Безпечне збереження конфіденційної інформації на серверах

https://doi.org/10.31713/MCIT.2021.42

Очко Олександр
кафедра автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій
Національний університет водного господарства та природокористування
Рівне, Україна
ochko_ak20@nuwm.edu.ua

Аврука Ірина
кафедра автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій
Національний університет водного господарства та природокористування
Рівне, Україна
a.s.avruka@nuwm.edu.ua

Abstract — **Виконано реалізацію і розробку алгоритму для забезпечення безпечного збереження конфіденційної інформації на серверах.**

Keywords **— кібербезпека; шифрування даних; хмарні рішення.**

I. Вступ

Хмарні рішення є дуже популярними для збереження та обробки даних. Проте таким рішенням важко довірити збереження своїх паролів та логінів для входу на посторонні сайти або іншу особисту інформацію.

Коли дані передаються на сервер, то вони передаються між сервером та клієнтом у зашифрованому вигляді(за наявності захищеного з'єднання), проте сервер бачить отримані дані у відкритому вигляді і сам вирішує як ними розпоряджатися.

Хмарні рішення – це добре, але ніхто не може гарантувати, що важливі дані не будуть «злиті» в публічний доступ через атаки на хостинг компанію або обліковий запис. Немає гарантій, що адміністрація або працівники, які мають доступ до серверу, не читають інформацію і не можуть її прочитати.

Збереження конфіденційної інформації на локальному комп'ютері у відкритому вигляді – це також небезпечно, бо може бути встановлене вірусне програмне забезпечення або до комп'ютера мають доступ треті особи. Також можлива повна втрата інформації через фізичну поломку.

Тому одним із рішень для збереження даних може бути забезпечення повного шифрування як на сервері так і комп'ютері користувача. Для реалізації цього необхідно розробити програмне забезпечення по заданому алгоритму та встановити його на сервер або хостинг. Приклад коду програмного забезпечення написаний на мові програмування PHP[1] із застосуванням фреймворку Laravel [2].

II. Способи захисту даних

A. На сервері

Щоб ускладнити доступ до конфіденційних даних їх потрібно зашифрувати. Проте, щоб при першому вдалому підібраному паролю не було отримано доступ до всієї інформації, необхідно використовувати різні паролі для шифрування файлів, в яких знаходиться важлива інформація, а їхня довжина повинна бути не менша 10 символів. Також необхідно прибрати можливу ідентифікацію зашифрованих файлів по користувачу, тобто ніхто не повинен знати, кому вони належать. Назви файлів повинні бути згенеровані випадковим чином і не містити будь-якої інформації, щоб могла на щось вказувати. В ідеальних умовах навіть адміністрація сервера не може знати, якому користувачу належать які файли.

B. На комп'ютері користувача

Щоб уникнути несанкціонованого доступу до інформації на стороні сервера і в разі втрати доступу до комп'ютера або отримання до нього доступу третьою особою, буде застосоване шифрування інформації[3](додатково до захищеного протоколу передачі даних HTTPS[4]), яка передається на сервер. Також це дає нам впевненість в тому, що програмне забезпечення на сервері не може читати інформацію, яку ми передаємо і коли він цю інформацію поверне, то вона не буде у відкритому вигляді.

III. Алгоритм роботи

Перед початком роботи необхідно зрозуміти, що повинно бути в результаті. Важливим критерієм в першу чергу є безпека, а далі зручність використання. Щодо зручності, то користувач повинен отримати ідентифікатор(який є звичайним рядком) для доступу до особистих даних та їхнього перезапису. Ввести дані та вказати пароль.

A. Особливості роботи алгоритму

Даний алгоритм розроблений таким чином, щоб сервер не міг отримати доступ до даних, які зберігаються у файлі, без запиту користувача, а

користувач, маючи ідентифікатор, не міг отримати доступ до даних без сервера та пароля.

B. Що являє собою ідентифікатор користувача

Ідентифікатор користувача – це закодований рядок в base64 [5], який містить зашифровану інформацію [6], а саме:

- Назва файлу
- Пароль шифрування файлу

Ключ шифрування ідентифікатора користувача знаходиться у базі даних на сервері. Структура бази даних [5] виглядає наступним чином:

ID	*Hash*	*Info*
int	string	string

ID – ідентифікатор рядка в базі даних.

