Lateral Movement

Objectives

One of the primary ways that attackers move laterally through a network is through remote services. This module will focus on several common remote service techniques, including:

- WMI
- DCOM
- WinRM
- scheduled tasks

What Makes a Good Target?

ဒီစာပိုဒ်ရဲ့အဓိကရည်ရွယ်ချက်က “**attackers တွေက ဘယ်လို system ကို target လုပ်မလဲ**” ဆိုတာကို defender သို့မဟုတ် forensics လုပ်နေတဲ့သူတွေ အနေနဲ့ နားလည်စေချင်တာပါ။ အရင်အချိန်တွေမှာ attacker တွေရဲ့လုပ်ပုံလုပ်နည်းက အရမ်းမစိစစ်သေးဘဲ၊ တစ်လုံးပေါက်ရင် အခြားလုံးတွေကို ဆက်တိုက် ထိုးဝင်သွားတဲ့ ပုံစံဖြစ်ပါတယ်။ အဲ့ဒီအချိန်မှာ local administrator password ကို computer အများစုမှာ တူတူထားတတ်တဲ့အတွက် workstation တစ်လုံးကို ပေါက်သွားရင် admin password ကိုရပြီး network ထဲက computer အများကြီးကို ဆက်ပြီးဝင်နိုင်ပါတယ်။

အများအားဖြင့် attacker က workstation တစ်လုံးထဲကို user အဆင့်နဲ့ပထမဆုံးဝင်ပါတယ်။ အဲ့ဒီနောက် privilege escalation လုပ်ပြီး local administrator အဖြစ်ပြောင်းတတ်ပါတယ်။ အဲ့ဒီအချိန်မှာ Windows ရဲ့ SAM database ထဲက local account password hash တွေကို dump လုပ်ပြီး ထုတ်ယူပါတယ်။ ဒီ hash ကို network ထဲက အခြား computer တွေမှာ စမ်းသုံးပြီး login ဝင်ကြည့်တာကို “spray” လုပ်တယ်လို့ခေါ်ပါတယ်။ အဲ့ဒီလိုနဲ့ system အများကြီးထဲကို ဝင်နိုင်သွားပြီး၊ နောက်ဆုံးမှာ domain admin လို လုပ်ပိုင်ခွင့်မြင့်တဲ့ user က login ဝင်ထားတဲ့ computer ကိုတွေ့သွားရင် LSASS memory ကို dump လုပ်ပြီး domain admin credential ကိုရယူနိုင်ပါတယ်။ ဒီလိုလုပ်နည်းကို ထပ်ခါထပ်ခါလုပ်ပြီး လုပ်ပိုင်ခွင့်မြင့်ဆုံးအထိ တက်သွားကြတာပါ။

ဒါပေမယ့် အခုခေတ်မှာတော့ organization အများစုက security ကို ပိုပြီးဂရုစိုက်လာကြပါတယ်။ Endpoint Detection and Response (EDR) လို tool တွေက suspicious behavior တွေကို စောင့်ကြည့်နိုင်လာပြီး Local Administrator Password Solution (LAPS) လိုနည်းလမ်းနဲ့ computer တစ်လုံးစီမှာ admin password ကို မတူအောင်ထားလာကြပါတယ်။ ဒီအခြေအနေမှာ attacker က အရင်လို system အများကြီးကို အလွယ်တကူ ဆက်တိုက်ထိုးဝင်ဖို့ မလွယ်တော့ပါဘူး။

ဒါကြောင့် attacker တွေက နည်းလမ်းပြောင်းလာကြပါတယ်။ အခုတော့ lateral movement ကို အများကြီးမလုပ်ဘဲ၊ သူတို့ရဲ့ footprint ကို သေးသေးလေးထားပြီး လိုအပ်တဲ့ system တချို့ကိုပဲ ရည်ရွယ်လာကြပါတယ်။ သူတို့က Active Directory ကို သေချာလေ့လာပြီး ဘယ် system က အရေးကြီးဆုံးလဲ၊ ဘယ် user တွေက ဘယ်နေရာမှာ login ဝင်တတ်လဲဆိုတာကို အရမ်းကောင်းကောင်း enumerate လုပ်လာကြပါတယ်။ ရည်ရွယ်ချက်က alert မတက်စေဘဲ၊ log ထဲမှာ ပုံမှန်လိုပဲမြင်ရအောင်လုပ်ပြီး objective ကို ပြီးမြောက်စေချင်တာပါ။

ဒီလိုအခြေအနေမှာ defender တွေက “logon အရေအတွက်များမှ attack” ဆိုတဲ့ threshold-based detection ကိုသာ မယုံသင့်ပါဘူး။ အရေးကြီးတာက “ဒီ login ဟာ ဒီ system မှာ ဖြစ်သင့်လား” ဆိုတဲ့ context ကို နားလည်ဖို့ပါ။ ဥပမာအားဖြင့် helpdesk user တစ်ယောက်က Domain Controller မှာ login ဝင်နေရင် log တစ်ကြောင်းတည်းပဲဖြစ်ပေမယ့် အရမ်းမမှန်တဲ့အပြုအမူဖြစ်ပါတယ်။ အဲ့ဒီလို unusual ဖြစ်တဲ့ authentication ကို သိနိုင်ဖို့ ကိုယ့် environment ကို ကိုယ်နားလည်ထားရပါမယ်။

ဒီ relationship တွေကို နားလည်ဖို့အတွက် BloodHound လို tool ကို အသုံးပြုပါတယ်။ BloodHound က Active Directory ထဲက user၊ group၊ computer တွေရဲ့ဆက်နွယ်မှုတွေကို map ဆွဲပြပေးပါတယ်။ အဲ့ဒီထဲမှာ privileged group ထဲဝင်ထားတဲ့ computer account တွေ၊ local administrators group ထဲမှာ user group ကြီးတွေ ပါဝင်နေတာတွေကို ကြည့်နိုင်ပါတယ်။ အဲ့ဒီလို system တွေက attacker အတွက် အလွန်အရေးကြီးတဲ့ target ဖြစ်ပါတယ်၊ ဘာကြောင့်လဲဆိုရင် user တစ်ယောက်ပေါက်ရုံနဲ့ admin အခွင့်အရေးရနိုင်တာကြောင့်ပါ။

ထို့အပြင် privileged user တွေ မကြာခဏ login ဝင်တတ်တဲ့ system တွေလည်း အန္တရာယ်ကြီးပါတယ်။ Domain admin၊ server admin တွေက နေ့စဉ်အသုံးပြုနေတဲ့ machine မှာ LSASS memory ထဲမှာ အရေးကြီး credential တွေရှိနေလေ့ရှိပါတယ်။ Attacker က အဲ့ဒီ system ကိုရောက်သွားနိုင်ရင် credential တစ်ခုထဲနဲ့ network အားလုံးကို ထိန်းချုပ်နိုင်သွားနိုင်ပါတယ်။

