Technologic Papers

FIDO2 - עולם ללא סיסמאות?

מאת אלון בר סלע ויהונתן אלקבס

הקדמה

FIDO Alliance (קיצור של Fast IDentity Online) היא קבוצת עבודה שהוקמה לצורך התמודדות עם אחת מחולשות אבטחת המידע הפגיעות ביותר באינטרנט ובכלל - **שימוש בסיסמאות**.

FIDO מורכבת מנציגים של כ-250 ארגונים שכוללים חברות טכנולוגיות (Google, Amazon ,Microsoft, Apple, ARM, META), חברות אבטחת מידע (Ping Identity, RSA, Akamai), ספקי שירותים (Pay-Pal, American Express, VISA) וגופים ממשלתיים (NIST, German Information Security Office, UK Cabinet Office). לכולם מטרה אחת ברורה - להפוך את האינטרנט למקום בטוח יותר.

FIDO2 היא מכלול תקנים שהוגדרו ע"י התאגדות FIDO במטרה לאפשר למשתמש הקצה הזדהות מהירה מול שירותים שונים באינטרנט ללא התבססות על סיסמאות (passwordless); אלא ע"י שימוש בהזדהות אקטיבית של המשתמש, בניית מפתחות א-סימטריים ייחודיים לכל שירות והגדרת פרוטוקול הזדהות בין השרת ודפדפן הלקוח ובין דפדפן הלקוח למכשיר האימות. נשמע מבטיח? בשנתיים האחרונות התרחבה מאוד התמיכה ב-FIDO2 למאות אתרי אינטרנט ושירותים שמנהלים את חיי היומיום שלנו - Gmail, Yahoo mail, Dropbox, Pay-Pal, Facebook, Twitter, Coinbase, Microsoft Azure, AWS GCP, Salesforce, GitHub, NordVPN, Tesla, Office 365 ועוד.

תמיכה ב-FIDO2 מאפשרת למשתמש להגדיר כי אופן ההזדהות לאתר או לשירות יתבצע באמצעות "חומרה מזהה" (Authenticator בלעז) אשר כוללת סט מכשירים כגון טלפון חכם (זיהוי פנים, טביעת אצבע או PIN), מחשב אישי (דוגמת Windows Hello) או חומרה ייעודית (Yubikey ,Bio-Key). המפתחות הפרטיים לא עוזבים את ה-Authenticator ועל מנת להפעיל אותו (לצורך חתימה על chellenge שמתקבל מהשרת) נדרשת התערבות פיזית של המשתמש וחיבור ישיר או תקשורת קצרת טווח בין החומרה המזהה והפלטפורמה (מחשב או טלפון נייד) ממנה ניגשים לשירות.

המאמר הקרוב הוא חלקה הראשון של סדרה בת 2 מאמרים אשר שם דגש על הסברים כלליים ומהווה הכנה טובה עבור המאמר השני אשר מיועד להיות משמעותית יותר hands-on עבור אלו שרוצים להבין כיצד האימות מתרחש "באמת" - איזה מידע עובר, כיצד הוא נראה וכיצד הוא ת'כלס מונע מתקפות.

קצת היסטוריה ועוד פרטים לא ממש חשובים לפני שמתחילים

התאגדות FIDO נוסדה בשנת 2012 ע"י חברות Pay-Pal, Lenovo וחברות נוספות. בשנת 2013 התרחבה ההתאגדות עם צירופן של Google ו-NXP Semiconductor.

בשנת 2014 פרסמה FIDO שני תקנים משמעותיים (שעדיין נתמכים באמצעי הגנה רבים):

1. **FIDO U2F** - Universal 2nd Factor מהווה תקן פתוח שנועד להסדיר את הצד הקריפטוגרפי למימוש אימות מבוסס מפתחות א-סימטריים, אופן המימוש של חומרת ה-2FA (הגדרת הממשקים הפיזיים דוגמת Bluetooth, USB או NFC, אופן השמירה על המפתחות) וקישור חומרת ה-2FA לדפדפן (ללא צורך בהתקנת יישום צד או דרייבר נוסף). FIDO U2F נחשב לתקן מוצלח ועזר להתפתחות שוק "חומרות ההזדהות" (Authenticators). ולמרות הנחיתות שלו ביחס למוצרי FIDO2 ניתן עדיין למצוא מגוון חומרות התומכות ב-U2F.
2. **FIDO UAF** - Universal Authentication Framework הינה מסגרת עבודה (שזה די דומה לתקן, אבל בלי להגדיר בדיוק את אופן המימוש) לאימות ללא סיסמא בין שרת ולקוח. המסגרת מגדירה כיצד מקשרים בין אמצעי הזיהוי לאתר האינטרנט, באילו שיטות הזדהות ניתן להתשמש, שימוש במפתחות א-סימטריים ועוד.

בשנת 2015 מייקרסופט הכריזה על תמיכה באימות מבוסס FIDO ב-WINDOWS 10. בנוסף, נוספו להתאגדות נציגי של ממשלות ארה"ב, גרמניה, אוסטרליה, סין ואנגליה.

בשנת 2016 ארגון ה-World Wide Web Consorium (W3C) הכריז על תמיכה בתקן הזדהות רשתית שמבוססת על ההצעה שהוגשה ע"י איגוד FIDO. בכך, החל שיתוף פעולה בין FIDO וה-W3C שבמסגרתה יוגדר תקן FIDO2.

בשנת 2017 הצטרפה facebook להתאחדות FIDO והחלה לאפשר למשתמשים להזדהות לשירותי facebook באמצעות FIDO 2FA.

בשנת 2018 תקן FIDO2 הבשיל לכדי מימוש ודפדפנים מובילים (Chrome, Firefox, Edge) הכריזו על תמיכה בתקן בדפדפנים שלהם. בנוסף, אנדרואיד הכריזה על תמיכה ושילוב התקן במערכת ההפעלה.

בשנת 2019 חלקים מתקן FID02 הוכרזו כסטנדרטים מוכרים ע"י איגוד התקשורת העולמי (ITU-T).

נעצור כאן, נראה שהתעמקנו מספיק בהיסטוריה של התאגדות FIDO, מה שחשוב שתזכרו מכל הפרק הנ"ל הוא שתקן FIDO2 מוכר ע"י תעשיות וממשלות מובילות, ואולי יום אחד (סביב 2050 לפי הקצב הנוכחי) תוכלו להזדהות עם FIDO2 גם מול שירותי ה-online של אחד הבנקים בישראל.

שיטות אימות 101

ישנן 3 דרכים לבצע אימות לזהות משתמש דיגיטלי או בשפה פשוטה יותר - דרכים להוכיח שאני מי שאני טוען שאני:

- באמצעות "משהו שאני יודע" (סוד) - סיסמה, שם של המורה המועדפת מכיתה ג'
- באמצעות "משהו שיש לי" - token (אֲסִימוֹן בעברית צחה) פיזי, טלפון חכם
- באמצעות "משהו שאני" - טביעת אצבע, רשתית עין, תווי פנים

אנחנו מקווים שזה ברור לכולם כי "משהו שאני יודע" היא כנראה האופציה הכי חלשה מבחינת אבטחת מידע. **אז למה בכל זאת בחרו בה?** התשובה הקצרה - כי היא הכי פשוטה למימוש. התשובה הארוכה - ובכן, דמיינו את האינטרנט של שנות ה-90 (אם אפשר בכלל לקרוא לזה אינטרנט לעומת מה שיש היום), מי בכלל חשב על זה שאדם צלול בנפשו ינסה להעביר סכום המקביל למיליון דולר במטבעות קריפטוגרפיים לצידו השני של הגלובוס?? במילים פשוטות - העברה של תמונות חתולים היו השאיפה הכי גדולה בזמנו, בטוח שלא צריך להמציא משהו פֶּנְסִי כמו פרוטוקול לאימות טביעת אצבע בשביל כך. שם משתמש "coolkid1978" וסיסמה "123456" הם די ויותר בשביל להתחבר לפורום הַחֲתוּלִיִים.

אוקיי אז הבנו ש-"משהו שאני יודע" היא כנראה החולייה החלשה וכנראה ש-"משהו שאני" הוא אמצעי האימות החזק ביותר מכיוון שלא ניתן לשכפל\ לגנוב אותו מבלי ש'קצת' תשימו לב לכך. אבל המחשבה הזו היא לא הכי אינטואיטיבית ברגע שמבינים כיצד מנגנוני אימות עובדים.

אבטחה ביומטרית הופכת במהירות לדרך המועדפת להגן בחברות ענק מפני גורמים עוינים שמטרתם הונאה וגניבת זהות. קוראי טביעות אצבע, סריקות רשתית העין וזיהוי פנים הפכו למיינסטרים, ובראשם חברות טכנולוגיה דוגמת Microsoft ו-Apple. פרטי האימות (סיסמה, אסימון, טביעת אצבע וכו') לרוב ישלחו וישמרו בצד שרת לאחר הפיכתם למידע דיגיטאלי והפעלת פונקציית גיבוב כזו או אחרת (משפחות MD, SHA). בעת ניסיון ההתחברות צד ה-client יריץ את פונקצית הגיבוב שוב ותתבצע השוואת string פשוטה בין זו שהתקבלה לזו שנשמרה בעבר ב-DB השרת. אבל מה יקרה אם השרת ישמור את פרטי ההזדהות שלנו כמו שהם ללא גיבוב?

