Federated Lab (FedLab): An Open-source Distributed Platform for Internet of Things (IoT) Research and Experimentation

Max Meijer*, Giacomo Tommaso Petrucci*, Matthijs Schotsman*, Luca Morgese‡
Thijs van Ede†, Andrea Continella†, Ganduulga Gankhuyag*, Luca Allodi*, Savio Sciancalepore*
*Eindhoven University of Technology, Eindhoven, Netherlands
†University of Twente, Enschede, Netherlands
‡TNO, Netherlands

***Abstract*— This paper introduces the Federated Lab (in short, FedLab), a virtual platform enabling shared research and experimentation on Internet of Things (IoT) devices, protocols, and functionalities in a network of geographically-distributed peers. In a nutshell, the FedLab allows multiple peers to share heterogeneous IoT hardware (devices) and software functionalities (capabilities) with a set of authenticated partners, easing the design of geographically-distributed experiments and providing a platform that enables joint research. The FedLab is conceived with ease of deployment and use in mind, focuses on compatibility-by-design, and it is easily deployable on multiple host operating systems as a plug-and-play tool. Besides motivating the design of the FedLab and its requirements, the paper also provides two use cases of the FedLab for the *IoT Security* research domain, demonstrating the potential of the FedLab to enable actual joint research. Finally, we release the source code of the components of the FedLab as open-source, to foster its independent adoption, deployment, and extension by the research community.**

***Index Terms*—IoT; Open Platform; Testbed; Experimentation.**

I. Introduction

The last years witnessed the pervasive diffusion of the Internet of Things (IoT) paradigm. Today, smart Internet-connected devices are increasingly deployed in several application scenarios, e.g., Smart Home, Buildings, Industrial, and Transportation, and a significant number of Cloud platforms, web-based applications, and functionalities are largely available on the market [1]. At the same time, companies and research centres worldwide are very active in the IoT domain, contributing an ever-increasing amount of innovative solutions for domain-specific challenges and issues [2].

In this context, shared research and experimentation take an increasingly important role in the design and evaluation of the proposed solutions [3]. Wherever research teams in universities and small companies provide expertise limited to specific application areas, collaboration in the form of consortia of possibly geographically-distributed teams is needed to develop reliable and accurately-tested IoT-based solutions [4], [5]. A stemming example of a research domain where shared research and experimentation are often required is *IoT security*. The security personnel working at companies and research centres often have expertise with a limited set of security tools and functionalities, e.g., defensive or offensive-only tools, resulting in security products and solutions to be tested against a small range of conditions and attack sets. In this context, a shared platform involving a set of trusted partners, each with distinguishing skills, might enable controlled shared experimentation and testing, resulting in more tight collaboration and, ultimately, more secure IoT products and functionalities [6].

However, shared research in the IoT domain is often challenging. Collaboration is often prevented by several factors, such as the geographical distribution of the teams, the confidentiality of the source code of the conceived tools and functionalities, the heterogeneity of the protocols and communication technologies available for the IoT, and the lack of a trusted collaboration platform where such tools can be made available securely to the consortium and robustly evaluated. As a result, solutions conceived in the IoT area are hard to test comprehensively, with a resulting negative impact on their overall quality, usability, security and, ultimately, impact on society.

Contribution. In this paper, we introduce the Federated Lab (*FedLab*), the first virtual platform enabling shared research and experimentation on IoT devices and functionalities across multiple geographically-distributed sites. The FedLab allows the participating partners to share with a set of authenticated peers locally-deployed IoT devices, supporting by-design multiple IoT protocols and architectures. Moreover, peers in FedLab can share locally-conceived IoT functionalities, namely, *capabilities*, and make them usable to other partners while not disclosing potentially-sensitive implementation details. In addition, the FedLab has been designed with compatibility and ease of use in mind, being deployable on multiple operating systems and running out of the box with minimal configuration. We discuss the motivations and design of the FedLab, the

This is a personal copy of the authors. Not for redistribution. The final version of the paper and the DOI will be available through the IEEExplore Digital Library soon.

rationale of the implementation, and the users' experience with the tool. We also present two use cases of the FedLab for the IoT security community, showing the role of the FedLab for allowing usage of capabilities and enabling distributed experimentation, respectively. Finally, we release the source code of FedLab as open-source, allowing interested readers to deploy further our solution and extend it with additional functionalities [7].

Roadmap. The paper is organized as follows. Sec. II motivates the design of the FedLab, Sec. III provides implementation details and the users' experience, Sec. IV shows two use cases of the FedLab, Sec. V reviews related work, and finally, Sec. VI concludes the paper and outlines future work.

II. FedLab Requirements and Design

In this section, we first introduce the requirements of the FedLab (Sec. II-A). Then, we present the architecture details (Sec. II-B) and the major supported operations (Sec. II-C).

