Micro-Id-Gym: a Flexible Tool for Pentesting Identity Management Protocols in the Wild and in the Laboratory*

Andrea Bisegna[1,2][0000−0002−8055−5262], Roberto Carbone[1][0000−0003−2853−4269], Giulio Pellizzari[3][0000−0001−6455−780X], and Silvio Ranise[1,4][0000−0001−7269−9285]

[1] Security & Trust, Fondazione Bruno Kessler, Trento (Italy)
{a.bisegna, carbone, ranise}@fbk.eu
[2] DIBRIS, University of Genova, Genova (Italy)
[3] DISI, University of Trento, Trento (Italy)
giulio.pellizzari@studenti.unitn.it
[4] Department of Mathematics, University of Trento, Trento (Italy)

Abstract. Identity Management (IdM) solutions are increasingly important for digital infrastructures of both enterprises and public administrations. Their security is a mandatory pre-requisite 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. In order to test whether known exploits can be reproduced in different environments, better understand their effects and facilitate the discovery of new vulnerabilities, we need to have a reliable testbed. For this, we present Micro-Id-Gym which abstractly supports two main activities: the creation of sandboxes with an IdM protocol deployment and the pentesting of IdM protocol deployments in the wild or in the laboratory (on the created sandboxes).

Keywords: security protocols · penetration testing · OAuth

1 Introduction

Identity Management (IdM) solutions are increasingly important for building trust in current and future digital ecosystems. The design and implementation of the IdM protocols underlying the most widely adopted IdM solutions is notoriously error-prone, as witnessed by several vulnerabilities reported in the last few years [3].

The activities for a pentester, especially in digital identity scenarios, require a deep knowledge of the protocols and the related implementation aspects. The lack of compliance with the IdM protocols defined in the standards or a missing check of the value of an HTTP messages can lead also to major security

* Partially supported by the innovation activity 19183 “Teîchos” of the Digital Finance action line of the EIT Digital, and by the joint laboratory “DigiMat Lab” between FBK and the Italian National Mint and Printing House (IPZS).

problems. Then considering also the problem of compliance with the Payment Services Directive 2 (PSD2) [1] regulations, the activities for a pentester increase significantly. Thus it is non-trivial for a pentester to have the skills and technical knowledge to perform all the pentesting activities required to ensure proper security posture for IdM protocols.

Several tools for automatic pentesting exist, but they usually target specific vulnerabilities, and few of them are able to spot all the relevant vulnerabilities for IdM protocols. In addition, in case a vulnerability has been detected, the burden of finding adequate mitigation measures is completely left on the tester who must also collect information about the identified problem and related fixes. Typically, such information is distributed in several sources ranging from the official standards and related security considerations to scientific papers addressing specific novel vulnerabilities.

Another issue with IdM protocols is the fact that in many cases it is not possible to perform a pentesting in the wild in the production environment due to several attacks with high impact like DoS or identity theft with serious legal implications. Therefore it is desirable to reproduce the production solution in a controlled environment. Unfortunately, the steps to reproduce the production environment are complicated and it is not always possible to create the right conditions to be able to spot the same vulnerabilities as in the production environment.

For this, we propose Micro-Id-Gym, offering on the one hand (in the laboratory) an easy way to configure the production environment in a sandbox where pentesters can develop hands-on experiences on how IdM solutions work, performing attacks with high impacts and better understand the underlying security issues. On the other hand (in the wild) a set of pentesting tools for the automatic security analysis of IdM protocols are provided. In [4] we have provided a high level overview of the main idea behind Micro-Id-Gym where the main focus was on the educational purposes. In this paper we are focusing on real environments describing the architecture and the main technical details of Micro-Id-Gym. We make the following four main contributions:

- We have provided a flexible environment for pentesting IdM protocols, which provides a set of deployments (by using container-based micro-services) and exploits the possibility to federate and set-up a local network among them.
- We have provided pentesting tools for the automatic security analysis of IdM protocols. To ease this phase, the penetration testing tools communicates with an application (MSC Drawer) that animates a message sequence chart (MSC) of the IdM protocols under consideration.
- We have assessed the penetration testing tools by analyzing a deployed service for PSD2 in the wild: the MSC Drawer has been extremely helpful to quickly show the differences between the expected MSC and the one obtained by analyzing the service. Then, we performed a finer-grained security analysis by reporting other relevant issues.
- We have assessed the environment for pentesting and the MSC Drawer in the laboratory experience: we used Micro-Id-Gym to create the proper experience,

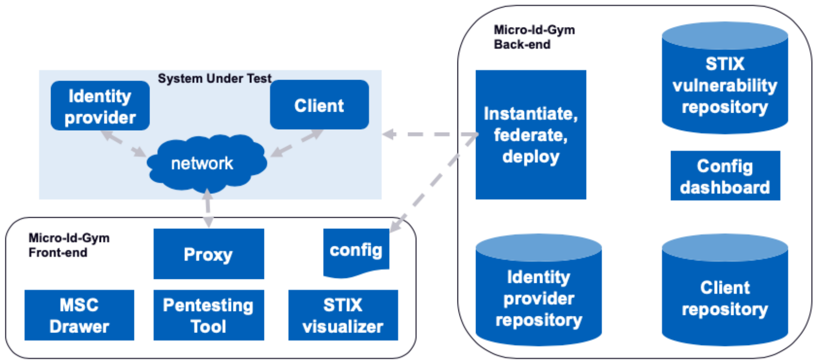


Fig. 1. Overview of Micro-Id-Gym.

by considering vulnerable scenarios, and assessed the effectiveness of MSC Drawer to help in finding vulnerabilities.

