אבטחת מידע מתקדם (סייבר)

שיעור 1 - מבוא

הערת אזהרה

- מטרת הקורס הינה הכרה טובה יותר של מערכות המחשב והפגעויות האפשריות בהם.
- כל המנסה את הנלמד בקורס שלא במסגרת הקורס, עושה זאת על אחריותו האישית.
- יש לראות בכל האמור מידע כללי בלבד.

ידע נדרש

- שפות עיליות (חובה)
  - C
- אסמבלי
  - ילמד בהקדמה לקורס Intel 32-bit IA32 assembly
- ידע בסיסי במערכות הפעלה
  - הבנת דרך הפעולה של מערכת הפעלה והמבנה שלה
  - השימוש בחלונות, כולל תכנות ודיבוג
  - ניסיון כמשתמש וירטואליזציה

הבא נתחיל במורשת קרב

מעביר המורשת: סא"ל (מיל') גיא לשם

מבצע "תרנגול 53"

- **מבצע תרנגול 53** הוא מבצע שנערך במהלך מלחמת ההתשה, בלילה שבין 26 ו-27 בדצמבר 1969, ובו השתלטו כוחות צה"ל על תחנת מכ"ם מצרית בראס ע'אריב, והביאו את המתקן לישראל.
- בפעולה השתתפו יחידות מחטיבת הצנחנים (גדוד 50 וסיירת צנחנים) ומחיל האוויר, והיא נחשבת לאחת הפעולות הנועזות והיצירתיות בתולדות צה"ל - "כלקוחה מעולם הסרטים", כך כינה אותה הרמטכ"ל, רב-אלוף חיים בר-לב.


כאן 11
212
099
חדשות הערב

כ-43 שנה אחרי...

העורף הווירטואלי

'האקרים מסין פרצו וגנבו מידע על כיפת ברזל'

לפי בלוג אבטחה, קבוצת ההאקרים שנהנית מחסות הממשל בבייג'ינג פרצה למחשבי שלוש חברות שהיו מעורבות בפיתוח המערכת, וגנבו גם מידע על "חץ 3" וכלי טיס בלתי מאוישים

ynet | פורסם: 29.07.14 , 12:51

Recommend 228

בלוג האבטחה KrebsOnSecurity מדווח היום (ג') כי האקרים סינים פרצו למחשבים של שלוש חברות שהיו מעורבות בפיתוח מערכת "כיפת ברזל", וגנבו כמויות עצומות של מסמכים רגישים הנוגעים לטכנולוגיית יירוט הרקטות הישראלית. לפי הדיווח, הפריצות אירעו בשנים 2011 ו-2012.

כיפת ברזל מיירטת רקטה באזור גוש דן
צילום: עידו ארז

אז והיום, מה השתנה ???

- אם נבחן את הקשר בין מבצע תרנגול אשר התרחש בשנת 1969 וגניבת תוכניות "כיפת ברזל" מפרסום שפורסם בשנת 2012, ניתן להבחין כי השחקנים הראשיים במשחק התחלפו, ההגעה הפיזית אל האתר על מנת להוציא מידע התחלף בחדירה לרשת האינטרנט, השומרים אשר שמרו על תחנת ה"מכם המצרי" ואף קיפחו את חייהם התחלפו באנשי אבטחת מידע.

אם כן איך ניתן להתמודד בעולם הטכנולוגי עם האתגריים העתידים של גניבת מידע\הריסת מידע\מניעת מידע ???

ברוכים הבאים ליום עיון
והשתלמות מעשית
באבטחת מידע
החברה לאוטומציה החדשה
גאה להציג
תרחיש איומי הסייבר

אוטומציה החדשה
one1 הבית הטכנולוגי שלך

FORTINET

מה זה "סייבר"?


if(parameters.contains("name"))
hql += " and p.name = :name";
}
if(parameters.contains("age")){
hql += " and p.age = :age";
}
TypedQuery<Person> query = em.
if(parameters.contains("name")
query.setParameter("name"
}
if(parameters.contains("age")
Integer

מי עושה סייבר?

- סייבר הפנאי (Recreational Cyber)
  - פצחנים (Hackers)
- פשעי סייבר (Cyber-Crime)
  - מאפיה (Mafia)

- סייבר תעשייתי (Industrial Cyber)
  - חיפוש אחר IP (Looking for IP)
- סייבר מלחמתי (CyberWar)
  - מדינות ומעצמות (States and Superpowers)

"סייבר" בתוכנה

- פונקציונאליות רדומה
- הוספת פונקציונאליות לתוכנה קיימת
- Exploitation
  - Find Bug
  - Exploit It!
  - Fame & Fortune

למה תוכנה מסובכת?

- אין שליטה על כל התוכנה שרצה
  - מערכת הפעלה
  - ספריות
  - קוד מיובא
  - **קומפילציה**

אז איך מתגוננים – מה כדאי לדעת?

- איך קוד רץ במחשב?
  - מערכת ההפעלה
  - מבנה תהליך
- מה באמת רץ במחשב?
  - משפה עילית לקוד מכונה
- איך אפשר לקרוא את זה?
  - Assembly
- מה אנחנו מכירים?

תכנית כללית לקורס

- מבוא.
- הכרת קוד מכונה *X86* (Assembly)
- הכרת מערכות הפעלה וטבעות הגנה (Protection Rings)
- הכרת כלי הנדסה לאחור (Reverse Engineering) – IDA, Immunity Debugger, HxD.
- הכרת פונקציות C מסוכנות (Dangerous C system calls)
- הכרת חולשות תוכנה:

Buffer Overflows, Stack overflow, Heap overflow, Integer overflow (signedness), Format string, Use after free, Double free, Dangling pointer, Uninitialized local variables.

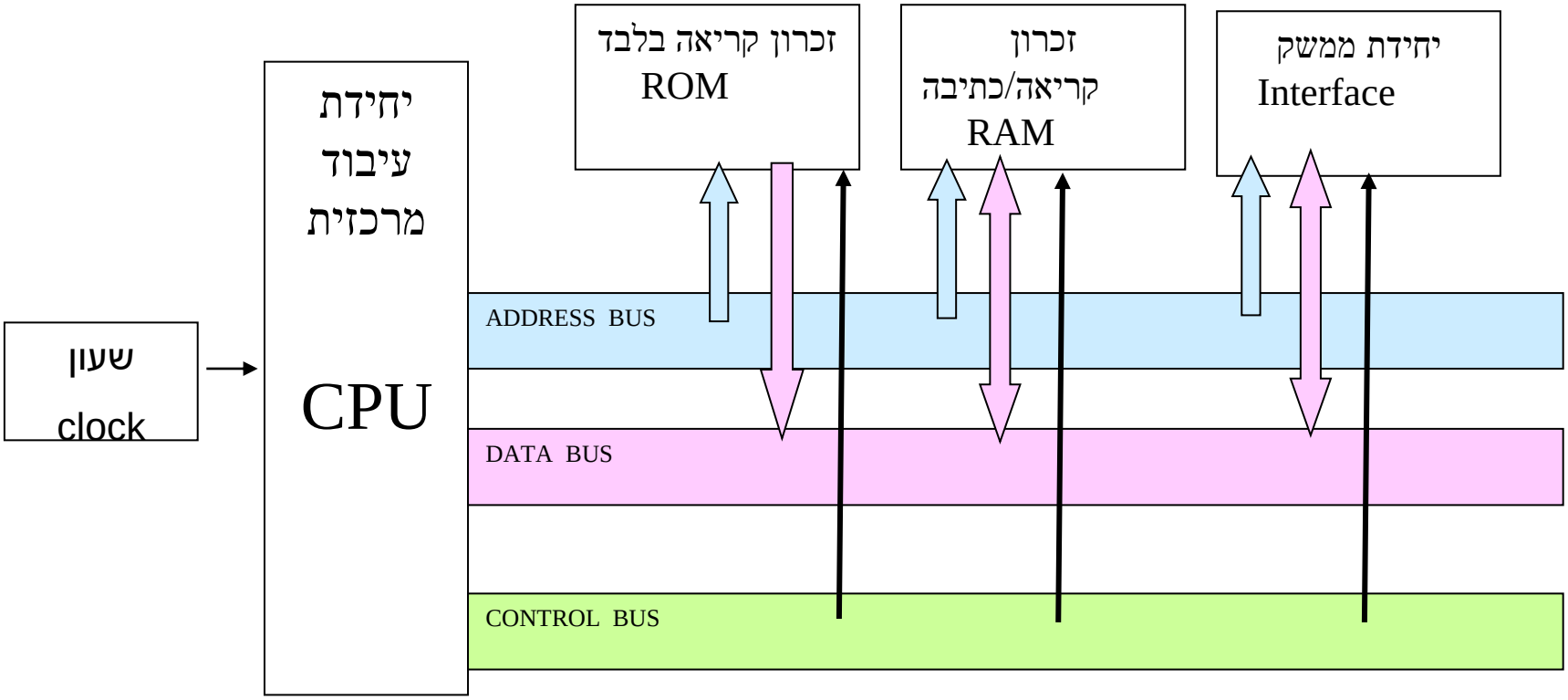
- הכרת תוכנות זדוניות כגון – וירוסים, Shellcoding, ועוד.
- הכרת כלים להתמודדות עם תוכנות זדוניות: cuckoo sandbox, כלי לינוקס.
- התמודדות עם תוכנות זדוניות - **ניתוח סטטי** העוסק בניתוח התוכנה מתוך הקוד שלה, כפי שהוא מופיע בקובץ ה- EXE <u>ללא הרצה של התוכנה.</u>
- התמודדות עם תוכנות זדוניות – **ניתוח דינמי** המתבצע <u>בזמן ריצה,</u> ניתוח דינמי מפעל את התוכנה הנחקרת ועוקב אחר פעולתה לצורך כך תהליך אחר, Debugger, שולט על התהליך הנחקר.
- הרצאות סטודנטים.

