digital whisper
Technologic Papers

Malware-less Persistence (Done) Right

מאת יהונתן אלקבס

הקדמה

עם עלייתן של מתקפות הפונות לניצול לוגי של מנגנונים פנימיים ואלמנטרים בתשתיות הסביבה הדומיינית, התחדדה ההבנה כי Active Directory מעולם לא נכתב בראיית Secure by Design אלא נדחף בעל כורחו למציאת פתרונות אבטחה עבור סט איומים שרק הלך וגדל עם התבגרותו בחלוף השנים.

מבט קצר על סט פרסרי הפרוטוקולים וה-APIs ש-AD מרכז, בצמוד לעובדה שהשירותים שהוא מייחצן חייבים להיות נגישים ו-trusted אל כל מכונה ברשת יתמצתו את ההסבר מדוע **ההגנה על AD היא עניין לא פשוט בכלל**. כראייה לכך ניתן לראות את המספר הרב של מוצרי אבטחה משלימים מחברות צד שלישי ופתרונות מבוססי Identity Security בְּיֶתֶר שְׂאֵת.

מתקפות כדוגמת Diamond\Golden\Silver Tickets ו-AdminSDHolder אשר הגיחו לפני פחות מעשור לאוויר העולם הכריחו את מגזר אבטחת המידע להכיר בעובדה כי טכניקות שרידות דלות אמצעים שאינן שורטות את הדיסק הן נחלת העולם החדש. הצורך בפריקת נוזקה בסביבות מרוחקות לשם הרצת קוד פשוט אינו נדרש עוד על מנת לשמר גישה עתידית אל אותן המערכות.

במאמר זה, אשר מהווה מקרה פרטי למאמר שפרסמתי במסגרת הגליון הקודם: One ACE to Rule Them All, מוצגים הרעיונות והמימושים של מתקפות מבוססות ACL-ים בדומיין לצורך שרידות ואחיזה רשתית בנוסף, נראה כיצד ניתן להטמיע את המנגנון בצורה מוסתרת כך שאפילו Domain Admins לא יוכלו לגלות אותו, ולבסוף נחליף כובעים ונבדוק מה הצד הכחול יכול לעשות (אם בכלל) בשביל להתגונן מכל אלו.

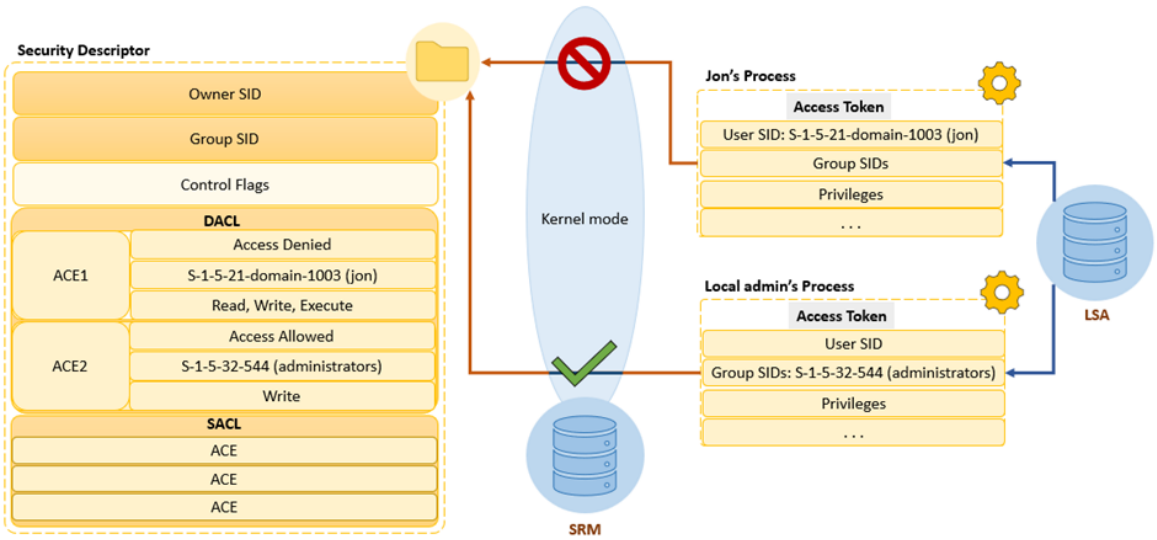
במידה ולא קראתם עוד את המאמר שהוזכר אני ממליץ שתתחילו ממנו. המאמר של היום מאוד טכני ומתמקד בעיקר בפן ההתקפי ללא הסבר מעמיק של הפן התיאורטי המלווה את הנושא. חוסר היכרות עם אבני הבניין עתידה לעשות את החיים הרבה יותר קשים למי שמעוניין להפיק את התשואה המרבית ממנו. אך עם זאת, כמובן שאלו רק מילים על דף דיגיטלי ואתם יותר ממוזמנים להמשיך את הקריאה.

אמל"ק

אובייקטים מאובטחים מוגדרים על ידי Microsoft כאובייקטים המחזיקים שדה Security Descriptor ב-Header שלהם. כחלק משדה זה, נמצא מערך DACL אשר מכיל את כללי הגישה אל האובייקט ומערך SACL אשר מגדיר כיצד נדרש (אם בכלל) להפיק לוגים כתוצאה מגישה לאותו אובייקט. כמעט כלל האובייקטים בדומיין (משתמשים, תיקיות שיתוף, OUs וכד') עונים על ההגדרה הנ"ל.

עם התחברות אינטראקטיבית של משתמש למערכת ההפעלה (בין אם בסביבה המקומית ובין אם בסביבה הדומיינית) מערכת ה-LSA לוקחת את ה-Logon Session בצמוד אל מידע אבטחתי נוסף אודות המשתמש ויוצרת את מבנה הנתונים של ה-**Access Token**. טוקן זה יצמד אל כל המשימות והתהליכים שירוצו תחת המשתמש על מנת לספק את סך ההרשאות הנדרשות עבורם כאשר הם באים במגע עם אובייקטים מאובטחים.

מערכת SRM הקרנלית היא האמונה מצד מ"ה על תהליך ההרשאה הנ"ל. בעת הגישה, ה-SRM מפרסרת את תוכן הטוקן של המשתמש ומשווה אותו אל שדות ה-ACEs שנמצאים בתוך ה-DACL כחלק משדה Security Descriptor של האובייקט המאובטח. אם המידע האבטחתי (SIDs שמאפיינים את המשתמש והקבוצות אליהן הוא שייך, ההרשאות שיש ברשותו וכו') מתאימים אל שדות של אחד ה-ACE - הגישה מאושרת. אם לא, מערכת ה-SRM תבטל את הפעולה. נשמע מורכב אבל תכל'ס זה די Straightforward.



עד כאן אמל"ק תיאורטי. מטרת המאמר הקרוב היא לתאר רעיון נישתי שנהגה על ידי מספר מועט של חוקרים והוא **קינפוג דקדקני של מערך ה-DACL באובייקטים מאובטחים נבחרים בדומיין לשם יצירת שרידות** ללא נוזקות והרצת קוד ב-AD ולאחר מכן להראות כיצד ניתן "**להחביא**" את המנגנון.

הערה א': למרות ששרידות על ידי שינוי קפדני של ACEs לאובייקטים נבחרים אכן מכריח רמה מסויימת של שינוי הסביבה (כל קוד בסופו של דבר יושב איפשהו), מבחינת עץ האובייקטים שמאפיין את מבנה הדומיין מדובר בשינוי מינימאלי. הרצת קוד (מבית) לא נחוצה כלל לתכלית האחיזה ואמצעי השרידות עתידים להישמר גם במקרים של כיבוי, החלפה, עדכון ושדרוג של השרתים והמכונות בדומיין. בהחלט מגניב.

הערה ב': כפי שכבר נאמר, תיאור מלא של כלל התיאוריה והסבר מפורט של מה הם בכלל Security Descriptor, DACL ועוד ראשי תיבות מפוצצים בסגנון מוצגים במאמר One ACE to Rule Them All.

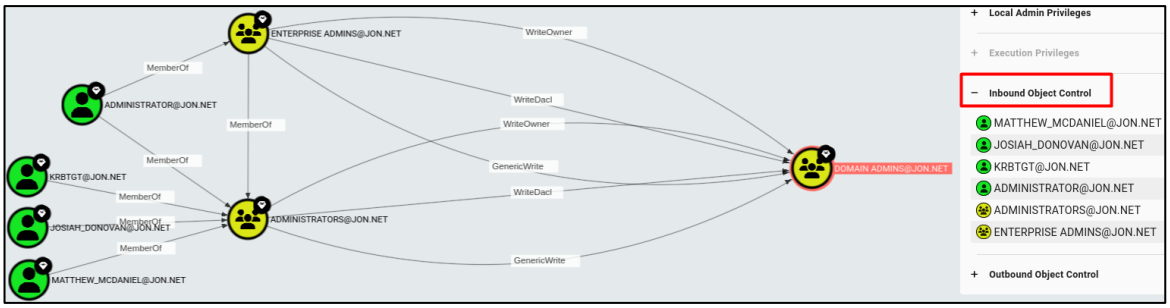
מי נגע ב-DACL שלי - Object Takeover

הרעיון של הטמעת ACEs עם שליטה לאובייקטים חזקים הוא בראש ובראשונה פיצ'ר ניהולי. כן כן, אם נסתכל למשל על שלבי התקנת Microsoft Exchange Server נוכל לראות כי היא משנה את כי היא משנה את ה-Domain Schema בשביל להוסיף את ה-Exchange Properties וקבוצות ייעודיות שלה אל AD. בדרך זו קבוצתExchange Enterprise Servers מקבלת **הרשאת WriteDacl על האובייקט של הדומיין עצמו.**

מה זה אומר? אממם, ובכן, לא הרבה, חוץ מכך שליטרלי **כל יישות** שתוסף לקבוצת Exchange Enterprise הלכה למעשה מקבלת גישה מלאה לכל מקום בדומיין. בהחלט מצב די הזוי ששרתי מייל מקבלים הרשאות שוות ערך לשרתי ה-DC אבל welp זה המצב וזו גם אחת הסיבות מדוע תוקפים אוהבים אותם כל כך.

