ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Η απειλή του Spectre στα σύγχρονα υπολογιστικά συστήματα

Γεώργιος Χ. Τερζάκης

Επιβλέπουσα: **Μέμα Ρουσσοπούλου,** Αναπληρώτρια Καθηγήτρια

ΑΘΗΝΑ

ΝΟΕΜΒΡΙΟΣ 2018

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Η απειλή του Spectre στα σύγχρονα υπολογιστικά συστήματα

Γεώργιος Χ. Τερζάκης

Α.Μ.: 1115200900249

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Μέμα Ρουσσοπούλου,** Αναπληρώτρια Καθηγήτρια

ΠΕΡΙΛΗΨΗ

Στόχος αυτής της βιβλιογραφικής πτυχιακής είναι η ανάλυση του ζητήματος ασφαλείας που αφορά σχεδόν όλους τους επεξεργαστές, λόγω του τρόπου λειτουργίας τους, που προέκυψε από το Spectre. Το Spectre έγινε γνωστό δημοσίως με την δημοσίευση του αντίστοιχου επιστημονικού άρθρου τον Ιανουάριο του 2018 από τον Jann Horn του Project Zero της Google μαζί με τους Paul Kocher και σε συνεργασία με τους Daniel Genkin, Mike Hamburg, Moritz Lipp και Yuval Yarom. Ώστοσο η ύπαρξη αυτού του θέματος ασφαλείας είχε γνωστοποιηθεί ήδη από την Google στους προμηθευτές και τις εταιρίες επεξεργαστών 7 μήνες πριν την έκδοση του επιστημονικού άρθρου, από τον Ιούνιο του 2017. Το Spectre βασίζει την λειτουργία του στην υποθετική εκτέλεση (speculative execution), εκτός σειράς εκτέλεση (out-of-order executing) και πρόβλεψη διακλάδωσης (branch prediction) που χρησιμοποιούν οι επεξεργαστές για να μεγιστοποιήσουν τις επιδόσεις τους. Παρ'όλα τα πλεονεκτήματα που προσφέρουν αυτές οι λειτουργίες στους επεξεργαστές (π.χ παράλληλη εκτέλεση προγραμμάτων, ταχύτητα στην εκτέλεση, βελτιστοποίηση στην χρήση των πόρων του συστήματος κ.α), επιτρέπουν στο Spectre, με τον τρόπο που λειτουργεί, να έχει τυχαία πρόσβαση στο χώρο μνήμης (memory space) των προγραμμάτων, με στόχο την πιθανή απόκτηση πληροφορίας η οποία μπορεί να περιέχει προστατευμένα δεδομένα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ασφάλεια Η/Υ

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: ζήτημα ασφαλείας, τρωτότητα, επεξεργαστής, Spectre, πρόγραμμα

ABSTRACT

The subject of this diploma thesis is the analysis of the CPU security issue, which emerged from the Spectre vulnerability exploits and affects nearly all modern processors. The Spectre bug was made public with the release of the scientific paper from Jann Horn of Google's Project Zero with Paul Kocher in collaboration with Daniel Genkin, Mike Hamburg, Moritz Lipp and Yuval Yarom on January 2018. However, Google had already informed all the affected companies and hardware vendors about that flaw on June 2017, seven months prior to the publication of the paper. Spectre exploits some basic CPU actions such as speculative execution, out-of-order executing and branch prediction. While those actions are quite beneficial features for the CPU (parallel execution, execution speed, optimization in system's resource management etc.), they allow Spectre to trick a program into accessing arbitrary locations in the program's memory space, in order to read the content of accessed memory, and thus potentially obtain sensitive data.

SUBJECT AREA: Computer Security

KEYWORDS: security issue, vulnerability, CPU, Spectre, program

Η παρούσα πτυχιακή εργασία είναι αφιερωμένη στη μητέρα μου.

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας πτυχιακής εργασίας θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια κ. Μέμα Ρουσσοπούλου για την συνεργασία της και τις πολύτιμες συμβουλές της που συνέβαλλαν στην ολοκλήρωσή της.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία εκπονήθηκε στο τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού Καποδιστριακού Πανεπιστημίου Αθηνών στο πλαίσιο της Πτυχιακής Εργασίας Ι που απαιτείται για την λήψη πτυχίου. Η έναρξη και ολοκλήρωση της εργασίας πραγματοποιήθηκε κατά την διάρκεια του χειμερινού ακαδημαϊκού εξαμήνου 2018-2019.

1. ΕΙΣΑΓΩΓΗ

Σε αυτή την ενότητα θα γίνει μία σύντομη παρουσίαση του υλικού και των λειτουργιών του επεξεργαστή και άλλων μερών του συστήματος, που είναι απαραίτητο να γνωρίζουμε ώστε να καταλάβουμε καλύτερα τη λειτουργία του υπολογιστή και κατ' επέκταση το πώς δουλεύει το Spectre, τί καταστάσεις εκμεταλλεύεται στον επεξεργαστή και στις μνήμες και πώς μέσα από αυτή τη λειτουργία του, επιτυγχάνει να υποκλέψει δεδομένα που θα έπρεπε υπό κανονικές συνθήκες να είναι προστατευμένα.

1.1 Επεξεργαστής και Μνήμες

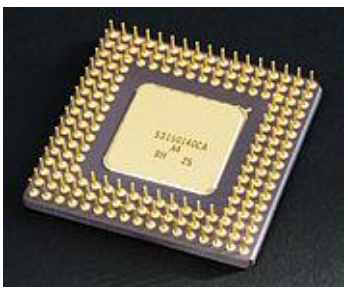
1.1.1 Κεντρική Μονάδα Επεξεργασίας (Central Processing Unit - CPU)

Ο επεξεργαστής(ή μικροεπεξεργαστής εάν είναι το μοναδικό ολοκληρωμένο κύκλωμα σε ένα σύστημα) είναι το κυριότερο τμήμα ενός υπολογιστικού μηχανήματος. Διαχειρίζεται δεδομένα, ελέγχει τις λειτουργίες του συστήματος και εκτελεί λειτουργίες διασύνδεσης και μεταβίβασης εντολών. Όπως είναι γνωστό, οι επεξεργαστές δεν σχετίζονται μόνο με τους υπολογιστές αλλά τους συναντάμε σε κάθε είδους ηλεκτρονική συσκευή, στην οποία απαιτείται η ύπαρξη υπολογιστικής ικανότητας.

Στην σημερινή εποχή, οι περισσότεροι επεξεργαστές υπολογιστών είναι μικροεπεξεργαστές, αφού εμπεριέχονται σε ένα τσιπ ολοκληρωμένου κυκλώματος, το οποίο περιλαμβάνει επιπλέον τις μνήμες, περιφεριακές διασυνδέσεις και άλλα υπολογιστικά τμήματα. Ένα τέτοιο πλήρες κύκλωμα ονομάζεται μικροελεγκτής (microcontroller) ή απλά σύστημα σε τσιπ (system on a chip - SoC).

Τέλος, οι περισσότεροι υπολογιστές χρησιμοποιούν πολυ-πύρηνους επεξεργαστές (multi-core processors), στους οποίους ένα τσιπ περίεχει 2 ή περισσότερους πυρήνες επεξεργασίας. Σε αυτή την περίπτωση τα τσιπ ονομάζονται υποδοχείς (sockets).

Εικόνα 1: Η πάνω όψη του επεξεργαστή Intel 80486DX2



Εικόνα 2: Η κάτω όψη του επεξεργαστή Intel 80486DX2

Εικόνα 3: Το σύστημα σε τσιπ (SoC) Raspberry Pi

1.1.2 Μνήμες

Η μνήμη ενός υπολογιστικού συστήματος είναι η τεχνολογία που του επιτρέπει να αποθηκεύει δεδομένα και πληροφορίες. Είναι μία από τις βασικότερες λειτουργίες ενός υπολογιστή και οργανώνεται σε διαφορετικά επίπεδα με βάση την ιεραρχία της μνήμης, που ορίζεται από το πόσο κοντά βρίσκονται τα μέσα αποθήκευσης και πόσο γρήγορα γίνεται η προσπέλαση των δεδομένων από τον επεξεργαστή. Επιπλέον όσο πιο κοντά βρίσκεται η μνήμη στον επεξεργαστή (και κατ' επέκταση η προσπέλαση είναι

γρηγορότερη) τόσο πιο ακριβή (από θέμα τιμής) και πιο μικρή (από θέμα αποθηκευτικής ικανότητας) γίνεται. Η ιεραρχία της μνήμης είναι η ακόλουθη:

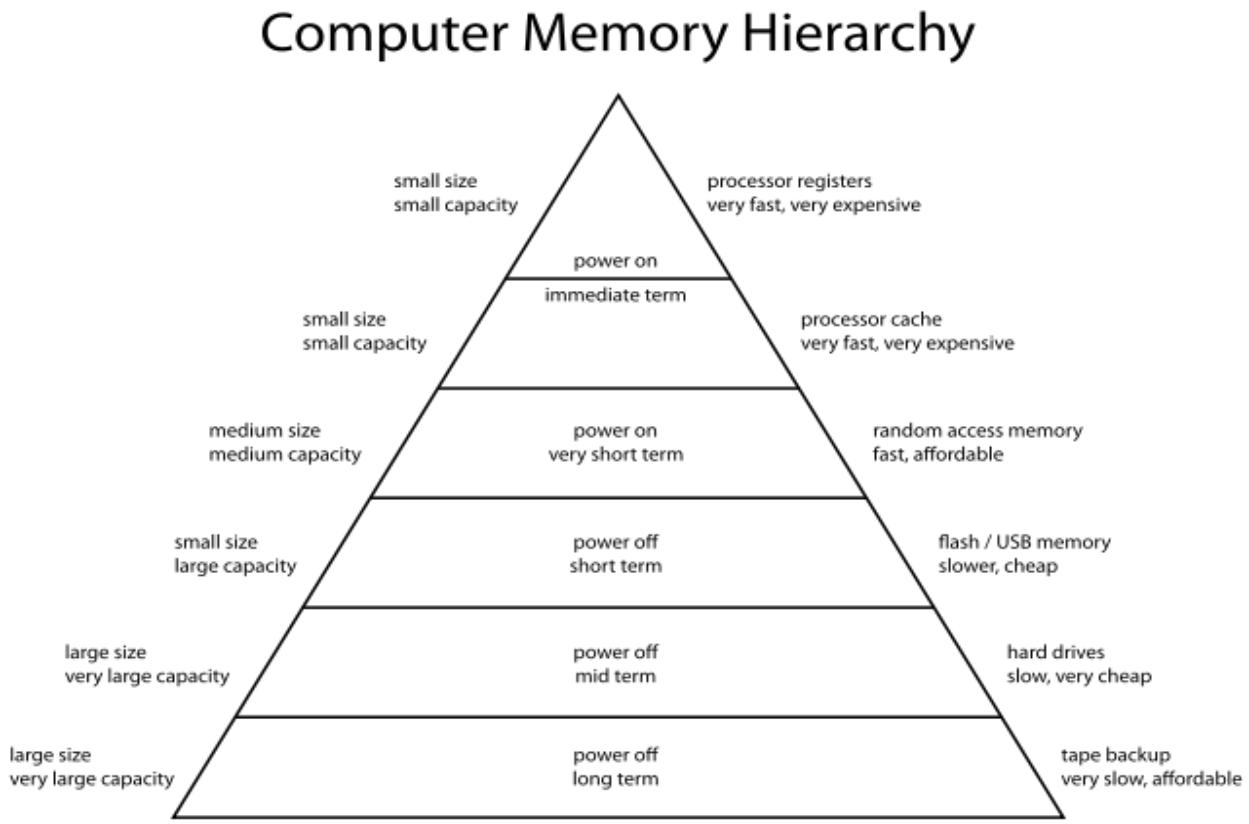
- Καταχωρητές (Registers): Αυτός ο τύπος της μνήμης βρίσκεται μέσα στο τσιπ του επεξεργαστή και είναι η γρηγορότερη μονάδα μνήμης του υπολογιστικού συστήματος. Συνήθως κάθε καταχωρητής έχει αποθηκευτικό χώρο όσο το μέγεθος μίας λέξης δεδομένων (32, 64 ή 128 bits).

