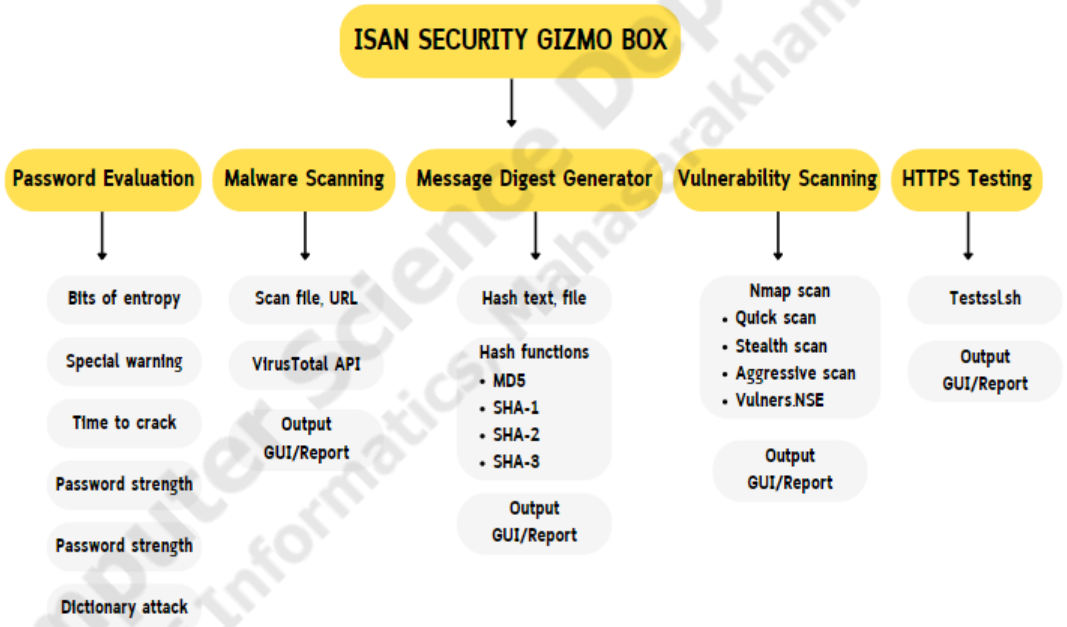
บทที่ 3

ขั้นตอนการดำเนินงาน

ในบทนี้จะกล่าวถึงการใช้ PyQt6 ร่วมกับภาษา Python เพื่อพัฒนา Graphical User Interface (GUI) ของเครื่องมือ ISAN Security Gizmo Box และขั้นตอนการดำเนินงานในการสร้างทั้ง 5 เครื่องมือ ได้แก่ Password Evaluation, Malware Scanning, Message Digest Generator, Vulnerability Scanning, HTTPS Testing โดยจะมีกรอบการดำเนินงานดังนี้

3.1 กรอบการดำเนินงาน



ภาพประกอบที่ 3.1 กรอบการดำเนินงาน

ภาพรวมของโครงงานปริญญานิพนธ์แสดงให้เห็นถึงกรอบการดำเนินงาน ดังนี้

1) ศึกษาการคำนวณหาข้อมูลเชิงปริมาณ Bits of entropy และนำผลลัพธ์ที่ได้มาคำนวณหาข้อมูลเชิงคุณภาพ

2) ศึกษามาตรฐาน NIST Special Publication 800-63B

3) ศึกษาเทคนิคการโจมตี Dictionary attack โดยแบ่งเป็น 2 รูปแบบ คือ Straightforward dictionary attack และ Skipping attack

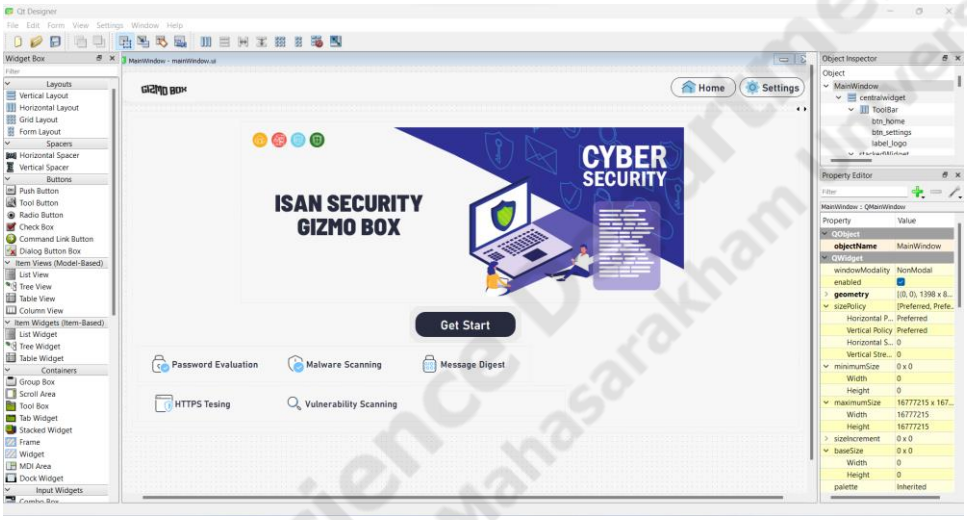
4) ศึกษาและทดสอบการเรียกใช้งาน VirusTotal API

5) ศึกษาและทดสอบ option Nmap และการใช้งาน Nmap ร่วมกับสคริปต์ Vulners.NSE

6) ศึกษาทดสอบ Testssl.sh, HSTS Preload

7) ศึกษาแฮชฟังก์ชันที่จะนำมาใช้ในเครื่องมือ Message Digest Generator ได้แก่ MD5, SHA-1, SHA-2(224, 256, 384, 512) , SHA-3(224, 256, 384, 512)

3.2 การพัฒนา Graphical User Interface (GUI) ด้วย Qt Designer



ภาพประกอบที่ 3.2 หน้า Qt Designer

พัฒนา Graphical User Interface (GUI) ด้วย Qt Designer เพื่อเป็นตัวกลางในการสื่อสารระหว่างผู้ใช้งานและเครื่องมือต่าง ๆ โดยด้านซ้ายจะเป็น Widget Box หรือ Components ที่ประกอบไปด้วย Layouts, Spacers, Buttons, item Views, Containers, Input Widgets และ Display Widgets ที่สามารถใช้งานได้และด้านขวาคือ Properties ที่เกี่ยวข้องกับการตกแต่งหรือปรับเปลี่ยน Style Sheet ให้กับแต่ละ Object โดยประกอบไปด้วย Object Inspector จะแสดง Object ที่มีใน Frame และส่วนของ Property Editor เช่น font, geometry, window title

```
15  class Main(QMainWindow):
16
17      def __init__(self):
18          super(Main, self).__init__()
19          loadUi("./assets/ui/mainWindow.ui", self)
```

ภาพประกอบที่ 3.3 เรียกใช้งานหน้า Graphical User Interface (GUI)