Replication လုပ်တဲ့ system တွေဖြစ်တဲ့ Domain Controller သို့မဟုတ် AD FS server တွေကိုလည်း အထူးသတိထားရပါမယ်။ Attacker က DCSync attack လုပ်ချင်ရင် replication လုပ်နေတဲ့ system ပေါ်ကနေ လုပ်ရင် ပုံမှန် activity နဲ့ ခွဲခြားဖို့ အရမ်းခက်ပါတယ်။ ထို့အပြင် network segment အကြားမှာ ရှိတဲ့ system တွေ၊ jump box တွေ၊ VDI system တွေဟာလည်း attacker အတွက် pivot လုပ်ဖို့ အလွန်ကောင်းတဲ့နေရာတွေဖြစ်ပါတယ်။

နောက်ဆုံးအနေနဲ့ technical အရေးကြီးမှုသာမက business အတွက် အရေးကြီးတဲ့ system တွေကိုပါ စောင့်ကြည့်ရပါမယ်။ Finance၊ HR၊ production system လိုနေရာတွေကို attacker က data ခိုးချင်တာ၊ service ဖျက်ချင်တာ စတဲ့ objective နဲ့ရည်ရွယ်လာနိုင်ပါတယ်။ ဒါကြောင့် defender အနေနဲ့ “ဘယ် system က အဖိုးတန်လဲ” ဆိုတာကို technical နဲ့ business နှစ်ဖက်လုံးအနေနဲ့ နားလည်ထားဖို့လိုပါတယ်။

အကြောင်းအရာအားလုံးကို စုပေါင်းပြောရရင်၊ ဒီစာပိုဒ်က “tool တစ်ခုတည်း၊ alert တစ်ခုတည်းနဲ့ attack ကိုမဖမ်းနိုင်ဘူး” ဆိုတာကို ပြောချင်တာပါ။ ကိုယ့် network ကို ကိုယ်သိရမယ်၊ high-value system တွေကို ခွဲခြားထားရမယ်၊ ပြီးတော့ attacker ရဲ့မျက်လုံးနဲ့ စဉ်းစားနိုင်ရမယ်။ အဲ့ဒီလိုလုပ်နိုင်ရင် log တစ်ကြောင်းလောက်ပဲရှိသော်လည်း “ဒီဟာ မမှန်ဘူး” ဆိုတာကို သိနိုင်ပြီး incident ကို အချိန်မီ ဖမ်းဆီးနိုင်ပါလိမ့်မယ်။

1) DCOM (Distributed COM) 🔌🪄

Main concept:

DCOM က COM ကို network ပေါ်ကနေ အသုံးပြုနိုင်အောင်လုပ်ပေးတဲ့ mechanism ဖြစ်ပြီး underlying transport အနေနဲ့ **RPC (Remote Procedure Call)** ကို သုံးပါတယ်။ Attacker အနေနဲ့ target machine ရဲ့ IP သိရုံနဲ့၊ အဲ့ဒီ COM object ကို launch လုပ်ဖို့ authorization ရှိရုံနဲ့ remote execution လုပ်နိုင်ပါတယ်။

Detail Explanation

COM ဆိုတာ Windows ထဲမှာ application တွေအချင်းချင်း ဆက်သွယ်ပြောဆိုဖို့အတွက် standard တစ်ခုပါ။ ရိုးရိုးပြောရရင် application တစ်ခုက COM object တစ်ခု register လုပ်ထားရင် အခြား process

တစ်ခုက အဲ့ဒီ COM object ကို ခေါ်သုံးပြီး method တွေကို execute လုပ်နိုင်ပါတယ်။

ဒီဟာကို legitimate use အနေနဲ့ Windows ထဲမှာ အများကြီးသုံးပါတယ်။ ဥပမာ PowerShell process ထဲကနေ Explorer ကို control လုပ်ချင်ရင် Shell.Application ဆိုတဲ့ COM object ကို instantiate လုပ်ပြီး Explorer ရဲ့ function တွေကို ခေါ်သုံးနိုင်ပါတယ်။ ဒါဆို COM ကိုယ်တိုင်က malicious မဟုတ်ဘူး၊ Windows ရဲ့ ပုံမှန် architecture တစ်စိတ်တစ်ပိုင်းပဲဖြစ်ပါတယ်။

COM က lateral movement အတွက် စိတ်ဝင်စားဖို့ကောင်းလာတာက attacker က **remote machine** ပေါ်က process တစ်ခုထဲမှာ method execute လုပ်ချင်တဲ့အချိန်ပါ။ ဒီနေရာမှာ **DCOM** ဝင်လာပါတယ်။ DCOM က COM ကို network ပေါ်ကနေ အသုံးပြုနိုင်အောင်လုပ်ပေးတဲ့ mechanism ဖြစ်ပြီး underlying transport အနေနဲ့ **RPC (Remote Procedure Call)** ကို သုံးပါတယ်။ Attacker အနေနဲ့ target machine ရဲ့ IP သိရုံနဲ့၊ အဲ့ဒီ COM object ကို launch လုပ်ဖို့ authorization ရှိရုံနဲ့ remote execution လုပ်နိုင်ပါတယ်။

လက်တွေ့မှာ lateral movement အတွက် အများဆုံး၊ အတည်ငြိမ်ဆုံး သုံးကြတဲ့ DCOM technique တစ်ခုက **MMC20.Application** COM object ကို အသုံးပြုတာပါ။ MMC ဆိုတာ Microsoft Management Console ဖြစ်ပြီး mmc.exe process ထဲမှာ run ပါတယ်။ ဒီ COM object မှာ **ExecuteShellCommand** ဆိုတဲ့ method ပါပြီး အဲ့ဒီ method ကို ခေါ်လိုက်ရင် mmc.exe က child process တစ်ခုကို launch လုပ်ပေးပါတယ်။ Attacker က ဒီ mechanism ကို အသုံးပြုပြီး remote machine ပေါ်မှာ command သို့မဟုတ် malware ကို run ခိုင်းနိုင်ပါတယ်။

Defender အနေနဲ့ log ထဲမှာ ဒီ activity ကို ဘယ်လိုမြင်ရလဲဆိုရင် Sysmon EventCode 1 (process creation) ထဲမှာ mmc.exe run တာကို တွေ့ရပါမယ်။ အထူးသဖြင့် mmc.exe ရဲ့ parent process ကို ကြည့်လိုက်ရင် `svchost.exe -k DcomLaunch -p` ဆိုပြီးပေါ်နေလေ့ရှိပါတယ်။ ဒါဟာ DCOM service က mmc.exe ကို launch လုပ်လိုက်တာဖြစ်တယ်ဆိုတဲ့ strong indicator ပါ။ ထို့အပြင် mmc.exe ရဲ့ command line ထဲမှာ `-Embedding` ဆိုတဲ့ parameter ပါနေတာကို တွေ့ရပါမယ်။ ဒီ `-Embedding` parameter က COM/DCOM ကနေ launch လုပ်ထားတဲ့ process တွေမှာ မကြာခဏတွေ့ရတဲ့ pattern ဖြစ်ပါတယ်။ mmc.exe မဟုတ်ဘဲ excel.exe၊ word.exe စတဲ့ အခြား application တွေနဲ့လည်း attacker က ဒီနည်းလမ်းကို သုံးနိုင်ပါတယ်။

