

Implementing Dynamic Taint Propagation to Enforce Domain Driven Security

Specification and Time Schedule

FREDRIK ADOLFSSON - FREADO@KTH.SE

Master in Computer Science
Date: March 7, 2018
Supervisor: Musard Balliu
Examiner: Mads Dam
Principal: Jonatan Landsberg & Simon Tardell
School of Computer Science and Communication

Contents

1	Introduction	1
2	Background	2
2.1	Related Work	3
2.2	Goal & Objective	4
3	Research Question & Method	5
4	Evaluation & News Value	7
5	Pre-study	8
6	Conditions & Schedule	9
6.1	Resources	9
6.2	Limitations	9
6.3	Company Supervisor	9
6.4	Time Plan	10
	Bibliography	11

Chapter 1

Introduction

1990 was the World Wide Web (Web) founded by Tim Berners-Lee and the creation have cause a huge impact on todays society. [21] The Web is a source for information and it connects the world in un unanimous platform. Many bushiness have decided to take advantage of this to gain accessability for their users to the system. But this accessability gain for the targeted user group dose not come without its drawbacks. The accessability is a weakness in the same manner as it is a strength. The Web Application is not only accessible for the targeted user group but for all user groups. Which entails that users who wishes to abuse and/or cause harm to the application have the accessability to possibly do so.

There are a number of possible attacks that a Web Application is vulnerable to and the attack that might be the most frequently conducted today will probably not be the same as the most conducted in the future. The organization Open Web Applications Security Project, mostly known for its shortening OWASP, is a online community which aim to provide knowledge about how to secure Web Applicationss. [10] OWASP have produced reports about the top 10 security risks for a Web Application and the latest was published 2017. Among those security risks is number one Injection Attacks and number seven Cross-Site Scripting. [11, 10, 3]

This thesis will look at the two named security risk and perform evaluations and benchmarks of a possible solution to prevent these kinds of attacks.

Chapter 2

Background

As said in the previous chapter is Injection Attacks the number one security vulnerability for Web Applications. Injection Attack is a collection name for any attack where the attacker's input changes the intent of the execution. Some possible versions of Injection Attacks is injection of queries that manipulates SQL, NoSQL, OS and LDAP executions. [11] The most common goals and result of a Injection Attacks are file destruction, lack of accountability, denial of access and data loss. [17]

Cross-Site Scripting (XSS) in other hand was number seven at OWASP top 10 security vulnerabilities published in 2017. [11] But the origin of the attack goes back to the beginning of the Web and one of the first XSS attacks was conducted just after the release of JavaScript. The attack was conducted through loading a malicious web application into a frame on the site that the attacker want to gain control of. The attacker could then through malicious scripts from the frame access any content that is visible or typed into the web application. The first prevention for XSS was then introduces through the standard of Same-Origin Policy. Same-Origin Policy restricts JavaScript to only access content from its own origin. [5, 15]

To prevent these form of security vulnerabilities in web applications have a variety of tools and methodologies been created. One of these is Dynamic Taint Propagation which goal is to prevent possible Injection and Cross-Site Scripting attacks in run time. This is done by marking input variables from sources, which is a marking point where malicious data might enter the system, as tainted through a taint flag attached to the variable. This taint flag follows the variable through-

out the application and propagates onto the other variables it comes in contact with. It is possible to detaint (remove the taint flag) a tainted variable but this is only done after the variable have been sanitized through validation. The taint values are checked in areas called sinks which is a marking for entrypoints to sensitive code such as SQL executions. [13, 18] The decision of what to do when a tainted variable tries to pass through a sink might vary depending on the application. But in general for a Dynamic Taint Propagation tool is the common reaction to stop the execution of the tainted code. But other actions such as logging or raising an alarm is not uncommon.

The described Dynamic Taint Propagation tool is a solution to protect the integrity of the system where there are security protocols detailing that no information from the sources are allowed to enter a sink without being sanitized through validation. The same application could however be used to enhance the confidentiality of the system. The security policy for doing this would be opposite of the integrity policy. The policy would be that data that comes from a sink must be sanitized and validated to not be sensitive before letting it pass through a source. This thesis will focus on the integrity policy which will reflect in the decided benchmarking. But the question of how it might help confidentiality policies as well is an interesting aspect.

But there is not only tools that have been created to help prevent Injection Attacks and XSS. One methodology that have been coined to help secure applications is the programming paradigm Domain Driven Security. Domain Driven Security aim to secure applications by focusing on the core domain models and making certain that validation of the value objects are correct. [20, 9]

The question that this thesis will evaluate is if we can combine Dynamic Taint Propagation and Domain Driven Security and develop a tool that enforces the security gains of Domain Driven Security.

