



## TEE – DIGITAL VAULT

### תקציר

מסמך אפיון – פרויקט גמר בקורס  
"מערכות להרצת קוד בסביבה בטוחה".

ליאב אריאל -212830871-

עמיעד קורמן -212608442-

מרצה: ברק עינב.

## תוכן עניינים

2	מושגי יסוד (תיאוריה)	
2	עקרונות TEE	←
3	עקרונות אבטחה במערכת הפעלה ראשית	←
3	הבעיה הבסיסית – מדוע מכשירי מחשוב אינם אמינים	←
4	מישורי תקיפה כיום	←
4	מישורי תקיפה במערכת TEE	←
5	תרשים מערכת TEE	←
6	מה הפרויקט ?	
6	מה השוק ?	←
6	ייחודיות הרעיון שלנו	←
7	HARDWARE VS. SOFTWARE	
7	יתרונות בסיסיים של כספת מקומית (לעומת תוכנה)	←
7	חולשות כלליות של כספת מקומית (לעומת תוכנה)	←
8	סקירה כללית של הפרויקט	
8	Context Diagram	←
9	UML - הצעה ראשונית	←
10	מימוש	
10	UML סופי	←
11	הסבר על 3 השכבות	←
11	גרסא ראשונה מעוצבת	←
12	ניתוח אבטחה	
12	זיהוי התוקף	←
12	CIA	←
13	מישורי תקיפה	←
14	GIT	←

# מושגי יסוד (תיאוריה)

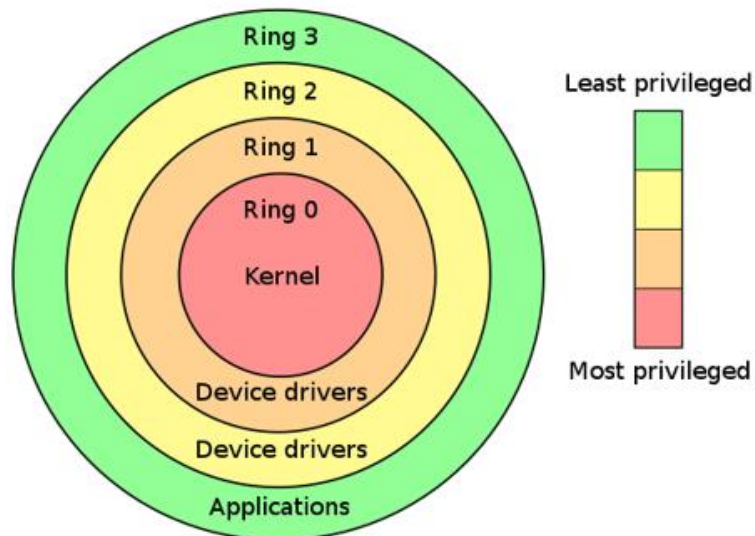
## הגדרה:

סביבת ביצוע מהימנה (TEE) היא אזור מאובטח של מעבד ראשי (בזה נעסוק בפרויקט המעשי). זה מבטיח שהקוד והנתונים הטעונים בפנים יהיו מוגנים ביחס לסודיות (Confidentiality) ושלמות (Integrity). TEE כסביבת ביצוע מבודדת מספק תכונות אבטחה כגון ביצוע מבודד, שלמות יישומים הפועלים עם ה-TEE, יחד עם סודיות הנכסים שלהם.

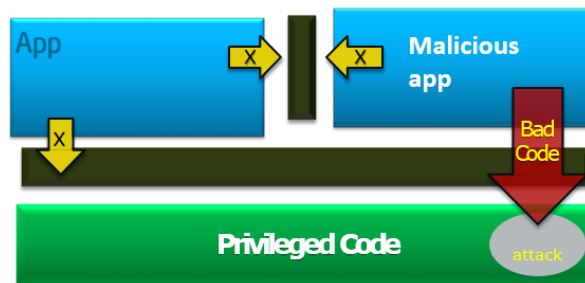
## עקרונות של TEE:

- ⇐ הפרדת מערכות. (מעבד, זיכרון, אכסון, קריפטוגרפיה).
- ⇐ חומרה ייעודית.
- ⇐ מערכת שבנויה מראש עבור security.
- ⇐ מערכת סגורה (או סגורה חלקית).
- ⇐ ערוץ תקשורת מוגדר היטב בין TEE ל-REE.

