Q138. Welcome!

Thank you for participating in our survey.

The purpose of this survey is to examine the security of several code fragments. These code fragments are part of a larger project by the Behavioural Security Research Group at the Institute for Computer Science of the University of Bonn. Our aim is to ensure that the code you will see here is as up-to-date and secure as it can be. You can choose to examine code fragments in up to two programming languanges, and you will receive 50 Euros per language for full and meaningful contributions. Please notify us after you have finished the survey to receive your payment.

If you have any questions or concerns, please do not hesitate to contact us:

Cüneyt Erem (s6cuerem@uni-bonn.de)
Lisa Geierhaas (geierhaa@cs.uni-bonn.de)

Participant Consent

Please read the consent form which is linked below carefully and keep the document.

[Consent Form]

Q135. I have read and understood the consent form and I agree to take part in this survey.

● I Consent

○ I Do Not Consent

Q1. Please state your age

25

Q4. Please state your current occupation (MSc Student, Software Engineer etc.)

Application Security engineer

Q5. How many years of professional experience do you have in software development?

4

Q6. Do you have experience in security concepts?

○ No

● Yes

Q7. Are your tasks in your current occupation related to security concepts?

○ No

● Yes

Q8. How familiar are you with security concepts for implementation of password storage in general?

○ Not familiar at all

○ Slightly familiar

○ Moderately familiar

● Very familiar

○ Extremely familiar

Q9. How familiar are you with implementation of password policy in general?

○ Not familiar at all

○ Slightly familiar

○ Moderately familiar

○ Very familiar

● Extremely familiar

Q10. How familiar are you with implementation of two factor authentication in general?

○ Not familiar at all

○ Slightly familiar

○ Moderately familiar

○ Very familiar

● Extremely familiar

Q11. In what programming language do you have the most experience? (Please select at most two languages)

☑ Python

☑ Java

□ Javascript

□ PHP

☐ GOLang

☐ C#

Q136. The main purpose of this study is to question whether the provided password code blocks (code snippets) are **safe** or how to improve them securely.

The code is presented in three categories: password storage, password policy and two factor authentication.

You are **not** obligated to answer questions that you are unsure of.

While checking the code blocks, you can use **ANY** resource you want to examine the code's security.

Estimated study duration is 10-30 minutes.

You will only be asked questions about the programming languages you choose.

We wish you a good survey.

Q106. **Python**

This code snippet implements password hashing for safe storage.

- It uses Argon2 as hashing algorithm
- The method hash_password is used to hash and salt a password
- The method verify is used to check whether a password matches a hash

Q19. **pip install argon2-cffi**

```
from argon2 import PasswordHasher
from argon2.exceptions import VerifyMismatchError

def hash_password(password):
    ph = PasswordHasher()
    pw_hash = ph.hash(password)

    return pw_hash

def verify(pw_hash, password):
    ph = PasswordHasher()
    try:
        return ph.verify(pw_hash, password)
    except VerifyMismatchError:
        return False

def main():
    #example password
    pw_hash = hash_password('s3cr3t')
    print(pw_hash)
    print(verify(pw_hash, 's3cr3t'))
    print(verify(pw_hash, 's3cr4t'))

    return None

if __name__ == '__main__':
    main()
```

Q12. Do you think the implementation of **hash_password** is up-to-date and secure?

● Yes

○ No (Please specify)

○ I am unsure (Please specify)

Q13. Do you think the implementation of **verify** is up-to-date and secure?

● Yes

○ No (Please specify)

○

I am unsure (Please specify)

Q14. Would you make any other changes to this code snippet?

● No

○ Yes (Please specify)

Q133. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

○ No

● Yes (Please specify)

`https://pythonlang.dev/repo/hynek-argon2-cffi/`

Q110. **Python**

This code snippet implements password hashing for safe storage.

- It uses Bcrypt as hashing algorithm
- The method hash_password is used to hash and salt a password
- The method verify is used to check whether a password matches a hash

Q20. **pip install bcrypt**

```
import bcrypt

def hash_password(password):
    pw_hash = bcrypt.hashpw(password.encode(), bcrypt.gensalt())

    return pw_hash

def verify(pw_hash, password):
    return bcrypt.checkpw(password.encode(), pw_hash)

def main():
    #example password
    pw_hash = hash_password('s3cr3t')
    print(pw_hash)

    print(verify(pw_hash, 's3cr3t'))
    print(verify(pw_hash, 's3cr4t'))

    return None

if __name__ == '__main__':
    main()
```

Q21. Do you think the implementation of **hash_password** is up-to-date and secure?

● Yes

○ No (Please specify)

○

I am unsure (Please specify)

Q22. Do you think the implementation of **verify** is up-to-date and secure?

● Yes

○ No (Please specify)

○ I am unsure (Please specify)

Q23. Would you make any other changes to this code snippet?

