Investigating Entra ID Attacks

Microsoft Entra ID (အရင်က Azure Active Directory လို့ ခေါ် ခဲ့တဲ့စနစ်) ဆိုတာက Cloud ပေါ်မှာ အလုပ်လုပ်တဲ့ Identity and Access Management စနစ်တစ်ခု ဖြစ်ပါတယ်။ အဖွဲ့အစည်းတွေက ဒီစနစ်ကို အသုံးပြုပြီး User တွေကို Login ခိုင်းတာ၊ Application တွေကို ခွင့်ပြုချက်ပေးတာ၊ Data နဲ့ Resource တွေကို ဘယ်သူသုံးလို့ရမလဲ ဆိုတာတွေကို ထိန်းချုပ်ကြပါတယ်။ အဖွဲ့အစည်းကြီးတွေ အများစုက Entra ID ကို အခြေခံပြီး သူတို့ရဲ့ Cloud Infrastructure ကို တည်ဆောက်ထားတဲ့အတွက် ဒီစနစ်ဟာ အလွန်အရေးကြီးတဲ့ အစိတ်အပိုင်းတစ်ခု ဖြစ်လာပါတယ်။

ဒါကြောင့်ပဲ Entra ID ဟာ Hacker တွေအတွက်လည်း အလွန်စိတ်ဝင်စားဖို့ကောင်းတဲ့ ပစ်မှတ်တစ်ခု ဖြစ်လာပါတယ်။ Entra ID ကို ထိန်းချုပ်နိုင်သွားရင် Application အများကြီးကို ဝင်ရောက်နိုင်ပြီး User အဖြစ် သုံးစွဲနိုင်သလို၊ တချို့အခါ Administrator အထိပါ တက်နိုင်ပါတယ်။ ဒါက Server တစ်လုံးကို Hack လုပ်တာထက်တောင် ပိုအန္တရာယ်ကြီးနိုင်ပါတယ်၊ အကြောင်းကတော့ Cloud ပေါ်မှာရှိတဲ့ Resource အကုန်လုံးကို သွယ်ဝိုက်ပြီး ထိန်းချုပ်နိုင်သွားနိုင်လို့ပါ။

ဒီ module ထဲမှာတော့ Hacker တွေက Entra ID ထဲကို ဘယ်လိုဝင်ရောက်ကြသလဲ၊ Authentication နဲ့ Permission စနစ်တွေကို ဘယ်လိုလှည့်စားအသုံးချကြသလဲ ဆိုတာတွေကို အလေးထားပြီး လေ့လာပါမယ်။ Authentication ဆိုတာက User ဟာ ဘယ်သူလဲ ဆိုတာကို အတည်ပြုတဲ့ လုပ်ငန်းစဉ်ဖြစ်ပြီး Permission ဆိုတာက အတည်ပြုပြီးတဲ့ User ကို ဘာလုပ်ခွင့်ပေးမလဲ ဆိုတာကို သတ်မှတ်ပေးတဲ့ အပိုင်းဖြစ်ပါတယ်။ Hacker တွေက ဒီအချက်နှစ်ချက်ကို မမှန်ကန်တဲ့ နည်းလမ်းနဲ့ အသုံးချပြီး Cloud Environment ထဲကို ဝင်ရောက်ကြပါတယ်။

ဒီ module မှာ အဓိက လေ့လာမယ့် တိုက်ခိုက်နည်း သုံးခုရှိပါတယ်။ ဒီတိုက်ခိုက်နည်းတွေဟာ Microsoft ရဲ့ Identity Platform ကို အသုံးချပြီး တရားဝင်လိုပဲ မြင်ရအောင် လုပ်ဆောင်ကြတဲ့ နည်းလမ်းတွေ ဖြစ်ပါတယ်။

ပထမနည်းလမ်းက OAuth နဲ့ Application Hijacking ဖြစ်ပါတယ်။ OAuth ဆိုတာက User ရဲ့ Password ကို မသိဘဲ Application တစ်ခုကို Access ခွင့်ပေးနိုင်တဲ့ စနစ်ဖြစ်ပါတယ်။ ဥပမာအားဖြင့် “Login with Microsoft” ဆိုတဲ့ ခလုတ်ကို နှိပ်လိုက်တဲ့အခါ Application က သင့် Email၊ Profile စတဲ့ အချက်အလက်တွေကို သုံးခွင့်ရသွားတာပါ။ Hacker တွေက ဒီ OAuth Flow ကို အသုံးချပြီး မကောင်းတဲ့ Application တစ်ခုကို ခွင့်ပြုခိုင်းပြီး User ရဲ့ Account ကို ထိန်းချုပ်သွားနိုင်ပါတယ်။ ဒါကို Application Hijacking လို့ ခေါ်ပါတယ်။

ဒုတိယနည်းလမ်းက Device Code Phishing ဖြစ်ပါတယ်။ ဒီနည်းလမ်းက Password မခိုးဘဲ User ကို ကိုယ်တိုင် Login လုပ်အောင် လှည့်စားတဲ့ နည်းပါ။ Hacker က Device Code တစ်ခု ပေးပြီး “ဒီ Code ကို Microsoft Site မှာ ထည့်ပြီး Login လုပ်ပါ” လို့ ပြောပါတယ်။ User က ယုံကြည်ပြီး Login လုပ်လိုက်တဲ့အခါ Hacker က User ရဲ့ Session ကို ရယူနိုင်သွားပါတယ်။ ဒီနည်းလမ်းက MFA (Multi-Factor Authentication) ပါရှိနေတောင် အသုံးချလို့ရနိုင်တဲ့အတွက် အလွန်အန္တရာယ်ကြီးပါတယ်။

တတိယနည်းလမ်းက Illicit Consent Grant ဖြစ်ပါတယ်။ Consent ဆိုတာက User က Application တစ်ခုကို “ဒီ App ကို ငါ့ Account သုံးခွင့်ပေးတယ်” လို့ ခွင့်ပြုခြင်းကို ဆိုလိုပါတယ်။ တချို့ Application တွေက Email ဖတ်ခွင့်၊ File ဖတ်ခွင့်၊ Calendar ပြင်ခွင့် စတဲ့ အခွင့်အရေးတွေကို တောင်းပါတယ်။ Hacker တွေက ဒီ Consent စနစ်ကို အသုံးချပြီး မသင့်တော်တဲ့ Permission အများကြီးကို တရားဝင်လိုပဲ

ရယူသွားကြပါတယ်။ User က “အလုပ်အတွက်လိုမယ်” ထင်ပြီး ခွင့်ပြုပေးလိုက်ရင် Hacker က Account ကို နောက်ကွယ်ကနေ အမြဲတမ်း အသုံးချနိုင်သွားပါတယ်။

ဒီ module ကို ပြီးဆုံးတဲ့အခါမှာ သင်ဟာ Hacker တွေက OAuth Flow ကို ဘယ်လိုအလွဲသုံးစားလုပ်ကြသလဲ ဆိုတာကို နားလည်လာပါလိမ့်မယ်။ ထို့အပြင် Entra ID ရဲ့ Audit Logs ထဲမှာ ဒီလို တိုက်ခိုက်မှုတွေကို ဘယ်လိုရှာဖွေဖော်ထုတ်ရမလဲ၊ ဘယ်အတိုင်းအတာထိ ထိခိုက်သွားနိုင်လဲ ဆိုတာကို စိစစ်နိုင်လာပါလိမ့်မယ်။

> Audit Log ဆိုတာက Entra ID ထဲမှာ ဖြစ်ပေါ် ခဲ့တဲ့ Login၊ Permission ပြောင်းလဲမှု၊ App Consent စတဲ့ အရာအားလုံးကို မှတ်တမ်းတင်ထားတဲ့ Log ဖြစ်ပါတယ်။

နောက်ဆုံးအနေနဲ့ Delegated Permissions နဲ့ Application Permissions တို့ရဲ့ ကွာခြားချက်ကိုလည်း နားလည်လာပါလိမ့်မယ်။ Delegated Permission ဆိုတာက User Login လုပ်ထားတဲ့ အချိန်မှာပဲ App က User ကိုယ်စား Access လုပ်နိုင်တာဖြစ်ပြီး Application Permission ဆိုတာက User မရှိဘဲ App ကိုယ်တိုင် Access လုပ်နိုင်တဲ့ အခွင့်အရေး ဖြစ်ပါတယ်။ Hacker တွေက အခြေအနေအလိုက် ဒီ Permission နှစ်မျိုးကို ရွေးချယ်ပြီး အသုံးချကြတာကို သင်သဘောတရားကျကျ နားလည်နိုင်လာပါလိမ့်မယ်။

Device Code Phishing

Device Code Flow ဆိုတာက Password ထည့်ပြီး Login လုပ်လို့ မဖြစ်နိုင်တဲ့ Device တွေအတွက် Microsoft က ပေးထားတဲ့ Authentication နည်းလမ်းတစ်ခု ဖြစ်ပါတယ်။ ဥပမာအားဖြင့် Smart Refrigerator, Smart TV, IoT Device, HoloLens လိုမျိုး Screen သေးသေး၊ Keyboard မရှိတဲ့ Device တွေမှာ Email နဲ့ Password ကို တိုက်ရိုက်ရိုက်ထည့်ဖို့ မလွယ်ကူပါဘူး။ ဒါကြောင့် Microsoft Identity Platform က Device Authorization Grant ဆိုတဲ့ Flow ကို ပေးထားတာပါ။

ဒီ Flow မှာ Device ကိုယ်တိုင် Login မလုပ်ဘဲ User က တခြား Device တစ်ခုဖြစ်တဲ့ ဖုန်း သို့မဟုတ် ကွန်ပျူတာကို အသုံးပြုပြီး Login လုပ်ပေးရပါတယ်။ အဲဒီနည်းလမ်းက Device Code Flow လို့ ခေါ်ပါတယ်။ ဒီ Flow ရဲ့ အဓိကအကြောင်းအရာက “ဒီ Code ကို ဒီ Website မှာ ထည့်ပြီး Login လုပ်ပါ” ဆိုတဲ့ ပုံစံနဲ့ User ကို ခိုင်းတာပါ။

Application တစ်ခုက Device Code Flow ကို စတင်ချင်ရင် Microsoft ရဲ့ `/devicecode` endpoint ကို API request ပို့ရပါတယ်။ အဲဒီ request ထဲမှာ Application ID နဲ့ Scope ဆိုတဲ့ Permission အမျိုးအစားတွေ ပါဝင်ပါတယ်။ Scope ဆိုတာက Application က User ရဲ့ ဘာအချက်အလက်တွေကို သုံးခွင့်တောင်းတာလဲ ဆိုတာကို ဖော်ပြတာပါ။ ဥပမာ User Profile ဖတ်ခွင့်၊ Email ဖတ်ခွင့် စတာတွေ ဖြစ်ပါတယ်။

`/devicecode` endpoint က User Code နဲ့ Device Code ကို ပြန်ပေးပါတယ်။ User Code ဆိုတာက လူက ထည့်ရမယ့် Code ဖြစ်ပြီး Device Code ဆိုတာက Application ဘက်မှာ သိမ်းထားရမယ့် Code ဖြစ်ပါတယ်။ User က `microsoft.com/devicelogin` ဆိုတဲ့ Microsoft ရဲ့ တရားဝင် Website မှာ User Code ကို ၁၅ မိနစ်အတွင်း ထည့်ရပါတယ်။ ဒီအချိန်အတွင်း Application က `/token` endpoint ကို အကြိမ်ကြိမ် မေးနေပါတယ်။ User က Login ဝင်ပြီး Consent ပေးလိုက်ရင် `/token` endpoint က Access Token ကို Application ဆီ ပြန်ပေးပါတယ်။

