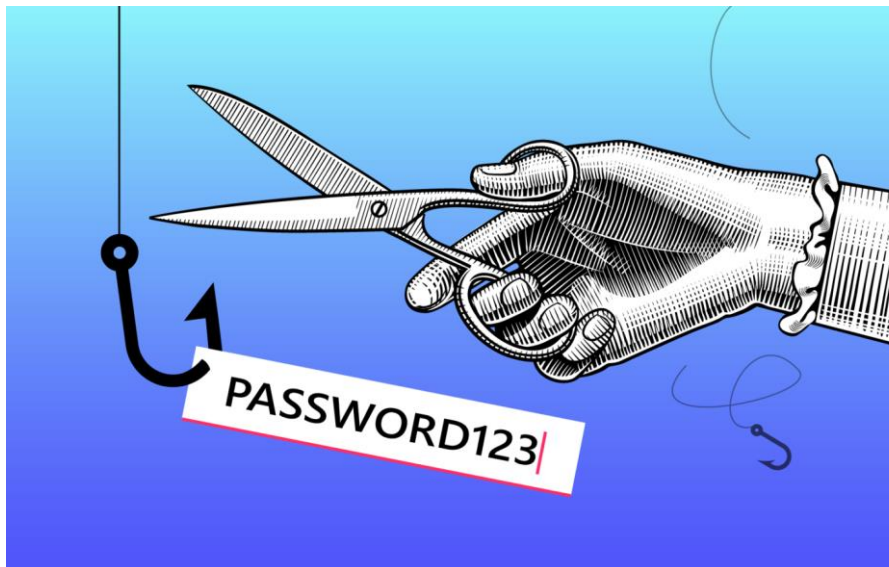


Project Security

Project Security IP Paper 22-23

“Better safe than sorry”



Nils van Witzenburg | 500849196 | IC202

12-11-2022

Versie 1.0



Inhoudsopgave

Inleiding.....	2
Problematiek	2
Aanpak en oplossing	3
Ontwerp en implementatie.....	4
Conclusie en advies.....	7
Bronnen.....	8
Bijlage 1 – Poster	9
Bijlage 2 – Poster v2.....	10
Bijlage 3 – Code van script	11
Bijlage 4 – Demo	15
Logboek.....	16
Docent feedback	16
Ontvangen en gegeven feedback	16
Planning.....	17
Deadlines	17
Sprint 1	18
Sprint 2	19
Sprint 3	20
Retrospective Sprint 1	21
Retrospective Sprint 2	21
Retrospective sprint 3	21
Sprints backlog.....	22
Methoden.....	23
Individuele reflectie	24

Inleiding

Bedrijf Y heeft net zijn deuren geopend en haar IT systeem ingericht. Het bedrijf heeft een mailserver die gebruikt wordt voor de klantenservice. Op de website kan een form worden ingevuld dat wordt doorgestuurd naar de mailserver. Gelukkig gaat het bedrijf onderzoek doen naar potentiële phishing-aanvallen. Ik, als ICT'er bij het bedrijf, heb de opdracht gekregen om de mailserver te beveiligen tegen deze aanvallen.

Mijn doel is om de hoeveelheid phishing e-mails te verminderen, die de medewerkers binnen krijgen via de form op de website. Vooral andere bedrijven en klanten vullen deze form in. Bij het invullen van de form moet degene hun eigen e-mail meegeven, waar zij een antwoord ontvangen in maximaal 5 werkdagen. Het doel van deze paper is om bedrijven op ideeën te brengen hoe je phishing-aanvallen tegen kan gaan, kleine of grote ICT afdeling.

Problematiek

Het bedrijf heeft mij dus gevraagd om onderzoek te doen naar het tegengaan van phishing-aanvallen.

Volgens Zscaler (2020) is door Covid-19 phishing, malicious websites en malware aanvallen, gerelateerd aan Covid, met 30.000% gestegen. Phishing-aanvallen zijn gestegen met wel liefst 667% en volgens APWG (2021) bereikte het aantal phishing-aanvallen een recordhoogte in 2021. Bedrijven en goede doelen zijn de meest voorkomende slachtoffers volgens de regering van het Verenigd Koninkrijk, GOV.uk (2022). Het blijkt dat Phishing-aanvallen de populairste vorm van cybercrime zijn. Daarom is het als beginnend bedrijf een van de belangrijkste uitdagingen om zoveel mogelijk van deze aanvallen te voorkomen en tegen te gaan.

Mijn probleemstelling is dus: Wat is voor bedrijf Y de beste maatregel om phishing-aanvallen tegen te gaan op korte termijn?

Aanpak en oplossing

Om te beginnen heb ik onderzoek gedaan naar welke mogelijke oplossingen er zijn tegen phishing-aanvallen.

Ik heb gekeken naar A.I. Based Protection. Dit is een goede oplossing volgens Santeri Kangas (2022), maar dit wordt al snel erg te duur en lastig om te onderhouden. Ik zou een eigen A.I kunnen programmeren, maar dit zal teveel tijd gaan kosten en dus op korte termijn minder veiligheid bieden.

Er worden veel cursussen aangeboden in het herkennen van (gerichte) phishing. Phishing e-mails worden met de dag 'echter' en het vergt energie en kosten om up-to-date te blijven. Verder zijn human errors nooit uit te sluiten. Maar dit is wel een goede maatregel om toe te voegen als het bedrijf meer personeel in dienst heeft.

Tenslotte ben ik zelf gaan denken over mogelijke alternatieve oplossingen. In het verleden heb ik geprogrammeerd in Python. Met mijn Python kennis zou ik een script kunnen maken die phishing e-mails kan detecteren op basis van een aantal veelvoorkomende eigenschappen. Ik heb daarbij gekozen om een Python script te maken, omdat het redelijk simpel te implementeren is, het efficiënt is en daarmee relatief goedkoop op korte termijn zekerheid voor het bedrijf wordt geboden.

Het script zal phishings emails moeten kunnen detecteren op basis van email adres en link(s). Er zal een tekst bestand worden aangemaakt, waar email adressen worden ingezet die een red flag geven. Deze red flag geeft aan dat dit een phishing email kan zijn. Verder wordt er gecheckt op interpunctie bij de domein naam van het email adres. Omdat de meeste e-mails binnen komen van klanten, wordt er gebruik gemaakt van populaire domain namen. Daarom zal bijvoorbeeld 'go-ogle' worden geflagged als verdacht. Als laatste zal er gekeken worden of er een link staat in de email. Het zal ongebruikelijk zijn als een klant een link stuurt waar een medewerker op moet gaan klikken. Daarom zal het script ook signaleren als er een link staat in de email. Verder roept het script een derde partij aan, die controleert of deze link malware, spam of phishing bevat.