ဒီ process creation မဖြစ်ခင် နည်းနည်းစောပြီး log ထဲမှာ **type 3 logon** event ကို တွေ့ရပါမယ်။ Type 3 logon ဆိုတာ network logon ဖြစ်ပြီး attacker က remote system ကို network ကနေ access လုပ်နေတယ်ဆိုတဲ့ အထောက်အထားပါ။ ထို့နောက် Sysmon EventID 3 (network connection) ထဲမှာ svchost.exe က port 135 (RPC endpoint mapper) ကို ချိတ်ဆက်တာကို တွေ့နိုင်ပါတယ်။ စိတ်ဝင်စားစရာက network connection event က mmc.exe execute ဖြစ်ပြီးမှ နည်းနည်းနောက်ကျပြီး ပေါ်လာတတ်တာပါ။ ဒါကြောင့် timeline ကို သေချာကြည့်ရင် DCOM behavior ကို ပိုမိုနားလည်နိုင်ပါတယ်။

Sample query (KQL):

Query ထဲမှာ inner join ကို သုံးပြီး HostName ကို join key အဖြစ်ထားပါတယ်။ ဒါက mmc.exe execute ဖြစ်တဲ့ host ပေါ်က event တွေကိုပဲ ဆက်စပ်ပြသချင်တာကြောင့်ပါ။ Timestamp difference ကို တစ်စက္ကန့်အတွင်းလောက် သတ်မှတ်ထားတာက DCOM activity တွေဟာ အချိန်အရမ်းကပ်ကပ် ဖြစ်ပေါ်တတ်လို့ပါ။

```
external_table('Winlog')
| where EventCode==1 and Image endswith "mmc.exe"
```

```
| project mmcTime = Timestamp, HostName, ParentCommandLine, ParentUser,
CommandLine, User
| join kind=inner
    (external_table('Winlog')
    | project Timestamp, HostName, EventCode, TargetUserName,
DestinationPort
    ) on HostName
| where (mmcTime - Timestamp) between (0s .. 1s)
| project Timestamp, EventCode, TargetUserName, mmcCommandLine = CommandLine
```

Notice

ဒါပေမယ့် DCOM technique အားလုံးက ဒီလို artifact အားလုံးကို မထုတ်ပေးပါဘူး။ ဥပမာ **ShellBrowserWindow** class က explorer.exe ရဲ့ already running process ထဲကနေ command ကို execute လုပ်ပါတယ်။ ဒီနည်းလမ်းက interactive user session တစ်ခု ရှိနေရပါမယ်၊ ထို user ရဲ့ credential ကို attacker က သိထားရပါမယ်။ ဒါကြောင့် အလွယ်တကူ မတွေ့ရဘဲ edge case လို ဖြစ်တတ်ပါတယ်။ ထို့အပြင် ဒီနည်းလမ်းနဲ့ run တဲ့ process က medium integrity ဖြစ်ပြီး၊ lateral movement နည်းလမ်းအများစုလို high integrity မဖြစ်ပါဘူး။

ဒီလို edge case တွေကို defender အနေနဲ့ handle လုပ်ချင်ရင် approach ကိုပြောင်းစဉ်းစားရပါမယ်။ Explorer.exe ရဲ့ child process တွေအတွက် LOLBin detection ရှိဖို့လိုပါတယ်။ ထို့အပြင် target user က system ထဲမှာ interactive login ဝင်ထားပြီးသား ဖြစ်နေတဲ့အချိန်မှာ အခြား host တစ်ခုကနေ type 3 logon လာတာကိုလည်း suspicious အနေနဲ့ကြည့်ရပါမယ်။ နောက်တစ်ခုက suspicious activity မဖြစ်ခင် နည်းနည်းစောပြီး administrative share တွေကို executable file သို့မဟုတ် script ရေးချင်တဲ့ ကြိုးပမ်းမှုတွေရှိလားဆိုတာ Windows Security EventID 5145 ကို ကြည့်ပြီး စစ်ဆေးရပါမယ်။

အကျဉ်းချုပ်ပြောရရင် DCOM lateral movement ဆိုတာ Windows ရဲ့ ပုံမှန် COM/DCOM architecture ကို အသုံးချထားတဲ့ attack ဖြစ်ပြီး noisy မဖြစ်ဘဲ remote execution လုပ်နိုင်လို့ attacker တွေအတွက် အရမ်းကြိုက်တဲ့နည်းလမ်းပါ။

Defender အနေနဲ့

- mmc.exe၊ svchost.exe -k DcomLaunch
- `-Embedding` parameter
- type 3 logon နဲ့
- RPC port 135 activity တွေကို context နဲ့ပေါင်းပြီး ကြည့်နိုင်ရင် ဒီ attack ကို သေချာဖော်ထုတ်နိုင်ပါတယ်။

---

2) DCOM as LOLBin chaining (mmc → child) 🧩➡️🖥️

Main concept:

mmc.exe ကို run တယ်၊ အဲ့ဒီနောက် type 3 logon (4624) ပေါ်လာတာ၊ ပြီးတော့ mshta.exe နဲ့ PowerShell လို process တွေ ဆက်တိုက် spawn ဖြစ်လာတာကို မြင်ရပါမယ်။ “PowerShell one or

two child processes deep” လို့ပြောထားတဲ့အဓိပ္ပါယ်က attacker က mmc.exe → mshta.exe → powershell.exe ဆိုပြီး အဆင့်အနည်းငယ်ဆင့်ဆင့် payload execute လုပ်နေတာကိုဆိုလိုတာပါ။ ဒီလို chaining လုပ်တာက direct PowerShell execute ထက် detection ကို ရှောင်နိုင်ဖို့အတွက် attacker တွေ မကြာခဏသုံးတဲ့နည်းလမ်းဖြစ်ပါတယ်။

Detail Explanation

mmc.exe ကို DCOM ကနေ launch လုပ်ပြီး attacker က mshta / powershell တို့ကို child အနေနဲ့ run ချတတ်ပါတော့ detection ရှောင်နိုင်တယ်။

> “DCOM + MMC ကိုသုံးပြီး lateral movement လုပ်တဲ့အခါ attacker က တကယ်တမ်း ဘယ် LOLBin ကို run ခဲ့လဲ”

