



Ελληνικό Μεσογειακό Πανεπιστήμιο

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πρόγραμμα Σπουδών Μηχανικών Πληροφορικής ΤΕ

Τίτλος:

Αυτοματοποίηση της διαχείρισης ευπαθειών λογισμικού με χρήση τεχνικών Robotic Process Automation.

Title:

Enabling Automatic Vulnerability Management Through a Robotic Process Automation Framework

Παπαχατζάκης Νικόλαος (τπ4294)

Επιβλέπων εκπαιδευτικός : Μαρκάκης Ευάγγελος

Επιτροπή Αξιολόγησης :

- **Μαρκάκης Ευάγγελος**
- **Παναγιωτάκης Σπυρίδων**
- **Στρατάκης Δημήτριος**

Ημερομηνία παρουσίασης: Δευτέρα 04 Ιουλίου 2022

Ευχαριστίες

Δεν θα μπορούσα να μην να εκφράσω τις θερμές μου ευχαριστίες στον καθηγητή μου Δρ. Ευάγγελο Μαρκάκη, καθώς και στους απόφοιτους μεταπτυχιακού Κεφαλούκο Ιωάννη και Αστουρακάκη Νικόλαο, για την πολύτιμη βοήθεια τους στην εκπόνηση και στη βελτίωση της πτυχιακής μου εργασίας, καθώς και στο εργαστήριο Pasirhae Lab για την υπέροχη συνεργασία και την γενναιόδωρη βοήθεια σε οτιδήποτε με απασχόλησε. Η βοήθειά τους, μα κυρίως η στάση τους, πέρα από ένα καλό αποτέλεσμα, με έκανε να καταλάβω πόσο πολύ μου αρέσει το αντικείμενο και το πόσο σημαντικό είναι να έχεις ανθρώπους που να σε στηρίζουν και να πιστεύουν σε εσένα.

Abstract

Internet these days has shaped our society in such a way that more and more things we do in our everyday lives tend to digitalization. The benefits behind that digitalization trend are that it makes our lives easier and more productive, especially with today's computing power and technological advancement where almost everything is possible to do online. To extend the functionality and usability of the current technology, an enormous number of new services are created every day. Considering that, every new service published, has the potential to become a target from malicious software and users as they might possess security vulnerabilities that are not yet identified or mitigated. The process of making an organization secure from such risk has become harder and more time-consuming than ever because of the eruption of new services, exposing their interfaces to the internet. For these reasons we propose an application that enables administrators to automate their way into managing software vulnerabilities in organizations, using Robotic Process Automation (RPA) technics. Our tool has all the functionality needed to help administrators automate the mitigation actions required for known vulnerabilities by recording the mitigation actions of the admin into a process, then our tool distributes robots that executes the recorded actions to multiple hosts, relieving administrators from such time-consuming tasks.

Περίληψη

Το Διαδίκτυο στις μέρες μας, έχει διαμορφώσει την κοινωνία μας με τέτοιο τρόπο που όλο και περισσότερα πράγματα που κάνουμε στην καθημερινή μας ζωή τείνουν προς την ψηφιοποίηση. Τα οφέλη πίσω από αυτήν την τάση ψηφιοποίησης είναι ότι κάνει τη ζωή μας ευκολότερη και πιο παραγωγική, ειδικά με τη σημερινή υπολογιστική ισχύ και την πρόοδο της τεχνολογίας όπου σχεδόν τα πάντα είναι δυνατά να γίνουν online. Με σκοπό την πρόοδο της λειτουργικότητας και της χρηστικότητας της τεχνολογίας, ο αριθμός των νέων υπηρεσιών που δημιουργείται καθημερινά είναι τεράστιος. Κάθε νέα υπηρεσία που δημοσιεύεται, είναι πιθανόν να γίνει στόχος από κακόβουλο λογισμικό, καθώς ενδέχεται να διαθέτει ευπάθειες ασφαλείας που δεν έχουν ακόμη εντοπιστεί. Η διαδικασία διασφάλισης ενός οργανισμού από τέτοιους κινδύνους έχει γίνει πιο δύσκολη και πιο χρονοβόρα από ποτέ λόγω αυτής της έκρηξης νέων υπηρεσιών. Για αυτούς τους λόγους, προτείνουμε μια εφαρμογή που επιτρέπει στους διαχειριστές να αυτοματοποιήσουν τον τρόπο διαχείρισης των ευπαθειών λογισμικού σε οργανισμούς και όχι μόνο, χρησιμοποιώντας τεχνικές Robotic Process Automation (RPA). Το εργαλείο μας έχει όλη τη λειτουργικότητα που απαιτείται από τους διαχειριστές για να αυτοματοποιήσουν τις ενέργειες που απαιτούνται για να αντιμετωπιστούν τα γνωστά τρωτά σημεία, καταγράφοντας τις ενέργειες τους σε μια διεργασία, όπου έπειτα διανέμονται και εκτελούνται οι ενέργειες αυτές σε πολλούς υπολογιστές, απαλλάσσοντας τους διαχειριστές από τέτοιες χρονοβόρες εργασίες.

Table of Contents

Ευχαριστίες	2
Abstract	3
Περίληψη	4
Table of Contents	5
List of Figures	7
List of Tables	8
1. Introduction	9
2. State of the Art	10
3. Technology Enablers	12
3.1. Robotic Process Automation	12
3.2. Microsoft UIA framework	12
3.3. Visual Studio	12
3.4. NMAP	13
3.5. Visual C#	14
3.6. PyCharm	14
3.7. Flask	14
3.8. JSON Web Token	15
3.9. PostgreSQL	15
3.10. Docker	15
3.11. Windows Forms	16
4. Implementation	17
4.1. Architecture	17
4.2. Server-side Logic (Backend)	17
4.2.1. Database	17
4.2.2. Server API	19
4.2.3. Endpoints	19
4.3. Client-side Logic (Frontend)	20
4.3.1. RPA Assistant	20
4.3.2. Desktop application	20
Sign in	20
Sign up	21
Assistant	21
Dashboard	22
Robots	23
Processes	24

Optimizations.....	35
Client API.....	36
Robots.....	36
5. Use-case Scenario.....	41
5.1. Methodology	41
5.2. Real-world application.....	43
6. Conclusion	45
7. References.....	46

List of Figures

Figure 1 Visual studio Community edition environment.....	13
Figure 2 Nmap Logo.....	13
Figure 3 Python logo.....	14
Figure 4 Flask logo.....	14
Figure 5 PostgreSQL Logo.....	15
Figure 6 Docker Logo.....	16
Figure 7 WinForms architecture.....	16
Figure 8 Architecture diagram.....	17
Figure 9 The database Schema.....	18
Figure 10 Sign in page.....	20
Figure 11 Sign up page.....	21
Figure 12 RPA Assistant Welcome page.....	22
Figure 13 Dashboard.....	23
Figure 14 Robots Page.....	23
Figure 15 New robot dialog.....	24
Figure 16 Processes page.....	24
Figure 17 Recording toolbox dialog.....	26
Figure 18 An example process.....	27
Figure 19 All the available tasks.....	29
Figure 20 Task states.....	29
Figure 21 Targets page overview.....	30
Figure 22 Discover host dialog.....	30
Figure 23 Vulnerability scan dialog.....	31
Figure 24 Targets board - Overview tab.....	32
Figure 25 Vulnerability Right click Context menu.....	32
Figure 26 Targets - Discovered vulnerabilities tab.....	32
Figure 27 Targets board - Timeline tab.....	33
Figure 28 Settings page.....	34
Figure 29 Profile page.....	35
Figure 30 Mouse click recording sequence.....	37
Figure 31 Playback robot flowchart.....	38
Figure 32 Example robot token data.....	39
Figure 33 Robot warning dialog - Postpone execution.....	40
Figure 34 Use-case diagram of Vulnerability management scenario.....	42
Figure 35 Sequence diagram of Vulnerability management scenario.....	42
Figure 36 Remediation process for XAMPP.....	43
Figure 37 Before remediation - Discovered vulnerabilities.....	44
Figure 38 After remediation - Discovered vulnerabilities.....	44

List of Tables

Table 1 API endpoints	19
-----------------------------	----

1. Introduction

As technology progresses and becomes more affordable, even more people get their hands on powerful and capable pieces of technology. Provided that the technology type of choice these days is the portable devices [1], i.e. (smartphones, tablets, and laptops), suggests that our information should be available from anywhere in the world. To accommodate that need, organizations focus on the development of web applications and distributed services. Although, some people may use the internet maliciously to benefit from others by stealing their private information. That is possible due to the huge number of bugs in the published apps and software vulnerabilities of web services that an attacker could exploit. As presented in [2], the number of discovered vulnerabilities keeps rising year by year. That is why we need to make sure that the apps, the services, and the organization's infrastructure are not vulnerable and secure from malicious users.

To have an acceptable level of security in an organization, most software vulnerabilities must be completely assessed and mitigated. Remediating all the possible vulnerabilities is a tedious and time-consuming process for the operators because software vulnerabilities may exist in any software an organization utilizes, whether it is the OS (Operating System), the web server, or even the smart lamp in the office. A typical organization usually consists of several employees that use computers with internet access, which exposes even more vulnerabilities and widens the attack surface of the organization, providing attackers with points of entry into the system. Thinking of all that, network administrators need help to stay fast and effective against cybercriminals. In this thesis, we propose an implementation of a Windows application made with network administrators in mind. [3] This application is essentially a Robotic Process Automation (RPA) tool that gives the administrators a way to automate the vulnerability mitigation process by allowing them to record the remediating actions needed in a remediating process or search for an already created process from other users of the platform. Using this process, robots can be deployed on multiple machines simultaneously and replicate the administrators' actions by executing the recorded process. Finally, operators can use the applications vulnerability scanning tools to rescan the targets for vulnerabilities and evaluate the effectiveness of the mitigation.

The rest of the paper is structured as follows. Section II presents the state of the art regarding automated vulnerability assessment and mitigation. Section III presents the technology enablers. Section IV presents the implementation details (System Architecture)

of the presented platform. Section V presents a Use-case scenario. Finally, Section VI concludes this paper with a short discussion on the results of this thesis and the presentation of foreseen future steps.

2. State of the Art

Vulnerability Management in an enterprise environment is a very critical and time-consuming task for the operators, because of the huge number of vulnerabilities discovered every day and the need to manage each of them for every machine in the infrastructure. Automations can be used to assist operators to free up valuable time that they spend on such repetitive tasks. Thus, in this section, we will analyze existing solutions for vulnerability assessment and remediation automations.

There have been several studies about Automated Software Vulnerability patching. Authors in [4] proposed a methodology for automated patching of known vulnerabilities in binary programs, using code from a non-defective version of the software and using this to patch the binary. Similarly, authors in [5] proposed an automatic patching technique, which overwrites the Global Offsets Table (GOT) and Procedure Linkage Table (PLT) of the vulnerable functions in a binary program. The authors also performed fuzzing and symbolic execution on binary program files to discover possible unknown vulnerabilities.

Moreover, the work in [6] studies a dynamic risk-aware patch scheduling mechanism to determine the order with which vulnerabilities should be patched, thus minimizing the security risks imposed by vulnerabilities. Similarly, the work in [7] proposes a machine-learning based automation framework to automate remediation decision analysis for electric utilities. Even though the presented solution was investigated within an organization selling electric utilities, it can also be applied to different networking environments as the authors claim. In this context, the authors in [8] proposed a machine-learning based solution for prioritizing vulnerability remediation actions by classifying vulnerabilities according to high- and low-risk, where high-risk vulnerabilities are those that have been exploited in actual organization networks. Moreover, the work in [9] verifies and tests any possible vulnerabilities by automatically pen-testing the system based on generated attack graphs.

Regarding automated vulnerability assessment, authors in [10] describe a vulnerability assessment framework that could be used to assess cyber threats on computer networks, specifically on Cyber-Physical Systems (CPS) and Aviation cyber-physical systems (ACPS). Additionally, the work in [11], [12] presents a machine learning model to extract vulnerability characteristics using the vulnerabilities' description, optimizing the vulnerability assessment on software. The presented machine-learning model utilizes a combination of character-level and word-level features to improve effectiveness.

Finally, there have been numerous research initiatives on remote, automated mitigation of vulnerabilities on computer networks. Nishant Sharma et al. in [13] proposed a proof of concept for remote automated vulnerability assessment and mitigation on a host-based solution, using custom vulnerability specific Python scripts. Similarly, authors in [14] presented ARMOUR, which mitigates the threat and secures the network by preventing the communication between vulnerable and critical assets, but it does not make any vulnerability remediation actions on the assets.

Most of the presented work does not propose a remote automated way of patching vulnerabilities, and even the work that does still requires operators to heavily interact with the remote machine and write custom scripts for every possible vulnerability. To address the above-

mentioned issue, we propose a framework that performs i) remote automated vulnerability assessment to multiple hosts and ii) remote and fully automated vulnerability patching to multiple hosts, utilizing RPA technics to minimize the time spent by network operators to manually patch common vulnerabilities; and shrink the attack surface of the network.

3. Technology Enablers

In this section, we will present all the technologies that we used and briefly explain the use and purpose of each one. All the technologies presented in this section are Open-Source and are available for academic purposes.

3.1. Robotic Process Automation

Robotic process automation is a technology that helps us create, deploy, and manage software robots. These robots often emulate human actions on computers, using the mouse and keyboard to interact with the user interfaces (UI) of various applications and systems. RPA robots can recognize what is on the user screen and even identify and extract data from the screen. It mainly executes a process of well-defined actions as a human would do.

3.2. Microsoft UIA framework

Microsoft UI Automation¹ is a framework developed by Microsoft² and is part of the .NET³ framework collection. UIA is a framework that it's mainly used for Accessibility applications because it provides programmatic access to the user interface (UI) and delivers information for what is on screen and can interact with the UI in more ways than just input from the user.

3.3. Visual Studio

Visual Studio⁴ is a complete Integrated Development Environment (IDE) developed by Microsoft that enables users to easily develop their applications with various programming languages like Visual C#, Visual C++, ASP.NET, and more.

¹ <https://docs.microsoft.com/en-us/dotnet/framework/ui-automation/>

² <https://www.microsoft.com/>

³ <https://docs.microsoft.com/en-us/dotnet/>

⁴ <https://visualstudio.microsoft.com/>

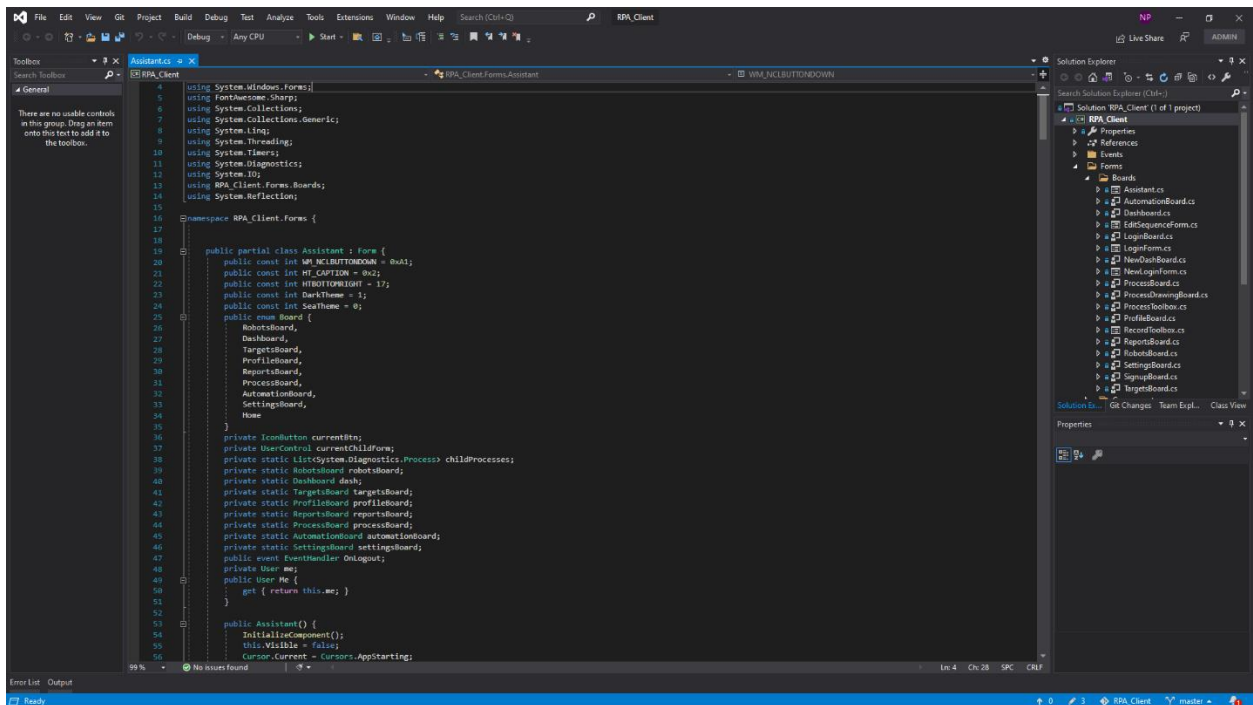


Figure 1 Visual studio Community edition environment

3.4.NMAP

NMAP⁵ is an open-source utility for network discovery and cybersecurity. It is very useful for network operators because of its easy and well-designed command-line interface (CLI) and because of the nicely structured data of the network that provides. NMAP also provides a scripting engine (NSE) that allows the expandability and scalability of the utility through custom scripts written in the LUA⁶ programming language

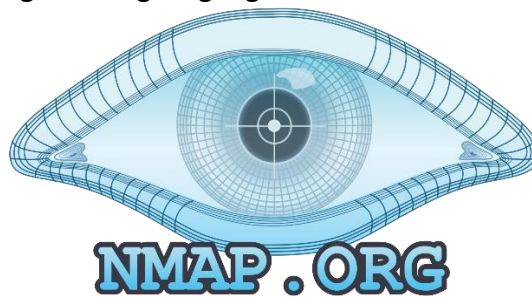


Figure 2 Nmap Logo

⁵ <https://nmap.org/>

⁶ <https://www.lua.org/>

3.5. Visual C#

Visual C# is a C-like programming language developed and maintained by Microsoft and is a basic part of the .NET framework. C# has a very flexible and scalable syntax and is quite easy to master. It supports declarative and non-declarative syntax and is very good for Graphical User Interface (GUI) applications.