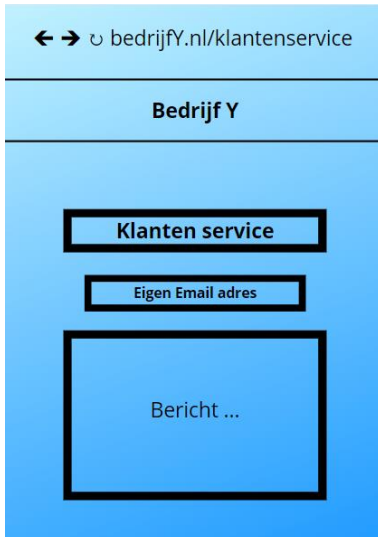
Het risico van een script is dat het de tekst in email niet goed kan lezen. Er wordt gebruik gemaakt van een Python library die de tekst leest en omzet in een string en dit wordt in de regel niet in 100% van de gevallen goed gedaan. Verder kan de medewerker de flags negeren en alsnog op een link klikken. Omdat er gebruik gemaakt wordt van een derde partij, heeft het bedrijf geen controle over de checks op de gevonden links.

Om het script te testen, ga ik een aantal test afbeeldingen gebruiken. Tijdens de DEMO video zal ik deze gebruiken. In deze test afbeeldingen staan afbeeldingen met een link, blacklisted email adressen en interpunctie email domein namen.

Als er helemaal geen maatregelen zijn tegen phising-aanvallen, is de kans groot dat er een succesvolle aanval voorkomt. Ransomware kan bijvoorbeeld het systeem blokkeren en het bedrijf zal dan veel geld moeten overmaken in vorm van cryptovaluta. Ook kunnen hackers spyware, worms, trojans en nog veel meer installeren op het systeem van het bedrijf. Gebruikersgegevens kunnen gestolen worden en op het Dark Web worden verkocht. Het is daarom super belangrijk om zo goed mogelijk phishing-aanvallen te voorkomen. De klanten willen natuurlijk zo snel mogelijk antwoord bij de klantenservice. Als er heel veel phishing e-mails doorkomen, vertraagd dit de antwoord snelheid naar 'echte' klanten.

Ontwerp en implementatie

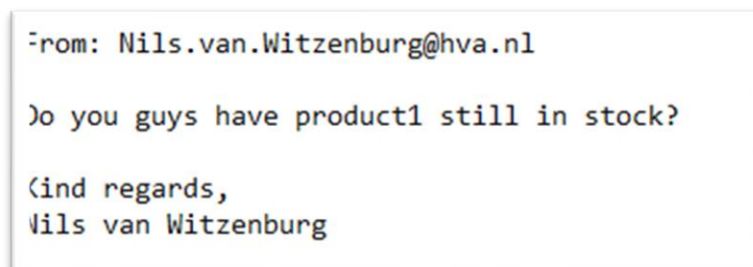
Het bedrijf heeft een website. Op deze website staat een klantenservice form, zie afbeelding 1 hieronder.



The screenshot shows a web browser address bar with 'bedrijfY.nl/klantenservice'. Below the address bar is a header 'Bedrijf Y'. The main content area has a light blue background and contains three input fields: 'Klanten service', 'Eigen Email adres', and a larger text area labeled 'Bericht ...'.

Afbeelding 1 – Klantenservice form

Op deze form zal de klant zijn gegevens invullen. Een ander programma in het systeem van het bedrijf controleert vervolgens of het een geldige email adres is. Als het email adres geldig is, zal de ingevulde form worden omgezet naar email en gestuurd worden naar de mailserver van het bedrijf. Een medewerker, die werkt in de klantenservice afdeling, zal deze email krijgen in zijn of haar mailbox, zoals in het voorbeeld hieronder.



The screenshot shows an email interface with the following text:
From: Nils.van.Witzenburg@hva.nl
Do you guys have product1 still in stock?
Kind regards,
Nils van Witzenburg

Afbeelding 2 – Email voorbeeld

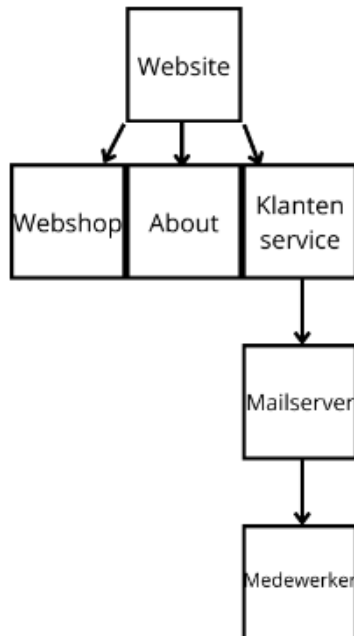
Het Python script zal uiteindelijk automatisch werken op de mailserver. Omdat ik tijdens dit project niet genoeg tijd heb om een mailserver te maken, heb ik besloten om een script te maken die handmatig gestart moet worden. In mijn Github¹ staat een beschrijving hoe het script gedraaid kan worden, verder is de code te zien in Bijlage 3. Het Python script controleert de interpunctie, email adres en links. Er wordt gebruik gemaakt van Python3, OpenCV-python en Python-tesseract.

Om te controleren of het een 'veilige' link is, wordt er gebruikt van een derde partij 'IPQualityScore'. Het heeft een fraud-prevention tool. Het detecteert 'bad actors' en blokkeert bots met gebruik van reputatiegegevens en gebruikersvalidatie. Ik roep de website API op en vervolgens geeft de website een JSON terug met data. Deze data bevat of de gecontroleerde website spam, malware of phishing kenmerken bevat.

¹ https://github.com/nilsvw/IP_ProjectSecurity

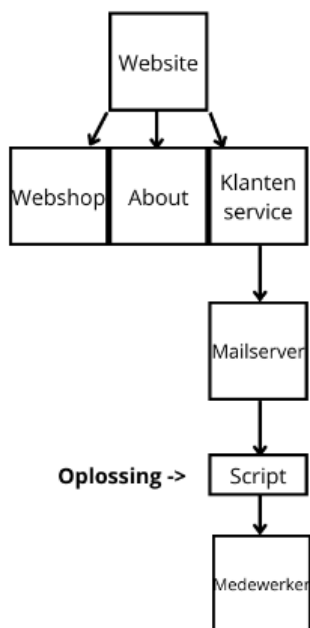
De script zal vragen om een deel tekst of een plaatje te scannen. Als de gebruiker gekozen heeft om een plaatje te scannen, zal het OpenCV en Pytesseract aanroepen.

Voor dat ik begon met het bedenken van een oplossing tegen phishing attacks, zag de infrastructuur van het bedrijf er zo uit.



Afbeelding 3 – Before

In afbeelding 4 staat waar mijn oplossing geïmplementeerd wordt.



Afbeelding 4 – After



Bij elke scan van een plaatje, wordt er een log bestand aangemaakt. In het log bestand staat de tijd van de scan, flags, checks en de volledig gescande tekst. De medewerker die de scan heeft uitgevoerd, kan de log bestand sturen naar mij als er enige twijfel is. De medewerker kan zelf ook de scan resultaten lezen uit de output van de terminal. Als de medewerker een flag ziet, moet hij of zij het direct melden bij mij.