כאשר דנים במימוש שרידות, לרוב יוצאים מנקודת הנחה שכבר יש שליטה כזו או אחרת על הדומיין וכעת מדובר על מנגנון Backdoor המאפשר גישה אל התשתית במועד מאוחר יותר. המכניזם נדרש להיות מוסתר שמה מחיקתו עתידה גם למנוע את האחיזה הקיימת וגם לסכן את כלל המבצע ההתקפי.

מפרספקטיבה אדומה אם נחפש היכן ***הכי משתלם*** לנו להחביא את ה-DACL Backdoor בסביבת AD, מעבר על Inbound Object Control Rights בממשק של BloodHound (בגרסה החדשה של הכלי) בהחלט עתיד ללמד אותנו לא מעט על סכמת השמות שנהוגה בדומיין עבור יישויות, היגיינת ה-DACL ומקומות טובים להחביא אובייקטים עם Nested Security Groups:



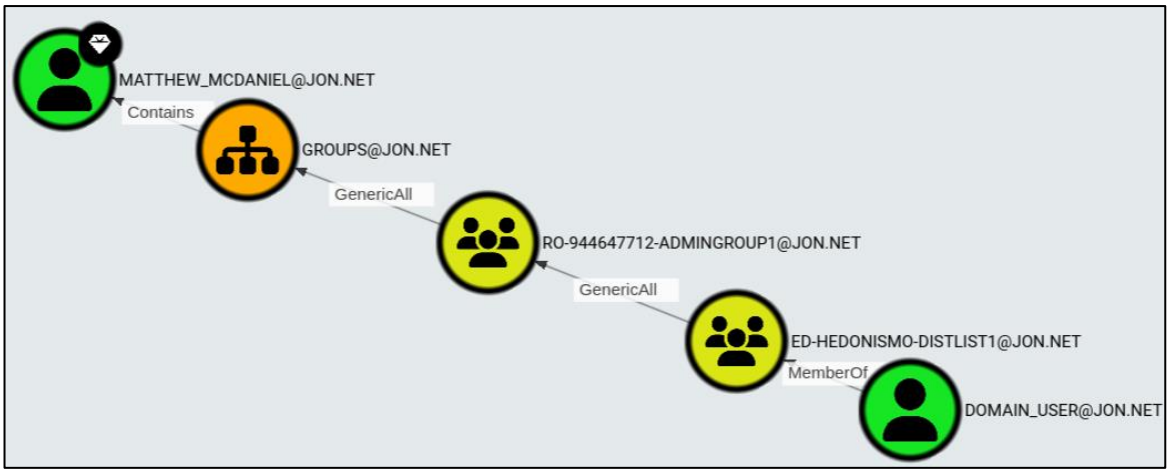
בשביל להבין את הסטטוס הנוכחי של כיצד מנהלי הרשת נוהגים לקנפג את ה-DACL בדומיין הנתקף, נקודה טובה להתחיל ממנה היא הסתכלות על קונפיגורציית קבוצת Domain Admins (או DA בקיצור ובשביל להישמע מגניבים) בממשק שניפתח מולנו. מדובר בנקודת מוצא מוצלחת מכיוון שקבוצת DA מוגדרת כקבוצה מוגנת (Protected Group) ועל פי כן ה-DACL שלה ישועתק מאובייקט ה-AdminSDHolder (אשר לו יש אינטרקציה עם קבוצות חזקות אחרות בדומיין).

נוכל להסיק מכך כי הטמנת Backdoor מבוסס ACEs בכל יישות דומיינית ב-Unrolled Members עם שליטה ב-DACL של קבוצת ה-DA יספקו לנו גישה חזקה על הקבוצה גם במועד מאוחר יותר.

אם נעקוב אחר הגרף ממקודם הרי שאם נשנה את ערכי ה-DACL של האובייקטים של Matthew_mcdaniel או של Jonsiah_donovan (פינה שמאל למטה) כך שיהיו ניגשים על ידי אובייקט אחר שברשותינו (אסביר עוד מעט כיצד ניתן לבצע זאת) נוכל להתמיע את השרידות בצורה יחסית שקטה.

קרי, **כמה שהאובייקט רחוק יותר בשרשרת השליטה מאובייקט היעד שלנו** (קבוצת DA) **כך הסיכוי שהצד הכחול ייחס לו חשיבות פוחת**. מה הכוונה? מתוך הנחה שהדומיין במצבו הנוכחי מקונפג "כמו שצריך", ככל שנבחר באובייקט בעל שליטה עקיפית רחוקה יותר מאובייקט המטרה שלנו כך הסיכוי לניטור מקיף סביבו יורדת.

על ידי משחק קצר עם ממשק BloodHound נוכל להסתכל לאילו אובייקטים יש גישה אל אותו Matthew_mcdaniel באמצעות חיפוש של יישויות המכילות ACEs מסוג GenericAll עליו. לשם הדוגמה נסתכל על ה-attack path הבא:



אם נביט על האובייקט של המשתמש החזק Matthew_mcdaniel הרי שהפעולה של הטמנת השרידות ב-OU שקרוב אליו (Groups) יעורר משמעותית יותר חשד מאשר הטמנת ה-Backdoor באובייקט אקראי של משתמש בשם domain_user (אותו יצרתי במכוון בשביל להמחיש שמדובר ב"סתם" משתמש נורמטיבי בדומיין).

אם נשלב את שני הגרפים שהצגנו הרי **שהטמנת ה-ACE זדוני אחד באובייקט של Domain_user הלכה למעשה תספק לנו שרידות על קבוצת Domain Admins.**

חשוב לציין שהדוגמה עוסקת רק בקבוצת DA מכיוון שהיא קבוצה חזקה דיפולטיבית בדומיין ומוכרת לכלל הקוראים. זה רק מן ההיגיון שבתרחישי תקיפה אמיתיים, לאחר למידת הסביבה בצורה מקיפה, יבחר אובייקט (או אובייקטים) אחר לביצוע Object Takeover המהולל.

כמה מסובך זה כבר להוסיף ACE לאובייקט שנמצא בשליטנו?

ת'אמת? כלל וכלל לא. במידה וביצענו את השלב של Privilege Escalation בצורה טובה ואנחנו רצים תחת משתמש חזק - מדובר בפקודה אחת בלבד. למעשה, צריך רק אחת מ-2 הרשאות ספציפיות על מנת לשנות לאובייקט את שדות ה-Security Descriptor. הראשונה היא **WriteDacl** אשר מאפשרת את עריכת ה-DACL וההרשאה השנייה היא **WriteOwner** מכיוון שלבעל האובייקט יש דיפולטיבית את הרשאת GenericAll על האובייקט עצמו.

יש שני כלים מוכרים שעושים את המשחק עם DACL ו-ACEs להרבה יותר פשוט (לעומת האלטרנטיבה של פרוצדורה בכתיבת פקודות ב-PS טהור ועבודה עם כלי DS כדוגמת dsacls).

- הכלי הראשון והמוכר ביותר, הוא לא אחר מאשר PowerView מאת harmj0y (Will Schroeder) על בסיסו נכתב לא מעט מקוד המקור של SharpHound עצמו (הקולקטור לאסיפת המידע בדומיין עבור BloodHound).
- הכלי השני הוא RACE מאת Nikhil Mittal. אם כי נדרש לטעון בנוסף אליו את ספריית Active Directory Module (שנמצאת בכל DC או ניתנת להורדה וטעינה לחוד), מדובר בכלי מאוד שימושי עם יכולות מבריקות שחבל שאינו מוצג ומתוחזק יותר.

כמובן שקיימים יותר מ-2 כלים למשימה אבל לצורך המאמר ועל מנת להדגים את האימפלמנטציה של הטמעת ACEs מסוגים שונים בפרק הקרוב אעשה שימוש ב-PowerView.

בשביל להוסיף למשתמש הפשוט שלנו (Domain_user) הרשאה לאיפוס סיסמה על אובייקט של משתמש חזק (Domain_admin) כל שנדרש הוא לטעון את הספריה לזכרון ולהריץ את הפקודה Add-DomainObjectAcl עם הפרמטרים הנכונים. נוכל לוודא כי הפעולה עבדה על ידי פילטור הפלט של פקודת Get-DomainObjectAcl. כדי לחזור למצב הקודם ולבטל את ה-ACE שהוספנו נשתמש בפקודת Remove-DomainObjectAcl.

