3DS2 and Strong Auth with PR API

Ian Jacobs, April 2018

Overview

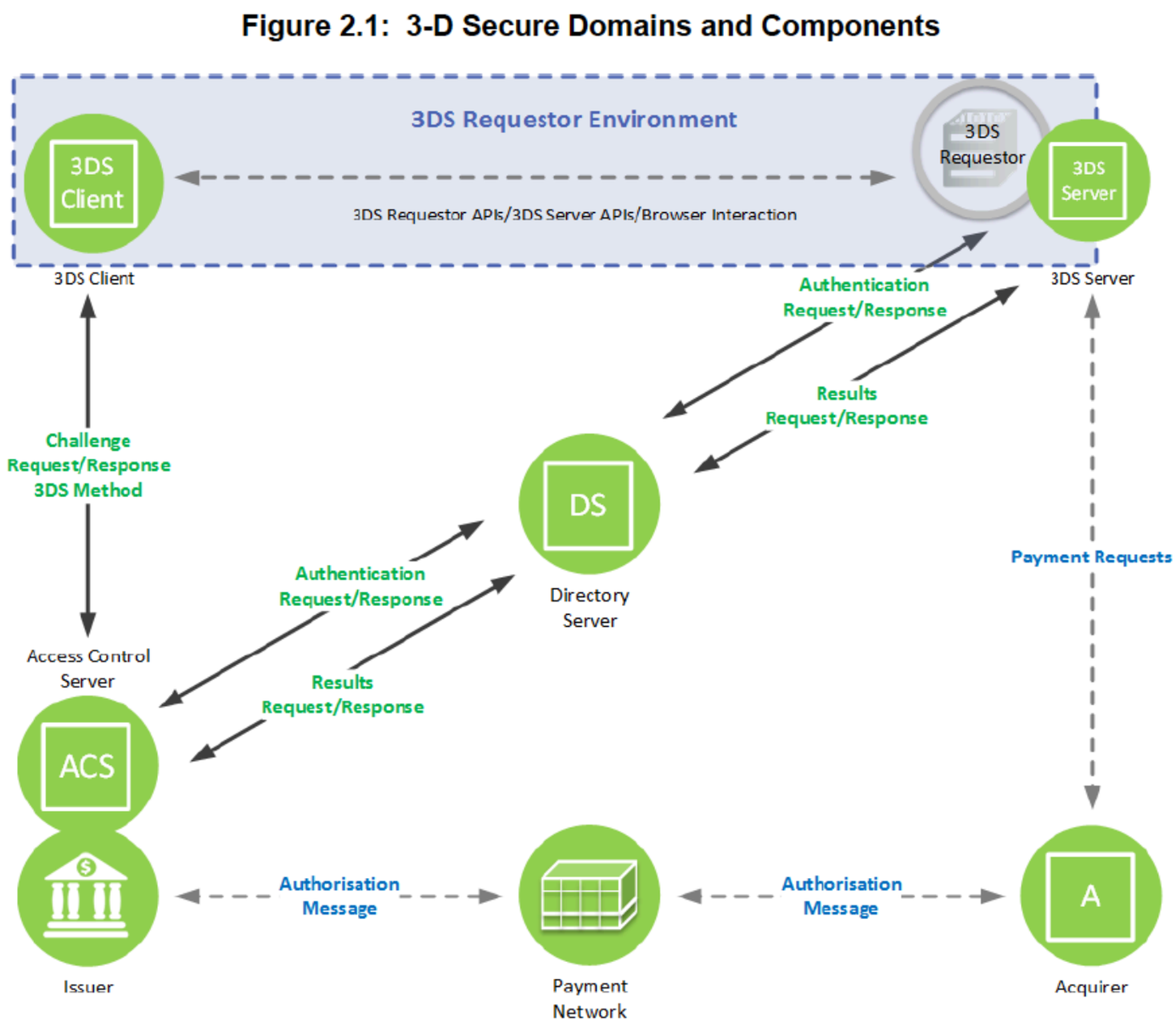
- 3DS2 Summary
- How best to pair 3DS2 as specified with PR API (e.g., for use cases where already required by regulation).
- Identify opportunities with some changes to either 3DS2 or PR API.
- Figure out how this relates to other SCA contexts such as PSD2.