3.6. Python

Python is an interpreted non-declarative programming language with the prime goal of easy, fast, and flexible software development. Python⁷ is cross-platform and open-source language and is one of the most famous and used programming languages.



Figure 3 Python logo

3.7. Flask

Flask⁸ is a lightweight Web Server Gateway Interface (WSGI) web application framework for Python which is designed for easy and fast web development. There are many extensions for Flask that extend its functionality and usability. In our research, we utilize flask as a REST API server.



Figure 4 Flask logo

⁷ <https://www.python.org/>

⁸ <https://palletsprojects.com/p/flask/>

3.8.JSON Web Token

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way of securely transmitting information structured in JSON format. JWT is a bearer authentication method and uses JSON data encoded as Hex string.

3.9.PostgreSQL

PostgreSQL⁹ is an open-source object-relational Database Management System (DBMS) compatible with Structured Query Language (SQL) that helps us manage and manipulate databases in a traditional fashion. It is popular for its robustness and superior performance.



Figure 5 PostgreSQL Logo

3.10. Docker

Docker¹⁰ is a software that enables containerization technology of Linux containers. Containers are stripped down Linux kernels that provide a separate environment with specific packages and drivers for our applications to run on independently. It supports a wide variety of custom containers, and it makes the deployment and the management of the containers easier.

⁹ <https://www.postgresql.org/>

¹⁰ <https://www.docker.com/>

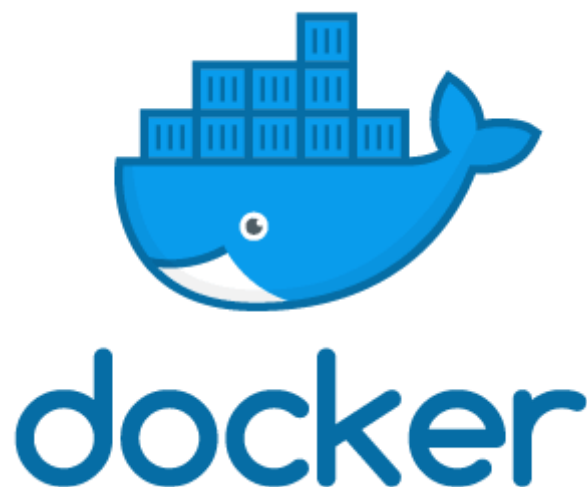


Figure 6 Docker Logo

3.11. Windows Forms

Windows Forms (WinForms) is a free and open-source Graphical User Interface (GUI) class library that is part of Microsoft .NET framework. WinForms makes the development of the GUI easy with all the of the shelf components and controls that offers. Although Windows Presentation Foundation (WPF) has come to replace WinForms, still lots of users prefer the old traditional way.

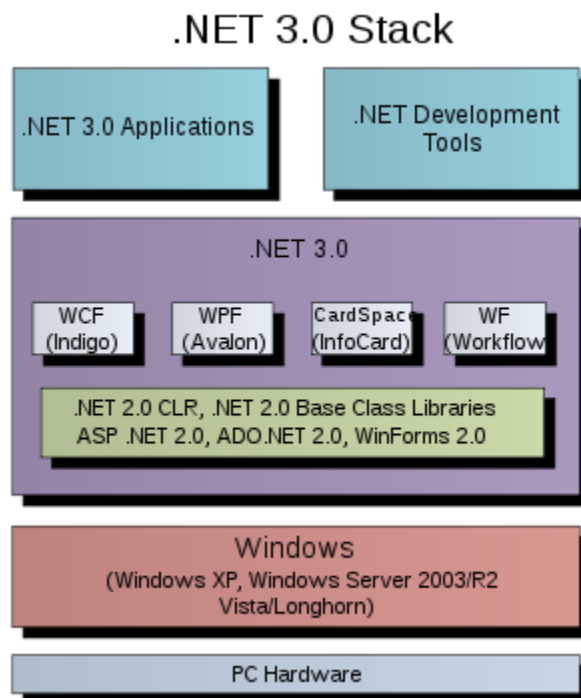


Figure 7 WinForms architecture

4. Implementation

4.1. Architecture

In this section, the architecture of the proposed framework will be presented. The application is designed to be cloud native therefore, the proposed architecture is split into two discrete parts, the server-side logic (backend) and the client-side logic (frontend).

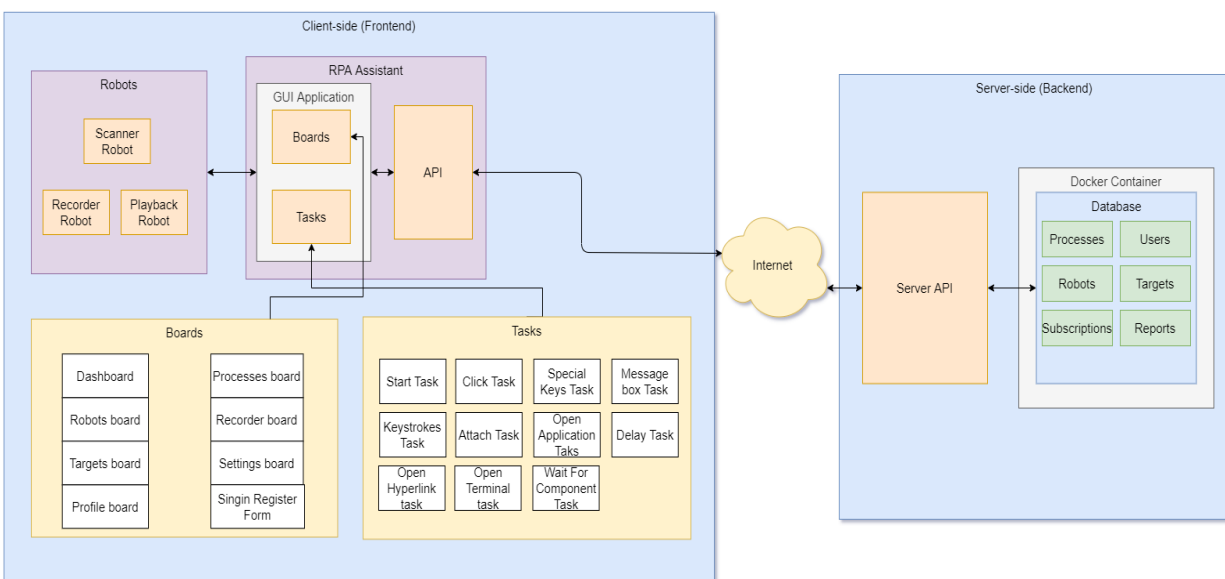


Figure 8 Architecture diagram

4.2. Server-side Logic (Backend)

4.2.1. Database

One of the key components of a cloud native application is the database, which is used to securely store all the user's data. In our implementation, we decided to use PostgreSQL¹¹ database management system (DBMS), since PostgreSQL supports JSON and XML datatypes which other commercial DBMS lack. To enhance scalability and load balancing, the database lives on Docker containers.

As seen in Figure 9, the database schema contains tables for users, processes, targets, reports, robots, and subscriptions. The table of users consists of fields for saving his ID in Universal Unique Identifier (UUID) format, first name, last name, email, password (as hashed SHA256¹² string), last login date, birthdate, and a subscription id that user have purchased. The fields of the processes table are the process UUID, the name and type of the process, creation date, its visibility (public or not), a description text, an array of tags for easier search, the list of actions that the robot will execute, the owner (User id) of the process, if it is a subprocess, and

¹¹ <https://www.postgresql.org/>

¹² <https://en.wikipedia.org/wiki/SHA-2>

the last modified date. The robots table consists of fields for robot UUID, the process id that it will execute, the name and the id of the target that it will be deployed, its status, last ran date, and the id of the user that created it. The targets table contains fields for saving the name of the target, its IP and MAC address, the subnet, the last scanned date, its state, vendor, the Operating system that is running, an array of discovered vulnerabilities, the id of the user that this machine belongs, SSH credentials and the ports that are open on the machine. The reports table also contains fields for the report id, the robot id that produced the report, the creation date, the status, and the data that builds the body of the report. Finally, the subscriptions table holds information about the available subscriptions the user can purchase. It has fields for the subscription name, the id, the price of the subscription, a description, and the purchase and expiration date.

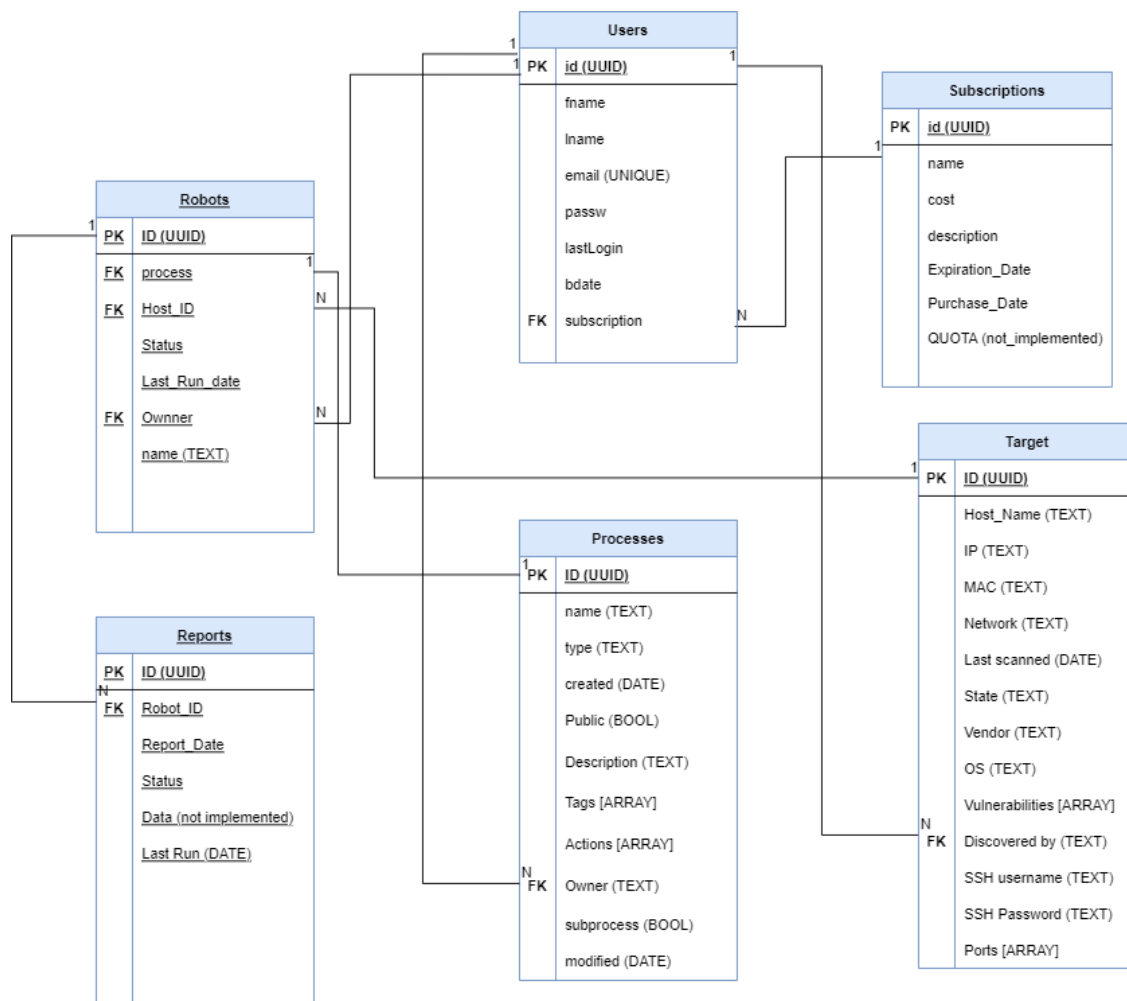


Figure 9 The database Schema

4.2.2. Server API

The server-side API is mainly a RESTful API written in Python using Flask and SQLAlchemy. We used the SQLAlchemy¹³ toolkit because it makes it easy to connect the DBMS with the Python by wrapping the complicated API of the DBMS into a user-friendly adapter and using models to interact with the database instead of SQL queries. The use of models makes the development easier and faster while keeping the code clean and providing the basic data type checking. The Flask web server exposes to the internet a series of endpoints where the frontend-logic and the robots can use to communicate. Flask was the framework of choice because it provides native support for JSON Web Token (JWT) and SQLAlchemy which makes the development process much quicker. The API is responsible for handling and processing all the user's data as well as the authentication of the user. If the user provides valid credentials, the API sends back to the client a bearer token (JWT token) that represents the user's access card. Using that token the API checks if the user is authorized to access the information, he requested for.

4.2.3. Endpoints

The following table shows the available endpoints of the server API.

URL	Method	Description
/api/v1/login	POST	Returns a Bearer token if credentials are valid
/api/v1/logout	GET	Logs out the user
/api/v1/signup	POST	Registers a new account with the posted data
/api/v1/process	GET/POST/PUT/DELETE	CRUD operations for processes.
/api/v1/scan/token	GET	Returns a special token for the scanner robot
/api/v1/users	GET/PUT/DELETE	Used to retrieve, update, and delete users account data
/api/v1/robots	GET/POST/PUT/DELETE	CRUD operations for the robots
/api/v1/robots/token	GET	Returns a Bearer token for the playback robot
/api/v1/robots/process	GET	Returns the process for the robot to execute
/api/v1/targets	GET/POST/PUT/DELETE	CRUD operations for the targets
/api/v1/subscriptions	GET/POST/DELETE	Used to create, retrieve, and delete subscriptions schemes
/api/v1/dashboard	GET	Returns users dashboard statistics
/api/v1/docs	GET	Shows API documentation

Table 1 API endpoints

¹³ <https://www.sqlalchemy.org/>

4.3. Client-side Logic (Frontend)

4.3.1. RPA Assistant

The assistant mainly consists of the main desktop application and the client API. We went with a traditional desktop application design because there was the need for things to happen on the user's hardware, such as network scanning and remote access. For the development of the assistant, we choose to use the Microsoft WinForms GUI class library and Microsoft Visual C# programming language that are under the .NET framework. The reason we settled with the older WinForms instead of newer technology is that it is supported by machines that run on Microsoft Windows, even older systems like Windows XP.

4.3.2. Desktop application

Having in mind that the application needed to be modular and easily expandable, we followed a container style approach on the assistant, which means that the main UI is just a container that fills up with custom boards and a side menu to navigate between them.

Sign in

Aside from the assistant's main form, we created a sign-in page, which is the first thing to appear when the application is started. On the sign-in page, the users must fill their login credentials, such as email and password. If the users do not remember their credentials, they can reset the password for this email using the link next to the Sign in button. Because the users may log in multiple times during a working day, we added a "Remember me" checkbox for the application to remember the last used credentials.