Plan of the paper. In Section 2 we give an overview of Micro-Id-Gym. Then, we provide more details of Micro-Id-Gym components in Section 3. To evaluate the effectiveness of Micro-Id-Gym, Section 4 reports the use of Micro-Id-Gym in the wild. Section 5 presents the results of a user study involving bachelor and master degree students in the laboratory. We conclude and overview future work in Section 6.

2 Overview of Micro-Id-Gym

To assist system administrators and testers in the deployment and pentesting of IdM protocol instances we propose Micro-Id-Gym. In this section we provide an overview of the tool before giving the details of the various components in the rest of the paper.

The IdM protocols are designed specifically for the transfer of authentication information and consist of a series of messages in a preset sequence designed to protect data as it travels through networks or between servers. All the IdM protocols provide standards for security to simplify access management, help in compliance, and create a uniform system for handling interactions between users and systems. For IdM protocols we refer to the web protocols where a Client (C) relies on a trusted third-party server called Identity Provider (IdP) for user authentication. Security Assertion Markup Language Single Sign-On v2.0 (hereafter SAML) [8] and OAuth 2.0 (OAuth) [6] / OpenID Connect (OIDC) [10] are two of the most known protocols providing this authentication pattern despite the fact that different names may be used to refer to the aforementioned entities. In the case of SAML, C takes the name of Service Provider (SP).

Abstractly, Micro-Id-Gym supports two main activities: pentesting of IdM protocol deployments and creating sandboxes with an IdM protocol deployment.

The first activity can be carried out on a deployment in the wild or one in a sandbox—say, in the laboratory—obtained by the second activity. We observe that the capability of creating sandboxes is useful to perform attacks with high impact like DoS or identity theft, the former dangerous for the service itself while the latter for legal and compliance issues.

Fig. 1 shows a high level view of the architecture of Micro-Id-Gym composed of two main components, namely Micro-Id-Gym Frontend (that supports pentesting) and the Micro-Id-Gym Backend (that supports the creation of a sandbox). In Fig. 1, it is also depicted the IdM protocol deployment that is supposed to be tested for security problems (called System Under Test, SUT). Below, we provide an overview of the two main components.

2.1 Micro-Id-Gym Backend

The Micro-Id-Gym Backend is used to recreate locally a sandbox as an instance of an IdP and a C and it can be done by uploading the own proprietary sandbox or by composing a new sandbox choosing the instances of IdPs and Cs provided by the tool as depicted in Fig. 1 in the IdP and C repositories. All the provided instances have been collected so far during our experience of using Micro-Id-Gym and they satisfy the requirements to be compatible with Micro-Id-Gym namely to have an instance of the IdP, at least one for the C and to use SAML or OAuth/OIDC as IdM protocols. The backend is also in charge to instantiate the selected instances, to federate each other, to exchange the required metadata, to perform the deployment of the sandbox and to set up the local network.

The process of creating a sandbox is straightforward: from the Dashboard the user picks Cs and an IdP from a set of available IdM protocol instances (Cs and IdPs repositories in Fig. 1) and sets the URLs and ports. Once the sandbox has been selected, the tool automatically connects the chosen instances and run them in the SUT which includes the entities of IdP and Cs properly instantiated, federated and deployed.

The Micro-Id-Gym Backend provides also Cyber Threat Intelligence (CTI) information useful for assessing vulnerabilities and threats related to the chosen instances. These data follow the Structured Threat Information Expression (STIX) format proposed by OASIS CTI TC.[5] This information is very useful since it immediately makes the pentester aware of possible specific attacks of that protocol known in the literature.

2.2 Micro-Id-Gym Frontend

The Micro-Id-Gym Frontend consists of tools to support user pentesting activities on the SUT, namely a Proxy, a set of Pentesting Tools, and two tools called MSC Drawer and MSC STIX Visualizer. As already mentioned, the SUT can be a sandbox or any IdM protocol available on Internet.

[5] https://www.oasis-open.org/committees/cti/

Proxy. It is a web proxy tool used to intercept the traffic between the user agent (i.e. a web browser) and the SUT. It provides a set of APIs used by the pentesting tools to inspect, modify, replay and drop the intercepted messages.

Pentesting Tool. It supports a user to perform pentesting of an IdM protocol deployment, by providing instruments to automatically detect security issues. The tools perform both passive and active tests. Passive tests analyze the HTTP messages exchanged between the browser and the servers, while active tests also intercept and modify those messages. An example of a passive test can be to check whether the format of the exchanged messages is compliant to what prescribed by the standard. Instead, an example of an active test can be the modification at run time of a parameter referred as required by the standard. With these tests, it is possible to detect vulnerabilities specifically due to an incorrect implementation of the IdM protocols. For instance, in case of a SAML implementation, the Pentesting Tool can identify a vulnerability leading to man-in-the-middle attack due to an incorrect implementation of the `RelayState` parameter, an identifier for the resource at the SP that the IdP will redirect the user to after successful login. All the security issues identified by the tools are reported in a table, including the suggestions to mitigate them.

MSC Drawer. The messages intercepted by the Proxy are then passed to the MSC Drawer which represents their flow as a MSC. The MSC Drawer provides a graphical overview of the authentication flow and allows easier inspection of the exchanged messages. For each HTTP message, the pentester can dive into headers, parameters, and body. Usually the standards for IdM protocols prescribe which are the mandatory/optional messages and their format, and the endpoints to invoke. Still, they usually do not prescribe anything about what happens between two subsequent requests to an endpoint. The messages of the standard can be thus interleaved with other "spurious" messages. For spurious messages we mean any HTTP traffic between two subsequent invocations prescribed by the standard (e.g., advertisements and images). Thus, being able to extract information about the standard is an extremely time consuming task for a pentester. Available state-of-the-art web proxy tools provide searching features, but it is still difficult to grasp the main messages, by selecting the relevant information referring to the standard, among the spurious messages.

