AI FOR INDUSTRY BASED CYBERSECURITY THREATS

SOLUTION DESIGN DOCUMENT

CONTENTS

EXECUTIVE SUMMARY

The key deliverable of this project was a prototype microservice system that will enable a software program to gather data relating to recent cybersecurity threats and vulnerabilities from reliable open-source repositories. With this data, the microservice will utilise machine learning to identify the highest priority threats and vulnerabilities impacting specific industries. When a user queries the service, the software will be capable of generating a list of high priority threats and vulnerabilities as well as relevant mitigation strategies for an industry of interest.

INTRODUCTION

2.1 Context and purpose

As at early 2022, Securemation undertake a manual “cyber security health-check” for clients across all industries. This involves assigning a specialised Security Analyst to manually research and assess industry-specific threats and vulnerabilities and develop appropriate recommendations to mitigate identified risks. This process requires a significant investment of time and labour resources and does not guarantee complete capture of all available data on current cyber security threats and vulnerabilities. Importantly, manual, unscalable processes hinder Securemation’s ability to take advantage of the rapid growth being forecast in the demand for cyber security consulting services.

Significant internal time and cost efficiencies could be realised by partly or fully automating this process, leading to improved timeliness, cost, and accuracy of services provided to clients. By doing so, Securemation stands to increase the robustness of its clients’ cyber security capabilities, enable them to pursue digitally-driven business strategies, and in turn support the growth of new and existing industries in Australia.

2.2 Target Audience/s

The primary audience / user of this prototype microservice are Securemation Security Analysts.
Once refined and deployed in a live environment, the microservice will be able to be used by current and potential Securemation clients (typically small to medium enterprises).

2.3 Scope

IN SCOPE	OUT OF SCOPE
- A tested and fully working prototype of software component that meets agreed user requirements listed as 'Must' and 'Should' in section 2.4. - As built' documentation capturing low-level design information.	- A bespoke, standalone graphical user interface. - A bespoke API. - Categorisation of threats/vulnerabilities by industry locales (geographic locations). - Categorisation of threats/vulnerabilities by subsector industries. - Enhanced toolkit of maturity assessments.

2.4 Requirements

ID	FUNCTION / NON-FUNCTIONAL	REQUIREMENT (PRESENTED AS USER STORIES)	MOSCOW RANKING
1	F & NF	As a Security architect user of a security management tool, I would like to be able to gather data relating to cyber security threats from reliable or pre-vetted online sources, so that I can be reliably informed of current cyber security threats in my industry.	Must
2	F	As a Security architect user of a security management tool, I would like any identified threats and vulnerabilities related to my selected industry to be prioritised, so that I can pursue proportionate mitigation strategies and make informed investment decisions.	Must
3	NF	As a Security architect user of a security management tool, I would like to know that the solution components are adequately protected against cyber security threats, so that I can place trust in the accuracy of the output.	Must
4	F	As a Security architect user of a security management tool, I would like to receive threat mitigation recommendations for all high priority threats/vulnerabilities identified, so that I can ensure my organisation is adequately protected from identified threats.	Must
5	NF	As a Security architect user of a security management tool, I would like each component of the solution to be integrated to ensure seamless communication and compatibility between existing and potential software components.	Must
6	F	As a Security architect user of a security management tool, I would like to receive (at a minimum) the name of identified cyber security threats, industry queried, priority level, mitigation recommendation/s, and a measure of confidence, so that I can be well-informed prior to undertaking more bespoke analysis and recommendations.	Must

7	F	As a Security architect user of a security management tool, I would like the output of the solution to be in JavaScript Object Notation (JSON) format, so that I can easily manipulate this into other forms such as display on a webpage.	Must
8	F	As a Security architect user of a security management tool, I would like identified threats and vulnerabilities for a selected industry to be categorised, so that I can quickly understand their nature and likely features.	Must
9	NF	As a Security architect user of a security management tool, I would like appropriate documentation outlining the functionality and user interaction of the solution, so that I can audit and build upon its functionality.	Must
10	NF	As a Security architect user of a security management tool, I would like any solution design to be flexible enough to accommodate additional data in future, so that I can further build upon its functionality.	Should
11	F	As a Security architect user of a security management tool, I would like the recency of threats to be accounted for in the threat prioritisation so that I can make informed judgements on the likely relevance of the threat/s and appropriate mitigation strategies.	Should
12	F	As a Security architect user of a security management tool, I would like to be able to access information from across the web so that I can have a better understanding of current cyber security threats.*	Could
13	F	As a Security architect user of a security management tool, I would like to be able to gain an understanding of what threats are likely to occur in the future, so that I can be better prepared in proactively preparing mitigation strategies for clients.	Could

*Downgraded as agreed between the solution development team and Securemation.

2.5 Key Terms (A-Z)

While not exhaustive, the following key terms provided a common understanding and shared language in the development of this prototype microservice.

DEFINITIONS

Common Vulnerability Scoring System (CVSS) - An open framework for communicating the characteristics and severity of software vulnerabilities on a scale of 0-10. CVSS is designed to measure the severity of a vulnerability and should be supplemented with a contextual analysis of the environment to form a complete picture of risk.

Common Vulnerabilities and Exposures (CVE) - A public list of security vulnerabilities in software or products, each with a unique identifying number, accessible here: https://cve.mitre.org/.

Indicator of Compromise (IoC) - An artifact (file name, hash, IP address etc.) that suggests unauthorised access or infection has occurred.

Industry – a grouping of companies that are related based on their primary business activities.

Malware Information Sharing Platform (MISP) - An open-source threat intelligence sharing platform that AusCERT uses to distribute IoC information, accessible here: https://www.misp-project.org/.

Threat - An incident that impacts – or has the potential to impact – an organisation's security. Also referred to as a malicious act.

Vulnerability - A specific weakness in a piece of software or system configuration, which an attacker can exploit.

Mitigation – An actionable strategy for negating or reducing the severity or likelihood of a threat or the exploitation of a vulnerability.

Industry benchmarks

Threats and mitigation strategies - The Australian Cyber Security Centre (ACSC) has developed prioritised mitigation strategies to help cyber security professionals in all organisations mitigate cyber security incidents caused by various cyber threats (ACSC, 2022). This tool was adopted for the development of the prototype architecture, and is available in full here: https://www.cyber.gov.au/acsc/view-all-content/publications/strategies-mitigate-cyber-security-incidents.

Vulnerability Severity Ratings - The Common Vulnerability Scoring System (CVSS) is an open framework for communicating the characteristics and severity of software vulnerabilities. The National Vulnerability Database (NVD) provides qualitative severity ratings of "Low", "Medium", and "High", detailed in full here: https://nvd.nist.gov/vuln-metrics/cvss. While not used for the development of the prototype microservice, it is recommended that further system refinement align with the vulnerability scoring system set out by NVD to maintain consistency with industry standards.

2.6 Tools used

The following tools were used in the development of the prototype microservice:

TOOL	NAME	VERSION
Containerization Platform	Docker	20.10.14
Workflow Program	jBPM Workbench Showcase Docker Image: - jBoss Wildfly - jBPM Workbench - KIE Server - jBPM Case Management Showcase	 23.0.2.Final 7.61.0.Final 7.61.0.Final 7.61.0.Final
ML / Data Analysis	Python & Python Modules: - Json - Pandas - NumPy - Matplotlib - Requests - Scikit Learn - SQL Alchemy - NLTK - **TensorFlow**	3.9.13 2.0.9 1.3.5 1.21.6 3.5.2 2.27.1 1.0.2 1.3.1 3.6.6 **2.8.0**
***Notebook Environment**	Jupyter Notebook	5.7.8
Data ETL Tool	Talend Open Studio for ESB	8.0.1
****API Testing Platform**	Postman	9.20.3

**Optional - for the purpose of containing executable python code and analysis results in figures and tables in a document format for data analysis purposes.*

***Optional – for the purpose of testing API endpoints, results can also be achieved using a browser alternatively*

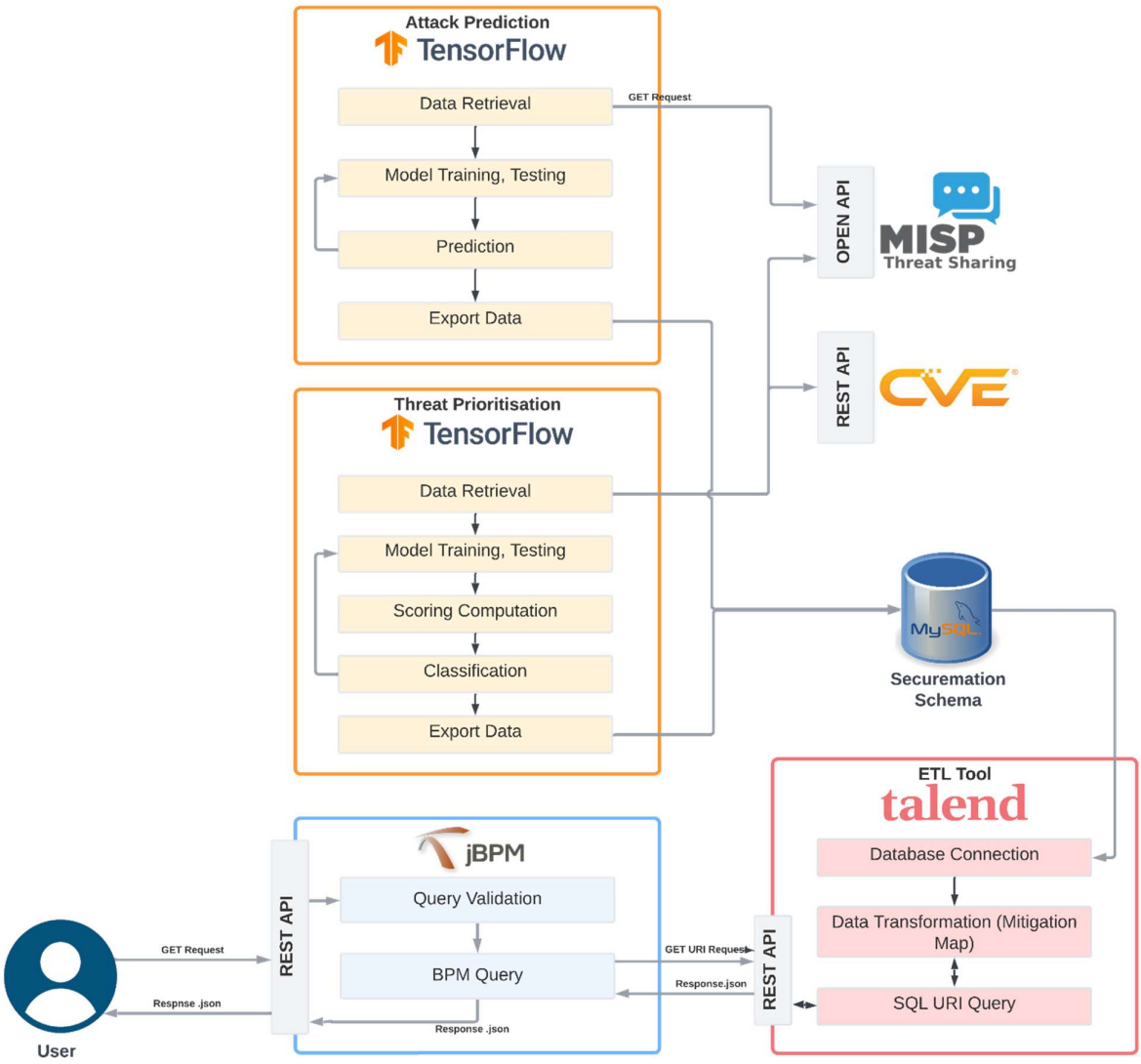


Figure 1. Prototype microservice system architecture

BUSINESS ARCHITECTURE

Deployment of the prototype microservice will fundamentally reconfigure the currently process (shown in Figure 1) and significantly reduce the time spent by the Security Analyst (as shown in Figure 2), reducing the time and resource investment by roughly 50%.

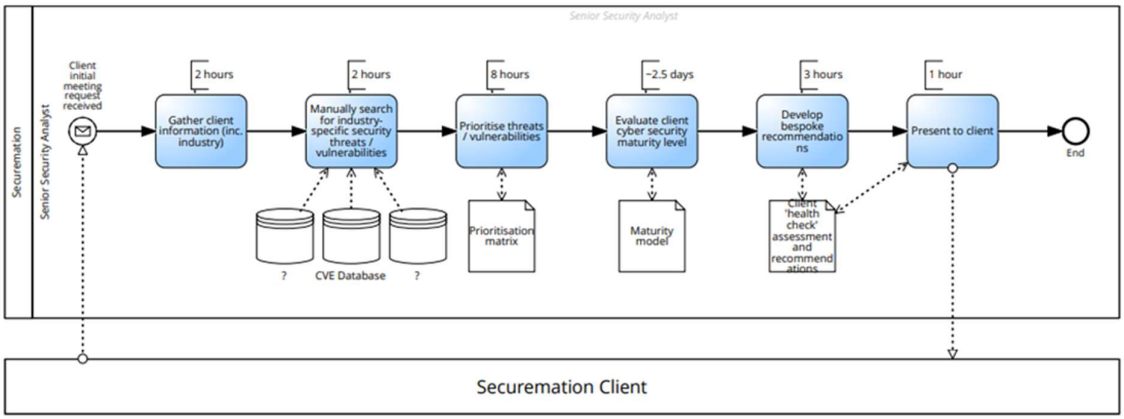


Figure 2. As-Is Business Process

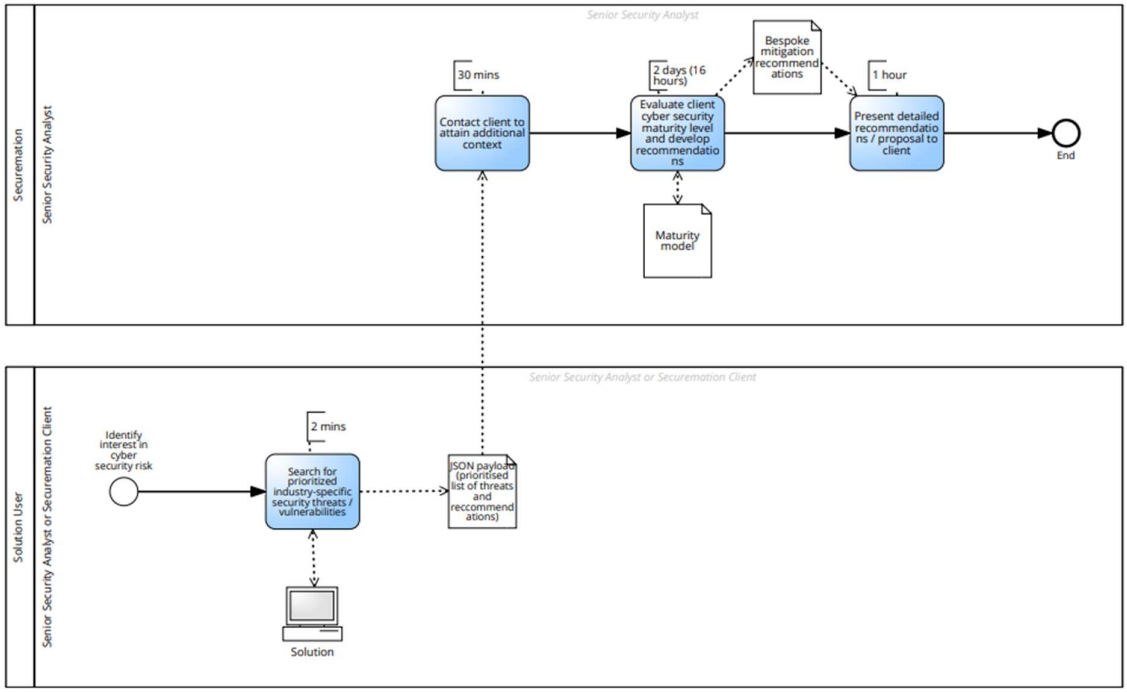


Figure 3 – To-Be Business Process

3.1 Business Risk

The design of the prototype microservice does not pose any known risk to Securemation's Business Architecture as outlined above. It is acknowledged that new risks may emerge as the solution is further developed and deployed in situ, though this is beyond the scope of this project.

ORGANISATION ARCHITECTURE

Detail of Securemation's broader organisation architecture is beyond the scope of this document.

4.1 Organisation Risk

In its current form, the design of the prototype microservice does not pose any known risk that might impact Securemation's Organisation Architecture.