כל הבלאגן נראה פחות או יותר ככה בתמונה אחת (מצרף את הדגל של Verbose לטובת השלם ההסבר):

```
PS C:\Users\domain_admin\Desktop> . .\powerview.ps1
PS C:\Users\domain_admin\Desktop> Get-DomainUser -Identity "CN=domain_user,CN=Users,DC=JON,DC=NET" -Properties objectsid

objectsid
S-1-5-21-1403467943-1367565381-54031474-1107


PS C:\Users\domain_admin\Desktop> Get-DomainObjectAcl -Identity "CN=domain_admin,CN=Users,DC=JON,DC=NET" | Where-Object { $_.SecurityIdentifier -eq 'S-1-5-21-140...1107' }
PS C:\Users\domain_admin\Desktop>
PS C:\Users\domain_admin\Desktop> Add-DomainObjectAcl -TargetIdentity "CN=domain_admin,CN=Users,DC=JON,DC=NET" -PrincipalIdentity "domain_user" -Rights ResetPassword -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC01.JON.NET/DC=JON,DC=NET
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|(samAccountName=domain_user)(name=domain_user)(displayname=domain_user))))
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC01.JON.NET/DC=JON,DC=NET
VERBOSE: [Get-DomainObject] Extracted domain 'JON.NET' from 'CN=domain_admin,CN=Users,DC=JON,DC=NET'
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC01.JON.NET/DC=JON,DC=NET
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(distinguishedname=CN=domain_admin,CN=Users,DC=JON,DC=NET)))
VERBOSE: [Add-DomainObjectAcl] Granting principal CN=domain_user,CN=Users,DC=JON,DC=NET 'ResetPassword' on CN=domain_admin,CN=Users,DC=JON,DC=NET
VERBOSE: [Add-DomainObjectAcl] Granting principal CN=domain_user,CN=Users,DC=JON,DC=NET rights GUID '00299570-246d-11d0-a768-00aa006e0529' on CN=domain_admin,CN=Users,DC=JON,DC=NET
PS C:\Users\domain_admin\Desktop> Get-DomainObjectAcl -Identity "CN=domain_admin,CN=Users,DC=JON,DC=NET" | Where-Object { $_.SecurityIdentifier -eq 'S-1-5-21-140...1107' }


ObjectDN                : CN=domain_admin,CN=Users,DC=JON,DC=NET
ObjectSID               : S-1-5-21-1403467943-1367565381-54031474-1108
ActiveDirectoryRights   : ExtendedRight
ObjectAceFlags          : ObjectAceTypePresent
ObjectAceType           : 00299570-246d-11d0-a768-00aa006e0529
InheritedObjectAceType  : 00000000-0000-0000-0000-000000000000
BinaryLength            : 56
AceQualifier            : AccessAllowed
IsCallback              : False
OpaqueLength            : 0
AccessMask              : 256
SecurityIdentifier      : S-1-5-21-1403467943-1367565381-54031474-1107
AceType                 : AccessAllowedObject
AceFlags                : None
IsInherited             : False
InheritanceFlags        : None
PropagationFlags        : None
AuditFlags              : None


PS C:\Users\domain_admin\Desktop> Remove-DomainObjectAcl -TargetIdentity "CN=domain_admin,CN=Users,DC=JON,DC=NET" -PrincipalIdentity "domain_user" -Rights ResetPassword -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC01.JON.NET/DC=JON,DC=NET
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|(samAccountName=domain_user)(name=domain_user)(displayname=domain_user))))
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC01.JON.NET/DC=JON,DC=NET
VERBOSE: [Get-DomainObject] Extracted domain 'JON.NET' from 'CN=domain_admin,CN=Users,DC=JON,DC=NET'
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC01.JON.NET/DC=JON,DC=NET
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(distinguishedname=CN=domain_admin,CN=Users,DC=JON,DC=NET)))
VERBOSE: [Remove-DomainObjectAcl] Removing principal CN=domain_user,CN=Users,DC=JON,DC=NET 'ResetPassword' from CN=domain_admin,CN=Users,DC=JON,DC=NET
VERBOSE: [Remove-DomainObjectAcl] Granting principal CN=domain_user,CN=Users,DC=JON,DC=NET rights GUID '00299570-246d-11d0-a768-00aa006e0529' on CN=domain_admin,CN=Users,DC=JON,DC=NET
True
PS C:\Users\domain_admin\Desktop> Get-DomainObjectAcl -Identity "CN=domain_admin,CN=Users,DC=JON,DC=NET" | Where-Object { $_.SecurityIdentifier -eq 'S-1-5-21-140...1107' }
PS C:\Users\domain_admin\Desktop>
```

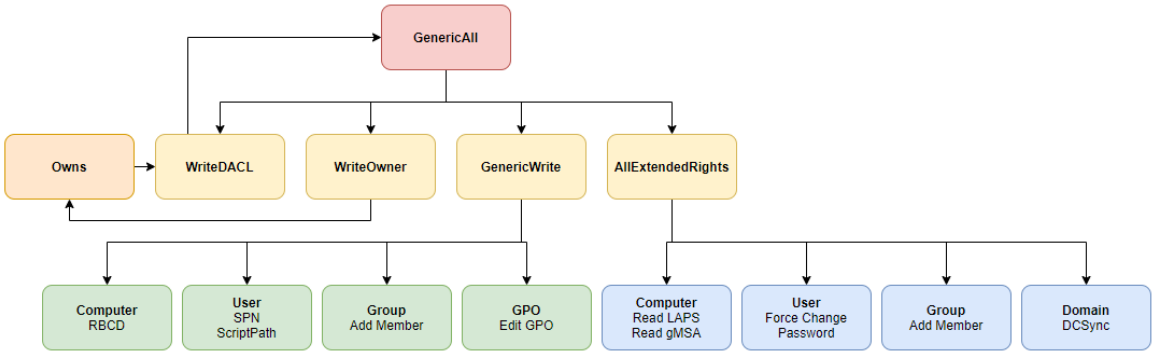
מצטער מראש שהצילום מסך ממש ממש לא נוח אבל עם קצת zoom in, כיווץ של העיניים ומשחק עם העכבר אפשר לראות מעולה את כלל הפלט.

מה אנחנו רואים שם? לאחר טעינה של הספריה לזיכרון, מורצת פקודת Get-DomainObjectAcl על משתמש domain_admin עם פילטור ה-SID של domain_user בשביל להמחיש שלפני הוספת ה-ACE לא היו לאובייקט הרשאות עליו. את ה-SID של המשתמש ניתן להשיג באמצעות פונ' Get-DomainUser או על ידי קצת PS shananigans בסגנון:

```
{ $_.SecurityIdentifier -eq (New-Object system.Security.Principal.NTAccount('domain_user')).
Translate([System.Security.Principal.SecurityIdentifier]).Value }
```

לאחר הרצת הפקודה Add-DomainObjectAcl עם הערך ResetPassword נוכל לראות כי כן יחזור אלינו פלט אם נריץ את Get-DomainObjectAcl בשנית.

אם שמתם לב, למרות שביקשנו מ-PowerView להוסיף לנו את ההרשאה של ResetPassword על אובייקט משתמש, קיבלנו דווקא את ההרשאה של **ExtendedRight** על האובייקט. זאת מכיוון שלא קיים ACE שמדבר במפורש על איפוס סיסמה לאובייקט אלא יש ACE המבטא סט פעולות בהתאם לסוג האובייקט הנבחר:



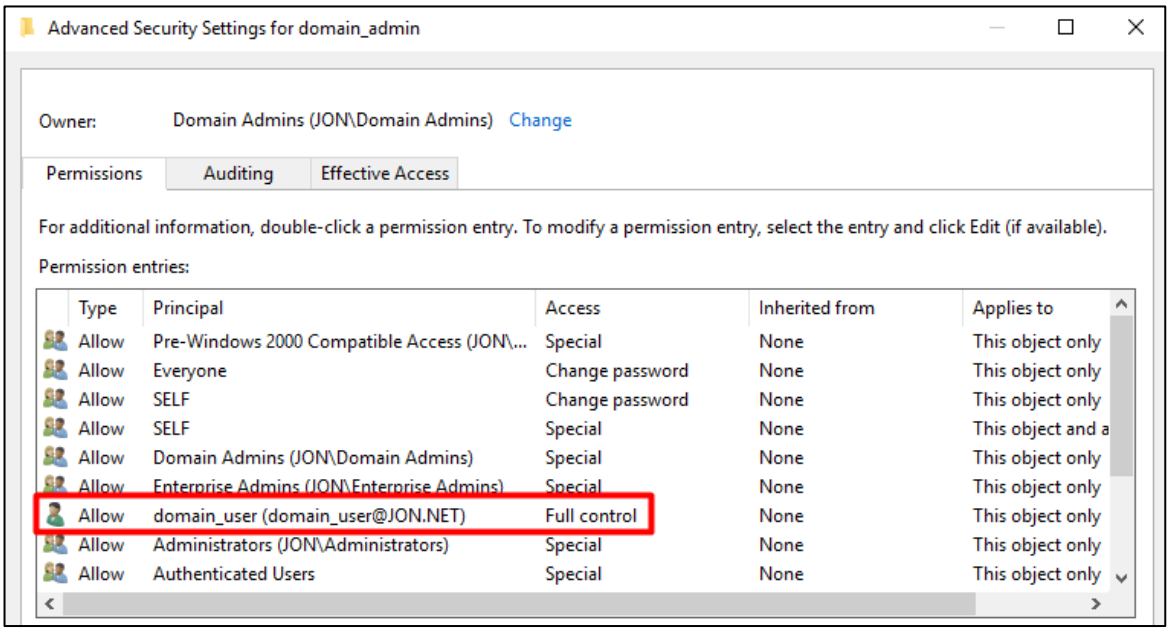
[מקור - https://ppn.snovvcrash.rocks/pentest/infrastructure/ad/acl-abuse]

במאמר One ACE to Rule Them All עברנו ממש על קוד המקור של Get-DomainObjectAcl ושל Get-DomainSearcher (שניתן לראות שקוראים להן לא מעט כחלק מהפלט של דגל verbose בצהוב) על מנת להבין כיצד הקסם של מציאת האובייקט ופירסור ה-ACEs שברשותו קורה מאחורי הקלעים.

בצורה דומה, אם נרצה להכניס שרידות חזקה (אך בולטת) יותר, נוכל להוסיף ACE עם **FullControl**:

```
Add-DomainObjectAcl -TargetIdentity "CN=domain_admin,CN=Users,DC=JON,DC=NET" -PrincipalIdentity "domain_user" -Rights All
```

כיצד בצד הכחול יראו את הדברים? אם ה-ACE הנ"ל נוסף בהצלחה ל-DACL של המשתמש domain_admin, באמצעות ממשק ה-GUI של ADUC נוכל לראות אותו כך:



אבל יונתן||| כבר יש מודעות גדולה לשיטות האלו ואפילו במאמר הקודם אתה בעצמך הזכרת לא מעט כלים כדוגמת PingCastle, PlumHound, GoodHound, Forest Druid ו-ADACLScanner שיכולים למצוא את אותן הרשאות מתירניות מבוססות ACL. מה הטעם במנגנון הישרדות הזה אם ניתן לגלות אותו בקלות על ידי שאילתת LDAP פשוטה?

ובכן, השאלה בהחלט במקום אבל היא מניחה מראש 3 נקודות מרכזיות:

1. כל הצוותים הכחולים אכן מודעים לטכניקות הנ"ל.
2. יש להם את הכלים והפתרונות בשביל למצוא, לנתח ולהסיר את אותן שיטות (ב-scale!)
3. בתור תוקפים לא הצלחנו להטמיע את השרידות באופן עמוק מספיק כך שתשתלב עם הסביבה בצורה שלא מעוררת חשד.

אם כי שתי הנקודות הראשונות בהחלט לא תלויות בנו כנציגים של הצד האדום, הנקודה השלישית לגמרי בשליטתנו. ולכן, השאלה שצריכה להישאל היא: **כיצד ניתן להפוך את כל מנגנון השרידות שהוצג להרבה יותר קשה לגילוי ו-anti-audit מטבעו?**

השלב הבא - Stealth Persistence

כזכור, הדרך בה מערכת ה-SRM בודקת את סדר ה-ACEs שנמצאים ב-security descriptor של כל אובייקט בזמן שאובייקט אחר ניגש אליו היא **בסדר קנוני:**

DACL
Explicit Deny ACEs
Explicit Allow ACEs
Inherited Deny ACEs from parent object
Inherited Allow ACEs from parent object
Inherited Deny ACEs from grandparent object
Inherited Allow ACEs from grandparent object
. . .