A. Federated Lab Requirements

The FedLab originates from the need for a trusted platform where to carry out shared research and experimentation within consortia of multiple geographically-distributed research groups. The design of an effective and efficient platform enabling such an objective is challenging, as it requires the fulfilment of several binding requirements. First, the FedLab should be a distributed platform (R1), supporting the connection of multiple geographically-distributed laboratories in a shared virtual network. At the same time, due to the potential side effects of the experiments on connected networks, the FedLab should be isolated from the private networks of the participating partners (R2), in a way not to affect business continuity and availability. The FedLab should also provide users' authentication, guaranteeing to share information only between the intended parties (R3), as well as accountability of the operations performed during the interaction with the platform. Moreover, the FedLab should support basic logging and auditing features (R4), so that performed operations could be always traced back to a given user. Looking at the functionalities to be provided, the FedLab should be able to support the sharing of both IoT devices and IoT-related functionalities (R5). For instance, users can decide to connect to the FedLab their IoT devices, to test the robustness of such devices against customized attacks, or evaluate if a functionality made available by one of the partners can work together with a shared device. To this aim, the FedLab should support the discovery of resources (devices and functionalities) currently available in the platform (R6), reporting also an indication of their current availability and usage. In addition, considering the large variety of IoT communication protocols and standards, the FedLab should be able to support by-design the connection of heterogeneous devices (R7), independently from the specific MAC-layer protocol and hardware features. Finally, to foster its adoption by all the partners, the FedLab should have appealing usability features, i.e., it should be easy to install (R8), with only minimal configuration needed by the user. At the same time, it should be able to run on multiple software platforms (R9), independently from the local hardware and operating system.

Table I
Relationship between Design Requirements, Components, and Functionalities of the FedLab.

Comp.	Functionality	R1	R2	R3	R4	R5	R6	R7	R8	R9
FedLab Server	VPN Server Resource Directory	✓		✓	✓		✓			
Bundle Box	VPN Client	✓		✓					✓	
	Devices and Capabilities Management					✓	✓	✓	✓	
	ACL		✓							✓
	Vagrant									

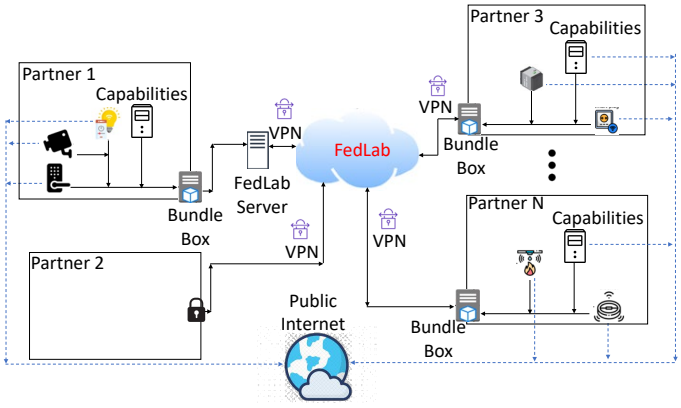

Figure 1. Network Architecture of the FedLab.

As we will show in Sec. V, the requirements described above are not supported by any available solution at the time of this writing, motivating the design of our solution. Tab. I anticipates the solutions we conceived to address the requirements discussed above in the design and implementation of the FedLab. Details follow in the next sections.

B. Architecture of the FedLab

Figure 1 shows the network architecture of the FedLab.

FedLab Server. The *FedLab Server* hosts the core processes and functionalities of the FedLab. First, it includes a *Virtual Private Network (VPN) Server*, available over the public Internet, in charge of establishing and managing connections to the FedLab. The VPN server allows connections only from legitimate peers, in possession of valid X.509 certificates, released to the users at joining time. Thus, in case of misbehaviour or voluntary leave by a partner, the FedLab Server stops the connection of such partners to the shared platform. Being in charge of managing connections among peers, the FedLab Server takes part in all platform activities. Such explicit involvement eases the deployment of basic logging and auditing functionalities, to be analyzed when debugging and tracing are needed. Furthermore, the FedLab server hosts the *FedLab*

Directory, i.e., the real-time list of all the resources available within the FedLab, reporting also the indication of the partner sharing such resources and the related status, either online (available for use) or offline (not available).

Bundle Box. At the client-side, the *Bundle Box* is the software tool allowing partners' connection to the FedLab, as well as smooth management of shared resources. In a nutshell, the Bundle Box is a piece of software, allowing the partner to connect to the FedLab and share resources. The Bundle Box includes a VPN client instance, taking care of the virtual connection of the partner's network to the FedLab. To this aim, at the partner's request to join the platform, the network administrator of the FedLab generates a dedicated X.509 certificate, to be used to connect securely. Moreover, the Bundle Box manages the sharing of local resources within the FedLab. It takes care of communicating to the FedLab server the set of resources shared by the partner, i.e., IoT devices and capabilities, together with the related status (online or offline). The FedLab partner can share specific IoT devices (e.g., smart doorbells, smart cameras, and smart lights, as sketched for the Partner 1 in Fig. 1), to be connected either directly or through multiple hops to the physical machine where the Bundle Box runs. If IoT devices use a different communication technology than IP (e.g., Bluetooth or Zigbee), the peer can make such IoT devices part of the FedLab by sharing the gateway as a device, and associating the various MAC addresses of the non-IP devices with the IP of the gateway. In this way, the traffic flowing to and from the non-IP devices will be integrated transparently into the FedLab.

The Bundle Box also connects to the FedLab one or more servers hosting *capabilities*, i.e., *homemade* functionalities developed and made available by a peer. Capabilities may include IoT applications or other services of any type (including also traditional IT services), and tools developed and/or deployed by the peers in the consortium. Capabilities are shared according to an Application Program Interface (API)-based logic: the responsible partner shares a set of procedures, namely, APIs, allowing all peers to create their own applications using the shared remote functionalities, without having access to the native code of such basic blocks. IoT devices and capabilities are explicitly shared one by one with the FedLab, by specifying their unique MAC address to the Bundle Box.

