Agile threat modeling and the "the devil is in the details" idiom

December 20, 2022

Share

Disclaimer

This post is based on the following elements:

1. My experience working as a developer (2003-2015) and then as a full-time Application Security Consultant (2015-present).
2. The collection of trainings I have recently followed about Threat Modeling activity.
3. My regular technical survey on the Application Security field.

📢 Therefore, it is quite possible that my point of view is wrong in some aspect or biased. In this case, I will be more than happy to get feedback to make my point of view evolve.

Glossary

This post uses the following terms:

- **User story:** A user story is an informal, general explanation of a software feature written from the perspective of the end user (source).
- **Abuser story:** An Abuser Story is the "evil" version of a user story (source).
- **Threat scenario:** A threat scenario is how an abuser story can happen from a technical perspective (source).

Context

As far as I can remember of my life as a developer, security was commonly limited in software specifications. For instance, one of them remains burned in my mind, it was regarding the Authentication and Authorization topic: "The software must be compliant with the OWASP Top 10" (top 10 is an awareness intended project, not one to really evaluate the security level of a web application).

With the time, methodologies used to develop software evolved to agile ones. The specification, I previously received, became a "user story" with more collaboration with my teammates and the client (it was a great and fun evolution 😊). However, the security aspect did not really change, still staying very foggy.

When I moved as a full-time Application Security guy in 2015, user stories still not contained any information related to the security aspect of the feature. Fortunately, this has since changed, but nowadays software created using an agile methodology still contains weaknesses like, for example, injection ones.

🤔 I asked myself, for a while, the simple question "**Why do we still have security weaknesses in our software?**". In this blog post, I will humbly try to:

1. Understand the reason. Perhaps I was doing something wrong or missing something important as a AppSec guy?
2. Contribute changing this state.

🔎 Therefore, I try to find an answer to my question through this post using all the information I had at the time of writing of this post in October 2022.

User Story and Abuser Stories: It is only the first part of the security path

Nowadays, a user story is complemented by one or several abuser stories representing the threats that can affect the features described by the story.

Let's take an example to define a working context that we will follow along this post.

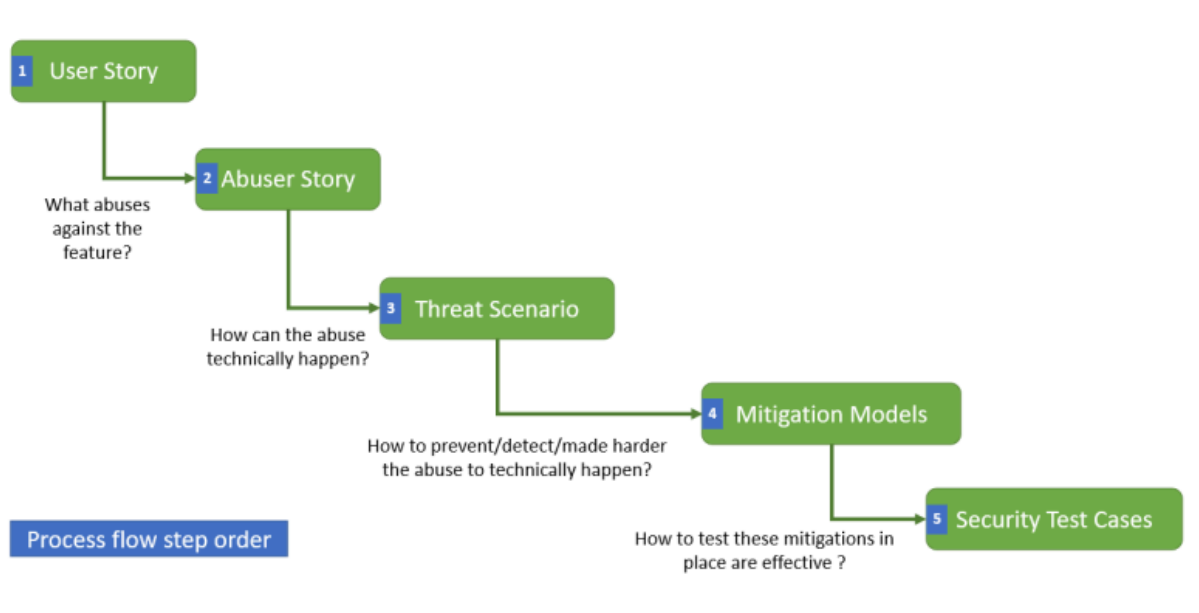
User story: As a client I want to download my account statement from the web banking so I can validate my spending of the current month.

