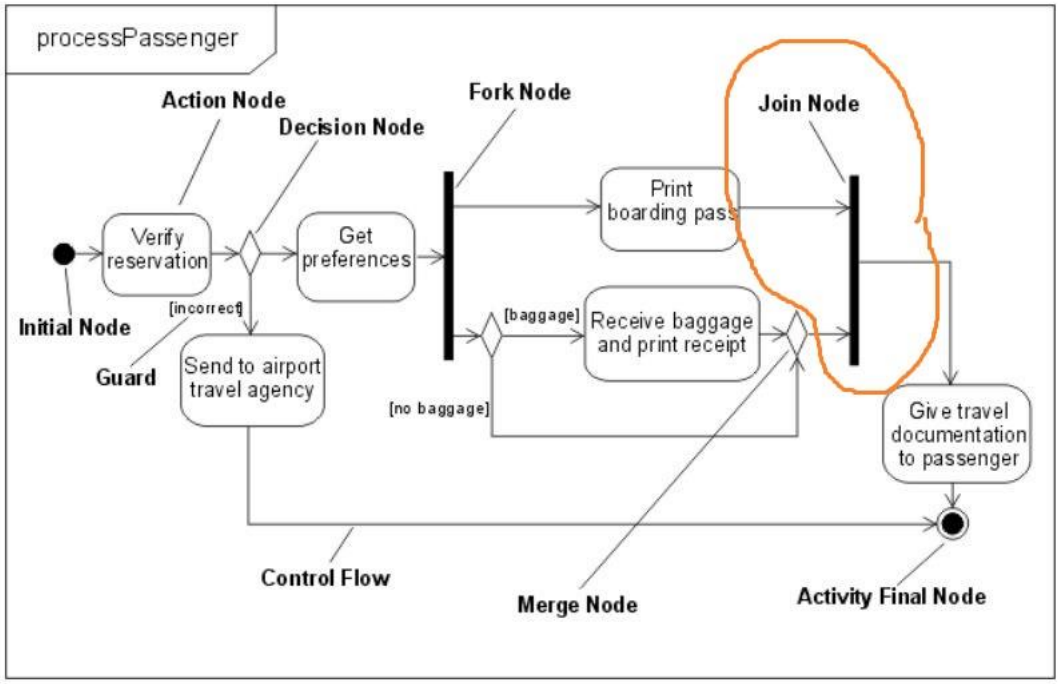
Collaborative Discussion – UML Flow Charts

Peer Responses

Contents

My Initial Post

> Great evidence of awareness of other externally-sourced material, such as CWE from Mitre. Good integration of it into this discussion.
>
> Excellent attention to detail here, Michael.
>
> One minor comment relating to the harnessing of UML: In relation to the flows into 'Persist failed authorisation result', there is an opportunity to use a UML join, which will unite these flows before they enter the activity.