○ No

●

Yes (Please specify)

```
def main():
    #example password
    pw_hash = hash_password('s3cr3t')
    print(pw_hash)

    print(verify(pw_hash, 's3cr3t'))
    print(verify(pw_hash, 's3cr4t'))

In these code blocks, it's not recommended to check passwords like this
form (hardcoded). If it is, possible, you can use systems like HashiCorp
Vault (called a secret management process) or at least get your input and
stored as variable.
This recommendation can apply to all code snippets to make sure that you
don't have any hardcode credential/password.
```

Q134. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

● No

○ Yes (Please specify)

Q111. **Python**

This code snippet shows an implementation of a password policy check.
- The method composition uses regexes to verify that the password meets composition criteria (at least one number, one special character, one capital letter and length between 8 and 64)
- The method check_strength uses the library zxcvbn to check a password's strength

Q25. **pip install zxcvbn**

```
import re
from zxcvbn import zxcvbn

def composition(password):
    number = "(?=.*\d)"
    upper_lower = "(?=.*[a-z])(?=.*[A-Z])"
    special = "[^A-Za-z0-9]"
    length = ".{8,64}$"

    test = []
    test.append(True if re.search(re.compile(number), password) else False)
    test.append(True if re.search(re.compile(upper_lower), password) else False)
    test.append(True if re.search(re.compile(special), password) else False)
    test.append(True if re.search(re.compile(length), password) else False)

    return test

def check_strength(password):
    result = zxcvbn(password, user_inputs=['John', 'Smith'])
    strength = { 0: "Worst", 1: "Bad", 2: "Weak", 3: "Good", 4: "Strong" }

    return [result['score'], strength[result['score']], result['feedback']]

def main():
    #example password
    password = 'Abcdefghi1.'
    x = composition(password)
    print("composition: ", x)
    y = check_strength(password)
    print("check_strength: ", y)

    return None

if __name__ == '__main__':
    main()
```

Q15. Do you think the implementation of this code snippet is up-to-date and secure?

○ Yes

○ No (Please specify)

● I am unsure (Please specify)

```
def main(): section

You can use user input to make it more realistic and not be hardcoded.
```

Q16. Would you make any other changes to this code snippet?

● No

○ Yes (Please specify)

Q135. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

○ No

● Yes (Please specify)

```
Make sure that you're not using this one.
https://pypi.org/project/zxcvbn-python/
```

Q112. **Python**

This code snippet shows an implementation of two factor authentication.
- It uses the library otp
- It generates and verifies a totp
- It creates a provisioning uri for the user

Q26. **pip install pyotp**

```
import pyotp

def generate_second_factor():
    shared_secret = pyotp.random_base32()

    return pyotp.TOTP(shared_secret)
```

```
def generate_uri(second_factor, user_mail):
    return second_factor.provisioning_uri(user_mail, issuer_name="Your Secure App")

def main():
    second_factor = generate_second_factor()
    print(second_factor.verify(second_factor.now()))
    user_mail = "alice@google.com"
    provisioning_uri = generate_uri(second_factor, user_mail)
    print(provisioning_uri)

if __name__ == '__main__':
    main()
```

Q17. Do you think the implementation of this code snippet is up-to-date and secure?

● Yes

○ No (Please specify)

○ I am unsure (Please specify)

Q18. Would you make any other changes to this code snippet?

● No

○

Yes (Please specify)

Q136. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

○ No

● Yes (Please specify)

https://medium.com/@taimoor.mirza9595/integrating-2fa-mfa-using-pyotp-193ad858f80d

Q113. **Plain Java**

This code snippet implements password hashing for safe storage.

- It uses Argon2 as hashing algorithm
- The method hashPassword is used to hash and salt a password
- The method verifyPassword is used to check whether a password matches a hash

Q27. **maven**

```
<dependency>
    <groupId>de.mkammerer</groupId>
    <artifactId>argon2-jvm</artifactId>
    <version>2.6</version>
</dependency>
```

gradle

```
compile group: 'de.mkammerer', name: 'argon2-jvm', version: '2.6'
```

ivy

```
<dependency org="de.mkammerer" name="argon2-jvm" rev="2.6"/>
```

```
import de.mkammerer.argon2.Argon2;
import de.mkammerer.argon2.Argon2Factory;
import de.mkammerer.argon2.Argon2Factory.Argon2Types;
import de.mkammerer.argon2.Argon2Helper;

// 1000 = The hash call must take at most 1000 ms
// 65536 = Memory cost
// 4 = parallelism
public static String hashPassword(String password) {
    Argon2 argon2 = Argon2Factory.create(Argon2Types.ARGON2id);
    int iterations = Argon2Helper.findIterations(argon2, 1000, 65536, 4);
    char[] passwordArray = password.toCharArray();
    String hash = argon2.hash(iterations, 65536, 4, passwordArray);
    argon2.wipeArray(passwordArray);
    return hash;
}

public static Boolean verifyPassword(String password, String hashed) {
    Argon2 argon2 = Argon2Factory.create(Argon2Types.ARGON2id);
    return argon2.verify(hashed, password);
}
```

Q29. Do you think the implementation of **hashPassword** is up-to-date and secure?

● Yes

○ No (Please specify)

○ I am unsure (Please specify)

Q30. Do you think the implementation of **verifyPassword** is up-to-date and secure?

● Yes

○ No (Please specify)

○ I am unsure (Please specify)

Q31. Would you make any other changes to this code snippet?

● No

○ Yes (Please specify)

Q137. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

○ No

●

Yes (Please specify)

Make sure all your libraries are not vulnerability affected. In this case, you can find below the current version. If you use a High/Critical affected library, it can be dangerous to your software. You can also use open-source tools like Owasp dependency check and Snyk to get flaws of your libraries.

https://mvnrepository.com/artifact/de.mkammerer/argon2-jvm/2.6

Q114. **Spring Java**

This code snippet implements password hashing for safe storage.