- Κρυφή μνήμη (Cache Memory): Είναι η μνήμη που μεσολαβεί μεταξύ των καταχωρητών και της κατά πολύ πιο αργής κύριας μνήμης. Στη κρυφή μνήμη αποθηκεύονται δεδομένα τα οποία είναι αντίγραφα τιμών που βρίσκονται είτε στην κύρια μνήμη, ώστε να γίνεται πιο γρήγορα η προσπέλαση από τους καταχωρητές, είτε στους καταχωρητές, ώστε να γίνεται αναναίωση των τιμών στην κύρια μνήμη. Ο κύριος στόχος αυτής της μνήμης είναι καθαρά η ταχύτητα μεταφοράς δεδομένων με στόχο την αύξηση της επίδοσης των επεξεργαστών. Γι' αυτό το λόγο οι περισσότεροι μοντέρνοι επεξεργαστές έχουν κρυφές μνήμες 3 ή και 4 επιπέδων. Όσο πιο υψηλό είναι το επίπεδο της κρυφής μνήμης τόσο πιο μεγάλη χωρητικότητα έχει.

- Κύρια Μνήμη (Random Access Memory - RAM): Αυτή η μνήμη είναι αρκετά πιο αργή από τις προηγούμενες και είναι πτητική (volatile). Αυτό σημαίνει ότι τα δεδομένα που αποθηκεύονται, παύουν να υπάρχουν όταν σταματίσει η τροφοδοσία ρεύματος στο σύστημα. Γι' αυτό το λόγο είναι προσωρινές διατάξεις αποθήκευσης δεδομένων και επιτρέπουν την πρόσβαση στα δεδομένα στον ίδιο χρόνο οπουδήποτε και αν βρίσκονται, δηλαδή με "τυχαία πρόσβαση". Σε αυτή την μνήμη αποθηκεύονται προγράμματα και δεδομένα ώστε να εκτελεστούν ή να επεξεργαστούν.

Εικόνα 4: Ένα σετ μνημών τυχαίας προσπέλασης (RAM)

Επιπλέον, υπάρχουν και οι δευτερογενείς μνήμες. Σε αυτή την κατηγορία ανήκουν οι σκληροί δίσκοι (hard drives), τα φλασάκια(usb flash drives), οι δίσκοι εγγραφής(CD/DVD ROMS) κ.α. Οι επεξεργαστές δεν έχουν άμεση επαφή με τέτοιου είδους μνήμες, αλλά ο υπολογιστής προσφέρει αντίστοιχες διεπαφές για την αποθήκευση και μεταφορά δεδομένων σε αυτού του τύπου μνημών. Το κύριο χαρακτηριστικό τους είναι ότι η μεταφορά δεδομένων είναι πολύ αργή (κόστος σε ταχύτητα), αλλά είναι φθηνοί και αρκετά μεγάλοι σε χωρητικότητα. Ένα άλλο χαρακτηριστικό τους είναι ότι σε αντίθεση με την κύρια μνήμη τα δεδομένα που αποθηκεύονται δεν επιρρεάζονται από την παροχή ρεύματος, δηλαδή είναι μη πτητικές (non-volatile) μνήμες.



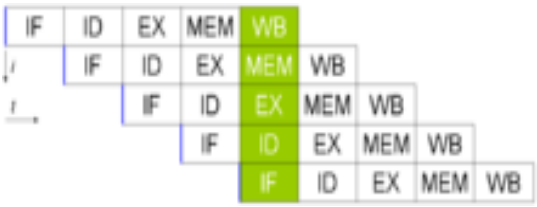
Σχήμα 1: Η ιεραρχία μνήμης του υπολογιστή

1.2 Λειτουργίες Βελτιστοποίησης Επεξεργαστή

Για την ταχύτερη λειτουργία του επεξεργαστή, ένα υπολογιστικό σύστημα δεν βασίζεται μόνο στην ταχύτητα που του προσφέρουν οι μνήμες. Ακόμα και ο ίδιος ο επεξεργαστής χρησιμοποιεί ένα σύνολο λειτουργιών που του επιτρέπει να εκτελεί παράλληλα διαφορετικές εντολές, αλλά και να μπορεί να προβλέψει το τί δεδομένα μπορούν να ζητηθούν στο μέλλον ώστε να τα έχει προσκομίσει πιο νωρίς από τις αντίστοιχες μνήμες που μπορεί να βρίσκονται. Με αυτό το τρόπο, αυξάνεται η ταχύτητα εκτέλεσης των προγραμμάτων και ως επέκταση η αποδοτικότητα του συστήματος. Οι σημαντικότεροι μέθοδοι και λειτουργίες του επεξεργαστή, γι' αυτό τον σκοπό είναι οι παρακάτω.

1.2.1 Σωλήνωση (Pipeline)

Είναι η μέθοδος που χρησιμοποιείται για να εκτελούνται διαφορετικές εντολές ταυτόχρονα. Η βασική ιδέα αυτής της μεθόδου είναι ο παραλληλισμός των εντολών στο υλικό του επεξεργαστή, ώστε κάθε μέρος του να μην βρίσκεται σε κατάσταση αδράνειας. Αυτό επιτυγχάνεται με το "σπάσιμο" των εντολών σε 5 διαφορετικά στάδια, το στάδιο της προσκόμισης εντολής(IF), το στάδιο της αποκωδικοποίησης και ανάγνωσης καταχωρητών(ID), το στάδιο της εκτέλεσης της εντολής ή του υπολογισμού διεύθυνσης(EX), το στάδιο προσπέλασης μνήμης(MEM) και το στάδιο επανεγγραφής αποτελέσματος σε καταχωρητές(WB).



Σχήμα 2: Τα 5 στάδια της σωλήνωσης

1.2.2 Εκτέλεση εκτός σειράς(Out-Of-Order Execution)

Παρ'ότι η σωλήνωση βοηθά αρκετά στον παραλληλισμό των εντολών, υπάρχουν περιπτώσεις που κάποιο στάδιο της σωλήνωσης μπορεί να είναι αδρανές. Αυτό οφείλεται σε κινδύνους εξάρτησης (dependency hazards) εντολών μεταξύ τους, δηλαδή μία εντολή που έπεται, χρειάζεται το αποτέλεσμα μίας εντολής που προηγείται και βρίσκονται σε διπλανά στάδια. Σε αυτή την περίπτωση ο επεξεργαστής αναγκαστικά θα πρέπει να περιμένει τουλάχιστον ένα κύκλο ρολογιού, ώστε το αποτέλεσμα της προηγούμενης εντολής να εγγραφεί στον καταχωρητή και στην συνέχεια να μπορεί να χρησιμοποιηθεί σαν είσοδος στην επόμενη εντολή. Σε αυτές τις περιπτώσεις εμφανίζεται αδράνεια η οποία μπορεί να αντιμετωπισθεί με την εκτέλεση εντολών εκτός σειράς.

Συγκεκριμένα, είναι μία μέθοδος υψηλής απόδοσης των επεξεργαστών στην οποία εκτελούνται οι εντολές, όχι με την σειρά που ορίζονται αυστηρά από το αντίστοιχο πρόγραμμα, αλλά με μία σειρά που ορίζεται από τον επεξεργαστή (αναδιάταξη εντολών), ώστε να αποφεύγονται τέτοιου είδους εξαρτήσεις μεταξύ των εντολών. Στην ουσία εάν εμφανιστεί μία τέτοια εξάρτηση, τότε για να μην υπάρξει αδρανές στάδιο, η αμέσως επόμενη εντολή που εκτελείται αντικαθίσταται με μία εντολή η οποία θα μπορεί να εκτελεσθεί άμεσα και ανεξάρτητα, και στην συνέχεια και αφού έχει ολοκληρωθεί η εγγραφή του αποτελέσματος από την προηγούμενη εντολή, θα εκτελεσθεί η "εξαρτημένη" εντολή σε δεύτερο χρόνο.

Σε αυτή την μέθοδο, ο τρόπος που εκτελούνται οι εντολές εξαρτώνται από την διαθεσιμότητα των δεδομένων και των καταχωρητών που απαιτούνται για την εκτέλεσή τους και όχι απαραίτητα με την αυστηρή σείρα στην οποία εμφανίζονται μέσα στο εκτελέσιμο πρόγραμμα. Με αυτό το τρόπο ο επεξεργαστής μπορεί να καταπολεμίσει τυχόν αδράνειες που εμφανίζονται κατά την εκτέλεση προγραμμάτων.