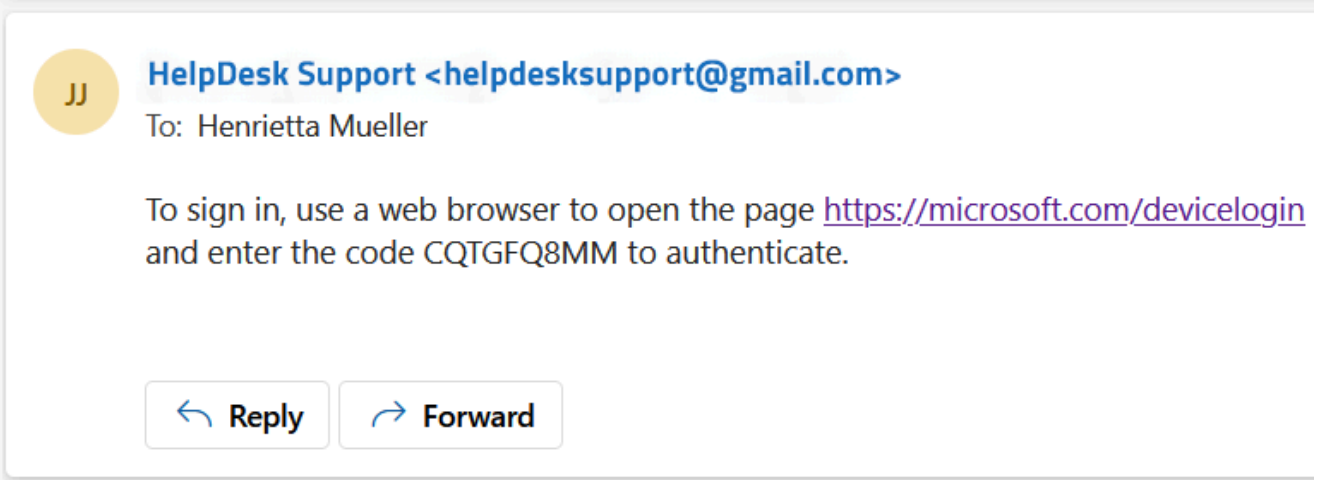
ဒီနေရာမှာ Device Code Flow ဟာ တရားဝင်နည်းလမ်းတစ်ခု ဖြစ်ပေမယ့် Hacker တွေက ဒီ Flow ကို အသုံးချပြီး Device Code Phishing ဆိုတဲ့ တိုက်ခိုက်နည်းကို လုပ်ကြပါတယ်။ Device Code Phishing ဆိုတာက Password မခိုးဘဲ User ကို ကိုယ်တိုင် Login လုပ်အောင် လှည့်စားတဲ့ Social Engineering တိုက်ခိုက်မှု ဖြစ်ပါတယ်။

Hacker က တရားဝင် Application တစ်ခုလိုပဲ `/devicecode` endpoint ကို သုံးပြီး Code တောင်းပါတယ်။ ဒီဥပမာမှာ Hacker က Microsoft Office ရဲ့ Application ID ကို အသုံးပြုပါတယ်။ ဒါကြောင့် Microsoft ဘက်က ကြည့်ရင် “Microsoft Office App က Login တောင်းနေတယ်” လို့ပဲ မြင်ရပါတယ်။ ဒီ Application ID ကို အသုံးပြုတာကြောင့် User က သံသယမရှိဘဲ ယုံကြည်သွားနိုင်ပါတယ်။

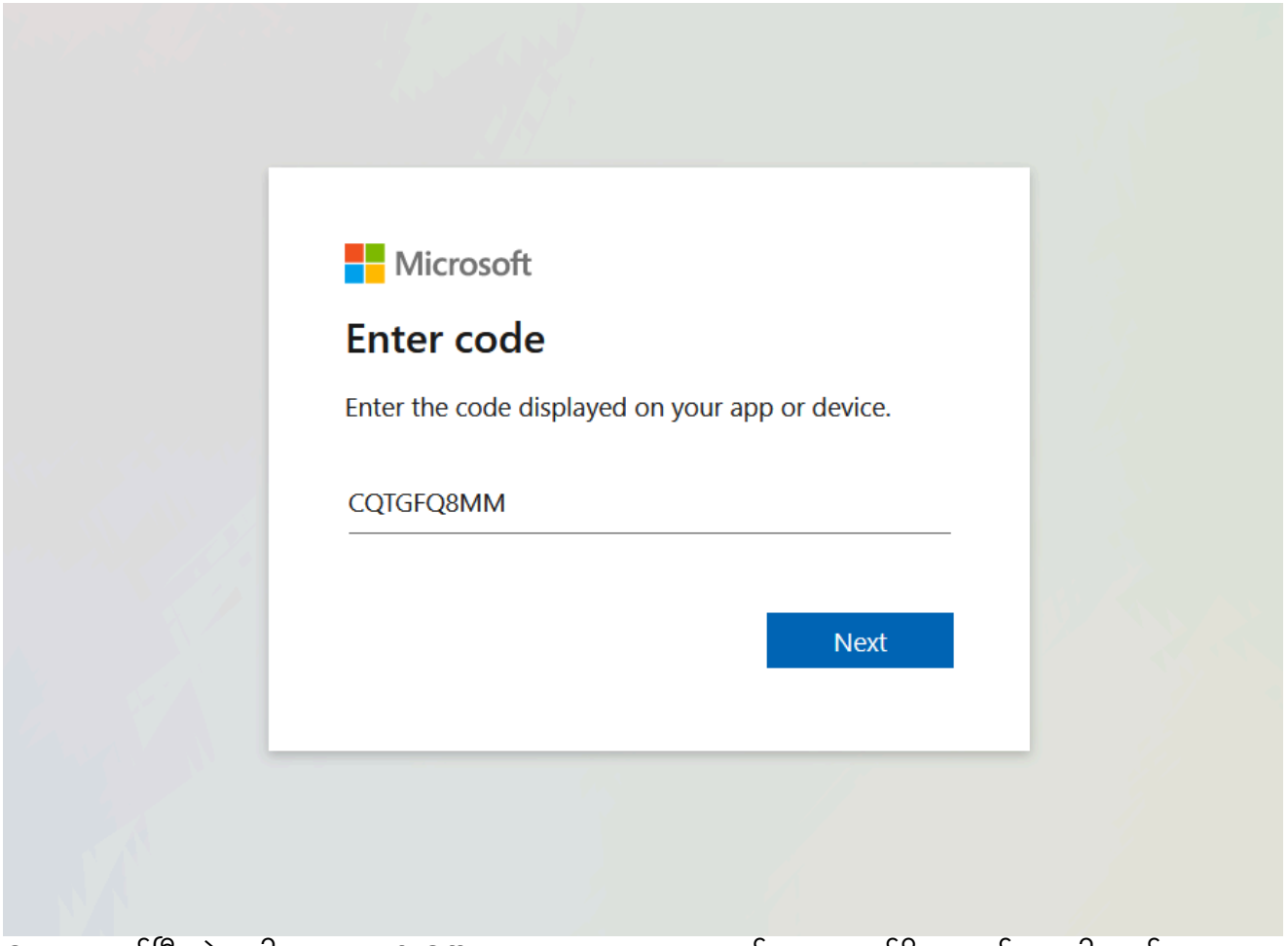
```
PS C:\Users\aceresponder> $ClientID = "d3590ed6-52b3-4102-aeff-aad2292ab01c"
$Scope = ".default offline_access"
$body = @{
"client_id" = $ClientID
"scope" = $Scope
}
$authResponse = Invoke-RestMethod -UseBasicParsing -Method Post -Uri "https://login.microsoftonline.com/common/oauth2/v2.0/devicecode" -Body $body
Write-Output $authResponse


user_code        : CQTGFQ8MM
device_code      : CAQABIQEAAADnfolhJpSnRYB1SVj-Hgd8dNnb5dya58hHdJf-4QKOYonVelLyLtee5_vV2EdYFJn0dtt4BiW3J9bziUlevRsSSCtWaAgnwfio35Vxh1-bqSvOH9aIQJM8O5wkqteXI2nEUmE4PtK_
                   jNBrmZAtSLIFynBUxCaZzZDpCdFOeXTTFAlrMjjd8UFxVg7Vl2LME-cgAA
verification_uri : https://microsoft.com/devicelogin
expires_in       : 900
interval         : 5
message          : To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code CQTGFQ8MM to authenticate.
```

Hacker က User Code နဲ့ Verification URL ကို Victim ဆီ ပို့ပါတယ်။



Email, Chat, Message တစ်ခုခုကနေ “ဒီ Code ကို Microsoft Site မှာ ထည့်ပြီး Login လုပ်ပါ” လို့ ပြောပါတယ်။ Victim က Link ကို ဖွင့်လိုက်တဲ့အခါ Microsoft ရဲ့ တရားဝင် Website ဖြစ်တာကို တွေ့ရတဲ့အတွက် ယုံကြည်ပြီး Code ကို ထည့်ပါတယ်။



Code ထည့်ပြီးတဲ့အခါ Microsoft Office Application အတွက် Login ဝင်ဖို့ တောင်းလာပါတယ်။