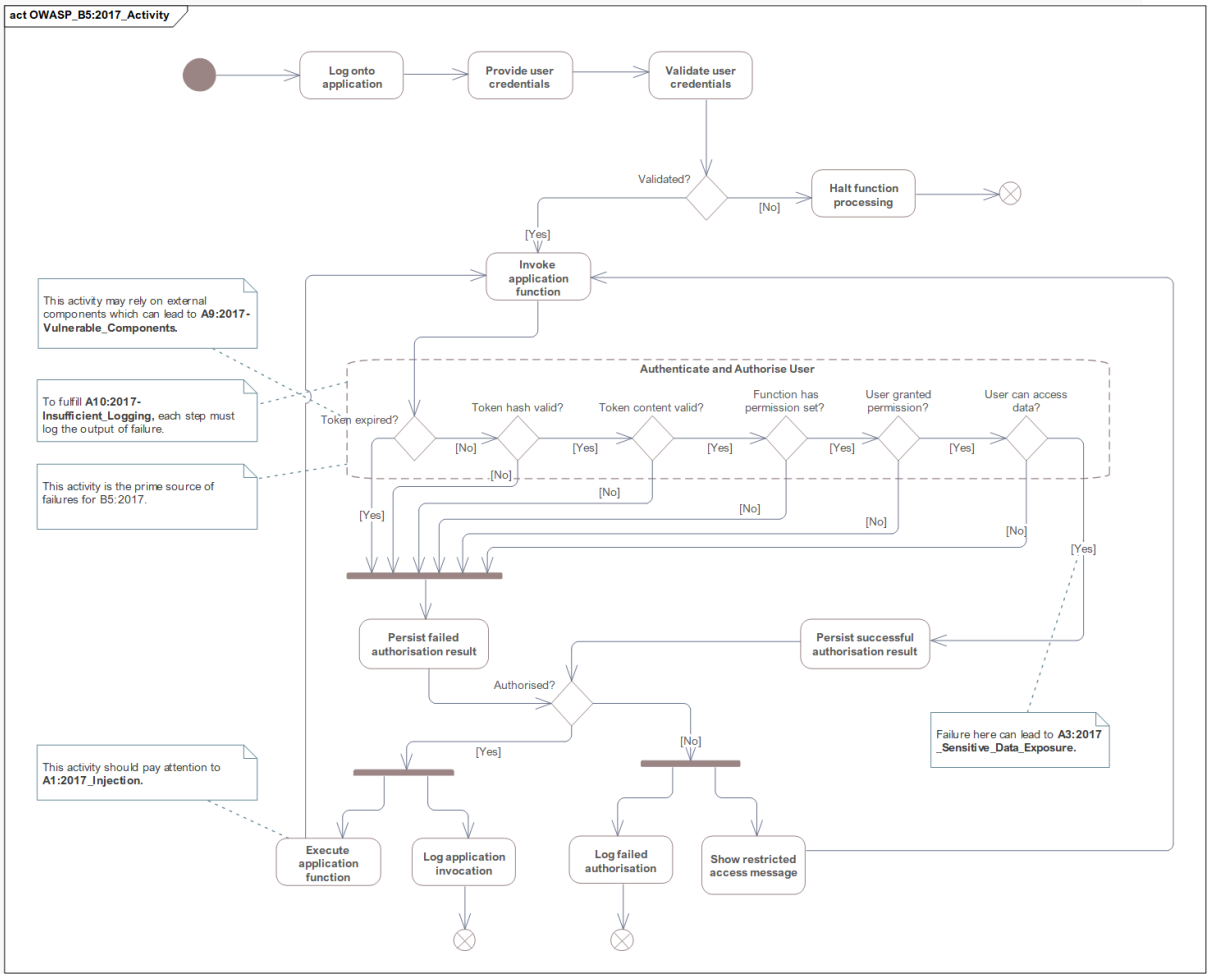
Best wishes,
Cathryn

Reply to Cathryn Peoples **from** Michael Justus

Re: Initial Post

Hi Cathryn,

Thank you very much for the valuable feedback! The use of a Join node into the "Failed Authorisation" activity was not utilised because the thinking is that a Join node joins multiple parallel flows. And, in the proposed UML flow, "Authenticate and Authorise User" is a sequential flow of activities. Therefore it was unclear if a Join node is valid. Thank you for the guidance.



