УДК 004.056.53

О. М. Панаско, *к.т.н., доцент,*
e-mail: lena.pa@ukr.net
М. В. Хроленко, *магістр з інформаційної безпеки,*
e-mail: mkhrolenko@gmail.com
Черкаський державний технологічний університет,
б-р Шевченка, 460, м. Черкаси, 18006, Україна

ЗАБЕЗПЕЧЕННЯ ЗАХИСТУ ПАРОЛІВ КОРИСТУВАЧІВ В ПРОЦЕДУРАХ АУТЕНТИФІКАЦІЇ

Стаття присвячена дослідженню підходу щодо забезпечення захисту паролів користувачів, які відіграють визначальну роль при реалізації процедур аутентифікації, авторизації та аудиті систем захисту із використанням до парольної інформації хешування із додаванням так званої «солі» – рядка випадкових даних, який подається на вхід хеш-функції одночасно із вихідними даними. Проведено аналіз відомих способів атак на систему аутентифікації, що пов'язані із отриманням паролів при зламі системи за їх хеш-значеннями, до яких, зокрема, відносяться метод повного перебору (bruteforce), пошук за словником, застосування спеціальних структур даних – таблиць пошуку, різних їх модифікацій, зокрема, зворотніх таблиць пошуку, а також таких, що відомі як «rainbow» таблиці. Зважаючи на сучасні інформаційно-технічні можливості, повністю запобігти зазначеним атакам неможливо, але при цьому є доцільним впливати на те, щоб значно знижувати їх ефективність. На основі проведеного аналізу розглядається напрямок, що здатний забезпечити захист паролів користувачів внаслідок ускладнення відновлення вихідних паролів за їх хеш-значеннями за попередньо сформованими «rainbow» таблицями, який ґрунтується на використанні хешування із додаванням до пароля так званої «солі».

Ключові слова: *аутентифікація, авторизація, системи захисту, пароль користувача, хешування, «rainbow» таблиці, застосування «солі».*

Постановка проблеми. В еру інформаційних технологій та зростання ролі всесвітньої павутини для сучасного суспільства дуже гостро постає питання забезпечення захисту даних, що обумовлюють ідентифікацію користувача та відповідно перевірку та підтвердження прав його доступу до системи, даних та функцій (аутентифікація та авторизація).

Отримання доступу до системи є результатом представлення користувачем відповідних даних, зокрема, імені користувача та пароля, що являє собою секретне слово або певну послідовність символів, призначену для підтвердження особи або її прав. Сформований користувачем пароль в подальшому зберігається в системі і наразі постає питання доцільності зберігання пароля у відкритому – незахищеному вигляді, оскільки це обумовлює зловмиснику можливість отримати несанкціонований доступ до системи. В даній статті досліджується аспект зберігання паролів у хешованому вигляді з використанням так званої «солі», що на сьогоднішній день вважається розповсюдженим способом підвищення надійності збереження паролів та їх захисту.

Аналіз останніх досліджень і публікацій.

Дослідження в галузі забезпечення захисту інформації в комп'ютерних системах та мережах характеризуються надзвичайною актуальністю на сьогоднішній день. В цьому контексті слід відзначити роботи М.В. Гайворонського, В.С. Галатенка, О.М. Новікова, А.Ю. Щеглова та інших. Одним із загально прийнятих підходів до забезпечення безпеки інформаційних ресурсів є виконання базових технічних вимог, що повинні бути впроваджені при розробці систем захисту: аутентифікації, авторизації та аудиті. В літературі цей підхід відомий як «Концепція трьох «А». Будь-яка ефективна система захисту інформаційних ресурсів повинна мати відповідні засоби для реалізації зазначених процедур [1,2,4,5]. Важливим аспектом захисту інформації в комп`ютерних мережах є реалізація основних криптографічних алгоритмів для шифрування файлів та паролів [1]. Для пере-

дачі та зберігання паролів використовують шифрування із застосуванням хешування. Аналіз наукових джерел дозволяє стверджувати про різноманітність сфер застосування процедури хешування інформації, зокрема, при обчисленні контрольних сум від даних з метою подальшого виявлення в них помилок, при побудові асоціативних масивів, при пошуку дублікатів в серіях наборів даних, при реалізації електронного підпису, а також при зберіганні парольної інформації в системах захисту у вигляді хеш-коду тощо. Протягом останніх років здійснюється потужна робота з приводу розробки нових, а також вдосконаленню існуючих методів криптографічного хешування. В цьому напрямку слід відзначити роботи Лужецького В. А. та Баришева Ю. В., Бойка А. О. та Горбенка І. Д. Значний внесок у розвиток алгоритмів хешування також внесли Д. Кнут, У. Пітерсон та Ханс Петер Лун.