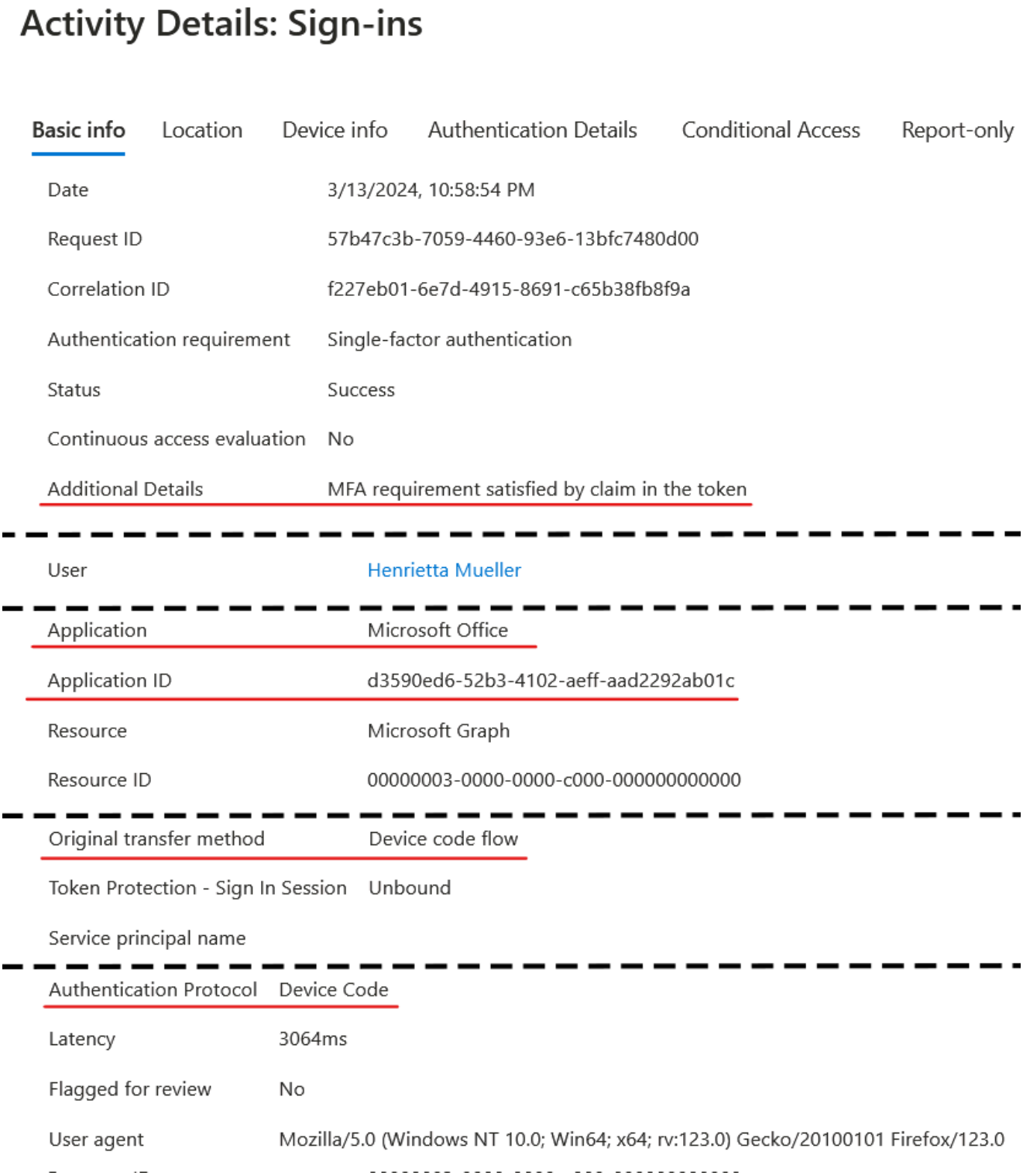
Victim က Email, Password နဲ့ MFA အပါအဝင် အားလုံးကို မှန်မှန်ကန်ကန် Login လုပ်ပြီး Consent ပေးလိုက်တဲ့အချိန်မှာ Hacker က Device Code ကို အသုံးပြုပြီး `/token` endpoint ဆီ Token တောင်းပါတယ်။ Microsoft က Token ကို Hacker ဆီ ပြန်ပေးလိုက်ပြီး Access Token နဲ့ Refresh Token ကို

ရရှိသွားပါတယ်။
The Token ထဲမှာ ဒီဟာတွေပါဝင်ပါတယ်။

```
{
    "token_type": "Bearer",
    "scope": "User.Read profile openid email",
    "expires_in": 3599,
    "access_token": "...",
    "refresh_token": "...",
    "id_token": "..."
}
```

Access Token က အချိန်ကန့်သတ်ရှိပြီး အချိန်ကုန်သွားရင် မသုံးနိုင်တော့ပါဘူး။ ဒါပေမယ့် Refresh Token ရှိနေရင် Access Token အသစ်တွေကို ထပ်မံ တောင်းယူနိုင်ပါတယ်။ အရေးကြီးတဲ့အချက်က Refresh Token ကို သုံးပြီး Access Token အသစ်တောင်းတဲ့ လုပ်ဆောင်ချက်တွေဟာ Audit Log မှာ မမြင်ရနိုင်တာပါ။ ဒါကြောင့် Attacker က အချိန်ကြာကြာ အသုံးချနေလည်း Detection ခက်ခဲပါတယ်။

Audit Log ထဲမှာ မြင်ရမယ့် အချက်တစ်ခုက MFA ဟာ Device Code Flow အောင်မြင်သွားတဲ့အတွက် Satisfied ဖြစ်နေတယ်ဆိုတာပါ။ နောက်တစ်ချက်က Requesting Application ဟာ Microsoft Office ဖြစ်နေတာပါ။ ဒါကြောင့် Security Team က ကြည့်ရင် ပုံမှန် Login လိုပဲ ထင်ရနိုင်ပါတယ်။ ဒါပေမယ့် Location tab ထဲမှာပါတဲ့ IP Address ကို ကြည့်ရင် Victim ရဲ့ IP မဟုတ်ဘဲ Attacker ရဲ့ IP ဖြစ်နေတတ်ပါတယ်။



ဒီနေရာမှာ ပိုပြီး အန္တရာယ်ကြီးတဲ့ အချက်တစ်ခုက Family of Client IDs (FOCI) ဆိုတာပါ။ ပုံမှန်အားဖြင့် Device Code Flow က Application ID နဲ့ Scope အတိုင်းပဲ Access Token ထုတ်ပေးရပါတယ်။ Refrigerator App က Outlook Email ကို ဝင်လို့မရသင့်ပါဘူး။ ဒါပေမယ့် Microsoft ရဲ့ Application တချို့ကို Family အဖြစ် သတ်မှတ်ထားပါတယ်။

FOCI ထဲမှာ ပါတဲ့ Application တွေက Family Refresh Token ဆိုတဲ့ အထူး Refresh Token ကို အသုံးပြုနိုင်ပါတယ်။ ဒီ Token က မူလ Consent ပေးထားတဲ့ Application တစ်ခုတည်းမက Family ထဲက အခြား Application တွေအတွက်ပါ Access Token တောင်းလို့ရပါတယ်။ ဒါက Microsoft Mobile App တွေအတွက် SSO လိုအပ်ချက်ကြောင့် ဖြစ်လာတဲ့ Design Quirk တစ်ခု ဖြစ်ပါတယ်။ The list of affected application IDs can be found here.

အဲဒီလို Application တွေထဲမှာ Microsoft Office, Outlook Mobile, Teams, Azure CLI, PowerShell, OneDrive, Authenticator စတဲ့ App တွေ ပါဝင်ပါတယ်။ Attacker က Microsoft Bing Search လို Application တစ်ခုနဲ့ Device Code Phishing လုပ်ခဲ့ရင်တောင် Family Refresh Token ရှိသွားရင် Microsoft Office အတွက် Access Token တောင်းယူနိုင်ပါတယ်။ အဲဒီအခါ Microsoft Office ရဲ့ Default Scope အတိုင်း Permission အများကြီးကို ရယူနိုင်သွားပါတယ်။

ဒါကြောင့် Device Code Phishing ကို တွေ့ရှိလိုက်ရင် “ဒီ App က ဒီ Permission ပဲရှိတယ်” လို့ မထင်သင့်ပါဘူး။ Family Refresh Token ရှိနိုင်တဲ့အတွက် Impact ဟာ အရမ်းကျယ်ပြန့်နိုင်ပါတယ်။ Identity Forensics လုပ်တဲ့အခါ FOCI Application တွေ ပါဝင်နေလားဆိုတာကို အထူးသတိထားစစ်ဆေးရပါမယ်။

Device Code Phishing Victim

Which user may have been the victim of a device code phishing attack?

ဒီ Investigation မှာ Device Code Phishing ဖြစ်နိုင်တဲ့ Authentication Event တွေကို ရှာဖို့ Entra ID Sign-in Logs ကို Query လုပ်ပါတယ်။ အသုံးပြုတဲ့ Query က `signInEventTypes:interactiveUser AND originalTransferMethod:deviceCodeFlow` ဖြစ်ပါတယ်။ ဒီ Query ရဲ့ အဓိပ္ပါယ်က User ကိုယ်တိုင် Login လုပ်ထားတဲ့ Event တွေအထဲက Device Code Flow ကို အသုံးပြုထားတဲ့ Authentication တွေကိုပဲ ရွေးထုတ်လိုက်တာပါ။

ဒီ Query ကို Run လုပ်တဲ့အခါ alexw@aceresponder.com အတွက် Event နှစ်ခု ထွက်လာပါတယ်။

>	Mar 17, 2024 @ 20:27:14.000	31.222.254.98	deviceCodeFlow	alexw@aceresponder.com
>	Mar 17, 2024 @ 20:27:05.000	31.222.254.98	deviceCodeFlow	alexw@aceresponder.com
>	Mar 17, 2024 @ 20:26:07.000	157.97.134.4	none	alexw@aceresponder.com
>	Mar 17, 2024 @ 20:24:57.000	157.97.134.4	none	alexw@aceresponder.com

Device Code Flow ကို User တစ်ယောက်က မကြာခဏ အသုံးပြုတာဟာ ပုံမှန်မဟုတ်တဲ့အတွက် ဒီအချက်တင်ပဲ သံသယဝင်စရာ ဖြစ်နေပါပြီ။ သို့သော် ဒီအဆင့်မှာပဲ “Attack ဖြစ်တယ်” လို့ မသတ်မှတ်သင့်သေးပါဘူး။

ဒါကြောင့် နောက်တစ်ဆင့်အနေနဲ့ alexw@aceresponder.com ရဲ့ အရင် Interactive Logon Event တွေကို ပြန်ကြည့်ပါတယ်။ အထူးသဖြင့် Source IP Address ကို နှိုင်းယှဉ်ကြည့်ပါတယ်။ Device Code Phishing Attack မှာ အရေးကြီးတဲ့ လက္ခဏာတစ်ခုက Login ကို Victim က ကိုယ်တိုင် လုပ်ပေးပေမယ့် Token ကို Request လုပ်တဲ့ Application ဘက်က IP Address ဟာ Victim ရဲ့ IP မဟုတ်ဘဲ Attacker ရဲ့ IP ဖြစ်နေတတ်တာပါ။

Consent Grant Attacks

Consent Grant Attack ဆိုတာက Microsoft Entra ID ရဲ့ OAuth Authorization Flow ကို တရားဝင် Application တစ်ခုလိုပဲ အသုံးချပြီး User ကို ကိုယ်တိုင် Permission ပေးအောင် လှည့်စားတဲ့ တိုက်ခိုက်

နည်း ဖြစ်ပါတယ်။ ဒီ Attack ဟာ Password ခိုးတာ မဟုတ်ဘဲ “ခွင့်ပြုချက်” ကို ခိုးတာဖြစ်လို့ User နဲ့ Security Team အတွက် သတိမထားရင် အလွန်လွယ်ကူစွာ ကျရောက်နိုင်ပါတယ်။