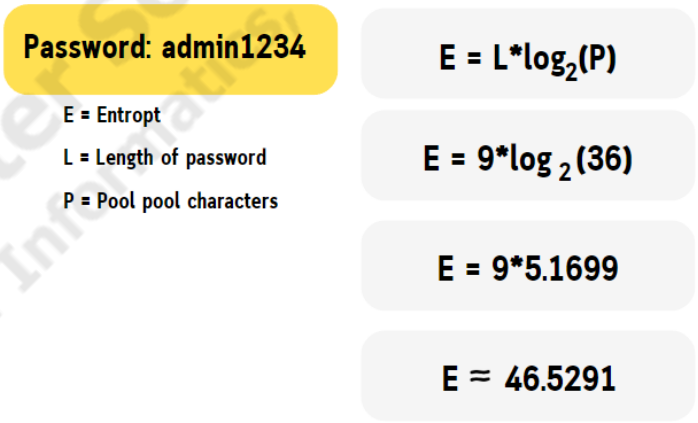
การเรียกใช้หน้า Graphical User Interface (GUI) ที่สร้างด้วย Qt Designer ทำการ import ฟังก์ชัน loadUi จาก PyQt6.uic เพื่อใช้ในการโหลดหน้า Graphical User Interface (GUI) ซึ่งคลาส Main จะเป็นสืบทอดมาจาก QMainWindow

3.3 เครื่องมือ Password Evaluation

3.3.1 การวิเคราห์ข้อมูลเชิงปริมาณ

Pool	Element	Pool size
Lower case	a-z	26
Upper case	A-Z	26
Number	0-9	10
Special symbols	~!@#$%^&*()_+{}[]?/	32

ภาพประกอบที่ 3.4 Pool of Characters



ภาพประกอบที่ 3.5 ตัวอย่างการคำนวณ Bits of Entropy

สำหรับเครื่องมือประเมินรหัสผ่านผู้ใช้งานสามารถป้อนรหัสผ่านที่ต้องการเพื่อใช้ในการคำนวณ Entropy โดยใช้สูตรการคำนวณ E = Length * Log2(Pool_of_Chars) และจะแสดงผลลัพธ์เป็นข้อมูลเชิงปริมาณและข้อมูลเชิงคุณภาพจากการศึกษาข้อมูลพบว่าเว็บไซต์ keepass.info ระบุข้อมูลเชิงคุณภาพของรหัสผ่านที่มีความแข็งแกร่งนั้นมีขนาดของเอนโทรปีอยู่ที่ 112-128 บิต ส่วนเว็บไซต์ barldung.com ระบบว่าข้อมูลเชิงคุณภาพของรหัสผ่านที่มีความแข็งแกร่งมีขนาดของเอนโทร

ปี อยู่ที่ 75-100 บิต จากข้อมูลดังกล่าว กล่าวโดยสรุปคือ keepass.info เป็นเว็บไซต์ Password Manager มีจุดประสงค์ให้คอมพิวเตอร์ตั้งรหัสผ่านให้ ทำให้เอนโทรปีของรหัสผ่านมีค่าที่สูงแต่รหัสผ่านที่ได้นั้นมีความยาวและยากเกินกว่ามีมนุษย์จะจดจำได้

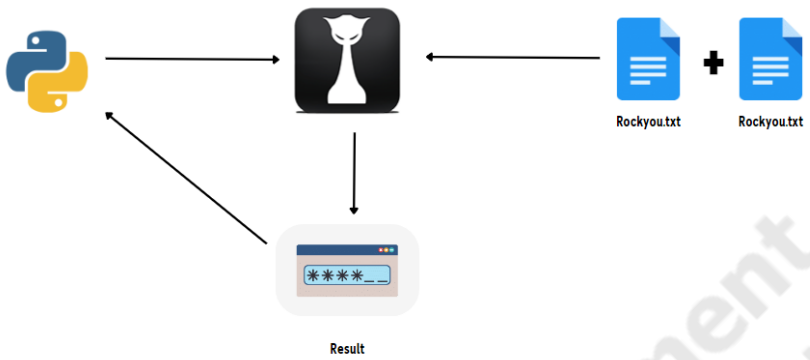
3.3.2 การวิเคราะห์ข้อมูลเชิงคุณภาพ

```
if password < 8 then
    return "Weak"
else
    if entropy < 50 then
        return "Weak"
    else if entropy < 50 then
        return "Medium"
        else
            return "Good"
        end
end
```

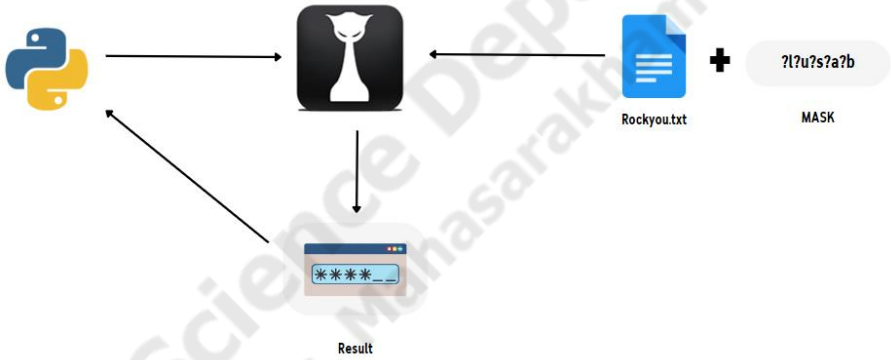
ภาพประกอบที่ 3.6 เงื่อนไขการแสดงข้อมูลเชิงคุณภาพ

จากภาพประกอบที่ 3.5 เงื่อนไขการแสดงข้อมูลเชิงคุณภาพ แสดงให้เห็นว่ารหัสผ่านที่มีความยาว 8 หลัก และเมื่อนำไปคำนวณจะได้ E = 8 * 7 โดย 7 บิต มาจาก $\log_2(128)$ แสดงให้เห็นว่ามีความหลากหลายของรูปแบบอักขระถึง 128 รูปแบบ ซึ่งเมื่อเทียบกับ pool of characters แล้วชี้ให้เห็นว่ารหัสผ่านที่มีค่า entropy ตั้งแต่ 50 บิต ขึ้นไปก็เพียงพอแล้วจึงสรุปได้ว่าหากรหัสผ่านที่มีค่า entropy ตั้งแต่ 50 บิต ขึ้นไปถือว่าเป็นรหัสผ่านระดับกลางแต่หาก entropy ตั้งแต่ 80 บิต ขึ้นไปโดยคำนวณได้จากสูตร E = 12 * 7 ซึ่งจะมีค่าเท่ากับ 84 บิต จะถือว่าเป็นรหัสผ่านที่มีความแข็งแกร่งหรือเป็นรหัสผ่านที่ดีต่อการใช้งาน โดยยังไม่ได้เปรียบเทียบกับรหัสผ่านใน wordlist ของ attacker จึงสรุปได้ว่า entropy ต่ำสุดที่จะยอมรับได้คือ 50 บิต และ entropy ที่ 80 บิต ขึ้นไปจะถือว่ามีความปลอดภัยโดยถ้าหากอยู่ระหว่าง 51-79 บิต หมายความว่ารหัสผ่านนั้นพอใช้ได้และอาจจะต้องทำให้ได้ถึง 80 บิต หลังจากนั้นจะมีการนำรหัสผ่านที่ป้อนเข้ามาไปตรวจสอบกับ Top 200 most common password of the years ย้อนหลัง 3 ปี ที่จัดโดย Nordpass.com

