Micro-Id-Gym: Identity Management Workouts with Container-Based Microservices

Andrea Bisegna[1], Roberto Carbone[1][0000−0003−2853−4269], Ivan Martini[2], Valentina Odorizzi[2], Giulio Pellizzari[2], and Silvio Ranise[1][0000−0001−7269−9285]

[1] Security & Trust, FBK, Trento, Italy {a.bisegna,carbone,ranise}@fbk.eu
[2] University of Trento, Trento, Italy
{ivan.martini,valentina.odorizzi,giulio.pellizzari}@studenti.unitn.it

Abstract. Identity Management (IdM) solutions are increasingly important for building trust in current and future digital ecosystems. Unfortunately, not only their secure deployment but even their usage are non-trivial activities that require a good level of security awareness. For this, we introduce Micro-Id-Gym, an easy to configure training environment in which users can develop hands-on experiences on how IdM solutions work and better understand the underlying security issues.

1 Introduction

As digital services are growing and the Internet of Things is getting larger, individuals need to be in control of their digital identities to manage their interactions with increasingly complex digital services. Failing to design and deploy usable and secure solutions for Identity Management (IdM) may result in a lack of trust in digital ecosystems with negative impact in terms of economy (e.g., higher fees on financial transactions) and society (e.g., limitation of electronic healthcare solutions). To avoid such negative effects, it is crucial to augment the understanding of the main security problems underlying the most widely adopted IdM solutions. This is important not only to security practitioners but virtually anyone dealing with online services because they are likely to use one of the available infrastructures provided by either enterprises (e.g., Google or Facebook) or public infrastructures for digital identities like the Europen electronic IDentification, Authentication and trust Services (eIDAS)[3] or the Italian Public System for Digital Identity (SPID).[4]

Unfortunately, increasing the security awareness on IdM solutions is far from being a trivial task. This is so because users get bogged down in the miriads of practical technical challenges that they are required to tame when trying to deploy or understand IdM solutions in realistic scenarios. To alleviate these problems, we propose an easy to configure training environment, called Micro-Id-Gym,[5] in which users can develop hands-on experiences on how IdM solutions

[3] https://ec.europa.eu/digital-single-market/en/policies/trust-services-and-eidentification
[4] https://www.agid.gov.it/it/piattaforme/spid
[5] https://sites.google.com/fbk.eu/micro-id-gym

work and increase their awareness related to the underlying security issues. We discuss in more details the motivations and design goals of Micro-Id-Gym in Section 2 by considering one of the most important IdM solutions, namely Single Sign-On (SSO). Then, we describe Micro-Id-Gym and a typical user experience in Section 3. We conclude and overview future work in Section 4.

2 A Motivating Use Case Scenario

To illustrate which are the problems related to understanding complex IdM solutions, we consider the SAML 2.0 Web Browser SSO Profile [6] (abbreviated with SAML SSO in the following), that is one of the most widely adopted authentication protocols.

Figure 1 shows a Message Sequence Chart (MSC) of the main steps of the SAML SSO protocol in which three entities are involved: a client (C), an identity provider (IdP) and a service provider (SP). The goal of C (typically a web browser with which a user interacts) is to get access to a service or a resource provided by SP. IdP authenticates C and issues corresponding authentication assertions that are trusted by SP (the trust relationship is depicted with a handshake in the figure). SP uses the assertions generated by IdP to decide on C's entitlement to the requested service or resource. A brief description of the protocol is as follows. C asks SP to provide the resource located at URI (step S1). SP then sends to C an HTTP redirect response (status code `302`) for IdP, containing an authentication request AuthReq(ID,SP), where ID is a (pseudo-)randomly generated string uniquely identifying the request (steps A1 and A2). A frequent implementation choice is to use the `RelayState` field to carry the original URI that C has requested (see [6]). Then IdP challenges C to provide valid credentials (dotted double arrows in the figure): this is not specified in the standard of the SAML SSO so to accommodate any authentication process offered by IdP. If the authentication succeeds, IdP builds an authentication assertion as the tuple AA = AuthAssert(ID, C, IdP, SP) and embeds it in a response message Resp = Response(ID, SP, IdP, $\{AA\}_{\mathrm{K}_{\mathrm{IdP}}^{-1}}$) where $\{AA\}_{\mathrm{K}_{\mathrm{IdP}}^{-1}}$ is the assertion signed with IdP's private key (the key icon in the figure). IdP then places Resp and the value of RelayState received from SP into an HTML form and sends the result back to C in an HTTP response (step A3) together with some script that automatically posts the form to SP (step A4). Finally, the SP sends to C an accepted HTTP response (status code `200`) containing the requested resource (step S2).