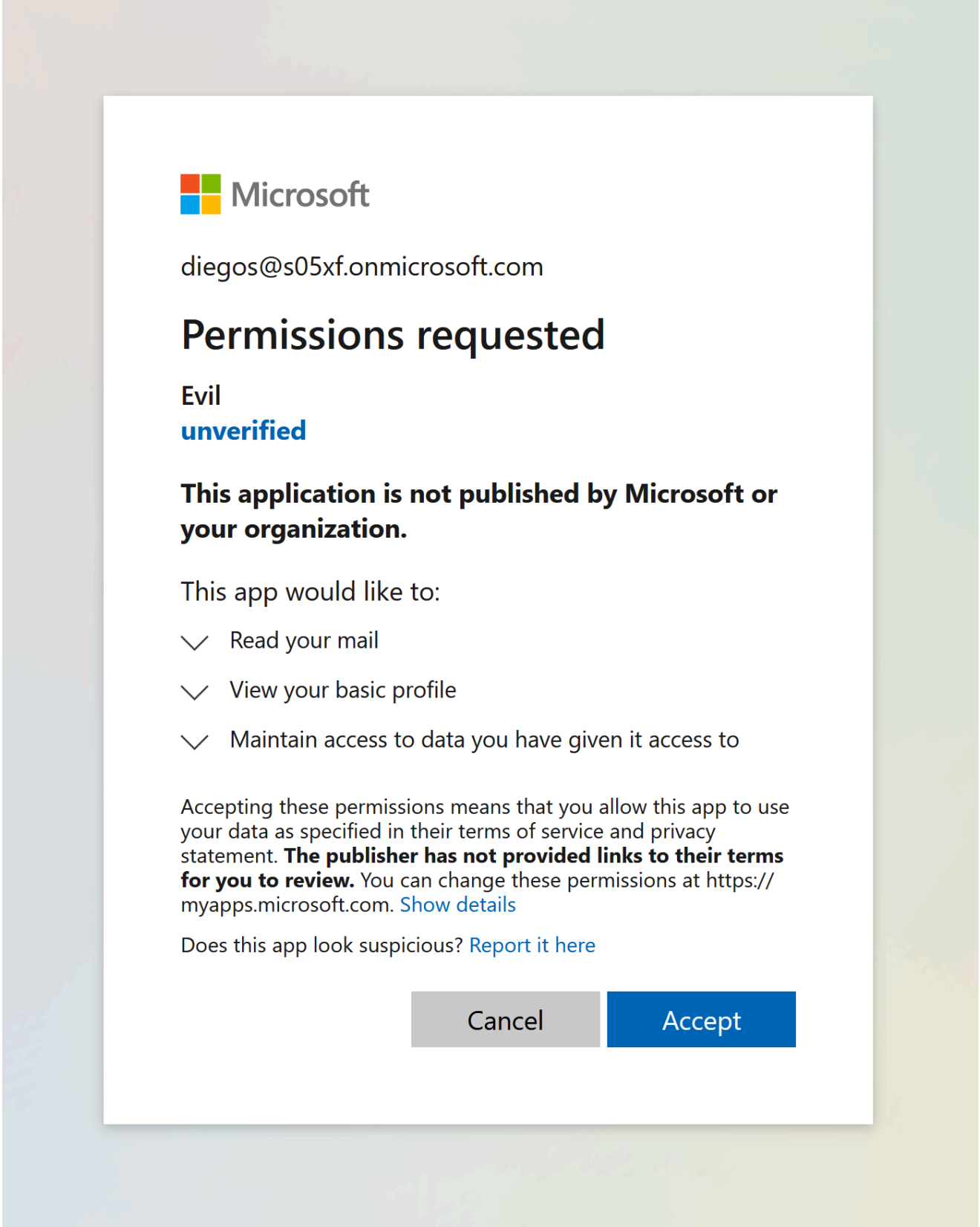
ပုံမှန် OAuth Flow မှာ Application တစ်ခုက User ရဲ့ Data ကို သုံးချင်ရင် Microsoft Identity Platform ကို Authorization Request ပို့ရပါတယ်။ User ကို Consent Screen ပြပြီး “ဒီ App ကို ဒီ Permission တွေ သုံးခွင့်ပေးမလား” လို့ မေးပါတယ်။ User က Accept လုပ်လိုက်ရင် Application က User ကိုယ်စား Access လုပ်နိုင်သွားပါတယ်။ Consent Grant Attack မှာ Attacker က ဒီတရားဝင် Flow ကိုပဲ အသုံးချပြီး မကောင်းတဲ့ Application တစ်ခုနဲ့ Social Engineering လုပ်တာပါ။

Attacker က အရင်ဆုံး OAuth Application တစ်ခုကို Register လုပ်ပါတယ်။ ဒီ Application ဟာ Microsoft ဘက်ကကြည့်ရင် တရားဝင် Application တစ်ခုလိုပဲ ဖြစ်နေပါတယ်။ ပြီးရင် Authorization URL တစ်ခုကို ပြုလုပ်ပါတယ်။

```
https://login.microsoftonline.com/common/oauth2/v2.0/authorize?
client_id=65a990d4-43e2-48d9-8856-fe536dca2b93
&response_type=code
&redirect_uri=http%3A%2F%2Fexample.com%2F
&response_mode=query
&scope=https%3A%2F%2Fgraph.microsoft.com%2Fmail.read
```

အဲဒီ URL ထဲမှာ `client_id` ဆိုတဲ့ အပိုင်းက Attacker ရဲ့ Application ID ဖြစ်ပြီး `redirect_uri` ဆိုတာက User က Consent ပေးပြီးတဲ့နောက် ပြန်သွားမယ့် Attacker ရဲ့ Server Address ဖြစ်ပါတယ်။ Scope ဆိုတာက Application က ဘာ Permission တောင်းနေတာလဲ ဆိုတာကို ဖော်ပြတာဖြစ်ပြီး ဒီဥပမာမှာ User ရဲ့ Mail ကို ဖတ်ခွင့် တောင်းထားတာကို မြင်ရပါတယ်။

Victim က ဒီ Phishing Link ကို Click လုပ်လိုက်တဲ့အခါ Microsoft ရဲ့ တရားဝင် Login Page ကို ရောက်သွားပါတယ်။ URL ကို ကြည့်ရင် login.microsoftonline.com ဖြစ်နေလို့ User က သံသယမရှိဘဲ ယုံကြည်သွားတတ်ပါတယ်။ Login ဝင်ပြီးတဲ့နောက် Microsoft က Consent Screen ပြပြီး “ဒီ Application ကို Mail.Read Permission ပေးမလား” လို့ မေးပါတယ်။



User က အလုပ်အတွက်လိုမယ် ထင်ပြီး Accept လုပ်လိုက်ရင် Attack စတင်ပြီးသား ဖြစ်ပါတယ်။

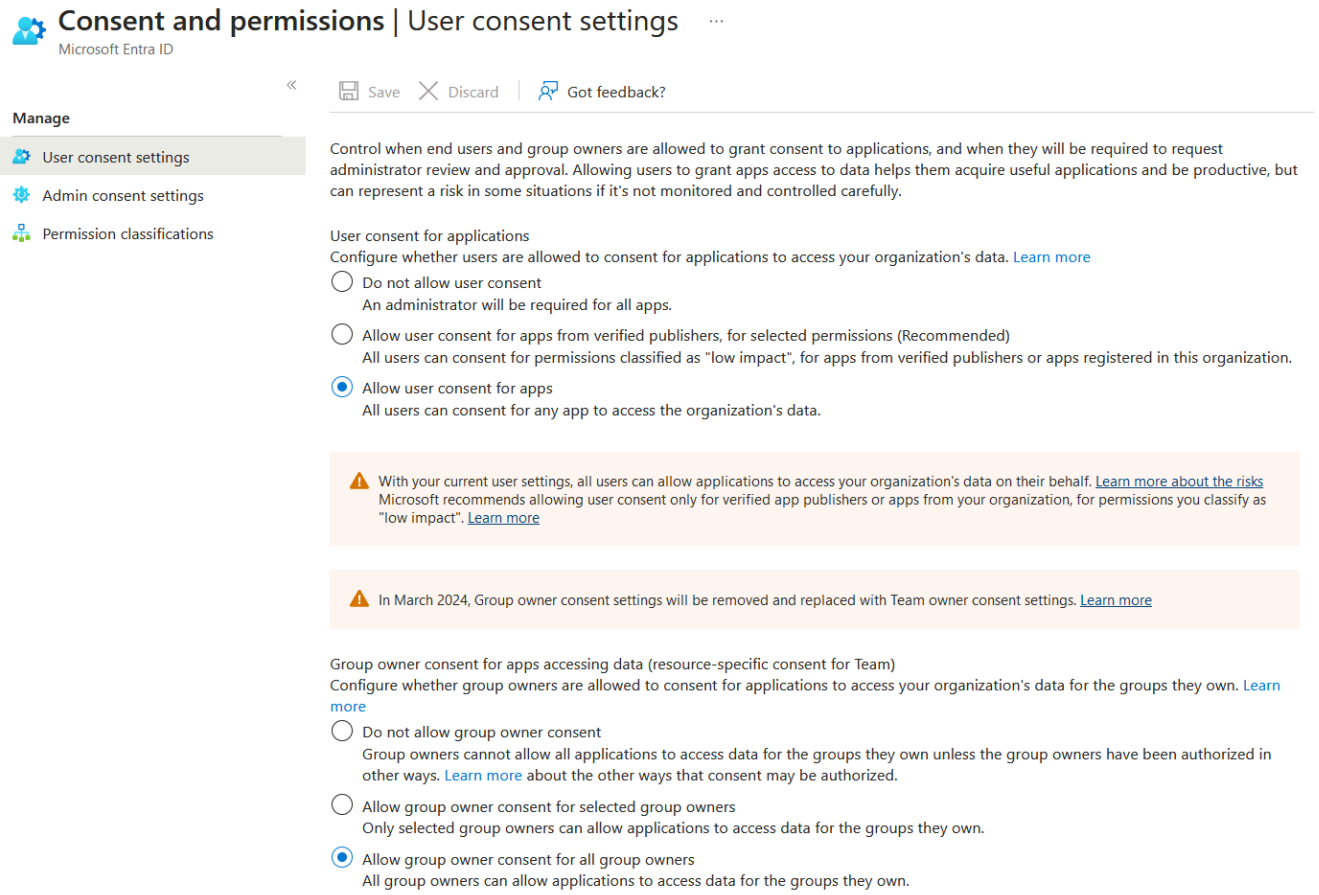
User က Consent ပေးလိုက်တဲ့အချိန်မှာ Microsoft Identity Platform က Authorization Code တစ်ခု ကို Attacker ရဲ့ redirect_uri ဆီ ပို့ပါတယ်။ ဒီ Authorization Code ကို အသုံးပြုပြီး Attacker က Access Token နဲ့ Refresh Token တွေကို တောင်းယူနိုင်ပါတယ်။ ဒီ Token တွေကို အသုံးပြုပြီး User ရဲ့ Email ကို နောက်ကွယ်ကနေ ဖတ်ရှုနိုင်သွားပါတယ်။ အရေးကြီးတာက User ရဲ့ Password ကို မသိဘဲ ဒီ အရာတွေကို လုပ်နိုင်တာပါ။

ဒီလို Consent Grant Attack ဖြစ်သွားတဲ့အခါ Entra ID Audit Log ထဲမှာ Activity သုံးမျိုး ထွက်ပေါ်လာ ပါတယ်။

Category	Activity
ApplicationManagement	Consent to application ဆိုတဲ့ Event က User ဘယ်သူက Consent ပေးခဲ့လဲ၊
UserManagement	ဘယ် Application ကို ပေးခဲ့လဲ
ApplicationManagement	ဘာ Permission တွေ ပေးလိုက်လဲ

Add app role assignment grant to user ဆိုတဲ့ Event က Application ကို User ကိုယ်စား လုပ်ဆောင်ခွင့် ပေးလိုက်တယ်ဆိုတာကို ပြပါတယ်။ Add delegated permission grant ဆိုတဲ့ Event က Application ကို ဘယ် Permission တွေ Delegated လုပ်လိုက်လဲ ဆိုတာကို ပြပါတယ်။ ဒီ Event သုံးခုထဲမှာ Consent to application Event ဟာ Investigation အတွက် အရေးကြီးဆုံး ဖြစ်ပါတယ်။

