Lab 10

More on Web Penetration, Web Crawlers, GPG and Ransomware

1. OWASP Zed Attack Proxy (ZAP)
   Let us install OWASP ZAP on Kali: `sudo apt install zaproxy`

   Turn on your Metasploitable VM and type `zaproxy` in the Kali terminal. OWASP ZAP is a scanning/exploit tool for web penetration. Once, the OWASP ZAP is up, select "Automated Scan" on the right panel and enter http://Meta2_IP/mutillidae on the "URL to attack" and hit the "Attack" button.

   It will take long to complete the full scan. You can press the stop button and click "Alerts" tab to see the vulnerabilities found. Click one of them to view or execute it on the web browser by right-clicking it. You can see the number of vulnerability of the website.

2. Simple Web Crawler Program for Searching Subdomains

   A while ago, we learned about subdomains. Recall that subdomains are used to represent servers or websites which belong to a particular domain. For example, eng.uow.edu.au, maps.uow.edu.au are all subdomains of the domain, uow.edu.au. The problem is that those subdomains are not secured enough as the main domain.

   We can find subdomains using various web-based information gathering tools, but we are going to write a web crawling program using Python to search for subdomains of a given domain.

   A convenient way to do this is to use the package `requests` in Python. A skeleton code to start is as follows.

```
import requests

domain = "uow.edu.au"
url = "https://"+domain

response = requests.get(url)
print(response)
```

   What is the output? Change `domain` to `"abc.uow.edu.au"`. What do you get now? Think about how you can write a function so that it passes when a url for non-existent subdomain is provided as input:

```
def check_subdomain(url)
   try:
        return requests.get(url)
   except requests.exceptions.ConnectionError:
```

```
            pass
```

Now, the remaining part is to provide possible urls for subdomain from a dictionary file. Download "subdomains.txt" from the subject Moodle site. Then, make the above program open this file and read the items in the file line by line. As there is a new line character "/n" after each (domain) word, we need to use `strip()` to remove that new line character. Also, make a Python function that takes a url as input and return `response`. In the main body of the program, you should have an if-statement to check whether this function returns something or nothing (in case a requested subdomain does not exist.) Run your program. (It will take a long time to try every domain names in the file. You can quit the program if it takes too long.)

3. We are going to use GPG on Kali and Ubuntu. Note that they are installed on those OS by default.

- Symmetric Encryption Using GPG

  On your Kali terminal, issue `gpg -c test.txt` to encrypt your file `test.txt`. To save your ciphertext in ascii format use: `gpg -c --armor test.txt` (Note1: the default gpg format is binary, which is not human-readable. Note2: "AES" is a default algorithm. Note3: `-a` is the same as `--armor`)

  To decrypt, issue `gpg test.txt.gpg`.

  Additionally, to create a hash value (digest) for any file using a SHA256 hash function, run `sha256sum` text.txt.

- Asymmetric Encryption (Public Key Encryption) Using GPG

  On your Kali terminal, issue `gpg --gen-key` to generate your public (and private) key.

  If prompted, enter your correct email address for your `uid`. The key generation may take some time as the default key size is 2048 bits.

  After you have generated your key, run `gpg --list-keys` to check your public key, fingerprint and `uid`. Note that your `uid` is your email address.

  Export your key by running `gpg -a --export uid > mypubkey.asc` (You can give any name you want.) After exporting the public key, send your key (`mypubkey.asc`) to someone (yourself in this lab task) via email.

  Now, turn on your Ubuntu VM and log in.

Imagine that you are your friend. Open your email and download the public key you sent via email. Import the key you received by issuing the following command on terminal: `gpg --import mypubkey.asc`.

Now create any text file for your plaintext, message.txt.

To use asymmetric (public-key) encryption, run

```
gpg –a –o ciphertext.asc -e -r uid message.txt
```

(Here, the output will be named as "`cipehrtext.asc`", `-e` is an option for encryption.) `Send the ciphertext.asc` to "you" on Kali via email.

On Kali, you download the `ciphertext.asc` from the email you received.

To decrypt, run `gpg -d ciphertext.asc` from the directory where `ciphertext.asc` is located. (Here, `-d` is an option for encryption.)

We may not use GPG every day, but it would be beneficial if we know how to use its basic functionalities.