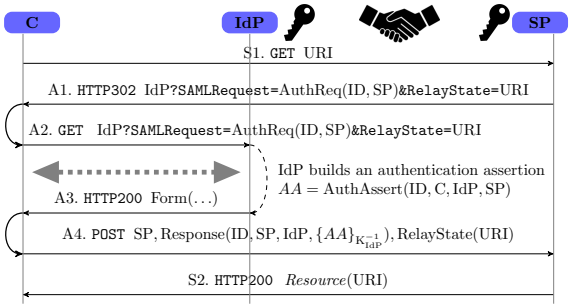

Fig. 1. The SAML SSO protocol [2].

The description of the SAML SSO protocol abstracts away several details that are crucial for security; we highlight some of the most important in the following. The trust relationship between SP and IdP (handshake icon) must be established before running the protocol by distributing appropriate meta-data between the two entities. The three entities shall exchange messages by using the Transport Layer Security (TLS) protocol for confidentiality and integrity. For this, both IdP and SP must have valid X.509 certificates. Users need to register at the IdP to receive credentials. Besides these aspects, additional details play an important role in the security of SAML SSO deployments including the configurations of auxiliary modules (e.g., omitting checks on macro expansion by XML parsers used for SAML assertions may lead to Denial of Service attacks [5]) and the format of messages (e.g., forgetting a field in a message may lead to man-in-the-middle attacks [1]).

The secure deployment of SAML SSO is a complex and error prone activity that requires a high level of awareness in several and heterogeneous aspects. It is thus not surprising that attacks have been discovered over the years (e.g., [3]). For instance, if the integrity of the RelayState field (cf. steps A1, A2, and A4 in Figure 1) is not adequately protected, it may allow hackers to mount injection attacks to SPs (we discuss this below). Notice that the SAML Approved Errata [7] emphasizes the importance of the sanitization of the RelayState.

3 Micro-Id-Gym

The main idea underlying Micro-Id-Gym is three-fold. First, its architecture uses container-based microservices and exploits the possibility to set-up a local network among them. So, each entity participating in an IdM solution (e.g., IdP and SP for the SAML SSO discussed in Section 2) is encapsulated in a microservice that may interact with the others through standard network primitives (e.g., HTTP requests and redirections). Additionally, containers—in our case those offered by Docker[6]—provide for high flexibility in terms of configuration and allow for managing the various technical aspects that are needed to deploy IdM solutions and reproduce realistic scenarios. Second, state-of-the-art penetration testing tools, such as Burp[7] and OWASP ZAP, [8] can be used to monitor the exchange of messages among the microservices in the local network so that users can understand when and which messages are exchanged among the various entities. This allows users for a better understanding of how IdM protocols work. To ease this phase, the penetration testing tool communicates with an application that animates a MSC of the protocol under consideration (e.g., the one depicted in Figure 1). We hope that the integration of standard penetration testing tools in Micro-Id-Gym would make their usage more wide-spread and ultimately contribute to increase the understanding of security problems of web applications. The third idea underlying Micro-Id-Gym is to provide an available

[6] `https://www.docker.com`
[7] `https://portswigger.net`
[8] `https://www.zaproxy.org`

set of pre-configured deployments that allow even users with a limited security background to get an hands-on experience of IdM solutions without getting lost in low level details and (hopefully) increase their understanding.

Micro-Id-Gym structures the use experience in four phases. First, *set-up* consists of describing (selected parts of) the configurations of the entities playing a role in the IdM solution and their relationships. Second, *routine* amounts to familiarizing users with the intended behavior of the IdM solution while assuming that there are no malicious entities. Third, *deviation* illustrates one or several ways in which the intended behavior of the IdM solution can be abused by attackers in presence of (exploitable) vulnerabilities. Fourth, *mitigation* builds on the description of the vulnerabilities identified in the previous step to propose possible mitigations to make their exploitation more difficult. For the sake of concreteness, we illustrate below these steps on the SAML SSO scenario of Section 2.

3.1 An overview of the user experience with Micro-Id-Gym

We elaborate on the injection attack to the SAML SSO protocol that exploits the lack of integrity checks on the RelayState field introduced in Section 2.

Set-up. The instructor guides the user to load the available scenario allowing for the exploitation of the RelayState vulnerability. This allows for the deployment of an IdP (e.g., Shibboleth[9]) and an SP (e.g., an SP based on Java), the assignment of X.509 certificates for TLS connections, the establishment of a trust relationship between them (federation), the formation of a virtual network by using Docker compose tool,[10] and the creation of an OpenLDAP module for handling user credentials. The instructor shows where configuration files are stored, discusses selected parts of them; for instance, he/she points outs that no sanitization is performed on the RelayState field by the IdP. Finally, he/she illustrates the metadata exchanged for the federation of IdP and SP.