MSC STIX Visualizer. It provides a graph of CTI information taken from the STIX vulnerability repository related to the intercepted authentication flow, currently only for SAML. Using the MSC Drawer UI the pentester can choose the granularity of CTI information he wants to look for. For instance, in case of a SUT using the SAML protocol, the pentester can look for CTI information regarding the `RelayState` parameter or, more generally, for CTI information related to a SAML IdP. The combination of these features with the pentesting tools makes the process of vulnerability identification and cyber risk assessment easier.

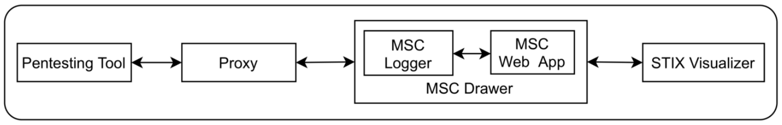


Fig. 2. Micro-Id-Gym Frontend.

Usage of the Micro-Id-Gym Frontend. The tools of the Micro-Id-Gym Frontend are used for the analysis of the HTTP messages generated during the authentication flow. The first step towards this process is to perform the authentication on the SUT. The messages exchanged during this process are displayed in the MSC Drawer. This is useful because at first glance the pentester can recognize whether the SUT follows the expected flow or not. The second step is to execute the automated tests provided by the Pentesting Tools. These automated tests verify if the SUT suffers the vulnerabilities tested by the tools. If a test was successful, the result will report the discovered vulnerability, otherwise, no alert will be reported.

Thanks to the vulnerability results, the pentester can identify where it is exposed, thus he knows where it has to be patched. Furthermore, using the CTI information provided by MSC STIX Visualizer, the pentester can assess how the vulnerability can be exploited and how severe it is.

3 The Components of Micro-Id-Gym

In this section we provide more details about components of the Micro-Id-Gym Frontend and the Micro-Id-Gym Backend.

3.1 The Components of Micro-Id-Gym Frontend

As depicted in Fig. 2, the Micro-Id-Gym Frontend is composed by Pentesting Tool, MSC Drawer (consisting of MSC Logger and MSC WebApp), MSC STIX Visualizer and a Proxy.

Proxy. It is a web proxy tool that intercepts the HTTP traffic between a browser and the servers of the SUT. It offers functionalities for inspecting, collecting, and modifying the HTTP messages, which are leveraged by the Pentesting Tool and MSC Drawer.

MSC Drawer. It is a tool for drawing MSC and allows the pentesters to quickly select the relevant messages, being able to spot a potential gap w.r.t. what prescribed by the standard and it is extremely helpful to save time, being more effective. IdM protocols are often expressed as a MSC with the advantage to immediately detect any incorrectness in the messages of the MSC and consequently

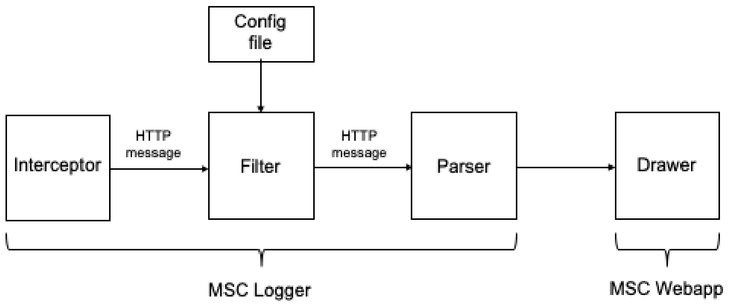


Fig. 3. MSC Drawer components.

identify any flaws. We evaluated some of these claims by performing a classroom experience, described in Section 5.

The Fig. 3 depicts the main components of the MSC Drawer. The MSC Logger is a Proxy's plugin reponsible to capture a selection of the HTTP messages, to filter and parse them according to the specific configuration, and send them through API to the MSC WebApp, a web application exposing a set of Restful API and responsible to draw a MSC.

As depicted in Fig. 3, the MSC Logger has the following functionalities:

- Configuration: it allows to set a password not allowing to overwrite a MSC already drawn and setup the URL of the MSC WebApp (the configuration information can be also provided through a configuration file, generated through the Dashboard of the Micro-Id-Gym Backend);
- Interceptor: it collects all the intercepted HTTP messages;
- Filter: it allows to add filters in terms of keywords of the HTTP messages that will be collected and reported in the MSC. The keywords can refer to `Host`, `Request Headers`, `Request Parameters`, `Response Headers` and `Response Body`; and
- Parser: it allows to add rules for mapping keywords or sequence of keywords in new terms in order to further improve the readability of the MSC. For instance, in a training context, it is possible to provide a MSC closer to the abstract view of the protocol under test (e.g., by mapping the actual URL of a Client with the label C).

Pre-configured filtering and parsing rules are currently available for SAML and OIDC/OAuth. In addition, the penstester can add custom rules.

To allow the MSC Drawer to create the MSC, the pentester has to navigate with the browser following the steps of the protocol and the corresponding MSC is dinamically generated in another browser at the URL set in the `Dashboard`. Directly in the MSC, the pentester can dive into the messages, and thanks to the interaction with the MSC STIX Visualizer he can check the available CTI information related to the parameters of the HTTP message.