2.1 Related Work

Stendahl [16] wrote a thesis in 2016 where he evaluated if a Domain Driven Security is able to prevent Injection Attacks and Cross-Site Scripting. He came to the conclusion that there is a security gains towards Injection Attacks and Cross-Site Scripting and the gained security comes from proper validation.

Halдар, Chandra, and Franz [6] have written a report about Dynamic Taint Propagation in Java where they try to solve the problem of not properly validating user input. They managed to construct a tool that is independent from the Web Applications source code and can see a gain in security. Halдар, Chandra, and Franz [6] ran their benchmarks on OWASP's project WebGoat [2] but acknowledged in their report that benchmarks of real Web Applications need to be tested.

There do exist two Dynamic Taint Propagation tools where Phosphor [14] is one and Security Taint Propagation [4] is another. Both are open source projects and developed for Java applications.

2.2 Goal & Objective

The goal of this thesis is to implement and benchmark a Dynamic Taint Propagation tool which aims to enforce the security gains of Domain Driven Security. The Dynamic Taint Propagation tool's meaning will be to prevent Injection Attacks and Cross-Site Scripting in run time where possible malicious code shall be prevented from executing and logged.

The principal, Omegapoint, is interested in everything that might validate, invalidate, evolve or bring a further value to the programming paradigm Domain Driven Security. The reason for this is because the concept of Domain Driven Security was born and is in development by Omegapoint consultants. Omegapoint also likes to see a prototype of a Dynamic Taint Propagation tool which is able to block attacks in run time.

Chapter 3

Research Question & Method

How can an implementation of a Dynamic Taint Propagation tool enforce the security gains of Domain Driven Security.

The assignment would be to benchmark and evaluate the implementation of a Dynamic Taint Propagation tool and discuss if it helps to enforce the security gains of Domain Driven Security. The process of this thesis would be to conduct, in order:

Literature Study The literature study is where information relevant to the thesis need to be gathered and presented. Details of what the focus will lie upon is presented in chapter 5.

Tainting & Detainting This step is the part where tainting and detainting rules are decided. These need to be decided since the next step is the implementation of the Dynamic Taint Propagation tool. I will focus on defining rules that enforces security policies to protect for integrity. But the question of how these can be reused or slightly modified to enforce policies of confidentiality protection should also be mentioned. The thought process of constructing these rules will also have Domain Driven Security in mind since the goal is to apply higher security for the applications that will be run on the Dynamic Taint Propagation tool.

Implementation The implementation step is where the Dynamic Taint Propagation tool is implemented. Omegapoint have developed a proof of concept product which I will continue my work upon. This tool is developed in and for Java with help of the Javassist

[7] which makes the manipulation of bytecode easier. The proof of concept is developed to check taint on HTTP query strings through a Spring server. I will continue the work and expand its coverage.

Benchmarking This step is where the Dynamic Taint Propagation tool will be benchmarked. The benchmarking will be a comparison to see how it relates to other similar applications. The two applications that will be the reference points are Phosphor [14] and Security Taint Propagation [4]. The applications to benchmark will be a mixture of some real-world applications and benchmarking applications. Omegapoint have two internal applications that can be used and Stanford SecuriBench [8] is a set of open source programs that is developed for evaluating security tools, which therefore is optimal to use for benchmarking.

The values that is in focus during the benchmark are the values in the table below.

- Added Time Complexity
- Prevention Rate
- False Positive Rate

The tools to use to test the previously named applications will be all or some of; OWASP Zed [12], w3af [19] and Loader [1].

Analysis The analysis step is where the benchmarking results is reflected upon and written into the report. The discussion will contain thought of how it helps to secure applications and what have been extra successful or the opposite.

Report Writing & Presentation The last steps is to finalize the report and present the thesis.

Chapter 4

Evaluation & News Value

The thesis is considered to reach its goal if a Dynamic Taint Propagation tool have been developed and is in working state where integrity policies are enforced. The tool should have been benchmarked where comparisons with current state of art applications. There should also be a evaluation containing well thought comments and observations about the benchmarking results. A comparison/analysis of the possibility for the Dynamic Taint Propagation tool to enforce the security gain of Domain Driven Security shall also be conducted.

The relevance in the thesis lies in the problem with software security. Since we are going towards an age where digitalization only grows larger is the question about how we can secure our software extremely relevant. The hypothesis is that we can help in the process of enforcing more secure software. But the question is with how much and if there are negative side effects such as to much overhead to the runtime.