INFORMATION ARCHITECTURE

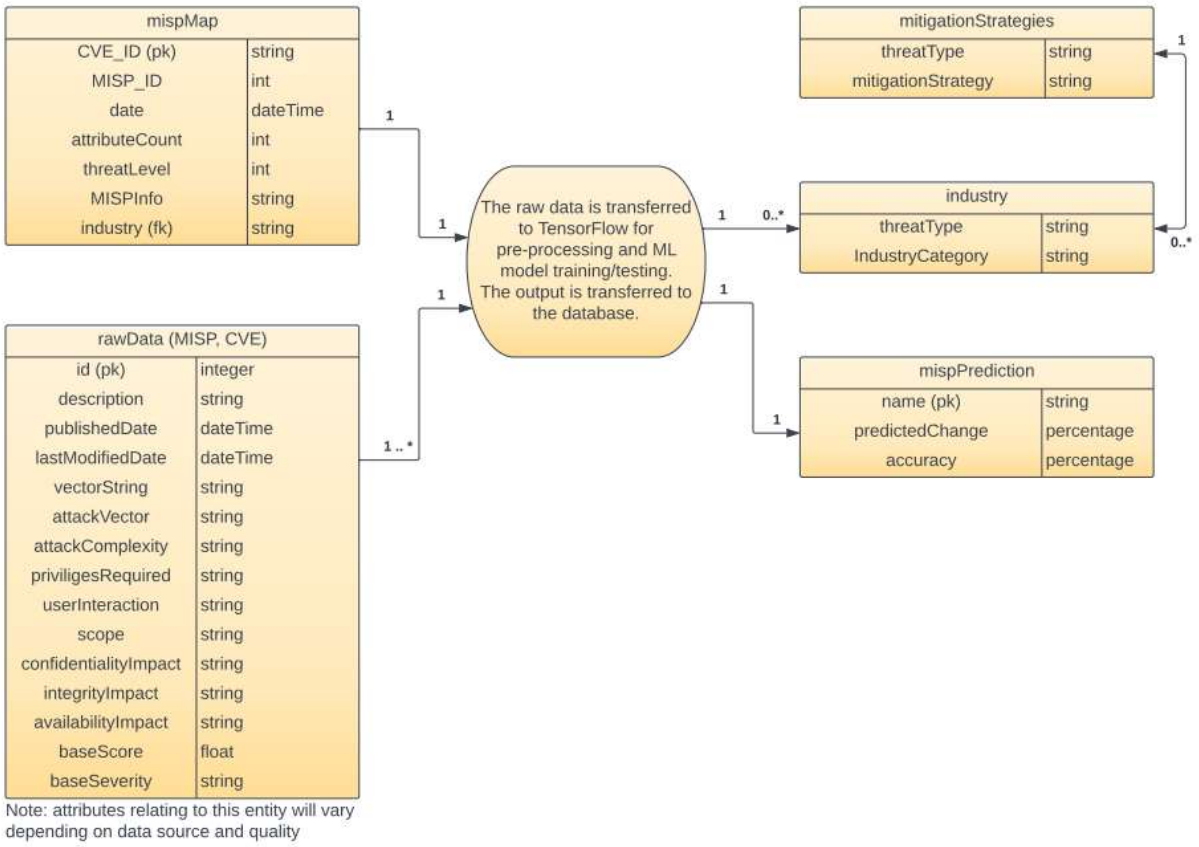
The information input by the user (their selected industry) is stored anonymously when querying the jBPM task. The information output is also housed in jBPM's process management on the KIE server and prioritisation and prediction results can be utilised in other BPM processes within jBPM. The prototype system's information can only be accessed locally at this level of development.

5.1 Information Risk

The design does not present any known risk to Securemation's information architecture as outlined above.

DATA ARCHITECTURE

The system collects data from two vetted, open-source repositories – MISP and CVE with raw data ingested in json format. Data sources have been screened for safety and verifiability of the overall dataset. Processing the structured qualitative and quantitative data is processed through data analytics and embedded methods for event descriptions. This structured dataset is stored in a MySQL database schema that is manipulated using the ETL tool Talend. MySQL utilises Access Control Lists (ACLs) for connections, queries and other operations as well as supports SSL-encrypted connections between MySQL clients and servers.



6.1 Data Risk

Risk to the data includes corruption of the sources due to the public platform mitigated by regular manual data backups of the processed MySQL database schema.

APPLICATION ARCHITECTURE

Applications required for functionality by this system include Docker, jBPM, 2 ML python components and Talend Open Studio. For testing this prototype system these software applications must be installed and up to date according to the version list as described in 2.6 Tools used. Accessory applications include Jupyter notebook for visualisation and notebook format for python modules and Postman for testing REST APIs. Communication between jBPM and Talend occurs via REST API. The python components connect to the MySQL database through the SQL Alchemy python module, that runs an engine managing DBAPI connections.

7.1 Application Risk

The design does not present any risk to Securemation's application architecture as outlined above.

TECHNOLOGY ARCHITECTURE

The infrastructure of this system is largely based on Microsoft OS however, jBPM and MySQL can be administered via Docker images. Talend is also deployable via a Docker image, however, this requires a paid subscription. To install the MISP database on Microsoft OS Docker was used to create the Linux VM required.

8.1 Technology Risk

The design does not present any known risk to Securemation's technology architecture.

SERVICE MANAGEMENT

The implementation and management of this design document will be handled by Securemation's Security team who will provide:

- Updates to applications as required
- Continue to manually map and test ML data/visualisations to suit Securemation's needs
- Audit process management of system API calls to jBPM
- Future developments (expanded upon in Section 13 - Future Considerations) for full deployment of system including
    - Automation of MISP event downloads
    - Automation of MISP mapping to CVEs
    - Integration of system output to other BPM within jBPM or direct to client

TEST CASE GUIDANCE

TEST CASE ID	TEST CASE	BUSINESS IMPACT
TCID-01	Test if jBPM KIE server deployment successful	LOW
TCID-02	Test if jBPM "front-end" BPM process instance creation is successful	LOW
TCID-03	Test if jBPM process instance returns full json without error in BPM	LOW
TCID-04	Test if jBPM DMN Mitigation 1,2,3 process instances map successfully	LOW
TCID-05	Test if Talend API "localhost:8088//industry?select={industry}" endpoints successful	LOW
TCID-06	Test if Talend API "localhost:8088//prediction" endpoint successful	LOW

TCID-07	Test if Talend tRESTResponse contains correct data	LOW
TCID-08	Test if TensorFlow and MySQL database connection is active	LOW
TCID-09	Test if TensorFlow Prioritisation model runs successfully and outputs 12 industry tables into MySQL schema	LOW
TCID-10	Test if TensorFlow Prediction model runs successfully and updates misp_prediction table in MySQL schema	LOW

*Recommended API Platform Postman – Postman system API collection available in artefact folder

USE CASES

USE CASE ID	USE CASE	BUSINESS IMPACT
UCID-01	User inputs correct industry name in jBPM process user task	LOW
UCID-02	User inputs incorrect industry name in jBPM process user task	LOW

UCID-01
System identifies that the input in the user task body call is correct and matches an industry that is available for query in the current prototype system. Returns the correct output for industry from the MySQL database schema.

UCID-02
System identifies the industry / query is incorrect and returns an error.

RECOVERY PLAN

12.1 Recovery plan overview

The design does not present any risk to Securemation's architecture. Recovery of the system will require maintenance by developers if the system does not run as prescribed. Recovery of the business process requires either direct access to MySQL schema/TensorFlow figures via Jupyter for manual revision or reversion to manual industry specific research process by a Security Analyst.

12.2 Recovery plan

TIME	TASK	EXPECTED RESULT	OBSERVED RESULT	PASS/FAIL
	Run BPM from last backed up MySQL schema	Output will be result from the most recent data backup		

IMPLEMENTATION PLAN

13.1 Implementation plan overview

To begin using this prototype microservice system each component must be implemented and tested stepwise according to the Implementation Schedule in section 13.2. The prototype is not developed for complete deployment and requires manual process for database connection. Data is automatically ingested from CVE via REST API to be fed into the TensorFlow Prioritisation component. However, MISP connection for the Prediction model has not been included into the system as it needs to connect to a localhost. A static download of example MISP events has been included for testing and demonstrative purposes under "MISP_atributes_all.json" from the default MISP feeds: "CIRCL OSINT Feed" and "The Botvrij.eu Data". This data was downloaded using MISP OPEN API endpoint (https://localhost/attributes) and detailed steps were described in Appendix 5.

The table below will provide all tasks necessary to deploy the design document mentioned above.

13.2 Implementation schedule

TIME	TASK	EXPECTED RESULT	OBSERVED RESULT	PASS/FAIL
	Task 1 – Environment set up	Required applications as per section 2.6 for microservice environment successfully installed. Python modules installed and up to at least recommended versions using pip.		
	Task 2 –Import artefacts	- jBPM Image pulled from Docker Hub and project assets imported - Python dependencies installed - Talend job imported - MySQL Image pulled from Docker Hub schema imported - Postman collection imported		
	Task 3 – Configure jBPM	Configuration of Work Item Handler within jBPM according to Appendix 3.4		
	Task 3 – Connect DB	Verify connection to MySQL schema in Talend query job under Metadata > DbConnections as described in Appendix 7.1.		

	Task 4 – Permit connection from ML to MySQL	In threat_priotisation.py and MISP_prediction.py the MySQL connection details must match the MySQL configuration. Currently set up as Environment Variables in Windows OS According to Appendix 8.4
	Task 5 – Run Prioritisation model	Model_training.pyt + Threat_prioritisation.py runs without error and creates/replaced 12 tables for each industry specified in Appendix 8.
	TCID-08	Tables created
	Task 6 – Run prediction model	Prediction model runs without error and creates/replaces misp_prediction schema in MySQL database.
	TCID-09	Table created
	Task 7 – Basic Run of Talend on local server	Success result on Talend for query job and deployed on localhost:8088
	TCID-05	Query success
	TCID-06	Query success
	TCID-07	Query success
	Task 8 – Deploy jBPM project on KIE server	Successful deployment of jBPM job "Securemation2.0" on KIE deployment server
	TCID-03	Query success using postman automated API collection
	Go/No Go	
	UCID-01	User can query jBPM "FrontEnd" and return full and correct json output

FUTURE CONSIDERATIONS

14.1 JBPM

- The final process instance variables can be utilised in other business processes within Securemation's current jBPM workflow or other BPM tools within the organisation.
- The result from jBPM could be deployed direct to client via a web application in the future.
- Extension or development of more complex development of BPM / DNMs with the data provided from machine learning models to create a larger or more specified output by using the available components within jBPM.

- Additional capability for jBPM to process instance variable data greater than 255 characters in order to provide alternate methods of housing and extraction of mitigation data.
- Add user privileges to the system for additional security through jBPM's user management according to organisational requirements.

14.2 Talend

- Adjust the SQL database query and overall REST response to business needs. Currently the output is the top 3 highest priority threats per industry however the query can be extended/altered in the tDBInput and tXMLMap components within Talend for more/less detail as desired. The limits selected are due to the 255 character limit of jBPM process instance variables.

14.3 TensorFlow (CVE+MISP) (Prioritisation)

- Automate running the script daily/weekly to update the database by creating a Task scheduler in Windows for a full deployment so updating the database does not have to be done manually.
- Add more databases to the existing prioritisation model. The scoring function can be modified to incorporate other data sources. However, some mapping procedures will be required (map to CVE dataset), and the data extracted from the new data sources must have similar categorization of threats (e.g. Gain Privilege, SQL injection, etc) and industries (healthcare, government, etc.) as the CVE dataset, because CVE dataset is adopted as a foundation dataset for building this prioritisation model.
- Development of a system that will automate the mapping of MISP events to related CVE vulnerabilities (extremely challenging due to data structure) – to improve on the prototype and further test the ML capabilities, more manual mapping can be done within Securemation with more specific cybersecurity background.
- Modify the prioritisation system and calculation scoring in Appendix 4.26 to return a more representative priority level of Low/Medium/High instead of the MISP multiplied Severity Score (i.e. the current output) as if there is a clear definition of the benchmark for Low/Medium/High priority across different industries. The calculation scoring can be adjusted to reflect more accurate organisational scoring practices.

14.4 TensorFlow (MISP) (Prediction)

- The prediction model can be altered to predict different time scales (i.e., daily, weekly, fortnightly, monthly etc.) or attribute categories by modifying value in the python file to further test suitability of the results related to organisational needs or to reflect the more current status of the cybersecurity attack landscape.
- Currently the model sometimes returns low prediction percentage however the system can be displayed a message if prediction percentage was unreliable.
- More MISP feeds incorporated into the prediction model by feeding from a more comprehensive MISP database.
- Automation of ingesting MISP attribute data into the model directly (rather than using the manually downloaded json) potentially more complicated due to how MISP runs on docker/Linux accessibility of python file.

14.5 MySQL

- Automatic backing up of schema to a suitable schedule for recovery plan, rather than manually.
- Include information on when schema was last updated in output from ML components by updating outputs to database.

- Add appropriate organisational user access privileges in MySQL schema or Securemation's database server.

14.6 Overall System

There were items that were considered out of scope during the development of this project due to time constraints. These items could be considered for further improving the functionality of the microservice system overall.

- Sorting industry sectors by subsector or geographic locale, this would require more specific datasets.
- Including the cybersecurity maturity matrix as an extension of the BPM based on Securemation's current maturity matrix questionnaire.
- Adding a web-scraping component for data ingestion. It was considered that this would not provide clear data sets however could be integrated as a separate component to see general news trends.
- Adding a Graphical User Interface (GUI) for client facing interactions with the system. The REST API endpoints could be utilised within the client facing application and display data in a clear and readable format. A GUI could also allow the user to select inputs from something similar to a dropdown menu to visualise the industries available. This would limit the available inputs a user could make, reducing the potential for incorrect inputs into the BPM.

CONTACT DETAILS – QUT TEAM MEMBERS

CHRISTINE CHOY
Solution Developer/Solution Tester

E: christine.choy@connect.qut.edu.au
P: 0420 585 282

CONNOR MCSWEENEY
Solution Developer/Solution Tester

E: connor.mcsweeney@connect.qut.edu.au
P: 0435 157 884

MASAYOSHI KAMIOKI
Solution Developer/Solution Tester

E: masayoshi.kamioki@connect.qut.edu.au
P: 0477 489 215

KOH HEI LUO
Solution Developer/Solution Tester

E: heiluo.koh@connect.qut.edu.au
P: 0435 098 347

BIANCA BEAUMONT
Project Manager / Team Leader / Business Analyst

E: be.beaumont@connect.qut.edu.au
P: 0421 107 271

DR BHARGAVI GOSWAMI
IFN711 Academic Supervisor

E: bhargavi.goswami@qut.edu.au
P: 0422 835 591

APPENDIX 1 – CALCULATION WEIGHTINGS

*NOTE: the values displayed in the following calculations are dummy values

CVE Model Threat Levels	Weight
Low	1
Medium	2
High	3
Critical	4

Threat Category	CVE Listed Priority (possible CVSS score)	Frequency	CVE Score per Threat lvl	Total CVE Score
A	Low	500	500	1140
	Medium	80	160	
	High	100	300	
	Critical	45	180	
B	Low	1000	1000	1250
	Medium	50	100	
	High	50	150	
	Critical	45	180	
C	Low	300	300	300
	Medium	0	0	
	High	0	0	
	Critical	0	0	

MISP Model Threat Levels ID	MISP Weight
4. Undefined	0
3. Low	1
2. Medium	2
1. High	3

Threat Category A

CVE ID	MISP ID	CVE Score	Frequency	MISP Model Threat Levels ID	MISP Weight	*MISP Multiplier	MISP Multiplied Severity Score
CVE-0001	MISP: 01	1	230	1	3	6.90	6.90
CVE-0002	MISP: 02	2	1279	2	2	25.58	51.16
CVE-0003	MISP: 01	3	230	3	1	2.30	6.90
CVE-0004	MISP: 03	4	400	4	0	0.00	4.00
CVE-0001	MISP: 02	1	1279	2	2	25.58	25.58
CVE-0005	NULL	4				0.00	4.00
			Total MISP Multiplied Severity Score for Threat Category A				**98.54**

* Frequency is reflective of instance attribute count
* MISP multiplier = Attribute Count x 0.01 x MISP Weight

Appendix 1.0 Estimated calculation scorings used in Severity scoring computation within the threat prioritisation model.

APPENDIX 2 – MITIGATIONS MAP

The following mitigation strategies can be accessed via the MitigationMapping.dmn Decision Model asset located in jBPM. Further documentation can be found in Appendix 3 – jBPM Configuration.

The mitigation strategies recommended below are general recommendations for each of the threat categories identified during the threat prioritisation process. The sources used to provide each of these strategies are listed below each threat type. Further development and use considerations for these strategies can be found in Section 13 - Further Considerations.

General software threats:

- Check and apply all available software updates immediately.
- Automate the update process and use a reliable update service provided directly from the vendor.

[Source: Cybersecurity and Infrastructure Security Agency - Essential Eight Maturity Model https://www.cisa.gov/uscert/security-publications/sql-injection]

Denial of Service (DoS):

- Regularly apply IT security patches to your website.
- Use a CDN or DDoS mitigation provider to front your online services.
- Be careful not to allow details about the address of your 'origin servers' to leak onto the internet, so that attackers cannot attempt to access it directly, bypassing the CDN or DDoS mitigation provider.
- Protect your 'origin servers' from direct access by implementing network filtering that limits access to traffic coming through your CDN or DDoS mitigation provider.
- Harden DNS servers against DDoS attacks.
- Consider mirroring part or all of your DNS infrastructure with DDoS resilient DNS providers.
- Run online services on different infrastructure to your critical business systems where practical.
- Have an incident response plan in place that accounts for DDoS attacks, and conduct exercises to ensure that the plan is effective.