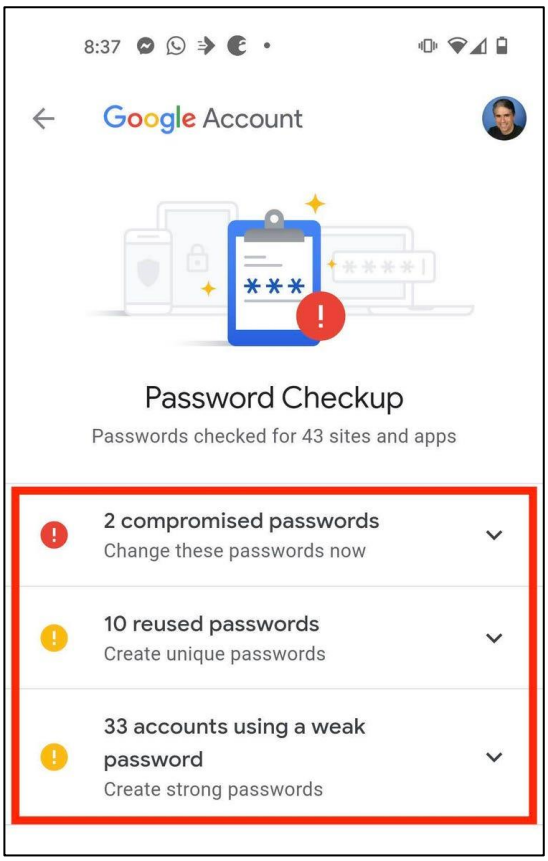
במידה ומדובר בסיסמה הרי שהיא תהיה כמו שהזנו אותה ב-cleat text אבל מה **אם אופן ההזדהות שלנו היה באמצעות מידע ביומטרי כמו טביעת אצבע?** כיצד הוא ישמר? ובכן, באמצעות פורמט fingerprint template כזה או אחר. הסיכון הכי גדול במקרה כזה הוא שללא פונקציית הגיבוב יהיה ניתן לשחזר את נתוני טביעת האצבע גם ללא האצבע הפיזית. משמעות הדבר היא שבמידה ותוקף פרץ ל-DB המכיל את פורמט טביעת האצבע שלכם **מעתה ועד סוף חייכם הוא יוכל להתחקות אחר הזהות שלכם**. גניבת זהות ביומטרית היא דבר מפחיד.

כמובן שיש אמצעים להגן על שמירת המידע הביומטרי שלכם במאגר בצורה בטוחה אבל פרצות כאלו מתגלות לצערנו לא מעט. במחקר שבוצע לפני 3 שנים על ידי צמד מוצלח ביותר של חוקרים ישראליים (נעם רותם ורן לוקאר) נמצא כי 2 מאגרי מידע של חברת Biostar היו נגישים לאינדוקס באמצעות שאילתות Elasticsearch שבאמצעותן הוציאו 27.8 מיליון רשומות ו-23GB של מידע בנוגע טביעות אצבע, נתוני זיהוי פנים, תמונות פנים של משתמשים, שמות משתמש וסיסמאות לא מוצפנים. שיטוט קצר בגוגל מספק עשרות מקרים דומים ושיטוט קצר אף יותר בכל פורום מבוסס שירות onion מספק פלטפורמה נוחה לקניית אותו מידע.

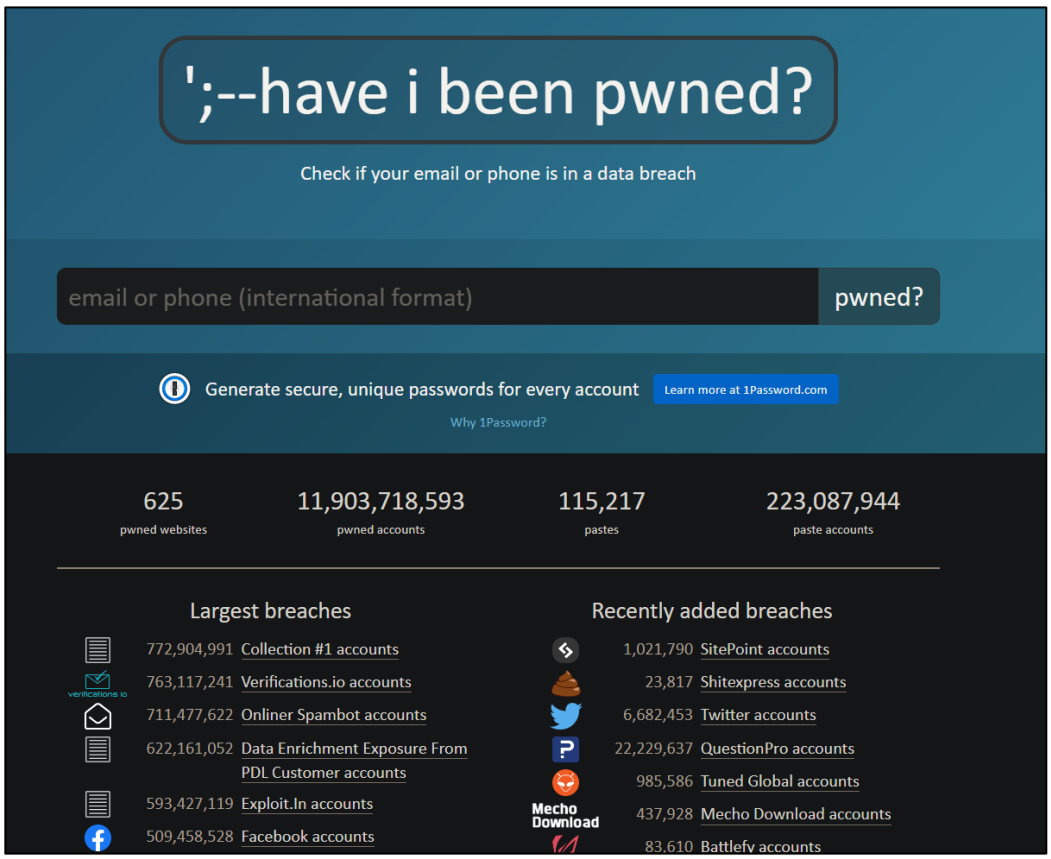
לסיכום, הזדהות ביומטרית היא ייחודית ויוצרת מפתח מורכב לפיצוח, אך פריצה לשירות בודד אשר לא קונפג כראוי עתיד להוות חולשה פוטנציאלית מול כלל השירותים מולם הזדהנו באמצעות המידע הביומטרי.

פתרון קבוע או פלסטר?

בשנים האחרונות הגיחו מספר פרויקטים גדולים, תוספים לדפדפן וסטארטפים; כל אחד עם הדרך המקורית שלו לנסות להתמודד עם הבעיה של שימוש בסיסמאות. Google Password Checkup אשר מתריע בפני משתמשיו מסיסמאות שנחשפו, סיסמאות חלשות וסיסמאות בשימוש חוזר הוא דוגמה פנטסטית לכיצד ענקית הטכנולוגיה מנסה להיאבק בחוסר מזלם (ופזיזותם) של משתמשיה:



שירות שונה אך עם כוונה זהה מוצג באתר haveibeenpwned.com:



תוספות מיוחדות לדפדפן (כמו RoboForm) וגם פתרונות זהים לסביבה הלוקאלית (כמו 1Password) פותחו, כל ייעודן הוא להחזיק מסד נתונים מוצפן לכל הסיסמאות שברשות המשתמש הפשוט על מנת שזה יוכל לשמור על סיסמאותיו בצורה מאובטחת יותר. יש עוד עשרות מוצרים שמצאו את דרכם לעולם הצרכנים אבל השאלה שצריכה להישאל היא *האם אתר/ תוסף/ מנגנון שאומר שכנראה כבר פרצו לי הוא הפתרון הטוב ביותר שיכולנו לבוא איתו או רק פלסטר זמני?* ובכן, אתם כבר יכולים לנחש מה הדעה שלנו.

אז מה הבעיה שלנו עם סיסמאות?

אימות מבוסס סיסמה הוא הצורה הנפוצה ביותר של אימות משתמשים. עם זאת, זוהי הדרך הפשוטה והנוחה ביותר לגניבת זהות והסלמת הרשאות בדגש על משתמשים חזקים ברשת.

