Strong Customer Authentication: Security Issues and Solution Evaluation

Master of Science in Technology thesis
University of Turku
Department of Future Technologies
Security of Networked Systems
May 2020
Pongku Kumar Paul

Supervisors:
Seppo Virtanen
Antti Hakkala

UNIVERSITY OF TURKU
Department of Future Technologies

PONGKU KUMAR PAUL: Strong Customer Authentication: Security Issues and Solution Evaluation

Type of thesis, 58 p., 0 app. p.
Security of Networked Systems
May 2020

In October 2015 PSD2 first adopted by the European Parliament to initiate a new method of payment system. Since then, it receives several amendment time to time. Strong Customer Authentication (SCA), one of the major requirements of PSD2 came into force from September 2019. However, European Banking Authority EBA found it is challenging to comply with this requirement fully, before the given deadline. Technical implementation challenge, complex payment systems across EU, bring-in all related actors under SCA needs to be resolved with profound solution to achieve the PSD2 success. Moreover, contradictory terms of the PSD2 with GDPR and inadequate protection for the user's privacy prevails account access issues that can be circumvented by the payment service providers. This article investigated the pros and cons of the PSD2, finds feasible solutions for SCA that seamlessly involves all actors in payment system. Despite the fact of technical implementation details, a leading PSP's SCA compliant solution integrated into an e-invoicing system as an specimen of an SCA compliant model. The model showcases the SCA conformity then test and verifies security of data and privacy of the user.

Keywords: Payment Service Directive 2, Strong Customer Authentication, Regulatory Technical Standards, Payment Service Provider, Account Servicing Payment Service Provider, Payment Initiation Service Provider, Account Information Service Provider, SAML2.0, OIDC

Contents

List Of Acronyms

PSD2 Payment Service Directive 2

SCA Strong Customer Authentication

RTS Regulatory Technical Standard

PISP Payment Information Service Provider

AISP Account Information Service Provider

PSP Payment Service Provider

PSU Payment Service User

2FA Two Factor Authentication

EBA European Banking Authority

SAML Security Assertion Markup language

OIDC OpenId Connect

CICD Continuous Integration Continuous Delivery

Chapter 1

Introduction

The recent evolution of the European revised Payment Services Directive (PSD2) [1] comprised of several objectives, has drastically emerged the new perception of redesigning the payment service models. The first payment service directive PSD1 [2] shown exemplary growth in the payments market. Following the significant growth changes, the adoption of PSD2 brought concrete advises accommodating the legal framework to support these developments. The expectations of this framework drive to certainty, competence, and coherence among the financial service providers. The outcome of such conformity ensures fraud protections, client protections, and secured transactions.

The PSD2 framework initiates two new ideas in payment services, called payment initiation services (PISP) and account information services (AISP). Both of these new facilities benefit over banks, previously the sole owner of the client's account information now follows an obligatory protocol to share their holds with third-party service providers free of charge, with the consent of the account holders. The inauguration of this access to personal information demands all related parties in this process to develop and maintain a secure authentication process that reflects the regulatory technical standards (RTS) and common and secure open standards of communication. Seemingly, PSD2 points to the wide development of the payment service market irrespective of security and privacy. This thesis aims to analyze the risks that are created by access to accounts followed by a

successful integration of SCA (Strong Customer Authentication) compliant model, one of the major requirements of the PSD2, into a system that demonstrates how this requirement can be achieved without building an extensive new interface for the businesses. Along the way, we will explore current technologies, as a probable solution that can meet the PSD2 technical requirements listed by the RTS. Viable recent open-source authentication solutions exist in the market and already trusted by many leading PSPs will be investigated thoroughly for reliable SCA implementation. Relatively, the business impacts might cause due to the new regulation in action, which will be justified by analyzing the stakeholder's opinions.

Chapter 2

Access to accounts: PSD2

PSD2 brought businesses advantages for Fintech companies that are dependent on some considerations. The companies should extend their existing payment systems and products in accordance with the new regulations. However, providing the services largely depends on the cooperation of the banks as the sole owner of all account information. Fortunately, PSD2 intervened with such strategies of limiting the market competition by facilitating payment service access to the payment initiation service providers and account information service providers.

Regardless of the benefits this third party access also resembles security and privacy risks. Since the user information e.g. transaction history, personal data, account information will be exposed, there are a significant amount of chances of abusing these pieces of information, for example blackmailing, identity theft, selling in black market or brokers. As a counter-work, categories of payment service providers listed under Article 1(1) of the PSD2 to minimize the risks of probable frauds. This list authorizes national central banks, member states and credit institutions or electronic money institutions (under a, b). Additionally, any organization that would like to provide payment services throughout the European Union, can obtain such rights having authorization from a competent national authority. On the other hand, companies are interested to provide account information services only, are exempted from required authorization (Article 33) except to apply for a

registration [1, p. 78].

As account servicing payment service providers, the banks receive fewer benefits fulfilling PSD2 requirements. The obligation of providing access to accounts to its users obligated as free of charge. Traditional banking systems are incapable of handling these matters at this moment. Therefore, upgrading current systems, and developing secure access could have costly effects. Top of that Article 73(2) says that any unauthorized or defective transactions even if it is initiated through payment initiation service provider, shall be refunded by the bank. In addition to that, now the bank's new competitors will get control over their accounts information, which leads to reduced interaction with customers. As a consequence, the bank might face a hard time selling its own products and services to customers. Whereas, Temenos expecting banks can also benefit from third-party innovative services developed from an agile mindset. Traditional business models and rigid operating frameworks can be overridden partnering legal service providers with reduced costs, more efficiency, replacing nonstandard ways of accessing user accounts (screen scraping). Value-added services, products can be developed to provide personalized services to the user through application store and smart devices e.g. smartphone, wearables. Interestingly, The banks will have the opportunities to facilitate interaction with clients from other banks by registering themselves as a service provider, which can be seen as multidimensional business scope. [3]

Gartner provided a more specific figure of probable annual increase of revenues for banks, approximately 30%. The strategy of the Programmable Business Model is expected to thrive over a decade that will let both internal and external users innovate and improve new business models with the aid of technologies. The user institutions can leverage the power of shared access, previously owned by banks only. Consequently, the banks can have the opportunities of generating multiplied revenues, evolving competent third parties capable of offering banking services straight to the customers. [4, p. 8]

2.1 Payment initiation service

PSD2 clearly defined the role of Payment information service (PIS) provider in Article 4(15) as "a service to initiate a payment order at the request of the payment service user with respect to a payment account held at another payment service provider". Namely, Sofort, iDEAL and Trustly are some of the giants, accordingly from Netherlands, Germany and Sweden, in payments industries performing PIS roles for a while. Services offered online, requires a secure bridge between the client and the merchant, that the payments are initiating from reliable sources. PSD recital 29, says this secured transactions ensures merchant to deliver the goods or services to the customer without any undue delay. Note that, bank must able to distinguish the properties of an payment order to locate it's origin. A separate interface handling orders from PIS providers should understand the difference among all required properties without any overlap. PIS provider service facility fails if it doesn't get processed by bank's dedicated interface.

PIS providers access to client accounts are regulated into Article 66 of PSD2 elaborately. PIS is noted as payers right to receive any online payment services. If services offered not available online dismisses the right to use PIS. In accordance with Article 64, when a payer gives explicit consent for payment services to be executed by a PIS, the bank must respect it to ensure payer's right. Paragraph 3(Article 66) make sure a PIS provider hold no rights to retain payer's funds, and can't save sensitive payments data for later use in addition to some other guidelines regarding it's services. Additionally, PIS are provisioned to request only the necessary data to the corresponding payments. The bank must make available all information related to the initiated payments and execution of them immediately on state change to the PIS provider. However, neither the bank nor the payment service provider requires to sign any agreement to process such payment requests. It is independent of any contractual business relationship between them, where the banks are not allowed to charge any fees for the services.

2.2 Account information service

Paragraph (27) of the decisions noted under PSD2, after the Directive 2007/64/EC, drastic change in financial market evolves a demand of new types of complementary online payment services that provide the payment service users required information of all accounts held at other payment service providers. The access should be made through the online interface provided by the corresponding account servicing payment service provider. PSD2 advises the security of consumer's accounts and transaction information enabling them to act under certain guidelines.

Primary vision of such service development indicates to receive collective information of several accounts held on different premises with single interface. A typical example of such intention includes count on user's average spending per month on different categories of products to offer customized service or advertisement. Also, interested financial advisors and credit reference agencies can be a part of this list as well.

Since the services can be accessed online only, Article 67 states regulation regarding the access which are similar to the payment initiation services guideline. Even though they share similar access protocols, deep analysis of account information service reveals disagreement with the regulations. Say for example, bullet point 2(f) of article 67 advised data cannot be stored, access or use other than the purpose of being used in account information service process, with the consent of the service user and data protection guidelines GDPR. Third party access to the data through account information service is a legal action. Formatted product offers are possible to create based on the data received by using this service. [5, p. 32]

Respecting the regulation, payment information obtained from the account information service without storing, should be good enough to use for personalized product marketing. The workaround will evolve proportionately once the open banking facilities are fully implemented, and available on demand. Moreover, with the consent of the user it is not clear whether the account information service provider is eligible to process the

data for other purposes. Point 2(f) of Article 67, advised data can be processed for other grounds as long as it respect to the General Data Protection Rules(GDPR), which partly disagree with the point 3(g) of article 66 "not use, access or store any data for purposes other than for the provision of the payment initiation service as explicitly requested by the payer". Unfortunately, no specification of user data found that access can be divided into allowed and prohibited classes for account information services. Regulation says no other information should revealed apart from the provision of corresponding payment service which is identical to the data minimisation principle of GDPR (Article 5(1)). Whereas, required information for the corresponding payment is dependent on the type of services offered.

The PSD2 aims to facilitate the account information services by registered service providers than only the bank. To satisfy such intention, limited access to the account data is a barrier that the insufficient access of data assigned to the service providers would leave them in a disadvantaged point compared to the sole owners e.g. banks. Therefore, assigning same rights as the user of online banking services to the account information service providers are vital to keep rolling the growth of financial market and to achieve PSD2's foreseen advantages. In the context of a payment transaction, usually account information service is involved with two users. In any event initiated by user1 to user2, including the data processing of user1, the service provider also process data of user2 without any explicit consent given by user2. There is no exception in PSD2 as well regarding this matter. The processing of this data should be based on the GDPR guidelines to protect the user2's privacy. Frequent use of account information services from user1 to user2, leave a risk of unveiling secret financial state of user2 to the service providers. In such cases, the silent users (user2) information can be put in large scale privacy risk. For that, It is important to control the extent of access over the data only in provision to the service asked for. No later use or forward to other interested parties to build a personalised profile shall be prohibited. [5, p. 33-34]

2.3 Secured access and protection of the service user

Security and privacy are priorities on the way of successful PSD2 implementation. A bank has the right to accept or deny access requests to any account for justified and evidenced reasons as long as they fall under the criterion of fraudulent or unauthorised. Point 5,6 of article 68 PSD2 highlights this issues in the context of service users privacy. In case of access rejected shall be informed to the user that the access is denied on the basis of terms and condition agreed in the form. It is also advised to inform the payer as soon as the access is denied or if possible beforehand, unless no objectively justified reasons related to data security or objection from any Union or national law prevent so. Details of the incident shall be reported to the competent authority whether it is related to the account information service provider or payment initiation service provider. The competent authority can investigate the relevant details of the reported issue and take measures accordingly if requires.