1.2.3 Πρόβλεψη Διακλάδωσης (Branch Prediction)

Είναι ένας μηχανισμός, στόχος του οποίου είναι η πρόβλεψη για το αν μία διακλάδωση στο πρόγραμμα (π.χ δομή if-then-else), πρέπει να εκτελεσθεί ή όχι. Στην ουσία πρόκειται για ακόμη μία μέθοδο που συμβάλλει στην ομαλή ροή της εκτέλεσης εντολών. Η πρόβλεψη διακλάδωσης κατέχει ένα σημαντικότατο ρόλο στην επίτευξη υψηλών επιδόσεων συστημάτων που κάνουν χρήση επεξεργαστών με σωλήνωση. Χωρίς αυτό τον μηχανισμό, οι επεξεργαστές θα ήταν κατά πολύ πιο αργοί, αφού ένα μεγάλο μέρος όλων των εκτελέσιμων προγραμμάτων αποτελείται από διακλαδώσεις. Σε αυτή την περίπτωση ο επεξεργαστής θα έπρεπε να περιμένει έως ότου η εντολή διακλάδωσης περάσει από το τρίτο στάδιο της σωλήνωσης (στάδιο EX), ώστε να γνωρίζει αν θα πρέπει να συνεχίσει με τις εντολές που αφορούν την διακλάδωση (branch taken) ή πρέπει να τις προσπεράσει (branch not taken).

Για την διαδικασία της πρόβλεψης μίας διακλάδωσης, ο επεξεργαστής χρησιμοποιεί τον Πίνακα Ιστορικού Διακλάδωσης (Branch History Table). Σε αυτό τον πίνακα αποθηκεύεται όλο το ιστορικό των διακλαδώσεων από την αρχή του προγράμματος. Με βάση αυτού, ο επεξεργαστής μπόρει να "μαντέψει" την μελλοντική συμπεριφορά του προγράμματος σε θέματα διακλαδώσεων. Κατά αυτό τον τρόπο μπορεί να αντιμετωπιστεί η αδράνεια σε περιπτώσεις διακλαδώσεων. Παρ' όλα αυτά, η επιτυχία πρόβλεψης δεν είναι ποτέ 100%. Αν και υπάρχουν πολλών ειδών Προβλέπτες Διακλαδώσεων (Branch Predictors), οι περισσότεροι από τους οποίους πραγματοποιούν σωστές προβλέψεις σε ποσοστά πάνω από 90%, πάντα θα υπάρχουν και οι περιπτώσεις στις οποίες η πρόβλεψη θα είναι λανθασμένη. Σε αυτή την περίπτωση, υπάρχει ποινή αποτυχίας πρόβλεψης (μικρή αύξηση του χρόνου εκτέλεσης εντολής), αφού η μέχρι εκείνη την στιγμή λανθασμένη εντολή που βρίσκονταν στο στάδιο της εκτέλεσης, αφαιρείται και το πρόγραμμα συνεχίζει να εκτελείται από τη σωστή αυτή τη φορά εντολή.

1.2.4 Υποθετική Εκτέλεση (Speculative Execution)

Είναι μία τεχνική βελτιστοποίησης στην οποία ένα υπολογιστικό σύστημα εκτελεί εντολές τις οποίες μπορεί εν' τέλη να μην τις χρειαστεί. Όπως εξηγήθηκε ήδη, ο επεξεργαστής εκτελεί εργασίες πριν του ζητηθούν για να αποτρέψει τυχόν καθυστερήσεις. Εάν στην συνέχεια αποδειχθεί ότι κάποια εργασία ήταν περιττή, τότε όλες οι αλλαγές που προκλήθηκαν από την συγκεκριμένη εργασία διαγράφονται και τα αντίστοιχα αποτελέσματα παραβλέπονται.

Ένα τέτοιο παράδειγμα είναι και η πρόβλεψη διακλάδωσης που αναλύθηκε προηγουμένως. Όταν υπάρχει λάθος πρόβλεψη, η εντολή η οποία δεν χρειάζεται να εκτελεσθεί, λόγω μη λήψης της διακλάδωσης, μπορεί να έχει ήδη εκτελεσθεί και μετά να οδηγηθούμε σε ποινή πρόβλεψης και επαναφορά αρχικής κατάστασης (rollback). Ο κύριος στόχος της υποθετικής εκτέλεσης είναι ο ταυτοχρονισμός (concurrency), δηλαδή η εκτέλεση εντολών, αλγορίθμων ή μέρους προγραμμάτων εκτός σειράς και χωρίς να επιρρεάζουν το τελικό αποτέλεσμα, εάν υπάρχουν διαθέσιμοι πόροι στον υπολογιστή. Με λίγα λόγια είναι η ικανότητα να διασπάσουμε ένα πρόγραμμα ή αλγόριθμο σε διαφορετικά μέρη που έχουν την ιδιότητα να είναι ανεξάρτητα μεταξύ τους, ώστε να μπορούν να εκτελεσθούν ταυτόχρονα, χωρίς να επιρρεάζει το ένα το άλλο, με απώτερο σκοπό την βελτιστοποίηση της απόδοσης του υπολογιστικού συστήματος.

1.3 Πώς λειτουργεί η Κρυφή Μνήμη

Ο επεξεργαστής χρησιμοποιεί την τεχνική της φύλαξης (caching) για να επιταχύνει την πρόσβαση του στις μνήμες. Όπως αναφέρθηκε, διαφορετικά είδη μνημών έχουν και διαφορετικούς χρόνους προσπέλασης, οι οποίοι εξαρτώνται από το πόσο κοντά βρίσκονται στη Κεντρική Μονάδα Επεξεργασίας. Έτσι, εάν ο επεξεργαστής χρειάζεται δεδομένα τα οποία βρίσκονται στη Κύρια Μνήμη, θα χρειαστεί πολύ περισσότερο χρόνο για να τα προσπελάσει, από ότι αν βρίσκονταν στους καταχωρητές ή στην Κρυφή Μνήμη του υπολογιστή. Όταν λοιπόν, ξεκινάει την εκτέλεση ένα πρόγραμμα, τότε ο επεξεργαστής φροντίζει να μεταφέρει τα δεδομένα που χρησιμοποιούνται πιο συχνά κατά την εκτέλεσή του, όσο πιο κοντά του γίνεται, δηλαδή στους καταχωρητές ή την κρυφή μνήμη πολλαπλών επιπέδων, ώστε να μειώσει τον χρόνο προσπέλασής τους, με απώτερο στόχο την ταχύτητα στην εκτέλεση.

Αυτή η μεταφορά δεδομένων από αργές μνήμες σε γρήγορες, μπορεί επιπλέον να γίνει και με τις μεθόδους της Υποθετικής Εκτέλεσης, Εκτέλεσης εκτός σειράς και Πρόβλεψης Διακλαδώσεων. Η κύρια ιδέα αυτών των μηχανισμών είναι η απόκτηση δεδομένων και η παραγωγή αποτελεσμάτων, πριν αυτά ζητηθούν από το πρόγραμμα, ώστε να είναι έτοιμα όταν αυτά ζητηθούν εν' τέλη από αυτό. Η χωρητικότητα των καταχωρητών και της κρυφής μνήμης όμως, είναι αρκετά περιορισμένη. Έτσι, όταν δεν υπάρχει άλλος χώρος για δεδομένα, και χρείαζεται να αποθηκευτούν νέα, διαγράφονται ορισμένα παλιά δεδομένα τα οποία δεν χρησιμοποιούνται πλέον ή δεν χρησιμοποιούνται τόσο συχνά για να καταλαμβάνουν χώρο στις συγκεκριμένες μνήμες και αντικαθιστούνται με τα καινούρια.

1.4 Προστατευμένη Μνήμη

Η προστετευμένη μνήμη είναι η βάση της ασφάλειας των υπολογιστών. Η κεντρική ιδέα είναι ότι καμία διεργασία ή πρόγραμμα δεν μπορεί να έχει άμεση πρόσβαση σε τέτοιου είδους δεδομένα χωρίς κάποιου τύπου άδεια από το λειτουργικό σύστημα. Το πιο σύνηθες σφάλμα κατά την απόπειρα πρόσβασης σε προστατευμένα δεδομένα είναι ο ξαφνικός τερματισμός του προγράμματος που επιδίωξε να τα προσπελάσει, ακολουθούμενο από ένα μήνυμα σφάλματος όπως παραβίαση πρόσβασης (access violation), σφάλμα εκτός ορίων (out-of-bounds error), σφάλμα καταμερισμού (segmentation fault) κ.α.

Παρ΄όλα αυτά ο έλεγχος που χρείαζεται να πραγματοποιηθεί για να οριστεί αν ένα πρόγραμμα είναι έγκυρο για να έχει πρόσβαση σε τέτοιου είδους δεδομένα, είναι αρκετά χρονοβόρος. Όμως, εάν κάποια προστατευμένα δεδομένα έχουν αποθηκευτεί ήδη στους καταχωρητές ή στην κρυφή μνήμη, λόγω της υποθετικής εκτέλεσης ή εκτέλεσης εκτός σειράς, τότε τα προγράμματα μπορούν να έχουν πρόσβαση σε αυτά τα δεδομένα πριν γίνει καν έλεγχος για την έγκρισή τους από το λειτουργικό σύστημα για την προσπέλασή τους. Με λίγα λόγια, η ταχύτητα προσπέλασης των δεδομένων από τους καταχωρητές και τις κρυφές μνήμες, είναι πολύ πιο γρήγορη από την διαδικασία εκτέλεσης του ελέγχου δικαιωμάτων (privilege check) για πρόσβαση και προσπέλαση των προστατευμένων δεδομένων.

2. ΤΟ ΖΗΤΗΜΑ ΤΡΩΤΟΤΗΤΑΣ SPECTRE

(SPECTRE VULNERABILITY ISSUE)

2.1 Πρόλογος

Στόχος αυτής της ενότητας είναι η ανάλυση της λειτουργίας του Spectre. Θα γίνει αναφορά στις διαφορετικές εκδοχές του προβλήματος και τί καταστάσεις εκμεταλλεύεται η κάθε μία, ποιές είναι οι επιπτώσεις τους και τέλος ο τρόπος με τον οποίο μπορούν να αντιμετωπιστούν.

2.1.1 Η Βασική Ιδέα

Το Spectre είναι μία τρωτότητα (vulnerability) που επιρρεάζει την πλειοψηφία των μικροεπεξεργαστών που χρησιμοποιούν την πρόβλεψη διακλαδώσεων και την υποθετική εκτέλεση. Πιο συγκεκριμένα, όταν ένας επεξεργαστής παρουσιάσει αποτυχία στην πρόβλεψη μίας διακλάδωσης και αποθηκεύσει δεδομένα (τα οποία μπορεί να είναι και προστατευμένα), θεωρώντας ότι θα τα χρειαστεί άμεσα κατά την διάρκεια της εκτέλεσης ενός προγράμματος στις "γρήγορες" μνήμες, μέχρι να υπολογισθεί η σωστή έκβαση μίας διακλάδωσης και όχι η πρόβλεψη που έγινε από τον επεξεργαστή με βάση τον Πίνακα Ιστορικού Διακλαδώσεων, μπορούν να είναι προσπελάσιμα από άλλα προγράμματα που έχουν πρόσβαση σε αυτές τις μνήμες. Με άλλα λόγια, μία αποτυχία πρόβλεψης διακλάδωσης μπορεί να αφήσει εμφανή "ίχνη" προστατευμένων δεδομένων στους καταχωρητές και την κρυφή μνήμη, τα οποία μπορούν να τα εκμεταλλευτούν άλλα προγράμματα.