นอกจากนี้เครื่องมือประเมินรหัสผ่านจะทำการนำรหัสผ่านที่ป้อนเข้ามาไปตรวจสอบกับ Dictionary ว่าเป็นรหัสผ่านที่ตรงกับ wordlist หรือไม่ โดยใช้วิธี Dictionary attack แบ่งเป็น 2 รูปแบบ คือ Straightforward Dictionary attack และ Skipping attack ซึ่ง Skipping attack ประกอบด้วย Combinator attack, Hybrid attack



ภาพประกอบที่ 3.7 เทคนิคการทำ Combinator attack ด้วย Python และ Hashcat



ภาพประกอบที่ 3.8 เทคนิคการทำ Hybrid attack ด้วย Python และ Hashcat

```
276         for char in password:
277             if char.isdigit():
278                 self.chk_numeric.setChecked(True)
279                 self.chk_numeric.setIcon(self.check_icon)
280             elif char.isupper():
281                 self.chk_upper.setChecked(True)
282                 self.chk_upper.setIcon(self.check_icon)
283             elif char.islower():
284                 self.chk_lower.setChecked(True)
285                 self.chk_lower.setIcon(self.check_icon)
286             elif char in '!@#$%^&*()_+-=':
287                 self.chk_special.setChecked(True)
288                 self.chk_special.setIcon(self.check_icon)
289             else:
290                 pass
291 
292 
293             if len(self.lineEdit_password.text()) >= PasswordEvaluation.minpasswordlength: # 8 chars
294                 self.chk_length.setChecked(True)
295                 self.chk_length.setIcon(self.check_icon)
296 
297         return password
```

ภาพประกอบที่ 3.9 การตรวจสอบองค์ประกอบของรหัสผ่าน

ฟังก์ชันนี้ใช้ในการตรวจสอบองค์ประกอบของรหัสผ่านโดยอ้างอิงตามค่ามาตรฐาน NIST Special Publication 800-63B ได้แก่ ตัวอักษรพิมพ์เล็ก ตัวอักษรพิมพ์ใหญ่ ตัวเลข อักขระพิเศษ และตรวจสอบ

จำนวนหลักของรหัสผ่านผู้ใช้งานโดยสามารถดูได้ว่ารหัสผ่านของตนเองมีองค์ประกอบอะไรบ้างและขาดอะไรไปบ้าง

3.3.3 การคำนวณ Estimating Password Cracking Times

บทความ "Estimating Password Cracking Times" ของเว็บไซต์ Better Buys กล่าวถึงระยะเวลาในการถอดรหัสรหัสผ่าน โดยอาศัยปัจจัยต่าง ๆ เช่น ความยาวของรหัสผ่าน ประเภทของรหัสผ่าน และการใช้เทคโนโลยีการถอดรหัสซึ่งเทคโนโลยีการถอดรหัสแบบใหม่ ๆ ช่วยให้สามารถถอดรหัสรหัสผ่านได้เร็วขึ้น อย่างไรก็ตาม ความยาวของรหัสผ่านยังคงเป็นปัจจัยสำคัญที่สุดในการป้องกันการถอดรหัสรหัสผ่าน สูตรที่ใช้ในการคำนวณคือ Password cracking time = (password length * number of combinations) / decryption speed โดย ความยาวรหัสผ่าน คือจำนวนตัวอักษรในรหัสผ่าน จำนวนชุดค่าผสม คือจำนวนรหัสผ่านที่เป็นไปได้ทั้งหมดสำหรับรหัสผ่านที่มีความยาวที่กำหนด ความเร็วในการถอดรหัส คือจำนวนรหัสผ่านที่สามารถถอดรหัสได้ในแต่ละหน่วยเวลา โดยตามข้อมูลของ John the Ripper โปรเซสเซอร์ที่คนใช้มากที่สุดในปี 2023 คือ AMD Ryzen 5 7600X ซึ่งมี GFLOPS 1.17 TFLOPS โปรเซสเซอร์รุ่นนี้ได้รับความนิยมเนื่องจากมีราคาไม่แพงและมีประสิทธิภาพเพียงพอเมื่อแปลง TFLOPS เป็น Keys Per Second จะมีค่าเท่ากับ 1,170,000 Keys Per Second

Password cracking time = (7 * 62^7) / 1,170,000
= 21069489.1
= 21069489.1 / (24 * 60 * 60)
= 2.9177 วัน หรือประมาณ 2 วัน 9 ชั่วโมง

ภาพประกอบที่ 3.10 ตัวอย่างการคำนวณ Password cracking time

```python
def calculate_entropy(self, password):
    # check if password is empty
    if password == '':
        self.chk_length.setIcon(self.warning_icon)
        self.chk_numeric.setIcon(self.warning_icon)
        self.chk_upper.setIcon(self.warning_icon)
        self.chk_lower.setIcon(self.warning_icon)
        self.chk_special.setIcon(self.warning_icon)
        return 0

    # Sum the number of possible characters
    possible_characters = 0
    if self.chk_numeric.isChecked(): # 0-9
        possible_characters += 10
    if self.chk_upper.isChecked(): # A-Z
        possible_characters += 26
    if self.chk_lower.isChecked(): # a-z
        possible_characters += 26
    if self.chk_special.isChecked(): # !@#$%^&*()_+-=
        possible_characters += 32
    if password.isspace() == True: # space
        possible_characters += 1

    # Calculate the entropy using the formula log2(possible_characters^password_length)
    entropy = log2(possible_characters**len(password))
    return entropy
```

ภาพประกอบที่ 3.11 การคำนวณ Bits of entropy

ฟังก์ชัน calculate_entropy ถูกใช้ในการคำนวณค่า entropy ของรหัสผ่านที่ผู้ใช้งานป้อนเข้ามาจะถูกตรวจสอบองค์ประกอบของรหัสผ่านโดยอ้างอิงตามค่ามาตรฐาน NIST Special Publication 800-63B และนำองค์ประกอบที่พบในรหัสผ่านไปคำนวณโดยใช้สูตร entropy = $\log_2$(possible_characters**len(password) ซึ่ง possible_characters คือ ค่าความเป็นไปได้ขององค์ประกอบรหัสผ่านและ len(password) คือ จำนวนหลักของรหัสผ่านที่ผู้ใช้งานป้อนเข้ามา

```python
import requests
import json

# Request the password list from NordPass
headers = {
    'sec-ch-ua': '"Not.A/Brand";v="8", "Chromium";v="114", "Google Chrome";v="114"',
    'Referer': 'https://nordpass.com/most-common-passwords-list/',
    'sec-ch-ua-mobile': '?0',
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0
    'sec-ch-ua-platform': '"Windows"',
}

response = requests.get('https://nordpass.com/json-data/top-worst-passwords/findings/all.json', headers=headers)

print(response.text)

# Convert the response to a JSON object
json_object = json.dumps(response.json())

# Write the JSON object to a file
with open("./data/nordpass_wordlist.json", "w") as outfile:
    outfile.write(json_object)
```