מנהלות

- ציון בקורס:
  - 10% תרגילי בית (ביצוע סדנאות שנתחיל בכיתה)
  - 10% הצגה פרויקט גמר
  - 80% ציון בחינה סיום

מפגש	נושא מרכזי ותחומים	משימות קריאה ומטלות הגשה
1	מבוא	
2	הכרת קוד מכונה (Assembly) X86	סדנת הגשה
3	טעינה ומבנה קובץ הרצה וביצוע דיס-אסמבלר והכרת כלי הנדסה לאחור IDA – (Reverse Engineering).	סדנת הגשה
4	ניתוח סטטי ודינמי של תוכנה	סדנת הגשה
5	ניתוח נוזקות - (Malware Analysis)	סדנת הגשה
6	שינוי ההתנהגות של מערכת (Hooking)	סדנת הגשה
7	חולשת תוכנה 1 – Overflows Buffer	סדנת הגשה
8	חולשת תוכנה 2 – Format string	סדנת הגשה
9	חולשת תוכנה 3 – WEB	סדנת הגשה
10	טיפול בחריגות וטכניקות נגד Anti-Reversing	סדנת הגשה
11	ניטור תעבורה והתמודדות עם מתקפות בארגון.	סדנת הגשה
12	טכניקות וכלים למציאת חולשות תוכנה (Fuzzing with Spike)	סדנת הגשה
13	הרצאות סטודנטים	הצגת פרויקט מסכם

מרכיבים עיקריים של מערכת מחשב



עיקרון עבודת המערכת מחשב

מערכת מבוססת מיקרו מחשב כוללת את שלושת המרכיבים הבאים:

- ***CPU*** **- (Microprocessor) Central Processing Unit** - יחידת עיבוד מרכזית (מעבד)
- ***Main Memory*** **-** זיכרון ראשי בו מאוחסנים פקודות ונתונים של התוכנית בזמן ביצוע
- ***Input/Output Devices*** **–** התקנים ליצירת קשר בין המעבד לבין המשתמש
- ***Memory and I/O Subsystem*** **–** פסים להעברת מידע בין המעבד לבין הזיכרון ובין המעבד לבין התקני קלט/פלט
- ***Address Bus*** **-** מעביר כתובת גישה ממעבד לתאי זיכרון או להתקני קלט/פלט
- ***Data Bus*** - מעביר נתונים
- ***Control Bus*** - מעביר אותות בקרה לניהול גישה לזיכרון ו להתקני קלט/פלט
  - כל מסלול הוא למעשה אוסף של קווים המקשרים את ה- CPU אל מרכיבי המערכת. כל מרכיבי המערכת מחוברים אל ה- CPU במקביל .
- כל משימה שהמערכת צריכה לבצע , כתובה כתוכנית בשפת תכנות כלשהי, מתורגמת לשפת מכונה ע"י קומפיילר ומאוחסנת בזיכרון כאוסף נתונים בינאריים בסדר מסוים (Set of Operation Codes - OpCodes).
- אל מנת לבצע המשימה ה-CPU פונה לאזור בזיכרון הראשי איפה נמצאת התוכנית המקודדת, מביא את ה-Opcode של הפקודה הנוכחית מזיכרון אל ה-CPU, מפענח את ה-Opcode ומבצע את הפקודה, שוב פעם פונה לזיכרון להבאת פקודה הבאה וכך עד סוף התוכנית.

מבחינה אחת לפחות, המיקרופרוסור הוא מכונה פשוטה מאוד. הוא חוזר כל הזמן על הרצף:

*הבא את ההוראה הבאה מהזיכרון- **Fetch***

*בצע את ההוראה - **Execution***

לך לשלב 1.

חמש שלבי ביצוע הפקודה

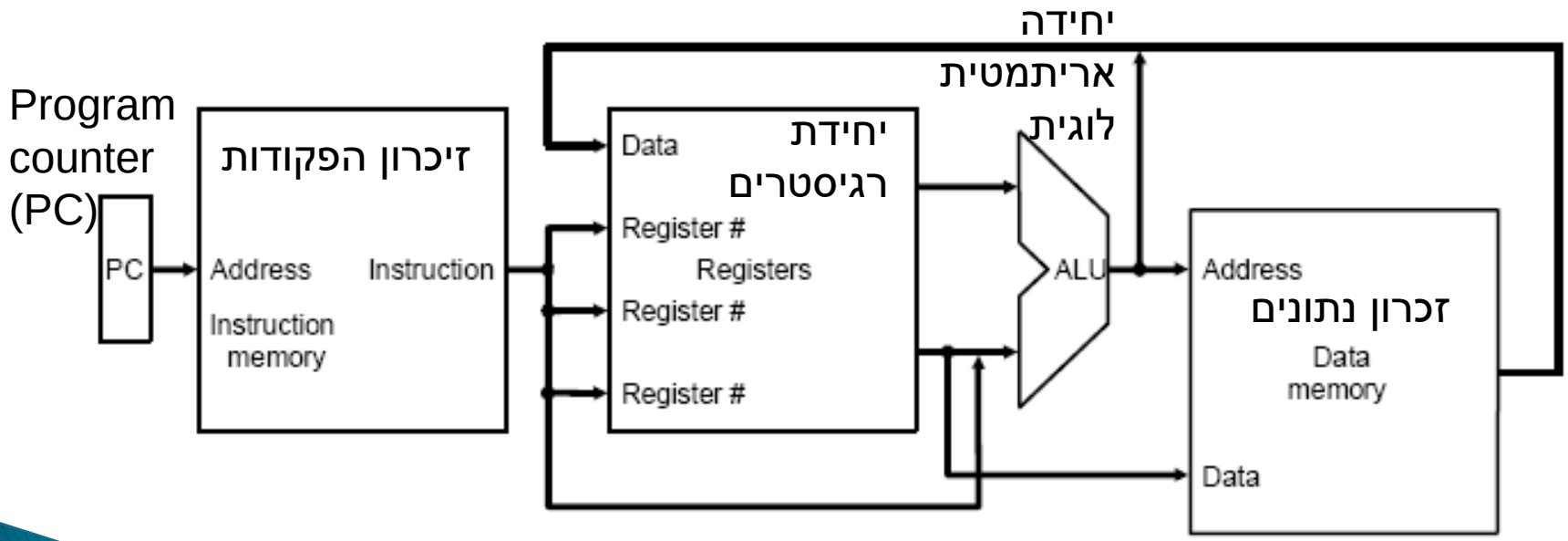
1) Instruction Fetch – הבאת Opcode של הפקודה לביצוע מזיכרון ראשי אל ה-CPU .
2) Instruction decode – פענוח הפקודה (Opcode) שנמצאת באוגר IP ע"י החומרה
3) Execution – ביצוע הפקודה (עיבוד אריתמטי , אחסון זמני)
4) Memory – קריאה\כתיבה מ\אל הזיכרון הראשי.
5) Write back – כתיבת התוצאות העיבוד חזרה ליחידת הרגיסטרים.

בסך-הכל קיימות חמש פעולות שונות או חמישה מחזורי פס אפשריים:

1) קריאה מהזיכרון.
2) כתיבה לזיכרון.
3) קריאה מקלט/פלט.
4) כתיבה לקלט/פלט.
5) סרק (פעולה פנימית שאינה דורשת גישה לזיכרון או למיפתח קלט/פלט).