מכיוון ש-Explicit Deny מקבל תיעדוף על פני Explicit Allow, עם הוספת ACE מנוסח היטב נוכל למנוע את האנומרציה של ה-Security descriptor אפילו על ידי תהליכים הרצים תחת קבוצות חזקות.

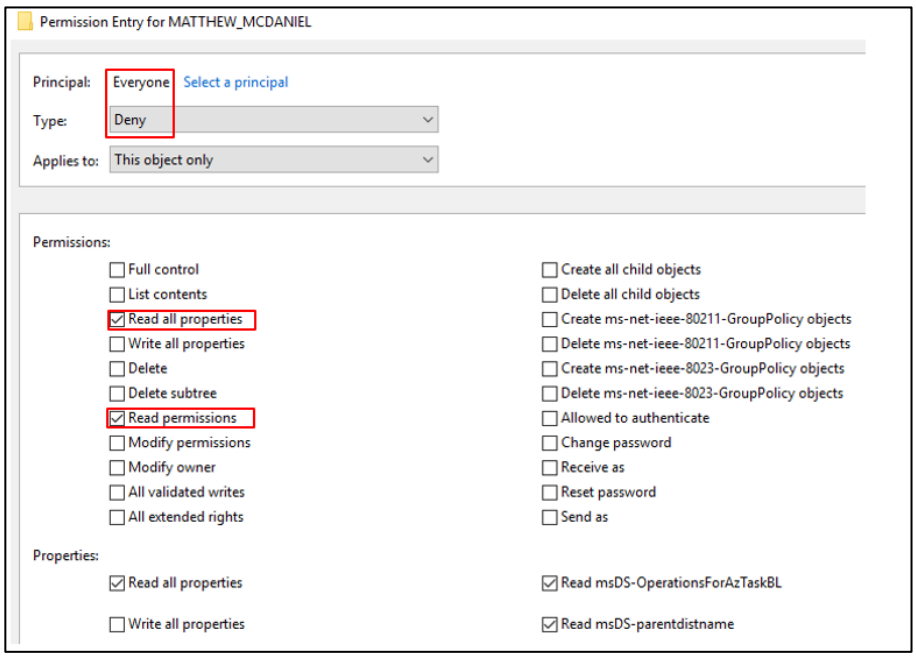
בנוסף, שווה לזכור שיש לבעלים של האובייקט את הרשאת GenericAll על אותו אובייקט שהם יצרו \ קיבלו בעלות עליו. **בדיקת ה-Owner עוקפת כל Explicit ACEs שנמצאים בראש ה-DACL**. מסיבה זו, במידה ונרצה להחביא את ה-DACL מהקבוצה שהיא הבעלים של כלל האובייקט נצטרך קודם כל לשנות את ה-owner של האובייקט.

בהתאם לכל אלו, על מנת להסתיר את כלל ה-DACL של אובייקט מאנומרציה של משתמש חזק אנחנו נדרשים לבצע **2 שלבים**:

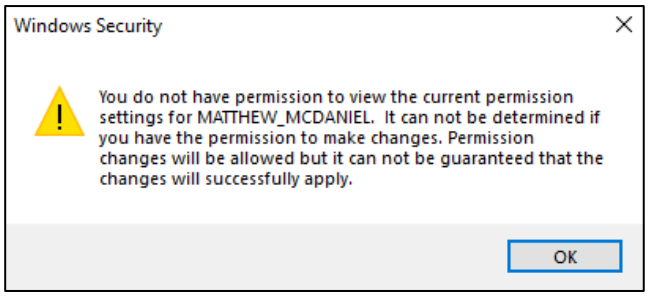
1. לשנות את ה-owner של האובייקט אל יישות שבשליטתנו
2. להוסיף ACE של Explicit Deny Read - Everyone

וזהו, הרעיון מאוד פשוט קונספטואלית. אדגים את המימוש של המתקפה דרך ה-GUI של ADUC על אובייקט משתמש בשם Matthew_mcdaniel בשביל להמחיש את הרעיון ואת העובדה שכעת אפילו Domain Admin לא יוכל לראות (!) את ה-DACL של האובייקט.

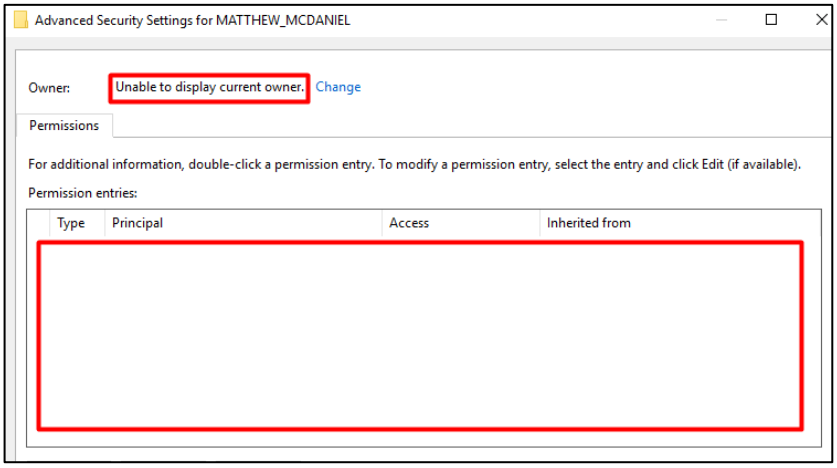
בהתאם למתכון, נשנה את ה-Owner של היוזר אל המשתמש שבשליטתנו ואז ניצור את ה-ACE הבא:



לאחר אישור, במידה וננסה לגשת אל ה-DACL של האובייקט עם משתמש בקבוצת Domain Admins נקבל את השגיאה הבאה:



והממשק יציג את ה-DACL בצורה ריקה לחלוטין:



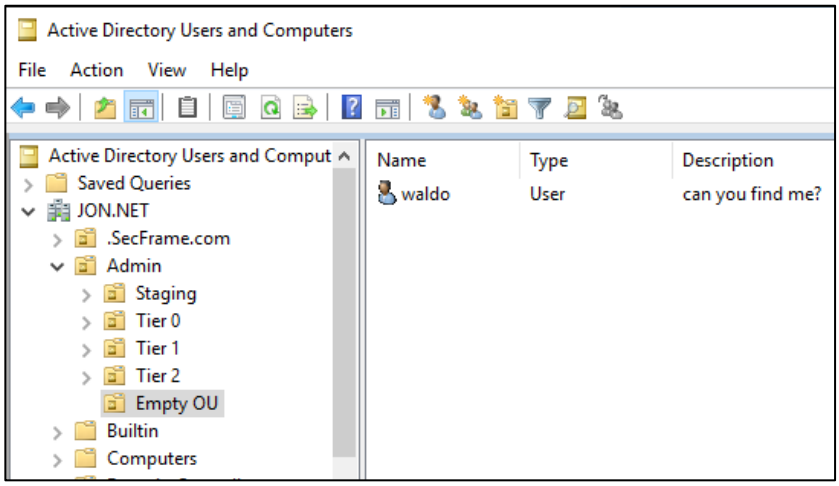
בדרך הזו, על מנת שמשתמש חזק יוכל לבצע אנומרציה על האובייקט של Matthew_mcdaniel הוא יידרש **קודם כל לקחת עליו בעלות** (מה שכבר הנראה אף כלי הגנתי לא יעז לעשות!) ורק אז לסקור את ה-DACL וה-ACEs שברשותו.

כלומר הצלחנו להחביא ה-security descriptor של אובייקט אבל מכיוון שלא ניתן לראות את ה-DACL שלו זה בפני עצמו מחשיד וכנראה שתקפוץ התראה על כך. אבל מה אם יכולנו פשוט "**להעלים**" את המשתמש? כלומר שמשתמש חזק לא יוכל בכלל לדעת שהמשתמש קיים? גם אם הוא ירוץ תחת הרשאת SYSTEM וישלח שאילתת LDAP במיוחד אליו?

מדובר בפרוצדורה טיפה יותר מורכבת מחסימת גישה אל ה-DACL אבל ניתן לבצע אותה ב-4 השלבים הבאים:

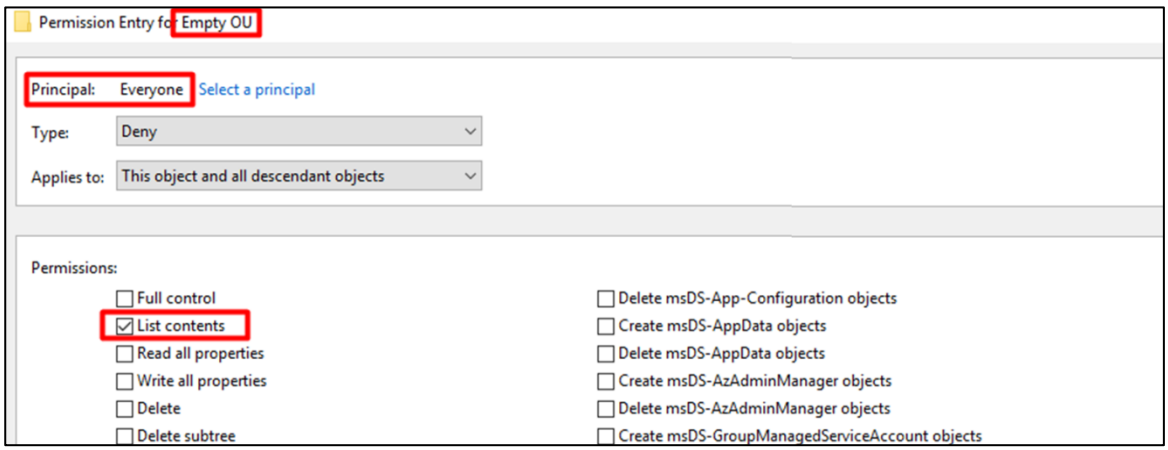
1. הוספת האובייקט שנרצה להעלים ל-OU (קיים או שיצרנו) ולחסום את ההרשאה להצגת תוכן ה-OU (List contents) אל קבוצת Everyone.
2. שינוי הבעלים של האובייקט לאובייקט עצמו
3. נתינת explicit control לאובייקט
4. חסימת כל ההרשאות של קבוצת Everyone לאובייקט ולכל מי שתחתיו.