အလွယ်ဆုံးနည်းလမ်းက mmc.exe ကနေ spawn ဖြစ်လာတဲ့ child process တွေကို တိုက်ရိုက်ရှာကြည့်တာ ဖြစ်ပါတယ်။ mmc.exe ကို LOLBin အဖြစ် အသုံးချထားတယ်ဆိုတာ သိပြီးသားဖြစ်တဲ့အတွက်၊ အဲ့ဒီ mmc.exe က ဘာကို ဆက်ပြီး run ခဲ့လဲဆိုတာကို ကြည့်ရင် attacker ရဲ့ နောက်တစ်ဆင့် action ကို ချက်ချင်းမြင်နိုင်ပါတယ်။

```
external_table('Winlog')
| where HostName startswith "alice-pc" and ParentImage endswith "mmc.exe"
| project Timestamp, ParentCommandLine, ParentUser, CommandLine, User
```

alice-pc ဆိုတဲ့ host ပေါ်မှာ mmc.exe ကို parent process အဖြစ်ထားပြီး run ဖြစ်လာတဲ့ child process တွေကိုရှာတာပါ။ ParentImage endswith "mmc.exe" ဆိုတာကြောင့် mmc.exe က spawn လုပ်လိုက်တဲ့ process တွေပဲ filter ဖြစ်ပါတယ်။ အဲ့ဒီ result ကိုကြည့်လိုက်တဲ့အခါ mshta.exe က remote .hta script ကို ခေါ်သုံးနေတာကို မြင်ရပါတယ်။ ဒီအချက်က အရမ်းအရေးကြီးပါတယ်၊ ဘာကြောင့်လဲဆိုရင် mshta.exe ကို Windows ရဲ့ built-in LOLBin တစ်ခုဖြစ်ပြီး attacker တွေက remote script execute လုပ်ဖို့ မကြာခဏ အသုံးချလေ့ရှိတာကြောင့်ပါ။ ဒီနေရာမှာ lateral movement ကို DCOM နဲ့လုပ်ပေမယ့် payload execute လုပ်တဲ့နေရာမှာ mshta.exe ကို အသုံးချထားတာကို သေချာမြင်ရပါတယ်။
ဒီလို child process တစ်ခုတည်းကြည့်တာနဲ့တင် အဖြေကိုရနိုင်ပေမယ့်၊ analyst အနေနဲ့ timeline တစ်ခုလုံးကို နားလည်ချင်ရင် join query ကို အသုံးပြုတာ ပိုကောင်းပါတယ်။

Detection / Indicators: ParentImage endswith `mmc.exe` and suspicious child processes; short-timeline join to catch pre/post events.
Sample query:

```
external_table('Winlog')
| where EventCode==1 and Image endswith "mmc.exe"
| project mmcTime = Timestamp, HostName,
mmcParentCommandLine=ParentCommandLine, ParentUser,
mmcCommandLine=CommandLine, User
| join kind=inner
    (external_table('Winlog')
    | where EventCode in (1, 4624)
    | project Timestamp, HostName, EventCode, TargetUserName,
```

```
DestinationPort, ParentCommandLine, CommandLine
    ) on HostName
| where (mmcTime - Timestamp) between (-2s .. 1s)
| project Timestamp, EventCode, TargetUserName, ParentCommandLine,
CommandLine
```

Detection / Indicators

ဒီ join result ကို timeline အနေနဲ့ကြည့်လိုက်ရင် mmc.exe run ဖြစ်လာတာ၊ အဲ့ဒီနောက် type 3 logon (4624) ပေါ်လာတာ၊ ပြီးတော့ mshta.exe နဲ့ PowerShell လို process တွေ ဆက်တိုက် spawn ဖြစ်လာတာကို မြင်ရပါမယ်။ “PowerShell one or two child processes deep” လို့ပြောထားတဲ့အဓိပ္ပါယ်က attacker က mmc.exe → mshta.exe → powershell.exe ဆိုပြီး အဆင့်အနည်းငယ်ဆင့်ဆင့် payload execute လုပ်နေတာကိုဆိုလိုတာပါ။ ဒီလို chaining လုပ်တာက direct PowerShell execute ထက် detection ကို ရှောင်နိုင်ဖို့အတွက် attacker တွေ မကြာခဏသုံးတဲ့နည်းလမ်းဖြစ်ပါတယ်။

---

3) PowerShell Remoting / WinRM (wsmprovhost.exe) ⚡🔁

Main concept:

WinRM ဆိုတာ Windows ထဲမှာ administrator တွေ remote computer တွေကို စောင့်ကြည့်၊ configure၊ manage လုပ်ဖို့အတွက် သုံးတဲ့ remote management protocol တစ်ခုဖြစ်ပါတယ်။ WinRM ကိုယ်တိုင်က “run command” မလုပ်ပါဘူး။ Remote execution လုပ်ဖို့ WinRM ကို **PowerShell** သို့မဟုတ် **WMI** လို execution technique တစ်ခုနဲ့ တွဲသုံးရပါတယ်။

Detection / Indicators:

Attacker တစ်ယောက်က PowerShell remoting ကို အသုံးပြုပြီး remote computer တစ်လုံးထဲကို session ဖွင့်လိုက်တဲ့အချိန်မှာ Windows အတွင်းမှာ process တစ်ချို့ အလိုအလျောက် ဖွင့်လာပါတယ်။ အရင်ဆုံး WinRM host process ကို DCOM ကနေ launch လုပ်ပါတယ်။ ဒီနေရာမှာ log ထဲမှာ မြင်ရတာက `svchost.exe -k DcomLaunch` ဆိုတဲ့ process ကနေ **wsmprovhost.exe** ဆိုတဲ့ process ကို spawn လုပ်ပေးပါတယ်။ mmc.exe DCOM launch မှာ `-Embedding` flag ပါသလိုပဲ၊ ဒီ wsmprovhost.exe မှာလည်း `-Embedding` ပါနေတာကို တွေ့ရပါမယ်။ ဒါဟာ PowerShell remoting session တစ်ခု ဖွင့်ထားတယ်ဆိုတဲ့ အရေးကြီးတဲ့ indicator ဖြစ်ပါတယ်။

Explanation

wsmprovhost.exe process က PowerShell remoting session ကို ကိုယ်စားပြုပြီး session မပိတ်မချင်း အဲ့ဒီ process က ဆက်လက် run နေပါတယ်။ အရေးကြီးတဲ့အချက်က attacker က native PowerShell command တွေပဲ အသုံးပြုနေသရွေ့ command execution အားလုံးဟာ **wsmprovhost.exe process context** ထဲမှာပဲ ဖြစ်နေပါတယ်။ ဆိုလိုတာက separate powershell.exe process အသစ် မမြင်ရဘဲ၊ activity အားလုံးကို wsmprovhost.exe အတွင်းမှာပဲ log တွေက ဖော်ပြပါလိမ့်မယ်။

