The Attack Surface of Wet Lab Automation

Naor Dalal, Yossi Oren, Yuval Dorfan, Jonathan Giron, and Rami Puzis

Abstract Robotic liquid handlers save human effort and are, in many cases, faster and more precise than a human operator. They can be operated and controlled remotely and do not require technical programming skills from their operators. Unfortunately, like many other high-tech products, robotic wet lab automation may have exploitable vulnerabilities and design weaknesses that allow subversion by an adversary. The distributed nature and remote control capabilities of wet lab automation expand its attack surface increasing the opportunities for an attack to interfere with the executed biological protocols, affect medical products, and alter test results. Perimeter defenses are known to be insufficient for proper protection of systems. Security needs to be considered throughout the entire pipeline of wet lab operations, including machinery, local- and cloud-based software, and even biological protocols. In this chapter, we review the most prominent types of robots in a biological laboratory through the lens of cyber-biosecurity and map the general attack surface of wet lab automation.

Keywords Cybersecurity · Laboratory automation · Liquid handlers · Cyberbiosecurity

---

N. Dalal
Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Y. Oren · R. Puzis (✉)
Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Cyber@BGU, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Y. Dorfan · J. Giron
Innovation center, Reichman University, Herzliya, Israel

1 Introduction

Pipetting, preparing, and transferring liquids require considerable time and effort in a traditional wet lab environment. Robotic liquid handlers save human effort and are, in many cases, faster and more precise than a human operator. Wet lab automation goes further, allowing a biologist to automate experiments or production via robotic control. It does not require technical programming skills and saves time and effort allowing the biologist to focus on the experimental design and data analysis. Wet lab automation frameworks can be operated and controlled remotely via a local network [1, 2] or even through a cloud [3, 4].

The core component of wet lab automation solutions is the lab robot. These robots have different capabilities, such as precise work with a pipette, liquid temperature control module, and precise liquid distribution, which can replace and scale up the work of a human lab technician. These robots carry out multiple steps in a biological protocol pipeline, starting with external biological inputs and ending with biological products, scientific data, or even clinical recommendations.

Unfortunately, like many other high-tech products, wet lab automation may have exploitable vulnerabilities and design weaknesses that allow subversion by an adversary. Regardless of the financial, ideological, or political motivation of the attackers, control over the production or experimental pipeline may result in serious adverse impacts ranging from disruption of the production to unintended and unanticipated dangerous biological byproducts.

The more distributed a wet lab automation control system is, and the more it is exposed to the Internet, the higher is the risk of an attack. Attacks can interfere with biological processes, affect medical products, and alter test results. Perimeter defenses, such as password-protected access and encrypted communication, are known to be insufficient for proper protection of systems. Security needs to be considered throughout the entire pipeline of wet lab operations, including machinery, local- and cloud-based software, and even biological protocols. Cross-site scripting, insecure applications, and insecure Internet-of-Things (IoT) controllers wired to the robots are just a few examples of potential attack vectors.

While there are many articles on cyberbiosecurity [5], biosafety and biosecurity [6, 7], cyberbiosecurity for DNA synthesis [8], assessing cyberbiosecurity vulnerabilities [9], protecting US food and agricultural system [10], harmful algal blooms (HABs) and the cyberbiosecurity of freshwater systems [11], and risk perceptions in the biotech sector [12], nevertheless, no previous work has discussed the particular security context of wet lab automation throughout the multiple steps of running the protocol pipeline.

In this research, we try to bridge this gap and try to shed light on the dangers and possible impacts of intervening with the running of a biological protocol in wet lab automation and the need to secure its proper execution.

Our contributions are as follows: First, we build a wet-lab automation ecosystem taxonomy and expand on each variable in the taxonomy. We also review a number of diverse robots in the field of biological laboratory automation and their capabilities. Next, we build and examine the pipeline of a running protocol, mapping the relevant parts for each step in the pipeline, and we describe what its role in the pipeline. For each step in the pipeline, we examine if it may be vulnerable and describe the required permission and access conditions which enable an adversary to attack this step. We create the connection between wet lab automation capabilities and the attack vectors, which attack vector can affect which capability. Finally, we perform a case study on several important lab automation protocols and show how an attacker can adversely intervene with them and what are the possible impacts of such attacks.

Wet lab automation is becoming more widespread supporting increased number of applications and deployment possibilities. Thus, it is important to consider the security aspects of wet lab automation as early as possible. By doing so, the community can prevent security-related configuration blunders with possibly fatal consequences.

2 The Wet Lab Automation Ecosystem

In this section, we analyze and present the taxonomy of the wet lab automation ecosystem as demonstrated in Fig. 1. The taxonomy shows the ecosystem of wet lab automation in general. Each leaf in the graph is variable of robot's criteria. Each wet lab automation robot can omit or add the variable in the taxonomy and implement him in his way. We present the different implementations and the generic way to implement it for each node in the taxonomy tree. The taxonomy breaks down the robot into logical parts, the ecosystem part that contains the hardware and software of the robot, and its commercial part.

2.1 Hardware

This section describes the hardware and physical (nonprogrammable) capabilities and component specification of the robot. This section is divided into several subsections; each subsection describes the hardware, ability, or physical feature of the robot.

2.1.1 Deployment

Deployment of robots involves placing the robots and their resources in specific location where they can perform their intended tasks. When you are setting up the robot laboratory infrastructure, you will be faced with multiple decisions, convenience, cost, and quality. There are two main options standing for you: on-premise robots and cloud robots (lab as a service). In this section, we introduce these two options and compare them.

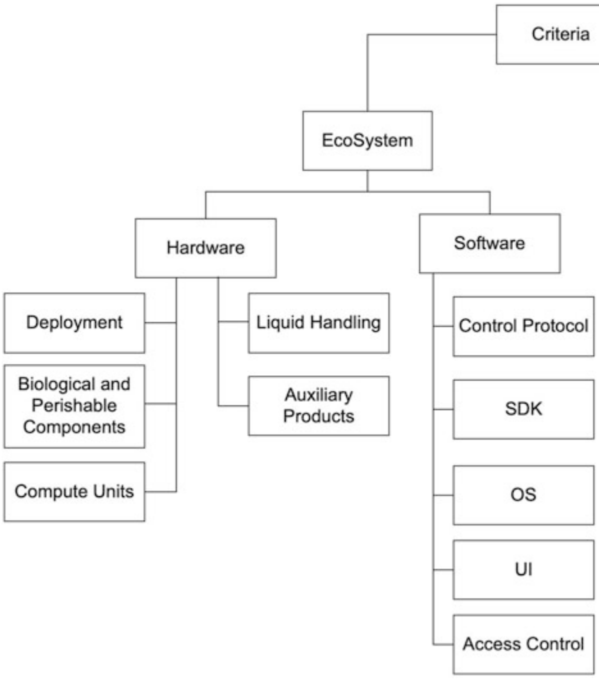

Fig. 1 Criteria hierarchy for wet lab automation ecosystem

On-Premise Most of the robots are 3D robots that provide open-source 3D models for do it yourself. You need to build the robots and store it in your lab. This requires you to access a 3D printer. You'll need to reserve an area in your lab for the robot and make sure you have basic knowledge of hardware assembly. You may need to purchase IoT devices (e.g., Arduino) for robot control and other connectable modules, for example, tip racks, well plates, and a syringe reactor. These robots are more dynamic and can be modified more easily and adapted to the needs of the laboratory, but they are less quality and simpler.