תיאור מפושט של מעבד MIPS

- זיכרון פקודות (Read Only, חוץ מזמן טעינת תכנית חדשה).
- זיכרון נתונים (Read/Write).
- רגיסטרים
- Program counter (PC)
- ALU (arithmetic logic unit)



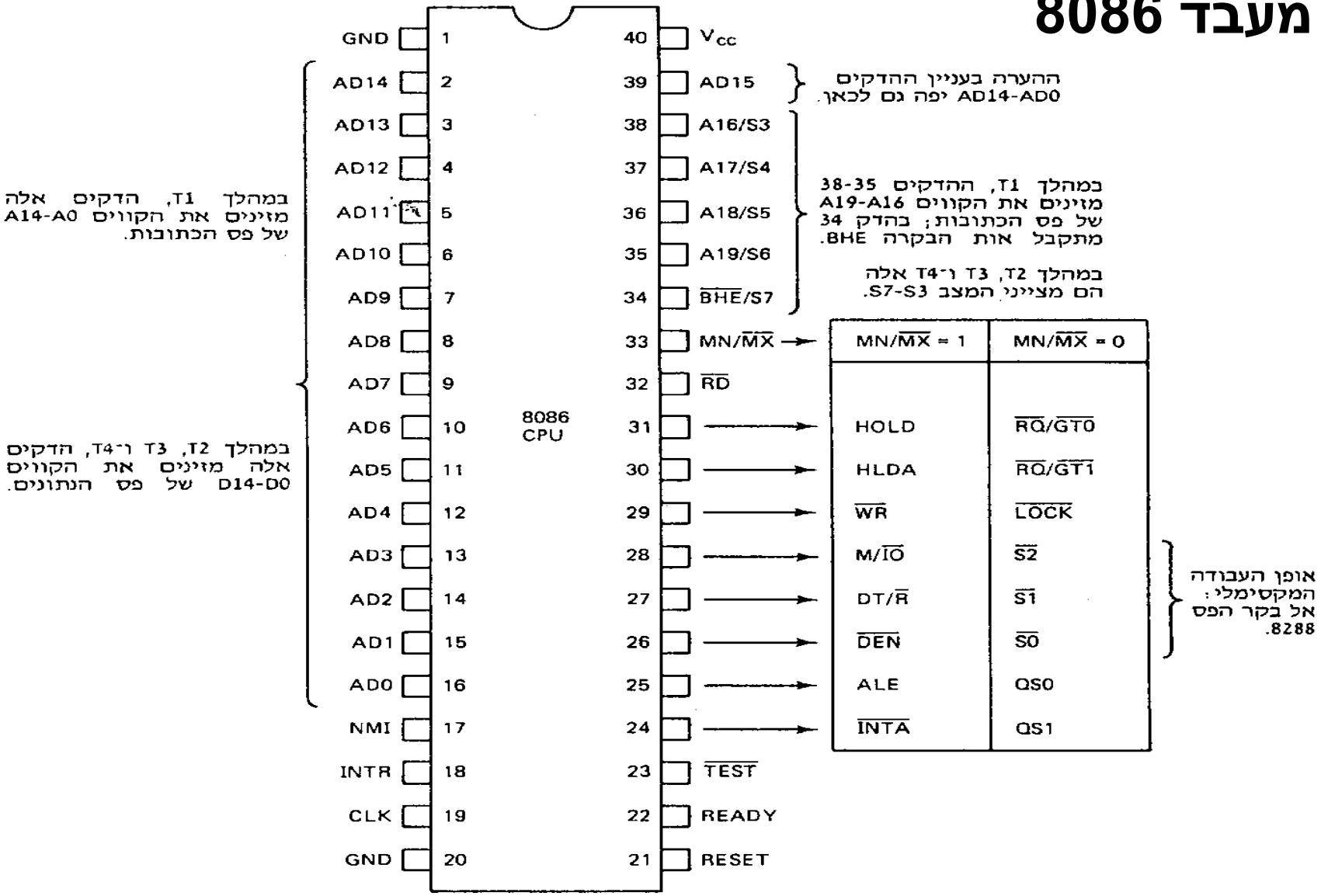
משפחת 80x86

- מתכנת בשפת סף, שהיא שפה נמוכה, צריך להכיר לא רק את מבנה ההוראות וכללי השפה אלא גם הארכיטקטורה של המחשב אליו מתייחסות ההוראות בשפה זו.
- המעבד שנציג הוא מעבד 8086 שייצרה חברת אינטל בשנת 1978.
- חברת אינטל החלה בפיתוח מעבדים למחשבים אישיים בשנות 1970, ומאז היא הוציאה לשוק גרסאות שונות של מעבדים המכונים 'דורות'.
- שמות המעבדים מחמשת הדורות הראשונים נקבעו על-ידי מספרים המתחילים בספרות 80 והם: 8086 ו-80186, 80286, 80386, 80486. החל מן הדור השישי, התחילה חברת אינטל לכנות את המעבדים בשם פנטיום (פנטה = 5 ביוונית) ולמספרם במספרים מ-40586 ואילך, זאת כדי להגן על זכויות היוצרים של החברה.

משפחת 80x86

- כל דור חדש של מעבדים בא לשפר את כושר העיבוד של המעבדים מן הדור הקודם ויחד עם זאת הוא שמר על עקרון התאימות (Compatibility).
- לפי עקרון התאימות, כל מחשב המבוסס על מעבד מדור חדש יהיה מסוגל להריץ גם תכניות הכתובות למעבדים מדורות קודמים.
- כדי לשמור על תאימות, המעבדים מבוססים על אותה ארכיטקטורה דומה ולכל המעבדים יש קבוצת הוראות (ISA) בסיסית, והשינויים מדור לדור מתבטאים בעיקר בהרחבת קבוצת ההוראות ובארגון מעבד שונה.
- בכל גרסה חדשה של מעבדים אפשר להגדיר צורת עבודה (הנקראת real mode) המאפשרת שימוש בארכיטקטורה של מעבד 8086

מעבד 8086



הגדרות ההדקים של ה-8086. להדקי הבקרה 24-31 תפקידים שונים באופן העבודה המינימלי ובאופן העבודה המקסימלי.

תכונות של המעבד 8086

המעבד 8086 יוצר ע"י חברת INTEL בשנת 1978.

המעבד 8086 בעל תכונות הבאות:

- רוחב פס נתונים - 16 סיביות
- רוחב אוגרים פנימיים - 16 סיביות , זאת אומרת שהוא יכול לעבד נתונים בעלי 16 סיביות בפעולות אריתמטיות - לוגיות
- רוחב פס כתובות כלפי זיכרון - 20 סיביות , זאת אומרת שהוא יכול לגשת לזיכרון ראשי בגודל 2^{20} = 1,048,576 = 1 Mbyte
- רוחב פס כתובות כלפי התקני קלט/פלט - 16 סיביות , זאת אומרת שהוא יכול להתחבר למספר התקנים: 2^{16} = 64535 Devices

- המעבד 8086 יכול לגשת לזיכרון להבאת נתונים בגודל 1 byte או 1 word=2 byte בגישה אחד, תלוי בפקודה.

- המעבד 8088 דומה מאוד למעבד 8086 חוץ מרוחב פס נתונים חיצוני שהוא בעל 8 סיביות - להבאת נתון בגודל 1 word=2 byte המעבד 8088 פעמיים ניגש לזיכרון .

מיבנה פנימי של המעבד 8086

- המעבד 8086 נקרא גם כן CPU מורכב בתוכו משתי יחידות פונקציונאליות עיקריות:
  - יחידה להבאת נתון מזיכרון – BIU – Bus Interface Unit

    יחידה זאת בונה את כתובת גישה לזיכרון, פונה לזיכרון, קוראת את הפקודה הבאה מהזיכרון לפס נתונים, מעבירה אותה ל-CPU ומאחסנת אותה לחוצץ של 6 בתים - תור פקודות(QUEUE).