Sample queries:

```
external_table('Winlog')
| where EventCode==1 and Image endswith "wsmprovhost.exe"
| project Timestamp, HostName, ParentCommandLine, ParentUser, CommandLine,
User
```

Query ကို run လိုက်တဲ့အခါ alice-pc ပေါ်မှာ wsmprovhost.exe process တစ်ခုပဲ မြင်ရပြီး parent command line က `svchost.exe -k DcomLaunch` ဖြစ်နေတယ်ဆိုရင် ဒီဟာက PowerShell remoting ကို DCOM + WinRM နဲ့ အသုံးပြုထားတယ်ဆိုတာကို သေချာပြောနိုင်ပါတယ်။ ထို့အပြင် user အနေနဲ့ Bob ရဲ့ account ပေါ်နေတယ်ဆိုရင် Bob က remote PowerShell session ဖွင့်ထားတယ်ဆိုတာကိုပါ သိနိုင်ပါတယ်။

ဒါပေမယ့် process တစ်ခုမြင်ရုံနဲ့ "attack" လို့ မဆုံးဖြတ်နိုင်ပါဘူး။ Administrator တွေလည်း PowerShell remoting ကို နေ့စဉ်အသုံးပြုကြတာမို့၊ Bob က attacker လား၊ admin လားဆိုတာကို **context** နဲ့ စစ်ဆေးရပါမယ်။ ဒါကြောင့် wsmprovhost.exe ရဲ့ ProcessId ကို အခြေခံပြီး timeline တစ်ခုလုံးကို ဆက်ပြီးကြည့်ရပါတယ်။ ProcessId ကို filter လုပ်ပြီး registry access၊ file access၊ PowerShell operational log (EID 4103) တွေကိုကြည့်ရင် attacker သို့မဟုတ် admin က ဘာလုပ်နေလဲဆိုတာ ပိုပြီးမြင်နိုင်ပါတယ်။

Filter by ProcessId to build timeline:

```
external_table('Winlog')
| where HostName startswith "alice-pc" and (ProcessId == 2372 or
ParentProcessId == 2372 or WinlogProcessPid == 2372)
```

ဒီ module မှာတော့ PowerShell log တွေရှိတဲ့အတွက် wsmprovhost.exe process အတွင်းမှာ attacker ဘာလုပ်နေတယ်ဆိုတာကို တိတိကျကျ မြင်နိုင်ပါတယ်။ ဒါပေမယ့် အချို့ environment တွေမှာ PowerShell logging မရှိတာလည်း ဖြစ်နိုင်ပါတယ်။ အဲ့ဒီလိုဆိုရင် analysis က ပိုခက်သွားပေမယ့် အားလုံးမဆုံးရှုံးသေးပါဘူး။ Analyst အနေနဲ့ wsmprovhost.exe က network connection ဘာတွေဖွင့်လဲ၊ child process အသစ်တွေ spawn လုပ်လား၊ file တွေကို ရေးလား၊ remote thread injection လုပ်လား စတဲ့ behavior အားလုံးကို စုပေါင်းကြည့်ရပါမယ်။ Attacker က ဒီအဆင့်ပြီးရင် SYSTEM privilege တက်ဖို့ သို့မဟုတ် host ရဲ့ အထူး feature တစ်ခုခုကို အသုံးချဖို့ ကြိုးစားလာနိုင်ပါတယ်။

နောက်ထပ် WinRM lateral movement နည်းလမ်းတစ်ခုက **winrshost.exe** ကို အသုံးပြုတာပါ။ ဒီနည်းလမ်းမှာ attacker က PowerShell remoting မဟုတ်ဘဲ WinRM shell ကို အသုံးပြုပြီး command run တတ်ပါတယ်။ Log ထဲမှာ ဒီဟာကို ဖမ်းချင်ရင် System process ကို port 5985 (HTTP) သို့မဟုတ် 5986 (HTTPS) နဲ့ network connection ဖြစ်လာတဲ့အချိန်နဲ့ winrshost.exe က child process spawn ဖြစ်လာတဲ့အချိန်ကို correlate လုပ်ရပါမယ်။ PowerShell log မရှိရင်လည်း anomalous Windows logon တွေနဲ့ winrshost.exe ရဲ့ strange child process တွေကို grouping လုပ်ပြီး detection လုပ်နိုင်ပါတယ်။

Summary

အကျဉ်းချုပ်ပြောရရင် PowerShell Remoting + WinRM lateral movement က Windows ရဲ့ ပုံမှန် admin workflow ကို အသုံးချထားတဲ့အတွက် attacker လား admin လား ခွဲခြားဖို့ context မရှိရင် အရမ်းခက်ပါတယ်။ ဒါကြောင့် defender အနေနဲ့ process pattern၊ logon type၊ user behavior နဲ့ timeline

correlation ကို အားလုံးပေါင်းပြီး စဉ်းစားနိုင်ရင် ဒီလို stealthy lateral movement ကို ဖော်ထုတ်နိုင်ပါလိမ့်မယ်။

Test : PowerShell Remoting (WinRM) session ထဲမှာ ipconfig.exe ကို ဘယ်လို execute ဖြစ်သွားတာလဲ

လွယ်ဆုံးနည်းလမ်းကတော့ အရင်ကတွေ့ထားပြီးသား wsmprovhost.exe process ကို အခြေခံပြီး စဉ်းစားခြင်းပါပဲ။ PowerShell remoting session တစ်ခု ဖွင့်ထားတဲ့အချိန်မှာ wsmprovhost.exe က attacker (သို့မဟုတ် admin) ရဲ့ PowerShell execution context ကို ကိုယ်စားပြုနေပါတယ်။

အရေးကြီးတဲ့ rule တစ်ခုကို ဒီနေရာမှာ မှတ်ထားရပါမယ်။

PowerShell commandlet တွေ၊ ဥပမာ Get-Process, Get-ChildItem လို cmdlet တွေကို run တာက wsmprovhost.exe process ထဲမှာပဲ execute ဖြစ်ပါတယ်။ အဲ့ဒီအတွက် child process အသစ် မမြင်ရပါဘူး။ ဒါပေမယ့် attacker က PowerShell ထဲကနေ ipconfig.exe လို external executable ကို run လိုက်ရင်တော့ wsmprovhost.exe က parent process ဖြစ်ပြီး child process အသစ်တစ်ခု spawn ဖြစ်လာပါတယ်။ ဒါကြောင့် wsmprovhost.exe ရဲ့ child process ကို ရှာလိုက်တာနဲ့ attacker က PowerShell session ထဲမှာ executable တစ်ခုခု run လုပ်ထားလားဆိုတာကို ချက်ချင်း သိနိုင်ပါတယ်။