רובנו אנשים של מספרים אז קצת סטטיסטיקות: כ-81% מגניבות הזהויות נובע משימוש בסיסמאות חלשות, אדם ממוצע עושה שימוש ב-107 חשבונות שונים (שלרוב מקושרים לחשבון מייל בודד) ועלות פריצות הסייבר לעסקים בארה"ב מגניבת זהויות גבוהה מ-5 מיליארד דולר בשנה.

חולשות מוכרות בשימוש בסיסמאות:

1) סיסמאות הן חלשות, חוזרות על עצמן וקלות לניחוש: מספיק להסתכל על הסיסמאות הנפוצות ביותר בשנים האחרונות כדי להבין את גודל הבעיה. ניקח את זה צעד קדימה, אתם יכולים לנחש איזו סיסמה **כבר עשור** כובשת את המקום הראשון בשכיחותה? אם עניתם כל דבר מתוחכם יותר מ- "123456" אתם טועים. גם בהגדרת מגבלות לבחירת סיסמה מורכבת לא צפויות הצלחות גדולות מדי. לדוגמה, אם נדרוש סיסמא בת 7 תווים, לפחות מספר אחד, ללא מספרים עוקבים וסימול ייחודי, במקום לקבל את mt2y@Qp, כנראה נסיים עם Admin1! המוכר. ה-meme של מישהו שם לעצמו מקל באופניים ומאשים אחרים מתאים פה בצורה מופתית.
2) מחזור סיסמאות במגוון פלטפורמות - נחשו מה, מבחינת אנטרופיה הסיסמה mt2y@Qp היא מעולה אבל אם הזנתם אותה בכל אתר אינטרנטי שקיים ופרצו לאחד מהם (באמצעות חולשה של הממשק, השרת עליו יושב ה-DB, הפרוטוקול שנעשה בו שימוש וכו') כנראה שתוך מספר שעות שאר השירותים (המייל, קופת הגמל, חברת האשראי וחלילה אף חשבון הנטפליקס) כעת פתוחים למרבה במחיר.
3) ישנם כלים ארוזים ומוכנים לפיצוח סיסמאות מגובבות. כרטיס מסך החדש GeForce RTX 4090 של Nvidia שיצא ממש לא מזמן יכול לפצח 300 מליארד (!) גיבובי NTLM בשנייה ו-200 אלף גיבובים של bcrypt. כן כן ממש לפי ה-benchmark של hashcat.
4) סיסמאות ניתנות למניפולציות חברתיות - הזנת פרטי חשבון לאתר phishing, מענה להודעת SMS לעדכון פרטי ההתקשרות לבנק, שיתוף הסיסמה עם גורם תמיכה אלמוני, וכמובן, הזנת פרטי חשבון לאחר ביצוע סקר עם 5 שאלות שבסופו תזכה בטלוויזיה 60 אינץ ודירה בהרצליה פיתוח.

אז מה היינו רוצים?

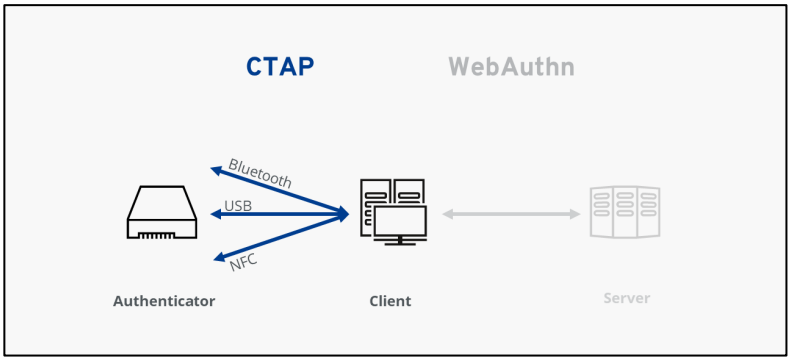
ת'אמת, לאחר הסעיפים הקודמים זה אמור להיות די straight forward - יכולת להגדיר סיסמה שמורכבת מעשרות תווים אקראיים, סיסמה ייחודית לכל שירות שאנחנו מנויים אליו, שהאתר יוודא שזה באמת אני שמנסה לגשת לחשבון שלי (בעזרת הזדהות ביומטרית או משהו בסגנון של הזדהות חזקה אחרת), שבצד השרת לא ידעו מה הסיסמא שלי (כלומר גם אם הוא יפרץ מתישהו לא יחשף שום מידע שישליך על שירותים נוספים) ולמען האמת עדיף שגם אני לא אדע את הסיסמא (ובכך לחסוך לעצמי ולארגון אפשרויות למניפולציות פסיכולוגיות ופישינג). בקיצור, **אנחנו רוצים את FIDO2.**

מרכיבי פרוטוקול FIDO2

פרוטוקול FIDO2 מורכב תת-פרוטוקולים שהוגדרו עבורו:

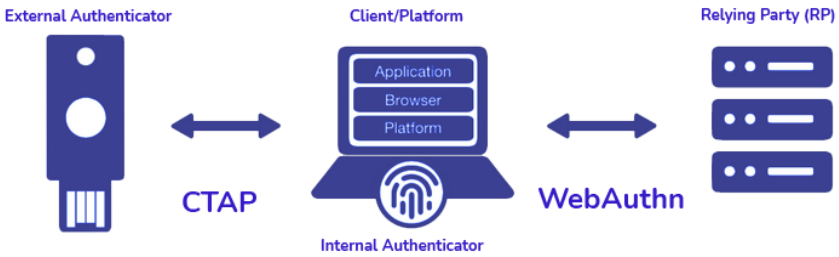
1. **CTAP** - Client to Authenticator Protocol הוא הפרוטוקול המאפשר תקשורת בין מאמת הנוכח בתצורת "חומרה נודדת" לאפליקציית לקוח. הוא מגדיר את אופן מימוש מכשיר ה"אימות" בהיבטי קריפטוגרפיה, שמירת הסוד (מפתח), ממשקים, תקשורת מול הדפדפן וצורות ההזדהות הנתמכות (טביעת אצבע, זיהוי פנים, הקלדות Pincode ועוד). CTAP הוא התפתחות של תקן FIDO U2F (אותו הצגנו בהתחלה) עם מספר שיפורים באופן המימוש, נקיון הקוד ומענה לחולשות הגנה.

בסופו של יום חשוב להבין כי CTAP נועד להבטיח שהתקשורת עם אמצעי האימות מתבצעת באמצעות ערוץ תקשורת מאובטח מבוסס טכנולוגיות 'קרובות'. תקשורות אלו דורשות חיבור פיזי (USB) או תקשורת קצרת טווח (NFC, BT) בין פלטפורמת המחשוב שמולה נדרש להזדהות ואמצעי ההזדהות (ה-Authnticater), נראה בהמשך מדוע זה נדרש, ולא ניתן לעשות שימוש בתקשורת סלולאר או Wifi בין אמצעי הזיהוי והמחשב.



2. **WebAuthn** - התקן הרשמי לאימות ללא סיסמה מול אתרי WEB. התקן מגדיר את התקשורת וחילוף המידע בין דפדפן המשתמש ושרתי הרשת מולם הוא רוצה להזדהות. התקן מגדיר שימוש במפתחות א-סימטריים, פונקציות ואוביקטים בשימוש, מימוש Challenge-Response בין ה-Client לשרתים (המכונים Relaying party - RP), יצירת מפתח ייחודי מול כל RP ועוד. במסגרת בניית התקן נכתבו ספריות למימוש Webauthn בסביבות פיתוח מוכרות ופורסמו כ-open source לכל המעוניין.

ניתן לראות את סיכום שני הסעיפים ולהמחיש את השימוש בשני הפרוטוקולים בתמונה הבאה:



WebAuthn מספק יתרונות אבטחה רבים ביניהם:

- ביטול סיסמאות ושימוש באימות חזק: שימוש בהצפנת מפתח ציבורי (Cryptography Public Key) לביצוע האימות ואבטחת התקשורת עם משתמשים על ידי אתרי האינטרנט.
- צמצום הסיכונים בפריצות נתונים: על ידי שימוש ב- WebAuthn אתרים מאחסנים רק את המפתח הציבורי של המשתמש, אשר נועד להיות ציבורי ואינו יוצר סיכוני אבטחה במקרה וזה דלף או נגנב.
- הפחתת איומי גניבת הזהות: WebAuthn משתמש במפתח ציבורי ייחודי עבור כל אתר אינטרנט. כלומר, גם במידה וגילו את ה-credentials לאתר מסויים לא ניתן לעשות איתם כלום באתרים אחרים. המפתח הפרטי שמייחד את זהות האתר למעשה לעולם לא נשלח לאתר ועל ידי העברת ואימות שם האתר בהודעה הראשונה שמגיעה מ-RP מתאפשר למנוע כל ניסיון פישינג וגניבת credentials של משתמשים שבטעות ביצעו הזדהות דרכם (כי מי לא קיבל הודעה מהנסיך הפרסי על כך שהוא זכה בירושה גדולה ויוכל לקבל אותה רק לאחר שיתחבר ל-PayPal דרך הלינק המצורף).

