Active Directory DACL Abuse

ယနေ့ခေတ် အဖွဲ့အစည်းများမှာ ကွန်ပျူတာစနစ်များကို အသုံးပြုပြီး အရေးကြီးတဲ့ အချက်အလက်တွေကို သိမ်းဆည်းထားရသလို၊ ဝန်ထမ်းတွေ အလုပ်လုပ်ရတာလည်း အလွယ်တကူဖြစ်စေဖို့ network infrastructure တွေကို အလွန်အမင်း အသုံးပြုလာကြပါတယ်။ ဒီလို အသုံးပြုလာတာနဲ့အမျှ အချက်အလက်တွေကို **လုံခြုံအောင် ထိန်းသိမ်းဖို့ အသုံးပြုရတဲ့ security infrastructure ကလည်း ပိုပြီး ရှုပ်ထွေးလာပါတယ်။** ဒီအရာတွေထဲမှာ Active Directory ဆိုတာ Windows အခြေခံ network environment တွေအတွက် အလွန်အရေးကြီးတဲ့ အစိတ်အပိုင်းတစ်ခု ဖြစ်ပါတယ်။

> **Active Directory ဆိုတာက အသုံးပြုသူတွေကို စီမံခန့်ခွဲပေးတဲ့ စနစ်တစ်ခုဖြစ်ပြီး၊ ဘယ်သူက ဘယ် ကွန်ပျူတာကို သုံးနိုင်မလဲ၊ ဘယ်ဖိုင်ကို ဖတ်နိုင်မလဲ၊ ပြင်နိုင်မလဲဆိုတာတွေကို ထိန်းချုပ်ပေးပါတယ်။**

အလွယ်ပြောရရင် Active Directory က network တစ်ခုလုံးရဲ့ **"အခွင့်အရေး စီမံခန့်ခွဲရေးဗဟို"** လို့ ဆိုလို့ရပါတယ်။ ဒါပေမယ့် ဒီလို အရေးကြီးတဲ့ စနစ်ဖြစ်တဲ့အတွက် configuration တွေကလည်း အလွန်ရှုပ်ထွေးပြီး၊ အမှားလုပ်မိရင် လုံခြုံရေး ပြဿနာတွေ ပေါ်လာနိုင်ပါတယ်။ ဒီလို အားနည်းချက်တွေဟာ administrator တွေအတွက် စိန်ခေါ်မှုဖြစ်သလို၊ တိုက်ခိုက်သူတွေ အတွက်တော့ အခွင့်အရေးတစ်ခု ဖြစ်လာပါတယ်။

> Active Directory ထဲမှာ object တစ်ခုချင်းစီကို ဘယ်သူက ဘာလုပ်ခွင့်ရှိလဲဆိုတာကို ထိန်းချုပ်ဖို့ **Discretionary Access Control List (DACL)** ဆိုတဲ့ mechanism ကို အသုံးပြုပါတယ်။

DACL ဆိုတာက object တစ်ခုအပေါ်မှာ **ဘယ် user ဒါမှမဟုတ် ဘယ် group က ဖတ်ခွင့်ရှိလဲ၊ ပြင်ခွင့်ရှိလဲ၊ ဖျက်ခွင့်ရှိလဲ** ဆိုတာတွေကို သတ်မှတ်ထားတဲ့ permission စာရင်းတစ်ခုလိုပါပဲ။ Administrator တွေ အနေနဲ့ DACL ကို အသုံးပြုပြီး object အများကြီးကို တစ်ခါတည်း permission ပေးနိုင်သလို၊ object တစ်ခုတည်းကိုပဲ အလွန်အသေးစိတ် permission ပေးလည်း ရပါတယ်။

ဒါပေမယ့် လက်တွေ့လုပ်ငန်းခွင်မှာ DACL တွေကို သတ်မှတ်တဲ့အခါ အမှားတွေ ဖြစ်လေ့ရှိပါတယ်။ ဥပမာ အားဖြင့်

- မလိုအပ်ဘဲ user တစ်ယောက်ကို အရေးကြီးတဲ့ object ကို ပြင်ခွင့်ပေးမိတာ၊
- ဒါမှမဟုတ် privilege မြင့်တဲ့ group ကို မသင့်တော်တဲ့ permission ပေးထားတာမျိုး

ဖြစ်နိုင်ပါတယ်။ ဒီလို misconfiguration တွေဟာ ပုံမှန်ကြည့်ရင် မမြင်သာပေမယ့်၊ **တိုက်ခိုက်သူတစ်ယောက်အတွက်တော့ privilege escalation လုပ်ဖို့၊ အခြား user တွေရဲ့ account ကို ထိန်းချုပ်ဖို့ အသုံးချနိုင်တဲ့ လမ်းကြောင်းတွေ ဖြစ်လာနိုင်ပါတယ်။ ဒီအရာကိုပဲ Active Directory DACL Abuse လို့ ခေါ်ပါတယ်။**

အဲ့ဒီကြောင့် incident responder၊ threat hunter၊ detection engineer လို security အလုပ်လုပ်သူတွေ အတွက် DACL မှာ ဘယ်နေရာတွေမှာ အားနည်းချက်တွေ ဖြစ်လေ့ရှိလဲ၊ တိုက်ခိုက်သူတွေက ဒီ permission အမှားတွေကို ဘယ်လို enumerate လုပ်ပြီး အသုံးချနိုင်လဲ ဆိုတာကို နားလည်ထားခြင်းက အလွန်အရေးကြီးပါတယ်။ ဒီအရာကို နားလည်ထားရင် မူမမှန်တဲ့ activity တွေကို စောစောသိနိုင်ပြီး၊ တိုက်ခိုက်မှု ဖြစ်နေတယ်ဆိုရင်လည်း ထိရောက်စွာ တုံ့ပြန်နိုင်ပါတယ်။

Active Directory Permissions

လက်တွေ့မှာ ကွန်ပျူတာတွေဗရှိတယ်၊ အသုံးပြုသူတွေရှိတယ်၊ printer တွေရှိတယ်၊ file တွေရှိတယ်ဆိုပြီး ဒီအရာတွေ အားလုံးပေါင်းပြီး network တစ်ခုကို ဖန်တီးထားပါတယ်။ Active Directory ဆိုတာက ဒီလို network ထဲမှာရှိတဲ့ အရင်းအမြစ်တွေ အကြောင်းအရာကို စုစည်းသိမ်းဆည်းထားတဲ့ **ချက်အလက်ဗဟိုတစ်ခုဖြစ်ပြီး**၊ ဘယ်သူက ဘာကို သုံးခွင့်ရှိလဲဆိုတာကို ထိန်းချုပ်ပေးတဲ့ စနစ်ပါ။ အလွယ်ပြောရရင် Active Directory က network ထဲက အရာအားလုံးအတွက် **"ဘယ်သူ ဘာလုပ်လို့ရမလဲ"** ဆိုတာကို ဆုံးဖြတ်ပေးတဲ့ အုပ်ချုပ်ရေးစနစ်တစ်ခု ဖြစ်ပါတယ်။

Securable Object

Active Directory က ဒီလို access control ကို လုပ်ပေးရာမှာ rule တွေကို အသုံးပြုပါတယ်။ ဒီ rule တွေကို securable object တွေအပေါ်မှာ ချိတ်ထားပါတယ်။ Securable object ဆိုတာက security descriptor တစ်ခု ချိတ်ထားလို့ရတဲ့ object ကို ပြောတာပါ။ ဥပမာအနေနဲ့

- user account တစ်ခု၊
- group တစ်ခု၊
- computer တစ်လုံး၊
- Organizational Unit တစ်ခု၊
- file သို့မဟုတ် folder တစ်ခု၊
- registry key တစ်ခု၊
- process တစ်ခုစတာတွေ

အားလုံးဟာ securable object တွေ ဖြစ်ပါတယ်။ ဆိုလိုတာက ဒီ object တွေအပေါ်မှာ permission သတ်မှတ်နိုင်ပါတယ်။

Security Descriptor

Object တစ်ခုချင်းစီမှာ security descriptor ဆိုတာ ရှိပါတယ်။ Security descriptor က object ရဲ့ လုံခြုံရေးဆိုင်ရာ အချက်အလက်အားလုံးကို စုထားတဲ့ အရာဖြစ်ပြီး၊ ဘယ်သူပိုင်ဆိုင်လဲ၊ ဘယ်သူတွေကို ဘယ်လိုအခွင့်အရေးပေးထားလဲ ဆိုတာတွေ ပါဝင်ပါတယ်။ ဒီ security descriptor ထဲမှာ အဓိကအားဖြင့်

- object owner နဲ့
- primary group၊ DACL လို့ခေါ်တဲ့
- discretionary access control list နဲ့ SACL လို့ခေါ်တဲ့ system access control list ဆိုပြီး အပိုင်းသုံးပိုင်း ပါဝင်ပါတယ်။

DACL

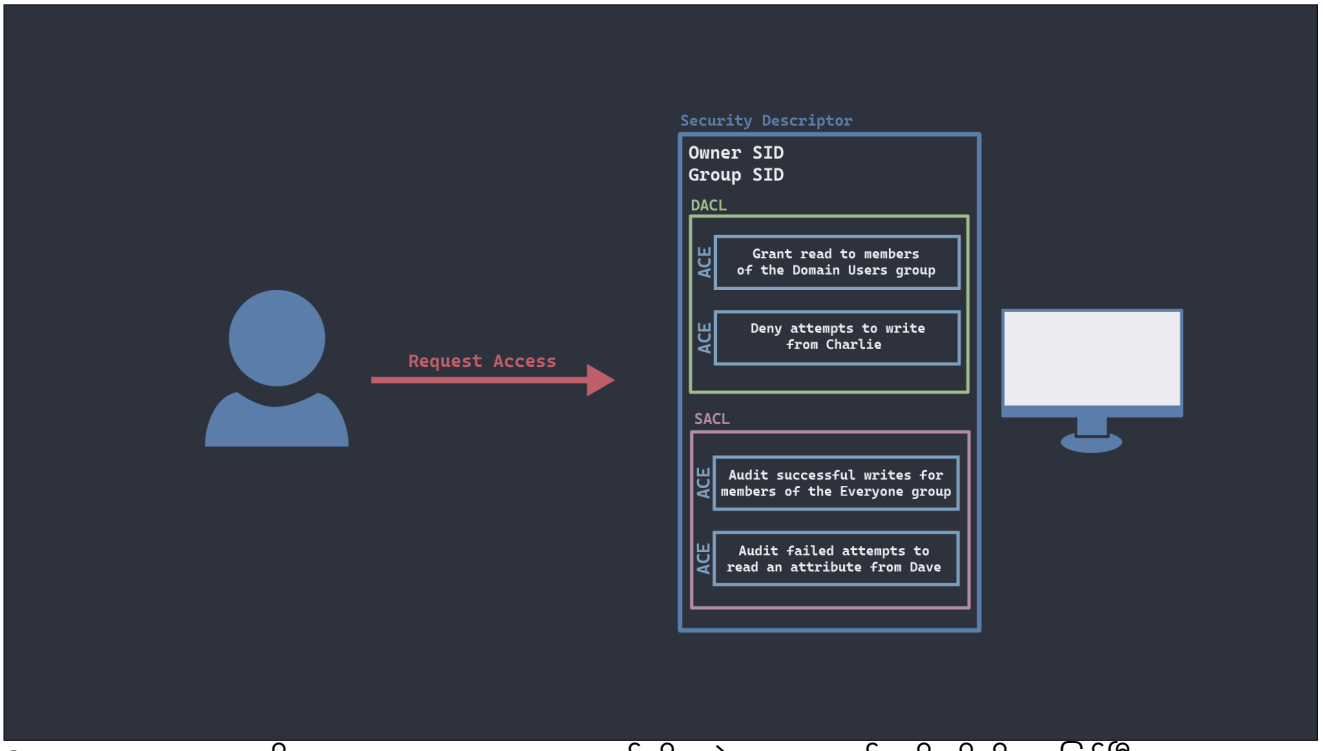
ဒီ module မှာတော့ အဓိကအားဖြင့် DACL ကိုပဲ စိတ်ဝင်စားပါတယ်။ DACL ဆိုတာက object တစ်ခုကို ဘယ် security principal က ဘယ်လို access ရရှိမလဲဆိုတာကို သတ်မှတ်ထားတဲ့ စာရင်းပါ။ DACL နဲ့ SACL (system access control list) နှစ်ခုစလုံးဟာ **ACE လို့ခေါ်တဲ့ access control entry** တွေကို စုထားတာ ဖြစ်ပါတယ်။

- DACL (Discretionary Access Control List) ထဲက ACE တွေကတော့ "ဒီ user သို့မဟုတ် group ကို ဒီ object ပေါ်မှာ ဘာလုပ်ခွင့်ပေးမလဲ၊ မပေးဘူးလား" ဆိုတာကို ဖော်ပြပါတယ်။

- SACL (System Access Control List) ထဲက ACE တွေကတော့ “ဒီ object ကို access လုပ်ဖို့ ကြိုးစားတဲ့အခါ ဘယ်လို audit လုပ်မလဲ” ဆိုတာကို သတ်မှတ်ပါတယ်။

ဥပမာအားဖြင့် အရေးကြီးတဲ့ object ကို ဖတ်ဖို့ သို့မဟုတ် ပြင်ဖို့ ကြိုးစားရင် log entry တစ်ခု ထွက်စေမယ် ဆိုပြီး သတ်မှတ်နိုင်ပါတယ်။ ဒီလို audit အချက်အလက်တွေဟာ detection engineer တွေနဲ့ incident responder တွေအတွက် အလွန်အသုံးဝင်ပါတယ်။