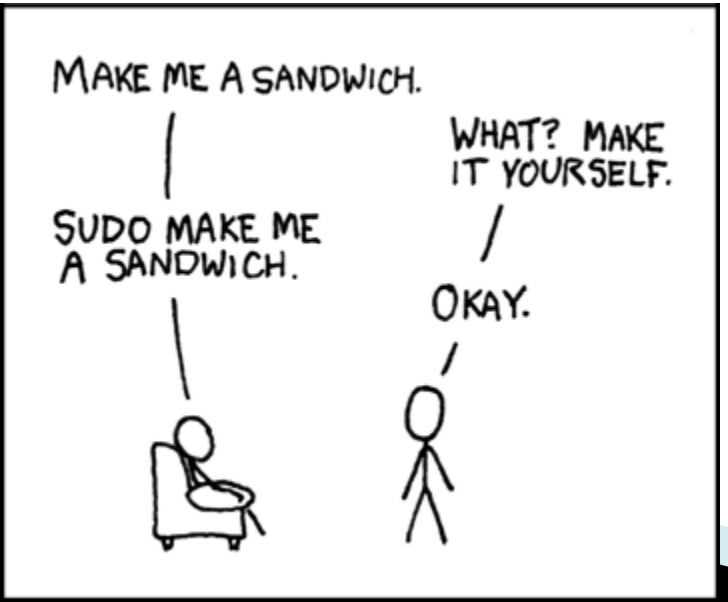
    יחידת BIU כוללת 4 אוגרי סגמנט: (1) Code Segment (2) Data Segment (3) Stack Segment (4) Extra Segment, ואוגר Instruction Pointer (כולם – 16 סיביות)
  - יחידת ביצוע – EU – Execution Unit

    יחידה זאת לוקחת פקודות מהחוצץ ומבצעת אותן ללא תלות ביחידת BIU.

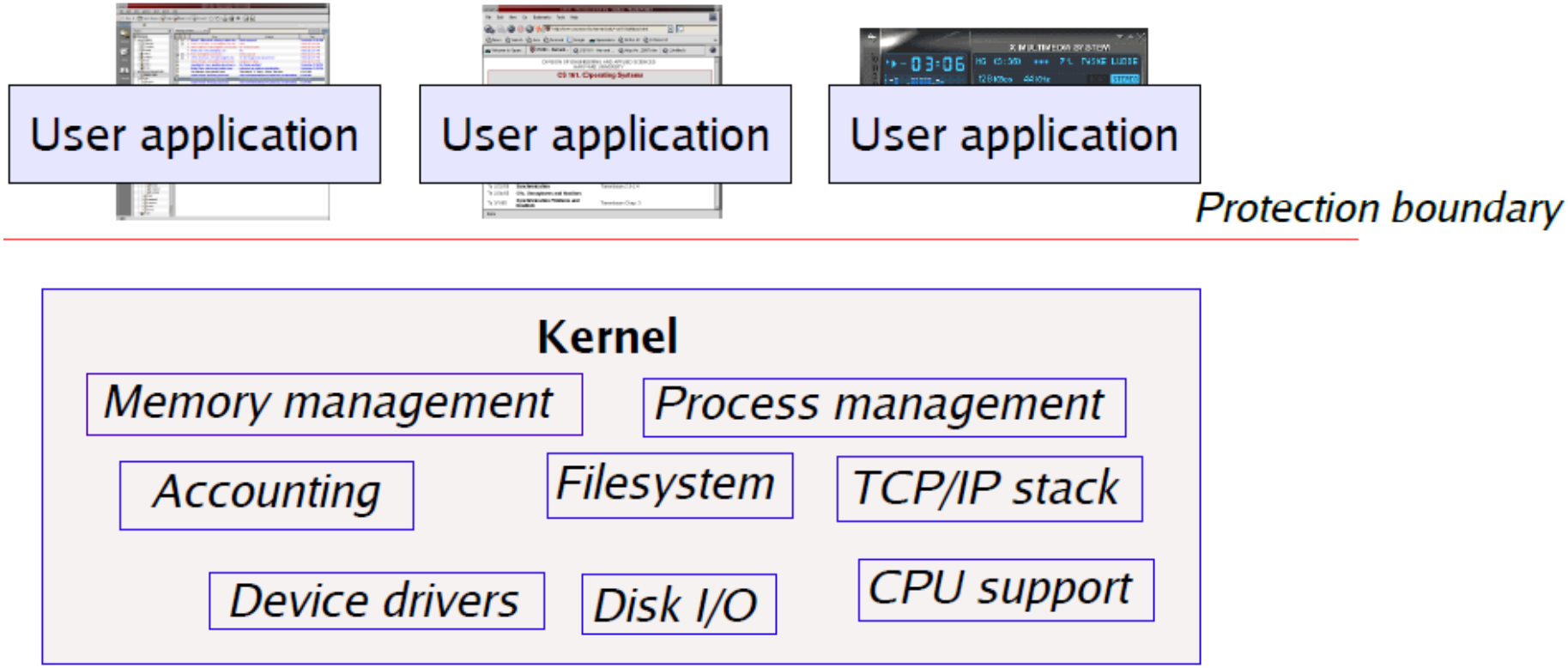
    החוצץ (QUEUE) עובד בעיקרון של FIFO – הפקודה שהוכנסה ראשונה לתור הפקודות ע"י יחידת BIU יוצת ראשונה לביצוע ע"י יחידת EU.
  - שתי היחידות האלו מפצלות את עומס עבודת ה-CPU כדי לעלות את מהירותו ואת יעילותו של המעבד.

מה עושה מערכת ההפעלה?

- מתווכת בין החומרה לבין התוכנה באופן "שקוף למשתמש"
- מנהלת משאבי המחשב
- מייצרת לכל תהליך סביבה שבה הוא רץ
  - הפרדה בין תהליכים
  - קשר בין תהליכים
  - מבני נתונים בזיכרון המייצגים את התהליכים
- אינה מאפשרת גישה ישירה לחומרה
- מגנה על עצמה
  - Ring0 vs Ring3



Operation System Overview



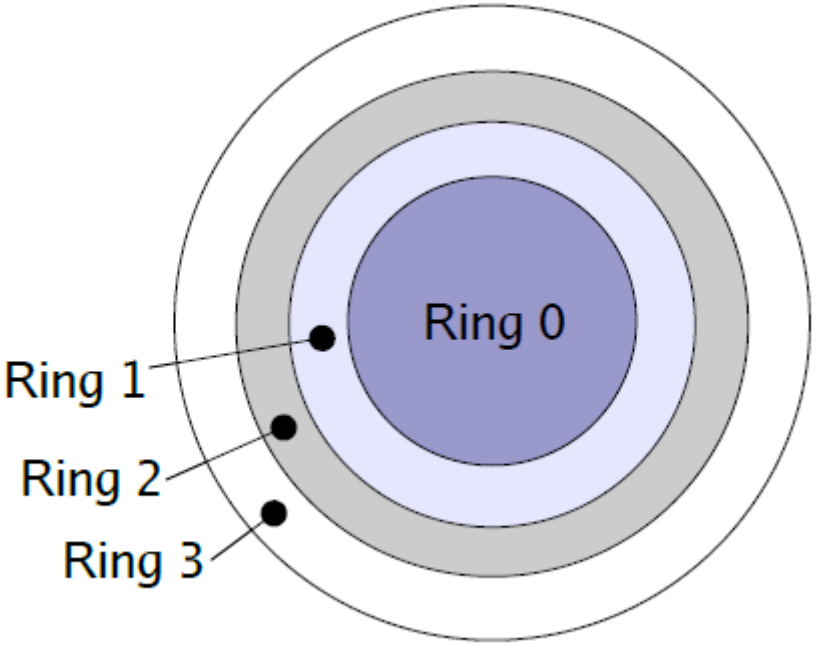
טבעות הגנה (Protection Rings)

- משפחת x86 של Intel כוללת ארבע "טבעות" (The Intel x86 family has four protection “rings“).
- טבעות ההגנה מציינות את סוג ההוראות והזיכרון הזמינים עבור קוד ההפעלה

