Design of Secure Data and Information Flow in JavaScript Applications

Michael Fürmann
University of Applied Sciences Augsburg
Email: michael.fuermann@hs-augsburg.de

***Abstract*—There are several techniques supporting the implementation of a secure data and information flow in applications. But many of the tools required to apply them are hard to find for complex JavaScript Projects. Nonetheless it is important to identify potential security risks in a software project as early as possible.**

This paper proposes new methods, applicable in early project stages, supporting the planning and implementation of a secure data and information flow. They aim to identify application components sensitive to data and information flow security, find potential countermeasures and rate their value for security analysing mainly data from stakeholder analysis, usage scenarios and a conceptual application model.

I. Introduction

Most users of online software systems never think about the data they disclose to the providers of their favourite applications. They just assume, their personal informations would be secure and protected against unauthorized access.

Even on large and prestigious platforms, this protection of sensitive information is not always given, as shown by the prominent case the platform Facebook fell victim to. The clever combination of data from userprofiles, accessible via official interfaces, enabled analysts to gain deep knowledge about single, identified users of the plattform [1].

Web applications nowadays face a wide range of different threats. The aims differ from disturbing the service itself or shutting it down temporarily to massive altering, deleting or copying of personal information stored on the systems.

The General Data Protection Regulation (GDPR) in the European Union obliges extensive duties for the protection of personal data in data processing systems. Thus not only the integrity and confidentiality of sensitive personal data has to be ensured, but also the possibility of unauthorized combination of pseudonymous data for profile generation has to be prevented. Therefore a developer or provider of an application must not only ensure to protect personal data of users, but also ensure, that pseudonymous data cannot be gathered, neither in legal or unrestricted way that allows to gain further personal information [2].

This increases the importance of identifying application sections significant for information security in early stages of the software project. Being aware of these, the conceptual model of the application can be designed to meet security requirements. Also developers can be made aware of identified, sensitive application components. So they are able to test critical code more conscientious or carefully examine required external JavaScript modules used in sensitive components to prevent the accidental inclusion of malicious code from external repositories [3].

II. Problem Description

In order to support the implementation of a secure data and information flow in an application, it is important to identify potential risks as early as possible. There already are some techniques to analyse the project, that provide valuable information in several stages of the project. But whilst methods in conceptual and planning phases are too generic to identify risks, those for the implementation phase mostly need working code and support of additional tools. For complex JavaScript project setups, those tools often reach their boundaries. For some techniques like the generation of a system dependence graph as described in III-C, it is hard to find tool support compatible to the projects code base.

Thus there is a need for methods, applicable in conceptual or planning phases of a project, that are specialized to find potential risks for data- and information flow. Because many innovative applications are planned and developed by small teams, they shouldn't require a staff of experts to apply those methods in due time and with affordable cost.

III. Basic Techniques

This section describes some generally accepted methods, that can be used to gather valuable information for the design of an application with a secure data- and information flow.

A. Stakeholder Analysis

The classic stakeholder analysis aims to secure wide range support for a project in the early concept- or planning phase. After listing all possible stakeholders, which includes persons and organizations having any contact points with the project, all their requirements and interests are worked out. Thus it is possible to detect not only requirements for the planning phase but also to identify stakeholder, supporting the project and those who are likely to slow down or block the project. Knowledge gathered about their interests allows now to plan countermeasures to convince and maybe turn them into supporters of the project, too [4, p.148ff].

B. Program Analysis

As technique to analyse written program code, there are many different approaches. The focus of this work lies on dynamic code analysis.

This approach is used to collect runtime data and to get insights into the programs behaviour at runtime, like executed control structures and function calls for specific input parameters. Therefore the application is compiled, executed, monitored and fed with different input data for the tested usage scenarios [5, p.339]. One example usage of this approach is an automated live debugger, executing the application having the possibility to intercept the code execution at any time, analyse the stack trace and previously executed control statements, to monitor the state of specific values or objects or to measure the performance of specific methods. For dynamic code analysis in JavaScript projects programs like Iroh[1] or DLint[2] can be used. In addition, this approach is used for dynamic and automated unit- or integration tests with tools like mocha[3] or jasmine[4].

C. Information flow Analysis

Using a system dependency graph, information flow Analysis is used to track the path of a specific piece of of information through the whole application. Beginning at the source of the information, the graph shows the way of all contained data fields towards a data sink. As an example, consider a function call submitting username and password, being processed by several internal method calls and ending up in an authentication call as illustrated in figure 1. Because the system dependence graph works on the level of real function calls, it requires existing program code or at least a very specific concept of the internal application structure [6, p.768].

