

**NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS**

**SCHOOL OF SCIENCE**

**DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION**

**BSc THESIS**

**Malicious Chrome Extensions**

**ILIAS A. RAFAIL**

|  |  |
| --- | --- |
| **Supervisor (or supervisors):** | **Name Surname, Title** |

**ATHENS**

**MONTH YEAR**



**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

**Κακόβουλες Επεκτάσεις του Chrome**

**Ηλίας Α. Ραφαήλ**

|  |  |
| --- | --- |
| **Επιβλέπων (ή Επιβλέπουσα ή Επιβλέποντες):** | **Όνομα Επώνυμο,** Τίτλος (π.χ Αναπληρωτής Καθηγητής) |

**ΑΘΗΝΑ**

**ΜΗΝΑΣ ΕΤΟΣ**

**BSc THESIS**

Malicious Chrome Extensions

**ILIAS A. RAFAIL**

**S.N.:** 1115201300149

|  |  |
| --- | --- |
| **SUPERVISOR:** | **Name Surname, Title** |

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

Κακόβουλες Επεκτάσεις του Chrome

**Ηλίας Α. Ραφαήλ**

**Α.Μ.:** 1115201300149

|  |  |
| --- | --- |
| **ΕΠΙΒΛΕΠΟΝΤΕΣ:** | **Όνομα Επώνυμο,** Τίτλος (π.χ Αναπληρωτής Καθηγητής) |

**ABSTRACT**

Today the use of extensions is widespread for users of all major web browsers such as Chrome. There is also a large increase in the number of personal data that search engines mediate, and as a result they become the target of criminals who, taking advantage of how extensions work, can gather various information from the user without this being perceived.

The goal of this thesis is to implement our own extension that a user can use to get  
informed about suspicious actions on extensions he wants to install from the Chrome webstore.

The way the extension works is initially connects to a Native Messaging Host, which sends information about the identity of the extension that the user has chosen to the application. The application will download the extension's source code and perform a static analysis of the code. Every time a suspicious action is detected, it will inform the user with a corresponding warning message and the file in which the action was detected.

**SUBJECT AREA**: Static analysis of Chrome extension code

**KEYWORDS**: π.χ. malicious/suspicious chrome extensions, chrome webstore, static analysis, native messaging host.

**ΠΕΡΙΛΗΨΗ**

Σήμερα η χρήση επεκτάσεων είναι διαδεδομένη στους χρήστες όλων των μεγάλων  
μηχανών αναζήτησης όπως είναι ο Chrome. Παρατηρείται επίσης μεγάλη αύξηση του πλήθους των προσωπικών δεδομένων τα οποία διατηρούν οι μηχανές αναζήτησης και ως αποτέλεσμα γίνονται στόχος εγκληματιών οι οποίοι εκμεταλλευόμενοι τον τρόπο λειτουργίας των επεκτάσεων μπορούν να συγκεντρώσουν διάφορες πληροφορίες από τον χρήστη χωρίς αυτό να γίνει αντιληπτό.

Στόχος της πτυχιακής εργασίας είναι να δημιουργήσουμε μια δική μας επέκταση την οποία θα μπορεί να χρησιμοποιήσει ο χρήστης για να ενημερωθεί αν υπάρχουν ύποπτες ενέργειες σε επεκτάσεις που θέλει να εγκαταστήσει από το webstore του Chrome.

Ο τρόπος με τον οποίο λειτουργεί η επέκταση είναι αρχικά να συνδέεται με έναν Native Messaging Host, ο οποίος στέλνει την πληροφορία για τη ταυτότητα της επέκτασης που έχει επιλέξει ο χρήστης στην εφαρμογή. Η εφαρμογή με τη σειρά της θα κατεβάσει τον πηγαίο κώδικα της επέκτασης και θα εκτελέσει στατική ανάλυση πάνω στον κώδικα αυτό. Κάθε φορά που εντοπιστεί κάποια ύποπτη ενέργεια θα ενημερώσει τον χρήστη με αντίστοιχο προειδοποιητικό μήνυμα αλλά και το αρχείο στο οποίο εντοπίστηκε η ενέργεια.

**ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ**: Στατική ανάλυση στον κώδικα επεκτάσεων του Chrome

**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ**: κακόβουλες/ύποπτες επεκτάσεις του chrome, chrome webstore, στατική ανάλυση, native messaging host.

*Στη σελίδα αυτή αναφέρονται οι αφιερώσεις. Η σελίδα αυτή είναι προαιρετική.*

**ΕΥΧΑΡΙΣΤΙΕΣ (ή AKNOWLEDGMENTS)**

Στη σελίδα αυτή αναφέρονται οι ευχαριστίες. Η σελίδα αυτή είναι προαιρετική. Παρατίθεται παράδειγμα ευχαριστιών.