## עקרונות אבטחה במערכת הפעלה ראשית:



## הבעיה הבסיסית: מדוע מכשירי מחשב אינם אמינים?



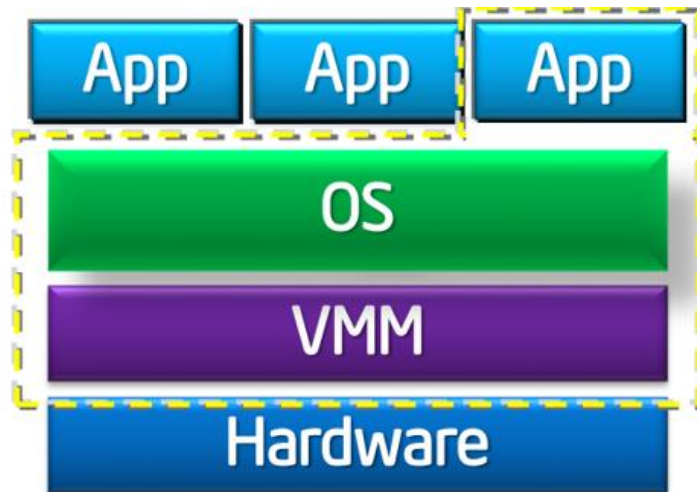
...and apps from each other ...


...UNTIL a malicious app exploits a flaw to gainfull privileges and then tampers with the OS or other apps

**Apps not protected from privileged code attacks**

Frank McKeen Intel Labs  
April 15, 2015

## מישורי תקיפה כיום:



Attack Surface 

## מישורי התקיפה במערכת TEE:

Application gains ability to defend its own secrets

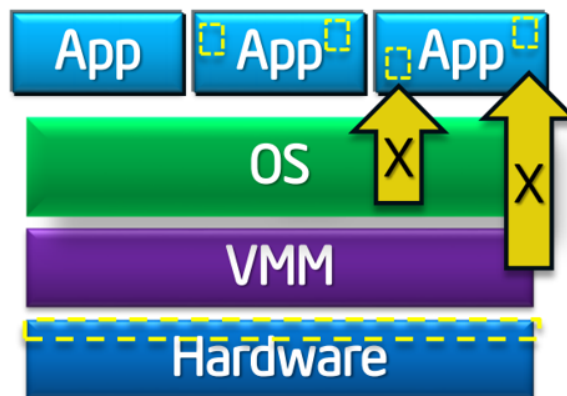
- Smallest attack surface (App + processor)
- Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets


Familiar development/debug

- Single application environment
- Build on existing ecosystem expertise

Familiar deployment model

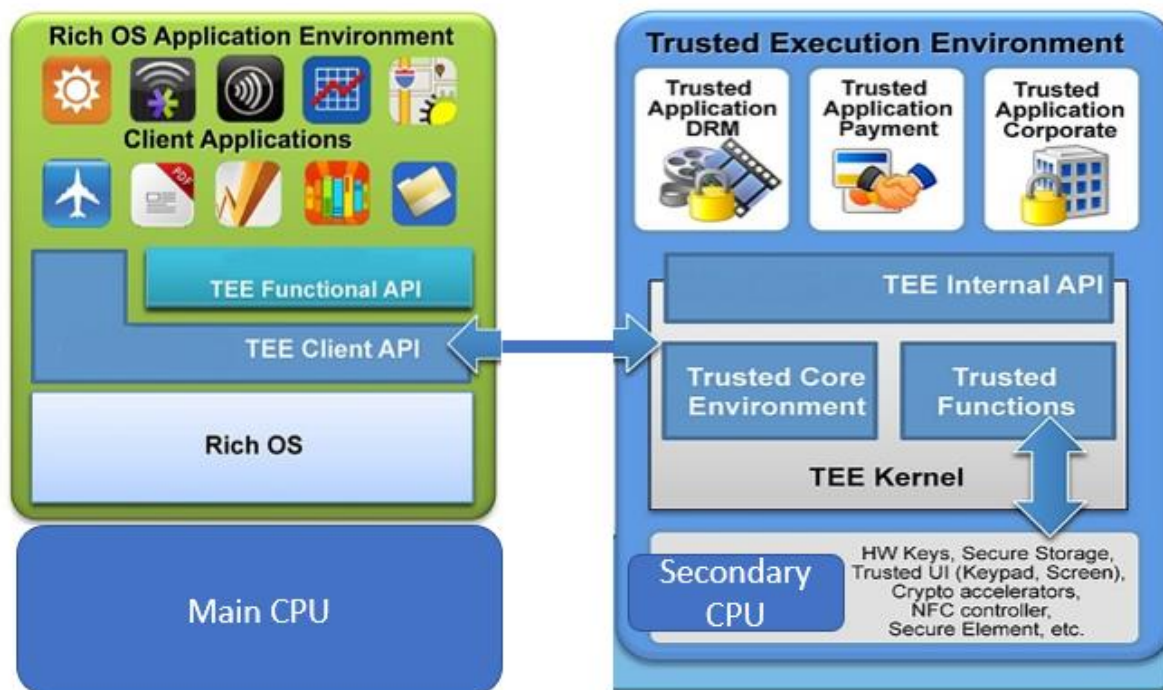
- Platform integration not a bottleneck to deployment of trusted apps



Attack Surface 

מספר מישורי התקיפה צומצם משמעותית.