Στο σημείο αυτό κρύβεται και η επικινδυνότητα αυτής της τρωτότητας αφού μία τέτοια διακλάδωση μπορεί να αφορά ζητήματα που έχουν σχέση με την ασφάλεια προστατευμένων δεδομένων. Αξίζει να σημειωθεί ότι όταν ο επεξεργαστής αντιληφθεί το λάθος που έγινε κατά την διάρκεια της εκτέλεσης του προγράμματος όσον αφορά κάποια διακλάδωση, τότε θα γίνει επαναφορά του προγράμματος στο σημείο της διακλάδωσης και θα συνεχιστεί η εκτέλεση από την σωστή αυτή τη φορά κατεύθυνση. Παρ'ολα αυτά, η κατάσταση επαναφοράς σε αρχική θέση, που πραγματοποιεί ο επεξεργαστής, αφορά μόνο την εκτέλεση του προγράμματος και όχι ό,τι άλλες ενέργειες πραγματοποιήθηκαν κατά την διάρκεια της λανθασμένης εκτέλεσης εντολών, όπως αποθήκευση κρίσιμων δεδομένων σε μνήμες ή διώρθωση του Πίνακα Ιστορικού Διακλαδώσεων, διότι κάτι τέτοιο θα ήταν αρκετά επιβαρυντικό για την επίδοση του συστήματος.

2.2 Ανάλυση Λειτουργίας του Spectre

2.2.1 Εισαγωγή

Το Spectre αποτελείται από μία ομάδα τρωτοτήτων υλικού (hardware vulnerabilities), τα οποία στοχεύουν στην "εξαπάτηση" προγραμμάτων, ώστε να αποκτήσουν πρόσβαση σε προστατευμένα δεδομένα της μνήμης. Η λειτουργία τους βασίζεται στην ιδέα της υποθετικής εκτέλεσης και συγκεκριμένα εστιάζουν στην περίπτωση της πρόβλεψης διακλάδωσης (Branch Prediction). Όντας πρόβλημα υλικού (hardware), είναι πολύ πιο δύσκολο να αντιμετωπιστεί σε σύγκριση με προβλήματα λογισμικού (software), αφού αντίστοιχες ενημερώσεις και επιδιορθώσεις τέτοιων προβλημάτων (patch) δεν

εξαρτώνται αποκλειστικά σε αλλαγή κώδικα, αλλά σε αλλαγή φιλοσοφίας κατασκευής και αρχιτεκτονικής επεξεργαστών.

Εικόνα 5: Το λογότυπο του Spectre

2.2.2 Εκδοχές του Spectre

Κάθε πρόγραμμα στον υπολογιστή κατά την διάρκεια της εκτέλεσής του, καταλαμβάνει συγκεκριμένο χώρο στην μνήμη έως ότου ολοκληρωθεί. Η μνήμη που αντιστοιχεί σε κάθε πρόγραμμα είναι ξεχωριστή και διακριτή από τα άλλα προγράμματα χωρίς αυτά να έχουν την άδεια να προσπελάσουν ή να προκαλέσουν αλλαγές σε δεδομένα μνήμης τα οποία δεν τους ανήκουν. Σε αυτήν ακριβώς την απομόνωση της μνήμης των προγραμμάτων επικεντρώνεται η λειτουργία του Spectre, προσπαθώντας να την αναιρέσει, παραπλανώντας τα φαινομενικά ασφαλή και χωρίς σφάλματα προγράμματα, τα οποία σε θεωρητικό επίπεδο "ακολουθούν" τις καλύτερες και πιο συμφέρουσες πρακτικές για την ταχύτητα της εκτέλεσή τους, ώστε να διαρρεύσουν πληροφορίες σε άλλα προγράμματα.

Σε αυτό το σημείο πρέπει να αναφερθεί ότι λόγω της φύσης του συγκεκριμένου ελλατώματος (bug), και τις πληθώρας διαφορετικών τρόπων το οποίο αυτό μπορεί να εφαρμοσθεί, ανακαλύπτονται συχνά καινούριες εκδοχές του. Ώστοσο, θα επικεντρωθώ στις 2 βασικές εκδοχές που ανακοινώθηκαν τον Ιανουάριο του 2018 οι οποίες είναι η Παράκαμψη Ελέγχου Ορίων (Bounds Check Bypass - BCB), που αφορά την πρώτη εκδοχή (variant 1) και η Παραπλάνηση Στόχου Διακλάδωσης (Branch Target Injection - BTI), που αφορά τη δεύτερη (variant 2).

2.3 Παράκαμψη Ελέγχου Ορίων (Bounds Check Bypass)

Η συγκεκριμένη εκδοχή έχει την ικανότητα να διαβάζει τυχαία κομμάτια μνήμης του συστήματος εκμεταλλευόμενη την πρόβλεψη διακλάδωσης. Συγκεκριμένα, για την επίτευξη της επίθεσης, το λειτουργικό θα πρέπει να εκτελέσει μία ακολουθία εντολών που αποτελείται από έναν έλεγχο ορίων πίνακα, ακολουθούμενο από μία ανάγνωση δεδομένων πίνακα, όπου ο δείκτης αυτής της ανάγνωσης είναι ορισμένος από το κακόβουλο πρόγραμμα.

Για την καλύτερη κατανόηση του προβλήματος και πώς αυτό δημιουργείται, ας εστιάσουμε στα παρακάτω βήματα από ένα κομμάτι κώδικα, όπως εμφανίζεται και στο Spectre paper του Paul Kocher και της Google ProjectZero:

- Δημιουργία 2 Πινάκων (Arrays) με συγκεκριμένο μέγεθος.

- Έλεγχος ορίου του πρώτου πίνακα μέσω μεταβλητής ορισμένη από τον χρήστη.

- Αν η μεταβλητή είναι εντός ορίων πίνακα, διάβασε τα δεδομένα της θέσης που ορίστηκε από την μεταβλητή από τον πρώτο πίνακα.

- Υπολόγισε μία άλλη μεταβλητή που θα προκύψει από το δεύτερο πίνακα σε σχέση με το στοιχείο που προέκυψε από την πρώτη μεταβλητή του πρώτου πίνακα, τροποποιημένη κατά μία τυχαία τιμή.

- Διάβασε στοιχεία από τον δεύτερο πίνακα της θέσης που ορίζει η δεύτερη μεταβλητή.

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

Εικόνα 6: Ο κώδικας του παραδείγματος

Όπως είναι προφανές η πρώτη μεταβλητή, μπορεί να έχει οποιαδήποτε τιμή σε κάθε επανάληψη, αφού αυτή ορίζεται από τον χρήστη. Στην περίπτωση της εκτέλεσης κώδικα χωρίς πρόβλεψη διακλάδωσης ή υποθετικής εκτέλεσης, η εκτέλεση του κώδικα θα σταματούσε στο δεύτερο βήμα, εάν η μεταβλητή (δείκτης) που ορίστηκε βρισκόταν εκτός ορίων πίνακα (out-of-bounds index). Σε αντίθετη περίπτωση όμως και μετά από ορισμένες σωστές επαναλήψεις και "εκπαιδεύοντας" με αυτό τον τρόπο την πρόβλεψη διακλάδωσης να αποδέχεται τον πρώτο έλεγχο ορίων με βάση τον Πίνακα Ιστορικού Διακλαδώσεων, ο επεξεργαστής χρησιμοποιεί μία τιμή που προέκυψε από δεδομένα που βρίσκονταν εκτός ορίων προσπέλασης δεδομένων (out-of-bounds read), και αποθηκεύει τα δεδομένα στην κρυφή μνήμη. Πιο αναλυτικά, κατά την εκτέλεση του κώδικα, υπάρχουν αστοχίες μνήμης (cache miss), δηλαδή τα δεδομένα που ζητούνται δεν βρίσκονται στην κρυφή μνήμη, κατά την διάρκεια ελέγχου της διακλάδωσης. Αυτό έχει σαν αποτέλεσμα ο επεξεργαστής, κάνοντας χρήση της Πρόβλεψης Διακλάδωσης, να προβλέψει λανθασμένα ότι η διακλάδωση θα είναι αληθής και να χρησιμοποιεί την μεταβλητή που ορίστηκε ως μία θέση του πίνακα, ζητώντας τα δεδομένα που της αντιστοιχούν.

Στην συνέχεια χρησιμοποιείται το αποτέλεσμα αυτής της πράξης για να υπολογισθεί μία άλλη διεύθυνση θέσης για τον δεύτερο πίνακα. Τα συγκεκριμένα δεδομένα λοιπόν που θα προκύψουν είναι τυχαία, αφού εξ' αρχής η μεταβλητή έθεσε μία θέση στον πίνακα που δεν είναι αποδεκτή (εκτός ορίων). Ως αποτέλεσμα αυτό έχει την αποθήκευση αυτών των τυχαίων δεδομένων στην μνήμη. Κατά την διάρκεια της αποθήκευσης, ο επεξεργαστής αντιλαμβάνεται το λάθος που έγινε στην πρόβλεψη διακλάδωσης και επιστρέφει το πρόγραμμα στο σημείο επίλυσης της διακλάδωσης. Ωστόσο, το αποτέλεσμα της υποθετικής εκτέλεσης του δεύτερου πίνακα που αφορά μία τυχαία διεύθυνση έχει ήδη καταγραφεί στην κρυφή μνήμη και μαζί με αυτό, και τα τυχαία δεδομένα που μπορεί να περιέχει. Η πρόσβαση πλέον σε αυτά τα δεδομένα που βρίσκονται στην κρυφή μνήμη μπορεί να πραγματοποιηθεί με την χρήση Επίθεσης Πλαϊνού Καναλιού (side-channel attack), όπως Άδειασμα+Επαναφόρτωση (Flush+Reload) ή Αρχικοποίηση+Διερεύνηση (Prime+Probe).

Χρησιμοποιώντας έτσι σε επανάληψη (loop) την ίδια τεχνική, μπορούμε να έχουμε πρόσβαση σε τυχαία δεδομένα (συμπεριλαμβανομένων και προστατευμένων δεδομένων), τα οποία υπό κανονικές συνθήκες δεν θα μπορούσαμε να προσπελάσουμε. Αξίζει να σημειωθεί ότι η επιτυχία ανάκαμψης δεδομένων με αυτή την τεχνική είναι αρκετά αποτελεσματική.