Pentesting Tool. It provides a set of tools to perform automatic pentesting of IdM protocol. Our idea is on the one hand to integrate in our tool (most of) the existing open source tools for automatic pentesting of IdM protocol, on the other hand to complement them with the pentesting techniques proposed in this paper. In addition, for every detected vulnerability, our tool returns the HTTP messages that may cause the flaw and suggests mitigations so to allow users to understand how to (manually) fix the issue. After an in-depth analysis of the state-of-the-art, we elicited those prominent and recent tests both for OAuth/OIDC and SAML, not yet covered by other plugins and added them in the Pentesting Tool. We developed two kinds of tests, the so-called passive and active tests, namely:

- Passive tests: tests done analyzing statically the intercepted HTTP messages without any interaction/modification of the HTTP messages during execution of the IdM protocol.
- Active tests: tests that need an interaction during the execution of the IdM protocol. After the initial execution, the user actions are stored and automatically re-executed while intercepting/changing the content of the messages before sending them to the C and IdP.

By using both active and passive tests, Pentesting Tool performs the following test categories: *(i)* Compliance with a given standard in terms of format of the messages, and mandatory fields, *(ii)* General security checks, performed on any collected HTTP message, and *(iii)* Specific tests for C and IdP roles. We decided to detail each test about OAuth/OIDC in Section 4 and not provide details regards the test for SAML because the scenario that we analyzed in this paper is based on OAuth/OIDC. The list of the SAML tests is available at the complementary material page.[6]

Table 1. Collection of Security Tests targeting *Any* role.

Security Test	Prot.	P/A	Description	Mitigation
Use of HTTPS	-	P	Check if all HTTP messages communicates over a secure channel.	Change conf. of the web server and forward unsecured HTTP messages to HTTPS.
Clickjaking Prevention	-	P	Check if all the HTTP messages has the header X-FRAME-OPTIONS set as DENY or SAME-ORIGIN.	Set DENY or SAME-ORIGIN the header X-FRAME-OPTIONS.

The set of pentesting tests is far from being complete. We started from that, but we would like to include in our tool (most of) the existing open source tools for pentesting IdM protocol. Our goal is to cover as many vulnerabilities/attacks as possible.

[6] https://stfbk.github.io/complementary/ETAA2020

We provide an excerpt of all the tests for OAuth/OIDC protocol we are currently supporting, and in detail the Table 2 reports the tests for IdP, and Table 1 for both IdP and C. For each test we set: *(i)* a name to identify the test (e.g., Use of HTTPS), *(ii)* the IdM protocol implemented in the environment where the test will be executed ("O" stands for OIDC), *(iii)* the type of test Passive (P) or Active (A), *(iv)* the description of the security test, and *(v)* the description of the mitigation.

3.2 The Components of Micro-Id-Gym Backend

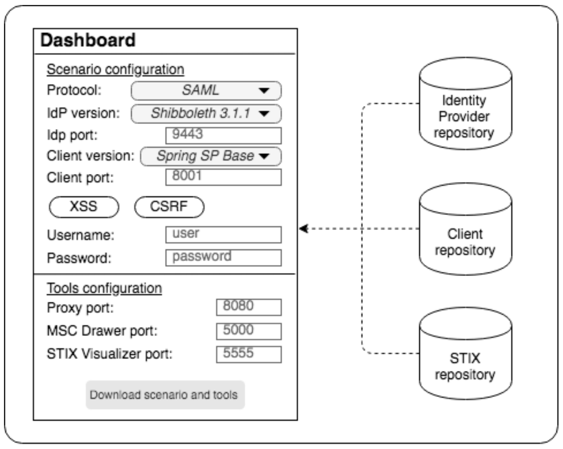
The goal of the Micro-Id-Gym Backend is by construction to provide a test environment generator tailored to IdM protocols and deploy the environment in the SUT. Given a set of available IdM protocol implementations collected while using the tool for third parties, the SUT automatically sets-up a working environment in a local network. The main reason is to allow system administrators to recreate locally in the laboratory their production environments, being able to pentest them in sandboxes. As depicted in Fig. 4, the Micro-Id-Gym Backend is composed by a set of IdPs and Cs instances both for OAuth/OIDC and SAML, a STIX notes repository and a Dashboard. The set of available instances is indeed a work in progress, and can be easily extended/updated over the time. By design, the architecture allows continuous integration of newer and different implementations.

The component editor (i.e. the person in charge to configure the SUT) through a Dashboard can select the IdM protocols (currently either SAML or OAu-

Table 2. Collection of Security Tests targeting *IdP* role.

Security Test	Prot.	P/A	Description	Mitigation
CSRF Prevention	O	P	Checks whether `state` parameter is used.	Introduce the `state` parameter in the flow.
Check Compliance with Standard	O	P	Checks whether all the parameters reported as REQUIRED in the Standard are in the HTTP messages of the considered flow.	Introduce the missing parameters in the flow.
Adopted PKCE	O	P	Checks whether the implementation used the parameter Proof Key for Code Exchange (PKCE).	Introduce the PKCE parameter in the flow.
Alteration `state` parameter	O	A	Changes in the Authorization Request the value of the `state` parameter.	Sanitize the value of `state` parameter.
Deletion state parameter	O	A	Deletes the value of the `state` parameter when sent to the AS with the Authorization Request.	Sanitize the value of `state` parameter.
Alteration `code_challenge` parameter	O	A	Changes the value of the `code_challenge` parameter when sent to the AS with the Authorization Request.	Sanitize the value of `code_challenge` parameter.
Deletion `code_challenge` parameter	O	A	Deletes the value of the `code_challenge` parameter when sent to the AS with the Authorization Request.	Sanitize the value of `code_challenge` parameter.
Alteration `code_challenge_method` parameter	O	A	Changes the value of the `code_challenge_method` parameter when sent to the AS with the Authorization Request.	Sanitize the value of `code_challenge_method` parameter.
Deletion `code_challenge_method` parameter	O	A	Deletes the value of the `code_challenge_method` parameter when sent to the AS with the Authorization Request.	Sanitize the value of `code_challenge_method` parameter.