Hash – це може бути результат перетворення ідентифікатора користувача алгоритмом хешування sha256 або sha512.

Info – містить інформацію щодо розшифрування ідентифікатора користувача.

Причина використання хешу – це те, що неможливо зрозуміти, з яких даних був отриманий даний хеш. Таким чином на сервері відсутня інформація щодо розшифрування файлів із даними користувачів.

C. Що являє собою файл із даними користувача?

Файл має випадково згенеровану назву. Його вміст вперше був зашифрований користувачем особистим паролем до того як інформація попала на сервер, а на сервері повторно зашифрована унікальним паролем, який знаходиться у зашифрованому ідентифікаторі користувача, який повертається користувачу після успішного збереження даних на сервері.

D. Чи можливо підібрати ідентифікатор користувача?

Довжина ідентифікатора залежить від технічної реалізації, у нашій реалізації програма генерує ідентифікатор із 376 символів, проте 2 символи вкінці завжди однакові. На даний момент не існує технологій, які могли б швидко вгадати подібну комбінацію.

E. Що буде, якщо втратити ідентифікатор користувача?

Втрата ідентифікатора – це втрата даних, до яких можна було доступитися за цим ідентифікатором. По-перше, якщо програмним забезпеченням користується декілька людей, то стає неможливим знайти необхідний зашифрований файл. По-друге, якщо і розшифровувати файл, то це займе значну частину часу, можливо вічність(залежить від алгоритму і довжини пароля, у нашому випадку 128 символів).

F. Що буде, якщо до ідентифікатора мають доступ треті лиця

Знання ідентифікатора користувача ще не означає доступ до даних. Так, сервер поверне дані, але вони і надалі у зашифрованому вигляді. Проте, якщо користувач встановив ненадійний пароль, то розшифрування цих даних може вдатися. Якщо було виявлено, що ідентифікатор може хтось знати, то потрібно виконати перезапис даних по даному ідентифікатору і запис конфіденційних даних в новий ідентифікатор.

IV. Застосування та висновки

Основна ідея алгоритму полягає в реалізації взаємозалежних відносин між користувачем і сервером, таким чином, щоб інформація, яка знаходиться у користувача, а саме ідентифікатор, не могла бути прочитана і так само дані користувача, які зберігаються на сервері, не могли бути прочитані. Проте, коли поєднуються дані обох сторін, тоді, і тільки тоді, можна було отримати дані, які відправляв користувач на сервер, тобто дані, які знаходяться у зашифрованому файлі. Наданий алгоритм можна застосовувати для збереження важливих даних і отримувати до них доступ з будь-якого місця, де є підключення до інтернету. Якщо доопрацювати фронтальну частину сайту[8], то вміст даних, які зберігає користувач, може не обмежуватися кількома рядками – це можуть бути повноцінні файли, фото, бази даних.

Література

[1] URL: https://www.php.net/ PHP: Hypertext Preprocessor

[2] URL: https://laravel.com/ Laravel – The PHP Framework For Web Artisans

[3] URL: https://github.com/ssashkaa01/security_cloud/blob/master/resources/views/main.blade.php#L93) Приклад реалізації коду для шифрування даних на стороні користувача у веб браузері

[4] URL: https://developers.google.com/search/docs/advanced/security/https How To Secure Your Site with HTTPS | Google Search Central

[5] URL: https://github.com/ssashkaa01/security_cloud/blob/master/app/Http/Controllers/SecurityCloudController.php#L84 Приклад коду, де декодується ідентифікатор користувача

[6] URL: https://github.com/ssashkaa01/security_cloud/blob/master/app/Http/Controllers/SecurityCloudController.php#L73 Приклад коду, де шифрується ідентифікатор користувача

[7] URL: https://github.com/ssashkaa01/security_cloud/blob/master/database/migrations/2021_08_31_113058_create_security_cloud_table.php) Приклад міграції бази даних на PHP фреймворку Laravel

[8] URL: https://github.com/ssashkaa01/security_cloud/blob/master/resources/views/main.blade.php) Приклад коду, де описано фронтальну частину сайту.