BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

APPLIED AUTOMATION BETWEEN SECURITY NETWORK COMPONENTS IN OPERATIONAL NETWORKS

APLIKOVANÁ AUTOMATIZACE MEZI BEZPEČNOSTNÍMI SÍŤOVÝMI PRVKY V PROVOZNÍCH SÍTÍCH

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Jakub Škoda

SUPERVISOR
VEDOUCÍ PRÁCE

Ing. Anna Kubánková, Ph.D.

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Bachelor's Thesis

Bachelor's study program **Information Security**

Department of Telecommunications

Student: Jakub Škoda ***ID:*** 211813

Year of study: 3 ***Academic year:*** 2021/22

TITLE OF THESIS:

Applied automation between security network components in operational networks

INSTRUCTION:

Study possibilities of automatization of security network components from Cisco and other vendors. Identify obsolete manual procedures associated with threat identification and with transfer of information context between security technologies in complex networks. Design automated scenarios to solve the shortcomings of manual procedures. Apply (implement) automated scenarios in a production environment on real devices. Describe the benefits of automated processes over traditional solutions.

RECOMMENDED LITERATURE:

[1] ALY, Bassem. Hands-On Enterprise Automation with Python.: Automate common administrative and security task. Packt Publishing, 2018. ISBN-13: 978-1788998512

[2] GOOLEY, Jason a Chris JACKSON. Cisco Certified DevNet Associate DEVASC 200-901 Official Cert Guide. Pearson Education, 2020. ISBN: 0136642969.

Date of project specification: 7.2.2022 ***Deadline for submission:*** 31.5.2022

Supervisor: Ing. Anna Kubánková, Ph.D.
Consultant: Ing. Jan Šimůnek, ALEF NULA, a.s.

doc. Ing. Jan Hajný, Ph.D.
Chair of study program board

ABSTRACT

This thesis deals with studying the possibilities for automation in network security. Identify a manual workflow and study the possibilities of how to design an automated replacement that will make it obsolete. In the new automated workflow, Cisco Umbrella and its DNS reporting and assessment tools for endpoint devices were used to create a new security metric, behaviour. The endpoint behaviour is then used on a selected use case of Remote Access VPN. Instead of using a Compliance check before establishing the VPN connection, the endpoint's behaviour is used instead.

KEYWORDS

Automation, Network, Cisco, Umbrella, Compliance, Behaviour, Firewall, VPN, Endpoint, Python, PostgreSQL, FreeRADIUS, Ubuntu, API, Git,

Extended abstract

Cílem této bakalářské práce bylo sezámit se s možnostmi automatizace bezpečnostní prvnků výrobce Cisco a třetích stran. Identifikovat možné manuální postupy spojené s identifikací hrozeb a předáváním kontextu mezi bezpečnostními prvky v provozních sítích. Následně navrhnout automatizované řešení, které by nahradilo překonaný manuální postup a poté toto řešení aplikovat na reálné zařízení.

V teoretickém úvodu se čtenář dozvídá základní informace o informační bezpečnosti a o tom, jak důležitou roli hraje v každodenním životě. Jsou zde také popsány dopady kybernetických útoků a jejich rapidní vzrůst během pandemie COVID-19. Dále se čtenář seznamuje s bezpečností koncových bodů a jejich rolí v dnešním produkčním prostředí.

V kapitole 1.2 se čtenář také dozví, jaké jsou rozdíly mezi Compliace a Behavior.

V další teoretické části se představuje výběr platformy. S každým produktem a nástrojem čtenáře obecně seznámí. Mezi použité nástroje patří Cisco ASA, Cisco Umbrella, Ubuntu a další. Vysvětlí jejích možnosti a odůvodní proč byl jednotlivý produkt a nástroj vybrán.

V první praktické kapitole se čtenáři představí vybraná manuální procedura. Vybraná manuální procedura je velmi časově náročná a v zásadě ji není nemožné bez automatizace vykonávat. Procedura spočívá v přidání další vrstvy ochrany do existující sítě za pomocí parametru chování samotné koncové stanice a Cisco Umbrella, která analyzuje DNS requesty koncových stanic. Na základě reputační databáze, AI a jiných nástrojů Cisco Umbrella může identifikovat škodlivé domény a zablokovat je. Tato funkce je sama o sobě skvělá, ale má své limity. Abychom tyto limity vyřešili musíme vytvořit novou vrstvu bezpečnosti, která bude využívat chování jako indikátor. Po podrobném popisu manuální procedury se čtenář přesouvá k samotnému řešení práce.

Na základě chování koncového bodu můžeme určit, zda je bezpečné připojení koncového bodu k interní síti či nikoliv. Pak můžeme použít tuto novou metriku pro vybraný nový způsob použití. Pro nás to bude VPN se vzdáleným přístupem. Automatizovaný postup splňuje všechny úkoly, které by administrátor musel dělat ručně s některými dalšími funkcemi.

Budeme vycházet z předpokladu, že existuje běžící databáze PostgreSQL s již před vyplněnými uživatelskými jmény, hesly a přidruženými osobními počítači pro každého uživatele. Koncová zařízení budou generovat provoz DNS, který bude odeslán a zpracován Cisco Umbrella. Jednou denně bude spuštěna systémová služba na serveru Ubuntu, vytvořená pro spuštění automatické aplikace. Tato aplikace načte všechny uživatele a jejich koncová zařízení. Dále bude filtrovat API dotazy

podle požadavků DNS z Umbrella podle požadovaných koncových zařízení a vybraných kategorií. V našem případě bezpečnostní kategorie a vynechání whitelistovaných uživatelů. Poté budou všechna tato data z rozhraní Umbrella Reporting API stažena a analyzována. Po provedení výpočtu, pro každý DNS request, bude prostřednictvím Umbrella Investigate API rozdělen do tří kategorií: Bezpečný, Problematický a Nebezpečný. Tyto spárované informace budou poslány do databáze. Následuje používaný postup uživatele. Uživatel, který se chce připojit do firemní sítě, používá svého VPN klienta. Po vyplnění se inicializuje Compliance kontrola uživatelského jména a hesla. Cisco ASA, která přijala požadavek na vytvoření VPN spojení, odešle serveru RADIUS request paket. Server FreeRADIUS pak hledá v databázi záznamy uživatelů. Server poté odešle odpověď RADIUS reply s dalšími parametry jako jsou: kategorie oprávnění uživatele a jedna ze tří kategorií definujících chování koncové stanice. ASA se poté podívá do své konfigurace, aby zjistila, co má dělat s přijatou RADIUS reply zprávou.

Výsledkem celé práce je, že jsme úspěšně překonali manuální postup s automatizací. Celé workflow je funkční. Koncové stanice uživatelů průběžně generují DNS requesty, které se automaticky zasílají do Umbrella. Periodicky spouštěná aplikace načte údaje z Umbrelly a vypočítá skore chování koncové stanice. Tento údaj se následně pošle do databáze. Při pokusu uživatele vytvořit VPN spojení se ASA zeptá FreeRADIUS server, jestli uživatel existuje a jaké má parametry. FreeRADIUS odpoví s daty z databáze. Pokud je parametr chování „nebezpečný“, tak je spojení s uživatelem ukončeno a je mu zobrazena zpráva, že má kontaktovat IT oddělení. Pokud je ale parametr chování „bezpečný“, tak je uživateli umožněno navázat dotazované VPN spojení. Pokud se administrátor po inspekci rozhodne zablokované uživatele odblokovat, má tuto možnost pomocí white list aplikace, která zařadí uživatele na určitou dobu na white list, který ho bude vyčleňovat z workflow aplikace.

ŠKODA, Jakub. *Applied automation between security network components in operationalnetworks*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications, 2022, 60 p. Bachelor's Thesis. Advised by Ing. Anna Kubánková, Ph.D.

Author's Declaration

Author:	Jakub Škoda
Author's ID:	211813
Paper type:	Bachelor's Thesis
Academic year:	2021/22
Topic:	Applied automation between security network components in operationalnetworks

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank the advisor of my thesis, Ing. Anna Kubánková, Ph.D., for her valuable comments, help, support, and professional suggestions. I also thank Ing. Honza Šimůnek for his leadership in this journey. Lastly, I thank Michal Svačina for his support and insights into the production environments.

Contents

List of Figures

List of Tables

Listings

Introduction

Security is today an indispensable topic for everyone. For the end-user, small businesses, especially big companies that store sensitive data. We live in a time of constant security breaches, data leaks and maliciously encrypted files and devices across all industries. The number of threats is rising every year, and everyone has to take some security measures to ensure that their data and devices are secure. Companies are investing more and more resources to keep their data and networks secure. From 2021 Report from IBM on Cost of Data breach [1] We see that breach costs have been on the rise for several years, despite the fact the Global Cybersecurity spending is exceeding $1 trillion from 2017-2021[2]. The amount of criminal activity we see is forcing the industry to spend tremendous amounts of money that analysts cannot accurately track. The impact of COVID-19 on cybersecurity created new challenges for businesses as they had to adapt to a different mode of working from home. This new standard was responsible for a big leap in digital transformation, and more security concerns as work from home bring its challenges to the security teams.[3]

Security teams have many tools and policies to manage and keep secure their assets. Incident response is becoming a very time consuming and repetitive without automation. The lack of automation and visibility in day to day operations is a hardship for 57% professionals that were surveyed.[4]

In this paper, We describe the creation of an open-source automation tool developed for administrators or security engineers that use Cisco platforms. This tool automates set procedures for checking DNS queries of monitored devices for their potential harmfulness. With the utilisation of Cisco Umbrella Reporting and Investigate capabilities for added insight over the DNS queries from monitored endpoints, this tool is able to bring the behaviour of these endpoints to the security industry.

Security is today an indispensable topic for everyone. For the end-user, small businesses, especially big companies that store sensitive data. We live in a time of constant security breaches, data leaks and maliciously encrypted files and devices across all industries. The number of threats is rising every year, and everyone has to take some security measures to ensure that their data and devices are secure. Companies are investing more and more resources to keep their data and networks secure. From 2021 Report from IBM on Cost of Data breach [1] We see that breach costs have been on the rise for several years, despite the fact the Global Cybersecurity spending is exceeding $1 trillion from 2017-2021[2]. The amount of criminal activity we see is forcing the industry to spend tremendous amounts of money that analysts cannot accurately track. The impact of COVID-19 on cybersecurity created new challenges for businesses as they had to adapt to a different mode of working from home. This new standard was responsible for a big leap in digital transformation, and more security concerns as work from home bring its challenges to the security teams.[3]

Security teams have many tools and policies to manage and keep secure their assets. Incident response is becoming a very time consuming and repetitive without automation. The lack of automation and visibility in day to day operations is a hardship for 57% professionals that were surveyed.[4]

In this paper, We describe the creation of an open-source automation tool developed for administrators or security engineers that use Cisco platforms. This tool automates set procedures for checking DNS queries of monitored devices for their potential harmfulness. With the utilisation of Cisco Umbrella Reporting and Investigate capabilities for added insight over the DNS queries from monitored endpoints, this tool is able to bring the behaviour of these endpoints to the security industry.

1 Endpoint security

We have seen a giant leap in endpoint security in the last two years due to the Covid-19 pandemic. Employees had to work from home, and administrators were forced to shift their focus domain. Endpoint security protects connections of various endpoints such as laptops, smartphones, IoT devices and others against attack and security threats. These endpoints access corporate networks and sensitive data, and it is crucial that they are secure and meet corporate set policies as well as standards.

In the beginnings of endpoint security, there were simple antivirus programs that had some predefined set of signatures. Today endpoint security is much more advanced using next-gen antivirus, threat detection with the investigation and threat response, device management and much more.[14]

1.1 Established practice in endpoint security

The established practice in endpoint security is to ensure that the company's assets are sufficiently protected while user restrictions are kept to a minimum. We can see this applied, for example, in the scenario when the employee on a home office with his personal company-issued computer wants to connect to the company's internal network. To do this, he will use a VPN tunnel. The VPN client application has pre-set requirements that the user has to comply with in order to be connected. Some companies do not have any special compliance requirements. Only a username and password and user is connected. Today this would be considered a security risk. That is why as a standard practice minimum, two-factor authentication is required. Then with the increasing value of the assets, more requirements are added. For example, in bank institutions, the VPN client has to check many parameters before the user is authorised to access internal resources: if an antivirus program is present with a recent malware-free scan of the machine, if valid security profiles are present and many more. Also, in these types of applications, a full-tunnel VPN is used, so a potentially malicious home network of the user is made of reach for the endpoint computer because all of the user's traffic and IP addresses are forwarded through the company and it's firewall and other security appliances.