About the data protection, article 94 states that, a member state shall allow processing of personal data by the payment service providers and payment systems to facilitate any investigation, prevention and detection of fraud payment activities. Service providers are eligible to store, process and access personal data only with the explicit consent of the service user in provision of the requested payment services. Following this, article 95 prescribes to manage all operational and security risks implementing proper framework including sufficient mitigation measures and control mechanisms that supports detection and classification of security incidents belongs to operational risks.

Continuing on users privacy, article 97 says, the service providers are bound to use "strong customer authentication" for both account information service and payment initiation service to dynamically link the transaction to the amount and the payee. The personalized security credentials of the service user shall remain confidential and integrity shall remain intact regardless of the payment initiated through a payment initiation service provider or not. The distinguishable elements of strong customer are authentication

knowledge, possession and inherence, elaborately something unique user knows, something that only the user possess and something the user is. Each of the properties are independent of others that the breach of one element shouldn't affect the credibility of the others. The definition of the strong customer authentication in article 4 clause 30 provides a detailed information.

Account information service providers are considered with comparatively less precaution compared to the other payment initiation service providers. Pursuant to article 33 clause 2, account information service providers that do not process any payment services does not belongs to the rules mentioned in article 94. However, these providers are still imposed with the data protection law of GDPR as long as they process personal user data. Therefore, this exception relates only to the clause 2 of article 94. Lack of elegant data protection rules and the consent received from user are sometimes controversial in a sense that, users often agree on processing personal data without truly understanding the terms and condition, even though it is manifested in plain form. The competent authorities of all member states shall provide adequate guidelines on colliding issues of PSD2 to facilitate the implementation process of PSD2 nationally. [5]

Chapter 3

Strong customer authentication and regulatory technical standards (RTS)

Security should be the first concern for all payment service providers when offering services online. Communication between the service providers and the receivers should take place in a secure manner that ensure secured authentication, and alleviates maximum possibilities of fraud. Transaction observation mechanism should be strong enough to detect any inappropriate use of user's personalised security credentials in case of, for example, lost, stolen and should also assure the corresponding service user's legitimacy where the permission is granted consciously to access its account information and initiate any transfer of funds. Additionally, strong customer authentication is necessary to to be applied as long as the authentication doesn't meet the requirements, no matter how many times a payer seek access to its payment account, start any transaction, or perform any action through a remote channel that carries a risk of fraud or abuse by fulfilling the standard procedure of authentication code/token generation. The generated authentication code/-token must be preventive even if it is forged entirely or revealed any of it's constitutive elements. The specification of strong customer authentication should be known by the user, requesting such services. [6]

Article 66 of PSD2 stipulates, payment service requests initiated by service users through payment initiation service providers must authorise the access using strong customer authentication. The formal definition of strong customer authentication (article 4, point 30) requires presence of at least two or more unique independent elements a user possesses on the process of authenticating a payment such as knowledge, possession and inherence. Since, the breach of one element shouldn't affect others, it is important to keep separate the use of personalized security credentials and the independent elements. Imperatively, legislative decision point 107, specifically nominated EBA (European Banking Authority) to draft regulatory technical standards for strong customer authentication with its expertise and knowledge for reliable, secured communication between parties involved in a payment service. It is also advised that the draft should be developed in cooperation with the ECB (European Central Bank) with aggregated consent of the other stakeholders.

RTS advised, it is possible to get exemption from SCA but contingent to couple of security catalysts level of risk, recurrence, amount and the channel used to execute the payment. The use of SCA also loose coupled for actions that pose low level risks, accessing balance and recent transactions maintaining the due confidentiality, previously setup or approved recurring payments through the use of SCA addressed to the same payee, and associated identical payer, payee and the PSP through former transactions. In above scenarios, the PISPs, PSPs rely on card based payment instrument and AISPs should request only the compulsory information from the ASPSP for the provision of any particular payment service consented by the service user. Contactless small amount payments at the point of sale, with a certain number of successive payment limits or limited maximum values of transaction can be executed without SCA to promote user friendly and low risk payment services. Unattended terminals and the end points where queues happen, should also be exempted from applying SCA due to the increased time of completing the process, operational, safety and relevant security risks. For example, applying SCA in toll gates, unattended ticket machine could results a in an underestimated long queue. How-

ever, RTS ascertain that if the exempted value of contactless payment doesn't set to a perfect threshold balance between the rising needs of augmented security for remote payments and the necessity of user friendly and practically accessible payments, the area of e-commerce can be affected. The threshold balance limit for low-value online purchases, should be considered in a sensible manner keeping in mind that, the execution of the payment is not physical with any payment instrument. Thus, the risks of fraud might be a little higher in online purchase than the physical purchase. RTS promotes the adoption of effective and risk based requirements for PSPs not interested on applying SCA in case of real time low risk transactions, which confirms the safety of sensible user data and funds, are eligible to get exemption. This risk based analysis should comprised of analysis of abnormal spending behaviour of the user, other risk factors, for instance, monetary based remote payment fraud rate threshold, calculated based on the location of payer and payee. If any payment request failed to pass this combined score of risk based analysis threshold, the PSP shall revert to SCA. Before confirming the maximum exempted risk based value for a PSP, average of all the fraud rates of all PSPs together with those went through SCA, within a certain period should take into account on the current basis. [6]

Article 10(2) of RTS, obliged PSPs to must apply SCA, when a PSU seek account information e.g. balance, transaction information for the first time and when more than 90 days elapsed since the last time the PSU was authenticated for processing any transaction request using SCA. Similarly, more specific restriction announced regarding the amount of contactless electronic payment initiated by the user. No more than EUR 50 is allowed for an individual single payment. In total, the cumulative amount of EUR 150 should not exceed or the total number of consecutive contactless transaction must be equal or below 5 from the date of last time the SCA applied.

3.1 Regulatory Tehnical Standards (RTS)

On 27 November 2017, finally RTS was accepted by European Commission after several contention about screen scraping authentication process with the EBA. Eventually, the discussion found a mutual point which released RTS on March 2018 that would be in effect from September 2019. Inclusion of RTS would complement the goals of PSD2. Article 115 given a deadline that the PSD2 must take effect from 13 January 2018. Since then, payment service providers are obliged to receive the benefit of account access. However, under article 115(5), service providers were active before January 2016 in a member state are permitted to continue their operation. This also brought a benefit to them that they are exempted from the security measures as long as the RTS is not in effect. Additionally, article 115(4) given them the freedom of applying the security measures (article 65, 66,67,97) are not mandatory during this transitional period, rather they are obliged to manage their operational and security risks to protect personal data.

3.2 Access Interfaces

Communication between the payer and the payee must have reliable security mechanism implemented, which is one of the payment service providers vital responsibilities. Any miscommunication through the use of smartphone application from an authorized user to unauthorised user or to any other service user's interface should be identified with proper risk mitigation. As a precaution, sessions linked between the service providers and users or any other entities can rely on unique session identifier, detailed transaction information logging and timestamps from a unified system. One ways out of many, such reliable system services can be ensured using a dedicated access interface.

Article 30 of RTS stipulates the service providers shall enforce a minimum of one interface that satisfy a few critical features. Firstly, the interface includes the ability to challenge any payment instrument issued based on card to accomplish identification

towards the account servicing payment service provider. Next, account information requests for authorised payment accounts or transactions from account information service providers maintain the integrity throughout the request life-cycle, and finally, the interface should secure the payment initiation request issued on behalf of the corresponding authorised user accounts with no unauthorised interruption while sending and receiving data and provide adequate accessible information regarding the execution of the transaction and account servicing payment service providers. RTS advised that, the interface shall build on the standards defined by any international or European standardisation organisations. Once the interface built, required documentation can be requested by the payment initiation service provider and account information service provider without any charge. Additionally, the interface facilitate testing including customer support, that payment service users will be able to test their corresponding application and software needed to offer services to user. Account servicing payment service providers will be under surveillance of competent authorities whether the interface is abiding by the obliged regulations.

According to article 31 of RTS account servicing payment service providers are given choices to implement either a dedicated interface or the user interface used for their clients. In case of second option chosen, still the service providers are not allowed to open the user interface to the payment initiation service users and account information service providers straight. To request any account and designated transaction information, the interface must identify and authenticate them granting access, associated with the corresponding payment service.

3.3 Consequences of ‘screen scraping’

To provide PSD2 compliant services, the service providers need access to the dedicated interface enabled by the bank. They are solely dependent on the bank’s functional interface to bridge the clients. Even though RTS enacts several preventive rules, this control

is considered a possibility that empowers the bank to frustrate the desired market growth for payment services. Say, for example, limiting the bandwidth and the availability of the interface can increase tension on the market. As stated above in section 3.1, RTS forced banks to maintain, update and make available all relevant technical specifications, testing facilities and any changes regarding communication to the interface.

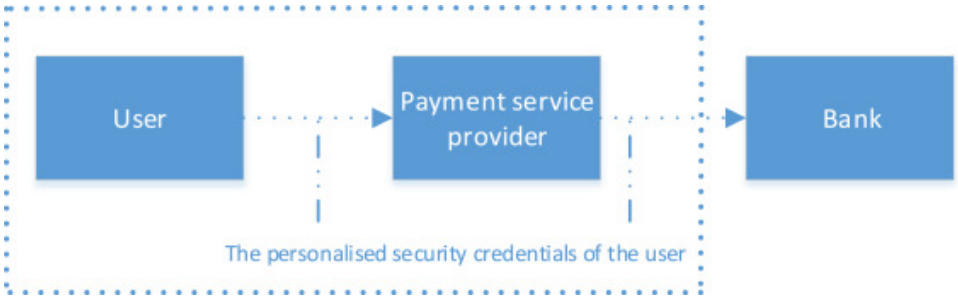

Figure 3.3.1: Screen Scraping [5]

Considering the facts of influence, the acceptance of screen scraping affects the content of RTS. Screen scrapping is straight access to the user accounts pretending to be a user itself, which totally obscures the genuine identity of the party gained access to the account. Alternatively, credentials belong to the user such as PIN, received to manage the access to the interface. Most of the biggest payment service providers before PSD2, for example, SOFORT from Germany, was providing financial services to their users through screen scraping.

