

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

1	Introduction	1
1.1	Problem	1
1.2	Aim	1
1.3	Definitions	1
1.4	Delimitations	1
1.5	Methodology	1
2	Background	2
2.1	Web Applications	2
2.2	Injection	2
2.2.1	Cross-site Scripting	2
2.2.2	SQL	2
2.3	Taint Propagation	2
2.3.1	Dynamic	2
2.3.2	Static	2
2.4	Domain Driven Design	2
2.4.1	Domain Driven Security	3
3	Implementation	4
3.1	Plain (Bad name)	4
3.2	Taint Propagation?	4
3.3	Domain Driven Security	4
4	Result	5
5	Discussion	6
6	Future Work	7
7	Conclusion	8

Bibliography	9
A Example	10

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 of them. [1, 4] DDD is more of a thought process and methodology to follow every step of the process. [3] In *Domain-driven design reference: definitions and patterns summaries* do Evans [2] describe DDD through 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. [5] 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. [3]

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 develop the product is communicated back and forth between the two parties. [5] 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. [2]

Evans [2] 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. [5]

2.4.1 Domain Driven Security

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] Steven C Banks. “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.
- [2] Eric Evans. *Domain-driven design reference: definitions and patterns summaries*. Dog Ear Publishing, 2015.
- [3] Eric Evans. *Domain-driven design : tackling complexity in the heart of software*. eng. Boston, Mass.: Addison-Wesley, 2004. ISBN: 0-321-12521-5.
- [4] 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.
- [5] Scott Millett. *Patterns, principles, and practices of domain-driven design*. Wrox, a Wiley brand, 2015.

Appendix A

Example