Routine. The instructor explains how to perform the step-by-step execution of the SAML SSO protocol. Users are presented with the situation depicted in Figure 2 where they can see the MSC of the protocol (top left corner), the Burp tool (right half), and the browser of the user (bottom left corner). The SAML SSO protocol is started by the user by interacting with the browser; Burp is configured to intercept relevant messages and to interact with the application animating the MSC. The goal of this step is to make students familiar with the protocol and bridge the gap between the high-level view specified in the MSC and the actual implementation.

[9] `https://www.shibboleth.net`
[10] `https://docs.docker.com/compose`

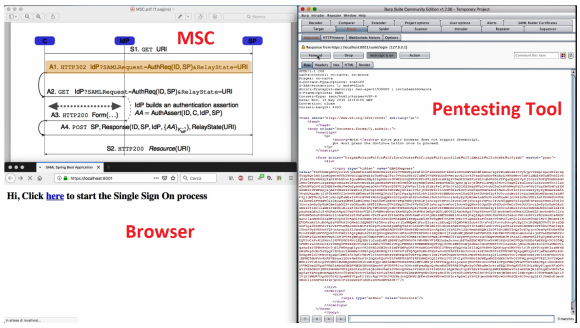

Fig. 2. Micro-Id-Gym: running the SAML SSO protocol.

Deviation. Once users are familiar with the protocol, the instructor discusses the possibility to use Burp not only to intercept messages but also to modify them and mount an attack by exploiting one or more vulnerabilities. As discussed in Section 2, the IdP in this scenario does not sanitize the RelayState field and this lack may be used as a vector for an injection attack. The instructor suggests to insert malicious code to be executed on the SP into the RelayState field of message A2—resulting from a redirection from the SP to IdP via the browser relay, recall Figure 1. An example of malicious code is a query exfiltrating sensitive information from the database handled by the SP whose results that can be returned as the resource in step S2. Once users have understood how the attack works, the instructor observes that, on the one hand, all SPs federated with the IdP are potential victims of the attack; but, on the other, for the attack to be successful the SP shall be vulnerable to the injection attack made possible by the modification of the RelayState.

Mitigation. Based on the anatomy of the attack and vulnerability performed in the previous step, the instructor explains that the best security practice to mitigate the attack is to update the configuration of the IdP to enable the sanitization of the RelayState field. After discussing where and how to modify the configuration file and to restart the IdP, the instructor asks users to attempt to mount the same injection attack and verify that it is not possible anymore.

4 Conclusions and Future Work

We have described Micro-Id-Gym, an easy to configure training environment in which users can develop hands-on experiences on how SAML SSO solutions work and better understand the underlying security issues. For ease of configuration and deployment, Micro-Id-Gym uses container-based microservices and state-of-the-art penetration testing tools. This allows one to create a catalogue of realistic scenarios in which different vulnerabilities and attacks can be re-created, analyzed, and mitigated without getting bogged down in the plethora of practical technical challenges that one is faced to tame when deploying complex SAML SSO solutions (especially in case of users with limited security skills).

There are two main lines of future developments in Micro-Id-Gym: (i) expand the catalogue of scenarios demonstrating vulnerabilities, attacks, and mitigations for SAML SSO, and (ii) provide support for other IdM solutions, including OAuth 2.0 [4] and OpenID Connect [8].

References

1. Armando, A., Carbone, R., Compagna, L., Cuellar, J., Tobarra, L.: Formal Analysis of SAML 2.0 Web Browser Single Sign-on: Breaking the SAML-based Single Sign-on for Google Apps. In: FMSE '08. ACM, New York, NY, USA (2008)
2. Armando, A., Carbone, R., Compagna, L., Cuéllar, J., Pellegrino, G., Sorniotti, A.: An Authentication Flaw in Browser-based Single Sign-On Protocols: Impact and Remediations. Computers & Security **33**, 41 – 58 (2013)
3. Engelbertz, N., Erinola, N., Herring, D., Somorovsky, J., Mladenov, V., Schwenk, J.: Security Analysis of eIDAS–The Cross-Country Authentication Scheme in Europe. In: 12th USENIX Workshop on Offensive Technologies (WOOT 18) (2018)
4. Hardt, D.: The OAuth 2.0 Authorization Framework (2012), IETF
5. Mladenov, V.: On the Security of Single Sign-On. Ph.D. thesis, Ruhr-Universität Bochum (2017)
6. OASIS: SAML V2.0 Tech. Overview. `http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf` (March 2008)
7. OASIS: SAML V2.0 Approved Errata. `http://docs.oasis-open.org/security/saml/v2.0/sstc-saml-approved-errata-2.0.pdf` (May 2012)
8. Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C.: OpenID Connect Core 1.0 incorporating errata set 1 (2014), OIDF