

УДК 004.056.53

Изучение уровня стойкости пароля для подтверждения авторизуемой личности

Носов Георгий Дмитриевич

ngd21ki185@student.bmstu.ru

Сосулев Лев Геннадьевич

slg21ki234@student.bmstu.ru

Ильичев Владимир Юрьевич

ilyichev.vyu@bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Россия

В настоящее время производительность компьютеров возрастает, но при этом главным механизмом аутентификации все еще является защита с использованием паролей, поэтому необходимо повышать их стойкость. Показана уязвимость слабого требования к парольной комбинации (CWE-521), а также рассмотрены последствия использования слабых паролей для безопасности систем и предложены рекомендации по повышению уровня защиты. В данной статье ставится цель повышения осведомленности о важности строгих политик безопасности паролей в разработке программного обеспечения.

Ключевые слова: стойкость защиты, слабый пароль, аутентификация, информационная безопасность

CWE-521 (слабое требование к паролю) — уязвимость, связанная с недостаточно строгими требованиями к паролям пользователей. Эта уязвимость проявляется в том, что система допускает использование слабых или легко поддающихся атакам паролей, что увеличивает риск несанкционированного доступа к информации или системе [1].

Цель работы — исследование устойчивости паролей для проверки подлинности пользователей.

Основные характеристики уязвимости CWE-521 включают:

- пароли низкой сложности: система позволяет пользователям устанавливать пароли, которые легко угадать или подобрать с помощью атак словаря или перебора;
- отсутствие требований к длине и составу пароля: система не накладывает минимальных требований к длине или сложности пароля, что делает его уязвимым к различным видам атак;
- отсутствие механизмов защиты от часто используемых паролей: система не проверяет новые пароли на принадлежность к списку распространенных или скомпрометированных, что делает их уязвимыми к атакам;
- отсутствие требований к периодической смене пароля: система не требует периодической смены пароля, что может привести к длительному использованию одного и того же слабого пароля.

Эта уязвимость может дать возможность злоумышленнику угадать пароли пользователей и получить несанкционированный доступ к системе, поэтому необходимо рассмотреть модель атак для создания сценариев атак на кон-

кретную систему [2]. Следующие шаблоны атак могут быть использованы для взлома системы, содержащую конфиденциальную информацию в открытом виде в соответствии с классификацией CAPEC (перечисление и классификация общих шаблонов атак) [3]:

- CAPEC-16: атака паролей на основе словаря;
- CAPEC-49: подбор пароля;
- CAPEC-55: взлом паролей с помощью радужных таблиц;
- CAPEC-70: перебор обычных или стандартных имен пользователей и паролей;
- CAPEC-112: грубая сила (brutforce);

Методами предотвращения уязвимости CWE-521 в современных компьютерных системах являются [4]:

- внедрение безопасного жизненного цикла разработки с экспертами по безопасности приложений для оценки и разработки механизмов управления, связанных с безопасностью и конфиденциальностью;
- создание и использование библиотеки безопасных шаблонов проектирования или предварительно подготовленных компонентов;
- применение моделирования угроз для критических аспектов аутентификации, контроля доступа, и логики бизнес-процессов;
- интеграция языка безопасности и механизмов управления в пользовательские сценарии;
- внедрение проверки подлинности на всех уровнях приложения (от интерфейса до логики сервера);
- разработка модульных и интеграционных тестов для подтверждения устойчивости к модели угроз всех критических потоков. Создание сценариев использования и небезопасного использования для каждого уровня приложения;
- разделение системного и сетевого уровня в соответствии с требованиями по защите и уязвимостям;
- ограничение потребления ресурсов для пользователей или служб.

Применим полученные знания для создания макета взаимодействия банковского приложения с сервером.

Для мобильности работы тестировщика на проникновение клиентскую часть банковского приложения удобно реализовать в виде приложения на операционной системе Android, написанном на языке Java, с использованием библиотеки Android и библиотеки PasscodeView [5] (рис. 1). При этом все конфиденциальные данные хранятся на сервере.

Приложение при входе требует секретный код от пользователя длиной 4 символа, валидация которого происходит на сервере и в зависимости от ответа сервера приложение переходит к следующему экрану после аутентификации.

Для проверки секретного кода пользователя был написан HTTP-сервер на языке python с помощью фреймворка для создания сетевого сервера — FLASK. Данный сервер слушает все подключения к своему IP-адресу на порту 25565 и обрабатывает POST-запрос к конечной точке /send_pin. Клиент должен отправить json с ключом pin_code и значением секретного кода. При

удачной валидации сервер отправляет код 200 и json с ключом flag и с значением секретных данных, переданных сервером (рис. 2).