[Source: Australian Cyber Security Centre https://www.cyber.gov.au/acsc/view-all-content/threats/denial-service]

SQL Injection:

- Deny access to the internet except through proxies for Store and Enterprise servers and workstations.
- Implement firewall rules to block or restrict internet and intranet access for databasesystems.
- Implement firewall rules to block known malicious IP addresses.
- Harden internal systems against the potential threat posed by a compromised system on the local network.
- Secure both the operating system and the application.
- Update and patch production servers regularly.
- Disable potentially harmful SQL stored procedure calls.
- Deny extended URLs.
- Sanitize/validate input.
- Ensure error messages are generic and do not expose too much information.
- Use principles of least privilege.
- Enforce best practice password and account policies.

- Document all database accounts, stored procedures, and prepared statements along with their uses.
- Perform regular audits and penetration testing.

[Source: Cybersecurity and Infrastructure Security Agency https://www.cisa.gov/uscert/security-publications/sql-injection]

Gain Privilege (Privilege Escalation Attacks):

- Fully manage the identity lifecycle, including provisioning and de-provisioning of identities and accounts to ensure there are no orphaned accounts that could be hijacked.
- Use a password management solution to consistently apply strong credential management practices (discovery, vaulting, central management, check-in, check-out) for both human and machines. This also entails eliminating default and hardcoded credential.
- Enforce least privilege: Remove admin rights from users and reduce application and machine privileges to the minimum required. Just-in-time access should also be implemented to reduce persistent or standing privileges.
- Apply advanced application control and protection to enforce granular control over all application access, communications, and privilege elevation attempts.
- Monitor and manage all privileged sessions to detect and quickly address any suspicious activity that might indicate a hijacked account or an illicit attempt at privilege escalation or lateral movement.
- Harden systems and applications: This complements the principle of least privilege and can involve configuration changes, removing unnecessary rights and access, closing ports, and more. This improves system and application security and helps prevent and mitigate the potential for bugs that leave vulnerability to injection of malicious code (i.e. SQL injections), buffer overflows, etc. or other backdoors that could allow privilege escalation.
- Vulnerability management: Continuously identify and address vulnerabilities, such as with patching, fixing misconfigurations, eliminating default and/or embedded credentials, etc.
- Secure remote access should always be monitored and managed for any form of privileged access since attacks can occur horizontally and vertically to exploit privileges.

[Source: Morey J. Haber (2021) Privilege Escalation Attack and Defense Explained https://www.beyondtrust.com/blog/entry/privilege-escalation-attack-defense-explained]

Buffer Overflow:

- Check and apply all available software updates immediately.
- Automate the update process and use an reliable update service provided directly from the vendor.

Bypass: check this

- Check and apply all available software updates immediately.
- Automate the update process and use a reliable update service provided directly from the vendor.

XSS:

- Allow own domain and subdomains only.
- Restrict to self, allow inline JavaScript.
- Allow everything from anywhere but block third-part scripts.
- Allow to self and approved external sources.
- Force all requests over HTTPS.
- Preventing users from posting HTML code into form inputs.
- Use a web application firewall.

[Source: Australian Cyber Security Centre https://www.cyber.gov.au/acsc/view-all-content/publications/protecting-web-applications-and-users]

Code Execution:

- Check and apply all available software updates immediately.
- Automate the update process and use an reliable update service provided directly from the vendor.
- Don't download applications from third-party download sites.
- Don't click on online ads to download applications and do use ad-blocking software.
- Don't download and install applications from peer to peer networks – you never know who has changed the files.
- Don't click on links in emails or instant messages, or execute attachments unless you are sure they are legitimate. Use a spam filter to protect yourself from malicious messages.
- Don't install applications received from contacts, say via email or USB sticks, without scanning them with your anti-virus application first.
- Use anti-virus software and automatically download signature updates daily. Learn about anti-virus software.
- Keep all your other software up to date too. Learn about updating sofware.
- Use strong passwords and passphrases. Learn how to create – and remember – strong passwords.
- Backup your files regularly – ideally every day. Learn about how to back up files.
- Disable Microsoft Office macros. (Macros are small programs used to automate simple tasks in
- Microsoft Office documents but can be used maliciously – visit the Microsoft website for information on disabling macros in your version of Office).
- Use safe behaviour online. Learn about how to use email safely and browse the web safely.
- Stay informed on the latest threats – sign up for the ACSC's Alert Service.
- Regularly check the software installed on your computer, tablet and other devices and uninstall any programs or software that is unused. If you see new programs or software that you did not agree to install, search the program name or ask your local computer repairer or retailer about the program, to see whether it is safe to use.

[Source: Australian Cyber Security Centre https://www.cyber.gov.au/acsc/view-all-content/threats/malware]

Directory Traversal:

- Prefer working without user input when using file system calls
- Use indexes rather than actual portions of file names when templating or using language files (ie value 5 from the user submission = Czechoslovakian, rather than expecting the user to return "Czechoslovakian")
- Ensure the user cannot supply all parts of the path – surround it with your path code
- Validate the user's input by only accepting known good – do not sanitize the data
- Use chrooted jails and code access policies to restrict where the files can be obtained or saved to
- If forced to use user input for file operations, normalize the input before using in file io API's, such as normalize().

[Source: Open Web Application Security Project Foundation Inc. https://owasp.org/www-community/attacks/Path_Traversal]

Cross-site request forgeries (CSRF):

- Check if your framework has built-in CSRF protection and use it. If framework does not have built-in CSRF protection add CSRF tokens to all state changing requests (requests that cause actions on the site) and validate them on backend.
- For stateful software use the synchronizer token pattern

- For stateless software use double submit cookies
- Implement at least one mitigation from Defense in Depth Mitigations section.
- Consider SameSite Cookie Attribute for session cookies but be careful to NOT set a cookie specifically for a domain as that would introduce a security vulnerability that all subdomains of that domain share the cookie. This is particularly an issue when a subdomain has a CNAME to domains not in your control.
- Consider implementing user interaction based protection for highly sensitive operations. # Consider the use of custom request headers.
- Consider verifying the origin with standard headers.
- Do not use GET requests for state changing operations. If for any reason you do it, protect those resources against CSRF.

[Source: Open Web Application Security Project Foundation Inc. https://cheatsheetseries.owasp.org/cheatsheets/CrossSite_Request_Forgery_Prevention_Cheat_Sheet.html]

HTTP Response Splitting:

- Carefully validate and sanitize any user-provided content that might be used to compose response headers.
- Encode dangerous characters such as \r and \n.
- Don't let users control the whole value in the `Location` header.
- When using a Content-Security-Policy, it should also be defined in the HTML

[Source: https://resources.infosecinstitute.com/topic/http-response-splitting-attack/]

File Inclusion:

- Install a Web Application Firewall (WAF to monitor user inputs and filter out malicious requests.
- Utilize crowdsourcing technology to maintain a continually updated database of compromised domains that serve as centralized distribution points for malware injection
- Restrict execution permission for upload directories and maintain a whitelist of allowable file types.

[Source: https://www.imperva.com/learn/application-security/rfi-remote-file-inclusion/]

Memory Corruption

- Educate employees who can perform money transfers about spear phishing emails
- Don't click on links in emails or messages, or open attachments, from people or organisations you don't know.
- Be especially cautious if messages are very enticing or appealing (they seem too good to be true) or threaten you to make you take a suggested action.
- Before you click a link (in an email or on social media, instant messages, other web pages, or other means), hover over that link to see the actual web address it will take you to (usually shown at the bottom of the browser window). If you do not recognise or trust the address, try searching for relevant key terms in a web browser. This way you can find the article, video or web page without directly clicking on the suspicious link.
- If you're not sure, talk through the suspicious message with a friend or family member, or check its legitimacy by contacting the relevant business or organisation (using contact details sourced from the official company website).
- Use a spam filter to block deceptive messages from even reaching you.

- Understand that your financial institution and other large organisations (such as Amazon, Apple, Facebook, Google, PayPal and others) would never send you a link and ask you to enter your personal or financial details.
- Use safe behaviour online. Learn how to use email safely and browse the web safely.
- Stay informed on the latest threats – sign up for the ACSC Alert Service. You can also find information about the latest scams on the Australian Government's Scamwatch website.

[Source: https://www.cyber.gov.au/acsc/view-all-content/publications/strategies-mitigate-cyber-security-incidents-mitigation-details]

APPENDIX 3 – JBPM CONFIGURATION

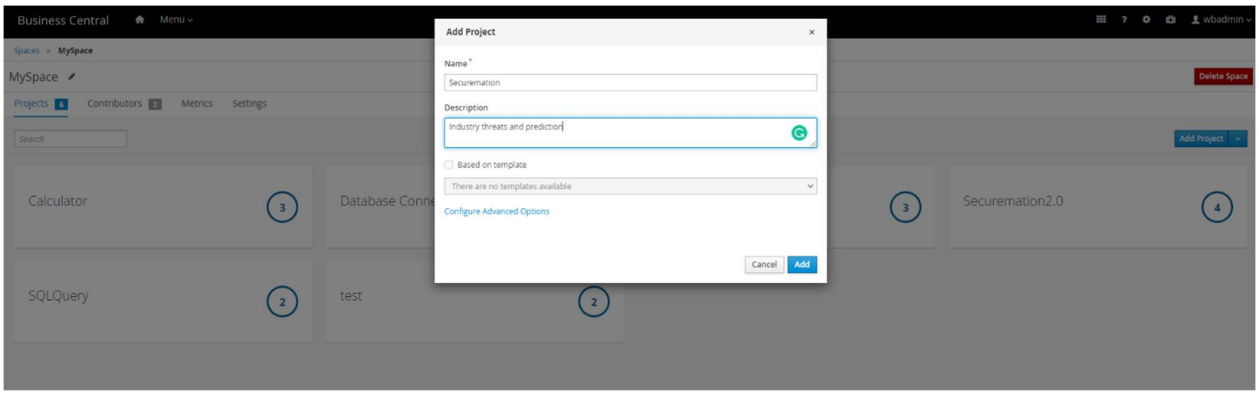
Included in this section are the installation and implementation steps required to set up jBPM in addition to current configuration settings.

Installation and Setup:

[Optional] Pull the Docker Image from 'connoraus/jbpm' and create a Docker container.

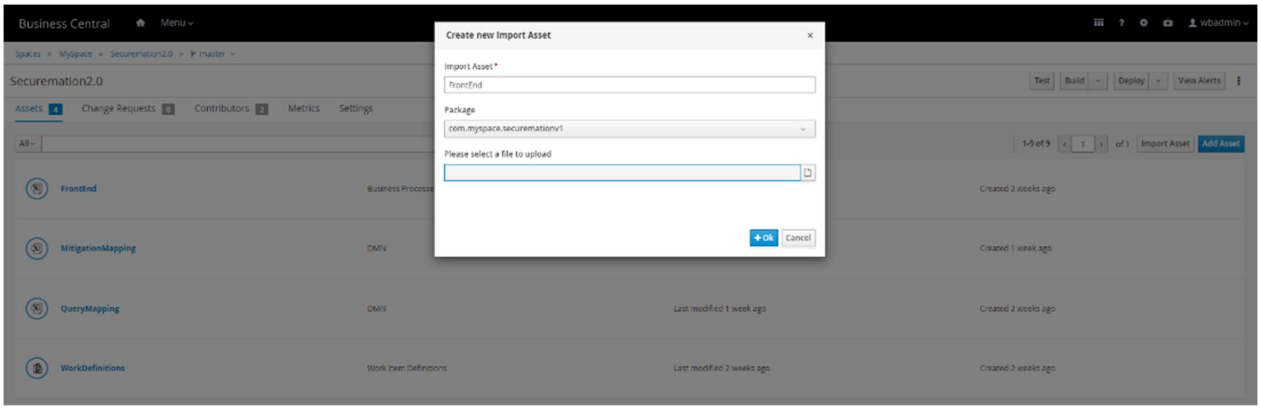
1. Once the container has been successfully imported/created, sign in to jBPM and select/create a 'Space' to house the project.

2. Next, create of a new project within jBPM. Select the 'Add Project' option and provide a relevant 'Name' and 'Description' as inputs.



Appendix 3.1 Add jBPM project

3. Appendix 3.2 displays the four assets within the jBPM project. Each of these assets should be imported directly from the project file "..Artefacts/jBPM/". This ensures the import of each asset's configuration settings and data.

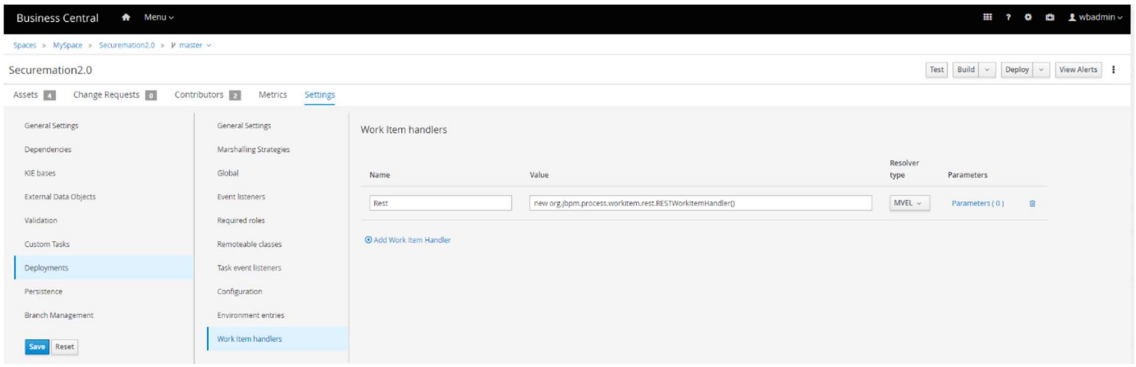


Appendix 3.2 Import Assets from ../Artefacts/jBPM/

4. Ensure that ServiceTask is set to ‘ON’ within the Custom Tasks Administration settings. This can be access by selecting the small cog (settings) shown in top right of Appendix 3.3, then selecting Custom Tasks Administration.

Appendix 3.3 Service Task

5. Selecting Settings, Deployments, Work Item handlers, and configuring the Work Item handler as seen in Appendix 3.4 will ensure that each of the API Service tasks within FrontEnd.jbpm are able to operate as expected.

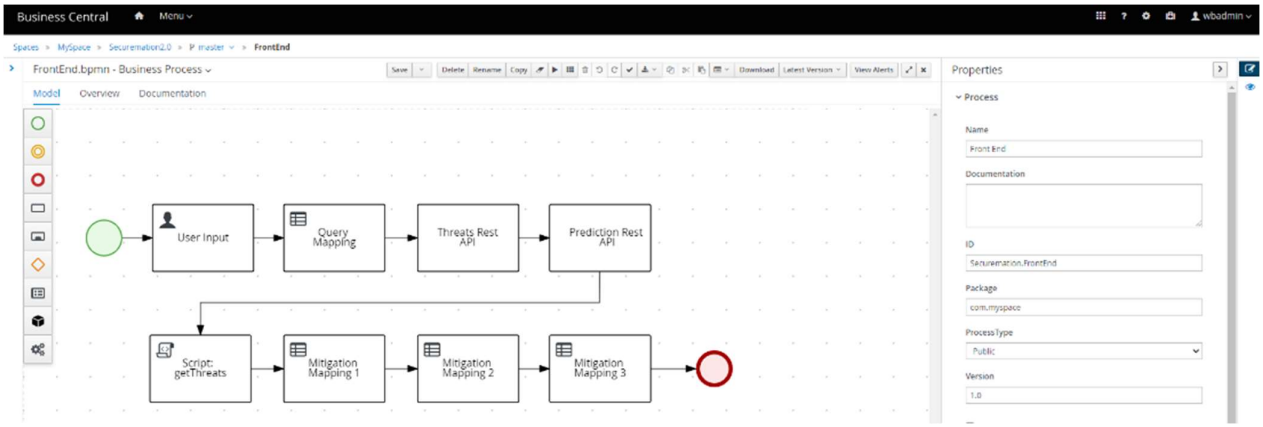


Appendix 3.4 Add Work Item Handlers

Existing Configuration:

Appendix 3.5 displays each of the processes within the FrontEnd.jbpm. The type of type of each process is as follows:

- User Task
  - User input
- Business Rule Task
  - Query Mapping
  - Mitigation Mapping 1, 2, and 3
- API Rest Service
  - Threats Rest API
  - Prediction Rest API
- Script Task
  - Script: getThreats



Appendix 3.5

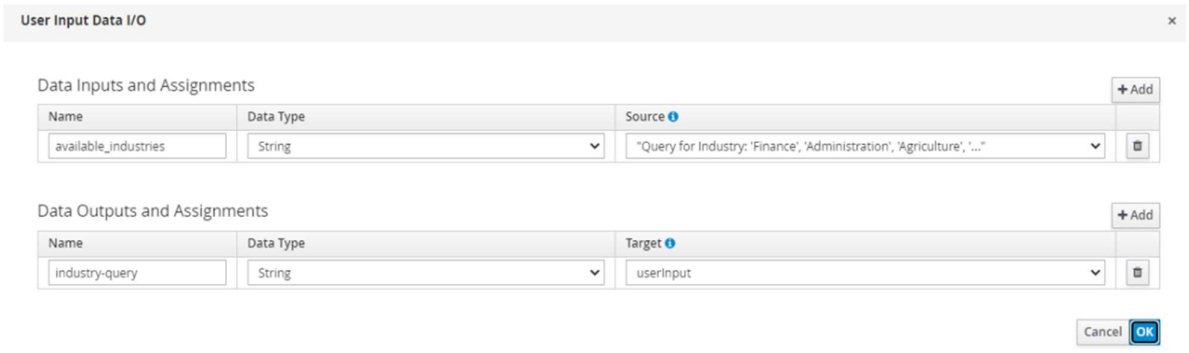
Expanding Process Data within the Properties tab will display all the process instance variables within FrontEnd.bpmn. They should be configured as per Appendix 3.6.