Om het script te testen zijn er een aantal nep e-mails gemaakt (zie github en Bijlage 4 - Demo). Uit de testen bleek dat het script een aantal zwakke plekken heeft. Als er twee links staan in de e-mail, crashed het script. Dit komt door de manier waarop het script voor links checkt. Dit kan opgelost worden door de code te herschrijven zodat het de tekst blijft scannen voor links tot aan het einde. Verder kan het script verborgen links onder bijvoorbeeld een afbeelding, knop of hyperlink niet vinden. Dit kan opgelost worden door in de pakketjes van een email te kijken, nu wordt alleen gekeken naar het plaatje van de e-mail. Als laatste zwakke punt is dat de Python Tesseract library af en toe de tekst niet goed scanned. Dit kan ook opgelost worden door in de pakketjes te kijken van de e-mail.



Conclusie en advies

Dus het script scant het email adres op interpunctie en de blacklist. Verder als er een link staat in de email, wordt dit gedetecteerd. Vervolgens wordt deze link gescand met behulp van een derde partij.

Advies is om het script niet volledig te vertrouwen, omdat het af en toe de tekst niet goed kan scannen. Script heeft nog zwakke punten. Ook moet je als medewerker het log bestand sturen naar ICT medewerker als je iets niet vertrouwd. Verder is het handig voor het bedrijf om een aantal cursussen over phishing-aanvallen te kopen. Organiseer een dag waarin iedere medewerker die werkt met klantenservice moet komen. Hier wordt uitgelegd wat het script doet, hoe je het kan gebruiken en dat het alleen een hulpmiddel is.

Uiteindelijk wil ik mijn script automatisch laten draaien op de mailserver. Het script zal in de pakketjes zelf kijken. Op deze manier lost ik het probleem op dat het script niet links kan vinden die onder een afbeelding, knop of hyperlink zitten.

Met het script verminder ik veel succesvolle phishing aanvallen. Medewerkers zullen minder snel in phishing emails trappen, wat het bedrijf een stuk veiliger maakt. Om even terug te komen naar mijn probleemstelling: Wat is voor bedrijf Y de beste maatregel om phishing-aanvallen tegen te gaan op korte termijn? De beste maatregel op korte termijn is dus een Python script die de emails controleert. Op lange termijn zal het Python script aangepast worden, zodat het automatisch gedraaid wordt op de mailserver, en er cursussen worden gegeven over phishing-aanvallen.

Bronnen

APWG. (2022, February 23). PHISHING ACTIVITY TRENDS REPORT. *Apwg.Org*.

https://docs.apwg.org/reports/apwg_trends_report_q4_2021.pdf

Cyber Security Breaches Survey 2022. (2022, October 27). GOV.UK.

<https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2022/cyber-security-breaches-survey-2022>

DataFlair. (2019). Pros and Cons of Artificial Intelligence – A Threat or a Blessing? *Data-Flair.Training*.

<https://data-flair.training/blogs/artificial-intelligence-advantages-disadvantages/>

Desai, D. (2021, February 24). *30,000 Percent Increase in COVID-19-Themed Attacks*. Zscaler.

<https://www.zscaler.com/blogs/security-research/30000-percent-increase-covid-19-themed-attacks>

Witzenburg, van, N. (2022, November 11). *IP_ProjectSecurity*. Github.

https://github.com/nilsvw/IP_ProjectSecurity

Why AI is the key to developing cutting-edge cybersecurity. (2022, July 25). World Economic Forum.

<https://www.weforum.org/agenda/2022/07/why-ai-is-the-key-to-cutting-edge-cybersecurity/>

Bijlage 1 – Poster

Versie 1 voor feedback

HOW TO PREVENT PHISHING ATTACKS

Aanvallen met **667%** gestegen
Populairste vorm van Cybercrime

Nils van Witzenburg
Nils.van.Witzenburg@hva.nl
500849196
IC202



SOLUTION:

- Klein programma dat snel en simpel is



Python Script dat E-mails controleert



"I'm so happy I used this script instead of something expensive!"

Bijlage 2 – Poster v2

Versie 2 na feedback

HOW TO PREVENT PHISHING ATTACKS

Aanvallen met **667%** gestegen
Populairste vorm van Cybercrime

Nils van Witzenburg
Nils.van.Witzenburg@hva.nl
500849196
IC202



Python Script dat E-mails controleert

SOLUTION:

- Klein programma dat snel en simpel is



"I'm so happy I used this script instead of something expensive!"

Bijlage 3 – Code van script

https://github.com/nilsvw/IP_ProjectSecurity/blob/main/scanner.py

```

1  import cv2
2  import pytesseract
3  from urllib.request import urlopen
4  import json
5  import random
6  import datetime
7
8
9  __author__ = "Nils van Witzenburg 500849196"
10 __version__ = 2
11
12
13 def start_up():
14     """
15     Prints necessary information to user
16     """
17     print("|-----|")
18     print("| Scan your Image or Text with this simple Python script |")
19     print("| How to use the Image Function: |")
20     print("| Path Example: C:\\Users\\NilsP\\Desktop\\tekst.png |")
21     print("| Or 'tekst.png' if script in same directory |")
22     print("|-----|")
23     print("DISCLAIMER: SCRIPT ONLY SCANS VISABLE LINKS AND EMAIL ADDRESSES\\nSCRIPT CAN FALSE FLAG")
24
25 def text():
26     """
27     Checks if input contains word(s) from 'Blacklist.txt'
28     Checks if input contains punctuation
29     """
30     foundFlags = []
31     blockedPun = ['-']
32     flag = 0
33     text = input("Text: ")
34
35     # Check if any of the blacklisted domains are in found text
36     with open('blacklist.txt') as f:
37         for line in f:
38             blacklistDomains = line.strip()
39             if text in blacklistDomains:
40                 flag = 1
41                 break
42
43     # Check if blockedPun is in Domain
44     for i in text:
45         for j in blockedPun:
46             if j in i:
47                 flag = 1
48                 break
49
50     # Add found flag to list
51     if flag == 1:
52         foundMessage = f'WARNING: Potentially dangerous domain found > {text}'
53         foundFlags.append(foundMessage)
54     if flag == 0:
55         print("Probably no Misspelled or Fraudulent Domain \u2713\\n")
56
57     # If flag found, print them
58     if len(foundFlags) >= 1:
59         print(f"Found:\\n{foundFlags}")
60