ภาพประกอบที่ 3.12 การเปรียบเทียบกับผ่านกับ NordPass

ใช้งานโมดูล requests เพื่อส่งคำร้องขอไปยัง https://nordpass.com/json-data/top-worst-passwords/findings/all.json เพื่อรับข้อมูลรายการรหัสผ่านที่ไม่ควรใช้จาก NordPass ผ่านการใช้ Request Headers ที่ระบุไว้ในตัวแปร headers ซึ่งมีทั้งหมด 200 รหัสผ่าน 2020-2022 เพื่อใช้เป็นฐานข้อมูลในการนำมาทำการปรียบเทียบกับรหัสผ่านว่าตรงกับรหัสผ่านที่ผู้งานในการนำมาทำการ

เปรียบเทียบกับรหัสผ่านว่าตรงกับรหัสผ่านที่ผู้งานป้อนเข้ามาหรือไม่หากพบว่าตรงกันจะมีข้อความแจ้งเตือน

```python
    def time_to_Crack(self, password):
        try:
            if password == '':
                self.chk_length.setIcon(self.warning_icon)
                self.chk_numeric.setIcon(self.warning_icon)
                self.chk_upper.setIcon(self.warning_icon)
                self.chk_lower.setIcon(self.warning_icon)
                self.chk_special.setIcon(self.warning_icon)
                return "0 seconds"

            possible_characters = 0
            if self.chk_numeric.isChecked(): # 0-9
                possible_characters += 10
            if self.chk_upper.isChecked(): # A-Z
                possible_characters += 26
            if self.chk_lower.isChecked(): # a-z
                possible_characters += 26
            if self.chk_special.isChecked(): # !@#$%^&*()_+-=
                possible_characters += 32
            if password.isspace() == True: # space
                possible_characters += 1

            combinations = possible_characters ** len(password)
            KPS_2020 = 17042497.3 # 17 Million

            seconds = combinations / KPS_2020
            seconds = f'{seconds:.0f}'
            seconds = int(seconds)
```

ภาพประกอบที่ 3.13 การคำนวณ Estimated time to crack

ฟังก์ชัน Time_to_crack ใช้ในการคำนวณระยะเวลาที่คาดว่าผู้โจมตีจะใช้ในการถอดรหัสผ่านสูตรที่ใช้คือ combinations=(possible_characters**len(password) ซึ่ง possible_characters คือ ค่าความเป็นไปได้ขององค์ประกอบรหัสผ่านและ len(password) คือ จำนวนหลักของรหัสผ่านที่ผู้ใช้งานป้อนเข้ามา จากนั้นนำ combinations มาหารค่าความแรงของเครื่องคอมพิวเตอร์ที่จะใช้ในการ Brute-force จะได้ค่า Estimated time to crack เป็นหน่วยวินาที

```python
    def run_hashcat(self, mode, wordlist, hash, password):
        if mode == "0": # Straight forward
            command = f"D:\Hashcat\hashcat-6.2.5\hashcat.exe -d 2 -m 0 -a {mode} {hash} D:\ISAN Security Gizmo B
        elif mode == "1": # Combination
            command = f"D:\Hashcat\hashcat-6.2.5\hashcat.exe -d 2 -m 0 -a {mode} {hash} {wordlist} {wordlist} |
        elif mode == "6": # Skipping 1
            command = f"D:\Hashcat\hashcat-6.2.5\hashcat.exe -d 2 -m 0 -a {mode} {hash} {wordlist} | findstr ':{
        elif mode == "7": # Skipping 2
            command = f"D:\Hashcat\hashcat-6.2.5\hashcat.exe -d 2 -m 0 -a {mode} {hash} {wordlist} | findstr ':{
        print(command)
        if command is None:
            return "No password found"
        process = subprocess.Popen(
            command,
            shell=True,
            stdout=subprocess.PIPE,
            stderr=subprocess.PIPE,
            text=True
        )

        for line in process.stdout:
            if line is None:
                return "No password found"
            self.update_text.emit(line)
        process.communicate()
        self.finished.emit()
```

ภาพประกอบที่ 3.14 การเรียกใช้งาน Hashcat

ฟังก์ชัน run_hashcat ใช้ในการรัน Hashcat เพื่อทดสอบการโจมตีด้วยเทคนิค Dictionary attack โดย option ที่ใช้ประกอบด้วย ดังนี้

1) -d 2 คือ เลือก device ในการโจมตี

2) -m 0 คือ รหัสผ่านที่ถูกแฮชด้วยฟังก์ชัน MD5

3) - a คือ โหมดที่ใช้ในการโจมตีซึ่งจะแบ่งตามที่ผู้ใช้งานเลือกโดยมีตัวเลือก ดังนี้

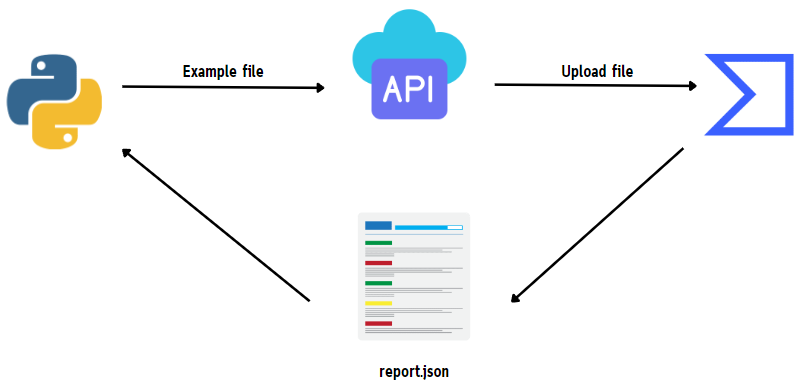
- 0 คือ การโจมตีแบบ Straightforward dictionary attack
- 1 คือ การโจมตีแบบ Combinator attack
- 6,7 คือ การโจมตีแบบ Hybrid attack

3.4 เครื่องมือ Malware Scanning

```
135     def FileScan(self):
136         print("File Scan")
137         input = self.lineEdit_malware.text()
138         url = MalwareScanning.api_file_scan
139         files = { "file": open(input, "rb") }
140         headers = {
141             "accept": "application/json",
142             "x-apikey": MalwareScanning.api_vt_key,
143         }
144 
145         response = requests.post(url, files=files, headers=headers)
```

ภาพประกอบที่ 3.15 การเรียกใช้งาน VirusTotal API-File Scan

ฟังก์ชัน FileScan ใช้สำหรับสแกนไฟล์ที่ผู้ใช้งานเพิ่มไฟล์เข้ามาว่ามีมัลแวร์หรือไม่โดยสร้าง dictionary ที่ประกอบด้วย key "file" และค่าคือไฟล์ที่ต้องการสแกนโดยใช้ open(input, "rb") เพื่อเปิดไฟล์ในโหมดอ่านแบบ binary กำหนด headers สำหรับคำขอโดยระบุ "accept" เป็น "application/json" และ "x-apikey" เป็นคีย์ API ของ VirusTotal ในตัวแปร headers และส่งคำขอ POST ไปยัง VirusTotal API โดยใช้ requests.post() ส่งไฟล์และ headers จากนั้น VirusTotal API จะทำการสแกนไฟล์ที่ถูกส่งไปและส่งผลลัพธ์กลับมา



