Kamil Najdek, 259011

Prowadzący: Mgr inż. Jacek Herold

Skrypt wykrywający złośliwą modyfikację strony WWW

25 stycznia 2023

Skrypt wykrywający złośliwą modyfikację strony WWW

Ten program jest przeznaczony do monitorowania zmian na stronie internetowej podanej w parametrze funkcji. Program pobiera aktualną wersję strony za pomocą biblioteki *requests* i przetwarza ją za pomocą biblioteki *BeautifulSoup*. Następnie oblicza skrót (hash) zawartości HTML strony oraz zasobów (obrazów, skryptów, plików CSS itp.) i zapisuje je w pliku "website_hashes.txt". Po pewnym okresie czasu program ponownie pobiera aktualną wersję strony i oblicza skróty dla nowych zasobów. Następnie porównuje nowe skróty z zapisanymi skrótami i jeśli znajdzie jakiekolwiek różnice, wyświetla komunikat *"ALERT: Zasób [...] został zmieniony!"*. Jeśli nie znajdzie żadnych różnic, wyświetla komunikat "Website is safe.".

Użyte biblioteki:

hashlib: Biblioteka ta udostępnia szereg bezpiecznych algorytmów haszujących, w tym SHA-256. Służy do obliczania hasha kodu HTML strony i wszystkich innych zasobów (obrazów, plików CSS i JavaScript) w celu sprawdzenia, czy nie zostały one zmodyfikowane.

requests: Biblioteka ta służy do wykonywania żądań HTTP do strony internetowej i jej zasobów. Skrypt używa tej biblioteki do pobrania kodu HTML strony i wszystkich innych zasobów, aby można było obliczyć hash każdego zasobu.

BeautifulSoup: Biblioteka ta jest używana do analizowania kodu HTML strony internetowej i wyodrębniania adresów URL wszystkich pozostałych zasobów. Skrypt używa tej biblioteki do znalezienia wszystkich znaczników *img*, *link* i *script* w kodzie HTML i wyodrębnienia atrybutów *src* lub *href* z każdego znacznika.

urljoin: Funkcja ta służy do dołączenia bazowego adresu URL do względnego adresu URL w celu uzyskania bezwzględnego adresu URL. Ta funkcja jest używana do łączenia bazowego adresu URL strony z względnym adresem URL każdego zasobu, tak aby każdy zasób mógł być pobrany przy użyciu biblioteki żądań.

Dokumentacja:

current_site = requests.get(url).content
używa biblioteki requests do wysłania żądania *get* do podanego adresu url i pobrania kodu HTML strony internetowej. Atrybut *.content* służy do pobierania zawartości odpowiedzi w postaci bajtów.

soup = BeautifulSoup(current_site, 'html.parser')
wykorzystuje bibliotekę BeautifulSoup do analizowania kodu HTML strony internetowej, który

został pobrany w poprzednim kroku. Funkcja BeautifulSoup przyjmuje dwa argumenty: kod HTML i parser (analizator składniowy) do użycia. W tym przypadku użyty został parser HTML. Wynik jest przechowywany w zmiennej *soup*.

resource_hashes = {}
tworzy pusty słownik o nazwie *resource_hashes*. Skrypt będzie używał tego słownika do przechowywania haseł wszystkich zasobów, w tym kodu HTML, obrazów, plików css, javascript itp. Słownik ten zostanie później wykorzystany do porównania aktualnej wersji strony z późniejszą wersją i wykrycia ewentualnych zmian.

current_hash = hashlib.sha256(current_site).hexdigest()
używa biblioteki hashlib do obliczenia skrótu SHA-256 kodu HTML strony internetowej, który jest przechowywany w zmiennej *current_site*. Funkcja *hashlib.sha256()* tworzy nowy obiekt hashowy sha256, a metoda *.hexdigest()* zwraca wynikowy hash jako ciąg szesnastkowy. Obliczony hash jest przechowywany w zmiennej *current_hash*.

resource_hashes[url] = current_hash
przechowuje obliczony hash w słowniku *resource_hashes* z adresem URL strony internetowej jako kluczem i hashem jako wartością. Jest to pierwszy zasób dodawany do słownika i jest nim kod HTML strony internetowej. Słownik ten zostanie później wykorzystany do porównania aktualnej wersji strony z późniejszą wersją i wykrycia ewentualnych zmian.