```

```
61 def image():
62     """
63     Scans all text of image
64     Checks if image contains word(s) from 'Blacklist.txt'
65     Checks if image contains punctuation (Email domain)
66     Checks if image contains a link
67     Creates Log file after scan
68     """
69     # Location where Tesseract is installed (Required on Windows)
70     pytesseract.pytesseract.tesseract_cmd = r'C:\Users\WilsP\AppData\Local\Tesseract-OCR\tesseract.exe'
71
72     foundFlags = []
73     foundSpam = []
74
75     blockedPun = ['-']
76     urlFound = ['https']
77     flag = 0
78     flagUrl = 0
79
80
81     # Scan image for text
82     path = input("Path Image: ")
83     img = cv2.imread(path)
84     # Store all found text in 'text'
85     text = pytesseract.image_to_string(img)
86     # Only check text after @ and before .
87     # This means 'Nils.van.Witzenburg@hva.nl' -> 'hva' will be scanned
88     emailFound = text.split('@')[1]
89     emailSplit = emailFound.split('.')[0]
90
91     # Check if any of the blacklisted words are in found text
92     with open('blacklist.txt') as f:
93         for line in f:
94             blacklistDomains = line.strip()
95             emailFound = text.split('@')[1]
96             emailSplit = emailFound.split('.')[0]
97             if emailSplit in blacklistDomains:
98                 flag = 1
99                 break
100
101
102     # Check if blockedPun is in Domain
103     for i in emailSplit:
104         for j in blockedPun:
105             if j in i:
106                 flag = 1
107                 break
108
109     # Check for links in the email
110     for i in text:
111         for j in urlFound:
112             if j in text:
113                 flagUrl = 1
114                 break
115
116     # If flag or flagUrl found == 1 otherwise 0
117     if flag == 1:
118         foundMessage = f"WARNING: Potentially dangerous domain found"
119         foundFlags.append(foundMessage)
120     if flag == 0:
121         print("Probably no Misspelled or Fraudulent Domain \u2713\n")
122
```

```

123     if flagUrl == 1:
124         warning = f"WARNING: Potentially dangerous link found"
125         foundFlags.append(warning)
126
127         # Remove any whitespace from text
128         # Prevents potential wrong-read text
129         foundText = ("").join(text.split())
130
131         # Save url found without the https://
132         # 'https://www.paypal.com donate now' -> 'www.paypal.com'
133         allText = foundText.split("https://")[1]
134         fullUrl = allText.split(' ')[0]
135
136         # API key
137         key = "GcN2FFKF10tVJy15517quA3ZJSbSzdNw"
138
139         # Do a JSON request and save found data
140         request = f"https://ipqualityscore.com/api/json/url/GcN2FFKF10tVJy15517quA3ZJSbSzdNw/{fullUrl}"
141         response = urlopen(request)
142         data_json = json.loads(response.read())
143         spam = str(data_json['spamming'])
144         malware = str(data_json['malware'])
145         phishing = str(data_json['phishing'])
146         foundSpam.append("Found Spam: " + spam)
147         foundSpam.append("Found Malware: " + malware)
148         foundSpam.append("Found Phishing " + phishing)
149     if flagUrl == 0:
150         print("No URL's found")
151
152     # If flag found, print them
153     if len(foundFlags) >= 1:
154         print(f"Found:\n{foundFlags}")
155
156     if len(foundSpam) >= 1:
157         print(f"Checks:\n{foundSpam}")
158
159     # Create log file
160     if len(foundFlags) >= 0 or len(foundSpam) >= 0:
161         randomId = random.randint(1,10000)
162         main = f"LOG FILE - ID:{randomId}"
163         currentTime = datetime.datetime.now()
164         date = currentTime.strftime("%Y-%m-%d %H:%M:%S")
165         found = f"FOUND:\n{foundFlags}\nChecks:\n{foundSpam}"
166         with open(f'log_{randomId}.txt', 'w') as logFile:
167             logFile.write(f"{main}\nDate of Scan: [{date}]\n\n{found}\n\nContent of Image:\n{text}")
168
169
170     print("Do you want to see all found text of image? (Y/n)")
171     answer = input()
172     if answer == 'Y'.lower():
173         print(f"**START**\n{text}\n**END**")
174
175
176

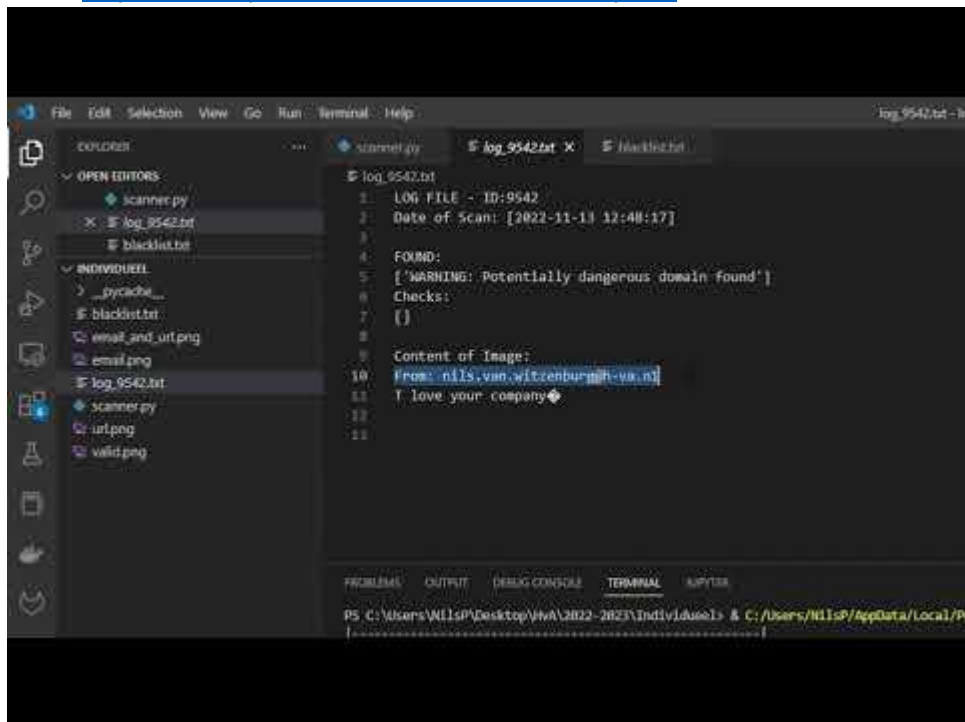
```

```
177 def option():
178     """
179     Asks user to scan text or image
180     """
181     print("Want to scan image or text? (I/t)")
182     answer = input()
183     if answer == 'I'.lower():
184         image()
185     if answer == 'T'.lower():
186         text()
187
188 def main():
189     """
190     Start program with the necessary functions
191     Provides user with extra information
192     Asks user to run program again
193     """
194     start_up()
195     option()
196
197     print("\033[1m" + "PLEASE NOTE"+
198           "\n- Double check the content of the email" +
199           "\n- Double check the email address" +
200           "\n- Don't click any links" +
201           "\n- Send the log file to IT department" + "\033[0m")
202
203     print("Do you want to run the script again? (Y/n)")
204     answer = input()
205     if answer == 'Y'.lower():
206         main()
207
208 if __name__ == '__main__':
209     main()
210
```

Bijlage 4 – Demo

Demo gaat over hoe het script werkt en het testen ervan.