ภาพประกอบที่ 3.16 แสดงการไหลของข้อมูลเมื่อเรียกใช้ VirusTotal API-File Scan

```
166         id = response.json()['data']['id']
167         print(id)
168         # File Analyses
169         MalwareScanning.fileAnalyses(self, id)
170 
171     def fileAnalyses(self, id):
172         url = MalwareScanning.api_file_analysis + "/" + id
173 
174         headers = {
175             "accept": "application/json",
176             "x-apikey": MalwareScanning.api_vt_key
177         }
178 
179         response = requests.get(url, headers=headers)
```

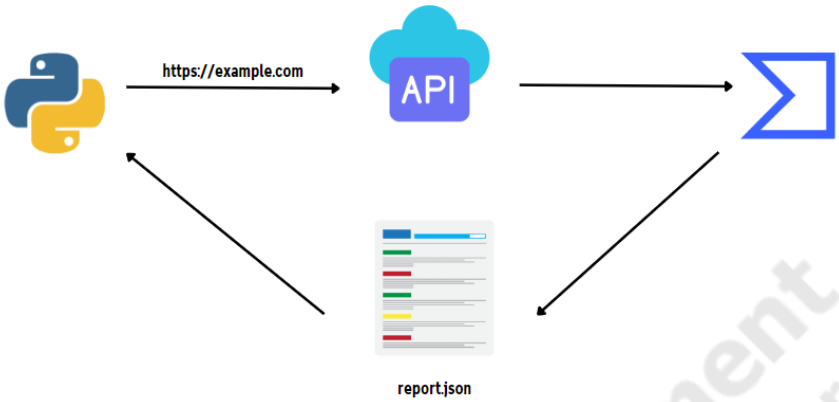
ภาพประกอบที่ 3.17 การวิเคราห์ไฟล์ของ VirusTotal

ฟังก์ชัน fileAnalyses ใช้สำหรับการขอข้อมูลการวิเคราะห์ไฟล์จาก VirusTotal API โดยใช้ id ที่คืนมาจากการสแกนไฟล์กำหนด headers สำหรับคำขอโดยระบุ "accept" เป็น "application/json" และ "x-apikey" เป็นคีย์ API ของ VirusTotal ในตัวแปร headers ส่งคำขอ GET ไปยัง VirusTotal API โดยใช้ requests.get() และส่ง headers ไปด้วยและ VirusTotal API จะส่งผลลัพธ์การวิเคราะห์ไฟล์กลับมา

```
215     def URLScan(self):
216         print("URL Scan")
217         input = self.lineEdit_malware.text()
218         url = MalwareScanning.api_url_scan
219 
220         payload = { "url": input }
221         headers = {
222             "accept": "application/json",
223             "x-apikey": MalwareScanning.api_vt_key,
224             "content-type": "application/x-www-form-urlencoded"
225         }
226 
227         response = requests.post(url, data=payload, headers=headers)
```

ภาพประกอบที่ 3.18 การเรียกใช้งาน VirusTotal API-URL Scan

ฟังก์ชัน URLScan ใช้สำหรับสแกน URL ของเว็บไซต์ที่ผู้ใช้งานเพิ่มไฟล์เข้ามาว่ามีมัลแวร์หรือไม่ โดยสร้าง payload ที่ประกอบด้วย key "url" และค่าคือ URL ของเว็บไซต์ที่ต้องการสแกนซึ่งใช้ { "url": input } และกำหนด headers สำหรับคำขอโดยระบุ "accept" เป็น"application/json", "x-apikey" เป็นคีย์ API ของ VirusTotal, และ "content-type" เป็น "application/x-www-form-urlencoded" ในตัวแปร headers และส่งคำขอ POST ไปยัง VirusTotal API โดยใช้ requests.post() และส่ง payload และ headers ด้วย VirusTotal API จะทำการตรวจสอบ URL ของเว็บไซต์ที่ถูกส่งไปและส่งผลลัพธ์กลับมา



ภาพประกอบที่ 3.19 แสดงการไหลของข้อมูลเมื่อเรียกใช้ VirusTotal API-URL Scan

```
248         id = response.json()['data']['id'].split('-')[1]
249         print(id)
250
251         MalwareScanning.URLReport(self, id)
252
253     def URLReport(self, id):
254         url = MalwareScanning.api_url_scan + "/" + id
255         print(url)
256         headers = {
257             "accept": "application/json",
258             "x-apikey": MalwareScanning.api_vt_key
259         }
260         response = requests.get(url, headers=headers)
```

ภาพประกอบที่ 3.20 การวิเคราะห์ URL ของ VirusTotal

ฟังก์ชัน URLReport ใช้สำหรับการขอข้อมูลการวิเคราะห์ไฟล์จาก VirusTotal API โดยใช้ id โดย id จาก response ถูกดึงมาจาก response.json()['data']['id'] และใช้ split('-')[1] เพื่อเอาเฉพาะส่วนที่ต้องการ 2จากนั้นกำหนด headers สำหรับคำขอโดยระบุ "accept" เป็น "application/json" และ "x-apikey" เป็นคีย์ API ของ VirusTotal ในตัวแปร headers และส่งคำขอ GET ไปยัง VirusTotal API โดยใช้ requests.get() และส่ง headers ด้วย. VirusTotal API จะส่งผลลัพธ์รายงานเพิ่มเติมเกี่ยวกับ URL กลับมา

```
410     def createReport(self):
411         print("Create Report Malware")
412
413         pdf_file_path = "./data/Reports/Malware_Scanning_Report.pdf"
414         self.btn_file_email_malware.setText(pdf_file_path.split('/')[-1])
415         target = self.lineEdit_malware.text()
416         malicious_number = self.label_maliciousResult.text()
417         suspicious_number = self.label_suspiciousResult.text()
418         undetected_number = self.label_undetectedResult.text()
419
420         # Create a PDF canvas
421         c = canvas.Canvas(pdf_file_path, pagesize=A4)
```

ภาพประกอบที่ 3.21 การสร้างไฟล์ PDF ของเครื่องมือ Malware Scanning

ฟังก์ชัน createReport ใช้สร้างรายงานประเภทไฟล์ PDF สำหรับแสดงผลบนเครื่องมือ Malware Scanning และส่งไปยังอีเมลในกรณีที่ผู้ใช้งานต้องการนำผลลัพธ์ไปวิเคราะห์ต่อ