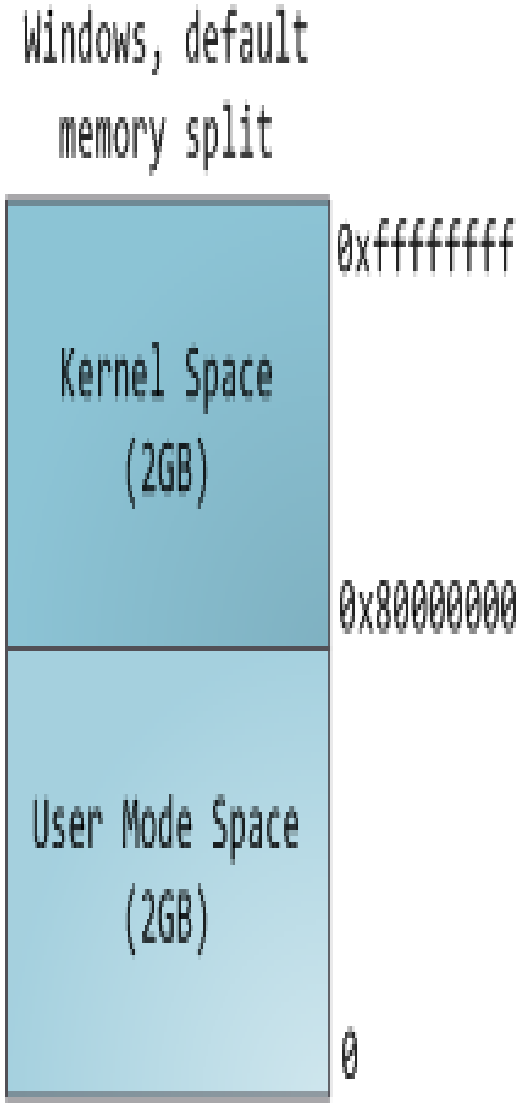
(The protection rings specify the type of instructions and memory available for the running code)

טבעות הגנה (Protection Rings)

- לטבעת 0 (הטבעת הפנימית) יש את ליבה של מערכת ההפעלה (מצב הקרנל).
- לטבעת 3 יש קוד יישום (מצב משתמש).
- בטבעות 1 ו -2 ניתן להשתמש עבור קוד מערכת חסוי פחות.



מרחב כתובות וירטואלי (Virtual address space)



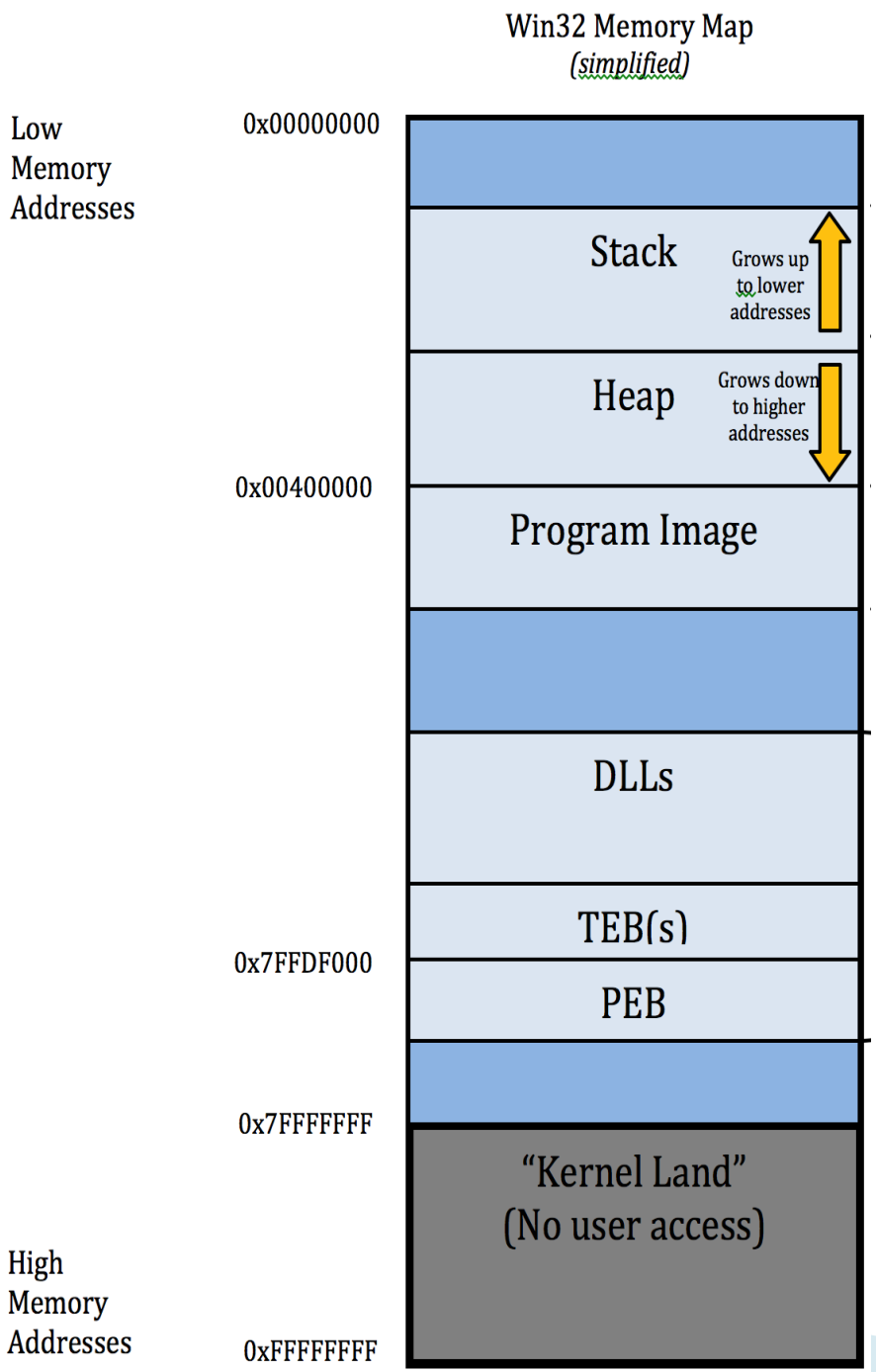
- לכל תהליך משתמש (user-mode process) יש מרחב כתובות וירטואלי משלו.
- כל הקוד הרץ\פועל במצב ליבה (kernel mode) משתף מרחב כתובות וירטואלי יחיד (שטח מערכת - *system space*).
- במערכת 32 סיביות, שטח הכתובות הווירטואלי הוא 2^{32} בתים (4 ג'יגה בייט).
- 2 ג'יגה בייט התחתון משמשים שטח המשתמש (user space)
- 2 ג'יגה בייט העליון משמשים שטח המערכת (*system space*).

מרחב כתובות וירטואלי (Virtual address space)

- לקוד הפועל במצב משתמש (user mode) יש גישה למרחב המשתמש (user space) אבל אין גישה למרחב המערכת (system space).
- לקוד הפועל במצב ליבה (kernel mode) יש גישה הן שטח המערכת (system space) והן לשטח המשתמש (user space) של תהליך המשתמש במצב הנוכחי (current user-mode process).

מפת הזיכרון של תהליך (Process memory map)

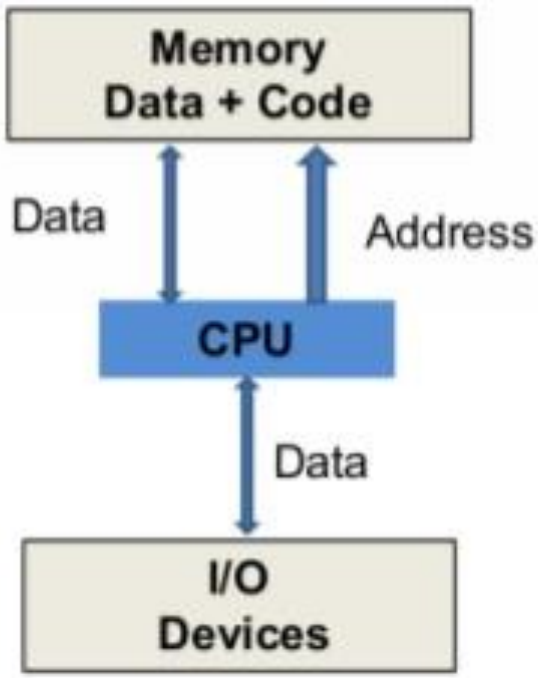
- PEB (Process Environment Block) – הוא מבנה זיכרון אשר מכיל מידע על התהליך, מידע כגון רשימת המודולים הטעונים, כתובת ה-HEAP, BaseAddress ועוד.
- TEB (Thread Environment Block) – דומה ל-PEB אבל מכיל פרטים על החוט הרץ.
- DLL (Dynamic-Link Library) - קובצי ה-DLL הרלוונטיים נטענים זיכרון הפיזי.