Link = <https://www.youtube.com/watch?v=PI4YaRyrZYY>



```
File Edit Selection View Go Run Terminal Help
EXPLORER
  OPEN EDITORS
    scanner.py
    log_9542.txt
    blacklist.txt
  INDIVIDUEEL
    _pycache_
    blacklist.txt
    email_and_url.png
    email.png
    log_9542.txt
    scanner.py
    url.png
    valid.png

log_9542.txt
1: LOG FILE ~ ID:9542
2: Date of Scan: [2022-11-11 12:48:17]
3:
4: FOUND:
5: ['WARNING: Potentially dangerous domain found']
6: Checks:
7: {}
8:
9: Content of Image:
10: from: nils.van.wittenburg@va.nl
11: I love your company
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL OUTPUT

PS C:\Users\NilsP\Desktop\YWA\2022-2023\Indiv\duo\1 & C:\Users\NilsP\AppData\Local\P...

Logboek

Docent feedback

06/10/2022

Na het geven van mijn pitch keek de docent uit naar waar mijn zwakke punten zitten in het script.

02/11/2022

“Is het belangrijk voor de gebruiker om te weten waar de scanner wel/niet op scant zodat die niet per ongeluk denkt 100% safe te zijn? Soort disclaimer? En afsluiten met twee adviezen. Advies om toch voorzichtig te zijn of advies om dit de moeite waard lijkt om te melden bij personen in de organisatie die over information security gaan of dat ze moeten controleren of ze het goede aan het scannen zijn. Dat soort zaken. Dan bij de alert: specifiek wat er waar gevonden is zodat de security persoon goed inzicht heeft? Soort rapportje/pdf misschien?”

Ik ben de volgende dag aan de slag gegaan om de feedback van 02/11/2022 te verwerken.

Ontvangen en gegeven feedback

De feedback die ik Noa gaf op zijn poster tijdens pitch: structuur veranderen, door bijvoorbeeld alle tekst links en dan plaatjes rechts te zetten. Of de tekst nummeren.

De feedback die ik Robin gaf in DLO op zijn poster: sommige delen tekst iets vergroten, maar verder duidelijke poster en creatief gemaakt.

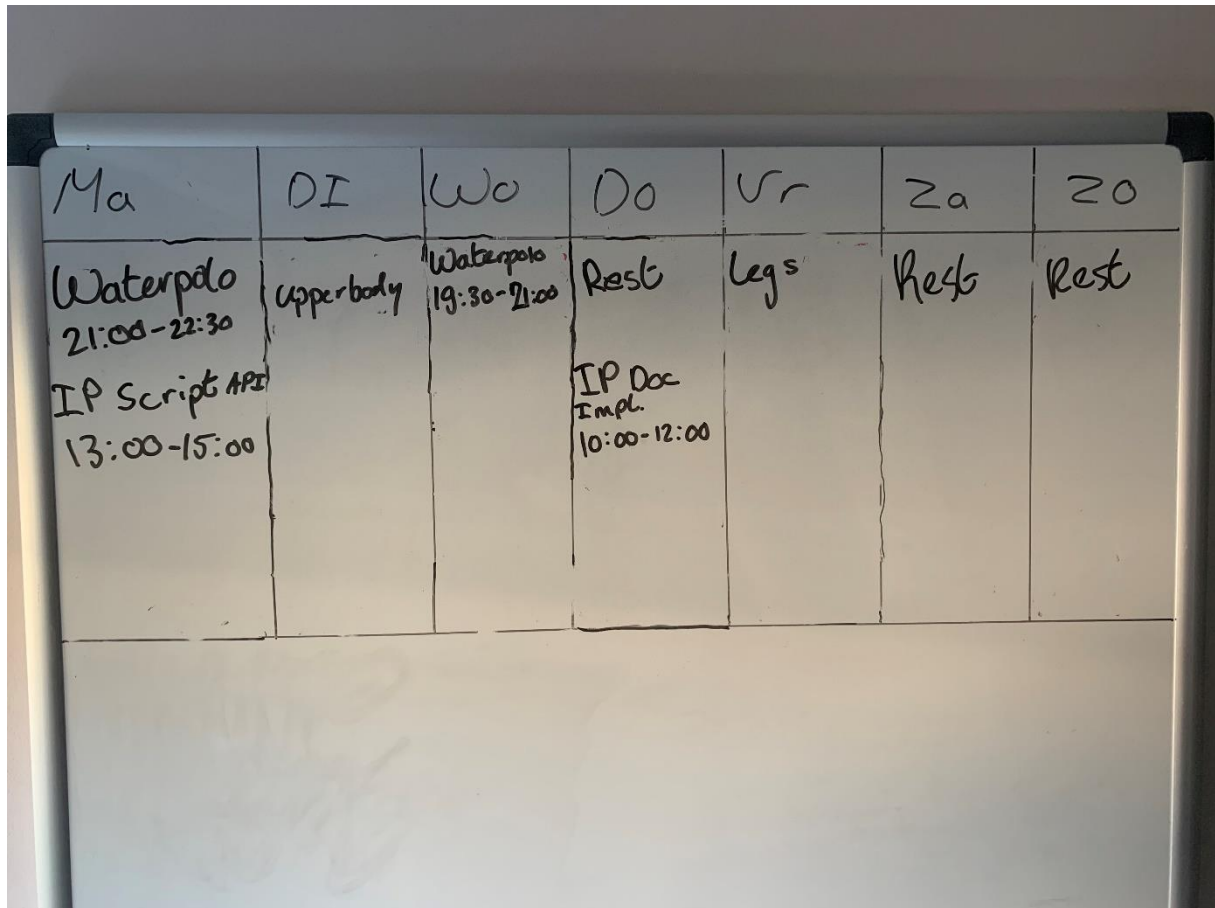
De feedback die ik Jan gaf in DLO op zijn poster: duidelijke structuur, plaatjes geven duidelijke aan waar de poster over gaat. Ik zou de stappen wat meer uitleg geven waarvoor ze zijn, dus misschien een titel geven.

Ontvangen feedback op pitch en poster: heb een duidelijke stem tijdens presenteren. Het is goed dat ik presenteerde over wat niet op mijn poster staat. Dus dat ik informatie gaf dat niet op mijn poster stond. Als tip om aan te geven waar de phishing-aanvallen voorkomen en waar ik ze ga scannen. Verder is het een mooie poster met een logische volgorde. Komisch element is een leuke toevoeging.

Ontvangen feedback op script: append functies compacter maken. Ziet er verder goed uit.

Planning

Ik heb ervoor gekozen om Trello te gebruiken als mijn planner. Hier kan ik duidelijk voor mijzelf weergeven wat ik nog moet doen, waar ik mee bezig ben en wat ik heb afgerond. De planning is verdeeld in drie sprints. In het hoofdstuk 'Werkwijze' wordt uitgelegd waarom. In de afbeelding per sprint staan alle taken. Verder heb ik onder de afbeelding bijgehouden aan welke taak ik heb gewerkt en op welke datum. Met behulp van Trello en mijn whiteboard thuis heb ik een planning gemaakt. Ik heb gekeken in Trello welke taken er gedaan moesten worden, en vervolgens verdeelde ik die over een week. Op mijn whiteboard schreef ik de tijd en taak per dag. Zie voorbeeld hieronder. Ik heb overige taken van andere vakken weggehaald, zodat het duidelijker is voor dit project.