Метою даної роботи є дослідження підходу щодо забезпечення захисту паролів користувачів в процедурах аутентифікації на основі застосування хешування із додаванням додаткового значення, так званої «солі», що здатний знизити ефективність відомих методів отримання паролів за їх хеш-значеннями.

Виклад основного матеріалу. Термін хешування пов'язаний із процедурою перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Це дає змогу зберігати паролі користувачів в системі чи на сервері в захешованому вигляді, що ускладнює можливість зловмисникам отримати несанкціонований доступ до даних, функцій чи програм.

Для процедури хешування використовуються хеш-функції, які будуються за ітеративною схемою, коли вихідне повідомлення розбивається на блоки певного розміру M_i , а над ними виконуються ряд перетворень з використанням як зворотних, так і незворотних операцій. На вхід кожного циклу хешування подається вихід попереднього циклу, а також черговий блок повідомлення, тобто $h_i = f(M_i, h_{i-1})$ (рис.1).

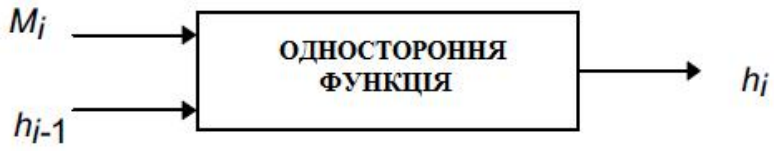

Рис. 1. Отримання хеш-значення

Розрізняють криптографічні та некриптографічні хеш-функції. Слід зазначити, що криптографічні функції хешування забезпечують дотримання наступних вимог:

– якщо відоме значення хеш-функції h, процес знаходження повідомлення M такого, що $H(M) = h$, повинен бути обчислювально складним;

– при заданому повідомленні M процес знаходження іншого повідомлення M', такого, що $H(M) = H(M')$, повинен бути обчислювально складним [2].

Важливою особливістю хеш-функцій є те, що вони не допускають зворотного перетворення, що полягає в неможливості отримання вихідного повідомлення за його дайджестом (хеш-значенням). При зломі системи існує потенційна загроза викрадення паролів, що вимагає використання криптостійких хеш-функцій для зберігання паролів користувачів, якщо зловмисником ініційовано атаку на систему аутентифікації. До найрозповсюдженіших криптостійких хеш-функцій можна віднести, зокрема, SHA256, SHA512, RipeMD та WHIRLPOOL. Вибір тієї чи іншої хеш-функції визначається специфікою задачі, що розв'язується.

Для перевірки коректності введеного пароля система реалізує порівняння між паролем, що був збереженим при реєстрації користувача, а також введеним під час процедури аутентифікації користувача. Для протидії загрозі несанкціонованого доступу, внаслідок того, що ідентичні рядки даних мають однакові хеш-значення, існує можливість перевірки на відповідність збереженого хеш-значення пароля користувача та згенерованого хеш-значення, що відповідає введеному паролю.

Внаслідок цього при реєстрації, коли вводиться ім'я користувача (логін) та пароль, останній хешується та зберігається відповідно в системі чи на сервері. Під час процедури

аутентифікації користувача до введеного пароля застосовують процедуру хешування і порівняння із збереженим хеш-значенням, що відповідає введеному логіну. Користувач отримує доступ до системи (авторизується) у випадку збіжності хеш-значень.

Застосування хешування унеможливлює визначення оригінального пароля для зловмисника і водночас надає можливість порівняння отриманого хеш-значення, що зберігається в системі, з хеш значенням введеного пароля користувача в процедурі аутентифікації. Внаслідок цього не можна стверджувати, що застосування криптографічних хеш-функцій для пароля користувача забезпечить його абсолютний захист.

Хоча хеш-функція і є односторонньою і відповідно неможливо провести зворотну хешуванню операцію та відновити вихідні дані, це не виключає здатність отримати необхідні дані іншим способом з урахуванням сучасних технологічних можливостей. З цього приводу можна навести бази даних, що сформовані з хеш-значень поширених слів і коротких фраз. Крім того прості паролі можна визначити за допомогою процедури повного перебору (bruteforce) або перебору за словником. Не слід виключати і можливість застосування пошукової системи для пошуку за хеш значенням – ввівши хеш-значення, з'являється ймовірність дізнатись відкритий пароль із напрацьованих на сьогодняшній день онлайн баз даних, в яких зберігаються хеші [3].