Moreover, the Bundle Box also includes an Access Control List (ACL), following a white-listing approach. Only hosts whose MAC address is specified in the ACL are shared with the FedLab, while packets from and to other IP-enabled devices are rejected. Using such an approach, the Bundle Box ensures that only traffic from and to shared resources is managed. On the one hand, IoT devices requiring a connection to the public Internet can use such online services without any limitations. On the other hand, the Bundle Box leaves to the FedLab partner the responsibility of isolating the shared resources from other parts of the local network. As IoT devices and capabilities may be available on dynamic IP addresses or can be disconnected, the Bundle Box also takes care of maintaining an updated association between the specific resource and the current IP address. To this aim, the Bundle Box interacts with the Address Resolution Protocol (ARP) tables of the host where it is deployed, to get the IP address associated with the MAC address of shared resources. The Bundle Box communicates to the FedLab server the list of available resources and their status, to maintain the FedLab Directory. When resources are not available, they are not shown in the ARP tables, and they are removed from the FedLab Directory after a given period. Similarly, when a partner would like to stop sharing resources, the corresponding entry can be deleted from the local ACL. Such a change is immediately communicated to the FedLab server, and the resource is deleted from the FedLab Directory.

Guest Access. When connection only to the FedLab is required, e.g., to use resources shared by other partners while not sharing any local resources, a partner can use the *FedLabGuest Access* (as shown for Partner 2 in Fig. 1). Compared to the Bundle Box, the guest access only provides the connection to the FedLab, as well as the tools to discover resources shared by other partners. Note that the partner connecting to the FedLab as a guest still receives its unique X.509 certificate, and thus, it is always authenticated.

C. FedLab Processes

The management of the FedLab requires several structured processes, managed jointly by the FedLab administrator, the network administrator of the local partner joining the FedLab, the FedLab server and the Bundle Box. We briefly describe the main processes below.

Joining the FedLab. When a partner would like to join the FedLab, it contacts the FedLab administrator. The FedLab administrator evaluates the request and, assuming the request is accepted, generates a unique X.509 certificate, to be used to authenticate the joining partner. Then, the FedLab administrator delivers to the system administrator of the local partner network the X.509 certificate and an instance of the Bundle Box, so that it can deploy the Bundle Box locally.

Leaving the FedLab. When a partner would like to leave the FedLab, it notifies the FedLab administrator. As experiments could be running, with the partner's consent, a grace period may be applicable. Meanwhile, other partners are also informed about the change. At the end of the grace period, the FedLab administrator revokes the certificate of the leaving partner, thus preventing it (and all its shared resources) from accessing the FedLab.

Adding Resources to the FedLab. Assume a partner would like to share a new resource with the FedLab. If the resource is an IoT device, the local administrator retrieves its MAC address, and it uses the device type (e.g., *smart camera*), MAC address, and description as parameters in a dedicated script within the Bundle Box. For capabilities, the local administrator should get also the MAC address of the host exposing the capability and the used port. The Bundle Box automatically adds the information listed above to the local list of shared resources, and it checks the resource status. To this aim, the Bundle Box checks if the provided MAC address is listed in the

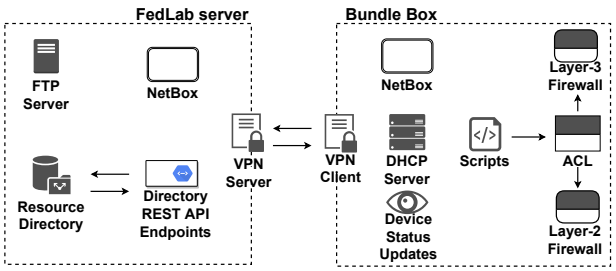

Figure 2. Implementation architecture of the FedLab server and bundle box.

local ARP table. If yes, it means that the resource is *available*; otherwise, the resource is *not available*. Then, the Bundle Box communicates to the FedLab server the information previously provided by the local administrator about the new resource, together with the related status. Finally, the FedLab server updates the FedLab Directory with the new information.

Removing Resources from the FedLab. When a partner would like to remove a shared resource from the FedLab, it simply disconnects the resource. The local Bundle Box detects the resource disconnection and communicates the event to the FedLab server, which marks the resource as *not available*. If the status of the resource does not change for a pre-defined time (e.g., two weeks), the local Bundle Box autonomously removes the resource from the list of shared devices.

Accessing Resources within the FedLab. When a partner would like to access resources available in the FedLab, it may first retrieve the updated FedLab Directory (see Tab. II). Using the directory, the user can identify the resource to be accessed, retrieve the IP address of the resource and, if necessary, the port. Then, it can connect to the resource via browser or terminal, using the IP address and port just retrieved, and use the resource according to the related usage terms. All the interactions are logged by the *FedLab server*.

III. Implementation Details and User Experience

This section describes the implementation of the FedLab (Sec. III-A) and provides an overview of the user experience when connecting to FedLab (Sec. III-B).

A. Implementation Details

We implemented a prototype of the FedLab by integrating several open-source technologies. As for the VPN, we used OpenVPN for both the client (integrated into the Bundle Box) and the server (part of the FedLab server) [8], and we configured the VPN server to accept connections only from valid clients, i.e., peers providing valid X.509 certificates, generated through the EasyRSA tool [9]. Figure 1 shows the architecture and information flow of our implementation.