Cloud Lab Automation as-a-Service (CLAaS) Another type of robots are the CLAaS. These robots contain work cells that are woven together by an integrated stack of control software. A robotic cloud lab is a deeply integrated technology stack of biology, hardware, and software made available to its users via the cloud. Unlike traditional on-premise robots, a robotic cloud lab flexibly supports multiple assay types and is built from the ground up to be controlled remotely. These robots are more complex and have more capabilities and are usually also of better quality, but sometimes it is more difficult to adapt them to the needs of the laboratory.

On-premise robots in contrast to CLAaS are more available because the robot is located in your lab and can easily adapt to your purpose. But on the other side, CLAaS is more maintained and the quality is higher. On-premise lab prices are according to the level of equipment of the robot and quantity of the pluggable modules you buy. There are robots that you can buy from the company instead of assembling it yourself (i.e., OT-2).

2.1.2 Biological and Perishable Components

Here we list the physical components required to operate biological protocols.

Single and Multichannel Pipette A pipette is a laboratory instrument used to measure out or transfer small quantities of liquid. Multichannel pipettes generally come with either 8 or 12 pipette heads, easily allowing for a single device to fill multiple wells at a single time.

Pipette Tip Pipette tips are disposable attachments for the uptake and dispensing of liquids using a pipette.

Tip Racks Holders and replacement trays for disposable pipette tips are designed and packaged to facilitate the reuse of pipette tip boxes to reduce the overall amount of plastic waste.

Well Plates The well plate is a flat plate that looks like a tray with multiple wells that are used as small test tubes.

Tube Rack Test tube racks are laboratory equipment used to hold upright multiple test tubes at the same time. They are most commonly used when various different solutions are needed to work with simultaneously, for safety reasons, for safe storage of test tubes, and to ease the transport of multiple tubes.

2.1.3 Compute Units

Next is a list of common hardware compute units that are responsible for communication, processing, and control of the robot actuators.

Stepper Motor Driver Carrier (i.e., DRV8825) Stepper motor drivers are specifically designed to drive stepper motors, which are capable of continuous rotation with precise position control, even without a feedback system. Stepper motors are used for moving the robots in multiple axes (2 and 3 axes) separately and simultaneously.

Arduino Arduino is an open-source electronic platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects. Arduino can be used for two purposes: as an endpoint that can be connected to robots via Wi-Fi and an actuator that communicates with the robots via a proprietary protocol.

Raspberry Pi Raspberry Pi is a tiny computer about the size of a deck of cards. It uses what is called a system on a chip, which integrates the CPU and GPU in a single integrated circuit, with the RAM, universal serial bus (USB) ports, and other components soldered onto the board for an all-in-one package. Raspberry is used for communicating withrobots, that is, EvoBot Raspberry sends the G-code commands to the robot through a USB connection.

2.1.4 Liquid Handling

Liquid handling is the act of transferring liquid from one location to another in a laboratory, usually for testing purposes. The robots have varied types of liquid handling capabilities:

Shake the Tube A hardware module controlled by firmware that is designed to mix liquids in different frequencies.

Vacuum Aspiration A hardware module for pulling liquid up into the pipette tip.

Blow Out A hardware module for pushing an extra amount of air through the pipette tip, so as to make sure that any remaining droplets are expelled.

Dispense Liquids A hardware module for pushing out liquid from the pipette tip into plate or another implement.

2.1.5 Auxiliary Products

Some of the robots came with connectable modules that optimize and help with the experiment. We mention a short list of these products:

Camera Module Some robots have the ability to put a camera on top of the robot that will record all the experiments; this helps in exploring and understanding the experiment.

Microscope Module It is a pluggable module that helps biologists better observe the liquid during the experiment. It is an instrument used to examine objects that are too small to be seen by the naked eye. The camera and microscope can combine together by recording the experiment through the microscope.

Temperature Module It is a pluggable module that can control accurately the temperature of the liquids. Temperature module is a hot and cold plate module.

Magnetic Module The magnetic module is a magnetic bead-based chemistry block for extraction and purification. It automatically engages and disengages high-strength magnetic bars to seated well plates for magnetic bead-based purification protocols.

Thermocycler Module Thermocyclers are instruments used to amplify DNA and RNA samples by the polymerase chain reaction.

High-Efficiency Particulate Air (HEPA) Module HEPA is an efficiency standard of air filter.

Sensors In addition to the auxiliary components listed above, some robotic frameworks for the wet lab also include various sensors, such as: motion sensor, ultrasonic sensor, sound sensor, and light sensor.

2.2 Software

This section describes the programmable parts in the robot, according to the taxonomy tree in Fig. 1. Programmable parts could be software, firmware, or even the protocol between the components of the robot. Each subsection describes these programmable parts.

2.2.1 Control Protocol

The robots use various control protocols, some of which are proprietary and some are known standards.

G-code G-code is a software programming language used to control a computer numerical control (CNC) machine. It is used mainly in computer-aided manufacturing to control automated machine tools and has many variants. Raspberry Pi sends the G-code commands to the robot through a USB connection.

uArm Swift Pro Protocol The uArm Swift Pro is an open-source Arduino-based robot arm designed for desktop use. Based on the standard G-code protocol, they add a new protocol head in front of the G-code so that it can be more easily used and debugged. What is more, it is designed to be compatible with the standard G-code.

Proprietary Protocols Some robot designers created their own simple control protocols suitable for their robot. They programmed Arduino using the analog write and read pin functions.

2.2.2 Software Development Kit (SDK)

SDK is a collection of software development tools in one installable package. The robots provide an SDK for controlling the robots; most of the robots provide a python SDK. The SDK contains functionality for full control of the robots. Usually the SDK simply sends a hypertext transfer protocol (HTTP) request to a server that actually controls the robots, but some run on the computer that controls the robots. In some devices, the SDK command translates to Extensible Markup Language (XML)-Remote Procedure Call (RPC) (XML-RPC), a protocol that uses XML to encode its calls and HTTP as a transport mechanism. You can automate the robot action and create protocols by python script and API the robots reveals to the user.

On-premise robots can be modified, and you can automate it yourself because you have the firmware of the IoT devices.

2.2.3 Operating System (OS)

Arduino lacks a full operating system, usually writing code that is interpreted by its firmware. However, Raspberry Pi has all the features of a computer; it needs an operating system to run and comes with a fully functional operating system called Raspberry Pi OS. In addition, sometimes there is a personal computer (PC) that controls robots or runs the HTTP server; its operating system can be any operating system that runs python (especially Windows or Linux).

2.2.4 User Interface (UI)

Several robots have interactive webpage graphical user interface (GUI) to control the robot, and some have smartphone applications. The GUI displays the entire protocol and robot control process and can be changed in any time. Behind the scenes, the beautiful GUI is converted to either code running on the IoT device or to Application Programming Interface (API) commands. OpenLH, for example, builds their GUI with Google's Blockly interface [13] which is converted to python code running on the computer which controls the arm of the robot. Another type of UI is the command-line interface (CLI); some robots provide commands that you can run from the CLI and automate the robots with it. Another type of robot does not provide GUI or CLI; the programmer needs to write the protocol using integrated development environment (IDE).