## תרשים מערכת TEE המשתמשת במערכת נפרדת:



\*זו המערכת שנשתמש בה בפרויקט זה.

## מה הפרויקט ?

מבלי לצלול עמוק בפנים לארכיטקטורה ומבנה הפרויקט, אלא ממבט של לקוח - המוצר הסופי אותו נרצה להשיג הוא אפליקציה / תוסף לדפדפן המאשר שמירת מידע אישי וחשוב, כגון סיסמאות או כל דבר אחר בזיכרון הפיזי של המחשב עם אקסטרה הגנה. מדובר על כספת סיסמאות TEE, עם תוסף לגוגל כרום. בנוסף, נציע פיצ'ר ליצירת סיסמאות חזקות באופן רנדומלי. (בעצם התוכנה תנהל לנו את כל הסיסמאות מבלי שנצטרך לזכור אותם ובצורה מאובטחת).

### מה השוק?

ישנן תוכנות רבות ושונות המאכסנות תכניות של משתמש, סיסמאות (למשל לאתרים מסויימים), מידע אישי, וכו'...  
לרוב הם מאכסנים אותו בענן, שתיאורטית נגיש מכל מקום.

### הרעיון שלנו – כספת סיסמאות מקומית ב-TEE:

במקום לאכסן את הנתונים הרגישים האלה בענן/תוכנה, יש לנו פתרון חומרה – נאחסן סיסמאות ב(TEE (Trusted Execution Environment. אפליקציה HOST במחשב יכולה לתקשר עם הTEE, לפתוח WebSocket (with local host שינהל את השיחה מול תוסף הגוגל chrome, ויאפשר למשתמש לשמור בקלות סיסמאות לפי url. בעצם האפליקציה כבר תדע להציע את הסיסמה והשם משתמש הנכונים עבור אתר ספציפי, תוך כדי התנהלות מול הTEE בלבד המוגן מפני פרצות של מערכת ההפעלה (למערכת ההפעלה אין גישה אליו). התקשורת בין TEE לתוסף האינטרנט תצטרך הצפנה א-סימטרית RSA, זה בעצם האזור היחיד שצריך הגנה נוספת מפני שלולא ההצפנה תוקף זדוני יוכל להאזין לWebSocket ולגלות את הסיסמה. מה השגנו? שימוש בטכנולוגיית TEE, יאפשר לנו גישה מהירה ובטוחה לדברים שנרצה לשמור – זה היתרון שלנו על מוצרים אחרים בשוק השומרים את הכל בענן ייעודי שלהם.

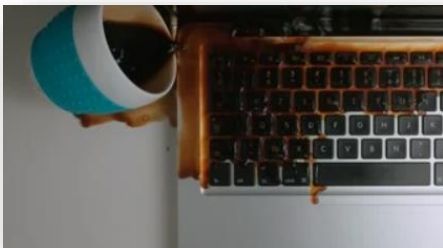
## HARDWARE VS. SOFTWARE

### יתרונות כלליים של כספת סימסאות מקומית (לעומת תוכנה):

- ✓ באופן סובייקטיבי, תמיד נכון לומר שחומרה מאובטחת יותר מתוכנה.
- ✓ בפתרון חומרתי, יש פחות באגים ודלתות אחוריות (backdoors).
- ✓ פחות משטחי תקיפה (התוקף חייב לרוץ במכונה המקומית ולהאזין לWebSocket).
- ✓ היישום והניהול הרבה יותר פשוטים.