Security Principal



Security principal ဆိုတာက authentication လုပ်လို့ရတဲ့ entity တစ်ခုကို ဆိုလိုတာဖြစ်ပြီး user၊ computer၊ group စတာတွေ ပါဝင်ပါတယ်။ Active Directory အတွင်းမှာ security principal တစ်ခုချင်းစီကို SID လို့ခေါ်တဲ့ **security identifier** နဲ့ ကိုယ်စားပြုထားပါတယ်။ **SID က unique ဖြစ်ပြီး system အနေနဲ့ identity ကို ခွဲခြားဖို့ အသုံးပြုပါတယ်။**

Access Control Entry(ECE)

DACL ထဲမှာရှိတဲ့ ACE တစ်ခုချင်းစီမှာ အရေးကြီးတဲ့ အချက်အလက်အချို့ ပါဝင်ပါတယ်။ အဲ့ဒါက

- ဒီ ACE က ဘယ် security principal အတွက်လဲ၊
- ဘယ် access rights တွေပေးထားလဲ၊
- access ကို ခွင့်ပြုထားတာလား ပိတ်ပင်ထားတာလား၊ နောက်ပြီး
- child object တွေကို inheritance လုပ်မလားဆိုတဲ့ rule တွေပါ ပါဝင်ပါတယ်။

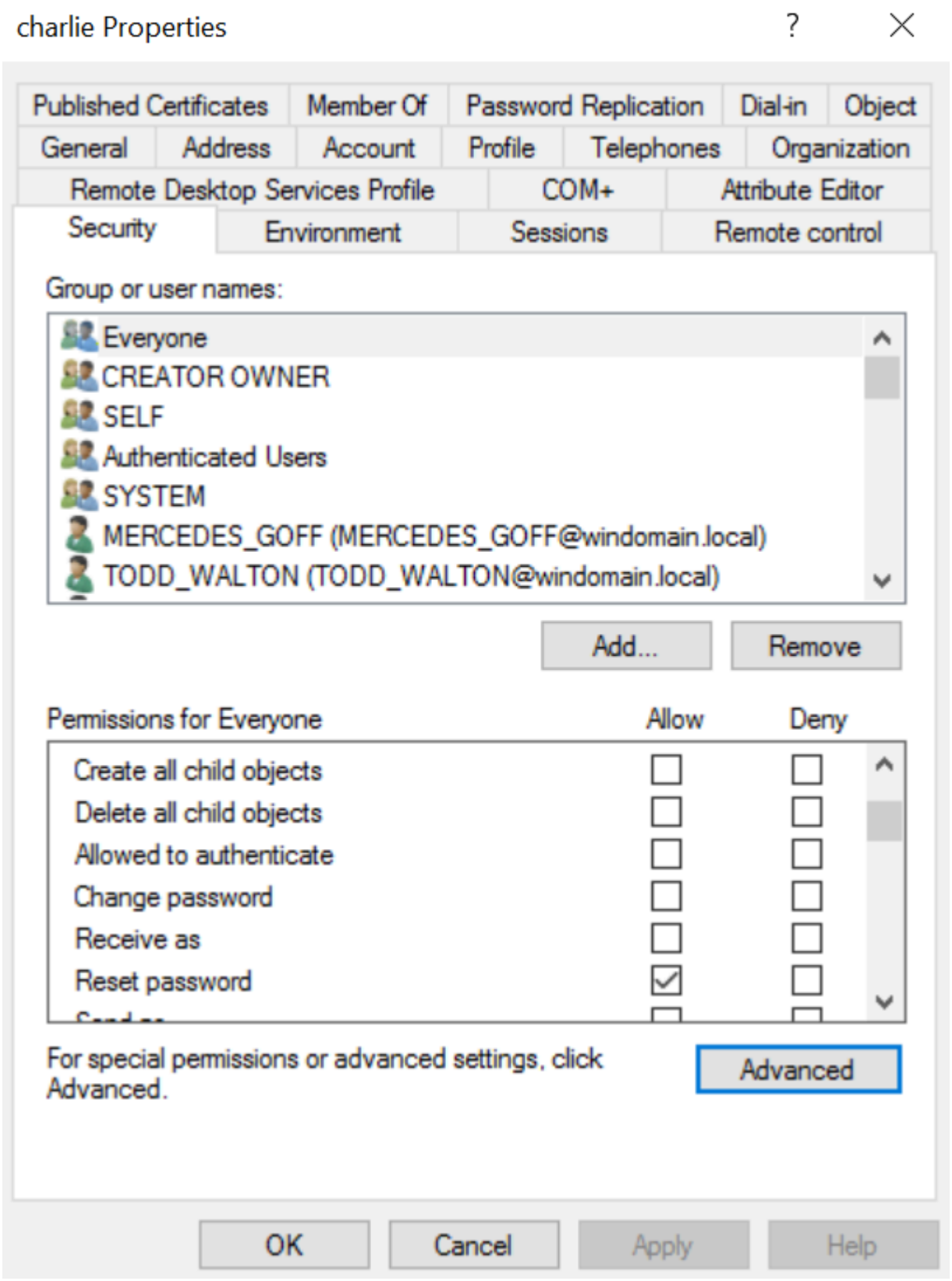
ဒီ ACE တွေကို *မမှန်ကန်စွာ* သတ်မှတ်ထားရင် attacker တွေအတွက် အခွင့်အရေးကြီးတွေ ဖြစ်လာနိုင်ပါတယ်။ တိုက်ခိုက်သူတွေအတွက် အလွန်တန်ဖိုးရှိတဲ့ access right တွေက
Access Right	Value for attackers
GenericAll လိုမျိုး	object ကို အပြည့်အဝ ထိန်းချုပ်ခွင့်ရတာ၊
GenericWrite လိုမျိုး	object ကို ပြင် ခွင့်ရတာ၊
WriteOwner လိုမျိုး	object ရဲ့ owner ဖြစ်အောင် ပြောင်းနိုင်တာ၊
WriteDACL လိုမျိုး	permission ကို ကိုယ်တိုင် ပြန်ပြင်နိုင်တာ၊

ForceChangePassword လိုမျိုး ၂အရင် password ကို မသိဘဲ user တစ်ယောက်ရဲ့ password ကို reset လုပ်နိုင်တာတွေ

ဖြစ်ပါတယ်။ လက်တွေ့မှာ permission တွေက ဒီထက်ပိုပြီး အသေးစိတ်တောင် သတ်မှတ်လို့ရပေမယ့် ဒီ module မှာတော့ ဒီလို အနှစ်ချုပ်ကြီးတဲ့ permission တွေကိုပဲ အဓိကထား စဉ်းစားပါတယ်။

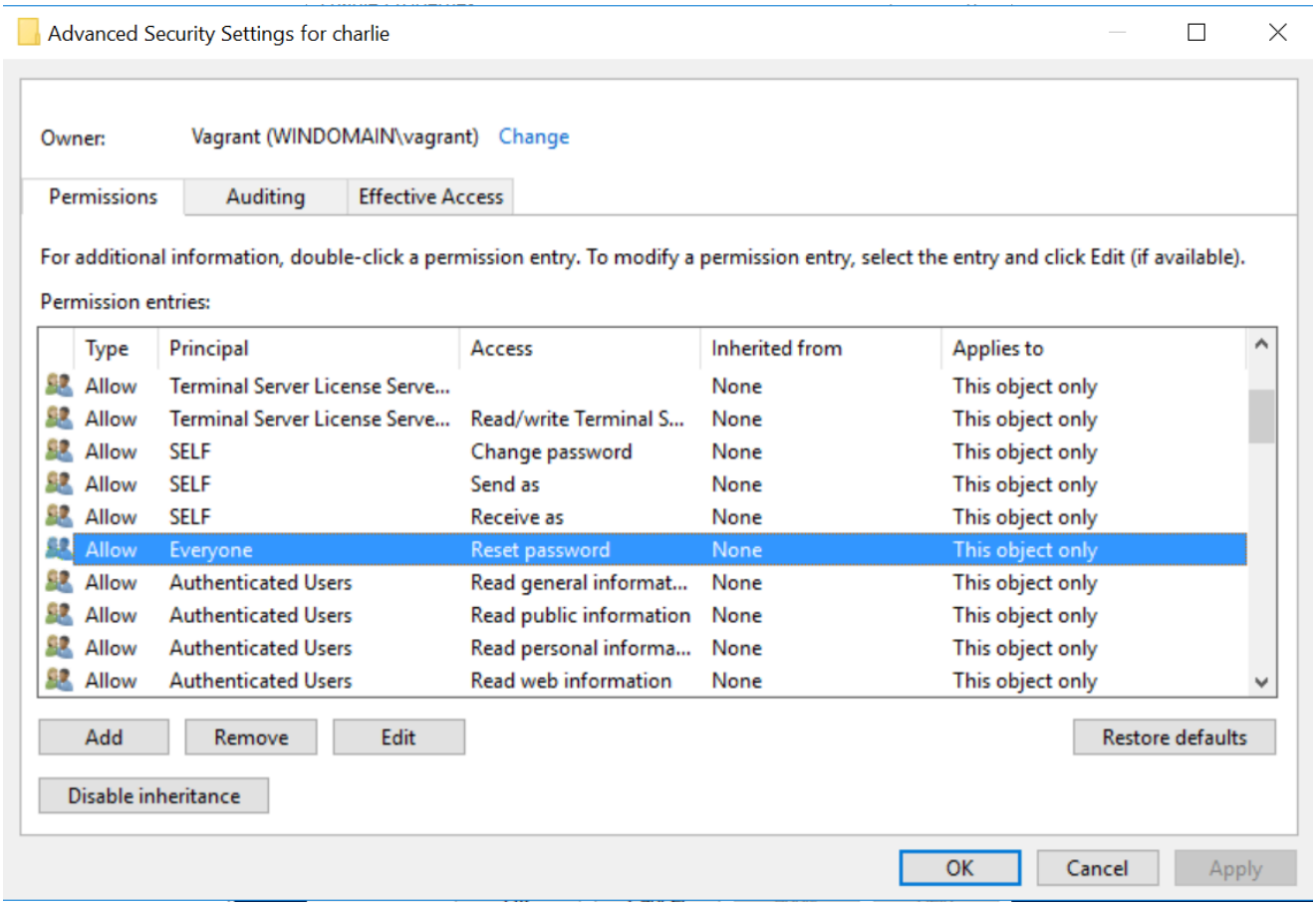
Active Directory permission တွေကို အသစ်လေ့လာနေတဲ့သူအတွက် ဒီအရာတွေက အလွန်ရှုပ်ထွေးတယ်လို့ ထင်ရနိုင်ပါတယ်။ အမှန်တကယ်လည်း လက်တွေ့မှာ အတော်ရှုပ်ပါတယ်။ နားလည်လွယ်အောင် စဉ်းစားရင် network firewall ACL ကို စဉ်းစားသလို စဉ်းစားနိုင်ပါတယ်။ Firewall မှာ source က destination ကို ဆက်သွယ်ဖို့ ကြိုးစားရင် rule တွေကို အပေါ်ကနေ အောက်ထိ စစ်သလိုပဲ၊ Active Directory မှာလည်း user တစ်ယောက်က object တစ်ခုကို access လုပ်ဖို့ ကြိုးစားရင် system က DACL ထဲက ACE တွေကို အစဉ်လိုက် စစ်ပြီး ခွင့်ပြုမလား၊ ပိတ်မလား ဆုံးဖြတ်ပါတယ်။

လက်တွေ့ဥပမာအနေနဲ့ Active Directory Users and Computers ဆိုတဲ့ tool ကို သုံးပြီး object တစ်ခုရဲ့ permission ကို ကြည့်နိုင်ပါတယ်။ User object တစ်ခုကို **right-click လုပ်ပြီး Properties ကို ဝင်လိုက်ရင် Security tab မှာ security descriptor ကို အကျဉ်းချုပ်ပုံစံနဲ့** မြင်ရပါတယ်။

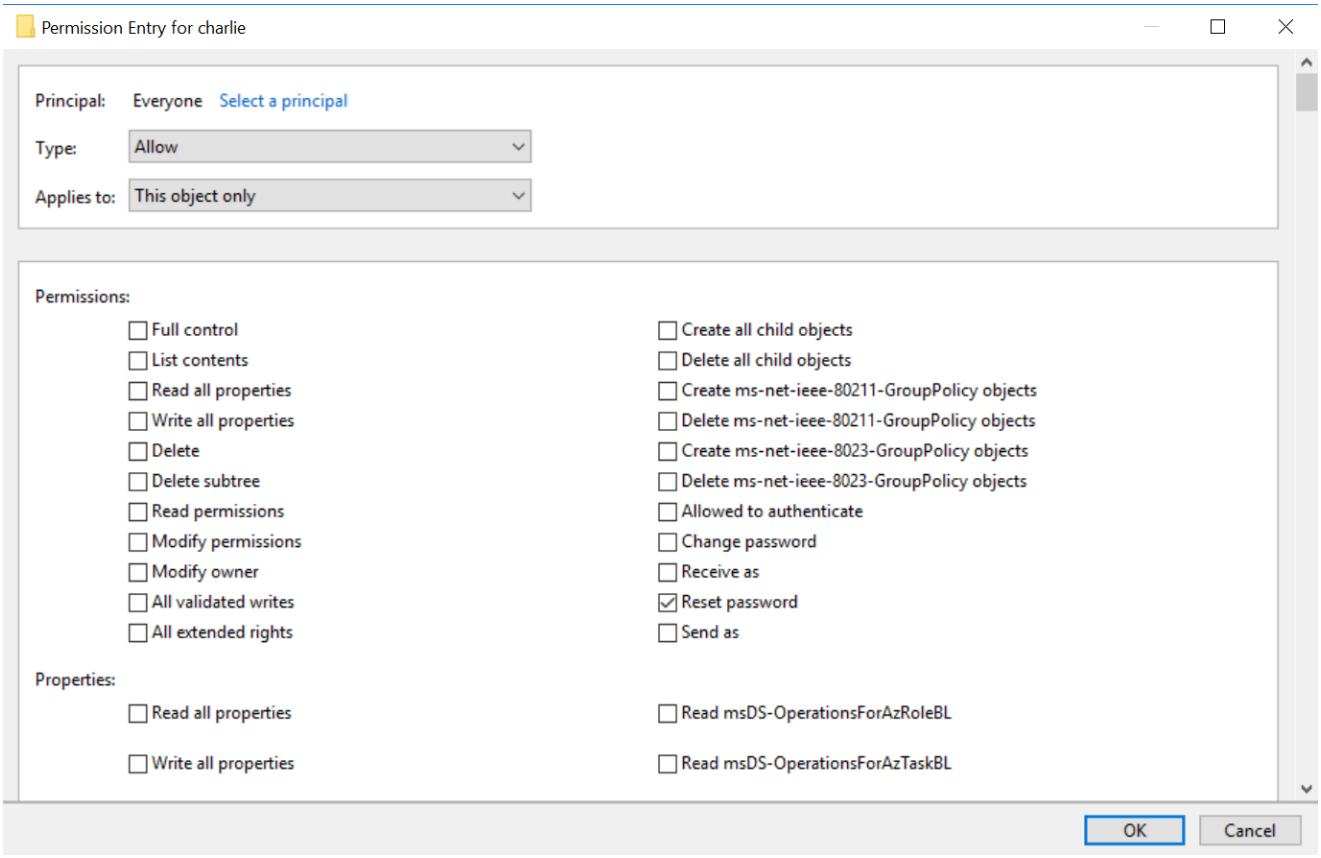


အဲ့ဒီမှာ Group or usernames ဆိုတဲ့နေရာမှာ security principal တွေရဲ့ စာရင်းကို မြင်ရပြီး၊ အောက်မှာတော့ အဲ့ဒီ principal တွေက object အပေါ်မှာ ဘာလုပ်ခွင့်ရှိလဲဆိုတာ ပြထားပါတယ်။