До відомих способів отримання паролів за їх хеш-значеннями, зокрема, належать:

– пошук хеш-значення за так званими «rainbow» (веселковими) таблицями – базами даних, де накопичуються заздалегідь визначені хеші для будь-яких можливих рядків;

– метод грубої сили (bruteforce), що полягає у переборі всіх можливих комбінацій символів і їхньому хешуванні доти, поки не отримано відповідне хеш-значення;

– пошук за словником, який подібний до попереднього способу, але для нього є характерним обмеження процедури перебору за рахунок того, що розглядаються не всі можливі комбінації символів, а їхня підмножина – всього декілька тисяч найпопулярніших паролів вигляду «qwerty», «asdf», «123», «password» та ін.

Найтривіальніший спосіб зламати хеш-код – це спробувати вгадати пароль, обчислюючи хеш-код для кожного з можливих варіантів і перевіряючи на збіжність отриманого хеш-коду із тим, що потрібно зламати.

Атака за словником заснована на використанні файлу, який представлено словами, фразами, поширеними паролями та іншими рядками, які з певною ймовірністю можуть бути використаними в якості пароля. Кожне слово в файлі захешовано з метою подальшого порівняння його хеш-значення із хеш-значенням пароля. Зазвичай, файли словників побудовані на основі «добування» слів з великих масивів тексту або з реальних баз даних паролів.

Сутність атаки повним перебором полягає у переборі всіх можливих комбінацій символів заданої довжини. Ці атаки вимагають дуже великих обчислювальних витрат, і, зазвичай, вони найменш ефективні за показником числа зламаних хеш-кодів по відношенню до часу виконання, а втім врешті-решт вони завжди знаходять пароль. В цілому, оскільки неможливо повністю запобігти атакам за словником або атакам повним перебором, доцільно намагатися знижувати їхню ефективність.

Достатньо оптимальним методом для дуже швидкого злому великої кількості хеш-кодів одного типу є застосування таблиць пошуку, в яких заздалегідь обчислені хеш-значення паролів зі словника паролів зберігаються в спеціальних структурах даних – таблицях пошуку. Особливість способу організації таблиць пошуку обумовлює високу швидкість процедури пошуку хеш-кодів.

Зворотні таблиці пошуку дозволяють зловмисникові застосувати атаку за словником або повним перебором до багатьох хеш-кодів одночасно без наявності попередньо обчисленої таблиці пошуку. Спочатку зловмисник створює таблицю пошуку, яка зіставляє кожен хеш-код пароля з скомпрометованої бази даних для користувача зі списком користувачів, які мали цей хеш-код, а потім – отримує хеш-значення кожного передбачуваного пароля і використовує таблицю пошуку для формування списку користувачів, чий пароль був визначений. Зазвичай кілька користувачів мають один і той же пароль, а тому цей вид атак особливо ефективний.

Найкращими таблицями пошуку вважаються «rainbow» таблиці, які набагато ефективніші за звичайні та зворотні таблиці пошуку, оскільки займають набагато менше

пам'яті при своїй побудові. «Rainbow» таблиці – це заздалегідь визначений набір даних, який містить хеш-функції, що відповідають множині комбінацій символів. Структура таблиць дозволяє дуже швидко знаходити відповідний пароль за відомим значенням хеш-функції. В основу механізму створення «rainbow» таблиць покладено побудову ланцюжків можливих паролів, початок яких – це певний випадковий пароль фіксованої довжини, а в середині – попеременне застосування хеш-функції та функції редукції [7].

З огляду на можливість отримання відкритого пароля за його хеш-значенням за вище описаними методами, а також враховуючи сучасні інформаційно-технічні можливості доцільно зазначити, що для забезпечення захисту паролів користувачів від пошуку за «веселковими» таблицями для процедури хешування рекомендується використовувати деяку додаткову комбінацію символів, так звану «сіль» (salt) [3,6,8].

У криптографії з цим поняттям пов'язано рядок випадкових даних, який подається на вхід хеш-функції разом з вихідними даними, і зазвичай використовується для подовження рядка пароля, що значно ускладнює відновлення вихідних паролів за допомогою попередньо побудованих «веселкових» таблиць. Слід відзначити, що «сіль» при цьому не здатна захистити від повного перебору для кожного пароля окремо – все залежить лише від затраченого для отримання потрібного результату часу. «Сіль» збільшить час підбору, що в свою чергу призведе до підвищення криптостійкості хеш-коду. Зберігати «сіль» обов'язково потрібно поруч з хеш-значенням, в якому її було додано до пароля, для подальшої перевірки вірності пароля при аутентифікації користувача. Для забезпечення унікальності кожного конкретного пароля «сіль» слід генерувати за допомогою генератора випадкових чисел.

