Chapter 6

Automated Assistance to the Security Assessment of API for Financial Services

By Andrea Bisegna, Roberto Carbone, Mariano Ceccato, Salvatore Manfredi, Silvio Ranise, Giada Sciarretta, Alessandro Tomasi and Emanuele Viglianisi

DOI: 10.1561/9781680836875.ch6

Published in *Cyber-Physical Threat Intelligence for Critical Infrastructures Security: A Guide to Integrated Cyber-Physical Protection of Modern Critical Infrastructures* by John Soldatos, James Philpot and Gabriele Giunta (eds.). 2020. ISBN 978-1-68083-686-8. E-ISBN 978-1-68083-687-5.

Suggested citation: Andrea Bisegna *et al.* 2020. "Automated Assistance to the Security Assessment of API for Financial Services" in *Cyber-Physical Threat Intelligence for Critical Infrastructures Security: A Guide to Integrated Cyber-Physical Protection of Modern Critical Infrastructures*. Edited by John Soldatos, James Philpot and Gabriele Giunta. pp. 94–103. Now Publishers. DOI: 10.1561/9781680836875.ch6.

now
the essence of knowledge

This chapter presents the challenges related to the security assessment and the automated synthesis of mitigation measures of APIs for financial services. The focus is on the APIs supporting the implementation of the new Payment Services Directive [PSD2]. It also gives an overview of an innovative approach to address these challenges by (i) the automated identification and mitigation of security misconfigurations underlying sessions based on Transport Layer Security [TLS], which is ubiquitously used to build a foundation layer of security; and (ii) the automated penetration testing and synthesis of mitigations for the functionalities provided by APIs built on top of it, both business (e.g., payments) and security (e.g., authentication or authorization). The main novelty of the proposed approach lies in the tight integration of identification and mitigation phases by means of actionable measures that allow users to significantly strengthen the security posture of the entire API ecosystem.

The Regulatory Landscape

The Electronic Identification, Authentication and Trust Services [eIDAS] Regulation is the keystone regulation that defines requirements granting legal validity throughout the internal market to electronic transactions, equivalent to previous paper-based documents. To that end, it regulates Qualified Certificates (QC), electronic seals and signatures, and trust service providers. Security guidelines for the appropriate use of QCs have been published by ENISA QTS [ENISA QTS].

The Revised Directive on Payment Services [PSD2] is intended to protect and promote competition in the internal market by mandating that Account Servicing Payment Service Providers (ASPSP)—most likely traditional banking institutions—open their services to Third-party Providers (TPP) of Services including account information (AISP) and payment initiation (PISP) providers.

The Regulatory Technical Standard [RTS] defines requirements on the use of QCs for website authentication and electronic seals for communication among TPPs and ASPSPs. Guidance on the use of QCs is included in [EBA-OP-2018-7]. The [ETSI TS 119 495] standard defines how to implement the requirements of the RTS for use of QCs to meet the regulatory requirements of PSD2. For instance, it defines the requirements for Qualified Website Authentication Certificates (QWACs), and it clarifies specifically that a QWAC "should be used to establish a secure TLS channel to protect the communication (in the transport layer) from potential attackers on the network."

Open Banking API Security Recommendations

Under PSD2, banks are to provide an interface for third parties to access account information and perform operations (e.g., payments) on behalf of the account holder. The regulation does not specify technical solutions.

The Berlin Group standards and harmonization initiative proposes several possible approaches in its detailed "Access to Account (XS2A) Framework," including XML/JSON data model and associated messaging, as well as OpenAPI files to assist developers with implementation. At its core, XS2A provides a detailed description of REST API and their usage for the purposes of authentication of involved parties and authorization to access Service resources, such as Account Information (AIS), Payment Initiation (PIS), and Confirmation of Funds (PIIS).

