Why character array is better than String for Storing password in Java

Why character array is better than String for storing password in Java was recent question asked to one of my friend in a java interview. he was interviewing for a Technical lead position and has over 6 years of experience.Both Character array and String can be used to store text data but choosing one over other is difficult question if you haven't faced the situation already. But as my friend said any question related to `String` must have a clue on special property of `Strings` like immutability and he used that to convince interviewer. here we will explore some reasons on why should you used `char[]` for storing password than String.

This article is in continuation of my earlier interview question post on String e.g. Why String is immutable in Java or How Substring can cause memory leak in Java, if you haven't read those you may find them interesting.Here are few reasons which makes sense to believe that character array is better choice for storing password in Java than String:

1) Since **Strings are immutable in Java** if you store password as plain text it will be available in memory until Garbage collector clears it and since `String` are used in String pool for reusability there is pretty high chance that it will be remain in memory for long duration, which pose a security threat. Since any one who has access to memory dump can find the password in clear text and that's another reason you should always used an encrypted password than plain text. Since Strings are immutable there is no way contents of Strings can be changed because any change will produce new String, while if you char[] you can still set all his element as blank or zero. So **Storing password in character array clearly mitigates security risk of stealing password**.

2) **Java itself recommends** using `getPassword()` method of `JPasswordField` which returns a `char[] and` deprecated `getText()` method which returns password in clear text stating security reason. Its good to follow advice from Java team and adhering to standard rather than going against it.

3) With String there is always a risk of printing plain text in log file or console but if use Array you won't print contents of array instead its memory location get printed. though not a real reason but still make sense.

```
String strPassword="Unknown";
char[] charPassword= new char[]{'U','n','k','w','o','n'};
System.out.println("String password: " + strPassword);
System.out.println("Character password: " + charPassword);

String password: Unknown
Character password: [C@110b053
```

That's all on *why character array is better choice than String for storing passwords in Java*. Though using `char[]` is not just enough you need to erase content to be more secure. I also suggest working with hash'd or encrypted password instead of plain text and clearing it from memory as soon as authentication is completed.

Other **Java Interview questions** you may like

Why multiple inheritance is not supported in Java
Why wait and notify are defined in Object class in java
Difference between Runnable and Thread in java
Why float and double are not suitable for storing monetary value
Why main method is public static in java
Why do you need marker interface in Java
Why Java doesn't support operator overloading in Java
Difference between ArrayList and LinkedList in Java

Posted by Javin Paul

Labels: Array , core java , core java interview question , data structure and algorithm
Location: United States

Menu
About Me
core java tutorial
data structure and algorithm
java interview questions
coding interview questions
java certification resources
java multithreading tutorials
best practices
java collection tutorials
Privacy Policy

Categories
programming (230)
coding (88)
interview questions (42)
jsp-servlet (33)
database (28)
linux (28)
Eclipse (19)
Java 8 (19)
design patterns (18)
books (14)
JDBC (13)
Java xml tutorial (9)

Recommended Best Practices
3 Method and Constructor overloading best practices
How string in switch case internally works in Java 7?
Top 10 Programming Best Practices to name variables and methods
How SSL, HTTPS and Certificates works in Java Application?
7 tips while dealing with password in Java?
Introduction of How Android works for Java Programmers
How substring() method works in Java?

Follow by Email
Email address... Submit

Blog Archive
► 2015 (119)
► 2014 (106)
► 2013 (136)
▼ 2012 (217)
► December (52)
► November (8)
► October (14)

15 comments :

sarat said...

Added to the above reasons Class String is also Serializable. So by using character array we also avoid the risk of serializing the password.

March 26, 2012 at 10:13 PM

Javin @ Java String replace Example said...

well said sarat, Indeed a worth noting point.

March 27, 2012 at 6:12 AM

Craig said...

Using char[] instead of String for passwords is (IMHO) a bad idea for a few reasons:

1. That's going to massively complicate things. Strings maintain information about character encodings - char[] (obviously) does not. So if you use char[], you have to be very careful to always use the same encoding. This is critical if you support non-US English users (and you should always do that!).

2. If a malicious user gets access to read core dump files in /tmp, it's game over anyways. At that point, it means that not only has a malicious user gained access to your server, but he's also gained either root or application user access - so he's fully compromised the system.

3. char[] are not pinned to RAM. The OS swaps char[] data just as easily as String data, writing it to disk in the swap file.

4. Strings may be immutable, making it more likely that they hang around for a while in memory due to string internment, but char[]'s aren't guaranteed to be removed from memory either due to the uncertainties of garbage collections. You can never be sure any memory is cleared in Java, nor can you tell Java to clear any memory.

Using char[] instead of String makes your code buggier, harder to maintain, harder to write, and only more secure in a totally impractical sense.

June 11, 2012 at 9:36 AM

Jun kun said...

Now I am confused between char[] and String to store password ? Which one I should use char[] or String ?

January 9, 2013 at 2:02 AM

Unknown said...