Georg Schardt, Managing Director of SOFORT GmbH, said: "Screen or web scraping is the simply technology of machine-reading websites, an omnipresent and indispensable technology for finding information on the internet, used for example for price-comparison websites, online portals and search-engines such as Google". He emphasized, the technology is safe to be used, facilitate sharing customer data with third-party payment service providers, that leading fin-tech company and the personalized AIS solutions of bank trust in this form of authentication over a decade. The idea of outlawing the screen scrapping in PSD2 is dismissed on his logic. If the omnipresent technology of reading websites is

inhibited, it might be said screen scrapping is now outlawed. But the reality is different. He mentioned "EBA itself explicitly says in Art. 27(2) draft EBA RTS that using the online-banking websites constitutes one option to provide a PSD2-compliant interface – this is screen scraping, as the content of the online-banking websites would be read by the TPP software". Therefore, the ban on screen scrapping is confusing and pointless. What EBA actually meant by this is, third party providers are now obliged to identify themselves towards the bank rather than just pretending to be a user, he added. Consequently, if third party providers are able to identify themselves to the bank, it is normal to use screen scraping and machine-read websites which is identical to the draft RTS in article 27(2). When asked about the ban on screen scraping and possible effects, he is optimistic about having a combination of screen scraping and identification. On the other hand, he fears, EBA is trying to impose some political decision, allowing the bank to open up a dedicated interface or direct access, assigning bank the role of a gatekeeper. Additionally, free data portability is a direct contrast of GDPR and limit innovative approach in the fragmented banking industry, in his opinion. [7]

The problem with the screen scrapping is, third party providers manage full access to the user account without identifying themselves. For any initiated communication to the bank, initially, it is unknown who is the other party that the bank is going to deal with. The party on the other side could be a service provider requesting on behalf of a user or it could be the user itself. Apparently, sensitive account information could be at risk of revealed to the unauthorized party. Which concerned the EBA and similar to Georg Schardt's opinion about screen scrapping later on draft RTS release. Article 31 of RTS, allowed banks to implement either a dedicated interface or the interface used to authenticate their own users [6]. In both cases, service providers must identify themselves towards the interface. Many payment service providers also lobbied against the ban of screen scraping on the basis of considering it as a back-up option, since the draft RTS mentioned nothing about it, but the stipulation on performance and availability of dedicated interface expected to be same

as their own user interface.[1]. However, the contingency measures provided on article 33 of RTS, prescribed that unplanned unavailability or a system breakdown shall allow the PSPs to use the interface available for their own users of the ASPSP until the dedicated interface is restored to the fully operational phase. The ASPSP must identify the PSPs and trust the authentication procedure of users provided by the ASPSP.

Kelly, operations manager of a renowned company expressed her concern on the weakness of security and latency of this access method. Corporate access which requires access to a large pool of data might take even 10 minutes to complete the retrieval. Since the tools are dependent on the portals intended to be used by customers, minor changes in portals might break the operation causing significant loss to the dependent businesses. However, she believes the widespread use of screen scraping apparently adopted because of no alternatives standard exists before. [8]

Literally, EBA put dispute against this fall-back option. Their viewpoint was, this might lead to abandoning the development of a dedicated interface completely, opting for the PSD2 compliant user interface. Development and maintenance of an additional interface concurrently with their own, would impose an additional burden and increased cost to the ASPSPs, they believe. It is disadvantageous to the PSD2 objective "to standardise access across the EU Member States and create a single EU payments market". Instead of the development of an identical communication standard across Europe, upcoming AISPs and PISPs will have to deal with a plethora of different customer-facing interfaces. Apart from that, EBA tried to present a substantial amount of argument that the technical abilities of the fallback interface wouldn't be much feasible than the dedicated interface. Top of that, EBA also questions the compatibility of the security requirements mentioned in PSD2 with the fallback interface. [9]

The commission retained the fallback option valid in the executable RTS. However,

[1] https://www.finextra.com/newsarticle/30772/eba-rejects-commission-amendments-on-screen-scraping-under-psd2

an ASPSP can get an exemption from this requirement under certain conditions met. The competent authorities can grant an exemption pursuant to Article 33 point 6 if the interface satisfies the obligations of article 32, designed and tested in accordance with the article 30 point 5, has been in operation for a minimum of three months by payment service providers, and any seeming problem is resolved immediately. Nevertheless, under no circumstances, the exempted banks are permitted anymore to use the method of screen scraping/user interface, since the transition period (Until September 2019) is over now.

3.4 Authentication provided by the ASPSP

Pursuant to article 30 point 2 of RTS, "For the purposes of authentication of the payment service user, the interface referred to in paragraph 1 shall allow account information service providers and payment initiation service providers to rely on all the authentication procedures provided by the account servicing payment service provider to the payment service user" [6]. So, the payment service providers do not need to develop any customized authentication mechanism to offer their services. The service providers can request the banks either including the payment order or seek account information to initiate the procedure to authenticate the corresponding user. There is no clear information provided in RTS that how this authentication mechanism can be implemented.

Two approaches can be considered to implement such an interface. In the first approach, which basically uses the bank's own client interface, the steps are

1. User creates a payment order through a payment service provider to the bank
2. Bank start the authentication procedure and request for information to the payment service provider
3. Payment service providers collect required information or personalized security credentials from the user
4. Bank received the credential from the service provider and given decision

In the second approach, the authentication can be resolved independently of the service provider. The steps are

1. User creates a payment order through a payment service provider to the bank
2. Bank starts the authentication procedure and request for information to the user
3. User provide necessary information as the bank ask for the purpose of authentication
4. Authentication decision redirects to the payment service provider

Since PSD2 discourages using the user's personal credentials and corresponding interface for the authentication, we would like to keep the discussion rolling on the second or decoupled approach of authentication. Regardless of types, the interface should manage an active session among the participants of an authentication request. The service provider will redirect the user to the bank, then the bank resolves the identity of the user independent of the payment service provider. This approach doesn't require the personalized credentials of the user as it is prohibited in PSD2. But, in any case, the payment service providers are not allowed to request, store and process the user's personal credentials. Article 5(1), sub-point (g) of the PSD2 made it compulsory to provide the description of the process in use to control access to the confidential account information.

Compared to the first approach of an interface implementation, the second approach imposes a minimum amount of fraud risks. No personalized credentials ever use in the process of authentication, therefore, no risk of abusing them by the PSPs. Since the confidentiality and integrity of the user credentials should remain intact, those must not be exposed to any employee of the PSPs. PSPs are bound to inform the user immediately as soon as any loss of confidentially happens in their capacity. Banks often warn their clients, never share login credentials even if it is asked by any staff of the bank. Else, any corrupted person would seduce the users pretending to be a PSP to share their private credentials and compromise the security.

3.5 Authentication provided by the PSP

PSP and AISP are bound to use strong customer authentication. However, instead of implementing own procedure, they can rely or not on the method provided by the ASPSP. PSD2 and RTS do not force them to use the procedure offered by the bank. They are open to use any other form of reliable two factor authentication as long as it serves the purpose securely. The PSP and the ASPSP authentication procedures are inspected under the same security standards defined by the PSD2 and RTS. Introducing any other form of authentication procedure might initiate some other form of additional unprecedented risks. Rather than using separate authentication procedure, using one of them, preferably from ASPSP might expose minimum risk of circumventing the system by criminals. Using both procedures of the PSP and the ASPSP, despite the fact of manipulating strong authentication for PSP similar to the ASPSP, can lead to a multiplied attack vectors. An ASPSP wouldn't be able to identify whether the user is different than the original owner, unless they use digital signature including their authentication mechanism. Verifying the consent given by the user original or fake to process the authorisation to a PSP might be in doubt. Consequently, the dependency for the verification shifts to a PSP then, which wouldn't be an ideal reflection of PSD2 regulations at all. Regardless of any authorisation consent given by the user, the PSPs are permitted in PSD2, to access the content of the user accounts in ASPSP. This indicates a risk of reveling user transaction information to unauthorised third parties, say for example, an incident of compromised PSP computer system which provides unlimited access to the user accounts.

Chapter 4

Identification and authentication management

As PSD2 demands the interaction of the payment service provider during the process of authentication, it is hard to find such technology as a pack that satisfy such triangular authentication needs. Precisely, there is no federated solution yet available. The solution must ought to protect the identity of the user and associated parties in the process of authentication. Moreover, article 4 of the RTS prescribed the standard an authentication procedure must comprised of. The authentication code generated using the elements categorized as knowledge, possession and inherence, will be accepted only once by the PSP to initiate transaction, grant online access to accounts, or to perform any other actions take place through a remote channel that are vulnerable to the risk of fraud. To accomplish this intention PSPs are advised to develop and deploy security measures, that must satisfy a few important requirements. First, disclosure of the authentication code must not reveal any confidential information regarding the user and the transaction. Second, previously generated authentication code and any knowledge extracted from that can not be reused to generate a new authentication code. Third, the security strength of the authentication code should ensure it cannot be reproduced. SImilarly, article 5 of the RTS and 97 of the PSD2

demands dynamic linking of each part of the transaction process. For instance, while a transfer initiate, the authentication code should bind to the specific amount and the payee, which must be visible to the payer. Altered values of any of those elements than when initiated, should invalidate the payment request immediately. The same goes for the PSP, while accepting the authentication code to a corresponding payment, that the amount and the payee is intact as when initiated and no alteration took place during the lifetime of the payment process. This way the integrity of the payment process can be ensured. Considering the current technological improvement, two major sustainable solution SOAP and REST can be think of, for web implementation of above requirements.

The final report of EBA on draft RTS [10] requires that, a TPP (payment service provider) can be verified by using qualified certificates as mentioned in article 3(30) and 3(39) of the eIDAS regulation. The verification held on web, must need this qualified certificate for create electronic seals to verify the identity of the TPP. If the XML based message used, European Standards Organization ETSI developed a digital signature mechanism which can be used for proving. However, the the electronic seal doesn't have to be a qualified one, but can be generated without a qualified electronic seal creation device as mentioned in the article 39 of the eIDAS regulation.

Security Assertion Markup Language(SAML) and OAuth version 2.0 can be considered as good options to build such triangular authentication relationship. Despite the different design patterns both comprised of, they are noteworthy to support the second approach of authentication mentioned above. With the aid of SOAP or REST based interface implementation both SAML and OAuth are substantially capable of handling the PSD2 compliant interface requirements.

4.1 Security Assertion Markup Language(SAML)

Figure 4.1.1, SAML process usually consists of three parties, the user, the SP (service provider) and the IDP (identity provider), who communicate with each other. The consumer of SAML assertion is the service provider. As mentioned in PSD2, the execution of a business process requires an authorisation from the user to ASPSP. SAML V2.0 supports this kinds of delegation with a standard way.