FedLab Server. The FedLab server is deployed on the host machine as a set of four (4) Docker containers, managed by docker-compose. We recall that Docker is a software platform that simplifies the process of creating, deploying, and running software applications, by allowing the developer to use *containers*, i.e., packages including applications and all their dependencies [10]. Docker-compose, instead, is used to configure and deploy multiple containers in a centralized way [11]. Each of the containers manages specific functionalities. The first runs the OpenVPN server, managing secure network connections to the FedLab. The second container runs an instance of *NetBox*, i.e., an Infrastructure Resource Modeling (IRM) application designed to empower network automation, used to easily keep track of the IP addresses allocated to the different parties participating in the FedLab [12]. The third container hosts an FTP server, used for storing user manuals and instructions for using capabilities. Finally, the fourth container in the FedLab server manages the FedLab Directory. The FedLab Directory is a Java web application, implemented using Spring, i.e., a framework easing the implementation and deployment of Java web applications [13]. The FedLab Directory exposes seven REST API endpoints, summarized in Tab. II, which can be queried via HTTP GET or POST requests. Overall, such endpoints automatize the interaction between the FedLab Directory and locally deployed Bundle Box instances. Moreover, the use of REST API also enables the participating parties to write their own applications, to fulfil specific automation needs.

Table II
List of REST API endpoints of the FedLab Directory, corresponding request type and description.

Endpoint	Request Type	Summary
/capabilities	GET	Request all capabilities
/capabilities	POST	Add capability to FedLab
/devices	GET	Request all devices
/devices	POST	Add device to FedLab
/HTMLdirectory	GET	Request Directory (HTML version)
/directory	GET	Request Directory (JSON version)
/statusUpdate	POST	Update device status
/removeDeviceRequest	POST	Remove a device from the FedLab
/removeCapabilityRequest	POST	Remove a capability from the FedLab

Bundle Box. The Bundle Box, installed by the partners locally to participate in the FedLab, consists of a Virtual Machine (VM) in the form of a Vagrant box. Vagrant is a cross-platform software used to create and run VMs using a hypervisor and the parameters specified in a dedicated configuration file, namely, the *Vagrantfile*. The Vagrantfile automatically manages the OS image used to build the VM, all the steps needed to configure the VM and install the scripts, and software dependencies. The user can run a textual command in the local terminal to provision and boot the specified VM, and then just use it as a regular VM. Thus, the usage of Vagrant allows achieving software independence, reproducibility of the deployment, and software flexibility, as any change would only require the modification of a line in the configuration file.

The Bundle Box includes several components. First, it includes

an OpenVPN Client instance, allowing to connect to the FedLab via the OpenVPN server. Moreover, it includes a firewall, consisting, in turn, of a layer-2 firewall and a layer-3 firewall. Overall, an ACL is used to manage which devices are allowed to connect to and receive traffic originating from the FedLab. The layer-2 firewall is used to block unauthorized ARP messages (at the link layer), leveraging local *arptables* and using the *default-deny* configuration. At the same time, the layer-3 firewall is used to block unauthorized IP traffic (at the network layer), leveraging local *iptables* and the *default-deny* configuration. Thus, as soon as a new resource is added to the FedLab by the partner, the Bundle Box sets corresponding `ACCEPT` rules for this new resource in the ACL, allowing traffic to flow to and from the resource.

The Bundle Box also integrates a Dynamic Host Configuration Protocol (DHCP) server, in charge of assigning IP addresses to IoT devices. To this aim, the Bundle Box integrates a Netbox instance, based on the code released on DockerHub [14], automatizing network configuration. Moreover, the Bundle Box includes automated processes for device status updates and for removing outdated entries from the ACL. It also includes scripts for sending and retrieving user manuals from the FedLab server, and an application for remotely starting and stopping the capture of packets on the segmented network created by the Bundle Box (see Sec. IV-B).

Finally, the Bundle Box includes a set of scripts, used for the automatic management of partners' operations in the FedLab. Ready-to-use scripts are available for adding a resource to the FedLab, removing a resource, exposing a resource, and stopping exposing a resource to the FedLab, to name a few.

Guest Access. We also configured an automated software tool for connecting to the FedLab only, namely, the *Guest Access* package. It includes an OpenVPN client instance and the unique X.509 certificate assigned to the specific partner.

We released the source code of the FedLab server and bundle box as open-source, to allow interested readers to deploy the FedLab and further customise it for shared research and experimentation [7].

B. Running a FedLab instance

At the time of this writing, three partners take part in the FedLab, within the NWO project INTERSECT [15]. The FedLab Server is deployed at TU/e, Eindhoven, Netherlands, while Bundle Box instances are running at TU/e (Eindhoven, Netherlands), University of Twente (Enschede, Netherlands), and TNO (Den Haag, Netherlands). Figure 3 shows a picture of the deployment at TU/e, Eindhoven. The FedLab server and two Bundle Box instances run on an Intel(R) Core(TM) i7-4700MQ laptop, equipped with 16GB of RAM and 230GB of HDD. The local IoT domain shared through the FedLab includes one Foscam IoT smart camera, one Ring IoT smart doorbell, and one TP-Link IoT smart plug. Two access points are included in the setup, just to ease physical connections.

