Implementing Dynamic Taint Propagation to Enforce Domain Driven Security

Specification and Time Schedule

FREDRIK ADOLFSSON - FREADO@KTH.SE

Master in Computer Science
Date: March 5, 2018
Supervisor: Musard Balliu
Examiner: Mads Dam
Principal: Jonatan Landsberg & Simon Tardell
School of Computer Science and Communication

Contents

Chapter 1

Background

One of the greatest strengths with deploying applications on the World Wide Web (web) is that they are accessible from everywhere where there exists a internet access. This is sadly one of its greatest weaknesses as well. The applications are easily accessible for people who whishes to abuse and/or cause them harm. Among the number of security risks that a web application is vulnerable to are two of the most common Injection Attack and Cross-Site Scripting. [7, 2]

The most common of the two is Injection Attacks. [8] A Injection Attack is any attack where the attacker's input changes the intent of the execution. Common result of Injection Attacks are file destruction, lack of accountability, denial of access and data loss. [13] Injection Attacks can be divided into two different subgroups. These two subgroups are SQL Injection and Blind SQL Injection. [13]

Cross-Site Scripting (XSS) have been a vulnerability since the beginning of the internet. One of the first XSS attacks was conducted just after the release of JavaScript. The attack was conducted trough loading a malicious web application into a frame on the site that the attacker want to gain information of. The attacker could then through JavaScript access any content that is visible or typed into the web application. To prevent this form of attack were the standard of Same-Origin Policy introduced. Same-Origin Policy restricts JavaScript to only access content from its own origin. [3, 11]

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

trough marking input variables as tainted through a taint flag attached to the input. This taint flag follows the input throughout the application and propagates onto the other variables it comes in contact with. It is possible to detaint a tainted input variable but this is only done after the variable have been sanitized trough validation. Taint values are checked in sinks which protects sensitive areas. Execution is halted if a tainted variable is detected in a sink. [10, 14] One of the methodologies that have been coined 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 object is correct. [16, 6]

1.1 Goal & Objective

The goal of this thesis is to implement and benchmark a Dynamic Taint Propagation tool. The Dynamic Taint Propagation tools meaning will be to prevent Injection Attacks and Cross-Site Scripting in run time. Where those events shall be blocked and logged. The benchmark will check the values; injection prevention rate, false positive rate and added time overhead. The tools to use as benchmark is all or some of; OWASP Zed [9], w3af [15] and Loader [1]. A discussion whether this tool also helps to enforce the programing paradigm Domain Driven Security is also to be conducted.

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.

1.2 Related Work

Previous work that have been conducted to Dynamic Taint Propagation is for example the work of Haldar, Chandra, and Franz [4] and Zhao et al. [17]. Both reports have implemented a Dynamic Taint Propagation tool for web deployed Java applications where the manipulation of the Java bytecode is done trough Javassist.

Domain Driven Security is somewhat more unknown, compared to Dynamic Taint Propagation, and there are not many reports that

discusses and analyses the domain. But one thesis that is noticeable, is the thesis of Stendahl [12].

Chapter 2

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 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.

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.

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 [5] which makes the manipulation of bytecode easier. The proof of concept is developed to check taint on HTTP query strings trough a Spring server.

Benchmarking This step is where the Dynamic Taint Propagation tool will be benchmarked. The Dynamic Taint Propagation tool should

be tested on a larger set of applications to make the result significant. The values that is in focus during the benchmark is the values in the table below.

- Injection Prevention Rate
- False Positive Rate
- Added Time Complexity

Analysis The analysis step is where the benchmarking results is reflected upon and written into the report.

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

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.

Chapter 3

Evaluation & News Value

There should be a discussion and evaluation of the implemented Dynamic Taint Propagation tool. This evaluation should contain well thought comments and observations about the benchmarking result. 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 work should be of interest for anyone wanting to see a gain in security. The core idea is to enforce more secure software through Dynamic Taint Propagation. However, since the relation between Dynamic Taint Propagation and Domain Driven Security will be discussed will the practitioners of Domain Driven Security find it extra interesting.

Chapter 4

Pre-study

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

- Web Applications
- Dynamic Taint Propagation
- Domain Driven Security
- Injection Attacks
- XSS
- 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 5

Conditions & Schedule

5.1 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 [5]. Applications to evaluate the implementation is also of need. The thesis is at the moment aimed towards web applications which means that a number, 10 should be sufficient, of web applications need to be gathered. Omegapoint have some internal systems which could be used. Other usable web applications can be found on open source platforms.

