Account and Transaction Protocol of the Open Banking Standard

Abdulaziz Almehrej[1], Leo Freitas[1] and Paolo Modesti[2]

[1] School of Computing, Newcastle University, UK
[2] Computer Science and Information Systems, Teesside University, UK

Abstract. To counteract the lack of competition and innovation in the financial services industry, the EU has issued the Second Payment Services Directive (PSD2) encouraging account servicing payment service providers to share data. The UK, similarly to other European countries, has promoted a standard API for data sharing: the Open Banking Standard. We present an overview of the result of a formal security analysis of the Account and Transaction API protocol.

1 Introduction

The lack of competition in the financial services industry has been one of the main factors that led the European Union to introduce the second version of the Payment Services Directive (PSD2) [14], which aims to improve competition by enabling and encouraging bank account holders to share, in a controlled and secure way, their account data. To provide a standard API for the sharing of customer data across different banks, the UK, similarly to other European countries, introduced the Open Banking Standard [13]. The regulation encompasses several API specifications suitable for different *Third Party Providers* (TPPs) who aim to service consumers that consent to sharing their data. The adoption of a standardised interface allows interoperability and simplifies the implementation of systems for sharing data between banks and TPPs.

Contribution In this paper, we present an overview of a formal security analysis of the Open Banking Standard APIs, focusing on the verification of the correctness of the Account and Transaction API protocol. The work relies on a previously proposed methodology [5] which provided a practical approach to protocol modelling and verification. The methodology utilises the Alice and Bob notation (AnB) [9] to specify a formal model of the protocol that can be formally verified with the OFMC model checker [2]. We formalised and verified a number of security goals that are implicit in the requirements. Although most goals were satisfied in our analysis, the lack of rigourous definition of security properties in the standard can be a source of ambiguity, potentially leading to different interpretations of the security requirements in the implementation. To the best of our knowledge, our model, fully presented in [1], is the first attempt to formally analyse Open Banking protocols. Recently, other authors [7] made an evaluation of the integration of a web application with the Danish Nordea's Open Banking APIs considering the security threats of the underlying technology, in light of OWASP Top 10 Web Application Security Risks list. However, they did not analyse the security of Open Banking itself considering and assessing security goals as we did. Therefore, we believe this formal analysis can be valuable for stakeholders considering the adoption of a standard that can have a significant and long impact on the efficiency and security of the financial sector.

2 Open Banking Standard

The Open Banking Standard [13] aims at two key outcomes. The first one is an open API for sharing data regarding the services offered by *Account Servicing Payment Service Providers* (ASPSPs), *e.g.* banks. The other one is an open API for sharing the account data of *Payment Service Users* (PSUs) provided by ASPSPs. Open Banking is not only concerned about the API endpoints (e.g. location of resources accessible by third parties, such as developers, to build banking and financial applications), but also about data and security standards. The *data standard* provides data models to the API data format. The *API standard* covers the API's operational requirements. The *security standard* covers API security requirements. An *Account Information Service Provider* (AISP) is a regulated entity allowed by ASPSPs to access a PSU's account data if the PSU provides their consent. This type of access is read-only as the AISPs are not expected to directly affect the payment accounts they are allowed access to. An AISP can then provide different services having the PSU's account and transaction data, including applications that provide a user-friendly view of the states of the different payment accounts held by the PSU, budgeting advice, price comparisons and product recommendations.

Account and Transaction Protocol The protocol is initiated with the PSU asking for information regarding their payment account(s) from an AISP (Step 1). The AISP then attempts to create an account access consent with the corresponding ASPSP, based on the access permissions agreed upon with the PSU. First, the AISP authenticates itself to the ASPSP through a client credential grant, which is an approach for machine-to-machine authentication. The ASPSP then provides the AISP with an access token used to request the creation of the consent resource (Step 2). At this point, the created account access consent has to be authorised to be used by the AISP to access the PSU's account data. This requires the PSUs to authenticate themselves to the ASPSP, followed by authorising the consent. During this phase, the PSU has to select the payment account(s) for which the chosen permissions should apply. The AISP then obtains an access token to the account data (Step 3). With this token, the AISP has to first retrieve the accessible accounts, including their unique IDs, through the accounts endpoint. The IDs can later be used to request the data of specific accounts (Step 4). To retrieve specific PSU account data (e.g. balances, transactions, direct debits, beneficiaries, *etc.*) the AISP will have to request the data via the appropriate link using the correct endpoint and method from the ASPSP.

3 Methodology and Security Goals

The formal verification of Open Banking API presented in this work is based on a protocol verification methodology proposed in [5]. The methodology utilises the Alice and Bob notation (AnB) [9] to specify a formal model of the protocol that can be formally verified through information flow (secrecy and authenticity) goals. Such notation abstracts from implementation details, but allows formal representation and analysis of the security-relevant characteristics of protocols.