The security of these APIs is based on both the transport and application layers. The first core technology explicitly identified by the guidelines is the Transport Layer Security [TLS] protocol: in particular, "the communication between the TPP and the ASPSP is always secured by using a TLS-connection using TLS version 1.2

or higher."[1] [XS2A-IG]. Additionally, [XS2A-IG] requires mutual authentication of TPP and ASPSP using eIDAS- and RTS-compliant QCs, which must include all the roles for which the TPP is authorized.

On the application layer, the core technology for authorization is the Open Authorization Protocol [OAuth2], in particular the "Authorisation Code Grant" flow is mandated for PIS and AIS. While other options are available and discussed below, OAuth is seen as preferable.

Strong Customer Authentication in XS2A

Strong Customer Authentication (SCA) is one of the main requirements set out by PSD2 (article 97) and RTS (Chapter III). The ASPSP must determine how to enforce SCA on a per-transaction basis, in compliance with those requirements.

In the XS2A framework, TPPs have three broad categories of options to allow compliance with SCA requirements:

1. Redirection—of users to their account holders and back to the TPP—using an authentication solution based on, e.g., OAuth 2, such as [OIDC];
2. Decoupling, in which the communication between user and account holder proceeds on an entirely separate channel; and
3. Embedding, in which the TPP has to embed the PSP's entire SCA flow in their own app.

Approach 3 involves a deep level of integration with every single account holder, which is much more work than the other options and requires an extremely high level of trust between the parties as it requires the sharing of user credentials. Approach 2 is more lightweight and scalable but incurs a higher risk of hanging business processes as the TPP must wait for notification of a completed operation on a separate channel. Option 1 is clearly seen as preferable.

Approach	Redirect (OAuth 2)	Decoupled	Embedded
SCA	Directly between user and PSP		Entirely at XS2A interface
Third-party Provider	Does not need detailed information about the individual steps of SCA	No impact on the user/provider interface	Needs SCA details for the user, e.g., displays challenge
Example	Standard interface, e.g., "scope" attribute of authentication request is linked to payment initiation or consent resource	Push notification with payment transaction details to dedicated mobile app or via any other application or device, independent of online banking front-end	Users enter username and password through their browser and are shown a QR code to be scanned

1. We note that TLS 1.2 is now officially marked as obsolete; TLS 1.3 is the current standard.

Automated Analysis of TLS

Transport Layer Security [TLS] consists of a set of cryptographic protocols designed to provide secure communications over a network. The popularity of TLS has encouraged attackers to find vulnerabilities and develop exploits. The variety of known attacks is the result of (i) maintaining backward compatibility and (ii) evolving use-case scenarios in which TLS is deployed.

One cannot "just deploy" TLS. Setting up a TLS server requires some amount of configuration, including:

- Choosing a set of cipher(s);
- Choosing the versions of TLS to be offered;
- Setting a certificate issued by a trustworthy CA;
- Coping with implementation issues (e.g., vulnerable libraries).

Several tools have been developed to help administrators deploy secure TLS instances. While such tools are quite effective in automatically finding vulnerabilities and issuing warnings about possible attacks, the burden of finding adequate mitigation measures is left to administrators who must first collect information about the identified problem and related fixes. Typically, such information is distributed in several sources ranging from scientific papers to blog posts. Even disregarding the effort to collect enough material to enact a mitigation, administrators should have enough skills to understand the often subtle details and turn the information in a concrete strategy to fix the problem. Additionally, each tool has varying degrees of coverage and does not specify mitigations for the issues identified. In other words, there is a problem in making the tools' reports actionable.

To address these issues, we developed TLS Assistant [MRS19], an open source tool that combines state-of-the-art TLS analyzers with a report system that shows the full set of viable attacks and suggests appropriate mitigations. The tool's architecture is summarized in Figure 6.1. Its goal is to assist an administrator in securing TLS configurations by:

- Detecting TLS and HTTPS misconfigurations;
- Providing
  - A brief attack description;
  - A mitigation description;
  - Mitigation code snippets (for Apache and nginx web server).