3DS2 summary

- Goals: Reduce fraud, increase approval rates, streamline UX
- Two flows:
    1. **Frictionless**: Risk analysis by issuer using data about transaction, merchant, user. Expected to majority use case.
    2. **Challenge**: When not satisfied, step up to SCA by issuer
- Flows anticipated to start “merchant side” (e.g., from gateway)
- Note: thank this is not a complete summary; for more information see previous presentations to the Working Group and the 3DS FAQ.

3DS2 detail

- Data gathering and SCA performed by JavaScript from user's issuing bank.
- To get that JS, 3DS2 protocol involves chain of trust among backend parties (3DS server, directory server, access control server, issuing bank).



Decoupled 3DS2 & PR API

1. Merchant collects card data from PR API (instead of form)

2. Merchant initiates 3DS2 flows

- Pro: No user action required. Specifications work as-is.

- Con: No easier for merchants.

Browser API for user data

1. If every issuing bank has to publish and maintain the same JavaScript to get data elements about the user environment, can this be done through a standard browser API instead?

- Pro: No user action required. Issuers no longer have to publish and maintain JS. Users could configure browser to prefer **not** to provide data and instead prefer challenge flow.

- Con: Browsers may not want to implement this specifically for 3DS2 (which would require browser vendors to track the evolution of that specification). Probably does not align with 3DS2 as specified.

Issuing bank payment handler

- 3DS2 seeks to connect user to issuing bank. This happens “for free” if user selects an issuing bank payment handler.

- Question: Is this expected to be a common use case?

- Pro: Less integration for merchant. More customer-facing interactions for issuing bank. More trust because user interacting with bank explicitly. Simpler protocol (backend communications no longer necessary) which may improve performance.

- Con: User needs to get issuing bank payment handler.

How this could work

1. Merchant provides data required for risk analysis through PR API (e.g., extra data directly or indirectly through URL accessed by issuing bank).

2. User selects issuing bank payment handler, which initiates risk analysis

3. Payment handler invokes challenge flows if needed (e.g., via Web Authentication).

4. Authentication value returned to merchant via PR API

Questions

- I understand there is a step where the merchant is authenticated (the 3DS server “validates the 3DS requestor”).
- How would this be carried out if flows move to the user side?
  - Would it suffice for the browser or payment handler to know the origin of the merchant?
  - Would it be necessary for the merchant to provide additional information through Payment Request API that user-side entities could use to validate the merchant?

Aggregator payment handler

- User-side entities may wish to provide 3DS services for issuing banks. These entities would be analogous to a gateway on the merchant side (and might be the same companies).

- Question: Is this expected to be a common use case?

- Pro: This begins to sound like user side version 3DS2, so it might be possible to reuse the 3DS2 model. Less integration for merchant.

- Con: User needs to get aggregator payment handler.

Questions

- What if the aggregator is the same as the merchant gateway?

- Not the same?

Browser initiates risk analysis

- Instead of issuing bank calling standard browser API for data elements, browser pushes data to issuing bank's ACS (or 3DS Server?)

- Browser returns result of risk analysis to merchant who can determine whether to initiate challenge flow.

- Pro: Rapid deployment of risk analysis since user does not have to get payment handler. User could configure browser to not do this and prefer challenge flow.

- Con: Browsers may not want to implement this; issuing banks or 3DS servers need to be able to respond to browser calls

Browser sheet as UX for strong authentication

- An **auth()** method in PR API (between **show()** and **complete()**) could be used to integrate issuer-provided UX into the sheet.