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας, θα θέλαμε να ευχαριστήσουμε τους επιβλέποντες, αν. καθ .Ευστράτιο Γεωργιάδη, Γρηγόριο Παπάμαλο, Αναστασία Γούσιου, Ξενοφών Παπαδόπουλο, για τη συνεργασία και την πολύτιμη συμβολή του στην ολοκλήρωση της.

**CONTENTS**

[PREFACE 14](#_Toc460936756)

[1. INTRODUCTION 15](#_Toc460936757)

[1.1 Μορφοποίηση Κειμένου 15](#_Toc460936758)

[1.1.1 Μέγεθος Σελίδας 15](#_Toc460936759)

[1.1.2 Ημερομηνίες 15](#_Toc460936760)

[1.1.3 Εξώφυλλο και 1o Εσώφυλλο (Σελίδα Τίτλου) 15](#_Toc460936761)

[1.1.4 2o Εσώφυλλο (Σελίδα έγκρισης) 16](#_Toc460936762)

[1.1.5 Περίληψη 16](#_Toc460936763)

[1.1.6 Αρίθμηση σελίδων 17](#_Toc460936764)

[1.1.7 Οι σελίδες του Κειμένου 17](#_Toc460936765)

[1.1.8 Ορολογία 18](#_Toc460936766)

[1.1.9 Συντμήσεις- Αρκτικόλεξα 18](#_Toc460936767)

[1.1.10 Βιβλιογραφικές Αναφορές 18](#_Toc460936768)

[1.2 Άλλες Παρατηρήσεις 19](#_Toc460936769)

[1.2.1 Λεζάντες 19](#_Toc460936770)

[1.2.2 Κεφαλίδες και Υποσέλιδα 19](#_Toc460936771)

[2. ΕΙΚΟΝΕΣ, ΠΙΝΑΚΕΣ ΚΑΙ ΣΧΗΜΑΤΑ 20](#_Toc460936772)

[2.1 Παράθεση Πινάκων 20](#_Toc460936773)

[2.2 Παράθεση Σχημάτων 20](#_Toc460936774)

[2.3 Παράθεση Εικόνων 21](#_Toc460936775)

[3. ΣΥΜΠΕΡΑΣΜΑΤΑ 22](#_Toc460936776)

[TABLE OF TERMINOLOGY 23](#_Toc460936777)

[ABBREVIATIONS - ACRONYMS 24](#_Toc460936778)

[ANNEX Ι 25](#_Toc460936779)

[ANNEX ΙΙ 26](#_Toc460936780)

[REFERENCES 27](#_Toc460936781)

**LIST OF FIGURES**

Σχήμα 1: Υπόδειγμα διαμόρφωσης ράχης βιβλιόδετου τόμου 14

Σχήμα 2: Στοιχεία ανάπτυξης Ανατολής, Βορρά και Δύσης 21

**LIST OF IMAGES**

[Εικόνα 1: Ερημικό τοπίο κατά το ηλιοβασίλεμασελ. 17](#_Toc220142383)

**LIST OF TABLES**

[Πίνακας 1: Συντομογραφίες χωρώνσελ. 15](#_Toc220142393)

[Πίνακας 2: Πίνακας ορολογίας με τις αντιστοιχίσεις των ελληνικών και ξενόγλωσσων όρωνσελ. 21](#_Toc220142394)

[Πίνακας 3: Ακρωνύμια και ανάπτυξή τουςσελ. 22](#_Toc220142395)

PREFACE

This thesis took place in Athens of Greece between… It is consisted by three parts, Reading, Implementation and Writing. For the first part, I read some papers in order to familiarize with related work and gather ideas for the subject. The implementation phase was about creating the extension and the application that would perform the analysis. I started creating the part that downloads the extension files (.crx file) the user has chosen and extract them. Then, I created two parsers, one for HTML files and the other for JavaScript, and I parsed each file of the extension. I used the parsing tree to check for code that matches with rules that indicate suspicious activity. In case of a match, I print a warning message that will inform the user about the suspicious action. The application is written in Python and can be further expanded by adding more rules. The final part is about the text which presents the way I worked on the project.

# INTRODUCTION

## Chrome Extension Composition

Google Chrome supports extensions written in JavaScript and HTML. Each extension also contains a mandatory manifest (.json file) that defines a set of properties such as the extension name, description and version number. The manifest is used by the browser to know the functionality offered by the extension and the permissions required to perform

Figure 1: manifest.json file.

## Installing Extensions

The Chrome Web Store is the official means for users to find and install extensions. But in addition to this, extensions can also be installed manually by the user or an external program. Installation of extensions outside the web store is referred as sideloading. In 2014 Chrome took steps to prevent sideloading by requiring all installed extensions to be hosted in the Chrome Web Store. However, it is still possible for programs to force silent installation of extensions, since the attacker already has control of the machine.