Advanced ကို ဝင်လိုက်ရင် security descriptor ကို ပိုပြီး အသေးစိတ်မြင်ရပါတယ်။



Permissions tab က DACL ကို ကိုယ်စားပြုပြီး Auditing tab ကတော့ SACL ကို ကိုယ်စားပြုပါတယ်။ ဒီနေရာမှာ ဘယ် user ၊ group၊ computer က “charlie” ဆိုတဲ့ user object ကို ဘာလုပ်လို့ရလဲဆိုတာကို အသေးစိတ်မြင်နိုင်ပါတယ်။ **DACL ထဲက ACE တွေကို ဒီနေရာမှာပဲ ထည့်လို့ရ၊ ဖျက်လို့ရ၊ ပြင်လို့ရပါတယ်။** ဥပမာပြထားတဲ့ ACE က Everyone group အတွက် access-allow ဖြစ်ပြီး inheritance မလုပ်ထားတဲ့ ACE ဖြစ်ပါတယ်။ ဆိုလိုတာက ဒီ permission ကို “charlie” object ပေါ်မှာ တိုက်ရိုက်ပေးထားတာ ဖြစ်ပါတယ်။



ACE ကို ဖွင့်ကြည့်လိုက်ရင် Everyone group ရဲ့ member တွေက “charlie” အပေါ်မှာ ဘယ် permission တွေ ရထားလဲဆိုတာကို တိတိကျကျ မြင်ရပါတယ်။ ဒီအချက်က အလွန်အန္တရာယ်ကြီးပါတယ်။

တိုက်ခိုက်သူတစ်ယောက်ရဲ့ မျက်လုံးနဲ့ ကြည့်မယ်ဆိုရင် PowerView လို tool တွေကို သုံးပြီး ဒီ ACE တွေကို enumerate လုပ်နိုင်ပါတယ်။

```
PS C:\Users\alice\Downloads> Get-ObjectAcl -SamAccountName charlie -ResolveGUIDs |?{$_.SecurityIdentifier -eq "S-1-1-0"}


AceQualifier           : AccessAllowed
ObjectDN               : CN=charlie,CN=Users,DC=windomain,DC=local
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : User-Force-Change-Password
ObjectSID              : S-1-5-21-1072563919-790901262-2159826448-4197
InheritanceFlags       : None
BinaryLength           : 40
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent
IsCallback             : False
PropagationFlags       : None
SecurityIdentifier     : S-1-1-0
AccessMask             : 256
AuditFlags             : None
IsInherited            : False
AceFlags               : None
InheritedObjectAceType : All
OpaqueLength           : 0
```

PowerView မှာ ပြထားတဲ့အတိုင်း attacker က User-Force-Change-Password ဆိုတဲ့ extended right ကို တွေ့နိုင်ပါတယ်။ ဒီ permission ရဲ့ display name က Reset Password ဖြစ်ပြီး ADUC GUI ထဲမှာ မြင်ရတဲ့ permission နဲ့ အတူတူပါပဲ။ အရေးကြီးတာက ForceChangePassword permission က အရင် password ကို သိစရာမလိုဘဲ password ကို ပြောင်းခွင့်ပေးတာ ဖြစ်ပါတယ်။

> ဒီလို misconfiguration ဖြစ်နေတဲ့အခါ Everyone group ထဲက မည်သူမဆို “charlie” ရဲ့ password ကို reset လုပ်နိုင်ပါတယ်။

Everyone group ဆိုတာက နာမည်အတိုင်းပဲ network ထဲက လူအားလုံး ပါဝင်တဲ့ group ဖြစ်ပါတယ်။ ဒါကြောင့် attacker တစ်ယောက်က network ထဲမှာ foothold တစ်ခုရပြီဆိုရင် ဒီ permission အမှားကို ရှာတွေ့တာနဲ့ “charlie” account ကို လွယ်လွယ်ကူကူ takeover လုပ်နိုင်ပါတယ်။

Attack Paths

တိုက်ခိုက်သူတစ်ယောက်က Windows environment တစ်ခုထဲကို foothold ရပြီဆိုတာနဲ့ Active Directory ကို စတင် စူးစမ်းလေ့လာတတ်ပါတယ်။

> Foothold ဆိုတာက user privilege နိမ့်နိမ့်နဲ့ system ထဲ ဝင်နိုင်သွားတဲ့ အခြေအနေကို ဆိုလိုတာပါ။

ဒီအဆင့်မှာ attacker ရဲ့ ရည်ရွယ်ချက်က **"ဒီ environment ထဲမှာ ဘယ် permission တွေ၊ ဘယ် configuration အမှားတွေကို အသုံးချလို့ရနိုင်မလဲ"** ဆိုတာကို နားလည်ဖို့ပါ။ အဲ့ဒီလို နားလည်ဖို့ attacker တွေက LDAP ဆိုတဲ့ protocol ကို အသုံးပြုပြီး enumeration လုပ်ကြပါတယ်။

LDAP (Lightweight Directory Access Protocol)

LDAP enumeration ဆိုတာက domain controller ကို မေးမြန်းပြီး Active Directory ထဲမှာ ရှိတဲ့

- object တွေ၊

- user တွေ၊
- group တွေ၊
- permission ဆက်နွယ်မှုတွေကို စုဆောင်းတာပါ။

ဒီအလုပ်ကို attacker က သူရထားတဲ့ low-privilege user account နဲ့ပဲ လုပ်နိုင်ပါတယ်။ LDAP က Active Directory ရဲ့ အခြေခံ protocol ဖြစ်တဲ့အတွက် permission မှန်ရင် information အများကြီးကို ပြန်ပေးပါတယ်။ Attacker တွေဟာ LDAP enumeration ရလာဒ်တွေကို local group enumeration၊ session enumeration နဲ့ vulnerability scanning လို technique တွေနဲ့ ပေါင်းပြီး environment ကို ပို ပြီး နက်နက်ရှိုင်းရှိုင်း နားလည်လာကြပါတယ်။

> Foothold ကနေ attacker ရဲ့ နောက်ဆုံးရည်မှန်းချက်အထိ သွားတဲ့ လမ်းကြောင်းတစ်လျှောက်လုံးကို attack path လို့ ခေါ်ပါတယ်။

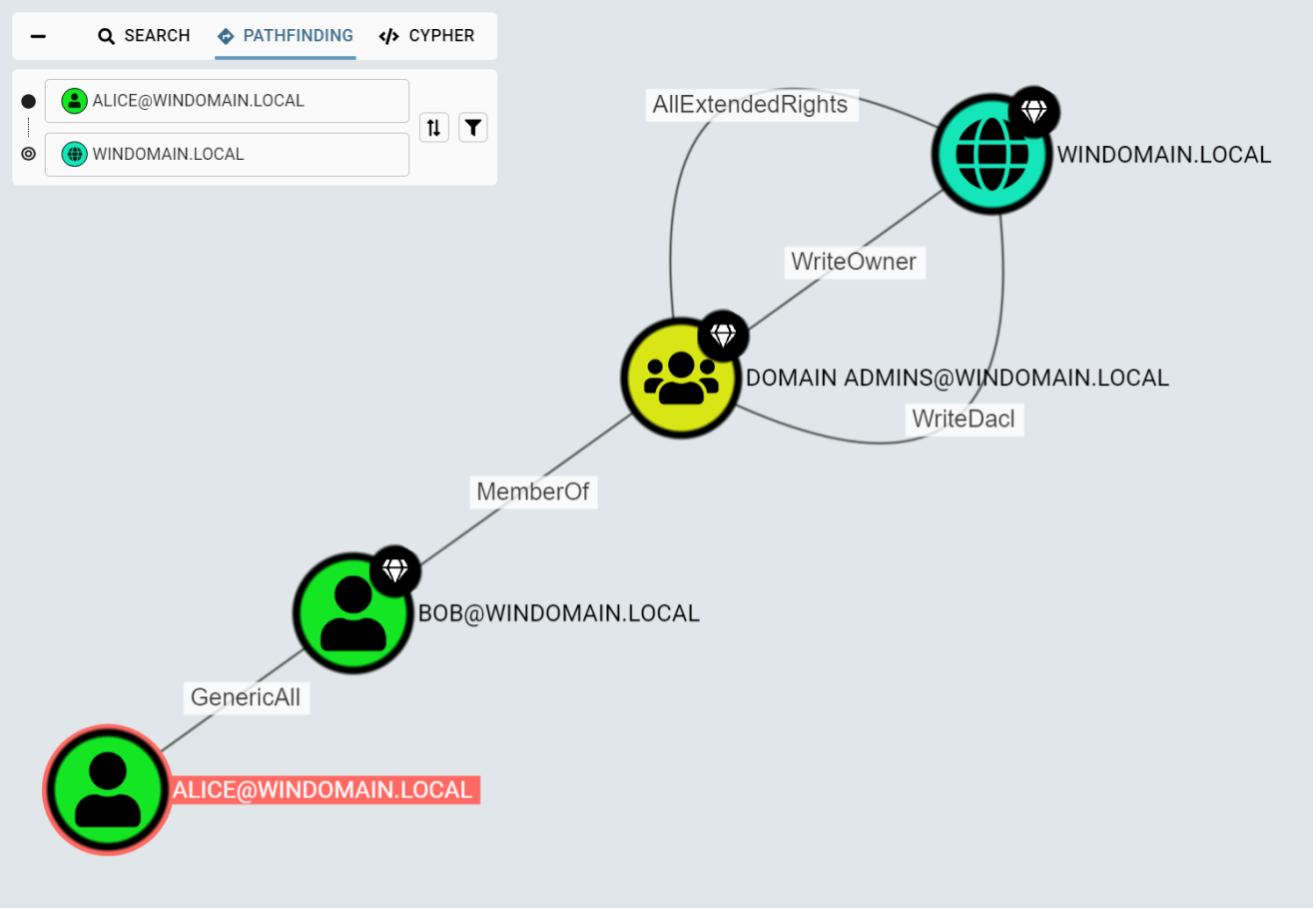
Attacker ဟာ အစပိုင်းမှာ privilege နိမ့်နိမ့်နဲ့ စပြီး misconfiguration တွေ၊ permission အမှားတွေကို အသုံးချရင်း privilege ကို တဖြည်းဖြည်း မြှင့်တင်သွားပါတယ်။ Attack path တစ်ခုကို ရွေးချယ်ရာမှာ attacker က environment ရဲ့ detection capability ကို စဉ်းစားပါတယ်။ ဆိုလိုတာက “ဘယ် လမ်းကြောင်းက ပိုမိုလွယ်ကူပြီး ဖမ်းမိနိုင်ခြေ နည်းလဲ” ဆိုတာကို စဉ်းစားပြီး attack path တစ်ခုကို ရွေးပါ တယ်။

BloodHound

ဒီ attack path တွေကို မြင်သာအောင် ပြပေးနိုင်တဲ့ tool အနေနဲ့ အလွန်နာမည်ကြီးတာက BloodHound ဖြစ်ပါတယ်။ BloodHound ဆိုတာ LDAP နဲ့ အခြား technique တွေကနေ စုထားတဲ့ Active Directory အချက်အလက်တွေကို ယူပြီး graph ပုံစံနဲ့ ပြပေးတဲ့ tool ဖြစ်ပါတယ်။ ဒီ graph ထဲမှာ **object တွေ အကြား ဆက်နွယ်မှုတွေ၊ permission အမှားတွေကို တစ်ချက်ကြည့်တာနဲ့ မြင်နိုင်ပါတယ်။** BloodHound အတွက် data စုတဲ့ collector tool ကတော့ Sharphound ဖြစ်ပြီး GUI ထဲကနေတောင် download လုပ် နိုင်တဲ့အထိ အသုံးပြုရ လွယ်ကူပါတယ်။