1.1.1 Risks of decentralised operations

The decentralisation of operations is bringing many risks that need to be addressed. Work from home brings risks such as insecure home networks, using personal devices for work, unencrypted file sharing, etc. The problem with decentralisation continues with the use of hybrid clouds bringing connection risks and Service Level Agreements

placing the whole security question on a third party. New security measures are needed to maintain the risks created by decentralisation. One of the risks is handled by this thesis endpoint security.

1.1.2 Future of endpoint security

As endpoints constitute an abstract gateway into the IT environments for users, neglecting their protection will lead to a disaster. Nevertheless, a big question arises, how will the future of endpoint security unfold. Instead of relying on classic approaches such as compliance checks, we could use AI and machine learning to create new monitoring tools and products that will focus on user behaviour, look out for malicious attempts on security and take action to thwart them. This new approach could mean a massive leap in endpoint security.

1.2 Behaviour vs Compliance

Today in cybersecurity and IT operations, compliance is an important topic. Every device, server, or user himself has to meet set requirements by the company to make the assets of the said company safe. From the psychological point of view, compliance refers to changing one's behaviour due to the request or direction of another person. In the technological space, compliance is more associated with complying with pre-set rules by the IT security department. While the person has the option to refuse the request, most of the times, he chooses to comply with per-set rules required so he can use tools, access a network etc. However, one aspect that is complying with set rules does not cover the behaviour itself of an employee, which is so crucial to any organisation's security. Today security is most often addressed by some type of compliance rules. For example, when a user wants to use the company laptop, he has to comply with rules such as: having a strong password, BIOS lock, encryption on the storage, and restrictions on options to execute, write and read files. If the user does not want to comply, he cannot have the company laptop. The reason why user behaviour was not addressed as much as compliance is because it was not possible to track, process and evaluate the user behaviour of every employee in the company. Fortunately, thanks to the progress in computation power, machine learning, and artificial neural networks, the factor of user behaviour is becoming a parameter that we can process and use in security operations.

1.2.1 Compliance

As mentioned before, compliance of users and endpoint devices in a production environment is handled by checking if said user/endpoint meets set requirements. Common practice is to authorise the user before giving him access to the network or resources. In the authorisation process, we check the compliance. Only if the user and his endpoint device meet all requirements in the compliance check is he authorised. To commonly observed attributes in the production environment include: 1.1

Attributes	Checks
Antivirus	Signatures
Operating system	Last version with knows vulnerability fixes
Personal Firewall	Enabled/Disabled
Disk encryption	Enabled/Disabled

Tab. 1.1: Compliance attributes

1.2.2 Behaviour

The use of compliance in the real world has its limitations. The workflow of checking if an endpoint device is compliant with internally set attributes is not always possible. Let us say that employee of an external company that needs to use the internal resources wants to connect to the internal network. In an ideal scenario, he would have to comply with the exact compliance check as an internal employee. This is not always possible. The internal IT department does not have management access to the endpoint devices of the external company. Therefore even if the external endpoint would have some attributes right in the compliance check, definitely not all due to the different device management. Some of the possible attributes to look out for when we are talking about behaviour are 1.2:

Attributes	Checks
Detected malware	Threat score (low/medium/high)
Blocked DNS requests by security tool	Threat score (low/medium/high)
IPS incidents on firewall connected to specific host	Threat score (low/medium/high)

Tab. 1.2: Behaviour attributes

Having the external company update, keep signatures and any other attribute is nearly impossible. That is why we need to find other ways to keep security. One of the ways is the behaviour of the user and his endpoint device. To make use of the behaviour, we need to select a platform on which we will be able to demonstrate the use of behaviour in network security in real life.

2 Platform selection

The selection of a platform is essential for creating an effective solution. Today many devices are being used outside of the internal network. The amount of corporate data traffic that bypasses the security perimeter has skyrocketed due to the forced work-from-home strategy with the start of the COVID-19 pandemic. The use of RDP (Remote Desktop Protocol) and VPN (Virtual Private Network) has seen an extensive rise of 41% and 33% respectively, in the first month of the outbreak [5]. This is why solutions such as Cisco Umbrella, which is based on DNS protection, are an essential part of the fight against open vulnerabilities that could be used for an attack.

In the past, traditional enterprise network architecture used only an on-premise model. Closed firewalls made access to the internal network from outside impossible. Today, in the time, of decentralisation, cloud services and RDP, the site perimeter is not defined. That is why we have to find new ways of protection and security. One of the many ways is endpoint security.

2.1 Automation and security products

Automation has become an essential part of all industries, including networking and cyber security. One of the most significant issues for network operation is the IT cost. The growth of data, devices, and traffic is outpacing the capabilities of IT departments rendering manual solutions nearly impossible to deploy. Cisco estimates that up to 95% of network changes are still performed manually, resulting in much higher operating costs compared to the cost of the network itself. As a result, companies that do not use centralised and remotely controlled automation in their operations will not be able to stay relevant to the market [6].

2.1.1 Third party automation and security products

The issue of decentralisation of workflows is taken very seriously in the network security industry. That is why companies are creating new ways and products to ensure communication and assets are secure.

There are many security products from a number of vendors such as Cisco, Fortinet, Check Point, Palo Alto Networks, IBM and many more. Our interest lies in the so-called Next-gen solutions and products. Among the most known is a next-gen firewall, which in contrast to traditional firewalls, can block modern threats such as malware, and application-layer attacks and bring Intrusion Prevention System (IPS) and VPN concentrators as an essential part of the decentralisation in operation

workflows. Then we have products that provide security in cloud services and hybrid cloud communication, and closest to us regarding this thesis, Automation security products.

2.2 Chosen solutions and tools used

Based on the chosen hardware platform, we must select a suitable programming language and application built on it. The core communication with Cisco services and tools is being done through APIs. Those can be accessed through a wide variety of tools and programming languages.

2.2.1 Cisco ASA

Adaptive Security Appliance (ASA) is Cisco's firewall product. ASA offers many firewall capabilities in many form factors - standalone appliances, blades for server applications, and virtual appliance used in public, private clouds and software-defined networks. Cisco has a new line of firewalls (Firepower) that is a complete next-get firewall with an IPS system and other new functions. The ASA code is still widely used for its low cost and excellent VPN capabilities that are used in this thesis.

2.2.2 Server - VMWARE VM_Ubuntu

As part of the solution is also a virtual server that handles communication with the Cisco Umbrella, firewall and database. The server runs a Linux distribution Ubuntu 20.4 as its operating system under the hypervisor of VMware ESXi. The configuration of the server and a more in-depth look is discussed in the implementation part of this thesis 4.6

Cron

To make the service (application) created in this thesis automated. We need to use some sort of job scheduler. Cron is a time-based job scheduling daemon found in many Unix systems such as Linux distribution. In our case, Cron handles automated job scheduling for the created application service 4.7 on a daily basis.

2.2.3 Programming language

The language used plays a vital role in the whole project. To meet the requirements, our candidate must be able to operate in a remote environment such as a full OS,

cloud or in a docker container. Also, it has to be easy to understand for it to be applied by other admins worldwide.

Python

Python is class-leading programming and scripting language that is used worldwide for all types of scenarios. Due to the fact that it also meets the above conditions, Python version 3 will be used to create the app for this automation.

Libraries

Two extra libraries were used in the python program: sqlalchemy, psycopg2-binary **sqlalchemy** is a python SQL toolkit and Object Relation Mapper. It provides high-performing database access with the user-oriented Python language in mind. **psycopg2-binary** is the most popular PostgreSQL database adapter for Python. It was designed to run on multi-threaded workloads. Implementation in C results in a very efficient and secure adapter.

2.2.4 API

Application Programming Interface is a set of functions, procedures, protocols and libraries that programmers worldwide use to create applications and other software. Today API is mostly used in creating mobile and web applications. The main function of an API is to create an interface for a connection between two devices or programs. [15]

REST API

All APIs that were used in this paper are REST APIs. The difference between API and REST API is that API is a set of functions and procedures that allow other computers or a program to access some feature of another application. REST is an architectural style developed for network applications. The primary communication is done in for of sever-client. The most significant advantage of REST is that the guidelines that define REST help build faster and easy to understand for third-parties.[16]

oauth2

oauth2 is an open standard security protocol that is very often used as a standard way of handling third-party access to an application without giving it the access credentials. Cisco Umbrella Reporting API uses this standard to generate a unique token that has a 1hour duration and is used to access the API.

2.2.5 Data formats

Choosing an appropriate data format is a crucial part of the whole project. APIs support many data formats. Some are used more frequently, and some are easier to use.

XML

XML data format was defined in 1996 and is used to this date. XML became very popular across all programming fields, but most importantly, it affected how the Internet we know works today. The syntax of XML is more challenging to use for humans. That is why other data formats emerged.

JSON

JSON (JavaScript Object Notation) is an XML open standard file format that is readable by humans and computers alike. The main difference between XML is its much more user-friendly readability and useability, as we can see in these examples: [17] This is why JSON was chosen as the used data format.

```
<employees>
  <employee>
    <firstName>Ema</firstName> <lastName>Rudiger</lastName>
  </employee>
  <employee>
    <firstName>Liam</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>James</firstName> <lastName>Brown</lastName>
  </employee>
</employees>
```

Source Code 2.1: XML Example

```
{"employees":[
  { "firstName":"Ema", "lastName":"Rudiger" },
  { "firstName":"Liam", "lastName":"Smith" },
  { "firstName":"James", "lastName":"Brown" }
]}
```

Source Code 2.2: JSON Example

2.2.6 AAA

Authentication, authorisation, and accounting is a security framework of services and protocols for controlling access to computer resources, auditing usage, enforcing policies, and providing the necessary information to bill for services. To this day, AAA is an industry-standard for authenticating users or machines to network services.

Authentication

The core of authentication is proving that the user is who he says he is. This is handled by the user providing credentials to create an identity. This identity is then compared by a AAA server with its database of stored information for said user. The server checks the username, password, and other parameters that are stored in the database. There are three types of authentication:[7]

Types	Examples
Something a person knows	Password, PIN
Something a person has	Swipe card, keys, USB stick
Something a person is	Fingerprint, face identification

Tab. 2.1: Authentication attributes

Authorization

Authorisation involves checking what the identity that has been authenticated has access to. User can be granted privileges to access certain parts of a network, files or a system. The permissions that are available to the user are stored in a database along with the user's identity. For example, the administrator will have different authorisation from a user and user from a guest. The administrator will have the right to create, edit and execute files. User will be able to read them, and guest will not have access to these files at all.

Accounting

The final part of the AAA framework is accounting, which keeps track of user activities while the user is logged in to a network. This can include the amount of time or data that a user used. Then the data is stored in logs for future use in administration, such as billing, analysis, and much more.