2.2.5 Access Control

For most of on-premise robots, there are no security aspects in the software. Some of them [1] created an open Wi-Fi by one of the IoT devices, and everyone in the same local area network (LAN) can control the robot. Others just need to connect to HTTP server through specific port, and you are free to go and run every protocol you want. However, CLAaS place more emphasis on security and use well-known security models such as hypertext transfer protocol secure (HTTPS) and two-factor authentication (2FA); 2FA is a security method that adds an additional layer of protection on top of just your username and password. It is a method of verifying that the person who is trying to access your account is who they say they are.

2.3 Commercial Aspects

This section describes the commercial aspect of the robots. This section describes the price ranges of wet lab automation robot types, the support that the developers of the robot give, and the community and the distribution of the robots.

2.3.1 Price

The robot prices range from $400 for open source and do-it-yourself robots to $9000 for robots that you got full assembly with multiple hardware components as described above. There are open-source robots that offer you full assembly instead of do-it-yourself. CLAaS robots are for subscription.

2.3.2 Support

Commercial robots run by companies are including contact support, return policy, warranty, and documentation. In contrast, open-source robots are less maintained; this is reflected in the lack of good documentation, contact support, and quality.

2.3.3 Community

Commercial robots are widely distributed; there are many companies that collaborate with the company that builds the robot; either it's CLAaS or on-premise. But some robots (do-it-yourself) are less distributed, and there are not too many sources on the community of these robots.

2.4 Summary

Detailing and mapping the taxonomy of wet lab automation ecosystem help us to understand better how the robots are built and what their capabilities are. It sheds light on where the security failures may be found and where a potential attacker could intervene in the system. We analyzed each property in the taxonomy considering whether this property may have security failures and whether an adversary can utilize it to his advantage.

3 Biological Laboratory Robots

In this section, we present some examples of wet lab automation robots. For each of them, we detail about its capabilities, hardware, and software. We showcase the uniqueness of each robot and how it differs from the other robots shown in this section. Furthermore, we attached figures of the robots and links for their open source and cited academic articles if exist.

Review Methodology In order to obtain the information about the robots, we read datasheets and published articles describing the robots [1, 14, 15]. To understand the developer's perspective on creating and running custom protocols, we examine the robots' APIs [16, 17] and attempt using the API ourselves. Further, in order to understand how a reviewed framework operates behind the scenes, we inspect the open-source code of the robots [18–22]. Such inspection often reveals issues not listed in the datasheets and API specifications. This review methodology is limited in a sense that we did not have access to the source code of all robots. The close source robots were examined in a less profound way. In Sect. 6, we elaborate the limitations in more detail.

3.1 Fully Integrable Noncommercial Dispensing Utility System (FINDUS)

FINDUS [1] is an on-premise open-source [20] 3D Printable Liquid-Handling Workstation for Laboratory Automation in Life Sciences. FINDUS hardware contains: (i) 3D-printed parts with an Anycubic 4Max printer; (ii) four stepper motors, for XY drives, Z drive, and pipet drive; (iii) DRV8825 controller boards and controlled stepper motors using a motor library provided by Laurentiu Badea; and (iv) two Arduino NodeMCU 1.0 (ESP-12E Module).

FINDUS software builds from python package for controlling the robot from PC through Wi-Fi and Arduino code that implements API server for commands from PC and controls the movements and shakers.

FINDUS is able to (i) start/stop shake the tube; (ii) start/stop vacuum aspiration; (iii) move in X, Y, and Z axes; (iv) move pipet; (v) move in X and Y axes simultaneously; and (vi) set position for X, Y, and Z axis pipette.

We can see in Fig. 2 the FINDUS workstation and its components. There are three-axis motion motors, syringe reactor, tip rack, and shaker motor, and more.

3.2 EvoBot

EvoBot [14] is an open-source [19], modular, liquid-handling robot for scientific experiments. Figure 3 shows a schematic view of the electronics of EvoBot and

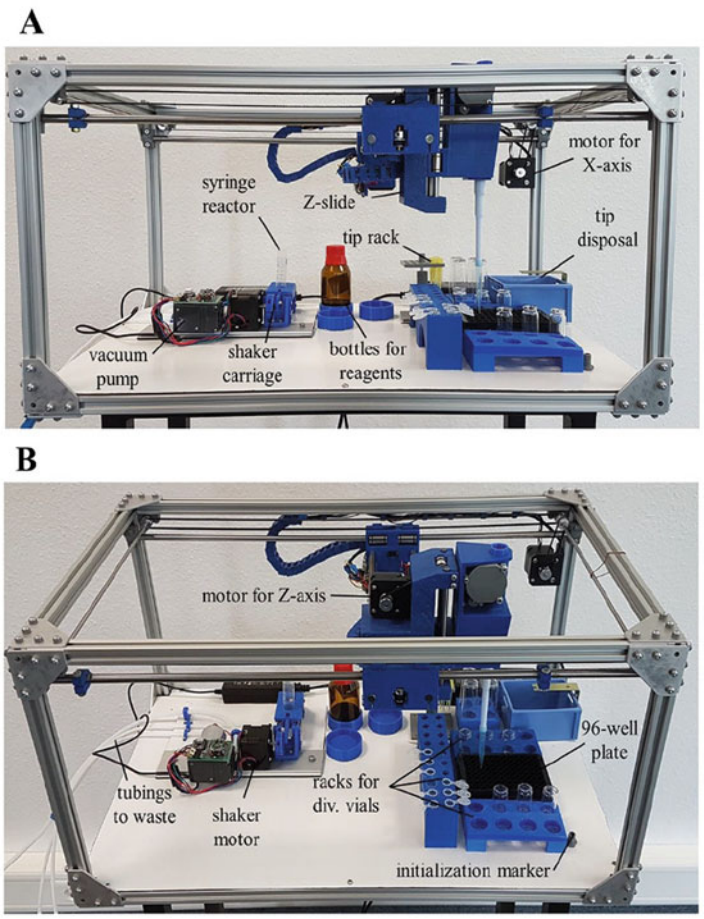

Fig. 2 FINDUS workstation, from FINDUS [1]

its different printed circuit boards (PCBs). The core of the electronics is based on electronics used in the open-source 3D printer community. EvoBot is built from the following, as shown in Fig. 4:

(i) Three layers: an actuation, an experimental, and an observation layers.
(ii) Actuation layer holds modules and can be moved in the horizontal plane by using two stepper motors.
(iii) Experimental layer supports the objects of the experiment such as petri dishes, microscope slides, or tubes.
(iv) Observation layer is optional, and most modules plugged into this layer are used to sense or observe the ongoing experiments, for example, camera and microscope.