We now describe what a typical workflow looks like from the perspective of a FedLab user, connecting to the FedLab via

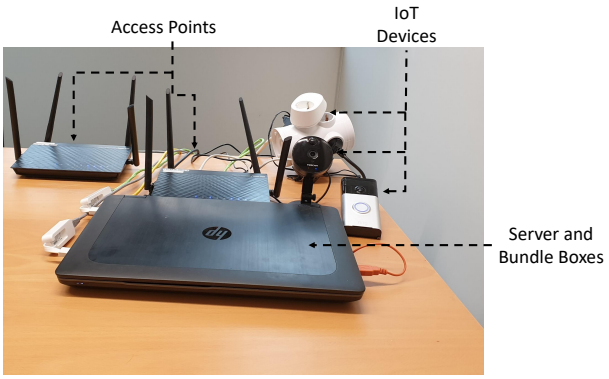

Figure 3. The test setup at TU/e, including the laptop hosting the FedLab server and two bundle boxes, and the local IoT domain, including a Foscam IoT smart camera, a Ring IoT smart doorbell, and a TP-Link IoT smart plug. Access Points are included in the setup to ease physical connections.

```
[intersect@fedlab vagrant box]$ vagrant ssh
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-109-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon May  9 14:56:38 UTC 2022

  System load:  0.24              Users logged in:         0
  Usage of /:   5.8% of 38.71GB   IPv4 address for enp0s3: 10.0.2.15
  Memory usage: 27%               IPv4 address for enp0s8: 192.168.5.1
  Swap usage:   0%                IPv4 address for tun0:   192.168.255.10
  Processes:    122


0 updates can be applied immediately.


*** System restart required ***
Last login: Mon May  9 14:08:42 2022 from 10.0.2.2
vagrant@ubuntu-focal:~$
```

Figure 4. Screenshot of the Bundle Box environment on the partner's host.

the Bundle Box. After the setup of the Bundle Box, the user can use any SSH client to access the Bundle Box. Figure 4 reports the screen shown at connection time. By default, the Bundle Box shows the information on the current system usage (computational load, memory usage, and running processes), as well as information on updates availability. The user can then connect to the desired resource, either using the browser or a client via the terminal, using information retrieved from the FedLab Directory.

IV. FedLab Use Cases

In this section, we showcase the potential of the FedLab in the *IoT security* research domain. Sec. IV-A shows how to use the capabilities in FedLab to test the robustness of Intrusion Detection Systems (IDSs), while Sec. IV-B describes a security-related experiment enabled by the FedLab. Both presented use cases focus on security aspects of IoT deployments, but applications to other domains, such as performance or reliability analysis, can be equally devised.

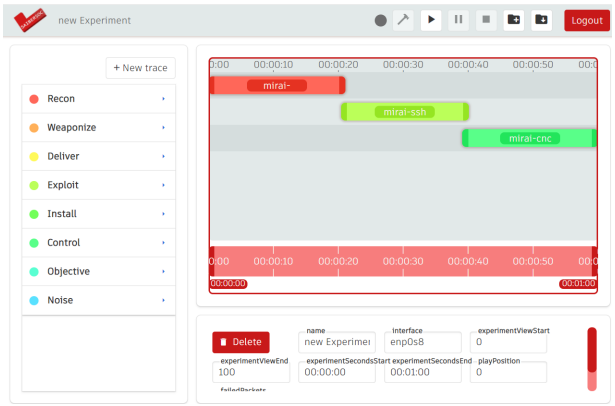

Figure 5. SAIBERSOC web interface, where the user has configured an attack trace made up of three consecutive phases.

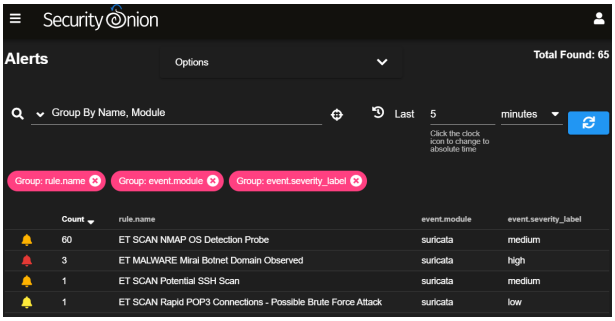

Figure 6. Web interface of the *TUe-SOC* capability, showing alerts due to the injection of malicious traffic from the *TUe-SAIBERSOC* capability.

A. Usage of FedLab Capabilities

In this section, we show an example of the usage of capabilities within the FedLab. Specifically, we leverage the SAIBERSOC tool described in [16]. SAIBERSOC is a tool enabling security researchers to assess the detection capabilities of a security monitoring infrastructure (such as a Security Operation Center, SOC) by injecting attack traces into the security sensors. The tool leverages the MITRE ATT&CK Framework and defines a procedure to automatically assemble and inject synthetic attacks to network domains monitored by an operational Security Operation Center (SOC). By analyzing metrics such as the detection accuracy, detection time, and time-to-investigation against such attacks, security researchers can evaluate the robustness, effectiveness, and efficiency of their monitoring infrastructure, as well as take actions to improve it in case of non-compliance to minimal requirements.

We deployed an instance of the SAIBERSOC tool on a dedicated machine, and we shared it as a *capability* within the FedLab, namely, the *TUe-SAIBERSOC*. Through the shared capability, users can specify how to build the attack traces, i.e., which attacks to imitate for the generation of the synthetic trace, how to combine those attacks, the packet interarrival times and throughput.