RADIUS

Several protocols use the elements of AAA to ensure identity security. One of which is the RADIUS protocol. Like many others, RADIUS uses a client-server model. Client NAS (Network Access Server) sends a request to a RADIUS server. The RADIUS server then processes the request and sends back a response. In the production networks RADIUS server is combined with a database such as MySQL, LDAP and more. An example of the RADIUS protocol would-be users who want to connect to a Wi-Fi AP. The user will perform authorisation, and the Wi-Fi AP (NAS in the RADIUS protocol) will send a request to the RADIUS server. The server then checks with the database if he exists. If so server responds to the request by either accepting it, rejecting it, or challenging it by asking for more information. [8]

FreeRADIUS is one of many RADIUS server products. While many are paid products such as Microsoft NPS and Cisco ISE. FreeRADIUS is, on the other hand, open-source. Since the whole project is developed in the C programming language, the server itself is very fast. In addition, FreeRADIUS has many modules that enable the integration of vast production environments. This enables easy deployments in extensive networks with a result of 100 million users per day around the world. [9]

2.2.7 Database

A database is an organised collection of data/information stored and accessed electronically in a computer system. The database design consists of a suitable data model, data representation, query languages, security, and privacy depending on its use. One of the most popular databases is a structured query language (SQL). SQL is used by most relation databases worldwide thanks to the SQL ANSI standard and many extensions from renowned companies such as IBM, Microsoft, and Oracle. [10]

PostgreSQL

PostgreSQL is an open-source object-relational database system that uses and extends SQL language. Runs natively on all major operating systems. It fully meets the ACID requirements and supports many data types, including JSON, XML, HSTORE, Geo-spacial and much more. Thanks to our previous experience with this database system and its use in a wide variety of enterprises across many industries, it was chosen as the best choice. [11]

pgAdmin4

The primary way of administration on PostgreSQL database is **psql** command-line program. It enables to enter SQL queries, write shell-like scripts... However, due to

the ability to use a tool with a graphical user interface, such a tool was chosen.

pgAdmin4 is a database administration tool for PostgreSQL with a graphical user interface. pgAdmin4 has many deployment options, such as a stand-alone desktop application or just the handler with a WEB interface. [12]

2.2.8 Umbrella

Cisco Umbrella was chosen for its security assessment capabilities, easy implementation into an existing production environment and workflow. Umbrella provides the first line of defence outside of the network for users on the go. It enables monitoring of all users' activity and blocks threats before they reach the internal network or other endpoints. Umbrella is an open cloud platform with easy integration to an already deployed security stack. Umbrellas Investigate module delivers live threat intelligence by analysing and learning from internet activity patterns and sourcing information from Cisco Talos, and other 3-rd party web reputation feeds.

The core component of Umbrella is monitoring Domain Name System (DNS) requests. When Umbrella receives a DNS request, it uses machine learning to resolve if the request is malicious, risky or safe.[13]

Risk Score

Umbrella Investigate module that is used in this solution, offers the use of a risk score attribute for a domain address. This attribute is calculated my selected security indicators such as: 2.2

Indicator	Checks
Geo Popularity Score	Measures the geographical request pattern in comparison to typical domains
Keyword Score	Measures the similarity of keywords in this domain to keywords found in malicious domains
Lexical	Measures the lexical similarity of this domain to malicious ones
TLD Rank Score	Ranks top level domains based on the amount of malicious activity they receive

Tab. 2.2: Umbrella Investigate - Risk score indicators

Machine learning

Machine learning that is used in Cisco Umbrella is a core part of any modern network security system. It uses statistics to find patterns in a large amount of data. Then the learning algorithm "learns" between good and bad files and behaviours. Machine learning in endpoint security helps find unusual patterns and vectors for potential malware attacks.

Cisco AnyConnect Secure Mobility Client

One of the many solutions available for endpoint security is the Cisco AnyConnect Secure Mobility Client. This solution is deployed on every endpoint PC in the company network. It offers to check if the endpoint device is compliant with many attributes such as BIOS version, antivirus check, up to date signatures and many more. The attribute used in the solution is from the Roaming Security module that handles sending DNS requests to Umbrella for future analysing.

2.3 Correlation of information

The correlation of information is a vital part of today's network security platform. Systems that maintain network safe like firewalls, IDS, Vulnerability scanning systems, SASE and other safety systems generate a lot of event information and logs. This information would not be used to its full potential if it were stored only on the data storage of its device. Administrators would be flooded with repeated alarm data that would be impossible to sort and act on them accordingly. This is why Automation and Correlation of information are so important. In this thesis, we use the risk score from Cisco Umbrella that is calculated with the help of a correlation of information about known vulnerabilities across the world. [18]

2.3.1 CVE Mapping

The Common Vulnerabilities and Exposures list is publicly known information-security vulnerabilities and exposures. Every flaw is assigned its unique CVE number by which it is identified. Then a short description of the flaw with the affected systems. Finally, a score that describes how serious the vulnerability is. The most important criteria for CVE are that the flaw has to be fixable and acknowledged by the affected vendor or documented. Then CVE entry is created so that everyone from an attacker to the affected business may respond.[19]

2.3.2 Mapping of know attack vectors

The attack vector is the path that attacker uses to gain access to a device or a network that he has as the target. Attackers choose their attack vectors according to found vulnerabilities. There is an excellent knowledge of common attack vectors in the hacking community as well in between security specialists. Mapping these attack vectors is a crucial task that has to be done to secure the potentially vulnerable application or network.

3 Manual procedure

3.1 Manual procedure, which will be surpassed by automation

The goal of this thesis is to identify obsolete manual procedures. Design an automated scenario to solve the shortcomings of said manual procedure and apply this automated scenario in a production environment on real devices. The manual procedure chosen to automate is very tedious and would be very labour intensive for the administrator. The procedure consists of adding another layer of security to the network through the behaviour of the endpoint device. Based on its reputation database, AI and other tools, Umbrella can identify malicious domains and block them. This itself is a very useful layer of added security. However, it has its limitations and therefore, to address these limitations, we need to create a new layer of security that will use behaviour as its indicator.

The current procedure for an administrator that would like to add DNS-layer security through the Cisco Umbrella. After deploying Umbrella on the endpoint device administrator, Umbrella can monitor and block DNS traffic. Nevertheless, this automated type of blocking has its drawbacks. Umbrella can generate false positives, and it is up to the administrator to manually check the reported DNS lookups in the Umbrella reporting module 3.1, copy the domain destination address, then copy this address to the Investigate module, enter the address and then check all the details 3.2 and verify if it is indeed correctly locked malicious DNS lookup or if it is false positive. Therefore, manually evaluate the information from Investigate module and assess potential risk. With the assessment for the endpoint, the administrator then has to somehow add the endpoint to a list of blocked devices. When he decides that he wants to unblock the endpoint, he has to unblock him manually.

This is a very tedious procedure with many problems. The administrator has to check manually for any potentially harmful DNS requests. Then he has to check if those requests are actually harmful. Assess the behaviour for the endpoint. Manually block it, keep track of when he blocked the endpoint (if he wants to block the devices for a set amount of time) and then remember to unblock him. In reality, this procedure is, without automation, nearly impossible to do on a regular reoccurring basis.

In the following screenshots, we can see the Umbrella Reporting module output with individual DNS requests. In a production environment, there would be ten thousand requests per day. Manually copying the "Destination" addresses of each DNS request, pasting them to the Investigate module and checking each one is

impossible. That is why a solution that will automate this process and solve the unusability of the Investigate data for this problem was created.