Defender ဘက်ကနေကြည့်ရင် BloodHound ကို ကျွမ်းကျင်အောင် သုံးနိုင်ခြင်းက အလွန်အရေးကြီးပါ တယ်။ အတွေ့အကြုံများတဲ့ incident responder တစ်ယောက်က တိုက်ခိုက်မှုကို မကြုံတင်သိထားဘဲနဲ့ ဖြစ်လာမယ့် လမ်းကြောင်းကို ခန့်မှန်းနိုင်တတ်တာဟာ Active Directory ကို attacker တစ်ယောက်လို နားလည်ထားလို့ပါပဲ။ ယနေ့ခေတ်မှာ BloodHound ဟာ ၂၀၁၂ ခုနှစ်လောက်က vulnerability scanner တွေလို အရေးပါလာပါတယ်။ Preventive နဲ့ detective နှစ်ဖက်စလုံးအတွက် BloodHound ကို workflow ထဲ ထည့်သုံးရင် Active Directory ရဲ့ အားနည်းချက်တွေကို သဘောတရားအနေနဲ့ နားလည်လာနိုင်ပါတယ်။

အရမ်းရှိုးရှင်းတဲ့ ဥပမာတစ်ခုကို စဉ်းစားကြည့်ရအောင်။



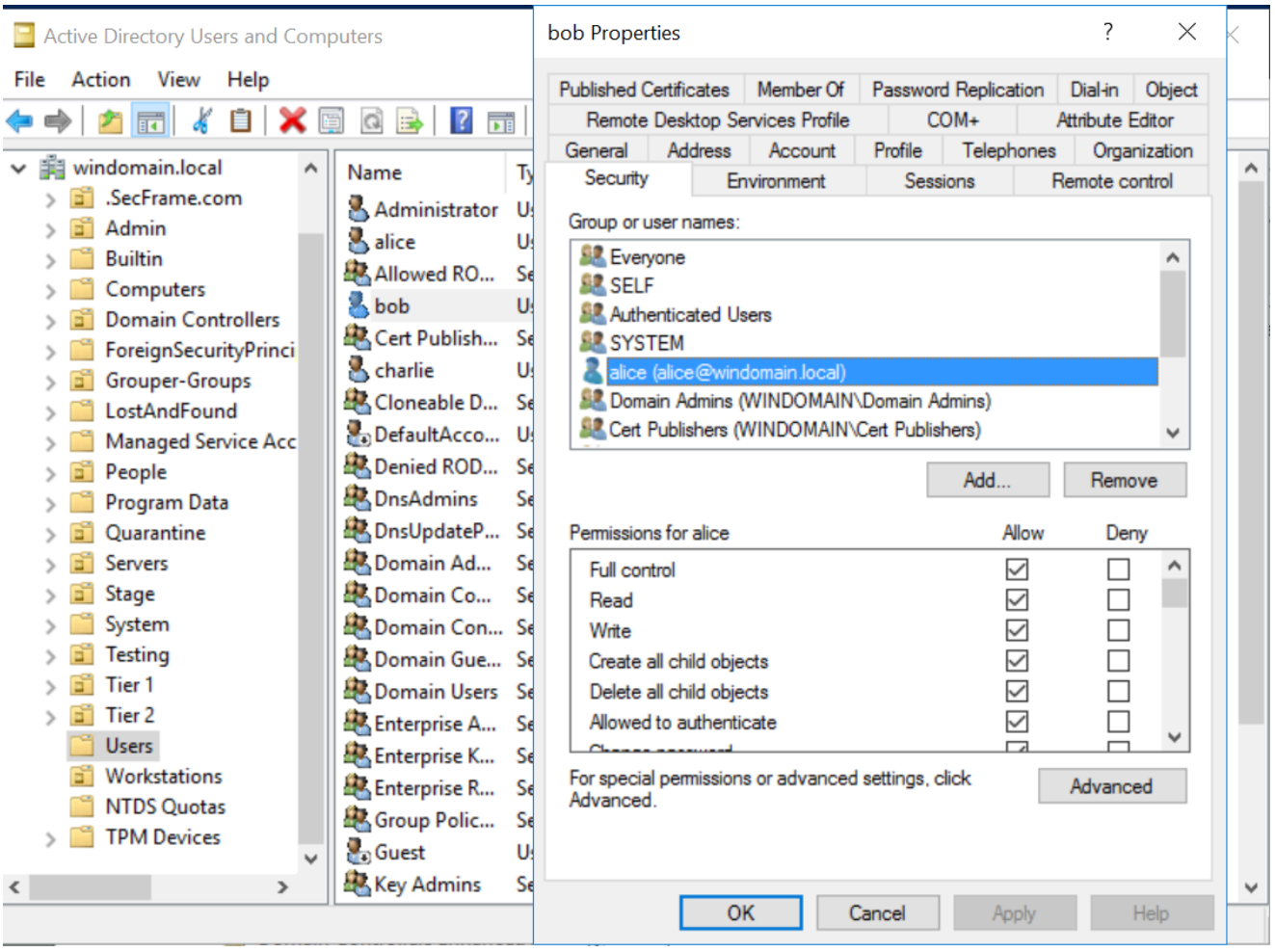
“Alice” ဆိုတဲ့ user က “bob” ဆိုတဲ့ user object ပေါ်မှာ **GenericAll permission** ရထားတယ်ဆိုပါစို့။

> GenericAll ဆိုတာ object ကို အပြည့်အဝ ထိန်းချုပ်နိုင်တဲ့ permission ဖြစ်ပါတယ်။

ဒီအခြေအနေမှာ attacker က Alice account ကို compromise လုပ်နိုင်ပြီဆိုရင် Bob ရဲ့ account ကို လုံးဝ takeover လုပ်နိုင်ပါတယ်။ Bob က Domain Admin ဖြစ်နေတယ်ဆိုရင် attacker ဟာ တစ်ဆင့်တည်း Domain Admin privilege ကို ရရှိသွားပါပြီ။

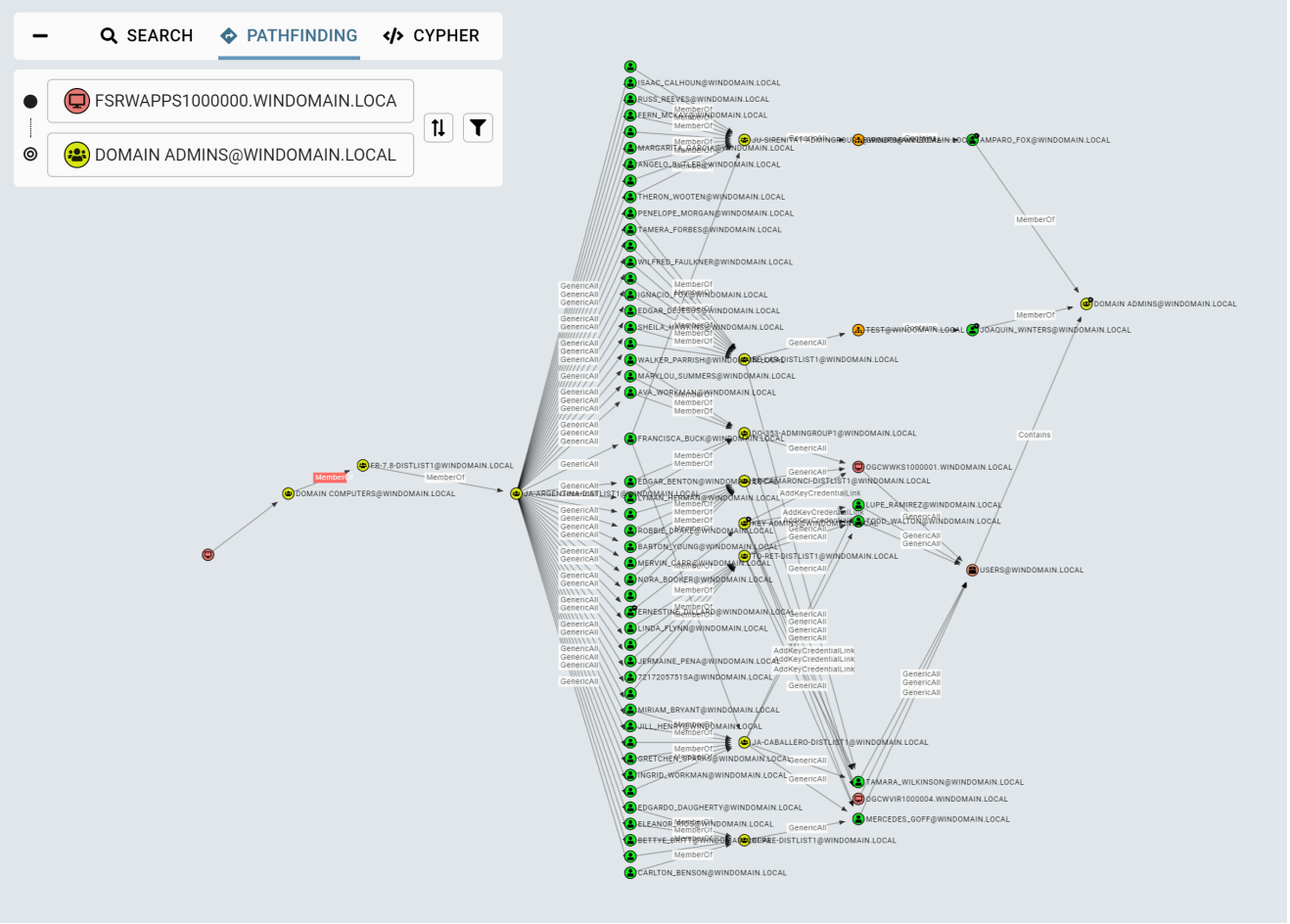
ဒီ attack path က ကြည့်ရင်လည်း အလွန်လွယ်လွယ်နားလည်နိုင်ပါတယ်။ Alice ကို compromise လုပ်တယ်၊ Bob ကို takeover လုပ်တယ်၊ Domain Admin ရပြီဆိုတဲ့ လမ်းကြောင်းပါပဲ။ BloodHound ထဲမှာ ဒီလို path တွေ ရှိနေပေမယ့် environment ကြီးလာရင် ဒီလို misconfiguration တွေကို ရှာဖို့ Cypher query တွေ အသုံးပြုရတတ်ပါတယ်။ အဲ့ဒါကြောင့် tool ကို သုံးတတ်ရုံနဲ့ မလုံလောက်ဘဲ logic ကိုပါ နားလည်ဖို့ လိုပါတယ်။

ဒီလို permission အမှားတွေ ဘာကြောင့် ဒီလောက်များနေလဲလို့ မေးစရာ ဖြစ်လာနိုင်ပါတယ်။ Administrator တွေက ဘာကြောင့် GenericAll လို permission တွေကို လွယ်လွယ်ပေးထားပြီး ဒီလို အန္တရာယ်ကြီးတဲ့ misconfiguration ကို မမြင်မိကြတာလဲ ဆိုတဲ့ မေးခွန်းပါ။



အကြောင်းအရင်းက ADUC လို GUI ထဲမှာ ကြည့်တဲ့အခါ permission ဆက်နွယ်မှုတွေကို attack path အနေနဲ့ မမြင်ရတာကြောင့်ပါ။ Alice-bob ဥပမာကို ADUC ထဲမှာ ကြည့်လိုက်ရင် အန္တရာယ်ကြီးတာကို ချက်ချင်း မသဘောပေါက်နိုင်ပါဘူး။

နောက်ထပ် ပို၍ပေါ်ထွေးတဲ့ attack path တစ်ခုကို စဉ်းစားကြည့်ရအောင်။



Domain Admin တွေအပေါ် DACL ကို သေချာထိန်းထားတယ်ဆိုပါစို့။ ဒါပေမယ့် attacker က computer account တစ်ခုကို compromise လုပ်နိုင်ခဲ့တယ်ဆိုရင် အခြေအနေ ပြောင်းလဲသွားပါတယ်။ Server တစ်ခုမှာ privileged process ကို exploit လုပ်တာ၊ workstation တစ်လုံးမှာ local privilege escalation လုပ်တာမျိုးနဲ့ attacker က NT AUTHORITY\SYSTEM context နဲ့ code execute လုပ်နိုင်သွားနိုင်ပါတယ်။ အဲ့ဒီ computer account က အချို့ group တွေရဲ့ member ဖြစ်ပြီး အဲ့ဒီ group တွေက user object အချို့ပေါ်မှာ permission ရထားနိုင်ပါတယ်။

ဒီ user တွေက နောက်ထပ် object တွေပေါ်မှာ permission ရထားတာကြောင့် misconfiguration တွေဟာ တစ်ခုချင်းစီ ပေါင်းပြီး နောက်ဆုံးမှာ Tier-0 group တစ်ခုထိ သွားနိုင်တဲ့ attack path တစ်ခု ဖြစ်လာပါတယ်။

> Tier-0 group ဆိုတာက Active Directory forest၊ domain controller တွေ၊ administrator account တွေအပေါ် တိုက်ရိုက် သို့မဟုတ် အလွယ်တကူ ထိန်းချုပ်နိုင်တဲ့ အမြင့်ဆုံး အဆင့်ရှိတဲ့ group တွေကို ဆိုလိုတာပါ။

ဒီလို ဆက်နွယ်မှုတွေကို attacker က attack primitives ဆိုတဲ့ နည်းလမ်းတွေကို အသုံးပြုပြီး exploit လုပ်ပါတယ်။ Attack primitive ဆိုတာက Active Directory ရဲ့ အတွင်းပိုင်း အလုပ်လုပ်ပုံကို အသုံးချပြီး misconfiguration ကနေ object takeover လုပ်နိုင်တဲ့ နည်းလမ်းတွေပါ။ ဥပမာအနေနဲ့ attacker က “alice” ရဲ့ access token ရထားတယ်ဆိုရင် Charlie နဲ့ Bob ရဲ့ account တွေကို password reset လုပ်ပြီး compromise လုပ်နိုင်ပါတယ်။ အဲ့ဒီလို လုပ်နိုင်တာက alice က Force-User-Change-Password ဆိုတဲ့ extended right ရထားလို့ပါပဲ။