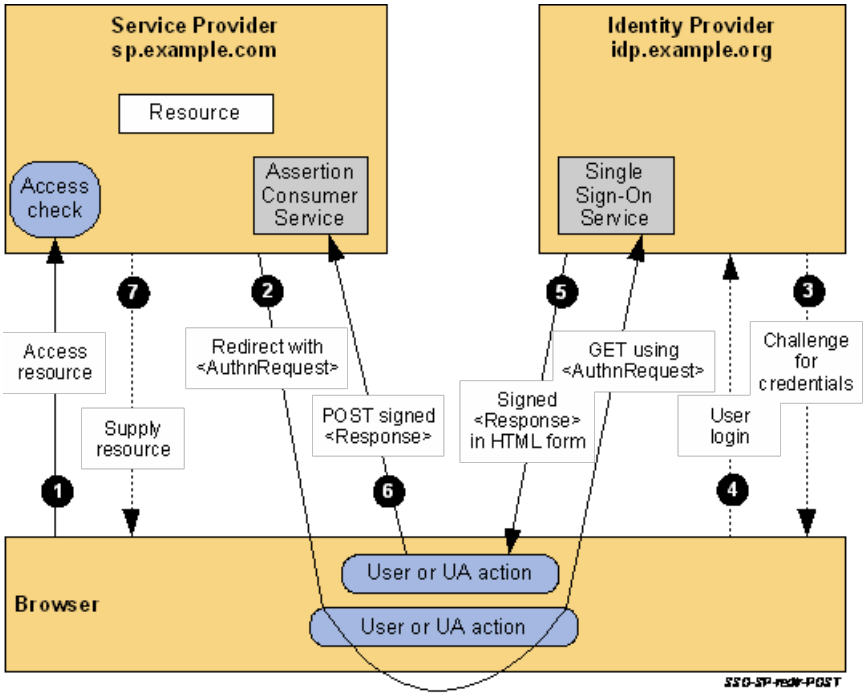

Figure 4.1.1: SP initiated Single Sign-On Service [11]

No specific technical method pointed as mandatory to link up the assertion made by SAML, say for example to bind the assertion with SOAP. However, REST based API stipulates to bind the assertion to a custom authorization scheme. The delegation variants of the SAML uses a field "DelegationRestrictionType", that names the authorised entities permitted to delegate the assertion. It makes easier the involvement of an ASPSP as fourth participant after the user, SP and the IDP in a typical SAML authorisation procedure. Another variants of SAML called forwarding has a main difference with the delegation version that, the user agents directly authenticates to the IDP, then received assertion delivers

to the intermediary SP. The intermediary SP acts as an subject forwarding the assertion to the another SP for authentication requirements. This another SP accept this assertion as long as it knows, the assertion is coming direct from the user agent. Whereas, delegation method doesn't pretend the intermediary SP to be a direct user agent while sending request to another SP, rather the SP authenticates requesting back to the IDP with the same subject. A new assertion then issued listing the intermediary SP as a delegate of the subject. In a nutshell, the delegation variant is intended to address multiple intermediary SPs involved in a transaction altering the path by each SP. These intermediary SPs are labeled as delegates and the assertion as delegate assertion. Such multiple tier delegation reflects the requirements of PSD2, therefore, made it one of the ideal candidates for the PSD2 compliant identification management interface. [12]

SAML assertion works both as authentication and authorization token. To use SAML assertion as a token in an API call, the same assertion should be added. Additionally, the assertion must include the information of interface access control, being aware of, the token will be used as an authorisation token for granting privileges, i.e. transaction, assigned to a user. The assertion can hold the authorisation statements using eXtensible Access Control Markup Language(XACML) [13], an attribute-based access control system (ABAC) that assign grants whether an user is authorised to access a resource. The statement is encapsulated inside the SAML assertion identity profile and considered secured from tampering. However, in case of exchanging the user access rights over non-XML based protocol, the JSON version of XACML should be a better choice. Regardless of the XML or JSON format of XACML used, the authorization statement will be encoded as standard SAML attributes that an ASPSP can verify it's integrity whether it is issued by the authentication interface, with a XACML system or any subset of XACML system. The assertion used for latter cases, for example long term access, probably issued for a certain amount of time compared to the one time use of transaction case. Remarkably, the security concern e.g. replay attack considered crucial in SAML. As a Consequence, main-

taining a list of consumed assertions as long as validity time is not expired are compulsory on the receiver end. [14]

4.2 OpenID Connect (OIDC)

Apart from the SAML architecture, OAuth 2.0 [15] is an another good solution to reflect on PSD2 requirements. The OAuth 2.0 addressed a few problems in the context of sharing a protected resource with third-party applications (TPA). The resource owner share it's personal credentials with the TPAs to grant consent of access, which is not only the consent, rather the full access to all resources protected under that credentials. This kind of action results into more severe problems and limitations.

1. The credentials are stored typically in clear text by TPA, for latter use
2. Regardless of weak password used that is vulnerable to break, the server needs to support password authentication
3. Unlimited access to owner's protected resources by TPA which is supposed to be for a particular duration and to the subset of resources
4. Cannot differentiate the TPAs, in case of resource owner want to revoke the access rights given earlier. Whether revoke all or none
5. Security break in TPAs literally compromise the credentials of the end user causing the protected resources open to uncountable unauthorised parties

The above issues are carefully considered for OAuth and the outcome was an additional authorisation layer separating the role of a client from the role of resource owner. Figure 4.2.2 shows the protocol flow of OAuth 2.0. In this refactored version, resource owner and resource server (hosts resources) controls the request of access by the clients. If authenticated, the clients are provided with a different set of credentials called access token, than those of the resource owner. The access token is well founded to a set of attributes, e.g. lifetime, scope and access grants, intended to be used by the client. Prior

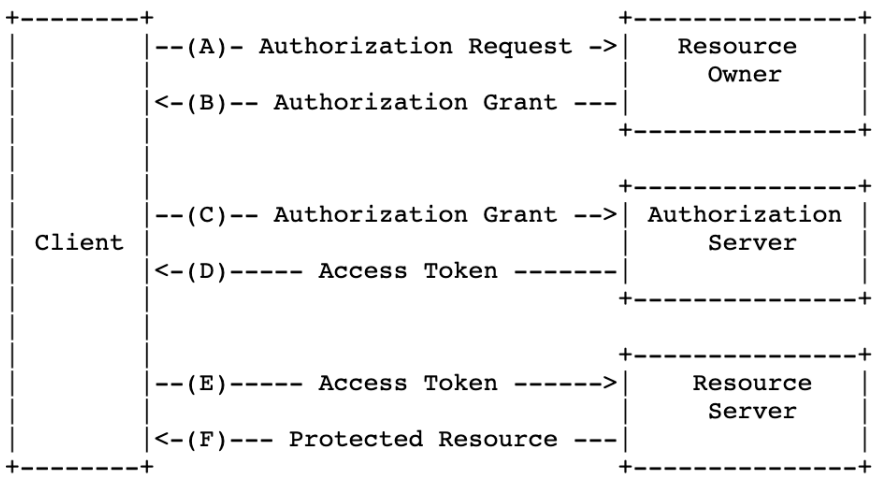

Figure 4.2.2: OAuth Protocol Flow [15]

the access token is issued, the resource owner approve the client to receive access grants with the authorisation server. Now the client is eligible to access the protected resources hosts by the resource server [15]

SAML usually needs wise consideration of how the assertion will be validated based on the execution context. On the other hand, OAuth 2.0 drives the SPs to receive a limited access to an HTTP service (ASPSP) on behalf of the resource owner having an interaction between the resource owner and the HTTP service (ASPSP) or by permitting the SPs to receive access on its behalf. Here the resource owner can be substituted with the service user to represent the compliance of PSD2 with OAuth 2.0. The OIDC which is the OAuth 2.0 built on, can be used for secured user authentication straight to the SPs. OIDC accepts different modes of operation depending on the use case scenario. Including a redirect URI with a "response_type" and "response_mode" parameter OIDC is able redirect the outcome of any authentication request through the user to the SP. The authentication scenarios are named Flows, which allows a resource owner to share protected resource belongs to him, without sharing his personalized credentials. The SP can use access token provided by the OAuth 2.0 server, to manage access to the protected resources delegating the resource owner. Among different types of Flows, the Authorization Code flows

mostly used for server side and client side applications. An ideal example of this flows reflects, how facebook and google accounts used to log in other service providers web application. The security of the payload, including ID Token, response and the request object, ensured using JSON Web Encryption(JWE). Also any confidential data sent by the SP to the user are encrypted and signed as long as the Request object used. The confidential data can also be transmitted as a XACML payload. To comply with the REST based structure a JSON version of XACML can be considered too. Precisely, OIDC follows two steps authorisation protocol to access a protected resource. First, the ID Token used to authenticate the service user against a SP. Second, an Access Token received in exchange of ID Token which is used as a key to access the restricted resources owned by the user. Any transaction request towards an ASPSP, attach this Access Token as Authorisation Header in an HTTP request, which sends it in an encrypted form. OIDC encourage to use a REST base API, since it doesn't advise any alternative ways of sending the authorisation information. However, any single authentication request can be handled as well using JWT encryption mechanism. [14]

The technical solutions for both SAML 2.0 and OIDC are necessarily resilient to the standards of the PSD2 in terms of identity and access management for the PSD2 compliant financial industry. Different approaches of these two solutions, met the necessary properties needed for the implementation of the PSD2. While SAML 2.0 offers a variety of composable specifications, still it would be nice to reconsider using its features and extensions in terms of defending certain security properties. Conversely, OIDC offers a kind of monolithic framework. Logically, the structure offers by OAuth 2.0 / OIDC, comparably easier to build the system on top of it than top of SAML 2.0.

4.3 Additional safety precautions

To apply additional security precautions, the PSPs could bring under the provision of one time or certain time interval authorization against the ASPSP's authentication procedure. This might not be a solution to the probable risks but an additional security step to the prevention of unauthorized access. For instance, an authentication completed on the PSP's end with the user's biometric, an ASPSP cannot guarantee the biometric originally belongs to the account holder. This gap of authentication procedure can be eliminated, if the PSP count on ASPSP's authentication procedure. Note that, the PSD2 never decides any certain structure of the authentication procedure of the ASPSP. Thus, adding a one-time authorization in the flow of authentication can add value to the secured implementation of the system.

But the refusal of access to the accounts until the SP is authorized to the ASPSP, shouldn't be right action since PSD2 and RTS has no explicit mention of such obligation. If we take a look back on the article 68 point 5 of PSD2, which allowed the ASPSP to deny access request for objectively justified and duly evidenced reasons in provision to the fraudulent access to the payment accounts. The risks are apparently related to the access right to a payment account, therefore, the possibility of being abused depends on granting access to the payment account. In addition to that, active fraud detection and transaction monitoring mechanism systems are compulsory for the PSP and ASPSP to detect fraud and unauthorized requests. When any PSP is objectively suspected of not using the PSD2 compliant authentication procedure, the ASPSP can stipulate one-time authorization to verify the association between the user and the account owner. Out of many, for instance, an abnormal spending pattern shall force the use of one-time authorization.

Article 68 of the PSD2 prescribed the service provider's use of payment instruments and access to the payment accounts. Point 1 suggests that the service user and the corresponding service provider(SP) can agree on the spending limit for the payment transactions initiated through a payment instrument, used for the purpose of giving consent. It

could be also possible, the exception of using the limit can be applied when the SP's authentication procedure doesn't comply with the authentication procedures of the ASPSP. The framework contract signed between the user and the SP or the ASPSP can have a reserved right to reject or interrupt the payment instrument for objectively justified reasons, any fraudulent and suspicious use of the payment instrument, or credit line of the payment instrument pose a significant risk of confidence, the payer wouldn't be able to pay back. Such unforeseen cases cannot be resolved beforehand but contact the payer where possible beforehand or immediately after the interruption with the reasons in an agreed manner, unless objectively justified security reasons compromise or forbidden by national legislation or relevant Union.