- It uses Argon2 as hashing algorithm
- The method hashPassword is used to hash and salt a password
- The method verifyPassword is used to check whether a password matches a hash

Q32. **maven**

```
<dependency>
   <groupId>org.springframework.security<groupId/>
   <artifactId>spring-security-crypto</artifactId>
   <version>5.2.2.RELEASE</version>
</dependency>

<dependency>
   <groupId>commons-logging</groupId>
   <artifactId>commons-logging</artifactId>
   <version>1.2</version>
</dependency>

<dependency>
   <groupId>org.bouncycastle</groupId>
   <artifactId>bcprov-jdk15on</artifactId>
   <version>1.64</version>
</dependency>
```

gradle

```
compile group: 'org.springframework.security', name: 'spring-security-crypto', version: '5.2.2.RELEASE'
compile group: 'commons-logging', name: 'commons-logging', version: '1.2'
compile group: 'org.bouncycastle', name: 'bcprov-jdk15on', version: '1.64'
```

ivy

```
<dependency org="org.springframework.security" name="spring-security-crypto" rev="5.2.2.RELEASE"/>
<dependency org="commons-logging" name="commons-logging" rev="1.2"/>
<dependency org="org.bouncycastle" name="bcprov-jdk15on" rev="1.64"/>
```

```
import org.springframework.security.crypto.argon2.Argon2PasswordEncoder;

public static String hashPassword(String password) {
   // Argon2PasswordEncoder(int saltLength, int hashLength, int parallelism, int memory, int iterations)
```

```
    // Spring Security uses default values that should be adjusted to your system
    Argon2PasswordEncoder encoder = new Argon2PasswordEncoder();
    return encoder.encode(password);
}

public static Boolean verifyPassword(String password, String hashed) {
    Argon2PasswordEncoder encoder = new Argon2PasswordEncoder();
    return encoder.matches(password, hashed);
}
```

Q34. Do you think the implementation of **hashPassword** is up-to-date and secure?

● Yes

○ No (Please specify)

○ I am unsure (Please specify)

Q35. Do you think the implementation of **verifyPassword** is up-to-date and secure?

● Yes

○

No (Please specify)

○ I am unsure (Please specify)

Q36. Would you make any other changes to this code snippet?

● No

○ Yes (Please specify)

Q138. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

○ No

●

Yes (Please specify)

```
https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/argon2/Argon2PasswordEncoder.html

https://www.programcreek.com/java-api-examples/?api=org.springframework.security.crypto.argon2.Argon2PasswordEncoder
```

Q115. **Plain Java**

This code snippet implements password hashing for safe storage.

- It uses Bcrypt as hashing algorithm
- The method hashPassword is used to hash and salt a password
- The method verifyPassword is used to check whether a password matches a hash

Q37. **maven**

```
<dependency>
    <groupId>org.mindrot</groupId>
    <artifactId>jbcrypt</artifactId>
    <version>0.4</version>
</dependency>
```

gradle

```
compile group: 'org.mindrot', name: 'jbcrypt', version: '0.4'
```

ivy

```
<dependency org="org.mindrot" name="jbcrypt" rev="0.4"/>
```

```
import org.mindrot.jbcrypt.*;

public static String hashPassword(String password) {
    return BCrypt.hashpw(password, BCrypt.gensalt());
}

public static Boolean verifyPassword(String password, String hashed) {
    return BCrypt.checkpw(password, hashed);
}
```

Q38. Do you think the implementation of **hashPassword** is up-to-date and secure?

● Yes

○

No (Please specify)

○ I am unsure (Please specify)

Q39. Do you think the implementation of **verifyPassword** is up-to-date and secure?

● Yes

○ No (Please specify)

○ I am unsure (Please specify)

Q40. Would you make any other changes to this code snippet?

● No

○ Yes (Please specify)

Q139. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

○ No

● Yes (Please specify)

```
https://snyk.io/vuln/maven:org.mindrot:jbcrypt@0.4
```

Q116. **Spring Java**

This code snippet implements password hashing for safe storage.
- It uses Bcrypt as hashing algorithm
- The method hashPassword is used to hash and salt a password
- The method verifyPassword is used to check whether a password matches a hash

Q41. **maven**

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-crypto</artifactId>
    <version>5.2.2.RELEASE</version>
</dependency>

<dependency>
```

```
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.2</version>
</dependency>

<dependency>
    <groupId>org.bouncycastle</groupId>
    <artifactId>bcprov-jdk15on</artifactId>
    <version>1.64</version>
</dependency>
```

gradle

```
compile group: 'org.springframework.security', name: 'spring-security-crypto', version: '5.2.2.RELEASE'
compile group: 'commons-logging', name: 'commons-logging', version: '1.2'
compile group: 'org.bouncycastle', name: 'bcprov-jdk15on', version: '1.64'
```

ivy

```
<dependency org="org.springframework.security" name="spring-security-crypto" rev="5.2.2.RELEASE"/>
<dependency org="commons-logging" name="commons-logging" rev="1.2"/>
<dependency org="org.bouncycastle" name="bcprov-jdk15on" rev="1.64"/>
```

```
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

public static String hashPassword(String password) {
    BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
    return (encoder.encode(password);
}

public static Boolean verifyPassword(String password, String hashed) {
    BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
    return(encoder.matches(password, hashed));
}
```

Q42. Do you think the implementation of **hashPassword** is up-to-date and secure?

● Yes

○ No (Please specify)

○

I am unsure (Please specify)

Q43. Do you think the implementation of **verifyPassword** is up-to-date and secure?

● Yes

○ No (Please specify)

○ I am unsure (Please specify)

Q44. Would you make any other changes to this code snippet?

● No

○

Yes (Please specify)

Q140. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

○ No

● Yes (Please specify)