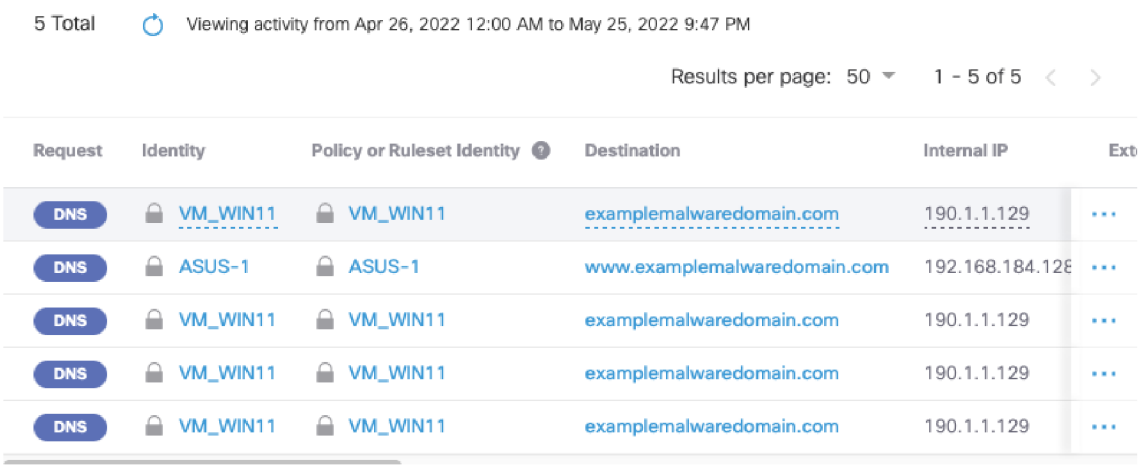

Fig. 3.1: Implementation - Cisco Umbrella Reporting

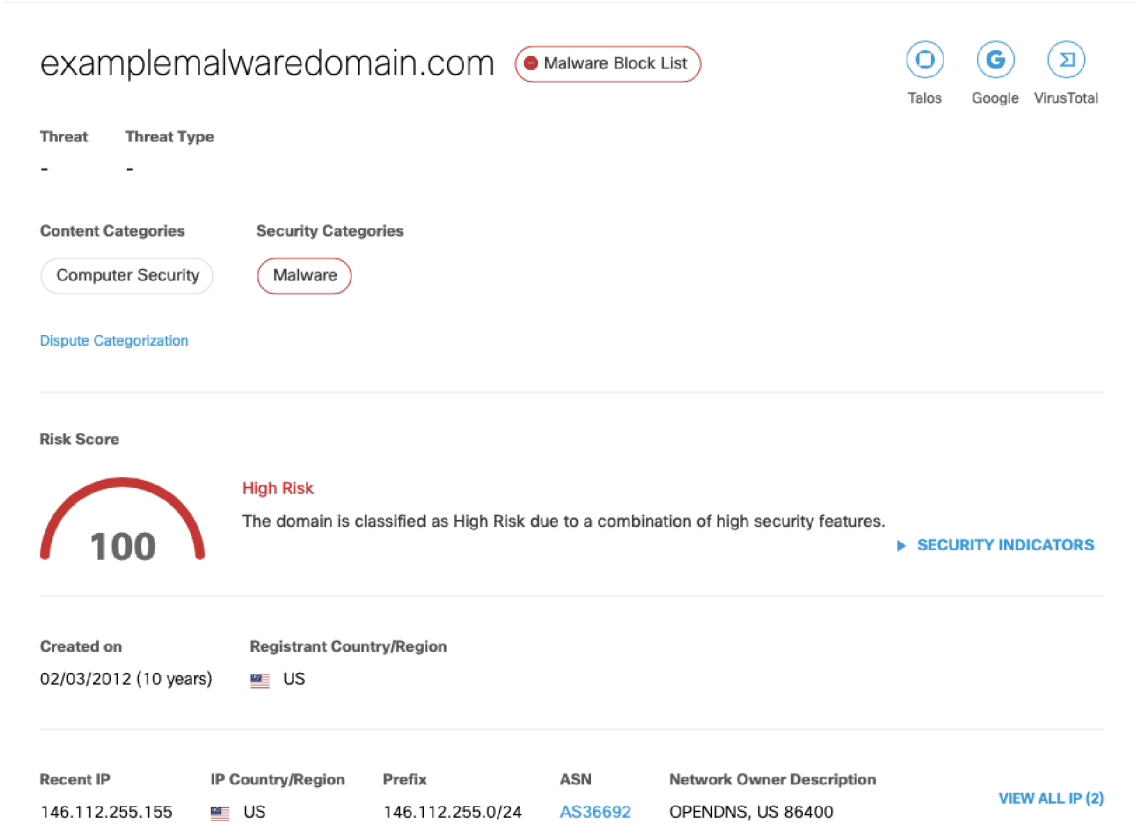

Fig. 3.2: Implementation - Cisco Umbrella Investigate

4 Implementation details

4.1 Solution

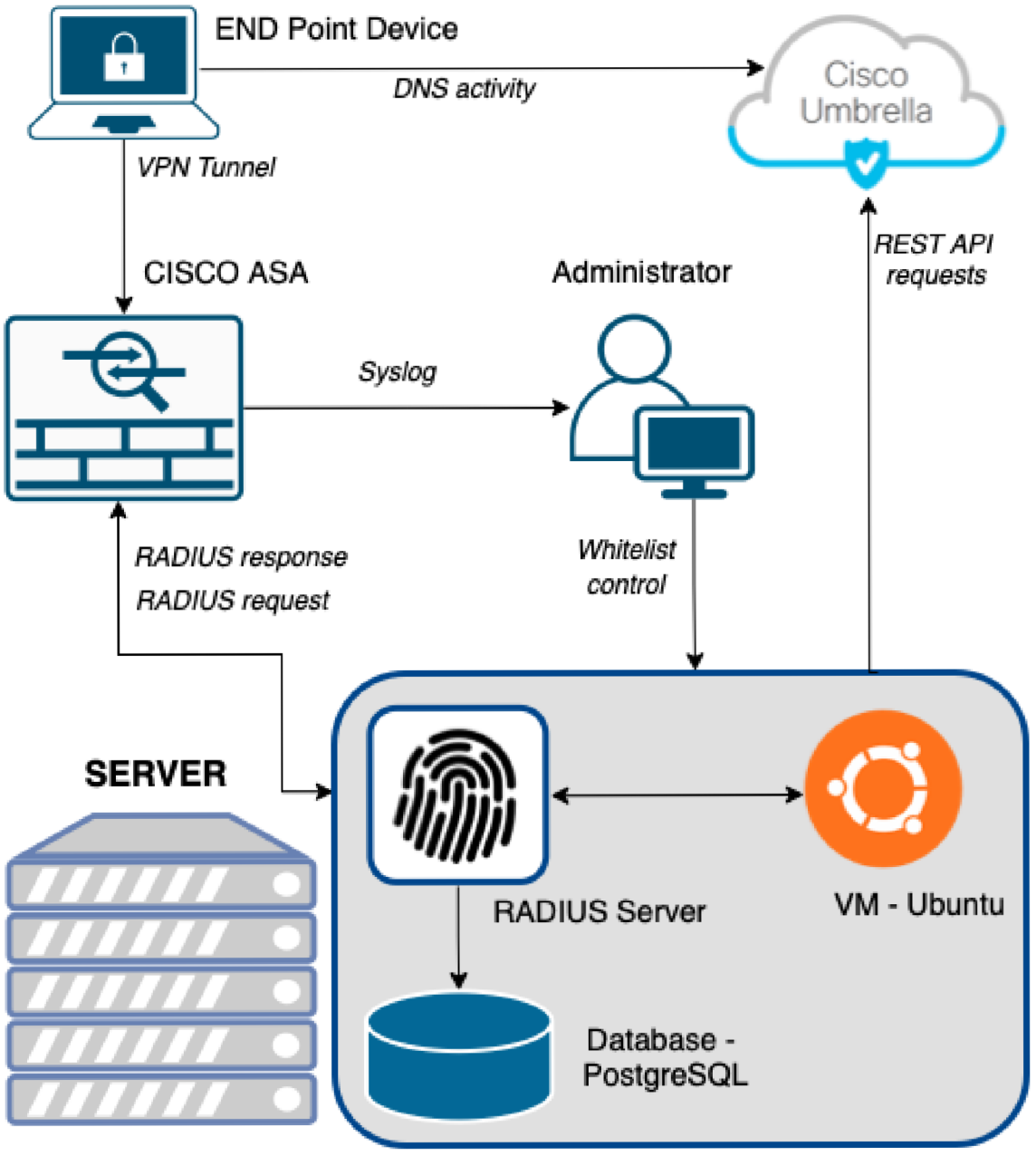

Fig. 4.1: Diagram of the solution

As we know, the currently used approach of checking the compliance of a device has its limitation. Nevertheless, with a new automated approach, we can utilize Umbrella and its DNS lookup assessment to create a new security metric, **behaviour**. With the endpoint's behaviour, we can determine if an endpoint is safe to be connected to the internal network or not. Then we can use this new metric for a new use case. For us, it will be a Remote Access VPN.

The automated procedure fulfils all tasks that the administrator would have to do manually with some additional features. Also, all query time frames mentioned in this paragraph are easily changeable in a configuration file (more on it later).

We will proceed from the assumption that there is a running PostgreSQL database with usernames, passwords and associated personal computers for each user. Endpoint devices will generate DNS traffic that will be sent and logged in Cisco Umbrella. Once a day, a system service on the Ubuntu server created to run the automated application will be launched. This application will load all users and their endpoint devices. Next on, will be filtering the DNS requests from Umbrella by desired endpoint devices and categories. In our case, security categories and omission of whitelisted users. Then all of this data from the Umbrella reporting API will be downloaded and parsed. After the calculation for each DNS request is done through Umbrella Investigate API, each request will be divided into three categories: Safe, Problematic and Dangerous. Next, this paired information will be updated in the database. This is the periodic part of the automated procedure.

The procedure in use works as follows. A user who wants to connect to the company network uses his VPN client. After filling out, the username and password checking sequence will initialize. Cisco ASA, which received a VPN request, will send a RADIUS request to the server. FreeRADIUS server then looks in the database for user entries. The server then sends RADIUS response to Cisco ASA with additional parameters such as: what privilege category the user is and one of the three security defining categories. ASA then looks in its configuration to see what should be done with the RADIUS response:

Categories	Outcomes
Safe	VPN tunnel is created
Problematic	VPN tunnel is created, but the user has limited access to the network
Dangerous	Connection with the user is terminated with a notification to contact the IT security department and Syslog message sent to CSIRT team

Tab. 4.1: Behaviour categories

When the "Dangerous" user contacts the IT security department with the issue of him being unable to create a VPN connection. They can inspect what is wrong with his device/behaviour. After the fix, they can add him to the whitelist for seven days so the Risk score can straighten out. After the seven day period, the whitelist entry will be automatically deleted. Finishing this automated procedure.

4.2 Configuration

Configuration of the application is done in the config.json file, where are settings for API URIs, Umbrella space identifier, query parameters such as how big the checking interval should be, risk security categories, database connection details, whitelist parameters and lastly, VPN behaviour groups.

4.2.1 Environment variables

Secrets are stored inside environment variables. For security reasons, it is a best practice to create a new user account for the application that will be run. In this scenario, account python_radius was created to handle the periodically run of the application, to store environment variables and as a user for the PostgreSQL database. In UNIX based systems, we use to access the environment variables with this command, where the string value is returned:

```
os.environ['auth_secret']
```

How to set environment variables:

Linux/Unix

```
export env_name=value
```

Windows

```
Powershell: \$env:env_name = 'value'
cmd: setx env_name "value"
```

Environment variables will be deleted after each reboot of the server. To store variables permanently, we need to add these variables to the home/bash.d file.

4.3 Cisco Umbrella

The graphical web user interface of Cisco Umbrella is easy to use. The configuration itself is very straightforward. To make devices forward traffic, the administrator

has to download the profile for the "Umbrella Roaming Security" module in the "Deployments" bookmark. The profile has the id of the Umbrella instance and the necessary keys for safe communication. After the endpoint is configured with the profile, the traffic is forwarded to the Umbrella.

Next, we need to generate keys with secrets needed for communication with the Umbrella APIs. Because the data from the Reporting API is filled with the individual endpoint traffic, there has to be added security in the communication. First, we generate key and secret in the Umbrella UI. Then for the actual communication, we need to get a token through authentication API 2.2.4 with the key and secret provided. In the case of the Investigate API, no token is needed because there is no user data in the API, so a separate key is sufficient. Now Umbrella is ready to receive the DNS traffic.[22]

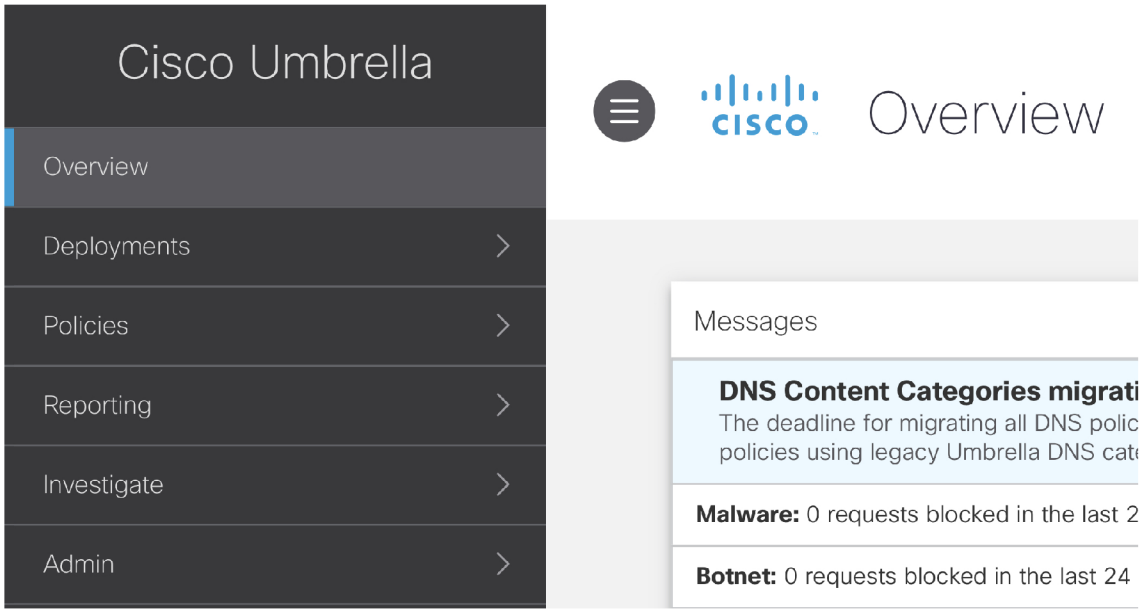

Fig. 4.2: Cisco Umbrella - config

4.4 END Point Device

Configuration on the endpoint device is a bit tricky. For the administrator to be able to use the profile from Umbrella, the endpoint has to have installed Roaming Client or Cisco AnyConnect with Umbrella Roaming Security Module. Before the installation itself, it is crucial that the downloaded profile from the Umbrella connection is placed in the appropriate folder before installation. After the installation, the profile is automatically added and ready to go.

Next, we need to change the DNS server IP address in the system. The IP address for the EU Umbrella server is **88.102.8.183**. Without this change, the request will not be sent to Umbrella for processing.

The most crucial part in the configuration of the endpoint is having a **lock** on the "Client Name" parameter. This is the system device name. In this scenario and in most production environments, the Client Name is matched in the database to its user. It is custom to have this option locked by the set privileges that come from centralized user management. Suppose the administrator does not have the option to manage the device. AnyConnect itself has the option to lock these parameters in the process of installation.

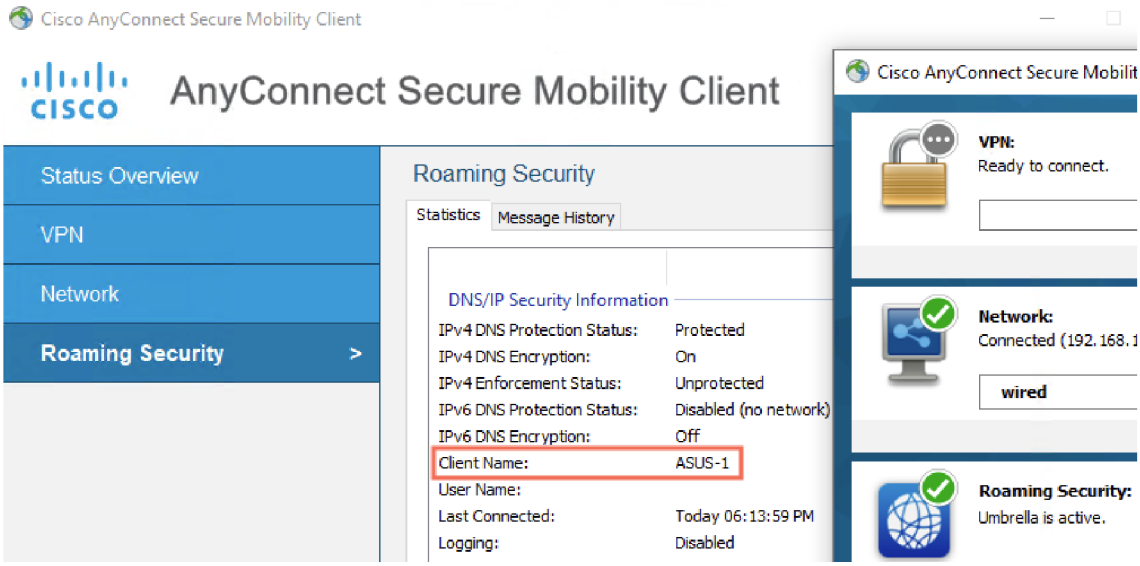

Fig. 4.3: Cisco AnyConnect - config

4.5 CISCO ASA

ASA software is very similar to the Cisco IOS software on routers. As Cisco ASA is a fairly old product, it still has the option to use a command-line interface, but it also has a JAVA application with a graphical interface.