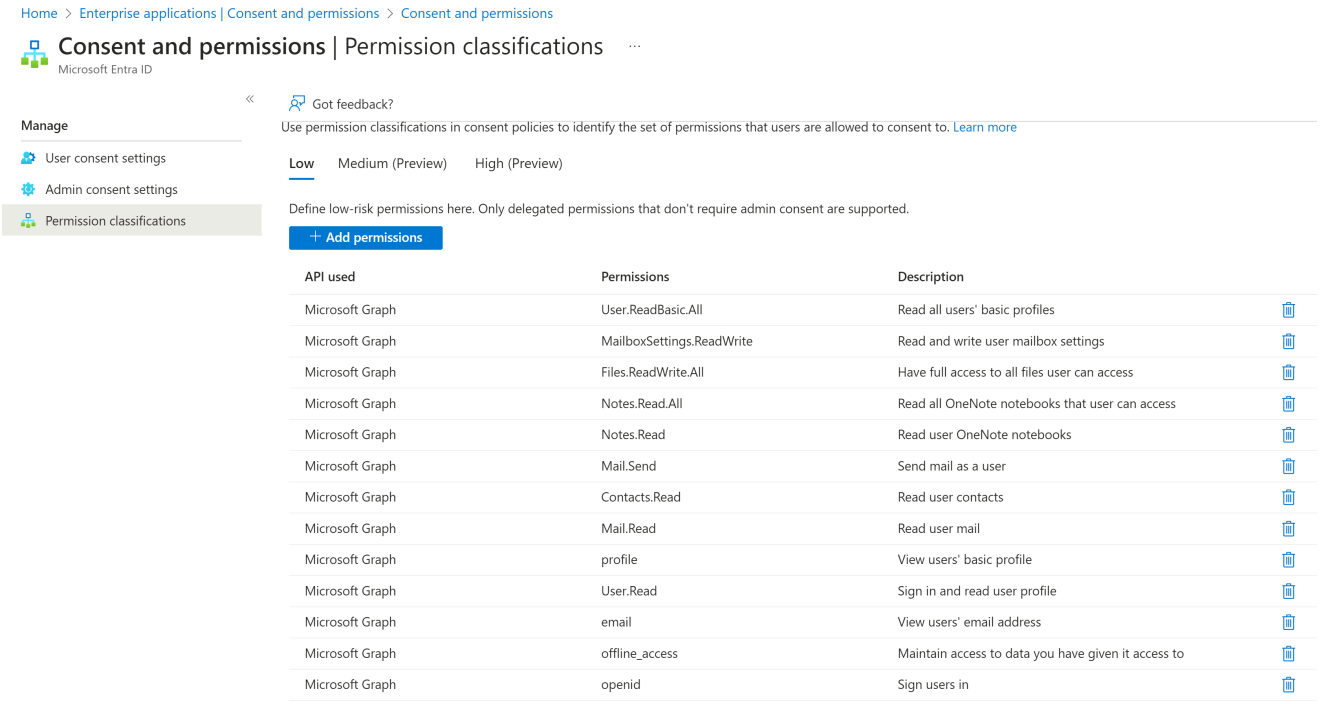
ပုံမှန်အားဖြင့် Microsoft Entra ID မှာ User တွေကို Application တိုင်းအတွက် Consent ပေးခွင့် မပေးထားပါဘူး။ Default Setting အရ Verified Application တွေအတွက် “Low Impact Permission” တွေကိုပဲ User က ကိုယ်တိုင် Consent ပေးလို့ရပါတယ်။



ဒီ Policy ကို Enterprise Applications ထဲက Consent and Permissions အပိုင်းမှာ သတ်မှတ်ထားပါတယ်။ အကယ်လို့ Organization က Setting ကို အားနည်းအောင် ချထားမယ်ဆိုရင် User တစ်ယောက်ဟာ Permission မရွေးဘဲ ဘယ် Application ကိုမဆို Consent ပေးနိုင်သွားပါတယ်။ ဒါက အလွန်အန္တရာယ်ကြီးတဲ့ အခြေအနေ ဖြစ်ပါတယ်။

Low Impact Permission ဆိုတာက User တစ်ယောက်တည်းရဲ့ Resource ကိုပဲ Access လုပ်နိုင်တဲ့ Permission တွေကို ဆိုလိုပါတယ်။ ဥပမာ User ရဲ့ Profile ဖတ်ခွင့်၊ Mail ဖတ်ခွင့်လိုမျိုး Permission တွေ ဖြစ်ပါတယ်။ Organization က ဘယ် Permission ကို Low Impact လို့ သတ်မှတ်မလဲ ဆိုတာကို ကိုယ်တိုင် ပြန်လည် သတ်မှတ်နိုင်ပါတယ်။ ဒါပေမယ့် ဒီ Permission တွေကိုတောင် Attacker က ရရှိသွား

ရင် User ရဲ့ Email တွေကို ဖတ်နိုင်ပြီး Internal Information တွေကို စုဆောင်းနိုင်တာကြောင့် Impact က မသေးပါဘူး။



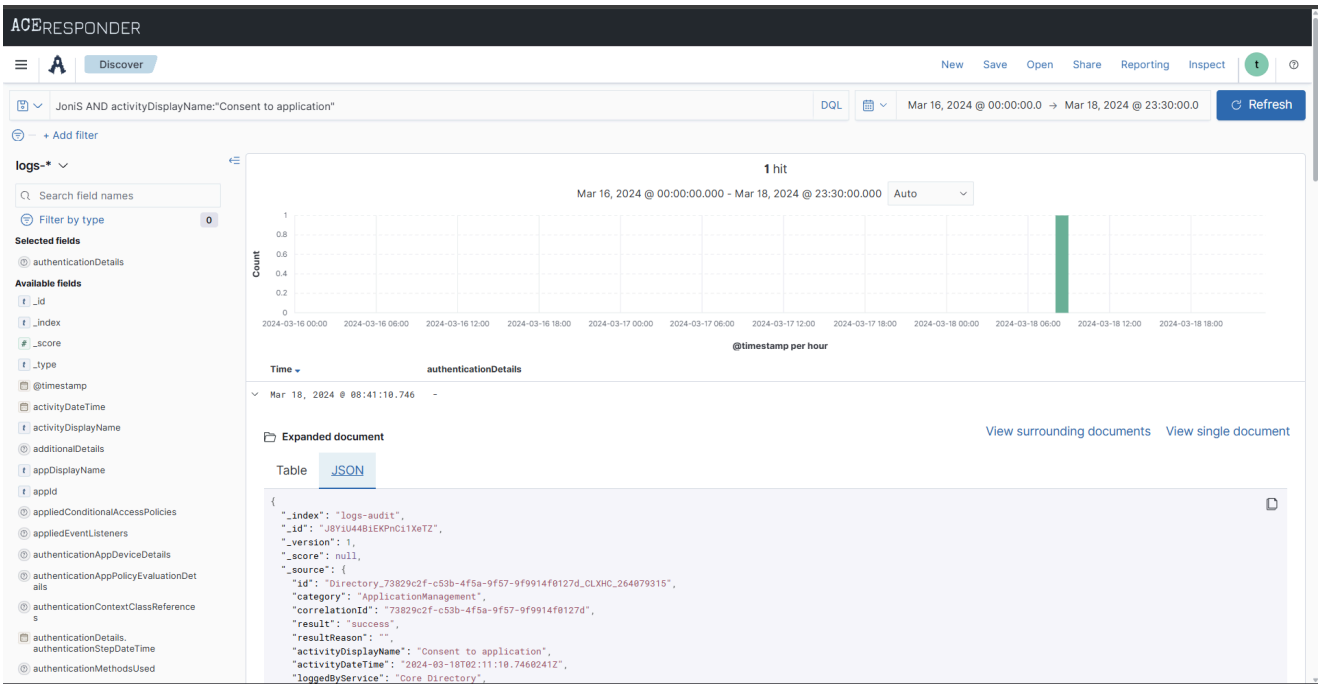
အကျဉ်းချုပ်ပြောရရင် Consent Grant Attack ဆိုတာ Microsoft ရဲ့ OAuth System ကို ချိုးဖောက်တာ မဟုတ်ဘဲ User ရဲ့ ယုံကြည်မှုကို အသုံးချပြီး တရားဝင် Permission ကို ကိုယ်တိုင် ပေးအောင် လှည့်စားတဲ့ တိုက်ခိုက်နည်း ဖြစ်ပါတယ်။ Password မခိုးဘဲ Account ကို ထိန်းချုပ်နိုင်တဲ့အတွက် Detection ခက်ခဲပြီး Cloud Identity Security မှာ အလွန်အရေးကြီးတဲ့ Threat တစ်ခု ဖြစ်ပါတယ်။ Forensics လုပ်တဲ့အခါ Consent to application Event ကို အဓိကထား စစ်ဆေးနိုင်ရင် ဒီလို Attack တွေကို ဖော်ထုတ်နိုင်ပါတယ်။

Illicit Consent Grant Victim

JoniS consented to an external application. What scope (permission) did the attacker request?

```
JoniS AND activityDisplayName:"Consent to application"
```

Audit Logs ထဲမှာ
`activityDisplayName:"Consent to application"`
ဆိုတဲ့ Event ကို စစ်တဲ့အခါ targetResources field ထဲမှာ Application ကို ပေးလိုက်တဲ့ Permission (Scope) ကို တိတိကျကျ မြင်ရပါတယ်။



targetResources


```
{
  "id": "27e44b4f-32a3-439a-ac17-3ac8a6479e16",
  "displayName": "Calendarify",
  "type": "ServicePrincipal",
  "userPrincipalName": null,
  "groupType": null,
  "modifiedProperties": [
    {
      "displayName": "ConsentContext.IsAdminConsent",
      "oldValue": null,
      "newValue": "\"True\""
    },
    {
      "displayName": "ConsentContext.IsAppOnly",
      "oldValue": null,
      "newValue": "\"False\""
    },
    {
      "displayName": "ConsentContext.OnBehalfOfAll",
      "oldValue": null,
      "newValue": "\"False\""
    },
    {
      "displayName": "ConsentContext.Tags",
      "oldValue": null,
      "newValue": "\"WindowsAzureActiveDirectoryIntegratedApp\""
    },
    {
      "displayName": "ConsentAction.Permissions",
      "oldValue": null,
      "newValue": "\"[] => [[Id: T0vkJ6MymkOsFzrIpkeeFoOXhVuK1ylAiXDE34zz3xgcgpakBbqxSpJUFVAj4xdK, ClientId: 27e44b4
970-c4df8cf3df18, ConsentType: Principal, Scope:  Files.Read.All  CreatedDateTime: , LastModifiedDateTime ]]; \""
    },
    {
      "displayName": "TargetId.ServicePrincipalNames",
      "oldValue": null,
      "newValue": "\"8b530358-9ad5-4ed2-b1da-f47893594989\""
    }
  ]
}
```