WebAuthen בבסיסו נועד להיות שיטה קלה למימוש עבור אתרי אינטרנט לעבור מאימות מבוסס סיסמה אל שימוש בשיטת אימות מאובטחת יותר. הפעלת WebAuthn דורשת שני שלבים ראשוניים:

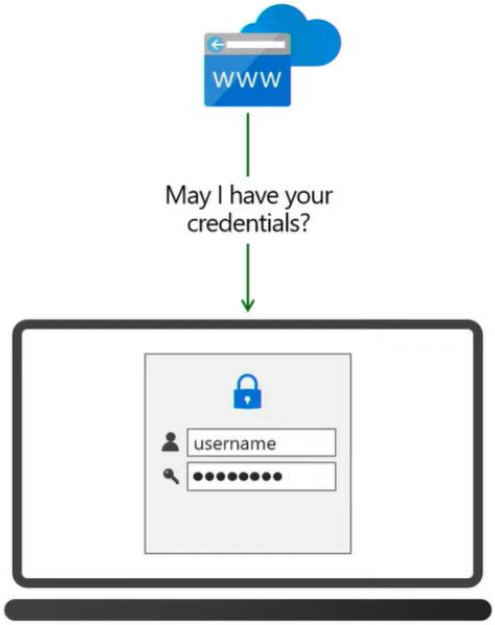
I. שינוי דפי הכניסה וההרשמה כך שיעשו שימוש בפרוטוקולי FIDO לאימות משתמשים
II. פריסת שירות אימות לקוחות התומך ב-FIDO תחת פלטפורמת CIAM (Customer Identity and Access Management) כזו או אחרת.

איך נראה FIDO2 מלמעלה

לא קיימת היום סטנדרזציה מחייבת לאימות משתמשים, לכן לפני שמציגים את שיטת האימות ש-FIDO2 מציעה, שווה לתאר בכלליות איך נראה תהליך אימות 'סטנדרטי'.

כאשר שרת רוצה לאמת את זהותו של משתמש קצה לרוב האינטרקציה ביניהם תראה בצורה הבאה: אתר האינטרנט יבקש הזנה של קלט (לרוב בתצורה של login form) בצד ה-client ← משתמש הקצה יזין את הנתונים המבוקשים (שם משתמש + סוד שידוע רק לו) ← צד ה-client יבצע גיבוב כלשהו על הסוד וימסור אותו בצמוד לשם המשתמש לשרת ← בצד השרת תתבצע שאילתה למסד הנתונים אשר תבדוק אם המשתמש קיים, האם הסוד שהוזן נכון ותחזיר את הרשאותיו של המשתמש לתוכנת ה-client (לרוב בתצורת token כזה או אחר) ← דף HTML מתאים יחזור ויטען עם משאבים שבהקצאת השרת ובהתאם להרשאותיו של משתמש הקצה.

כמובן שנשאף שכל תהליך ההתקשרות הנ"ל בין השרת והלקוח יבוצע על תווך תקשורת מוצפן (TLS).

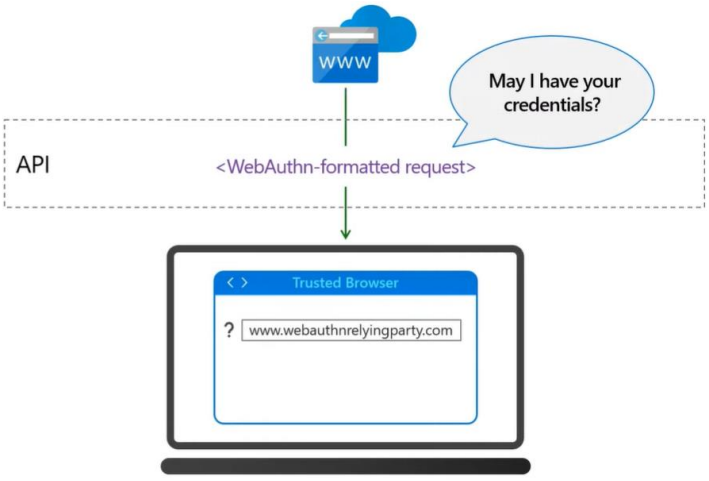


[מקור כל התמונות בפרק הקרוב - Microsoft Security]

אע"פ שהתהליך הנ"ל נשמע אלמנטרי ובסיסי, מכיוון שאין מכניזם סדיר לאימות *(וכל מפתח שני חושב שהוא כשיר להמציא את הגלגל מחדש במקום לעשות שימוש בספרייה קיימת)*, התרחיש הפשוט הנ"ל חושף את שני הצדדים למשטחי תקיפה פוטנציאליים - התחזות למשתמש אחר וגניבת זהות על ידי אתר זדוני הם הבולטים ביותר.

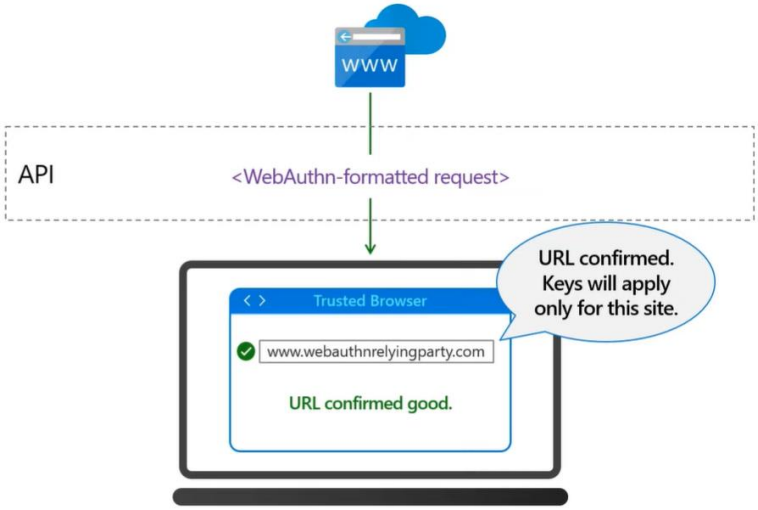
אז איך נראה אימות באמצעות FIDO2?

במקום שלכל אתר יהיה את ה-Login form שלו, ב-FIDO2 האתר פשוט קורא אל API של JavaScript שמוצע על ידי הדפדפן עצמו (כל הדפדפנים הגדולים תומכים ב-WebAuthen API). ממשק ה-JS מכיל מספר פיצ'רים שמאפשרים לאתר לתשאל את הדפדפן בנוגע למספר מאפיינים אודות המשתמש כדי לאמת את זהותו.



לפני שהדפדפן מביא לאתר את המידע הוא קודם מוודא שהאתר הוא בעצמו מי שהוא טוען שהוא. זה מתבצע באמצעות טכנולוגיה שקיימת כבר שנים - תעודות דיגיטליות.

בשפה קצת יותר טכנית - הדפדפן אחראי לוואלידציה של שדה ה-origin המוכל ב-request שהאתר שלח. מדובר בבדיקה של ה-domain או ה-subdomain שממנו הגיעה הבקשה של האתר אל מול התעודה הדיגיטלית שהוא מציג. אותה תעודה לרוב תהיה משורשרת עם תעודה אחרת שחתומה על ידי certificate authority (CA), גוף גדול ומוכר שתפקידו לאשר ולוודא כי בעל האתר אכן אוטנתי. תוכנת ה-client שלנו (הדפדפן) תקבל את התעודה של ה-CA בפורמט ASN.1 ולאחר סיריאליזציה זריזה תשווה את התוכן שלה לתעודות ששמורות אצלינו במחשב ב-Certificate Store. אם הכל עבר כשורה הדפדפן ירנדר את תוכן ה-DOM ונקבל גישה.



רעיון זה בא לפתור את הבעיה של גניבת זהויות על ידי אתר זדוני. כבני אדם קשה לנו להתמיד באופן קבוע ולבדוק שהאתר שאנחנו מזדהים אליו הוא באמת האתר שהתכוונו לגשת אליו.