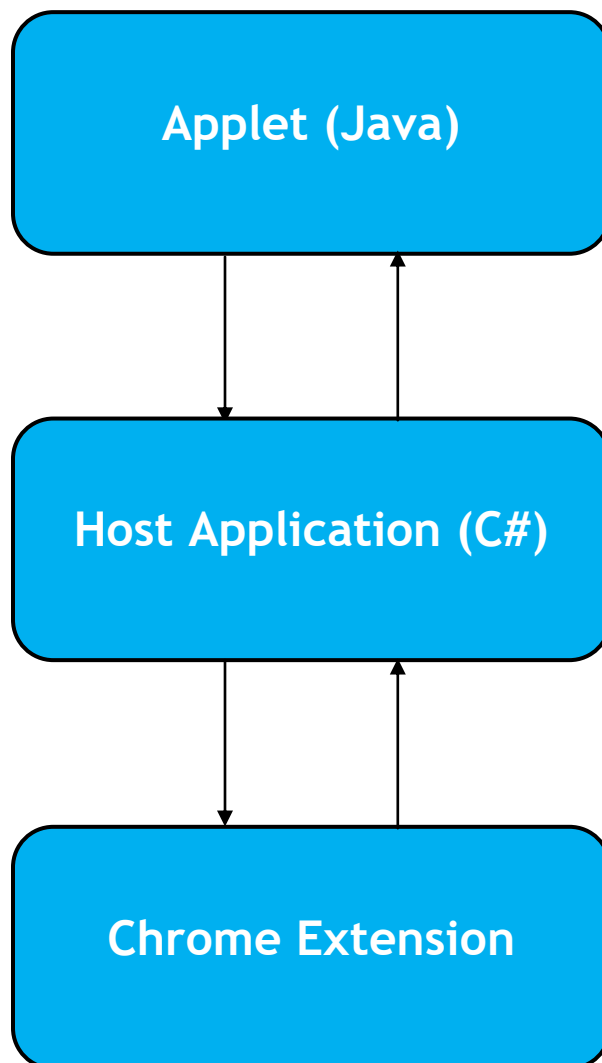
### חולשות כלליות של כספת מקומית (לעומת תוכנה):

- ✗ לא נגיש אם המשתמש לא בקרבת המחשב האישי. אמנם השגנו שלימות וסודיות אבל איבדנו זמינות.
- ✗ לא ניתן לנייד בקלות למחשב אחר (אפילו אם מדובר במעבר באופן קבוע) – צריך פיזית להתקין את הרכיב במחשב החדש (ומדובר ברכיב פנימי במחשב, ולא בדיסק-און-קי). גישה אליו מותנת בצידוד יקר ובתנאי מעבדה.
- ✗ לא ניתן להרחבה (מבחינת גודל הזיכרון).
- ✗ אם משהו קורה לחומרה, המידע שעליו פשוט נאבד. (אנחנו לא מגבים אותו באף מקום, כי אם נעשה זאת המוצר שלנו לא יהיה שונה מכל מוצר אחר בשוק המגבה את המידע בענן). אך הסבירות שמשהו יקרה לרכיב נמוכה מאוד, כמובן תמיד יש את הגורם האנושי 😊