An AnB specification comprises of several sections. The *Types* section declares the different identifiers used in the protocol. This includes the agents, constant and variable (random) numbers and transparent functions. Transparent functions are user-defined through their signature, thereby abstracting from their implementation details (*i.e.* they are uninterpreted). The *Knowledge* section describes the initial data each agent has before running the protocol. Fresh values are initialised at runtime. The information flow is described in the *Actions* section, where details about messages exchanged by agents are specified. Furthermore, the model can be used to verify specific security properties, such as (weak and strong) authentication and secrecy goals:

- **A weakly authenticates B on M**: agent A has evidence that the message M has been endorsed by agent B with the intention to send it to A (i.e. non-injective agreement [8]);
- **A authenticates B on M**: weak authentication plus evidence of the freshness of the message M (i.e. injective agreement [8]);
- **M secret between A, B**: message M is kept secret among listed agents.

The formal model captures the protocol requirements [12]. While the Open Banking API describes in details the information-flow, it lacks definitions of security goals that the exchanges between agents are meant to convey. Therefore, part of our work consisted in identifying suitable goals for the protocol model.

For the verification, we used the Open-Source Fixed-Point Model-Checker (OFMC) [10], a symbolic model-checker supporting the AnB notation. Moreover, the AnBx Compiler and Code Generator [11] was used to pre-process the model to benefit from a stricter type system and support the extension to AnB that allows named expression abstractions (*Definitions* section).

The goals we identified (and verified) are based on our understanding of the protocol and on its dependencies. For example, OAuth 2.0 security considerations [6, P.52-P.60], protocol use cases in [13, P.20-P.23] and our expectations of the protocol.

We identified eight goals: four on message secrecy, and four on authentication.

```
fAISPSecret(AISP) secret between AISP,aspspA             #G1
fPSUSecret(PSU) secret between PSU,aspspA                #G2
ClientToken, AuthToken secret between AISP,aspspA,aspspR  #G3
```

```
# FAILED initially + Fixed #A2.3
PSU authenticates aspspR on fGetIntent(Intent)              #G4
aspspR authenticates PSU on SelectedAccounts                #G5
PSU weakly authenticates AISP on ASPSPAuthPSUEndP,AISPEndP #G6

# FAILED + Fixed #A4.1  #A4.2
Accounts secret between AISP, aspspR                        #G7
AISP authenticates aspspR on Accounts                       #G8
```

Two goals (G1 and G2) are obvious: the exchanged secrets/credentials between the AISP and PSU and the authorisation server remain secret whilst requesting for a client token (Action 2.1 in the specification) and acquiring consent authorisation (A3.1.2 and A3.3.3). That is because if the AISP credentials are leaked (A2.1), many attacks would be possible (for instance [6, Sect. 10.2] discusses client impersonation). Another secrecy goal (G3) states that various exchanged tokens (A2.2–A2.3 and A3.3.4–A4.1) remain secret between the AISP and the authorisation and resource servers. As tokens are AISP bound, a compromised token cannot be directly used. However, [6, Sect. 10.3] requires tokens to be confidential, to prevent attacks involving valid token injection [6, Sect. 10.12]. These goals clearly indicate the inherited potential vulnerabilities of the Account and Transaction Protocol (ATP) dependencies. The final secrecy goal (G7) is about the resource server message to the AISP (A4.2) and is obvious: account information must remain secret.

The authentication goals relate to the PSU authenticating the consent resource to authorise (G4), the resource server authenticating the PSU's selected accounts information (G5) and the AISP authenticating the PSU's account information from the resource server (G8). This last goal between the AISP and the resource server is crucial in verifying the integrity of the account data sent to the AISP by the resource server. In addition to direct data modification, it is important to verify that old data cannot be replayed. For instance, in the case of affordability check, if the PSU was an intruder and modified the data, they could trick an AISP into providing a product they are not eligible for. This goal also enforces fraud detection: if the transactional data can be modified by an intruder to hide fraudulent activity. Given the redirections from the PSU to the authorisation server and AISP (A3.1.1 and A3.3.1–A3.3.2), we weakly authenticate that those endpoints cannot be modified by an intruder to help avoid redirected URI manipulation [6, Sect. 10.6] and phishing attacks [6, Sect. 10.11] (G6).

Model Development. The Open Banking ATP is complex and with multiple dependencies. The AnB model aims to provide an abstract and accurate view of its essential aspects and to verify key properties. The initial AnB model was overly detailed with unnecessary data exchanges. To reach the right level of abstraction, we then decided to first determine the protocol goals prior to abstracting. Even after such endevour, verification was unwieldy: it ran for over two days without response. As is common within model checking problems, state explosion must be tackled beyond abstracting details, abstract on irrelevant data.

Restricting the role of the PSU, where it had to be different from the AISP and servers, considerably reduced the state space. This led to termination with goal verification to be reduced to about seven hours. This enabled us to identify further steps to abstract related to data, which reduced the verification time to about six minutes. A final abstraction, related to the various TLS-related steps, was to abstract them using AnB bullet channels, used to model channels providing authentication and/or secrecy properties at the end-points. The internal efficiency of OFMC dealing with such channels led the final version to verify within eight seconds. This exponential efficiency (up to 5 orders of magnitude) increase is not uncommon in model checking problems, so long the right abstractions are taken alongside expert knowledge of the tool's implementation.