- **Abuser story 1:** As a malicious client I want to download an account statement of an arbitrary client so I can access to his/her spending information to spy on him/her.
- **Abuser story 2:** As a malicious client I want to perform a wire transfer using a negative amount so I can achieve the specified amount to be added to my account instead of being subtracted.
- **Abuser story N:** …

The abuser story is described to allow the inclusion of non-technical people (like a business analyst for example) in the process of identification the abuser stories (threats modeling activity) and gain their insights and point of view. This inclusion is very important to gather technical and business threats.

💬 As repeated several times, by Abhay Bhargav, trainer of the modules I followed: "Threats modeling activity is a collaboration work between people having different skills and mindset.".

It is the reason why, at this stage, the work is not done yet regarding the security aspect of the user story. Indeed, the whole picture of the process is the following:



The occurrences of the different entity relations are the followings:

Relation	Occurrences
User Story to Abuser story	One user story has one or several abuser stories.
Abuser story to Threat scenario	One abuser story has one or several threat scenarios.
Threat scenario to Mitigation models	One threat scenario has one or several mitigation models.
Mitigation models to Security test cases	One mitigation has one or several security tests cases.

🤔 So, if an abuser story, like the one above, is given to the developer AS IS, then he/she cannot implement counter measures because there is no technical information about how to achieve the corresponding protection to the threat described.

The second part of the security path

Once an abuser story was identified, the technical deep dive can start to clearly materialize how it can happen from a technical perspective? Let us use the example defined to make things easier to understand.

Abuser story 1 – As a malicious client I want to download an account statement of an arbitrary client so I can access to his/her spending information to spy on him/her.

😈 So, as an attacker, how can I achieve the goal underlined?

- Via an injection against the account statement storage or generation feature: SQL injection, Path Traversal, etc.
- Via an abuse on the account statement identifier leveraging exposure to an Insecure Direct Object References vulnerability combined with an Authorization (access control) vulnerability.
- By abusing the caching system used by the storage or generation feature to access files already present in caching systems.
- And so on…

😈 Every technical representation identified above is a **threat scenario**. To identify them, an offensive mindset is needed to be able to build one or several attack scenarios using the abuser story as objective.

📚 Depending on the company, such profile, would not be available (more on this aspect, later in the post). In this case, referential like CAPEC (Common Attack Pattern Enumerations and Classifications) can be used, focusing on the "Software" domains of attack, to identify applicable collection of attacks enabling to reach the abuser story evil objective. In addition, the "OWASP Automated Threats to Web Applications" project, can help too in the identification of automated attacks applicable to the abuser story.

🤔 There is still no technical information allowing him/her to implement the corresponding protection to the threat scenarios identified.

The third part of the security path

Once a threat scenario was identified, it is crucial that the corresponding countermeasures are identified. Indeed, it is really such kind of information that will allow the developer to implement the protection against the threat represented by the threat scenario.

🎯 There are different ways to handle a threat:

- **Treat it** by mitigate it.
- **Terminate it** by removing the subject of the risk.
- **Transfer it** by, for example, subscribing to an insurance covering the impact of the threat.
- **Tolerate it** by accepting the risk.

✅ More precisely, there are several ways to treat a threat:

- You can directly prevent its realization.
- You can detect its occurrence as quick as possible to limit its impact.
- You can make its exploitation significantly harder.

🛡️ The collection of elements put in place to mitigate a **threat scenario** is called a **mitigation models**. It is recommended to combine elements to increase the deepness and resilience of the protection. Let us use again the example defined to make things easier to understand.

Threat scenario 1 – Via an injection against the account statement storage or generation feature: SQL injection, Path Traversal, etc.

🏹 So, as a defender, how can I handle this threat scenario?

- Prevent its realization, by leveraging, in combination:
  - SQL prepared statement.
  - Strict input validation because I can define the format of the account statement identifier.
- Detect occurrence of the threat:
  - Trigger a security event if the feature returns an access denied to a request for an account statement (illegal access tentative).
  - Trigger a security event if the amount of data returned by an HTTP response is larger than the average of an HTTP response normally returned by the feature (data exfiltration pending).
- Make the exploitation of the threat significantly harder:
  - Use a format of an account statement identifier not prone to guessing or iteration like for example UUID v4 or truly random one (look at this OWASP cheat sheet for more details).
  - Revoke the session/access token or lock the user account if the amount of data returned by an HTTP response is larger than the average of an HTTP response normally returned by the feature.