Process Variables

Name	Data Type	Tags
threatsURL	String	
threatsResul	Object	
userInput	String	
predictionRe	Object	
threatType1	String	
threatType2	String	
threatType3	String	

Appendix 3.6

Select the User Input User Task, Properties, Implementation/Execution and the Edit symbol next the Assignments to access the input/output (I/O) data for this task, as seen in Appendix 3.7. Data Inputs lists all the possible valid user inputs as a string data type under Source. Data Outputs contains the expected user input (industry-query) and sets that to the UserInput process instance variable.



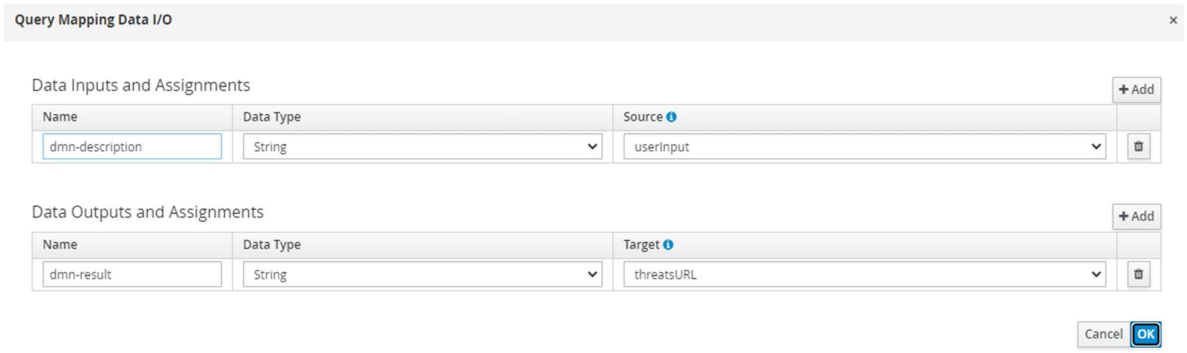
Appendix 3.7

Ensure that the Actors section includes all users that should have access to the project. This can be accessed by selecting the User Input User Task followed by selecting Properties. An example is shown in Appendix 3.8.

Appendix 3.8

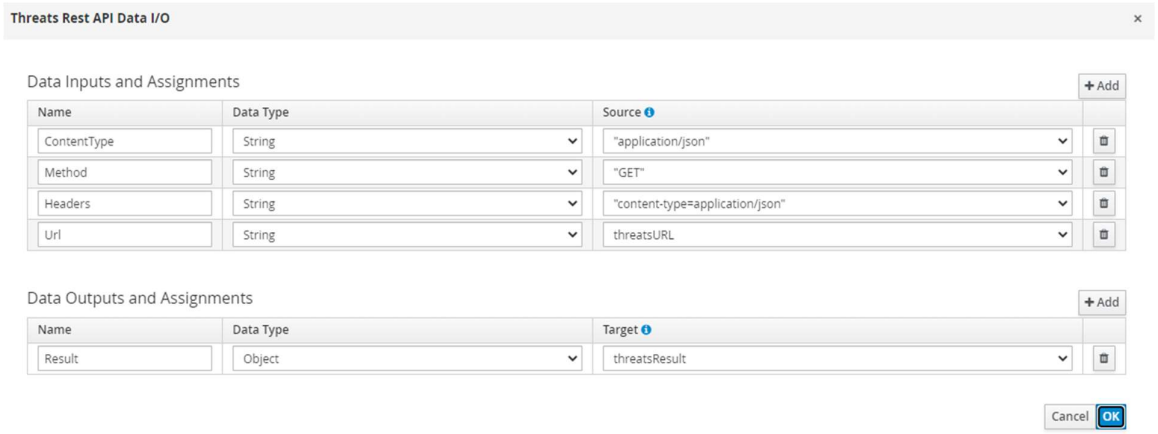
Appendix 3.9 displays the I/O data for the Query Mapping Decision Model. It takes the userInput process instance variable setting it to the dmn-description variable used by the linked DMN asset (QueryMapping.dmn). It returns the dmn-result and is set to the threatsURL process instance variable.

* Note: information relating to the QueryMapping DMN configuration and process can be found below.



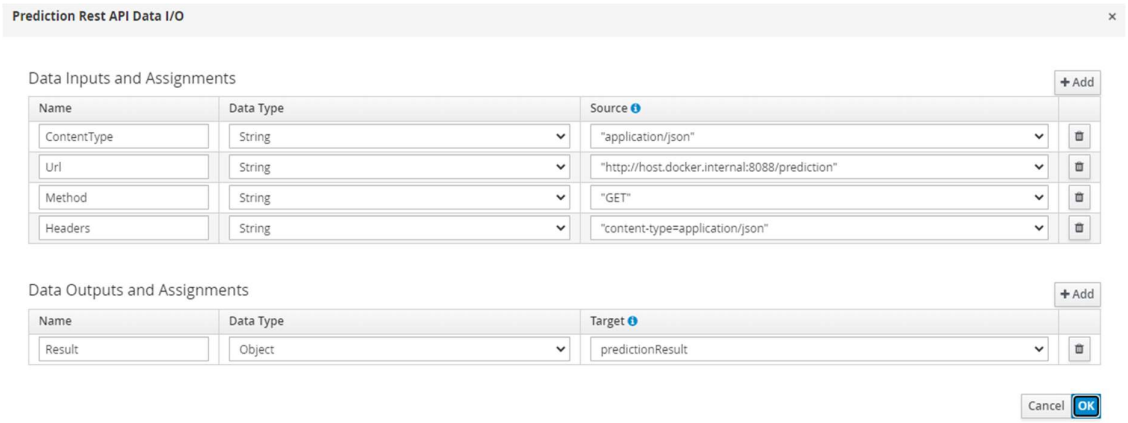
Appendix 3.9

Appendix 3.10 shows how the Threats Rest API Rest Service Task uses the threatsURL process instance variable and further custom properties to execute and REST query. This allows it to communicate directly with Talend – specifically the Talend job tasked with extracting and returning data from the MySQL database. The result is set to the threatsResults process instance variable.



Appendix 3.10

Appendix 3.11 shows how the Prediction Rest API Rest Service Task uses a hard coded URL and further custom properties to execute and REST query. This allows it to communicate directly with Talend – specifically the Talend job tasked with extracting and returning data from the MySQL database. The result is set to the predicitonResults process instance variable.



Appendix 3.11

Appendix 3.12

Appendix 3.12 displays the custom code executed by the Script: getThreats Script Task. This code is housed within the Script field and can be found by selecting the Task, Properties, and expanding Implementation/Execution. It contains three 'for' loops that work to match any of the threat types listed within the threatTypes string array to the 'temp' variable. The 'temp' variable is the threatResult process instance variable converted from JSON format to a string. Each for loop with set a threat type to one of the threatType process instance variables - there are three. Adding or removing any 'for' loops will affect the number of threat types found and must be matched by an updated SQL query within Talend as well as an updated number of threatType process instance variables and Mitigation Mapping Business Rule tasks.

Script

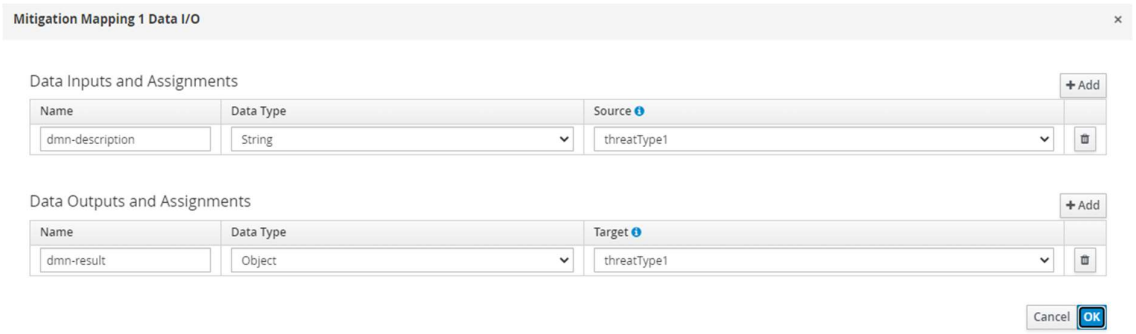
```
var temp = kcontext.getVariable("threatsResult");
String tempString = temp.toString();
int threatCount = 0;

String[] threatTypes = {"Gain Privilege", "Denial of Serv
    "Buffer Overflow", "Bypass", "XSS", "Code Execution (
    "Directory Traversal", "Cross-site request forgeries

for (int i = 0; i < threatTypes.length; i++)
{
    if (tempString.contains(threatTypes[i])) {
        if (threatCount == 0)
```

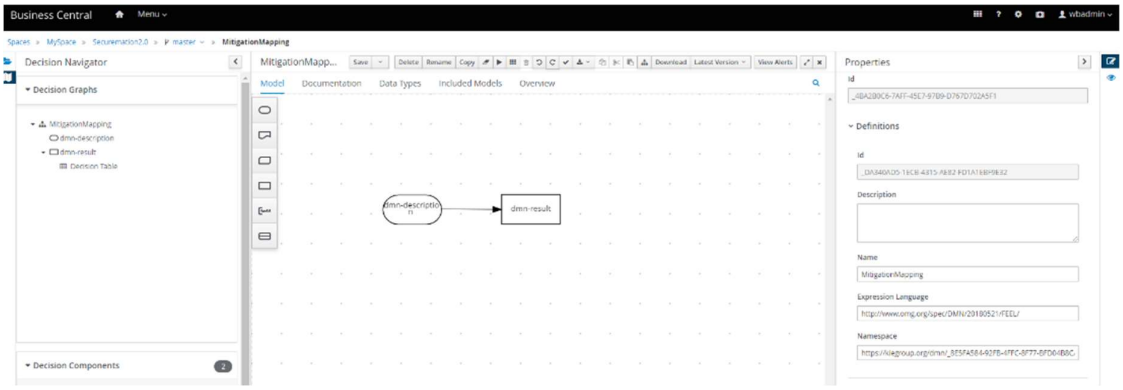
Java

Appendix 3.13 shows the I/O variables for the Mitigations Mapping Business Rule Task. It takes the threatType1 process instance variable, setting it to the dmn-description to be used by the MitigationMapping.dmn. The output is set to threatType1. This configuration and process is identical for each of the MitigationMapping Business Rule Tasks within FrontEnd.jbpm excluding the I/0 variable name – the number is incremented for each variable (e.g. threatType2).



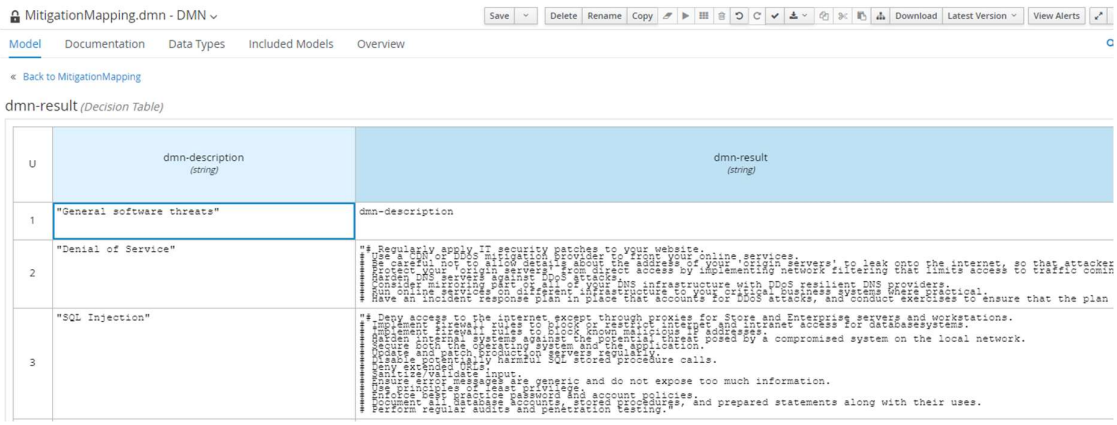
Appendix 3.13

Appendix 3.14 displays the Input Data and the DMN Decision relationship within MitigationMapping.dmn. It is important that the Name listed in the properties tab matches the DMN Model name listed within the properties of the linked Business Rule Task within FrontEnd.jbpm. It is also important that the names of each of these nodes match the names of the I/O variables used within the linked Business Rule Task.



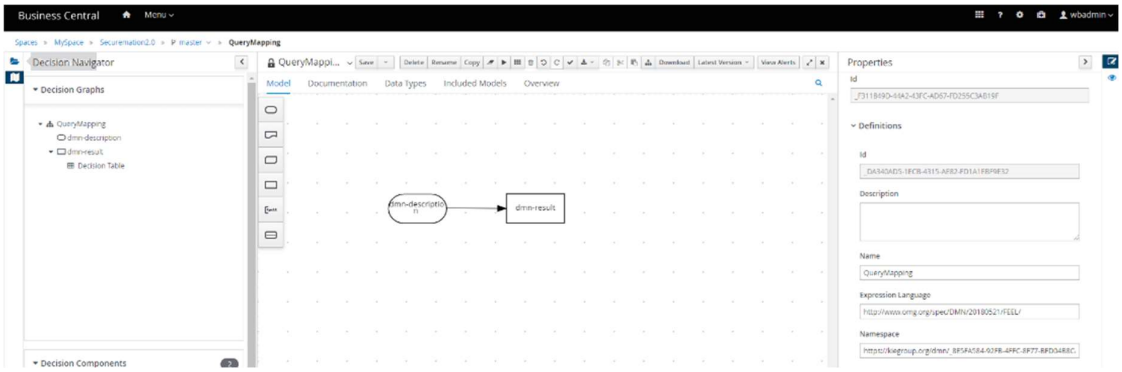
Appendix 3.14

Appendix 3.15 displays an example of the description and result mapping relationships. The value of the input data (dmn-description) must match one of the cells within the dmn-description column of this table for the mapping to be successful. Once achieved, value of the matching dmn-result cell will be returned to the Business Rule Task within FrontEnd.jbpm. This data can be further explored and edited to meet the requirements of the project.



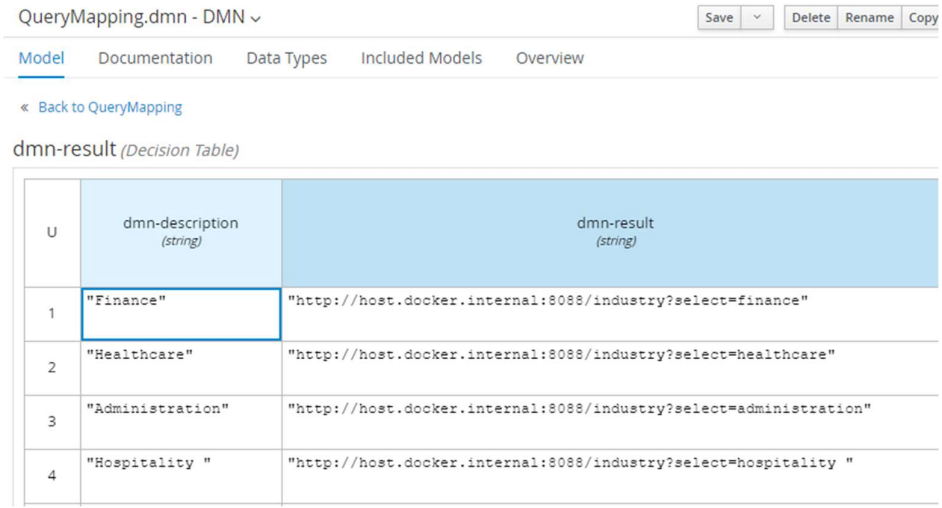
Appendix 3.15

Appendix 3.16 displays the Input Data and the DMN Decision relationship within QueryMapping.dmn. The rules for configuration match that of the MitigationMapping.dmn.



Appendix 3.16

Appendix 3.17 displays an example of the description and result mapping relationships within QueryMapping.dmn. The values within the dmn-description column must match the valid inputs provided to the user from the UserInput User Task within FrontEnd.jbpm.



Appendix 3.17

APPENDIX 4 – TENSORFLOW (CVE+MISP) ANALYSIS CONFIGURATION

Retrieve CVE Data by industry keywords from NVD using the getdata(industry) function

*This data is used as a foundation for the scoring function.