נמחיש את הרעיון על קבוצת OU בשם "Empty OU" עם משתמש בשם "**waldo**" שחבר בקבוצת Domain Admins שיצרתי בדומיין:

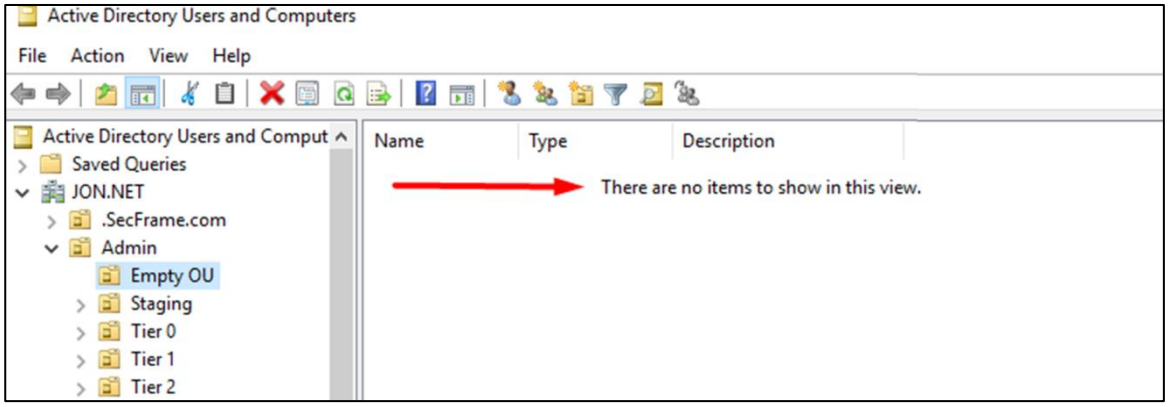


כלל ה-ACEs שעברו בירושה לאובייקט מסודרים ברמות לפי הסדר בו הורשו. כלומר גישה שהגיעה מירושה של אובייקט האב (*Empty OU*) ממוקמת לפני גישה שניתנה בירושה מאובייקט הסב (*Admin*) וכך הלאה. על ידי קינפוג של OU שמהווה את אובייקט האב של האובייקט שאנחנו רוצים להעלים אנו יכולים להחיל ACE עם Explicit Deny שיעקוף את כלל ה-ACEs של ה-OU עצמו ושל כל האובייקטים שמעליו. זאת פעולה חזקה מאוד שמאופשרת **בגלל הדרך בה Microsoft בחרו לממש את המנגנון של גישה לאובייקטים ולא ניתן לשנות אותו.**

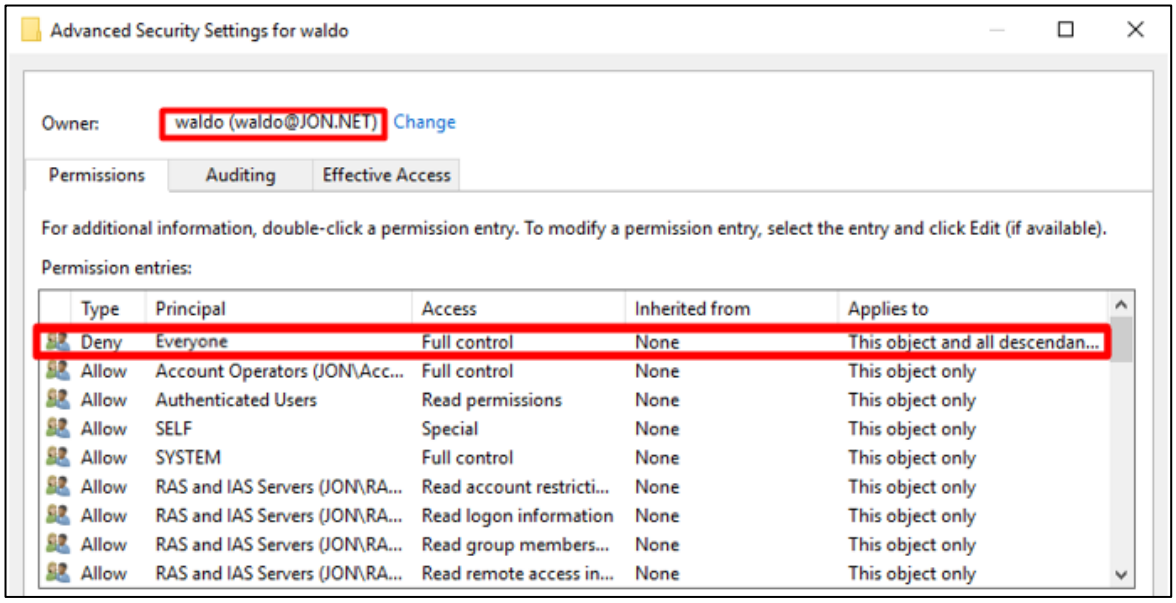
פעולת שינוי גישת ה-OU אליו הוא משתייך:



בצורה הזו ניתן "להעלים" אובייקט לגמרי ולא יהיה אפשר לתשאל את ה-OU לקיומו של האובייקט באמצעות LDAP ולא באמצעות הממשק גרפי עצמו של ADUC (שכאמור רץ תחת הרשאות מנהלן ב-DC):



כלומר, אנחנו יודעים שמשתמש waldo אמור להיות שם אבל לא ניתן לראות אותו. לאחר מכן נשנה את הבעלים של האובייקט לעצמו ונערוך את ה-DACL:

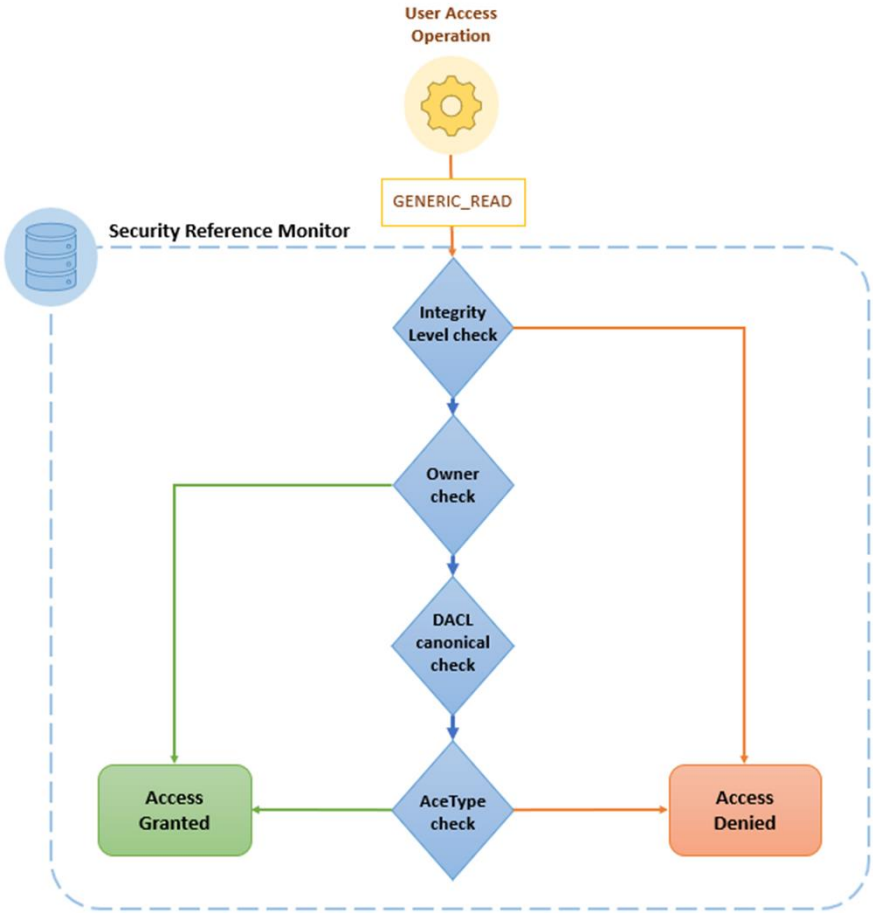


כל שנותר כעת הוא לבדוק שאנחנו אכן יכולים להתחבר באמצעות RDP או אימות אינטראקטיבי למשתמש Waldo בשביל אחיזה ב-DC:

```
PS C:\Users\domain_admin\Desktop\SysinternalsSuite> runas.exe /user:jon.net\waldo cmd
Enter the password for jon.net\waldo:
Attempting to start cmd as user "jon.net\waldo" ...
cmd (running as jon.net\waldo)

C:\Windows\system32>hostname && whoami && whoami /all | findstr "Domain Admins"
DC01
jon\waldo
JON\Domain Admins                           Group            S-1-5-21-1403467943-1367565381-54031474-512  Gr
```

למרות שעל פניו חסמו את האינטרקציה של האובייקט על **כלל המשתמשים בדומיין** (כולל הוא עצמו) ואנחנו עדיין יכולים להשתמש בו כרוכה בהבנה שהרשאת Owner (המספקת **GenericAll** דיפולטיבית על האובייקט) נבדקת על ידי ה-SRM **לפני שנבדק כלל שדה ה-DACL** שמכיל את אותו ה-ACE:



על מנת להמחיש את הנקודה, נפתח טרמינל כמשתמש SYSTEM, נטען את ספריית Powerview ונראה **שאפילו בהרשאות הכי גבוהות ב-DC לא ניתן לראות את המשתמש המוסתר**:

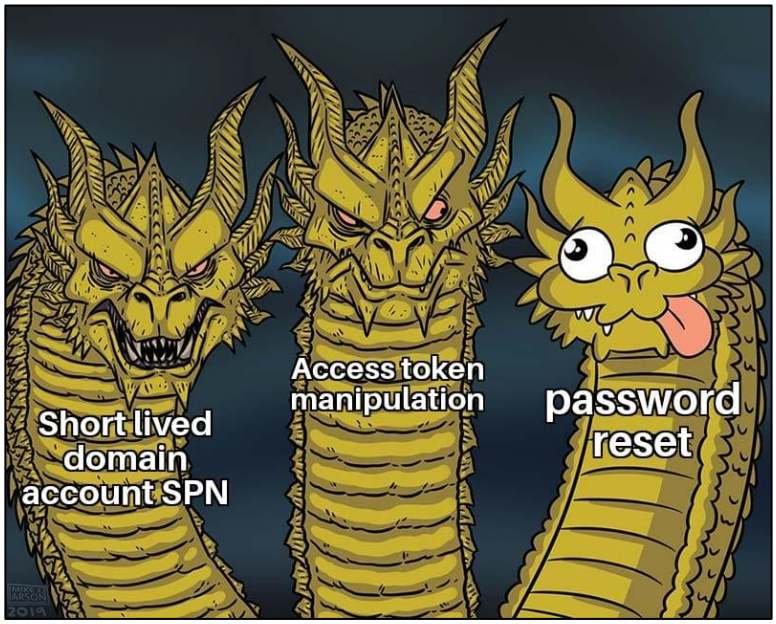
```
PS C:\users\domain_admin\desktop> whoami
nt authority\system
PS C:\users\domain_admin\desktop>
PS C:\users\domain_admin\desktop> . .\powerview.ps1
PS C:\users\domain_admin\desktop>
PS C:\users\domain_admin\desktop> get-domainuser -identity waldo
PS C:\users\domain_admin\desktop>
```

בצורה הזו כעת יש ברשותינו אובייקט משתמש חזק **שרק אנחנו מודעים לקיומו.**

אז איך בתור תוקפים עלינו להתנהל עם ה-Backdoor הזו? מעתה ואילך השימוש באובייקט לשם שרידות חייב להיות נמוך ככל הניתן, עם OpSec מקסימאלי ורק לשם קבלת אחיזה כי אנחנו לא רוצים לייצר יותר לוגים ממה שצריך עבורו. כמו כן, מכיוון שעדיין ניתן לראות את קיומו של ה-OU יש תיעדוף למציאת OU ריק מראש או יצירה של אחד (או כמה) עם ה-name-schame של הסביבה הקיימת.

ניתן להפוך את כלל המנגנון לשקט בהרבה על ידי שירשור של מספר אובייקטי proxy כאלה ביחד (OU בתוך OU, קבוצה מוחבאת שמסתירה קבוצה שמסתירה משתמש וכד'). בנוסף, לא נדרש להשאיר סט הרשאות מלא לאובייקט שבו תלויה השרידות אלא בסה"כ הרשאת WriteDacl או WriteOwner על אובייקט עם שרשרת שליטה לאובייקט המטרה שלנו (כדוגמת משתמש בקבוצת DA) בדומיין.

שווה גם לומר כמה מילים על **הדרך בה אנחנו מתכננים להשיג את האחיזה בחזרה על אותו אובייקט מטרה**. למשל אם מדובר באובייקט משתמש, כנראה שנצטרך את פרטי ה-Credentials שלו כי לאפס לו את הסיסמה בצורה ישירה זו פעולה רועשת פיצוציסמם (ואין משהו שמעצבן יותר את המשתמשים על צוות ה-IT מאשר זה ששינו להם את הסיסמה בלי לומר מראש).



סודות ה-SACL (מה בכל זאת ניתן לעשות מהצד הכחול?)

אחד המאפיינים הכי ליבתיים של עולם אבטחת המידע הוא היותו **פאסיבי**. כלומר, הגנה מבוססת ריאקציה. אנחנו לא עושים כלום אם לא קרה כלום, פשוטו כמשמעו. זו קונספציה מחשבתית שחשוב להבין.

סריקות מחזוריות של מוצרי EDR\AV, פעולות אקטיביות לחיפוש תוקף ברשת (Threat Hunting) ותשלום עבור Threat Intelligence הן חלק בלתי נפרד מארסנל הכלים של צוות הגנה מודרני; אך התמונה הכללית נשארת זהה - לעולם לא נוכל לחזות מהיכן תגיח האנומליה הבאה.

אם לפשט את התוכן הפילוסופי עד כה למשפט קוהרנטי אחד הרי **שיצירת תמונת מצב עדכנית של הרשת בצמוד לניטור כל דבר שזז בתוכה הוא ההימור הכי טוב שיש לנו כרגע בשביל למנוע נזק ממתקפות סייבר**.

מהצד האדום, ניתן אומנם למחוק, לערוך או לכבות את מנגנוני אסיפת הלוגים עם גישה חזקה לרשת אך במערכות מקונפגות היטב ישנן פעולות שפשוט לא ניתן לבטל את יצירת הלוגים עבורן מבלי להקפיץ את כל צוות ה-SOC על הרגליים.

בפרק הקרוב אסביר את <u>השימוש הפרקטי ב-SACL</u> (כאמור System Access Control List) לשם אסיפת לוגים המעידים על עריכת ההרשאות של אובייקטים נבחרים בדומיין. **באמצעות SACL ניתן לנטר כל אובייקט עם Security Descriptor.** מדובר בחלק בלתי נפרד ממוצרי אבטחה רבים ומתהליכי הניטור שמבוצעים לשם אסיפת לוגים אל עבר ה-SIEM מעמדות הקצה.

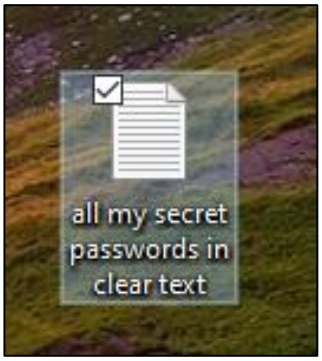
לכן, בפרק הקרוב נסתה קצת מהאג'נדה האדומה של המאמר ונסתכל על כיצד ניתן להשתמש ב-SACL הן ברמת הדומיין והן ברמת המחשב הבודד (קרי, standalone) לפעולות ניטור בשביל לשפוך קצת אור על יכולות שכולם יכולים לבצע בחינם כבר ממחר בשביל להעלות את רמת הניטור במקומות רגישים.

הערה: מכיוון שהמאמר עסק במתקפות מבוססות DACL, פתרון ההגנה שאציג יהיה מבוסס SACL. כמובן שיש פתרונות אבטחה רבים אחרים וכתיבת ספר חוקות על פי תרחישי תקיפה מוכרים (כדוגמת SIGMA ו\או Suricata לזיהוי התנהגויות אנומאליות ו-YARA לזיהוי קבצים חשודים) הוא יותר מנדרש.

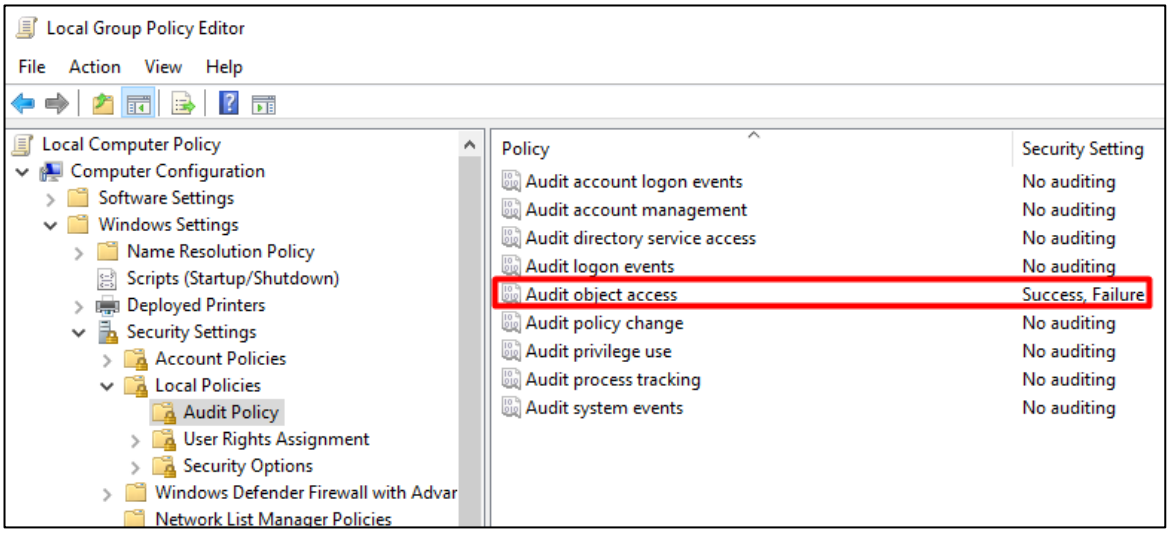
המטרה שלנו היא לזהות במהירות, בזול וביעילות פעילויות חריגות במכונה מבלי להתמקד בריצה של תהליכים ו-threads. הרעיון הוא די פשוט - באמצעות SACL ניתן לבצע Audit ל-ACEs לאובייקטים כדוגמת קבצים או מפתחות ברגיסטרי ולתעד אם גישה אליהם צלחה או לא.

בעוד שאת DACL ניתן להקביל לגדר המגנה על הגישה לאובייקט, ולכן צריך למצוא דרך לקפוץ או לחפור מתחתיה, מ-SACL לא ניתן לחמוק - אם הגדרנו לוג על כל קריאה של קובץ מפתח ה-SSH שלנו - יופק לוג. אז בהינתן הקינפוג הנכון, <u>האם יתכן לייצר log events שיצביעו על כך שאנחנו כבר compromised?</u>

בשביל להמחיש את הרעיון נתחיל ממשהו פשוט - ניטור אחר פתיחה של קובץ decoy *סופר סודי* בשולחן העבודה שלנו:



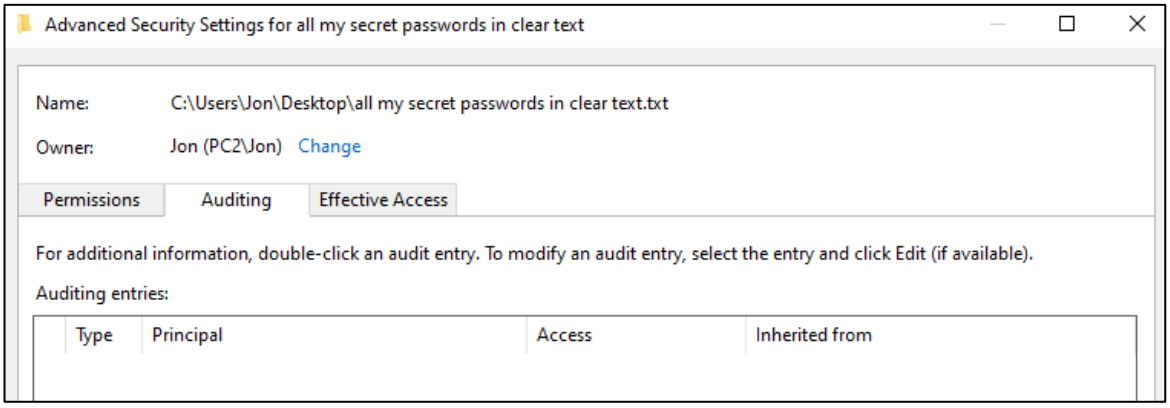
השלב הראשון יהיה להגדיר ב-Local Group Policy שאנחנו מעוניינים לנטר גישה אל אובייקטים:



לאחר מכן, באמצעות ה-GUI נכנס להגדרת ה-SACL של אובייקט הקובץ על ידי:

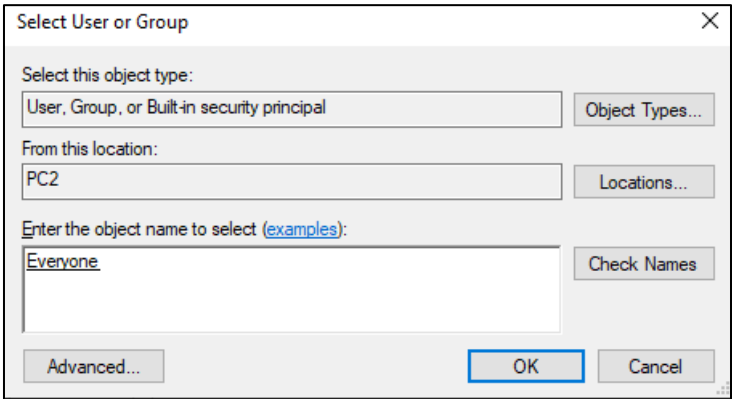
Propertise → Security → Advanced →Auditing

כעת, לפני הוספת ה-ACE, ניתן לראות שה-SACL ריקה:

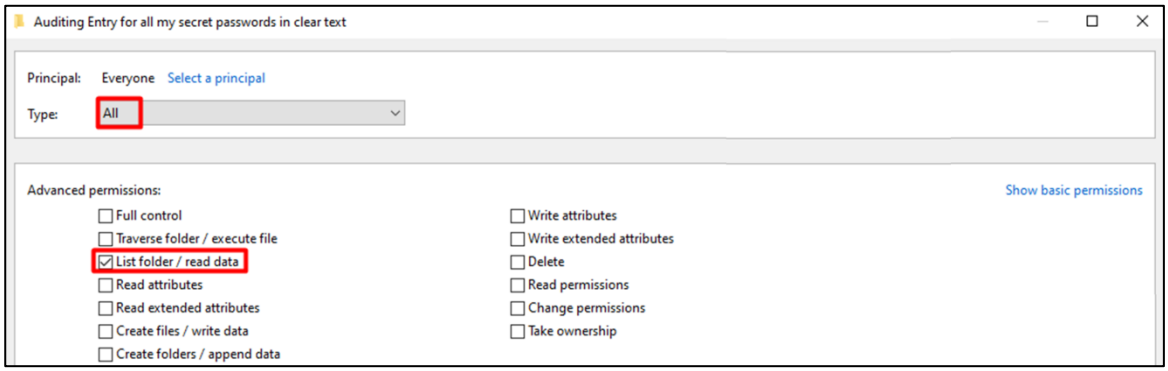


על ידי לחיצה על Add ואז על Select a principal נגיע לחלון של הוספת היישויות עליהן נרצה לבצע ניטור.

מכיוון שאנחנו מעוניינים לנטר כל גישה לקובץ נבחר ב-Everyone:

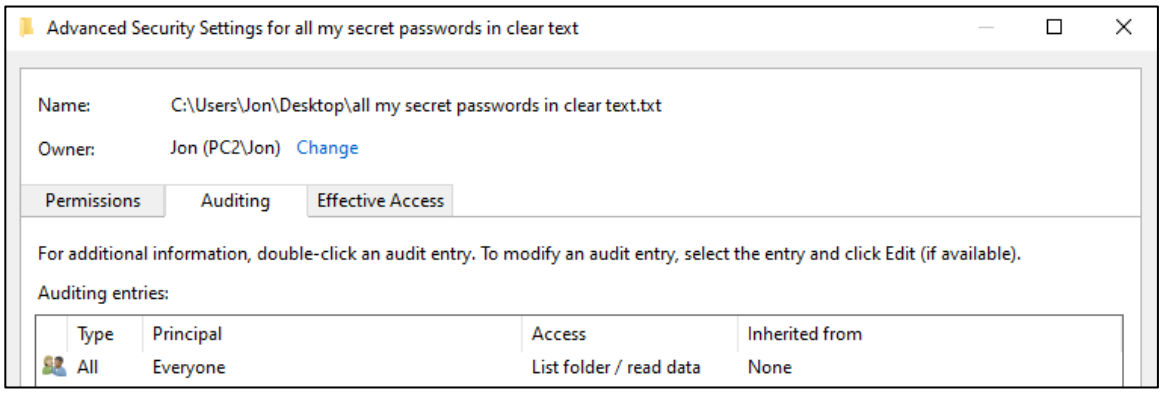


עם אישור נקבל חלון חדש. נעבור אל Show advanced permissions ונסמן את ההרשאות של קריאה בלבד **הן על גישה מוצלחת והן על גישה כוזבת**:



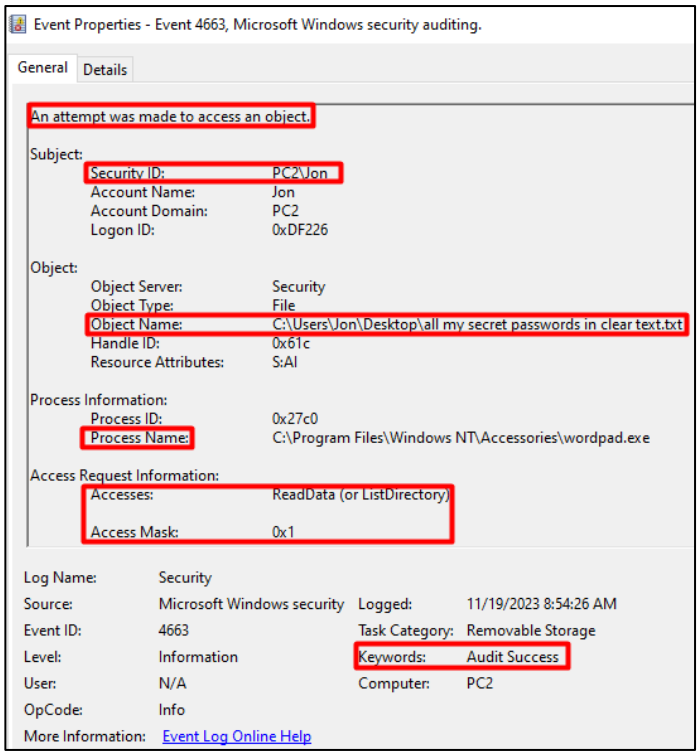
שימו לב שלמרות שיש לנו הרבה יותר אפשרויות למעקב אחרי פעולות עם האובייקט (כמו קריאה של מאפייניו, ניסיון להרצתו וכד') אנחנו לא מעוניינים לנטר פעולות כאלה מכיוון שהן לא מעידות שהסוד שהיה בתוכו נחשף. בנוסף, חשוב לציין כי הגדרה של ACE המכיל מספר רב של הרשאות יגרור ליצירה רבה מאוד של even-ים ול-**Alert Fatigue**, מה שבתור מגנים אנחנו חייבים להימנע ככל הניתן.

כעת נוכל לראות שנוצר ACE חדש:



נאשר הכל ונצא.

כעת, בשביל לוודא שהפקת הלוג באמת עובדת, כל שצריך לעשות הוא לפתוח את הקובץ ולחפש את לוג 4663 (File System Object Access) ב-event viewer. ואכן:



אפשר לראות בלוג המון שדות מעניינים כמו התהליך שניגש לאובייקט (wordpad.exe), ה-SID של ה-User, האם הפעולה הצליחה או לא ועוד. בקיצור, לא מעט שדות שניתן לייצר אגרגציה לפיהם ולפלטר באמצעותם.

שימוש בממשק משתמש תמיד טוב בשלבי ההסבר אך ברור לכולם כי בסביבה אמיתית, בשביל להימנע מטעויות ועבודה סיזיפית, חייבים לנטוש את ה-GUI ולהכניס **אוטומציה**. לכן, כנראה תשמחו לשמוע שניתן לבצע את כלל השלבים שהצגנו עד כה בסקריפט קצר ב-Powershell:

```
$AuditUser = "Everyone"
$AuditRules = "ReadData"
$InheritType = "None"
$PropagationFlags = "None"
$AuditType = "Success"

$FileReadSuccessAudit = New-Object
System.Security.AccessControl.FileSystemAuditRule($AuditUser,$AuditRules,$InheritType,$PropagationFlags,$AuditT
ype)

$FilePath = "$ENV:USERPROFILE\Desktop\all my secret passwords in clear text.txt"

$Acl = Get-Acl $FilePath -Audit                # Get the ACL with Audit ACEs
$Acl.SetAuditRule($FileReadSuccessAudit)       # Set the ACE
$Acl | Set-Acl | Out-Null                      # Apply the ACL
```

וואלה נראה מגניב אבל איך אפשר ליישם את זה מפני המתקפות מבוססות ACLs?

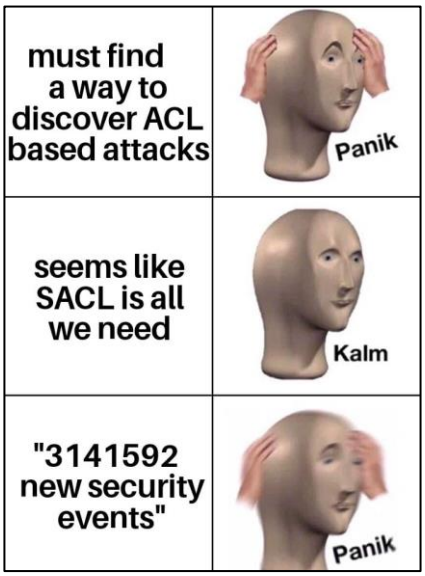
אם הבנתם את הרמה הבסיסית של קינפוג SACL על אובייקט פשוט בעמדת stand-alone אתם מבינים גם כיצד הסיפור נראה בדומיין. השוני העיקרי הוא במיקום הפקת הלוג. בעוד ש-ACEs ב-DACL מייצגים את טיב האינטרקציה של יישויות עם האובייקט, ה-ACEs ב-SACL מייצגים את סוגי האינטרקציות שעבורן יש להפיק לוג ב-security event log **של ה-Domain Controller.**

אז איזה לוגים שווה לנטר בשביל לגלות אם שמישהו משחק לנו עם ACL-ים בדומיין?