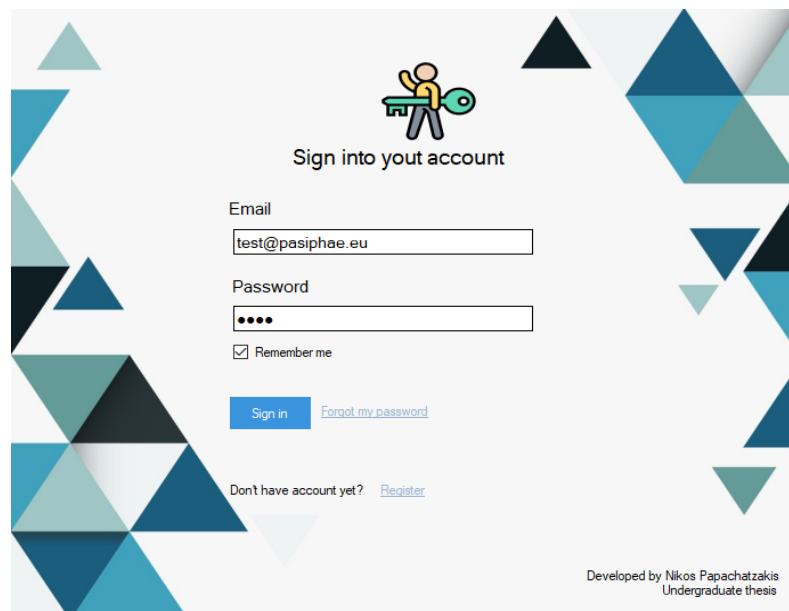
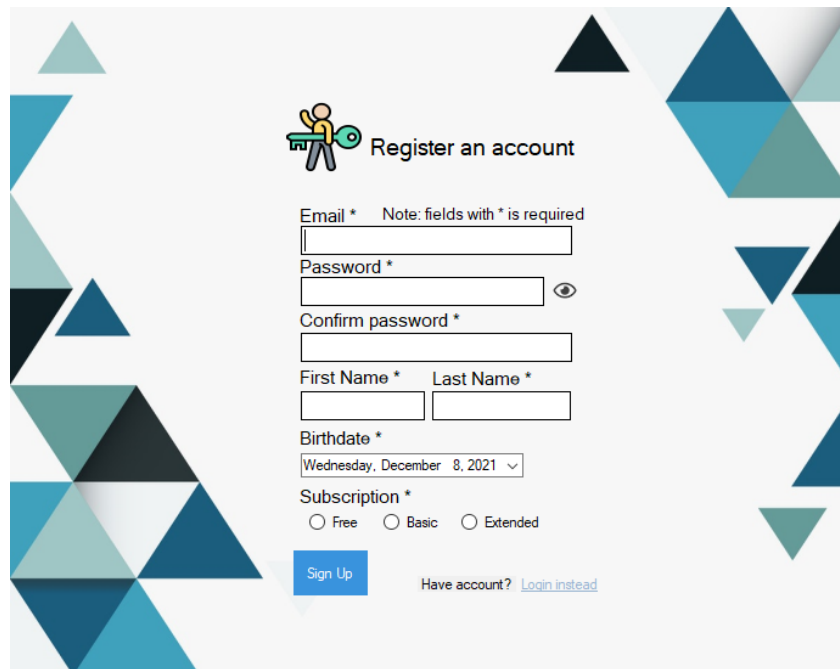


Figure 10 Sign in page

Sign up

On the Sign-in page, clicking the “Register” link reveals the Sign-up page. On this page, the user can create a new account by filling in all the necessary information indicated with the star “*” character. The subscription that the user selects, changes the monthly cost of the application, and changes the functionality of the application. Although, in the scope of the thesis, the subscription mechanism was not implemented.



Register an account

Email * Note: fields with * is required

Password *

Confirm password *

First Name * Last Name *

Birthdate *
Wednesday, December 8, 2021

Subscription *
 Free Basic Extended

[Sign Up](#) [Have account? Login instead](#)

Figure 11 Sign up page

Assistant

After the user successfully login, the main form of the application starts. As seen in Figure_12 the application consists of the container space (marked with gray color) where all the content boards are displayed and the top panel where we find the title of the active board, the profile button, and the window controls. Moving on, on the left side we find the navigation side panel with all the available boards. By clicking on any item in the navigation side panel, the specific board will be displayed in the center of the application. Lastly, at the bottom, the status bar with a progress bar on the right is depicted. This progress bar indicates the progress of the robot deployment. Right next to the progress bar, we find a button to resize the whole window. For aesthetic purposes, the application lacks the default window border, therefore, making the resize button mandatory.

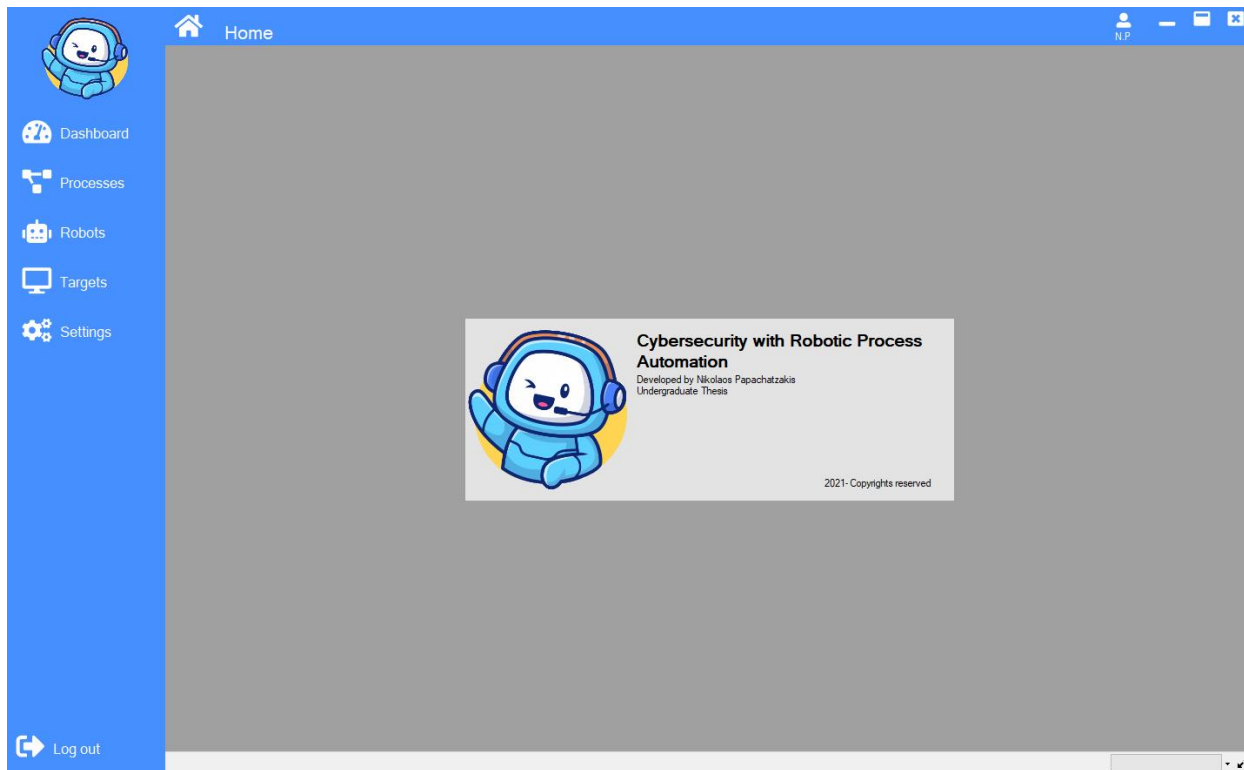


Figure 12 RPA Assistant Welcome page

Dashboard

The dashboard is the page that shows to the user various information about their activity on the platform. Some of this information are the processes that the user has made, the robots, the discovered targets, reports, robot status, and the sum of the discovered vulnerabilities.

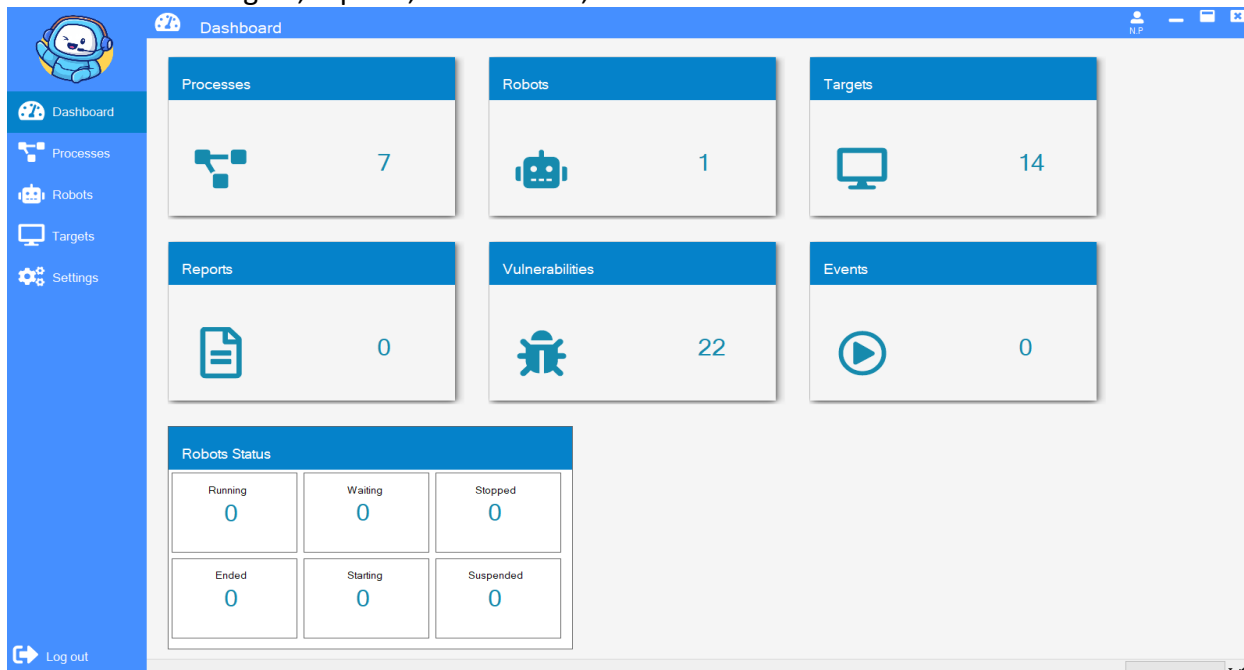


Figure 13 Dashboard

Robots

Robot's page main purpose is for listing and handling the user's robots. As seen in [Figure 14](#), in the center of the page, there is a list of all the robots of the user. The information that is saved for every robot such as the name, the IP address of the machine to deploy on, its status of execution (started, ended, running, etc.), the last run date, the user who created it, the process name to be executed and the UUID of the robot.

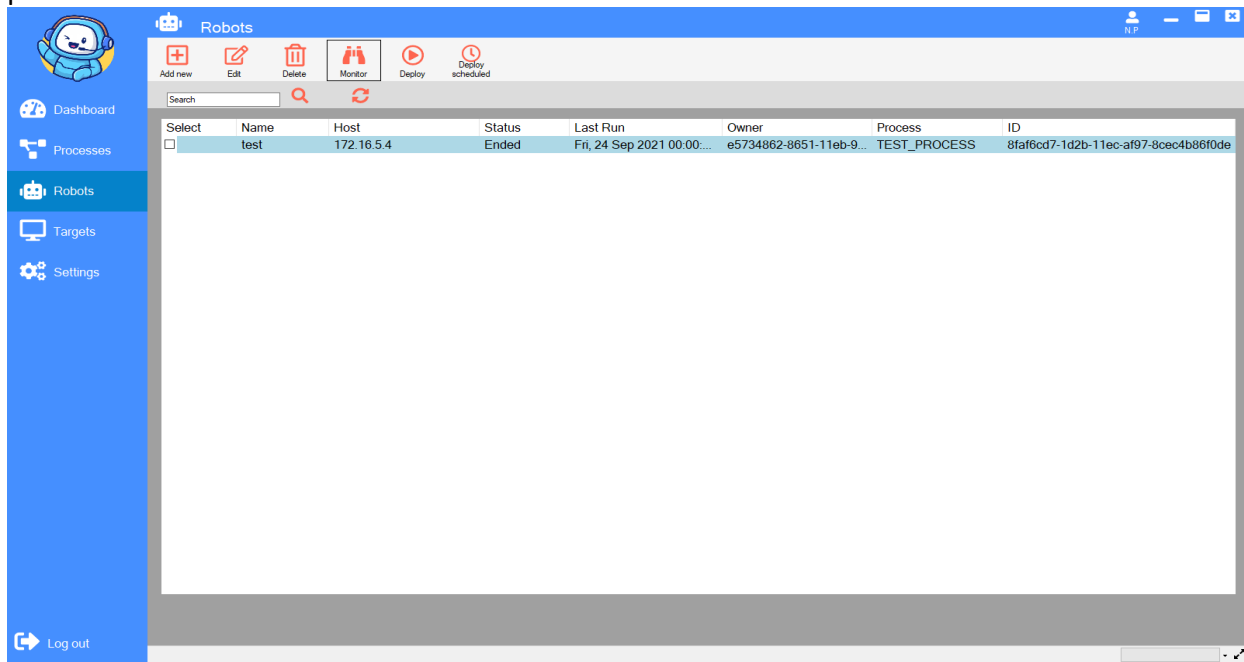


Figure 14 Robots Page

At the top of the page is the menu with all the available actions. The “add new” button, when clicked, will show a dialog window prompting the user to insert the name of the new robot, the IP address of the machine to deploy on, and the process to execute. Similarly, the “edit” button shows a dialog with the robots’ details already filled for the user to change. The “delete” button deletes all the selected robots. The “deploy” button deploys all the selected robots from the list on the specified machines for each robot. Lastly, the “deploy scheduled” also deploys the selected robots but gives the robots a specified starting time and date. Finally, right below the main menu is a field where the user can search for a robot using its name.

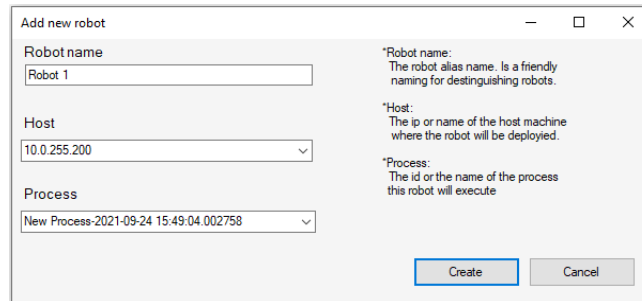


Figure 15 New robot dialog

Processes

The processes board provides the ability to the user to create, edit and delete the processes. The page consists of the following UI components.

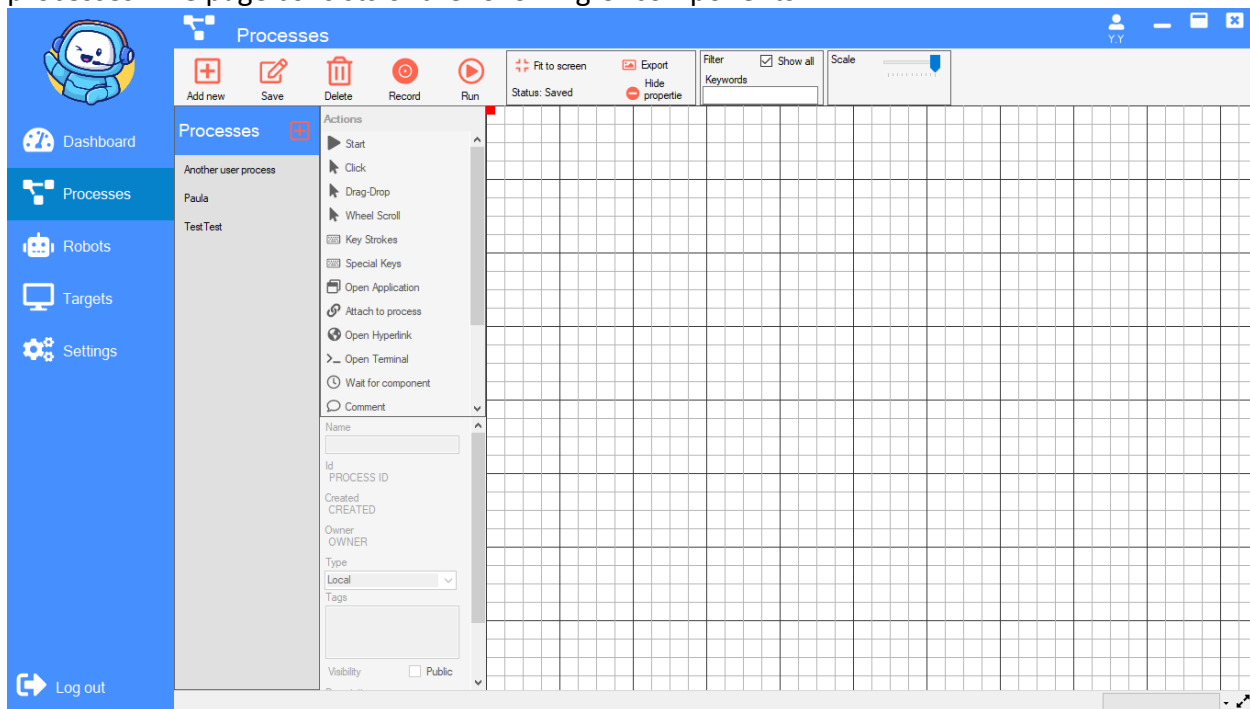


Figure 16 Processes page

Actions menu

As seen on most pages, on the top of the page is a menu bar containing all the available functionality of the specific page. The process page is no exception to that, having a menu with buttons to create, save, delete, record, and run the selected process. Going on, we find a status label that describes if the process is saved or not, a Fit to screen button that resets all the tasks of the process to the origin¹⁴, the Export button, which exports the current process to a PNG format image for easy sharing, and the Hide properties button that minimize-maximize the Task

¹⁴ Origin is the point where the (x, y) coordinates are (0, 0) respectively. The origin point is presented on the drawing board with a small red square.

list and process details panels to extend the usable area of the drawing board. On the right of all these, there is a field that is used to filter the processes on the process list by their name, a checkbox that when checked the process list contains all the public processes across our platform and if not, then only the users' processes will be displayed. Finally, there is a slider that is used to scale the process to the desired size.