Application of one-time authorisations is contingent based on the fact that RTS and PSD clarified nothing regrading the circumstance it can be applied. The user account and personal data are essentially secured with the use of it. But, the market competition can be affected because of this additional step of authentication before the access is granted. Nevertheless, the intended security should be a priority against the feeling of an added additional obstacle to the authentication process. Rather than a transaction request rejected by the ASPSP, adding this extra layer of security might facilitate the authentication process sometimes. However, PSD2 aims to widen the financial market, therefore actions halt its acceleration should be carefully considered before the implementation.

Chapter 5

Development of an SCA integration model

The discussion above clarifies the fact that SCA strongly relates to the PSP and ASPSP and the user. The new regulation forced all PSPs to be registered and identify themselves against the authentication interface of the ASPSP. Even in the case of a screen scraping access method used, the procedure should be the same. The valid PSPs are obliged to register to their local competent authority. The SCA integration model depends on third party PSP API called Stripe [16] to continue it's business payment services. The host business model is an e-invoicing company, facilitates the process of invoicing for their customers electronically. Since the customers are basically involved with the construction business, their subcontractors and employees deal with the collection of spare parts and the installation of those spare parts, which also need to be billed to their employers.

Irrespective of places and time employees e.g. plumbers should be able to send invoices to the accounting department to create proper invoices to be sent to the end-user who consumed the services. The invoices usually include the price of spare parts as well as the wages per hour. Figure 5.0.1 depicts a graphical flow of the whole business process.

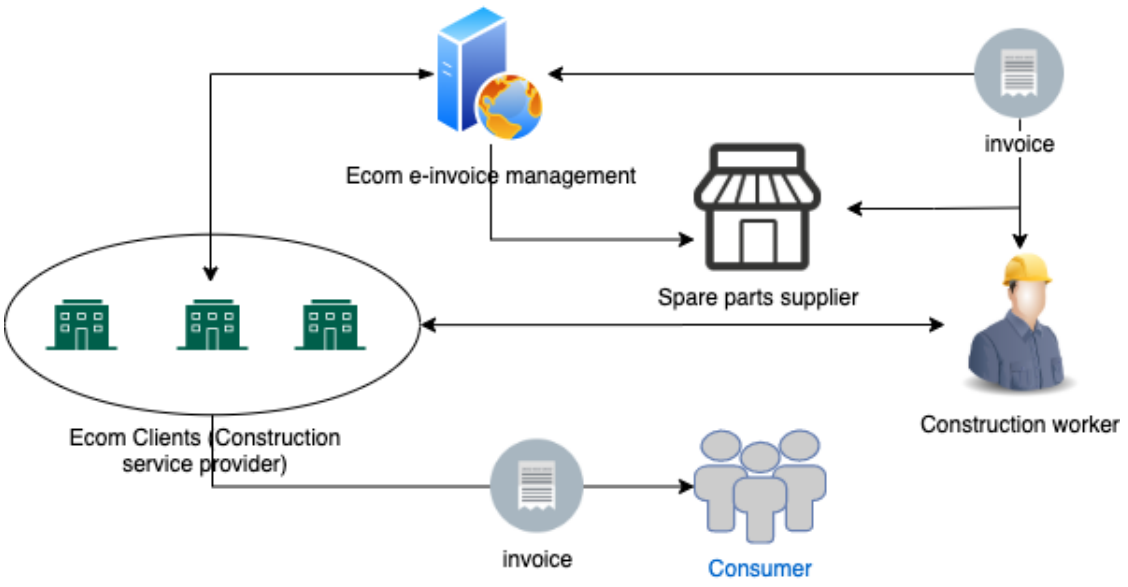

Figure 5.0.1: Ecom business model

5.1 Subscription and payment process

The service offered to the customers has a monthly billing period comprised of several different package options. Packages are combined facilities for the admins and users to access resources. The more resource a customer use or the facilities seek for, the more he/she pays. However, the price comes as a fixed wholesome doesn't vary on the amount of resource use. Precisely, the customer opts first what packages are suitable for their operation instead of pay as you go. Of course, they are open to extending it later if wishes. Since, the business model exchanges services against the money with customers, it depends on a PSP to deal with the billing to reduce the overhead of manual billing, fraud, and improve reliability. The host business has its own automated billing controller, that initiates the payment request to the PSP on due dates.

In the context of PSD2 and SCA, The host business model is inevitably associated with the payment service provider or PSP. Ecom accepts only CNP (Card Not Present) based payments to buy a subscription. Handling payments on the business end usually imposes a handful of fraud risks, since the identity of the payment initiator is unknown.

The additional authentication step 3D secure was a proven cure associated with the card-based payment fraud. VISA and MasterCard already using 3D secure 1 before PSD2. Another successor 3DS2 (3D secure 2) compliant with PSD2 about to be released soon. Undoubtedly, this would be appreciated as well to strengthen the card-based payment security. 3DS2 enables the bank to receive more necessary data for the corresponding payments from the PSP and businesses e.g. shipping address, device ID and transaction history. The more information is available regarding the payer and payments, the fast and frictionless authentication procedure it follows. However, when it fails to prove as a legitimate user and owner of the card, the user is forwarded through a challenge, that needs an additional authentication confirmation for example, from the bank's app, installed in the user's smart device. These uncertain authentication situations, additional authentication call depends on the time, place, user, amount, transaction history, etc. Concurrently, different banks may have different procedures and authentication solutions to reflect on PSD2. No doubt, it would be time-consuming, cost-oriented to develop, cumbersome, need continuous maintenance and attention which probably appear as too much overhead along with maintaining own business. Top of that, deploying own fraud protection mechanisms wouldn't be a cheap solution to buy. Stripe as a PSP handles all those matters, introduced by the ASPSPs as per PSD2 regulations.

The use of smartphones makes 3DS2 easier to implement through the use of the application provided by the ASPSPs. This out-of-band communication, accept confidential credentials from the user that denotes the legitimate use of the smart device and application. Instead of forcing the cardholder to remember a password, it can authenticate the user through the smart device's built-in secured fingerprint authentication or face recognition. Undoubtedly, it's a remarkable improvement compared to the 3D secure 1's registration and additional time-consuming authentication steps. In addition to this fast authentication process, 3DS2 executes on the client site without the full page redirect. The challenge flows now embedded within the web itself. The cardholder wouldn't be redi-

rected to a new page rather the 3DS2 authentication flow appears on the checkout site as a modal. This is a feature of improved user experience. A PSP can request an exemption from SCA to 3DS2 based on its own calculated threshold for a payment request. If the request is allowed to pass through the friction-less flow, then the PSP won't be benefited from the liability shift in case of fraud claims. The widespread use of 3DS2 by the ASP-SPs wouldn't take place so fast since the adoption of all of the individual card issuer is crucial. Partially, country and region are variables as well to its successful adoption.

Stripe supports both 3DS2 and 3D secure 1 on their payment API. As the card holder's bank ask for version 1 or 2, it automatically falls back into the requested version. Handling this process happens dynamically for high-risk payments. Mobile applications, both IOS and Android support an in-app authentication process, that prevents clients from redirecting outside of the application offering a smooth on-site experience. The automatic fallback option to 3DS2 or 3D secure 1 remains the same for the mobile OS.

5.2 Multi-factor authentication

Ecom considers data security seriously to tie the user's trust to the business values. Apart from the user name and password-based authentication, multi-factor authentication activation is available as an optional layer of security at this moment. Optional means, the authentication is not forced to use, but the user chooses whether to activate it or not. Breaking digital accounts weigh easier, safe from unveiling the identity than physical breaking inside any protected resource. Organized cybercrimes e.g. malicious attacks, session theft, data breach, repudiation, hacks are always rising rather slowing down. Traditional security systems are obsolete to the sophisticated cyber attacks resulting in the personal data breach. Being aware of severe financial and business loss, consumers are given the option to reinforce their account's security through the use of multi-factor authentication.

2FA (Two factor authentication), a sub-type of multi-factor authentication is implemented as an additional layer of security that represents something the user has, something the user is and something the user knows. Which is similar to the strong customer authentication requirements. Instead of providing access, this additional layer of security comes in action after the user completes the username and password authentication. 2FA asks for another secret from the users that agreed during the time of enabling 2FA. It could be a PIN, OTP (one time password), answering secret questions or a keystroke pattern. Since both authentication layers are independent of each other, it is highly unlikely that even if the user's first layer of security credentials (username and passwords) are stolen, the attacker will gain any access to the account. But still, the user needs to be aware of this and enable the security measures properly, for instance, lock smartphone with a passcode or bio-metric data, use a random PIN for the authentication application that can't be guessed in one or two attempts.

Given so many options available, we opted Time Based One Time Password (TOTP) for this second layer of user accounts security with Google authenticator. Even though, this authenticator application might not be the best choice to be used for clients, at this moment it is good enough to satisfy the second layer of security needs before the next viable solution implements. The user sends a request to enable 2FA to the server, then the server response back generating a unique random secret key to be used on google authenticator API that generates a QR image. Users can use the google authenticator app to scan the image that specifically created for the corresponding user. A 6 digit TOTP keeps generating in the authenticator application against the user's profile as long as the application is open in the smartphone. Each key will remain valid for 30 seconds. Once the time expired a new key will be generated. Since the OTP is time-based, it always uses real-time for the key generation.

Despite the google authenticator implementation, there are a few more 2FA options available to choose from as a replacement in the future. SMS based 2FA has been in mass

use for a while. This form of authentication interacts directly with the user's registered smartphone connection. Once the user passes through the first layer of identification, an OTP token is sent to its smartphone as a text message. As soon as the OTP arrived, the user should type it on the required field and send back to the authentication server to verify. If succeed the user is given access to the intended resources. Similar to the text OTP, not common anymore, voice-based OTP delivery is another option. Unfortunately, SMS or voice-based 2FA is marked as the least secure methods of identity verification in recent times. Therefore, it should be avoided for websites that deal with high risks operations e.g. payment, invoicing, money transfer, store personal transaction data.