Legenda: "P": Passive Test, "A": Active Test. "-": any protocol


Dashboard

Scenario configuration

Protocol: SAML

IdP version: Shibboleth 3.1.1

Idp port: 9443

Client version: Spring SP Base

Client port: 8001

XSS CSRF

Username: user

Password: password

Tools configuration

Proxy port: 8080

MSC Drawer port: 5000

STIX Visualizer port: 5555

Download scenario and tools

Fig. 4. Micro-Id-Gym Backend.

th/OIDC), an IdP instance and one or more C instance(s) he wants the SUT to deploy, among the ones available. At the moment of the selection, the Dashboard consults the STIX notes repository and shows which known security issues the selected entities might suffer (e.g., XSS and CSRF in Fig. 4). The user can also insert customized credentials to authenticate on the IdP and configure the ports where C and IdP will run. Once the selection has been completed the SUT will *(i)* generate and deploy the sandbox, *(ii)* create a local network within the agents, *(iii)* perform the federation of the entities and *(iv)* set the credentials for each IdP instance.

Finally, the Dashboard can be used to configure some components of the Micro-Id-Gym Frontend. In particular, customized ports can be selected for the Proxy, the MSC Drawer and the MSC STIX Visualizer.

3.3 Implementation

In this section, we provide the technical details about the implementation of the Micro-Id-Gym.

All the instances are Docker-based and the currently available implementations are depicted in Table 3. For each instance, we report the version, the protocol running on that implementation and the technology used. The proxy we decided to adopt is the Community Edition of Burp Suite (hereafter Burp). This choice has been done because it provides a set of useful and easy-to-use APIs for the pentesting tools development.

A web application (hereafter webapp) developed in NodeJs provides the Dashboard for the configuration of the instances, the Proxy, and the Pentesting Tool. Through a form, the user can customize the ports where the services will run, insert custom credentials to authenticate at the IdP, and give a name to the environment he is creating. The form submission starts the generation of the needed files. A folder with the chosen environment name is created and the selected implementation is copied into it. Docker and Burp configuration files are then customized through a find-replace process. Targeted placeholders are substituted with the information inserted by the user. Finally, the steps to run the environment together with the location of the environment folder are displayed in the dashboard. The same information is also written in a Readme file added to the environment folder.

Following the provided instructions, the user has to run the frontend. Once opened the terminal and reached the folder where the customized environment is located, he finds three directories: `proxy`, `tools` and `sso`. The `proxy` folder contains Burp, part of the Pentesting Tool, and the JSON files used for the automatic configuration. The user can run the Proxy and load tools and configuration with `java -jar` command. The folder `tools` contains MSC Drawer and MSC STIX Visualizer. Both are developed in NodeJs and deployed as Docker containers. A `docker-compose` command automatically deploys and run them. The last folder, `sso`, contains the configured instances. As well as the tools, the instances are deployed and run through a `docker-compose` command.

For the interested reader, the Micro-Id-Gym tool is online[7] and a demo video is available at the complementary material page.

Table 3. Collection of C and IdP instances.

Version	**Prot.**	**Technology**	**Provider**
Base	S	Spring SP	C
Supported c14n algorithm	S	Spring SP	C
DTD enabled	S	Spring SP	C
DTD disabled and c14nWithComments algorithm	S	Spring SP	C
RelayState validation enabled	S	Spring SP	C
RelayState validation disabled	S	Spring SP	C
Sample Webapp	O	KeyCloak	C
Simple Webapp	O	MitreID	C
3.3.x	S	Shibboleth IdP	IdP
3.2.x	S	Shibboleth IdP	IdP
3.3.3 with RelayState sanitization enabled	S	Shibboleth IdP	IdP
1.3.3 without redirect_uri validation	O	MitreID	IdP
1.3.3 with redirect_uri validation	O	MitreID	IdP
10.0.1 without redirect_uri validation	O	Keycloak	IdP
10.0.1 with redirect_uri validation	O	Keycloak	IdP

Legenda: “S”: SAML, “O”: OAuth/OIDC

[7] https://github.com/stfbk/micro-id-gym/

4 **Micro-Id-Gym** at work in the wild: pentesting of a PSD2 deployment

In the context of an European project we had to assess a PSD2 service provided by an important Italian identity provider. PSD2 is the second Payment Services Directive, designed by the countries of the European Union. Strong Customer Authentication (SCA) is one of the fundamental building blocks for building secure PSD2 deployments composed by Multi Factor Authentication and dynamic linking. The best solution to implement this scenario (such as proposed by many like the Berlin group) can be done by adopting OAuth which allows delegation to third parties to access resources. It could revolutionize the payments industry, affecting everything from the way we pay online, to what information we see when making a payment. Security of electronic payments is fundamental for ensuring the protection of users and all payment services offered electronically should be carried out in a secure manner, adopting technologies able to guarantee secure user authentication.