Використання «солі» характеризується динамічністю: навіть при використанні двох однакових паролів із додаванням випадково згенерованої «солі» їхні хеш-значення будуть різні. Окресливши переваги використання «солі» доцільно розглянути механізм формування паролю користувача при реєстрації в системі на основі хешування із додаванням «солі» (рис.2).

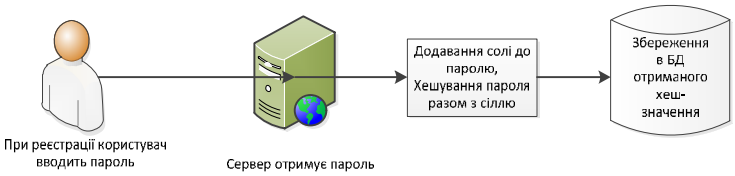

Рис. 2. Схема реєстрації користувача з використанням «солі» під час хешування

Наведеній на рис.2 схемі реєстрації користувача відповідають окремі фрагменти програмних реалізацій мовою програмування C#.

Після введення користувачем імені та пароля для реєстрації відбувається генерація випадкової «солі»:

```
public string doSalt(){
      Random rnd = new Random();
      string salt = rnd.Next()
            .ToString();
      return salt;
}
```

Згенерована «сіль» додається до введеного користувачем пароля:

```
string soltPassw=password+doSalt();
```

В даному фрагменті `password` являє собою змінну, в яку записується пароль, а `doSalt()` – метод генерації «солі».

Виконання хешування пароля з «сіллю» відбувається на основі методу `doHash()`, який виконує процедуру хешування за допомогою алгоритму SHA512:

```
doHash(soltPassw);
public string doHash(string passw){
      byte[] HashValue;
      byte[] password =
      System.Text.Encoding.UTF8
            .GetBytes(passw);
      SHA512 SHhash = new
SHA512Managed();
      HashValue = Shhash
            .ComputeHash(password);
      string s =
            System.Text.Encoding
                  .UTF8.GetString(
                        HashValue, 0,
                        HashValue.Length)
}
```

В результаті формується рядок, який відповідає захешованому паролю з «сіллю».

```
string hashPassw = doHash(soltPassw);
```

Отримане хеш-значення записується разом з відкритою «сіллю» та іменем користувача (в даному прикладі в базі даних MongoDB).

```
var client =
new MongoClient(
     "mongodb://localhost:27017");
var dbUsers = client.GetDatabase(
     "users");
var usersData =
     dbUsers.GetCollection<UsersInfo>(
          "usersData");
usersData.InsertOne(uInfo);
```

В наведеному фрагменті usersData являє собою об'єкт типу UsersInfo, в якому зберігається ім'я користувача, отримане хеш-значення та «сіль».

Коли в подальшому користувачу потрібно повторно надавати свої ім'я та пароль, процедура аутентифікації відбуватиметься відповідно до блок-схеми (рис.3):