ဒီ scenario မှာ ParentProcessId ကို 2372 (wsmprovhost.exe ရဲ့ PID) လို့ filter လုပ်လိုက်တဲ့အခါ ipconfig.exe ကို child process အဖြစ် မြင်ရပါတယ်။ ဒီအချက်တစ်ခုတည်းနဲ့တင် Bob က PowerShell remoting session ထဲကနေ ipconfig.exe ကို run ခဲ့တယ်ဆိုတာကို သေချာပြောနိုင်ပါတယ်။ ipconfig.exe က system discovery အတွက် မကြာခဏ အသုံးပြုတဲ့ command ဖြစ်တဲ့အတွက် attacker activity ဖြစ်နိုင်သလို legitimate admin activity လည်း ဖြစ်နိုင်ပါတယ်။ ဒါကြောင့် နောက်ထပ် context ကို ဆက်ပြီးကြည့်ဖို့ လိုပါတယ်။

ဒီနေရာမှာ စိတ်ဝင်စားစရာက PowerShell Pipeline Execution event (Event ID 4103) ရဲ့ timing ပါ။ သာမန်အားဖြင့် PowerShell command execution ဆိုရင် 4103 event ကို အရင်မြင်ရမယ်လို့ ထင်လို့ရပါတယ်။ ဒါပေမယ့် ဒီ timeline ကို ကြည့်လိုက်တဲ့အခါ ipconfig.exe process က အရင်ဆုံး execute ဖြစ်သွားပြီးမှ 4103 event ပေါ်လာတာကို တွေ့ရပါတယ်။ ဒီအချက်က PowerShell internals ကို နားလည်ဖို့ အရေးကြီးပါတယ်။

အကြောင်းရင်းက PowerShell pipeline execution log (4103) က “PowerShell pipeline ကို build လုပ်ပြီး execute လုပ်ပြီးသွားပြီ” ဆိုတဲ့အချိန်မှာမှ log ထုတ်ပေးတာပါ။ PowerShell ထဲကနေ external executable တစ်ခုကို ခေါ်လိုက်တဲ့အချိန်မှာ PowerShell က process creation ကို အရင်ဆုံးလုပ်ပြီး ipconfig.exe ကို OS ကို handed off လုပ်ပေးပါတယ်။ အဲ့ဒီနောက်မှ pipeline execution အဖြစ် PowerShell log ထဲမှာ 4103 event ကို ရေးပါတယ်။ ဒါကြောင့် timeline မှာ ipconfig.exe → 4103 ဆိုတဲ့ order ကို တွေ့ရတာက ပုံမှန် behavior ဖြစ်ပါတယ်၊ log error မဟုတ်ပါဘူး။

ဒုတိယ query က ဒီအချက်ကို verify လုပ်ဖို့အတွက် timeline ကို ပိုကျယ်ကျယ်ဖမ်းထားတာပါ။ wsmprovhost.exe ရဲ့ ProcessId နဲ့ ParentProcessId ကိုပါ filter လုပ်ထားပြီး၊ PowerShell pipeline event (4103) နဲ့ ipconfig.exe terminate event (4689) ကိုပါ ထည့်ကြည့်ထားပါတယ်။ ဒီလိုလုပ်လိုက်တဲ့အခါ wsmprovhost.exe session စတင်တာ၊ ipconfig.exe spawn ဖြစ်တာ၊ ipconfig.exe ပြီးဆုံး

သွားတာ၊ ပြီးမှ PowerShell pipeline event ပေါ်လာတာဆိုတဲ့ အစီအစဉ်ကို တစ်ကြောင်းတည်းလို timeline အနေနဲ့ မြင်နိုင်ပါတယ်။

ဒီဟာက defender အတွက် အရေးကြီးတဲ့ သင်ခန်းစာတစ်ခုကို ပေးပါတယ်။ PowerShell Remoting session ထဲမှာ child process တွေကို တွေ့ရပြီဆိုရင် အဲ့ဒါက cmdlet မဟုတ်ဘဲ external executable run ခဲ့တယ်ဆိုတဲ့ strong signal ဖြစ်ပါတယ်။ ထို့အပြင် PowerShell log event တွေရဲ့ timing ကို နားမလည်ရင် “ဘာကြောင့် ipconfig.exe အရင်၊ 4103 နောက်” ဆိုပြီး လွဲမှားစွာ ထင်မြင်နိုင်ပါတယ်။ Timeline ကို နားလည်ပြီး context နဲ့ကြည့်နိုင်ရင် attacker behavior ကို ပိုမိုတိကျစွာ ခွဲခြမ်းစိတ်ဖြာနိုင်ပါလိမ့်မယ်။

---

4) WMI lateral movement (WmiPrvSE.exe) 🍫 🛠️

Main concept:

WMI lateral movement ဟာ Windows ရဲ့ legitimate management feature ကို အသုံးချထားတာဖြစ်ပြီး, WmiPrvSE.exe ရဲ့ child process တွေ၊ ADMIN$/C$ share activity တွေ၊ SMB + process execution timeline ကို ပေါင်းကြည့်နိုင်ရင် attacker ရဲ့ movement ကို သေချာစွာ ဖော်ထုတ်နိုင်ပါတယ်။

Full Explanation

Attacker က အရင်ဆုံး target system နဲ့ authentication / connection ကို DCOM သို့မဟုတ် WinRM နဲ့ set up လုပ်ပါတယ်။ အဲ့ဒီနောက် **WMI ရဲ့ Win32_Process object** ကို အသုံးပြုပြီး process အသစ်တစ်ခုကို create လုပ်ခိုင်းပါတယ်။ Win32_Process.Create method က command line string တစ်ခုလက်ခံပြီး Windows process အသစ်ကို spawn လုပ်ပေးနိုင်ပါတယ်။ ဒါကြောင့် attacker အနေနဲ့ “remote cmd execute” လုပ်ဖို့ WMI ကို အသုံးချနိုင်ပါတယ်။

WMI lateral movement ကို detection လုပ်တဲ့အခါ **အရေးကြီးဆုံးမှတ်ထားရမယ့်အချက်** က command ကို execute လုပ်တဲ့အချိန်မှာ **WmiPrvSE.exe** (WMI Provider Service) process က **child process အသစ်တစ်ခု spawn လုပ်မယ်** ဆိုတာပါ။ ဒါကြောင့် hunting သို့မဟုတ် forensics လုပ်တဲ့အခါ WmiPrvSE.exe ရဲ့ child process တွေကို ကြည့်တာက အလွန် practical ဖြစ်ပါတယ်။ Spawn ဖြစ်လာတဲ့ child process က မကြာခဏ attacker က upload လုပ်ထားတဲ့ executable ဖြစ်တတ်သလို, cmd.exe, powershell.exe လို LOLBin တစ်ခုလည်း ဖြစ်နိုင်ပါတယ်။

```
external_table('Winlog')
| where HostName startswith "alice-pc" and ParentImage endswith
"WmiPrvSe.exe"
| project Timestamp, ParentCommandLine, ParentUser, CommandLine, User
```