Push notification is a popular form of 2FA nowadays. However, it has dependencies on the internet. User's smartphone, retailer and the 2FA has a triangular secure relationship among them in this process. It eliminates the use of OTP from the verification process. The user's smartphone or device should be capable of installing apps, which can receive the push notification and prompt the user for consent. Based on the access request the user can response back through the app. Conveniently, the risks of phishing, a man in the middle attack or any other form of penetration must be very hard to compromise the security of this identification method. The downside of this method is a consistent internet connection, but in the context of security, no doubt this could be one of the best options for now. [1]

5.3 Scope of this Thesis

PSD2 identified significant scopes to develop the financial market to it's maximum possible within Europe. It is not worth and limited by time to address everything n this thesis. We would like to see how the payment service providers equipped themselves against the new rules and regulations. Since the real-time market analysis data is not pos-

[1] https://authy.com/what-is-2fa/

sible to retrieve, we need to depend on assumptions only for the future market prospects. However, Ecom as an e-invoicing business company more or less have to deal with the situations, because of the transaction happens with most of the banks in Finland. We want our customers to experience a smooth billing process without any friction on the payment system. To facilitate the whole system, the business process is contingent on the friction-less payment system. The fraud protection mechanism is underlying within the Stripe. A reasonable amount of discussion took place in different parts of this thesis already. We are not going to emphasize on it anymore. The integration code is more than what we have put below and the whole model is connected from all angels of our system customers. As a result, depicting all the connected corners wouldn't be possible throughout the integration process described below. In most cases, we will try to stick with Stripe acquisition, initialization, customer creation, how to charge the customer on session, creating a relationship between the customer and the payment method they choose and most importantly PSD2 compliance handling.

5.4 Integration of PSD2 compliant Stripe API

As it is mentioned beforehand Stripe [16] handles the money moving part from customers to Ecom accounts. It offers versatile payment options for the checkout flow e.g. cards, wallets, local payments. The checkout flow is optimized pretty well for mobile applications and mobile web as well. In our context, only CNP payment checkout flow is relevant to discuss in detail, seeing that only CNP based payments are accepted at this moment to build professional business relationship with Ecom. A customer must have to have a credit/debit card to pay their bills.

Initially, the integration model divided into two main areas, client-side implementation and server-side implementation. The whole process of integration altogether can be fit into 5 major steps. Since we aim to recurring payments from the customers in the future,

we save the card details and customer information for later use. However, Ecom only charge customers on session, which means when the customers are themselves on the checkout flow. We don't charge customer off session. We will discuss later on the step how to bring the customer on session for payments that charge later before the due date.

Step 01, Install Stripe (Server-side): Stripe should be set up on the server-side as an initial step. Official libraries are available for most of the popular platforms to access the API from the application. In our case, we choose *npm* version and install the package in our back-end server. To initialize the Stripe library with our business account, remember we need an active Stripe account that provides us a secret API key.

```
1    const stripe = require('stripe')(your_secret_API_key);
```

Step 02, Create Customer (Server-side): Creating a new customer on Stripe is a must to set up any intents. When a customer appears in Ecom checkout flow, we check whether we created this customer in Stripe's Customers table or not. On successful creation, we save unique *customer_id* on Ecom storage associating with the customer information, else we find the payment method from the customer information and avoid creating a new one. The request initiates from the client-side, and fetching or creating a new customer id happens on the server-side then returns securely back to the client-side. Figure 5.4.2 shows how the process takes place.

```
1  app.post('subscription', async (req, res) => {
2    const { systemCustomer } = req;
3    const payload = {
4      name: systemCustomer.name,
5      email: systemCustomer.email,
6      metadata: {
7        'Ecom Jet SystemCustomer ID': systemCustomer.id
8      }
9    });
10   const response = await stripe.customers.create(payload);
11   res.json({ client_id: response.id });
12 };
```

Figure 5.4.2: Create customer in Stripe

Step 03, Create Setup Intent (Server-side): We want our customers to set up for fu-

ture recurring payments. To do so, in Stripe we need to create an intent with Stripe *Setup Intent API*. Precisely, we are saving the credentials to charge the customer in the future. Once the payment method is set up, it is possible to reuse the method, for different use cases. The *SetupIntent* object keeps updating during the process of setting up the card e.g, validity check, authenticating customer. A card set up for off session payments certainly needs permission from the cardholder. An agreement is necessary between the service receiver and provider. In terms of SCA, it is mandatory to get the cardholder's permission before we save it for future charges. The terms of service of the service providers should include information at a minimum, consent for recurring payments, payment amount determination policy, and what interval the charge will be made. Then, all off session payments are tagged as Merchant initiated transaction (MIT) by Stripe, which is identical to the request exemption from SCA that let customers be independent of their presence while making a charge. Literally, the success of the exemption request is contingent on the card issuer bank. Only they decide whether to pass this request through a friction-less gateway.

```
app.post('/generate-intent', async (req, res) => {
  // send customer id (optional) if created already in Stripe
  const intent =  await stripe.setupIntents.create({
    customer: customer_id,
    usage: on_session
  });
  res.json({ client_secret: intent.client_secret });
});
```

Figure 5.4.3: Create setup intent

Nevertheless, as mentioned earlier, Ecom doesn't charge customers off session but on session yet. However, the UI and the backend server is continuously improving that the off session charge might be available before this writing finished. A parameter called *usage* passed to the *Setup Intents API*, while creating an intent (Figure 5.4.3) tells Stripe to charge the customer now. Basically, this parameter is designed to succeed payment requests initiated with the presence of the cardholder on session. Note that, it is possible

to charge a card off session, previously which was saved as on session. The concern is, the card issuer bank is highly probable to send the request through the challenge flow since the initial card setup wouldn't be identical to the request. Stripe marked *usage* parameter as an optimization to the payment request. Therefore, the authentication from the customer could be obvious in either case. Consequently, a recovery process is intended to exist to take over on failure. So that the customer can be notified about the failure and bring him on session in the checkout flow to complete the payment process. At this moment the recovery process implementation is under development since charging customer off session is not available now. The recovery process might include but not limited to notifying customers via SMS, email immediately after the payment process failed for off session payments.


```
{
  "id": "seti_1GMMAPIuKQ0PCijjgQX2qOYg",
  "object": "setup_intent",
  "cancellation_reason": null,
  "client_secret": "seti_1GMMAPIuKQ0PCijjg
  "created": 1584139185,
  "description": null,
  "last_setup_error": null,
  "livemode": false,
  "next_action": null,
  "payment_method": "pm_1GMMAuIuKQ0PCijj5
  "payment_method_types": [
    "card"
  ],
  "status": "succeeded",
  "usage": "off_session"
}
```


Figure 5.4.4: SetupIntents Object

Figure 5.4.4 represents an *SetupIntent* object comprised of fields regarding a card-holders detailed credentials. Notice carefully, the red marked field *client_secret*, a unique identifier key returned by Stripe to be tied with the client's card. This intent is not tied to any card yet, rather a usable registered intent from the stripe. Now a customer card can be bind with this intent for future use. The purpose is basically to create a confidential

separation between the client and service providers e.g Ecom, that never exposes the card number directly to the service provider for security. *client_secret* uniquely links to the customer's card and lets us perform several actions, for example, set up confirmation, updating payment details, authenticate and validate the card without exposing the customer. Note that, due to the *client_secret*'s confidential priority it shouldn't be logged, embed in any parts of our application.

Step 04, Collect Card Details (Client-side): For the client-side setup we start with installing npm package *react-stripe-elements*. This time we use React library [2]. Before all, we include Stripe.js script in our application and load it before calling any of the Stripe components. It has full integration support of *Setup Intents API*.

```
npm install --save react-stripe-elements
```

After the Stripe.js script loaded, we initialize a *StripeProvider* component with Ecom publishable key as props and wrap *Elements* [3] and all sub-elements under this component. *Elements* is the parent of a set of nested UI components e.g. inputs, buttons, to build a card data collection form on the client-side. All sensitive information lies within this are tokenized, and handled by Stripe securely without any interaction with our application. We skip some underlying component implementation details and their configuration to keep this reading comprehensive enough. Our main aim is to collect user card details. Therefore, we need *CardElement* wrapped with *InjectStripe* function component, that provide us the Stripe *handleCardSetup* functions to link up the setup intent with the customer card. *CardElement* is an optimized Stripe react form, that collects all necessary card details in a single input and securely sends it to the Stripe server for verification.

The setup is ready now to confirm a setup intents. Please see Figure 5.4.5 for the whole process. While filling up card details verification process keep running accordingly by Stripe. On success, submit button will be visible to make final charge. Remember, as we discussed earlier of SCA regulations, customer must be informed that we are saving

[2] `https://stripe.com/docs/payments/save-and-reuse`

[3] `https://stripe.com/docs/stripe-js#react`

```
1    const handleSubmit = async (ev) => {
2    // stop default page refresh.
3    ev.preventDefault();
4    setInProgress(true)
5
6    // create a card set up intent
7    const paymentIntentClientSecret = await fetch('POST','/generate-
     intent')
8
9    // card set up intent succeed, set up the card now
10   const result = await stripe.handleCardSetup(
     paymentIntentClientSecret, {
11     payment_method_data: {
12       billing_details: {
13         name: cardHolderName,
14         email: billingEmail,
15       }
16     }
17   })
18   if (result.error) {
19     // handle error
20   } else {
21     // Card set up done, create a payment method and attach to
     customer
22     await fetch('POST','/payment-method/'+ result.setupIntent.
     payment_method)
23   }
24 }
```

Figure 5.4.5: Card set up and payment process

their card for possible reuse later. Stripe *CardElement* is highly customisable for this sort of intent, to add a few lines of text regarding how we are going to use this card details later. We submit the form, and that send a request (line 7) now for a payment intents to the backend server as mentioned in step 03. As soon as the response arrives, we extract the *client_secret* field and Stripe suggests we handle it with security precaution in mind. Along with the *client_secret*, on the next line we call *handleCardSetup* function that set the payment card in Strip server for later reuse. Additional parameter billing details are saved as well to tie our customer with the card setup. Setup intents field *result.setupIntent.payment_method* returns with an unique identifier value assigned to it which we save to the provided customer on Stripe.

Chapter 6

Verification and testing

An integral part of any software development is testing the software. Malfunctioning of a software product which might have extreme consequences, asks for reasonable efforts for its validation. For instance, a software that drives an airplane landing facility, in terms of failure affects is different from a parking location finder program. It indicates that a software project must focus on all required attributes before the product initialization. A number of attributes to ensure the quality of the software is often considered as obvious. The software is reliable, has usability, easily testable, efficient, portable and maintenance is easy. Combining everything in a software project may deem unnecessary, sometimes irrelevant or limited by the deploy environment. Therefore, prioritizing the relevant software attributes would be a wise choice at the start of the project else the development would lead to a fragile structural architecture. Apparently, software testing is a laborious process that often resembles incompleteness. [17]

6.1 Software testing and data collection

One of the very basic and crucial elements of software testing is collecting the test data. Necessarily, test data should resemble both the inputs and the outcomes of using those inputs in different parts of the software. The inputs can be sorted as a group of inputs,

individually, and a combination of two or more based on test requirements that should also determine the specific outputs for each of them to be used. Notice that, the second approach of determining the output can be very difficult for crucial feature test, since simulating the proper output for any big process most often cannot be done by manual calculation. It is also insufficient for critical systems to be tested.

Primarily we can divide software testing into two major parts. Manual and automated testing. As the name implies, manual testing is a process of executing the software manually on the intended delivery environment, testing each function consecutively as it should do in the real environment. Ont the other hand, in an automated procedure, the manual test execution is replaced with test scripts, composed of diversified tools which are able to execute the whole testing procedures pretending to be steered and asserting on test cases by a human.

Software can be tested from a number of different execution angels. Out of many, we will discuss a few that are most relevant to our technical solution.