- Pro: Consistent UX for strong authentication.

- Con: Requires browsers to implement.

Browser-assisted cache of issuing bank SCA

- While visiting issuing bank site, user is strongly authenticated.

- Issuing bank knows about the user's cards but not which cards the user has stored in the browser. Bank asks browser if user has stored any of the cards in the browser. If so, cards are linked to the origin and authenticated session (some token with associated ambient data such as date and duration).

- When the user selects one such card to pay, the response data includes the origin and token. Origin can be used by merchant's PSP as part of security check. Token is used during authorization.

- Token can be refreshed whenever user visits bank site. Note that no payment handler is involved; just a visit to the bank origin.

- Pro: Avoids UX friction because bank SCA result can be reused multiple times within a period of time defined by the bank. Avoids device fingerprinting.

- Con: User has to visit bank site and be strongly authenticated. No per-transaction risk analysis.

Questions

- PSD2 requires 2-factor authentication, one of which must be transaction specific. Could the browser hash/sign some transaction data including the authentication secret?

- Another option is when the user visits the bank site, instead of storing SCA value, the issuing bank’s handler is registered and associated with the card. When 3DS is required, if user selects card-in-browser, the browser automatically invokes the authorized payment handler.

Card network payment handlers

- Each card network (not issuers!) implements a payment handler. Users install the handler when visiting an issuing bank's site. At transaction time, the handler displays the issuing bank cards for selection.

- At installation time, the issuing bank site is the main window, so it can gather the data it wants. It then wraps that up securely in some way for management by the handler.

- At transaction time, if 3DS is needed, the handler sends the previously-gathered data payload (e.g., token); this avoids the need for JavaScript insertion. In the case of the challenge flow, the handler uses the Open Window Algorithm of the Payment Handler API to show the challenge url sent back by the issuer.

- Pro: Issuers control what data they wish to gather and in a context where they have permission to do so; depends almost completely on existing specifications or browser behaviors.

- Con: Requires card networks to distribute payment handlers. User needs to get payment handler.

Browser-stored cards associated with issuing bank services

- User visits issuing bank site which results in the following associated data stored in the browser: (1) user-added cards (2) data gathered by bank (3) risk analysis endpoint of bank (4) auth endpoint of bank.
    - Endpoints might be managed by card networks or other trusted parties.
    - Question: Could this registration happen at another site (e.g., directory server type site)?
- When user selects browser-stored card to pay:
    - Browser initiates risk analysis flow at risk analysis endpoint using stored data from bank
    - If step-up is required, browser invokes the authentication endpoint of the bank in the still-open sheet.
- Browser returns authentication value along with card (basic or tokenized) in response data.

- Pro: Issuing banks and card networks do not have to issue payment handlers. It might be possible to make this generic and thus applicable to a variety of risk analysis and authentication approaches.
- Con: Requires browsers to call endpoints.

Summary of opportunities

- Decoupled 3DS & PR API
- Standard browser API for user data elements
- Issuing bank payment handler
- Aggregator payment handler
- Browser initiates risk analysis
- Browser integrates strong authentication
- Browser-assisted cache of issuing bank SCA
- Card network payment handlers
- Browser-stored cards associated with issuing bank services

Observations

- To facilitate deployment, identify opportunities that maximally leverage browsers and card networks, to lower the burden on merchants and issuing banks.

- The opportunities listed here are discussion starters.

- Thanks to Adam Solove, Peter Saint-Andre, and Dom Hazael-Massieux for recent discussions about some of these.

General questions

- Which of these makes sense? Is most interesting? If more than one, how would we stage them?

- How does Web Authentication fit in?

- Could integration of strong authentication in the browser also enable consistent handling of other use cases (e.g., credit transfers under PSD2)?

- 3DS2 and Payment Request Api rely on JavaScript. Is there anything to do when JavaScript not supported or turned off?