We will be based on the scenario where an internal network is already set up, and all we need to do is to set up ASA as a RADIUS client and all the necessary policies and user groups.

4.5.1 Group Policies

First Group Policies will be created. In production, networking Group Policies are used to create different user types, each with a different set of privileges. We have created three types of users. Guest, User and Administrator. Each user type has its privileges and access on the network. A guest has access only to the internet, the user can access the production section of the network, and the administrator has full access. The Group Policies created are named like this: Group_Guest, Group_User, Group_Administrator.

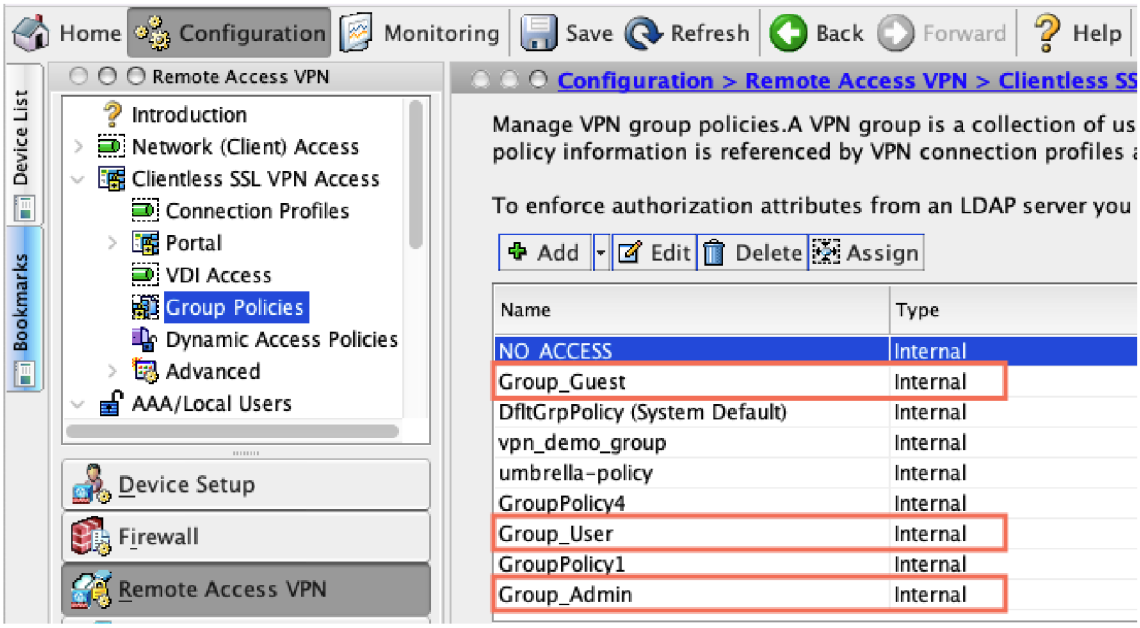

Fig. 4.4: ASA - Group Policies

4.5.2 Dynamic Access Policies

Dynamic Access Policies (DAP) are used as another method of controlling to witch network resource a user is authorized to access. Each policy is evaluated for matching criteria during VPN session establishment. There are many options on how to limit access to the user. We can apply ACL filters, disable file browsing, display a message or terminate the connection before it even forms.

DAP policies are the main control function for the behaviour aspect of end-point security addressed in this document 4.2:

DAP	Functions
Safe	User is free to use the network
Problematic	Restricts the user to only access parts of the network and blocks connection to production servers
Dangerous	Terminates the connection and displays a message to contact the CSIRT team
DfltAccessPolicy	Default DAP with no restrictions

Tab. 4.2: Dynamic Access Policies

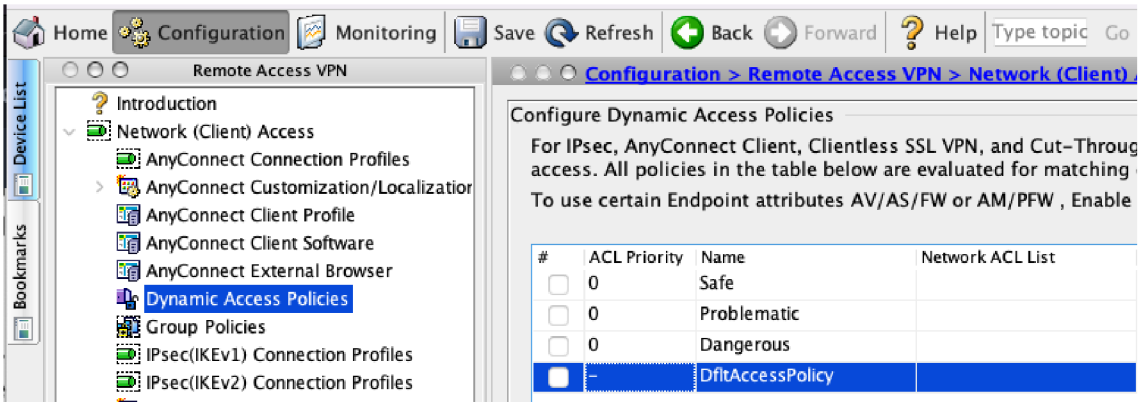

Fig. 4.5: ASA - Dynamic Access Policies

4.5.3 AAA Server Groups

To make DP and DAP policies work, we need to set up the connection to the server. ASA has the ability to work with multiple AAA servers with different protocols. We chose RADIUS. The setup of the AAA server is very straightforward. The administrator has to know the IP address of the RADIUS server, on which interface it is connected, server authentication and accounting ports and finally Secret Key. After the setup administrator can test the connection with the RADIUS server through the test window in the ASA interface.

After all this setup and connection testing, the Cisco ASA is ready to be used and included in the scenario.

4.6 Server

The virtual machine with Ubuntu operating and Cisco Umbrella are the core components of this playbook system that handles many tasks in this scenario. The server runs many services from the core Ubuntu OS running the application, FreeRADIUS

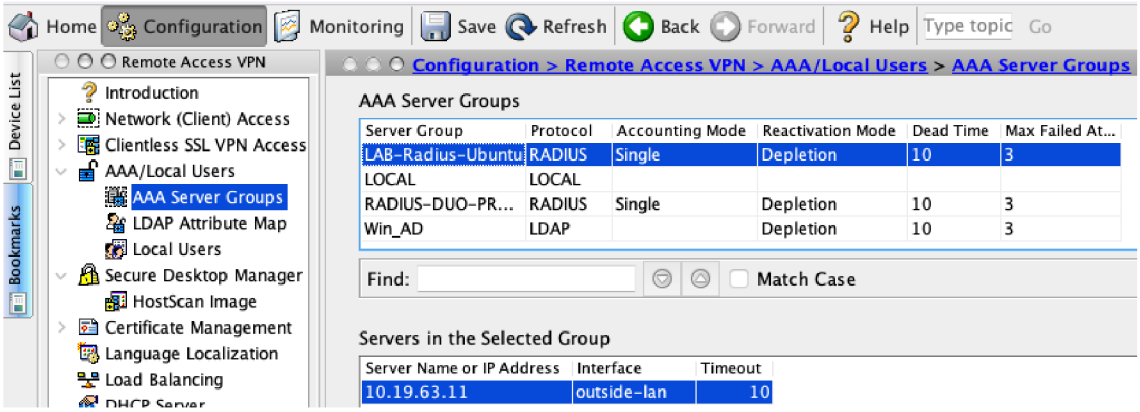

Fig. 4.6: ASA - AAA

server and database. The following steps describe the configuration necessary for our scenario.

4.6.1 VM - Ubuntu

First step on the Ubuntu OS is to create a new user that will be handling the periodic application launch and connection with the database. After the user **python_radius** is created we need to add environment variables that are necessary for the proper function of the application to the home/bash.d file 4.3:

Environment variable	Where to obtain it
umb_reporting_key	is obtained from Cisco Umbrella
umb_reporting_secret	is obtained from Cisco Umbrellas and used with the key to get authorization token from Reporting API.
umb_investigate_secret	is created in Cisco Umbrella and is used to get information from the Investigate API
db_pass	is used as a password for the database

Tab. 4.3: Environment variables

Then we need to install the required Python packages (pip) 2.2.3:

```
pip install -r src/requirements.txt
```

4.6.2 FreeRADIUS server

To install the FreeRADIUS server that will be compatible with we need to proceed as follows:

```
sudo apt update && upgrade
```

Install FreeRADIUS 3.0 and PostgreSQL modules:

```
sudo apt install freeradius freeradius-utils freeradius-postgresql
```

Enable SQL connection:

```
sudo cd /etc/freeradius/3.0/mods-enabled
sudo ln -s ../mods-available/sql sql
```

Delete **-sql** and **#sql** code on sites default and inner-tunnel:

```
sudo nano /etc/freeradius/3.0/sites-available/default
udo nano /etc/freeradius/3.0/sites-available/inner-tunnel
```

Edit FreeRADIUS connection:

```
sudo nano /etc/freeradius/3.0/mods-available/sql

driver = "rlm_sql_postgresql"
dialect = "postgresql"
server = "localhost"
port = 5432
login = "radius"
password = "radpass"
radius_db = "radius"
read_clients = yes
client_table = "nas"
```

Enable, start and check running FreeRADIUS service:

```
sudo systemctl start freeradius
sudo systemctl enable freeradius
sudo systemctl status freeradius
```

To test the freeRADIUS service, we have to configure the database next.

4.6.3 PostgreSQL

PostgreSQL is available for many distributions by default, but it does not guarantee that it will be up to date. The best practice is to install the software from the PostgreSQL Apt repository itself:

```
# Create the file repository configuration:
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt
$(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

# Import the repository signing key:
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc
| sudo apt-key add -

# Update the package lists:
sudo apt-get update

# Install the latest version of PostgreSQL.
# If you want a specific version, use 'postgresql-12' or similar
# instead of 'postgresql':
sudo apt-get -y install postgresql
```

[20] To check the PostgreSQL service status:

```
service postgresql status
```

If the service is running, we can continue in the database configuration with PgAdmin4.

In a real production environment, some type of database will be already in place with users, passwords, and all the necessary information filled out. If that were the case, the administrator would have to amend this procedure.