IV. Explanation of proposed methods

While the methods described in chapter III provide valuable input about information being processed by the planned application, they do not examine potential risks in early project stages. In order to find appropriate countermeasures for security issues, they are required to be recognized as early as possible.

To be able to do so, this chapter introduces two new analysis methods. These are not meant to replace existing methods as described in chapter III, but to identify risks for information security and to suggest possible solutions in early stages of a software project that support the implementation of a secure information flow in the whole software development lifecycle [7]. Thus they follow the concept of "Security by modelling, verification and enforcement" [8, p.111].

All examples on how to use the methods in the next subsections refer to a research project at University of Applied Sciences Augsburg, which the author currently participates in. Aim of the project named Autark[5] is to build an application for teenagers suffering from Autism Spectrum Disorder to help them and their parents to structurize and organize their everyday life.

A. Data Source and Sink Analysis

In order to identify potential risks, it is mandatory to know which information are submitted to and requested by whom from an application and which of them possibly could reveal personal data. To gather this knowledge, Data Source and Sink Analysis uses the result of a previous stakeholder analysis to create a diagram (as shown in figure 2), that shows all stakeholders of the application along with all information that are provided or requested by them and those, that are generated from details provided to the application. Also information, that is implicitly generated by the usage of the application (like log file entries) is added to the diagram.

The presentation in figure 2 further displays information being personal or combinable to personal different from non personal information. Also restricted or unwanted requests by stakeholders can be distinguished from regular requests.

The graph developed this way now enables to identify information flows between users by using the planned application. Furthermore it highlights data access that needs to be controlled or permitted as well as sensitive personal data and information that should be prevented from being combined by users not authorized to personal data access.

B. Conceptional Information Flow Simulation

Tracing the way of the information through the conceptual model offers the possibility to identify security flaws for the data- and information flow security in the conceptual structure. As basis for this simulation it is obligatory to define usage scenarios for the ways users exchange information with and through the application. There should be at least one usage scenario that connects information requested by a user to at least one information source (as worked out in IV-A). For the demonstration of the simulation, the following usage scenario has been used:

1) Teenager performs a task
2) Teenager confirms the successful completion of the task to the system
3) System adds the successful completion of the task to the protocol
4) Teenager gets a confirmation by the system
5) Parents get informed about the successful completion of the task

Now a conceptual model of the application with user interactions and internal modules is used to visualize for every usage scenario the way information-carrying data objects are taking from their data sources to data sinks through components of the application as shown in figure 3. If required, the conceptual model is extended by further application components required to deal with the defined usage scenarios. This simulation is based on the system dependence graph used for security slicing in [6, p.768], but uses conceptual

[1] https://maierfelix.github.io/Iroh/
[2] https://berkeley-correctness-group.github.io/DLint/
[3] https://mochajs.org/
[4] https://jasmine.github.io/
[5] https://www.hs-augsburg.de/Informatik/Autark.html

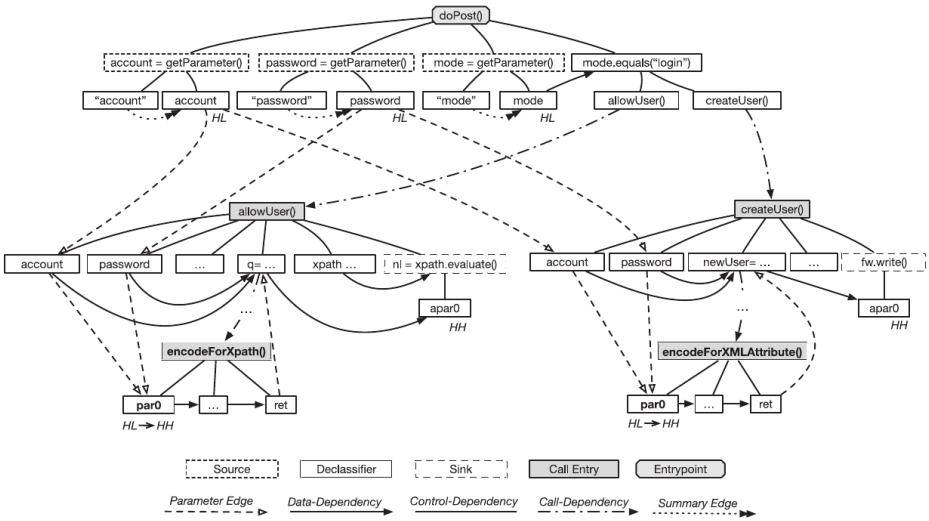

Fig. 1. Example system dependence graph [6, p.768]

