Conditional Access Policies

מאת ספיר פדרובסקי

הקדמה

במאמר זה אנסה להיכנס לעולם של CAP, הלוא הם Conditional Access Policies. (ואל תקראו להם GPO, כי זה באמת פשוט מעליב...). אצלול להגדרות השונות שניתן להחיל, ואציג מעקפים פשוטים ומסובכים יותר על כל אחת מהן.

אני כן אציין שהחלק של המעקפים המסובכים מתבסס על כמה מהמאמרים הקודמים שלי, בעיקר על המאמר בנושא PRT.

אז... מה זה?

אם נסתכל בהגדרה היבשה באתר של מיקרוסופט, CAP הן:

> *"Conditional Access policies at their simplest are if-then statements; if a user wants to access a resource, then they must complete an action."*

האמת היא שזה תיאור מצוין. בעצם הרעיון של CAP היא ליצור תנאים שיאפשרו או ימנעו גישה למשאבים.

כמובן שזה מאוד מופשט, נראה בהמשך כמה חזקות היכולות של CAP.

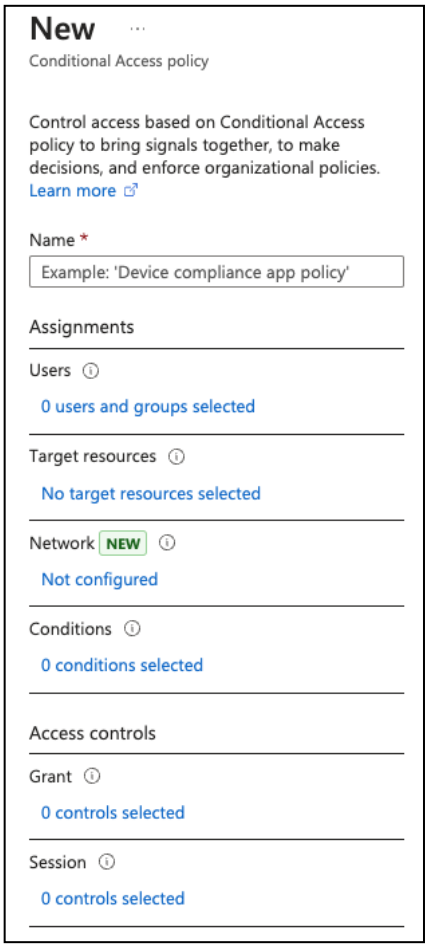
אחת הבעיות הנפוצות בשימוש ב-CAP, היא היווצרות של תחושת ביטחון. כאשר בפועל, קונפיגורציה נכונה ומוצלחת של CAP היא לא פשוטה בכלל, ומעבר לכך, בקלות מאוד ניתן לבצע קונפיגורציה בעייתית, מתירנית ומסוכנת.

המטרה של המאמר הזה היא חלילה לא להגיד לכם כיצד לסדר את ה-CAP שלכם, אלא להכיר לכם את העולם והסכנות הטמונות בו. (ותכלס המטרה האמיתית היא שרציתי להעמיק את הידע שלי ב-CAP וחיפשתי תירוץ)

אז נסתכל על מיסקונפיגורציות נפוצות, מעקפים מסובכים יותר ופחות, וכל מיני אפשרויות מעניינות להגברת האבטחה בארגון שלנו :)

יאללה לקנפג!

נתחיל, והדרך הטובה ביותר בעיני להבין משהו הוא פשוט לעשות אותו. אז בואו ננסה ליצור CAP. לשם כך נכנס ל-Entra ומשם ל-Security -> Conditional Access -> Policies כעת נבחר באופציה ליצור פוליסה חדשה: New Policy.



אני חושבת שרק מהתמונה בצד שמאל ניתן ללמוד המון, אך אנו נצלול לתוך כל אחת מהאופציות כאן ונבין מה האפשרויות שלה, ומה ההשלכות.

הרעיון של CAP הוא פוליסה מסוג אם->אז , ניישם את ההבנה הזו על התמונה שאנו רואים.

כל החלק הראשון עד ה-Access Controls הוא החלק של "אם" החלק מתחת, הוא ה"אז".