```
https://security.snyk.io/vuln/SNYK-JAVA-ORGSPRINGFRAMEWORKSECURITY-570203

I use the link to ensure the library version is not affected by any
significant flaw.
```

Q117. **Plain Java**

This code snippet shows an implementation of a password policy check.

- The method composition uses regexes to verify that the password meets composition criteria (at least one number, one special character, one capital letter and length between 8 and 64)
- The method check_strength uses the library zxcvbn to check a password's strength

Q45. **maven**

```
<dependency>
    <groupId>com.nulab-inc</groupId>
    <artifactId>zxcvbn</artifactId>
    <version>1.5.2</version>
</dependency>


import com.nulabinc.zxcvbn.Strength;
import com.nulabinc.zxcvbn.Zxcvbn;
import java.util.HashMap;
import java.util.Map;

public static String composition(String password) {
    String number = "(?=.*\\d).{2,}";
    String upper_lower = "(?=.*[a-z])(?=.*[A-Z]).{2,}";
    String special = "(?=.*[^A-Za-z0-9]).{2,}";
    String length = ".{8,64}";

    return "" + password.matches(number) + ", " + password.matches(upper_lower) + ", " +
password.matches(special) + ", " + password.matches(length);
}

public static String check_strength(String password) {
    Zxcvbn zxcvbn = new Zxcvbn();
    Strength strength_levels = zxcvbn.measure(password);

    Map map = new HashMap();
    map.put(0, "Worst");
    map.put(1, "Bad");
    map.put(2, "Weak");
    map.put(3, "Good");
    map.put(4, "Strong");

    return "" + strength_levels.getScore() + ", " + map.get(strength_levels.getScore()) + ", " +
strength_levels.getFeedback();
}

public static void main(String[] args) {
    //example password
    String password = "Abcdefghi1.";
    System.out.println("composition: " + composition(password));
    System.out.println("check_strength: " + check_strength(password));
}
```

Q46. Do you think the implementation of this code snippet is up-to-date and secure?

○ Yes

● No (Please specify)

```
String password = "Abcdefghi1.";

It's good to leave it as hardcoded as an example. Use secret management
software in a production environment to safely store your passwords.

https://www.vaultproject.io/
```

○

I am unsure (Please specify)

Q47. Would you make any other changes to this code snippet?

○ No

● Yes (Please specify)

```
String password = "Abcdefghi1.";

password: $Stored_password
```

Q158. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

○ No

● Yes (Please specify)

```
https://www.vaultproject.io/docs

https://www.tabnine.com/code/java/classes/com.nulabinc.zxcvbn.Zxcvbn
```

Q118. **Plain Java**

This code snippet shows an implementation of two factor authentication.

- It uses the library otp
- It generates and verifies a totp
- It creates a provisioning uri for the user

Q48. **maven**

```
<dependency>
    <groupId>j256.two-factor-auth</groupId>
    <artifactId>two-factor-auth</artifactId>
    <version>1.3</version>
</dependency>
```

```
public static int generate_second_factor(String shared_secret) throws Exception{
    return TimeBasedOneTimePasswordUtil.generateCurrentNumber(shared_secret);
}

public static String generate_uri(String keyId, String shared_secret) {
    return TimeBasedOneTimePasswordUtil.generateOtpAuthUrl(keyId, shared_secret);
}

public static void main(String[] args) throws Exception {
    String shared_secret = TimeBasedOneTimePasswordUtil.generateBase32Secret();
    int second_factor_code = generate_second_factor(shared_secret);
    System.out.println(TimeBasedOneTimePasswordUtil.validateCurrentNumber(shared_secret,
second_factor_code, 10));
    String keyId = "alice"; String imageURL = TimeBasedOneTimePasswordUtil.qrImageUrl(keyId,
shared_secret);
    System.out.println(generate_uri(keyId, shared_secret));
}
```

Q49. Do you think the implementation of this code snippet is up-to-date and secure?

○ Yes

○ No (Please specify)

●

I am unsure (Please specify)

We have to store securely secret-key to the database, we can also use Vault to improve the security storage mechanism

Q50. Would you make any other changes to this code snippet?

● No

○ Yes (Please specify)

Q141. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

○ No

● Yes (Please specify)

https://github.com/j256/two-factor-auth

Q119. **Javascript**

This code snippet implements password hashing for safe storage.

- It uses Argon2 as hashing algorithm
- The method hash_password is used to hash and salt a password
- The method verify_password is used to check whether a password matches a hash

This question was not displayed to the respondent.

Q51. **brew install gcc**
npm install -g node-gyp CXX=g++-9
npm install argon2