3.5 เครื่องมือ Message Digest Generator

```
149     def checkFile_Text(self):
150         if os.path.exists(self.lineEdit_MSdigest.text()) == True: # check if file exists
151             print("File")
152             MessageDigest.state_detect = 1
153             self.btn_md5.clicked.connect(lambda: MessageDigest.fileExtract(self, "md5", MessageDigest.getPath(sel
154             self.btn_sha1.clicked.connect(lambda: MessageDigest.fileExtract(self, "sha1", MessageDigest.getPath(se
155             self.dropdown_sha2.activated.connect(lambda: MessageDigest.fileExtract(self, "sha2_" + self.dropdown_
156             self.dropdown_sha3.activated.connect(lambda: MessageDigest.fileExtract(self, "sha3_" + self.dropdown_
157         else:
158             MessageDigest.state_detect = 0
159             print("Plaintext")
160             self.btn_md5.clicked.connect(lambda: MessageDigest.hash(self, "md5"))
161             self.btn_sha1.clicked.connect(lambda: MessageDigest.hash(self, "sha1"))
162             self.dropdown_sha2.activated.connect(lambda: MessageDigest.getdropdown_sha2(self))
163             self.dropdown_sha3.activated.connect(lambda: MessageDigest.getdropdown_sha3(self))
```

ภาพประกอบที่ 3.22 การตรวจสอบ Input ของเครื่องมือ Message Digest Generator

ฟังก์ชัน checkFile_Text ใช้สำหรับตรวจสอบข้อมูลที่ผู้ใช้ป้อนเข้ามาว่าไฟล์ที่ถูกระบุในช่องข้อมูล self.lineEdit_MSdigest.text() มีอยู่หรือไม่ โดยใช้ os.path.exists() และเก็บผลการตรวจสอบในตัวแปร MessageDigest.state_detect ถ้าไฟล์มีอยู่ MessageDigest.state_detect ถูกตั้งค่าเป็น 1 และถ้าไม่มี MessageDigest.state_detect ถูกตั้งค่าเป็น 0 และดำเนินการแฮชตามฟังก์ชันที่ผู้ใช้งานเลือกต่อไป

```
181     def hash(self, type):
182         #print(self.dropdown_sha2.currentText())
183         if type == "md5":
184             self.lineEdit_outputTextMSDigest.setText(MessageDigest.md5(self, self.lineEdit_MSdigest.text())) \
185                 if self.lineEdit_MSdigest.text() != '' else self.lineEdit_outputTextMSDigest.setText('')
186             self.algorithm = 'MD5'
187 
188         elif type == "sha1":
189             self.lineEdit_outputTextMSDigest.setText(MessageDigest.sha1(self, self.lineEdit_MSdigest.text())) \
190                 if self.lineEdit_MSdigest.text() != '' else self.lineEdit_outputTextMSDigest.setText('')
191             self.algorithm = 'SHA-1'
```

ภาพประกอบที่ 3.23 การเรียกใช้งานแฮชฟังก์ชัน

ฟังก์ชัน hash ใช้สำหรับคำนวณค่าแฮชของข้อมูลที่ผู้ใช้งานป้อนเข้ามาและเรียกใช้งานฟังก์ชันตามประเภทการแฮชที่ผู้ใช้งานเลือก เช่น หากเลือกการแฮชประเภท MD5 จะเรียกฟังก์ชัน md5 ของคลาส MessageDigest เพื่อคำนวณค่าแฮช MD5 ของข้อมูลที่ถูกระบุใน self.lineEdit_MSdigest.text() และแสดงค่าแฮชนั้นใน self.lineEdit_outputTextMSDigest

```python
def fileHash(self, type, path):
    MessageDigest.LoadAPIKey(self)
    text_type = type
    if "_" in type:
        text_type = text_type.replace("_", " ")
    self.label_type.setText(text_type.upper())

    if type == "md5":
        init_hash = hashlib.md5()
        file = path
        BLOCK_SIZE = 65536
        with open(file, 'rb') as f:
            fb = f.read(BLOCK_SIZE)
            while len(fb) > 0:
                init_hash.update(fb)
                fb = f.read(BLOCK_SIZE)
        file_hashed =  init_hash.hexdigest()
        print (f"This is file hash {type}: {file_hashed}")
        self.lineEdit_outputTextMSDigest.setText(f'{file_hashed}')
```

ภาพประกอบที่ 3.24 การแฮชไฟล์

ฟังก์ชัน fileHash ใช้สำหรับคำนวณค่าแฮชของไฟล์ที่ผู้ใช้งานป้อนเข้ามาตาที่ระบุใน path โดยจะนำไฟล์ที่ถูกป้อนเข้ามาแบ่งการอ่านไฟล์เป็นบล็อกขนาด 65536 ไบต์ และใช้ข้อมูลนั้นในการอัปเดตคำนวณค่าตามแฮชฟังก์ชันที่ผู้ใช้งานเลือกโดยการแฮชไฟล์จะทำการเรียก Python library เช่นเดียวกับการแฮชข้อความ

```python
def processLineKey(self):
    if self.lineEdit_MSdigest.text() == '':
        print("Data to send Empty")
        self.lineEdit_outputTextMSDigest.setStyleSheet("border: 1px solid red;")
        self.lineEdit_outputTextMSDigest.setPlaceholderText("Empty")
        return
    type = self.label_type.text()
    message = self.lineEdit_outputTextMSDigest.text() + "\nHash Algorithms: " + type
    token = self.lineEdit_tokenMSDigest.text()
    try:
        if token != '':
            getQR = "./data/MessageDigest-QRCode.png"
            url = "https://notify-api.line.me/api/notify"

            headers = {"Authorization": "Bearer " + token}
            payload = {"message": message}

            with open(getQR, "rb") as image_file:
                files = {"imageFile": image_file}
                response = requests.post(url, headers=headers, params=payload, files=files)
```

ภาพประกอบที่ 3.25 การตรวจสอบ Line Notify

ฟังก์ชัน processLineKey ใช้สำหรับส่งข้อความและรูปภาพ QR Code ไปยัง Line Notify โดยใช้ Token ที่ระบุใน self.lineEdit_tokenMSDigest และข้อมูลเกี่ยวกับค่าแฮชและประเภทแฮชที่ถูกคำนวณซึ่งผู้ใช้งานจะต้องมี Line API Token เพื่อใช้ในการส่งข้อมูล

```python
def qrCodeGenerator(self, hash):
    if self.lineEdit_outputTextMSDigest == '': # check current text output is empty
        print("Error: QR-Code is Not Generated")
        self.lineEdit_outputTextMSDigest.setStyleSheet("border: 1px solid red;")
        self.lineEdit_outputTextMSDigest.setPlaceholderText("Empty")
        return
    # Generate QR Code
    qr = qrcode.QRCode(
        version=1,
        box_size=10,
        border=5
    )
    qr.add_data(hash)
    qr.make(fit=True)
    img = qr.make_image(fill_color="black", back_color="white")
    img.save("./data/MessageDigest-QRCode.png")
    print("QR Code Generated")
    return img
```

ภาพประกอบที่ 3.26 การสร้าง QR Code

ฟังก์ชัน qrCodeGenerator ใช้สำหรับสร้างรูปภาพ QR Code จากค่าแฮชที่ถูกคำนวณและแสดงใน self.lineEdit_outputTextMSDigest สร้างรูปภาพ QR Code จากข้อมูลดังกล่าวโดยใช้ library qrcode เมื่อสแกน QR Code จะได้ค่าแฮชในรูปแบบข้อความ