נעבור על כמה סוגי "אם":

Users:

אין כל-כך מה להרחיב כאן, ניתן לבחור משתמשים או קבוצות (או גם וגם) ולהכניס אותם תחת "Include" או "Exclude"

Target resources:

כאן בעצם יש 3 קטגוריות:

Resources - בעצם מדובר באפליקציות אליהן נזדהה. אפשר לבחור אפליקציות ספציפיות, או באופציה All Cloud Apps (או בשמה החדש All resources). כמו כן אפשר להשתמש ב custom security attributes שזה בגדול סוג של ביטוי רגולרי על שדות ואני מאוד לא ממליצה על זה :)

- User actions - פה ניתן לבחור מבין 2 אופציות:
  1. Register security information
  2. Register or join device - אני **ממש** ממליצה לפחות לסמן את זה, לא חסרות מתקפות המבוססות על כך שלא נדרש MFA כדי להוסיף Device, לקבל PRT ולעשות בלאגנים.

- Authentication context - זה בעצם כמו תגית (Label) שנוכל לתת לאפליקציה על מנת שהפוליסה תחול עליה.

Network:

פה נוכל ממש לבחור טווחי IP, או מיקומים שרק מהם נאשר התחברות (למשל, רק תקשורת ממדינת ישראל). או לחילופין, כל מי שמתחבר מהם, תשלל לו ההתחברות או שיחויב ב-MFA וכו. (כמובן שזה ניתן למעקף על ידי VPN, זו דוגמא קלאסית להגדרה שגורמת לנו להרגיש בטוחים, כאשר היא אחת ההגדרות שהכי קל לעקוף)

Conditions:

אני מאוד אוהבת את הסעיף הזה למרות שכמעט תמיד ניתן לעקוף אותו. הוא מכיל דברים כגון:

- Device platforms (למשל, רק משתמשי Windows)
- Locations (אני דיי בטוחה ש-Networks יחליף את Locations בהמשך)
- Client apps (לדוגמא, רק גישות מהדפדפן)
- ניתן גם לבחור device-ים ספציפיים או Device-ים שמתאימים לביטוי רגולרי כלשהוא (למשל כל אלו עם המחרוזת PROD בשם)
- Authentication flows - זה אחד נחמד! לדוגמא, ידוע שבאמצעות Device Code Flow מריצים הרבה מתקפות פישיניג, אז אפשר להגדיר שאם יש הזדהות באמצעות Device code flow, לחייב ב-MFA או כל מיני תנאים נוספים

סך הכל, התנאים ממש מגניבים בעיניי ואני חושבת שאפשר לשחק איתם ולעשות דברים ממש נחמדים.

ועם זאת, ניתן לעקוף כמעט כל אחד ואחד מהם:

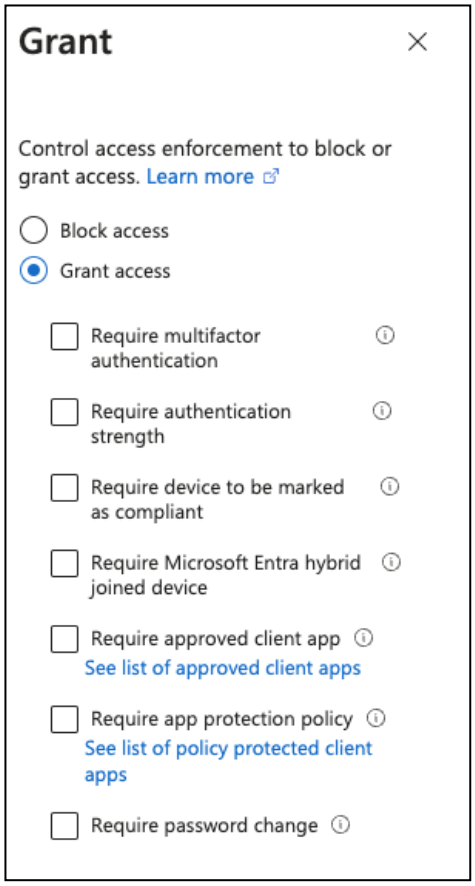
- Device platforms - מסתמך על הדפדפן, לכן כל שיטה המאפשרת לשנות את הדפדפן מאפשרת לעקוף את התנאי הזה.
- Locations - כמובן ניתן למעקף על ידי שימוש ב-VPN
- ClientApps - את זה הכי קל לעקוף באמצעות שימוש בדפדפן, כי הסיכוי ששימוש בדפדפן חסום הוא ממש ממש אפסי
- Authentication flows - דווקא דיי טוב, אבל בסופו של דבר אם לא נצליח להשתמש ב-Device code, (לצורך הדוגמא) פשוט נסה flow אחר