📋 The level of technical details of a mitigation model **must enable a developer to implement the corresponding protection without asking himself/herself how to do it from a technical perspective.**

📚 Duplicate threat scenarios will be identified across different abuser stories, it is normal. It is a good opportunity to centralize common/reusable mitigation in a Kknowledge Base (in a wiki for example) to unify the protection, make easier their update as well as speeding up the task to define mitigation models.

🏗️ Indeed, building with the time, a KB of reusable mitigation (like *recipes*) will reduce the time spent on mitigation models definition. This KB will contain such information, among others:

- Libraries and corresponding code snippets to use for mitigation like input validation, output encoding, file type validation, data secure deserialization, etc.
- Regular expressions or features of framework to use to validate a specific type of business data like IBAN, credit card number, social security number, etc.
- Libraries and corresponding code snippets to use for data cryptographic operations (ciphering, hashing, random content generation, data integrity protection).
- Algorithms and code snippets to handle sensitive according to their sensitivity level defined by the company data classification referential.
- Preconfigured project templates, secure by default, to use for the different framework used in the company.
- Any other technical reusable materials…

💬 As repeated several times, by Abhay Bhargav, trainer of the modules I followed: "Developers are our customers, so, making their job easier is important by making security the more transparent possible for them.".

🤔 So, once again, if the mitigation models provided for each threat scenario is too vague, incomplete, or ineffective then the protection implemented by the developer will be in the same state and then vulnerability will be present.

The fourth part of the security path

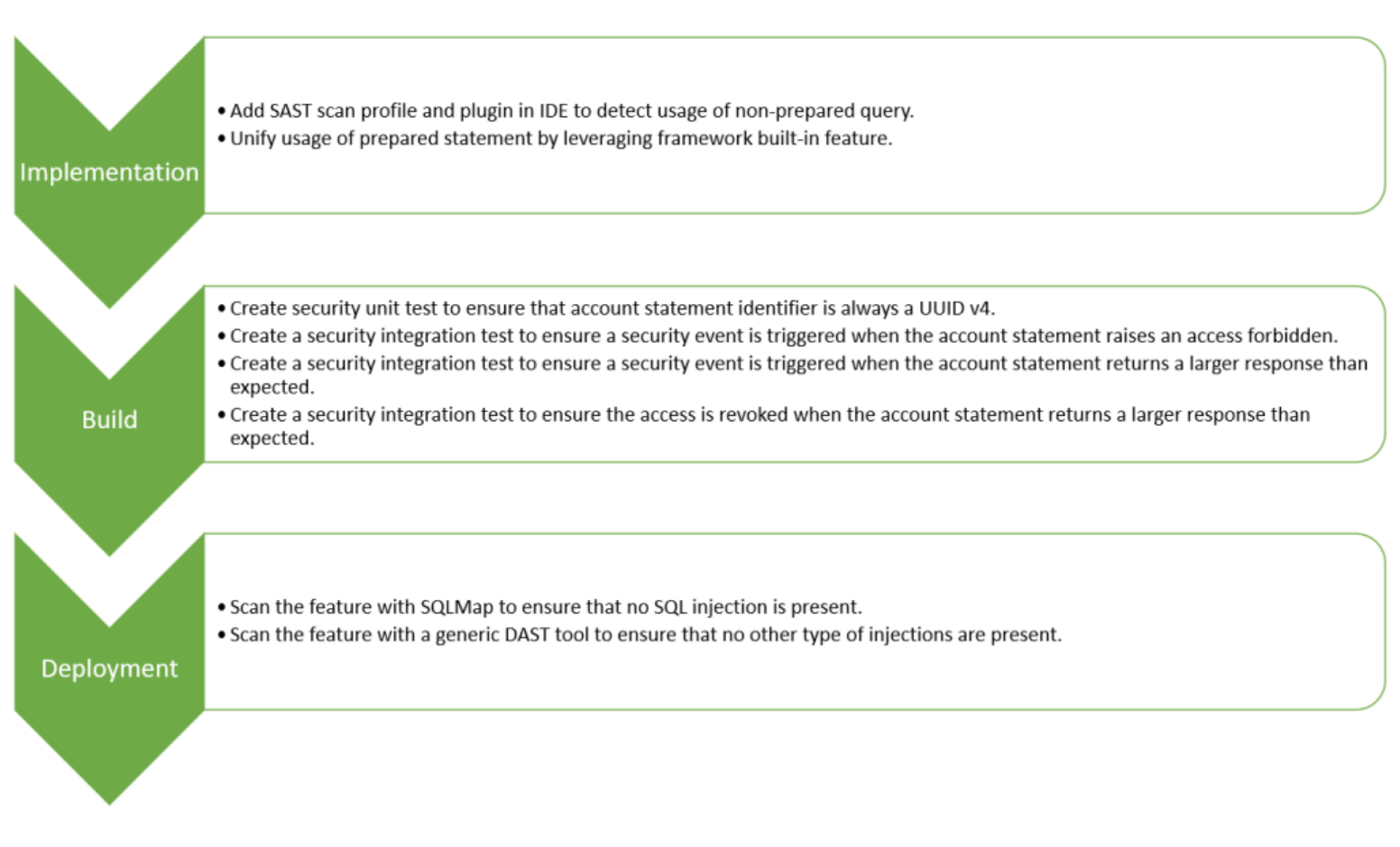
Once a mitigation model was identified and implemented, it is important to ensure that it is effective and stay in place. The test in charge of ensuring this state is named a **security test case**.