PgAdmin4

Thanks to the graphical interface of PgAdmin4, the configuration and possible management is relatively easy to do. To create the database as a first step, we need to register a new server. After filling out the name, IP address, username and password, the database server is ready 4.8. To make the database compatible with freeRADIUS, we need to query the database with a schema for FreeRADIUS. `https://github.com/FreeRADIUS/freeradius-server/blob/master/raddb/mods-config/sql/main/postgresql/schema.sql` then we add a table for users identity:

```
-- Table users_identity
CREATE TABLE public.users_identity
(
    id serial NOT NULL,
    radcheck_id integer NOT NULL,
    umbrella_label character varying(32) NOT NULL,
    whitelist_from date,
    PRIMARY KEY (id)
);

ALTER TABLE IF EXISTS public.users_identity
    OWNER to radius;

ALTER TABLE IF EXISTS public.users_identity
    ADD CONSTRAINT radcheck_id FOREIGN KEY (radcheck_id)
    REFERENCES public.radcheck (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE
    NOT VALID;
```

Source Code 4.1: Database - users identity

When all those steps are successful we can proceed to adding the users: Commands

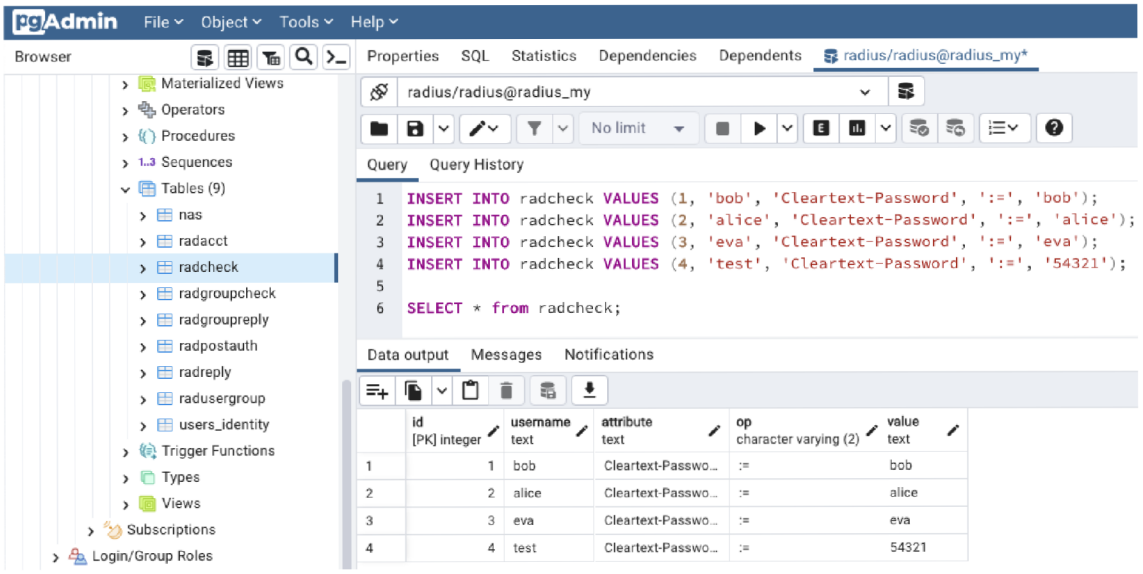

Fig. 4.7: pgAdmin4 - User insert

to test the communication between FreeRADIUS and PostgreSQL database:

```
sudo radtest demo 12345 localhost 10 testing123
sudo radtest test 54321 localhost 10 testing123
```

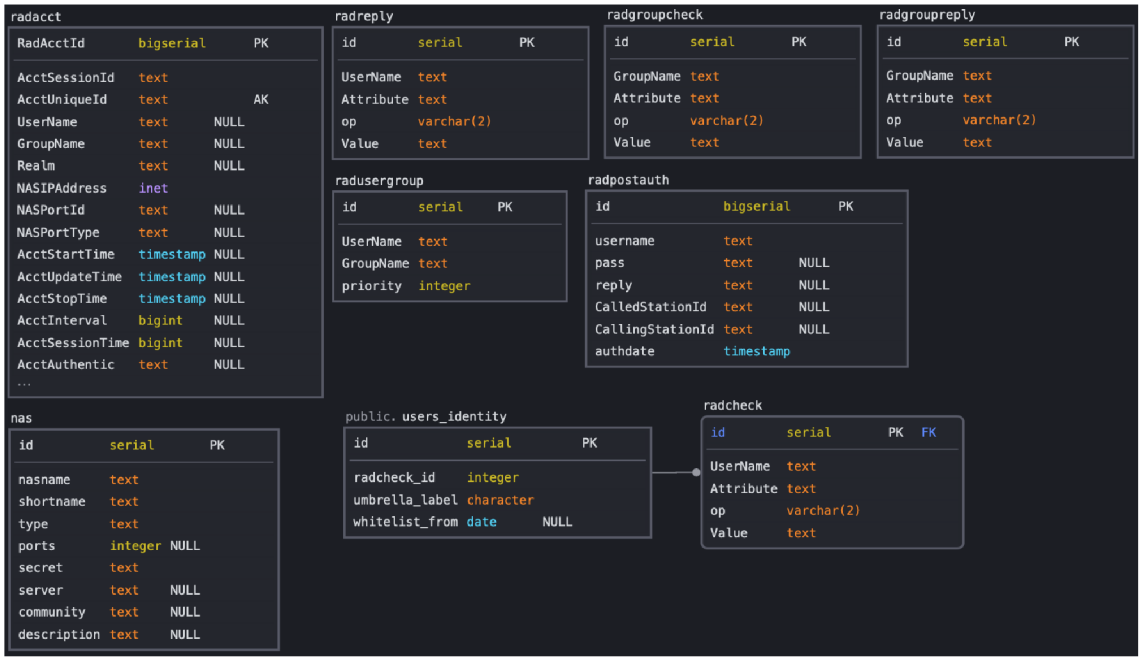

Fig. 4.8: Database diagram

The finished database provides for Cisco ASA the group category (guest, user, admin) and DAP policy calculated from the risk score (Safe, Problematic, Dangerous). Important note: We need to specify a product-specific vocabulary for the communication with Cisco ASA and FreeRADIUS. This vocabulary provides FreeRADIUS with the specific way to assemble RADIUS reply packet 4.2. `https://github.com/redBorder/freeradius/blob/master/share/dictionary.cisco.asa`

```
ATTRIBUTE       Framed-AppleTalk-Network        38      integer


VENDOR          Cisco-ASA                       3076

BEGIN-VENDOR    Cisco-ASA

ATTRIBUTE       ASA-Group-Policy        25      string

ATTRIBUTE       ASA-WebVPN-Customization        113     string

ATTRIBUTE       ASA-Member-Of           145     string
```

```
END-VENDOR      Cisco-ASA
```

Source Code 4.2: dictionary - FreeRADIUS

4.7 Code implementation

The main core of this implementation is the application itself that handles the communication with Cisco Umbrella APIs, a database with stored user data and logic behind the endpoint security, which provides limits to user access in dependence to their behaviour. Due to the complexity and extensiveness of the whole application, only interesting and core parts of the application will be described. The whole repository is on GitHub:

`https://github.com/jakuSk/umbrella_behaviour-security`

Security features

The security features of the code implementation include the use of HTTPS for all API calls, with an even higher level of security with the use of oauth2 for data that is personalised. As an industry standard, all keys, secrets and tokens are stored as system variables. This ensures that even if the code is available, no one can use these security variables. This also has the advantage that if the code is run on a server that is managed by other people. They also can not access these security variables because they are locked under a different user.

Suppose this code would not be open source but a private product. A possible way to make the deployment even more secure would be to obfuscate the deployed code. This approach is also very used in production environments, as it is nearly impossible to reverse engineer the code or logic behind it. Example of obfuscated "def main" method 4.3. The unobfuscated method that is used in this implementation and deployed on the server 4.4

```
def main ():#line:1
    ""#line:2
    OOOOOOOOOOOOOOOO0 =time .perf_counter ()#line:4
    OOOOOOOOOOOOOOOO0 =ConfigService (config_name ='config.json')#line:5
    OOOOOOOOOOOOOOOO0 =DatabaseService (OOOOOOOOOOOOOOOO0 )#line:6
    OOOOOOOOOOOOOOOO0 =UmbrellaReportingService (OOOOOOOOOOOOOOOO0 )#line:7
    OOOOOOOOOOOOOOOO0 =OOOOOOOOOOOOOOOO0 .get_identites ()#line:11
    OOOOOOOOOOOOOOOO0 =OOOOOOOOOOOOOOOO0 .get_users_identities
    (OOOOOOOOOOOOOOOO0 )#line:13
```

```
    for OOOOOOOOOOOOOOOO in range (len (OOOOOOOOOOOOOOOO )):#line:16
        process_users_activity (OOOOOOOOOOOOOOOO ,OOOOOOOOOOOOOOOO
        ,OOOOOOOOOOOOOOOO ,OOOOOOOOOOOOOOOO )#line:18
    OOOOOOOOOOOOOOOO =time .perf_counter ()#line:20
    print (f'Time taken to process {len(OOOOOOOOOOOOOOOO)} users is
    {OOOOOOOOOOOOOOOO-OOOOOOOOOOOOOOOO}')
```

Source Code 4.3: app.py - Main method - obfuscated

4.7.1 config.json

As mentioned in the 4.2 section. The application has its configuration file with all variable parameters that the administrator can change. The most useful ones are organization_id for Umbrella space, query parameters, query categories that the administrator wants to monitor and the user whitelist length. (More in-depth 4.7.7).

```
"reports": {
            "organization_id": "7966517",
            "query_parameters": {
                "from": "-30days",
                "to": "now",
                "limit": "5000",
                "offset": "0"
            }
```

Query security categories: 4.4

4.7.2 Main file app.py

An auto-run python application for free deployment on environments with Python 3.0 is open source for everyone on the public GitHub repository. The main file of the automation playbook app.py stores the Main method of the app with all of the high-level logic. Added classes and configuration files to store the app's core from the load- ing the configuration, communication with the APIs and all the necessary filtration and parsing of outputs.

Throughout the app is used a standard syntax for handling GET requests. GET requests are generally assembled of URI (Uniform Resource Identifier) for a specific resource, headers with some form of authentication and a format in which to accept the data from the APIs. This approach is applied in every method throughout the app. [21]

ID	Label
60	Drive-by Downloads/Exploits
61	Dynamic DNS
62	Mobile Threats
63	High Risk Sites and Locations
64	Command and Control
65	Command and Control
66	Malware
67	Malware
68	Phishing
106	Unauthorized IP Tunnel Access
108	Newly Seen Domains
109	Potentially Harmful
110	DNS Tunneling VPN
150	Cryptomining

Tab. 4.4: Query security categories

```
def main():
    """Main method of the application."""
    # Service creation
    t1 = time.perf_counter()
    config_service = ConfigService(config_name='config.json')
    db_service = DatabaseService(config_service)
    umbrella_service = UmbrellaReportingService(config_service)

    # Application logic
    # First get active identities from umbrella top-identities API
    identities_dict = umbrella_service.get_identites()
    # Then we get the mapping table from db that maps umbrella_label with username
    users = db_service.get_users_identities(identities_dict)

    # Then we get the reports for each user
    for users_index in range(len(users)):
        process_users_activity(umbrella_service, users,
                               users_index, db_service)

    t2 = time.perf_counter()
    print(f'Time taken to process {len(users)} users is {t2-t1}')
```

Source Code 4.4: app.py - Main method program with all high-level logic

After importing all classes needed for the Main method, the app loads the configuration. Then it starts calling methods inside corresponding classes for individual API calls to get the data needed for determining if the user has harmful behaviour or not. Then "process_users_activity" method is called that will process the user activities, get the risk score from the Investigate API and save it to the database where the FreeRADIUS server reads the data 4.4.

Exit codes: 4.5

Exit code	Reason
100	Unknown exceptions
101	Failed to obtain bearer token
102	Non 2XX status code on API call

Tab. 4.5: Exit codes

We have, in total, four services (classes) that support the run of the application:

4.7.3 Service - config_service.py

The config_service.py loads the configuration from the config.json file, returns it as a dictionary. Then it finds asked elements by the application by key_path after splitting the key_path for each element. Next it returns the value from the configuration file.

```
def __get_config_member(self, key_path: str):
    """Finds given element by key_path in __config after splitting the
    key_path for each element"""
    split_path = key_path.split(':')
    temp_config = self.__config

    for key in split_path:
        temp_config = temp_config.get(key)
    return temp_config

def get_value(self, property_path: str):
    """Returns value from configuration file. Uses remembered path."""
    value = self.__get_config_member(property_path)
    return value
```

Source Code 4.5: config_service.py

4.7.4 Service - umbrella_reporting_service.py

This class handles the main communication with the Umbrella APIs. First, we need to get an authorisation token from the authorisation API that will be used with the Umbrella Reporting API. This method handles loading the secrets necessary for obtaining token and handles error messages.

```
def __get_secrets(self, key, secret) -> (str, str):
        """Method to get the secrets from the environment variables."""
        try:
            key = os.environ[key]
            secret = os.environ[secret]
            return key, secret
        except KeyError as error:
            print(f'Environment variable {error} not found... Exiting')
            sys.exit(1)
        except Exception as error:
            print(f'Error: {error}')
            sys.exit(100)
```