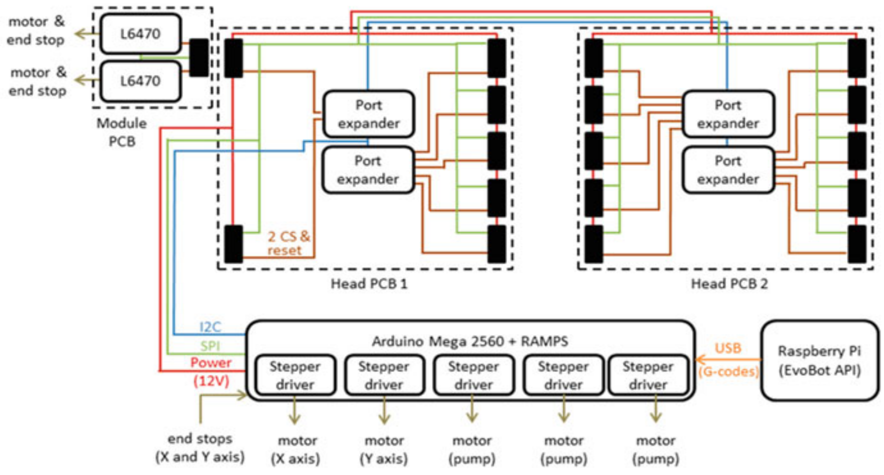

Fig. 3 EvoBot electronic schematic view, from EvoBot [14]

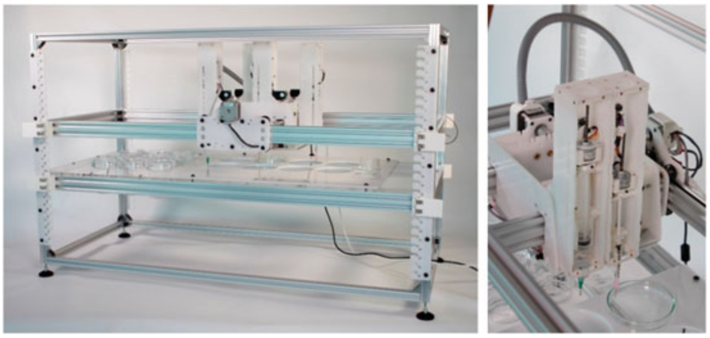

Fig. 4 EvoBot liquid-handling robot, from EvoBot OpenLH [14]

(v) Three different kinds of modules: a syringe module, a pump-based dispensing module, and a heavy payload module (microscope, three-dimensional scanner).
(vi) Arduino and Raspberry Pi 3.
(vii) Stepper motors.

EvoBot includes a software part that contains the following:

(i) Arduino runs a modified version of the Marlin firmware, which is widely used to control 3D printers using G-code.
(ii) Raspberry Pi sends the G-code commands to the robot through a USB connection.
(iii) Python API gives users access to control the robot.

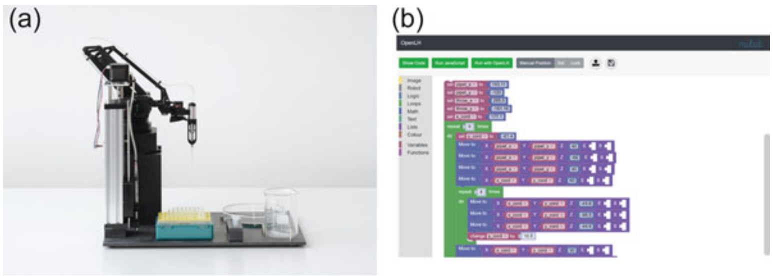

Fig. 5 OpenLH, from Ref. [21]. (**a**) uArm Swift Pro. (**b**) OpenLH blockly interface

(iv) Users can interact directly with the robot using a GUI, or they can run programs directly on the Raspberry Pi.

EvoBot is able to perform the following:

(i) The syringe module moves liquids with precision, it can move the syringe up and down in addition to the movement of the plunger.
(ii) The syringes can be easily replaced by just loosening and tightening one screw.
(iii) The dispensing module can pump up to four liquids and is used to wash Petri dishes or dispense pure reagents into vessels' start/stop vacuum aspiration.
(iv) Heavy payload module to hold a 3D scanner.

3.3 *OpenLH*

The OpenLH [15] is an open-source [21] liquid-handling system based on an available robotic arm platform (uArm Swift Pro) which allows for creative exploration by biologists and bio-enthusiasts. OpenLH is built from three main parts: (i) an open-source robotic arm, uArm Swift Pro [18]; (ii) a linear actuator-operated syringe pump; and (iii) the custom-made liquid-handling attachment, as can be seen in Fig. 5a.

The uArm runs on top of an Arduino Mega 2560 with a custom version of Marlin firmware (available under GPL license). The robot operates using G-code definitions sent through universal asynchronous receiver transmitter (UART) protocol. OpenLH software is built from several parts as the following:

(i) The user may generate different programs manipulating the arm using Google's Blockly interface [13] as can be seen in Fig. 5b.
(ii) The generated program is then compiled to python code, using the Swift API (which compiles to G-code commands).
(iii) It is possible to save programs for later use and upload images for the Bitmap to bioprint feature.

OpenLH has the following main features:

(i) Move To: Move the arm to a specific location. To use it, just generate a new move to block (from "Robot" section) as well as the relevant coordinate block (from "Robot" section). In the coordinate block, X Y Z stands for the coordinates, E for extrusion level, and S for movement speed.
(ii) Move Wrist: Rotate arm's wrist with the required angle. It is useful to drop used tips from the arm to a disposal area.
(iii) Bitmap to Bioprint: It is an interface that would load a portable network graphics (PNG) bitmap, select all the pixels of a single color, and print these pixels with the OpenLH. To use it, just generate a new image block (from "Image" section) as well as the relevant coordinate blocks (from "Robot" section).
(iv) Manual Position: Puts the arm in disjoint mode, allowing the user to move it around manually and sample coordinates. After reaching a desired location, a tip to pick up, for example, hit set button to generate the location's coordinates as a new usable block.

3.4 Opentrons OT-2

Opentrons [2] OT-2 is an open-source [22] liquid-handling robot. Opentrons OT-2 is built from following three sections:

(i) Labware – You must tell the protocol context about what should be present on the deck (well plate, tube rack), Labware Library.
(ii) Pipettes – You define the instruments required for your protocol. You tell the protocol context about which pipettes should be attached and which slot they should be attached to (11 slots on the deck).
(iii) Commands define the commands that make up the protocol. The most common commands are aspirate, dispense, pickup tip, and drop tip. Opentrons OT-2 pipette configurations: Single- and eight-channel pipetting, two-pipette mounts, for a configuration of one or two single- or eight-channel pipettes. Pipettes are easily interchangeable. Opentrons OT-2 contains 11 deck slots that enable countless configurations; deck slots are compatible with standard SBS dimensions. Deck also includes a removable trash bin. Its connectivity is through Wi-Fi 2.4 GHz IEEE 802.11b/g/n, USB 2.0. It can be applied to connectable pluggable hardware modules; modules are peripherals that attach to the OT-2 to extend its capabilities: (i) temperature, (ii) magnetic, and (iii) thermocycler modules, as shown in Fig. 6b.

Opentrons OT-2 is an open-source do-it-yourself but can be bought from Opentrons starting at $5000. The OT-2 Python Protocol API is a simple python framework designed to make writing automated biology lab protocols easy. The python script is running by Opentrons App.