The client using the service also has to specify one or more IP addresses, uniquely identifying which devices to target for the attack. Then, using the web interface, the client has to press the button *Build* to assemble the attack traces into a single one, and then press *Start* to launch the assembled attack against the specified device(s). Figure 5 shows a screenshot of the web interface available for *TUe-SAIBERSOC*, where the tool has been configured to execute a complex attack in three phases.

At TU/e, the local Bundle Box also exposes the *TUe-SOC* capability. In brief, *TUe-SOC* is a collection of security monitoring tools, often used in deployed SOCs to monitor and identify the presence of malicious traffic. *TUe-SOC* leverages Security Onion, a free and open Linux distribution for threat hunting, enterprise security monitoring and log management [17]. It uses Suricata as the core IDS, while the data pipeline is built upon Elastic Stack tools, such as Beats, Logstash, Elasticsearch, and Kibana. The *TUe-SOC* capability supports the input of a specific network interface, to be monitored for detection of potentially malicious traffic. Therefore, as a use case of the combined usage of shared capabilities within the FedLab, a user interested to evaluate the performance of the *TUe-SOC* can inject synthetic traffic generated through the *TUe-SAIBERSOC* capability, and then look at the metrics of interest to deduce the effectiveness of the monitoring infrastructure. We ran the attacks build as in Fig. 5 on a specific network interface, and we configured the *TUe-SOC* capability to analyze traffic on the same interface. As shown in the screenshot of Fig. 6, the *TUe-SOC* detects the presence of malicious activity (see the alerts shown on the dashboard, as generated by the *Suricata* module), and provides additional information on the type of threat affecting the monitored network interface. Besides, the *TUe-SOC* also indicates the severity of the attack.

FedLab users can use the *TUe-SAIBERSOC* and *TUe-SOC* capabilities to run the desired shared experimentation. For example, partners interested in assessing the performance of their monitoring platform against specific IoT attacks can deploy such tools as a local capability, share it through the FedLab, configure *TUe-SAIBERSOC* to build several attack traces, and then run such attacks on the same interface monitored by the monitoring tools, to assess their robustness. At the same time, if new attacks are conceived by a partner, they can be shared through the FedLab and configured to generate malicious traffic on the same network interface monitored by the *TUe-SOC* capability, to evaluate the robustness of the attack against an example monitoring infrastructure. Likewise, the *TUe-SOC* can be used as a capability in its own right to monitor specific traffic in and from IoT devices of interest, for example, to detect potentially unwanted open connections towards suspicious servers (or servers in specific countries).

B. Enabling shared experimentation through the FedLab

In this section, we show how to enable shared experimentation through the FedLab. Specifically, we rely on *MUDscope*, an IoT network intrusion detection approach [18]. MUDscope first captures anomalous traffic affecting an IoT device in a network environment by leveraging the device's Manufacturer

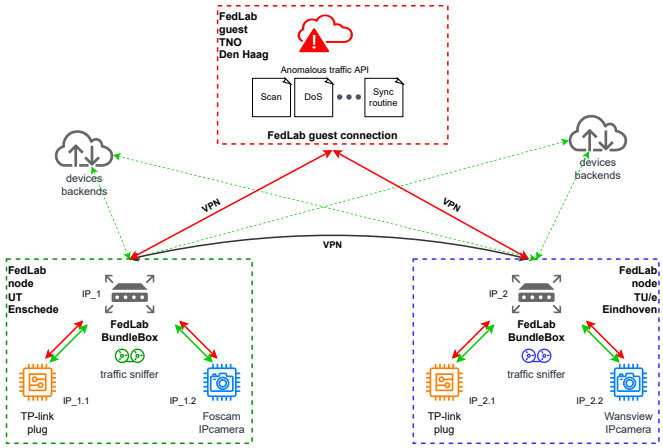

Figure 7. Architecture of the shared experiment using the FedLabṪwo Bundle Boxes are deployed in Eindhoven and Enschede, exposing networks hosting two equivalent pairs of IoT devices. Guest access is enabled in Den Haag.

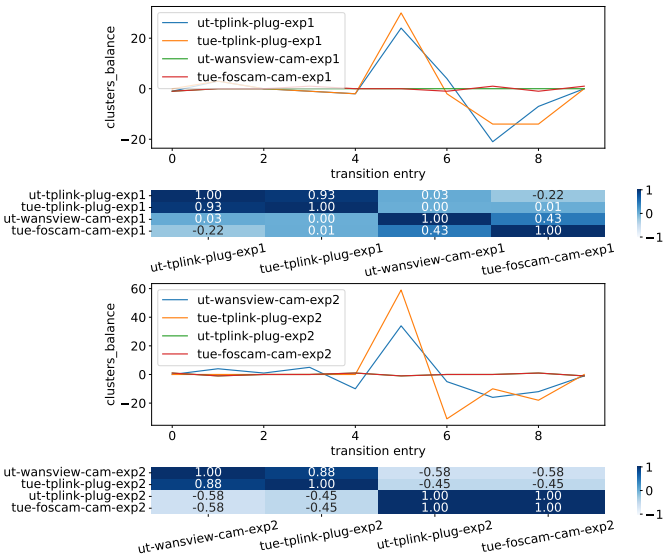

Figure 8. Sample of the results from [18], showing the signature fluctuations over one of the metrics used to detect network anomalies, namely, (`clusters_balance`), for two attacked devices (blue and orange lines) and two non-attacked devices (green and red lines). The matrices below the figures show the respective correlation values. The upper plot considers as attacked devices the IP camera in Enschede and the IP camera in Eindhoven, while the bottom one considers as attacked devices the IP camera in Enschede and the smart plug in Eindhoven. The generated anomalous signatures are matching, while negligible fluctuations are recorded for non-attacked devices.