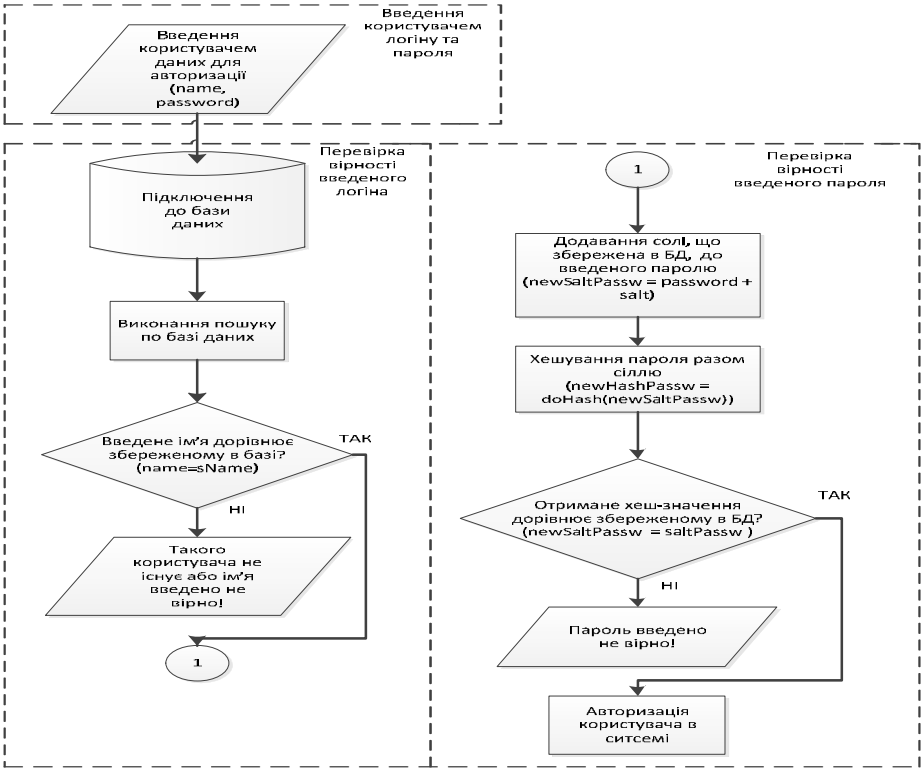

Рис. 3. Блок-схема процедури аутентифікації користувача

Після введення користувачем логіну та пароля відбувається перевірка вірності введеного пароля:

```
public void FindUsers(
     string userName, string pasw){
var client =
new MongoClient(
     "mongodb://localhost:27017");
var dbUsers =
     client.GetDatabase("users");
var usersData =
     dbUsers.GetCollection<UsersInfo>
          ("usersData");
```

Відбувається пошук в базі даних (БД) інформації користувача за його іменем – захешованих пароля та «солі» та відкритої «солі»:

```
var filter1 = Builders<UsersInfo>
     .Filter.Eq(
          c => c.Name,
          userName);
var users = usersData.Find(filter1)
     .ToList();
```

До введеного пароля додається збережена в базі «сіль» та виконується процедура хешування (для знайденого запису користувача):

```
foreach (var doc1 in users){
     string paswSalt = pasw + doc1.Salt;
     Hashing hashPasw = new Hashing();
     String hashP = hashPasw.doHash(
     paswSalt);
}
```

Після отримання хеш-значення виконується пошук в БД об'єкта, захешовані пароль

з «сіллю» якого рівні попередньо отриманому хеш-значенню. Навіть якщо у різних користувачів паролі будуть однаковими, програма знайде лише одного користувача з ідентичними згенерованим хеш-значенням та збереженим, оскільки «сіль» у кожного користувача різна.

```
var filter2 = Builders<UsersInfo>
     .Filter.Eq(c => c.Passw, hashP);
var usersPasw = usersData
     .Find(filter2).ToList();
foreach (var doc2 in usersPasw){
     MessageBox.Show(doc2.Name +
          " Вас авторизовано!");
}
```

Користувача буде успішно авторизовано при знайденій відповідності в БД згенерованого хеш-значення та збереженого при реєстрації для введеного імені користувача. Це демонструє відповідний фрагмент програмної реалізації, в якому для введених даних (`userName` та `password`) за допомогою методу `FindUsers()` шукаються ідентичні ім'я та пароль в БД:

```
string userName =  NewUser;
string password = password12345;
MongoDB mdb = new MongoDB();
mdb.FindUsers(userName, password);
```

В цілому, використання випадкової «солі», яка є унікальною для кожного користувача, забезпечить захист паролів користувачів в процедурах аутентифікації від атак на основі використання «rainbow» таблиць, оскільки кожному паролю відповідатиме своя унікальна «сіль» і це в свою чергу обумовить необхідність генерування зловмисником значної кількості окремих веселкових таблиць, що з практичної точки зору вважається недоцільним. Генерація «солі» не потребує затрат часу та додаткових ресурсів системи, порівняно із відомим методом подвійного хешування. Внаслідок того, що «сіль» можна зберігати у відкритому вигляді, процедура аутентифікації не передбачає додаткових витрат на маніпуляції з нею. В результаті для підвищення надійності збереження паролів користувачів та забезпечення їхнього захисту вважається доцільним проводити хешування із використанням додаткового значення – «солі».

Висновки. В процесі даного дослідження розглянуто підхід щодо забезпечення захисту паролів користувачів в процедурах аутентифікації на основі використання хешування із додаванням унікального параметру, так званої «солі», що в результаті проведення аналізу основних методів отримання паролів за їх хеш-значеннями здатно знизити ефективність застосування зазначених методів, зокрема, за попередньо сформованими «rainbow» таблицями.