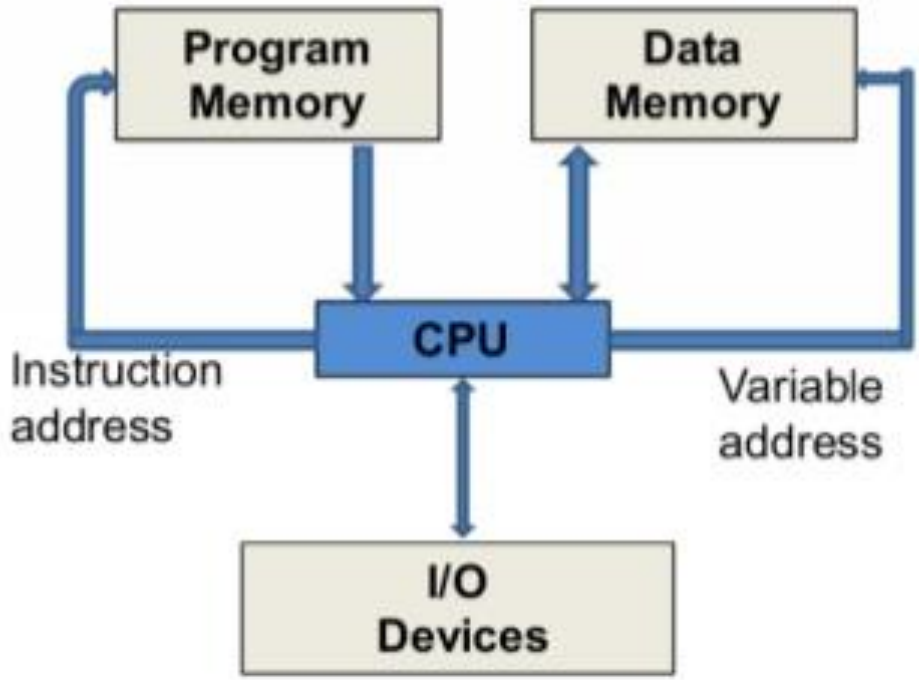
מחסנית, ערימה ותמונת התוכנית
Stack, Heap, Program Image

- מחסנית (Stack) – אזור בזיכרון המאחסן משתנים מקומיים ופרמטרים פונקציונליים.
- ערימה (Heap) – אזור בזיכרון המספק הקצאת זיכרון בזמן ריצה. מיועד לנתונים שחייבים לחרוג מהפונקציה של ההקצאה.
- תמונת התוכנית (Program Image) – אזור בזיכרון המכיל מיקום קובץ ההפעלה. כולל את קטע הטקסט (text section) - קוד הפעלה של התוכנית, קטע הנתונים (data section) – המכיל נתונים גלובליים של התוכנית.

Von Neumann vs. Harvard Architecture



Von Neumann Machine



Harvard Machine

קצת על malware (תוכנה זדונית) וה"בונבונים" שכתבו\השתמשו בו

- נראה מול מי ומה אנו מתמודדים ?

האקר וקראקר

- **האקר** (בעברית: פַּצְחָן) הוא מושג המשמש לתיאור אדם המתעניין בטכנולוגיה ובשימוש יצירתי בה. בדרך כלל הכוונה היא למומחי מחשבים המשתמשים בהם בצורה יצירתית.
- משמעות המושג בהקשר של מחשבים השתנתה לאורך השנים, כיום הציבור והתקשורת משתמשים במושג האקר הן לחיוב והן לשלילה.
- דוגמה חיובית אפשר למצוא למשל בתיאור התקשורתי של לינוס טורבאלדס אשר יצר את גרעין מערכת ההפעלה לינוקס. יש הרואים באנשים מסוגו של טורבאלדס כהאקרים האמיתיים, אלו המשתמשים ביכולותיהם לבניית מוצרים ופיתוח רשת האינטרנט.
- למרות זאת, ישנם אלו המשתמשים במושג האקר גם כאשר מדובר ב**קראקרים** (בעברית **פרצן**) המנצלים את הרשת לפעילות העוברת על החוק.
- הקראקרים מסוגלים להשתלט באופן עוין על מחשבים מרחוק דרך הרשת, ולפרוץ מערכות אבטחה של מחשבים תוך גרימת נזק, לעתים בלתי הפיך, לתוכנות המחשב ולרשת.

הנה "נוער הזהב"

Jeremy Jaynes, North Carolina, USA

Spammer (earns estimated at 24M$, sentenced 9 years)

Jeanson James Ancheta, California, USA

Botnet Operator, adware, DDoS, spam (generated more than 107k$ in advertising), arrested in 2005

Farid Essebar, Morocco

Author of Zotob worm in 2005 (at age 18) to facilitate credit card forgery scams

Maria Zarubina, Russia (wanted)

DDoS extortion in 2004

Michael and Ruth Ha'Efrati, Israel

Trojans for industrial espionage in 2004

רוגלה (Spyware)

- **רוגלה** (Spyware) או **תוכנת ריגול**, היא תוכנית מחשב העוקבת בחשאי אחר הרגלי הגלישה של המשתמש במחשב שבו היא מותקנת, ומעבירה מידע על הרגלים אלו לאתרים ששתלו אותה. מטרת הרוגלות היא לזהות את העדפותיו של המשתמש ותחומי העניין שלו, בדרך כלל על מנת להתאים לו פרסומות אישיות בעת הגלישה ולהפיק מכך רווח כלכלי.
- הרוגלות מצורפות לתוכנות תמימות לכאורה (למשל קאזה). תוך כדי הסכמה להתקנת התוכנה מסכים המשתמש גם להתקנת הרוגלה ולפעולתה על מחשבו, בדרך כלל בלי להיות מודע לכך.
- תוכנות ריגול שמטרתן ריגול טהור (ולא בהכרח הפקת רווחים) מסוגלות לדעת כמעט כל דבר שבוצע על ידי משתמש המחשב, כגון יעדי הגלישה, תאריכים, הסיסמה שהקליד וכדומה.
- הרוגלה מתגלה לעתים עקב המטרד שהיא גורמת, בדמות חלונות פרסומת שקופצים תוך כדי הגלישה, חלונות שמקורם אינו באתר שאליו פונה הגולש. קיימות תוכנות אחדות המשמשות לניקוי המחשב מרוגלה שמותקנת בו.
- משתמשים מתקדמים נוהגים להתקין במחשביהם תוכנת רחרחן (סניפר). מטרת תוכנה זו היא לעקוב אחר תעבורת הרשת במחשב ולרשום מידע זה בקובץ. על ידי כך מאפשרת התוכנה לזהות האם תוכנה שהותקנה זה עתה היא באמת תוכנה תמימה או שמא רוגלה, וכן האם קיימת במחשב פעילות של סוס טרויאני.

תולעת מחשב

- **תולעת מחשב** היא תוכנית מחשב בעלת יכולת שכפול עצמי. התולעת עושה שימוש ברשת כדי להעתיק את עצמה לצמתים אחרים (המהווים מסופים ברשת) ללא הכרח במעורבות משתמשים. בשונה מוירוס מחשב, התולעת לא חייבת להיצמד לתוכנית קיימת. תולעים פוגעות ברשת (ולו רק בשל העובדה שהן צורכות רוחב סרט יקר) בעוד שוירוסים פוגעים בקבצים על מחשב היעד.
- **סוגי תולעי מחשב:**
  - **תולעי דואר אלקטרוני** תולעים אלו מתפשטות באמצעות הודעות דואר אלקטרוני. התולעת תגיע לרוב כדואר אלקטרוני שבו גוף ההודעה או הקובץ המצורף יכילו את קוד התולעת.
  - **תולעי הודעות מיידיות** ההפצה היא באמצעות שירותי הודעות מיידיות. השיטה היא שליחת קישורים, המפנים לאתרים נגועים, לכל הכתובות אשר ברשימת הקשר המקומית.
  - **תולעי IRC** ערוצי צ'ט הם היעד המרכזי. שיטת ההפצה זהה לשיטות שצויינו לעיל-שליחת קבצים נגועים או קישורים המפנים לאתרים נגועים.
  - **תולעי רשתות שיתוף קבצים** התולעת מעתיקה עצמה לתוך תיקיית קובצי השיתוף.
  - **תולעי אינטרנט** תולעים התוקפות פורטי TCP/IP ישירות, במקום להתמקד בפרוטוקולים ברמה גבוהה יותר כמו דואר אלקטרוני או IRC.
  - **תולעי מכשירים ניידים** קטע קוד מתולעת למכשירים ניידים. ההתפשטות היא באמצעות פרוטוקול שן כחולה תולעים אלו הן מהסוג החדש ביותר. הן הופיעו לראשונה בשנת 2004. תפוצת התולעים היא לרוב באמצעות פרוטוקול השן הכחולה המקשר בין מכשירים ניידים, מדפסות, סורקים ומחשבים בטווח של 10 מטרים.