We successfully tested the use of TLSAssistant in the deployment of an eIDAS solution based on the new Italian identity cards before its submission for eIDAS notification, discovering that the first release was prone to Lucky 13 [AFP13]

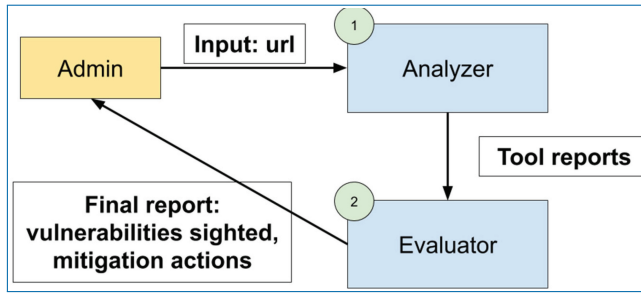

Figure 6.1. TLSAssistant workflow.

and 3SHAKE [BDLFPS14]. The server-side vulnerabilities issues were promptly patched, and the report was judged to be both easy to read and complete.

RESTful API Security Testing

API security issues can have a serious impact on all the applications that depend on them. Indeed, not only is there a growing business for API management [GMQAPI19] but there is a dedicated [OWASP API] top 10 security issue list, of which we highlight "API2:2019 Broken User Authentication" and "API7:2019 Security Misconfiguration." For example, the Harbor enterprise docker container management service was found to expose a "POST /api/users" registration API in which new users could self-register and inject a "HasAdminRole=true" attribute, thereby mounting an escalation of privilege attack remotely on any service exposing this API—see [CVE-2019-16097].

Specifically in the financial sector, a report by TrendMicro [HMcAM19] highlights challenges arising from the new paradigm, for instance, due to the different trust model underpinning the open banking framework. Among several issues, the basic building block of authorization protocols is still a work in progress.

While OAuth 2.0 is arguably the de facto standard for authorization protocols, it is a family of profiles tailored to specific use cases and scenarios. The higher security requirements inherent to the financial sector and the intrinsic novelty of exposing banking APIs to third parties have prompted the establishment of a working group for a dedicated Financial-grade API profile [FAPI], designed to harden OAuth under more adversarial circumstances—for instance, by assuming that sensitive tokens can be leaked by the user's browser or operating system, as is the case for many man-in-the-middle attacks, and allowing for the possibility that API endpoints may be misconfigured. Several mitigations have been proposed, for instance, requiring the use of mutual TLS between third parties and account providers; nevertheless, researchers in [FHK19] found that the expected security properties

did not appear to hold in all cases, for instance, allowing malicious actors to force an honest TPP to perform write-like operations (e.g., payment authorizations) from the attacker's device on an honest user's account.

We note that the use of OAuth on its own for authentication is considered improper; the OpenID Connect [OIDC] protocol builds an authentication layer on top of OAuth, and indeed, this is used in FAPI.

Automated Black-box Testing of RESTful APIs

We developed a synthesis of functional and security black-box tests, to appear in [VDC20]. It allows the automatic generation of test cases for RESTful API against errors and vulnerabilities. Indeed, errors can be indicators of potential vulnerabilities that may be exploited to mount attacks.

The tool's architecture is summarized in Figure 6.2. It takes as input an OpenAPI specification, containing all the necessary information to reach the API and the description of the endpoints. The first module generates an Operation Dependency Graph that, together with the Swagger specification, is given as input parameter to the Nominal Tester module in order to test the API's nominal behavior. The Nominal Tester outputs the nominal test cases and a set of structured reports that are given as input to both Error Tester and Security Tester. The former tests the