Opentrons OT-2 has advanced control: sometimes, you may write a protocol that is not suitable for execution through the Opentrons App. Perhaps it requires user

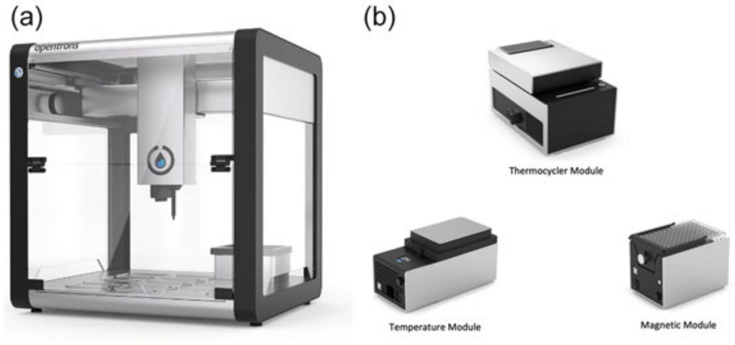

Fig. 6 Opentrons, from Ref. [2]. (**a**) OT-2 (**b**) Pluggable Modules

input; perhaps it needs to do a lot of things it cannot do when being simulated. There are two ways to run a protocol on the OT-2 without using the Opentrons App: Jupyter Notebook and CLI.

Opentrons OT-2 can be used for many purposes, for example, the following articles [23–25] show how it was used for COVID-19 polymerase chain reaction (PCR) testing automation.

3.5 Strateos

Strateos [3, 26] is a CLAaS [27] solution provider. The company's platform enables scientists to design, run, and analyze experiments remotely utilizing Strateos' robotic cloud labs. In addition, Strateos designs, builds, and implements modular cloud labs in their clients' facilities. Clients have the option of toggling between their own on-site facilities and Strateos' remote-controlled cloud labs for small molecule drug discovery, biologics, and synthetic biology workflows, advancing the digitization of laboratories via a hybrid lab solution.

Strateos' core technology is their lab control software that integrates and controls various instrument types, ranging from liquid handlers, bioreactors, and high content imagers to an array of ultrahigh-throughput screening instruments and devices. This modular, cloud-based software addresses common challenges in research operations and scientific experiment execution, even in labs with no automation currently. Strateos' software also focuses on solving operational challenges found in labs, such as managing experimental requests, asset and workflow calendaring, and executing between teams and individual users and enabling automatic data capture and centralization of scientific workflows to accelerate the design-make-test-analyze cycle and generate AI-enabled data that will aid in the discovery of new scientific insights. The software is modular and scalable from control of work cells to multiple client facilities as can be seen in Fig. 7b.

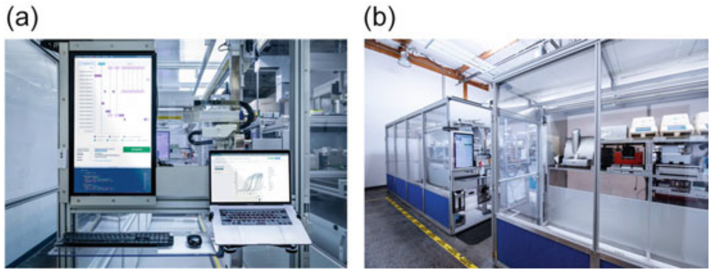

Fig. 7 Strateos, from Ref. [3]. (**a**) Strateos CLAaS solution. (**b**) Strateos work cell

Strateos developed and maintained Autoprotocol [28], as the open-source standard helping define experiments that are run over the Internet on remote robotic automation, moving research into the cloud. Open-source software packages are used to organize a collection of protocols and allow customers to build protocols using Python, or alternatively clients can access Strateos GUI to build and automate protocols. Autoprotocol is a JavaScript Object Notation (JSON) formatted data structure that provides a precise way of describing and automating biological and chemical protocols in the lab. A run can be submitted by posting properly formatted Autoprotocol to the server via the Strateos API.

3.6 Summary

In Table 1, we summarize the main components of each robot. Most solutions are deployed on-premise with a great deal of customization and lack of standardized security controls. This naturally increases the attack surface of open-source wet lab automation frameworks. Access control in majority of the solution is based on plain HTTP allowing man-in-the-middle attacks. We also observed high similarity among the biological protocol processing, server components, and control in different solutions. Vulnerabilities in the implementation and processing of biological protocols as well as components responsible for the protocol's execution may have the most severe impacts and thus deserve the most attention of security researchers.

4 Attack Surface

This section describes the potential attack vectors among the wet lab automation ecosystem. In this section, we analyze each entry point of the system and the all-pipeline of running protocol from his design and planning until it is running. We examine and describe each step in the pipeline and analyze whether it is possible

Table 1 Comparison of wet lab automation solutions

Component	Robot				
	FINDUS	EvoBot	OpenLH	Opentrons OT-2	Strateos
Deployment	On-premise	On-premise	On-premise	On-premise	CLAaS
Auxiliary products (short)	None	Camera	Sensors, temperature, electromagnet, and camera	Magnetic, thermocycler, temperature, and HEPA	Magnetic, thermocycler, temperature, and Illumina sequence
Control proto-Col	Proprietary protocol	G-code	G-code	G-code	Secretive
SDK	Python SDK	Python SDK and API	Python SDK and API	Python SDK and API	Python SDK and API
OS	Arduino	Marlin firmware and raspberry pi OS	Marlin firmware	Proprietary embedded hardware	Secretive
UI	IDE	GUI	GUI	GUI and CLI	GUI and CLI
Access control	Open Wi-fi	XML-RPC	HTTP server	HTTP server	HTTPS server and 2FA

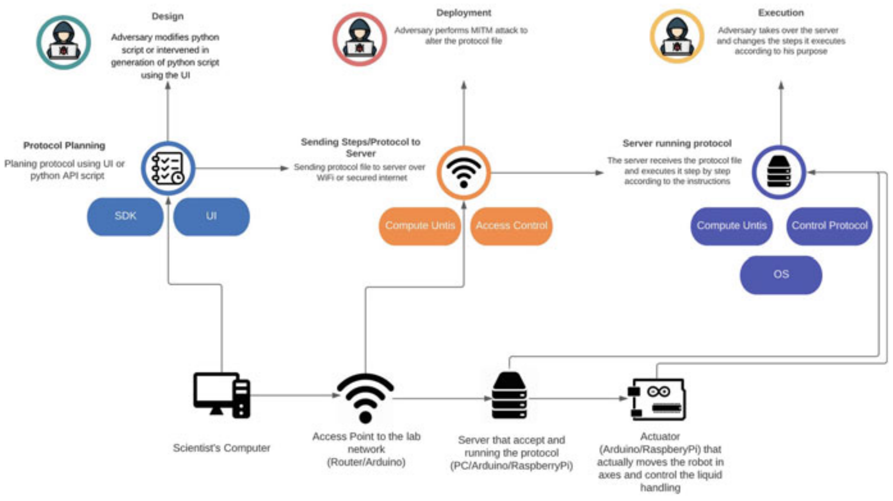
Design
Adversary modifies python script or intervened in generation of python script using the UI
Deployment
Adversary performs MITM attack to alter the protocol file
Execution
Adversary takes over the server and changes the steps it executes according to his purpose
Protocol Planning
Planing protocol using UI or python API script
SDK
UI
Sending Steps/Protocol to Server
Sending protocol file to server over WiFi or secured internet
Compute Untis
Access Control
Server running protocol
The server receives the protocol file and executes it step by step according to the instructions
Compute Untis
Control Protocol
OS
Scientist's Computer
Access Point to the lab network (Router/Arduino)
Server that accept and running the protocol (PC/Arduino/RaspberryPi)
Actuator (Arduino/RaspberyPi) that actually moves the robot in axes and control the liquid handling

Fig. 8 Attack surfaces on wet lab automation ecosystem

to intervene at this step in the final protocol, and if possible, we describe how can adversary do this.

