Categorized Questions

Here we present the full list of questions we identified and categorized. The *#Y.X* notation corresponds with the tags used in strategy and assumption analysis.

Developer Planning and Self-Reflection #1.X

What was I looking for? #1.1

What do I know now? #1.2

What should I do first? #1.3

Do I understand? #1.4

Have I seen this before? #1.5

What was that again? #1.6

What's the next thing I should be doing? #1.7

Where am I (in the code)? #1.8

Is this worth my time? #1.9

Is this my responsibility? #1.10

Why do I care? #1.11

Have I used this package before? #1.12

Am I making the right decision for my code? #1.13

How long have we been talking? #1.14

Vulnerability Severity and Rank #2.X

How serious is this vulnerability? #2.1

Are all these vulnerabilities the same severity? #2.2

How do the rankings compare? #2.3

What do the vulnerability rankings mean? #2.4

Data Storage and Flow #3.X

How is data put into this variable? #3.1

Does data from this method/code travel to the database? #3.2

Where does this information/data go? #3.3

How do I find where the information travels? #3.4

How does the information change as it travels through the program? #3.5

What does this variable contain? #3.6

Is any of the data malicious? #3.7

Where is the data coming from? #3.8

What are the contents of this SQL file? #3.9

Where along the data pipeline is this method? #3.10

Where along the pipeline do you want to make sure the data is secure? #3.11

Application Context/Usage #4.X

What is this method/variable used for in the program? #4.1

Are we handling secure data in this context? #4.2

What is the context of this vulnerability/code? #4.3

How does the system work? #4.4

Will usage of this method change? #4.5

Is this method/variable ever being used? #4.6

Is this code used to test the program/functionality? #4.7

Does test utils get sent to production code? #4.8

How many layers of code/security have we passed through before reaching this point? #4.9

Control Flow and Call Information #5.X

What is the call hierarchy? #5.1

How can I get calling information? #5.2

Who can call this? #5.3

Where is the method being called? #5.4

What causes this to be called? #5.5

Are all calls coming from the same class? #5.6

What gets called when this method gets called? #5.7

How often is this code called? #5.8

Is this called from another less secure API? #5.9

How frequently does this method get called? #5.10

How many calls does it take to reach the vulnerability? #5.11

What are the parameters called? #5.12

Is the entire set of possible parameters known in advance? #5.13

Resources and Documentation #6.X

What type of information does this resource link me to? #6.1

What is the documentation? #6.2

Can my team members/resources provide me with more information? #6.3

Where can I get more information? #6.4

What information is in the documentation? #6.5

Is this a reliable/trusted resource? #6.6

How do resources prevent or resolve this? #6.7

How should I word my search to get the right information? #6.8

Is this the OWASP site? #6.9

Does the method have a javadoc? #6.10

Understanding Alternative Fixes and Approaches #7.X

Why should I use this alternative method/approach to fix the vulnerability? #7.1

What are the alternatives for fixing this? #7.2

Does the alternative function the same as what I'm currently using? #7.3

When should I use the alternative? #7.4

Is the alternative slower? #7.5

Are there other considerations to make when using the alternative(s)? #7.6

How does my code compare to the alternative code in the example I found? #7.7

Is secure random good? #7.8

Is prepared statement a Java thing or a 3rd party library? #7.9

How secure is secure random? #7.10

Code Background and Functionality #8.X

Who wrote this code? #8.1

Why is this code needed? #8.2

Is this library code? #8.3

Are there tests for this code? #8.4

Why was this code written this way? #8.5

What does this code do? #8.6

Is this code doing anything? #8.7

How much effort was put into this code? #8.8

Why are we using this API in the code? #8.9

Is there a list of files we are going to iterate over? #8.10

Is rand static or final? #8.11

Why are we collecting this information in the first place? #8.12

Is there a variable called ``dir'' at the top? #8.13

How does the code accomplish design goals? #8.14

What is the name of the current class? #8.15

Could the pointer be null? #8.16

Does it throw an exception instead #8.17

Locating Information #9.X

Where is this used in the code? #9.1

Where are other similar pieces of code? #9.2

Where is this artifact? #9.3

Is this artifact located in this class? #9.4

Where is this method defined? #9.5

Where is this class? #9.6

Where is the next occurrence of this variable? #9.7

How do I track this information in the code? #9.8

How do I navigate to other open files? #9.9

Where is this class instantiated #9.10

Where was the class instantiated #9.11

Assessing the Application of the Fix #10.X

How hard is it to apply a fix to this code? #10.1

How do I use this fix in my code? #10.2

How do I fix this vulnerability? #10.3

Is there a quick fix for automatically applying a fix? #10.4

Will the code work the same after I apply the fix? #10.5

Can these fix suggestions be applied to my code? #10.6

Will the error go away when I apply this fix? #10.7

Does the code stand up to additional tests prior to/after applying the fix? #10.8

What other changes do I need to make to apply this fix? #10.9

Preventing and Understanding Potential Attacks #11.X

How can this vulnerability lead to an attack? #11.1

How can I replicate an attack that exploits this vulnerability? #11.2

Why is this a vulnerability? #11.3

What are the possible attacks that could occur? #11.4

How can I prevent this attack? #11.5

How should I address this problem? #11.6

What is the problem (potential attack)? #11.7

Is this a real vulnerability? #11.8

How do I find out if this is a real vulnerability? #11.9

How much value would you get from randomly injecting stuff? #11.10

Is releaseForms handling data securely? #11.11

Relationship Between Vulnerabilities #12.X

Does this other piece code have the same vulnerability as the code I'm working with? #12.1

Are all the vulnerabilities related in my code? #12.2

Are all of these notifications vulnerabilities? #12.3

End-User Interaction #13.X

Is there input coming from the user? #13.1

Does the user have access to this code? #13.2

Does user input get validated/sanitized? #13.3

Notification Text #14.X

What is the relatinoship between the error message and the code? #14.1

What code caused this error message to occur? #14.2

What does the error message say? #14.3

Understanding Concepts #15.X

What is the term for this concept? #15.1

Do these words have special meaning related to this concept/problem? #15.2

How does this concept work? #15.3

What is this concept? #15.4

Is Java's true/false considered enum? #15.5

What is the nomenclature for testing suites? #15.6

Confirming Expectations #16.X

Is this doing what I expect it to? #16.1

Uncategorized #17.X

Have I exhausted all references in this method? #17.1

Is this method deprecated? #17.2

What does the constructor and datatype look like? #17.3

Was the author of this code aware of security issues? #17.4

Has the package I'm using been tested? #17.5

Is it done the same way in Python? #17.6

Are there non-malicious ways the code could break? #17.7

How can I transfer text from the IDE to the browser? #17.8

What parts of the code do you trust with this data? #17.9

What do you want to trust? #17.10

Understanding and Interacting with Tools #18.X

Why is the tool complaining? #18.1

What is the tool output telling me? #18.2

Can I verify the information the tool provides? #18.3

What is the tool keybinding? #18.4

What is the tool's confidence? #18.5

What tool do I need for this? #18.6

How is the information presented by the tool organized? #18.7

What is the related stack trace for this method? #18.8

How can I annotate that these strings have been escaped and the tool should ignore the warning? #18.9

Discarded (5) #Discard