Applying dynamic taint propagation in order to enforce domain driven security

FREDRIK ADOLFSSON

Master in Computer Science
Date: January 25, 2018
Supervisor: Musard Balliu
Examiner: Mads Dam
Swedish title: Tillämpa dynamic taint propagation för att genomdriva domändriven säkerhet
School of Computer Science and Communication

Abstract

Sammanfattning

Contents

Chapter 1

Introduction

1.1 Problem

1.2 Aim

1.3 Definitions

Definition 1.3.1. Domain

Definition 1.3.2. Domain Model

1.4 Delimitations

1.5 Methodology

Chapter 2

Background

2.1 Web Applications

2.2 Injection

2.2.1 Cross-site Scripting

2.2.2 SQL

2.3 Taint Propagation

2.3.1 Dynamic

2.3.2 Static

2.4 Domain Driven Design

There exists a plethora of tools who aim to help in the process of developing complex domain models, but Domain Driven Design (DDD) is not one if them. [2, 5] DDD is more of a thought process and methodology to follow every step of the process. [4] In *Domain-driven design reference: definitions and patterns summaries* do Evans [3] describe DDD trough three core ideas:

- Focus on the core domain.
- Explore models in a creative collaboration of domain practitioners and software practitioners.

- Speak a ubiquitous language within an explicitly bounded context.

The core domain is the part of your product that is most important and often is your main selling point compared to other similar products. [7] A discussion and even possible a documentation describing the core domain is something that will help the development of the product. The idea is to keep everybody on the same track heading in the same direction. [4]

The second idea is to explore and develop every model in collaboration between domain practitioners, who are experts in the given domain, and software developers. This ensures that important knowledge needed to successfully develope the product is communicated back and forth between the two parties. [7] The third idea is important to enable and streamline the second. By using a ubiquitous language will miscommunication between domain and software practitioners be minimized and the collaboration between the two parties can instead focus on the important parts which is to develop the product. [3]

Evans [3] do as well argue about the weight of clearly defining the bounded contexts for each defined model, and this needs to be done in the ubiquitous language created for the specific product. The need of this exists because of the otherwise great risk of misunderstandings and erroneous assumptions in the collaborations between the different models. [7]

2.4.1 Domain Driven Security

Wilander [9] and Johnsson [6] created 2009 a blog post each in a synchronous manner where they together introduces the concept of Domain Driven Security (DDS) to the public. They describe DDS as the intersection between Domain Driven Deign (DDD) and application security. DDD is about developing complex domain models and one of the most basic rule of application security is to always validate input data. DDS in other hand, is about the importance of creating and maintaining domain models who are reflecting the product correctly and they are validated so they cant be populated with erroneous data. [9, 6, 1, 8]

Chapter 3

Implementation

3.1 Plain (Bad name)

3.2 Taint Propagation?

3.3 Domain Driven Security

Chapter 4

Result

Chapter 5

Discussion

Chapter 6

Future Work

Chapter 7

Conclusion

Bibliography

[1] Johan Arnör. "Domain-Driven Security's take on Denial-of-Service (DoS) Attacks". In: (2016), p. 54. URL: http://kth.diva-portal.org/smash/get/diva2:945831/FULLTEXT01.pdf.

[2] Steven C Bankes. "Tools and techniques for developing policies for complex and uncertain systems Introduction: The Need for New Tools". In: (). URL: http://www.pnas.org/content/99/suppl%7B%5C_%7D3/7263.full.pdf.

[3] Eric Evans. *Domain-driven design reference: definitions and patterns summaries*. Dog Ear Publishing, 2015.

[4] Eric Evans. *Domain-driven design : tackling complexity in the heart of software*. eng. Boston, Mass.: Addison-Wesley, 2004. ISBN: 0-321-12521-5.

[5] Rabia Jilani et al. "ASCoL: A Tool for Improving Automatic Planning Domain Model Acquisition". In: *AI*IA 2015 Advances in Artificial Intelligence*. Ed. by Marco Gavanelli, Evelina Lamma, and Fabrizio Riguzzi. Cham: Springer International Publishing, 2015, pp. 438–451. ISBN: 978-3-319-24309-2.

[6] Dan Bergh Johnsson. *Dear Junior - Letters to a Junior Programmer: Introducing Domain Driven Security*. 2009. URL: http://dearjunior.blogspot.se/2009/09/introducing-domain-driven-security.html (visited on 01/25/2018).

[7] Scott Millett. *Patterns, principles, and practices of domain-driven design*. Wrox, a Wiley brand, 2015.

[8] Jonas Stendahl. "Domain-Driven Security". In: (2016), p. 39. URL: http://kth.diva-portal.org/smash/get/diva2:945707/FULLTEXT01.pdf.

[9] Johan Wilander. *OWASP Sweden: Domändriven säkerhet / Domain-Driven Security*. 2009. URL: http://owaspsweden.blogspot.se/2009/09/domandriven-sakerhet-domain-driven.html (visited on 01/25/2018).

Appendix A

Example