Παρ'όλα αυτά μπορεί να υπάρξουν σφάλματα στην ανάγνωση των δεδομένων. Αυτά τα σφάλματα μπορεί να προκύψουν από τριτογενείς παράγοντες, όπως για παράδειγμα η αποθήκευση όλων των δεδομένων του δεύτερου πίνακα εξ'αρχής στην κρυφή μνήμη ως αποτέλεσμα της προφόρτωσης υλικού (hardware prefetching) ή άλλων ενεργειών του λειτουργικού συστήματος, ή και ενεργειών άλλων προγραμμάτων, στην περίπτωση που ο δεύτερος πίνακας αντιστοιχεί σε στοιχεία μνήμης που αφορούν διαμοιραζόμενες βιβλιοθήκες (shared libraries).

Σε αυτές τις περιπτώσεις θα πρέπει να γίνουν επαναλαμβανόμενες "επιθέσεις" για να εξακριβωθεί η ορθότητα των δεδομένων που έχουν αποθεκευτεί στην κρυφή μνήμη με την χρήση εντολών που βασίζονται στην χρονική απόκριση του επεξεργαστή στην προσπέλαση δεδομένων, δηλαδή αν ο επεξεργαστής εκτελέσει γρήγορα αυτή την προσπέλαση, τότε τα δεδομένα βρίσκονται πράγματι στην κρυφή μνήμη, σε κάθε άλλη περίπτωση θα πρέπει να ξεκινήσει η διαδικασία ανάγνωσης από την αρχή. Τέτοιες εντολές για την εξακρίβωση απόκρισης χρόνου είναι αυτές που βασίζονται στην RDTSCP (Read Time-Stamp Counter and Processor ID) και σε συνδιασμό με επαναλαμβανόμενες εκτελέσεις μπορούν να αποφέρουν αποφυγή σφαλμάτων εξακρίβωσης αποτελεσμάτων σε ποσοστό μεγαλύτερου του 99,9%.

2.4 Παραπλάνηση Στόχου Διακλάδωσης (Branch Target Injection)

Σε αυτή την εκδοχή, η οποία ονομάζεται και Δηλητηρίαση Έμμεσων Διακλαδώσεων (Poisoning Indirect Branches), το πρόγραμμα επίθεσης (adversary program) επιδιώκει να "εκπαιδεύσει" τον επεξεργαστή να κάνει λάθος επιλογή κατά την επίλυση έμμεσων διακλαδώσεων, με στόχο την προσπέλαση τυχαίων στοιχείων μνήμης που ανήκουν σε άλλα προγράμματα. Οι έμμεσες διακλαδώσεις είναι αρκετά συχνές και χρησιμοποιούνται ευρέως σε κάθε είδους προγράμματα. Στην περίπτωση που ο υπολογισμός για το αποτέλεσμα μίας διεύθυνσης σε μία έμμεση διακλάδωση καθυστερήσει, λόγω αστοχίας της κρυφής μνήμης, η υποθετική εκτέλεση θα προχωρίσει στην εκτέλεση κώδικα που έχει προβλέψει με βάση προηγούμενων συμπεριφορών εκτέλεσης κώδικα. Το αποτέλεσμα αυτής της εκπαίδευσης οδηγεί στην εκτέλεση κώδικα σε άλλες, μη έγκυρες υπό κανονικές συνθήκες, περιοχές μνήμης, οι οποίες ορίζονται κατάλληλα από το πρόγραμμα επίθεσης. Αυτό το είδος της επίθεσης θεωρείται και ως το πιο επικίνδυνο, αφού το πρόγραμμα επίθεσης μπορεί με αυτό τον τρόπο να θέσει εκτεθειμένα δεδομένα μνήμης από το πρόγραμμα θύμα (victim program), ακόμα και στην περίπτωση μη ύπαρξης εκμεταλλεύσιμης εξαρτημένης συνθήκης διακλάδωσης (exploitable conditional branch).

Ένα απλό παράδειγμα που επιδεικνύει την λειτουργία αυτής της επίθεσης είναι το παρακάτω. Ας υποθέσουμε ότι το πρόγραμμα επίθεσης, το οποίο έχει πρόσβαση σε 2 καταχωρητές κατά την διάρκεια εκτέλεσης μίας έμμεσης διακλάδωσης, επιδιώκει να διαβάσει δεδομένα από ένα άλλο πρόγραμμα. Στην συνέχεια το πρόγραμμα επίθεσης χρειάζεται να εντοπίσει και να έχει πρόσβαση σε ένα "μέσο" (Spectre gadget) του οποίου η υποθετική εκτέλεσή του από τον επεξεργαστή θα του επιτρέψει να μεταφέρει δεδομένα σε ένα "κρυφό" κανάλι (covert channel), του οποίου έχει πρόσβαση το πρόγραμμα επίθεσης. Ένα τέτοιο μέσο θα μπορούσε να αποτελείται από 2 εντολές, η οποία η πρώτη θα εκτελούσε μία πράξη πρόσθεσης πάνω σε μία διεύθυνση μνήμης της

οποίας ο καταχωρητής θα ελέγχεται από το πρόγραμμα επίθεσης και η οποία αποθηκεύεται σε έναν άλλο δεύτερο καταχωρητή που ελέγχεται από το ίδιο πρόγραμμα, ακολουθούμενο από μία εντολή που επιτρέπει το διάβασμα δεδομένων μνήμης της διεύθυνσης του δεύτερου καταχωρητή. Σε αυτή την περίπτωση, το μέσο της επίθεσης επιτρέπει στο πρόγραμμα, ελέγχοντας τον πρώτο καταχωρητή, να επιλέξει την διεύθυνση στην οποία θα πραγματοποιηθεί η διαρροή πληροφορίας και ελέγχοντας τον δεύτερο καταχωρητή, τον τρόπο με τον οποίο η συγκεκριμένη πληροφορία σχετίζεται με την διεύθυνση που διαβάζεται μέσω της δεύτερης εντολής.

Στους επεξεργαστές που ελέχθηκαν για την εξακρίβωση της λειτουργίας της συγκεκριμένης εκδοχής του Spectre, το μέσο που επιλέχθηκε για την πραγματοποίηση της επίθεσης θα έπρεπε να είναι εξ' αρχής στοιχείο της μνήμης που θα εκτελεσθεί από το πρόγραμμα θύμα, ώστε να ενεργοποιηθεί η υποθετική εκτέλεση από τον επεξεργαστή. Ωστόσο, τα πράγματα γίνονται πολύ πιο εύκολα αν αναλογιστούμε το μέγεθος των διαμοιραζόμενων βιβλιοθηκών που χρησιμοποιούνται από κοινού στις περισσότερες διεργασίες. Επομένως, ένα κακόβουλο πρόγραμμα έχει αρκετές επιλογές στην εύρεση ενός μέσου, χωρίς να περιορίζεται εξ'ολοκλήρου και αποκλειστικά στο σύνολο του κώδικα του θύματος, για την εκτέλεση της επίθεσης.

'Οπως είναι φανερό από την παραπάνω ανάλυση, αρχικός στόχος του κακόβουλου προγράμματος είναι να εκπαιδεύσει την πρόβλεψη διακλάδωσης μέσω του δικού του κώδικα, ώστε το πρόγραμμα θύμα να εκτελέσει το μέσο της επίθεσης κατά την διάρκεια εκτέλεσης του κώδικά του. Στην ουσία το κακόβουλο πρόγραμμα μιμείται το μοτίβο των διακλαδώσεων του θύματος έως ότου φτάσει στο σημείο της διακλάδωσης που πρέπει να προβλεφθεί λάθος για την πραγματοποίηση της επίθεσης. Στο σημείο αυτό αξίζει να σημειωθεί ότι οι προϋποθέσεις για την εκπαίδευση της πρόβλεψης διακλάδωσης διαφέρουν από επεξεργαστή σε επεξεργαστή.

Επιπλέον, αυτή η διαδικασία της μίμισης του μοτίβου διακλαδώσεων πραγματοποιείται στο επίπεδο των εικονικών διευθύνσεων. Οι φυσικές διευθύνσεις, η χρονική στιγμή και το αναγνωριστικό διεργασίας (process ID), δεν φαίνεται να επιρεάζουν την έκβαση του αποτελέσματος. Ωστόσο αυτή η εκπαίδευση της πρόβλεψης διακλάδωσης θα πρέπει να γίνει στον ίδιο πυρήνα του επεξεργαστή, διότι κάθε ένας πυρήνας χρησιμοποιεί διαφορετική πρόβλεψη διακλάδωσης και για το λόγο αυτό δεν μπορούν να επηρεάστουν τα αποτελέσματα της μίας από κάποια άλλη.

Επίσης παρατηρήθηκε ότι οι προβλέπτες διακλαδώσεων μπορούν να επηρεαστούν από μεταβιβάσεις για διάβασμα δεδομένων σε σημεία του συστήματος που δεν είναι επιτρεπτά. Σε αυτές τις περιπτώσεις θα υπάρξει σφάλμα εξαίρεσης (exception) κατά την διάρκεια εκτέλεσης του κώδικα, το οποίο μπορεί να γίνει διαχειρίσιμο από κάποιον χειριστή σημάτων (signal handler). Με αυτό τον τρόπο η πρόβλεψη διακλάδωσης θα εκπαιδεύεται να στέλνει όλες τις άλλες διεργασίες στην ίδια θέση εικονικής διεύθυνσης (virtual address space) του θύματος, στην οποία θα προκαλείται η εκτέλεση του μέσου της επίθεσης.

Τέλος, σύμφωνα με τις μελέτες που έχουν πραγματοποιηθεί σε συγκεκριμένους σύγχρονους επεξεργαστές, η επιτυχία δηλητηρίασης των διακλαδώσεων φτάνει και το ποσοστό που μπορεί να ξεπεράσει και το 98%.

2.5 Επιπτώσεις του Spectre

Όπως φαίνεται και από τις 2 εκδοχές του Spectre, το ζήτημα ασφαλείας που προέκυψε είναι το μεγαλύτερο και το δυσκολότερο, ως προς τον τρόπο αντιμετώπισής του, που έχει εμφανιστεί μέχρι στιγμής. Αν και όλες αυτές οι περιπτώσεις που αναλύθηκαν ήταν

αποτέλεσμα στοχευμένων πειραμάτων, αυτό δεν σημαίνει ότι οι συνέπειες αυτού του προβλήματος δεν θα είναι καταστρεπτικές στην περίπτωση που επεκταθεί και εφαρμοσθεί εκτός ελεγχόμενων ορίων.

