Project OVVL – Threat Modeling Support for the entire secure development lifecycle

Andreas Schaad [1]

Abstract: OVVL (the Open Weakness and Vulnerability Modeller) is a tool and methodology to support threat modeling in the early stages of the secure software development lifecycle. We provide an overview of OVVL (https://ovvl.org), its data model and browser-based UI. We equally provide a discussion of initial experiments on how identified threats in the design phase can be aligned with later activities in the software lifecycle (issue management and security testing).

Keywords: Threat Modeling, Secure Software Development Lifecycle

1 Introduction

Threat Modeling is an effort to identify potential security design flaws and vulnerabilities as early as possible in the software development lifecycle to avoid costly fixes and re-engineering activities in the later stages. Apart from STRIDE [Sh14] and TAM2 [SB12], surprisingly little has been done so far regarding practical and lightweight "hands-on" methodologies as well as automated tool support - although it is conventional wisdom that early identification of security issues is good engineering and management practice. We thus present OVVL, a methodology and tool that integrates into the different stages of the software lifecycle to further drive technical and managerial acceptance of threat modeling. The goal is to not only identify vulnerabilities, but to track them throughout the later lifecycle stages and thus increase the quality of cybersecurity in real-world systems.

2 OVVL (the Open Weakness and Vulnerability Modeller)

2.1 Underlying Methodology & Data Model

OVVL [SR19] is based on (and extends) the commonly accepted STRIDE methodology for threat modeling [Sh14]. A complex system design can be abstracted to data flow diagrams and a first match is made with respect to possible STRIDE threats (Spoofing, Tampering, Repudiation, Information Disclosure, Denial, Elevation of Privileges). What distinguishes the OVVL approach from the known body of work is that the OVVL data model allows to enrich a design with information about the components that may be used for realising a system. For example, a software architect can describe that the underlying

[1] Hochschule Offenburg, Fakultät Medien, Badstr. 24, Offenburg, andreas.schaad@hs.offenburg.de

doi:10.18420/sicherheit2020_10

database will be a MongoDB with a certain release level. Based on that information, OVVL will query existing vulnerability databases and identify CVE entries, i.e. known and reported real-world vulnerabilities. Information about the identified threats (at design level) and known vulnerabilities (at operational level) can then be pushed to available tooling in the software development lifecycle. In the context of this paper we discuss bug tracking (3.1) and vulnerability testing (3.2).

2.2 Technical Architecture

OVVL is a pure web application enabled by Spring Boot. The javascript application logic is consumed through an Angular UI / frontend. Data is managed in a MongoDB and all OVVL services are available through Swagger-generated REST APIs. The OVVL server can be operated as a virtual machine to be used internally where required.

2.3 Case Study 1: Threat modeling of a cloud application (CloudProtect)

OVVL has been developed as part of the CloudProtect project[2] and is currently used to provide an initial security analysis of the CloudProtect architecture (Figure 1). This analysis yielded over a dozen potential design threats as well as 5 known existing major vulnerabilities (CVE entries > 7.0) which at this stage of the project serves as a good basis for technical discussions.