Список літератури

1. Шнайер Б. Прикладная криптография: протоколы, алгоритмы и исходные тексты на языке Си. 2-е изд. Москва: Триумф, 2002. 688 с.
2. Горбенко И. Д. Функции хеширования. Понятия, требования, классификация, свойства и применение. *Радиоэлектроника и информатика.* 1998. № 1. С. 64–69.
3. «Соленое» хеширование паролей: делаем правильно. URL: http://www.internet-technologies.ru/articles/article_1807.html.
4. Чунарьова А. В. Аналіз існуючих шаблонів систем автентифікації в інформаційно-комунікаційних системах та мережах. *Безпека інформації.* 2012. № 2. С. 65–70.
5. Єсіна М. В., Горбенко І. Д. Багатофакторна аутентифікація: використання механізмів двофакторної автентифікації для захисту від несанкціонованого доступу. *Компьютерное моделирование в наукоемких технологиях (КМНТ-2014):* труды научно-технической конференции с международным участием, 28–31 мая 2014 г. Харьков: Харьк. нац. ун-т им. В. Н. Каразина, 2014. С. 159–162.
6. Хэширование паролей. URL: http://phpfaq.ru/tech/hashing.
7. Словарные атаки на хэш-функции. URL: http://www.panasenko.ru/articles/168/168.html.
8. Pritesh N. A., Jigisha K., Paresh V. Crypto graphy Application using Salt Hash Technique. *International Journal of Application or Innovation in Engineering & Management.* 2013. № 6. С. 236–239.

References

1. Shnayer, B. (2002) Applied Cryptography: Protocols, Algorithms, and Source Code in C. Moscow: Triumf, 2002, 610 p.
2. Gorbenko, I. D. (1998) Hashing functions. Concepts, requirements, classification, properties and applications. *Radyoelektronyka y ynformatyka,* No. 1, pp. 64–69.

3. "Salty" hashing of passwords: we do it right. URL: http: //www.internet-technologies.ru/ articles/article_1807.html.
4. Chunarova, A. V. (2012) Analysis of existing patterns of authentication systems in information and communication systems and networks. *Bezpeka informatsii*, No. 2, pp. 65–70.
5. Yesina, M. V., Horbenko, I. D. (2014) Multifactor Authentication: Using two-factor authentication mechanisms to protect against unauthorized access. *Kompyuternoe modelirovanie v naukoemkih tehnologiyah (KMNT-2014): Trudyi nauchno- tehnicheskoy konferentsii s mezhdunarodnyim uchastiem, 28-31 maya 2014 g.* Harkov: Harkovskiy natsio nalnyiy universitet im. V. N. Karazina, pp. 159–162.
6. Password hashing. URL: http://phpfaq. ru/tech/hashing.
7. Dictionary attacks on hash functions. URL: http:// www.panasenko.ru/articles/168/168.html.
8. Pritesh, N. A, Jigisha, K., Paresh, V. (2013) Cryptography Application using Salt Hash Technique. *International Journal of Application or Innovation in Engineering & Management,* No. 6, pp. 236–239.

O. M. Panasko, *Ph.D., associate professor,*
e-mail: lena.pa@ukr.net**,**
M. V. Khrolenko, *Master in cybersecurity*
e-mail: mkhrolenko@gmail.com
Cherkassy State Technological University,
Shevchenko blvd., 460, Cherkassy, 18006, Ukraine

PROVISION OF USER PASSWORDS PROTECTION IN AUTHENTICATION PROCEDURES

The article is devoted to the study of the approach to ensuring the user passwords protection, which play a decisive role in the implementation of authentication, authorization and audit security systems of using password-based hashing with the addition of so-called "salt" – a string of random data that is submitted to the input of hash functions simultaneously with source data. The analysis of known methods of attack on the authentication system, which is associated with the receipt of passwords for their hash values, which include, in particular, the bruteforce method, dictionary search, the use of special data structures - search tables, various modifications, in particular, "rainbow" tables is shown in this article. Given the current information and technical capabilities, it is not possible to completely prevent these attacks, but it is advisable to reduce their effectiveness. On the basis of the analysis, a direction is considered that can provide protection of user passwords due to the difficulty of restoring output passwords on their hash values for pre-formed "rainbow" tables, which is based on the use of hashing with the additional value which is known as "salt".

Keywords: *authentication, authorization, protection systems, user password, hashing, "rainbow" tables, Salt Hash Technique.*

Статтю представляє О. М. Панаско, к.т.н., доцент