**ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ**

**ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ**

**ΠΑΡΟΥΣΙΑΣΗ / ΕΞΕΤΑΣΗ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ**

**Πλέλης Κωνσταντίνος**

**Μεταπτυχιακός Φοιτητής**

**Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης**

Επόπτης Μεταπτ. Εργασίας: Καθηγητής Ε. Μαρκάτος

**Παρασκευή, 31/3/2017, 14:00**

**Αίθουσα Τηλεδιάσκεψης Κ206, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης**

# **“ Υλοποίηση και μέτρηση απόδοσης μηχανισμών ασφαλείας στο ChakraCore, μια σύγχρονη μηχανή JavaScript"**

## ΠΕΡΙΛΗΨΗ

Το λογισμικό, όπως οι φυλλομετρητές διαδικτύου, οι εφαρμογές προβολής και επεξεργασίας εγγράφων, μπορούν να επεκτείνουν την λειτουργικότητά τους υποστηρίζοντας την εκτέλεση script σε ένα εικονικό περιβάλλον. Για παράδειγμα, όλοι οι φυλλομετρητές διαδικτύου υποστηρίζουν την εκτέλεση κώδικα JavaScript. Αρχικά, η εκτέλεση γινόταν σε ένα διερμηνέα. Ωστόσο, η εκτέλεση script σε διερμηνέα έχει μειωμένη απόδοση σε σύγκριση με την εκτέλεση κώδικα μηχανής, με αποτέλεσμα οι εξελιγμένες μηχανές JavaScript να υποστηρίζουν πλέον παραγωγή εντολών κώδικα μηχανής σε πραγματικό χρόνο.

Η παραγωγή κώδικα σε πραγματικό χρόνο εισάγει νέο κώδικα στην διεργασία που εκτελείται, ο οποίος, εκτός αν προστατευθεί κατάλληλα, μπορεί να δημιουργήσει επιπρόσθετα ρίσκα ασφαλείας. Η προστασία αυτού του δυναμικού κώδικα είναι περισσότερο πολύπλοκη σε σχέση με την προστασία εκτελέσιμων ή πηγαίου κώδικα. Ενώ τα εκτελέσιμα και ο πηγαίος κώδικας μπορούν να αναλυθούν στατικά και να εφαρμοσθούν πολύπλοκοι αλγόριθμοι, ο δυναμικός κώδικας μπορεί μόνο να αναλυθεί κατα τη διάρκεια της εκτέλεσης. Αυτό είναι πολύ σημαντικό, καθώς ο λόγος ύπαρξης του δυναμικού κώδικα είναι η γρηγορότερη εκτέλεση, κάτι που μπορεί να μειωθεί σημαντικά με την εφαρμογή τέτοιων αλγορίθμων.

Σε αυτή την εργασία, εφαρμόζουμε ενα σύνολο από προτεινόμενες αμυντικές τεχνικές σε μία σύγχρονη μηχανή JavaScript, το ChakraCore, και μελετάμε την επίδρασή τους στην απόδοση. Εφαρμόζουμε διαφορετικές λύσεις και με διάφορους συνδυασμούς, και δείχνουμε πως συγκεκριμένες τεχνικές μπορούν να μειώσουν σημαντικά το κέρδος που προσφέρει η παραγωγή κώδικα σε πραγματικό χρόνο.

Ανάμεσα στις άμυνες που αξιολογούμε είναι το Control Flow Guard, μια καινοτόμος υλοποίηση Control Flow Integrity από τη Microsoft, η οποία είναι διαθέσιμη αποκλειστικά σε Windows 8.1 και Windows 10. Κυρίως επικεντρωνόμαστε στο να επεκτείνουμε τον υπάρχοντα μηχανισμό κωδικοποίησης σταθερών κατα την παραγωγή κώδικα, καθώς και εισάγουμε την κωδικοποίηση σταθερών που δημιουργούνται έμμεσα, όπως για παράδειγμα αυτές που εμφανίζονται κατά την παραγωγή κώδικα σε μπλοκ υπό συνθήκη. Ο στόχος των αμυντικών τεχνικών που υλοποιούμε και αναλύουμε είναι να αποτρέψουν επιθέσεις Return Oriented Programming (ROP), είτε σταματώντας την παραβίαση της ροής εκτέλεσης του προγράμματος, είτε ανεβάζοντας τον πύχη για τον επιτιθέμενο κατά την κατασκευή των απαραίτητων ROP gadgets.

**Plelis Konstantinos**

**M.Sc. Thesis**

**Computer Science Department**

**University of Crete**

**Master's Thesis Supervisor: Professor E. Markatos**

**Friday, 31/3/2017, 14:00**

**Room K206, Computer Science dept., University of Crete**

**“Implementation and performance evaluation of security defenses in ChakraCore, a state of the art JavaScript engine”**

## ABSTRACT

Software, such as web browsers, document viewers and processors, extend their functionality by running scripts in a virtualized environment. For example, all web browsers support executing JavaScript code. Initially the execution was performed using an interpreter. However, interpreted script execution suffers in performance compared to running native code, therefore most sophisticated engines support Just-in-Time (JIT) compilation of bytecode to native instructions.

JIT compilation introduces new code in the running process, which, unless correctly hardened, can pose additional security risks. Hardening JIT code is much more complicated compared to hardening binaries or source code. While binaries and source code are analyzed off-line, and complex algorithms can be of use, JIT code can only be analyzed at runtime. This is important, since JIT compilation happens primarily for faster execution and, thus, such complex algorithms can reduce the performance gain significantly.

In this thesis, we apply a set of proposed hardening solutions in a state of the art JavaScript engine, ChakraCore, and we evaluate all defenses in terms of performance. We apply different solutions and in various combinations, and we demonstrate how certain defenses can reduce the performance gain of the JIT engine significantly.

Among the defenses we evaluate is Control Flow Guard, a novel Control Flow Integrity security mitigation by Microsoft, available exclusively on Windows 8.1 and Windows 10. Mainly, we focus on extending the existing constant blinding mechanism, as well as introducing blinding of implicit constants, such as those produced in code generated from conditional blocks. The target of the mitigations that are implemented and analyzed is to prevent Return Oriented Programming (ROP) attacks, either by stopping the initialization of the exploit or by raising the bar for the attacker to generate the necessary ROP gadgets.