We can see in Fig. 8 the pipeline of running protocol and the potential intervention of adversary. The bottom items describe the physical components and their connection to the proper step in the pipeline of running protocol. Each step is accompanied by a description and variable in the taxonomy tree in Fig. 1. The upper arrows coming out of each step express the possible intervention in the protocol or in the laboratory spec at this step. The variables that accompanied each step were designed to explain which parts are relevant to the intervention at this step. There are several articles that describe similar general structures such as on cyber-physical system (CPS) networks [29] but not on wet lab automation ecosystem.

4.1 Design

When the biologists start planning their protocol, mostly they have two main options to design the protocol as described in taxonomy Sect. 2: SDK or UI. Adversary that aims to change the protocol can do it with a grip on the biologist's computer by the following techniques:

Script Injection Adversary can inject himself to protocol script and inject his steps to protocol flow and prevent some steps in the protocol. This technique assumes that the adversary knows the protocol and knows how to replace specific steps to gain his goal.

Script Generator Intervention Adversary can inject himself to the UI application that generates protocol script and controls the protocol that will be generated. This technique doesn't assume anything about the knowledge of the adversary with the generated protocol; if the adversary knows the protocol the biologist intends to create, he can replace only specific steps with minimal intervention to gain his goal. But if he doesn't know the protocol the biologist intends to create, he can replace the generated protocol file to gain his goal.

Script Replacement If the adversary doesn't know the biologist's protocol, the adversary can treat the protocol file as a black box, and instead of intervening in an existing protocol, he can replace the protocol file with another file as he wishes. This technique assumes that the adversary doesn't know the protocol the biologist intends to create.

Labware Spec Intervention Some robots use labware spec or manifest. The spec uses to provide metadata and parameters and describes both labware's dimensions and properties. Adversary can intervene in this manifest and manipulate it in a malicious way to influence or damage the results of the protocol.

Affected Capabilities Design state script manipulation can influence among other things the following capabilities:

(i) Temperature module. Adversary can change the temperature of the temperature module and affect the proper procedure of the protocol.
(ii) Magnetic Module. Adversary can raise the magnets to induce a magnetic field in the labware.

(iii) Thermocycler module. Adversary can change the temperature of the block in which samples are located and temperature of the lid heating pad.
(iv) Vacuum aspiration. Adversary can change the amount of liquid that pulls into the pipette tip.
(v) Dispense liquids. Adversary can change the amount of liquid that push out from pipette tip into plate.
(vi) Blow Out. Adversary can prevent from blowing out the remaining droplets.

4.2 Deployment

After the biologist created the protocol script, either by SDK or UI, he needs to send the protocol (the script itself or JSON file that represents the protocol (Autoprotocol [28])). Adversary that aims to change the protocol can do it with a grip on the biologist's lab network/LAN by the following techniques:

Man in the Middle (MITM) Attack MITM is a known approach in many cases including CPS networks [30, 31]. Most of the open-source on-premise robots come with unsecured Wi-Fi and HTTP server and not HTTPS which reveals the biologist to MITM attacks. In such robots, adversary can perform MITM between the biologist's computer to HTTP server and modify the command the protocol that is sent to the server without the knowledge of the biologist. In addition, the attacker can change the labware spec to manipulate the lab environment for malicious purpose and damage or manipulate experiment results.

Impersonate Biologist Due to insecure control on on-premise robots, adversary can impersonate biologist's computer and control the robot and run any protocol he wants. This technique assumes that adversary has grip on some device in biologist's lab, that is, another computer in the network, the access point (Arduino).

Wi-Fi Sniffing Adversary can sniff the traffic on an open Wi-Fi using grip if a malware is present on some computer near the robot. Operating sniffing tools in a monitoring mode to collect traffic in open Wi-Fi networks does not require authentication. Thus, unencrypted Wi-Fi channel opened by one of the robot's components facilitates leakage of information about the running protocols in the lab and their results.

4.3 Execution

Finally, the protocol (Python script or JSON file) arrived to server that controls the robot and runs the protocol. Because in some robots the protocol represents using python script that runs as is in the server, adversary could run an arbitrary code (remote code execution (RCE)) in the server without the knowledge of the biologist. Running on the server that controls the robot could lead to dire consequences on the lab and protocol results. Execution is too late for intervention in the labware spec. Adversary with a grip on the server can make the following actions:

File System Manipulation Adversary that changed the protocol using one of the ways we mentioned above can manipulate the file system by inserting python commands into the protocol that accesses the file system. Adversary can do this because script execution is performed in an unsafe environment and not using restricted python [32]. Lack of restricted python and trust in the script itself without integrity validation of the script can lead to unwanted results of the protocol and leakage of results of previous protocols running on the robot. This attack vector assumes that adversary has grip on the server that runs the protocols and the protocols are sent to the server, which is a Python script that the server is running.

G-Code Command Intervention Most of the robots are controlled by G-code commands as mentioned in the taxonomy of wet lab automation. The server that runs the protocol actually sends G-code commands to the robot according to the command in the script. Adversary with knowledge on the G-code commands that are sent to the robot, which is not an unfounded requirement because all the robots we mention are open-source, with grip in the server can create its own G-code commands and send them to the robot to manipulate the running of the protocol or run preliminary steps to control the results of the protocol that the server intends to run.

Actuator Hijacking Actuator is the controller (Arduino/Raspberry Pi) that actually moves the robot in axes and controls the liquid handling. Those actuators are usually unsecured and written in the simplest way; in most of the types, it is Arduino that gets the G-code commands through USB or UART. If the adversary could hijack actuator through a vulnerability he exploits via USB or UART, he can run arbitrary code on the actuator and actually do whatever he wants, run commands as desired, skip commands from the biologist, and even leak the protocol using Wi-Fi that is sometimes found on these actuators. Adversary can control the Arduino using one of the vulnerabilities it exposed as detailed in security analysis and exploitation of Arduino devices in the Internet of Things [33].

5 Misuse Cases

In this section, we describe several automation processes in the biological field; we explain the process and how automation fits into them and its importance. In each of the processes, we explain how an attacker can intervene in it and what the possible damages are to such an attack.

5.1 Personalized Medicine

As genetic technology is improving, personalized medicine will replace conventional treatment [34, 35]. Together with the great hopes of tailor-made medicine, we identify a greater potential of damage due to automation failures.

For example, tissue testing is applied in cases of cancer to choose the best biological chemotherapy or its combined treatment using automated screening methods to comply each specific case.

DNA or RNA aptamer use for general and personalized therapeutics has shown great promise [36, 37]; production of template RNA/DNA aptamers is commonly done using automated DNA synthesis methods [38] that produce the required strands in a sequential way.

Attack on the DNA synthesis is presented in [39], who show acoustic side-channel attack methodology which can be used on DNA synthesizers to breach their confidentiality and steal valuable oligonucleotide sequences. The potential attack could result in null effect of the treatment or damage to the tissue.

High-throughput screening of dedicated medicine and factors including repurposing of generic drugs to measure hit conformation with a robust effect on the tested tissue.