6.1.1 Unit test

The title 'Unit test' itself is self-describing. Unit test refers to the testing of the smallest units of your software. Units can be a standalone functional component that provides specific output for given a set of inputs. Tests are conducted with a variety of production standard input data to simulate the function execution efficiencies in the production environment. The test output should match the actual result of the expected output to qualify for integration. While testing, the function can be stressed to its maximum providing rigorous inputs that should be handled without error. Unit testing can add great values, for instance, on a monetary operation to check the accuracy. [1]

[1] `https://www.atlassian.com/continuous-delivery/software-testing`

6.1.2 Integration test

A test that covers multiple units to work as a whole to provide a single output can be said as an integration test. Even though all units pass through the unit test, still it is highly necessary to continue the integration test. Individual unit execution shouldn't have a doubt, but the integration of different units, for instance, can have a flaw in execution order, incorrect input and output flow between units, input type mismatch. Often, the acquisition of third-party libraries needed to accelerate the development speed and to avoid development flaws. Writing unit tests for third party libraries seems inessential, but integration tests. The dependencies are usually mocked to return the desired result.

6.1.3 End to end test

Integration test occupies multiple units together, where an end to end test depicts the simulation of full user experience. Recall that, the dependencies and the back-end server requests are usually faked in an integration test which is not the case in an end to end test. A typical example would have the following steps:

1. Click "Create new user"
2. Read the instruction and fill-up the form
3. Submit

Here the network request to the API wouldn't be mocked rather routed to the correct path and then the request is processed by the corresponding functions before it returns the final output. Since this process simulates the real user behavior and environment, the underlying network protocol by default get involved in the test to establish a connection between the UI and API, and returns to the proper request made from the UI after the result is processed. Precisely, all the layers of a software execution are involved here that verifies the accuracy of the software parts being tested.

6.2 Test automation with CICD

A very well known popular term in software testing is CICD, a process of integration and deployment to the production, which stands for continuous integration and continuous delivery. CICD facilitates all testing plans and strategies into a single pipeline that executes all entities sequentially to deliver the codes into production on success. For instance, a CICD pipeline can have the following levels of steps:

1. Run all test cases
2. Merge codes on success into the master branch
3. Staging or production deployment

Elaborately, the first step of the process is, a developer pushed his completed task (codes) into the CICD pipeline. The execution of the predefined testing levels in the pipeline starts immediately on the task one by one. If the task successfully passes through the test execution pipeline, generally the next step is to merge the codes of the task into, for example, master branch of the project. The tests passed, task merged, now it should be fine to start the deployment of the task to the production or staging and it goes for the deployment. All steps are well scripted that let the whole process run and executes on required places automatically. In case of any failure in the CICD pipeline, the user is immediately informed through predefined communication methods, e.g, SMS, email halting the merge and deployment. Additionally, visual implementation of the CICD pipeline command execution is pretty common that allows users to see on-screen what went wrong. The user can take relevant actions to fix the task and push back again to the CICD pipeline. Most of the programming language platforms have built their own ecosystem for testing. Different types of programming-language specific tools and utilities are built to ease the development of testing suites that are available to download using the predetermined package manager.[1]

Adjacently, additional tools to perform test execution and test facilities development

are very handy as well. Suitable third party test runner packages can be installed to dispense the final output from the test suite. One of the widely used tools is measuring code coverage. This tool demonstrates how much of the actual codes of the whole codebase is covered under the tests written. At the end of the test, a table of code coverage report is printed that highlights the percentage of codes and different parts of each file, e.g, functions, statements covered in the test. Conveniently, most of the code editors also support the visual demonstration of code coverage with the color marking on each line. Which comes as pretty comprehensive to read and distinguish between the lines of codes that are covered under test.

Contemporary test methodologies have replaced the customary methods. Previously, testing software used to consider as a task of the Quality Assurance (QA) team. No more that methodology is in active use nowadays, rather it is obsolete and the responsibility is now tied to the developers as well. Developer's empathy for the customers and products considerably helps to build quality products and codes, else the absence of developers in quality assurance typically lets the issues dormant into the code base for an unspecified duration. Consequently, exposing these issues later requires extravagant efforts to fix them. Inadvertently, this extravagant efforts may finally extends to hiring a new QA team to handover the liability. CICD encourage developers to be aware of the customer empathy and product's quality to deliver a flawless user experience. The tests cases written by the developers should cover the feature they work on properly. Overseeing them in advance, being executed in the production environment also offer developers to maintain the quality of the assigned task.[1]

CICD workflow offers enormous business benefits. As we discussed above, traditionally, the responsibilities of the QA team comprised of software testing and maintain product quality. Chronologically, the liability has shifted to the developers now. It facilitates the release, QA and testing faster than the traditional approach and also cuts the cost to spend after a managing separate QA team.

6.3 Testing SCA integration model

While discussing testing, we mentioned programming language platforms have their own built-in ecosystem for testing. The integration model has been developed in React, therefore, related choice for testing libraries are Jest[2] or Enzyme[3]. We preferred Jest to continue as a basis of the testing framework. Jest has extensive support for a variety of JavaScript frameworks e.g. Typescript, Babel, Node, React, Angular, Vue, which makes it very popular to choose for software projects around the world. You need almost zero configuration to start with jest apart from the installation part. Of course, it has tools and options to configure your project as you want. One of the powerful features jest offers is, tests are possible to run in parallel that uses the maximum of your machine's computing abilities saving time. It runs the failed test first on re-run and has filtering options as well to filter out particular test files only to run rather whole. The user is given a handful of options to filter out the test files that need to be run, for example, failed tests only, by filename, etc.. At the end of the test, jest presents a table of test report contains how much of the functions, statements, branch, lines are tested for each test file. Also, errors are color-marked which is visually very comprehensive to read and troubleshoot. All unit tests and integration tests are gracefully written in jest to be executed. The assertion libraries are free to choose from the market. Having this flexibility we tied React testing library[4] with jest to provide us plenty of helpful DOM testing utilities. The utilities are helpful functions that can be used to manipulate the DOM efficiently, for instance, get an element from the DOM by its *data-testid*, use a regular expression to search all elements have text *"Name"* and a lot more. Moreover, Mocha[5] and Chai[6] are also integrated to write asynchronous tests and make assertions on those test cases accordingly. If we take a

[2] `https://jestjs.io/`

[3] `https://enzymejs.github.io/enzyme/`

[4] `https://github.com/testing-library/react-testing-library`

[5] `https://mochajs.org/`

[6] `https://www.chaijs.com/`

look back once at everything together, Jest works as a test runner, Mocha used for writing asynchronous test cases (also synchronous), react testing library provides DOM utility functions and finally Chai, helps on asserting the test cases.

```
/* eslint-disable no-undef */
describe('Subscription', () => {
    it('should subscribe to PRO and allow user manager', function () {
        cy.login('free.freemium@ecomjet.test')

        // settings->users should not be accessible
        cy.visit('/settings')
        cy.wait(500)
        cy.get('a').contains('Käyttäjät').parents('a').should('have.class', 'Mui-disabled')

        cy.visit('/settings/subscription')
        cy.subscribePro()

        // settings->users should be accessible
        cy.visit('/settings')
        cy.wait(500)
        cy.get('a').contains('Käyttäjät').parents('a').should('not.have.class', 'Mui-disabled')
    })

    it('should cancel subscription and reactivate without trial period', function () {
        cy.login('free.freemium@ecomjet.test')
        cy.visit('/settings/subscription')

        cy.subscribePro()
        cy.get('button').contains('Lopeta tilaus').click()
        cy.get('button').contains('Hyväksy').click()
        cy.contains('Tilaus on päättynyt') // Subscription is canceled

        cy.get('button').contains('Uudista tilaus').click() // Renew subscription
        cy.subscribePro()
    })
})
```

Figure 6.3.1: End to end Cypress test

Jest test runner setup, at this moment we kept it limited to create unit tests and integration tests. What lacks from the Jest test is an end to end testing, to test the whole software setup altogether with the API. Cypress[7] filled-up this gap efficiently introducing end to end test for Ecom software projects. It is fast, robust and very user friendly to debug the tests. Top of that the package is open source for everyone to use. Writing test cases in cypress is relatively easier than writing test cases in Jest. Since all API requests are mocked in jest setup, we eliminate this limitation by using Cypress to create a real-time simulation of user experiences and behavior. Surprisingly, Cypress generates videos for all test cases that it runs, therefore debugging and finding out the reasons in case of test fails are pretty comprehensive. Our SCA integration model necessarily involved with the API to securely handle the connection to the Stripe and the secret access token. It is good to know that Stripe offers a test API for our implemented model to be tested against it. On

[7] https://www.cypress.io/

successful user and payment setup, the data is available in the test API. Manual checking or writing automatic test cases can validate the information. However, The whole process is not brought into the end to end test flow, instead, the required user subscription process is simulated on end to end test flow. Figure 6.3.1 exhibit an end to end Cypress test scenario for new user subscription and cancel a subscription. This parent file reused a couple of underlying functions defined in the default configuration file that deals with the API requests, database connection, and manipulation, and utility functions, e.g. *cy.login()*, *cy.visit()*, *cy.subscribePro()*. Function *login* completes the user login with the required data and makes sure the user is logged in, then *visit* navigates to the intended path.

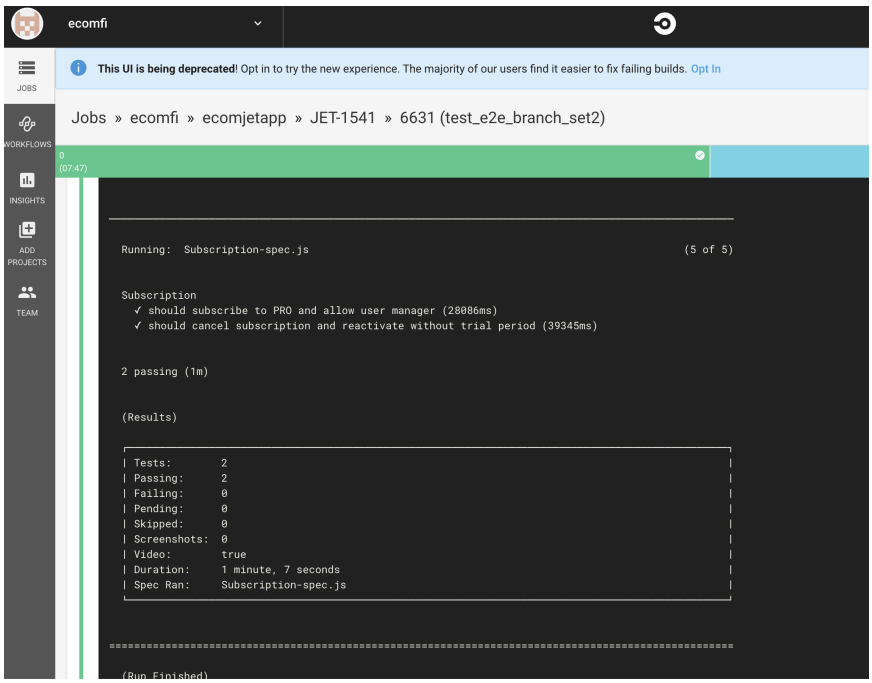

Figure 6.3.2: CI test result

Cypress test report represented in Figure 6.3.2 speaks for the test cases written in subscription-spec.js file (Figure 6.3.1). Two test cases as shown in the report passed after running for 1 minute and 7 seconds. The test running duration usually varies in the basis of computing resource availability. Except for the passing and failing, the report also contains additional fields, for instance, pending, skipped, screenshots, which are

mostly empty in this context but the video is true. The graphical test execution process is recorded, and available at the end of the test in the corresponding path given in the configuration file.