```
const argon2 = require('argon2');

async function hash_password(password) {
  try {
    return await argon2.hash(password, {type: argon2.argon2id});
  } catch (err) {
    console.log("error1: " + err)
  }
}

async function verify_password(pw_hash, password) {
  try {
    if (await argon2.verify(pw_hash, password)) {
      return true;
    } else {
      return false;
    }
  } catch (err) {
    console.log("error2: " + err)
  }
}

async function main() {
  //example password
  const hash = await hash_password("s3cr3t");
  console.log("hash: " + hash);

  console.log(await verify_password(hash, "s3cr3t"));
  console.log(await verify_password(hash, "s3cr4t"));
}

main();
```

This question was not displayed to the respondent.

Q52. Do you think the implementation of **hash_password** is up-to-date and secure?

This question was not displayed to the respondent.

Q53. Do you think the implementation of **verify_password** is up-to-date and secure?

This question was not displayed to the respondent.

Q54. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q142. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q120. **Javascript**

This code snippet implements password hashing for safe storage.
- It uses Bcrypt as hashing algorithm
- The method hash_password is used to hash and salt a password
- The method verify_password is used to check whether a password matches a hash

This question was not displayed to the respondent.

Q78. **npm install bcrypt**

```
const bcrypt = require('bcrypt');

async function hash_password(password) {
  const saltRounds = 10;
  try {
    return await bcrypt.hash(password, saltRounds);
  } catch (err) {
    console.log("error1: " + err);
  }
}

async function verify_password(pw_hash, password) {
  try {
    if (await bcrypt.compare(password, pw_hash)) {
      return true;
    } else {
      return false;
    }
  } catch (err) {
    console.log("error2: " + err);
  }
}

async function main() {
  //example password
  const hash = await hash_password("s3cr3t");
  console.log("hash: " + hash);

  console.log(await verify_password(hash, "s3cr3t"));
  console.log(await verify_password(hash, "s3cr4t"));
}

main();
```

This question was not displayed to the respondent.

Q81. Do you think the implementation of **hash_password** is up-to-date and secure?

This question was not displayed to the respondent.

Q80. Do you think the implementation of **verify_password** is up-to-date and secure?

This question was not displayed to the respondent.

Q79. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q143. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q121. **Javascript**

This code snippet shows an implementation of a password policy check.
- The method composition uses regexes to verify that the password meets composition criteria (at least one number, one special character, one capital letter and length between 8 and 64)
- The method check_strength uses the library zxcvbn to check a password's strength

This question was not displayed to the respondent.

Q55. **npm install zxcvbn**

```
function composition(password) {
  var number = /(?=.*\d)/;
  var upper_lower = /(?=.*[a-z])(?=.*[A-Z])/;
  var special = /[^A-Za-z0-9]/;
  var length = /.{8,64}$/;

  return [number.test(password), upper_lower.test(password), special.test(password), length.test(password)];
}

function check_strength(password) {
  var zxcvbn = require('zxcvbn');
  var result = zxcvbn(password);
  var strength = { 0: "Worst", 1: "Bad", 2: "Weak", 3: "Good", 4: "Strong" }

return [result.score, strength[result.score], result.feedback.warning, result.feedback.suggestions];
}

//example password
var password = 'Abcdefghi1.';
console.log('composition: ' + composition(password));
console.log('check_strength: ' + check_strength(password));
```

This question was not displayed to the respondent.

Q56. Do you think the implementation of this code snippet is up-to-date and secure?

This question was not displayed to the respondent.

Q57. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q144. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q122. **Javascript**

This code snippet shows an implementation of two factor authentication.

- It uses the library otp
- It generates and verifies a totp
- It creates a provisioning uri for the user

This question was not displayed to the respondent.

Q55. **npm i totp-generator**

```
const OTPAuth = require('otpauth');

function generate_second_factor(shared_features) {
  return new OTPAuth.TOTP(shared_features);
}

function generate_token_delta(second_factor) {
  let token = second_factor.generate();
  let delta = second_factor.validate({ token: token, window: 1 });

  return delta;
}

function generate_uri(second_factor) {
  let uri = second_factor.toString();
  let parsedTotp = OTPAuth.URI.parse(uri);

  return parsedTotp;
}

function main() {
  let second_factor = generate_second_factor({ issuer: 'Your Secure App', label: 'alice@gmail.com', });
  let delta = generate_token_delta(second_factor);
  if (delta == 0) {
    console.log("delta: " + true);
  } else {
    console.log("delta: " + false);
  }

  let parsedTotp_uri = generate_uri(second_factor);
  console.log("parsedTotp_uri: " + parsedTotp_uri);
}
```

```
main();
```

This question was not displayed to the respondent.

Q53. Do you think the implementation of this code snippet is up-to-date and secure?

This question was not displayed to the respondent.

Q54. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q145. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q123. **PHP**

This code snippet implements password hashing for safe storage.

- It uses Argon2 as hashing algorithm
- The method hash_password is used to hash and salt a password
- The method verify is used to check whether a password matches a hash

This question was not displayed to the respondent.

Q77.
```
function hash_password(&$password) {
    return password_hash($password, PASSWORD_ARGON2ID);
}

function verify(&$password, &$pw_hash) {
    if (password_verify($password, $pw_hash)) {
        return 'true';
    } else {
        return 'false';
    }
}

#example password
$password = 's3cr3t';
echo 'hash: ', $hash = hash_password($password), "\r\n";

$password1 = 's3cr3t';
$password2 = 's3cr4t';

echo 'verify: ', $verify = verify($password1, $hash), "\r\n";
echo 'verify: ', $verify = verify($password2, $hash), "\r\n";
```

This question was not displayed to the respondent.

Q76. Do you think the implementation of **hash_password** is up-to-date and secure?

This question was not displayed to the respondent.

Q75. Do you think the implementation of **verify** is up-to-date and secure?

This question was not displayed to the respondent.

Q74. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q146. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q124. **PHP**

This code snippet implements password hashing for safe storage.

- It uses Bcrypt as hashing algorithm
- The method hash_password is used to hash and salt a password
- The method verify is used to check whether a password matches a hash

This question was not displayed to the respondent.

Q73.