- **4662** - An operation was performed on an object
- **5137** - A directory service object was created
- **5136** - A directory service object was modified
- **5141** - A directory service object was deleted
- **4670** - permissions on an object were changed

חשוב לומר שזהו רק קצה הקרחון בעבודה עם SACL וניתן לנטר כל אובייקט עם Security descriptor בצורה דומה. ניתן לנטר בכל הדומיין יצירה של אובייקטים מסוג OU, GPO, משתמשים וקבוצות, מחשבים, עריכה של סכמת הדומיין ואפילו אובייקטים של DNS בשביל להפיק מהם לוגים להזנה בכלי אנליזה מתקדמים. פיצ'רים אלו הם חלק מסט הכלים שמגיעים עם מוצרי אבטחה תעשייתיים. שווה להכיר!

הערה: בפרספקטיבה דומה לפילוסופיה של הגנה מבוססת ריאקציה מתחילת הפרק, אנחנו יכולים לנטר את השינויים שמתרחשים באובייקט ולא את מה שכבר נעשה. כלומר, אם תוקף השתלט לנו על הדומיין לפני שהתחלנו לנטר את תצורת קינפוגי ה-DACL של כלל האובייקטים בדומיין - **לא נוכל להשתמש במכניזם הנ"ל בשביל לתפוס שינויים שכבר התרחשו בעבר**! מאחורי הקונספט הזה טמונה ההבנה מדוע שווה להכניס מוצרי הגנה לסביבת הדומיין כמה שיותר מוקדם.



האם ניתן להסתיר את מנגנון השרידות עוד יותר?

כמו שכבר הוצג, ניתן לשרשר את המנגנון של ההסתרה עם מספר אובייקטים נוספים ובכך לסבך ולהקשות את הגילוי עוד יותר. כמו כן, כמובן שניתן להצליב את השיטה עם שיטות שרידות מוכרות אחרות (משימות מתוזמנות, ניצול מנגנונים Kerberos-ים, יצירת SSP כדוגמת mimilsa ו-Sneaky Active Directory Persistence Tricks נוספים) אבל מכיוון שב-ACL עסקינן - ישנן דרכים ערמומיות עוד יותר.

לפי דעתי, למרות שלא מצאתי מישהו שכתב על זה לעומק, אפשר בהכרח להסתיר Backdoor המבוססת על DACL בלבד לביצוע Object Takeover בצורה טובה ושקטה יותר בסביבה דומיינית.

הרעיון של Security Descriptor וכל השדות שבתוכו הוא לא נחלתם הבלעדית של אובייקטים דומיינים (AD objects) אלא של כל האובייקטים המאובטחים באשר הם - גם אלו שמוגדרים בסביבה המקומית של מערכת ההפעלה.

על יתר סט, בעמוד 8 במאמר הקודם הצגתי כי למעשה ישנם 2 סוגים של ACEs: **Generic ACEs** אשר מתייחסים אל כלל האובייקטים ו-**Object-specific ACEs** אשר משרתים directory service. קרי, קבוצת ה-object specific מכילה צמד GUIDs (ObjectType ו-InheritedObjectType) נוספים שבעצם מהווים שדות אקסטרה שאפשר לברור ולקטלג מדיניות ACL לפיהם עבור האובייקטים עצמם ב-AD. אני מוכן להמר שלא מעט מוצרי הגנה מפלטרים לפיהם בלבד ומזניחים את קבוצת Generic ACEs באלגנטיות.

מכיוון שאימות מקומי מתרחש בצורה שונה מאימות דומייני (מוזמנים לדפדף במאמר Access Tokens Granted Right בשביל להבין בדיוק את סך ההבדלים), ניתן לייצר מנגנון persistence שמערב השמה של DACL על משתמש אדמין מקומי מוסתר בגבולות מכונת קצה ומנהלי הרשת לא יהיו מודעים בכלל להיווצרותו.

מכיוון שמשתמש מקומי בהגדרה (ובהתאם למנגנון בו הוא נוצר ונשמר במ"ה) אינו חלק מהדומיין, הוא לא יורש חוקות group policies דומייניות וכתוצאה מכך לא ניתן להחיל עליו מגבלות, לעקוב אחר פעולותיו ולנהל אותו מרחוק ברמת ה-AD. שווה לומר שמכיוון שאנחנו יוצאים מנקודת ההנחה שכבר הוסגה שליטה חזקה בדומיין, ניתן לבטל התייחסות אל סוגיית קיומם של LGPOs, הגדרות מגבילות אחרות ופתרונות EPP.

במידה וננקוט ב-"פעולות הסתרה" דומות לאלו שהוצגו בסביבה הדומיינית בסגרת המאמר כעת, נוכל לייצר אדמין מקומי שאף אחד לא מודע לקיומו - אפילו לא מי שיושב על אותה מכונה.

רעיון ערמומי נוסף הוא עריכת ה-ACEs של מפתחות מסויימים ב-Registry כך שלא ידרשו הרשאות גבוהות לשם גישה (מקומית או מרוחקת) אל הנתיב שמאפיין את כלל ה-Cached Domain Credentials (שדיפולטיבית שומר מקומית את כלל הפרטים של 10 ההתחברויות אחרונות למכונה) ובכך **ליצור שרידות דומיינית באמצעות משתמש מקומי חלש בלבד**.

לכן, במקום לערוך את ה-DACL של אובייקט AD חזק כדוגמת מחשב או יוזר בקבוצה Tier 0 של הארגון לשם שרידות אנחנו יכולים לערוך את ה-DACL של אובייקט לואקלי במכונה מרכזית ברשת ודרכה לחזור לאחיזה רשתית.

ללא ספק נדרשת עוד מחשבה ומחקר בשביל למצות את הכיוון הנ"ל אבל מאיך שאני קורא את הדברים, מדובר בשרידות חזקה במידה ולא קיים פתרון EDR או EPP מקיף מאוד על עמדת הקצה (ותכל'ס כנראה שגם אם כן).

סיכום

אני רוצה להאמין שכלל הנושא של מתקפות מבוססות ACLs בסביבה דומיינית הוא זה שהיווה את ההשראה להאמרה המפורסמת של ג'ון למברט בנושא אופי המאבק בין הצוות האדום לצוות הכחול - "מגנים חושבים ברשימות, תוקפים חושבים בגרפים וכל עוד זה המצב - התוקפים ינצחו". אם נצרף אליו משפט מפורסם עוד יותר של אבי תורת הלחימה המודרנית קרל פון קלאוזביץ אשר קבע כי "**קו ההגנה** (ב-AD) **לעולם ייפרץ**" נוכל אולי להשלים את התמונה ולהבין מדוע מתקפות מבוססות ACLs כה נפוצות וכאן כדי להישאר.

מהצד האדום, ניהול אופרציית OpSec נבונה הכוללת ביצוע אנומרציה של מערך ה-SACL לפני אנומרציה של ה-DACL היא הכרחית על מנת להבין את טיב הניטור ברשת עוד לפני שמתחילים להתבונן ולחפש נתיבי הסלמה מבוססי רשימות גישה. כמו כן, במידה והצלחתם להשיג שרידות מבוססת ושקטה על האובייקט מטרה שלכם, תשמרו עליו שקט או כפי שאמר אובמה: "don't do stupid shit".

מהצד הכחול, welp אין ספק שניטור אחר מיסקונפיגורציות של ACL-ים הוא **עניין מורכב**, במיוחד בסביבות גדולות. רוב רובם של ה-event-ים שנוצרים על ידי הניצול של ACL בכלל לא מופקים דיפולטיבית ב-Windows ולכן נדרש להגדיר אותם מבעוד מועד. קינפוג SACL באובייקטי מפתח ב-AD, בהתאם לכלל סט המתקפות מבוססות DACL בצמוד אל פיזור אובייקטי decoy ומלכודות דבש ברחבי הדומיין עשויים בהחלט להגביר את הסיכוי לגילוי תוקף בזמן אמת ומזעור נזקים בצורה משמעותית.

אבל מאיפה הכל התחיל? איך יצא שהגענו למצב העגום הזה שפיקוח רציף אחרי כמעט כל אובייקט בדומיין והפקת אינסוף לוגים עבור ההרשאות של אובייקטים השייכים אל Tier 0 בארגון הם הנורמה? מדוע לא קיים מנגנון אבטחה מובנה ויעיל יותר?

ובכן, בדיוק מהסיבה האלמנטרית ש-AD (בדומה אל רוב מערכות מבוססות זהויות) נועד בראש ובראשונה בשביל לחבר רשתות וגופים שונים לכדי סביבה אחת. הסיבה שקבוצת Exchange Enterprise Servers מקבלת הרשאת WriteDacl על האובייקט של הדומיין עצמו היא מכיוון שהפעולות שחבריה עושים בדומיין

הן **ליבתיות** מטבען. מה הטעם לאחד את כולם לסביבה אחת אם אי אפשר להחליף הודעות וקבצים (מיילים בשפה עממית) בין חברי הארגון?

יישומים רוחביים בצמוד אל רציפות ארגונית מאז ומתמיד היוו את קטר הביזנס בעוד שאבטחת מידע היא ה-by-product שנובעת ממנו. Love it or hate it זהו טבעו של השוק. מכניזם הגישה לאובייקטים הוא רק משיכת מכחול אחת מני רבות בתמונה של מערכת ההפעלה.

זה גם כנראה שורש הסיבה מדוע למרות מאמרים ומספר רב מאוד של כלים התקפיים בנושא, Microsoft כבר שנים מתעלמים באלגנטיות מהבעייתיות של נתבי תקיפה מבוססי ACL ובדומה לסל מתקפות לוגיות אחרות שמכבכות מזה שנים ב-Active Directory טוענים שזה חלק מאופרציה תקינה של הדומיין.

על הכותב

יהונתן אלקבס, חוקר אבטחת מידע, חובב סוקולנטים, טיולים ואוהב את המדינה שלנו.