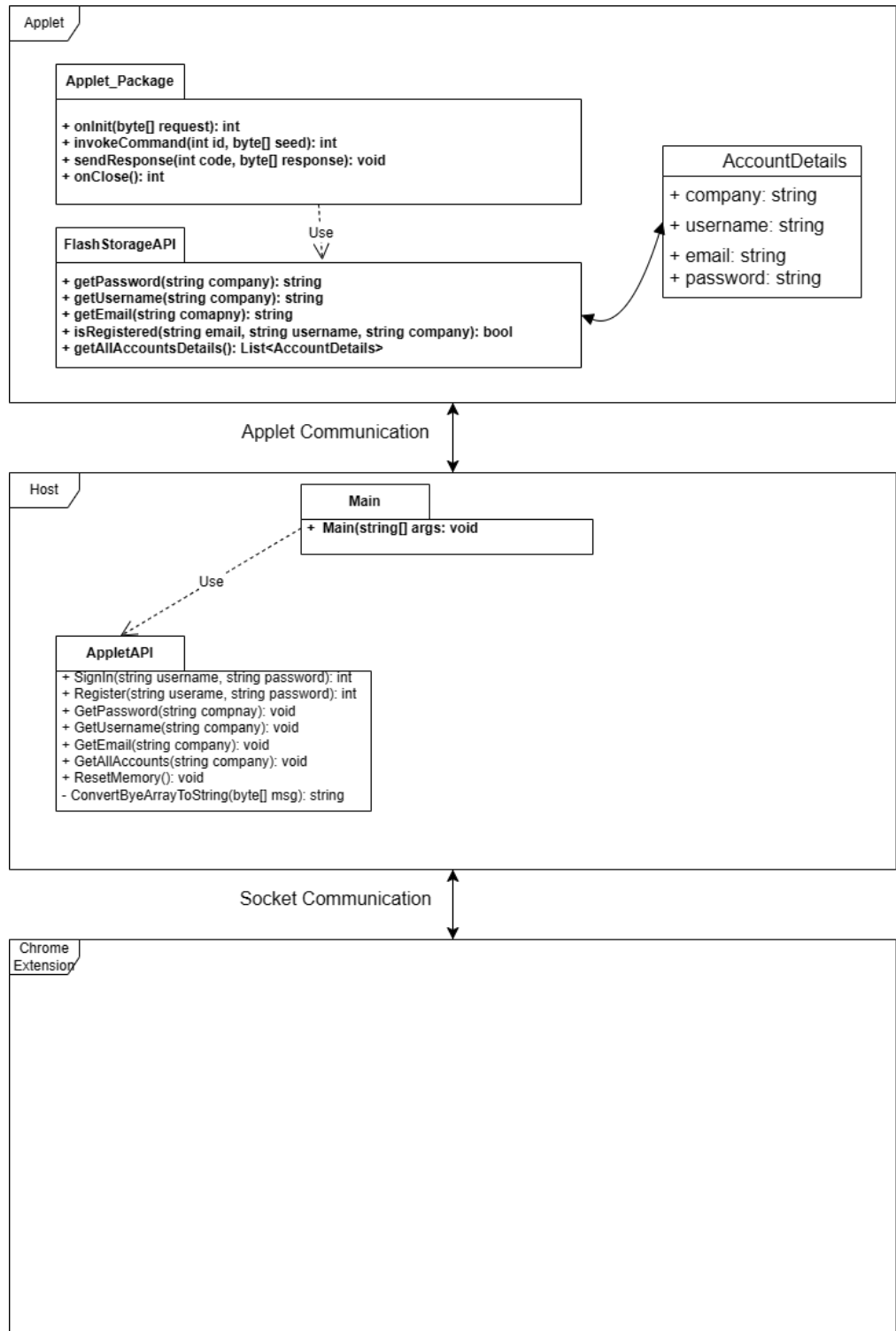




:Context Diagram

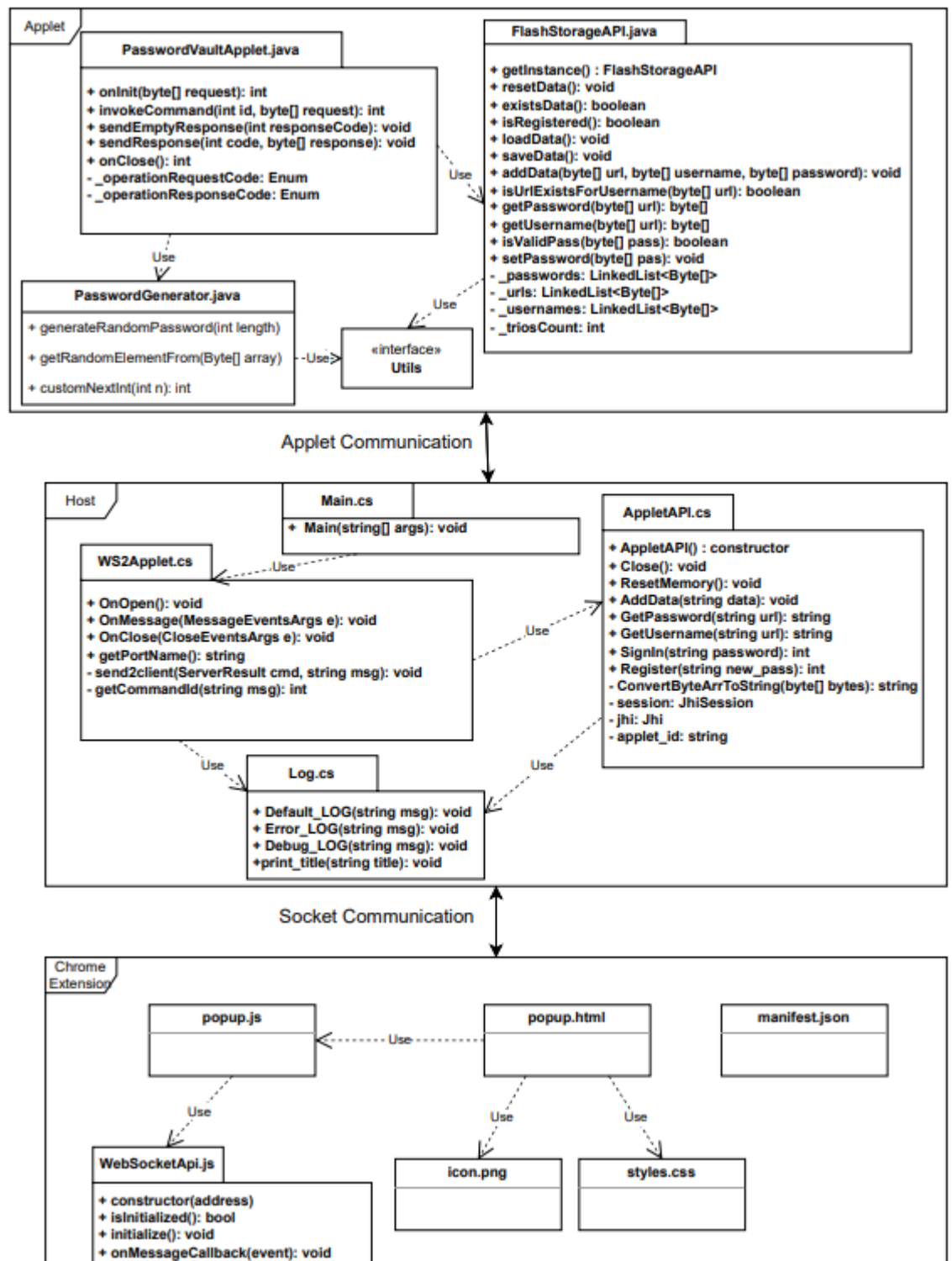


## UML – הצעה ראשונית:



# מימוש

## UML – סופי:



## הסבר על 3 השכבות:

1. **Applet** – השכבה האחראית על התקשורת מול Intel Dal. נכתב בjava ורץ על סביבה מהימנה (TEE). מנהלת זיכרון משלה (FlashStorage). זו השכבה ששומרת הדאטה שלנו בצורה מאובטחת. ומספקת לנו אותו כשאנו צריכים.

2. **Host** – שכבת הביניים, מהווה כמגשר בין chrome extension לApplet. כל תפקידה, לקבל את הבקשה מהתוסף, לעבד אותה, ליצור בקשה מתאימה לApplet אשר יספק את המידע הדרוש / יעדכן את המידע הרלוונטי. ולבסוף שליחת תגובה מתאימה בחזרה לchrome extension. השכבה בנוסף, אחראית על הרצת console log (כמתואר בצד שמאל למטה). נכתב בC#, רץ על אמולטור של Intel Dal (סביבה מהימנה).

3. **Chrome Extension** – ה front end של הפרויקט, יודעת לנהל event שמתבצעים בתוסף, לייצר הודעת request מתאימה לפי פרוקטול שהגדרתי, לשלוח לHost לקבל תגובה לעבד אותה ולעדכן את הgui באופן דינמי.

## גרסה ראשונה מעוצבת:

The image displays a web application interface for a login system, overlaid on a server console log. The web interface, titled 'facebook.com', features a login form with fields for 'URL:', 'USERNAME:', 'PASSWORD:', and 'MESSAGE:'. Below these fields are buttons for 'Set Master Key', 'Sign In', 'Get Password', 'Get Username', 'Add Data', and 'Reset Memory'. A 'Log In' button is prominently displayed in blue, with a 'Forgot password?' link below it. At the bottom, there is a 'Create new account' button in green. The server console log on the left shows the following sequence of events:

```
*****
SERVER CONSOLE LOG
*****
SYS_LOG: Enter q! to stop
SYS_LOG: WS server started on ws://127.0.0.1:5789/WS2Applet
SYS_LOG: AppletApi > Installing the applet.
SYS_LOG: AppletApi > Opening a session.
SYS_LOG: Client opened a socket.
SYS_LOG: ==> Message sent to client: RES_SUCCESS Successfully connected to Server!
