Peer Response 1

Context: https://www.my-course.co.uk/mod/hsuforum/discuss.php?d=270870

Hi Kieron,

This is a clear and concise introduction to challenges faced when using SQL as part of a user-facing system. Prevention of these attacks is an interesting topic- many attacks rely on the usage of characters such as a single quote mark or semicolon so that extra text will be parsed as SQL query operators (w3resource, 2021). The challenge, however, is that the characters themselves may be a necessity- for example, a web application that stores blog posts made by users might need to avoid filtering quote marks and semicolons to preserve the user's input. Although sanitized queries and prepared statements eliminate this risk (as you mention), in performance-critical environments, it may be difficult to use this measure (OWASP, 2021). One interesting approach was proposed by Halfond et al. (2006), which relies on tracking the flow of untrusted inputs while factoring in the characters surrounding the suspicious text to prevent false positives. This method has a low performance overhead and the paper itself still has relevance to the discussion surrounding SQL injection prevention (Loughran et al., 2018), but based on what I could find, it doesn't seem that this strategy achieved widespread adoption. In a hypothetical scenario where a performance-critical application was being built, do you think it would be reasonable for a developer to consider this strategy, although it's not directly endorsed by OWASP?

With regards to the UML diagram, the comments make it much easier to contextualize the actions taken in the activity diagram, though it could be possible to add the comments directly to the diagram as decision nodes or individual actions- do you think this would be worthwhile?

References

Halfond, W., Orso, A. & Manolios, P. (2006) 'Using positive tainting and syntax-aware evaluation to counter SQL injection attacks.' *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering (SIGSOFT '06/FSE-14)*. Portland: Oregon, November 5-11. USA: ACM. 175–185.

Loughran, D., Salih, M. & Subburaj, V. (2018) All About SQL Injection Attacks. Journal of The Colloquium for Information System Security Education (CISSE) 6(1): 1-24.

OWASP. (2021) SQL Injection Prevention Cheat Sheet. Available from: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html [Accessed 24 August 2021].

w3resource. (2021) SQL Injection Tutorial. Available from: https://www.w3resource.com/sql/sql-injection/sql-injection.php [Accessed 24 August 2021].

Peer Response 2

Context: https://www.my-course.co.uk/mod/hsuforum/discuss.php?d=270384

Hi Andrey,

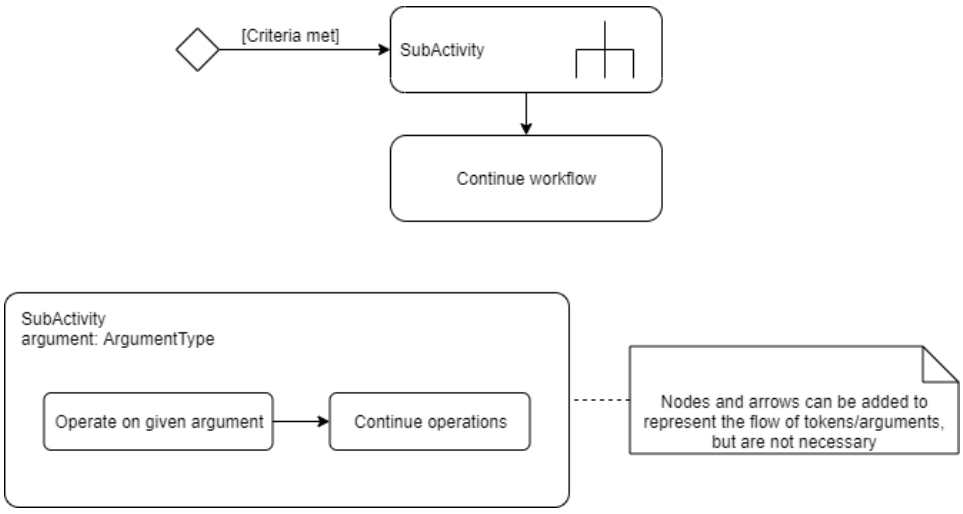
This is a really well-done activity diagram and the usage of specialized symbols makes it easy to understand the precise, intended behavior and states of the system. Considering that subactivities are represented with ExpansionNodes, it becomes easier to follow the flow of data and see behaviors from the point of view of the various actors in the system, which would make development easier (Oracle, 2007). For example, and if I understand it correctly, the IncidentResponse subactivity and the DDoSMitigation subactivity take some contextual data, and representing this flow of data is great because it clarifies the responsibilities of the subactivity within the context of the system, and makes it possible to describe behaviors at a lower level of detail while maintaining relevance to the overall diagram.

One thing I noticed is that the diagram is beginning to expand in size and is beginning to represent more complex behaviors, which might become tricky to follow when discussing specific workflows or user stories in the context of the diagram. What are your thoughts on using UML CallBehaviorActions (Object Management Group, 2015)? This notation would make it possible to represent a subactivity as a self-contained block without the need for ExpansionNodes, increasing abstraction, which has the effect of making the subactivities scalable, and would make the main diagram easier to read. A (minimal) example is attached below.

References

Object Management Group. (2015) OMG Unified Modeling Language ™ (OMG UML) Version 2.5. Available from: https://www.omg.org/spec/UML/2.5.1/PDF [Accessed 23 August 2021].

Oracle. (2007) Getting Started With Activity Modeling. Web: Oracle.



Response to Peer Response: Andrey Smirnov

Hi Shan,

Thank you for your thoughtful remarks. We are of one mind on the usefulness of Expansion Regions for modelling sub-activities or groups of actions that operate on incoming data. That said, I might have gone against the UML conventions when adding the DDoSMitigation activity, as according to my research, expansion regions are expected to have explicit outputs (Bock, 2005). I had also considered Loop Nodes before deciding on the final design. There are differing interpretations of the meaning and applicability of these nodes, and their specification in the UML standard is rather opaque (Bock, 2005). Ultimately, I do not believe in hard rules for diagramming; my personal litmus test is to see whether the intended audience understands the designer's intent. I must also note that I was not able to find up-to-date academic literature on these components.

As for your suggestion to utilize Call Activity Nodes to "detach" the detailed representation of sub-activities and unburden the overall flow, I believe that would be an excellent improvement. A few days ago I suggested the same approach to my team; one of the activity diagrams that we have been working on has also become too unwieldy.

References

Bock, C. (2005) UML 2 Activity and Action Models Part 6: Structured Activities. Journal of Object Technology 4(4): 43-66.

Störrle, H. (2004) Structured Nodes in UML 2.0 Activities. Nordic Journal of Computing 11(3): 279-302.