Assignment on Securing Software for CS50 Cybersecurity

1 message

Google Forms <forms-receipts-noreply@google.com>
To: ekrishnachaitanya2004@gmail.com

11 August 2024 at 10:48

Google Forms

Thanks for filling in Assignment on Securing Software for CS50 Cybersecurity

Here's what was received.

Assignment on Securing Software for CS50 Cybersecurity

All assignments in CS50 Cybersecurity are out of 10 points. A score of 7 points or better (70%) is required to be considered to have "passed" an assignment in this course. **Please do not resubmit an assignment if you have already obtained a passing score**--we consider that spam, and if detected, the submission will be deleted, meaning you will not receive the score back anyway. You don't receive a final grade at the end of the course, so it will have no bearing on your certificate, and it will only slow down our graders!

Unlike CS50x, assignments in this course are graded on a set schedule, and depending on when you submitted, it may take up to three weeks for your work to be graded. Do be patient! Project scores and assignment status on cs50.me/cs50cy (e.g. "Your submission has been received...") will likely change over time and are not final until the scores have been released.

Email *

ekrishnachaitanya2004@gmail.com

Name *

KRISHNA CHAITANYA

edX Username *

eKRISHNA2004

What is your GitHub username? *

If you do not already have a GitHub account, you can sign up for one at https://github.com/join. You can then use this account to log in to cs50.me/cs50cy to track your progress in the course (your progress will only show up after you have received at least one score release email from CS50 Bot, so do be patient!). Don't worry about seeing a 'No Submissions' message on submit.cs50.io, if you find that. The course collects submissions using Google Forms, and only the gradebook on cs50.me/cs50cy is important!

- **Be certain the username you provide is correct!** If you provide the wrong username, you will not be able to see your scores.
- **Your GitHub username should not be changed while you are taking this course.** The current gradebook system is not designed to accommodate name changes.
- **Be sure to remove extraneous characters**, such as an @ prefix. Do not input a URL or email address, just your username.

ekrishnachaitanya2004

City, State, Country *

Srikalahasthi,Andhra Pradesh,India

This course is graded by human graders, and has a ZERO TOLERANCE plagiarism and collaboration policy. If *any* of your answers are copied and pasted from, or obviously based on (a) an online source, including non-course-sanctioned generative AI tools or (b) another student's work in the course, in *any* of the course's five assignments or the final project, you will be reported to edX and removed from the course immediately. There is no opportunity for appeal. There are no warnings or second chances. *

It is far better, we assure you, to leave an answer blank rather than risk it. This may be an online course, but it is offered by Harvard, and we're going to hold you to that standard. **The full essence of all work you submit to this course should be your own.**

◉ I understand this policy and agree to its terms; I hereby affirm that I will not plagiarize any

answers in this course.

Via what technique(s) might a malicious actor "crack" software (that is, bypass registration/payment to use it)?

Keygen, Reverse engineering, Patch

Distinguish the nature of two types of "cross-site" attacks we discussed: cross-site scripting (XSS) and cross-site request forgeries (CSRF).

XSS focuses on injecting malicious code to compromise the client-side, while CSRF exploits the authenticated session to manipulate the server-side. Both attacks pose significant risks to web applications, but understanding their differences is essential for effective prevention and mitigation.

Why do we need to *escape* certain characters in inputs?

Escaping certain characters in inputs is essential to prevent injection attacks like SQL injection and cross-site scripting, which can compromise security by manipulating queries or injecting malicious scripts. It ensures safe URL handling by encoding special characters, and prevents Lightweight directory access protocol injection, ensures proper parsing in HTML and XML, maintains data integrity by preventing command injection. Overall, escaping characters ensures that user inputs are treated as data, not code, protecting the web application's functionality and security.

In the context of SQL, what is a *prepared statement*?

A prepared statement in SQL is a precompiled query that separates the SQL commands from the data values. This enhances security by preventing SQL injection attacks and can also improve the speed of query execution.

Why is client-side validation considered "less secure", perhaps, than server-side validation?

Client-side validation is less secure because it can be easily changed by the user by disabling JavaScript or manipulating code. Server-side validation occurs on the server and cannot be altered by the user. While client-side validation enhances user experience, it should never

replace server-side checks.

GeekHero Comic

Pragmatically, the above comic may be correct as it pertains to the behavior of most *users* towards open-source software. Even if that is your attitude towards it, why might it still be a good thing to consider using more open-source software (or, developing it), from a security-minded perspective?

Using and developing open-source software can improve security by allowing anyone to see and improve the code, resulting in the identification and correction of vulnerabilities. A global developer community's collective oversight facilitates continuous security enhancements and audits, ensuring robustness against threats. Furthermore, the ability to customize and adapt the software gives users greater control over their security posture, while avoiding the pitfalls of security through obscurity that may be relied on by closed-source software. Because of these collaborative and transparent development practices, open-source frequently results in more secure, trustworthy software.

How are package managers similar to app stores (such as Apple's App Store, Google Play Store, Microsoft Store, etc.), from a cybersecurity perspective?

Package managers and app stores are similar in that they both provide a centralized platform for distributing software, which helps ensure that users are downloading trusted and secure applications. Both systems often involve a vetting process to minimize the risk of malicious software and offer features like automatic updates to keep software secure. Additionally, they manage dependencies, ensuring that all necessary components are present and up to date, and they may include security scanning to check for vulnerabilities. Overall, both package managers and app stores help maintain a safer software environment for users.

What threat does use of Content-Security-Policy fields in our source code help to defend against?

Content-Security-Policy defends against cross-site scripting (XSS) and data injection attacks.

Provide a specific example of a situation when you might want to use the HTTP POST method instead of the HTTP GET method.

Password Through a login form

Heartbleed Logo

Image Source: Wikimedia

More precisely known as CVE-2014-0160 but better known as Heartbleed, the discovery of this exploit caused quite an internet panic in 2014, resulting in one of the first times a bug was actively publicized in mass media outlets, as cybersecurity researchers scrambled to make the public aware of the bug and to encourage rapid download of the fix.

Read up on Heartbleed either via the Wikipedia article linked in the previous paragraph, or via any other research you like (such as this video).

Why was Heartbleed such a threat to a user's security?

Heartbleed posed a serious threat to user security because it took advantage of a flaw in OpenSSL's implementation of the TLS heartbeat extension, allowing attackers to read memory from affected servers. Private keys, user passwords, and personal information could all be exposed as a result of this. The bug was especially dangerous because it had been present in widely used OpenSSL versions for about two years before it was discovered, and the attacks could take place undetectedly, leaving no trace. This made determining the extent of the information that may have been compromised difficult. Because OpenSSL is widely used to secure internet communications, a large portion of the web was affected, prompting urgent calls for patches, certificate renewals, and user caution.

Feedback

How did you find the difficulty of this assignment?
*

	1	2	3	4	5	
Too easy	○	○	◉	○	○	Too hard

About how many MINUTES would you say you spent on this assignment? *

Just to set expectations for future students.

300