SYS_LOG: <== Received message from Echo client: SIGN_IN 1234
DEBUG_LOG: WS2Applet on 'SIGN_IN' operation.
DEBUG_LOG: AppletApi on 'Signin' operation.
SYS_LOG: User successfully signed in.
SYS_LOG: ==> Message sent to client: RES_SUCCESS Successfully signed in.
SYS_LOG: <== Received message from Echo client: GET_USERNAME facebook.com
DEBUG_LOG: WS2Applet on 'GET_USERNAME' operation.
DEBUG_LOG: AppletApi on 'GetUsername' operation.
SYS_LOG: Username retrieved.
SYS_LOG: ==> Message sent to client: RES_USERNAME_RETRIEVED liavha00@walla.com
SYS_LOG: <== Received message from Echo client: GET_PASSWORD facebook.com
DEBUG_LOG: WS2Applet on 'GET_PASSWORD' operation.
DEBUG_LOG: AppletApi on 'GetPassword' operation.
SYS_LOG: Password retrieved.
SYS_LOG: ==> Message sent to client: RES_PASSWORD_RETRIEVED Aa123456
SYS_LOG: AppletApi > Closing the session.
SYS_LOG: AppletApi > Uninstalling the applet.
SYS_LOG: AppletApi > Installing the applet.
SYS_LOG: AppletApi > Opening a session.
SYS_LOG: Client opened a socket.
SYS_LOG: ==> Message sent to client: RES_SUCCESS Successfully connected to Server!
SYS_LOG: <== Received message from Echo client: SIGN_IN 1234
DEBUG_LOG: WS2Applet on 'SIGN_IN' operation.
DEBUG_LOG: AppletApi on 'Signin' operation.
SYS_LOG: User successfully signed in.
SYS_LOG: ==> Message sent to client: RES_SUCCESS Successfully signed in.
SYS_LOG: <== Received message from Echo client: GET_USERNAME facebook.com
DEBUG_LOG: WS2Applet on 'GET_USERNAME' operation.
DEBUG_LOG: AppletApi on 'GetUsername' operation.
SYS_LOG: Username retrieved.
SYS_LOG: ==> Message sent to client: RES_USERNAME_RETRIEVED liavha00@walla.com
SYS_LOG: <== Received message from Echo client: GET_PASSWORD facebook.com
DEBUG_LOG: WS2Applet on 'GET_PASSWORD' operation.
DEBUG_LOG: AppletApi on 'GetPassword' operation.
SYS_LOG: Password retrieved.
SYS_LOG: ==> Message sent to client: RES_PASSWORD_RETRIEVED Aa123456
```