🛡️ Ideally, a mitigation model must be backed by several **security test cases** present at different stages of the code life:

1. At **implementation** time: Via live checks in the IDE level via static code analysis (using static application security testing tools).
2. At **build** time: Via security unit tests (when applicable) and static code analysis.
3. At **deployment** time: Via security integration tests and a dynamic application scan (using dynamic application security testing tools).
4. During **external manual assessment** (security code review or intrusion tests): By providing the mitigation models to external auditors to allow them to validate their effectiveness.

🐜 Indeed, it brings a resilient validation process in case a test would not be accurate or would be buggy. Let's use the example defined to make things easier to understand.

✅ The following schema map possible different security test cases to verify the mitigation models for threat scenario 1:



🤝Remark: It is interesting to provide the information of your threat modeling (abuser stories + threat scenarios + mitigation models) to auditors during an external assessment. Indeed, they can validate the effectiveness of your mitigation models. In addition, they can notify you regarding any missing abuser story, threat scenario or mitigation model. It brings a new level of collaboration between the auditors and your teams.

🤔 So, if the security test cases are absent or incomplete then ineffective mitigation can be present as well as associated vulnerability.

The Security Champion role to the rescue…

Along this post, many times, references were made to task requiring low or advanced security skills in the "software" domain. As also mentioned, depending on the company (size, business, culture) such skills might not present and then making hard or impossible performing the full threat modeling of a user story to consider its security aspect.

🌎 It is a well-known problem in the software industry. It is why the term, and then, role of "Security Champion" was created. To cite the description of such role from SecurityJourney:

*"A security champions program is designed to **embed security expertise throughout the entire organization.** This includes soliciting volunteers from each team to become advocates for good security practices with their peers […] Security champions **are the ambassadors of security** and are looked upon as subject matter experts among developers."*

💡 So, creating such role and profile in the company can be a way to address the issue in the achieving of the complete threat modeling of user stories.

📡 The following podcast provides useful insight about creating Security Champion in a company:

- A different way of approaching Security Champions
- Training Security Champions
- Level up your Security Champions

Few words on the process…

The table below tries to map the different phase of the threat modeling of a user story in terms of profile of people and phase in an agile methodology:

Threat modeling part	Profile	Agile phase
Abuser story	Business analyst, technical lead, developer, security team member	Spring planning meeting
Threat scenario	Technical lead, developer, security team member	After the spring planning meeting and before the implementation
Mitigation models	Technical lead, developer, security team member	After the spring planning meeting and before the implementation
Security test case	Technical lead, developer, security team member	During the implementation

💡 "Definition of done" or "Delivery point" should include a statement that every mitigation models must be covered by, at least, one security test case.

Conclusion

🏭 Usage of agile methodologies in software development allows considering the security aspect of every user story. However, it is important to consider the full scope of an abuser story. Providing the expected materials for the implementation of related protections enables their validation and effectiveness.

✅ Regarding my initial question, I have my answer: When a vulnerability is present then it is because something was missing/incomplete in the security path of the affected user story.

Additional useful technical references

- https://owasp.org/www-project-security-culture/v10/
- https://owasp.org/www-community/Source_Code_Analysis_Tools
- https://cheatsheetseries.owasp.org/
- https://owasp.org/www-project-secure-headers/
- https://catalog.securityjourney.com/resources/appsec-podcast

Author

Dominique Righetto

General | Modeling, Security, Threat

Search

Search …