Internal Illicit Consent Grant

Internal Illicit Consent Grant ဆိုတာက Organization အတွင်းမှာရှိပြီးသား Account တစ်ခုကို Attacker က ရရှိထားပြီးနောက် ဆက်လက် အသုံးချတဲ့ OAuth Consent Attack အမျိုးအစား ဖြစ်ပါတယ်။ အများစုသော Organization တွေမှာ Verified မဟုတ်တဲ့ External Application တွေကို User က ကိုယ်တိုင် Consent ပေးခွင့် မရှိပါဘူး။ ဒါကြောင့် Attacker တစ်ယောက်အနေနဲ့ အပြင်ဘက်က App တစ်ခုနဲ့ တိုက်ခိုက်ဖို့ မလွယ်ကူပါဘူး။ ဒီအခြေအနေကြောင့် Internal Illicit Consent Grant Attack ဟာ

Attacker က Organization ထဲမှာ အခြေခံအဆင့် Access တစ်ခုရရှိပြီးနောက် ပိုမိုတွေ့ရတဲ့ Attack ဖြစ်လာပါတယ်။

Attacker က User Credential တစ်ခုခုကို ရရှိထားရင် Organization အတွင်းမှာ Application တစ်ခုကို ကိုယ်တိုင် Create လုပ်နိုင်ပါတယ်။ Microsoft Entra ID ရဲ့ Default Setting အရ User အများစုကို Application Create လုပ်ခွင့် ပေးထားပါတယ်။ ဒါကြောင့် Administrator မဟုတ်တဲ့ Low-Privilege User တစ်ယောက်နဲ့တောင် Attack ကို စတင်နိုင်ပါတယ်။ ဒီအချက်က Internal Consent Grant Attack ကို အန္တရာယ်ကြီးစေတဲ့ အဓိကအချက် ဖြစ်ပါတယ်။

Internal Consent Grant Attack ကို Investigation လုပ်တဲ့အခါ Entra ID Audit Logs ထဲမှာ အထူး သတိထားစစ်ဆေးရမယ့် Event သုံးမျိုး ရှိပါတယ်။

Category	Activity
ApplicationManagement	Add application
ApplicationManagement	Update application – Certificates and secrets management
ApplicationManagement	Consent to application

ပထမ Event က ApplicationManagement Category အောက်က Add application ဖြစ်ပါတယ်။ ဒီ Event က Organization အတွင်း Application အသစ်တစ်ခု Create လုပ်ထားတယ်ဆိုတာကို ပြပါတယ်။ ဒုတိယ Event က Update application – Certificates and secrets management ဖြစ်ပြီး Application အတွက် Client Secret သို့မဟုတ် Certificate တစ်ခု ထည့်ထားတယ်ဆိုတာကို ပြပါတယ်။ တတိယ Event က Consent to application ဖြစ်ပြီး User တစ်ယောက်က အဲဒီ Application ကို Permission ပေးလိုက်တယ်ဆိုတာကို ဖော်ပြပါတယ်။

Internal Illicit Consent Grant Attack ရဲ့ လုပ်ငန်းစဉ်ဟာ External Consent Grant Attack နဲ့ အခြေခံ အားဖြင့် တူညီပါတယ်။ Social Engineering ကို အသုံးပြုပြီး User ကို Consent ပေးအောင် လှည့်စားရ ပါမယ်။ ဒါပေမယ့် Internal Attack ဖြစ်တဲ့အတွက် မဖြစ်မနေ ပါဝင်ရမယ့် အဆင့်တစ်ခုက Application ကို Create လုပ်ခြင်း သို့မဟုတ် ပြင်ဆင်ခြင်း ဖြစ်ပါတယ်။ ဒီအချက်က Investigation အတွက် အရေးကြီး တဲ့ ခြားနားချက် ဖြစ်ပါတယ်။

ပုံမှန်အားဖြင့် Attacker က Low-Privilege User အဖြစ် Application တစ်ခု Create လုပ်ပါတယ်။ အဲဒီ Application မှာ သူလိုချင်တဲ့ Permission တွေကို ထည့်ပါတယ်။ ပြီးရင် Application ကို Token တောင်း နိုင်ဖို့ Secret သို့မဟုတ် Certificate တစ်ခု Create လုပ်ပါတယ်။ အဲဒီအပြီးမှာ အခြား User တွေကို Target လုပ်ပြီး “ဒီ App ကို Consent ပေးပါ” ဆိုတဲ့ Social Engineering ကို စတင်လုပ်ဆောင်ပါတယ်။ User တစ်ယောက်က Consent ပေးလိုက်တာနဲ့ Attacker က Token တွေကို ရယူနိုင်ပြီး User ကိုယ်စား Cloud Resource တွေကို Access လုပ်နိုင်သွားပါတယ်။

ဒါပေမယ့် Attack တွေဟာ အမြဲတမ်း ဒီပုံစံအတိုင်းပဲ မဖြစ်ကြပါဘူး။ Attacker က Permission အဆင့် မြင့်တဲ့ Account ကို ရရှိထားရင် ရှိပြီးသား Application တစ်ခုကို Modify လုပ်နိုင်ပါတယ်။ ဥပမာအားဖြင့် Application ရဲ့ Redirect URI ကို မကောင်းတဲ့ Server တစ်ခုဆီ ညွှန်ပြလိုက်နိုင်ပါတယ်။ အဲဒီလိုလုပ်နိုင်ရင် User က Login လုပ်ပြီး Consent ပေးလိုက်တဲ့ Authorization Code ကို Attacker ရဲ့ Server ဆီ ပို့သွားနိုင်ပါတယ်။

အခြားတစ်ဖက်မှာ Attacker က ရှိပြီးသား Application ရဲ့ Client Secret ကို ခိုးယူနိုင်သော်လည်းကောင်း၊ အသစ်တစ်ခု ထပ်ထည့်နိုင်သော်လည်းကောင်း အဲဒီ Application ကို အသုံးပြုပြီး Token တွေကို တောင်းယူနိုင်ပါတယ်။ ဒီလိုအခြေအနေတွေမှာ Attack ဟာ ပိုမိုဖုံးကွယ်ပြီး Detection ခက်ခဲလာပါတယ်။ ဒါဟာ Attacker ရဲ့ ရရှိထားတဲ့ Permission အဆင့်၊ လုပ်ဆောင်နိုင်တဲ့ အတိုင်းအတာနဲ့ အချိန်ကုန်ကျမှုကို ချိန်ဆပြီး ရွေးချယ်တဲ့ နည်းလမ်း ဖြစ်ပါတယ်။

အကျဉ်းချုပ်ပြောရရင် Internal Illicit Consent Grant Attack ဆိုတာ Organization အတွင်းမှာ ယုံကြည်ထားတဲ့ Application နဲ့ User Permission စနစ်ကို အသုံးချပြီး Cloud Identity ကို တိုက်ခိုက်တဲ့ နည်းလမ်း ဖြစ်ပါတယ်။ External App မသုံးနိုင်တဲ့ အခြေအနေတွေမှာ Attacker တွေက ဒီနည်းလမ်းကို အသုံးပြုကြပြီး Application Creation, Secret Management နဲ့ Consent Event တွေကို Audit Log ထဲမှာ ဆက်စပ်ပြီး စစ်ဆေးနိုင်ရင် ဒီလို Attack တွေကို ဖော်ထုတ်နိုင်ပါတယ်။

Internal Application

What is the DisplayName of the internal application created by JoniS?

Internal Illicit Consent Grant Investigation မှာ ပထမဆုံး စစ်ရမယ့် အချက်က Organization အတွင်း Application အသစ်တစ်ခု Create လုပ်ထားတဲ့ Event ဖြစ်ပါတယ်။ Entra ID Audit Logs ထဲမှာ Application အသစ် Create လုပ်တဲ့ လုပ်ဆောင်ချက်တွေကို `Add application` Event အနေနဲ့ မှတ်တမ်းတင်ထားပါတယ်။ ဒါကြောင့် JoniS က Application တစ်ခု Create လုပ်ထားလားဆိုတာကို စစ်ဖို့

```
JoniS AND activityDisplayName:"Add application"
```

ဆိုတဲ့ Query ကို အသုံးပြုပါတယ်။

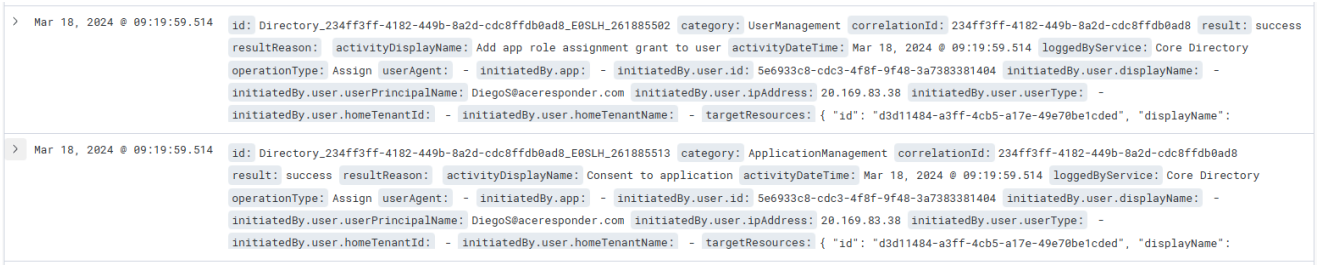
ဒီ Query ကို Run လုပ်လိုက်တဲ့အခါ JoniS က Application တစ်ခု Create လုပ်ထားတဲ့ Event ကို တွေ့ရပါတယ်။ အဲဒီ Event ထဲက `targetResources` Field ကို ကြည့်လိုက်ရင် Create လုပ်ထားတဲ့ Application ရဲ့ DisplayName ကို တွေ့နိုင်ပါတယ်။ အဲဒီ DisplayName က **FinanceMgr** ဖြစ်ပါတယ်။ ဒါကြောင့် JoniS က Create လုပ်ထားတဲ့ Internal Application ရဲ့ အမည်ဟာ FinanceMgr ဖြစ်တယ်လို့ သတ်မှတ်နိုင်ပါတယ်။

နောက်ထပ် အရေးကြီးတဲ့ အချက်တစ်ခုက FinanceMgr Application မှာ `AppAddress` ဆိုတဲ့ Field ပါရှိနေတာပါ။ ဒီ AppAddress က User တစ်ယောက် Consent ပေးလိုက်တဲ့အခါ Microsoft Identity

Platform က Authorization Code သို့မဟုတ် Token ကို ပို့ပေးမယ့် Destination ဖြစ်ပါတယ်။ ပုံမှန် Application တစ်ခုအတွက် AppAddress ဟာ Trusted Endpoint ဖြစ်သင့်ပေမယ့် ဒီ Case မှာတော့ Attacker က ထိန်းချုပ်ထားတဲ့ Server ကို ညွှန်ပြနေပါတယ်။ ဒါက FinanceMgr ဟာ Legitimate Internal App မဟုတ်ဘဲ Illicit Consent Grant Attack အတွက် Create လုပ်ထားတဲ့ Malicious Application ဖြစ်နိုင်ကြောင်းကို ပြသနေတဲ့ အရေးကြီးတဲ့ Evidence ဖြစ်ပါတယ်။

Test : Leveraging Consent

What did the attacker do as a result of the internal consent grant attack?