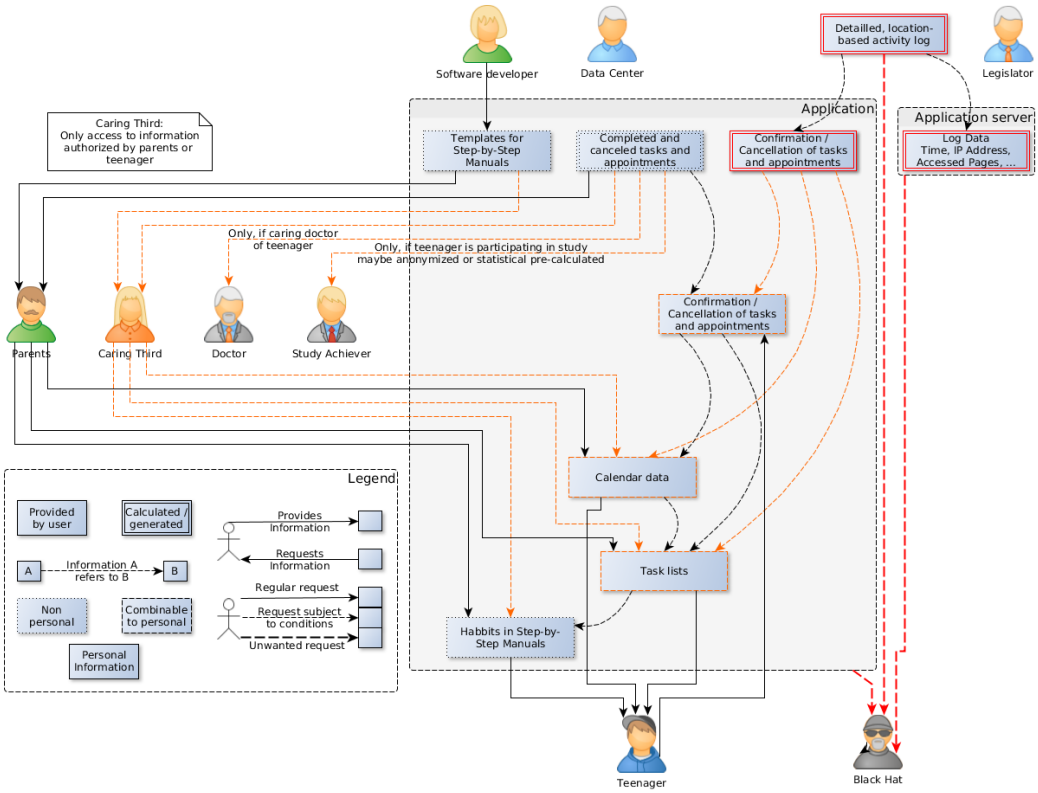

Fig. 2. All stakeholders of the application and their provided and requested information

application components instead of existing code statements and can therefore be used in conceptual stages of the project.

A data object might not always carry verified confidential personal data as well as application components might not be as trustworthy to be allowed to compute them or require verified data to prevent code injections. To take this circumstance into account, all data objects are classified regarding to their level of confidentiality and the reliability of their data integrity as described in figure 4. Furthermore, because the internet should be considered as an insecure medium, any data object being transferred between client- and serverside of the application should further be classified as unverified. Also the application components get classified by their trustworthiness to handle confidential data and their need for data integrity as