Process List

The process list is the left most panel of the process page, and its purpose is to display the available processes ordered by their name in ascending order. Users can use this list to select the process they want to edit. By right clicking on an item of the list, a context menu appears which gives us the ability to clone the specific process, delete or edit the process.

Tasks List

The tasks list is located on the right of the process list and there is a list of all the available tasks the user can insert into a process. It helps the user to manually edit the process if needed. By clicking on a task, it automatically inserts a task of that kind on the drawing board.

Process details

Right below the tasks list, we can find the process details panel. This panel is responsible for displaying all the details of the current process. Such details are the process's name, id, creation date, owner, type, tags, visibility, and its description. Some of this information use a text field to display because we wanted the user to be able to edit these details of the process. The type of the process defines where this process was recorded. The tags of the process are used to make the search of the process easier and to provide some information regarding the content of the process. The visibility of the process is to determine if other users on the platform can view and use this process.

Process Recording toolbox

When the user decides to record a process, by clicking the "Record" button on the Actions menu, the assistant gets minimized and the recording toolbox dialog shows up. This dialog gives the user the time needed to prepare for the recording while providing a variety of different recording modes. As seen in [Figure 17](#), the toolbox consists of:

- "Start recording" button starts recording any action made by the user.
- "Click" button records a single mouse click action.
- "Keystrokes" button records the user's keystrokes.
- "Special key" button records a special key combination (i.e., CTRL + SHIFT + V keys).
- "Command window" button records a terminal session.
- "Save" button saves the recordings and terminates the procedure.

Pressing any of the above-mentioned buttons, the toolbox gets minimized, allowing the user to perform the remediating actions. By pressing the Escape (ESC) button, the user can stop the recording and show the toolbox.

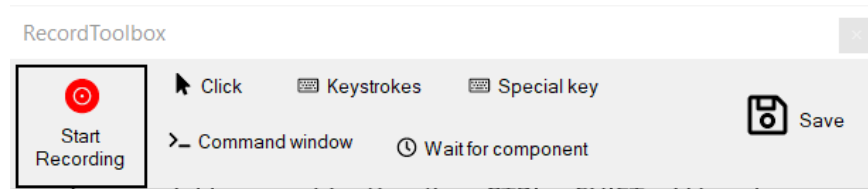


Figure 17 Recording toolbox dialog

Process Drawing board

Finally, the most important part of the processes page is the process drawing board. Using this component, the user can design and view a process. A process is represented as a directed graph in which each node is a task of the process, and each edge, points to the next task to execute in the process. By design, the tasks can have multiple inward and outward connections but, in our implementation, we do not take advantage of that feature yet.

Moving on to the process design, the user can insert a task into the drawing board manually or by recording the user's actions. The recorded processes are complete and ready to execute. That is not the case when the user manually inserts tasks into the process. To be able to execute a task, it must be connected to a start task, or a sequence of tasks starting with the start task. After the successful insert of a recorded process, the user can edit the process manually using the controls described below.

- Holding the right mouse button down and moving the mouse, moves the whole board around.
- Holding the left mouse button down and mouse move on the board reveals a dotted rectangle where every task that interacts with this rectangle gets selected. The selected tasks are lighter in color and feature a blue border.
- Holding the left mouse button down and moving the mouse on a task, moves the selected tasks on the board.
- A single left click on the board deselects everything.
- A single left click on a task adds it to the selection while deselecting everything else.
- Holding the CTRL key while single clicking on a task adds it to the selection.
- Holding the CTRL key and left mouse button while moving the mouse, creates a grouping rectangle. Every task that intersects the rectangle, becomes a child of it. The grouping rectangles are meant for grouping tasks with similar functionality. They have a title, size, and color properties that by right clicking on it, the user can change their values through the context menu.
- Holding the left mouse button down while moving the mouse on a grouping rectangle, moves the rectangle and its tasks around on the board.
- DEL key deletes everything selected on the drawing board.
- CTRL + S keys combination saves the process
- CTRL + C keys combination copies the selected items on the drawing board

- CTRL + V keys combination pastes the previously copied tasks, on the board

There are two ways to insert a task manually. One of these is to insert a task through the task list. The second way to insert a task is while connecting to another task. To connect the tasks, the user needs to click on a green plus (+) button at the bottom center of the task. When this button gets clicked, an arrow appears on the board indicating that the connection is taking place. The user then needs to click on another task to connect them. If the user does not have a task to connect on, by clicking on the board, a context menu appears, and the user has the option to add a task on that spot and connect to it.

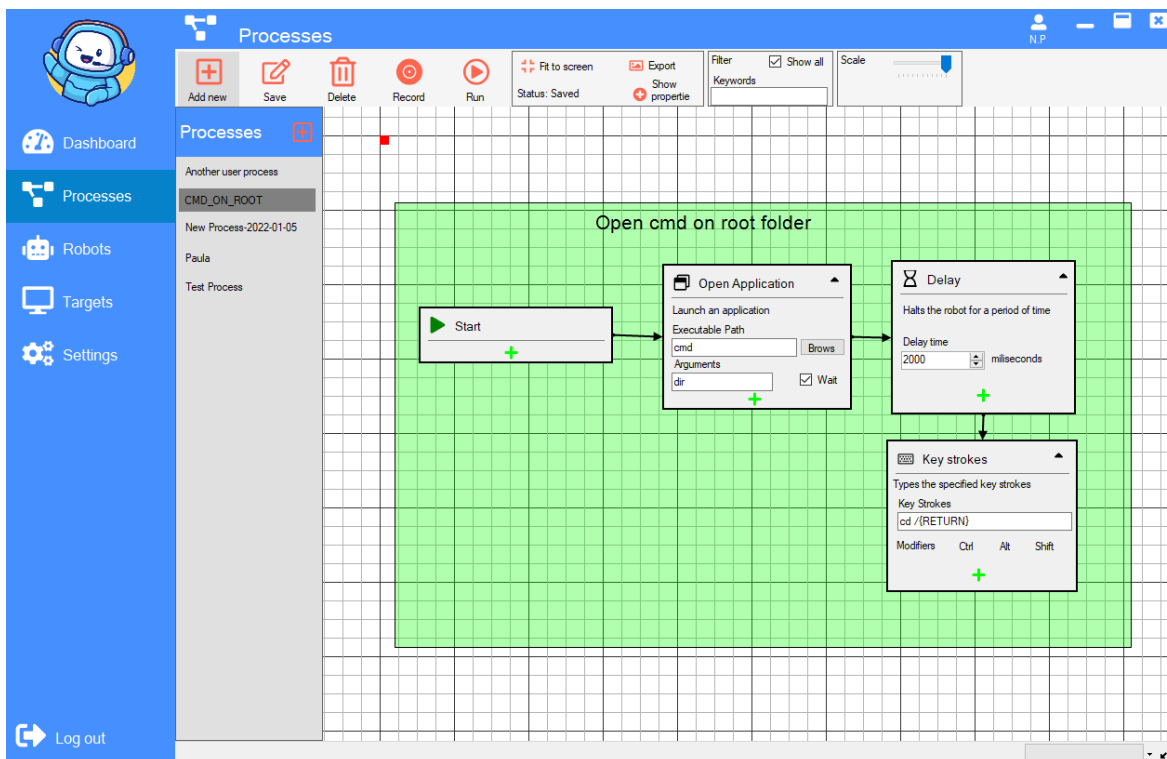


Figure 18 An example process

Tasks

The key components of a process are the tasks. The tasks are the actual instructions that the robot will follow to complete a process. We have created a collection of tasks for the user to choose from.

- a) The start task is a nonoperational task that indicates the start of a process. The robots always execute the start task first and for that reason, any task that is not connected to a start task will not be executed.
- b) The click task is used to direct the robot to position the cursor onto a specific control on the user's desktop and perform a click, double click, or a right click.
- c) The special keys task is responsible for pressing special keys combos. By clicking on the toggle buttons on the task the user can specify the keys to press.
- d) The message box task is used for displaying a message to the user while the robot is running.
- e) The keystrokes task is used for emulating keyboard strokes. Along with the keystrokes, the user can define a key modifier by toggling the modifiers buttons.
- f) The attach task attaches the robot to a specific window on the user's desktop. This is useful because it helps the robot to search for the controls more efficiently.
- g) The open application task starts an instance of the specified program with the specified arguments.
- h) The delay task halts the execution of the process for the time the user specifies in milliseconds.
- i) The open hyperlink task opens the specified URL on the system's default browser.
- j) The open terminal starts a new instance of a terminal window on the specified directory.
- k) The wait for the component task halts the execution of the robot until the specified control appears on the screen. For safety reasons, the task will timeout after 10 minutes if the control never shows up.

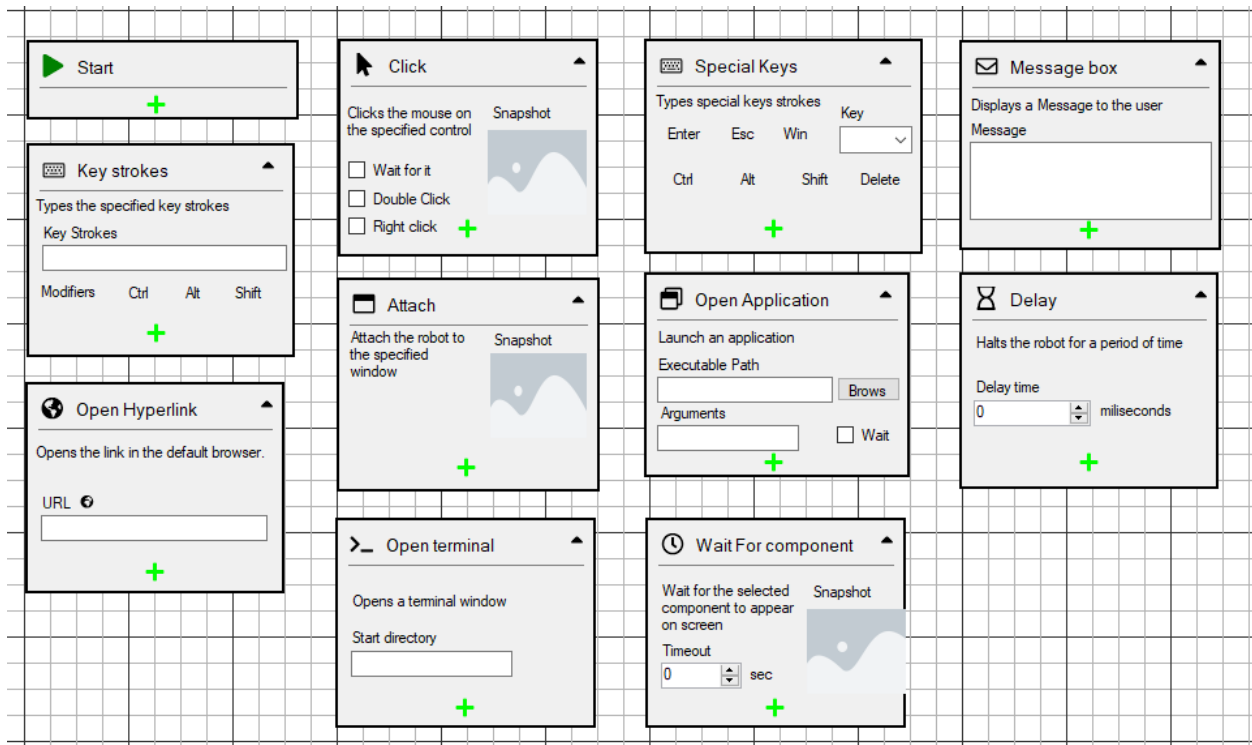


Figure 19 All the available tasks

For aesthetic and usability purposes, the tasks can be displayed on four (4) different states. The normal appearance, when selected, when disabled, and when collapsed.

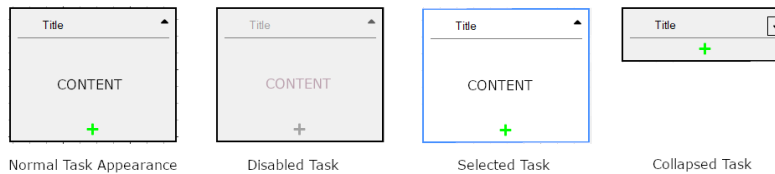


Figure 20 Task states

Targets

The targets page is dedicated to display and handle all the necessary information about the discovered hosts of the user. Its main purpose is to provide the user a way to manage discovered hosts, scan for new hosts on the network, perform vulnerability test and manage the vulnerabilities of each one, and finally to program timed events for a host. The targets page consists of the following UI.

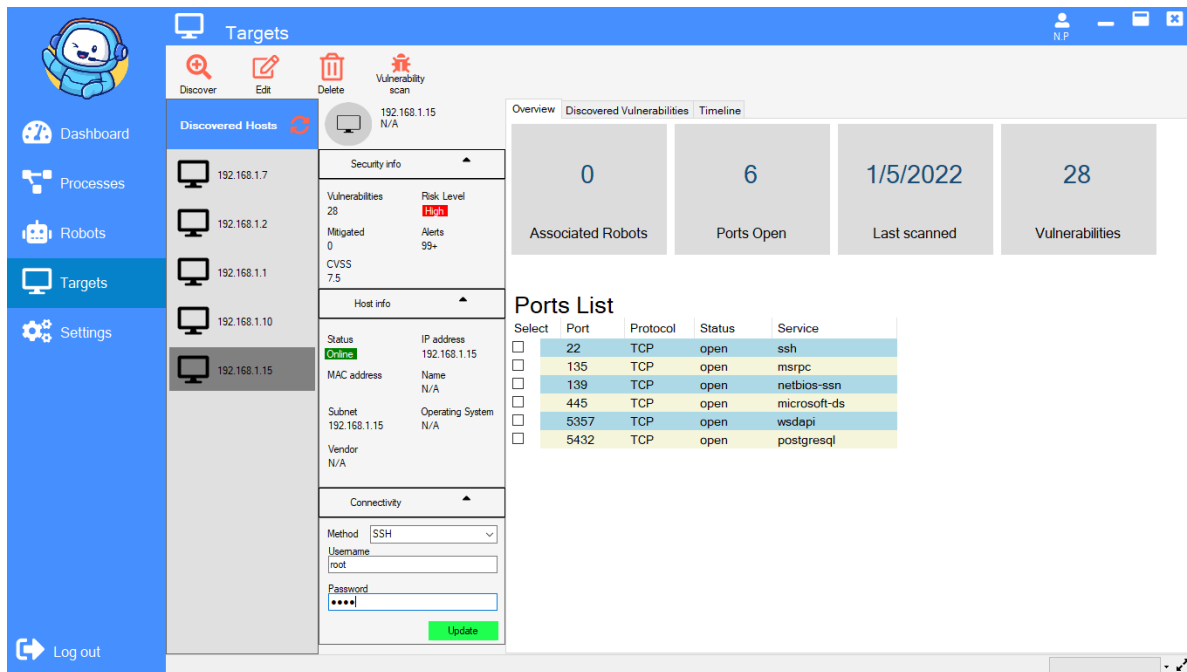


Figure 21 Targets page overview

Action menu

On the top of the page is a menu bar containing all the available functionality of the specific page. Regarding the targets page, the discover button allows the user to scan a network for active hosts while also choosing the method of the scan and the scanner to be used. The delete button forgets the selected host and deletes it from the list. If the user scans again and the deleted hosts are active, they will be displayed again. Finally, the vulnerability scan button allows the user to scan for vulnerabilities on specific targets or on all known targets.

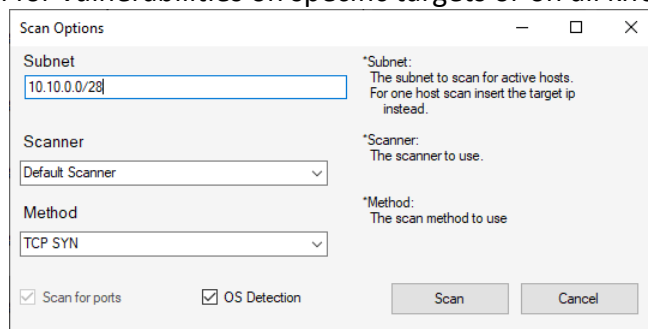


Figure 22 Discover host dialog

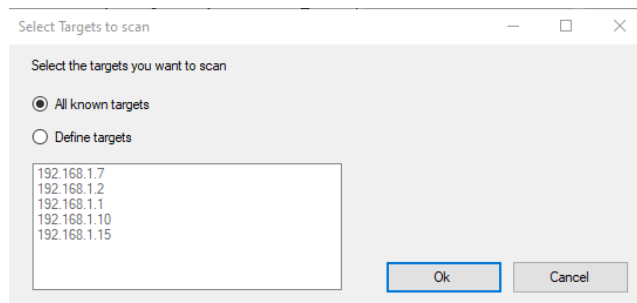


Figure 23 Vulnerability scan dialog

Targets list

On the left side of the page, we can find the discovered hosts list. This list displays all known hosts. Next to the title of the panel, is a refresh button that is used to ping all the known targets to determine if they are still online or not. By clicking on an item on the list, the user selects a target for further information. On the other hand, by right clicking on an item of the list, a context menu appears that gives us the options to rescan the host with a specified method, to vulnerability scan, characterize the machine as critical infrastructure, monitor its network traffic, delete it from the list, open an SSH shell window, and send a shutdown signal.