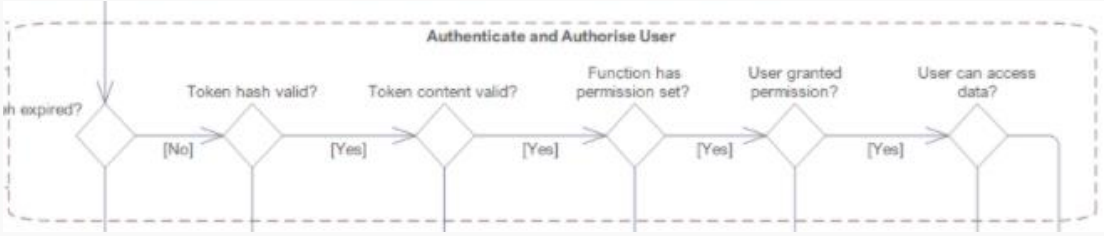
Post by Hendrik Van Rooyen

Peer Response

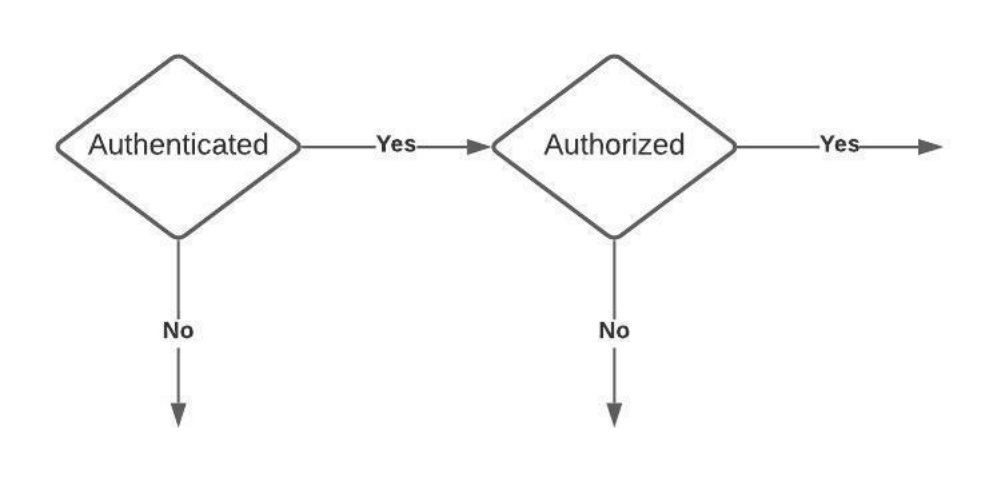
Hi Michael,

I really like your diagram. You did a good job at illuminating the details. In addition, I really enjoyed reading the referenced paper that had me interested from the first sentence. This is something that I am pretty sure resonates with many that have been in the industry for a couple of years. Based on the contents of that paper, it made me think about this article where Schulte (2017) talks about mobile first, security second. A term that I feel hits the mark for much of what we have seen.

Regarding your diagram, I was just wondering if one could simplify this part?



Instead of going into detail with the authentication process, you can encapsulate everything authenticated related in one node and then have the details documented elsewhere, if the stakeholders would need more detail as depicted below. This is by no means a complete diagram, just one to emphasize the flow between authentication and authorization.



References

Schulte, M.A. (March 1, 2017) Mobile First, Security Second? *E-3 Magazin.* Available from: https://e-3.de/mobile-first-security-second/ [Accessed 21 August 2021].

Reply to Hendrik Van Rooyen **from** Andrey Smirnov

Re: Peer Response

Hi Hendrik,

Thank you for referencing a very interesting article that elucidates how companies are lagging to recognize the importance of mobile security. I had to use Google Translate as my German is not sufficiently advanced to discern all meaning, so I hope my understanding is correct that the author also hints at a certain tension that these companies experience between the need to innovate their mobile offerings on the one hand, and to ensure sufficient security on the other. This is an ever fascinating trade off.

According to the recent survey by Mishra & Thakur (2019), there is more than a dozen potential mobile security attack vectors that occur on application, web, network and physical levels. Some of the more common mobile security problems are cross-site scripting attacks, insecure communications and data storage, as well as malware (Au & Choo, 2016). While there is probably no complete solution to all of these challenges, we have seen progress in development of effective defense mechanisms such as biometric authentication. From the perspective of organizations, implementation of threat management, secure policies, and a secure environment for mobile devices, are all needed to protect critical business data (Nagarjun & Ahamad, 2018). Overall, it can be said that preventing security issues on mobile devices is a shared responsibility of application developers, application hosting providers, mobile device and OS manufacturers, as well as end users.

References

Au, M. & Choo, K. (2016) *Mobile Security and Privacy: Advances, Challenges and Future Research Directions*. 1st ed. Syngress.

Mishra, S. & Thakur, A. (2019) A Survey on Mobile Security Issues. Proceedings of Recent Advances in Interdisciplinary Trends in Engineering & Applications (RAITEA). DOI: http://dx.doi.org/10.2139/ssrn.3372207

Nagarjun, P. & Ahamad, S. (2018) Review of Mobile Security Problems and Defensive Methods. *International Journal of Applied Engineering Research* 13(12): 10256-10259.

Response to Shan Swanlow's Post

Post by Michael Justus

Peer Response

Hi Shan,

The use of a Structured Activity is intelligent and shows a deeper understanding of UML.

The activity "Copy Token from URL" implies the generation of a token value. I assume this is where the failure occurs, referred to in the post? If so, out of interest, what is the likelihood of six sequential token values occurring? The Java Documentation mentions that java.util.random is "not cryptographically secure", and the recommendation is to use instead "SecureRandom" (Java, 2021) because it adheres to FIPS 140-2 standard (section 4.9.1, "Power Up Tests", FIPS (2001))

You raised a rather intriguing recommendation in the post to utilise a microservice approach when generating RNG-based tokens. How would such an approach work if, for example, a library intends to generate a cryptographically secure token of, say, 128 bytes in length? I suppose even the microservices would require some form of limits to prevent overloading them with requests or the potential to produce sequential numbers from their internal RNGs.

References

FIPS (2001) FIPS PUB Security Requirements for Cryptographic Modules. Available from https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf [Accessed 18 Aug 2021]

Java (2021) Java Platform Standard Edition 8: Class SecureRandom. Available from https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html [Accessed 18 Aug 2021].