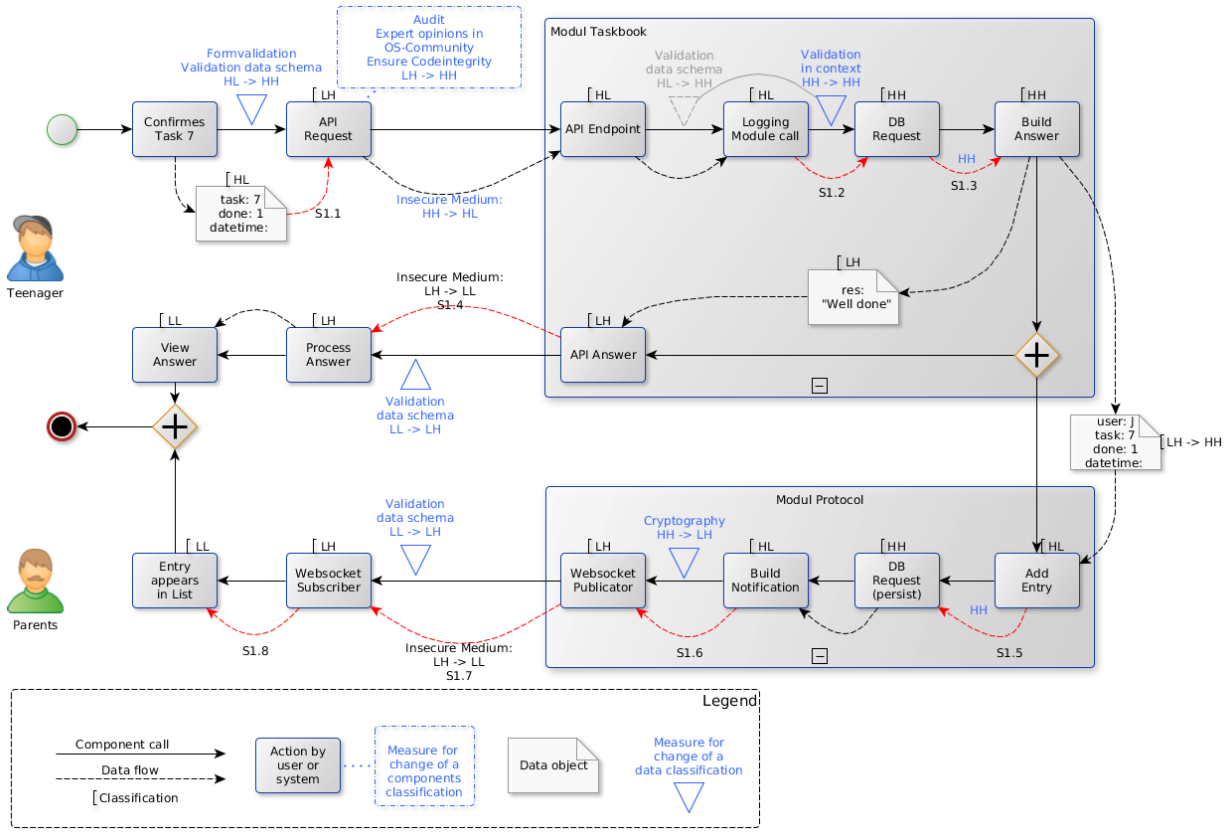

Fig. 3. Security Classified Information Flow Simulation with labelled breaks of security classification rules and suggestions for countermeasures (Solution for S1.2 was split up into two countermeasures for architectural reasons)

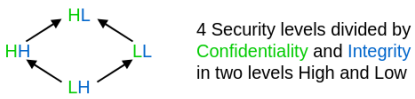

Fig. 4. Security Lattice defining allowed information flow to application components [6, p.771]

suggested by [6, p.771].

Data objects are only allowed to be handled by application components having the same or a higher security classification as shown in figure 4. All transitions, violating this requirement have been highlighted in figure 3 and labelled with an ascending number. These are potential security flaws in the application concept found by running an information flow simulation on it.

To achieve a secured data and information flow during implementation, for every break of the security classification a potential solution must be found. The possibilities range from actions to rise a components trustworthiness to reclassification of data objects, like improving integrity by validations or lower confidentiality by pseudonymisation or encryption, as also shown in figure 3. If required or convenient for the planned structure of the application, a solution can also be achieved by implementing multiple reclassification components, as shown by the countermeasures for violation S1.2 in figure 3.

TABLE I
PROPOSED SCORING SCHEMA FOR IDENTIFIED VIOLATIONS

Risk points for every determined violation	
+2	Affected information is personal (according to IV-A)
+1	Affected information can be combined to personal (see IV-A)
Additional risk points for classification breaks	
+3/2/1	Transfer of data object in contradiction with security classification (1st time/2nd time/further, see IV-B)
+1	Access to information is subject to conditions (see IV-A)
Additional risk points for not requested information	
+1	Information never requested by a user or only by unwanted drains (see IV-A)

C. Derivation of Results

The aim of this last step is to extract requirements for implementation and testing of the application as well as scored advisories for conceptual changes. The score serves as a measure for the countermeasures value in data and information flow security of the application.