ActivityDisplayName ကို ကြည့်လိုက်တော့ Consent to application ပြီးသွားတော့ Add app role assignment grant to user ဆိုတာ တက်လာတယ်။ အတော့ user create ဆိုတာဖြစ်နိုင်တယ်။ ပြီးတော့ permission ပါတစ်ခါတည်းပေးလိုက်တယ်ဆိုတာ အဲ့အပေါ်က log တွေကနေ သိနိုင်တယ်။

Test : Leveraging Consent 2

Which role did the attacker assign to JoniS?

```
FinanceMgr And activityDisplayName:"Add"
```

လို့ ရှာလိုက်တော့ အောက်ကဟာလေးထွက်လာတယ်။


```
{
"id": "a496821c-ba05-4ab1-9254-155023e3174a",
"displayName": null,
"type": "User",
"userPrincipalName": "JoniS@aceresponder.com",
"groupType": null,
"modifiedProperties": [
    {
    "displayName": "Role.ObjectID",
    "oldValue": null,
    "newValue": "\"47abfbed-3dec-489f-9e65-e129576acc27\"" },
    {
    "displayName": "Role.DisplayName",
    "oldValue": null,
```

```
    "newValue": "\"Global Administrator\""
    },...
```


The answer : Global Administrator

Test : Modifying Roles

What scope (permission) did the attacker use to add the Global Administrator role to JoniS?


```
targetResources
{
  "id": "d3d11484-a3ff-4cb5-a17e-49e70be1cded",
  "displayName": "FinanceMgr",
  "type": "ServicePrincipal",
  "userPrincipalName": null,
  "groupType": null,
  "modifiedProperties": [
    {
      "displayName": "ConsentContext.IsAdminConsent",
      "oldValue": null,
      "newValue": "\"True\""
    },
    {
      "displayName": "ConsentContext.IsAppOnly",
      "oldValue": null,
      "newValue": "\"False\""
    },
    {
      "displayName": "ConsentContext.OnBehalfOfAll",
      "oldValue": null,
      "newValue": "\"False\""
    },
    {
      "displayName": "ConsentContext.Tags",
      "oldValue": null,
      "newValue": "\"WindowsAzureActiveDirectoryIntegratedApp\""
    },
    {
      "displayName": "ConsentAction.Permissions",
      "oldValue": null,
      "newValue": "\"[] => [[Id: hBTR0_-jtUyhfknnC-HN7YOXhVuK1ylAiXDE34zz3xjIM2lew82PT59IOnODOBQE, ClientId: d3d11484-a3ff-4cb5-a17e-49e
70be1cded, PrincipalId: 5e6933c8-cdc3-4f8f-9f48-3a7383381404, ResourceId: 5b859783-d78a-4029-8970-c4df8cf3df18, ConsentType: Principal,
Scope:  RoleManagement.ReadWrite.Directory, CreatedDateTime: , LastModifiedDateTime ]]; \""
```

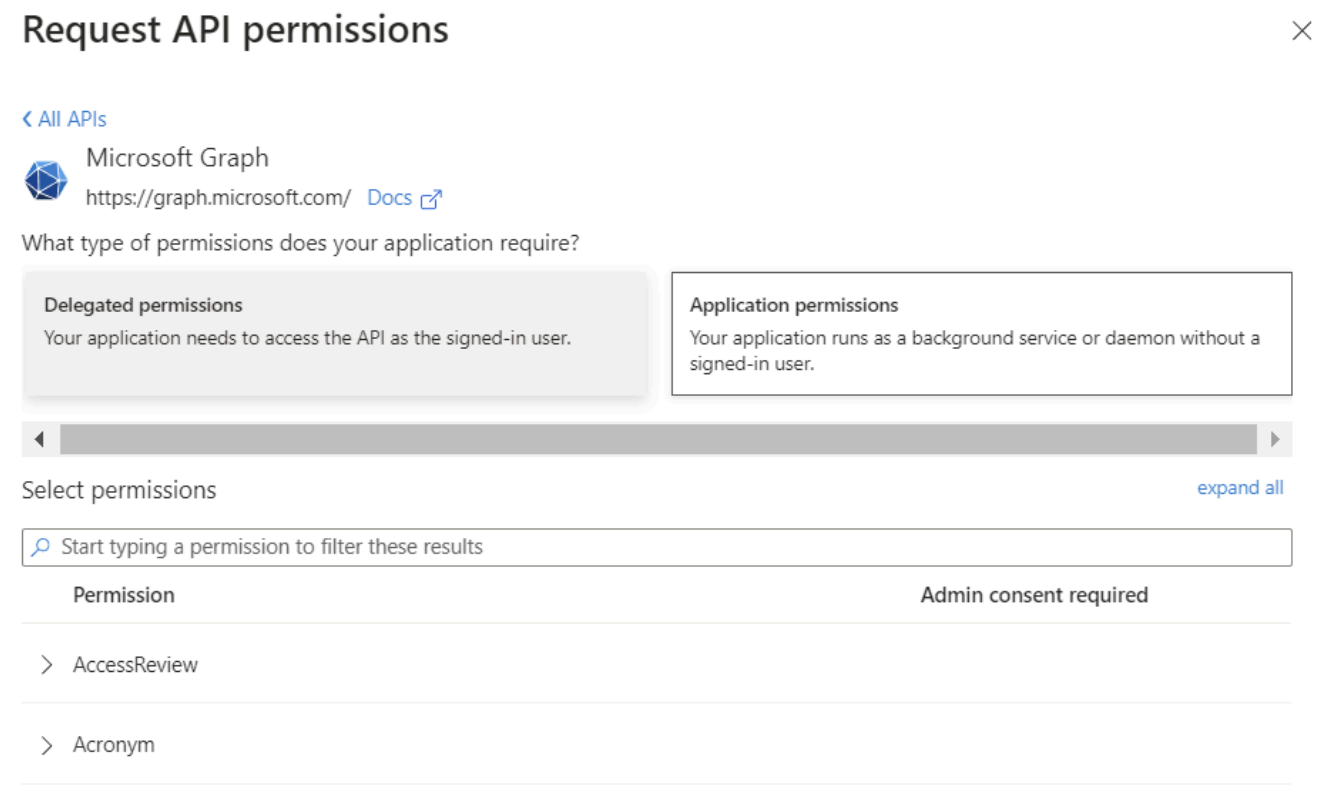
OAuth App Hijacking

အခုအထိ ကျွန်တော်တို့ ဆွေးနွေးလာခဲ့တဲ့ Attack အများစုက Delegated Access ကို အလွဲသုံးစားလုပ်တဲ့ နည်းလမ်းတွေ ဖြစ်ပါတယ်။ Delegated Access ဆိုတာက Application တစ်ခုက User ကိုယ်စား Resource တွေကို Access လုပ်တာကို ဆိုလိုပါတယ်။ ဒီ Access အမျိုးအစားမှာ Permission ကို User ကိုယ်တိုင် Consent ပေးရပြီး Application က User ရဲ့ Identity ကို အခြေခံပြီး လုပ်ဆောင်နိုင်တာပါ။

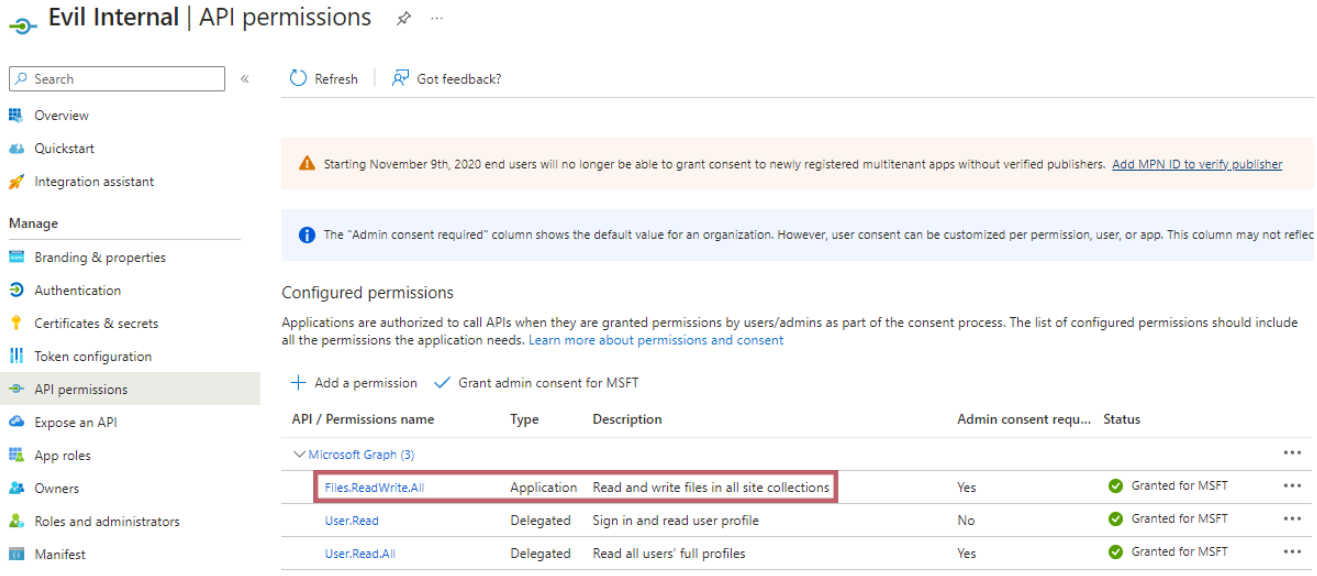
Delegated Permission ရရှိထားတဲ့ Application တစ်ခုဟာ User ရဲ့ Permission အကန့်အသတ်အောက်မှာပဲ အလုပ်လုပ်နိုင်ပါတယ်။ အဓိကအားဖြင့် User ကိုယ်တိုင် Access လုပ်လို့ရတဲ့ Resource တွေကိုပဲ Application က Access လုပ်နိုင်ပြီး User က Consent ပေးထားတဲ့ Permission အတိုင်းပဲ လုပ်ဆောင်နိုင်ပါတယ်။ ဒါကြောင့် Delegated Access ဟာ User အပေါ် အလွန်မှုတည်နေတဲ့ Access အမျိုးအစား ဖြစ်ပါတယ်။

Delegated Access ရဲ့ အားနည်းချက်တစ်ခုက Application က Resource ကို Access လုပ်ချင်တိုင်း User က Sign in လုပ်ထားရပါမယ်။ Access Token နဲ့ Refresh Token တွေ Expire ဖြစ်သွားရင် ဒါမှမဟုတ် Admin က Revoke လုပ်လိုက်ရင် User က ထပ်မံ Consent ပေးရပါမယ်။ အဲဒီအတွက် Attacker အတွက် Delegated Access ဟာ အမြဲတမ်း တည်တံ့နေတဲ့ Control မဟုတ်ပါဘူး။

အဲဒီလိုအခြေအနေတွေကို ကျော်လွှားချင်ရင် Application Permissions ကို အသုံးပြုရပါတယ်။ Application Permission ဆိုတာက User မရှိဘဲ Application ကိုယ်တိုင် Resource တွေကို Access လုပ်နိုင်တဲ့ Permission ဖြစ်ပါတယ်။ ဒီ Permission အမျိုးအစားမှာ Application က Service Principal အနေနဲ့ Authenticate လုပ်ပြီး User ကိုယ်စား မဟုတ်ဘဲ ကိုယ်တိုင် Identity တစ်ခုအဖြစ် အလုပ်လုပ်ပါတယ်။