Usage Description (MUD) profile—a manufacturer-provided networking whitelist defined according to the MUD IETF specification [19]. The tool then characterizes how this traffic changes over time and generates signatures for the captured anomalous activities. As network threats targeting multiple devices produce similar effects on the devices' MUD-rejected traffic, MUDscope correlates anomalies collected from multiple geographically-distributed devices, and it detects evolving IoT threats.

In this context, the FedLab has been used to instantiate a distributed testbed to evaluate MUDscope, enabling local traffic logging and information sharing across multiple locations. Recall from Sec. III that any instance of the Bundle Box supports the default *sniff* functionality, logging in a dedicated file all the traffic flowing from and to local resources shared to the FedLab. As depicted in Fig. 7, we deployed two equivalent pairs of IoT devices at distinct geographical locations, i.e., TU/e (Eindhoven, Netherlands) and UT (Enschede, Netherlands). These deployments use Bundle Boxes, as they share resources to the FedLab. Guest access was then enabled for a server at TNO, Den Haag (Netherlands), used to launch malicious traffic to the devices in Enschede and Eindhoven.

Through several experiments, we launched the same attack routine from the TNO FedLab guest to selected devices across the deployments at Enschede and Eindhoven. In each experiment, we recorded the local traffic received by the IoT devices at both locations through the default *sniff* Bundle Box functionality. Then, the recorded traces were processed with *MUDscope* from [18]. As shown in Fig. 8, MUDscope was able to identify a high correlation score in the anomaly signatures obtained from the MUD-rejected traffic of the attacked devices, to enhance IoT network threat situational awareness. Therefore, the FedLab proved instrumental in easily setting up a distributed IoT testing environment for this research case.

Table III
QUALITATIVE COMPARISON OF THE FEDLAB AND WITH RELATED WORK.

Ref.	R1	R2	R3	R4	R5	R6	R7	R8	R9
[4]	○	●	●	●	○	●	○	●	●
[5]	●	-	●	○	◐	○	●	-	-
[6]	●	-	-	●	○	●	●	◐	●
[20]	○	-	○	●	○	●	○	●	●
[21]	●	-	●	○	◐	○	●	●	●
[22]	●	-	●	●	○	●	●	-	◐
[23]	○	●	○	●	◐	●	●	●	●
FedLab	●	●	●	●	●	●	●	●	●

V. RELATED WORK

Many IoT-related testbeds have been proposed in the scientific literature in the last years, with a focus on either networking functionalities or security capabilities.

The MoteLab testbed offered a solution for multiple simultaneous firmware updates [4]. It provides a web interface to instruct a Command & Control server to distribute firmware updates to network nodes, or to schedule a job. Although it is released as open-source, MoteLab does not allow remote join of the testbed (req. R1 in Sec. II-A), and neither does remote communication. Also, it does not support either the sharing of capabilities (R5) or multiple IoT protocols (R7).

The Kansei testbed in [5] is a distributed platform supporting multiple hardware and protocols. However, it only allows sharing of IoT devices, not capabilities (R5). It also does not support central logging and auditing (R4), and it does

not include a mechanism for resource discovery (R6). At the same time, it is not clear how users can install the software needed for joining the platform (R8), and if such software is compatible with multiple operating systems (R9).

NetEye [20], focused on Wireless Sensor Networkss (WSNs), included 130 TelosB motes with IEEE 802.15.4 radios and 15 Linux laptops equipped with IEEE 802.11a/b/g radios. However, it is not distributed (not meeting the req. R1). Also, it does not support authentication (R3), remote sharing of devices and capabilities (R5), and heterogeneous IoT devices (R7).

Wisebed [21] is the testbed closest to the FedLab. It is a distributed platform easing experimentation across multiple distributed locations, supporting peer authentication and heterogeneous devices. However, Wisebed is meant to share devices only, not capabilities (R5). Also, users can connect to the platform only using a web page (R9), and automated resource discovery is not supported (R6).

Communication among several remote nodes is offered by FIT IoT-LAB, interconnecting $2,700$ sensors across France [22]. The user can get bare-metal access to the nodes via a web interface, upload firmware, and run tests. However, the platform does not allow the sharing of capabilities (R5), and it runs only on a few software platforms (R9).

Considering security-oriented solutions, the authors in [23] recently discussed the current lack of means to structurally evaluate the security of IoT devices. To close the gap, they discussed the architecture of a system to evaluate IoT devices' security, providing also a reference implementation. Although supporting some of our features, it is not geographically distributed (R1), it does not support peer authentication (R3), and public device access is not supported (R5). Similar issues can be found in the testbed proposed in [6], focusing on the automation of security tests. Such a testbed also supports access control, but it is not a plug-and-play solution (R8), and it is not clear how it can support isolation and authentication.