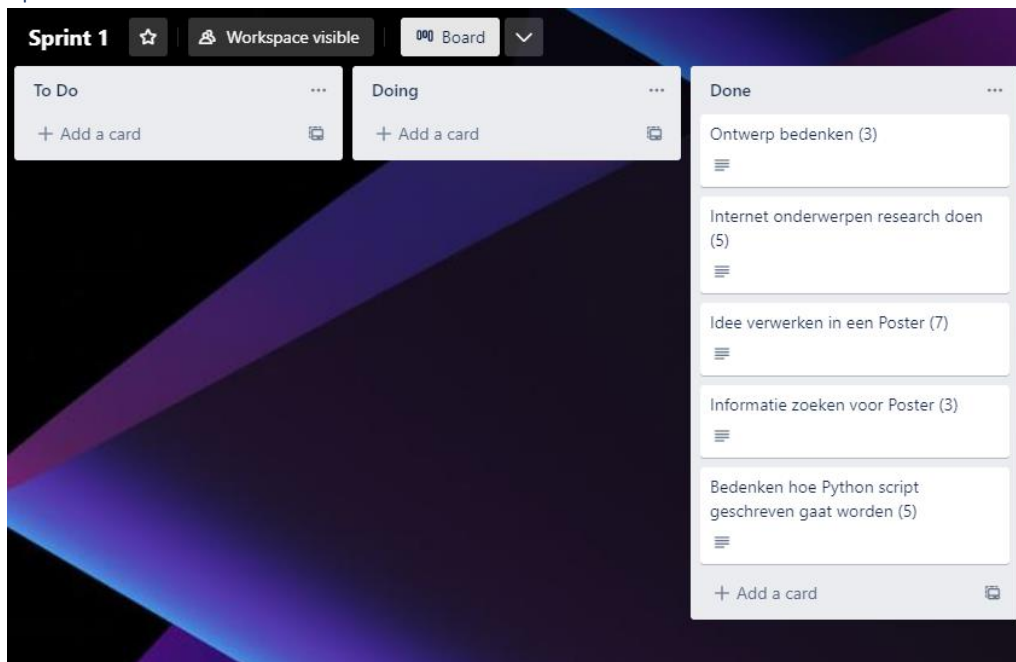
Ma	Di	Wo	Do	Vr	Za	Zo
Waterpolo 21:00-22:30 IP Script API 13:00-15:00	Upperbody	Waterpolo 19:30-21:00	Rest IP Doc Impl. 10:00-12:00	Legs	Rest	Rest

Deadlines

06/10/2022 Pitch Poster

18/11/2022 Inleveren Individueel Product

Sprint 1



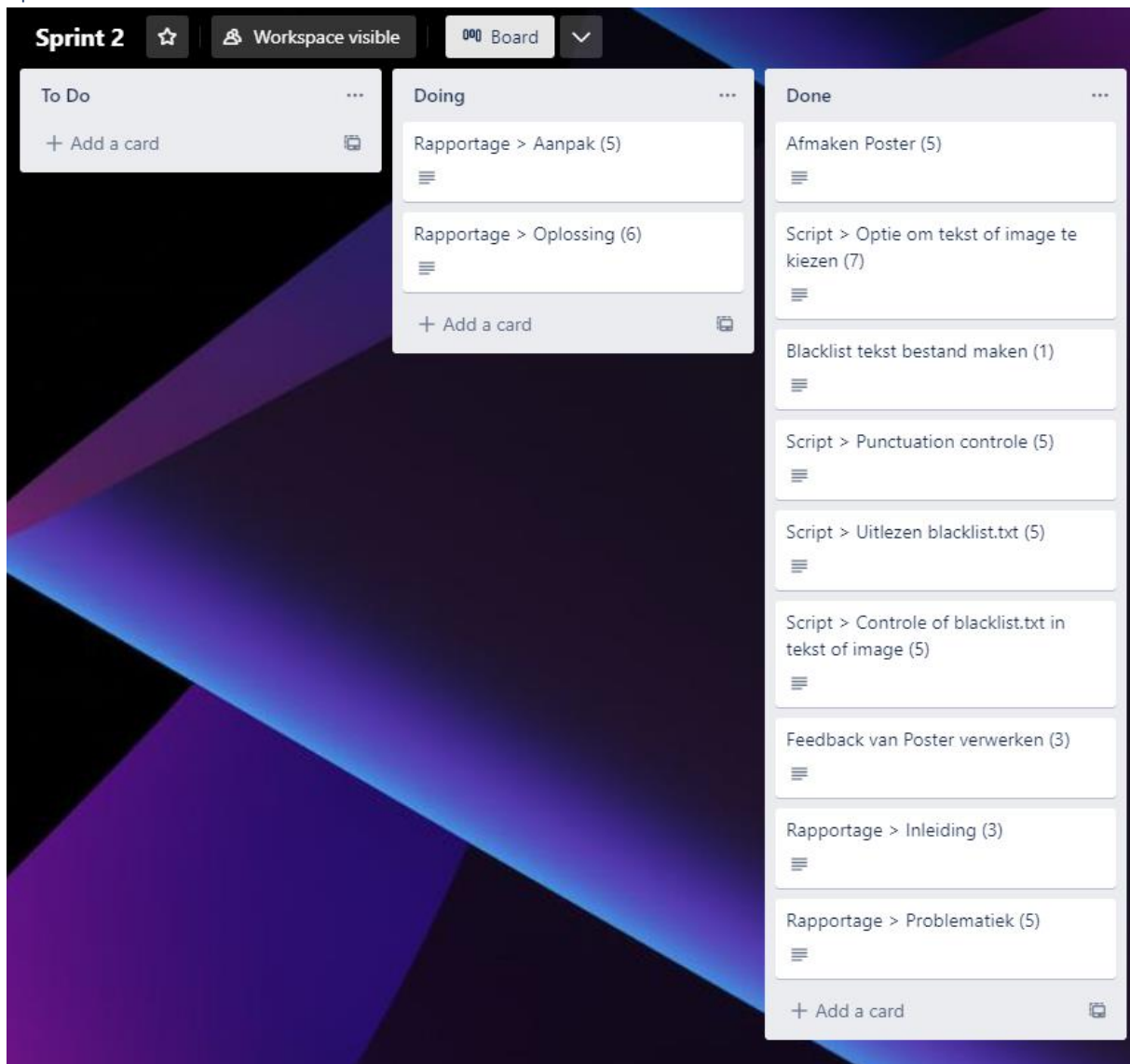
06/09 en 07/09 Ben ik begonnen met het bedenken van Cyber Security problemen. Ik heb op internet opgezocht of er veel te vinden is over het onderwerp en of het wel mogelijk is om gekozen probleem op te lossen.

09/09 Ben ik begonnen met het maken van mijn poster voor de Pitch. Hier heb ik mijn ideevorming vast gesteld: Script die phishing emails kan detecteren.

11/09 Heb ik de studiehandleiding doorgelezen. Bekeken wat er allemaal in de poster moet.

16/09 Ben ik onderzoek gaan doen over mijn onderwerp. Dit is nodig voor de Rapportage en Poster. Bronnen genoteerd in Rapportage en mijn poster ingericht. Ook ben ik begonnen met het schrijven van mijn script (voorbereiding voor sprint 2). Dit is nog heel erg de basis en het kan alleen nog maar de keuze van de gebruiken vragen, Image of Tekst scannen.

Sprint 2



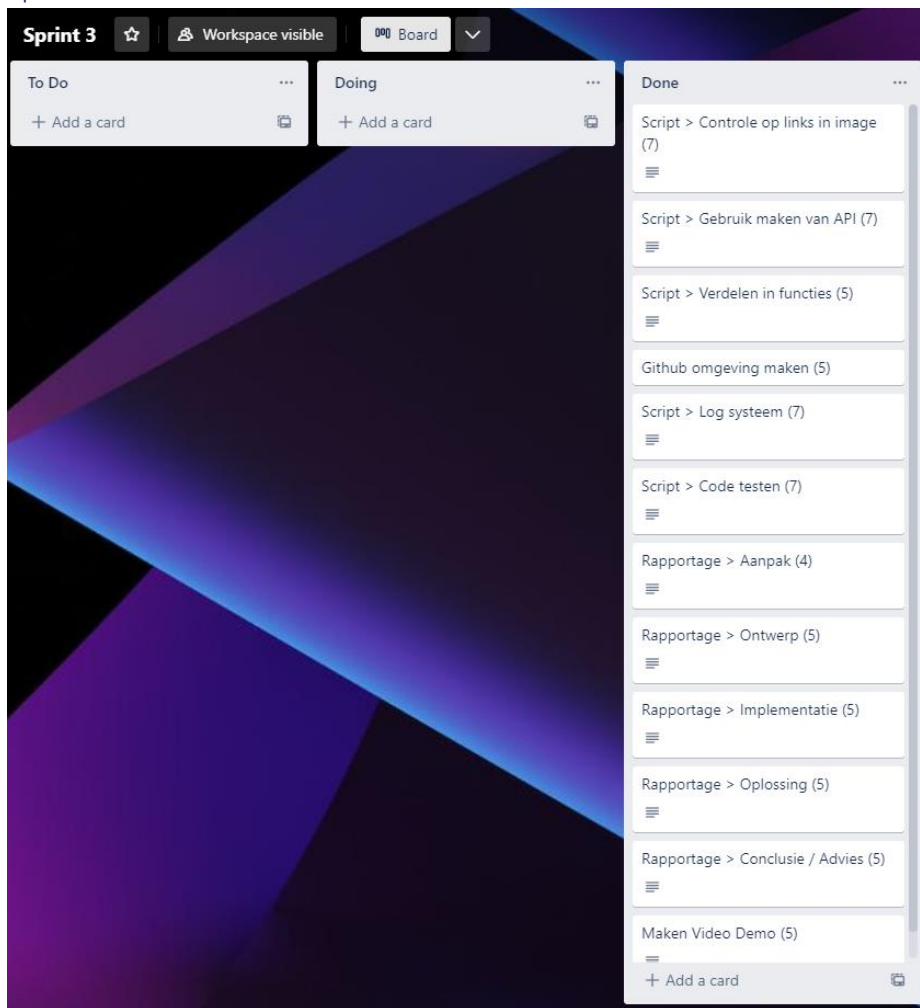
21/09 Script, afmaken optie om tekst of image te scannen. Ook poster afgemaakt voor de Pitch.

22/09 Script, tekst bestand gemaakt genaamd 'Blacklist.txt'. Code geschreven om tekst bestand uit te lezen. Rapportage inleiding en problematiek gemaakt.

06/10 Pitch gegeven met poster. De feedback ontvangen van medestudenten. Ben dezelfde dag feedback gaan toepassen op de poster. Begonnen met schrijven van Aanpak en Oplossing. Dit zijn momenteel alleen steekwoorden, omdat het lastig is om nu al alles 100% te weten.

07/10 Implementeren en schrijven code om tekst uit een plaatje te lezen en op te slaan. Toevoeging dat bij beide opties (tekst en image), er gecontroleerd wordt of er enige interpunctie in email-domein staat.

Sprint 3



09/10 Maken code dat controleert of er een link staat in de tekst van de image.

14/10 API gebruiken die de link gebruikt uit de image. Dit betekent het opvragen van een API-key om de API te kunnen gebruiken. Daarna moet de code een link vinden, opschonen en doorgeven aan de API.

25/10 Code in verschillende functies zetten zodat de code efficiënter en overzichtelijker is. Verder heb ik vandaag de code getest met behulp van verschillende situaties.

27/10 Gewerkt aan de Rapportage. Aanpak en Oplossing afgemaakt. Verder heb ik gewerkt aan Ontwerp en implementatie. Dit 90% afgemaakt.

28/10 Code in Github omgeving zetten. Test afbeeldingen, script en blacklist.txt toegevoegd. Verder een README bestand gemaakt waarin staat hoe je zelf thuis het script kan gebruiken.

03/11 Alle bugs uit het script gehaald. Log systeem gemaakt.

04/11 Afmaken van Aanpak en Oplossing in Rapportage. Ook Ontwerp en Implementatie afgemaakt.