כעת, אנחנו מגיעים לשלב של "אז", זאת אומרת: **אם** אחד מפרטי ההזדהות מתאימים לאחד מהתנאים שהגדרנו.. **אז**:

Grant:

כאן אנחנו יכולים להגיד, תן גישה או תחסום גישה, בהתאם לתנאים לבחירתנו (ספוילר, זה החלק שהמעקפים הכי מעניינים קורים בו)

אפשר לראות את התנאים, הם דיי ברורים:



בדרך כלל, מסמנים את אחד מארבעת התנאים הראשונים:

- לדרוש MFA
- לדרוש עוד Authentication Factor חזק, למשל FIDO
- לדרוש Device יהיה Compliant (במוצר MDM כמו למשל Intune)
- לדרוש Device יהיה Joined

<u>אגב כמה מילים על Complaint:</u>

כאשר אנו אומרים Complaint, אנחנו מתכוונים שה-Device נרשם ב-Intune (או מוצר ניהול אחר - MDM) ועבר מספר בדיקות.

בדיקות כאלו יכולות להיות למשל

- Device שייך ל-AD Domain מסוים
- ל-Device מערכת הפעלה מסוימת
- וכו

הבדיקות האלו לא נוצרות באופן דיפולטי, מנהל הסביבה נדרש ליצור אותן, לכן יש סיכוי טוב שאין לכם בכלל תנאים מיוחדים כדי להפוך Device ל-Complaint ואתם רק צריכים לרשום אותו ל-Intune (שזה מסתכם בגדול בהנפקת תעודה).

דיי מזמן יצא מאמר ממש חביב המסביר כיצד לזייף Device ולרשום אותו כ-Complaint , כאשר הוא בכלל לא קיים!

https://aadinternals.com/post/mdm/

אני מוכנה להתערב שזה עדיין עובד ☺

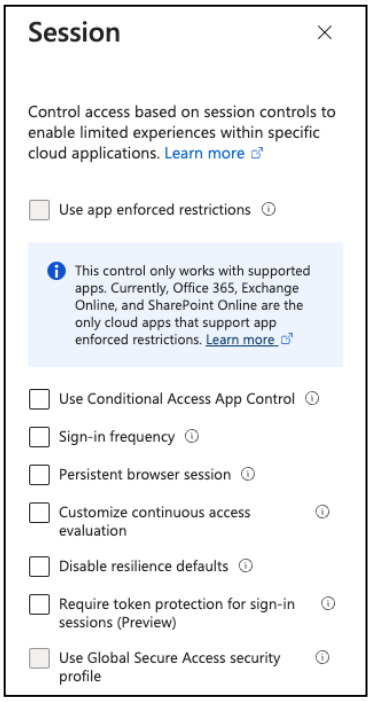
בגדול, הקוד פשוט שולח הודעה שאומרת "ה-Device סבבה!", ו-Azure רושם אותו כ-Complaint. בקטן, Azure לא בודק מי אמר לו שהרכיב סבבה, זו סתם קריאת API. אז כל מה שצריך לעשות, זה להשיג Token ל-b730954-1685-4b74-9bfd-dac224a7b8941. (הלוא הוא AADGraph שבגדול אמור להיות Deprecated) ולשלוח קריאת API שעושה Set לערך Is Complaint של ה-Device.

ולהלן:

```
$body=@{}

if($Compliant)  {$body["isCompliant"] = "true"}
if($Managed)    {$body["isManaged"] =   "true"}
if($Intune)     {$body["deviceManagementAppId"] = "0000000a-0000-0000-c000-000000000000"}


$headers=@{
    "Authorization" = "Bearer $AccessToken"
}

# Get the object Id if not given
if([string]::IsNullOrEmpty($ObjectId))
{
    $ObjectId = Get-DeviceObjectId -DeviceId $DeviceId -TenantId $tenantId -AccessToken $AccessToken
}

# Set the device compliance
Invoke-RestMethod -UseBasicParsing -Method Patch -Uri "https://graph.windows.net/$tenantId/devices/$ObjectId`?api-version=1.61-internal"

Get-DeviceCompliance -ObjectId $ObjectId -AccessToken $AccessToken
```

זה היה אמור להיות בחלק של מעקפי CAP אבל זה התפלק ל-Section הזה במאמר, כנראה מהתרגשות... ☺

נמשיך, דיברנו על Grant, עכשיו נדבר על האופציה האחרונה-Session:



זה גם אחד אהוב עלי, בעיקר **sign-in frequency**!