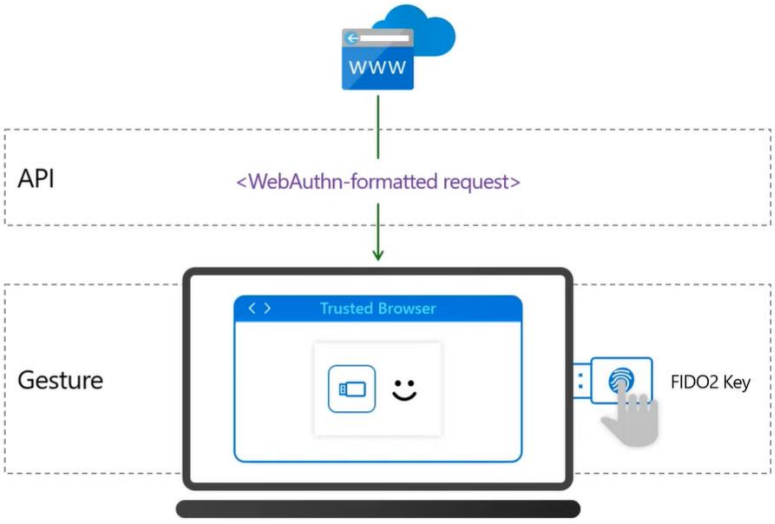
ההבדל בין www.microsotf.com אל www.microsoft.com הוא דוגמה טובה לכך. דפי Login הם בסך הכל דפי HTML וניתן להעתיק אותם במינימום מאמץ במסגרת URL spoofing. ניתן לגרום למשתמשים להזדהות מול אתרים זדוניים ובאמצעות ביצוע redirect פשוט להפנות אותם לאחר מכן לאתר המקורי מבלי שהם ירגישו בכלל שקרה משהו לא בסדר.

מחשבים ממירים את תווי ה-ASCII שמוצגים ב-URL לבתים ולכן הם רגישים יותר אפילו לשינויים המינוריים ביותר. תפקיד הדפדפן הוא לזהות באופן חד ערכי ש-domain שאנחנו הולכים להזדהות מולו הוא אכן האתר שמגיע לו לקבל את ה-credentials שלנו.

בדוקומנטציה של NIST 800-63 שאחראי על Digital identity guidelines הם מתייחסים לכל הסאגה הזו בשם verifier impersonation resistance שזו בסה"כ דרך פנסי לומר שאי אפשר לעבוד על מחשב בדרך שעובדים עלינו.

מרגע שנוצר לנו מפתח עבור Microsoft.com הוא יוכל לשמש לזיהוי ולאימות שלנו רק מול Microsoft.com ולא שום אתר אחר - בטח שלא microsotf.com. ולכן מרגע שיצרנו מפתח לאתר מסוים גם אם נמסור אותו בטעות לאימות באתר אחר - כלום לא יקרה.

חשוב כבר עכשיו לעצור ולומר ששום מפתחות \ סיסמאות לא ישלחו על ידי הדפדפן מבלי שהמשתמש יאשר ו\או יבצע פעולה כלשהי באופן **אקטיבי**. בדוקומנטציה קוראים לפעולה הנ"ל Gesture.



בנוסף לבקשת כניסה לאתר, תקן FIDO דורש מהמשתמש לבצע 'מחווה' אקטיבית על מנת להתקדם בתהליך ההזדהות מול האתר. על מנת שאמצעי הזיהוי יבצע חתימה של ה-challenge-response באמצעות המפתח הפרטי וההודעה תשלח לשרת בכדי להתקדם במתן הרשאת הגישה לשירות על המתשמש לבצע סריקת טביעת אצבע, זיהוי פנים, הקשת קוד או פעולות אחרות שיאמתו מול אמצעי האימות שהוא אכן הבעלים החוקי של אמצעי הזיהוי .

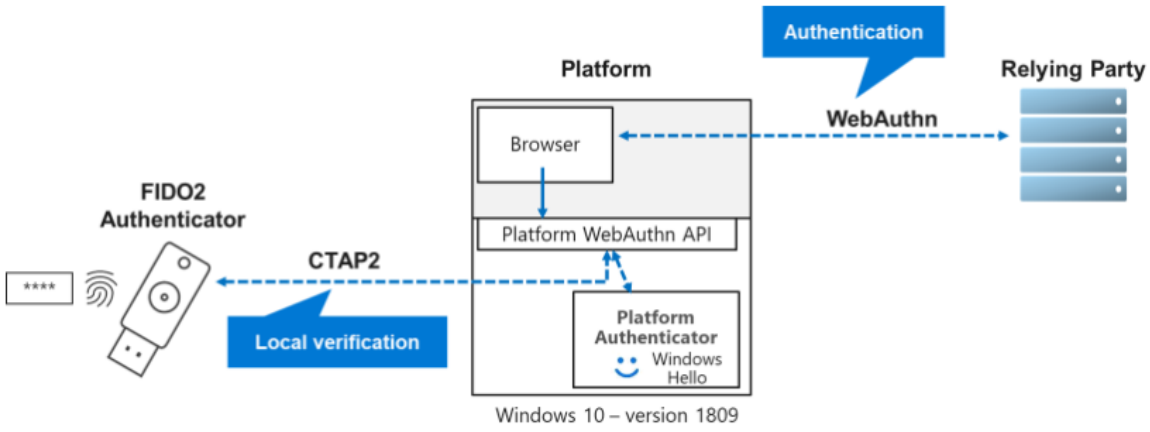
הסטנדרט משאיר פתוח את אופן מימוש ההזדהות. פעולות ה-Gesture יכולות להיות swipe up במסך לאחר שבוצע זיהוי facial recognition במכשיר, אימות מבוסס טביעת אצבע באמצעות FIDO key או שיטות עתידניות יותר דוגמת מגע בשעון יד שמודד לנו את הדופק (שתאמינו או לא אופייני רק לכם). בכך, קונצוריום FIDO מעודד תעשיות לפתח טכנולוגיות פשטות, אמינות וחדשניות להזדהות מאובטחת של משתמשים.

הקטע היפה בסטנדרט של אימות מבוסס FIDO הוא העובדה שאין מגבלה לאילו שיטות gesture ניתן לבצע את האימות באמצעותן. טכנולוגיות כמו NFC, USB, BLE שממשיכות להתפתח כבר עשרות שנים הגיעו לשלב בגרות וכעת אפשר לעשות בהן שימוש בשביל לפתח פתרונות הזדהות נוספים. התעשייה וה-ecosystem סביב FIDO יכולים לצמוח לאיזה כיוון שהכי יתאים לצרכן ובכך ליצור פתרון בצורה הענפה ביותר. User experience תמיד מהווה אבן ביניין באינטרנט.

בנוסף, **הסטנדרט מכסה את הרעיון שהפלטפורמה או מערכת ההפעלה תספק את פתרונות האימות**. ככה למשל ב-Windows 10\11 ניתן להשתמש במכניזם של Windows Hello בשביל לספק Credentials של FIDO אל הממשק של WebAuthn.

אוקיי, אז עשינו שימוש ב-gesture כזו או אחרת בשביל לאמת את זהות המשתמש, מה קורה לאחר מכן? למה אנחנו צריכים את זה? התשובה הקצרה - קריפטוגרפיה של מפתח ציבורי.

כל מאמת חייב להכיל אזור אחסון מאובטח עבור המפתחות הפרטיים ולאחר ביצוע ה-gesture נעשה שימוש במפתח הפרטי בשביל לחתום על ה-cedentials שונים עבור אתר האינטרנט ואמצעי אבטחה נוספים כמו חותמות זמן, מונים אינקרומנטליים וכו'. נרחיב על אמצעי ההגנה והשדות שמאפייני כל בקשה (ממש ברמת הקוד) במאמר ההמשך.



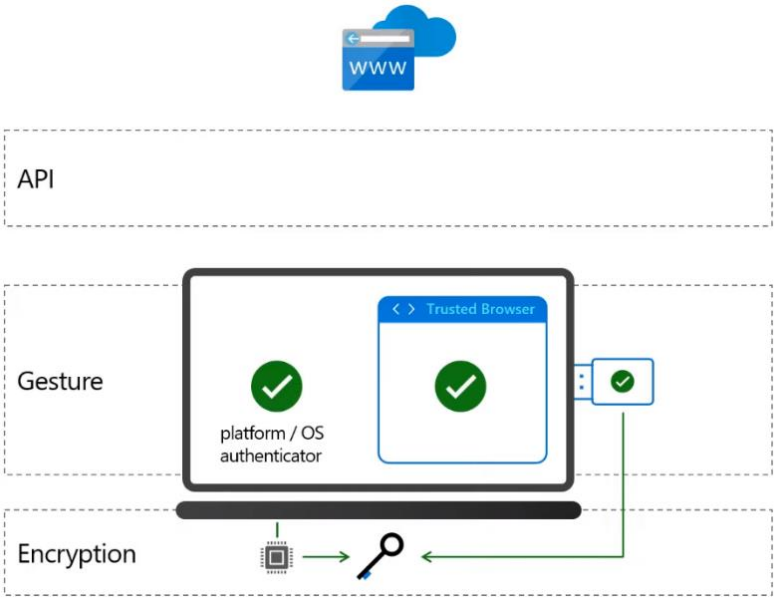
נקודה שחשוב לתת עליה את הדגש היא העובדה כי ה-Credentials (צמד המפתחות) נוצרים רק על ידי המאמת. הוא זה שבסופו של דבר יוצר, מחזיק וחותם על המידע גם בתהליך הרישום וגם בתהליך האימות.