soup.find_all(['img', 'link', 'script'])
jest używane do znalezienia wszystkich tagów HTML odpowiadających za zasoby na stronie (w tym przypadku <img>, <link>, <script>)

tag.get('src') or tag.get('href')
jest używane do pobrania adresu URL zasobu z tagu HTML

urljoin(url, src)
jest używane do połączenia podstawowego adresu URL strony z adresem URL zasobu (jeśli zasób jest podlinkowany relatywnie)

resource = requests.get(resource_url).content

pobiera zasób z podanego adresu URL

resource_hash = hashlib.sha256(resource).hexdigest()

jest używane do obliczenia skrótu SHA-256 dla pobranego

```
with open("website_hashes.txt", "w") as f:
    f.write(str(resource_hashes))
```

Linijki te odpowiadają za zapisanie w pliku "*website_hashes.txt*" obliczonych skrótów (hashy) zasobów strony internetowej (HTML, obrazy, pliki CSS, pliki JavaScript itp.) w postaci ciągu znaków. Plik ten będzie następnie używany do porównania z nowymi hashami obliczonymi w późniejszej części programu, w celu wykrycia ewentualnych zmian na stronie.

```
new_site = requests.get(url).content
soup = BeautifulSoup(new_site, 'html.parser')
new_resource_hashes = {}
```

Te linijki kodu odpowiadają za pobranie nowego kodu HTML strony za pomocą modułu requests i przetworzenie go za pomocą modułu BeautifulSoup. Pobrany kod jest następnie przetwarzany przez BeautifulSoup w celu wyszukania wszystkich zasobów (takich jak obrazy, pliki CSS, skrypty JavaScript itp.) na stronie. Hashe tych zasobów są przechowywane w nowym słowniku *new_resource_hashes*.

```
new_hash = hashlib.sha256(new_site).hexdigest()
new_resource_hashes[url] = new_hash
```

Linijki te odpowiadają za pobranie nowej wersji strony i obliczenie jej hasha przy użyciu algorytmu SHA-256. Następnie nowy hash jest przypisywany do słownika "*new_resource_hashes*" pod kluczem "*url*". Hash jest użyty jako sposób na porównanie nowej wersji strony z poprzednią i wykrycie ewentualnych zmian.

```
for tag in soup.find_all(['img', 'link', 'script']):
    src = tag.get('src') or tag.get('href')
    if src:
```

```
        resource_url = urljoin(url, src)
        try:
            resource = requests.get(resource_url).content
            resource_hash = hashlib.sha256(resource).hexdigest()
            new_resource_hashes[resource_url] = resource_hash
        except requests.exceptions.RequestException as e:
            print(e)
            continue
```

Te linijki kodu odpowiadają za pobieranie zawartości wszystkich zasobów (obrazów, skryptów i linków) znajdujących się na stronie internetowej i obliczanie dla nich skrótu SHA256. Następnie skróty te są dodawane do słownika "*new_resource_hashes*" z kluczem będącym adresem URL zasobu. W przypadku wystąpienia wyjątku podczas pobierania zasobów, kod wyświetla komunikat o błędzie.

```
with open("website_hashes.txt", "r") as f:
    stored_hashes = eval(f.read())
for resource_url, new_hash in new_resource_hashes.items():
    if resource_url not in stored_hashes or new_hash != stored_hashes[resource_url]:
        print("ALERT: Zasób {} został zmieniony!".format(resource_url))
    else:
        print("Strona jest bezpieczna.")
```

Te linijki kodu odpowiadają za otwarcie pliku "*website_hashes.txt*" w trybie odczytu, odczytanie z niego zapisanych poprzednio hashy zasobów, a następnie porównanie tych hashy z hashami obecnie pobranych zasobów. Jeśli którykolwiek z hashy jest różny, wyświetlany jest komunikat "*ALERT: Zasób {} został zmieniony!*". Jeśli wszystkie hashe są takie same, wyświetlany jest komunikat "*Strona jest bezpieczna*.".

```
check_website("https://www.wp.pl")
```

Linijki kodu odpowiadają za wywołanie funkcji "*check_website()*" i przekazanie do niej adresu URL strony internetowej "*https://www.wp.pl*" lub każdej innej, jako argumentu.