3.6 เครื่องมือ Vulnerability Scanning

```python
    target = VulnerabilityScanning.validate_input(self, target)
    # Select Type of Scan and set the command
    if type == "Quick Scan":
        option = "-T4 -F" # -T4 = Aggressive, -F = Fast Scan

    elif type == "Stealth Scan":
        option = "-sS" # -sS = Stealth Scan

    elif type == "Aggressive Scan":
        option = "-T4 -A" # -T4 = Aggressive, -A = Agressive Scan Os Detection, Version Detection, Script Sc

    elif type == "Vulners.NSE Script":
        option = "-sV --script=vulners"

    else:
        option = ""

    # Set the command in the lineEdit
    self.lineEdit_commandvulner.setText("nmap " + option + " " + target)
    self.lineEdit_commandvulner.setStyleSheet("border: 2px solid green;")
    # Start Scan
    VulnerabilityScanning.start_scan(self, option, target)

def start_scan(self, option, target):
    self.textEdit_ResultScan.clear() # Clear the textEdit
    # Start Thread
    VulnerabilityScanning.scan_thread = threading.Thread(target=VulnerabilityScanning.run_scan, args=(self,
    VulnerabilityScanning.scan_thread.start()
```

ภาพประกอบที่ 3.27 การกำหนดค่า option ของเครื่องมือ Vulnerability Scanning

ฟังก์ชันนี้ใช้สำหรับกำหนดค่า option ขึ้นอยู่กับการเลือกของผู้ใช้งานฟังก์ชันเริ่มต้นด้วยการใช้ VulnerabilityScanning.validate_input เพื่อตรวจสอบความถูกต้องของเป้าหมายที่ระบุใน Target

และตรวจสอบความถูกต้องว่าเป็น IP address หรือ โดเมนเนมซึ่งในการสแกนซึ่งแบ่งเป็น 4 โหมด ได้แก่ Quick scan, Stealth scan, Aggressive scan, Vulners.NSE script

3.6.1 Quick scan คือการสแกนเครือข่ายหรือเครื่องคอมพิวเตอร์โดยใช้การตั้งค่าที่ชิดเจาะจงเพื่อทำให้การสแกนเสร็จสิ้นเร็วขึ้น โดยทั่วไปแล้วมักจะเน้นการสแกนเพียงบางพอร์ตที่สำคัญเท่านั้นเพื่อลดเวลาการสแกนและการรบกวนในเครือข่ายปลายทาง สแกนแบบ Quick scan มักใช้สำหรับวัตถุประสงค์การตรวจสอบพอร์ตที่สำคัญหรือการทดสอบความเร็วของการเชื่อมต่อในเครือข่ายการสแกนแบบ Quick scanใช้คำสั่ง nmap -F -T4 <target> จะช่วยให้การสแกนเสร็จสิ้นอย่างรวดเร็วและเหมาะสำหรับการตรวจสอบพอร์ตที่สำคัญเท่านั้นหรือการทดสอบความเร็วของการเชื่อมต่อในเครือข่าย, และอาจลดความรบกวนในเครือข่ายปลายทางเมื่อเป็นจำเป็น

3.6.2 Stealth Scan เป็นการสแกนเครื่องคอมพิวเตอร์แบบไม่เปิดเผยตัวเอง โดยการสแกนนี้จะทำการสแกนพอร์ตบนเครื่องคอมพิวเตอร์โดยไม่ส่ง packet ชนิด SYN ไปยังเครื่องที่ต้องการสแกน ซึ่งวิธีการเชื่อมต่อแบบนี้จะไม่เปิดเผยตัวเองต่อเครื่องคอมพิวเตอร์ที่เรากำลังสแกนอยู่ การใช้งาน Nmap Stealth Scan สามารถทำได้โดยใช้คำสั่ง nmap -sS <target> โดยที่ <target> คือ IP address หรือชื่อโดเมนของเครื่องคอมพิวเตอร์ที่ต้องการสแกน ซึ่งสามารถช่วยป้องกันการโจมตีจากผู้ไม่หวังดีได้ในบางกรณี

3.6.3 Aggressive scan ใน Nmap เป็นการสแกนพอร์ตโดยใช้เทคนิคการสแกนที่เร็วและเจาะจงมากขึ้น โดยการสแกนแบบ Aggressive จะเป็นการสแกนทุกรูปแบบที่เป็นไปได้และใช้เทคนิคการสแกนที่หลากหลาย เพื่อหาข้อมูลเพิ่มเติมเกี่ยวกับระบบเครือข่ายที่กำลังถูกสแกน การสแกนแบบ Aggressive อาจทำให้ระบบเครือข่ายที่ถูกสแกนเกิดความเสี่ยงต่อการโจมตีได้ง่ายขึ้น โดนใช้คำสั่ง nmap -A การใช้งานคำสั่งดังกล่าวจะทำการสแกนพอร์ตของเครื่องเป้าหมายโดยใช้เทคนิคการสแกนที่หลากหลาย เช่น การสแกนด้วย OS detection, version detection, script scanning, traceroute และอื่นๆ ซึ่งจะช่วยให้ผู้ใช้ได้ข้อมูลเพิ่มเติมเกี่ยวกับระบบเครือข่ายที่ถูกสแกนได้มากขึ้น

3.6.4 Vulners.NSE เป็น script สำหรับเครื่องมือสแกนระบบเครือข่ายโดย Nmap ช่วยให้ผู้ใช้สามารถสแกนหาช่องโหว่ในระบบเป้าหมายได้ โดยการตรวจสอบกับฐานข้อมูลช่องโหว่ Vulners.com และเปรียบเทียบผลลัพธ์กับพอร์ตและบริการที่เปิดใช้งานบนระบบเป้าหมาย script นี้สามารถใช้งานได้กับ Nmap ทุกเวอร์ชัน และสามารถทำงานกับระบบปฏิบัติการที่แตกต่างกันได้

```python
def run_scan(self, option, target):
    option = option.split()
    #print(option)
    # Start Progress Bar 0 to 99 until scan is complete
    self.progressBar_vulnerScan.setVisible(True)
    for i in range(100):
            self.progressBar_vulnerScan.setValue(i + random.randint(30, 40))
            threading.Thread(target=time.sleep(0.1)).start()
    try:
        #ใช้ *option เพื่อแยก args ทำให้สามารถใช้ใน subprocess.check_output ได้อย่างยืดหยุ่น

        result = subprocess.check_output(['nmap', *option, target], text=True)
        VulnerabilityScanning.update_result_text(self, result)
    except subprocess.CalledProcessError as e:
        print(f"Scan Error: {e.stderr}")
        self.progressBar_vulnerScan.setValue(100)
        VulnerabilityScanning.update_result_text(self, f"Scan Error: {e.stderr}")
```