Query ကို run လိုက်တဲ့အခါ alice-pc ပေါ်မှာ WmiPrvSE.exe က parent process ဖြစ်ပြီး child process အနေနဲ့ command တွေ run နေတာကို မြင်ရပါတယ်။ `whoami` နဲ့ `net localgroup administrators` လို command တွေက အထူးသဖြင့် စိုးရိမ်စရာပါ။ ဘာကြောင့်လဲဆိုရင် `whoami` က “ငါဘယ် user context နဲ့ run နေလဲ” ဆိုတာစစ်တာဖြစ်ပြီး, `net localgroup administrators` က local admin group ကို enumerate လုပ်တာဖြစ်လို့ attacker ရဲ့ discovery / privilege validation behavior ဖြစ်နိုင်ပါတယ်။

ဒီ timeline ထဲမှာ ပထမဆုံး စိတ်ဝင်စားစရာ event က **o2bpbxt.exe** ဆိုတဲ့ executable ပါ။ နာမည်က random လိုဖြစ်နေတဲ့အတွက် legitimate software ဖြစ်နိုင်ချေ နည်းပါတယ်။ Event အရေအတွက်လည်း မများတဲ့အတွက် executable နာမည်ကို အခြေခံပြီး log တွေအားလုံးကို ချိတ်ဆက်ကြည့်နိုင်ပါတယ်။ Message field ထဲမှာ o2bpbxt.exe ပါတဲ့ event တွေကို ကြည့်လိုက်တဲ့အခါ **Windows Security EID 5145 (Detailed File Share)** နဲ့ **Sysmon EID 3 (Network Connect)** ကို တွေ့ရပါတယ်။

Hunt by message containing a suspicious exe:

```
external_table('Winlog')
| where Message contains "o2bpbxt.exe"
```

ဒီ events တွေက အရမ်းအရေးကြီးပါတယ်။ **5145 event** က remote share တစ်ခုကို file write လုပ်ထားတယ်ဆိုတဲ့ အဓိပ္ပါယ်ဖြစ်ပြီး, Sysmon network connection event ထဲမှာ mshta.exe attack မှာ မြင်ခဲ့တဲ့ malicious IP address ကို ချိတ်ဆက်နေတာကို တွေ့ရပါတယ်။ ဒါကြောင့် o2bpbxt.exe က **C2 beacon သို့မဟုတ် malware payload** ဖြစ်နိုင်ချေ အရမ်းမြင့်ပါတယ်။

နောက်တစ်ဆင့်အနေနဲ့ defender အနေနဲ့ “ဒီ `file` ဘယ်ကလာတာလဲ” ဆိုတာကို စုံစမ်းရပါမယ်။ အဲ့ဒီအတွက် 5145 event တွေကို ပြန်ကြည့်လိုက်ရင် Bob ရဲ့ account က **alice-pc ရဲ့ ADMIN$ share** ထဲကို o2bpbxt.exe ကို upload လုပ်ထားတာကို တွေ့ရပါတယ်။ ADMIN$ share ဆိုတာ Windows ရဲ့ default administrative share ဖြစ်ပြီး remote admin privilege ရှိတဲ့ user တွေပဲ write လုပ်နိုင်ပါတယ်။ ဒါကြောင့် Bob account ဟာ alice-pc ပေါ်မှာ admin privilege ရှိပြီး lateral movement လုပ်နိုင်တဲ့အခြေအနေမှာရှိနေတယ်ဆိုတာကို ထပ်မံအတည်ပြုနိုင်ပါတယ်။

ဒီလို random နာမည်ပါတဲ့ executable ကို ADMIN$ သို့မဟုတ် C$ share ထဲ upload လုပ်ပြီး execute လုပ်တာဟာ **commodity malware** နဲ့ **pentester tool** တွေမှာ အရမ်းတွေ့ရတဲ့ pattern ဖြစ်ပါတယ်။ Pentester တစ်ယောက်က stealth ကို အဓိကမထားရင် ဒီလို out-of-the-box tool တွေကို တိုက်ရိုက်သုံးတတ်ပါတယ်။ Advanced attacker တွေလည်း environment ကို “detection မကောင်းဘူး” လို့ ခန့်မှန်းမိရင် အချိန်ခွေတာဖို့ ဒီလိုနည်းလမ်းကို သုံးနိုင်ပါတယ်။ တခါတရံ C2 framework ထဲက default executable name ကို rename မလုပ်ဘဲ မေ့လျော့သွားတာလည်း ဖြစ်နိုင်ပါတယ်။

> ဒါပေမယ့် ဒီနေရာမှာ အရေးကြီးတဲ့ သင်ခန်းစာက **“executable နာမည်တစ်ခုတည်းနဲ့ မဆုံးဖြတ်ပါနဲ့”** ဆိုတာပါ။ Name ကို IOC အနေနဲ့ သုံးနိုင်ပေမယ့် အဓိကက environment context, user behavior, timeline correlation ကို အခြေခံပြီး ဆုံးဖြတ်ရပါမယ်။

နောက်ဆုံးအနေနဲ့ WmiPrvSE.exe ရဲ့ အခြား child process တွေကို ကြည့်ရင်

```
cmd.exe /Q /c net localgroup administrators 1> \\127.0.0.1\ADMIN$\__xxxx 2>&1
```

လို command line တွေကို တွေ့ရပါတယ်။ ဒါတွေက **impacket wmiexec.py** tool ရဲ့ artifact ဖြစ်ပါတယ်။ wmiexec.py က WMI ကို အသုံးပြုပြီး semi-interactive shell တစ်ခု ဖန်တီးပေးတဲ့ tool ဖြစ်ပါတယ်။

wmiexec.py ရဲ့ အလုပ်လုပ်ပုံက အရမ်းရိုးရှင်းပါတယ်။ အရင်ဆုံး SMB နဲ့ remote machine ကို ချိတ်ပြီး ADMIN$ share ထဲမှာ file တစ်ခု create လုပ်ပါတယ်။ ပြီးတော့ command တစ်ခုကို execute လုပ်ပြီး stdout နဲ့ stderr ကို အဲ့ဒီ file ထဲ redirect လုပ်ပါတယ်။ နောက်တစ်ဆင့်မှာ attacker က SMB နဲ့ အဲ့ဒီ file

ကို ပြန်ဖတ်ပါတယ်။ ဒီလုပ်ငန်းစဉ်ကို ထပ်ခါထပ်ခါလုပ်ပြီး interactive cmd.exe လို experience ရအောင်လုပ်ထားတာပါ။

ဒီနည်းလမ်းက stealth မကောင်းပေမယ့် environment အချို့မှာတော့ ယနေ့အချိန်ထိ အသုံးများနေဆဲပါ။ Attacker က command line ကို obfuscate လုပ်နိုင်ပေမယ့် **pattern အနေနဲ့တော့ "file ကို share ထဲ upload → command execute"** ဆိုတာ မပြောင်းလဲပါဘူး။ ဒီ pattern ကို နားလည်ထားရင် WMI lateral movement ကို ဖော်ထုတ်ဖို့ အများကြီးအထောက်အကူဖြစ်ပါလိမ့်မယ်။