## Access Control Settings

Access control settings is a protection mechanism currently used by all browsers (based on Chromium, Firefox and Microsoft Edge). Browsers rely on a set of configuration options included in a manifest file.



Figure 2: Example of manifest containing API permissions,  
content scripts, background script and CSP.

### Permissions

The permission system is designed in the spirit of least privilege, with the goal of limiting the resources available to an extension. It determines which sites an extension can access, the allowed API call, and the use of binary plugins.

In Figure 2 example, the extension requests host permission for https://www.google.com/, which allows it to access cookies and webRequest APIs for the specified domains. Wildcarding can also be used where the extension requests access to \*://\*.facebook.com. This permission allows access to all subdomains of facebook.com via any URL scheme. Additionally, <all\_urls> is a special token used for matching any URL.

### Content Scripts

Content scripts is a list that indicates JavaScript files that will run inside of the web page. The execution of a content script not only can modify the DOM tree of other scripts, but it can also issue authenticated web requests like POST.

In Figure 2, we can see 2 JavaScript files included, that will be run in the context of the page for any URLs matching the http://www.yahoo.com/ pattern.

### Background Pages

Background pages often contain the logic and state an extension needs for the entirety of the browser session and do not have any visibility to the user.

Figure 2 shows how background.js is specified as a background page.

### Content Security Policy

A Content Security Policy (CSP) header is specified by servers and used by the browser in order to determine the sources from which it can include objects of the page. CSP can also specify other options, such as whether to allow the page to perform an eval or to embed inline JavaScript.

Figure 2 shows an example where the extension specifies its CSP in order to include source from foo.com and execute eval.

# RELATED WORK

Modern web browsers are characterized by third-party add-ons that extend their functionality. Chrome browser provide a rich API where extensions can make network requests, access the local file system, get low-level information about running processes and many more. Although a permission system is used by Chrome in order to curtail an extension’s privileges and avoid misuse, malicious extensions can allow attackers to gain access to a wide range of private, sensitive data and computer resources.

At following, we refer related work has been done on detecting malicious behavior in browser extensions and a series of attacks by which malicious extensions can steal data, track user behavior and collude to elevate their privileges. In special, we refer to Hulk which is a tool for detecting malicious behavior in Google Chrome extensions.

## Hulk Architecture

Hulk is a dynamic analysis system that detects malicious behavior in Chrome extensions. It dynamically loads extensions in a monitored environment and observes the interaction of the extension with the loaded web pages. It uses a set of heuristics to identify potentially dangerous behavior and label extensions as malicious, suspicious or benign.

### HoneyPages

There are extensions that their activation is based on the content of a web page. In order to analyze such cases, Hulk uses HoneyPages which are specially crafted pages that dynamically create an environment for the extension to perform the actions it needs. That means when an extension queries for the presence of a specific element, such as an iframe DOM element, the HoneyPage will create the element, inject it in the DOM tree and return it to the extension.

### Fuzzer

Extensions can register callbacks that respond to certain browser-level events using an event-based model such as the chrome.webRequest API. HoneyPages are not able to trigger callbacks for network events that require special properties like a specific URL or HTTP header. Therefore, Hulk uses event handler fuzzing where all event callbacks are invoked with mock event objects. At the same time, a HoneyPage is loaded in the active tab which enables Hulk to monitor all changes that the extension attempts to make on the page.

## Malicious Behavior Detection

In Hulk’s presentation there were cases of extensions abusing Chrome’s extension API. Specifically, by monitoring the chrome.management.uninstall API calls, malicious behavior was detected where an extension uninstalled other extensions.

A malicious extension can also interfere with tabs that point to the extension configuration page (chrome://extensions) either by replacing the URL with a different one, or by removing the tab completely. As a result, the malicious extension denies the access to Chrome’s extension configuration page and the user is unable to uninstall any extension.

Using callbacks in the webRequest API, a malicious extension can manipulate HTTP headers. Extensions can use the webRequest API to effectively perform a man-in-the-middle attack on HTTP requests and responses before they are handled by the browser. This behavior is often malicious (or at least dangerous) since there are cases of extensions that remove security-related headers, such as Content-Security-Policy or X-Frame-Options, through the use of callbacks such as webRequest.onHeadersReceived and webRequestInterval.eventHandled.

In addition to dropping security related headers, extensions can change or add parameters in URLs before the HTTP request is sent. Such suspicious behavior is common, especially among extensions that request permissions on shopping related sites such as Amazon, eBay, and others.