```
function hash_password(&$password) {
    return password_hash($password, PASSWORD_BCRYPT);
}

function verify(&$password, &$pw_hash) {
    if (password_verify($password, $pw_hash)) {
        return 'true';
    } else {
        return 'false';
    }
}

#example password
$password = 's3cr3t';
echo 'hash: ', $hash = hash_password($password), "\r\n";

$password1 = 's3cr3t';
$password2 = 's3cr4t';
echo 'verify: ', $verify = verify($password1, $hash), "\r\n";
echo 'verify: ', $verify = verify($password2, $hash), "\r\n";
```

This question was not displayed to the respondent.

Q72. Do you think the implementation of **hash_password** is up-to-date and secure?

This question was not displayed to the respondent.

Q71. Do you think the implementation of **verify** is up-to-date and secure?

This question was not displayed to the respondent.

Q70. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q147. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q125. **PHP**

This code snippet shows an implementation of a password policy check.
 - The method composition uses regexes to verify that the password meets composition criteria (at least one number, one special character, one capital letter and length between 8 and 64)
 - The method check_strength uses the library zxcvbn to check a password's strength

This question was not displayed to the respondent.

Q69. **composer require bjeavons/zxcvbn-php**

This question was not displayed to the respondent.

Q108. Click to write the question text

This question was not displayed to the respondent.

Q62. Do you think the implementation of this code snippet is up-to-date and secure?

This question was not displayed to the respondent.

Q61. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q148. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q126. **PHP**

This code snippet shows an implementation of two factor authentication.
 - It uses the library otp
 - It generates and verifies a totp
 - It creates a provisioning uri for the user

This question was not displayed to the respondent.

Q58. **composer require pragmarx/google2fa**
composer require bacon/bacon-qr-code

```php
require_once(__DIR__ . '/vendor/autoload.php');

# Create the 2FA class
$google2fa = new PragmaRX\Google2FA\Google2FA();

function generate_second_factor(&$google2fa) {
    return $google2fa->generateSecretKey();
}

function generate_uri(&$google2fa, &$name, &$user_mail, &$second_factor) {
    return $google2fa->getQRCodeUrl( $name, $user_mail, $second_factor );
}


$user->shared_secret = generate_second_factor($google2fa);
print $user->shared_secret;
print "\r\n";

$second_factor_now = $user->shared_secret;
print $second_factor_now;
print "\r\n";

$valid = $google2fa->verifyKey($user->shared_secret, $second_factor_now);
echo 'verify: ', ($valid) ? "true": "false", "\r\n";

$name = "alice";
$user_mail = "alice@google.com";
$second_factor = $user->shared_secret;

$qrCodeUrl = generate_uri($google2fa, $name, $user_mail, $second_factor);
print $qrCodeUrl;
```

This question was not displayed to the respondent.

Q59. Do you think the implementation of this code snippet is up-to-date and secure?

This question was not displayed to the respondent.

Q60. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q149. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q127. **GoLang**

This code snippet implements password hashing for safe storage.
- It uses Argon2 as hashing algorithm
- The method hash_password is used to hash and salt a password
- The method verify is used to check whether a password matches a hash

This question was not displayed to the respondent.

Q82. **import project from https://github.com/alexedwards/argon2id**
create package named 'main'

```
package main

import (
    "fmt"
    "log"
    "github.com/alexedwards/argon2id"
)

func hash_password(password string) string {
    hash, err := argon2id.CreateHash(password, argon2id.DefaultParams)
    if err != nil {
        log.Fatal(err)
    }

    return hash
}

func verify(pw_hash string, password string) bool {
    match, err := argon2id.ComparePasswordAndHash(password, pw_hash)
    if err != nil {
        log.Fatal(err)
    }

    return match
}

func main() {
    //example password
    hash := "" + hash_password("s3cr3t")
    fmt.Println("hash: ", hash)

    fmt.Println(verify(hash, "s3cr3t"))
    fmt.Println(verify(hash, "s3cr4t"))
}
```

This question was not displayed to the respondent.

Q92. Do you think the implementation of **hash_password** is up-to-date and secure?

This question was not displayed to the respondent.

Q94. Do you think the implementation of **verify** is up-to-date and secure?

This question was not displayed to the respondent.

Q89. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q150. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q128. **GoLang**

This code snippet implements password hashing for safe storage.
- It uses Bcrypt as hashing algorithm
- The method hash_password is used to hash and salt a password
- The method verify is used to check whether a password matches a hash

This question was not displayed to the respondent.

Q83. **import project from https://pkg.go.dev/golang.org/x/crypto/bcrypt**
create package named 'main'