סוס טרויאני

- **סוס טרויאני** הינו תוכנת מחשב מזיקה שמנסה לחדור למחשב תוך התחזות לתוכנה תמימה (השם שאול מהסיפור על הסוס הטרויאני במלחמת טרויה).
- סוס טרויאני מופיע בדרך כלל כקובץ המצורף לדואר אלקטרוני או כתוכנה חופשית להורדה, ובעת הפעלתו יבצע פעילות משעשעת או מועילה, כדי לגרום למקבל הוירוס לשלוח אותה הלאה לחברים נוספים. אותה פעילות משעשעת (למשל, סרטון קצר) היא הסוואה לכך שהתוכנה מתקינה את עצמה במחשב, ועלולה לגרום נזק.
- ישנם סוסים טרויאנים שתפקידם לתת הרשאות למשתמש אחר להיכנס למחשב הנפגע מרחוק. פעולה זו נקראת גם התקנת 'דלת אחורית' (Back Door) סוסים טרויאנים אחרים הם רוגלה, כלומר אוספים מידע (למשל, מספרי כרטיסי אשראי שהמשתמש מקליד) מהמחשב שבו הותקנו ושולחים אותו ליעד מוגדר מראש.
- ישנם שימושים נוספים לתוכנות בלתי-רצויות מסוג זה כמו למשל גניבת סיסמאות והתחזות לבעל המחשב בכניסה לאתרים שונים. תוכנות סוס טרויאני שימשו בישראל לריגול תעשייתי של חברות מפורסמות במשק הישראלי.
- סכנה נוספת היא הפיכת המחשב ל"זומבי" באמצעות תוכנת הסוס הטרויאני. מחשב זומבי הוא מחשב שנשלט מרחוק באמצעות המנגנון של תוכנת הסוס הטרויאני, ואף שרוב הזמן הוא מתפקד כרגיל, ניתן להורות לו מרחוק לבצע פעולה בניגוד לרצון בעליו. כאשר מחשב כזה מחובר לרשת הוא עלול לשמש לביצוע מתקפת מניעת שירות, שליחת דואר זבל אלקטרוני וכו'.

נקודות תורפה\חולשות תוכנה – Vulnerabilities (פגיעויות)

- בכל מערכת (תכנה או חמרה) קיימות נקודות תורפה (חורי אבטחה).
- יצרניות מערכות תכנה\תכנה-חמרה, כמו גם חוקרים מקהיליית האבטחה, מחפשים באופן תמידי אחר נקודות תורפה במערכות.
- לדוגמא: Microsoft מחויבת לפרסם מידע על נקודת התורפה, תוך 90 יום מזמן שנודע לה עליה
- כיום מפרסמת Microsoft פעם בחודש תיאור של חורי האבטחה, ותיקונים (Patches) עבורם
- מירב ה-vulnerabilities הם תוצאה של בגים בתכנות או בתכנון
- חלק הארי נובע מחוסר בדיקות תקינות של מספר, תחום ואורך של פרמטרים

Zero day vulnerability/attack

- מירב ההתקפות (וירוסים, תולעים) הגדולות נצלו חורי אבטחה, שפורסמו עבורם תיקונים (patches).
- מירב כלי ההגנה (Anti-Virus, IPS - Intrusion Prevention Systems) מתגוננים כנגד התקפות ידועות, או התקפות שמנצלות חורי אבטחה ידועים.
- התקפה או חור אבטחה שטרם התפרסם עבורם תיקון (Patch)נקראים Zero day attack או Vulnerability

.

פתרון 1: הנדסה לאחור

- **הנדסה לאחור** היא תהליך של גילוי עקרונות טכנולוגיים והנדסיים של מוצר דרך ניתוח המבנה שלו ואופן פעולתו.
- הנדסה לאחור (RE) גם כוללת גילוי של מבני פרוטוקולים ושל קוד תוכנה, וכן גילוי מידע לא ידוע על מערכת ע"י ניתוח המבנה שלה ואופן פעולתה.
- בעידן התעשייתי, עוד לפני עידן המחשב, הנדסה לאחור הייתה נפוצה מאד
  - עוד לפני חוקי הפטנטים וזכויות היוצרים
- קל יחסית לבצע הנדסה לאחור של מכונות
  - רכיבים פיסיים גדולים
  - אבל לעיתים מסתמכים על חומרים ייחודיים
    - בעלי תכונות מסוימות
    - למשל בשעונים

פתרון 1: הנדסה לאחור של חומרה

- בעידן המחשב – מסובך יותר – בגלל הקוטן והמורכבות
  - והקושי של פירוק למרכיבים
- אבל לא בלתי אפשרי
  - יתכן שהצורה של המוצר מדליפה מידע או שניתן לזהות כזה באופן ויזואלי
  - במעבדים מודרניים:
    - מיקרוסקופים ומיקרוסקופים אלקטרוניים
  - מדידת מתח בזמן הפעולה
  - הזרקת מתח למעבדים ומעקב אחרי התגובה
  - גרימה לשגיאות בחישוב, ומעקב אחרי השינויים
  - קריאת זיכרון, או האזנה ל-bus , אם אפשר
  - ועוד

פתרון 1: הנדסה לאחור של תוכנה

- הבנת קוד תוכנה בשפה עילית
- הבנת קבצי הרצה (כגון: EXE או a.out).
  - אנליזה סטטית – דה-קומפילציה
  - אנליזה סטטית – שפת מכונה
  - אנליזה דינמית – דיבגרים
- קוד ביניים ו- JIT – .NET & Java bytecode.
- רלוונטי גם ל-
  - אופטימיזציה של קוד ע"י המהדר
    - בהינתן קוד המקור או ייצוג פנימי של המהדר
  - Post-link optimization
    - כלומר אופטימיזציה בהינתן קבצי הרצה
  - בדיקות אבטחה של קוד
    - תלויות במה הקוד עושה

פתרון 1: הנדסה לאחור של פרוטוקולים וקבצים

- הבנת המבנה של פרוטוקולי תקשורת וקבצים
- טכניקות יכולות לשלב
  - האזנה לתקשורת (sniffing)
  - RE של התוכנה המתקשרת
  - שינוי נתונים שנשלחו, או משלוח חבילות חדשות, ומעקב אחרי התגובות
  - בחינת קבצים
- בכל המקרים הללו החוקר מתנהג כבלש: בוחן עדויות, חוקר, מחפש מידע חדש, מנתח מידע שהשיג, ואפילו משנה תוכן ועוקב אחרי השינויים

פתרון 2: ניתוח סטטי ודינמי

- ניתוח סטטי עוסק בניתוח התוכנה מתוך הקוד שלה
  - כפי שהוא מופיע בקובץ ההרצה
  - ללא הרצה של התוכנה
- תוכנות שתומכות בניתוח סטטי כוללות
  - דיס-אסמבלרים – מתרגמים את הקוד לאסמבלי
  - דה-קומפיילרים – מתרגמים לשפה עילית, אם ניתן
  - מפענחי מבני קובץ הרצה
    - מפרטים את המבנה של קובץ ה- PE
- גם מועיל שיש
  - Hex dump, strings וכו'

פתרון 2: מה זה דיס-אסמבלר?

- דיס-אסמבלר הוא תכנית, שקוראת קובץ הרצה
  - ממירה את תוכן הקובץ משפת מכונה לאסמבלי
  - מנקודת ראות מ"ה והמעבד הוא סתם תכנית
- דיס-אסמבלר צריך
  - להכיר את מבנה קובץ ההרצה)למשל PEשל חלונות(
  - לפרש פקודות אסמבלי
  - לקשר כתובות לשמות פונקציות ולשמות משתנים
    - כולל לקישורים אל DLL-ים
  - זיהוי מיקום פקודות המכונה
    - במעבדים בהם גודל פקודת מכונה אינו קבוע זה לא תמיד פשוט
  - לזהות בין קוד למשתנים, ובין סוגי משתנים שונים
    - למשל להדפיס נכון מחרוזות, שלמים, וכו'

פתרון 2: Ida Pro

- Ida Pro הוא דיס-אסמבלר אינטראקטיבי
- מיועד ל-RE עם יכולות דיבוג
- Ida Pro Free
  - גרסה קודמת המתאימה לצרכינו
  - תוכנה חופשית
  - הגרסאות הבאות בעלות יכולות נוספות, לא חופשיות
- מאפשרת רישום הערות ומתן שמות לתאי זיכרון
  - למשל משתנים וקוד
  - יכולת מומלצת ביותר
    - מאפשרת לעבוד עם קוד קריא יותר אחרי שזיהינו חלק מרכיביו
    - וכך נעשה