အကျဉ်းချုပ်ပြောရရင် Attack Path ဆိုတာက misconfiguration တစ်ခုချင်းစီ မဟုတ်ဘဲ Active Directory ထဲမှာရှိတဲ့ permission အမှားတွေ အားလုံးကို ချိတ်ဆက်ပြီး privilege escalation လုပ်နိုင်တဲ့

လမ်းကြောင်းတစ်ခုဖြစ်ပါတယ်။ Defender တစ်ယောက်အနေနဲ့ ဒီလို path တွေကို attacker ရဲ့ မျက်လုံးနဲ့ မြင်နိုင်ဖို့ BloodHound နဲ့ Active Directory logic ကို နားလည်ထားခြင်းက အလွန်အရေးကြီးပါတယ်။

ForceUserChangePassword

ForceUserChangePassword ဆိုတာ **Active Directory ထဲမှာ အလွန်အန္တရာယ်ကြီးတဲ့ permission တစ်ခုဖြစ်ပါတယ်။** ဒီ permission ရထားတဲ့ user တစ်ယောက်ဟာ အခြား user တစ်ယောက်ရဲ့ အရင် password ကို မသိဘဲ password ကို reset လုပ်နိုင်ပါတယ်။ အဲ့ဒီအချက်က attacker အတွက် account takeover လုပ်ဖို့ အလွန်လွယ်ကူတဲ့ လမ်းကြောင်းတစ်ခု ဖြစ်လာစေပါတယ်။

ဥပမာအားဖြင့် attacker တစ်ယောက်ဟာ ForceUserChangePassword permission ကို ရထားတယ်ဆိုပါစို့။ ဒါမှမဟုတ် user account တစ်ခုအပေါ် owner ဖြစ်နေတာ၊ GenericAll လို ပိုကြီးတဲ့ permission ရထားတယ်ဆိုရင်လည်း အတူတူပါပဲ။ ဒီအခြေအနေမှာ attacker က administrator တစ်ယောက်လိုပဲ target user ရဲ့ password ကို reset လုပ်နိုင်ပါတယ်။ Password reset လုပ်ပြီးတာနဲ့ target account ကို ချက်ချင်း ထိန်းချုပ်နိုင်သွားပြီး legitimate user ကိုတောင် login မလုပ်နိုင်အောင် လုပ်နိုင်ပါတယ်။

ဒီလို misconfiguration ဟာ လက်တွေ့လုပ်ငန်းခွင်မှာ အရမ်းတွေ့ရတတ်ပါတယ်။ အထူးသဖြင့် help desk user တွေ၊ lower-level administrator တွေကို compromise လုပ်နိုင်တဲ့အခါ ပိုပြီး အန္တရာယ်ကြီးလာပါတယ်။ Help desk user တွေကို ပုံမှန်အားဖြင့် “user တွေရဲ့ password reset လုပ်ခွင့်” ပေးထားရပါတယ်။ ဒါပေမယ့် တစ်ခါတစ်ရံမှာ မလိုအပ်ဘဲ higher-tier account တွေရဲ့ password ကိုပါ reset လုပ်နိုင်တဲ့ permission ပေးထားမိတတ်ပါတယ်။ ဒီလိုဖြစ်နေခဲ့ရင် attacker က help desk account တစ်ခုကို compromise လုပ်တာနဲ့ privilege escalation ကို လွယ်လွယ်ကူကူ လုပ်နိုင်ပါတယ်။

တစ်ချို့ environment တွေမှာ tier separation ကို အတိုင်းအတာတစ်ခုထိ လုပ်ထားကြပေမယ့် ForceUserChangePassword permission ကို attacker က အသုံးချနိုင်သေးပါတယ်။ Attacker က ဒီ primitive ကို အသုံးပြုပြီး user account အများကြီးကို takeover လုပ်နိုင်ပါတယ်။ Account များများကို ထိန်းချုပ်နိုင်လာတာနဲ့ network ထဲမှာ lateral movement လုပ်ရတာ ပိုလွယ်လာပါတယ်။ Lateral movement ဆိုတာက computer တစ်လုံးကနေ နောက်တစ်လုံးကို ခွေ့ပြီး access ကို တဖြည်းဖြည်း ကျယ်ပြန့်လာစေတဲ့ လုပ်ရပ်ပါ။

Account takeover လုပ်နိုင်ပြီဆိုရင် attacker က LSASS process ထဲက credential တွေကို dump လုပ်နိုင်ပါတယ်။

> LSASS(Local Security Authority Subsystem Service) ဆိုတာ Windows authentication နဲ့ credential ကို handle လုပ်တဲ့ process ဖြစ်ပြီး၊ privileged access ရလာတဲ့အခါ အလွန်တန်ဖိုးရှိတဲ့ credential တွေကို ရယူနိုင်ပါတယ်။

ဒီအခြေအနေကနေ ကြည့်ရင် tier separation ကို အပြည့်အဝ မထိန်းထားတဲ့ user တစ်ယောက်ကို attacker က ထိန်းချုပ်နိုင်သွားခဲ့ရင် domain တစ်ခုလုံးအတွက် အန္တရာယ်ကြီးမားလာပါတယ်။

Detection ဘက်ကို ပြန်ကြည့်ရင် user တစ်ယောက်ရဲ့ password ကို reset လုပ်တဲ့အခါ Windows မှာ log event နှစ်ခု ထွက်လာပါတယ်။ ပထမတစ်ခုက **Event ID 4724** ဖြစ်ပြီး `“An attempt was made to reset an account’s password”` ဆိုတဲ့ event ပါ။ ဒါက administrator တွေအတွက်လည်း

အကျွမ်းတဝင်ရှိတဲ့ event ဖြစ်ပါတယ်။ SIEM ထဲမှာ ဒီ event ကို ကြည့်လိုက်ရင် ဘယ် user က ဘယ် user ရဲ့ password ကို reset လုပ်ခဲ့လဲဆိုတာကို မြင်နိုင်ပါတယ်။

ဒုတိယ event က **Event ID 4738** ဖြစ်ပါတယ်။ ဒီ event က user account attribute တစ်ခုခု ပြောင်းလဲသွားတဲ့အခါ ထွက်လာတဲ့ event ဖြစ်ပါတယ်။ Password reset လုပ်တဲ့အခါ `Password Last Set` ဆိုတဲ့ attribute ပြောင်းလဲသွားတဲ့အတွက် ဒီ event လည်း ထွက်လာပါတယ်။ Log ကို ကြည့်လိုက်ရင် `Password Last Set` ရဲ့ timestamp က event timestamp နဲ့ ကိုက်ညီနေတာကို တွေ့ရပါလိမ့်မယ်။ ဒါပေမယ့် ဒီ event က attacker ကို identify လုပ်ဖို့ အထူးအသေးစိတ် information မပေးနိုင်တဲ့အတွက် detection အနေနဲ့ တစ်ခုတည်းကို အားထားလို့ မရပါဘူး။

ဒီနေရာမှာ “environment ကို ဘယ်လောက်နားလည်ထားလဲ” ဆိုတာ အလွန်အရေးကြီးလာပါတယ်။ ForceUserChangePassword attack ကို အထိရောက်ဆုံး ကာကွယ်နိုင်တဲ့နည်းက permission audit ကို ပုံမှန်လုပ်ပြီး least privilege principle ကို လိုက်နာတာပါ။ User တစ်ယောက်ကို သူ့အလုပ်အတွက် တကယ်လိုအပ်တဲ့ permission ပဲ ပေးရပါမယ်။ Help desk user တွေကို higher-tier account တွေရဲ့ password reset လုပ်ခွင့် မပေးထားသင့်ပါဘူး။

မဖြစ်မနေ ဒီ permission ကို ပေးရတဲ့ အခြေအနေတွေမှာလည်း administrative control နဲ့ tier separation ကို သေချာလုပ်ထားရပါမယ်။ Tier-0 account တွေ၊ Domain Admin တွေနဲ့ အနိမ့်အဆင့် user တွေကို တစ်လမ်းတည်း မဆက်သွယ်နိုင်အောင် ခွဲထားရပါမယ်။ ဒီလိုလုပ်ထားမှ ForceUserChangePassword လို attack primitive တစ်ခု compromise ဖြစ်ခဲ့ရင်တောင် domain တစ်ခုလုံး မပျက်စီးအောင် ထိန်းချုပ်နိုင်မှာ ဖြစ်ပါတယ်။

Test : Net User

What did the attacker set Bob’s password to?

ForceUserChangePassword ကို အသုံးပြုပြီး user တစ်ယောက်ရဲ့ password ကို reset လုပ်တဲ့အခါ **Event ID 4724** ထွက်လာတယ်ဆိုတာကို ပြောခဲ့ပါတယ်။ ဒါပေမယ့် 4724 event တစ်ခုတည်းကို ကြည့်လိုက်ရုံနဲ့ “ဒီ password reset ကို ဘယ်လိုနည်းလမ်းနဲ့ လုပ်ခဲ့တာလဲ” ဆိုတာကို မသေချာနိုင်ပါဘူး။ အထူးသဖြင့် net user command လို command-line နည်းလမ်းကို အသုံးပြုထားလား၊ GUI ကနေလုပ်ထားလားဆိုတာကို ခွဲခြားဖို့ context ကို ထပ်မံစစ်ဆေးဖို့ လိုအပ်ပါတယ်။

```
external_table('Winlog')
| where EventCode == 4724
| where TargetUserName contains "bob"
| project Timestamp, Message
```

ဒီ context ကို သိနိုင်တဲ့နည်းလမ်းက domain controller ပေါ်မှာ password reset ဖြစ်မတိုင်ခင် အချိန်အနည်းငယ်အတွင်း ဘာ activity တွေ ဖြစ်ခဲ့လဲဆိုတာကို ပြန်ကြည့်ခြင်းပါပဲ။ အဲ့ဒါကြောင့် ဒီနေရာမှာ timeline ကို ချဲ့ထွင်ပြီး 4724 event မဖြစ်ခင် အချိန်အနည်းငယ်ကို ပြန်လည် စစ်ဆေးပါတယ်။ ဥပမာအနေနဲ့ password reset ဖြစ်တဲ့ timestamp ကို ၁ စက္ကန့် နောက်ပြန်ဆုတ်ပြီး အဲ့ဒီအချိန်အတွင်း ဖြစ်ပေါ် ခဲ့တဲ့ event တွေကို ကြည့်တာပါ။

```
"Timestamp": "2023-08-29T12:06:18.047Z",
"Message": "An attempt was made to reset an account's password.
```

ဒီမှာ Sysmon Event ID 1 ကို ထည့်သွင်းစစ်ဆေးထားပါတယ်။ Sysmon EID 1 ဆိုတာ process creation event ဖြစ်ပြီး system ပေါ်မှာ process အသစ်တစ်ခု စတင် run လိုက်တဲ့အခါ ထွက်လာတဲ့ log ဖြစ်ပါတယ်။ Attacker က net user command ကို အသုံးပြုပြီး password reset လုပ်ခဲ့တယ်ဆိုရင် domain controller ပေါ်မှာ cmd.exe သို့မဟုတ် powershell.exe ကနေ net.exe process run ဖြစ်ခဲ့တာကို Sysmon log မှာ ဖမ်းမိနိုင်ပါတယ်။

```
external_table('Winlog')
| where Timestamp between (datetime("2023-08-29T12:06:17.047Z") ..
datetime("2023-08-29T12:06:18.064Z")) and EventCode == 1
| project Timestamp, EventCode, User, ParentCommandLine, CommandLine
```

The answer : newpassword

```
"CommandLine": "C:\\Windows\\system32\\net.exe user bob newpassword /dom"
```

Setting SPNs

အရင်အပိုင်းတွေမှာ attacker တစ်ယောက်ဟာ ForceUserChangePassword ကို အသုံးပြုပြီး user account တစ်ခုကို တိုက်ရိုက် takeover လုပ်နိုင်တယ်ဆိုတာကို ကြည့်ခဲ့ပါပြီ။ ဒါပေမယ့် attacker တွေဟာ အမြဲတမ်း password reset လုပ်ချင်ကြတာ မဟုတ်ပါဘူး။ Password ပြောင်းလိုက်ရင် legitimate user က သတိထားမိနိုင်သလို SOC က threshold-based detection တွေနဲ့ ဖမ်းမိနိုင်ခြေ ပိုများပါတယ်။ အဲ့ဒီလို risk တွေကို ရှောင်ရှားဖို့ attacker တွေက ပိုပြီး stealth ဖြစ်တဲ့ နည်းလမ်းတွေကို အသုံးပြုတတ်ကြပါတယ်။ SPN ကို set လုပ်တာဟာ အဲ့ဒီလို နည်းလမ်းတွေထဲက တစ်ခုဖြစ်ပါတယ်။

> Service Principal Name သို့မဟုတ် SPN ဆိုတာ Kerberos authentication မှာ service တစ်ခုကို ကိုယ်စားပြုတဲ့ identifier တစ်ခုပါ။

ပုံမှန်အားဖြင့် SPN တွေကို service account တွေမှာ သတ်မှတ်ထားပြီး client က service ကို access လုပ်ချင်တဲ့အခါ Kerberos ticket ကို တောင်းဖို့ အသုံးပြုပါတယ်။ ဒါပေမယ့် Active Directory မှာ user object တစ်ခုရဲ့ ServicePrincipalName attribute ကို write လုပ်ခွင့် ရထားရင် attacker က service မရှိတဲ့ user account တစ်ခုကို service account လို သဘောထားပြီး SPN တစ်ခု ထည့်နိုင်ပါတယ်။