Рис. 1. Внешний вид клиента банковского приложения на Android

- ▼ Hypertext Transfer Protocol
 - ▼ POST /send_pin HTTP/1.1\r\n
 - > [Expert Info (Chat/Sequence): POST /send_pin HTTP/1.1\r\n]
 - Request Method: POST
 - Request URI: /send_pin
 - Request Version: HTTP/1.1
 - Content-Type: application/json; charset=utf-8\r\n
 - > Content-Length: 19\r\n
 - Host: 192.168.43.54:25565\r\n
 - Connection: Keep-Alive\r\n
 - Accept-Encoding: gzip\r\n
 - User-Agent: okhttp/4.12.0\r\n
 - \r\n
 - [Full request URI: http://192.168.43.54:25565/send_pin]
 - [HTTP request 1/1]
 - [Response in frame: 6358]
 - File Data: 19 bytes
 - ▼ JavaScript Object Notation: application/json
 - ▼ Object
 - ▼ Member: pin_code
 - [Path with value: /pin_code:6788]
 - [Member with value: pin_code:6788]
 - String value: 6788
 - Key: pin_code
 - [Path: /pin_code]

Рис. 2. HTTP запрос клиента к серверу в программе Wireshark

Таким образом, в случае отправки верного секретного ключа, произойдет авторизация и сервер отправит конфиденциальные данные (рис. 3).



Рис. 3. Окно после аутентификации в клиентском приложении

```
1 import requests
2
3 # Задайте адрес вашего сервера
4 SERVER_URL = "http://127.0.0.1:25565/send_pin"
5
6 def brute_force_pincode():
7     for pin in range(10000):
8         formatted_pin = str(pin).zfill(4)
9         response = requests.post(SERVER_URL, json={"pin_code": formatted_pin})
10
11         if response.status_code == 200: # Если пин-код верный
12             print(f"Найден верный пин-код: {formatted_pin}")
13             print("Флаг:", response.json().get("flag"))
14             break
15         elif response.status_code == 403:
16             #print(f"Неверный пин-код: {formatted_pin}")\
17             pass
18         else:
19             print("Ошибка запроса:", response.status_code)
20
21 if __name__ == "__main__":
22     brute_force_pincode()
```

P length: 834 lines: 22 Ln: 5 Col: 1 Pos: 125 Windows (CR LF) UTF-8 INS

C:\Windows\System32\cmd.exe

```
D:\My Data\reverse\ctfs\K2CTF>python bruter.py
Найден верный пин-код: 6788
Флаг: K2CTF{@ndr0id_BrUt3_1s_3@sy}

D:\My Data\reverse\ctfs\K2CTF>
D:\My Data\reverse\ctfs\K2CTF>
D:\My Data\reverse\ctfs\K2CTF>
D:\My Data\reverse\ctfs\K2CTF>
D:\My Data\reverse\ctfs\K2CTF>
D:\My Data\reverse\ctfs\K2CTF>
D:\My Data\reverse\ctfs\K2CTF>
D:\My Data\reverse\ctfs\K2CTF>
D:\My Data\reverse\ctfs\K2CTF>
```

Рис. 4. Скрипт для перебора секретного кода и результат его работы

Так как секретный код ограничен длиной в 4 символа, то его возможно перебрать методом грубой силы за приемлемое время. Для эксплуатации данной уязвимости был написан скрипт для перебора секретного кода, который подключается к серверу и при получении кода ошибки отправляет новый секретный код, пока сервер не вернет успешный код и не произойдет авторизация (рис. 4).

В результате проведенного исследования был создан программный продукт для исследования устойчивости паролей для проверки подлинности пользователей, согласно пункту 522 общего перечня недостатков безопасности (CWE-522).

Литература

- [1] MITRE Corporation. CWE-521 Weak Password Requirements. URL: <https://cwe.mitre.org/data/definitions/521.html> (accessed November 10, 2024).
- [2] ImmuniWeb. Weak Password Requirements Vulnerability. URL: <https://www.immuniweb.com/vulnerability/weak-password-requirements.html> (accessed November 11, 2024).
- [3] MITRE Corporation. CAPEC — Common Attack Pattern Enumeration and Classification. URL: <https://capec.mitre.org/data/index.html> (accessed November 6, 2024).
- [4] OWASP Foundation. A04:2021 Insecure Design. URL: https://owasp.org/Top10/A04_2021-Insecure_Design (дата обращения 26.03.2023).
- [5] Keval Patel. PasscodeView. Open-source project. URL: <https://github.com/kevalpatel2106/PasscodeView> (accessed November 2, 2024).

The research of the password strength level to confirm the authorized person's identity

Nosov G.D.	ngd21ki185@student.bmstu.ru
Sosulev L.G.	slg21ki234@student.bmstu.ru
Ilyichev V.Y.	ilyichev.vyu@bmstu.ru

BMSTU, Moscow, Russia

Currently, computer capabilities are being improved, and the main authentication mechanism is still a password, so it is necessary to increase the strength of the memorized secrets. This research demonstrates the vulnerability of weak password requirements (CWE-521) and discusses the consequences of using weak passwords for system security, along with recommendations for improving the level of protection. This article aims to raise awareness about the importance of strict password security policies in software development.

Keywords: weak password requirements, authentication, information security