The involved scenario that we analyzed uses OAuth and OIDC as an IdM protocol. The OAuth protocol is an authorization standard allowing a user (resource owner, RO)—which interact with a user agent (UA), typically a web browser—is to delegate to a C the access to his resources stored on another web server controlled by an Authorization Server (AS). OIDC is an identity layer on top of the OAuth protocol that is used for authentication purposes. The aim of the user which interacts with UA is to get access to a service provided by C, leveraging AS as IdP. Fig. 5 shows a MSC providing a simplified view of the OAuth Authorization Code flow. A brief description of the protocol is as follows:

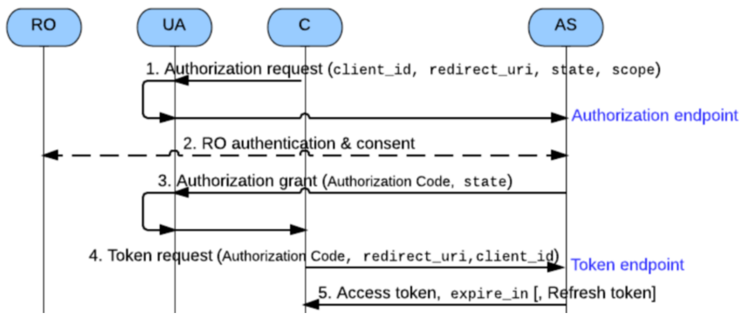

RO
UA
C
AS
1. Authorization request (client_id, redirect_uri, state, scope)
Authorization endpoint
2. RO authentication & consent
3. Authorization grant (Authorization Code, state)
4. Token request (Authorization Code, redirect_uri,client_id)
Token endpoint
5. Access token, expire_in [, Refresh token]

Fig. 5. OAuth Authorization Code Flow (simplified view).

1 C initiates the flow by directing RO's UA (typically a web browser) to the authorization endpoint. C includes its identifier (`client_id`), requested scope (`scope`), local state (`state`), and a redirection URI (`redirect_uri`) to which the AS will send UA back once access is granted (or denied).

2 The AS authenticates the RO (via the UA) and establishes whether the RO grants or denies the C's access request.
3 Assuming RO grants access, AS redirects UA back to C using the redirection URI provided earlier (in the request or during C registration). The redirection URI includes an authorization code and any local state provided by C earlier.
4 C requests an access token from AS's token endpoint by including the authorization code received in the previous step. When making the request, C authenticates with AS. C includes the redirection URI used to obtain the authorization code for verification.
5 AS authenticates C, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect C in step 3. If valid, AS responds back with an access token and, optionally, a refresh token.

The description of OAuth abstracts away several details that are crucial for security. A trust relationship between C and AS must be established before running the protocol by distributing appropriate meta-data between the two entities. The user must possess credentials to access AS. The configurations of auxiliary modules must be properly performed, and the format of the messages must be properly set and checked.

To perform the security assessment, we leveraged the Pentesting Tool component of Micro-Id-Gym, and we were able to identify some vulnerabilities and a misconfiguration of the OIDC protocol. In particular, we followed the following two steps:

– Being the SUT an OIDC implementation, we set the MSC Drawer on the OIDC mode, in such a way to collect all the relevant traffic concerning OIDC. We have been thus able to spot very quickly the differences between the expected OIDC flow and the one shown by the MSC Drawer in terms of the intercepted HTTP messages. Thanks to MSC Drawer we were able to identify two HTTP messages exchanged between entities not encrypted using Transport Layer Security (TLS). The usage of HTTPS protects against man-in-the-middle attacks, eavesdropping and tampering. Thus this provides a reasonable assurance that one is communicating with the intended website without interference from attackers.
– Then, a finer-grained security analysis has been conducted by using our plugin included in the Pentesting Tool. The tests reported in Section 3 have been automatically executed and a number of relevant issues reported.
  About the compliance, some required parameters according to the OIDC standard were not present in the protocol. On the contrary, some parameters which are not expected in the OIDC standard have been included in the protocol. In detail the tool detected that `redirect_uri` and `state` parameters were not set properly and both were not verified in different steps of the process by the IdP.
  Regards the vulnerability in the `redirect_uri` parameter, AS, authorization endpoint, and client redirection endpoint can be improperly configured and

operate as open redirects. An open redirect is an endpoint using a parameter to automatically redirect a user-agent to the location specified by the parameter value without any validation. Open redirects can be used in phishing attacks, or by an attacker to get end-users to visit malicious sites by using the URI authority component of a familiar and trusted destination. In addition, if AS allows C to register only part of the redirection URI, an attacker can use an open redirect operated by C to construct a redirection URI that will pass the authorization server validation but will send the authorization code or access token to an endpoint under the control of the attacker [12].

Regards the `state` parameter, it is a recommended parameter in OAuth [2]. It is an opaque value used by C to maintain state between the request and callback. The AS includes this value when redirecting the user-agent back to the C. The `state` parameter is set by C in the Authorization Request, in step 1 in Fig. 5, and it is checked by C when C receives it in step 3. If the integrity of the `state` parameter field is not adequately protected, it may allow hackers to mount cross-site request forgery attacks (CSRF) [9]. To exploit these vulnerabilities an attacker causing the target browser to send the target site a request containing the attacker's own authorization code or access token. As a result, the target site might associate the attacker's protected resources with the target user's current session; possible undesirable effects could include saving user credit card details or other sensitive user data to an attacker-controlled location.

These results pointed out the need of tools, like Micro-Id-Gym, capable to support developers in order to properly implement, configure and test the OIDC implementations. Another interesting consideration emerges from our assessment. Some parameters were included in the protocol, meaning that we could detect them while checking the HTTP traffic. Yet, some of them were not used in end. For instance, the values of some parameters were not checked at all by the IdP. This highlights the need of active tests to check whether the parameters are indeed used properly.