Intervention in personalized medicine process can be performed by the following:

(i) Damage of the tissue while using nondrug-related influence and temperature can cause mismatches of hit confirmation and treatment of patients using noneffective drugs.
(ii) Damage the process of the drug administration for the screening.
(iii) Its weak point is its flexibility because it supports many types of treatments, so it can be disrupted relatively easily and substances can be omitted or added to the drug if adversary has grip on the robot.

5.2 COVID-19 PCR Tests

Automation of PCR testing [40, 41] is already used everywhere. PCR is the duplication and amplification of short and long DNA oligos. This process allows for the detection of minuscule samples of DNA [42] and is used to detect the presence of the COVID-19 virus in human samples. The impact of mistakes here could be dramatic on public health.

Intervention in automation of PCR testing can cause several damages as the following:

(i) Integrity. Adversary can damage the integrity of the test through swapping the samples of people and cause impairment of the test's integrity. Moreover, adversary can damage the integrity using malware on the PCR reader that alters the screening report and swapping people results. Adversary using a malware on the robot can manipulate the liquid's temperature and cause false-negative or positive results.

(ii) Confidentiality. Adversary with grip on the automation ecosystem can release the PCR test results to the public.

5.3 Sportsmen Doping Test Control

Doping is an old well-known issue in professional sport [43]. Throughout the process of competitive competition, many drug tests are conducted. The process includes sampling the sportsmen and women and identifying illegal substances in their blood or urine. The preparation and measurements of the samples using automated measures could provide a solution for many cases where the results are needed in a short time. While automation provides great advantages, it also poses risks to the integrity and coding of the samples and possibilities of cyber-attacks that can meddle in the analysis and reporting process of the results.

Similar to PCR tests, adversary can swap the samples of sportsmen and cause impairment of the test's integrity, swapping test of sportsmen that took drugs with test of clean sportsmen to evade punishment. In addition, adversary can contaminate the test with the standard that the test compares to and cause to a clean athlete to be considered to have taken drugs.

5.4 On-site Drug Production and Dispensing

In many cases, on-site production is necessary [44, 45]. It can reduce shipping costs and improve the quality of multiple products, especially medicine, such as vaccines that need special preserving conditions. The on-site robots will prepare the drugs using liquid and powder handling automation. Furthermore, they will provide necessary dispensing of ready-made tablets and liquids per client.

While providing great economic advantages, on-site production also poses threats through cyber-attacks that interfere or meddle in the preparation or tagging process of the prepared drugs.

Because of the automation of the process, adversary can cause robots to return the wrong type of medicine without the knowledge of the patient. Adversary also can damage the production of the medicine by omitting important substances of the medicine. Moreover, adversary could change the labels of allergen on the medicine package and cause people to have allergic reaction that could endanger their lives.

6 Summary and Conclusions

Key Takeaways In this chapter, we have analyzed the potential attack surface in the wet lab automation ecosystem. The most important finding from the analysis is that the biological protocol implemented as a Python script is dangerous. Vulnerabilities in the protocol editing tools, components that interpret and execute the protocols, and components operated by the protocols can lead to severe adverse consequences as described in Sect. 5. Some consequences can be prevented digitally by signing the protocol and executing it in a secured environment such as restricted python [32]. In general, a user-provided script should never be considered trusted.

Furthermore, many components in the robot environment are distributed and wireless. This allows the attacker to intervene with the robot operation in several stages. Therefore, secure operation of the robot requires hardened communication between the components.

Limitations Our collection data methodology for each device has some limitations because there are robots we haven't their source code and only API and datasheet documentations. It could be that some of the data we had on the rest of the robots does not quite fit these robots. Moreover, we didn't actually implement a real attack on these robots; it could be that the vulnerabilities we mentioned are not existing in some robots also because of a possible mismatch between the design of the system and its actual implementation. On the one hand, the attacks can be less deadly and dangerous than we have described, but on the other hand, the opposite is also true and attacks can be even more dangerous than we described.

Related Work Cyberbiosecurity is proposed as a new discipline at the interface of cybersecurity, CPS, and biosecurity to help safeguard the bioeconomy [5]. The first paper on cyberbiosecurity was focused on biotechnology and its security concerns [6]; it explained that biotechnology workflows are cyber-physical processes and illustrated with biomanufacturing process [7]. map the cyberbiosecurity landscape, in biotechnology and digitization of traditional technology. They explained about biosecurity on automation processes similar to us and cyberbiosecurity on artificial intelligence (AI) techniques across the biology sector [8]. illustrate malicious DNA injection performed by a remote cyber-criminal in DNA synthesis process and offer some mitigations. Protecting US food and agricultural system is reviewed [10], the cyberbiosecurity concepts from food production to the end user are explored, challenges are described, and solutions to integrate cyberbiosecurity in food and agricultural sectors are recommended. According to [12], cyberbiosecurity risks are difficult to characterize due to their diversity in types of threats, targets, and potential impacts.

Future Work The prevalence of attack vectors in wet lab automation frameworks suggests that we cannot rely only on perimeter protection and standard security controls. In order to continue providing the flexibility and power of custom design of biological protocols, future wet lab automation frameworks should be robust against subversion of their components.

Future research is required to illustrate the dangers of cyber-attacks on biological protocols and offer better protections of the wet lab automation systems: (i) identifying vulnerable protocol whose results can be manipulated without alerting the biologist, (ii) investigating process signing approaches for biological protocols, and (iii) designing adversary resilient distributed wet lab automation systems where every component ensures the correct operation of other components.

Acknowledgments This study was partially supported by the Cyber Security Research Center at the Ben-Gurion University of the Negev. All images in this chapter, except Fig. 7, are used under open-source licenses of their respective owners. Figure 7 is reproduced with permission of its owner.

References

1. F. Barthels, U. Barthels, M. Schwickert, T. Schirmeister, Findus: An open-source 3d printable liquid-handling workstation for laboratory automation in life sciences. SLAS TECHNOLOGY: Translating Life Sciences Innovation **25**(2), 190–199 (2020). https://doi.org/10.1177/2472630319877374. PMID: 31540570
2. Opentrons ot-2 – opentrons open source lab automation. https://opentrons.com/ot-2
3. Strateos – cloud-base lab automation solution. https://strateos.com/strateos-control-our-lab/
4. Automata labs – cloud-base lab automation for life sciences. https://automata.tech/products/automata-labs/
5. R.S. Murch, W.K. So, W.G. Buchholz, S. Raman, J. Peccoud, Cyberbiosecurity: An emerging new discipline to help safeguard the bioeconomy. Front. Bioeng. Biotechnol. **39** (2018)
6. J. Peccoud, J.E. Gallegos, R. Murch, W.G. Buchholz, S. Raman, Cyberbiosecurity: From naive trust to risk awareness. Trends Biotechnol. **36**(1), 4–7 (2018)
7. L.C. Richardson, N.D. Connell, S.M. Lewis, E. Pauwels, R.S. Murch, Cyberbiosecurity: A call for cooperation in a new threat landscape. Front. Bioeng. Biotechnol. **7**, 99 (2019)
8. R. Puzis, D. Farbiash, O. Brodt, Y. Elovici, D. Greenbaum, Increased cyber-biosecurity for DNA synthesis. Nat. Biotechnol. **38**(12), 1379–1381 (2020)
9. D.S. Schabacker, L.-A. Levy, N.J. Evans, J.M. Fowler, E.A. Dickey, Assessing cyberbiosecurity vulnerabilities and infrastructure resilience. Front. Bioeng. Biotechnol. **7**, 61 (2019)
10. S.E. Duncan, R. Reinhard, R.C. Williams, F. Ramsey, W. Thomason, K. Lee, N. Dudek, S. Mostaghimi, E. Colbert, R. Murch, Cyberbiosecurity: A new perspective on protecting us food and agricultural system. Front. Bioeng. Biotechnol. **7**, 63 (2019)
11. G. David, I.I.I. Schmale, A.P. Ault, W. Saad, D.T. Scott, J.A. Westrick, Perspectives on harmful algal blooms (habs) and the cyberbiosecurity of freshwater systems. Front. Bioeng. Biotechnol. **128** (2019)
12. K. Millett, E. Dos Santos, P.D. Millett, Cyberbiosecurity risk perceptions in the biotech sector. Front. Bioeng. Biotechnol. **7**, 136 (2019)
13. Blockly – client-side library for the programming language javascript for creating block-based visual programming languages and editors. https://developers.google.com/blockly
14. A. Faiña, B. Nejati, K. Stoy, Evobot: An open-source, modular, liquid handling robot for scientific experiments. Appl. Sci. **10**(3) (2020). https://doi.org/10.3390/app10030814. ISSN 2076-3417. https://www.mdpi.com/2076-3417/10/3/814