ภาพประกอบที่ 3.28 การเรียกใช้งานคำสั่ง Nmap

ฟังก์ชัน run_scan ใช้ในการรันคำสั่ง Nmap ฟังก์ชันเรียก subprocess ในการสแกนโดยใช้คำสั่ง Nmap โดยใช้ subprocess.check_output โดยส่งคำสั่ง Nmap พร้อมกับ argument *option และเป้าหมาย target เข้าไปผลลัพธ์ของการสแกนถูกจัดเก็บในตัวแปร result

```python
def createReport(self):
    # Create a PDF canvas
    current_time = datetime.now()
    file_name = "./data/Reports/Vulner_Scanning_Report.pdf"
    self.btn_file_email_vulner.setText(file_name.split('/')[-1])
    target = self.lineEdit_vulner.text()
    option_scan = self.dropdown_typeScan.currentText()
    if option_scan == "Type Scan":
        option_scan = "Default Scan"
    command = self.lineEdit_commandvulner.text()

    c = canvas.Canvas(file_name, pagesize=A4)
```

ภาพประกอบที่ 3.29 การสร้างไฟล์ PDF ของเครื่องมือ Malware Scanning

ฟังก์ชัน createReport ใช้สร้างรายงานประเภทไฟล์ PDF สำหรับแสดงผลบนเครื่องมือ Vulnerability Scanning และส่งไปยังอีเมลในกรณีที่ผู้ใช้งานต้องการนำผลลัพธ์ไปวิเคราะห์ต่อ

3.7 เครื่องมือ HTTPS Testing

```python
def validate_input(self, target):

    has_special = any(char in "<>!@#$%^&*()_+-=?&" for char in target)
    if has_special:
        print("Special Characters Detected")
        self.lineEdit_https.setPlaceholderText("Invalid Input")
        # wait 2 seconds
        self.lineEdit_https.setStyleSheet("border: 2px solid Red;")
        self.lineEdit_https.setText('')
        return False
    else:
        self.lineEdit_https.setStyleSheet("border: 2px solid green;")
        return target.lower()
```

ภาพประกอบที่ 3.30 การตรวจสอบ Input ของเครื่องมือ Vulnerability Scanning

ฟังก์ชัน validate_input ในการตรวจสอบความถูกต้องของข้อมูลที่ผู้ใช้งานป้อนเข้ามาว่ามีอักขระพิเศษหรือไม่หากมีตัวอักษรพิเศษปรากฏ ฟังก์ชันจะแสดงข้อความ "Special Characters Detected" และทำการตั้งค่า placeholder ของช่องข้อมูลนำเข้า lineEdit_https เป็น "Invalid Input" แต่ถ้าหากไม่มีตัวอักษรพิเศษฟังก์ชันจะเปลี่ยนสีกรอบของช่องข้อมูลนำเข้าเป็นสีเขียวและส่งค่า target ที่ถูกแปลงเป็นตัวอักษรเล็กออกมา

```
33      def createReport(self):
34          # Create a PDF canvas
35          current_time = datetime.now()
36          file_name = f"./data/Reports/HTTPS_Testing_Report.pdf"
37          self.btn_file_email_https.setText(file_name.split('/')[-1])
38          target = self.lineEdit_https.text()
39
40          c = canvas.Canvas(file_name, pagesize=A4)
```

ภาพประกอบที่ 3.31 การสร้างไฟล์ PDF ของเครื่องมือ HTTPS Testing

ฟังก์ชัน createReport ใช้สร้างรายงานประเภทไฟล์ PDF สำหรับแสดงผลบนเครื่องมือ HTTPS Testing และส่งไปยังอีเมลในกรณีที่ผู้ใช้งานต้องการนำผลลัพธ์ไปวิเคราะห์ต่อ

```
┌──(kali㉿kali)-[~/testssl.sh]
└─$ ./testssl.sh isanmsu.com

###########################################################
    testssl.sh       3.2rc2 from https://testssl.sh/dev/
    (8f295cb 2023-10-07 15:09:11)

      This program is free software. Distribution and
             modification under GPLv2 permitted.
      USAGE w/o ANY WARRANTY. USE IT AT YOUR OWN RISK!

       Please file bugs @ https://testssl.sh/bugs/

###########################################################

 Using "OpenSSL 1.0.2-bad (1.0.2k-dev)" [~183 ciphers]
 on kali:./bin/openssl.Linux.x86_64
 (built: "Sep  1 14:03:44 2022", platform: "linux-x86_64")


 Start 2023-10-19 05:10:27        -->> 203.159.92.102:443 (isanmsu.com) <<--

 rDNS (203.159.92.102):  --
 Service detected:       HTTP


 Testing protocols via sockets except NPN+ALPN

 SSLv2      not offered (OK)
 SSLv3      not offered (OK)
 TLS 1      not offered
 TLS 1.1    not offered
 TLS 1.2    offered (OK)
 TLS 1.3    offered (OK): final
 NPN/SPDY   not offered
 ALPN/HTTP2 h2, http/1.1 (offered)
```

ภาพประกอบที่ 3.32 การทดสอบ Testssl.sh

ทดสอบการเรียกใช้งาน Testssl.sh ใช้ในการทดสอบความปลอดภัยของเซิร์ฟเวอร์และเครือข่ายในสภาพการทำงานจริงโดยตรวจสอบการกำหนดค่าความปลอดภัยของเว็บเซิร์ฟเวอร์ การเชื่อมต่อที่ปลอดภัยที่ใช้ HTTPS และปัญหาความปลอดภัยอื่น ๆ

3.8 การส่งรายงานไปยังอีเมล

```python
class SendEmail():

    sender_email = 'secgizmobox@gmail.com'
    sender_password = 'eevl vyqp heju suol'

    def sending(self, receiver_email, subject, body, attachment_path):
        name = str(attachment_path.split('/')[-1])
        # Create a multipart message
        message = MIMEMultipart()
        message['sender_email'] = self.sender_email
        message['receiver_email'] = receiver_email
        message['subject'] = subject

        # Add body to email
        message.attach(MIMEText(body, 'plain'))

        # Open the file in bynary
        with open(attachment_path, 'rb') as attachment:
            # Add file as application/octet-stream
            # Email client can usually download this automatically as attachment
            part = MIMEApplication(attachment.read(), Name=name)
            part['Content-Disposition'] = f'attachment; filename="{name}"'
            message.attach(part)

        # Log in to server using secure context and send email
        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as server:
            server.login(self.sender_email, self.sender_password)
            server.sendmail(self.sender_email, receiver_email, message.as_string())
            server.quit()
```

ภาพประกอบที่ 3.33 การส่งรายงานไปยังอีเมล

คลาส SendEmail ใช้ในการส่งอีเมลผ่านการเชื่อมต่อกับเซิร์ฟเวอร์ SMTP ของ Gmail โดยใช้ library smtplib และ library email.mime สำหรับการสร้างและส่งอีเมลในฟังก์ชัน sending, สร้างอีเมลในรูปแบบ MIMEMultipart และกำหนด header อีเมล เช่น ผู้ส่ง (sender) ผู้รับ (receiver) และหัวข้อ (subject) นอกจากนี้ยังทำการเข้าสู่เซิร์ฟเวอร์ SMTP ของ Gmail โดยใช้โปรโตคอล SMTP_SSL ที่ทำให้อัตโนมัติเชื่อมต่อผ่านพอร์ต 465 และใช้การเข้าสู่ระบบด้วยอีเมลและรหัสผ่านของผู้ส่ง