פתרון 2: ניתוח דינמי

- ניתוח דינמי מפעיל את התוכנה הנחקרת, ועוקב אחרי פעולתה
- לצורך כך, תהליך אחר, הנקרא דיבגר, שולט על התהליך הנחקר
  - קובע מתי יעשה מה
  - יכול לעצור ולחדש את הפעולה
  - עוקב ויכול לשנות את כמעט כל משאב של התהליך המדובג
- בשונה מניתוח סטטי, הניתוח נעשה בזמן ריצה
  - ולא על קובץ ההרצה ללא ריצה
  - לכן, ניתן לבחון נתונים שאינם זמינים בניתוח סטטי

פתרון 2: ארגז החול (Sand Box) לבודד את הסיכון

- **ארגז חול** (**Sandbox**) - משמש לתיאור סביבה שבה מאפשרים למשתמשים לעשות ככל העולה על רוחם, ללא חשש מנזק למערכת ההפעלה.
- **אזור חסין לצורך אבטחת מידע -** סביבת מחשוב המגבילה לצורך אבטחה את הגישה לקבצים ולמשאבי מחשוב אחרים. באופן זה, תוכניות המופעלות בתחום 'ארגז החול' אינן יכולות לפגוע בשאר תוכניות המחשב ובמידע המאוחסן בו.
- **מכונה וירטואלית** יכולה לעזור ליצור ארגז חול: גם מי שיש לו שליטה מלאה על המכונה הווירטואלית, לא יכול לצאת ממנה למכונה המארחת. לכן מכונה וירטואלית יכולה לאפשר הדמיית מערכת הפעלה או מכשיר אחד, בתוך מכשיר אחר, בלי האפשרות לפגוע במכשיר האחר.

פתרון 2: מה זה דיבגר ?

- Debugger מאפשר למשתמש
  - הרצת הקוד
    - תוך כדי בדיקת האוגרים והזיכרון של התהליך המדובג
  - שינוי כל אחד מהערכים האלו
    - כולל של הקוד עצמו
  - עצירת ריצת התוכנית (breakpoints) לפי מספר שיטות
  - לעיתים גם פירוש מבנים מסוימים בזיכרון
    - למשל ניתוח מבנה קובץ ההרצה
- כלים נפוצים בחלונות:
  - Ollydbg - מדבגים תהליך ב-user mode בלבד
  - Immunity Debugger - מדבגים תהליך ב-user mode בלבד
  - Windbg - מדבג הכל, הכלי היחידי שמדבג kernel בחלונות
- לינוקס
  - gdb

פתרון 3: Hooking (הוקינג)

- Hooking היא שיטה המשמשת לשינוי ההתנהגות של מערכת ההפעלה, תוכנה או קוד מכונה באמצעות יירוט קריאה לשגרה או הודעה המועברת בין תוכניות.
- **קטע הקוד האחראי ליירוט נקרא Hook.**
- hooking משמש למטרות רבות ובהן חיפוש באגים והרחבת אפשרויות השימוש בשגרות קיימות. באמצעות טכניקה זו ניתן ללכוד את המידע העובר בין המקלדת או העכבר למחשב או לזהות קריאה לפונקציות ספרייה כדי לבחון את התנהגותן.
- ניתן להשתמש בטכניקה זו גם כדי ליצור תוכנות זדוניות. לדוגמה, Rootkit-ים רבים משתמשים בטכניקה זו כדי להתחבא בתוך תהליכים מוכרים כדי להסוות את פעילותם.

פתרון 3: שימושי הוקינג

- ניטור תוכנות
  - מעקב אחרי פונקציות מסוימות (קלט\פלט וזמן ריצה).
  - משמש מפתחים לצורך בדיקת פונקציות מסוימות
    - לדוגמא ניטור פונקציות הקצאה ושחרור זיכרון ע"י תוכנת **valgrind** (שהיא תוכנה ללינוקס המספקת כלים חזקים לדיבוג הזיכרון, פרופיילינג וכדומה. הקונספט בה הוא פשוט: מוסיפים דגל קומפילציה (g-) מקמפלים ומאותו רגע ניתן להריץ את קובץ ההרצה באמצעות valgrind, כשיש ל–valgrind אפשרות לתת לכם הערות אינפורמטיביות).
  - משמש לאיתור התנהגות לא תקינה של תוכנות מסוימות ע"י תוכנת אנטי וירוס
  - כלי עזר ל- Reverse Engineering
  - קוד עוין שמצורף לתכנית
    - למשל למעקב אחרי המשתמש

פתרון 3: שימושי הוקינג

- שינוי תוכנה
  - שינוי קטע קוד / פונקציה מסוימת
  - שימוש לגיטימי – הרחבת האפשריות בתוכנה מסוימת
    - למשל Babylon מוסיפה אפשרות תרגום לכל התוכנות הרצות
  - וירוסים מסוימים ישנו את התנהגות המערכת על מנת להסתתר מהמשתמש
- יש עוד מספר שימושים.
- עבורנו הוקינג חשוב משתי סיבות
  - האחת, נרצה להשתמש בהוקינג בזמן ניתוח קוד דינמי
    - להצגת ערכי ביניים וניטור
- השנייה, לזהות הוקינג בקוד שאנחנו בוחנים
  - למשל כאשר נוזקות ביצעו הוקינג לצרכיהן
  - וכמובן גם לבטל הוקינג כזה

פתרון 4: מערכות אנטי-וירוס

- מערכות אנטי-וירוס מגנות נגד מגוון נוזקות נפוצות
  - וירוסים, תולעים
  - רוגלות, תוכנות פרסום
  - סוסים טרויאניים
  - malicious Browser Helper Objects (BHOs)
  - .key-loggers ,browser hijackers
  - כופרה -ransomeware
  - rootkit ,backdoors
  - malicious LSPs, dialers, fraudtools
- בנוסף, הן מגנות נגד בעיות אבטחה נוספות, כגון
  - הנדסה חברתית
  - עוגיות ריגול
- הן מוגבלות ביכולתן להגן נגד נוזקות לא מוכרות
  - ובפרט נגד נוזקות ייעודיות שנכתבו במיוחד למטרה מסוימת

פתרון 4: זיהוי נוזקות

- זיהוי נוזקות נעשה במגוון של אלגוריתמים היוריסטיים
  - זיהוי שינויים בקובץ
  - בדיקת תוכן קובץ
  - זיהוי התנהגות
  - בדיקת שינויים במערכת הקבצים (white list)
- שני מקרים עקרוניים
  - זיהוי סטטי – בדיקת קובץ ההרצה
    - למשל אחרי הורדת קובץ מהרשת
    - או בהכנסת דיסק USB
  - זיהוי דינמי
    - בזמן ריצה
- וכמובן אפשר לשלב
  - למשל זיהוי חשד באופן סטטי, ווידוא דינמי
    - ואז מציאת חתימה לווירוס שהתגלה, וזיהויו באופן סטטי
  - או קבצי הרצה באמצעות חתימות, וקבצי מערכת באמצעות white list

אתגרים חדשים

- קיימות טכניקות לזיהוי וירטואליזציה בפוגענים Anti Virtualization.
- מתייחס לכלל מוצרי הוירטואליזציה הקיימים היום בשוק: Sandbox, VirtualBox , VMware ועוד..
- נועד להקשות על החוקרים לבצע תהליך של הנדסה אחורית שתפקידה לנסות להבין כיצד הפוגען עובד מאחורי הקלעים, וכיצד ניתן לנטרל אותו.
- כתוצאה מהתקדמות של טכנולוגיה זו, וירוסים ופוגענים צריכים לדעת להתמודד עם המצב החדש, ולכן פיתחו לעצמם כל מיני טכניקות (ידועות יותר, וידועות פחות) עקיפה שונות.

משחק מוחות - טכניקות נגד (אנטי חקירה\הנדסה לאחור) שקיימות בפוגענים

- טכניקת לזיהוי מספר מעבדים
- טכניקת Magic Number - ישנו מספר שנקרא Number Magic המייצג ערך לזיהוי עבור פורט מסוים
- זיהוי באמצעות חיפוש ערכים קיימים בRegistry (ערכים ברגיסטרים)
- טכניקת בדיקת הנתיב של הקובץ
- טכניקת זיהוי כתובות MAC קבועות
- זיהוי –Process-ים של מכונות וירטואלית
- טכניקת זיהוי ה–DLL-ים הנטענים לזיכרון
- טכניקת זיהוי פיצ'ר Folder Shared
- טכניקת גודל הדיסק הקשיח
- טכניקות לזיהוי —Sandbox-ים
- טכניקת "Anti-Evasion-Evasion Trick"
- ...