Kerberoasting

Attacker တစ်ယောက်မှာ target user အပေါ် GenericAll ဒါမှမဟုတ် ServicePrincipalName attribute ကို write လုပ်နိုင်တဲ့ permission ရှိတယ်ဆိုပါစို့။ ဒီအခြေအနေမှာ password ကို မပြောင်းဘဲ target user အတွက် SPN တစ်ခုကို set လုပ်နိုင်ပါတယ်။ SPN ကို set လုပ်ပြီးတာနဲ့ attacker က Kerberos ကို အသုံးပြုပြီး အဲ့ဒီ SPN အတွက် service ticket ကို တောင်းနိုင်ပါတယ်။ ဒီလုပ်ရပ်ကို **Kerberoasting** လို့

ခေါ်ပါတယ်။ Kerberos service ticket ထဲမှာ user ရဲ့ password hash ကို အခြေခံထားတဲ့ encrypted data ပါဝင်တဲ့အတွက် attacker က ဒီ ticket ကို offline မှာ crack လုပ်ပြီး password ကို ရယူနိုင်ပါတယ်။

Cracking

ဒီနည်းလမ်းရဲ့ အားသာချက်က password reset လို တိုက်ရိုက်ထင်ရှားတဲ့ event မဖြစ်တာပါ။ Kerberos ticket request တစ်ကြိမ်ပဲ ဖြစ်တဲ့အတွက် threshold-based detection တွေကို ရှောင်နိုင်ပါတယ်။ ထို့အပြင် attacker က RC4 encryption မဟုတ်ဘဲ AES encryption နဲ့ ticket ကို တောင်းနိုင်ပါတယ်။ AES ticket ကို crack လုပ်ရတာ ပိုခက်ပေမယ့် SOC အများစုက RC4 downgrade detection ကို ပို အလေးထားပြီး single AES ticket request ကို အရေးမကြီးဘူးလို့ ထင်တတ်ကြပါတယ်။ အဲ့ဒီအချက်ကို attacker တွေက အကျိုးရှိအောင် အသုံးချပါတယ်။

Detection

user account တစ်ခုမှာ SPN attribute ကို ထည့်လိုက်တဲ့အခါ domain controller မှာ **Event ID 4662** ထွက်လာပါတယ်။

> ဒီ event က object တစ်ခုရဲ့ attribute ကို access သို့မဟုတ် modify လုပ်ခဲ့တယ်ဆိုတာကို ပြတဲ့ event ဖြစ်ပါတယ်။

SPN ကို set လုပ်တဲ့အခါ 4662 event ထဲမှာ ServicePrincipalName attribute ကို ကိုယ်စားပြုတဲ့ GUID ကို တွေ့နိုင်ပါတယ်။ ဒီ event က “SPN attribute ကို ပြောင်းခဲ့တယ်” ဆိုတဲ့ အချက်ကို သိနိုင်စေပေမယ့် SPN တန်ဖိုးကို တိတိကျကျ မပြသပါဘူး။

ဒီနေရာမှာ ပိုပြီး အသုံးဝင်တာက **Event ID 5136** ဖြစ်ပါတယ်။

> 5136 event က directory object တစ်ခုရဲ့ attribute တန်ဖိုး ပြောင်းလဲသွားတဲ့အခါ ထွက်လာတဲ့ event ဖြစ်ပြီး၊ **target user က ဘယ်သူလဲ၊ ဘယ် SPN value ကို ထည့်လိုက်လဲ** ဆိုတာကို တိတိကျကျ ပြပေးပါတယ်။

Incident response အတွက် ဒီ event က အလွန်အရေးကြီးပါတယ်။ ဘာကြောင့်လဲဆိုတော့ ပုံမှန် user account တွေမှာ SPN အသစ် ထည့်တာဟာ အလွန်ရှားပါးတဲ့ လုပ်ရပ်ဖြစ်လို့ပါ။

SPN ကို set လုပ်ပြီးနောက် attacker က Kerberoasting လုပ်တဲ့အခါ **Event ID 4769** ထွက်လာပါတယ်။

> ဒီ event က Kerberos service ticket တစ်ခု ထုတ်ပေးလိုက်တယ်ဆိုတာကို ပြတဲ့ event ဖြစ်ပြီး ServiceName field ထဲမှာ target user name ကို တွေ့ရနိုင်ပါတယ်။

Ticket encryption type က RC4 ဖြစ်ဖြစ် AES ဖြစ်ဖြစ် အဓိက မဟုတ်ပါဘူး။ အဓိကအချက်က user account တစ်ခုအတွက် SPN ကို မမှန်ကန်စွာ set လုပ်ထားတဲ့ လုပ်ရပ်ကို ဖမ်းမိဖို့ပါပဲ။ 4769 event ကတော့ “ဒီ SPN ကို အသုံးချပြီး attack primitive တစ်ခု ဖြစ်နေပြီ” ဆိုတဲ့ context ကို ထပ်ပေးတဲ့ အချက်အလက်ဖြစ်ပါတယ်။

အကျဉ်းချုပ်ပြောရရင် Setting SPNs ဆိုတာ attacker တွေအတွက် အလွန် stealth ဖြစ်တဲ့ privilege abuse နည်းလမ်းတစ်ခုဖြစ်ပါတယ်။ Password ကို မပြောင်းဘဲ Kerberoasting လုပ်နိုင်တဲ့အတွက် detection ကို ရှောင်နိုင်ပြီး offline cracking နဲ့ credential ရယူနိုင်ပါတယ်။ Defender အနေနဲ့ user

account တွေမှာ SPN အသစ်ဖန်တီးတာကို အလွန်သတိထားကြည့်ရှုရပါမယ်။ Event ID 5136 မှာ SPN creation တွေကို စနစ်တကျ audit လုပ်နိုင်ရင် Kerberoasting attack ကို စောစောဖမ်းမိနိုင်ပါလိမ့်မယ်။

Test : Kerberoast

Which additional user account did the attacker kerberoast?

Kerberoasting ဖြစ်နေတယ်ဆိုတာကို log ထဲကနေ သိနိုင်ဖို့ Kerberos service ticket request event ကို စောင့်ကြည့်ရပါတယ်။ Windows မှာ Kerberos service ticket တစ်ခု ထုတ်ပေးလိုက်တိုင်း **Event ID 4769** ဆိုတဲ့ log event ထွက်လာပါတယ်။ ဒါကြောင့် အခြေခံအနေနဲ့ 4769 event တွေကို စစ်ဆေးရပါမယ်။

```
external_table('Winlog')
| where EventCode == 4769
| project Timestamp, EventCode, TargetUserName, TargetDomainName,
ServiceName
```

ပေးထားတဲ့ query က Winlog table ထဲက EventCode 4769 ဖြစ်တဲ့ event တွေကို ရွေးထုတ်ထားပါတယ်။ ဒီ query ရဲ့ အဓိကရည်ရွယ်ချက်က “alice" ဆိုတဲ့ user က service ticket request လုပ်ထားတဲ့ activity တွေကိုသာ ဖော်ထုတ်ဖို့ပါ။ TargetUserName contains "alice" လို့ သုံးထားတာက alice account ကို အသုံးပြုပြီး Kerberos service ticket တောင်းထားတာရှိမရှိကို စစ်ချင်လို့ ဖြစ်ပါတယ်။ ဆိုလိုတာက alice account ကို attacker က compromise လုပ်ပြီး Kerberoasting လုပ်နေတယ်လို့ သံသယရှိတဲ့ context ပါ။

```
external_table('Winlog')
| where EventCode == 4769 and TargetUserName contains "alice" and
ServiceName !endswith "$" and ServiceName != "krbtgt"
| project Timestamp, EventCode, TargetUserName, TargetDomainName,
ServiceName
```

ဒီ query မှာ ServiceName နဲ့ ပတ်သက်ပြီး filter နှစ်ခု ထပ်ထည့်ထားတာကို တွေ့ရပါမယ်။ ServiceName !endswith "$" ဆိုတာက computer account တွေအတွက် ထုတ်တဲ့ service ticket တွေကို ဖယ်ထုတ်ဖို့ပါ။ Active Directory မှာ computer account တွေရဲ့ နာမည်ဟာ ပုံမှန်အားဖြင့် $ နဲ့ဆုံးပါတယ်။ Computer account တွေအတွက် service ticket တောင်းတာက ပုံမှန် activity ဖြစ်တတ်တဲ့အတွက် Kerberoasting detection မှာ noise ဖြစ်နိုင်ပါတယ်။ ဒါကြောင့် user သို့မဟုတ် service account ကိုပဲ အာရုံစိုက်ဖို့ $ နဲ့ဆုံးတဲ့ ServiceName တွေကို ဖယ်ထားတာပါ။

ServiceName != "krbtgt" လို့ ထပ်ပြီး filter လုပ်ထားတာက krbtgt account ကို ဖယ်ထုတ်ဖို့ ဖြစ်ပါတယ်။ krbtgt ဆိုတာ Kerberos authentication ရဲ့ အခြေခံ service account ဖြစ်ပြီး domain ထဲမှာ အမြဲတမ်း ticket issue လုပ်နေရတဲ့ account ဖြစ်ပါတယ်။ krbtgt ကို Kerberoasting target အနေနဲ့ မသုံးကြသလို log ထဲမှာ အမြဲတွေ့ရတဲ့ account ဖြစ်တဲ့အတွက် detection အတွက် အရေးမကြီးပါဘူး။ ဒါကြောင့် false positive မဖြစ်အောင် krbtgt ကို ဖယ်ထုတ်ထားတာပါ။

The result :

```
2023-08-29 13:44:48.335 | 4769 |alice@WINDOMAIN.LOCAL | WINDOMAIN.LOCAL
ADRIAN_HOFFMAN
```

Disable Pre-Auth (AS-REP Roasting)

Disable Pre-Auth ဆိုတဲ့နည်းလမ်းဟာ အရင်က ပြောခဲ့တဲ့ SPN ကို set လုပ်ပြီး Kerberoasting လုပ်တာနဲ့ သဘောတရားတူပါတယ်။ ရည်ရွယ်ချက်က password ကို မပြောင်းဘဲ user ရဲ့ password hash ကို ရယူပြီး offline မှာ crack လုပ်ဖို့ပါပဲ။ ဒီနည်းလမ်းကို **AS-REP Roasting** လို့ ခေါ်ပါတယ်။ Attacker က user account တစ်ခုရဲ့ attribute တစ်ခုတည်းကို ပြောင်းလိုက်ရုံနဲ့ ဒီတိုက်ခိုက်မှုကို လုပ်နိုင်တာကြောင့် အလွန်အန္တရာယ်ကြီးပါတယ်။

Kerberos Ticket

Kerberos authentication မှာ ပုံမှန်အားဖြင့် pre-authentication ဆိုတဲ့ လုပ်ငန်းစဉ်တစ်ခု ရှိပါတယ်။ Pre-authentication enabled ဖြစ်နေတဲ့အခါ user က login လုပ်ဖို့ ကြိုးစားရင် domain controller က **“မင်းဟာ ဒီ user တကယ်ဟုတ်ကြောင်း သက်သေပြပါ”** ဆိုပြီး timestamp တစ်ခုကို user ရဲ့ password ကို အခြေခံပြီး encrypt လုပ်ပေးဖို့ တောင်းပါတယ်။ Domain controller က အဲ့ဒီ encrypted timestamp ကို စစ်ပြီးမှ authentication ကို ဆက်လုပ်ပါတယ်။ ဒီအဆင့်ကြောင့် attacker က password ကို မသိဘဲ Kerberos ticket ကို မရနိုင်ပါဘူး။

ဒါပေမယ့် attacker က user account တစ်ခုမှာ Kerberos pre-authentication ကို disable လုပ်နိုင်သွားရင် အခြေအနေ လုံးဝ ပြောင်းလဲသွားပါတယ်။ Pre-authentication မလိုအပ်တော့တဲ့အခါ domain controller က user ကို စစ်ဆေးမနေတော့ဘဲ အဲ့ဒီ user အတွက် **encrypted Ticket Granting Ticket (TGT)** ကို တန်းပြန်ပေးလိုက်ပါတယ်။ ဒီ TGT ထဲမှာ user ရဲ့ password ကို အခြေခံထားတဲ့ encrypted data ပါဝင်နေတဲ့အတွက် attacker က ဒီ TGT ကို offline မှာ crack လုပ်ပြီး password ကို ရယူနိုင်ပါတယ်။ ဒီလုပ်ရပ်ကိုပဲ AS-REP Roasting လို့ ခေါ်ပါတယ်။

Detection

လက်ရှိ environment အများစုမှာ Kerberos pre-authentication ဟာ default အနေနဲ့ enabled ဖြစ်နေပါတယ်။ ဒါကြောင့် user account တစ်ခုမှာ pre-authentication ကို disable လုပ်ဖို့ ကြိုးစားတဲ့ action ဟာ ပုံမှန်လုပ်ငန်းဆောင်တာ မဟုတ်ဘဲ အလွန်သံသယဖြစ်စရာကောင်းတဲ့ activity တစ်ခုအဖြစ် သတ်မှတ်လို့ရပါတယ်။ Defender ဘက်ကနေ ကြည့်ရင် ဒီလို action တွေကို ချက်ချင်း သတိထားစောင့်ကြည့်သင့်ပါတယ်။

Detection အပိုင်းကို ကြည့်ရင် pre-authentication ကို disable လုပ်တဲ့အခါ domain controller မှာ log event နှစ်ခု ထွက်လာပါတယ်။ ပထမတစ်ခုက **Event ID 4662** ဖြစ်ပါတယ်။

> ဒီ event က directory object တစ်ခုရဲ့ attribute ကို access သို့မဟုတ် modify လုပ်ခဲ့တယ်ဆိုတာကို ပြတဲ့ general object access event ဖြစ်ပါတယ်။