As preparation for the derivation of results, all data object transactions violating the security classification scheme in IV-B are listed in a table of violations along with all information from IV-A that are, if at all, requested by unwanted drains. For each violation it is noted, if the affected information

reflects by itself or combined with others personal data. The table further shows, whether the information is not at all or only requested by unwanted drains or the access to the information is subject to further conditions (according to IV-A). The violation level displays the position of an identified violation in a chain of classification breaks by the same data object since its last classification according to IV-B. This data is now used to calculate risk points for every single entry using the scoring scheme proposed in table I. The proposed risk point scoring scheme presented here, is based on the judgement of the author, developed for the current project. It has to be confirmed whether it is generally applicable.

Using this scheme, violations are ranked by the confidentiality of affected information (personal, non personal or personal when combined with further data, according to IV-A). The first break of the security classification rules as shown in figure 4, is scored with more risk points (+3) than the next violation caused by the same object (+2) or any further violations afterwards by the same object (+1). A countermeasure solving the first violation in that chain in most cases will also solve all follow up violations. So it is important to focus on this first. Beside that, follow up violations scored the same way as the first one in the chain had too much influence on the rating of countermeasures in the last step. Last, also the access to the affected information is scored for either conditional access, unwanted access or never accessed information according to IV-A.

That way a weighted overview of all identified security flaws is gained, as shown in table II.

As final result for guidance on conceptual changes, countermeasures from IV-B, that solve one or more violations are collected and scored by accumulating the solved violations risk points as shown in table III. Some identified risks, especially from IV-A, might not be solvable by conceptual changes, but by requirements for the planned infrastructure on which the application will run. Those are also noted as solutions in the result table and scored the same way as the conceptual changes.
The calculated score reflects the benefit of the countermeasures found for the data- and information security in the application. If not all advisories are affordable, it can be used in a cost-benefit-assessment to decide about their realization.

In addition the graph created in IV-B allows to derive requirements for the implementation and testing of the application to achieve a more secure data and information flow. First and foremost, this concerns application components classified as "HL" or "HH" and thus are needed to be trustworthy as well as all added measures for reclassification of data objects. These, as listed in table IV, are subject to more intense testing and code reviews. At last violations like S1.5 deserve consideration. Even if it was indirectly solved by countermeasures for S1.2 and S1.3, "Add Entry" as a classified "HL" component is still allowed to handle HL-classified data objects. So, in order to avoid a security flaw at S1.5, it is required to treat "Add Entry" as HH-classified component during implementation.

V. Results

As shown by the example sliced from the research project Autark, it was possible to gather valuable information about security flaws (table II) along with advisories on how to fix them (tables III, IV). These recommend different countermeasures for violations, implementable in all phases of the project. Furthermore a scoring system provided a value of countermeasures for the security of the data and information flow.

The methods introduced have been shown to be applicable according to the requirements sentenced in II, at least for this project. However, a statement about universal applicability of the approach still requires further investigation.

VI. Conclusion

As described in IV-C, the introduced methods provide valuable input for all project phases from conception over planning to implementation and testing. In combination with existing basic techniques (as described in III), it enabled to identify personal data and combinable personal information as well as potential security flaws in the data- and information flow of a conceptual application model in an early project phase. For the implementation and testing phase, advice was gathered, which parts of the application will require more intense testing and reviews. Furthermore, requirements for securing the later infrastructure running the application have been worked out. A scoring model for risks and possible solutions allows at last to decide about the realization of countermeasures in a cost and benefit assessment.

Chapter V already mentioned, that, although this approach worked for the examples taken from the research project Autark, it can not yet be seen as generally applicable. Especially the risk point scoring schema, as proposed in IV-C, being based on the authors knowledge of information security and experience as software developer, needs further research and development by being applied to further application models. If the shown methods are practicable for other JavaScript software projects or maybe for software development in general has to be subject to further studies.

Furthermore there is still a need for research and development regarding supporting analysis tools for JavaScript. While some programs for code analysis (III-B) catch up to the latest language specification, convenient tools for the generation of System Dependence Graphs (used in information flow analysis as described in III-C) of a complex JavaScript code base are still hard to find.

Acknowledgment

The author would like to thank Prof. Dr.-Ing. Alexandra Teynor from University of Applied Sciences Augsburg for her kind support and lots of valuable discussions on this research topic. This research is part of the research project Autark at University of Applied Sciences Augsburg[6].