באמצעות האופציה הזו נוכל לחייב את המשתמש להזדהות מחדש בכל X שעות/דקות, ואם אנחנו ממש אכזריים ומחפשים חוויית משתמש מזעזעת במיוחד, נוכל לבחור בצמד המילים האיימתי Every time . מה שיכריח את המשתמש להתאמת מחדש בכל פעם שהוא ניגש ל-Client חדש.

עוד קונספט שנכנס פה הוא **CAE - Continuous access evaluation:**
הפיצ'ר הזה בעצם מאפשר ל-Entra לעשות Revoke ל-Token קיים באופן אוטומטי במידה והגיע אירוע מסוים.

לדוגמא, באמצע ה-Session שלי מול SPO (SharePoint Online), האדמין של הארגון שלי קינפג שאני חייבת לבצע MFA.

SPO יודע לדבר עם Entra, לקבל ממנו עדכון בלייב (או, עד דילאיי של 15 דקות לטענת מיקרוסופט) שה-Token שלי לא ולידי יותר כי אין בו MFA, ולעשות לו Revoke, מה שידרוש ממני להתאמת מחדש (הפעם עם MFA)

אוסיף לינק להרצאה שמסבירה על הקונספט הזה יותר לעומק, ועל רשימת הדרישות הנחוצות לתמיכה ב-CAE (למשל כיום, רק אפליקציות של מיקרוסופט תומכות בפיצ'ר).

אחד אחרון שנדבר עליו, הוא **Token protection**, זה בעצם Token binding. המטרה של הדבר הזה, היא להוכיח שאכן השתמשנו ב-Token שלנו מה-Device שהיינו אמורים להשתמש בו. כיצד נעשה זאת? נצטרך להשתמש בסוד שנמצא רק על ה-Device, ולא נוכל לגנוב אותו באמצעות קוד מרוחק. אני מדברת כמובן על מפתחות הצפנה שנשמרים ב-TPM! (ספוילר, עקפו גם את זה, stay tuned!).

אז הקונספט כאן הוא לחתום את הבקשה שלנו ל-AT (Access Token) באמצעות ה-Session Key, זה הוא בעצם POP (Proof of possession) . בהמשך אסביר על המעקף הזה, ונצלול יותר לעומק למה בדיוק מוצפן פה, ואיך בדיוק עוקפים את זה :)

דברים שחשוב לדעת על CAP

1. בשונה מ-GPO למשל, ניתן שיחולו מספר CAP על ישות במקביל, ואין "עדיפות" לאף אחד מהם. כמובן, שבמידה ואחד מהם מחמיר יותר מהשני (ביצוע פעולת Deny), הוא זה שיתפוס פיקוד.

2. כאשר אין CAP, נכנס לפעולה משהו בשם Security Defaults, מדובר בעצם בסט של חוקים בסיסיים החלים על ה-Tenant. (למשל, האם יש לכל המשתמשים MFA?). הנה הרשימה השלמה של כלל ה-Security Defaults:

   - Requiring all users to register for multifactor authentication
   - Requiring administrators to do multifactor authentication
   - Requiring users to do multifactor authentication when necessary
   - Blocking legacy authentication protocols
   - Protecting privileged activities like access to the Azure portal

   במידה וניצור אפילו CAP אחד, ה-Security Defaults באופן אוטומטי נכנסים למצב Disable. לכן שימו לב שאם קנפגתם CAP, עליהם לכסות את כל מה שה-Security defaults כיסו קודם לכן, אחרת את מפסידים שכבת הגנה!

3. הבדיקה על ה-CAP מתחילה רק לאחר שסיימנו את החלק הראשון של ההזדהות (למשל, הכנסנו שם משתמש וסיסמא נכונים).

מעקפים שאני אוהבת

בכנס Black Hat Europe האחרון, הוצגה הרצאה ממש מעניינת (קישור למצגת), ובהתבסס עליה פורסם מאמר חדש מצוין, על מעקף CAP ספציפיים, של Require complaint device.
האמת היא שממש אהבתי את המעקף הזה! הוא בעצם מבוסס על Client ID פגיע.