Table III summarizes our discussion and highlights the features supported (or not) by the surveyed platforms, including our FedLab. Although the FedLab shares similarities with the discussed testbeds, none of the available solutions fulfils all the identified requirements, at the same time. Such requirements are supported, instead, by the FedLab, as previously detailed in Tab. I. Therefore, the FedLab fulfils by-design all the identified requirements, emerging as the most suitable solution for shared research and experimentation on IoT across geographically-distributed consortia.

VI. Conclusions and Future Work

In this paper, we presented the FedLab, the first open-source virtual platform enabling shared research and experimentation of IoT functionalities across multiple geographically-distributed sites. Through the FedLab, partners in a consortium can share IoT devices and IoT-based functionalities conceived by other peers, easily carrying out shared research and tests in a distributed environment. As important added values, FedLab has been conceived with interoperability and ease of use in mind, being compatible with multiple IoT technologies and usable out of the box with minimal configuration. We also showed the FedLab at work, providing two use-cases in the IoT security area where its deployment enabled shared experiments and contributed to easing experimentation involving multiple geographically-distributed partners.

Future work will focus on the support of additional features, such as the booking of resources and scheduled experiments.

Acknowledgments

This work has been supported by the INTERSCT project, Grant No. NWA.1162.18.301, funded by Netherlands Organisation for Scientific Research (NWO). The findings reported herein are solely responsibility of the authors.

References

[1] D. Singh, G. Tripathi, and A. J. Jara, "A survey of Internet-of-Things: Future vision, architecture, challenges and services," in *IEEE World Forum on IoT*, 2014, pp. 287–292.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, et al., "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] J. Kim, J. Yun, S. Choi, et al., "Standard-based IoT platforms interworking: implementation, experiences, and lessons learned," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 48–54, 2016.

[4] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: A wireless sensor network testbed," in *Int. Symp. on Information Processing in Sensor Netw.* IEEE, 2005, pp. 483–488.

[5] E. Ertin, A. Arora, R. Ramnath, et al., "Kansei: A testbed for sensing at scale," in *Proc. of 5th Int. Conf. on Information processing in sensor networks*, 2006, pp. 399–406.

[6] O. Waraga, M. Bettayeb, Q. Nasir, et al., "Design and implementation of automated IoT security testbed," *Computers & Security*, vol. 88, p. 101648, 2020.

[7] Max Meijer, Giacomo Tommaso Petrucci, Matthijs Schotsman, "Open-source code of the FedLab Platform," https://github.com/GTP95/FedLab, 2022, (Accessed: 2022-Jun-03).

[8] "Openvpn documentation," (Accessed: 2022-Jun-03). [Online]. Available: https://openvpn.net/community-resources/reference-manual-for-openvpn-2-4

[9] Easy RSA, "Easy-RSA 3," https://easy-rsa.readthedocs.io/en/latest/, 2022, (Accessed: 2022-Jun-03).

[10] "Networking using a macvlan network," 2021, (Accessed: 2022-Jun-03). [Online]. Available: https://docs.docker.com/network/network-tutorial-macvlan/#prerequisites

[11] "Overview of Docker Compose," 2022, (Accessed: 2022-Jun-03). [Online]. Available: https://docs.docker.com/compose/

[12] "Netbox documentation," (Accessed: 2022-Jun-03). [Online]. Available: https://netbox.readthedocs.io/en/stable/

[13] "Spring framework," (Accessed: 2022-Jun-03). [Online]. Available: https://spring.io/projects/spring-framework

[14] "Netbox docker image," (Accessed: 2022-Jun-03). [Online]. Available: https://hub.docker.com/r/netboxcommunity/netbox

[15] "INTERSECT: Towards an Internet of Secure Things," 2022, (Accessed: 2022-Jun-03). [Online]. Available: https://intersct.nl

[16] M. Rosso, M. Campobasso, G. Gankhuyag, et al., "SAIBERSOC: Synthetic Attack Injection to Benchmark and Evaluate the Performance of Security Operation Centers," in *Annual Computer Security Applications Conf.*, 2020, pp. 141–153.

[17] Security Onion Solutions, LLC, "Security Onion," https://securityonionsolutions.com/, 2022, (Accessed: 2022-Jun-03).

[18] L. Morgese, "Stepping out of the MUD: contextual network threat information for IoT devices with manufacturer-provided behavioural profiles," Master's thesis, University of Twente, 2021.

[19] E. Lear, D. Romascanu, and R. Droms. (2019-03) IETF RFC 8520 - manufacturer usage description (MUD) specification. [Online]. Available: https://tools.ietf.org/html/rfc8520

[20] X. Ju, H. Zhang, and D. Sakamuri, "NetEye: a user-centered wireless sensor network testbed for high-fidelity, robust experimentation," *Int. Journal of Communication Systems*, vol. 25, no. 9, pp. 1213–1229, 2012.

[21] I. Chatzigiannakis, S. Fischer, C. Koninis, et al., "WISEBED: an open large-scale wireless sensor network testbed," in *Int. Conf. on Sensor Applications, Experimentation and Logistics*. Springer, 2009, pp. 68–87.
[22] C. Adjih et al., "Fit iot-lab: A large scale open experimental iot testbed," *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015.
[23] S. Siboni, V. Sachidananda, Y. Meidan, et al., "Security testbed for Internet-of-Things devices," *IEEE Trans. on Reliability*, vol. 68, no. 1, pp. 23–44, 2019.