[6]https://www.hs-augsburg.de/Informatik/Autark

TABLE II
IDENTIFIED VIOLATIONS WITH DETAILS TO PERSONAL REFERENCES OF AFFECTED DATA FROM SOURCE AND SINK ANALYSIS (IV-A), INCREASING VIOLATION LEVEL FOR EACH DATA OBJECT COUNTING CLASSIFICATION FLAWS SINCE ITS LAST CLASSIFICATION IN INFORMATION FLOW SIMULATION (IV-B), SCORED WITH RISK POINTS ACCORDING TO THE SCORING SCHEMA PROPOSED IN TABLE I

Violation	Personal data (see fig. 2)	Combined personal data (see fig. 2)	Violation level (see fig. 3)	Not / only unwanted accessed (see fig. 2)	Restricted access (see fig. 2)	Risk points (see tab. I)
S1.1	n	(+1) y	(+3) 1	n	n	4
S1.2	n	(+1) y	(+3) 1	n	n	4
S1.3	n	(+1) y	(+2) 2	n	n	3
S1.4	n	n	(+3) 1	n	n	3
S1.5	(+2) y	n	(+3) 1	n	n	5
S1.6	(+2) y	n	(+2) 2	n	n	4
S1.7	(+2) y	n	(+1) 3	n	n	3
S1.8	(+2) y	n	(+3) 1	n	n	5
Detailed presentation of daily personal routine	(+2) y	n	-	(+1) y	n	3
Log data	(+2) y	n	-	(+1) y	n	3
Detailed activity log	(+2) y	n	-	(+1) y	n	3

TABLE III
RESULT: CONCEPTUAL CHANGES SCORED BY ACCUMULATION OF SOLVED VIOLATIONS RISK POINTS FROM TABLE II

Action	Solves	Score
Reclassifications		
Formvaidation, Validation of data schema before API call, ensure code integrity of API library	S1.1	4
Validation of data scheme at API Endpoint, further validation in context of module call	S1.2, S1.3, S1.5	12
Validate data schema of received confirmation	S1.4	3
Secure data object cryptographically before sending out to parents	S1.6, S1.7, S1.8	12
Validate data schema of received protocol entry	S1.7	3
Systems and Infrastructure		
Deactivate server logs	D2, D3	6
Spread Data across multiple systems	D1, D3	6
Secure SSH access to systems	D1, D2, D3	9

TABLE IV
REQUIREMENTS FOR IMPLEMENTATION AND TESTING

Subject to intense testing, additional reviews	
All reclassification actions	
Taskbook	API Endpoint
Taskbook	Logging Module call
Taskbook	DB Request
Taskbook	Build Answer
Protocol	Add Entry
Protocol	DB Request
Protocol	Build Notification
Restrictions / Relabeling	
Protocol - Add Entry	Treat as HH

REFERENCES

[1] C. Cadwalladr and E. Graham-Harrison. (2018) How Cambridge Analytica turned Facebook 'likes' into a lucrative political tool. [Online]. Available: https://www.theguardian.com/technology/2018/mar/17/facebook-cambridge-analytica-kogan-data-algorithm

[2] (2017) DSGVO: Grundsätze für die Verarbeitung personenbezogener Daten. intersoft consulting. [Online]. Available: https://www.datenschutzbeauftragter-info.de/dsgvo-grundsaetze-fuer-die-verarbeitung-personenbezogener-daten/

[3] F. Y. Rashid. (2017) Malicious code in the Node.js npm registry shakes open source trust model. [Online]. Available: https://www.csoonline.com/article/3214624/security/malicious-code-in-the-node-js-npm-registry-shakes-open-source-trust-model.html

[4] U. Holzbaur, M. Bühr, D. Dorrer, A. Kropp, E. Walter-Barthle, and T. Wenzel, *Die Projekt-Methode: Leitfaden zum erfolgreichen Einsatz von Projekten in der innovativen Hochschullehre (German Edition)*. Springer Gabler, 2017. [Online]. Available: https://link.springer.com/book/10.1007/978-3-658-15462-2

[5] L. Sampaio and A. Garcia, "Exploring context-sensitive data flow analysis for early vulnerability detection," *Journal of Systems and Software*, vol. 113, pp. 337–361, mar 2016.

[6] J. Thomé, L. K. Shar, D. Bianculli, and L. Briand, "Security slicing for auditing common injection vulnerabilities," *Journal of Systems and Software*, vol. 137, pp. 766–783, mar 2018.

[7] G. Deepa and P. S. Thilagam, "Securing web applications from injection and logic vulnerabilities: Approaches and challenges," *Information and Software Technology*, vol. 74, pp. 160–180, jun 2016.

[8] M. Bugliesi, S. Calzavara, and R. Focardi, "Formal methods for web security," *Journal of Logical and Algebraic Methods in Programming*, vol. 87, pp. 110–126, feb 2017.