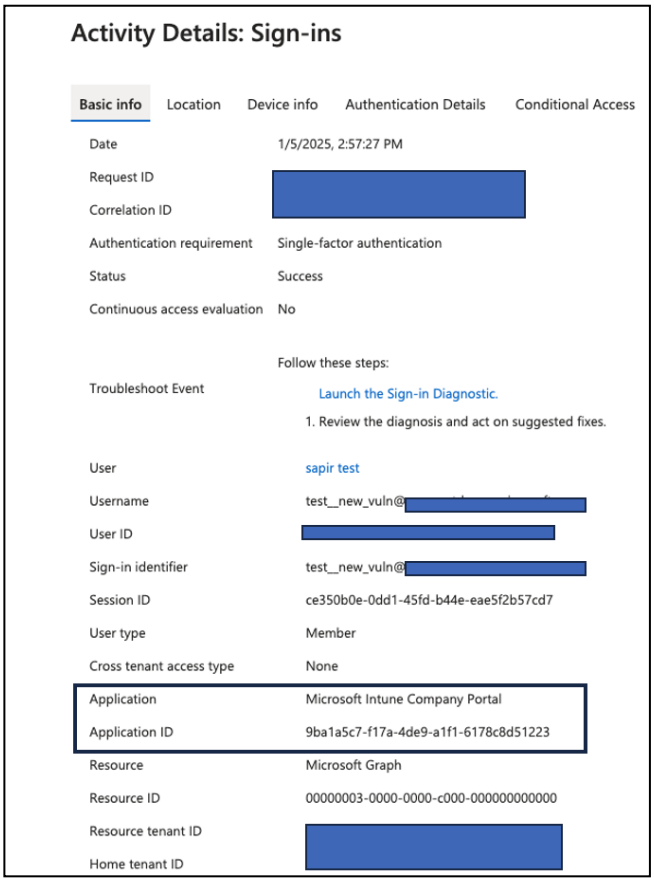
מדובר על ba1a5c7-f17a-4de9-a1f1-6178c8d512239, ערך שבעצם מייצג את האפליקציה המוכרת: "Microsoft Intune Company Portal" .

אני די בטוחה כי הסיבה שה-Client ID הזה פגיע, היא משום שעל מנת לרשום device חדש, הוא כמובן לא יכול להיות כבר Compliant, לכן כאשר מזדהים אל ה-Client הזה, מדלגים בבדיקה על ה-CAP המדובר.

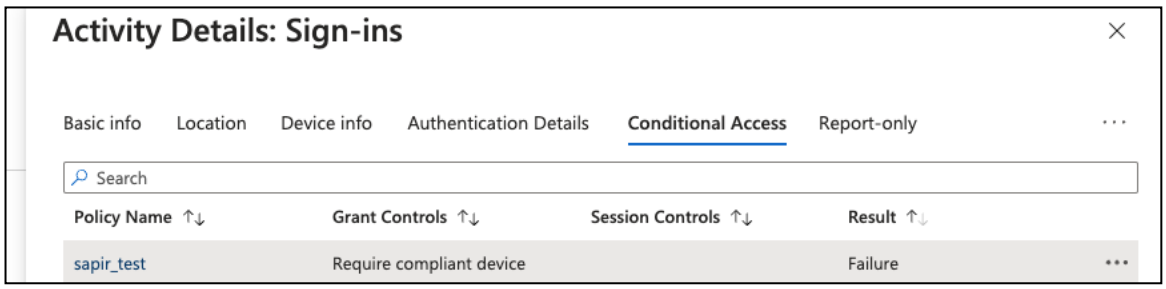
אז הרעיון הוא להזדהות עבור ה-Client ID שציינתי, (חשוב להבהיר כאן, המתקפה רק עוקפת את ה-CAP המדובר, אבל אתם עדיין צריכים את ה-Creds של המשתמש על מנת לבצע אותה). לאחר ההזדהות , נוכל להשתמש בעובדה שה-Client ID הזה הוא חלק מאותו ה-FOCI של Microsoft azure powershell (שהוא ה-Client בו נשתמש כאשר נרצה להשתמש ב-RoadRecon, AADInternals, MSGraph וכל החברים), ולהחליף את ה-Token שלנו ב-Token ל-Microsoft Azure Powershell. ובהתאם להרשאות המשתמש, נוכל לבצע פעולות או לקרוא מידע שמעניין אותנו :)

בקיצור, מעקף סופר מגניב לדעתי, דרך נחמדה לזהות אותו, היא לאתר התחברות מוצלחת, כאשר בחלק של CAP נראה כישלון.