All the discovered issues were notified to the company and fixed by their developers. In detail, about the HTTP messages exchanged not over TLS, the developers reviewed all the URL inside the application and modified the web server so that all messages will pass in HTTPS. Regards the missing sanitization of the `redirect_uri` parameter, the developers implemented a method inside the application to sanitize the parameter and do not allow any manipulation from an attacker. For the not adequately protection of the `state` parameter, the developers added some code to adequate protect the parameter. Lastly, about the redundant parameters not included in the protocol, the developers removed all of them from the flow.

5 Micro-Id-Gym at work in the lab: evaluation of the effectiveness

While in Section 4, we reported a use case where we used Micro-Id-Gym in the wild in this section, we evaluate it in the laboratory. We create a couple of scenarios of deployments for IdM protocol based on OIDC where some implementations have implemented the protocol correctly and others where there are some vulnerabilities to identify. In order to evaluate if and how the tool it is used effectively we have organized a structured user experience in a couple of workshops involving 8 bachelor and 30 master students from the Department of Information Engineering and Computer Science of the University of Trento, playing the role of inexperienced security testers. Bachelor students have quite a broad background on information security, because they attended (among others) the course *Introduction to Computer and Network Security*, that covered the OIDC protocol. Master students attended other courses about security, including the course *Security Testing* that covered, in particular, attacks to OIDC implementations.

In this lab we are focusing on the MSC Drawer component of Micro-Id-Gym. The goal of the study is to evaluate the effectiveness of understanding an OIDC execution trace, when visualized either by MSC Drawer or by OWASP Zap, [8] one of the most popular free security tools, by conducting an experimental user study. Below, we detail the settings of the experimentation (designed following the template and guidelines by Wohlin et al. [5]) and summarize the main results. The experimental setting starts with the definition of the research question: Do the usage of MSC *increase* the effectiveness to find the *correct* OIDC implementation by a security tester? The research question aims to evaluate whether MSC helps security tester to increase security effectiveness of OIDC implementations. By answering this research question (see Section 5.4), we will be able to assess the added value in adopting MSC Drawer instead of available state-of-the-art proxy tools.

5.1 Context

The *context* of this study consists of the *participants* involved in the experiment and the software *systems* whose deployment contains vulnerabilities.

Participants. As far as the participants are concerned, we are aware that the expertise of students may be different from that of professionals. However, finding professionals available to conduct a demanding experiment as the one we designed is not easy. We mitigated this limitation by considering students with different levels of education and by making sure that participants had enough knowledge on OIDC protocols and its related vulnerabilities. All in all, the use of undergraduate students as a proxy of junior developers to draw conclusions is a common practice in empirical software engineering that is largely accepted and validated [7,13,11].

[8] https://www.zaproxy.org/

Systems. The systems used to conduct the experiment are two different deployments of the OIDC protocol:

- **S**$_1$ a defective OIDC implementation vulnerable to a missing sanitization of the `redirect_uri` parameter; and
- **S**$_2$ a defective OIDC implementation vulnerable to a not adequate protection of the `state` parameter.

The selected systems are comparable in terms of complexity of the operations required to detect the problem. It is important to note that these systems are representative of realistic OIDC implementations and both vulnerabilities and their related attacks are described in Section 4. To fit the time constraint of our experiment, only one vulnerability is present in each system.

5.2 Variables Selection

To measure the support of vulnerability detection and to conduct a corrective maintenance on OIDC implementations, we identified as the main factor of the experiment—that acts as an independent variable—the presence of the MSC during the execution of the task. In our experiment, the *base* treatment case TR_{zap} consists of using OWASP ZAP; and TR_{mig} consists of using MSC Drawer, that includes not only the list of the HTTP intercepted messages, but also generates a MSC of the intercepted traffic. Moreover, we instrumented the experimental settings to measure **Correctness** of each corrective tasks performed by participants.

5.3 Experiment Design and Procedure

We adopt a counter-balanced experimental design intended to fit two lab sessions. Participants are randomly assigned to four groups (despite they work alone), each one working in two labs on different systems with different treatments. The design allows for considering different combinations of *Systems* and *Treatments* in different order across *Labs* (see Table 4).

Before our experiment, participants were properly trained with lectures and exercises on OIDC protocol, on MSC Drawer and on OWASP ZAP, to provide/recall the required background. The purpose of training is to make participants confident about the kind of tasks they are going to perform and the environment they will have available. The experiment was carried out according to the following procedure. Participants had to *(i)* complete a pre-experiment profiling

Table 4. Experimental design.

	Group A	**Group B**	**Group C**	**Group D**
Lab 1	S_1 with TR_{mig}	S_2 with TR_{zap}	S_2 with TR_{mig}	S_1 with TR_{zap}
Lab 2	S_2 with TR_{zap}	S_1 with TR_{mig}	S_1 with TR_{zap}	S_2 with TR_{mig}

survey questionnaire, *(ii)* perform the detection task for the first lab, *(iii)* perform the detection task for the second lab, and *(iv)* complete a post-experiment survey questionnaire.

The pre-experiment profiling survey collects background knowledge about the participants, such as their previous experience with Proxy tool and their knowledge of OIDC protocol. Post-experiment survey questionnaire (reported in Appendix A) deals with the clarity of the tasks, cognitive effects of the treatments on the behavior of the participants and perceived usefulness of MSC Drawer.