The work should be of interest for anyone wanting to see a gain in security. Especially for individuals or business who do already have Java applications who they wishes to increase the security for.

Chapter 5

Pre-study

The literature study will focus on gathering the relevant information needed for the report. These areas are listed in the table below:

- Java Applications
- Injection Attacks
- XSS
- Dynamic Taint Propagation
- Domain Driven Security
- Javassist

Research into JVM modifications must also be included since it is needed for the implementation of the Dynamic Taint Propagation tool. The information will be obtained by researching for relevant books, reports and other possible material. Two of the founders of the concept of Domain Driven Security work at Omegapoint and are accessible for questions. Conduction interviews with the founders might be of interest.

Chapter 6

Conditions & Schedule

6.1 Resources

To save some time will the development of the Dynamic Taint Propagation tool continue on the work that Simon Tardell have started. Which is a tool developed in and for Java with help of the Java library Javassist [7]. Applications to use to evaluate the implementation is also of need. But example of these have been mentioned in chapter 3.

6.2 Limitations

- The Dynamic Taint Propagation tool dose not have to be a production ready. The goal is to develop a prototype.
- Web Applications is the targeted applications.
- The scope of the thesis will not contain Static Taint Propagation.
- The tool is developed in Java with Javassist.

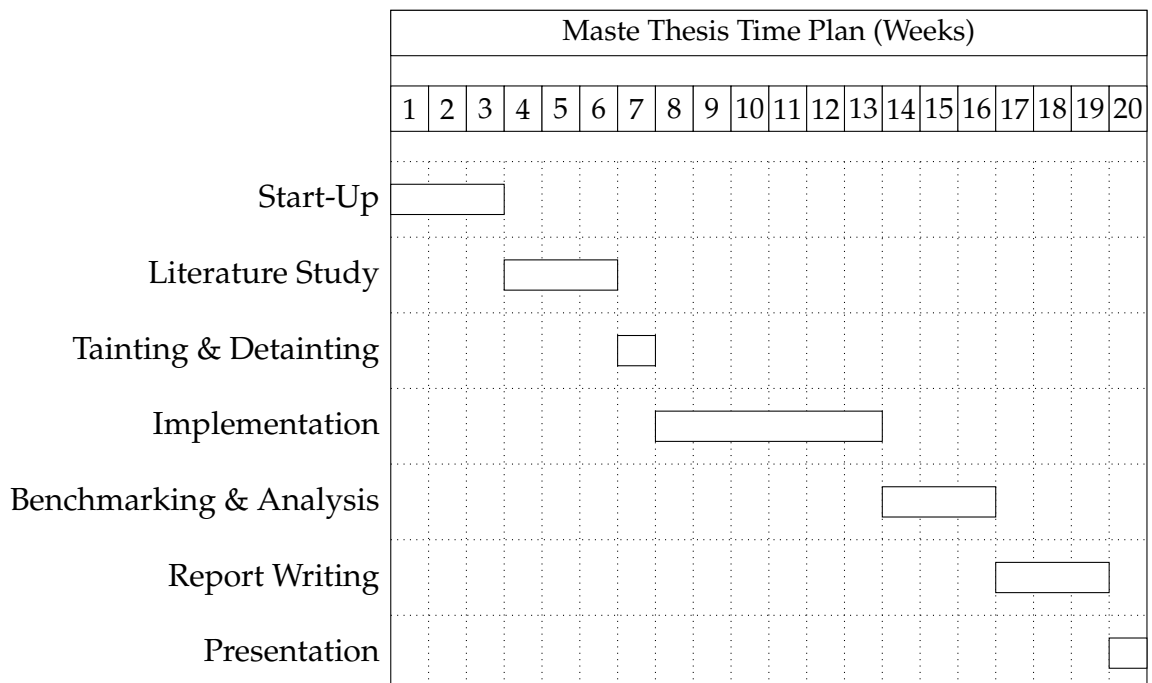
6.3 Company Supervisor

- **Jonatan Landsberg:** Will assist with supervision on the academic part if the thesis.

- **Simon Tardell:** Supervisor in the technical parts of the thesis. He is also the author of the first draft of the Dynamic Taint Propagation tool which this thesis will continue its work upon.

6.4 Time Plan

Below is my time plan for the Masters Thesis. The goal is to continuously, throughout all phases, add to the report. But I have also reserved a couple of weeks in the end for writing the report. I believe that this time can be used to add to or rewrite sections if needed.