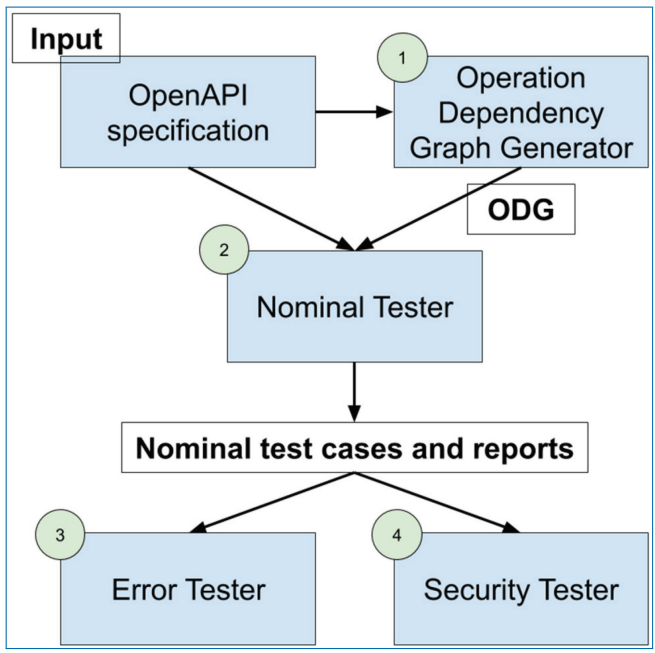

Figure 6.2. Black-box tool workflow.

correct error handling in case of malformed requests, for example, missing required parameters. The latter tests the API against common security vulnerabilities issues, such as SQL injection.

The execution scenarios generated by Nominal Tester, Error Tester, and Security Tester are run in the RESTful-API-under-test and its responses are monitored to spot the presence of programming mistakes, errors, and vulnerabilities. A set of *oracles* are defined to this aim, which check responses across multiple dimensions, such as error status code, data consistency with the OpenAPI specification, syntax and well-formed output data, traces of injection vulnerabilities.

Interesting execution scenarios generated by nominal, error and security testers are output as a set of test cases, consisting of JSON description of steps and java code using swagger codegen, to document and reproduce the issues.

OAuth/OIDC Testing

We also developed a tool for automated OAuth/OIDC penetration testing as a plug-in for the Burp Suite, designed to be integrated in our security training and pen-testing environment Micro-ID-Gym [BCMOPR19]. Our plug-in performs both passive and active tests over the traffic generated during an OIDC flow.

Passive tests do not interfere with the flow itself but analyze the recorded traffic, checking, for instance, standard compliance—whether exchanged messages conform to specifications—and Cross-Site Request Forgery (CSRF) protection—e.g., by correct implementation of Proof Key for Code Exchange (PKCE). Active tests verify the behavior of the endpoints when subject to unexpected, modified, or removed input parameters during the OAuth flow.

The plug-in is built on top of Burp Proxy, a tool which allows testers to intercept all requests and responses and leverages the selenium-webdriver browser automation library. The input is a recorded test track, used as a guide for a selenium instance. The track contains the instructions to guide the selenium driver through an OAuth/OIDC flow. The track can be played back so that a tester may observe whether the browser, controlled by the selenium driver, is performing as expected. The tool is designed to pinpoint the step of the flow in which incorrect behavior has been sighted, and courses of action to mitigate against it are to be integrated.

Summary

Our proposed approach to TLS and API security is one that integrates the generation of actionable intelligence and offers concrete courses of action for the

mitigation of vulnerabilities. Our ongoing work includes the integration of TLSAssistant in the FinSec platform, the identification of compliance impacts of identified vulnerabilities, and models for continuous risk assessment. In future work, we aim at extending API testing with new penetration testing functionalities, bundle them to build a set of cooperating security services, and integrate the resulting component in a suitable platform.

Acknowledgments

Black-box and white-box security tests for REST API were developed as part of Teîchos, an EIT Digital Finance project.

TLSAssistant was developed in a joint lab with IPZS and is currently being enhanced and integrated in FINSEC, a H2020 Critical Infrastructure Innovation Action project (Contract Number: 786727), which is co-funded by the European Commission in the scope of its H2020 program.

References

[AFP13] N. J. Al Fardan and K. G. Paterson: "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols." 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, 2013, pp. 526–540, doi: 10.1109/SP.2013.42.