1. Define a function to extract information such as CVE ID, CVE Description, CVSS Version 3 Score, published date, last Modified date from the result dictionary. The CVSS Version 3 Score is retrieved in the form of a dictionary, therefore requiring the use of the pd.series function to convert it from a dictionary to a series. The output of this function is a merged dataframe of the result series of the CVSS v3 Score and the other information extracted.

```
#API Call get data from NIST
def getdata(industry):
    response = requests.get("https://services.nvd.nist.gov/rest/json/cves/1.0/?resultsPerPage=2000&keyword="+industry)
    Dictionary = response.json()
    id_ = []
    description = []
    score = []
    published_ = []
    lastModified_ = []
    dict_ = Dictionary['result']['CVE_Items']
    for element in dict_:
        id_.append(element['cve']['CVE_data_meta']['ID'])
        for i in element['cve']['description']['description_data']:
            description.append(i['value'])
        score.append(element['impact'])
        published_.append(element['publishedDate'])
        lastModified_.append(element['lastModifiedDate'])

    lists = [id_, description, score, published_, lastModified_]
    df = pd.DataFrame(lists).transpose()
    df.columns = ['CVE_ID', 'description','score','published_','lastModified']

    df = df[df.score != {}]
    df.reset_index(drop=True, inplace=True)
    df_score = df["score"].apply(pd.Series)
    df_metricV3 = df_score["baseMetricV3"].apply(pd.Series)
    df_cvssV3 = df_metricV3["cvssV3"].apply(pd.Series)
    result_df = pd.concat([df, df_cvssV3], axis=1)
    return result_df
```

Appendix 4.1

2. Retrieve CVE data for different industries (e.g. Administrative and support services industry) and the keyword used is 'administrative service'. The list of keywords used for data retrieval for each industry is listed in **Table 1**. Drop rows with NULL values. Add a new column named 'Industry' to the result dataframe from the getdata(industry) function.

```
# Administrative
AS = getdata('administrative service')
AS.drop(AS[AS['attackComplexity'].isnull() == True].index, inplace=True)
AS['Industry'] = 'administration'

# Accommodation and food services
#'accommodation' or 'food' or 'beverage' or 'hotel' or 'catering' or 'hospitality',
#Accom = getdata('accommodation')
Food = getdata('food')
Beverage = getdata('beverage')
Hotel = getdata('hotel')
Catering = getdata('catering')
Hospitality = getdata('hospitality')
Accom_food = pd.concat([Food, Beverage, Hotel, Catering, Hospitality], ignore_index=True, sort=True)
Accom_food.drop(Accom_food[Accom_food['attackComplexity'].isnull() == True].index, inplace=True)
Accom_food['Industry'] = 'hospitality'

#Agriculture, Forestry & Fisheries
Agri = getdata('agriculture')
Farm = getdata('farm')
#Fish = getdata('fisheries')
Agri_Forest_Fish = pd.concat([Agri, Farm], ignore_index=True, sort=True)
Agri_Forest_Fish.drop(Agri_Forest_Fish[Agri_Forest_Fish['attackComplexity'].isnull()
                                                    == True].index, inplace=True)
Agri_Forest_Fish['Industry'] = 'agriculture'
```

```
#Education
education = getdata('education')
student = getdata('student')
uni = getdata('university')
insti = getdata('institute')
school = getdata('school')
education_industry = pd.concat([education, student, uni, insti, school], ignore_index=True, sort=True)
education_industry.drop(education_industry[education_industry['attackComplexity'].isnull()
                                                    == True].index, inplace=True)
education_industry['Industry'] = 'education'

#'financial' or 'bank' or 'insurance' or 'finance'
financial = getdata('financial')
bank = getdata('bank')
insurance = getdata('insurance')
finance = getdata('finance')
investment = getdata('investment')
loan = getdata('loan')
mortgage = getdata('mortgage')
wallet = getdata('wallet')
forex = getdata('forex')
treasury = getdata('treasury')
cryptocurrency = getdata('cryptocurrency')

finance_industry = pd.concat([financial, bank, uni, insurance, finance, investment,
                              loan, mortgage, wallet, forex, treasury,cryptocurrency],
                             ignore_index=True, sort=True)
finance_industry.drop(finance_industry[finance_industry['attackComplexity'].isnull()
                                                    == True].index, inplace=True)
finance_industry['Industry'] = 'finance'
```

```
#Government
government = getdata('government')
government.drop(government[government['attackComplexity'].isnull()
                                                        == True].index, inplace=True)
government['Industry'] = 'government'
government

#Health & Medical
health = getdata('health')
medical = getdata('medical')
hospital = getdata('hospital')
clinic = getdata('clinic')
nurse = getdata('nursing')
patient = getdata('patient')
health_industry = pd.concat([health, medical, hospital, clinic, nurse,
                             patient],
                            ignore_index=True, sort=True)
health_industry.drop(health_industry[health_industry['attackComplexity'].isnull()
                                                        == True].index, inplace=True)
health_industry['Industry'] = 'healthcare'

#Rental, real estate & hiring
rental = getdata('rental')
real_es = getdata('real estate')
rental_real = pd.concat([rental, real_es],
                            ignore_index=True, sort=True)
rental_real.drop(rental_real[rental_real['attackComplexity'].isnull()
                                                        == True].index, inplace=True)
rental_real['Industry'] = 'realestate'
```

```
#Retail Wholesale
retail = getdata('retail')
wholesale = getdata('wholesale')
supplier = getdata('supplier')
retail_wholesale = pd.concat([retail, wholesale,supplier],
                             ignore_index=True, sort=True)
retail_wholesale.drop(retail_wholesale[retail_wholesale['attackComplexity'].isnull()
                                                    == True].index, inplace=True)
retail_wholesale['Industry'] = 'retail'

#Tourism
tourism = getdata('tourism')
travel = getdata('travel')
tour_indus = pd.concat([tourism, travel],
                             ignore_index=True, sort=True)
tour_indus.drop(tour_indus[tour_indus['attackComplexity'].isnull()
                                                    == True].index, inplace=True)
tour_indus['Industry'] = 'tourism'

#Postal
postal = getdata('postal')
warehouse = getdata('warehouse')
airline = getdata('airline')
bus = getdata('bus booking')
ticket = getdata('bus ticket')
air_cargo = getdata('air cargo')
postal_warehouse_transport = pd.concat([postal, warehouse, airline, bus, ticket, air_cargo],
                             ignore_index=True, sort=True)
postal_warehouse_transport.drop(postal_warehouse_transport[postal_warehouse_transport['attackComplexity'].isnull()
                                                    == True].index, inplace=True)
postal_warehouse_transport['Industry'] = 'transportation'
```

```
#Telecom
telecom = getdata('telecom')
telecom.drop(telecom[telecom['attackComplexity'].isnull() == True].index, inplace=True)
telecom['Industry'] = 'telecommunication'

#Manufacturing industry
PLM = getdata("Oracle Agile PLM")
PLM.drop(PLM[PLM['attackComplexity'].isnull() == True].index, inplace=True)
PLM['Industry'] = 'manufacturing'
```

Appendix 4.2

	CVE_ID	description	score	published_	lastModified	attackComplexity	attackVector	availabilityImpact	baseScore	baseSeverity
0	CVE-2022-22977	VMware Tools for Windows(12.0.0, 11.x.y and 10...	{'baseMetricV3': {'cvssV3': {'version': '3.1',...	2022-05-24T19:15Z	2022-06-09T15:52Z	LOW	LOCAL	HIGH	7.1	HIGH
2	CVE-2022-29171	Sourcegraph is a fast and featureful code sear...	{'baseMetricV3': {'cvssV3': {'version': '3.1',...	2022-05-06T00:15Z	2022-05-17T13:39Z	LOW	NETWORK	HIGH	7.2	HIGH
3	CVE-2021-25309	The telnet administrator service running on po...	{'baseMetricV3': {'cvssV3': {'version': '3.1',...	2021-03-02T01:15Z	2022-04-26T16:00Z	LOW	NETWORK	HIGH	9.8	CRITICAL

	confidentialityImpact	integrityImpact	privilegesRequired	scope	userInteraction	vectorString	version	Industry
	HIGH	NONE	LOW	UNCHANGED	NONE	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	3.1	Administrative and support services
	HIGH	HIGH	HIGH	UNCHANGED	NONE	CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H	3.1	Administrative and support services
	HIGH	HIGH	NONE	UNCHANGED	NONE	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	3.1	Administrative and support services

Appendix 4.3 Result dataframe for industry 'Administrative'

Industry	Keywords
Administrative and support services industry	'administrative service'
Accommodation & food services	'food', 'beverage', 'hotel', 'catering' and 'hospitality'
Agriculture, Forestry & Fisheries	'agriculture', 'farm'
Education	'education', 'student', 'university', 'institute', 'school'
Finance & Insurance	'financial', 'bank', 'insurance', 'finance', 'investment', 'loan', 'mortgage', 'wallet', 'forex', 'treasury', 'cryptocurrency'
Government	'government'
Health & Medical	'health', 'medical', 'hospital', 'clinic', 'nursing', 'patient'
Rental, real estate & hiring	'rental', 'real estate'
Retail & wholesale trade	'retail', 'wholesale', 'supplier'
Tourism	'tourism', 'travel'
Transport, postal and warehousing	'postal', 'warehouse', 'airline', 'bus booking', 'bus ticket', 'air cargo'
Telecommunications	'telecom'
Manufacturing	'Oracle Agile PLM'

Table 4.1

3. Merge all the result dataframe together. Remove the duplicate column, i.e. CVE ID.

```
#Concatenate all
df = pd.concat([AS, Accom_food, Agri_Forest_Fish, education_industry, finance_industry,
                government, health_industry, rental_real, retail_wholesale, tour_indus,
                postal_warehouse_transport, telecom, PLM], ignore_index=True, sort=True)
df.drop_duplicates(subset='CVE_ID', keep='first', inplace=False)
df.reset_index(drop=True, inplace=True)
df
```

Appendix 4.4

	CVE_ID	description	lastModified	published_	score	attackComplexity	attackVector	availabilityImpact	baseScore	baseSeverity	confidentialityImpact
0	CVE-2022-22977	VMware Tools for Windows(12.0.0, 11.x.y and 10...	2022-06-09T15:52Z	2022-05-24T19:15Z	{'baseMetricV3': {'cvssV3': {'version': '3.1',...	LOW	LOCAL	HIGH	7.1	HIGH	HIGH
1	CVE-2022-29171	Sourcegraph is a fast and featureful code sear...	2022-05-17T13:39Z	2022-05-06T00:15Z	{'baseMetricV3': {'cvssV3': {'version': '3.1',...	LOW	NETWORK	HIGH	7.2	HIGH	HIGH
2	CVE-2021-25309	The telnet administrator service running on po...	2022-04-26T16:00Z	2021-03-02T01:15Z	{'baseMetricV3': {'cvssV3': {'version': '3.1',...	LOW	NETWORK	HIGH	9.8	CRITICAL	HIGH
3	CVE-2017-18101	Various administrative external system import ...	2022-04-22T20:40Z	2018-04-10T13:29Z	{'baseMetricV3': {'cvssV3': {'version': '3.1',...	LOW	NETWORK	NONE	6.5	MEDIUM	LOW
4	CVE-2020-25154	An open redirect vulnerability in the administ...	2022-04-21T20:22Z	2022-04-14T21:15Z	{'baseMetricV3': {'cvssV3': {'version': '3.1',...	LOW	NETWORK	NONE	6.1	MEDIUM	LOW
...	...	...	...	...	...	...	...	...	...	...	...
2555	CVE-2016-5515	Unspecified vulnerability in the Oracle Agile ...	2016-11-28T20:26Z	2016-10-25T14:29Z	{'baseMetricV3': {'cvssV3': {'version': '3.0',...	LOW	NETWORK	HIGH	8.8	HIGH	HIGH

integrityImpact	privilegesRequired	scope	userInteraction	vectorString	version	industry
NONE	LOW	UNCHANGED	NONE	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	3.1	administration
HIGH	HIGH	UNCHANGED	NONE	CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H	3.1	administration
HIGH	NONE	UNCHANGED	NONE	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	3.1	administration
LOW	NONE	UNCHANGED	NONE	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N	3.1	administration
LOW	NONE	CHANGED	REQUIRED	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N	3.1	administration
...	...	...	...	...	...	...
HIGH	LOW	UNCHANGED	NONE	CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	3.0	manufacturing

Appendix 4.5: Result dataframe of concatenating all the dataframe of different industries

4. Categorize the data records into different vulnerabilities type via searching for keywords in their CVE description (Refer to Table 4.2 for the keywords used for different threat types). If a data record's description column meets the conditions (i.e. contains the keywords for a specific vulnerability), the data record will be assigned a True value under the specific column for that vulnerability, otherwise a False value will be assigned.

```
import numpy as np
cond_list = [df['description'].str.contains('denial of service' or 'flood attack' or 'denial-of-service', case=False),
             df['description'].str.contains('buffer overflow' or ('buffer' and 'overwrite') or
             'stack overflow' or 'buffer over-read' or 'heap overflow' or 'heap-buffer-overflow'
            or 'integer overflow' or 'heap-overflow',case=False),
             df['description'].str.contains('code execution' or 'command execution' or
             ('execute' and 'arbitrary' and ('code' or 'command')) or ('execute' and ('code' or 'command')), case=False),
             df['description'].str.contains('memory consumption issue' or ('memory' and 'corrupt'), case=False),
             df['description'].str.contains(('sql' and 'inject'), case=False),
             df['description'].str.contains('XSS' or ('cross' and 'site') or 'Cross-Site Scripting' or 'Cross-site Scripting'
                                            case=False),
             df['description'].str.contains('Directory Traversal' or 'Path traversal' or
             ('InHand Networks InRouter 900 Industrial 4G Router before v1.0.0.r11700' and 'arbitrary') or
             ('Jenkins' and 'permission' and 'arbitrary'), case=False),
             df['description'].str.contains('Http Response Splitting' or ('request' and 'splitting'), case=False),
             df['description'].str.contains('Bypass', case=False),
             df['description'].str.contains('information disclosure' or 'gain information' or 'information exposure',
                                            case=False),
              df['description'].str.contains(('gain' and 'privilege') or 'privileged' or 'user privilege' or
              'privilege escalation' or ('unprivileged' and ('user' or 'attacker' or 'actor')), case=False),
              df['description'].str.contains('CSRF' or 'Cross-site request forgery', case=False),
              df['description'].str.contains('LFI' or ('file' and 'inclusion'), case=False)
             ]
choice_list = ['Denial of Service', 'Buffer Overflow', 'Code Execution', 'Memory Corruption', 'SQL injection',
'XSS', 'Directory Traversal', 'Http Response Splitting', 'Bypass', 'Gain Information', 'Gain Privilege', 'CSRF',
               'File Inclusion']
```

```
df['Denial of Service'] = np.where(cond_list[0], True, False)
df['Buffer Overflow'] = np.where(cond_list[1], True, False)
df['Code Execution'] = np.where(cond_list[2], True, False)
df['Memory Corruption'] = np.where(cond_list[3], True, False)
df['SQL Injection'] = np.where(cond_list[4], True, False)
df['XSS'] = np.where(cond_list[5], True, False)
df['Directory Traversal'] = np.where(cond_list[6], True, False)
df['Http Response Splitting'] = np.where(cond_list[7], True, False)
df['Bypass'] = np.where(cond_list[8], True, False)
df['Gain Information'] = np.where(cond_list[9], True, False)
df['Gain Privilege'] = np.where(cond_list[10], True, False)
df['CSRF'] = np.where(cond_list[11], True, False)
df['File Inclusion'] = np.where(cond_list[12], True, False)
```

Appendix 4.6

Threat type	Keywords
Denial of Service	'denial of service', 'flood attack', 'denial-of-service'
Buffer overflow	'buffer overflow', 'buffer' and 'overflow', ''stack overflow', 'buffer over-read', 'heap overflow', 'heap-buffer-overflow', 'integer overflow', 'heap-overflow'
Code Execution	'code execution', 'command execution', 'execute' and 'arbitrary' and 'code', 'execute' and 'arbitrary' and 'command', 'execute' and 'code', 'execute' and 'command'
Memory Corruption	'memory consumption issue', 'memory' and 'corrupt'
SQL Injection	'sql' and 'inject'
XSS	'XSS', 'cross' and 'site', 'Cross-Site Scripting', 'Cross-site Scripting'
Directory Traversal	'Directory Traversal', 'Path traversal', 'InHand Networks InRouter 900 Industrial 4G Router before v1.0.0.r11700' and 'arbitrary' or 'Jenkins' and 'permission' and 'arbitrary'
Http Response Splitting	'Http Response Splitting', 'request' and 'splitting'
Bypass	'Bypass'
Gain Information	'information disclosure', 'gain information', 'information exposure'
Gain Privilege	'gain' and 'privilege', 'privileged', 'user privilege', 'privilege escalation', 'unprivileged' and 'user', 'unprivileged' and 'attacker', 'unprivileged' and 'actor'
CSRF	'CSRF', 'Cross-site request forgery'
File Inclusion	'LFI', 'file' and 'inclusion'

Table 4.2

5. Convert the baseSeverity into numerical form for the scoring function.

```
df['Severity Score'] = df['baseSeverity'].map({'LOW': 1,
                                               'MEDIUM': 2,
                                               'HIGH': 3,
                                               'CRITICAL': 4})
```