Target information panel

Next to the targets list, is a panel that displays general information about the selected target. This panel consists of three collapsible panels that are used to categorize the information. The first one holds security metrics and info about the target. The second holds information about the network properties of the target and finally the third holds the SSH credentials of the target.

Main tabbed control

At the center of the page, there is a tabbed control that is used to provide more information about the selected host. There are three different tabs on that control. The overview tab mostly displays general information about the associated robots, the open ports, when it was last scanned and the amount of the discovered vulnerabilities. There is also a list of the open ports of the host.

The screenshot displays a web interface for vulnerability management. On the left is a navigation sidebar with icons for Dashboard, Processes, Robots, Targets, and Settings. The main area is titled 'Targets' and contains a 'Discovered Hosts' list with IP addresses: 192.168.1.7, 192.168.1.2, 192.168.1.1, 192.168.1.10, and 192.168.1.15. The host 192.168.1.15 is selected. To the right of the list is a detailed view for this host, showing 'Security info' (28 vulnerabilities, High risk level), 'Host info' (IP address 192.168.1.15, Status Online), and 'Connectivity' (SSH method, Username root, Password masked). Further right is a 'Ports List' table with columns for Select, Port, Protocol, Status, and Service. The table lists five open ports: 22 (ssh), 135 (msrpc), 139 (netbios-ssn), 445 (microsoft-ds), and 5357 (wsdapi). A 'Ports List' summary card shows 6 ports open and 28 vulnerabilities. The 'Overview' tab is active, showing 0 associated robots, 6 ports open, last scanned on 1/5/2022, and 28 vulnerabilities.

Select	Port	Protocol	Status	Service
<input type="checkbox"/>	22	TCP	open	ssh
<input type="checkbox"/>	135	TCP	open	msrpc
<input type="checkbox"/>	139	TCP	open	netbios-ssn
<input type="checkbox"/>	445	TCP	open	microsoft-ds
<input type="checkbox"/>	5357	TCP	open	wsdapi
<input type="checkbox"/>	5432	TCP	open	postgresql

Figure 24 Targets board - Overview tab

Moving on, the discovered vulnerabilities tab gives us a more graphical way to manage the discovered vulnerabilities of the target. As shown in Figure 25, it consists of a button to scan this target for vulnerabilities on the top right, a pie chart that displays the vulnerabilities based on their Common Vulnerability Scoring System (CVSS) score, and a list with more details about the vulnerabilities. If the user right clicks on an item of the list, as seen in Figure 24, a context menu appears which gives us the following options.

- View vulnerability’s reference webpage
- Create a mitigation process
- Search for existing mitigation processes on the platform
- Ignore this vulnerability for all targets or for this target only.

	CVE	CVSS	Severity	Status	Port/Protocol	Reference
<input type="checkbox"/>	CVE-2009-1...	7.1	High	New	80/tcp, 443/t...	https://vulners.com/cve/CVE-2009-1891
<input type="checkbox"/>	CVE-2009-1...	7.1	High	New	80/tcp	https://vulners.com/cve/CVE-2009-1890
<input type="checkbox"/>	CVE-2012-0...	6.9	Medium	New	80/tcp	https://vulners.com/cve/CVE-2012-0883
<input type="checkbox"/>	CVE-2018-1...	6.8	Medium	New	80/tcp	https://vulners.com/cve/CVE-2018-1112
<input type="checkbox"/>	CVE-2017-9...	6.4	Medium	New	80/tcp, 443/t...	https://vulners.com/cve/CVE-2017-9888
<input type="checkbox"/>	CVE-2009-3...	5.8	Medium	New	80/tcp, 443/t...	https://vulners.com/cve/CVE-2009-3555

Figure 25 Vulnerability Right click Context menu

Figure 26 Targets - Discovered vulnerabilities tab

Finally, the timeline tab, lets the user create timed events for the selected targets. An event could be a robot deployment, a scheduled vulnerability scan, or a network scan. Currently, this tab is not functional, but it will be implemented in the future.

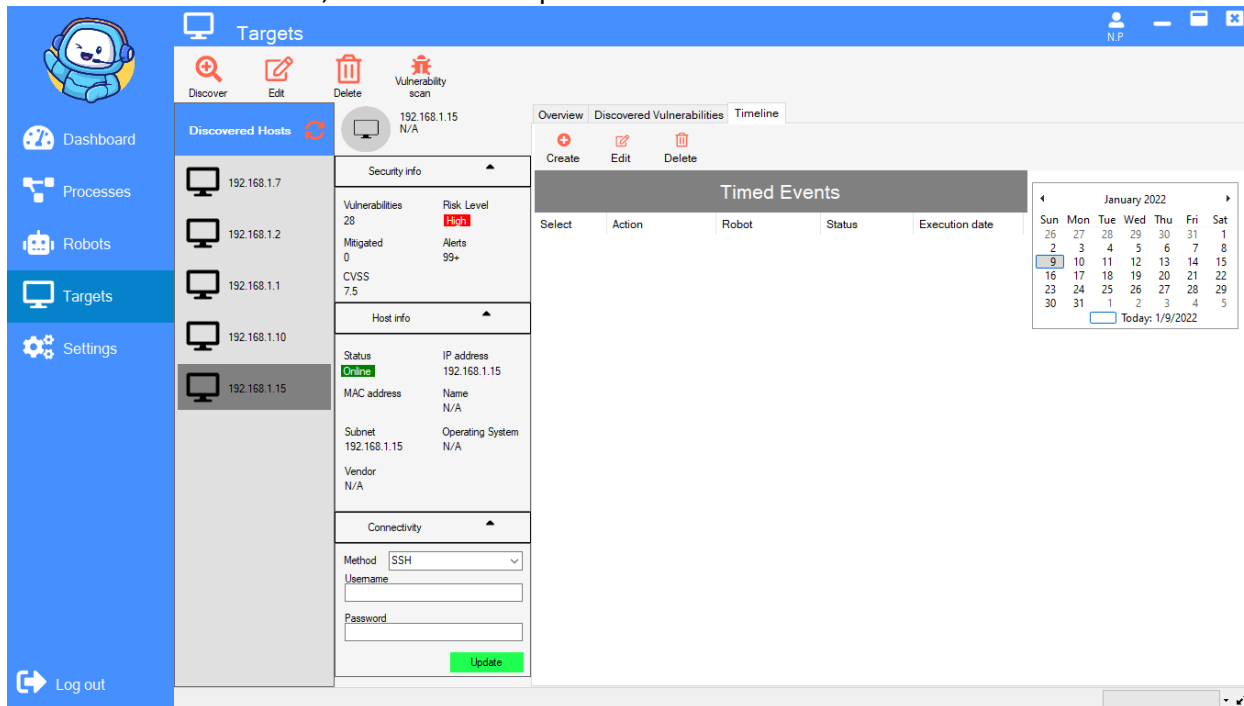


Figure 27 Targets board - Timeline tab

Settings

The settings page contains all the available settings and preferences that the user can change and personalize. The settings are categorized into the environment, automations, scanning, robots, and vulnerability settings for easier navigation. Again, in the context of this thesis, the settings are a prototype and are not yet been implemented.

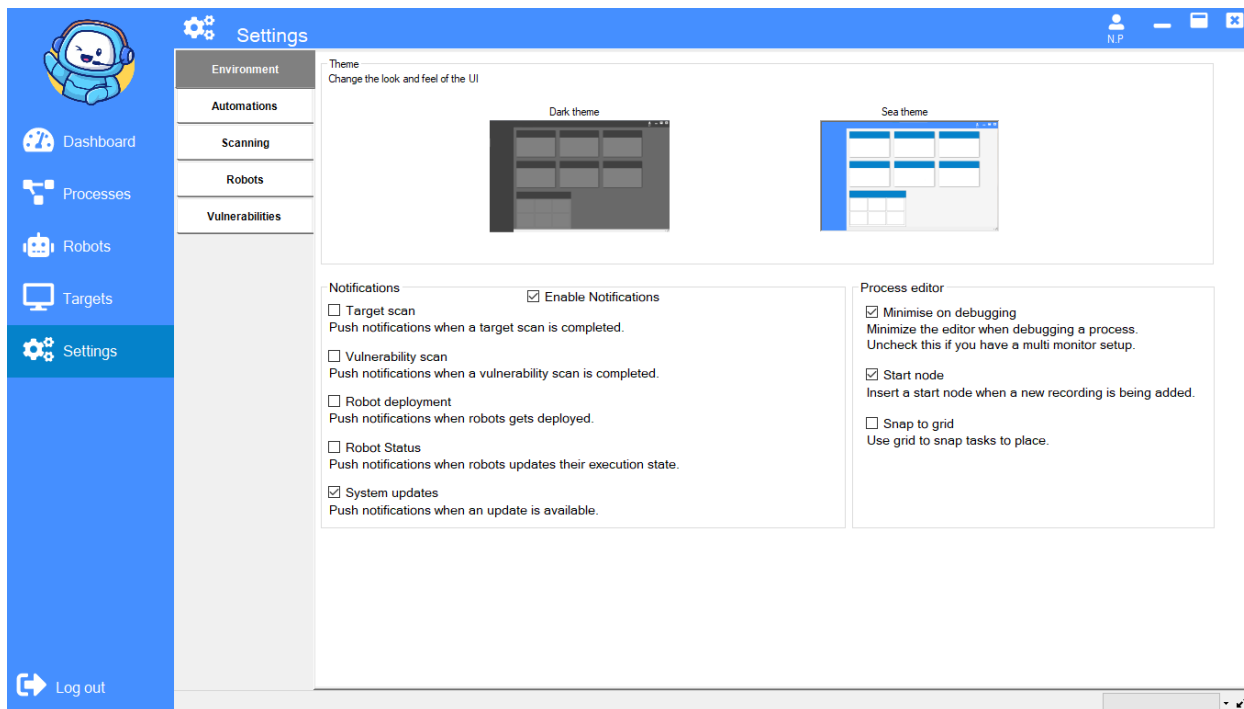


Figure 28 Settings page

Profile

The last page in assistant is the profile page. To navigate to this page, on the top right corner is the profile button. Clicking on this button reveals a context menu where the view profile option is the one that lands the user on the profile page. Here, the users can review their personal info and act about their profile. The users can edit their personal information, change profile picture, change password, update the subscription purchased and delete the account altogether.

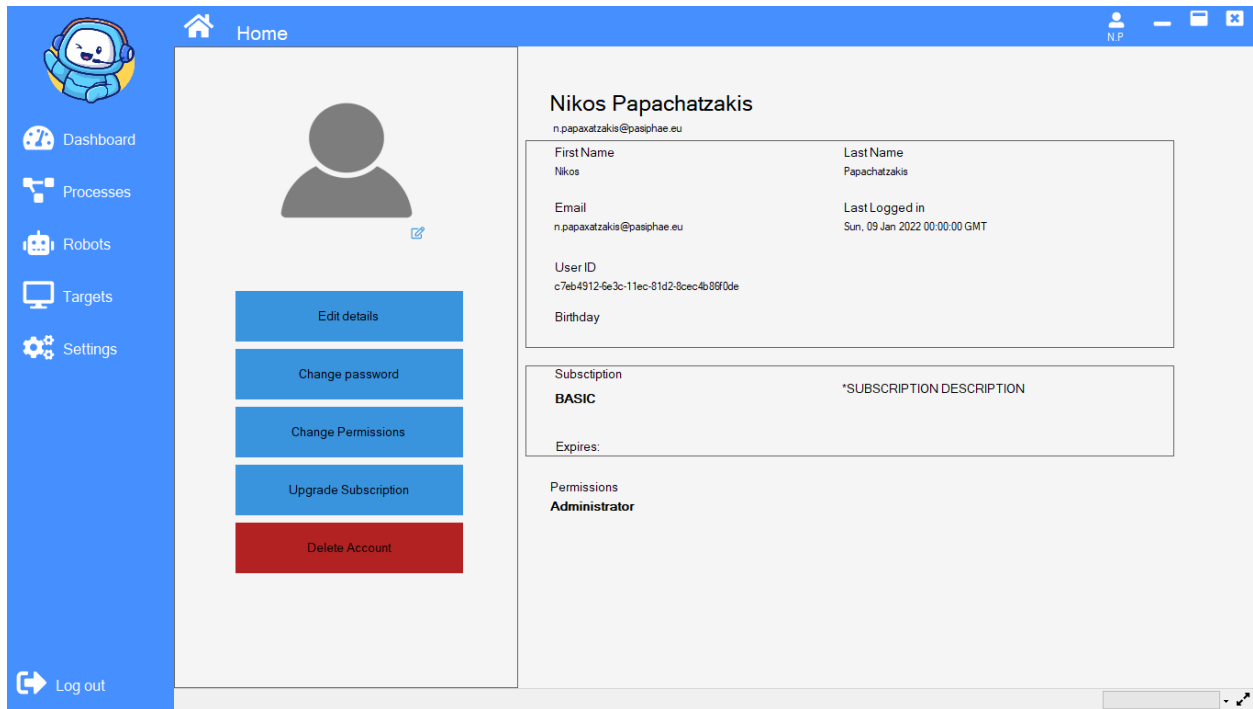


Figure 29 Profile page

Optimizations

Because WinForms is based on GDI+ graphics engine, it does not support hardware-accelerated graphics which means that all the graphics calls are handled by the main thread on the Central Processing Unit (CPU). This means that the performance is quite poor, even for a few components. We came across that problem when the user started making processes longer than a few tasks which made the application run very slow. So, we decided to create our own drawing routines to optimize the drawing procedure by implementing a custom Double buffered graphics object using the GDI+. With all that, we were able to effectively draw to screen a huge number of tasks, and eliminate the screen tearing, stutter, and flickering while keeping the performance on an acceptable level.

We managed to do all that by making the tasks to save a screenshot of their client area into a bitmap and refresh this bitmap every time the task gets invalidated. Then when the user attempts to move the process around, the drawing board prevents the application from automatically drawing the tasks by changing their visibility to false. After that, our custom redraw function pastes the pre-drawn bitmap of each task in the same location as the task is and then writes the whole bitmap on a secondary buffer. When the drawing has been completed, the function copies the secondary buffer to the primary buffer to update the actual graphics on the screen and then moves to compute the next frame on the secondary buffer.

Client API

Moving on to the client API, we designed it to be a singleton instance object that wraps all the communication with the cloud in simple and easy to use functions. These functions are asynchronous with the rest of the application and therefore an event driven model was implemented, to notify the application that the requested data is ready. When the application makes a call to the client API, it builds the request body and sends it to the cloud. When a response comes back from the cloud, it raises a custom-made event and invokes every subscribed function on the application. The “subscriber” functions then handle the response from the cloud.

Robots

Lastly, we have the robots within the frontend package. In robotic process automation, the robots are small pieces of software that mimic the human interaction with the computer by executing strictly defined steps. In our implementation, we used Python programming language to create the robots and compiled them into standalone applications with the PyInstaller¹⁵ utility. Because the robots are compiled into binary files, the deployment process becomes very easy as they come with all the dependencies they need to run on any windows-based machine. The robots are meant to communicate with the backend as well as with the assistant, they need a special token that gives them the authorization needed. For this reason, the main application asks the backend for a special token for the robot beforehand and then moves on using the robot.

Recorder Robot

The recorder robot is used to record the actions of the user on the screen and encode these actions into a process. To recognize the actions that happen on the UI by the user, we used the Microsoft UIA framework. Firstly, the robot constantly monitors the keyboard and the mouse for input. If for example the left mouse button gets clicked, then we make a call to the UIA framework to find out what is the control that got clicked on the screen. The framework then gives us some details of the control such as an automation id, a name, the type of the control, and much more. Then the robot formats the info on a string of characters along with a screenshot of the control (for visualization purposes) and prints the action on the standard output while waiting for more input. Finally, the recording stops when the user hit the Escape (ESC) key on the keyboard. The app listens for output from the recorder and saves it to the current process.