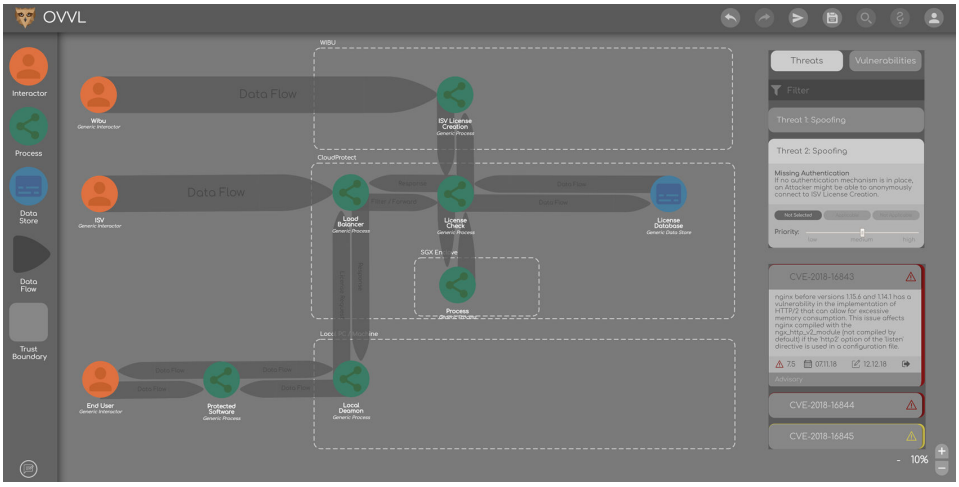


Fig. 1: OVVL Model of CloudProtect (with examples of threats and vulnerabilities)

[2] BMBF KMU-Innovativ “CloudProtect” FKZ: 16KIS0850

2.4 Case Study 2: Retrospect Threat Modeling (green&easy)

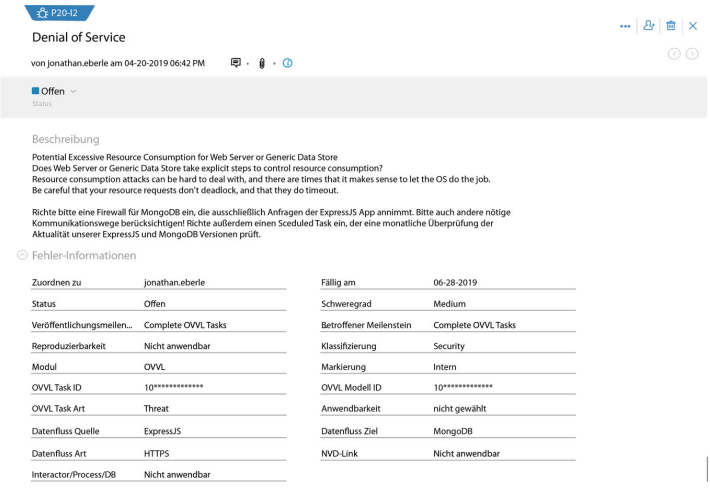
We used OVVL to model a web application called “green&easy” [Eb19]. The interesting fact here is that the responsible architect was not security trained and used OVVL in retrospect after he and his team had already provided a first working proof of concept.

The OVVL-based system design consisted of 2 interactors, 7 processes, 1 database grouped over 4 trust boundaries. 22 threats were automatically identified by OVVL out of which 20 threats were deemed as critical enough to be pushed into the projects issue tracker (Zoho). The majority of these identified threats (16) addressed possible elevation of privileges. On basis of the provided metadata, 13 immediate vulnerabilities where flagged with respect to used components such as MongoDB, ExpressJS and Redux. In fact, the analysis made in [Eb19] provides detailed estimates that the overall design effort is even reduced from 808 to 702 developer hours when using OVVL (now already including a detailed threat and vulnerability analysis).

3 OVVL Integration into the Secure Development Lifecycle

3.1 Bugtracker Integration

OVVL supports integration with the Zoho bugtracker. This allows to automatically push identified threats and associated metadata into an off-the-shelf project management tool (Figure 2). Based on this, a project manager can track whether identified threats or vulnerabilities are resolved throughout the project lifecycle. A detailed analysis of the Zoho API and corresponding data items in OVVL has been provided [Eb19].


P20-I2

Denial of Service

von jonathan.eberle am 04-20-2019 06:42 PM

Offen

Status

Beschreibung

Potential Excessive Resource Consumption for Web Server or Generic Data Store
Does Web Server or Generic Data Store take explicit steps to control resource consumption?
Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job.
Be careful that your resource requests don't deadlock, and that they do timeout.