This is awesome. Points noted and very mind blowing. Java needs further reading

January 10, 2013 at 1:46 PM

Dave Conrad said...

There are a few mistakes here. First, there is no real "danger" of a password String being logged, since it would only happen if you explicitly log it.

Second, you can easily print the contents of an array using Arrays.toString(), so if you're afraid your fingers will type logger.info(password), well, they might also accidentally type logger.info(Arrays.toString(password)).

Finally, there is no safety from serialization using an array instead of a String. Yes, it's true that Strings are Serializable; so are arrays.

January 24, 2013 at 12:41 PM

Anonymous said...

System.out.println(charPassword);

The above line prints password in clear text.

February 20, 2013 at 1:59 PM

Anonymous said...

```
public class printPwd {
public static void main(String[] args) {
String strPassword = "Unknown";
char[] charPassword = new char[] { 'U', 'n', 'k', 'w', 'o', 'n' };
System.out.println("String password: " + strPassword);
System.out.println("Character password: " + String.valueOf(charPassword));
}
}
```

surprize output is
String password: Unknown
Character password: Unkwon

March 11, 2013 at 11:09 AM

Anonymous said...

Strings don't keep any information about encoding -- they just have a char[], and a char is a unicode code point in utf-16. You can create a String from a byte[] and an encoding, and given a String you can get a byte[] if you give it an encoding -- but the String itself has no encoding other than the utf-16 of its chars.

April 23, 2013 at 7:14 PM

The One said...

There is API provided by Java itself to get the characters in an array. Please have a look at below code snippet, there is no use of array over string other in this scenario but memory wise yes it is:
System.out.println("Character password: " + Arrays.toString(charPassword));

Character password: ['U','n','k','w','o','n']

September 12, 2013 at 12:41 AM

Anonymous said...

As you said that if someone who have acess to memory dump can know the passwords if it is string form,and in char array it will be known only the address, As he know the memory dump Wont he able to find the passsowrd using that address which he knows.

April 9, 2014 at 10:33 PM

Unknown said...

"String are used in String pool for reusability" is generally not correct in this case.

Only certain types of strings are interned in the permanent generation. Strings created dynamically at runtime, i.e. strings in a HTTP request, are not. Consider this: if every String pulled in a HTTP request was interned, you'd consume all of the permanent gen quickly.

This may be different for nonstandard JVMs, but for the Oracle VM this is correct.

May 15, 2014 at 2:34 AM

Tushar jain said...

Hi.Can u plz explain us the below code more Clearly.

char[] charPassword = new char[] { 'U', 'n', 'k', 'w', 'o', 'n' };
Character password: [C@110b053

How is this output possible in case of Char Array.??

November 27, 2014 at 12:36 AM

Anonymous said...

Why would you store the password in the first place!!! Its a bad design in itself IMHO.
Defer it to authentication system (webserver/ldap etc) or store it's hash or in any other encrypted form.

Then you wouldn't have to care about char[] or String, i personally do not see any threat of using String.

Someone, who can access memory dump, could as well run remote debug / implement code manipulation APIs to view passwords right on screen.

Design your system well.

December 21, 2014 at 8:30 PM

Dave Cronin said...

In reply to Craig points, about the use of char[] being a bad idea, I would reply
1. yes, using char can complicate things, but then a password is just a sequence of characters, so a char array is suitable to store a password.
2 and 3. the char array data could indeed get into a core dump, or be swapped to virtual memory. However, the password will not be accessible in RAM, so it does reduce the risk. Also, by using a char array, you can delete the password data immediately, before the garbage collector removes the array object.
4. True, the char[] array object would hang around in memory until removed by the garbage collector. However, the point is that the data inside the array should be explicitly cleared out, for example

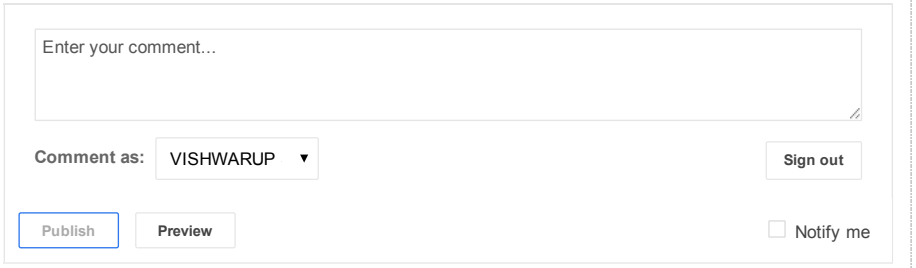
// password data
char [] charPassword = {'s', 'e', 'c', 'r', 'e', 't'};

// empty the password array
Arrays.fill(charPassword, '\0');

Also it is worth noting that the Swing class JPasswordField has deprecated the getText method using String, for security reasons. The new method getPassword instead returns a char array.

September 21, 2015 at 5:12 AM

Post a Comment



Newer Post Home Older Post

Subscribe to: Post Comments (Atom)