# METHOLOGY AND DESIGN

## Native Messaging Host

Extensions and apps can exchange messages with native applications using a messaging passing API. The native application must register a native messaging host that knows how to communicate with the extension. Chrome starts the host in a separate process and communicates with it using standard input and standard output streams.

We use this technique to send the extension ID that the user chose to the native application which will download the source code and run the static analysis.

## Parsers

As we know, extensions support JavaScript and HTML. Malicious code can be found in both, therefore we used two parsers, one for each type. We use the Abstract Syntax Tree as a representation of the abstract syntactic structure of the source code written in each file of the extension.

## Rules

Now that we have the Abstract Syntax Tree (AST), we can start adding rules. As rule we describe a set of elements, such as call expressions, variables, keywords etc., that indicate suspicious activity when they are found in the source code.

In order to define a rule, we use lists and flags.

The callees list contains all callees of a call expression. For example, if the call expression is chrome.webRequest.onBeforeRequest.addListener, the list will contain: [chrome, webRequest, onBeforeRequest, addListener].

The CallExpression list contains all the call expressions or in other words the list of the callees that we described before.

The decs list contains all the callees of a variable declaration. For example, if the variable declaration is var document.forms, the list will contain: [document, forms].

The Vars list contains all the variable declarations, or in other words the decs list that we already described.

# IMPLEMENTAYION

# EVALUATION

# CONCLUSION

Στο τέλος της εργασίας υπάρχουν τα συμπεράσματα που προκύπτουν από την έρευνα.

# 

# REFERENCES

1. I.E. Sutherland, R.F. Sproull, and R.A. Schumaker, A Characterization of 10 Hidden-Surface Algorithms, *ACM Computing Surveys*, vol. 6, no. 1, Mar. 1974, pp.1-55.
2. W.M. Newman and R.F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill, 1979, p. 402.
3. M.A., Arbib, ed., *The Handbook of Brain Theory and Neural Networks*, MIT Press,1998.
4. M. Weiser, “Program Slicing”, *Proc. 14th Int'l Conf. Data Eng. (ICDE 98)*, IEEE CS Press, 1998, pp. 439-449.
5. Y. Yao et al., “Web Intelligence (WI): Research Challenges and Trends in the New Information Age*”*, *Web Intelligence: Research and Development, LNAI 2198*, N. Zhong et al., eds., Springer-Verlag, 2001, pp. 1-17.
6. R. Focardi and R. Gorrieri, eds., *Foundations of Security Analysis and Design*, LNCS 2171, Springer-Verlag, 2001.
7. B. Fagin, “A Parallel Execution Model for Prolog”, doctoral dissertation, Dept. Computer Sciences, Univ. California, Berkeley, 1987.
8. Ν. Πλατής, «Τεχνικές πολλαπλών αναλύσεων στην απλοποίηση τριγωνικών και τετραεδρικών πλεγμάτων», Διδακτορική Διατριβή, Τμήμα Πληροφορικής & Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, 2005.
9. M. Nichols, “The Graphical Kernel System in Prolog”, master's thesis, Dept. Computer Science and Eng., Rensselaer Polytechnic Inst., 1985.
10. Β. Πρεμέτη, «Ασφάλεια στην ηλεκτρονική διακυβέρνηση μεθοδολογία και διεθνής εμπειρία», Διπλωματική Εργασία, Τμήμα Πληροφορικής & Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, 2003.
11. F. Kaplan, “From Baghdad to Manila: Another Lousy Analogy for the Occupation of Iraq”, *Slate*, 21 Oct. 2003; <http://slate.msn.com/id/2090114/>. [Προσπελάστηκε 15/1/08]
12. R. Bartle, “Early MUD History» Nov. 1990; [www.ludd.luth.se/mud/aber/mud-history.html](http://www.ludd.luth.se/mud/aber/mud-history.html) [Προσπελάστηκε 15/1/08]
13. *Recommendation ITU-R BT.601, Encoding Parameters of Digital Television for Studios,* Int'l Telecommunications Union, 1992.
14. *IEEE Std. 1596-1992, Scalable Coherent Interface (SCI)*, IEEE, 1992.
15. *MPEG-21 Overview*, ISO/MPEG N5231, MPEG Requirements Group, Oct. 2002.
16. *N. Mitra, SOAP v.1.2, Part 0: Primer,* World Wide Web Consortium (W3C) recommendation, June 2003; <http://www.w3.org/TR/soap12-part0/> [Προσπελάστηκε 15/1/08]
17. D. Box et al., *Simple Object Access Protocol (SOAP)* 1.1 World Wide Web Consortium (W3C) note, May 2000; <http://www.w3.org/TR/SOAP/>. [Προσπελάστηκε 15/1/08]
18. *Unix System V Interface Definition*, no. 2, vol. 2, AT&T, Murray Hill, N.J., 1986.