Source Code 4.6: umbrella_reporting_service.py - get secrets

Then method add_query_string handles query requests by changing URLs with the appropriate parameters that are stored in the config.json file. Because the Umbrella API does not support pre-filtering the query by the **label** parameter (hostname of the endpoint), we need to get an internal Umbrella identity id for each endpoint. This task is handled by the get_identities method. Then we load the desired query categories stored in config.json. In our case, all security categories. Following report URL is created in get_report_url method, which is used in get_report_user method that will get the DNS requests data itself for each user and store it in JSON format for parsing that will be handled by process_dns_queries method. Lastly, the get_investigate_data method gets data from Investigate API for all DNS requests pre-filtered by desired categories and returns a dictionary with risk score values that will be used to determine the behaviour of the endpoint.

```
def get_investigate_data(self, domain: str) -> dict:
    """Get data from Investigate API. Returns dict with risk_score value"""
    return_dict = {}
```

```
    return_dict['domain'] = domain
    api_token = os.environ['umb_investigate_token']
    url = self.__config.get_value('umbrella:urls:investigate') + domain
    api_headers = {}
    api_headers['Authorization'] = f'Bearer {api_token}'
    api_headers['Accept'] = 'application/json'

    response = requests.get(url, headers=api_headers, verify=True)
    if response.status_code != 200:
        print(
           f'Non 200 status code - {response.status_code} on investigate api.')
        print(response.json())
        sys.exit(1)

    # Here we add the risk_score for the domain into out return_dict
    return_dict['risk_score'] = response.json()['risk_score']

    return return_dict
```

Source Code 4.7: umbrella_reporting_service.py - get investigate data

4.7.5 Service - database_service.py

database_service.py, as the name suggests, manages data from the database. After the sqlalchemy is imported, we load the configuration through the config_service.py class. Next, we initialise the database itself. Then the class has many methods. One handles getting the end-device labels and adding them to the string. Then getting active endpoints and their identity id necessary for use in getting the report data. The last method updates the user parameters with the appropriate DAP policy.

```
def get_users_identities(self, identities_dict) -> list:
    """Method to get users identities"""
    labels = self.__get_users_labels_as_string(identities_dict)
    query = f'''SELECT rc.username, ui.umbrella_label
        FROM radcheck rc
        JOIN users_identity ui
        ON ui.radcheck_id = rc.id
        WHERE ui.umbrella_label IN ({labels})
        AND (ui.whitelisted_from IS NULL
            OR ui.whitelisted_from <= CURRENT_DATE -{self.__whilist_eff});'''

    users_query_result = self.__connection.execute(query).fetchall()
    return_list = []
    for user in users_query_result:
```

```
            return_list.append((user['username'],
                                user['umbrella_label'],
                                identities_dict[user['umbrella_label']]))

    return return_list
```

Source Code 4.8: umbrella_reporting_service.py - user identities

4.7.6 Service - risk_score_calculation.py

The risk_score_calculation class is very straightforward. The class has four methods. The first method calculates an average risk score from pre-filtered DSN requests for each endpoint. The paired method will store this risk score and its domain for later use. The following method checks if there is a DNS request with a risk score prevailing of 55. The last method calculates the risk score. If an endpoint has one or more dangerous requests (risk score higher than 55), the method returns the requests risk score. Else we return the average from all pre-filtered DNS requests.

```
def calculate_risk_score(domain_list: list) -> int:
    """Calculate risk score"""
    if (__is_very_risky(domain_list)):
        return __get_highest_risk(domain_list)
    else:
        return __calculate_risk_score_average(domain_list)
```

Source Code 4.9: risk_score_calculation.py - calculate risk score

4.7.7 whitelist.py

The whitelist script is essential for the whole playbook. When the user is blocked, there is no way to clear the logs in the Umbrella. IT security teams have to have this data logged in case they need to investigate some malicious behaviour when there is a breach in the network. Without the whitelist scrip, administrators would not be able to unblock users with bad behaviour statistics rendering this whole playbook undeployable to the production environment. Then user contacts the proper administrator with his connection problem. The administrator will inspect the user's behaviour and consult the reason why his endpoint behaves maliciously. When the problem is sorted administrator, can whitelist the user from the application process. Then the entry to the database will be deleted after the previously set time expiration (days):

```
./whitelist.py -u <username>
```

5 Results

The final results are that we successfully overcame the manual procedure with automation. After everything from the chosen solution is configured, we can start the created workflow. As a first step, we are using API calls on the Cisco Umbrella Reporting module to load all DNS requests from the endpoint that fall into security categories 5.1. After parsing all of this data, we take all hostname addresses and run them through the Umbrella Investigate module. Investigate will provide a lot of useful information. We are interested in the Risk score attribute for each hostname address 5.2. With this new information, we pair the DNS requests from the user's endpoint device and their risk score. Then by set thresholds, we determine if the endpoint has dangerous, problematic, or safe behaviour. After we know the final behaviour status of each endpoint, we send this information to the database 5.1, 5.2. If the user wants to initialise remote access VPN connection, Cisco ASA sends a RADIUS request to the FreeRADIUS server, which then checks with the database if said user exists, has his password right, his privileges and finally, his behaviour category. Cisco ASA then checks the behaviour category. If user is in the dangerous category, terminates the connection 5.4, 5.5. If his behaviour category is safe, assigns him appropriate authorization category and connects him 5.6, 5.7, 5.3. If the administrator wants to whitelist this user after investigating the reasons for his bad behaviour, he can easily add him for a specified time to the whitelist, where he will be excluded from the workflow 5.8.


```
"data": [
    {"returncode": 0, "externalip": "88.102.8.183", "allapplications": [],
        "date": "2022-05-19", "internalip": "190.1.1.129", "time": "17: 37: 48"
        "querytype": "A", "policycategories": [
            {"id": 67, "type": "security", "label": "Malware", "integration":
            False, "deprecated": False
            }
        ], "type": "dns", "categories": [
            {"id": 67, "type": "security", "label": "Malware", "integration":
            False, "deprecated": False
            },
            {"id": 113, "type": "content", "label": "Computer Security",
            "integration": False, "deprecated": False
            }
        ], "verdict": "blocked", "domain": "examplemalwaredomain.com",
        "timestamp": 1652981868000, "blockedapplications": [],
        "allowedapplications": [], "identities": [
            {"id": 585728824, "type": {"id": 34, "type": "anyconnect", "label":
            "Anyconnect Roaming Client"
                }, "label": "VM_WIN11", "deleted": False
```

```
            }
        ], "threats": []
    }
]
```

Source Code 5.1: API call - user report data

```
{'domain': 'examplemalwaredomain.com', 'risk_score': 100}
```

Source Code 5.2: API call - Investigate data

```
18  SELECT * from radcheck;
```

Data output | Messages | Notifications

	id [PK] integer	username text	attribute text	op character varying (2)	value text
1	1	bob	Cleartext-Passwo...	:=	bob
2	2	alice	Cleartext-Passwo...	:=	alice
3	3	eva	Cleartext-Passwo...	:=	eva
4	4	test	Cleartext-Passwo...	:=	54321

Fig. 5.1: Database - users table

```
20  SELECT * from radreply;
```

Data output | Messages | Notifications

	id [PK] integer	username text	attribute text	op character varying (2)	value text
1	1	eva	ASA-Member-...	:=	Dangerous
2	2	bob	ASA-Member-...	:=	Safe
3	3	alice	ASA-Member-...	:=	Problemat...

Fig. 5.2: Database - behaviour table

```
ASA-ALEF-LAB-01# show vpn-sessiondb anyconnect

Session Type: AnyConnect

Username     : bob                    Index        : 14151
Public IP    : 10.32.0.240
Protocol     : AnyConnect-Parent
License      : AnyConnect Premium
Encryption   : AnyConnect-Parent: (1)none
Hashing      : AnyConnect-Parent: (1)none
Bytes Tx     : 0                      Bytes Rx     : 0
Group Policy : Group_Guest            Tunnel Group : LAB-RadiusTEST
Login Time   : 01:08:56 UTC Mon May 30 2022
Duration     : 0h:00m:04s
Inactivity   : 0h:00m:00s
VLAN Mapping : N/A                    VLAN         : none
Audt Sess ID : c0a8140103747000629419 28
Security Grp : none
```