Ονομαστικά, κάποιος που θα μπορούσε να εκμεταλλευτεί στο έπακρο τις δυνατότητες του Spectre, θα είχε την ευχέρεια να αποκτήσει πρόσβαση σε κρυφά κλειδιά που αφορούν διαδικασίες κρυπτογράφησης, απόκτηση κωδικών για πρόσβαση σε κλειδωμένες (προστατευμένες) λειτουργίες ενός υπολογιστικού συστήματος ή ακόμα και πρόσβαση σε στοιχεία που αφορούν συναρτήσεις κατακερματισμού (hash functions), οι οποίες μπορουν να έχουν συσχέτιση με συναρτήσεις αθροίσματος ελέγχου (Cyclic Redundancy Check - CRC), υπολογισμό ψηφίου ελέγχου (Check Digit), δακτυλικά ψηφιακά αποτυπώματα (Digital Fingerprints), κώδικες ελέγχου λαθών (Error Correcting Codes) κ.α.

Αρχικά, η λειτουργία του Spectre ήταν περιορισμένη και επηρέαζε διεργασίες μόνο εντός του συστήματος του οποίου βρισκόταν. Αργότερα όμως ανακαλύφθηκαν τρόποι για την εκτέλεσή του εξ'αποστάσεως. Με αυτό τον τρόπο οι επιθέσεις Spectre έχουν την δυνατότητα να εισάγονται και να εκτελούνται μέσω κώδικα JavaScript, εντός των φυλλομετρητών (browsers), με στόχο την απόκτηση προσωπικών δεδομένων, κωδικών σε ιστοσελίδες κτλ. Επιπλέον τέτοιες επιθέσεις θα μπορούσαν να γίνουν και σε υπολογιστικές υπηρεσίες Νέφους (Cloud Computing Services), με στόχο την παρακολούθηση, την πρόσβαση σε στοιχεία βάσεων δεδομένων, έλεγχο εξυπηρετητών (servers) κ.α.

Τέλος, το Spectre μπορεί να εφαρμοσθεί και εντός εικονικών μηχανών (virtual machines). Αν και σε αυτή την περίπτωση η διαδικασία είναι πιο περίπλοκη και η απόκτηση πληροφορίας γίνεται με πιο αργό ρυθμό, η επιτυχία της εξαρτάται από την απόκτηση πρόσβασης του κακόβουλου προγράμματος στο δακτύλιο προστασίας 0 (protection ring 0). Από αυτό το σημείο και μετά, το Spectre μπορεί να λάβει πληροφορίες για το σύστημα πάνω στο οποίο λειτουργεί η συγκεκριμένη εικονική μηχανή, πληροφορίες για την Διάταξη Τυχαιοποίησης του Χώρου Διεθύνσεων (Address Space Layout Randomizer - ASLR) αναλύοντας τον Πίνακα Ιστορικού Διακλαδώσεων και να έχει πρόσβαση στο επίπεδο 3 της κρυφής μνήμης, όπως και στην διάταξη της φυσικής μνήμης, χρησιμοποιώντας την Παραπλάνηση Στόχου Διακλάδωσης, με την βοήθεια του "μέσου" επίθεσης.

3. ΤΡΟΠΟΙ ΑΝΤΙΜΕΤΩΠΙΣΗΣ

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα αναλυθούν οι τρόποι με τους οποίους μπορούμε να περιορίσουμε την αποτελεσματικότητα των επιθέσεων του Spectre. Λόγω της φύσης του ζητήματος και τον τρόπο που αυτό επηρεάζει τις λειτουργίες του συστήματος, έχουν προταθεί διαφορετικά μέτρα που το κάθε ένα εστιάζει ειδικά στους διαφορετικούς μηχανισμούς που εκμεταλλεύεται το Spectre. Τέλος, θα αναφερθούν και τα μειονεκτήματα ή οι πιθανοί περιορισμοί που μπορούν να προκύψουν και αφορούν την αδυναμία της τωρινή τεχνολογία υλικού και αν αυτή μπορεί να αλλάξει στο μέλλον.

3.1.1 Παρεμπόδιση της Υποθετικής Εκτέλεσης

Βασικό στοιχείο της επίθεσης του Spectre είναι η υποθετική εκτέλεση. Ως αποτέλεσμα, η παρεμπόδιση μίας τέτοιας λειτουργίας, θα ήταν αρκετή για να καταστήσει το Spectre αδύνατο για να αποσπάσει δεδομένα στα οποία δεν θα μπορούσε να είχε πρόσβαση υπό κανονικές συνθήκες. Σε μία τέτοια περίπτωση, οι εντολές θα εκτελούνται μόνο ύστερα από την εξακρίβωση ελέγχου της ροής του προγράμματος και πάντα με την σειρά με την οποία έχουν ρητά οριστεί εντός του κώδικα. Ωστόσο, όσο αποτελεσματική είναι αυτή η λύση, τόσο πιο μεγάλο αντίκτυπο έχει στην υποβάθμιση της επίδοσης των επεξεργαστών.

Μία άλλη εκδοχή αυτής της λύσης είναι η επιλεκτική απενεργοποίηση της υποθετικής εκτέλεσης σε κρίσιμα σημεία του κώδικα που έχουν να κάνουν με προσπέλαση προστατευμένων δεδομένων ή την δυνατότητα ορισμένα υπολογιστικά συστήματα να εναλλάσσουν την χρήση των επεξεργαστών τους ανάλογα με το αν είναι ασφαλής η χρήση της υποθετικής εκτέλεση ή όχι. Παρ'όλα αυτά η υλοποιήση μίας τέτοιας τεχνικής δεν είναι εύκολα υλοποιήσιμη για την άμεση αντιμετώπιση του προβλήματος.

Μία άλλη προσέγγιση θα ήταν από πλευράς λογισμικού. Η εφαρμογή μεθόδων όπως σειριοποίηση (serializing) ή προσωρινή παρεμπόδιση υποθετικής εκτέλεσης (speculation blocking), θα διασφάλιζαν ότι οι εντολές δεν θα εκτελούνται έκτος σειράς ή υποθετικά αντίστοιχα. Ένας τρόπος για την εφαρμογή αυτής της λύσης που έχουν προτείνει από κοινού η Intel και η AMD είναι η χρήση της εντολής LFENCE σε κάθε έκβαση των διακλαδώσεων. Κάτι τέτοιο όμως θα καθυστερούσε υπερβολικά τις διακλαδώσεις καθώς θα έθετε εμμέσως εκτός λειτουργίας την πρόβλεψη διακλάδωσης.

Αντιθέτως, μία βελτιωμένη εκδοχή αυτής της λύσης θα ήταν η χρήση στατικής ανάλυσης για τον καλύτερο εντοπισμό των διακλαδώσεων των οποίων η πρόβλεψη θα πρέπει να εμποδιστεί, αφού αρκετές διακλαδώσεις εξ' ορισμού δεν έχουν την δυνατότητα να διαβάσουν και να διαρρεύσουν στοιχεία μνήμης που βρίσκονται εκτός των ορίων τους (out-of-bounds).

Επιπλέον η εισαγωγή εντολών σειριοποίησης θα βοηθούσε στην αντιμετώπιση της δηλητηρίασης διακλαδώσεων, αφού η σειρά της εκτέλεσης των εντολών θα είναι συγκεκριμένη και δεν θα αλλάζει ανάλογα με προβλέψεις ή εκτελέσεις εκτός σειράς.

Τέλος, η εισαγωγή των εντολών LFENCE πριν από έμμεσες διακλαδώσεις θα εξασφάλιζε ότι η ροή των εντολών της διακλάδωσης κατά την διάρκεια της σωλήνωσης θα έχει ολοκληρωθεί με την σωστή σειρά και εγκαίρως, χωρίς να χρειαστεί η εφαρμογή των μηχανισμών της πρόβλεψης διακλάδωσης. Αυτό με την σειρά του θα οδηγήσει στην

μείωση των εντολών που εκτελούνται υποθετικά στην περίπτωση που υπάρχει απόπειρα δηλητηρίασης διακλάδωσης. Η εφαρμογή όμως της συγκεκριμένης προσέγγισης, υποχρεώνει να υποβληθούν σε αλλαγή όλα τα υπάρχοντα προγράμματα, εκτελέσιμα αρχεία και βιβλιοθήκες λογισμικού. Επιπροσθέτως, αυτή η αλλαγή θα είναι ακόμα πιο δύσκολη να εφαρμοσθεί σε πεπαλαιωμένα λογισμικά (legacy software).

3.1.2 Παρεμπόδιση Πρόσβασης σε Κρυφά Δεδομένα

Υπάρχουν αρκετοί τρόποι για να επιτεφχθεί αυτή η απαγόρευση πρόσβασης σε κρυφά δεδομένα. Πιο συγκεκριμένα, ο φυλλομετρητής Chrome της Google, έχει την ικανότητα να εκτελεί κάθε ιστοσελίδα που "ανοίγεται" σαν διαφορετική διεργασία. Με το τρόπο αυτό το Spectre δεν θα μπορεί να διαβάσει δεδομένα χρησιμοποιώντας JavaScript, αφού οι πληροφορίες των διαφορετικών σελίδων θα είναι ορισμένες σε θέσεις μνήμης που ανήκουν σε διαφορετικές διεργασίες.

Το WebKit (λογισμικό φυλλομετρητών) χρησιμοποιεί 2 άλλες τεχνικές για τον περιορισμό της πρόσβασης σε κρυφά δεδομένα από την υποθετική εκτέλεση κώδικα. Η πρώτη αφορά τα όρια των πινάκων και η δεύτερη τον τρόπο πρόσβασης σε δείκτες. Η πρώτη τεχνική υλοποιείται με την αντικατάσταση των ορίων των πινάκων με ένα κρυφό δείκτη μάσκας. Στην περίπτωση αυτή, αντί να ελέγχεται η μεταβλητή για την θέση ενός πίνακα για το αν είναι εντός ορίων ή όχι, ελέγχεται ο συγκεκριμένος δείκτης ο οποίος είναι κατάλληλα επιλεγμένος ώστε να αντιπροσωπεύει το μέγεθος του πίνακα. Έτσι, στην περίπτωση που η ζητούμενη θέση βρίσκεται εκτός ορίων πίνακα, το λογισμικό αποτρέπει την πρόσβαση του προγράμματος σε στοιχεία που δεν του επιτρέπονται.

Στην δεύτερη τεχνική, οι δείκτες του προγράμματος τυχαιοποιούνται με την χρήση μίας τυχαίας τιμής. Με αυτό τον τρόπο το κακόβουλο πρόγραμμα δεν θα μπορεί να γνωρίζει κάθε στιγμή ποιος δείκτης έχει πρόσβαση σε ποια δεδομένα, χωρίς να γνωρίζει την τυχαία επιλεγμένη τιμή.