¹⁵ <https://www.pyinstaller.org/>

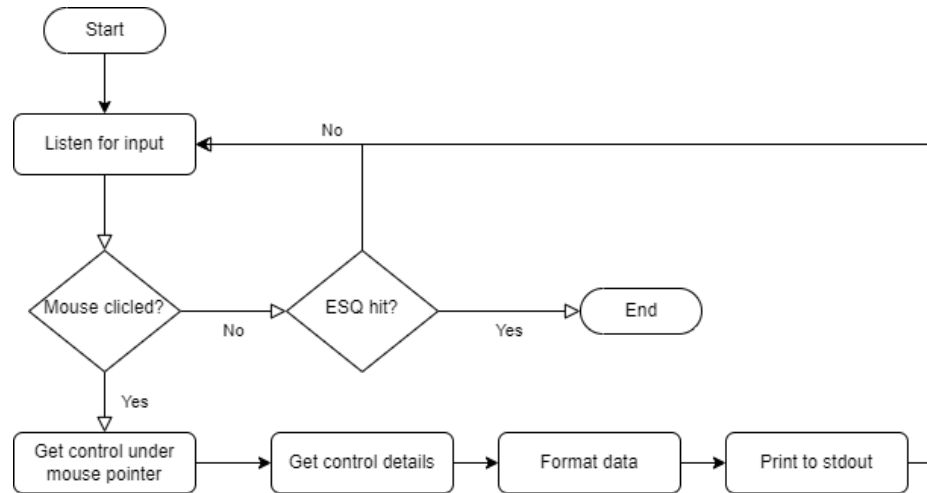


Figure 30 Mouse click recording sequence

Playback Robot

The playback robot is used to replicate the actions that the user recorded. Simply put, the robot takes a process as an input and executes it step by step. To achieve that, the playback robot gets the encoded process string and decodes it into discrete commands. Each command is unique and represents one action. For example, 'lclick[args]' is the command that tells the robot to execute a mouse left button click. The 'args' part of the command comes with all the necessary information about the control that the robot needs to click on. Then, the robot gets the arguments and using the UIA framework, searches the UI for control with matching identifiers. If the robot finds a control that matches, find a clickable (x, y) point on the control, moves the mouse over it, and then perform the left click. If the robot does not find a matching control, halts for 5 seconds and tries again. After three unsuccessful tries, logs an error and stops the execution. After the action has been completed successfully, the robot continues to the rest actions in the process until there are no more actions to do or a fail-safe trigger is triggered. Fail-safe, triggers when the user moves the mouse to the top-left corner of the screen. Having a fail-safe trigger helps when for whatever reason the user wants to stop the execution of the robot.

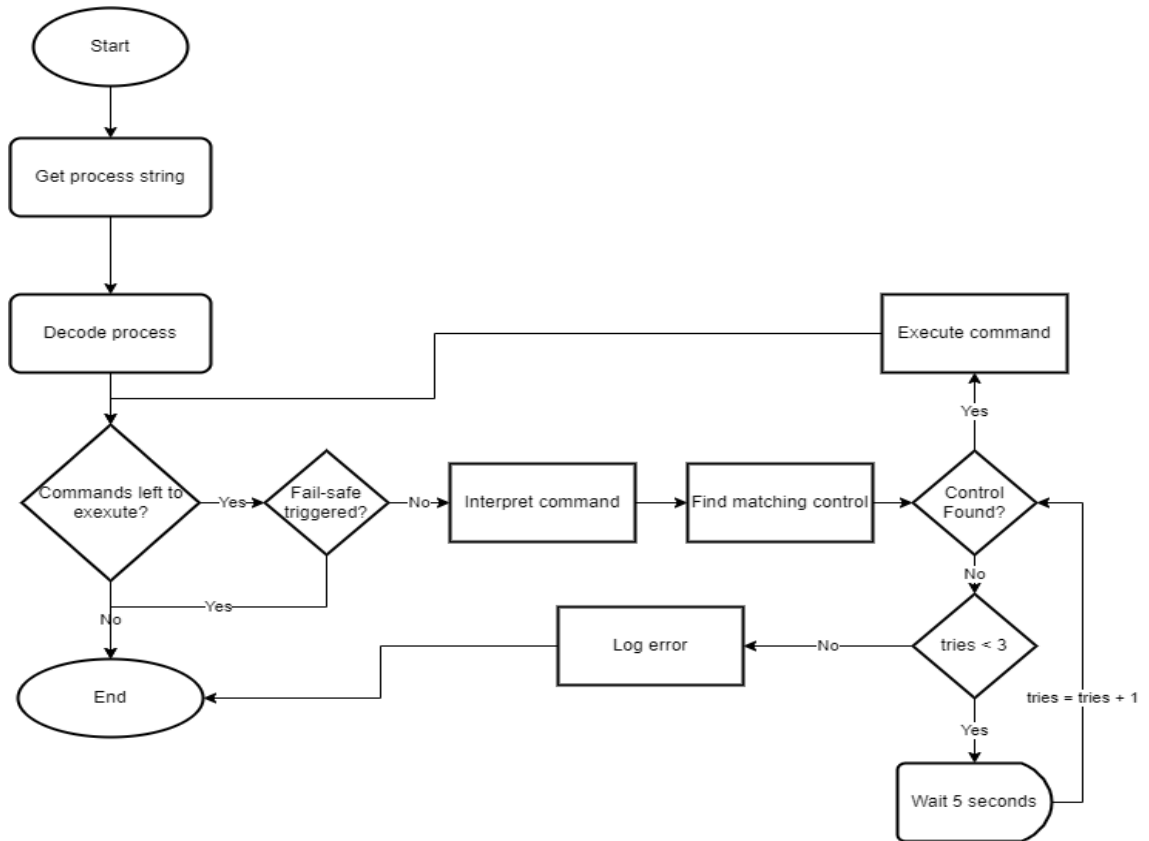


Figure 31 Playback robot flowchart

Scanner Robot

The scanner robot is used to scan the network for machines and vulnerabilities. In our implementation, the scanner robot has three (3) different functions. 1) The host discovery, where we search a network for online machines and their open ports. 2) The Online test, where we ping the known machines to determine if they are still online, and 3) Vulnerability test, where using the Vulners¹⁶ script for NMAP, we determine the vulnerabilities of a specified machine. All this is possible due to NMAP¹⁷ network discovery software, which provides a robust and easy way to scan a network.

¹⁶ <https://nmap.org/nmapdoc/scripts/vulners.html>

¹⁷ <https://nmap.org/>

Robot Deployment

The Robot deployment starts when the user clicks the deploy button on the robot's page. Firstly, the application gets the list of the robots that the user selected for deployment, and for each robot in that list, makes a request to the backend for a specialized token. As we can see in Figure 11, there is a lot of useful information about the robot, for example, the *isHuman* keyword describes if this token is issued to a human user or a robot, and the *robotId* and *processId* give us the robot id and process id respectively. Moreover, the keyword *sub* indicates the username (email) of the user that issued this token and the *runNow* keyword describes that the robot should start execution right away or wait for the token to expire before starting up. We used the expiration date of the token to give the robot a specific time and date of the execution.

```
{
  "typ": "JWT",
  "alg": "HS256"
}
{
  "fresh": false,
  "iat": 1640270249,
  "jti": "02ec7f09-6941-416d-a2e7-ae9fddd7961c",
  "type": "access",
  "sub": "test@pasiphae.eu",
  "nbf": 1640270249,
  "exp": 1640270489,
  "isHuman": false,
  "robotId": "8faf6cd7-1d2b-11ec-af97-8cec4b86f0de",
  "target": "172.16.5.4",
  "processId": "dfc056d7-1d29-11ec-9732-8cec4b86f0de",
  "runNow": "False"
}
```

Figure 32 Example robot token data

After the application successfully retrieves the token from the backend, using the SSH credentials of the target machine, starts an SSH connection to that machine and using the SFTP protocol of the SSH, uploads the binary of the robot to the target machine. Along with the binary file, a configuration file gets transferred which has the encoded token string and a runner script, written in BATCH. After the transfer is completed successfully, the application executes the runner script. The script is responsible for executing the robot. We went with this approach, because the SSHd (SSH daemon), does not let you start applications with GUI (Graphical user interface). That happens because, the SSHd runs as a service, where services have no access to the user desktop, and to start an application of that kind, you need access to the user desktop. We found a workaround to that problem by utilizing a script that creates a Scheduled task using Windows Scheduled tasks. This task is an event type task that waits until something triggers it. The script then triggers the task, and the task launches the robot with elevated permissions. After the robot has successfully launched, it decodes its token and gets the expiration time of the token and waits until then. One minute before the token expires, a dialog appears on the user screen which informs the user that an update is about to happen. The user can postpone¹⁸ the execution of the robot for ten (10) minutes for only one time with the purpose of saving any unsaved work

¹⁸ Note. The postpone mechanism is not yet implemented.

and stopping using the computer. If the time run out or the *Ok* button gets pressed, the robot, using its token, requests the process that it is about to run from the backend. If the token is valid, the backend responds back to the process. The robot decodes the process and executes it. If the process is successfully executed, the robot posts to the backend its status as “Ended”. If not, depending on the situation, the robot might post status like “Crashed”, “Stopped”, “Running” or “Waiting”.

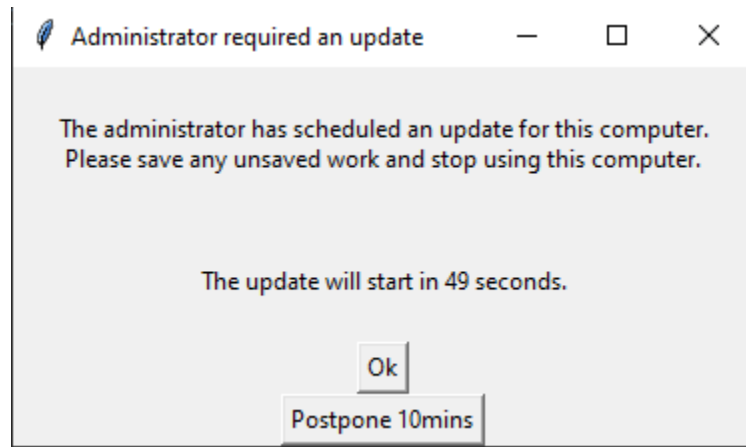


Figure 33 Robot warning dialog - Postpone execution

5. Use-case Scenario

In this section, a specific use-case scenario will be presented. This use-case is about the actions of the user, to completely remediate one or more vulnerabilities on a remote machine.

5.1. Methodology

As described in [15], the network operator must do a sequence of actions, to complete the Vulnerability management cycle (VMC) procedure. Regarding that, in our example, the first thing the user must do is to login into the RPA Assistant using their registered account credentials. In the case that the user does not have an account yet, by using the sign-up form, a new account can be created. After the successful login, through the “Targets” board on the sidebar, the user can scan the whole network for hosts or scan for a specific target, by clicking the “Discover” button on the header menu. Then, a popup window appears, where the user fills up the network address (with CIDR¹⁹ identifier) or the specific host address, selects the scanner, and the method to be used. Upon successful completion of the network scan, the discovered hosts are depicted on the sidebar. By clicking on the target of choice, details of that target are shown in the sidebar. The user must set up the connection method to that machine by filling up the SSH credentials of the machine. After that, through the “Discovered vulnerabilities” tab, the user can start a vulnerability scan on that specific target, or by clicking the “Vulnerability scan” button on the header menu, a multiple target scan can be initiated. Moving on, when the vulnerability scan on the target is completed, on the “Discovered vulnerabilities” tab of the target, a list containing the discovered vulnerabilities will be shown. By right clicking on one or multiple entries on that list, a context menu appears that let the user “Create mitigation process” for these vulnerabilities or search for an already created process from other users on the platform. In our example, we assume that there is no such process, so the user moves on to create a remediating process. That can be done using the built-in GUI recorder tool to record the remediating actions automatically or manually by adding and connecting tasks together. For debugging purposes, the user can run the process locally using the “Run” button on the header menu. When the user has finished the process, on the “Robots” board is a list of the user’s robots. Using the “Create robot” button, the user creates a robot with the desired target and selects the process that the robot will execute. When the robot is created, by selecting the robot and clicking on the “Deploy” button, the assistant will connect to the machine using the user defined SSH credentials and deploy the robot automatically on the target machine. The robot gets uploaded to the target and then initiates the remediation procedure. After the successful execution of the robot, the vulnerabilities should be remediated. To verify that, the user must perform a vulnerability scan on that target as described above and manually identify that the remediated vulnerabilities have gone.

¹⁹ https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

Enabling Automatic Vulnerability Management Through a Robotic Process Automation Framework
 Papachatzakis Nikolaos

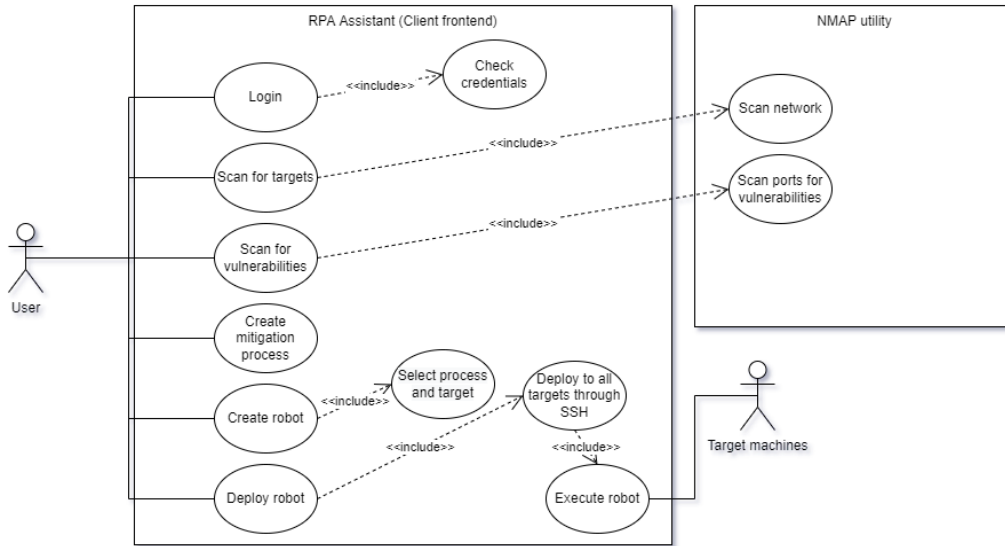


Figure 34 Use-case diagram of Vulnerability management scenario

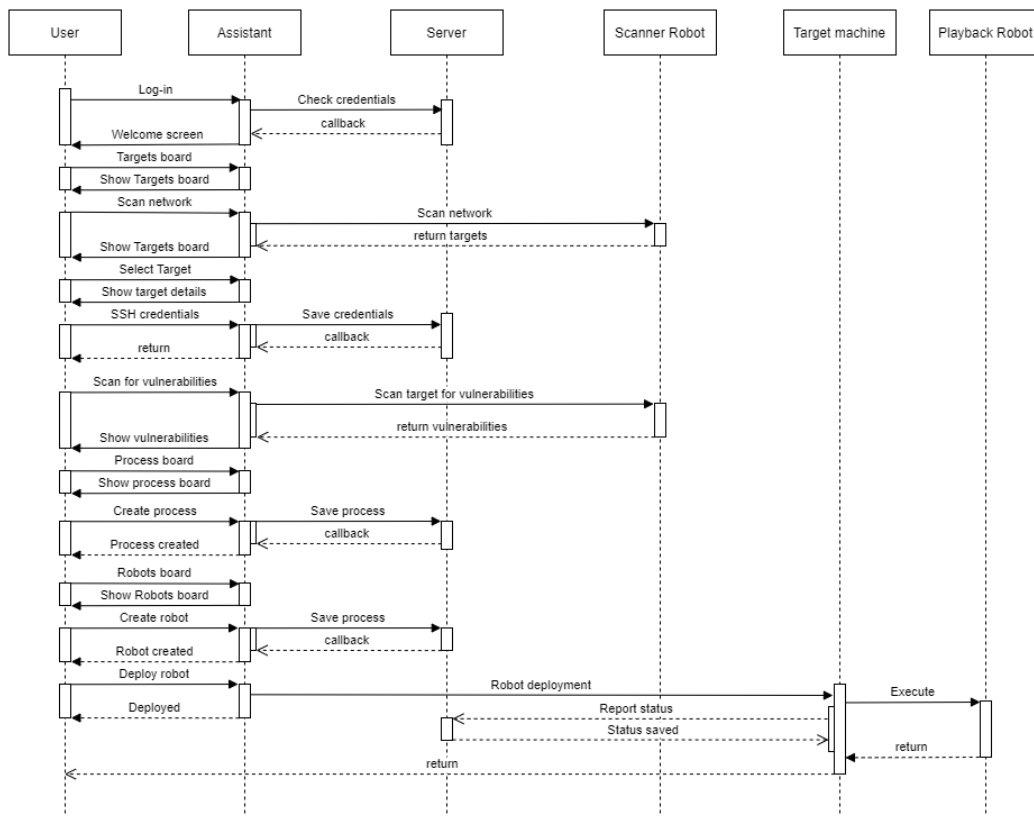


Figure 35 Sequence diagram of Vulnerability management scenario

that the robots have successfully finished the job, the next thing to do was to perform a second vulnerability scan on those two machines. Doing that, as seen in [Figure 36](#), [Figure 37](#) we found out that the remediation was indeed successful because there were no vulnerabilities reported by the scan and the website was online as expected.