Reply to Michael Justus **from** Shan Swanlow

Re: Peer Response

Hi Michael,

Thanks for your response. The reason why it's possible to obtain six sequential values is because of how the java.util.Random library is built. If a developer wishes to use the library, they'll begin by creating an instance of Random itself, which will then set up an internal state. When a developer wants to get some random value, they would call one of the "next" methods (e.g. nextInt, nextFloat, nextBytes). Due to the internal state of the RNG, future results are always influenced by previous results, so the "next" methods effectively create a deterministic sequence. In the context of a password reset token, a developer might choose to use nextBytes() to generate a token, however, this would mean that if an attacker requests a password reset six times in a row, they're effectively calling nextBytes() six times in a row, providing a set of sequential values which can be used to determine the generator's internal state. This does make the assumption that no other users request a password reset within that timeframe, however, in certain cases, it's still possible to determine internal state even if values are skipped (tailcall, 2017).

With regards to your second point, I know of a quantum RNG that is offered via a web API, which has been implemented in different libraries and languages (ANU QRNG, n.d.). Although it's possible to do the same for a cryptographically secure RNG, it would introduce network-based risks that need to be managed. In my opinion, those risks outweigh the benefits, because it would be creating a (subtle) single point of failure- what do you think? Microservices would improve security when combined with a secure offline library so that each instance is truly standalone, but as you mention, additions would be necessary. Apart from ensuring that each microservice instance has a unique, truly random internal state, implementing a rate limiting strategy is a good suggestion as it would reduce the risk of a single user obtaining a sequence of values and has the additional benefit of improving overall resilience (Google, 2021). I imagine you're thinking something along the lines of allowing each unique IP to send one reset request every X hours? Another key idea is load balancing- if requests are consistently distributed across different microservice instances, and each instance has a truly random internal state, it wouldn't be possible for an attacker to observe sequential tokens.

References

ANU QRNG. (n.d.) Frequently asked questions. Available from: https://qrng.anu.edu.au/contact/faq/#downloads [Accessed 18 August 2021].

Google. (2021) Rate-limiting strategies and techniques. Available from: https://cloud.google.com/architecture/rate-limiting-strategies-techniques [Accessed 18 August 2021].

tailcall. (2017) Cracking RNGs: Linear Congruential Generators. Available from: http://tailcall.net/posts/cracking-rngs-lcgs/ [Accessed 18 August 2021].

Reply to Shan Swanlow **from** Michael Justus

Re: Peer Response

Hi Shan,

I find the response excellent because you clearly show the underlying reason why such token generation can fail. The reference to the quantum RNG project is also a valued addition because (reading about the project) is a fascinating product developed by Down Unddah, whose goal is to produce almost random numbers based on some quantum field fluctuations.

Regarding the use of microservices in generating RNG-based tokens, with the ANU project referenced, it is very likely these two components can quickly resolve the issue identified. Each microservice contains a unique state, and the yonder machine generates such uniqueness. But, networks introduce points of failure. After all, attacks inevitably involve network access (yes, if attackers are internal to an organisation, with direct access to hardware and applications, that is another concern). So one thought is that leveraging microservices must be hardened against failure (security or otherwise). Rudrabhatla (2020) makes a case for three pertinent microservice concerns: secure the attack surface, encrypting everything; and, continuous monitoring. Jindal et al. (2019) put forward approaches such as sandboxing, determining microservice capacity and using load generators like Kubernetes Cluster.

References

Jindal, A., Podolskiy, V. & Gerndt, M. (2019) Performance modelling for cloud microservice applications. Proceedings of the 2019 acm/spec international conference on performance engineering: 25-32.

Rudrabhatla, C.K. (2020) Security Design Patterns in Distributed Microservice Architecture. arXiv:2008.03395.

Reply to Michael Justus **from** Andrey Smirnov

Re: Peer Response

Hi Michael,

Thank you for providing those references, I found them extremely interesting to go through. Indeed, with the adoption of distributed architectures we have seen a widening of the attack spectrum, and many security techniques that were successful with monolithic web applications are not as effective when applied to microservices. One approach towards increasing MSA security that has seen good adoption in the industry revolves around the use of token-based authentication/authorization mechanisms (e.g., OAuth, JSON Web Tokens), particularly coupled with the API gateway pattern. In this design approach, the API gateway acts as a single point of entry into the system and enforces token validation before routing the frontend requests to backend services. Some organizations that deal with sensitive customer data (ABN AMRO being one example) take this concept one step further and employ a secondary (Internal/Private) API gateway that adds another layer of security. This can mean that every request from the frontend application needs to pass through up to 4 layers before ending up in the actual backend service responsible for handling the call. While this setup arguably adds latency costs, the benefits from the "defence in depth" approach, for some organizations, largely outweigh these costs.