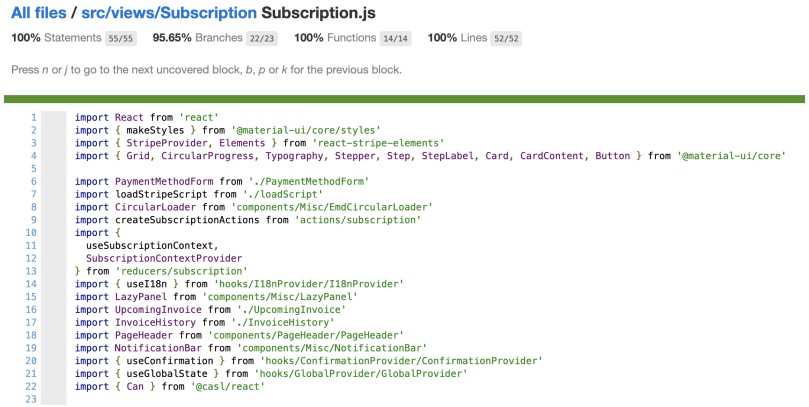

Figure 6.3.3: Code coverage report

It is worth checking how much of the actual codes have been covered in your tests. sometimes, inappropriate test inputs discard the execution of conditional statements that result in test incompleteness. Code coverage report helps to resolves this incompleteness, demonstrating the missing and covered codes in the test file by the containing elements e.g, statements, functions, lines. The code coverage report(Figure 6.3.3) for our integration model confirms almost 100% code coverage, which is visible at the top of the figure.

Fortunately, the Stripe testing API offers all set of failures and success cases to simulate the payment flow. To make sure the integration was a success, Stripe provides a set of test cards and other information that helps to trigger the flow implemented in the system. It is a strict "No" to use any genuine card number in the test environment. Triggering the flow with test card information replicate the real user behavior with necessary events, webhook call, and error handlers. Also, certain test card numbers are provided that trigger 3D challenge on-screen to test the challenge flow. On successful completion of the challenge trigger corresponding events or webhook call, that can be asserted on UI. Required cypress test cases are written to handle the complete subscription process in real-time

with Stripe test API that includes handling the customer creation, subscription management, charging the customer card, informing customers immediately on charge fail, and canceling the subscription. Developers locally make sure all test cases are passed, before sending it to the final CICD pipeline.

The test setup and code integration are achieved through an in-house built CICD pipeline. Once a new feature is built and pushed into the remote repository, the execution of the commands starts on the feature branch from the CICD pipeline. The setup always varies based on the demands of the developer's team or software project. In the Ecom context, merging a feature branch into the master continues on test fails, alternatively, deployment to the staging/production remains due until the feature branch passes all tests along with other procedures remaining in the CICD pipeline.

Chapter 7

Discussion

The principal investigation focus of this thesis originally initiated from the revised payment service directive, PSD2. Throughout the exploratory journey above, we revealed the unforeseen business benefits would bring by the PSD2 in financial markets. The sole ownership of user data from ASPSPs, which will be in the shared domain along with PISPs and AISPs, can have multidimensional compromised and improved business use cases. Possibilities of data being abused by the adversaries were in the broad discussion if it leads to the gradual decline of monetary exchanges in the financial market. Even though ASPSPs are losing controls on their own data because of PSD2, we tried to answer could they stress out the market willingly, limiting the amount of data to be shared with the PISPs and AISPs. Moreover, the shared domain would be operated under the control of the ASPSPs. Naturally, ASPSPs will be assigned with these additional tasks of handling the authentication process, accepting and declining transaction requests, confirming the privacy of the user is not disclosed to an unauthorized party. Since PSD2 itself didn't provide much insight on this, except the technical guide of implementing the services, there are unidentified issues still left which have been adequately speculated in different related parts of this dissertation. Strong Customer Authentication or SCA is one of the major authentication changes requested in PSD2 to exchange the services between AISP, PISP, and ASPSP. The fear, whether introducing an additional layer of security challenge

could affect the market or not got mixed opinions from a few giant stakeholders. However, PSD2 SCA guideline answers who, when, what in the context of related parties but lacked specific technical solutions. Later, Regulatory Technical Standards, RTS backed up the technical requirements and formatting options without implying to any specific technical framework that leaves the host open to choose their own. We put our efforts to bridge this technical implementation gap inspecting a couple of reliable open source web technology's affordability in chapter 04.

Many PSPs have already developed their own solution following the RTS guidelines to cope up with PSD2 before the due date. This new acquisition also forced their client companies to upgrade their own local implementation to receive the payment services. The model we developed and integrated into our system has tested most corner cases, assuring compliance with the SCA regulation. Finally, the integration is backed up with automatic test runner and deployment, which confirm the model is up and running.

Chapter 8

Conclusion

To what extent the security of the user data is ensured, in relation to the given access of PISP and AISP to the client information and transaction history, addressed in this article. The consequence of compromising data security to uplift the market share can have a reverse effect on the decline of trade volume too. Lack of clear guideline in PSD2 put user's privacy in stake. A wide range of service coverage by account information service indicates a diverse increase in market growth, but do not address the pool of data processing can payoff user's privacy.

The normal ways of accessing user accounts, e.g, screen scraping were advocated for a prohibition by the EBA since it cannot distinguish between the owner and other entities seeking access. Being concerned about indistinguishable full account access, later version of RTS made an exception, the user interface can be accessed only as a redundant system. The solution is trustworthy during the normal operational time, however, falling back into 'screen scraping' for a long time could let the adversaries circumvent the issue.

Strong customer authentication, SCA eases the security and privacy deficiencies in open banking, introduced by the PSD2. Despite the fear, this additional layer of security might not be appreciated and the overall impact would outweigh the security benefits. SCA needs a draft model or ideas to be integrated within the collaborative authentication process between PSP and ASPSP. RTS either didn't resolve any implementation details

of SCA. We figured out a couple of draft methods to be useful in the integration process with its own pros and cons. Nevertheless, the PSPs have the freedom to rely on ASPSP's authentication system, or they can implement their own. However, implementing own would be expensive in return to the actual benefits.

PSD2 wouldn't be successful without strong cooperation among all involved parties in the finance market. Marked weak spots or unclear terms of PSD2 do not point to the inadequate protection of user's privacy, instead, it demands amendment on incomprehensible parts. On the other hand, it would be too early to predict it's future now unless the financial market runs for a while under this new regulation. Inevitably, the competent authorities of the member states should be supervised, monitored accordingly, to reduce the pressure on user account and data security.

SAML and OIDC are two viable solutions that can interact with the PSP during the authentication process, in provision of any service request. The limitation of involving the PSP, ASPSP and the user into an authentication mechanism let us sought methods that comply with the intent. In addition to that, it should be remembered that an ASPSP reserves the right to reject any suspected payment request. This means, even though a request is coming through a valid PSP, that doesn't mark the request as clean. Therefore, fraud detection and response to the incidents are essential for gatekeeping on the PSP side.

we exhibit an SCA integration model integrated into an e-invoicing system. A PSD2 compliant payment service provider Stripe offers a versatile range of payment services and APIs, to be implemented on the client-side. Since payment requests will be failed unless they are PSD2 compliant, it helps companies to be updated with the latest PSD2 changes in payment services to maintain successful payment flow in everyday business. The model is then attached to the CICD pipeline with an automated test runner, to ensure nothing going to get wrong without undiscovered.

References

[1] European Union. Directive (eu) 2015/2366 of the european parliament and of the council of 25 november 2015 on payment services in the internal market, amending directives 2002/65/ec, 2009/110/ec and 2013/36/eu and regulation (eu) no 1093/2010, and repealing directive 2007/64/ec [2015] oj l337/35. *Official Journal of the European Union*, OJ L 337(2366):35–127, 12 2015.

[2] European Union. Directive 2007/64/ec of the european parliament and of the council of 13 november 2007 on payment services in the internal market amending directives 97/7/ec, 2002/65/ec, 2005/60/ec and 2006/48/ec and repealing directive 97/5/ec [2007] oj l319/1. *Official Journal of the European Union*, OJ L319/1(64):1–36, 11 2007.

[3] Temenos. How temenos enables open banking and the revised payments directive (psd2). 2019.

[4] Kristin Moyer. Hype cycle for open banking apis, apps and app stores. Technical report, July 2015.

[5] PTJ Wolters and BPF Jacobs. The security of access to accounts under the psd2. *Computer law & security review*, 35(1):29–41, 2019.

[6] European Union. Commission delegated regulation (eu) 2018/389 of 27 november 2017 supplementing directive (eu) 2015/2366 of the european parliament and of the

council with regard to regulatory technical standards for strong customer authentication and common and secure open standards of communication (text with eea relevance.). *Official Journal of the European Union*, OJ L 69:23–43, 3 2018.

[7] Melisande Mual. Psd2 rts on secure communication and screen scraping. Interview, The Paypers, 05 2017. https://thepaypers.com/interviews/psd2-rts-on-secure-communication-and-screen-scraping/768765-38.

[8] Finextra. Open banking vs. screen scraping: looking ahead in 2019, 01 2019. Accessed: 2020-02-30.

[9] European Banking Authority. Opinion of the european banking authority. Technical report, 09 2017.

[10] European Banking Authority. Final report, draft regulatory technical standards on strong customer authentication and common and secure communication under article 98 of directive 2015/2366 (psd2)). 2 2017.

[11] John Hughes and Eve Maler. Security assertion markup language (saml) v2. 0 technical overview. *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*, pages 29–38, 2005. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html.

[12] Internet2 Scott Cantor. Saml v2.0 condition for delegation restriction version 1.0, 11 2009. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation.html.

[13] OASIS Standard. extensible access control markup language (xacml) version 3.0. 2013.

[14] Tobias Wich, Daniel Nemmert, and Detlef Hühnlein. Towards secure and standard-compliant implementations of the psd2 directive. *Open Identity Summit 2017*, 2017.

[15] Dick Hardt et al. The oauth 2.0 authorization framework. Technical report, Internet Engineering Task Force (IETF), 12 2012.

[16] Stripe. `https://stripe.com/en-fi`, 2019. Accessed: 2020-02-25.

[17] W Richards Adrion, Martha A Branstad, and John C Cherniavsky. Validation, verification, and testing of computer software. *ACM Computing Surveys (CSUR)*, 14(2):159–192, 1982.