### זיהוי התוקף:

כל מי שירצה להשיג את הנתונים שבפנים / להרוס את הנתונים שבפנים.

### CIA – חשיבת התוקף ואיך המוצר מוגן מפני זה:

#### ⇐ סודיות (Confidentiality):

- תוקף ירצה לגשת למידע הסודי (סיסמאות) ולגלות אותם.
- **המידע מאובטח ב TEE ומאובטח על ידי הצפנת ה-WebSocket.**

#### ⇐ שלימות הנתונים (Integrity):

- סביר להניח שלתוקף לא יהיה איכפת לשנות את הנתונים, כי לפי הגדרת התוקף מניע של "להרוס את הנתונים" רלוונטי עבורו על מנת לגרום לכספת לא לעבוד.
- יתכן שהתוקף ירצה ליצור סיסמאות משלו ולספק אותם למשתמש.
- **על מנת להשיג את 2 המטרות הנל, יצטרך התוקף להצליח לרוץ על מכונה מקומית. להתממשק ל IntelDal, לגלות את ה coden עבור datan של הכספת סיסמאות ולכתוב אליה. אם התוקף הצליח לרוץ על מכונה מקומית: נזק אחר כבר נעשה. מערכת ההפעלה יודעת למנוע את זה.**

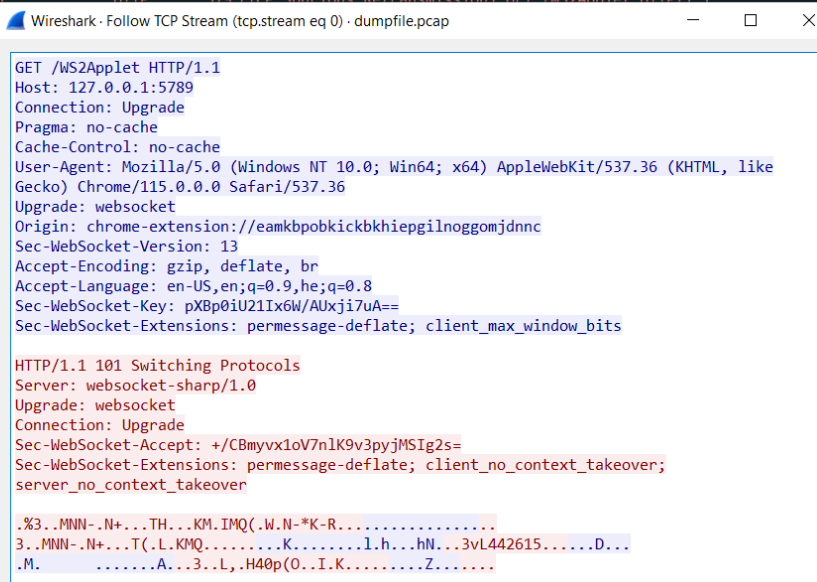
#### ⇐ זמינות (Availability):

- תוקף יכול לחשוב על מתקפת כופר, נעילת הסיסמאות תמורת כסף.
- **על מנת לעשות את זה, התוקף צריך גישה למכונה המקומית ולפרוסס של Host ולנעול אותו. שוב, גישה למכונה מקומית לא דבר שמתאפשר בקלות ומערכת ההפעלה עצמה מונעת את זה.**

## מישורי תקיפה:

← **קלט משתמש:** אם יש שיבוש בקלט, הפרוטוקול שיחה משתבש גם, והמערכת יודעת לזהות את זה ולהגיב בהתאם.