15. G. Gome, J. Waksberg, A. Grishko, I.Y. Wald, O. Zuckerman, Openlh: Open liquid-handling system for creative experimentation with biology, in *Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction*, (2019), pp. 55–64. https://doi.org/10.1145/3294109.3295619
16. Ot-2 python protocol api version 2. https://docs.opentrons.com/v2/
17. Strateos developer center. https://developers.strateos.com/docs
18. uarm developer – python library for uarm software. https://github.com/uArm-Developer/pyuarm
19. Evobot developer – software for the evobot robot. https://bitbucket.org/afaina/evobliss-software/src/master/
20. Findus developer- an open-source 3d printable liquid-handling workstation for laboratory automation in life sciences. https://github.com/FBarthels/FINDUS
21. Idc milab openlh – open source liquid handling system. https://github.com/idc-milab/openlh
22. Opentrons open source – source code for the opentrons api and ot app. https://github.com/Opentrons
23. J. Rader, K. Watson, Affordable covid-19 testing automation with the opentrons ot-2
24. Fernando L'azaro-Perona, Carlos Rodriguez-Antol'ın, Marina AlguacilGuill'en, Almudena Guti'errez-Arroyo, Jesu's Mingorance, Julio Garc'ıaRodriguez, and SARS-CoV-2 Working Group, Evaluation of two automated low-cost rna extraction protocols for sars-cov-2 detection. PLoS One **16**(2), e0246302 (2021)
25. Jos'e Luis Villanueva-Can˜as, Eva Gonzalez-Roca, Aitor Gastaminza Unanue, Esther Titos, Miguel Juli'an Mart'ınez Yoldi, Andrea Vergara G'omez, and Joan Anton Puig-Butill'e, Implementation of an open-source robotic platform for sars-cov-2 testing by real-time rt-pcr. PLoS One **16**(7), e0252509 (2021)
26. Robert P Goldman, Puja Trivedi, Daniel Bryce, Matthew DeHaven, Alex Plotnick, Peter L Lee, Joshua Nowak, Vanessa M Biggers, Trissha R Higa, and Jeremy P Hunt. A bayesian model for experiment choice in synthetic biology
27. J. Hayes, Technology laboratory automation: Labs go auto. Engineering & Technology **16**(7), 58–60 (2021)
28. Autoprotocol – language for specifying experimental protocols. https://autoprotocol.org/
29. A. Chattopadhyay, A. Prakash, M. Shafique, Secure cyber-physical systems: Current trends, tools and open research problems, in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, (IEEE, 2017), pp. 1104–1109
30. C. Cheh, A. Fawaz, M.A. Noureddine, B. Chen, W.G. Temple, W.H. Sanders, Determining tolerable attack surfaces that preserves safety of cyber-physical systems, in *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*, (IEEE, 2018), pp. 125–134
31. D. Antonioli, N.O. Tippenhauer, Minicps: A toolkit for security research on cps networks, in *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or Privacy*, (2015), pp. 91–100
32. Restrictedpython – provide a program input into a trusted environment. https://pypi.org/project/RestrictedPython/
33. C. Alberca, S. Pastrana, G. Suarez-Tangil, P. Palmieri, Security analysis and exploitation of arduino devices in the internet of things, in *Proceedings of the ACM International Conference on Computing Frontiers*, (2016), pp. 437–442. https://doi.org/10.1145/2903150.2911708
34. M.J. Joyner, N. Paneth, Seven questions for personalized medicine. JAMA **314**(10), 999–1000 (2015)
35. K. Gorshkov, C.Z. Chen, R.E. Marshall, N. Mihatov, Y. Choi, D.-T. Nguyen, N. Southall, K.G. Chen, J.K. Park, W. Zheng, Advancing precision medicine with personalized drug screening. Drug Discov. Today **24**(1), 272–278 (2019)
36. M.M. Soldevilla, D. Meraviglia-Crivelli, A.P. de Caso, Menon, and Fernando Pastor., Aptamer-irnas as therapeutics for cancer treatment. Pharmaceuticals **11(4):108** (2018)

37. M.S. Nabavinia, A. Gholoobi, F. Charbgoo, M. Nabavinia, M. Ramezani, K. Abnous, Anti-muc1 aptamer: A potential opportunity for cancer treatment. Med. Res. Rev. **37**(6), 1518–1539 (2017)
38. S. Kosuri, G.M. Church, Large-scale de novo dna synthesis: Technologies and applications. Nat. Methods **11**(5), 499–507 (2014)
39. S. Faezi, S.R. Chhetri, A.V. Malawade, J.C. Chaput, W. Grover, P. Brisk, M.A. Al Faruque, Oligo-snoop: A non-invasive side channel attack against DNA synthesis machines, in *Network and Distributed Systems Security (NDSS) Symposium 2019*, vol. 2019,
40. F. de Jesus, D.G. Cortez, D. Tandel, P.V. Robinson, D. Seftel, D.M. Wilson, D.M. Maahs, B.A. Buckingham, K.W.P. Miller, C.-t. Tsai, Automation of a multiplex agglutination-pcr (adap) type 1 diabetes (t1d) assay for the rapid analysis of islet autoantibodies. SLAS Technology (2021)
41. The opentrons covid-19 testing system. https://blog.opentrons.com/how-to-use-opentrons-to-test-for-covid-19/
42. J. Bartlett, D. Stirling, A short history of the polymerase chain reaction, in *PCR protocols*, (Springer, 2003), pp. 3–6
43. J. Salm, O. Sefiha, Restorative justice in sports: Does restorative justice have a place in anti-doping governance? Sport in Society, 1–16 (2021)
44. Emelie Ohnstedt, Hava Lofton Tomenius, Peter Frank, Stefan Roos, ̈E. V°agesj ̈o, and Mia Phillipson. Accelerated wound healing in minipigs by on-site production and delivery of cxcl12 by transformed lactic acid bacteria. Pharmaceutics, 14(2):229, 2022
45. Chloe Kent, Drug dispensing goes digital. https://www.pharmaceutical-technology.com/features/robotic-drug-dispensing-digital-pharmacy/