Richte bitte eine Firewall für MongoDB ein, die ausschließlich Anfragen der ExpressJS App annimmt. Bitte auch andere nötige Kommunikationswege berücksichtigen! Richte außerdem einen Sceduled Task ein, der eine monatliche Überprüfung der Aktualität unserer ExpressJS und MongoDB Versionen prüft.

Fehler-Informationen

Zuordnen zu	jonathan.eberle	Fällig am	06-28-2019
Status	Offen	Schweregrad	Medium
Veröffentlichungsmeilen...	Complete OVVL Tasks	Betroffener Meilenstein	Complete OVVL Tasks
Reproduzierbarkeit	Nicht anwendbar	Klassifizierung	Security
Modul	OVVL	Markierung	Intern
OVVL Task ID	10**********	OVVL Modell ID	10**********
OVVL Task Art	Threat	Anwendbarkeit	nicht gewählt
Datenfluss Quelle	ExpressJS	Datenfluss Ziel	MongoDB
Datenfluss Art	HTTPS	NVD-Link	Nicht anwendbar
Interactor/Process/DB	Nicht anwendbar		

Fig. 2: OVVL Threat managed in the Zoho Bugtracker

3.2 Vulnerability Testing

As described earlier, OVVL can help to identify design threats as well as existing vulnerabilities. We thus analysed whether reported CVEs can be used as a reference when applying automated vulnerability scanners such as Nessus (as well as openvas and nmap_vulners) in the later stages of the software lifecycle. As shown in figure 3, a test engineer can then identify whether an issue reported by Nessus had already been identified in OVVL (and thus early) or whether a new vulnerability was found. OVVL provides the needed interfaces and reporting UIs to Nessus.

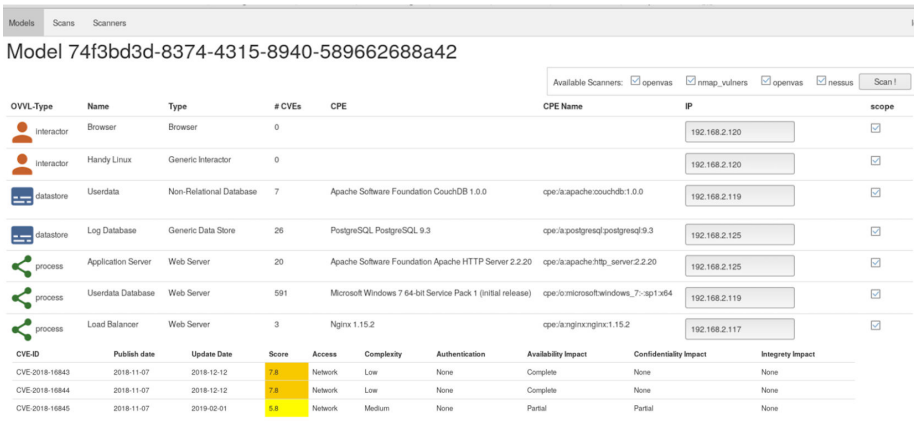


Fig. 3: OVVL data used by vulnerability scanners (Nessus, nmap-vulners, openvas)

4 Summary & Conclusion

OVVL is an open-source project available at https://github.com/OVVL-HSO. OVVL differs from existing open source tools by distinguishing between design threats and technical vulnerabilities. OVVL data can be used by other tools in the development chain such as issues trackers or vulnerability scanners.

Bibliography

[Eb19] Jonathan Eberle: „Konzeption eines agilen Software-Sicherheitsmanagements am Beispiel einer Mobile App “, Hochschule Offenburg, 2019, Nr. 1676005854

[SB12] Andreas Schaad, Mike Borozdin “TAM^2: Automated threat analysis”, ACM SAC 2012

[Sh14] Adam Shostack “Threat Modeling: Designing for Security” Wiley, 2014

[SR19] Andreas Schaad, Tobias Reski: "Open Weakness and Vulnerability Modeler" (OVVL): An Updated Approach to Threat Modeling. ICETE (2) 2019: 417-424