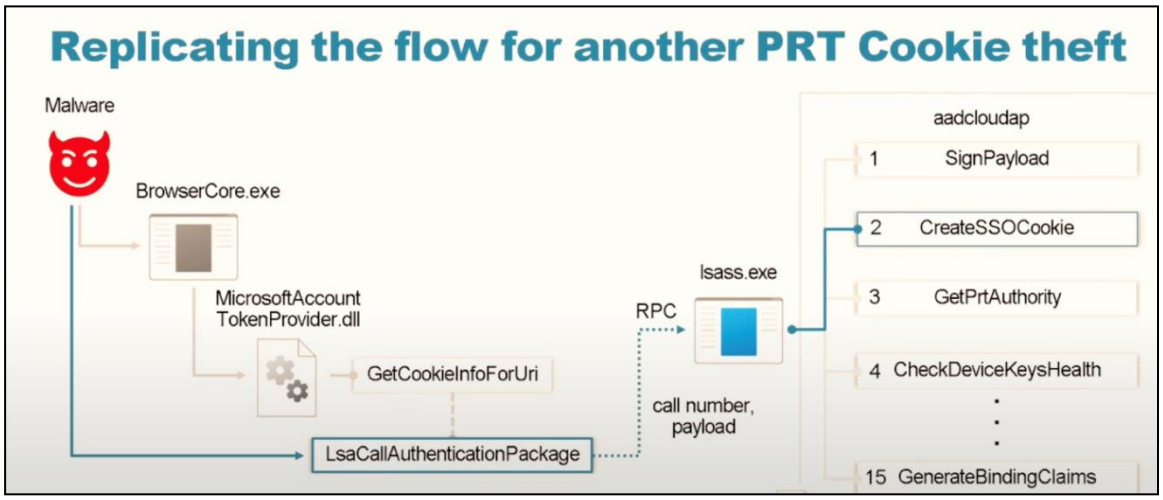
כי כידוע לנו, על מנת לעבור את ה-CAP, אנו צריכים לא להיכשל באף אחד מהם, אין היררכיה או תיעדוף. זאת אומרת, שאם ראינו כישלון אחד לפחות, ההתחברות הייתה אמורה להיכשל. הנה דוגמא, כאן נוכל לראות את האפליקציה המדוברת, וכמובן שההתחברות היתה מוצלחת:



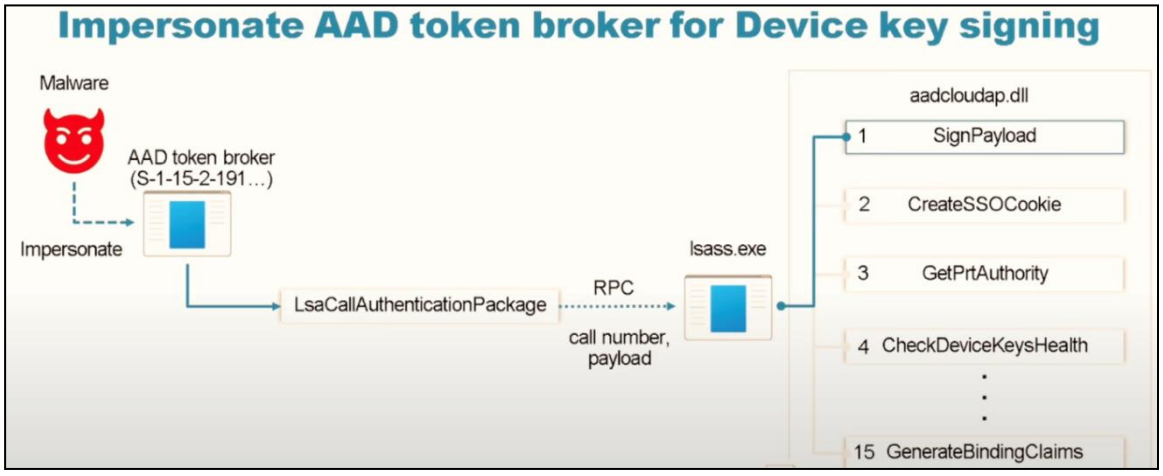
וכאן נוכל לראות את הכישלון ב-CAP:



