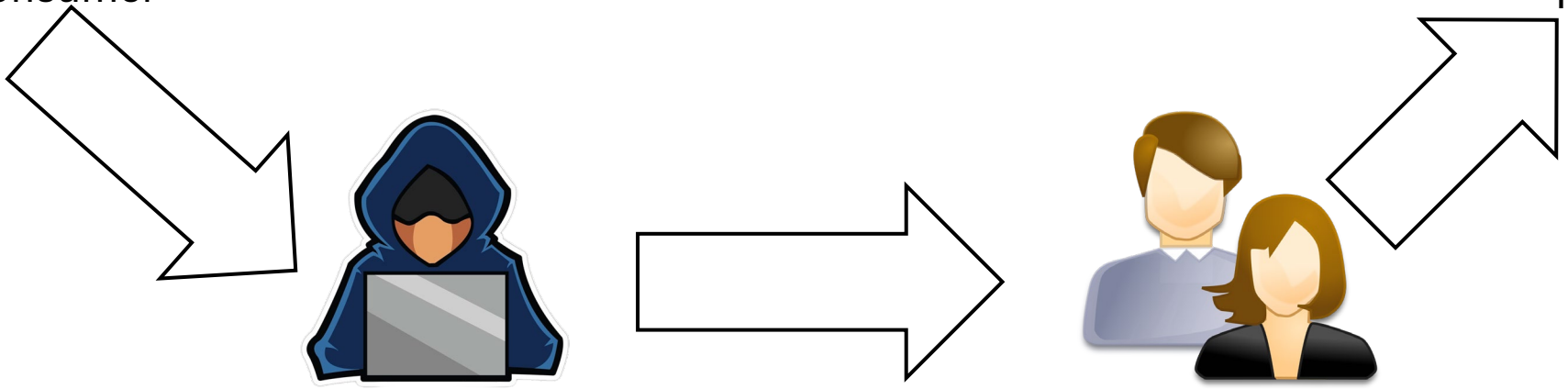


Federated Scenario - MITM between C and TC

Token Consumer

```
302 Moved temporarily
Location:
https://token.provider/authorize
Include-Referer-Token-Binding-Id: true
```

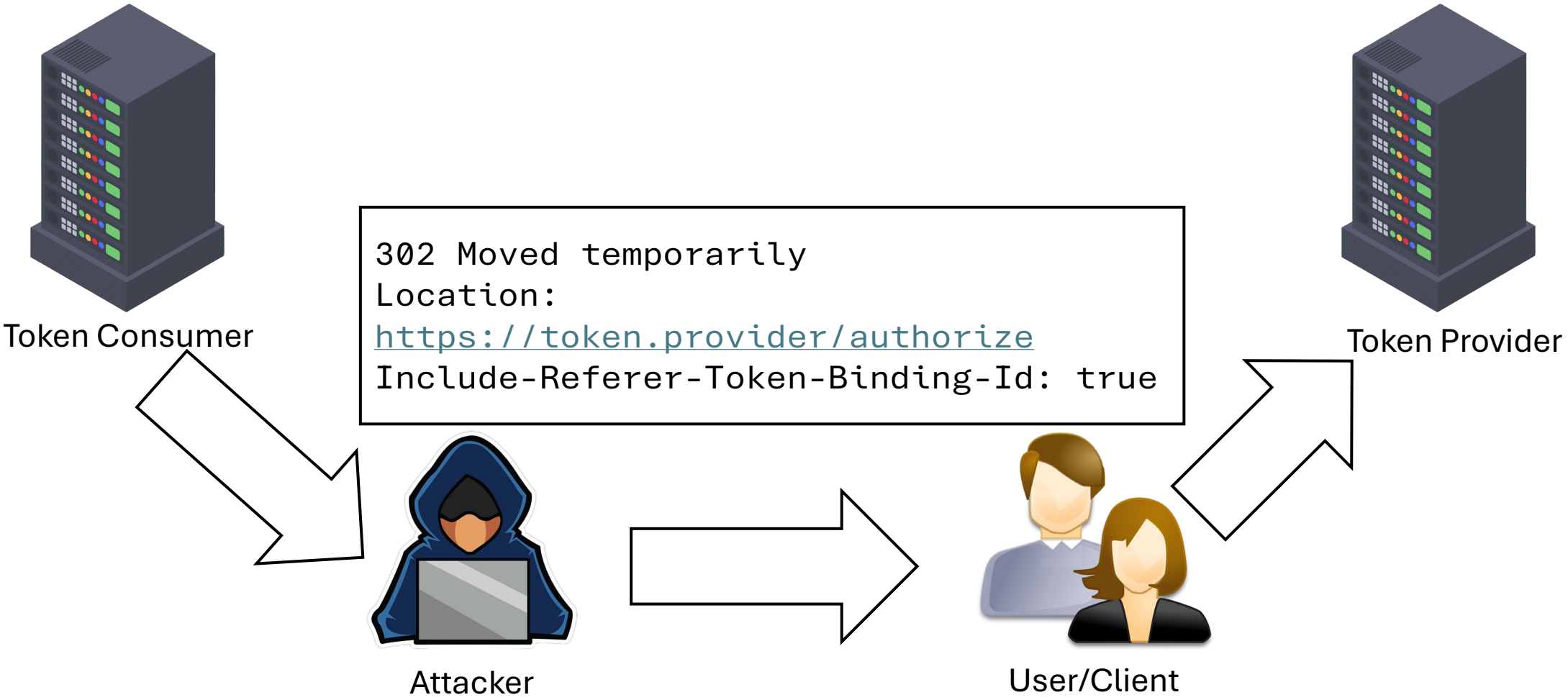
Token Provider



Attacker

User/Client

Federated Scenario - MITM between C and TC



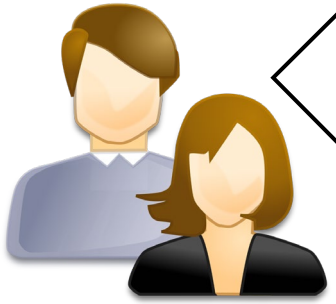
Federated Scenario - MITM between C and TC

Token Consumer

```
HTTP/1.1 GET /authorize
Host: token.provider

headers : {Sec_Token_Binding = ETBMSG[
[{EKM_CTP, TBID_CTP, provided}KsCTP],
[{EKM_CTP, TBID_CTC, referred}KsCTC]
]}
```

Token Provider

Attacker



User/Client

Privacy considerations

- The scope of the Token Binding must not be broader than the scope of the tokens defined in the application protocol
  - As a rule of thumb: one key pair per second level domain

- Token binding key pairs do not have an expiration time
  - Potentially allows for server side tracking
  - Clients should be able to discard key pairs

- Same level of control over the lifetime of Token Binding key pairs as over cookies or other potential tracking mechanisms.

Deployment

- RFC 8471, 8472, 8473 standardized in 2018
- Violates layer separation principles
- Poor intra layer API support
- Today supported only by Microsoft Ecosystem
  - Chrome removed support in 2018
  - Firefox never supported it (tracking)

Any questions?