5.4 Summary of Findings

We collected all the results and noticed that: (i) the distributions of correct/wrong answers when using MSC Drawer is respectively 262 and 80, (ii) the distributions of correct/wrong answers when using OWASP ZAP is respectively 222 and 120, (iii) 8 participants over 38 were able to detect correctly and completely the vulnerabilities when they were using MSC Drawer, and (iv) only 3 participants were able to detect correctly and completely the vulnerabilities when using OWASP ZAP.

The feedback questionnaire was positive and from the post-experiment survey we can learn that 87.5% of the students considers TR$_{mig}$ more useful and 84.4% assessed that TR$_{zap}$ is more complex to understand. In addition, all the students positively recommend our tool. Here we report some comments: "Very clear in the presentation of information.", "Simple to visualize the HTTP messages with directions.", "It is easier and faster to read all the information.".

Finally, looking at their subjective feedback, it seems that participants agree with our claim that MSC Drawer is highly beneficial in detecting vulnerabilities in OIDC implementations and it will help to increase the security awareness.

6 Conclusions and Future Work

We have described Micro-Id-Gym a flexible tool for pentesting IdM protocols easy to configure and in which users can develop hands-on experiences on how IdM protocols work, performing attacks with high impacts and better understand the underlying security issues. For ease of configuration and deployment, Micro-Id-Gym uses container-based micro-services and state-of-the-art penetration testing tools. We have improved Micro-Id-Gym by supporting new IdM protocols and a catalog of realistic scenarios in which different vulnerabilities and attacks can be re-created, analyzed, and mitigated. Secondary, we have analyzed a real use-case scenario involving a PSD2 service provided by an important Italian identity provider. Finally, we have validated the user experience and security awareness provided by our framework by using the results of a user-study experimentation involving students from university.

As future work, we plan to extend Micro-Id-Gym by *(i)* integrating new pentesting tools, *(ii)* supporting other multi-party web applications, and *(iii)* supporting STIX also for OAuth/OIDC.

References

1. PSD2 (accessed june 23, 2020). `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32015L2366`
2. Security Considerations OAuth (accessed june 23, 2020). `https://tools.ietf.org/id/draft-bradley-oauth-jwt-encoded-state-08.html#rfc.section.6`
3. 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: Proceedings of the 6th ACM workshop on Formal methods in security engineering. pp. 1–10 (2008)
4. Bisegna, A., Carbone, R., Martini, I., Odorizzi, V., Pellizzari, G., Ranise, S.: Micro-Id-Gym: Identity Management Workouts with Container-Based Microservices. In: International Journal of Information Security and Cybercrime 8(1). pp. 45–50 (2019)
5. C. Wohlin, P. Runeson, M.H.M.O.B.R.A.W.: Experimentation in software engineering. Softw. Test., Verif. Reliab. (2001). https://doi.org/10.1002/stvr.230
6. Hardt, D.: The OAuth 2.0 Authorization Framework (RFC6749). Internet Engineering Task Force (IETF) (2012)
7. Höst, M., Regnell, B., Wohlin, C.: Using students as subjects—a comparative study of students and professionals in lead-time impact assessment. Empirical Software Engineering **5**(3), 201–214 (2000)
8. Hughes, J., Maler, E.: Security assertion markup language (saml) v2.0 technical overview. OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08 pp. 29–38 (2005)
9. Li, W., Mitchell, C.J.: Security issues in oauth 2.0 sso implementations. In: International Conference on Information Security. pp. 529–541. Springer (2014)
10. Sakimura, N., Bradley, J., Jones, M., De Medeiros, B., Mortimore, C.: OpenID Connect Core 1.0 incorporating errata set 1. The OpenID Foundation, specification **335** (2014), `https://openid.net/specs/openid-connect-core-1_0.html`
11. Salman, I., Misirli, A.T., Juristo, N.: Are students representatives of professionals in software engineering experiments? In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. vol. 1, pp. 666–676. IEEE (2015)
12. Svahnberg, M., Aurum, A., Wohlin, C.: Redirect uri attack. In: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. pp. 288–290 (2008)
13. Svahnberg, M., Aurum, A., Wohlin, C.: Using students as subjects-an empirical evaluation. In: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. pp. 288–290 (2008)

A Post-questionnaire

The following table shows the content of the post-experiment survey questionnaire mentioned in Section 5. It deals with object clarity of the tasks, cognitive effects of the treatments on the behavior of the subjects and perceived usefulness of MSC Drawer. The first set of questions (Q1-Q6) needs to be answered twice (one answer for each performed lab) while the remaining set only needs to be answered once as it refers to the overall session.

Table 5. Post-experiment survey questionnaire.

ID	Applies to	Question
Q1	Each lab	I had enough time to perform the tasks. (1-5).
Q2	Each lab	I experienced no difficulty in detecting the vulnerability. (1-5).
Q3	Each lab	Which operations (e.g., mouse over steps, open tab, search, ...) did you perform to understand whether the protocol was vulnerable to the mentioned vulnerability?
Q4	Each lab	Did you consult internet to find help to answer the questionnaire? If yes, which online queries did you search(e.g., keywords used)? Which content was helpful?
Q5	Overall	Which tool did you find more useful to answer the questions? (Report of Lab 1-2).
Q6	Overall	Which tool did you find more intuitive? For which tool was more difficult to find the proper information about the protocol in order to answer the questions? (Report of Lab 1-2).
Q7	Overall	Which tool would you use for your work? Motivate your answer (to the previous question). (open question).
Q8	Overall	Do you know any tool that performs similar tasks? (open question).
Q9	Overall	Do you have any suggestion related to the tool usage? (open question).
Q10	Overall	What do you think is the main advantage using MSCDrawer? Would you add more information to the MSCDrawer? (open question).