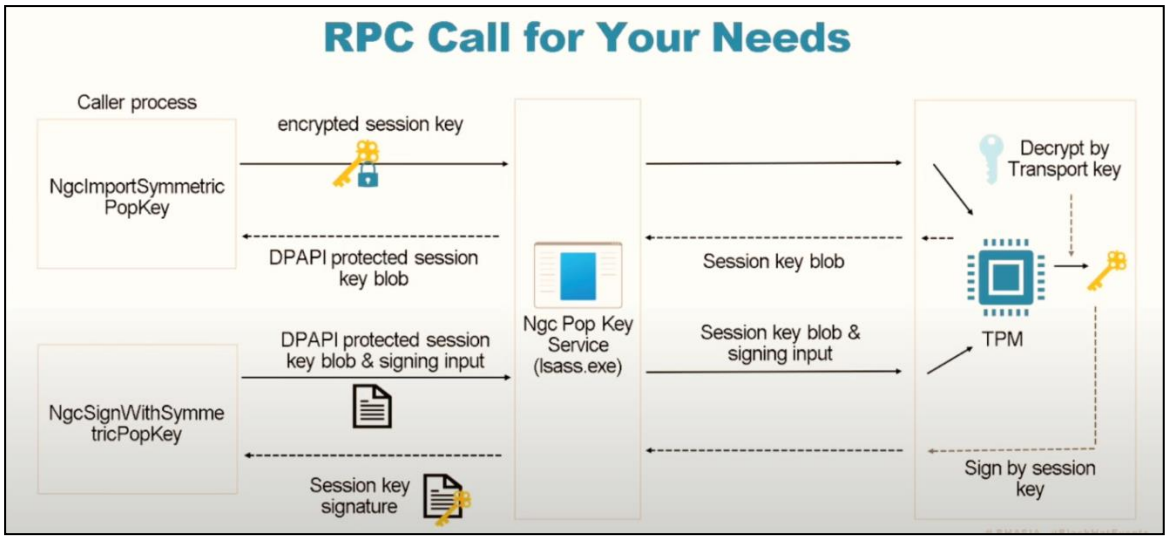
עוד מעקף שממש אהבתי, מצליח בעצם לגנוב PRT, וגם לחתום אותו עם ה-Session Key, פעולה שמתבצעת באמצעות ה-TPM מבלי להיות פיזית על המחשב. השיטה מאוד דומה לשיטות שהצגתי במאמר שלי על PRT, שימוש בפונקציות בהן משתמש תוסף הדפדפן BrowserCore.exe. אני לא אחפור יותר מידי על ה- low level, אלא אשתמש בתמונה מתוך ההרצאה שמסבירה ממש נחמד את השיטה:



בגדול מה שקורה כאן, הוא שהתוקף קורא ישירות לפונקציה LsaCallAuthenticationPackage שנמצאת בתוך MicrosoftAccoutTokenProvider.dll, שהוא dll שנטען על ידי התוסף לדפדפן. הפונקציה הזו מדברת עם aadCloudap שהוא הפלאגין של Entra המשמש את Lsass. כך בעצם הוא קורא לפונקציה CreateSSOCookie ומקבל לעצמו PRT.



זוכרים שדיברנו על כך שאם יש Token binding נצטרך לחתום אותו עם מפתח ששמור ב-TPM? כאן בעצם מציגים קריאה לפונקציה SignPayload, שגם היא יושבת בתוך aadCloudAp, הפונקציה הזו חותמת את ה-Token עם ה-session Key. יש לציין שלצורך השימוש בפונקציה, עליך לבצע Impersonation ל-Process ספציפי בשם Microsoft.AAD.BrokerPlugin.exe. כמו כן אני מוסיפה כאן תמונה שמסבירה על הפונקציות הדרושות על מנת לבקש מה-TPM לפענח את ה-Session Key, כדי שנוכל לחתום איתו את ה-PRT ב-SignPayload:



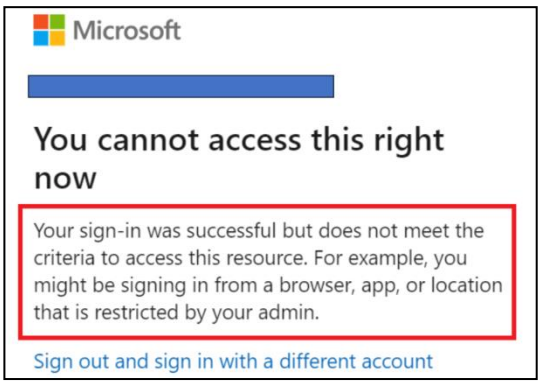
אני ממש ממליצה לצפות בהרצאה, היא ממש מרתקת! ☺

מבחינת זיהוי של מתקפה כזו, אני לא חושבת שיש דרך לזהות אותה באמצעות Entra, אבל אם יש לכם יכולות זיהוי חזקות ב-On-Prem , הייתי מסתכלת מי משתמש בפונקציות האלו, או לחילופין אולי יהיה קל יותר לזהות את ה-Impersonation ל-Microsoft.AAD.BrokerPlugin.exe.