Fig. 5.3: ASA - VPN session for bob with - Group Policy assignment

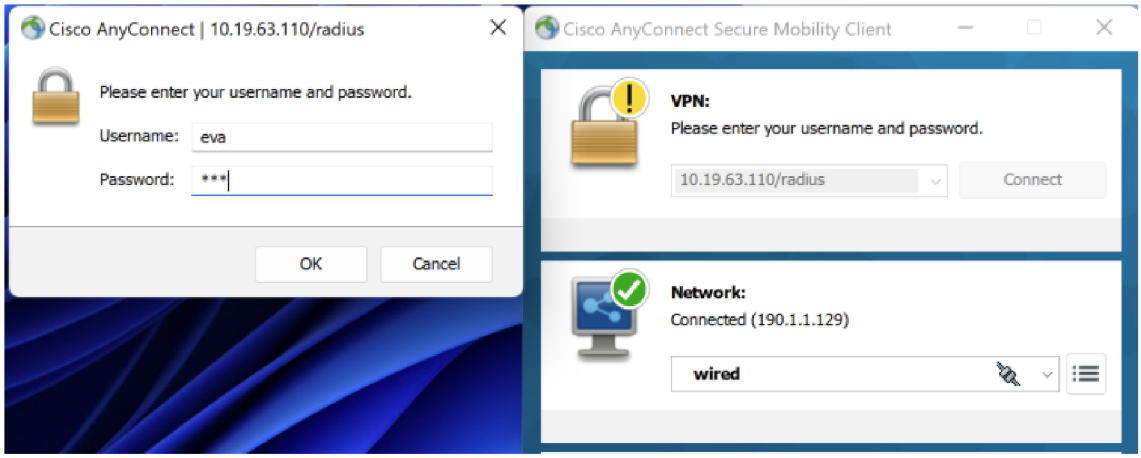

Fig. 5.4: Anyconnect eva - connecting

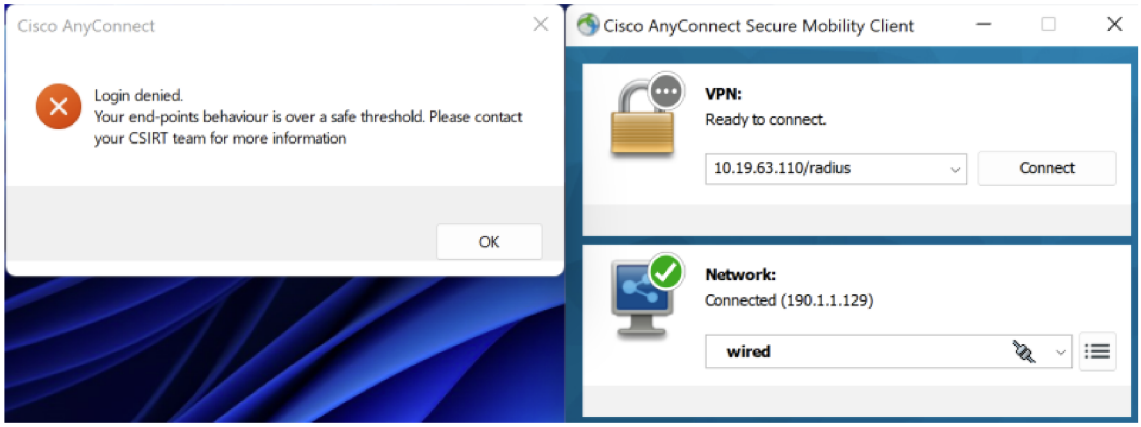

Fig. 5.5: Anyconnect eva - terminated

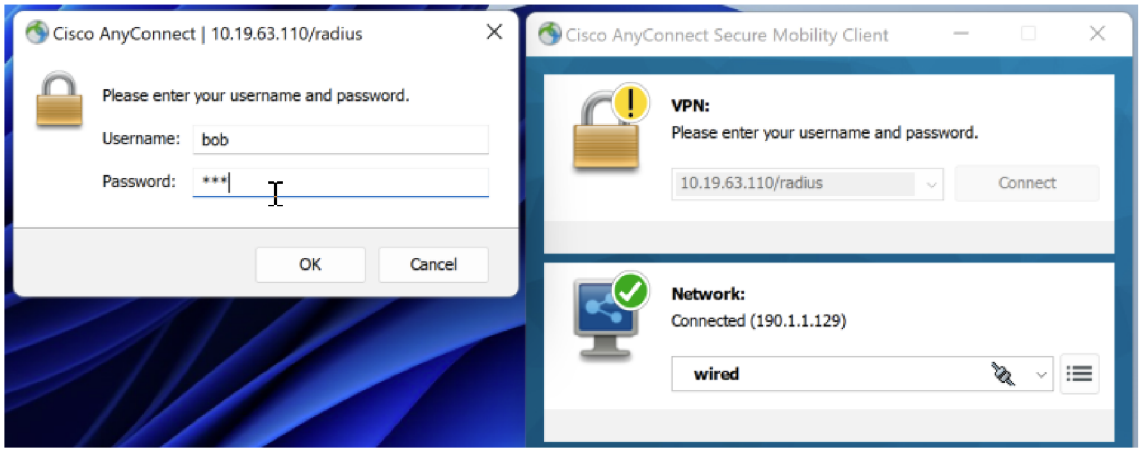

Fig. 5.6: Anyconnect bob - connecting

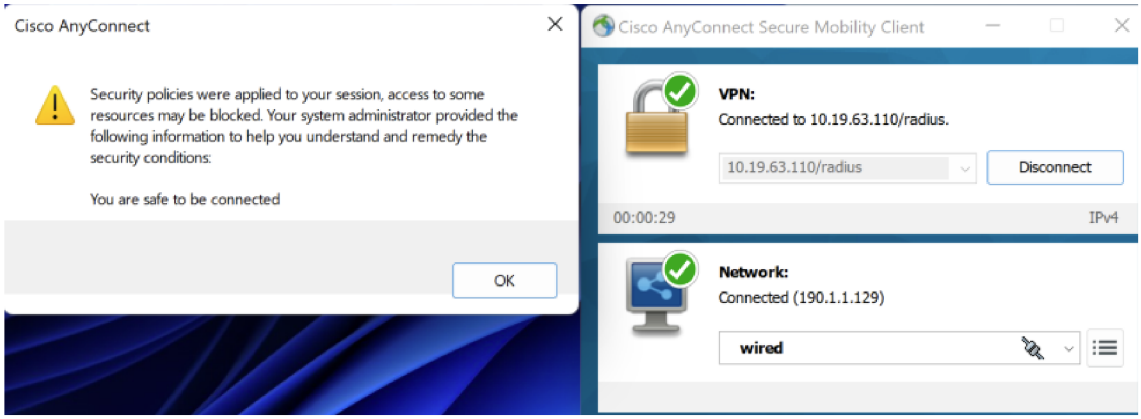

Fig. 5.7: Anyconnect bob - safe

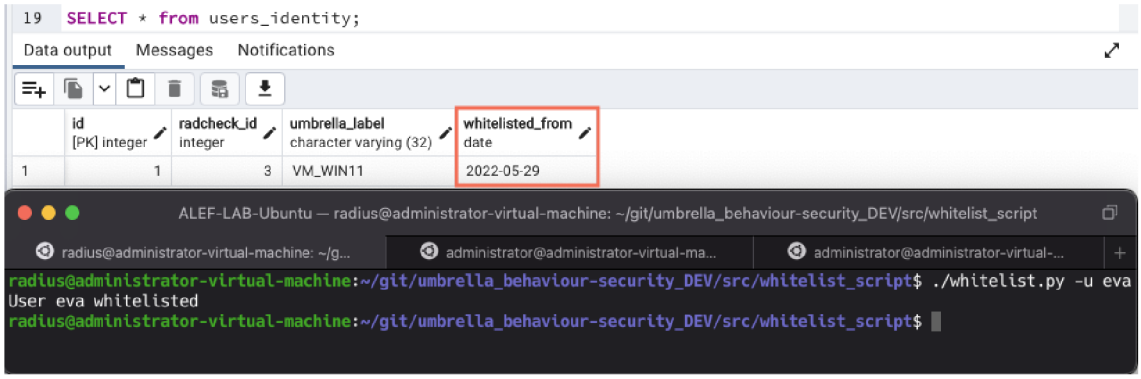

Fig. 5.8: Database - whitelist

Conclusion

The goal of this bachelor thesis was to study the possibilities of automation of security network components, identify obsolete manual procedures associated with threat identification with transfer of information context between security technologies in complex operational networks. Design an automated scenario that will solve the shortcoming of manual procedures and apply this scenario on real devices. The goal was successfully fulfilled. The thesis described the pitfalls in network security, end-point security and their developing trends for the future. Importance for security against new threats and protection of assets due to the increasing risks in dependence to the global shift in connectivity to cloud services, remote work from home caused by the COVID-19 pandemic and many other factors has grown. Reality that attacks are getting more complex resulting in higher resilience to convention defences, with the outcome that businesses can no longer solely rely on turnkey solution like firewalls, antivirus software lead to the development of new products from business leaders in cyber security focusing on user behavior, information correlation and automation.

Then after selecting obsolete manual procedure, solution was created using Cisco Umbrella and its capabilities in DNS security, on which an automation workflow was designed. This workflow is acquiring information from Cisco Umbrella from which it is able to determine the behaviour of an users end-point device. This gained knowledge is then utilized when end-point device is initializing connection with remote access VPN to connect the the company's internal network. If the users behavior is found dangerous, connection is terminated. If his behavior is safe, proceeds with establishing VPN tunnel connection. Then giving an option to the administrator to whitelist the user, after investigating the reasons for the bad behavior.

The future of this scenario is to implement it to the main security orchestration product from Cisco SecureX. This will make the solution even easier to deploy resulting in much grater and broader use. This could render this scenario in to a standard security product deployed in production networks. But before this happens the whole code implementation is available under the open source MIT licence on Github: `https://github.com/jakuSk/umbrella_behaviour-security`.

Bibliography

[1] IBM. 2020. Cost of a Data Breach Report 2020. *Cost of a Data Breach Report 2020* [online]. New Orchard Road Armonk, NY 10504: IBM Corporation. Available at: https://www.ibm.com/downloads/cas/RZAX14GX

[2] MORGAN, Steve. 2019. Global Cybersecurity Spending Predicted To Exceed $1 Trillion From 2017 – 2021. *Global Cybersecurity Spending Predicted To Exceed* 1 Trillion From 2017-2021 [online]. 1 Harbor Drive, Sausalito CA 94965: Cybersecurity Ventures. Available at: https://cybersecurityventures.com/cybersecurity-market-report/

[3] Coronavirus update: In the cyber world, the graph has yet to flatten. 2020. Check Point Software Tehcnologies LTD [online]. San Carlos 959 Skyway Road Suite 300 San Carlos, CA 94070: Check Point. Available at: https://blog.checkpoint.com/2020/04/02/coronavirus-update-in-the-cyber-world-the-graph-has-yet-to-flatten/

[4] Automation, Visibility Remain Biggest Issues For Cybersecurity Teams, From New Fidelis Report. 2019. Fidelis Cybersecurity [online]. Bethesda, MD: Fidelis Cybersecurity. Available at: https://fidelissecurity.com/newsroom/state-threat-detection-2019/

[5] RDP and VPN use skyrocketed since coronavirus onset. 2020. ZDNet [online]. www.zdnet.com: Catalin Cimpanu. Available at: https://www.zdnet.com/article/rdp-and-vpn-use-skyrocketed-since-coronavirus-onset/

[6] Network automation [online]. 2022. Corporate Headquarters 170 West Tasman Dr. San Jose, CA 95134 USA: Cisco Systems. Available at: https://www.cisco.com/c/en/us/solutions/automation/network-automation.html

[7] AAA Security: What is Authentication, Authorization, and Accounting (AAA)?. © 2022. AAA Security [online]. 899 Kifer Road Sunnyvale, CA 94086 US: Fortinet. Available at: https://www.fortinet.com/resources/cyberglossary/aaa-security

[8] RADIUS System Components. ©2021. The RADIUS protocol [online]. 100 Centrepointe Dr, Suite 200 Ottawa, ON K2G 6B1 Canada: NetworkRADIUS. Available at: https://networkradius.com/doc/3.0.10/concepts/introduction/components.html

[9] FreeRADIUS [online]. © 2018. The FreeRADIUS Server Project and Contributors. Available at: https://freeradius.org/about/#project

[10] What Is a Database?. © 2022. ORACLE [online]. 500 Oracle Parkway, M/S 5op7 Redwood Shores, CA 94065 Attention: Trademark and Copyright Legal Department: Oracle Corporation. Available at: https://www.oracle.com/database/what-is-database/

[11] What is PostgreSQL?. © 2022. PostgreSQL [online]. he PostgreSQL Global Development Group. Available at: https://www.postgresql.org/about/

[12] PgAdmin [online]. © 2021. PostgreSQL Community Association of Canada. Available at: https://www.pgadmin.org/docs/pgadmin4/latest/index.html

[13] Cisco Umbrella at a glance. 2017. Cisco Umbrella [online]. Cisco Systems, Inc. 170 West Tasman Dr. San Jose, CA 95134 USA: Cisco. Available at: https://www.insight.com/content/dam/insight-web/Canada/PDF/partner/cisco/cisco-umbrella-at-a-glance.pdf

[14] Endpoint Security. 2021. CheckPoint [online]. San Carlos 959 Skyway Road Suite 300 San Carlos, CA 94070: Check Point. Available at: https://www.checkpoint.com/solutions/endpoint-security/

[15] What is an API? 2017. RedHat [online]. 100 E. Davie St. Raleigh, NC 27601: RedHat. Available at: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

[16] What is a REST API? 2017. RedHat [online]. 100 E. Davie St. Raleigh, NC 27601: RedHat. Available at: https://www.redhat.com/en/topics/api/what-is-a-rest-api

[17] JSON vs XML. W3Schools [online]. W3Schools. Available at: https://www.w3schools.com/js/js_json_xml.asp

[18] S. Zhang, Y. Gao, M. Zhang, J. Ge and S. Wang, "The Study of Network Security Event Correlation Analysis Based on Similar Degree of the Attributes," 2013 Fourth International Conference on Digital Manufacturing & Automation, 2013, pp. 1565-1569, doi: 10.1109/ICDMA.2013.375.

[19] CVE [online]. 2021. MITRE Corporation. Available at: https://cve.mitre.org/index.html

[20] PostgreSQL Installation Steps. © 2022. PostgreSQL [online]. The PostgreSQL Global Development Group. Available at: https://www.postgresql.org/download/linux/ubuntu/

[21] GOOLEY, Jason a Chris JACKSON. Cisco Certified DevNet Associate DEVASC 200-901 Official Cert Guide.Pearson Education, 2020. ISBN: 0136642969

[22] Cloud Security APIs [online]. 2021. Cisco Systems, Inc. 170 West Tasman Dr. San Jose, CA 95134 USA: Cisco DevNet. Available at: https://developer.cisco.com/docs/cloud-security/

[23] Cisco Umbrella reporting - Activity search [online]. 2021. Europe: Cisco. Available at: https://dashboard.umbrella.com

[24] Fixing git HTTPS Error: "bad key length" on macOS 12 [online]. 2021. Stackoverflow: User: nd. Available at: https://stackoverflow.com/questions/69734654/fixing-git-https-error-bad-key-length-on-macos-12

A Contents of the electronic attachment

The entire application is in an electronic attachment and the directory structure is described below.

- `/` — root directory of the appended archive
  - `.vscode/` — vscode settings
    - `launch.json`
    - `settings.json`
  - `help_scripts/` — help scripts
    - `get_categories.py` — API call to get all categories and their IDs
  - `src/` — all source files
    - `behaviour_risk_calculator/` — main code of the application
      - `services/` — support classes for the main application
        - `config_service.py`
        - `database_service.py`
        - `umbrella_reporting_service.py`
      - `app.py` — main application
      - `config.json` — configuration file
      - `requirments.txt` — requirements needed for the app to run
      - `risk_score_calculation.py`
    - `whitelist_script/` — white list app
      - `config.json` — configuration file
      - `config_service.py`
      - `requirments.txt`
      - `whitelist.py` — application for whitelisting desired user endpoint
  - `.gitignore` — file that ignores unwanted files from uploading to repository
  - `CURLS.md` — markdown file with curls
  - `LICENCE` — main file of the slides for presentation
  - `README.md` — description file of the repository