11/11 Maken conclusie in Rapportage en maken van DEMO video. Hierbij sluit ik ook sprint 3 af.

Retrospective Sprint 1

Aan het einde van een sprint ga ik reflecteren op mijn proces en werkwijze. Tijdens sprint 1 heb ik vooral onderzoekend werk gedaan. Dit is van belang omdat ik dan goed een beeld krijg van wat ik wil gaan maken. Ik ben tevreden over hoe het gegaan is tijdens deze sprint. Ik heb al mijn taken kunnen afmaken. Bij de volgende sprint ben ik van plan om meer taken te doen. Ik heb vooruit gekeken over wat ik allemaal nog moet doen, en zie dat ik meer taken moet doen tijdens sprint 2 en 3. Volgende sprint ga ik beginnen aan het coderen van het script. Volgende keer ga ik bij mijn poster dieper in details.

Looptijd: 06-09 / 20-09

Retrospective Sprint 2

Tijdens sprint 2 ben ik hard te werk gegaan met het coderen van mijn script. Ik had gepland om wat meer aan de rapportage te werken, maar ik kwam er achter dat dit nog niet kon. Dit kon pas wanneer ik mijn script bijna af had. Dit is geen probleem, omdat ik de twee taken meeneem naar sprint 3. Ik ben tevreden over hoe het gegaan is tijdens sprint 2. Volgende sprint ga ik meer documenteren en de laatste loodjes leggen bij het script. Ik merk dat er een aantal fouten zitten in het Python script, deze wil ik zo snel mogelijk eruit halen. Ook zie ik dat een Python library niet goed werkt. Ik ben toch van plan om dit te gebruiken omdat het een proof of concept is, maar volgende keer wil ik eerder de libraries testen of ze het wel goed doen.

Looptijd: 21-09 / 07-10

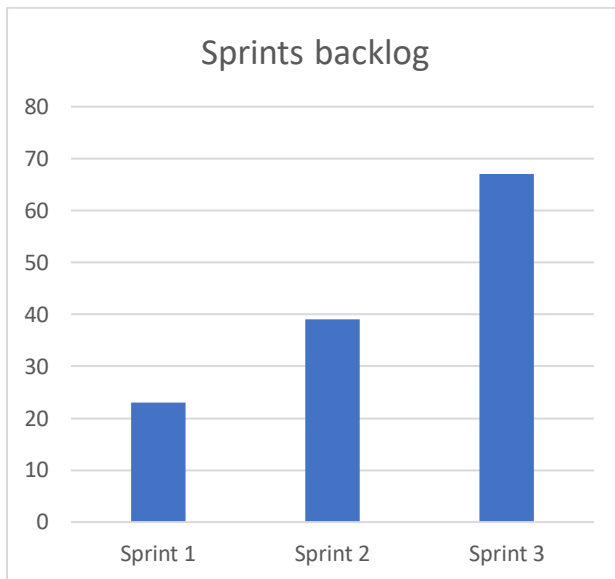
Retrospective sprint 3

In sprint 3 heb ik de Rapportage en Script af kunnen maken. Omdat ik goed had ingeschat hoeveel tijd ik ongeveer nodig had om taken af te ronden. Deze sprint is langer dan de andere twee sprints omdat ik 2 weken van de sprint niet aan het individuele product heb gewerkt. Door goed te plannen had ik genoeg tijd om het project af te maken. Ik had deze sprint kunnen verdelen in sprint 3 en 4, maar dit heb ik niet gedaan. Dit heb ik gedaan omdat ik het individuele project in drie delen had verdeeld; onderzoeken, documenteren en coderen. Ik vond het handiger om het op 3 sprints te houden daarom. Ik werk niet met een groep dus daarom kan ik flexibel zijn met de looptijd van de sprints. Volgend project wil ik meer feedback vragen aan de docent tijdens de les. Omdat ik een oplossing gekozen had waar ik al ervaring mee had (Python) had ik geen vragen. Dus volgende keer wil ik een uitdagendere oplossing maken. Ik ben blij met de resultaten van mijn Rapportage en technisch product, het voldoet aan mijn eisen.

Looptijd: 09-10 / 11-11

Sprints backlog

Op basis van de punten die elke userstory gekregen hebben, is er een sprints backlog gemaakt. Het geeft visuele inzicht over de voortgang per sprint.



Wat opvalt is dat het bijna een lineaire stijging is. Het is erg lastig om al met de documentatie van het product te beginnen als er nog niet een volledig product gemaakt is. Daarom zijn veel punten in sprint 3 documentatie. In sprint 1 is er onderzoek gedaan en een poster gemaakt van mijn gekozen onderwerp. Ook is er gepland voor sprint 2 en 3. In sprint 2 is er begonnen aan het script, wat meer tijd kost, wat ook te zien is. Wat ik merkte is dat sprint 2 en 3 mij veel meer tijd gingen kosten, daarom is hier rekening mee gehouden.