Αυτές οι τεχνικές είναι ιδιαίτερα χρήσιμες για μεταγλωτιστές και διερμηνείς (interpreters) κώδικα, που θέλουν να έχουν τον έλεγχο ως προς τον κώδικα ο οποιός εκτελείται και να έχουν την ευχέρεια ως προς τον περιορισμό των δεδομένων που τα προγράμματα μπορούν να προσπελάσουν.

3.1.3 Παρεμπόδιση Εισαγωγής Δεδομένων σε Κρυφά Κανάλια

Αναφέρεται στην ιδιότητα των επεξεργαστών να γνωρίζουν εάν τα δεδομένα που μεταφέρθηκαν στις κρυφές μνήμες είναι αποτέλεσμα υποθετικής εκτέλεσης, και στην περίπτωση αυτή να μπλοκάρεται η προσπέλασή τους έως ότου γίνει ο έλεγχος δικαιωμάτων από το σύστημα. Δυστυχώς, αυτή η δυνατότητα δεν υποστηρίζεται από τους τωρινούς επεξεργαστές και την σημερινή τεχνολογία.

3.1.4 Περιορισμός Εξαγωγής Αποτελεσμάτων από Κρυφά Κανάλια

Ένας από τους τρόπους που χρησιμοποιείται για την αντιμετώπιση αυτού του προβλήματος στους φυλλομετρητές, είναι η εισαγωγή ενός τύπου καθυστέρησης για την μείωση της ταχύτητας με την οποία εκτελείται ο κώδικας JavaScript. Επιπλέον αυτού του είδους οι επιδιορθώσεις λογισμικού, απενεργοποιούν την οντότητα SharedArrayBuffers, που μπορεί να χρησιμοποιηθεί ως βάση για την καταμέτρηση

χρονικής διάρκειας από κακόβουλα προγράμματα. Ωστόσο, αυτές οι βελτιστοποιήσεις αν και επιτυγχάνουν με τον τρόπο τους την μείωση της αποτελεσματικότητας των επιθέσεων, δεν τον εξαλείφουν πλήρως. Αυτή η λύση που χρησιμοποιείται μπορεί να θεωρηθεί προσωρινή, αφού δεν υπάρχει ακόμα δυνατότητα από τους τωρινούς επεξεργαστές να απενεργοποιήσουν εντελώς την λειτουργία των κρυφών καναλιών.

3.1.5 Αποτροπή Δηλητηρίασης Διακλαδώσεων

Για την εφαρμογή μίας τέτοιας τεχνικής, η Intel και η AMD έχουν επεκτείνει από κοινού το σύνολο εντολών αρχιτεκτονικής (Instruction Set Architecture - ISA) των επεξεργαστών για να τον εμπλουτίσουν με διαφορετικούς μηχανισμούς διαχείρισης έμμεσων διακλαδώσεων.

Η πρώτη είναι η Περιορισμένη Εκτέλεση Έμμεσων Διακλαδώσεων (Indirect Branch Restricted Speculation - IBRS), στην οποία ο επεξεργαστής, κατηγοριοποιεί τις διακλαδώσεις με βάση την σημαντικότητά τους και αποτρέπει την επιρροή της πρόβλεψης διακλάδωσης μεταξύ διαφορετικών κατηγοριών.

Στην συνέχεια υπάρχει η Έμμεση Πρόβλεψη Διακλάδωσης Ενιαίου Νήματος (Single Thread Indirect Branch Prediction - STIBP), η οποία αναιρεί την ιδιότητα της πρόβλεψης διακλαδώσεων μεταξύ προγραμμάτων που εκτελούνται σε κύρια νήματα του ίδιου πυρήνα επεξεργαστή. Επιπλέον γίνεται και η χρήση του Φραγμού Πρόβλεψης Έμμεσων Διακλαδώσεων (Indirect Branch Predictor Barrier - IBPB), ο οποίος εμποδίζει τα προγράμματα που άρχισαν την εκτέλεσή τους πριν την δημιουργία του φραγμού, να επηρεάζουν προγράμματα που άρχισαν να εκτελούνται μετά την δημιουργία του. Με αυτό τον τρόπο το σύστημα μπορεί να προστατεύσει κρίσιμες για την ασφάλεια διακλαδώσεις, ώστε να μην ενεργούν με βάση το προηγούμενο ιστορικό διακλαδώσεων που μπορεί να έχει επηρεαστεί από κακόβουλα προγράμματα.

Ωστόσο αυτές οι αλλαγές του υλικού έχουν επιπτώσεις στην επίδοση της ταχύτητας των επεξεργαστών που εξαρτώνται από το πλήθος των μηχανισμών που επιλέγονται να ενεργοποιηθούν, τον τρόπο εφαρμογής τους (πλήρης έλεγχος όλων των πυρήνων, εφαρμογή σε κάθε διεργασία κτλ.) αλλά και των δυνατοτήτων του κάθε επεξεργαστή ξεχωριστά.

Αντίστοιχα η Google έχει προτείνει έναν άλλο μηχανισμό για την αντιμετώπιση του προβλήματος που τον ονομάζει retpolines (return + trampoline). Στην ουσία πρόκειται για ένα κομμάτι κώδικα που αντικαθιστά τις έμμεσες διακλαδώσεις με εντολές return. Κατά την διάρκεια εκτέλεσης έμμεσων διακλαδώσεων το σύστημα μπαίνει σε έναν ατέρμων βρόγχο (endless loop) μέχρι τον υπολογισμό της διακλάδωσης και την επιστροφή στην κανονική ροή των εντολών μέσω της εντολής return. Αν και αυτή η τεχνική μπορεί εν' μέρη να αποτελέσει λύση για το πρόβλημα της δηλητηρίασης διακλαδώσεων, δεν αποτελεί μόνιμη λύση. Στην περίπτωση που η Στοίβα Αποθήκευσης Εντολών Επιστροφής (Return Stack Buffer) γεμίσει, τότε το σύστημα κάνει χρήση της Μνήμης Στόχου Διακλάδωσης (Branch Target Buffer), η οποία είναι εξίσου ευάλωτη σε επιθέσεις λογισμικού.

4. ΣΥΜΠΕΡΑΣΜΑΤΑ

Όπως φαίνεται και από την παραπάνω ανάλυση, το Spectre δεν είναι ένα εύκολα αντιμετωπίσιμο πρόβλημα. Οι βασικές εκδοχές του έχουν την δυνατότητα να επηρεάζουν και να εκμεταλλεύονται πολλαπλούς μηχανισμούς των υπολογιστικών συστημάτων και ακόμα και σήμερα το συγκεκριμένο ζήτημα ασφαλείας ολοένα και εξελίσσεται, αφού οι εξακριβωμένες εκδοχές του Spectre έχουν φτάσει πάνω από 8, με την κάθε μία να στοχεύει σε ένα διαφορετικό εκμεταλλεύσιμο μηχανισμό του συστήματος.

Αναφορικά, οι επεξεργαστές που είναι ευάλωτοι σε τέτοιες επιθέσεις ξεπερνούν συνολικά τους 2.800 και αφορούν την Intel, AMD, ARM, IBM, APPLE και VIA, όχι μόνο προσωπικών συστημάτων αλλά και εξυπηρετητών. Αν και μέχρι στιγμής δεν έχει γίνει γνωστή κάποια επίθεση μέσω Spectre εκτός εργαστηριακών πλαισίων, αυτό οφείλεται εν' μέρη στην πολυπλοκότητα της μεθόδου της επίθεσης καθώς και στο γεγονός ότι η επίθεση πρέπει να πραγματοποιηθεί εντός του συστήματος που θέλουμε να προσβάλουμε ή με την χρήση JavaScript μέσω προγραμμάτων φυλλομετρητών και όχι με κάποιον άλλο ευκολότερο τρόπο εξ' αποστάσεως.

Επικεντρώνοντας στην ουσία των εκμεταλλεύσιμων μηχανισμών των συστημάτων, μπορούμε να εντοπίζουμε ένα μοναδικό κοινό που έχουν όλοι μεταξύ τους και αυτό είναι η αύξηση της αποδοτικότητας των επεξεργαστών μέσω της ταχύτητας εκτέλεσης εντολών. Αυτή η ιδέα είναι εκείνη που έχει επικρατήσει από την αρχή της δημιουργίας και της αργότερα εξέλιξης των επεξεργαστών και προγραμμάτων λογισμικού. Φαίνεται όμως πως μία τέτοια εξέλιξη της ταχύτητας έρχεται με το κόστος της ασφάλειας. Όπως αναφέρθηκε και στο κεφάλαιο της αντιμετώπισης του Spectre, όλες οι ενδεικτικές λύσεις που έχουν προταθεί, στοχεύουν στην προσωρινή αντιμετώπιση του προβλήματος, αφού η πλειοψηφία τους απενεργοποιεί κάποιον μηχανισμό του επεξεργαστή ή του λογισμικού που χρησιμοποιείται για ταχύτερη εκτέλεση και αύξηση της αποδοτικότητας των επεξεργαστών.

Δυστυχώς μερικές από αυτές τις λύσεις δεν είναι υλοποιήσιμες με την χρήση της τωρινής τεχνολογίας. Για την καλύτερη αντιμετώπισή του θα πρέπει να αλλάξει εξ' ολοκλήρου ο τρόπος λειτουργίας των επεξεργαστών και το σύνολο εντολών της αρχιτεκτονικής τους.

Συγκεκριμένα, η Intel ανέφερε τον Μάρτιο του 2018, ότι θα επανασχεδιάσει τους επεξεργαστές της για την αντιμετώπιση του Spectre και ότι τα καινούρια τσιπ θα είναι έτοιμα με την άφιξη της επόμενης γενιάς των επεξεργαστών της μέχρι το τέλος του χρόνου. Αντίστοιχα βήματα πρόκειται να ακολουθήσουν και οι υπόλοιπες εταιρίες, είτε με επανασχεδιασμό ή με προσθήκη επιδιορθωμένου κώδικα στους ήδη υπάρχοντες επεξεργαστές.