Appendix 4.7

6. Define a function to retrieve data for training the Convolutional Neural Network model. However, this function is different from the getdata(industry) function as the keyword being searched in this instance is the different threat categories.

```
def getdata(threat_cat):
    response = requests.get("https://services.nvd.nist.gov/rest/json/cves/1.0/?resultsPerPage=2000&keyword="+threat_cat)
    Dictionary = response.json()
    id_ = []
    description = []
    score = []
    published_ = []
    lastModified_ = []
    dict_ = Dictionary['result']['CVE_Items']
    for element in dict_:
        id_.append(element['cve']['CVE_data_meta']['ID'])
        for i in element['cve']['description']['description_data']:
            description.append(i['value'])
        score.append(element['impact'])
        published_.append(element['publishedDate'])
        lastModified_.append(element['lastModifiedDate'])

    lists = [id_, description, score, published_, lastModified_]
    df = pd.DataFrame(lists).transpose()
    df.columns = ['id', 'description','score','published_','lastModified']

    df = df[df.score != {}]
    df.reset_index(drop=True, inplace=True)
    df_score = df["score"].apply(pd.Series)
    df_metricV3 = df_score["baseMetricV3"].apply(pd.Series)
    df_cvssV3 = df_metricV3["cvssV3"].apply(pd.Series)
    result_df = pd.concat([df, df_cvssV3], axis=1)
    return result_df
```

Appendix 4.8

7. Get the data using the keyword.

```
df_overflow = getdata('overflow')
df_overflow['Category'] = 'Overflow'

df_DOS = getdata('denial of service')
df_DOS['Category'] = 'DOS'

df_code_exec = getdata('Code Execution')
df_code_exec['Category'] = 'Code Execution'

df_memory_corruption = getdata('memory corruption')
df_memory_corruption['Category'] = 'Memory Corruption'

df_sql = getdata('SQL Injection')
df_sql['Category'] = 'SQL Injection'

df_xss = getdata('XSS')
df_xss['Category'] = 'XSS'

df_directory_tra = getdata('Directory Traversal')
df_directory_tra['Category'] = 'Directory Traversal'

df_http = getdata('Http Response Splitting')
df_http['Category'] = 'Http Response Splitting'

df_bypass = getdata('bypass')
df_bypass['Category'] = 'Bypass'

df_gain_info = getdata('gain information')
df_gain_info['Category'] = 'Gain Info'

df_gain_privi = getdata('gain privilege')
df_gain_privi['Category'] = 'Gain Privilege'

df_CSRF = getdata('CSRF')
df_CSRF['Category'] = 'CSRF'

df_FI = getdata('File Inclusion')
df_FI['Category'] = 'File Inclusion'
```

Appendix 4.9

8. Concatenate the resulting dataframes into a single dataframe. Drop the data records with null values for score.

```
result_df = pd.concat([df_overflow, df_DOS, df_code_exec, df_memory_corruption, df_sql, df_xss, df_directory_tra,
                       df_http, df_bypass, df_gain_info, df_gain_privi, df_CSRF, df_FI], ignore_index=True, sort=False)
result_df.loc[result_df['attackComplexity'].isnull() == True]
result_df.drop(result_df[result_df['attackComplexity'].isnull() == True].index, inplace=True)
result_df = result_df.reset_index(drop=True)
result_df
```

Appendix 4.10

9. Define function for pre-processing the description data prior to the model training.

```
def decontract(sentence):
    # specific
    sentence = re.sub(r"won't", "will not", sentence)
    sentence = re.sub(r"can\'t", "can not", sentence)

    # general
    sentence = re.sub(r"n\'t", " not", sentence)
    sentence = re.sub(r"\'re", " are", sentence)
    sentence = re.sub(r"\'s", " is", sentence)
    sentence = re.sub(r"\'d", " would", sentence)
    sentence = re.sub(r"\'ll", " will", sentence)
    sentence = re.sub(r"\'t", " not", sentence)
    sentence = re.sub(r"\'ve", " have", sentence)
    sentence = re.sub(r"\'m", " am", sentence)
    return sentence

def cleanPunc(sentence):
    cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
    cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
    cleaned = cleaned.strip()
    cleaned = cleaned.replace("\n"," ")
    return cleaned

def keepAlpha(sentence):
    alpha_sent = ""
    for word in sentence.split():
        alpha_word = re.sub('[^a-z A-Z]+', '', word)
        alpha_sent += alpha_word
        alpha_sent += " "
    alpha_sent = alpha_sent.strip()
    return alpha_sent
```

```
from nltk.corpus import stopwords
stop = stopwords.words('english')

stemmer = SnowballStemmer("english")
def stemming(sentence):
    stemSentence = ""
    for word in sentence.split():
        stem = stemmer.stem(word)
        stemSentence += stem
        stemSentence += " "
    stemSentence = stemSentence.strip()
    return stemSentence
```

Appendix 4.11

10. Apply the functions to clean the description data.

```
result_df['description'] = result_df['description'].str.lower()
result_df['description'] = result_df['description'].apply(decontract)
result_df['description'] = result_df['description'].apply(cleanPunc)
result_df['description'] = result_df['description'].apply(keepAlpha)
result_df['description'] = result_df['description'].apply(lambda x: ' '.join(
    [word for word in x.split() if word not in (stop)]))
result_df['description'] = result_df['description'].apply(stemming)
```

Appendix 4.12

11. Train the model.

```
from keras.layers import Dense, Activation, Embedding, Flatten, GlobalMaxPool1D, Dropout, Conv1D
max_words = 20000
filter_length = 300
maxlen=200

model = Sequential()
model.add(Embedding(max_words, 20, input_length=maxlen))
model.add(Dense(512, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)))
#model.add(Dropout(0.5))
model.add(Conv1D(filter_length, 3, padding='valid', activation='relu', strides=1))
model.add(GlobalMaxPool1D())
model.add(Dense(4))
model.add(Activation('sigmoid'))
#model.add(Dropout(0.5))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=[tf.keras.metrics.AUC()])

callbacks = [
    tf.keras.callbacks.ReduceLROnPlateau(),
    tf.keras.callbacks.ModelCheckpoint(filepath='model-conv1d.h5', save_best_only=True)
]
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 200, 20)           400000
_________________________________________________________________
dense (Dense)                (None, 200, 512)          10752
_________________________________________________________________
conv1d (Conv1D)              (None, 198, 300)          461100
_________________________________________________________________
global_max_pooling1d (Global (None, 300)               0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1204
_________________________________________________________________
activation (Activation)      (None, 4)                 0
=================================================================
Total params: 873,056
Trainable params: 873,056
Non-trainable params: 0
_________________________________________________________________
```

```
epochs = 30
# Fit the model using the train and test datasets.
#history = model.fit(x_train, train_labels,validation_data= (x_test,test_labels),epochs=epochs )
history = model.fit(train_ds.shuffle(2000).batch(128),
                    epochs= epochs ,
                    validation_data=valid_ds.batch(128),
                    verbose=1)
```

Appendix 4.13

12. Save the model as 'tf_cnnmodel', the tokenizer used as a pickle object.

```
import os
import sys

save_path = sys.path[0]
file_name = 'tf_cnnmodel'
modelsave = os.path.join(sys.path[0], file_name)
print(modelsave)
model.save(modelsave)
```

```
C:\Users\heilu\tf_cnnmodel
INFO:tensorflow:Assets written to: C:\Users\heilu\tf_cnnmodel\assets
```

```
import pickle

# saving
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

Appendix 4.14

13. Load the saved model and the tokenizer.

```
new_model = tf.keras.models.load_model(modelsave)
new_model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 200, 20)           400000
_________________________________________________________________
dense_4 (Dense)              (None, 200, 512)          10752
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 198, 300)          461100
_________________________________________________________________
global_max_pooling1d_2 (Glob (None, 300)               0
_________________________________________________________________
dense_5 (Dense)              (None, 4)                 1204
_________________________________________________________________
activation_2 (Activation)    (None, 4)                 0
=================================================================
Total params: 873,056
Trainable params: 873,056
Non-trainable params: 0
_________________________________________________________________
```

```
with open('tokenizer.pickle', 'rb') as handle:
    tokenizer1 = pickle.load(handle)
```

Appendix 4.15

14. Prepare the MISP mapped CVE dataset: During the preparation of the dataset, you can have null values for the 'DATE' column and 'MISP info' column. However, for every other column, the presence of null value would hinder the computation of the MISP multiplied severity score. The value within the 'Industry' column must be consistent with the CVE foundation dataset, i.e. government, healthcare, education and etc.

CVE_ID	MISP_ID	DATE	Attribute_count	Threat_lvl	MISP info	Industry
CVE-2009-3129	42	20/1/2015	56	1	OSINT Ana	education
CVE-2013-5065	17	13/11/2014	53	1	OSINT Exp	education
CVE-2013-3346	17	13/11/2014	53	1	OSINT Exp	education
CVE-2012-1723	17	13/11/2014	53	1	OSINT Exp	education
CVE-2013-2729	17	13/11/2014	53	1	OSINT Exp	education
CVE-2012-4681	17	13/11/2014	53	1	OSINT Exp	education
CVE-2009-3129	17	13/11/2014	53	1	OSINT Exp	education
CVE-2013-5065	42	20/1/2015	56	1	OSINT Ana	education
CVE-2013-3346	42	20/1/2015	56	1	OSINT Ana	education
CVE-2012-1723	42	20/1/2015	56	1	OSINT Ana	education
CVE-2013-2729	42	20/1/2015	56	1	OSINT Ana	education
CVE-2012-4681	42	20/1/2015	56	1	OSINT Ana	education

Appendix 4.16

15. Insert MISP-mapped CVE data from mysql

```
#Database connection credentials
username = os.environ.get('MySQLroot')
password = os.environ.get('MySQLrootpw')
hostname = 'localhost'
dbname = 'securemation'
```

```
#Create SQLAlchemy engine to connect to mySQL db
engine = create_engine("mysql+pymysql://{user}:{pw}@{host}/{db}"
                       .format(host=hostname, db=dbname, user=username, pw=password))
dbConnection = engine.connect()
```

```
MISP_cve = pd.read_sql("select * from securemation.misp_map", dbConnection);
```

Appendix 4.17

16. Define a function to obtain the CVSS V3 Score for every MISP mapped CVE data record.

```
def getscore(id):
    response = requests.get("https://services.nvd.nist.gov/rest/json/cve/1.0/"+ id + "?addOns=dictionaryCpes")
    Dictionary = response.json()
    id_ = []
    description = []
    score = []
    dict_ = Dictionary['result']['CVE_Items']
    for element in dict_:
        id_.append(element['cve']['CVE_data_meta']['ID'])
        for i in element['cve']['description']['description_data']:
            description.append(i['value'])
        if "baseMetricV3" in element['impact']:
            score.append(element['impact'])
        else:
            score.append({'baseMetricV3': {'cvssV3': {'version': '3.1','vectorString': 'NULL',
                                                      'attackVector': 'NULL','attackComplexity': 'NULL',
                                                      'privilegesRequired': 'NULL','userInteraction': 'NULL',
                                                      'scope': 'NULL','confidentialityImpact': 'NULL',
                                                      'integrityImpact': 'NULL','availabilityImpact': 'NULL',
                                                      'baseScore': 0,'baseSeverity': 'NULL'},
                                           'exploitabilityScore': 0,'impactScore': 0}})

    lists = [id_, description, score]
    df = pd.DataFrame(lists).transpose()
    df.columns = ['CVE_ID', 'description','score']

    df = df[df.score != {}]
    df.reset_index(drop=True, inplace=True)
    df_score = df["score"].apply(pd.Series)
    df_metricV3 = df_score["baseMetricV3"].apply(pd.Series)
    df_cvssV3 = df_metricV3["cvssV3"].apply(pd.Series)
    result_df = pd.concat([df, df_cvssV3], axis=1)
    return result_df
```

Appendix 4.18

17. Create a new column called 'MISP Severity Score' in the dataframe matching the Key value pair from the 'Threat_lvl' column. Compute the 'misp-multiplier' for every data record.

Misp multiplier = 0.01 x Attribute count x MISP Severity Score

Threat_lvl	MISP Severity Score
1	3
2	2

3	1
4	0

```
df_misp_mapped = pd.concat([MISP_cve, MISP], axis=1)
df_misp_mapped.loc[df_misp_mapped['baseSeverity'] == 'NULL']
df_misp_mapped['MISP Severity Score'] = df_misp_mapped['Threat_lvl'].map({1: 3,
                                        2: 2,
                                        3: 1,
                                        4: 0})
df_misp_mapped['misp-multiplier'] = df_misp_mapped['MISP Severity Score']* (0.01 * df_misp_mapped['Attribute_count'])
df_misp_mapped = df_misp_mapped[['CVE_ID', 'Attribute_count', 'Industry', 'description','baseSeverity','MISP Severity Score',
df_misp_mapped = df_misp_mapped.reset_index(drop = True)
df_misp_mapped = df_misp_mapped.T.drop_duplicates().T
df_misp_mapped
```

	CVE_ID	Attribute_count	Industry	description	baseSeverity	MISP Severity Score	misp-multiplier
0	CVE-2009-3129	56	education	Microsoft Office Excel 2002 SP3, 2003 SP3, and...	NULL	3	1.68
1	CVE-2013-5065	53	education	NDProxy.sys in the kernel in Microsoft Windows...	NULL	3	1.59
2	CVE-2013-3346	53	education	Adobe Reader and Acrobat 9.x before 9.5.5, 10....	NULL	3	1.59
3	CVE-2012-1723	53	education	Unspecified vulnerability in the Java Runtime ...	NULL	3	1.59
4	CVE-2013-2729	53	education	Integer overflow in Adobe Reader and Acrobat 9...	NULL	3	1.59
...	...	...	...	...	...	...	...
188	CVE-2015-4902	114	utilities	Unspecified vulnerability in Oracle Java SE 6u...	NULL	2	2.28
189	CVE-2014-4076	114	utilities	Microsoft Windows Server 2003 SP2 allows local...	NULL	2	2.28
190	CVE-2015-2545	142	utilities	Microsoft Office 2007 SP3, 2010 SP2, 2013 SP1,...	NULL	1	1.42
191	CVE-2015-2545	120	utilities	Microsoft Office 2007 SP3, 2010 SP2, 2013 SP1,...	NULL	2	2.4
192	CVE-2019-13720	14	utilities	Use after free in WebAudio in Google Chrome pr...	HIGH	1	0.14

Appendix 4.19

18. Assign the data records into different vulnerabilities type via searching for keywords in their CVE description.

```
import numpy as np
cond_list = [df_misp_mapped['description'].str.contains('denial of service' or 'flood attack' or 'denial-of-service', case=Fa
             df_misp_mapped['description'].str.contains('buffer overflow' or ('buffer' and 'overwrite') or
             'stack overflow' or 'buffer over-read' or 'heap overflow' or 'heap-buffer-overflow'
            or 'integer overflow' or 'heap-overflow',case=False),
             df_misp_mapped['description'].str.contains('code execution' or 'command execution' or
             ('execute' and 'arbitrary' and ('code' or 'command')) or ('execute' and ('code' or 'command')), case=False),
             df_misp_mapped['description'].str.contains('memory consumption issue' or ('memory' and 'corrupt'), case=False),
             df_misp_mapped['description'].str.contains(('sql' and 'inject'), case=False),
             df_misp_mapped['description'].str.contains('XSS' or ('cross' and 'site') or 'Cross-Site Scripting' or 'Cross-sit
             df_misp_mapped['description'].str.contains('Directory Traversal' or 'Path traversal' or
             ('InHand Networks InRouter 900 Industrial 4G Router before v1.0.0.r11700' and 'arbitrary') or
             ('Jenkins' and 'permission' and 'arbitrary'), case=False),
             df_misp_mapped['description'].str.contains('Http Response Splitting' or ('request' and 'splitting'), case=False)
             df_misp_mapped['description'].str.contains('Bypass', case=False),
             df_misp_mapped['description'].str.contains('gain information' or ('information' and 'leak') or
             'information exposure' or 'information disclosure', case=False),
              df_misp_mapped['description'].str.contains(('gain' and 'privilege') or 'privileged' or 'user privilege' or
              'privilege escalation' or ('unprivileged' and ('user' or 'attacker' or 'actor')), case=False),
              df_misp_mapped['description'].str.contains('CSRF' or 'Cross-site request forgery', case=False),
              df_misp_mapped['description'].str.contains('LFI' or ('file' and 'inclusion'), case=False)
             ]
choice_list = ['Denial of Service', 'Buffer Overflow', 'Code Execution', 'Memory Corruption', 'SQL injection',
'XSS', 'Directory Traversal', 'Http Response Splitting', 'Bypass', 'Gain Information', 'Gain Privilege', 'CSRF', 'File Inclus


df_misp_mapped['Denial of Service'] = np.where(cond_list[0], True, False)
df_misp_mapped['Buffer Overflow'] = np.where(cond_list[1], True, False)
df_misp_mapped['Code Execution'] = np.where(cond_list[2], True, False)
df_misp_mapped['Memory Corruption'] = np.where(cond_list[3], True, False)
df_misp_mapped['SQL Injection'] = np.where(cond_list[4], True, False)
df_misp_mapped['XSS'] = np.where(cond_list[5], True, False)
df_misp_mapped['Directory Traversal'] = np.where(cond_list[6], True, False)
df_misp_mapped['Http Response Splitting'] = np.where(cond_list[7], True, False)
df_misp_mapped['Bypass'] = np.where(cond_list[8], True, False)
df_misp_mapped['Gain Information'] = np.where(cond_list[9], True, False)
df_misp_mapped['Gain Privilege'] = np.where(cond_list[10], True, False)
df_misp_mapped['CSRF'] = np.where(cond_list[11], True, False)
df_misp_mapped['File Inclusion'] = np.where(cond_list[12], True, False)
```

Appendix 4.20

19. Pre-process the description column within the MISP mapped CVE dataset before it is used as an input to the machine learning model to obtain a baseSeverity value for those records with baseSeverity = NULL.