[BDLFPS14] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Pironti and P. Strub: "Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS". IEEE Symposium on Security and Privacy 2014: 98–113.

[BCMOPR19] A. Bisegna, R. Carbone, I. Martini, V. Odorizzi, G. Pellizzari and S. Ranise: "Micro-Id-Gym: Identity Management Workouts with Container-Based Microservices." IJISC 8 (1), pp. 45–50, 2019.06.28.

[CVE-2019-16097] NIST National Vulnerability Database: Common Vulnerabilities and Exposures #2019-16097. URL: https://nvd.nist.gov/vuln/detail/CVE-2019-16097

[EBA-OP-2018-7] "Opinion of the European Banking Authority on the use of eIDAS certificates under the RTS on SCA and CSC." URL: https://eba.europa.eu/file/58802/

[eIDAS] "Regulation (EU) No. 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC." URL: http://data.europa.eu/eli/reg/2014/910/oj

[ENISA QTS] "ENISA studies on qualified trust services." URL: https://www.enisa.europa.eu/topics/trust-services/qualified-trust-services

[ETSI TS 119 495] "Electronic Signatures and Infrastructures (ESI); Sector Specific Requirements; Qualified Certificate Profiles and TSP Policy Requirements under the payment services Directive (EU) 2015/2366". V1.4.1, November 2019. URL: https://www.etsi.org/standards-search#page=1&search=TS119495

[FAPI] OpenID Financial-grade API (FAPI) Working Group. URL: https://openid.net/wg/fapi/

[FHK19] D. Fett, P. Hosseyni and R. Kuesters: "An Extensive Formal Security Analysis of the OpenID Financial-Grade API." Proceedings of S&P 2019, pp. 1054–1072. doi: 10.1109/SP.2019.00067.

[GMQAPI19] Gartner "Magic Quadrant for Full Life Cycle API Management" 2019. URL: https://www.gartner.com/doc/reprints?id=1-1OGPZC68&ct=190905&st=sb

[HMcAM19] F. Hacquebord, R. McArdle, F. Mercês and D. Sancho: "Ready or Not for PSD2: The Risks of Open Banking." TrendMicro, 2019. URL: https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/the-risks-of-open-banking-are-banks-and-their-customers-ready-for-psd2

[MRS19] S. Manfredi, S. Ranise and G. Sciarretta: "Lost in TLS? No More! Assisted Deployment of Secure TLS Configurations." In: DBSec 2019: Data and Applications Security and Privacy XXXIII pp. 201–220. LNCS 11559. doi: 10.1007/978-3-030-22479-0_11. URL: https://stfbk.github.io/tools/TLSAssistant

[OAuth2] "The OAuth 2.0 Authorization Framework." IETF proposed standard. URL: https://tools.ietf.org/html/rfc6749

[OIDC] "OpenID Connect". URL: https://openid.net/connect/

[OWASP API] OWASP foundation Top 10 API security issue list. URL: https://owasp.org/www-project-api-security/

[PSD2] "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." URL: http://data.europa.eu/eli/dir/2015/2366/oj

[RTS] "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)." URL: http://data.europa.eu/eli/reg_del/2018/389/oj

[TLS] "The Transport Layer Security (TLS) Protocol". IETF proposed standard. URL: https://tools.ietf.org/html/rfc8446 (v1.3), https://tools.ietf.org/html/rfc5246 (v1.2 – obsolete).

[VDC20] E. Viglianisi, M. Dallago, M. Ceccato: "RESTTESTGEN: Automated Black-Box Testing of RESTful APIs." Accepted to appear in ICST 2020 Research Papers.

[XS2A-OR] NextGenPSD2 Access to Account Interoperability Framework – Operational Rules V1.3 2018.12.21. URL: https://www.berlin-group.org/nextgenpsd2-downloads

[XS2A-IG] NextGenPSD2 Access to Account Interoperability Framework – Implementation Guidelines V1.3.4 2019.07.05. URL: https://www.berlin-group.org/nextgenpsd2-downloads