5.2 Limitations

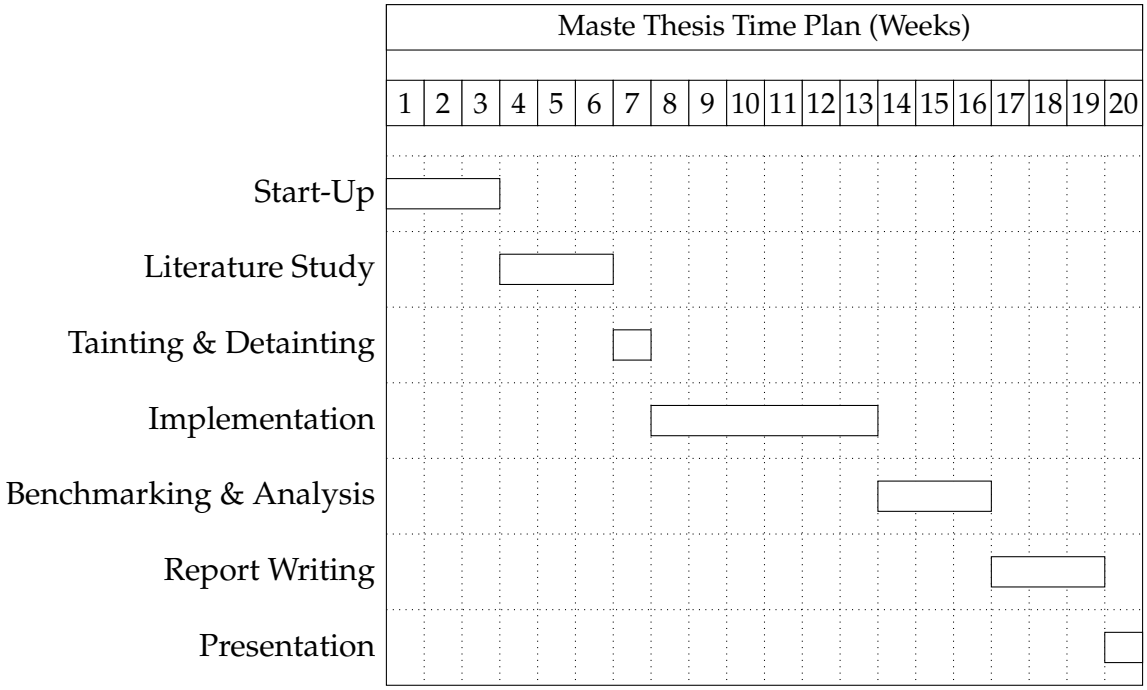
- The Dynamic Taint Propagation tool dose not have to be a production ready. The goal is to develop a prototype.
- Web Applications is the targeted applications.
- The scope of the thesis will not contain Static Taint Propagation.
- The tool is developed in Java with Javassist.

5.3 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.

5.4 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] Michael Cross. *Developer's guide to web application security*. eng. Rockland, MA: Syngress Publishing, 2007. ISBN: 1-281-06021-6.

[3] Seth Fogie. *XSS attacks cross-site scripting exploits and defense*. eng. Burlington, MA: Syngress, 2007. ISBN: 1-281-06024-0.

[4] Vivek Haldar, Deepak Chandra, and Michael Franz. "Dynamic Taint Propagation for Java". In: (). URL: https://pdfs.semanticscholar.org/bf4a/9c25889069bb17e44332a87dc6e2651dce86.pdf.

[5] *Instrumentation (Java Platform SE 8)*. URL: https://docs.oracle.com/javase/8/docs/api/java/lang/instrument/Instrumentation.html (visited on 02/27/2018).

[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] Open Web Application Security Project. *OWASP*. URL: https://www.owasp.org/index.php/Main%7B%5C_%7DPage (visited on 02/01/2018).

[8] 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.

[9] *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).

[10] 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.

[11] *Same Origin Policy - Web Security*. URL: https://www.w3.org/Security/wiki/Same%7B%5C_%7DOrigin%7B%5C_%7DPolicy (visited on 02/07/2018).

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

[13] Praveenkumat H Subbulakshmi T. *Secure Web Application Deployment Using Owasp Standards: An Expert Way of Secure Web Application Deployment*. Createspace Independent Publishing Platform, 2017.

[14] 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.

[15] *w3af - Open Source Web Application Security Scanner*. URL: http://w3af.org/ (visited on 03/05/2018).

[16] 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).

[17] Jingling Zhao et al. "Dynamic taint tracking of web application based on static code analysis". In: *Proceedings - 2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2016* (2016), pp. 96–101. DOI: 10.1109/IMIS.2016.46.