```
df_misp_mapped['description'] = df_misp_mapped['description'].str.lower()
df_misp_mapped['description'] = df_misp_mapped['description'].apply(decontract)
df_misp_mapped['description'] = df_misp_mapped['description'].apply(cleanPunc)
df_misp_mapped['description'] = df_misp_mapped['description'].apply(keepAlpha)
df_misp_mapped['description'] = df_misp_mapped['description'].apply(lambda x: ' '.join(
    [word for word in x.split() if word not in (stop)]))
df_misp_mapped['description'] = df_misp_mapped['description'].apply(stemming)
```

Appendix 4.21

20. Separate the dataframe into two dataframes (one in which all records have baseSeverity = NULL and another one in which all records have an actual CVE baseSeverity value). After that, convert the description values into a vector using tokenizer. Use pad_sequences() function to pad variable length sequences.

```
df_notnull = df_misp_mapped.loc[df_misp_mapped['baseSeverity'] != 'NULL']
df_null = df_misp_mapped.loc[df_misp_mapped['baseSeverity'] == 'NULL']
df_null = df_null.reset_index(drop = True)
df_null
```

Appendix 4.22

```
x  = np.array(tokenizer1.texts_to_sequences(df_null['description'].tolist()) )
x = pad_sequences(x, padding='post', maxlen=200)
```

Appendix 4.23

21. Generate predictions of the baseSeverity for the records with baseSeverity = NULL.

```
# Generate predictions (probabilities -- the output of the last layer)
# on test  data using `predict`
print("Generate predictions for all null data")
predictions = new_model.predict(x)
print(predictions)
predict_results = predictions.argmax(axis=1)
```

```
Generate predictions for all null data
[[2.37971544e-04 9.54990804e-01 1.04997754e-02 1.27494335e-04]
 [1.58387423e-03 5.53691626e-01 3.18065649e-05 2.70176530e-02]
 [1.58387423e-03 5.53691626e-01 3.18065649e-05 2.70176530e-02]
 [9.99960661e-01 2.01811908e-05 1.45459175e-03 9.87979770e-03]
 [9.99960661e-01 2.01811908e-05 1.45459175e-03 9.87979770e-03]
 [1.60164267e-01 4.29242849e-04 4.36574221e-04 9.89532292e-01]
 [1.60164267e-01 4.29242849e-04 4.36574221e-04 9.89532292e-01]
 [9.82865095e-01 6.28608465e-03 4.06056643e-04 8.94188881e-04]
 [9.82865095e-01 6.28608465e-03 4.06056643e-04 8.94188881e-04]
 [9.64184165e-01 7.62718916e-03 3.57359648e-04 1.24776363e-03]
 [9.64184165e-01 7.62718916e-03 3.57359648e-04 1.24776363e-03]
 [8.72926950e-01 1.00270182e-01 6.08652830e-04 3.34799290e-04]
 [8.72926950e-01 1.00270182e-01 6.08652830e-04 3.34799290e-04]
 [9.99960661e-01 2.01811908e-05 1.45459175e-03 9.87979770e-03]
 [9.99960661e-01 2.01811908e-05 1.45459175e-03 9.87979770e-03]
 [1.58387423e-03 5.53691626e-01 3.18065649e-05 2.70176530e-02]
 [1.58387423e-03 5.53691626e-01 3.18065649e-05 2.70176530e-02]
 [2.62574895e-06 9.87930775e-01 2.30557998e-05 3.57057154e-02]
```

```
df_null['baseSeverity']= predict_results
df_null['baseSeverity'] = np.where((df_null.baseSeverity == 0),'CRITICAL',df_null.baseSeverity)
df_null['baseSeverity'] = np.where((df_null.baseSeverity == '1'),'HIGH',df_null.baseSeverity)
df_null['baseSeverity'] = np.where((df_null.baseSeverity == '2'),'LOW',df_null.baseSeverity)
df_null['baseSeverity'] = np.where((df_null.baseSeverity == '3'),'MEDIUM',df_null.baseSeverity)
```

Appendix 4.21

	CVE_ID	Attribute_count	Industry	description	baseSeverity
0	CVE-2009-3129	56	education	microsoft offic excel sp sp sp sp offic mac op...	HIGH
1	CVE-2013-5065	53	education	ndproxi sys kernel microsoft window xp sp sp s...	HIGH
2	CVE-2013-3346	53	education	adob reader acrobat x x x allow attack	HIGH

Appendix 4.22: All the data records in df_NULL have an assigned baseSeverity level.

22. Merge the resulting dataframe (with assigned baseSeverity level) and the dataframe in which all records have an actual CVE baseSeverity value.

```
df_misp_mapped = pd.concat([df_null, df_notnull], ignore_index=True)
df_misp_mapped
```

Appendix 4.23

23. Create a new column called 'Severity Score' in the dataframe matching the Key value pair from the 'baseSeverity' column.

baseSeverity	Severity Score
LOW	1
MEDIUM	2
HIGH	3
CRITICAL	4

```
df_misp_mapped['Severity Score'] = df_misp_mapped['baseSeverity'].map({'LOW': 1,
                                                                        'MEDIUM': 2,
                                                                        'HIGH': 3,
                                                                        'CRITICAL': 4})
df_misp_mapped
```

Appendix 4.24

24. Set misp-multiplier of the foundation CVE dataset as 0. Merge the foundation CVE dataset and the MISP-mapped CVE dataset together while ensuring that all columns to be merged for both datasets are of the same column name.

```
df['misp-multiplier'] = 0
df_scoring = pd.concat([df[['CVE_ID', 'Industry', 'baseSeverity', 'Denial of Service',
                            'Buffer Overflow', 'Code Execution', 'Memory Corruption', 'SQL Injection',
                            'XSS', 'Directory Traversal', 'Http Response Splitting', 'Bypass', 'Gain Information',
                            'Gain Privilege', 'CSRF', 'File Inclusion', 'Severity Score', 'misp-multiplier']],
                        df_misp_mapped[['CVE_ID', 'Industry', 'baseSeverity', 'Denial of Service', 'Buffer Overflow',
                                        'Code Execution', 'Memory Corruption', 'SQL Injection', 'XSS',
                                        'Directory Traversal', 'Http Response Splitting', 'Bypass',
                                        'Gain Information', 'Gain Privilege', 'CSRF', 'File Inclusion',
                                        'Severity Score', 'misp-multiplier']]], ignore_index=True)
df_scoring
```

Appendix 4.25

25. Define a scoring function that is able to compute the severity score for different threat type and in different industries. Refer to Appendix 1 for better understanding of how the computation of MISP multiplied Severity Score is done.

```
def scoring(df, ind):
    df_industry = df.loc[df['Industry']== ind]
    severity_score = []
    MISP_severity_score = []
    choice_list = ['Denial of Service', 'Buffer Overflow', 'Code Execution', 'Memory Corruption', 'SQL Injection',
'XSS', 'Directory Traversal', 'Http Response Splitting', 'Bypass', 'Gain Information', 'Gain Privilege', 'CSRF',
                   'File Inclusion']
    for threat in choice_list:
        df_threat = df_industry.loc[df_industry[threat] == True][[threat, 'misp-multiplier','Severity Score']]
        values = df_threat['misp-multiplier'] * df_threat['Severity Score']
        df_threat['MISP Severity Score'] = values.where(df_threat['misp-multiplier'] != 0, other=df_threat['Severity Score'])
        MISP_severity_score.append(df_threat['MISP Severity Score'].sum())
        severity_score.append(df_threat['Severity Score'].sum())

    scoring = pd.DataFrame()
    scoring['threatType'] = choice_list
    scoring['severityScore'] = severity_score
    scoring['MISPSeverityScore'] = MISP_severity_score
    scoring = scoring.sort_values(by=['MISPSeverityScore'], ascending=False)
    scoring.reset_index(drop=True, inplace=True)
    return scoring
```

Appendix 4.26

26. Test whether the scoring function works: Loop through a list of Indusry and get their MISP-multiplied Severity Score.

```
industrylist = ['administration', 'hospitality', 'agriculture', 'finance', 'government',
                'healthcare', 'realestate', 'retail', 'tourism', 'transportation',
                'telecommunication', 'manufacturing', 'utilities']
```

Appendix 4.27

```
for i in industrylist:
    print(i)
    print(scoring(df_scoring, i))
```

```
administration
                threatType  severityScore  MISPSeverityScore
0           Gain Privilege            179                179
1         Denial of Service             75                 75
2             SQL Injection             38                 38
3                       XSS             25                 25
4                    Bypass             22                 22
5           Code Execution             19                 19
6                      CSRF             15                 15
7          Buffer Overflow              7                  7
8          Gain Information              7                  7
9       Directory Traversal              3                  3
10        Memory Corruption              0                  0
11  Http Response Splitting              0                  0
12           File Inclusion              0                  0
hospitality
                threatType  severityScore  MISPSeverityScore
0           Gain Privilege            369             362.10
1             SQL Injection             83              83.00
2         Denial of Service             63              60.39
3                       XSS             36              36.00
4           Code Execution             18              10.17
5                    Bypass              8              8.00
6                      CSRF              8              8.00
7       Directory Traversal              6              6.00
8   Http Response Splitting              2              2.00
9          Buffer Overflow              3              0.39
10        Memory Corruption              0              0.00
11         Gain Information              0              0.00
12           File Inclusion              0              0.00
```

Appendix 4.28

27. Export the table to mysql database.

```
#Database connection credentials
username = os.environ.get('MySQLroot')
password = os.environ.get('MySQLrootpw')
hostname = 'localhost'
dbname = 'securemation'

#Create SQLAlchemy engine to connect to mySQL db
engine = create_engine("mysql+pymysql://{user}:{pw}@{host}/{db}"
                       .format(host=hostname, db=dbname, user=username, pw=password))
```

```
for i in industrylist:
    print("Created MySQL table for: " + i)
    scoring(df_scoring, i).to_sql(i, engine, index=False, if_exists='replace')
```

Appendix 4.29

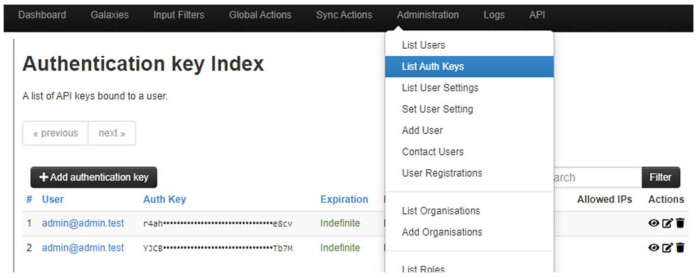
APPENDIX 5 – TENSORFLOW (MISP) PREDICTION CONFIGURATION

Environment set up and MISP attribute file download:

1. Download and install Ubuntu terminal environment with Windows Subsystem for Linux (WSL) from Microsoft Store.
2. Install Docker Engine on Ubuntu and run. https://docs.docker.com/engine/install/ubuntu/
3. Install coolacid/docker-misp (https://github.com/coolacid/docker-misp) on Ubuntu and run.

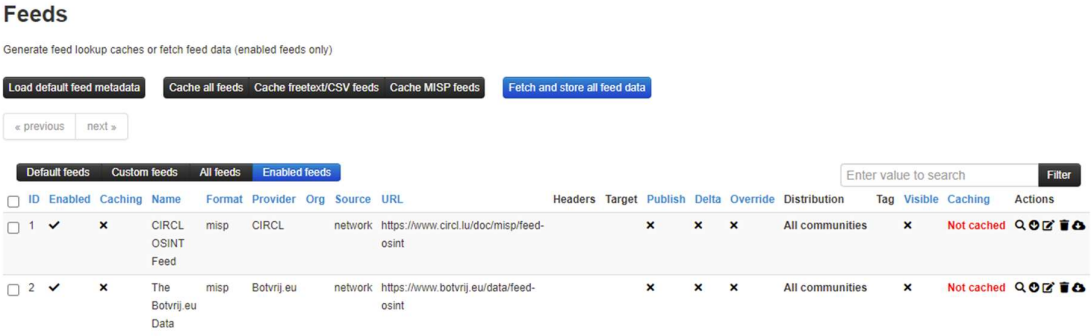
Appendix 5.1

4. Access to a localhost server (https://localhost) from host OS (Windows) and login to MISP. https://www.circl.lu/doc/misp/quick-start/
5. Add a new authentication key for API and record the key value.



Appendix 5.2

6. Activate default feeds (CIRCL OSINT Feed , The Botvrij.eu Data) and fetch feed data. https://www.circl.lu/doc/misp/managing-feeds/



Appendix 5.3

7. Download attribute data by using a GET request using Curl with OpenAPI.

```
C:¥Users¥       >curl  --header "Authorization:                                    " --header "Accept: applicati
on/json" --header "Content-Type: application/json" https://localhost/attributes --insecure -o MISP_atributes_all.json

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 39731  100 39731    0     0  22050      0  0:00:01  0:00:01 --:--:-- 22048
```

```
curl  --header "Authorization: YOUR_AUTHENTICATION_KEY"
      --header "Accept: application/json"
      --header "Content-Type: application/json"
      https://localhost/attributes --insecure -o MISP_atributes_all.json
```

Appendix 5.4

Data processing with Python:

8. Run 'MISP_prediction.py' (see Appendix 8 – MySQL configuration) in the same directly containing saved 'MISP_attributes_all.json' file in the step 7.
9. The predicted increase/decrease percentage* and prediction accuracy percentage** is automatically exported to the database.
10. Repeat from step 6 and update 'MISP_attributes_all.json' file manually when necessary.

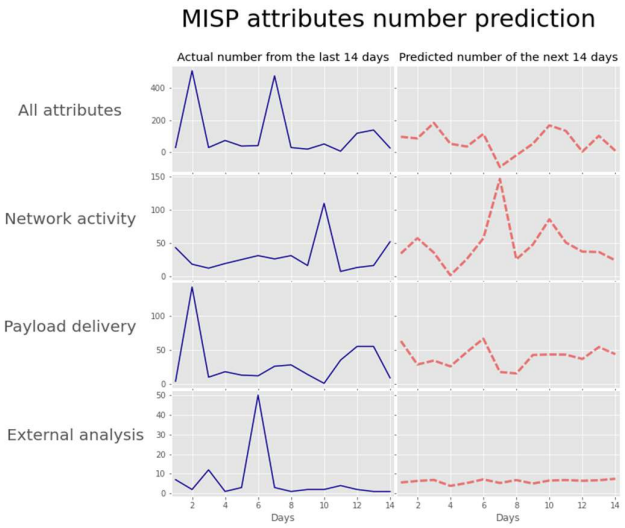
*** Prediction data in the final output are summed the total values for the last 14 days and upcoming 14 days in order to accommodate limitations of both MISP data (attributes are not recorded every day and there are blank days) and jBPM functionality (255 character limit).**

**** The prediction accuracy of future attribute numbers is calculated as below.**

100 – (test data – prediction data / test data) *100 (%)

Data visualisation:

11. To see the detail test results and prediction increase/decrease graphs, open 'MISP_prediction_graph.ipynb" file from Jupyter Notebook (https://jupyter.org/) and run the code with updated 'MISP_attributes_all.json' file.



	numbers_last_14days	predicted_numbers_14days	predicted_change	prediction_accuracy
All attributes	1579	927	-41.3%	58.8%
Network activity	419	732	74.7%	25.2%
Payload delivery	422	607	43.8%	56.0%
External analysis	91	92	1.1%	98.2%

Appendix 5.5

Change prediction settings:

12. To change the prediction period, change the value of variable 'daynum' in the python file (default value = 14).