לכל מכשיר מאמת (כדוגמת YubiKey, מכשיר סלולארי וכד') הפועל ב-FIDO2 יש תעודת X.509 אשר מאוחסנת כאשר המכשיר מיוצר; אלו הן Attestation Certificate אשר מייצגות כל מאמת. כלומר מפתח נוסף שמשותף לכלל המוצרים מאותו דגם. המפתח הפרטי צרוב במכשיר המאמת ולא ניתן לייצאו ולהתערב בהפקתו. יתרה מזאת, המכשיר חתום מבחינה קריפטוגרפית, כלומר אם תוקפים ינסו ליירט את בקשת רישום ולהחליף אותה בערך שלהם, הם לא יוכלו להחליף את המפתח הציבורי שנוצר במפתח אחר כי החתימה על ה-Attestation לא תהיה תואמת למפתחות המאמת.

בנוסף, לעיתים קרובות הארגון מעוניין להגביל את השימוש באמצעי זיהוי ולאפשר שימוש באמצעים ספציפיים בלבד. זאת לשם שימוש באמצעים מאובטחים יותר (לדגומה שימוש בחומרה ייעודית ולא לאפשר הזדהות שמבוססת על המכשיר הנייד Cross platform שנחשבת פחות בטוחה). לצורך כך נדרש לוודא מהו הדגם (והטכנולוגיה) של אמצעי הזיהוי.

הזכרנו את נושא **התעודות הדיגיטליות** כבר פעמיים, פעם אחת בדרך בה הדפדפן מאמת את זהותו של האתר ופעם שנייה בצורה בה המאמת מוכיח את זהותו לדפדפן. במידה ואתם מעוניינים להבין יותר לעומק כיצד אימות מבוסס תעודות עובד, מי מנהל את כל הסיפור, היכן בכלל התעודות שמורות לכם במחשב ואיך הכל אשכרה נראה - אני (בדעה חצי אובייקטיבית) ממליץ לקרוא את המאמר "על תעודות דיגיטליות - איך לסמוך על הסינים באינטרנט" מגליון 122 במגזין (מאמר סבבה פלוס, אם תשאלו אותי).

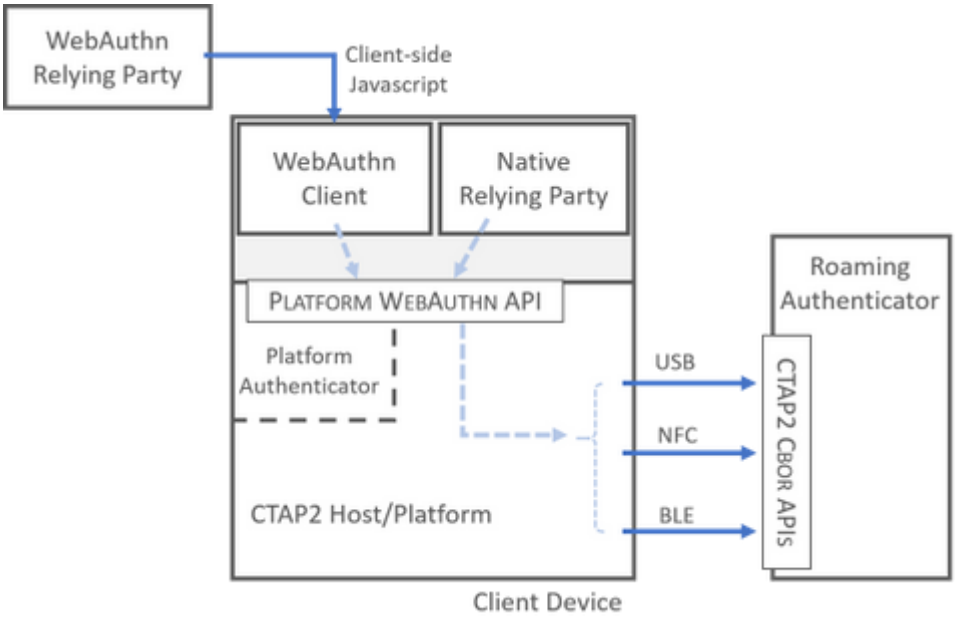
נקודה נוספת שחשוב לשים עליה את הדגש היא שאמצעי אימות בודד מאפשר הזדהות (ושמירת מפתחות) למספר רב של אתרים. בצורה כזו לא נדרש להחזיק מיליון טוקנים שונים (מונח שקיבל את השם *necklace of tokens*) בשביל לענות על הצורך של הזדהות לאתרים שונים - אחד מספיק.



לאחר שהמפתח שוחרר ממכשיר המאמת וחתם על ה-credentials עבור האתר, הדפדפן מוסר אותם לאתר עצמו לאחר שווידא כי זהות האתר היא הנכונה והוא זה שביקש את ה-original request.

התמונה הגדולה

אם לסכם בכלליות את כל מה שהוצג בפרק האחרון, CTAP2 ו-WebAuthn מגדירים ביחד שכבת אבסטרקציה שיוצרת ecosystem עבור אימות חזק בסביבת Web. נוכל לקחת צעד קדימה ולהביט על ארכיטקטורת FIDO2 הכללית בצורה הבאה:



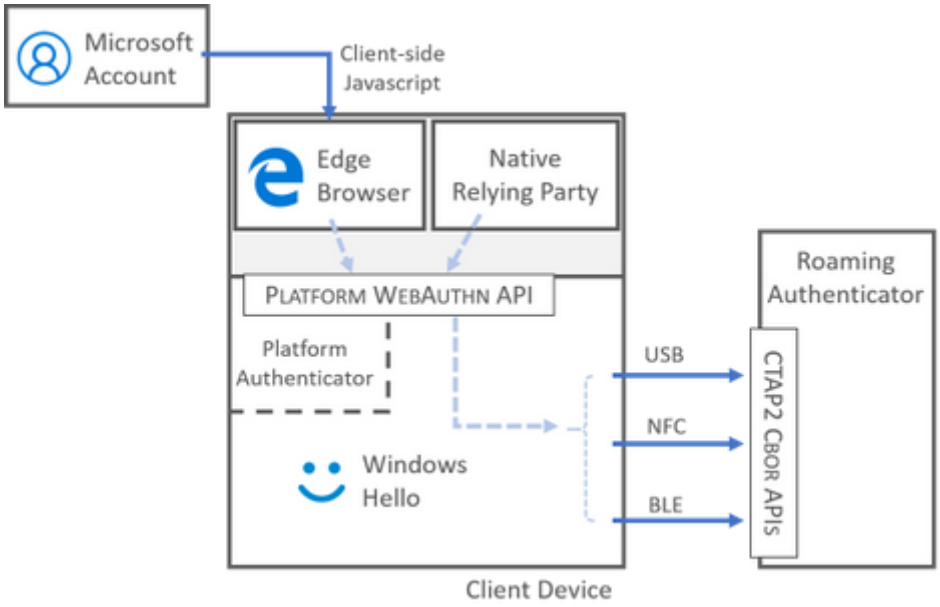
[מקור - Microsoft]

הערה: החצים המקווקים בתכלת מייצגים אינטראקציות התלויות ביישום הספציפי של ממשקי ה-API של הפלטפורמה וניתנים למודפיקציה. נרחיב על החלקים השונים בשרטוט:

- **Relying Parties & Clients** - אפליקציות web או מערכות native שרוצות לעשות שימוש ב-FIDO2. השוני העיקרי הוא כיצד הן מממשות את האימות. באפליקציות web האתר שרוצה לעשות שימוש ב-FIDO לא יכול לפנות ישירות אל ממשק ה-WebAuthn API וחייב לעשות שימוש במתווך (הדפדפן). באפליקציות native המצב שונה ויכול להיות מצב שבו ה-RP ירוצו על מכשירי הלקוח ובכך ישמשו גם בתור WebAuthn client בשביל לשלוח ולקבל בקשות של WebAuthn. חשוב לציין שאין שום תזכור לשימוש ב-Single sign-on (SSO) בארכיטקטורה הנ"ל.
- **Client Device** - ממש החומרה שמבצעת את האימות. פלטפורמת המחשוב שממנה המשתמש מבקש לצרוך את השירות (שעבורו הוא נדרש להזדהות).
- **WebAuthn API** - הממשק שמאפשר ללקוח לשלוח בקשות למכשיר האימות.
- **Platform Authenticator** - אימות מובנה על מכשיר הלקוח. לא ניתן לגשת אליו בפרוטוקולים cross-platform כמו USB, NFC, BLE אלא רק על ידי עבודה ישירה מול החומרה שעל המכשיר. סורק טביעת אצבע על מחשב לפטופ\ טלפון חכם הוא דוגמה טובה לכך.
- **CTAP2 Platform/Host** - החלק במכשיר הלקוח שאחראי לניהול המשא ומתן עם המאמת.

- **Roaming Authenticator** - מאמת המסוגל להתחבר למספר מכשירי לקוח וחייב לעבור אינטראקציה עם לקוח הקצה בשביל לחתום באמצעות הסוד שהוא מכיל על ה-credentials. דוגמאות למאמתים נודדים - כרטיסי קרבה מבוססים NFC, טלפונים חכמים מאופשרי BLE, מפתחות על גבי USB וכו'. החלק החשוב הוא שמכשירים האלו יתמכו בפרוטוקולים CTAP1 ו\או CTAP2.