ဒီအဆင့်မှာ attacker က user object ရဲ့ attribute ကို ပြောင်းလိုက်ပြီဆိုတာကို သိနိုင်ပေမယ့် ဘာကို ပြောင်းလိုက်လဲဆိုတာကိုတော့ တိတိကျကျ မသိရသေးပါဘူး။

ဒုတိယ event က **Event ID 4738** ဖြစ်ပါတယ်။

> ဒီ event က user account attribute ပြောင်းလဲမှုကို အသေးစိတ် ပြပေးတဲ့ event ဖြစ်ပြီး ဒီအခါမှာ **“Don’t Require Preauth – Enabled”** ဆိုတဲ့ အချက်ကို တိတိကျကျ မြင်နိုင်ပါတယ်။

ဒီစာသားကို မြင်လိုက်တာနဲ့ attacker က Kerberos pre-authentication ကို ပိတ်လိုက်တယ်ဆိုတာကို ချက်ချင်း သဘောပေါက်နိုင်ပါတယ်။ Defender အတွက် ဒီ event ဟာ အလွန်တန်ဖိုးရှိတဲ့ indicator ဖြစ်ပါတယ်။

Pre-authentication ကို disable လုပ်ပြီးပြီးချင်း attacker က AS-REP Roasting ကို ဆက်လုပ်လေ့ရှိပါတယ်။ ဒီအဆင့်မှာ domain controller က user အတွက် TGT ကို ထုတ်ပေးတဲ့ event ဖြစ်ပြီး **Event ID 4768** ထွက်လာပါတယ်။ ဒီ event ထဲမှာ

> PreAuthType ဆိုတဲ့ field ရှိပြီး အဲ့ဒီတန်ဖိုးက 0 ဖြစ်နေရင် pre-authentication မပါဘဲ TGT ကို issue လုပ်လိုက်တယ်ဆိုတဲ့ အဓိပ္ပါယ်ပါ။

timeline မှာ 4768 နဲ့ PreAuthType 0 ကို တွေ့ရရင် AS-REP Roasting ကို တကယ်လုပ်ပြီးသွားပြီလို့ သတ်မှတ်နိုင်ပါတယ်။

ဒီလို timeline ကို ချိတ်ဆက်ကြည့်မယ်ဆိုရင် attack flow ကို ပြတ်ပြတ်သားသား မြင်နိုင်ပါတယ်။ ပထမဆုံး user attribute ကို ပြောင်းတယ်ဆိုတဲ့ 4662 ထွက်လာတယ်၊ နောက် user account configuration ပြောင်းသွားတယ်ဆိုတဲ့ 4738 မှာ Don’t Require Preauth Enabled ကို မြင်ရတယ်၊ အဲ့ဒီနောက် pre-authentication မပါတဲ့ TGT ကို issue လုပ်တယ်ဆိုတဲ့ 4768 PreAuthType 0 ထွက်လာတယ်။ ဒီသုံးခုကို ချိတ်ဆက်ကြည့်နိုင်ရင် AS-REP Roasting attack ကို ယုံကြည်စိတ်ချစွာ သတ်မှတ်နိုင်ပါတယ်။

Test : More Disabling Pre-Auth

Which additional user account did the attacker disable pre-authentication for?

Event ID 4738 ဟာ user account ရဲ့ attribute တစ်ခုခုကို ပြောင်းလိုက်တဲ့အခါ domain controller မှာ အမြဲတမ်း ထွက်လာတဲ့ event ဖြစ်ပါတယ်။ Disable Pre-Auth attack မှာ attacker က “Don’t Require Preauth” ဆိုတဲ့ setting ကို Enabled လုပ်လိုက်တာကြောင့် ဒီ event ထဲမှာ အဲ့ဒီစာသားကို တိတိကျကျ တွေ့နိုင်ပါတယ်။ ဒါကြောင့် Defender အနေနဲ့ AS-REP Roasting ကို စတင်ဖို့ ပြင်ဆင်နေတဲ့ လက္ခဏာကို ရှာချင်ရင် 4738 event တွေထဲမှာ “preauth” ဆိုတဲ့ စကားလုံးပါဝင်နေတာကို ရှာဖို့ပဲ လိုပါတယ်။

```
external_table('Winlog')
| where EventCode == "4738" and Message contains "preauth"
| project Timestamp, TargetUserName, EventCode, EventAction, Message
```

Shadow Credentials

အရင်က ပြောခဲ့တဲ့ Disable Pre-Auth attack က user ရဲ့ password ကို အခြေခံတဲ့ Kerberos authentication ပေါ်မှာ မူတည်ပါတယ်။ ဒါပေမယ့် တချို့ organization တွေမှာ smartcard သုံးတယ်၊ PKI (Public Key Infrastructure) သုံးတယ်ဆိုရင် user က password ကို မရိုက်တော့ပါဘူး။ Smartcard သို့မဟုတ် certificate ကို သုံးပြီး login လုပ်ပါတယ်။ ဒီလိုအခြေအနေမှာ “password မသုံးဘူးဆိုရင် Kerberos pre-authentication ကို ဘယ်လိုလုပ်လဲ” ဆိုတဲ့ မေးခွန်း ပေါ်လာပါတယ်။

ဒီအခါမှာ Active Directory က user object ထဲမှာ **msDS-KeyCredentialLink** ဆိုတဲ့ attribute ကို အသုံးပြုပါတယ်။ ဒီ attribute ထဲမှာ user ရဲ့ **public key** ကို သိမ်းထားပါတယ်။ Authentication လုပ်တဲ့ အချိန်မှာ client ဘက်မှာရှိတဲ့ smartcard (သို့) certificate ရဲ့ **private key** နဲ့ timestamp ကို encrypt လုပ်ပြီး domain controller ဆီကို ပို့ပါတယ်။ Domain controller က user object ထဲမှာ သိမ်းထားတဲ့ public key နဲ့ decrypt လုပ်ပြီး မှန်ကန်ကြောင်း စစ်ဆေးပါတယ်။ ဒီလိုနည်းလမ်းကို Microsoft က **key trust model** လို့ ခေါ်ပါတယ်။ ယုံကြည်မှုဟာ password မဟုတ်တော့ဘဲ msDS-KeyCredentialLink ထဲမှာရှိတဲ့ public key ကို အခြေခံထားတာပါ။

ပြဿနာကတော့ attacker က ဒီ msDS-KeyCredentialLink attribute ကို ပြောင်းလဲနိုင်တဲ့ permission ရထားရင် ဖြစ်ပါတယ်။ ဥပမာ GenericWrite သို့မဟုတ် GenericAll လို permission တွေ ရထားရင် attacker က ကိုယ်ပိုင် public key တစ်ခုကို target user ရဲ့ msDS-KeyCredentialLink ထဲကို ထည့်နိုင်ပါတယ်။ အဲဒီအချိန်ကစပြီး domain controller အနေနဲ့ “ဒီ public key ကို ယုံကြည်ရမယ်” လို့ ယူဆသွားပါတယ်။ အဲ့ဒီနောက် attacker က ကိုယ်ပိုင် private key ကို သုံးပြီး target user အဖြစ် Kerberos authentication လုပ်နိုင်သွားပါတယ်။ ဒီနည်းလမ်းကို **Shadow Credentials attack** လို့ ခေါ်ပါတယ်။

ဒီ attack ရဲ့ အန္တရာယ်က password ကို မပြောင်းဘူး၊ password hash ကို ချက်ချင်း crack လုပ်စရာလည်း မလိုဘူးဆိုတဲ့ အချက်ပါ။ Attacker က msDS-KeyCredentialLink ကို တစ်ကြိမ်ပဲ ပြောင်းလိုက်ရုံနဲ့ target account ရဲ့ TGT ကို ရယူနိုင်ပြီး NTLM hash ကိုပါ ရနိုင်ပါတယ်။ User ကိုယ်တိုင်က password ပြောင်းမချင်း attacker ရဲ့ access က ဆက်လက် အသက်ရှင်နေပါတယ်။

Detection

domain controller logs ကို ကြည့်ရင် attacker က msDS-KeyCredentialLink ကို ပြောင်းတဲ့အချိန်မှာ **Event ID 4662** နဲ့ **Event ID 5136** တွေ့ ထွက်လာပါတယ်။ 4662 event ထဲမှာ msDS-KeyCredentialLink attribute ကို ကိုယ်စားပြုတဲ့ GUID ကို တွေ့ရပါလိမ့်မယ်။

> ဒီ 4662 event က “object attribute တစ်ခုကို access သို့မဟုတ် modify လုပ်လိုက်တယ်” ဆိုတာကို ပြောပြပါတယ်။

> အဲ့ဒီနောက် 5136 event ထဲမှာတော့ ဘယ် object ကို ပြောင်းလိုက်တာလဲဆိုတာကို ObjectDN အနေနဲ့ တိတိကျကျ ဖော်ပြပေးပါတယ်။

အရေးကြီးတဲ့အချက်က msDS-KeyCredentialLink ဆိုတဲ့ attribute ကို **သာမန် domain user တွေက ပြောင်းဖို့ မသင့်တော်ပါဘူး**။ Helpdesk user ဖြစ်စေ၊ normal user ဖြစ်စေ ဒီ attribute ကို modify လုပ်တာတွေ့ရင် အလွန်အန္တရာယ်ရှိတဲ့ လက္ခဏာဖြစ်ပါတယ်။ ဒါကြောင့် SOC သို့မဟုတ် IR အဖွဲ့အနေနဲ့ ဒီလို event တွေကို မြင်တာနဲ့ ချက်ချင်း စုံစမ်းသင့်ပါတယ်။

နောက်ဆုံးအနေနဲ့ attacker က target account အဖြစ် TGT တောင်းတဲ့အချိန်မှာ **Event ID 4768** ထွက်လာပြီး PreAuthType က 16 ဖြစ်နေတာကို တွေ့နိုင်ပါတယ်။ ဒါဟာ smartcard authentication ကို ညွှန်းတာပါ။ သို့သော် ဒီ event တစ်ခုတည်းက malicious ဖြစ်မဖြစ် ဆုံးဖြတ်လို့ မရပါဘူး။ Environment တစ်

ခုလုံး smartcard သုံးနေတယ်ဆိုရင်တော့ သာမန်အခြေအနေ ဖြစ်နိုင်ပါတယ်။ တကယ့်အရေးကြီးတာက msDS-KeyCredentialLink ကို ဘယ်သူ ပြောင်းလဲခဲ့လဲ ဆိုတဲ့ အချက်ပါ။

Script Path

Active Directory ထဲမှာ user တစ်ယောက် login ဝင်တဲ့အချိန်မှာ အလိုအလျောက် run ပေးနိုင်တဲ့ **logon script** ဆိုတာ ရှိပါတယ်။ ဒီ script ကို အသုံးပြုပြီး drive mapping လုပ်တာ၊ environment variable set လုပ်တာ၊ သို့မဟုတ် company policy အရ command တွေ run ပေးတာမျိုး လုပ်ကြပါတယ်။ ဒီ logon script ရဲ့ path ကို AD user object ထဲက **scriptPath** ဆိုတဲ့ attribute မှာ သိမ်းထားပါတယ်။

ပြဿနာက attacker က target user ရဲ့ scriptPath attribute ကို ပြောင်းလဲနိုင်တဲ့ permission ရထားရင် ဖြစ်ပါတယ်။ ဥပမာ GenericWrite, GenericAll လို permission တွေ ရထားရင် attacker က scriptPath ကို malicious script တစ်ခုရှိတဲ့ network share သို့မဟုတ် local path တစ်ခုကို ပြောင်းနိုင်ပါတယ်။ အဲ့ဒီနောက် target user က login ဝင်တာနဲ့ attacker ထည့်ထားတဲ့ script က **user ရဲ့ context နဲ့ အလိုအလျောက် run သွားပါတယ်**။ Target user က privileged user ဖြစ်ရင် attacker အတွက် privilege escalation သို့မဟုတ် lateral movement လုပ်ဖို့ အလွန်လွယ်သွားပါတယ်။

> ဒီ attack ရဲ့ အန္တရာယ်က password ပြောင်းစရာမလို၊ exploit run စရာမလို၊ user ကို login ဝင်အောင်ပဲ စောင့်ရတာပါ။ User ကိုယ်တိုင် login ဝင်တဲ့အချိန်မှာ attacker ရဲ့ code ကို run ပေးသွားတာကြောင့် detection မလုပ်နိုင်ရင် တိတ်တိတ်လေး compromise ဖြစ်သွားနိုင်ပါတယ်။

Logs အနေနဲ့ ကြည့်မယ်ဆိုရင် ဒီ attack မှာ အရင်တွေ့ခဲ့ဖူးတဲ့ pattern ကို ထပ်မံတွေ့ရပါတယ်။ အရင်ဆုံး **Event ID 4662** ထွက်လာပါတယ်။ ဒီ event က AD object attribute ကို access သို့မဟုတ် modify လုပ်လိုက်တယ်ဆိုတာကို ပြပါတယ်။ ဒီ event ထဲမှာ scriptPath attribute ကို ကိုယ်စားပြုတဲ့ GUID `bf9679a8-0de6-11d0-a285-00aa003049e2` ကို တွေ့နိုင်ပါတယ်။ ဒီ GUID ကို တွေ့ရင် “scriptPath ကို ပြောင်းလိုက်ပြီ” ဆိုတာကို သိနိုင်ပါတယ်။

အဲ့ဒီနောက် **Event ID 4738** ထွက်လာပါတယ်။ ဒီ event က user account ရဲ့ attribute တစ်ခုခု ပြောင်းလဲလိုက်တယ်ဆိုတာကို ပိုပြီး လူဖတ်လွယ်တဲ့ပုံစံနဲ့ ပြပေးပါတယ်။ ဒီ event ထဲမှာ **target user account ရဲ့ နာမည်နဲ့ scriptPath အသစ် ဘာကို ပြောင်းလိုက်တယ်ဆိုတာကို တိတိကျကျ တွေ့နိုင်ပါတယ်။**
Investigation လုပ်တဲ့အချိန်မှာ ဒီ scriptPath value ကို ကြည့်ရင် malicious share သို့မဟုတ် မသင့်တော်တဲ့ path ဟုတ်မဟုတ်ကို ချက်ချင်း ခန့်မှန်းနိုင်ပါတယ်။

တချို့ environment တွေမှာတော့ **Event ID 5136** လည်း ဆက်ပြီး ထွက်လာနိုင်ပါတယ်။ ဒီ event က directory object modification ကို ပိုပြီး အသေးစိတ် ပြပေးပါတယ်။ **ObjectDN နဲ့ attribute change details တွေ ပါဝင်တာကြောင့် forensics လုပ်တဲ့အခါ အထောက်အကူ ဖြစ်ပါတယ်။**

အရေးကြီးတဲ့အချက်က scriptPath attribute ဟာ ပုံမှန် environment တစ်ခုမှာ **အရမ်းမပြောင်းလဲသင့်တဲ့ value** ဖြစ်ပါတယ်။ User တစ်ယောက်အတွက် logon script ကို တစ်ခါ set လိုက်ပြီးရင် အချိန်ကြာကြာ မပြောင်းဘဲ သုံးနေကြတာ များပါတယ်။ ဒါကြောင့် scriptPath ကို ပြောင်းလိုက်တယ်ဆိုတဲ့ event တွေကို တွေ့ရင် “သာမန် administrative task” လို့ မထင်ဘဲ **အမြဲတမ်း investigation လုပ်သင့်ပါတယ်**။

Test : More Script Path

Which additional user did the attacker set a scriptpath for?