```
package main

import (
    "fmt"
    "log"
    "golang.org/x/crypto/bcrypt"
)

func hash_password(password string) string {
    hash, err := bcrypt.GenerateFromPassword([]byte(password), 14)
    if err != nil {
        log.Fatal(err)
    }

    return string(hash)
}

func verify(pw_hash string, password string) bool {
    err := bcrypt.CompareHashAndPassword([]byte(pw_hash), []byte(password))

    return err == nil
}

func main() {
    //example password
    hash := "" + hash_password("s3cr3t")
    fmt.Println("hash: ", hash)

    fmt.Println(verify(hash, "s3cr3t"))
    fmt.Println(verify(hash, "s3cr4t"))
}
```

This question was not displayed to the respondent.

Q93. Do you think the implementation of **hash_password** is up-to-date and secure?

This question was not displayed to the respondent.

Q95. Do you think the implementation of **verify** is up-to-date and secure?

This question was not displayed to the respondent.

Q88. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q151. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q129. **GoLang**

This code snippet shows an implementation of a password policy check.
    - The method composition uses regexes to verify that the password meets composition criteria (at least one number, one special character, one capital letter and length between 8 and 64)
    - The method check_strength uses the library zxcvbn to check a password's strength

This question was not displayed to the respondent.

Q84. **sudo apt install golang-github-nbutton23-zxcvbn-go-dev**

```
import (
    "fmt"
    "regexp"
    "github.com/nbutton23/zxcvbn-go"
)

func composition(password string) string {
    number, _ := regexp.MatchString(".*[0-9]", password)
    upper_lower, _ := regexp.MatchString(".*[a-zA-Z]", password)
    special, _ := regexp.MatchString(".*[^\\d\\w]", password)
    length, _ := regexp.MatchString(".{8,64}", password)

    return fmt.Sprintf("%t, %t, %t, %t", number, upper_lower, special, length)
}

func check_strength(password string) string {
    result := zxcvbn.PasswordStrength(password, nil)
    strength := map[interface{}]interface{}{ 0: "Worst", 1: "Bad", 2: "Weak", 3: "Good", 4: "Strong", }

    return fmt.Sprintf("%v, %v", result.Score, strength[result.Score])
}
```

```
func main() {
    //example password
    var password string = "Abeeqfghi1."

    fmt.Println("composition: ", composition(password))
    fmt.Println("check_strength: ", check_strength(password))
}
```

This question was not displayed to the respondent.

Q91. Do you think the implementation of this code snippet is up-to-date and secure?

This question was not displayed to the respondent.

Q87. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q152. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q130. **GoLang**

This code snippet shows an implementation of two factor authentication.

- It uses the library otp
- It generates and verifies a totp
- It creates a provisioning uri for the user

This question was not displayed to the respondent.

Q85. **import project from https://github.com/pquerna/otp**
create package named 'main'

```
import (
    "github.com/pquerna/otp/totp"
    "fmt"
)

func main() {
    key, err := totp.Generate(totp.GenerateOpts{ Issuer: "Your Secure App", AccountName:
"alice@gmail.com", })
    if err != nil {
        panic(err)
    }

    passcode := key.Secret() //give your passcode from user instead of key.Secret()
    valid := totp.Validate(passcode, key.Secret())

    if valid {
        println("true")
    } else {
        println("false")
    }

    fmt.Printf("%s\n", key.URL())
}
```

This question was not displayed to the respondent.

Q90. Do you think the implementation of this code snippet is up-to-date and secure?

This question was not displayed to the respondent.

Q86. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q153. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q131. **C#**

This code snippet implements password hashing for safe storage.

- It uses Argon2 as hashing algorithm
- The method hash_password is used to hash and salt a password
- The method verify is used to check whether a password matches a hash

This question was not displayed to the respondent.

Q96. **dotnet add package Konscious.Security.Cryptography.Argon2 --version 1.0.9**

```
using System;
using System.Security.Cryptography;
using Konscious.Security.Cryptography;
using System.Text;

private static byte[] CreateSalt() {
    var buffer = new byte[128];
    var generator = RandomNumberGenerator.Create();
    generator.GetBytes(buffer);

    return buffer;
}

private static byte[] hash_password(string password, byte[] salt) {
    var argon2 = new Argon2id(Encoding.UTF8.GetBytes(password));
    argon2.DegreeOfParallelism = 16;
    argon2.MemorySize = 8192;
    argon2.Iterations = 40;
    argon2.Salt = salt;

    return argon2.GetBytes(128);
}

private static bool verify(byte[] pw_hash, string password, byte[] salt) {
    var new_hash = hash_password(password, salt);

    return pw_hash.SequenceEqual(new_hash);
}

public static void Main() {
    var salt = CreateSalt();
    //example password
    var pw_hash = hash_password("s3cr3t", salt);
    Console.WriteLine($"hash: '{ Convert.ToBase64String(pw_hash) }'.");

    var result1 = verify(pw_hash, "s3cr3t", salt);
    Console.WriteLine(result1 ? "True!" : "False!");

    var result2 = verify(pw_hash, "s3cr4t", salt);
    Console.WriteLine(result2 ? "True!" : "False!");
}
```

This question was not displayed to the respondent.

Q104. Do you think the implementation of **hash_password** is up-to-date and secure?

This question was not displayed to the respondent.

Q105. Do you think the implementation of **verify** is up-to-date and secure?

This question was not displayed to the respondent.

Q106. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q154. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q132. **C#**

This code snippet implements password hashing for safe storage.

- It uses Bcrypt as hashing algorithm
- The method hash_password is used to hash and salt a password
- The method verify is used to check whether a password matches a hash

This question was not displayed to the respondent.

Q97. **dotnet add package BCrypt.Net-Next --version 4.0.2**