כתוקף, איך אוכל לדעת איזה CAP מכשיל אותי?

אז האמת שהשגיאות יכולות להיות דיי אינפורמטיביות, אשים כאן מספר דוגמאות. למשל, כאן ברור לנו שהמשתמש חייב להתחבר ממחשב מנוהל:

כאן נוכל להבין שכנראה נפלנו בחלק של conditions. זה לא יגיד לנו בדיוק על איזה תנאי נפלנו, אך יצמצם לנו את האפשרויות:



מה אני כמגן יכול לעשות?

קודם כל, לפני כל המתקפות מסובכות עם ה-PRT, ככל הנראה שבמידה ותוקף יצליח לעקוף את ה-CAP בסביבה שלך, זה בגלל בעיות קונפיגורציה. לכן אני אתחיל מלהמליץ על הכלי הנהדר הזה. המאפשר לכם לדמות כל מיני תרחישים של CAP, ולבחון את החוזק שלהם, ולוודא שמה שתכננתם לקנפג ב-CAP אכן עובד, ואין לכם שום טעויות קונפיגורציה. שנית, אני ממליצה להשתמש ב-Sign In logs על מנת לבדוק כשלונות ב-CAP. תוכלו לראות באמצעות הלוגים באיזו פוליסה נכשלה ההתחברות. בנוסף, התחברות מוצלחת עם CAP כושל , זה דבר מאוד מאוד חשוד.

בשיטה זו תוכלו לראות את המעקפים היותר מסובכים, אך מעקפים כמו שימוש ב-VPN כדי לעקוף Locations condition, הם הרבה יותר קשים לזיהוי. לצערי כיום אין דרך מאוד מוצלחת לזהות אותם, לכן ההמלצה היא לעשות CAP מוצלח והדוק ככל הניתן. ולא לשכוח MFA לכל המשתמשים!

סיכום

Conditional Access Policies הן כלי רב עוצמה שיכול להוסיף שכבת אבטחה איכותית על הסביבה שלנו.
אך עם זאת, את רובן ניתן לעקוף, עם מאמץ כזה או אחר, לכן חשוב להבין שהן לא מספיקות כדי להבטיח הגנה מושלמת על הסביבה, ועלינו לעשות כמיטב יכולתנו באמצעות אכיפת MFA על כל המשתמשים, פוליסת סיסמאות נכונה, והגברת המודעות למתקפות פישינג.

כן הייתי ממליצה לעקוב אחר המעקפים החדשים שיוצאים, שכן הרבה מהם כן משאירים סימנים בלוגים , ולפחות אותם יהיה מעניין לנסות לזהות. מקווה שנהניתם ושנפתח לכם התאבון! אם כן, קפצו לביביליוגרפיה :)

על המחברת

@sapirxfed - ביום יום עושה שטויות ב-CrowdStrike, ובהפסקות עושה retweet לכל דבר שקשור ל-AD או ל-AAD (טוב, ולאחרונה גם AWS וסתם דברים רנדומלים של ענן... ☺)

אוהבת לכתוב קוד, מנסה למצוא חולשות, ומעריצה שרופה של DigitalWhisper!

ביבליוגרפיה

הרצאה חמודה על CAE:

https://www.youtube.com/watch?v=m3309aUKET8

Bypassבאמצעות זיוף של Complaint device:

https://aadinternals.com/post/mdm/

ההרצאה המעולה על bypass באמצעות פונקציונאליות של תוסף הדפדפן:

https://www.youtube.com/watch?v=JItnI6b9DII

מעקף מעולה באמצעות client פגיע:

https://labs.jumpsec.com/tokensmith-bypassing-intune-compliant-device-conditional-access/#45;bypassing&%2345;intune&%2345;compliant&%2345;device&%2345;conditional&%2345;access/

הרצאה ארוכה אבל פשוט תענוג לצפות בה, גניבת PRT וקצת טעימה על WHFB:

https://www.youtube.com/watch?v=tNh_sYkmurI

דוקומנטציה של מיקרוסופט , תמיד טוב שיש:

https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-conditional-access-policies

מיסקונפיגורציות נפוצות:

https://trustedsec.com/blog/common-conditional-access-misconfigurations-and-bypasses-in-azure