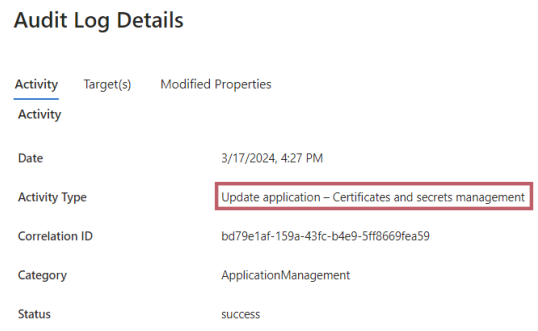
ဥပမာအားဖြင့် Application တစ်ခုမှာ `Files.ReadWrite.All` ဆိုတဲ့ Application Permission ရှိနေမယ်ဆိုရင် Organization အတွင်း User တွေရဲ့ File တွေကို User Login မရှိဘဲ ဖတ်နိုင်၊ ပြင်နိုင်သွားပါတယ်။ ဒါက Delegated Permission ထက် အန္တရာယ်ပိုကြီးတဲ့ Permission ဖြစ်ပါတယ်၊ အကြောင်းကတော့ User Interaction မလိုအပ်တော့လို့ပါ။



> OAuth App Hijacking ဆိုတာက Attacker တစ်ယောက်က Application Permission ပါရှိတဲ့ OAuth Application ကို ထိန်းချုပ်သွားနိုင်တဲ့ Attack ကို ဆိုလိုပါတယ်။

Attacker က Application ရဲ့ Client Secret သို့မဟုတ် Certificate ကို ရရှိသွားရင် Application အနေနဲ့ Authenticate လုပ်နိုင်သွားပါတယ်။ ဒီအချိန်မှာ User Account တစ်ခုမှ မလိုအပ်တော့ဘဲ Application Identity ကို အသုံးပြုပြီး Resource တွေကို တိုက်ရိုက် Access လုပ်နိုင်သွားပါတယ်။

Attacker က Secret ကို ခိုးယူနိုင်တာမျိုးသာမက Permission ရှိထားရင် Application ထဲကို Secret အသစ် သို့မဟုတ် Certificate အသစ်ကို ထပ်ထည့်နိုင်ပါတယ်။ အဲဒီလိုလုပ်နိုင်တာနဲ့ Attacker က တရားဝင် Application ပိုင်ရှင်လိုပဲ Login လုပ်နိုင်ပြီး Application Permission တွေကို အလွဲသုံးစားလုပ်နိုင်ပါတယ်။ ဒီအခြေအနေဟာ Account Compromise ထက်တောင် ပိုဆိုးနိုင်ပါတယ်၊ အကြောင်းကတော့ Application Permission တွေက Organization အဆင့် Resource တွေကို ထိန်းချုပ်နိုင်လို့ပါ။



ဒီလို OAuth App Hijacking ဖြစ်နေတယ်ဆိုတာကို Entra ID Audit Logs ထဲမှာ Identify လုပ်နိုင်ပါတယ်။ အထူးသဖြင့် `ApplicationManagement` Category အောက်က `Update application – Certificates and secrets management` ဆိုတဲ့ Activity ကို သတိထားစစ်ဆေးရပါမယ်။ ဒီ Event က Application ရဲ့ Secret သို့မဟုတ် Certificate ကို ထပ်ထည့်တာ၊ ပြင်ဆင်တာ၊ ပြောင်းလဲတာတွေကို မှတ်တမ်းတင်ထားပါတယ်။

Test : App Hijacking

An attacker hijacked an internal application and logged on with the secret they created. Which user principal performed the attack?

Entra ID Audit Logs ထဲမှာ Application ရဲ့ Secret သို့မဟုတ် Certificate ကို Create လုပ်တာ၊ ပြင်ဆင်တာတွေကို `Update application – Certificates and secrets management` Activity အနေနဲ့ မှတ်တမ်းတင်ထားပါတယ်။ Investigation လုပ်တဲ့အခါ `activityDisplayName:"secrets management"` ဆိုတဲ့ Query ကို အသုံးပြုပြီး Application Secret နဲ့ Certificate ဆိုင်ရာ Event တွေကို ရှာပါတယ်။

ဒီ Case မှာ အဲဒီလို Event နှစ်ခု တွေ့ရပါတယ်။ တစ်ခုက JoniS ဖြစ်ပြီး သူက Internal Application ကို Create လုပ်တဲ့အချိန် Secret တစ်ခု ထည့်ခဲ့တာဖြစ်ပါတယ်။ ဒါဟာ Application Create လုပ်တဲ့ Flow အတွင်း ပါဝင်နိုင်တဲ့ ပုံမှန်လုပ်ဆောင်ချက်လည်း ဖြစ်နိုင်ပါတယ်။


```
"userPrincipalName": "JoniS@aceresponder.com", "ipAddress":
"185.216.201.89", "userType": null, "homeTenantId": null, "homeTenantName":
null } }, "targetResources": [ { "id": "e57c43f5-c2e5-4495-bda8-
b6fc3f7a17c4", "displayName": "FinanceMgr", "type": "Application",
"userPrincipalName": null, "groupType": null, "modifiedProperties": [ {
```

```
"displayName": "KeyDescription", "oldValue": "[]", "newValue": "[\"
[KeyIdentifier=7d0b85c9-5144-459d-9adc-
ec7bd887497e,KeyType=Password,KeyUsage=Verify,DisplayName=t]\"]" }, {
"displayName": "Included Updated Properties", "oldValue": null, "newValue":
"\"KeyDescription\"" }
```


ဒါပေမယ့် နောက်ထပ် Secret သို့မဟုတ် Certificate ကို ထပ်ထည့်ထားတဲ့ User က AlexW@aceresponder.com ဖြစ်ပါတယ်။ AlexW က Application Owner မဟုတ်ဘဲ Secret ကို ထပ်ထည့်ထားတယ်ဆိုတာက Application Identity ကို ထိန်းချုပ်ဖို့ ကြိုးစားနေတဲ့ လက္ခဏာတစ်ခု ဖြစ်ပါတယ်။ Secret တစ်ခု ရရှိသွားတာနဲ့ Application Permission တွေကို အသုံးပြုပြီး User Login မလိုအပ်ဘဲ Authenticate လုပ်နိုင်သွားပါတယ်။ ဒါက OAuth App Hijacking ရဲ့ အဓိက အချက်ပါ။


```
"user": { "id": "9338e1cc-40d4-48d5-a6d3-5ecb21a32d29", "displayName": null,
"userPrincipalName": "AlexW@aceresponder.com", "ipAddress": "92.119.19.131",
"userType": null, "homeTenantId": null, "homeTenantName": null } },
"targetResources": [ { "id": "b139e28f-bf49-420a-a0ce-bf935b102672",
"displayName": "TestAdmin", "type": "Application", "userPrincipalName":
null, "groupType": null, "modifiedProperties": [ { "displayName":
"KeyDescription", "oldValue": "[\"[KeyIdentifier=8e0ee66e-b6d8-4e13-8de0-
5efcc9df69e5,KeyType=Password,KeyUsage=Verify,DisplayName=test]\"]",
"newValue": "[\"[KeyIdentifier=8e0ee66e-b6d8-4e13-8de0-
5efcc9df69e5,KeyType=Password,KeyUsage=Verify,DisplayName=test]\",\"
[KeyIdentifier=8bdac207-1dca-4604-8f64-
8573dc87554c,KeyType=Password,KeyUsage=Verify,DisplayName=t]\"]" }, {
"displayName": "Included Updated Properties", "oldValue": null, "newValue":
"\"KeyDescription\"" }
```

Test : App Hijacking Actions

What did the attacker do after hijacking TestAdmin?

AlexW က TestAdmin Application ထဲကို Client Secret တစ်ခု ထပ်ထည့်လိုက်တဲ့အခါ OAuth App Hijacking ဟာ အောင်မြင်သွားပါတယ်။ Secret တစ်ခု ရရှိသွားတာနဲ့ Application ကို User Account မလိုအပ်ဘဲ Authenticate လုပ်နိုင်သွားပါတယ်။ ဒီအချိန်မှာ ဖြစ်ပေါ်လာတဲ့ Sign-in Event ဟာ User Sign-in မဟုတ်ဘဲ `Service Principal Sign-in` ဖြစ်ပါတယ်။ Service Principal Sign-in ဆိုတာက Application Identity ကို အသုံးပြုပြီး Login လုပ်ထားတယ်ဆိုတဲ့ အဓိပ္ပါယ်ပါ။

	Time	activityDisplayName	initiatedBy.user.userPrincipalName	appDisplayName	signInEventTypes
›	Mar 17, 2024 @ 22:03:16.127	Add user	-	-	-
›	Mar 17, 2024 @ 22:02:35.000	-	-	TestAdmin	servicePrincipal
›	Mar 17, 2024 @ 22:01:40.962	Update application	AlexW@aceresponder.com	-	-
›	Mar 17, 2024 @ 22:01:40.961	Update application - Certificates and secrets management	AlexW@aceresponder.com	-	-
›	Mar 17, 2024 @ 22:01:40.695	Update service principal	AlexW@aceresponder.com	-	-

TestAdmin Application မှာ Excessive Application Permissions ရှိနေတဲ့အတွက် Attacker က User Account မရှိဘဲ Organization အတွင်း Administrative လုပ်ဆောင်ချက်တွေကို လုပ်နိုင်သွားပါတယ်။

Service Principal အနေနဲ့ Authenticate လုပ်ပြီးတဲ့နောက် Attacker က Microsoft Graph API ကို အသုံးပြုပြီး User အသစ်တစ်ယောက်ကို Create လုပ်လိုက်ပါတယ်။ ဒီလုပ်ဆောင်ချက်ဟာ Organization အတွက် အလွန်အန္တရာယ်ကြီးတဲ့ Impact ဖြစ်ပါတယ်။

targetResources


```
{
  "id": "0c6968fc-0511-4414-8ebc-fbd4aa21544b",
  "displayName": null,
  "type": "User",
  "userPrincipalName": "BachD@aceresponder.com",
  "groupType": null,
  "modifiedProperties": [
    {
      "displayName": "AccountEnabled",
      "oldValue": "[]",
      "newValue": "[true]"
    },
    {
      "displayName": "DisplayName",
      "oldValue": "[]",
      "newValue": "[\"Bach Door\"]"
    },
    {
      "displayName": "MailNickname",
      "oldValue": "[]",
      "newValue": "[\"BachD\"]"
    },
    {
      "displayName": "StsRefreshTokensValidFrom",
      "oldValue": "[]",
      "newValue": "[\"2024-03-18T03:03:15Z\"]"
    },
    {
      "displayName": "UserPrincipalName",
      "oldValue": "[]",
      "newValue": "[\"BachD@aceresponder.com\"]"
    }
```


ဒါကြောင့် App Hijacking ပြီးသွားတဲ့နောက် Attacker က လုပ်ခဲ့တဲ့ အဓိက Action ဟာ **User အသစ်တစ်ယောက်ကို Create လုပ်ခဲ့တာ** ဖြစ်ပါတယ်။ ဒီ User အသစ်ဟာ Persistence ရဖို့၊ နောက်ထပ် Attack တွေ ဆက်လုပ်ဖို့ အသုံးပြုနိုင်တဲ့ Backdoor Account ဖြစ်နိုင်ပါတယ်။