כל מכשיר לקוח (כמו דפדפן או אפליקצית native) יכול לממש את הסטנדרט עם כל מכשיר אימות. לשם הדוגמה נסתכל על היישום של חברת Microsoft ל-FIDO2 בפלטפורמת Windows עוד מ-2018:



[מקור - Microsoft]

אכן, הארכיטקטורה נראית מאוד זהה לתרשים הכללי של FIDO2 אך כעת כל מרכיב מקבל תפקיד מוגדר:

- **Microsoft Account** מממשת את ה-WebAuthn RP. באמצעותה ניתן להירשם למספר שירותים ב-Windows החל משירותיים ליבתיים כמו Outlook וחלה בשירותים ביזריים שספק מדוע הם עוד קיימים כמו Xbox. בכל פעם (כמעט) שתראו שירות Windows שדורש Sign-in תהיה אופציה לשימוש ב-FIDO2 על ידי שימוש ב-client side javascript בשביל לטרגר את הדפדפן Microsoft Edge לתקשר עם הממשקים של WebAuthn APIs.
- **Microsoft Edge** מהווה את ה-WebAuthn client. דפדפן הדגל של Microsoft תומך בהתממשקות מלאה מול פיצ'רים של פרוטוקולי CTAP2 ו-WebAuthn בסביבת Windows וגם בהרחבה של AppID שמהווה תאימות לאחור עבור U2F. עם זאת, כשזה מגיע ל-Android דפדפן Edge לא תומך בכלל ב-WebAuthn. זאת כבר בעיה של גוגל ו-Chrome.

הערה: במקרה הספציפי של Windows, מכיוון ש-Microsoft Account צריכה פיצ'רים והרחבות שקיימות בלבד במאמתים של FIDO2 CTAP2, לא ניתן לעשות שימוש ב-credentials שהושגו דרך CTAP1 (UAF). זאת למרות ש-Edge תומך בסוגי מאמתים שעושים שימוש גם ב-CTAP1 וגם ב-CTAP2 ולכן הוא יכול ליצור גם U2F וגם FIDO credentials.

- **Windows 11 \ Windows 10** מייצגים את הפלטפורמה. שניהם מכילים את Win32 Platform WebAuthn APIs.
- **אין מוצר proprietary של Microsoft** שמהווה את Roaming Authenticators בשירטוט מדוע? ובכן, הסיבה לכך היא מכיוון שיש כבר ecosystem מבוססת של מוצרים שמתמחים באימות חזק, ולכל לקוח (בין אם תאגידים או אנשים) יש דרישות שונות לאבטחה, קלות שימוש, הפצה ושחזור חשבון. Microsoft פשוט לא הייתה צריכה להיכנס לתחום הזה ולהשקיע ב-R&D למימוש מוצר משלה כשיש כמעט 1000 (!) מוצרים קיימים. שווה לציין כי Windows Hello מהווה ענף נפרד מאשר מאפשר אימות מבוסס FIDO2 אבל למרות שהוא מגיע עם מערכת ההפעלה הוא לא תמיד יהווה את הבחירה האינטואיטיבית.

למידע נוסף על הרשימה ההולכת וגודלת של מאמתים עם אישור FIDO2 שווה לקפוץ לקישור הבא. הרשימה כוללת מאמתים מובנים, מאמתים נודדים ואפילו יצרני שבבים בעלי designs מאושרים. סה"כ כמעט 1000 מאמתים שונים בעת כתיבת המאמר.

מקרה בוחן - Uber breach 2022

נסכם את המאמר הראשון מבין השניים ואת אפקטיביות ההגנה שמספקת הזדהות מבוססת FIDO2 עם דוגמה פרקטית מהתקופה האחרונה - ניתוח אירוע הפריצה האחרון לחברת Uber.

ב-15 לספטמבר 2022 השיגו תוקף(ים) את פרטי ה-Login (שם משתמש וסיסמה) של עובד בחברת Uber לשירות ה-VPN של החברה. שירות ה-VPN משמש עובדים מרחוק לצורך חיבור לרשת הפנימית של Uber (ולאפשר עבודה מהבית תוך מתן גישה לשירותים ברשת הפנימית של החברה).

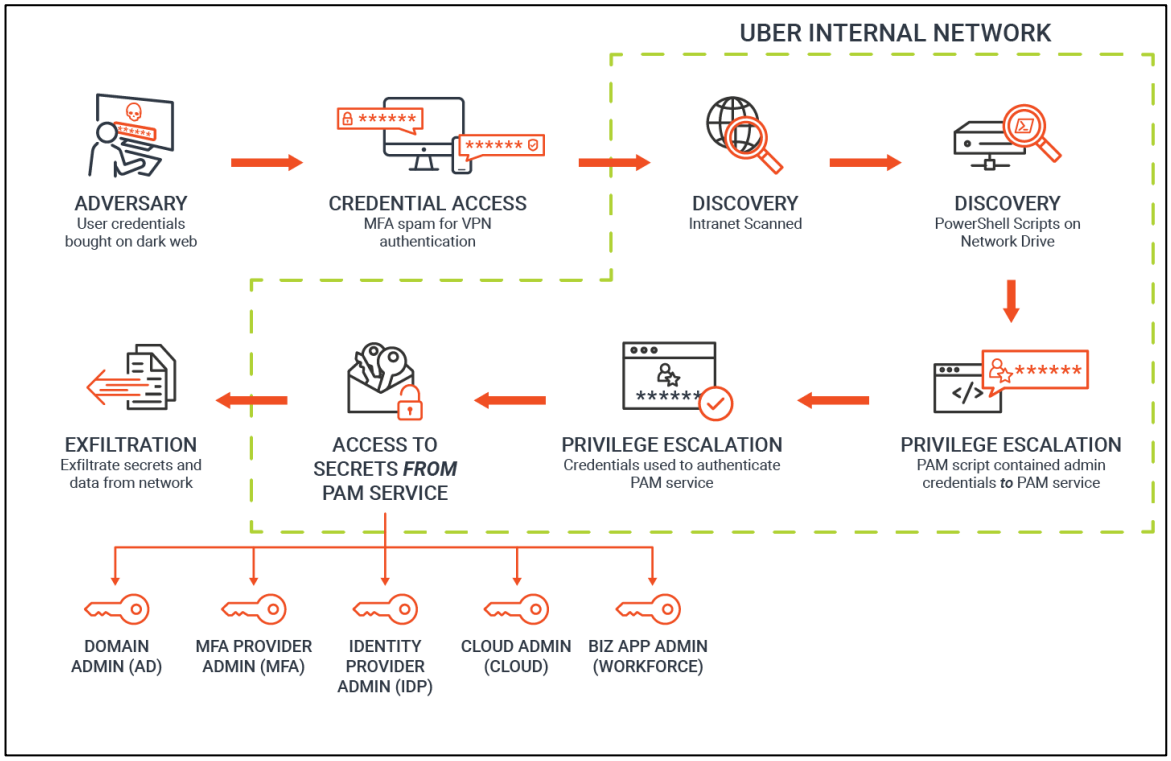
לא ידוע כיצד השיגו התוקפים את פרטי ה-Login, האם באמצעות נוזקה שהותקנה על מחשב העובד ושלחה את פרטי ההתקשרות לתוקפים, באמצעות קניית threshold access credentials בפורום מחתרתי או עקב ביצוע Phishing לעמוד הכניסה ל-VPN אליו הוזנו פרטי העובד.

לצורך התחברות לחשבון ה-VPN נדרש העובד להזין את פרטי Login, ובנוסף לאשר את החיבור באמצעות MFA שבמבוסס על אישור ההתחברות בהודעת Push notification שנשלחת למכשיר הסלולארי של העובד שעליו ללחוץ עליה לאישור את ההתחברות ל-VPN.

על מנת להתגבר על מנגנון ה-MFA התוקפים עשו שימוש בהנדסה חברתית (Social engeneering) מבוססת **MFA Fatigue** (התשת ההזדהות הדואלית). בכל נסיון חיבור של התוקפים ל-VPN נשלחה לעובד הודעה על נסיון התחברות (שלא הוא ביצע). לאחר עשרות סירובים של העובד לביצוע ההתחברות, התקבלה הודעת מייל מהתוקפים שהתחזו ל"צוות ה-IT" של Uber על תקלה בהתחברות לשירותי ה-VPN וביקשו מהעובד לאשר את ההתחברות על מנת להפסיק את הבקשות הבלתי פוסקות.