```
############ Number of days for prediction ############
daynum = 14
```

Appendix 5.6

13. To filter old data input, change the value of pd.to_timedelta (default value = 1500days).

Data prep - All attribution

```
#extract id,event_id,timestamp
df_all = df_dict[['id','event_id','timestamp']]
#get timestamp and convert unix time to date and drop hours/minutes
df_all = pd.to_datetime(df_all['timestamp'], unit='s').dt.date
#convert object to datetime
df_all = pd.to_datetime(df_all)
#group by date and name headers
df_all = df_all.value_counts().rename_axis('Date').reset_index(name='Num').sort_values(by=['Da
#filter last X days
df_all = df_all[df_all['Date'] > datetime.now() - pd.to_timedelta("1500days")]
#cut outliers
q = df_all.Num.quantile(0.95)
df_all = df_all.query('Num < @q')
```

Appendix 5.7

14. Three attribute categories (External analysis, network activity and payload delivery) were selected for the prediction model to ensure there was sufficient number of days of data input to train the model (e.g., External analysis has 840 days of data input). To predict a different attribute category, change attribute name in str.contains.

Data prep - grouped by Category

```
#for misp_atr_category
df_cat = df_dict[['id','event_id','timestamp','category']]
#get timestamp and convert unix time to date and drop hours/minutes
df_cat['timestamp'] = pd.to_datetime(df_cat['timestamp'], unit='s').dt.date
#convert object to datetime
df_cat['timestamp'] = pd.to_datetime(df_cat['timestamp'])
df_cat = df_cat.value_counts(['timestamp', 'category']).reset_index(name='Num').sort_values
print(df_cat['category'].value_counts())
```

```
External analysis        840
Network activity         839
Payload delivery         785
Other                    356
Artifacts dropped        333
Payload installation     107
Attribution               68
Antivirus detection       61
Persistence mechanism     54
Financial fraud           20
```

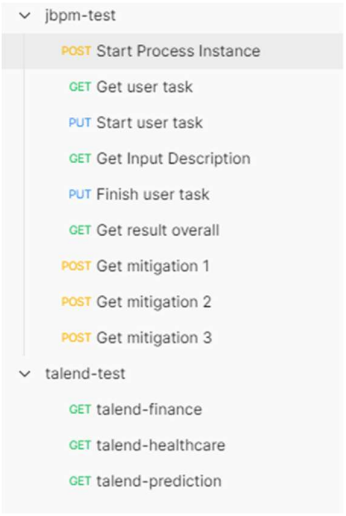
Model - External analysis

```
df_cat_EA = df_cat[df_cat['category'].str.contains("External analysis")]
#cut outliers
q = df_cat_EA.Num.quantile(0.95)
df_cat_EA = df_cat_EA.query('Num < @q')
```

Appendix 5.8

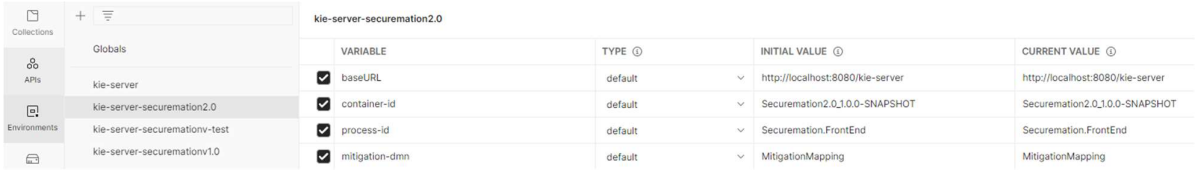
APPENDIX 6 – POSTMAN CONFIGURATION

Appendix 6.1 displays all the API tests. They are separated into two collections – jbpm-test and talend-test. To successfully execute the jbpm-test collection, ensure that the jBPM project, the Talend project, and the MySQL database are deployed and running. To execute the talend-test, only Talend and MySQL database need be running.



Appendix 6.1 Postman API collections

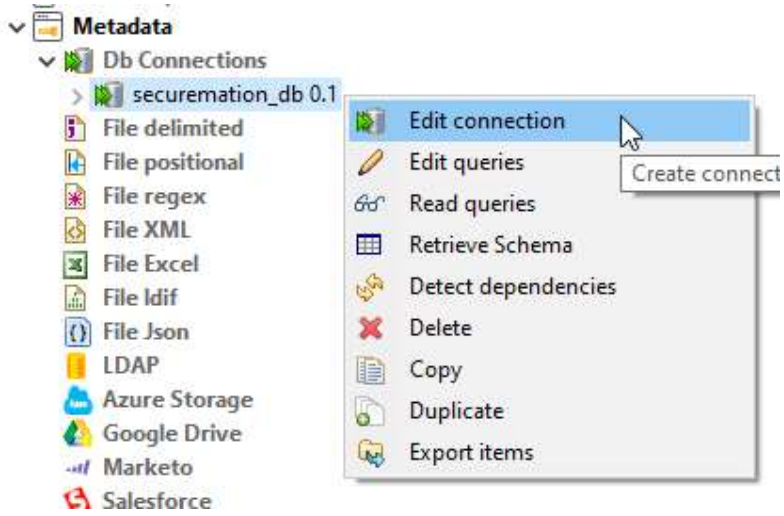
Appendix 6.2 displays all the environmental variables within Postman that are used by each of the Postman collections. Please ensure that each of these variables matches their corresponding identifiers within jBPM for the tests to execute successfully.



Appendix 6.2 KIE server environment

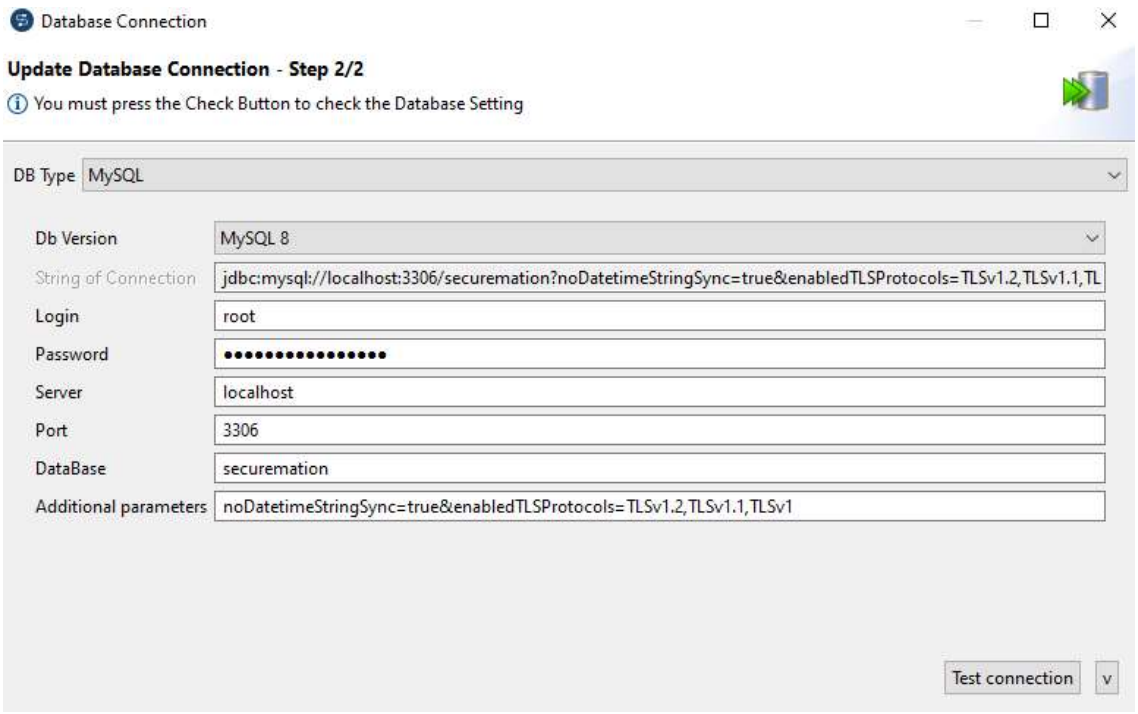
APPENDIX 7 – TALEND CONFIGURATION

1. To begin configuring Talend the first step is to verify the connection to the database after importing the Talend project located at Artefacts/Talend/query/..



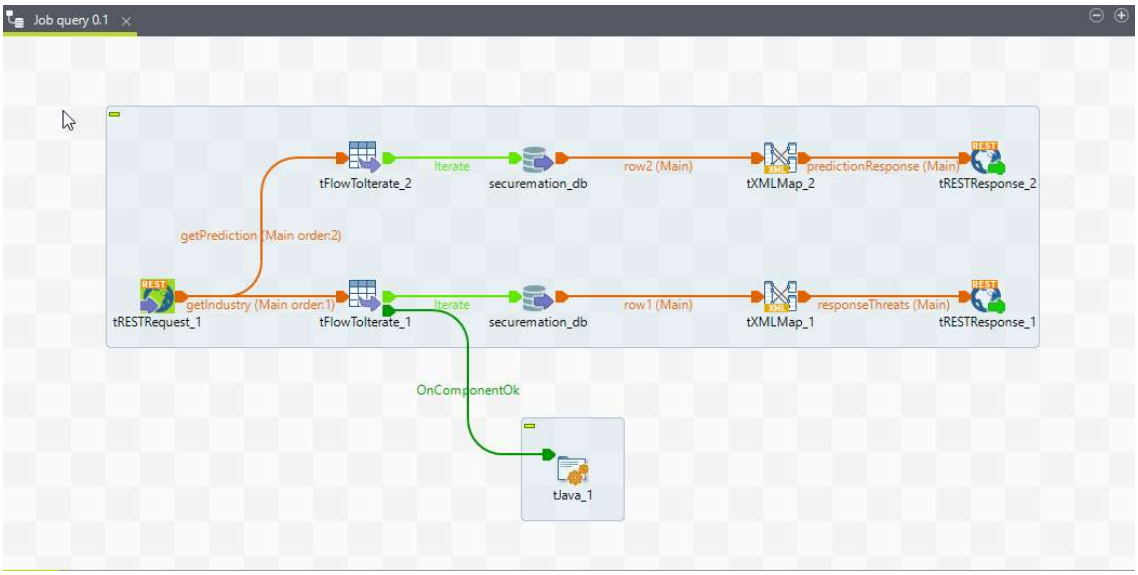
Appendix 7.1 Open "Edit connection" wizard to verify correct connection to MySQL database

2. Test the connection and verify that the login and password is correct for the MySQL schema selected.

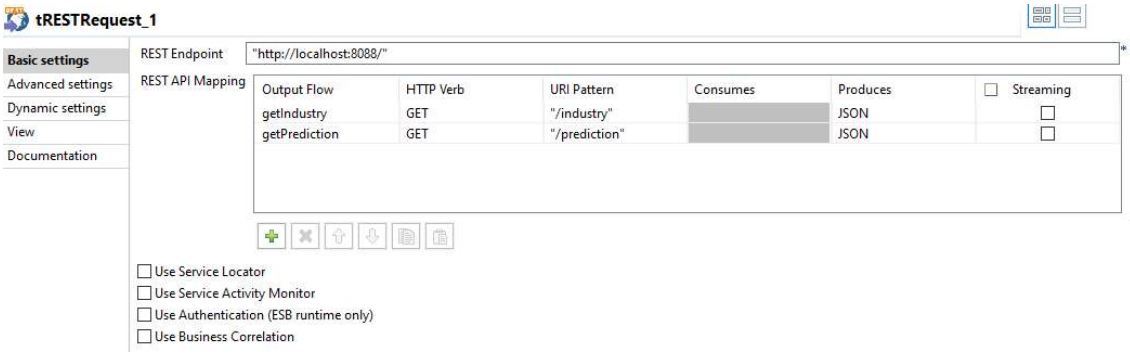


Appendix 7.2 MySQL connection Update database connection wizard in Talend.

3. Two tRESTResponse endpoints have been created from one request that allows the extracted data to be returned to jBPM. The URL description and configuration for the REST API can be seen in Appendix 7.3 by selecting the tRESTRequest_1 component.

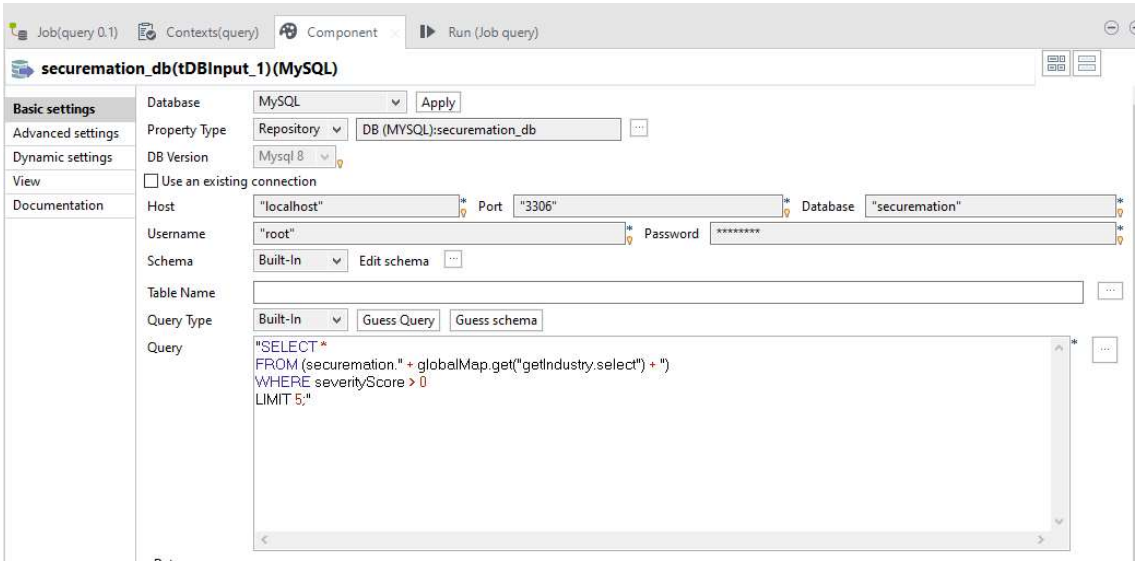


Appendix 7.2 Talend REST API job

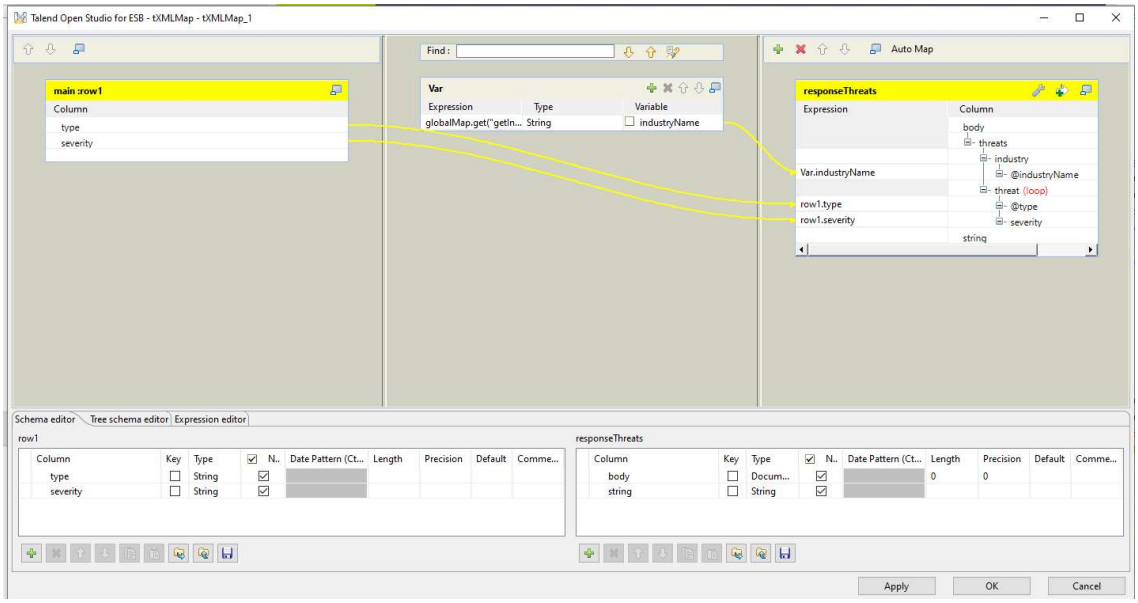


Appendix 7.3 tRESTRequest_1 Component details.

4. Changing the structure of the query and output can be done in the DBInput and tXML map components below in Appendix 7.4 and 7.5. The tXMLmap can be configured to adjust the final output.



Appendix 7.4 SQL query in the tDBInput component



Appendix 7.5 tXMLMap

APPENDIX 8 – MYSQL CONFIGURATION

1. Import the MySQL schema from MySQL/Securemation_database.sql into local MySQL server.

Appendix 8.1 MySQL schema containing industry tables, prediction result and optionally mapped data.

threatType	severityScore	MISPSeverityScore
Gain Privilege	423	421.32
SQL Injection	393	397.4
XSS	121	121
Code Execution	93	93
Denial of Service	81	77.64
Bypass	62	62
Buffer Overflow	31	31
Directory Traversal	17	17
CSRF	16	16
Gain Information	5	5
Memory Corruption	0	0
Http Response Spl...	0	0
File Inclusion	0	0

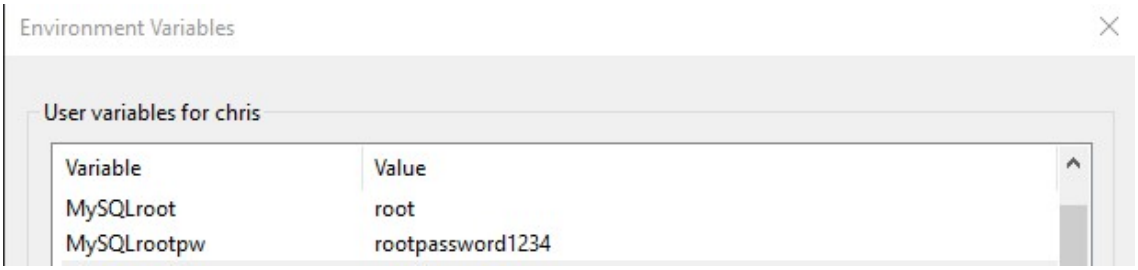
Appendix 8.2 Prioritisation model output for "Healthcare" industry table.

2. After running the prioritisation models 12 tables for industries will be automatically generated/updated in the database schema according to tables listed in Appendix 8.1.

attribute	numbers_last_14days	predicted_numbers_14days	predicted_change	prediction_accuracy
All attributes	1579	932	-41.0%	59.1%
Network activity	419	779	85.9%	14.1%
Payload delivery	422	307	-27.3%	72.8%
External analysis	91	96	5.5%	94.1%

Appendix 8.3 Prediction model output

3. After running the prediction model the output will be in a singular table as above.



Appendix 8.4 Windows OS Environment variables required for Python ML connection.

4. Create Environment Variables in Windows OS. Variable names must match image 8.4, Value must be equal to the username and password of the mySQL database being used.