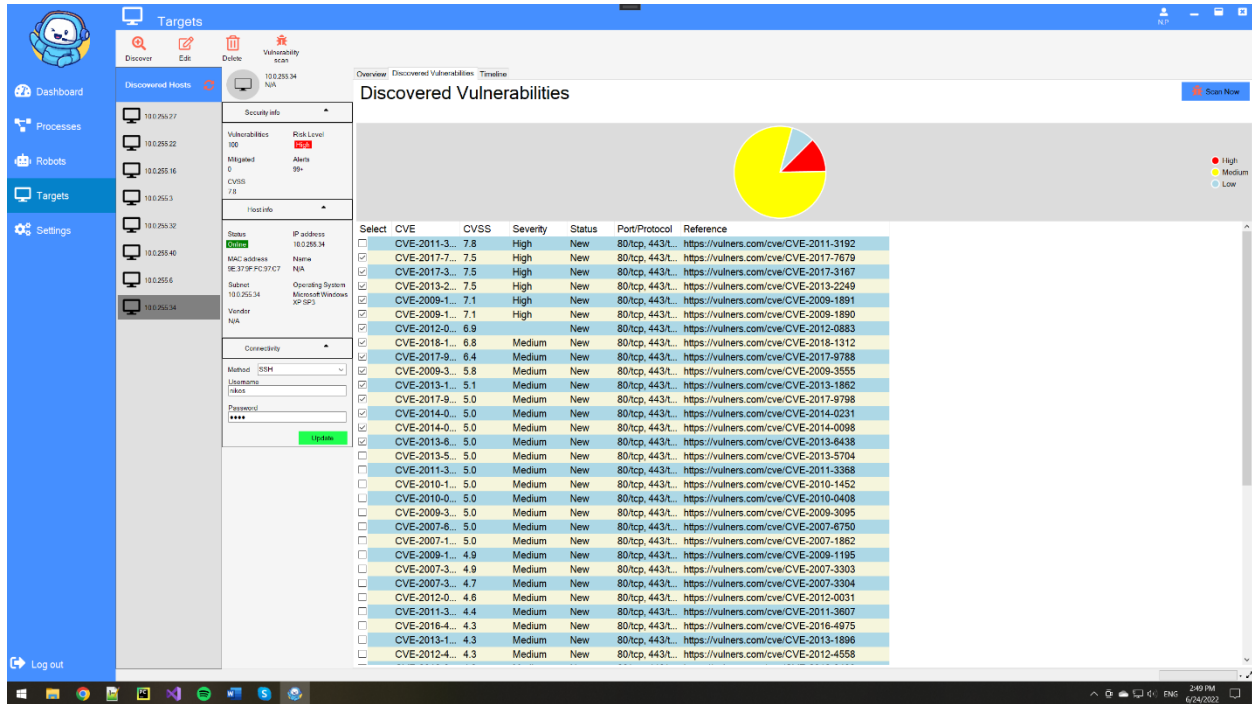


Figure 37 Before remediation - Discovered vulnerabilities

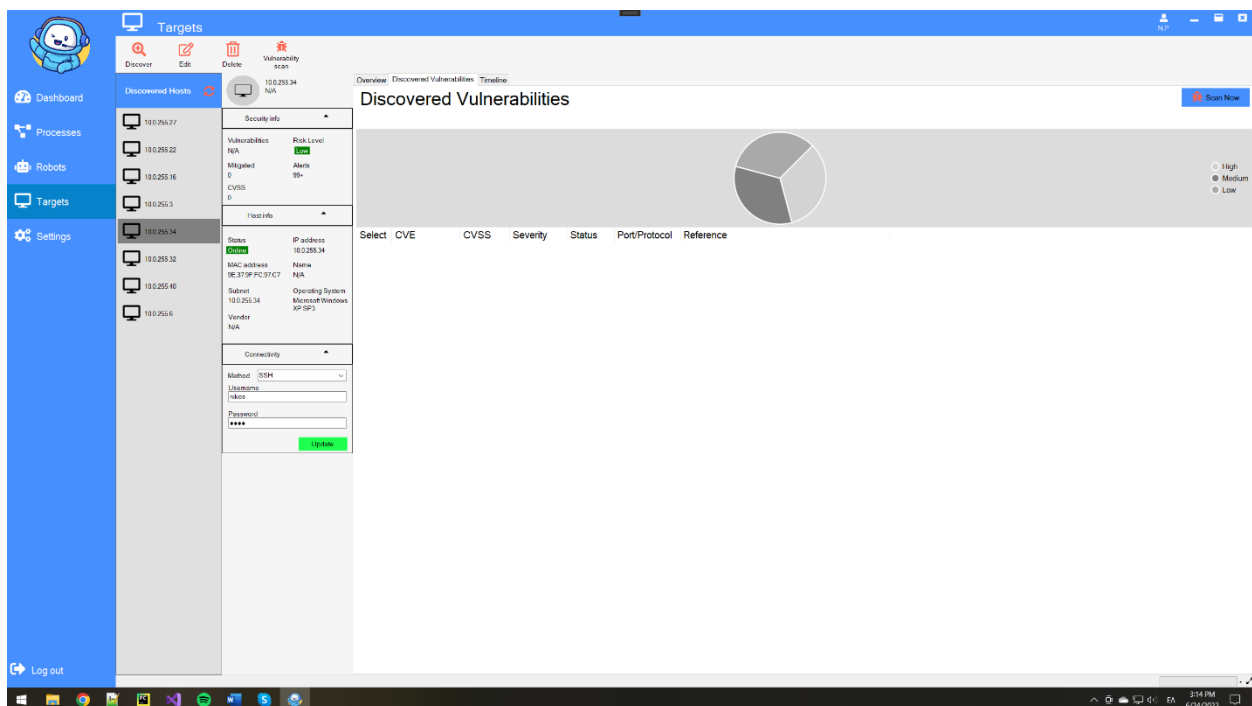


Figure 38 After remediation - Discovered vulnerabilities

6. Conclusion

In this thesis, we presented a cloud native Robotic Process Automation application that is used to help the network administrators automate the procedure of vulnerability management. It can automatically scan the network for targets, scan the targets for known vulnerabilities, and with the help of the RPA technics, automates the remote deployment of the remediation procedure on multiple organization assets. Moreover, we showcased a real-world use-case scenario using the application to detect, assess and remediate some vulnerabilities on two vulnerable machines on the network running an old version of Apache web server. In the future, we plan to upgrade the recorder robots to run remotely, allowing the administrators to record the remediation process on a remote machine directly. Finally, we plan to extend the support of the application to other operating systems, including Apple's Mac OSX and Debian based Linux distributions.

7. References

- [1] D. G., "TechJury." .
- [2] "CVEDetails." .
- [3] K. Karampidis, S. Panagiotakis, M. Vasilakis, E. K. Markakis, and G. Papadourakis, "Industrial cybersecurity 4.0: Preparing the operational technicians for industry 4.0," *IEEE Int. Work. Comput. Aided Model. Des. Commun. Links Networks, CAMAD*, vol. 2019-September, Sep. 2019, doi: 10.1109/CAMAD.2019.8858454.
- [4] Y. Hu, Y. Zhang, and D. Gu, "Automatically patching vulnerabilities of binary programs via code transfer from correct versions," *IEEE Access*, vol. 7, pp. 28170–28184, 2019, doi: 10.1109/ACCESS.2019.2901951.
- [5] J. Jurn, T. Kim, and H. Kim, "An Automated Vulnerability Detection and Remediation Method for Software Security," *Sustainability*, vol. 10, no. 5, p. 1652, May 2018, doi: 10.3390/su10051652.
- [6] F. Zhang and Q. Li, "Dynamic Risk-Aware Patch Scheduling," *2020 IEEE Conf. Commun. Netw. Secur. CNS 2020*, 2020, doi: 10.1109/CNS48642.2020.9162225.
- [7] F. Zhang, P. Huff, K. McClanahan, and Q. Li, "A Machine Learning-based Approach for Automated Vulnerability Remediation Analysis," Jun. 2020, doi: 10.1109/CNS48642.2020.9162309.
- [8] J. Jacobs, S. Romanosky, I. Adjerid, and W. Baker, "Improving vulnerability remediation through better exploit prediction," *J. Cybersecurity*, vol. 6, no. 1, Jan. 2020, doi: 10.1093/cybsec/tyaa015.
- [9] D. Malzahn, Z. Birnbaum, and C. Wright-Hamor, "Automated Vulnerability Testing via Executable Attack Graphs," Jul. 2020, pp. 1–10, doi: 10.1109/cybersecurity49315.2020.9138852.
- [10] S. A. P. Kumar and B. Xu, "Vulnerability Assessment for Security in Aviation Cyber-Physical Systems," in *Proceedings - 4th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2017 and 3rd IEEE International Conference of Scalable and Smart Cloud, SSC 2017*, Jul. 2017, pp. 145–150, doi: 10.1109/CSCloud.2017.17.
- [11] T. H. M. Le, B. Sabir, and M. A. Babar, "Automated software vulnerability assessment with concept drift," in *IEEE International Working Conference on Mining Software Repositories*, May 2019, vol. 2019-May, pp. 371–382, doi: 10.1109/MSR.2019.00063.
- [12] Y. Nikoloudakis, E. Pallis, G. Mastorakis, C. X. Mavromoustakis, C. Skianis, and E. K. Markakis, "Vulnerability assessment as a service for fog-centric ICT ecosystems: A healthcare use case," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 5, pp. 1216–1224, Sep. 2019, doi: 10.1007/S12083-019-0716-Y/TABLES/2.
- [13] and S. P. Nishant Sharma, H. Parveen Sultana, Asif Sayyad, Rahul Singh, "Remote Automated Vulnerability Assessment and Mitigation in an Organization," 2020, pp. 219–227.
- [14] N. Nakhla, K. Perrett, and C. McKenzie, "Automated computer network defence using ARMOUR: Mission-oriented decision support and vulnerability mitigation," Oct. 2017, doi: 10.1109/CyberSA.2017.8073389.
- [15] E. Heaslip, "Nightfall." .
- [16] Y. Nikoloudakis *et al.*, "Towards a Machine Learning Based Situational Awareness Framework for Cybersecurity: An SDN Implementation," *Sensors 2021, Vol. 21, Page 4939*, vol. 21, no. 14, p. 4939, Jul. 2021, doi: 10.3390/S21144939.