Model Correctness. We used the OFMC model checker [2] to verify the eight goals described above. At first, three goals (*G4*,*G7*,*G8*) about PSU intent authentication and account information secrecy and integrity failed. This led us to check these goals independently in order to study their reason for failure quickly. The witness for the PSU authentication failure (*G4*) relates to the resource server authenticating with an unknown agent rather than the PSU. This was fixed by having the resource server being aware of the PSU's identity early on when setting up the access consent with the AISP (A3.2). Thus, this failure identifies a previously undocumented vulnerability, which our modification fixes.

The account information goals fail due to a limitation of bullet channels: they do not protect against replay attacks, hence their use here allowed breaking both secrecy (*G7*) and integrity (*G8*). The intruder could respond to the AISP's request for account data by replaying a previous message. This breaks integrity as the response received by the AISP, and perceived to be the account data, has been modified. As the data replayed is known to the intruder, it also breaks secrecy of the account data. However, the TLS protocol does protect against message replay [4, P.93-P.94]. To deal with this limitation and ensure that freshness would resolve the issue, we modified the model to include a nonce generated and sent by the AISP when requesting for the PSU account data and is expected to be part of the response.

These modifications enable checking all goals for one session. Multiple sessions verification is important as there could be attacks relying on multiple concurrent protocol runs. Due to increased state space and limited hardware, we were unable to fully verify the model for two parallel sessions. As customary in under such conditions (e.g. [3] for iKP and SET), we were able to obtain partial results by increasing the search space up to the available resource limits (search space depth: 15 plies, 14.5GB RAM, 50 hours to run) without being able to reach any attack state.

4 Conclusion

The novel Open Banking Account and Transaction protocol is a security-critical protocol, which is being enforced on the largest banks in Europe. Given the protocol's significance and expected wider use, verifying its correctness is crucial. Our findings were disseminated as part of a presentation on PSD2 at a UK Finance event, with representatives from Visa and MasterCard, as well as several banks. Some of the identified goals were known, others not. The audience was particularly keen on the time/cost analysis. Our future work will focus on the modelling of the protocol's state and transparent functions specification in VDM-SL: this is aimed at discovering underlying vulnerabilities related to the myriad of dependant technologies (*e.g.* OAuth2, TLS, *etc.*).

References

1. Almehrej, A., Freitas, L., Modesti, P.: Security analysis of the Open Banking Account and Transaction API Protocol. ArXiv (2020), `https://arxiv.org/abs/2003.12776`
2. Basin, D., Mödersheim, S., Viganò, L.: Ofmc: A symbolic model checker for security protocols. International Journal of Information Security (June 2005)
3. Bugliesi, M., Calzavara, S., Mödersheim, S., Modesti, P.: Security protocol specification and verification with AnBx. Journal of Inf. Security and Applications **30**, 46–63 (2016). https://doi.org/10.1016/j.jisa.2016.05.004
4. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2 (August 2008), `https://tools.ietf.org/html/rfc5246`. Last accessed date [22 August 2019]
5. Freitas, L., Modesti, P., Emms, M.: A Methodology for Protocol Verification applied to EMV. In: Formal Methods: Foundations and Applications - 21th Brazilian Symposium, SBMF 2018, Salvador, Brazil, November 28-30, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11254. Springer (2018)
6. Hardt, D.: The OAuth 2.0 Authorization Framework (October 2012), `https://tools.ietf.org/html/rfc6749`. Last accessed date [22 August 2019]
7. Kellezi, D., Boegelund, C., Meng, W.: Towards secure Open Banking architecture: An evaluation with OWASP. In: International Conference on Network and System Security. pp. 185–198. Springer (2019)
8. Lowe, G.: A hierarchy of authentication specifications. In: CSFW'97, pp. 31–43. IEEE Computer Society Press (1997)
9. Mödersheim, S.: Algebraic properties in Alice and Bob notation. In: Int. Conf. on Availability, Reliability and Security (ARES 2009). pp. 433–440 (2009). https://doi.org/10.1109/ARES.2009.95
10. Mödersheim, S., Viganò, L.: The open-source fixed-point model checker for symbolic analysis of security protocols. In: Foundations of Security Analysis and Design V. pp. 166–194. Springer (2009)
11. Modesti, P.: AnBx: Automatic generation and verification of security protocols implementations. In: Foundations & Practice of Security. LNCS 9482, Springer (2015)
12. Open Banking Limited: Account and Transaction API Specification - v3.1.1, `https://tinyurl.com/qs643hq`. Last accessed date [22 August 2019]
13. Open Banking Working Group: The Open Banking Standard (February 2016)
14. The European Parliament and the Council of the European Union: DIRECTIVE (EU) 2015/2366. Official Journal of the European Union (November 2015)