Active Directory ထဲမှာ user account အများစုဟာ logon script ကို မသုံးပါဘူး။ ဒီလို account တွေအတွက် ScriptPath value က ပုံမှန်အားဖြင့် `-` သို့မဟုတ် `%%1793` လို system placeholder value ဖြစ်နေတတ်ပါတယ်။ ဒီ values တွေက “ဒီ user အတွက် logon script မသတ်မှတ်ထားပါဘူး” ဆိုတဲ့ အဓိပ္ပါယ်ပါ။ ဒါကြောင့် ScriptPath attribute ကို အမှန်တကယ် အသုံးပြုထားတဲ့ account တွေကို ရှာချင်ရင် ဒီ placeholder values တွေကို ဖယ်ရှားပစ်ရပါမယ်။

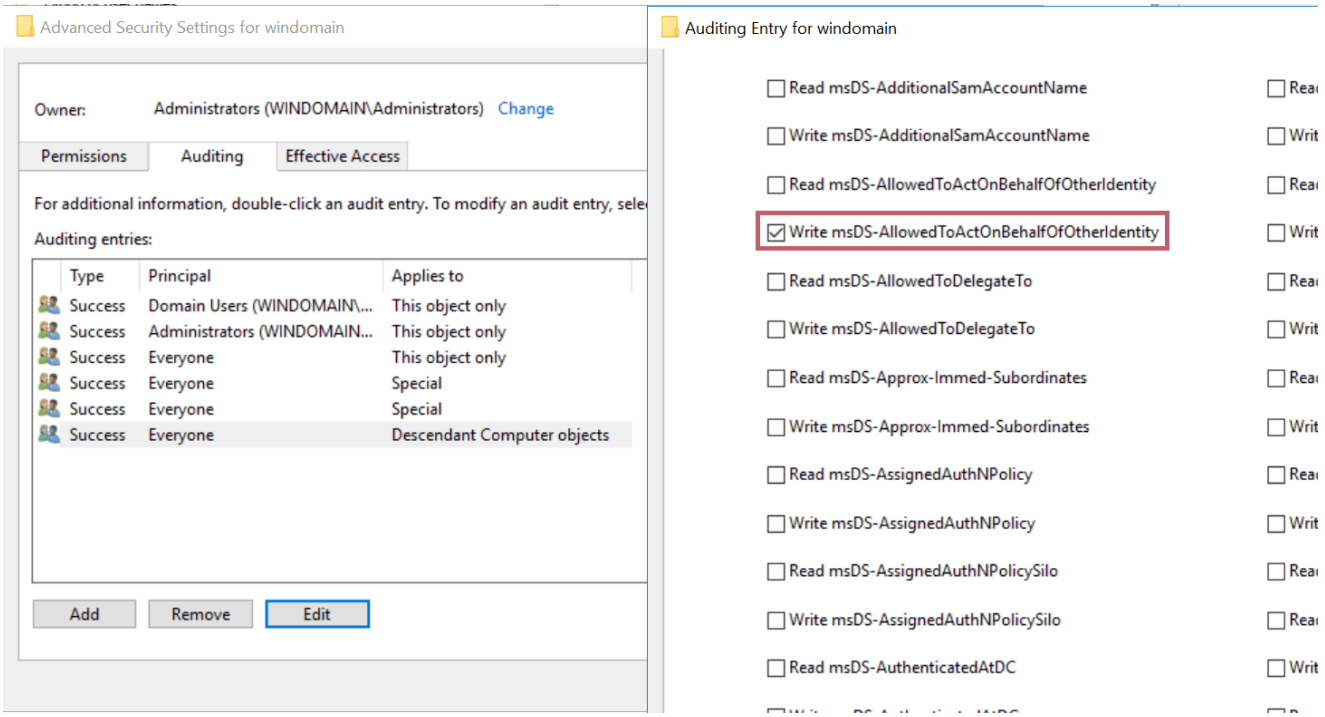
```
external_table('Winlog')
| where EventCode == "4738" and ScriptPath !in ("-", "%%1793")
| project Timestamp, EventCode, SubjectUserName, TargetUserName, Message,
ScriptPath
```

ဒီ query က အဲ့ဒီအလုပ်ကို လုပ်ပေးပါတယ်။ EventCode 4738 ဖြစ်တဲ့ event တွေထဲက ScriptPath value က `-` မဟုတ်ဘူး၊ `%%1793` မဟုတ်ဘူး ဆိုတဲ့ event တွေကိုပဲ ရွေးထုတ်ပါတယ်။ အဲ့ဒီလိုလုပ်လိုက်တဲ့အခါ **logon script ကို set လုပ်ထားတဲ့ user account တွေရဲ့ event တွေအားလုံး** ကို တစ်နေရာတည်းမှာ မြင်နိုင်ပါတယ်။ ဒါဟာ ScriptPath abuse detection အတွက် အလွန်အသုံးဝင်ပါတယ်။

RBCD Computer Object Takeover

အရင်ဆုံး **Kerberos delegation** ဆိုတဲ့ အယူအဆကို နားလည်ရပါမယ်။ ပုံမှန်အားဖြင့် user တစ်ယောက်က service တစ်ခုကို access လုပ်တဲ့အခါ Kerberos service ticket (TGS) ကို အသုံးပြုပါတယ်။ ဒီ ticket ဟာ အဲဒီ service တစ်ခုတည်းအတွက်ပဲ အသုံးပြုလို့ရပါတယ်။ ဒါပေမယ့် real-world application တွေမှာ web server တစ်ခုက database server ကို user အစား access လုပ်ရတာလိုအပ်တတ်ပါတယ်။ ဒီလိုအခါ service က user ကို “ကိုယ်စားပြု” လုပ်ပြီး တခြား resource တွေကို access လုပ်နိုင်ဖို့ **delegation** ကို အသုံးပြုပါတယ်။

Traditional delegation မှာ “ဒီ service ကို delegation လုပ်ခွင့်ပေးမလား” ဆိုတာကို service account ဘက်က သတ်မှတ်ရပါတယ်။ ဒါပေမယ့် **Resource-Based Constrained Delegation (RBCD)** မှာတော့ အယူအဆ ပြောင်းပြန်ဖြစ်သွားပါတယ်။ Resource ဖြစ်တဲ့ server (ဥပမာ database server) က “ဘယ် service ကို ငါ့ဆီမှာ user ကိုယ်စား impersonate လုပ်ခွင့်ပေးမလဲ” ဆိုတာကို ကိုယ်တိုင် သတ်မှတ်နိုင်ပါတယ်။ ဒီ trust relationship ကို computer account ရဲ့ **msDS-AllowedToActOnBehalfOfOtherIdentity** attribute ထဲမှာ သိမ်းထားပါတယ်။



ဒီနေရာမှာ attacker အတွက် အခွင့်အလမ်း ပေါ်လာပါတယ်။ Attacker က computer account တစ်ခု အပေါ် **write permission** ရထားရင် အဲဒီ attribute ကို ပြောင်းလဲနိုင်ပါတယ်။ ပြောင်းလိုက်တဲ့အခါ "ဒီ target computer က attacker ထိန်းချုပ်ထားတဲ့ service ကို ယုံကြည်ပြီး user တွေကို impersonate လုပ်ခွင့်ပေးမယ်" ဆိုတဲ့ trust ကို တိတ်တိတ်လေး ထည့်သွင်းလိုက်တာ ဖြစ်ပါတယ်။ အဲဒီနောက် attacker က Domain Admin အပါအဝင် privileged user တွေကို impersonate လုပ်ပြီး target computer ပေါ်မှာ code execution လုပ်နိုင်သွားပါတယ်။

Enviroment

ဒီ attack ကို လုပ်ဖို့ delegation အသုံးပြုမယ့် account မှာ **SPN (Service Principal Name)** လိုအပ်ပါတယ်။ Attacker မှာ SPN ပါတဲ့ service account မရှိဘူးဆိုရင်လည်း ပြဿနာမရှိပါဘူး။ Windows domain တွေမှာ default အနေနဲ့ normal domain user တစ်ယောက်က **computer account 10 ခု အထိ create လုပ်နိုင်ပါတယ်**။ Attacker က computer account အသစ်တစ်ခု create လုပ်လိုက်ရုံနဲ့ SPN ပါတဲ့ account ကို ရရှိသွားပါတယ်။ ဒီ action ကို domain controller logs မှာ **Event ID 4741** အနေနဲ့ တွေ့ရပါတယ်။

အဲ့ဒီနောက် attacker က target computer ရဲ့ msDS-AllowedToActOnBehalfOfOtherIdentity attribute ကို ပြောင်းလိုက်ပါတယ်။ ဒီ attribute ပြောင်းလဲမှုက **Event ID 4742** ကို ဖြစ်စေပါတယ်။ ဒါပေမယ့် ပြဿနာက ဒီ event က အလွန်အကျူးမရှိပါဘူး။ ဘယ် delegation ကို ထည့်လိုက်လဲ၊ ဘယ်သူကို ယုံကြည်လိုက်လဲ ဆိုတာကို မပြောပြနိုင်လောက်အောင် information နည်းပါတယ်။ ဒီကြောင့် RBCD attack ဟာ default logging နဲ့ဆိုရင် အရမ်းလွယ်လွယ်နဲ့ လွတ်သွားနိုင်ပါတယ်။

ပိုပြီး ဆိုးတာက default configuration မှာ **msDS-AllowedToActOnBehalfOfOtherIdentity attribute ကို write လုပ်တာကို domain controller က မ audit ပါဘူး**။ ဆိုလိုတာက attacker က RBCD ကို configure လုပ်လိုက်ပေမယ့် SOC အနေနဲ့ log ထဲမှာ ဘာမှ မမြင်နိုင်တာပါ။ ဒါကြောင့် Defender ဘက်က proactive ပြင်ဆင်မှု လိုအပ်လာပါတယ်။

ဖြေရှင်းနည်းက domain object ရဲ့ **SACL (System Access Control List)** ကို update လုပ်ခြင်းပါ။ Domain object မှာ ACE တစ်ခု ထည့်ပြီး **Everyone group က descendant computer object တွေရဲ့ msDS-AllowedToActOnBehalfOfOtherIdentity attribute ကို write လုပ်တဲ့အခါ audit**

လုပ်ပါ ဆိုပြီး သတ်မှတ်နိုင်ပါတယ်။ ဒီလို ပြင်ဆင်လိုက်ရင် attacker က RBCD configure လုပ်တဲ့အချိန်တိုင်း domain controller က အသုံးဝင်တဲ့ security event တစ်ခု ထုတ်ပေးပါလိမ့်မယ်။

ဒီ event ကို detection rule တစ်ခုအဖြစ် SIEM ထဲမှာ ထည့်ထားနိုင်ရင် **RBCD Computer Object Takeover ကို အလွန်အစောဆုံး အဆင့်မှာပဲ ဖမ်းမိနိုင်ပါတယ်**။ အထူးသဖြင့် AD environment က ရှုပ်ထွေးပြီး service account password တွေ အားနည်းနေတယ်ဆိုရင် ဒီ detection ဟာ Tier-0 compromise ကို ကာကွယ်နိုင်တဲ့ အရေးကြီးဆုံး control တွေထဲက တစ်ခု ဖြစ်လာပါတယ်။

Conclusion

အကျဉ်းချုပ်ပြောရရင် RBCD attack ဟာ computer account permission နည်းနည်းပဲ ရထားရင်တောင် Domain Admin impersonation အထိ ရောက်နိုင်တဲ့ အလွန်အန္တရာယ်ကြီးတဲ့ Active Directory abuse နည်းလမ်းပါ။ Default logging ကို မယုံရဘဲ SACL ကို သေချာ configure လုပ်ပြီး msDS-AllowedToActOnBehalfOfOtherIdentity write operation တွေကို audit လုပ်ထားနိုင်ရင် Defender အနေနဲ့ attacker ရဲ့ attack path ကို အစောဆုံး ဖြတ်တောက်နိုင်ပါလိမ့်မယ်။