8. Appendix

8.1. Assistant- Drawing board source code

```
usingFontAwesome.Sharp;
usingRPA_Client.Events;
usingSystem.Reflection;
usingSystem.Runtime.InteropServices;
usingSystem.Threading;
usingSystem.Windows.Forms;

namespaceRPA_Client.Forms{
publicpartialclassProcessDrawingBoard:UserControl{
publiceventEventHandlerOnProcessChanged;
privateintindx=0;
privatePointpointOfClick;
privatebooldragging=false;
privateboolscaling=false;
privateboolselecting=false;
privateboolconnecting=false;
privateboolgroupingTasks=false;
privateboolrectDragging=false;
privateList<Task>selectedTasks;
privatePointselectionStart;
privatePointselectionEnd;
privatePointconnectingFrom;
privatePointorigin;
privateboolbackgroundDisabled=false;
privateboolexporting=false;

[DllImport("user32.dll")]
privateexternstaticIntPtrSendMessage(IntPtrhWnd,intmsg,intwParam,IntPtrlParam);

privateconstintWM_SETREDRAW=11;
set{
this.scaling=value;
}
}
publicboolEnableTaskGroupSelection{
get{returngroupingTasks;}
set{
this.groupingTasks=value;
if(value){
this.Cursor=Cursors.Cross;
}else{
this.Cursor=Cursors.Default;
}
}
}
publicintIndex{
get{returnindx;}
```

```
;
handler?.Invoke(null,newEventArgs());
}

public ProcessDrawingBoard(){
InitializeComponent();
this.SetStyle(ControlStyles.UserPaint | ControlStyles.AllPaintingInWmPaint,true);
this.SetStyle(ControlStyles.OptimizedDoubleBuffer,true);
ctx
connections=new Dictionary<int,List<int>>();
toolsAvailable=new List<Task>();
clipBoard=new List<Task>();
groupRectangles=new List<TaskSelectionRectangle>();
selectedGroupRectangles=new List<TaskSelectionRectangle>();
initialised=true;
foreach(Task t in Controls){

selectedTasks.Add(t);
t.Selected=true;
}
}

private void RecreateBuffers(){
//Check that we aren't disposing or this could be invalid.
if(!initialised | | this.IsDisposed)
return;
ctx.MaximumBuffer=new Size(this.Width+1,this.Height+1);
bgctx.MaximumBuffer=new Size(this.Width+1,this.Height+1);
//Dispose of old backbuffer Graphics (if one has been created already)
if(graphics!=null)
graphics.Dispose();
if(bgGraphics!=null)
bgGraphics.Dispose();
graphics=ctx.Allocate(this.CreateGraphics(),
new
if(EnableScaling){
Zoom(v);
}else{
[16]
}
}
selectedTasks.Clear();
Redraw();
this.Invalidate();

}

private void OnMouseDown(object sender, MouseEventArgs e){
TaskSelectionRectangle rect=CursorInSelectionRectangle();

if(e.Button==MouseButtons.Right){
pointOfClick=this.PointToClient(Cursor.Position);
dragging=true;
}
```



```
this.Cursor=Cursors.SizeAll;
(!rect.Lock){
if(!rect.Selected){

rect.Selected=true;
selectedGroup=rect;
selectedGroupRectangles.Add(rect);

}
pointOfClick=this.PointToClient(Cursor.Position);
rectDragging=true;
this.Cursor=Cursors.SizeAll;
}
}

}
ReDraw();
InvalidateAndUpdate();

}

privateTaskSelectionRectangleCursorInSelectionRectangle(){
Pointc=this.PointToClient(Cursor.Position);
foreach(TaskSelectionRectanglegroupRectangles){
if(rect.Rectangle.Intersects(newRectangle(c,newSize(1,1))){
returnrect;
}
}
returnnull;
}

privateList<Task>GetGroupedTasks(TaskSelectionRectanglerect){
List<Task>grouped=newList<Task>();
foreach(Taskinthis.Controls){
if(rect.Rectangle.Intersects(t.Bounds)){
grouped.Add((Task)t);
}
}
returngrouped;
}

privatevoidOnMouseMove(objectsender,MouseEventArgse){
if(dragging){
Pointmpos=this.PointToClient(Cursor.Position);
foreach(Taskchildinthis.Controls){
if(child.ParentGroup==null){
child.Location=newPoint(child.Left+(mpos.X-pointOfClick.X),child.Top+(mpos.Y-pointOfClick.Y));
}
}
foreach(TaskSelectionRectanglegroupRectangles){
rect.Translate((mpos.X-pointOfClick.X),(mpos.Y-pointOfClick.Y),driver:null);
}
//updateworldorigin
origin.X+=(mpos.X-pointOfClick.X);
origin.Y+=(mpos.Y-pointOfClick.Y);
```

```
pointOfClick=mpos;
ReDrawBackground();
ReDraw();
DrawTasks(false);
InvalidateAndUpdate();
return;
}
if(rectDragging){
Pointmpos=this.PointToClient(Cursor.Position);
TranslateProcess(GetRectOutOfBorderDisplacement(selectedGroup.Rectangle));
foreach(TaskSelectionRectanglerectinselectedGroupRectangles){
rect.Translate(mpos.X-pointOfClick.X,mpos.Y-pointOfClick.Y,driver:null);
}
pointOfClick=mpos;
ReDraw();
DrawTasks(false);
InvalidateAndUpdate();
return;
}
if(selecting){

selectionEnd=this.PointToClient(Cursor.Position);
SetSelectionRect();
ReDraw();
InvalidateAndUpdate();
return;
}
privatevoidDrawTasks(boolv){
if(tasksVisible==v)return;

foreach(TasktinControls){
t.Visible=v;
}
tasksVisible=v;
InvalidateAndUpdate();
}

privatevoidInvalidateAndUpdate(){
this.Invalidate();
this.Update();
}

privatevoidOnMouseUp(objectsender,MouseEventArgse){

TaskSelectionRectanglerect=CursorInSelectionRectangle();
if(e.Button==MouseButton.Right){
if(rect!=null){
if(!rect.Lock){
groupContext.Show(Cursor.Position);
selectedGroup=rect;
}
}
}else{
```

```
if(!EnableTaskGroupSelection){
ClearSelected();
}
}

if(connecting){
connecting=false;
ReDraw();
this.Invalidate();
newTaskContext.Show(Cursor.Position);
return;
}
if(dragging){
dragging=false;
DrawTasks(true);
this.Cursor=Cursors.Default;
}elseif(rectDragging){
rectDragging=false;
DrawTasks(true);
this.Cursor=Cursors.Default;
}elseif(selecting){
selecting=false;
if(selectionStart==this.PointToClient(Cursor.Position)){
ClearSelected();
return;
}
SetSelectionRect();
if(groopingTasks){

TaskGroupTitleDialogdialog=newTaskGroupTitleDialog();
DialogResultres=dialog.ShowDialog();

if(res==DialogResult.OK){

Randomr=newRandom();
TaskSelectionRectangleg=newTaskSelectionRectangle(){
Rectangle=selection,
FillColor=Color.FromArgb(80,r.Next(0,255),r.Next(0,255),r.Next(0,255)),
BorderColor=Color.Black,
Selected=false,
Title=dialog.GroupTitle
};
List<Task>children=GetGroupedTasks(g);
foreach(Tasktinchildren){
g.Addchild(t);
t.ParentGroup=g;
}
groupRectangles.Add(g);
EnableTaskGroupSelection=false;
}

}else{
SelectOverlappedTasks();
```

```
}
ReDraw();
InvalidateAndUpdate();
}
((ProcessBoard)this.Parent).SaveProcess();
}

publicvoidTranslateToOrigin(){
Rectangleareaofinterest=GetAreaOfInterest();
Pointtranslation=newPoint((-1)*areaofinterest.X+10,(-1)*areaofinterest.Y+10);
TranslateProcess(translation);
ReDraw();
this.Invalidate();
}

privatevoidSetSelectionRect(){
intx,y;
intwidth,height;
x=selectionStart.X>selectionEnd.X?selectionEnd.X:selectionStart.X;
y=selectionStart.Y>selectionEnd.Y?selectionEnd.Y:selectionStart.Y;
width=selectionStart.X>selectionEnd.X?selectionStart.X-selectionEnd.X:selectionEnd.X-selectionStart.X;
height=selectionStart.Y>selectionEnd.Y?selectionStart.Y-selectionEnd.Y:selectionEnd.Y-selectionStart.Y;
selection=newRectangle(x,y,width,height);
}

publicvoidZoom(floatdelta){
if(scaling||false){//FIXME-----FIXME
//return;//FIXME-----FIXME-----FIXME-----FIXME-----FIXME-----FIXME-----FIXME-----FIXME-----FIXME
//foreach(Controlchildinthis.Controls){
//child.Scale(newSizeF(delta,delta));
//child.Invalidate();
//}

graphics.Graphics.ScaleTransform(delta,delta);
bgGraphics.Graphics.ScaleTransform(delta,delta);
globalScaling=globalScaling+delta;
Console.WriteLine("SCALE+{0}",globalScaling);
ReDrawBackground();
ReDraw();
InvalidateAndUpdate();
ProcessChanged();
}
}

privatevoidSelectOverlappedTasks(){
ClearSelected();
foreach(Controlcinthis.Controls){
if(cisTask){
if(selection.Intersects(c.Bounds)){
((Task)c).Selected=true;
selectedTasks.Add((Task)c);
}
}
}
}
```

```
}  
}  
foreach(TaskSelectionRectanglecingroupRectangles){  
if(selection.IntersectsWith(c.Rectangle)){  
selectedGroupRectangles.Add(c);  
c.Selected=true;  
}  
}  
}  
privateTaskGetTaskByIndex(intindx){  
foreach(Taskinthis.Controls){  
if(t.Index==indx){  
returnt;  
}  
}  
returnnull;  
}  
protectedoverridevoidOnPaint(PaintEventArgse){  
if(!this.IsDisposed&&graphics!=null){  
//bgGraphics.Render(e.Graphics);  
graphics.Render(e.Graphics);  
  
}  
}  
publicstaticfloatGetWindowsScaling(){  
return(float)(Screen.PrimaryScreen.Bounds.Width/System.Windows.SystemParameters.PrimaryScreenWidth);  
}  
privatevoidReDrawBackground(){  
if(!initialised)return;  
Graphicsg=bgGraphics.Graphics;  
g.Clear(Color.White);  
Penp=newPen(Color.FromArgb(255,150,150,150),1);  
//Drawbackgroundgrid  
for(inty=-origin.Y;y<Height-origin.Y;y++){  
if((y%(80*GetWindowsScaling()))==0){  
p.Color=Color.FromArgb(255,50,50,50);  
  
g.DrawLine(p,newPoint(0,y+origin.Y),newPoint(Width,y+origin.Y));  
continue;  
}  
if((y%(20*GetWindowsScaling()))==0){  
p.Color=Color.FromArgb(255,180,180,180);  
g.DrawLine(p,newPoint(0,y+origin.Y),newPoint(Width,y+origin.Y));  
continue;  
}  
  
}  
p.Width=1;  
for(intx=-origin.X;x<Width-origin.X;x++){  
if((x%(80*GetWindowsScaling()))==0){  
p.Color=Color.FromArgb(255,50,50,50);  
g.DrawLine(p,newPoint(x+origin.X,0),newPoint(x+origin.X,Height));  
}elseif((x%(20*GetWindowsScaling()))==0){
```

```
p.Color=Color.FromArgb(255,180,180,180);
g.DrawLine(p,newPoint(x+origin.X,0),newPoint(x+origin.X,Height));
}

}
g.FillRectangle(Brushes.Red,newRectangle(origin.X,origin.Y,10,10));
}

privatevoidReDraw(Graphicsg=null){
if(gisnull){
g=graphics.Graphics;
}
//Clearthegraphics
g.Clear(Color.White);
if(RenderBackground){
gGraphics.Render(g);
}
//drawashedrectangewhileselecting
g.

}
//setpenwithcustomtriangleendcap
Penpen=newPen(Color.Black,2F);
AdjustableArrowCaparrowCap=newAdjustableArrowCap(5,5);
pen.CustomEndCap=arrowCap;
pen.StartCap=LineCap.RoundAnchor;
//drawarrowforthenewconnection
if(connecting){
g.DrawLine(pen,connectingFrom,this.PointToClient(Cursor.Position));
}
foreach(TaskSelectionRectanglerectingroupRectangles){
rect.Draw(g);
}
//loopthroughallconnectionsanddrawarrowsforeveryone
foreach(KeyValuePair<int,List<int>>pairinconnections){
Task=GetTaskByIndex(pair.Key);
if(t==null)continue;
foreach(intiinpair.Value){
Task1=GetTaskByIndex(i);
if(t1==null)
continue;
Rectangleb1=t.Bounds;
Rectangleb2=t1.Bounds;
if(b1.Top>b2.Bottom){
if(b1.Right<b2.Left){
//ConnectRightBottom
g.DrawLine(pen,newPoint(b1.Right,b1.Top+b1.Height/2),newPoint(b2.Left+b2.Width/2,b2.Bottom));
}elseif(b1.Left>b2.Right){
//ConnectLeftBottom
g.DrawLine(pen,newPoint(b1.Left,b1.Top+b1.Height/2),newPoint(b2.Left+b2.Width/2,b2.Bottom));
}else{
//ConnectTopBottom
g.DrawLine(pen,newPoint(b1.Left+b1.Width/2,b1.Top),newPoint(b2.Left+b2.Width/2,b2.Bottom));
}
```

```
}
}elseif(b1.Bottom<b2.Top){
if(b1.Right<b2.Left){
//ConnectRightTop
g.DrawLine(pen,newPoint(b1.Right,b1.Top+b1.Height/2),newPoint(b2.Left+b2.Width/2,b2.Top));
}elseif(b1.Left>b2.Right){
//ConnectLeftTop
g.DrawLine(pen,newPoint(b1.Left,b1.Top+b1.Height/2),newPoint(b2.Left+b2.Width/2,b2.Top));
}else{
//ConnectBottomTop
//starttask(bottom)toendtask(top)
g.DrawLine(pen,newPoint(b1.Left+b1.Width/2,b1.Bottom),newPoint(b2.Left+b2.Width/2,b2.Top));
}
}else{
if(b1.Left>b2.Right){
//ConnectLeftRight
g.
g.DrawImage(t.Bitmap,t.Location);
}
}
}

//RECURSIVEFUNCTION
publicvoidPrintProcess(intindx){
if(indx==0){
indx=connections.Keys.Min();
Console.WriteLine("Root"+indx);
}
try{
foreach(intiinconnections[indx]){
strings="Node"+i;
s.PadRight(7+i);
Console.WriteLine(s);
PrintProcess(i);
}
}catch(Exception){
Console.WriteLine("NodeHasnochilds");
}

}

publicstringGetProcess(){
//Returnsstringrepresentingtheprocess
stringprocess="[TASKS]";
foreach(TasktinControls){
if(tisStartTask){
process+="\n"+"[START_NODE,"+t.Left+","+t.Top+]";
}else{
process+="\n"+t.Index+","
+t.GetType().Namespace+"."+t.GetType().Name//allthisis"namespace,ClassName"identifier
+"","+t.Left+","+t.Top+","+t.Collapsed+', '+t.Data;
}
}
process+="\n[CONNECTIONS]";
```

```
foreach(KeyValuePair<int,List<int>>pairinconnections){
foreach(intnextinpair.Value){
process+="\n"+pair.Key+", "+next;
}
}
process+="\n[GROUPS]";
foreach(TaskSelectionRectanglerectinggroupRectangles){
process+="\n"
+rect.Rectangle.X+", "
+rect.Rectangle.Y+", "
+rect.Rectangle.Width+", "
+rect.Rectangle.Height+", "
+rect.FillColor.ToArgb()+", "
+rect.Title+", "
+"[";
foreach(Taskinrect.Children){
process+=t.Index+", ";
}
process.Substring(0,process.Length-1);
process+=']';
}
returnprocess;
}
privateRectangleGetAreaOfInterest(){
intminX=int.MaxValue,minY=int.MaxValue;//representsNegativeInfinity
){
minX=Math.Min(minX,rect.Rectangle.X);
minY=Math.Min(minY,rect.Rectangle.Y);
maxX=Math.Max(maxX,rect.Rectangle.X+rect.Rectangle.Width);
maxY=Math.Max(maxY,rect.Rectangle.Y+rect.Rectangle.Height);
}
returnnewRectangle(minX,minY,maxX-minX,maxY-minY);
}
privatevoidTranslateProcess(Pointtranslation){
//applythetranslationtogroups
foreach(TaskSelectionRectanglerectinggroupRectangles){
rect.Translate(translation.X,translation.Y,driver:null);
}
//applythetranslationtonongroppedtasks
foreach(TaskinControls.OfType<Task>()){
if(t.ParentGroup==null){
t.Location=newPoint(t.Location.X+translation.X,t.Location.Y+translation.Y);
}
}
origin.Y+=translation.Y;
origin.X+=translation.X;
}

publicvoidExportProcess(stringfile){
exporting=true;
SuspendDrawing(this);
//newlyaddedcode-----
```



```
foreach(TaskinControls){
t.Invalidate();//invalidateeverytasktoupdateitsbitmapforcorrectprintonpng
}
//newlyaddedcode-----

RectangleareaOfInterest=GetAreaOfInterest();
Pointtranslation=newPoint();
//Determineifprocesshasnegativecoordinatesandfindoutthetranslationneeded
if(areaOfInterest.X<0){
translation.X=(-1)*areaOfInterest.X+10;
}
if(areaOfInterest.Y<0){
translation.Y=(-1)*areaOfInterest.Y+10;
}
//applythetranslations
TranslateProcess(translation);

Bitmappp=newBitmap(8000,8000);//thebiggestpossiblebitmap.Wewillcropitlater
Graphicsgg=Graphics.FromImage(pp);
RenderBackground=false;
ReDraw(gg);
RenderBackground=true;
Bitmappp=pp.Clone(newRectangle(areaOfInterest.X+translation.X,areaOfInterest.Y+translation.Y,areaOfInterest.Wi
dth+20,areaOfInterest.Height+20),pp.PixelFormat);
p.Save(file,System.Drawing.Imaging.ImageFormat.Png);
p.Dispose();
pp.Dispose();
//rollbacktranslations
TranslateProcess(newPoint(-translation.X,-translation.Y));
exporting=false;
ResumeDrawing(this);
}
publicvoidEmptyDrawingBoard(){
ClearSelected();
groupRectangles.Clear();
selectedGroup=null;
foreach(TaskinControls){
t.OnTaskClicked-=OnTaskClicked;
t.OnCreateConnection-=OnCreateConnection;
t.OnDeleteConnections-=OnDeleteConnections;
t.OnTaskCollapsedChange-=OnTaskCollapsedChange;
t.OnTaskMouseDown-=OnTaskMouseDown;
t.OnTaskMouseUp-=OnTaskMouseUp;
t.OnTaskMoved-=OnTaskMoved;
t.Dispose();

}
origin=Point.Empty;
Controls.Clear();
connections.Clear();
GC.Collect();
ReDraw();
InvalidateAndUpdate();
```

```
}
privateTaskCreateStartTask(){
//Startnode
Taskstart=newStartTask();
start.Index=0;
start.Left=200;
start.Top=0;
start.OnTaskClicked+=OnTaskClicked;
start.OnDeleteConnections+=OnDeleteConnections;
start.OnCreateConnection+=OnCreateConnection;
start.OnTaskMoved+=OnTaskMoved;
start.OnTaskMouseDown+=OnTaskMouseDown;
start.OnTaskMouseUp+=OnTaskMouseUp;
returnstart;
}
publicvoidCreateProcess(List<string>instructions){
if(this.locked)return;
SuspendDrawing(this);
if(instructions==null){
return;
}elseif(instructions.Count<=0){
return;
}
if(GetStartTask()!=null){
prevTask=null;
}else{
Taskstart=CreateStartTask();
this.Controls.Add(start);
prevTask=start;
}

//Everyothernode
foreach(stringcomininstructions){
string[]sp=com.Split(' ');
stringcommand=sp[0];//.Substring(0,sp[0].Length);
Task;
switch(command){
case"lclick":
t=newClickTask();
((ClickTask)t).Button=MouseButtons.Left;
((ClickTask)t).Command=command;
break;
case"rclick":
t=newClickTask();
((ClickTask)t).Button=MouseButtons.Right;
((ClickTask)t).Command=command;
break;
case"double-click":
t=newClickTask();
((ClickTask)t).IsDoubleClick=true;
((ClickTask)t).Command=command;
break;
case"attach":
```

```
t=newAttachTask();
break;
case"type":
t=newKeyStrokesTask();
break;
case"terminal":
t=newOpenTerminalTask();
break;
default:
t=newTask();
Console.WriteLine("Taskwithcommand{0}doesnotexists.",command);
break;
}
t.Data=com;
if(prevTask!=null){
Tasktask=AddTask(t,newPoint(prevTask.Left,prevTask.Top+200),NextIndex);
ConnectTasks(prevTask,task);
prevTask=task;
}else{
prevTask=AddTask(t,newPoint(200,0),NextIndex);
}
}
ProcessChanged();
ResumeDrawing(this);
}
publicvoidLoadProcess(stringinstruction){
SuspendDrawing(this);
EmptyDrawingBoard();
Index=0;
if(instruction==" "||instruction==null){
ReDraw();

ResumeDrawing(this);
this.Invalidate();
this.Update();
return;
}
string[]process=instruction.Split('\n');
boolreadingTasks=true;
boolreadingGroups=false;
foreach(stringcominprocess){
if(com.Equals("[TASKS]")){
readingTasks=true;
readingGroups=false;
continue;
}
if(com.Equals("[CONNECTIONS]")){
readingTasks=false;
readingGroups=false;
continue;
}
if(com.Equals("[GROUPS]")){
readingTasks=false;
```

```
readingGroups=true;
continue;
}
if(com.StartsWith("[START_NODE"]){
if(GetStartTask()!=null){
}else{
Taskt=CreateStartTask();
string[]pos=com.Split(',');
t.Left=Int32.Parse(pos[1]);
t.Top=Int32.Parse(pos[2].Substring(0,pos[2].Length-1));
Controls.Add(t);
prevTask=t;
}
continue;
}
string[]details=com.Split(',');
if(readingTasks&&!readingGroups){
try{
intindex=Int32.Parse(details[0]);
Typeptp=Type.GetType(details[1],true);
Tasktool=(Task)Activator.CreateInstance(tp);
tool.Data=details[5];
Taskt=AddTask(tool,newPoint(Int32.Parse(details[2]),Int32.Parse(details[3])),index);
if(Index<index){
Index=index;
}
t.Collapsed=bool.Parse(details[4]);
}catch(FormatException){
MessageBox.Show(
"Errorwhileloadingprocess.\nOneormoretasksaremismisconfigured.Theywillberejectedfromtheprocess."
,"Taskscorrupted",MessageBoxButtons.OK,MessageBoxIcon.Error);
}catch(TypeLoadException){
MessageBox.Show(
"Errorwhileloadingprocess.\nOneormoretasksaremismisconfigured.Theywillberejectedfromtheprocess."
,"Taskscorrupted",MessageBoxButtons.OK,MessageBoxIcon.Error);
}
}elseif(!readingTasks&!readingGroups){
intstart=Int32.Parse(details[0]);
intend=Int32.Parse(details[1]);
ConnectTasks(start,end);
}else{
TaskSelectionRectangle=newTaskSelectionRectangle(){
Rectangle=newRectangle(int.Parse(details[0]),int.Parse(details[1]),int.Parse(details[2]),int.Parse(details[3])),
FillColor=Color.FromArgb(int.Parse(details[4])),
BorderColor=Color.Black,
Selected=false,
Title=details[5]
};
string[]children=details[6].Substring(1,details[6].Length-2).Split(';');
foreach(stringiinchildren){
if(i==""){
continue;
}
}
```

```
Taskt=GetTaskByIndex(int.Parse(i));  
if(t==null){  
continue;  
}  
g.Addchild(t);  
t.ParentGroup=g;  
}  
groupRectangles.Add(g);  
}  
}  
ReDraw();  
ResumeDrawing(this);
```