```
using System;
using BCryptNet = BCrypt.Net.BCrypt;

private static string hash_password(string password) {
    string pw_hash = BCryptNet.HashPassword(password);

    return pw_hash;
}

private static bool verify(string pw_hash, string password) {
    return BCryptNet.Verify(password, pw_hash);
}

public static void Main() {
    //example password
    string pw_hash = hash_password("s3cr3t");
    Console.WriteLine(pw_hash);

    var result1 = verify(pw_hash, "s3cr3t");
    Console.WriteLine(result1);
    var result2 = verify(pw_hash, "s3cr4t");
    Console.WriteLine(result2);
}
```

This question was not displayed to the respondent.

Q102. Do you think the implementation of **hash_password** is up-to-date and secure?

This question was not displayed to the respondent.

Q103. Do you think the implementation of **verify** is up-to-date and secure?

This question was not displayed to the respondent.

Q107. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q155. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q133. **C#**

This code snippet shows an implementation of a password policy check.

- The method composition uses regexes to verify that the password meets composition criteria (at least one number, one special character, one capital letter and length between 8 and 64)
- The method check_strength uses the library zxcvbn to check a password's strength

This question was not displayed to the respondent.

Q98. **dotnet add package zxcvbn-core --version 7.0.92**

```
using System;
using System.Text.RegularExpressions;
using Zxcvbn;

private static string composition(string password) {
    Regex number = new Regex(@"(?=.*\d)");
    Regex upper_lower = new Regex(@"(?=.*[a-z])(?=.*[A-Z])");
    Regex special = new Regex(@"[^A-Za-z0-9]");
    Regex length = new Regex(@".{8,64}$");

    Match match1 = number.Match(password);
    Match match2 = upper_lower.Match(password);
    Match match3 = special.Match(password);
    Match match4 = length.Match(password);

    bool[] test = new bool[4];
    test[0] = match1.Success;
    test[1] = match2.Success;
    test[2] = match3.Success;
    test[3] = match4.Success;

    return "" + test[0] + ", " + test[1] + ", " + test[2] + ", " + test[3];
}

private static string check_strength(string password) {
    var result = Zxcvbn.Core.EvaluatePassword(password);
    Dictionary strength = new Dictionary();
    strength.Add(0, "Worst");
    strength.Add(1, "Bad");
    strength.Add(2, "Weak");
    strength.Add(3, "Good");
    strength.Add(4, "Strong");

    string last = "" + result.Score + ", " + strength.FirstOrDefault(x => x.Key == result.Score).Value + ", " +
result.Feedback.Warning + ", " + result.Feedback.Suggestions;

    return last;
}

public static void Main() {
    //example password
    string password = "Abcdefghi1.";

    string a = composition(password);
    Console.WriteLine(a);

    string b = check_strength(password);
    Console.WriteLine(b);
}
```

This question was not displayed to the respondent.

Q101. Do you think the implementation of this code snippet is up-to-date and secure?

This question was not displayed to the respondent.

Q108. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q156. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q134. **C#**

This code snippet shows an implementation of two factor authentication.
- It uses the google authenticator library
- It generates and verifies a totp
- It creates a provisioning uri for the user

This question was not displayed to the respondent.

Q99. **install .NET 5.0 or lower**
sudo apt-get install -y libgdiplus
dotnet add package GoogleAuthenticator --version 2.4.0

```
using System;
using Google.Authenticator;

static void Main(string[] args) {
    string key = Guid.NewGuid().ToString().Replace("-", "").Substring(0, 10);
    string issuer = "Your Secure App"; string accountTitle = "alice@gmail.com";

    TwoFactorAuthenticator tfa = new TwoFactorAuthenticator();
    SetupCode setupInfo = tfa.GenerateSetupCode(issuer, accountTitle, key, false, 3);

    string second_factor_now = tfa.GetCurrentPIN(key, false);

    Console.Write(tfa.ValidateTwoFactorPIN(key, second_factor_now) + "\r\n");

    var provisionUrl = string.IsNullOrWhiteSpace(issuer) ? $"otpauth://totp/{accountTitle}?secret={key.Trim('=')}" : $"otpauth://totp/{Uri.EscapeDataString(issuer)}:{accountTitle}?secret={key.Trim('=')}&issuer={Uri.EscapeDataString(issuer)}";

    Console.Write(provisionUrl + "\r\n");
}
```

This question was not displayed to the respondent.

Q100. Do you think the implementation of this code snippet is up-to-date and secure?

This question was not displayed to the respondent.

Q109. Would you make any other changes to this code snippet?

This question was not displayed to the respondent.

Q157. Did you use any additional resources while checking the code? If yes, please provide a link or description of your resource.

This question was not displayed to the respondent.

Q109. Do you have any other suggestions, questions or comments?

As a sample, it looks like libraries are securely implemented. The problem comes after when you use to store the password. You must avoid plain hardcoded secrets and use secret management software to keep critical things in it. Although, it has to be checked regularly library version to affect new vulnerabilities, for instance, password guesses and brute-force attacks. In an advanced password policy, you can use the most common password list to make users aware of how simple it is. If the user chooses passwords like 'Passw0rd!' it's convenient to your policy, but this type of password can easily guess by attackers. Recently, companies have used the method to avoid easy passwords. Please, pay attention to the below link to know further about it. https://docs.microsoft.com/en-us/azure/active-directory/authentication/concept-password-ban-bad#global-banned-password-list

Embedded Data

Pseudonym: 8918871171

Location Data

Location: (40.3909, 49.8759)

Source: GeoIP Estimation