Kind regards,
Andrey

Response to Suresh Sigera's Post

Post by Michael Justus

Peer Response

Hi Suresh,

Thanks for your diagram. I thought the reference to the "content.salt.security" report was really good because they provide some good material regarding APIs' use. Your post mentions that "66% of organizations concede to have delayed the rollout of a new application into production because of API security concerns". Such a high number is valid when considering further that, according to the report, 61% of respondents use APIs for platform or system integrations, and 55% have experienced API-related vulnerabilities. This supports the security concerns you expressed of sensitive data via APIs.

According to OWASP (A10:2017-Insufficient_Logging) (OWASP1), systems require more support for logging their activities, so it is likely beneficial if your diagram could show logging a message after the "Check domain is whitelisted-->No" flow. The post refers to A3:2017-Sensitive_Data; but, it is not obvious from the diagram _where_ such sensitive data is accessed or how the proposed flow guards against such access. Would the sensitive data perhaps be related to the "Register service" activity?

References

OWASP1 (2021) OWASP Top Ten 2017:A10:2017-Insufficient Logging and Monitoring. Available from https://owasp.org/www-project-top-ten/2017/A10_2017-Insufficient_Logging%2526Monitoring [Accessed 18 Aug 2021]

Response to Taylor Edgell's Post

Post by Michael Justus

Peer Response

Hi Taylor,

Your diagram clearly represents what a broken authentication flow looks like from a user's point of view. Following the flow, what happens if the system being attacked does not properly authenticate but does have rate limits in place? Does your flow terminate given such a condition? If it does, the poor hacker has to start afresh from password #1 until the system rejects further attempts. Another good point that stands out from the flow shown in the diagram is how closely related it is to a Denial of Service (DOS) attack--flooding a target system with requests.

In the flow, I assume the attacker knows whether or not the target system uses TLS/SSL or if they must encrypt the password before submitting. Would the proposed flow change slightly if these conditions hold?

Reply to Michael Justus **from** Taylor Edgell

Re: Peer Response

Hi Michael,

Thank you for your response. At the moment, within my diagram, I have made the assumption that there is no rate limit for submitting passwords. This is an explicit situation that I think would be useful to add in further implementations. One note regarding rate limits is that it is a practise that can vary between websites, and company policies. For instance some website login facilities may have you wait up to 24 hours after 3 failed login attempts, on the other hand others only require the users cache to be cleared and the site to be accessed again. There is currently no set standard for this protective mechanism.

In regards to TLS/SSL they are not necessarily applicable to the exact error I am trying to show can be exploited. The primary error I am trying to demonstrate is poor password quality, and the lack of reasonable limits on password attempts. TLS/SSL would become relevant in relation to an external source trying to intercept the password once it has been submitted by a user. For instance if a site was not protected then a malicious party could simply apply a form of man-in-the-middle attack to take the information they want instead of trying to guess the security information. Further to this, without TLS/SSL, the external party could arrange for the website to inject, or trick the user into downloading, malicious code onto their systems.

Reply to Taylor Edgell **from** Michael Justus

Re: Peer Response

Hi Taylor,

Great, thank you. You mentioned "poor password quality". Do you think that such password policies have any impact on improving against this type of attack?

For example, organisations (mysteriously) have their own set of password quality requirements, but as Adams & Sasse (1999), note, "System-generated passwords are essentially the optimal security approach; however, user-generated passwords are potentially more memorable and thus less likely to be disclosed". So, one assumes that activity "Input List of 10000 passwords" essentially loads a list of memorable, easy-to-crack passwords.

Interesting, too, Grobler et al., (2020) consider the role that "social identity" plays in user passwords. For example, as they state email addresses that end in ".com.de" likely indicate a German speaker who likely formulates passwords using the German vocabulary.

References

Adams, A. & Sasse, M.A. (1999) Users are not the enemy. Communications of the ACM, 42(12):40-46.

Grobler, M., Chamikara, M.A.P., Abbott, J., Jeong, J.J., Nepal, S. and Paris, C. (2020) The importance of social identity on password formulations. Personal and Ubiquitous Computing:1-15.