← **הסנפוט:** מתבצעת הצפנה של הWebSocket מקצה לקצה.  
תצלום של הסנפה:

1	0.000000	127.0.0.1	127.0.0.1	TCP	52 5789 → 55585 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
2	0.006005	127.0.0.1	127.0.0.1	TCP	52 55585 → 5789 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
3	0.006005	127.0.0.1	127.0.0.1	TCP	40 55585 → 5789 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
4	0.007003	127.0.0.1	127.0.0.1	TCP	40 [TCP ACKed unseen segment] 5789 → 55585 [ACK] Seq=1 Ack=540 Win=2619648 Len=0
5	0.007003	127.0.0.1	127.0.0.1	HTTP	579 [TCP Spurious Retransmission] GET /WS2Applet HTTP/1.1
58	0.036581	127.0.0.1	127.0.0.1	 Wireshark · Follow TCP Stream (tcp.stream eq 0) · dumpfile.pcap GET /WS2Applet HTTP/1.1 Host: 127.0.0.1:5789 Connection: Upgrade Pragma: no-cache Cache-Control: no-cache User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.0.0 Safari/537.36 Upgrade: websocket Origin: chrome-extension://eamkbpobkickbkhiiepgilnoggomjdnn Sec-WebSocket-Version: 13 Accept-Encoding: gzip, deflate, br Accept-Language: en-US,en;q=0.9,he;q=0.8 Sec-WebSocket-Key: pXBp0iU21Ix6W/AUXji7uA== Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits  HTTP/1.1 101 Switching Protocols Server: websocket-sharp/1.0 Upgrade: websocket Connection: Upgrade Sec-WebSocket-Accept: +/CBmyv1oV7nIK9v3pyjMSIg2s= Sec-WebSocket-Extensions: permessage-deflate; client_no_context_takeover; server_no_context_takeover  .%3..MNN-.N+...TH...KM.IMQ(.W.N-*K-R..... 3..MNN-.N+...T(.L.KMQ.....K.....l.h...hN...3vL442615.....D... .M. ....A...3..L..H40p(O..I.K.....Z.....	
59	0.036581	127.0.0.1	127.0.0.1		
60	0.036581	127.0.0.1	127.0.0.1		
61	0.036581	127.0.0.1	127.0.0.1		
62	5.881350	127.0.0.1	127.0.0.1		
63	5.881722	127.0.0.1	127.0.0.1		
72	5.912013	127.0.0.1	127.0.0.1		
73	5.912013	127.0.0.1	127.0.0.1		
74	9.789447	127.0.0.1	127.0.0.1		
75	9.789447	127.0.0.1	127.0.0.1		
84	9.799447	127.0.0.1	127.0.0.1		
85	9.799447	127.0.0.1	127.0.0.1		
86	10.964445	127.0.0.1	127.0.0.1		

Frame 61: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface 0  
Ethernet II, Src: Intel(R) Ethernet Adapter, Dst: 127.0.0.1, Protocol: TCP  
Transmission Control Protocol, Src Port: 5789, Dst Port: 55585, Seq: 5789, Len: 79  
Hypertext Transfer Protocol, Content-Type: text/html, Content-Length: 101  
WebSocket

Frame 86: 10.964445 seconds on wire (87.755556 Kbytes), 10.964445 seconds captured (87.755556 Kbytes) on interface 0  
Ethernet II, Src: Intel(R) Ethernet Adapter, Dst: 127.0.0.1, Protocol: TCP  
Transmission Control Protocol, Src Port: 5789, Dst Port: 55585, Seq: 5789, Len: 79  
Hypertext Transfer Protocol, Content-Type: text/html, Content-Length: 101  
WebSocket

← **תקיפות חומרה:** החומרה שאנו משתמשים בה היא intel dal.

החומרה מוגנת היטב מפריצות, ועל מנת לדלות ממנה מידע צריך תנאי מעבדה מאוד יקרים.

## GIT

**קישור לפרויקט:**

[https://github.com/liav2002/TEE5873\\_8442\\_0871/tree/FinalProject](https://github.com/liav2002/TEE5873_8442_0871/tree/FinalProject)