Bibliography

- [1] *Application Load Testing Tools for API Endpoints with loader.io*. URL: <https://loader.io/> (visited on 03/05/2018).
- [2] *Category:OWASP WebGoat Project - OWASP*. URL: https://www.owasp.org/index.php/Category:OWASP%7B%5C_%7DWebGoat%7B%5C_%7DProject (visited on 03/06/2018).
- [3] Michael Cross. *Developer's guide to web application security*. eng. Rockland, MA: Syngress Publishing, 2007. ISBN: 1-281-06021-6.
- [4] *Dynamic Security Taint Propagation in Java via Java Aspects*. URL: https://github.com/cdaller/security_taint_propagation (visited on 03/06/2018).
- [5] Seth Fogie. *XSS attacks cross-site scripting exploits and defense*. eng. Burlington, MA: Syngress, 2007. ISBN: 1-281-06024-0.
- [6] Vivek Halder, Deepak Chandra, and Michael Franz. "Dynamic Taint Propagation for Java". In: (). URL: <https://pdfs.semanticscholar.org/bf4a/9c25889069bb17e44332a87dc6e2651dce86.pdf>.
- [7] *Instrumentation (Java Platform SE 8)*. URL: <https://docs.oracle.com/javase/8/docs/api/java/lang/instrument/Instrumentation.html> (visited on 02/27/2018).
- [8] *Introduction to Stanford SecuriBench*. URL: <https://suif.stanford.edu/%7B~%7Dlivshits/securibench/> (visited on 03/07/2018).
- [9] 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).

- [10] Open Web Application Security Project. OWASP. URL: https://www.owasp.org/index.php/Main%7B%5C_%7DPage (visited on 02/01/2018).
- [11] OWASP. "OWASP Top 10 - The Ten Most Critical Web Application Security Risks". In: *Owasp* (2017), p. 22. URL: https://www.owasp.org/images/7/72/OWASP%7B%5C_%7DTop%7B%5C_%7D10-2017%7B%5C_%7D%7B%5C_%7D28en%7B%5C_%7D29.pdf.pdf%7B%5C_%7D0Ahttp://scholar.google.com/scholar?hl=en%7B%5C_%7DbtnG=Search%7B%5C_%7Dq=intitle:OWASP+Top+10+-2010%7B%5C_%7D1.
- [12] OWASP Zed Attack Proxy Project - OWASP. URL: https://www.owasp.org/index.php/OWASP%7B%5C_%7DZed%7B%5C_%7DAttack%7B%5C_%7DProxy%7B%5C_%7DProject (visited on 03/05/2018).
- [13] Jinkun Pan, Xiaoguang Mao, and Weishi Li. "Analyst-oriented taint analysis by taint path slicing and aggregation". In: *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS 2015-November* (2015), pp. 145–148. ISSN: 23270594. DOI: 10.1109/ICSESS.2015.7339024.
- [14] *Phosphor: Dynamic Taint Tracking for the JVM*. URL: <https://github.com/gmu-swe/phosphor> (visited on 03/06/2018).
- [15] *Same Origin Policy - Web Security*. URL: https://www.w3.org/Security/wiki/Same%7B%5C_%7DOrigin%7B%5C_%7DPolicy (visited on 02/07/2018).
- [16] Jonas Stendahl. "Domain-Driven Security". In: (2016), p. 39. URL: <http://kth.diva-portal.org/smash/get/diva2:945707/FULLTEXT01.pdf>.
- [17] Praveenkumat H Subbulakshmi T. *Secure Web Application Deployment Using Owasp Standards: An Expert Way of Secure Web Application Deployment*. Createspace Independent Publishing Platform, 2017.
- [18] Guru Venkataramani et al. "FlexiTaint: A programmable accelerator for dynamic taint propagation". In: *Proceedings - International Symposium on High-Performance Computer Architecture* (2008), pp. 173–184. ISSN: 15300897. DOI: 10.1109/HPCA.2008.4658637.

- [19] *w3af - Open Source Web Application Security Scanner*. URL: <http://w3af.org/> (visited on 03/05/2018).
- [20] Johan Wilander. *OWASP Sweden: Domändriven säkerhet / Domain-Driven Security*. 2009. URL: <http://owaspsweden.blogspot.se/2009/09/domanddriven-sakerhet-domain-driven.html> (visited on 01/25/2018).
- [21] *World wide web skapas – nu kan internet bli en publiksuccé | Internetmuseum*. URL: <https://www.internetmuseum.se/tidslinjen/www/> (visited on 03/06/2018).