Methoden

Er is ervoor kozen om de scrum-methode te gebruiken tijdens dit project. Omdat dat het een individuele opdracht is, ben ik zelf de Scrum Master en Product Owner. Zelf bedenk ik de User Stories, dit zijn korte en eenvoudige beschrijving van taken. Deze User Stories worden gezet in Trello als beschrijving per taak. Dit is direct mijn planning, met behulp van een whiteboard. Scrum is eigenlijk bedoelt voor team projecten, maar tijdens het individuele project wordt het toch gebruikt. Dit is gedaan omdat mijzelf dan forceer om te gaan reflecteren over mijn werk. Verder helpt om goed overzicht te houden over al mijn taken.

Er is voor gekozen om de sprints niet te verdelen per tijd periode maar per onderdeel van het project. Sprint 1 is meer onderzoekend gericht, sprint 2 is meer coderend gericht en sprint 3 is meer documenterend gericht. Maar het plan is wel om sprint 1 en 2 twee weken lang te doen. Dit wordt gedaan omdat ik mijzelf dan forceer om binnen twee weken onderzoek te doen naar het onderwerp, en mijn context en probleemstelling duidelijk te hebben. Hetzelfde geldt met sprint 2, daar forceer ik mijzelf om al een groot deel van het technisch product af te hebben. Een sprint wordt geeindigd met een retrospective. Bij een retrospective wordt er gereflecteerd op mijzelf hoe de afgeronde sprint is gegaan. In sprint 3 gaat er vooral gedocumenteerd worden. Verder worden de laatste loodjes gelegd op het technisch product.



Individuele reflectie

Het einde van het individuele product 1 is nu aangekomen. Ik ben begonnen met onderzoek, heb daarna een technisch product gemaakt en als laatste een documentatie. Ik vind dat ik een goede keuze heb gemaakt om het Scrum te werken. Met behulp van Scrum heb ik mij gedurende het project gereflecteerd op mijn werk. Ook vind ik het een goede keuze om de sprints te verdelen in taken in plaats van tijd. Omdat ik voor mijzelf twee weken 2x goed gewerkt heb, had ik veel tijd over in sprint 3. Dit is een goede zet geweest omdat ik wist dat ik tijdens sprint 3 het erg druk zou krijgen met andere vakken.

Wat ik volgend individuele project beter kan doen is een onderwerp kiezen waar ik minder verstand van heb. Nu heb ik het over een onderwerp gehad waar ik al wat van wist en mijn oplossing is in Python, waar ik al redelijk goed in ben.

De waarde van het eindproduct is goed. Ik heb nieuwe trucjes geleerd in Python en ik heb een beter beeld over hoe een bedrijf zich kan verdedigen tegen phishing-aanvallen. Als criteria op het project heb ik wel dat het meer eerst een oplossing vinden en dan pas een probleem. Het is erg lastig om een probleem te kiezen, waar je nog geen oplossing voor kan bedenken. Dus misschien voor in de toekomst dat de docent per groepje een probleem al geeft. Hierbij vermijdt je dus dat je eerst een oplossing bedenkt voor het probleem.