לאחר אישור הבקשה, קיבלו התוקפים גישה לרשת הפנימית של Uber. מכאן, הכניסה לרשת UBER הייתה סלולה בפני התוקפים לחיפוש חולשות, סיסמאות, misconfiguration, ושימוש בכלי תקיפה מוכרים ברשת הפנימית. נמצא Share point פנימי עם קבצי Powershell שכללו סיסמאות גלויות (clear text) למערכת ה-PAM של Uber (Privileged Access Management - מערכת לניהול הרשאות וסיסמאות ליישומים ארגוניים). מכאן קצרה הדרך לקבלת הרשאות מנהלן ל-Slack, למערכת ניטור הסייבר sentinelOne, לשירותי ה-AWS, ה-VMware sphere ואל hackerOne (שככול הנראה מכילה פגיעויות בקוד של יישומי uber).

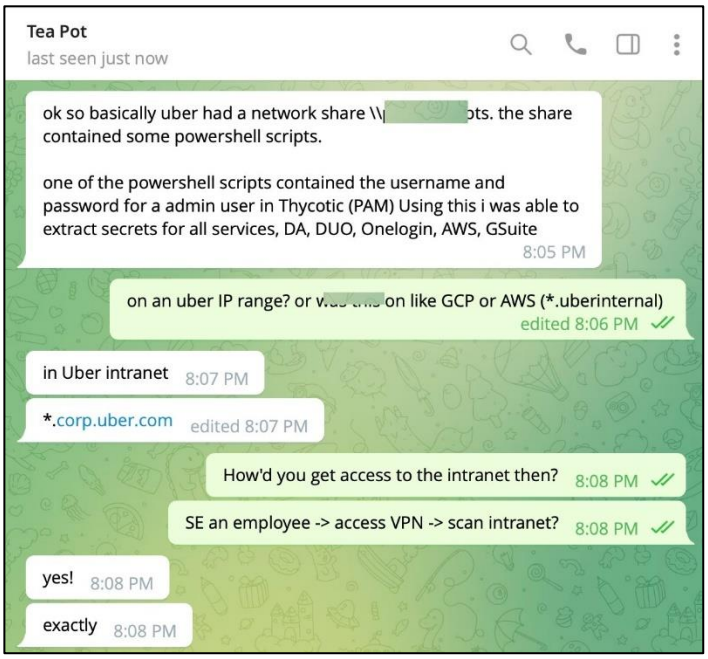
ל-CyberArk יש תרשים פנטסטי שמתאר את מהלך התקיפה:



[מקור - https://www.cyberark.com/resources/blog/unpacking-the-uber-breach]

למזלה של Uber התקיפה לא בוצעה למטרות כופר או ריגול תעשייתי, אלה להאדרת שמו של התוקף (שהתברר כנער בן 17, ספק אם חבר ב-Lapsus$, ולכאורה הספיק לפרוץ גם ל-GTA6 *ולעבור dox על ידי לא אחר מ-pompompurin הבעלים של BreachForums*) ולכן התוקף הקפיד לעדכן בזמן אמת על ההתקדמות התקיפה.

תמונות כמו זו, שותפו בכל מקום:



ניתן למצוא מגוון פערי אבטחה שאפשרו את התקיפה (וכנראה שבראשם שמירת סיסמאות גלויות למערכת ה-PAM) אבל באמצעות שימוש באימות מבוסס FIDO2 (במקום מימוש ה-MFA הלקוי שביצעו Uber) ניתן היה למנוע את הגישה של התוקפים אל חיבור ה-VPN ולמנוע את התקיפה מלכתחילה:

1. בניגוד למימוש ה-MFA שבוצע ב-Uber שהסוד כלל לא נשמר עליו (ה-MFA שימש רק כאישור נוסף לפרטי ה-Login שהוזנו לאתר, כך שה"סוד" שאיפשר גישה לשירות הוא שם המשתמש והסיסמא) באימות מובסס FIDO2 הסוד עצמו שמור על ה-Authenticater וכלל איננו נגיש למשתמש (שאיננו מודע לפרטי ה-Private key ששמורים על אמצעי הזיהוי, ולא יכול למסור אותם לגורם חיצוני). הפיצול שעשו ב-Uber בין הסוד ואמצעי ה-MFA אפשרה את הפריצה ל-VPN.
2. אמצעי הזיהוי ב-FIDO2 דורשים חיבור פיזי של אמצעי הזיהוי (ה-Authenticator) לפלטפורמה ממנה מבקשים את השירות (לדוגמה: נדרש חיבור USB של אמצעי האימות למחשב), או קרבה פיזית בין המזהה לפלטפורמה (לדוגמה: בהזדהות לשירות במחשב, אמצעי הזיהוי מקושר למחשב בתקשורת קצרת טווח מבוססת BLE או NFC) - באופן זה, לא ניתן להוציא את פרטי ה-Login ללא גישה פיזית לאמצעי. לדוגמה: רק עם חיבור ה-Yubikey authenticator לממשק ה-USB, והזנת פרטי ה-Login לאתר, תנתן גישה לשירות ה-VPN. בחלק מאמצעי הזיהוי של FIDO2 המשתמש נדרש להזין טביעת אצבע (בין אם בטלפון או במחשב) על מנת לבצע את ההתחברות. פרטים אלו אינם ניתנים להעברה, ומונעים אפשרות להוצאה במרמה של פרטי ה-Login.
3. כפי שראינו, FIDO2 כולל מנגנון הגנה מובנה מפני תקיפות Phishing (שליחת שם האתר המדויק במסגרת הודעת ה-Challenge\Response הראשונה שנשלחת למשתמש מהשרת) - תמיכה של שרת ה-VPN בפרוטוקול WebAuthn היה מונע את השגת פרטי ה-Login של העובד לשרת ה-VPN.

לסיכום, בגישה לשירותים קריטיים בפרימטר המאוים של הארגון (חיבור המשתמשים מהבית לרשת הפנימית ע"ג רשת האינטרנט) היה נכון להשתמש באמצעי אימות מאובטח יותר כדוגמת FIDO2 ובכך לדרוש הזדהות באמינות גבוהה עם קרבה פיזית של ה-Authenticator לפלטפורמה ממנה מתבצעת ההתחברות. את זה אי אפשר לעשות מלונדון אל אמריקה בקלות רבה כל כך.

סיכום

הזדהות משתמשים באמצעות FIDO2 מונעת שימוש בסיסמאות חלשות או סיסמאות חוזרות, גישה לשירותים באמצעות גניבת זהות המתשמש ומקשה מאוד על תקיפות Social engeneering ו-Phishing. בפועל, משתמש הקצה כבר לא נדרש לזכור או להזין סיסמאות אלה לעשות שימוש ב-Authenticator עליו נשמרים הסודות לכל שירות אליו הוא נרשם.

נכון להיום, התשתית לשימוש ב-FIDO2 קיימת - מרבית הדפדפנים תומכים בתקן, ישנן Frameworks עתירות כוכבים ב-Github למימוש FIDO2 במגוון שפות תכנות ומתפתח שוק רחב של אמצעי הזדהות במחירים שווים לכל נפש. למרות זאת, רק מעט שירותים תומכים כיום בהזדהות מבוססת FIDO2.

בשונה ממה שאומרים בבחירות, בתהליכי טרנספורמציה דיגיטלית השינוי מתחיל מלמעלה ולא מלמטה. סטאטוס של גדולות הטכנולוגיה (שתכל'ס די מצופה מהן להוביל את התהליך) הוא שיקבע את כניסת השינוי והמעבר למימוש מלא ורחב של טכנולוגיות חדשות כמו FIDO2.

כשאנחנו מביטים על ענקי הדפדפנים והיצרניות שדוחפות את הקידמה בהם: Chrome ,Chrome Desktop Android ,Microsoft Edge ,Firefox ,Safari macOS ,Safari iOS לא תמיד נִרְאֶה שמחוגי הטכנולוגיה נעים בקצב שהיינו מקווים לו (מספיק להסתכל על הסטאטוס של Apple בנושא התמיכה ב-CTAP2 שכבר שנים עומד במקום) אבל חשוב לזכור כי מדובר בתהליך. תהליך שמטרתו בסופו של יום, עם כל הקיטשיות, **להפוך את האינטרנט למקום בטוח יותר.**

כאן מסתיים לו המאמר הראשון מבין השניים שהכנו. כלל המאמר עסק בהסבר Hige level ובלא מעט ניפנופי ידיים בנוגע לטכנולוגית FIDO2 ולמעשה היווה הכנה נוחה למאמר הבא אשר מיועד להיות משמעותית יותר hands-on עבור אלו שרוצים להבין כיצד האימות מתרחש "באמת" - איזה מידע עובר, כיצד הוא נראה וכיצד הוא ת'כלס מונע מתקפות בשלב האימות. יש למה לצפות.

על הכותבים

יהונתן אלקבס, בן 28, חוקר אבטחת מידע בחברה לא קטנה ולא פרטית. חובב סוקולנטים, סודה וקפה.

אלון בר סלע, אבא של כרמל, הלל, אוריה ואמה. מנהל אבטחת מידע בחברה לא קטנה ולא פרטית.