Κλείνοντας να τονιστεί ότι το Spectre μέσω των δεδομένων των οποίων εξάγει, έχει μεγαλύτερες επιπτώσεις σε συστήματα στα οποία διαμοιράζουν τους πόρους τους σε διαφορετικούς χρήστες κάθε στιγμή, όπως κοινόχροιστα συστήματα με λειτουργία εξυπηρέτησης πολλαπλών επισκεπτών (guests), εξυπηρετητές και συστήματα Νέφους, και αυτό διότι τα προσωπικά δεδομένα που εν δυνάμει μπορούν να εξαχθούν είναι μεγαλύτερης σημασίας από κάθε άλλο σύστημα, αφού εστιάζουν σε πληροφορίες που αφορούν βάσεις δεδομένων, κωδικούς ασφαλείας προσωπικών συστημάτων και λογαριασμών που δεν αφορούν μόνο έναν χρήστη άλλα όλο το σύνολο των χρηστών που χρησιμοποιούν το συγκεκριμένο σύστημα.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Central Processing Unit	Κεντρική Μονάδα Επεξεργασίας
Microcontroller	Μικροελεγκτής
System on a Chip	Σύστημα σε τσιπ
Multi-Core Processor	Πολυ-πύρηνος επεξεργαστής
Sockets	Υποδοχείς
Registers	Καταχωρητές
Cache Memory	Κρυφή μνήμη
Random Access Memory	Κύρια Μνήμη ή Μνήμη Τυχαίας Προσπέλασης
Volatile/Non-Volatile	Πτητικός/Μη-πτητικός
Hard Drives	Σκληρός Δίσκος
Pipeline	Σωλήνωση
Out-of-Order Execution	Εκτέλεση εκτός σειράς
Dependency Hazards	Κίνδυνοι εξάρτησης
Branch Prediction	Πρόβλεψη διακλάδωσης
Branch taken/not taken	Αποδοχή/Απόρριψη διακλάδωσης
Branch History Table	Πίνακας Ιστορικού Διακλάδωσης
Speculative Execution	Υποθετική Εκτέλεση
Rollback	Επιστροφή σε αρχική κατάσταση
Concurrency	Ταυτοχρονισμός
Caching	Φύλαξη
Access Violation	Παραβίαση πρόσβασης
Out-of-Bounds Error	Σφάλμα εκτός ορίων
Segmentation Fault	Σφάλμα καταμερισμού
Privilege Check	Έλεγχος δικαιωμάτων
Vulnerability	Τρωτότητα
Hardware	Υλικό Υπολογιστή
Hardware Vulnerability	Τρωτότητα σχετική με υλικό υπολογιστή
Software	Λογισμικό
Patch	Επιδιόρθωση
Bug	Ελλάτωμα (σε υπολογιστή)
Bounds Check Bypass	Παράκαμψη Ελέγχου Ορίων
Variant	Εκδοχή
Branch Target Injection	Παραπλάνηση Στόχου Διακλάδωσης
Out-of-Bounds Index	Δείκτης εκτός ορίων
Cache Miss	Αστοχία μνήμης
Side Channel Attack	Επίθεση πλαϊνού καναλιού
Flush+Reload	Άδειασμα+Επαναφόρτωση
Prime+Probe	Αρχικοποίηση+Διερεύνηση
Loop	Επανάληψη
Hardware Prefetching	Προφόρτωση Υλικού
Poisoning Indirect Branches	Δηλητηρίαση έμμεσων διακλαδώσεων
Adversary Program	Κακόβουλο πρόγραμμα
Victim Program	Πρόγραμμα θύμα
Exploitable Conditional Branch	Εκμεταλλεύσιμη εξαρτημένη διακλάδωση
Spectre Gadget	Μέσο επίθεσης Spectre
Covert Channel	Κρυφό κανάλι

Process ID	Αναγνωριστικό διεργασίας
Exception	Εξαίρεση
Signal Handler	Χειριστής σημάτων
Virtual Address Space	Εικονική διεύθυνση μνήμης
Cyclic Redundancy Check	Συνάρτηση αθροίσματος ελέγχου
Check Digit	Ψηφίο ελέγχου
Digital Fingerprints	Ψηφιακά αποτυπώματα
Error Correcting Codes	Κώδικες ελέγχου λαθών
Browser	Φυλλομετρητής
Cloud Computing Services	Υπολογιστικές υπηρεσίες Νέφους
Servers	Εξυπηρετητές
Virtual Machines	Εικονικές μηχανές
Protection Ring 0	Δακτύλιος προστασίας 0
Address Space Layout Randomizer	Διάταξη Τυχαιοποίηση Χώρου Διεθύνσεων
Serializing	Σειριοποίηση
Speculation Blocking	Παρεμπόδιση υποθετικής εκτέλεσης
Legacy Software	Πεπαλαιωμένο λογισμικό
Instruction Set Architecture	Σύνολο εντολών αρχιτεκτονικής
Indirect Branch Restricted Speculation	Περιορισμένη Εκτέλεση Έμμεσων Διακλαδώσεων
Single Thread Indirect Branch Prediction	Έμμεση Πρόβλεψη Διακλάδωσης Ενιαίου Νήματος
Indirect Branch Predictor Barrier	Φραγμός Πρόβλεψης Έμμεσων Διακλαδώσεων
Endless Loop	Ατέρμων βρόγχος
Return Stack Buffer	Στοίβα Αποθήκευσης Εντολών Επιστροφής
Branch Target Buffer	Μνήμη Στόχου Διακλάδωσης

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

CPU	Central Processing Unit
SoC	System on Chip
RAM	Random Access Memory
USB	Universal Serial Bus
IF	Instruction Fetch
ID	Instruction Decode
EX	Execute
MEM	Memory
WB	Write Back
BHT	Branch History Table
BCB	Bounds Check Bypass
CRC	Cyclic Redundancy Check
ASLR	Address Space Layout Randomizer
ISA	Instruction Set Architecture
IBRS	Indirect Branch Restricted Speculation
STIBP	Single Thread Indirect Branch Prediction
IBPB	Indirect Branch Predictor Barrier
RSB	Return Stack Buffer
BTB	Branch Target Buffer

ΠΑΡΑΡΤΗΜΑ Ι

[1] Spectre & Meltdown - Computerphile - https://www.youtube.com/watch?v=I5mRwzVvFGE [Προσπελάστηκε 21/10/2018]
[2] Meltdown & Spectre - The Worst CPU Bug Ever? - https://www.youtube.com/watch?v=d7ILCoU9d4k [Προσπελάστηκε 21/10/2018]
[3] Spectre and Meltdown attacks explained understandably - https://www.youtube.com/watch?v=mgAN4w7LH2o [Προσπελάστηκε 21/10/2018]
[4] In-depth Understanding of Spectre and Meltdown Vulnerabilities - https://www.youtube.com/watch?v=6Rsl3UYmrCI [Προσπελάστηκε 21/10/2018]
[5] RuhrSec 2018: "The Story of Meltdown and Spectre", Dr. Daniel Gruss & Jann Horn - https://www.youtube.com/watch?v=xP4YATpTGmA [Προσπελάστηκε 21/10/2018]

ΑΝΑΦΟΡΕΣ

[1] https://spectreattack.com/ [Προσπελάστηκε 21/10/2018]
[2] https://spectreattack.com/spectre.pdf [Προσπελάστηκε 21/10/2018]
[3] https://en.wikipedia.org/wiki/Spectre_(security_vulnerability) [Προσπελάστηκε 21/10/2018]
[4] https://www.csoonline.com/article/3247868/vulnerabilities/spectre-and-meltdown-explained-what they-are-how-they-work-whats-at-risk.html [Προσπελάστηκε 21/10/2018]
[5] https://en.wikipedia.org/wiki/Side-channel_attack [Προσπελάστηκε 21/10/2018]
[6] https://en.wikipedia.org/wiki/Vulnerability_(computing) [Προσπελάστηκε 21/10/2018]
[7] https://en.wikipedia.org/wiki/Memory_protection [Προσπελάστηκε 21/10/2018]
[8] https://blog.barkly.com/meltdown-spectre-patches-list-windows-update-help [Προσπελάστηκε 21/10/2018]
[9] https://googleprojectzero.blogspot.com/search?q=spectre [Προσπελάστηκε 21/10/2018]
[10] https://www.theguardian.com/technology/2018/jan/04/meltdown-spectre-worst-cpu-bugs-ever-found-affect-computers-intel-processors-security-flaw [Προσπελάστηκε 21/10/2018]
[11] https://en.wikipedia.org/wiki/Cpu [Προσπελάστηκε 21/10/2018]
[12] https://en.wikipedia.org/wiki/Branch_predictor [Προσπελάστηκε 21/10/2018]
[13] https://en.wikipedia.org/wiki/Speculative_execution [Προσπελάστηκε 21/10/2018]
[14] https://en.wikipedia.org/wiki/Concurrency_(computer_science) [Προσπελάστηκε 21/10/2018]
[15] https://en.wikipedia.org/wiki/Instruction_pipelining [Προσπελάστηκε 21/10/2018]
[16] https://en.wikipedia.org/wiki/Computer_data_storage [Προσπελάστηκε 21/10/2018]
[17] https://en.wikipedia.org/wiki/Random-access_memory [Προσπελάστηκε 21/10/2018]
[18] https://en.wikipedia.org/wiki/CPU_cache [Προσπελάστηκε 21/10/2018]
[19] https://en.wikipedia.org/wiki/Memory_hierarchy [Προσπελάστηκε 21/10/2018]
[20] https://en.wikipedia.org/wiki/Cache_hierarchy [Προσπελάστηκε 21/10/2018]
[21] https://en.wikipedia.org/wiki/Computer_memory [Προσπελάστηκε 21/10/2018]
[22] https://en.wikipedia.org/wiki/Out-of-order_execution [Προσπελάστηκε 21/10/2018]
[23] https://en.wikipedia.org/wiki/Pipeline_(computing) [Προσπελάστηκε 21/10/2018]
[24] https://blog.barkly.com/meltdown-spectre-bugs-explained [Προσπελάστηκε 21/10/2018]
[25] https://en.wikipedia.org/wiki/Branch_target_predictor [Προσπελάστηκε 21/10/2018]
[26] https://www.bleepingcomputer.com/news/security/mozilla-confirms-web-based-execution-vector-for-meltdown-and-spectre-attacks/ [Προσπελάστηκε 21/10/2018]
[27] https://www.techarp.com/guides/complete-meltdown-spectre-cpu-list/ [Προσπελάστηκε 21/10/2018]
[28] https://www.theregister.co.uk/2018/01/04/intel_amd_arm_cpu_vulnerability/ [Προσπελάστηκε 21/10/2018]
[29] http://blogs.cornell.edu/classeit/2018/01/05/meltdown-and-spectre-vulnerability/ [Προσπελάστηκε 21/10/2018]
[30] https://stackoverflow.com/questions/48089426/what-is-a-retpoline-and-how-does-it-work [Προσπελάστηκε 21/10/2018]
[31] https://en.wikipedia.org/wiki/Cloud_computing [Προσπελάστηκε 21/10/2018]
[32] https://en.wikipedia.org/wiki/Covert_channel [Προσπελάστηκε 21/10/2018]