---

5) Scheduled Tasks (Task create/delete) ⏰ 📋

Main concept:

Scheduled Tasks lateral movement ဟာ Windows ရဲ့ ပုံမှန် administration feature ကို အသုံးချထားတာဖြစ်ပြီး detection က log တစ်ခုတည်းနဲ့ မလုံလောက်ပါဘူး။ Random task name, random executable, create → delete အချိန်အလွန်တိုတာ စတဲ့ anomaly တွေကို မြင်ရင် သတိထားရမယ်။ နောက်ဆုံးအနေနဲ့ 4698 event ကို network logon, RPC traffic, file share activity တွေနဲ့ **context အပြည့်အစုံနဲ့** ချိတ်ဆက်ကြည့်နိုင်ရင် remote scheduled task attack ကို သေချာဖော်ထုတ်နိုင်ပါလိမ့်မယ်။

Detai Expanation

Windows Scheduled Task ဆိုတာ administrator တွေ ပုံမှန်သုံးနေတဲ့ feature ဖြစ်ပြီး၊ သတ်မှတ်ထားတဲ့ အချိန်မှာ သို့မဟုတ် condition တစ်ခုဖြစ်လာရင် command / program တစ်ခု run ဖို့ အသုံးပြုကြပါတယ်။ Attacker တွေက ဒီ legitimate feature ကိုပဲ အသုံးချပြီး malicious execution လုပ်သွားကြတာပါ။

Attacker က remote system ပေါ်မှာ scheduled task တစ်ခု create လုပ်တဲ့အချိန်မှာ underlying mechanism အနေနဲ့ **RPC (Remote Procedure Call)** ကို အသုံးပြုပါတယ်။ ပထမဆုံး connection က **port 135** ကို svchost.exe ဆီ ချိတ်ပါတယ်။ ဒါက RPC endpoint mapper ဖြစ်ပြီး "scheduled task service ကို ဘယ် port မှာ ဆက်ရမလဲ" ဆိုတာကို negotiate လုပ်ဖို့ပါ။ အဲ့ဒီနောက် Windows က random high port တစ်ခုကို ထပ်ဖွင့်ပြီး task ကို တကယ်တမ်း create လုပ်သွားပါတယ်။ Network အဆင့်မှာကြည့်ရင် port 135 → random port ဆိုတဲ့ pattern ကို တွေ့ရတတ်ပါတယ်။

Detection အနေနဲ့ဆိုရင်

scheduled task lateral movement ကို ဖမ်းဖို့ **special technique တစ်ခုလိုအပ်တယ်** ဆိုတာမဟုတ်ပါဘူး။ အကောင်းဆုံးနည်းလမ်းကတော့ **scheduled task anomaly detection** ပဲ ဖြစ်ပါတယ်။ Windows ကိုယ်တိုင် log ထုတ်ပေးထားတဲ့ event တွေထဲမှာ အရေးကြီးဆုံးက **Windows Security Event ID 4698** ဖြစ်ပါတယ်။ 4698 ဆိုတာ scheduled task အသစ်တစ်ခု create လုပ်လိုက်တယ်ဆိုတဲ့ event ဖြစ်ပြီး task name, task content (ဘာ command run မလဲ) စတဲ့ အချက်အလက်တွေကို ထုတ်ပေးပါတယ်။ Task ကို delete လုပ်လိုက်ရင်တော့ **Event ID 4699** ကို တွေ့ရပါမယ်။

```
external_table('Winlog')
| where HostName startswith "alice-pc" and (EventCode == 4698 or EventCode
```

```
== 4699)
| project Timestamp, EventCode, TaskName, TaskContent, Message
```

Query ကို run လိုက်တဲ့အခါ alice-pc ပေါ်မှာ 4698 (task creation) နဲ့ 4699 (task deletion) event နှစ်ခုကို **တစ်စက္ကန့်အတွင်းမှာပဲ** တွေ့ရပါတယ်။ ဒီအချက်က အရမ်း suspicious ဖြစ်ပါတယ်။ Legitimate admin တွေက scheduled task တစ်ခု create လုပ်ပြီး ချက်ချင်း delete လုပ်တာ များမများမဟုတ်ပါဘူး။ ထို့အပြင် task name က `\spDNoafD` လို random string ဖြစ်နေပြီး, run ခိုင်းထားတဲ့ executable ကလည်း C:\ root ထဲမှာ random နာမည်နဲ့ရှိနေတယ်ဆိုရင် malicious ဖြစ်နိုင်ချေ အရမ်းမြင့်ပါတယ်။

ဒါပေမယ့် ဒီ event ကိုကြည့်လိုက်ရင် စိတ်ရှုပ်စရာတစ်ခုရှိပါတယ်။ Event 4698 ထဲမှာ **"remote ကနေ လုပ်တယ်"** ဆိုတာကို တိုက်ရိုက်ပြောထားတာ မရှိပါဘူး။ Event ထဲမှာ ပြထားတဲ့ user က **local Administrator account** ဖြစ်နေတယ်ဆိုရင် "local admin က legitimate task create လုပ်တာလား" လို့ ထင်နိုင်ပါတယ်။ အရင် event တွေမှာတော့ Bob account က network logon လုပ်ပြီး execution လုပ်နေတာကို မြင်လို့ remote attack လို့ ခန့်မှန်းလို့ရခဲ့ပါတယ်။ ဒီနေရာမှာတော့ scheduled task ကို local Administrator context နဲ့ create လုပ်ထားတဲ့အတွက် confusion ဖြစ်စရာပါ။

Defender Note

အဲ့ဒီကြောင့် defender အနေနဲ့ scheduled task event တစ်ခုတည်းနဲ့ မဆုံးဖြတ်သင့်ပါဘူး။ **4698 event ကို network activity နဲ့ correlate လုပ်ဖို့လိုပါတယ်**။ ဥပမာအားဖြင့် 4698 ဖြစ်တဲ့အချိန်အနီးမှာ port 135 ကို svchost.exe ဆီ network connection ဖြစ်လာလား, type 3 logon (network logon) ဖြစ်လာလား, ADMIN$ သို့မဟုတ် C$ share ကို file write လုပ်ထားတဲ့ 5145 event တွေရှိလားဆိုတာကို အတူတကွကြည့်ရပါမယ်။ ဒီလို grouping လုပ်လိုက်မှ "ဒီ scheduled task ဟာ remote attacker က RPC ကနေ create လုပ်ထားတာ" ဆိုတာကို ယုံကြည်စရာ evidence နဲ့ပြောနိုင်ပါလိမ့်မယ်။

---

6) Remote Service Creation & Named Pipes (RPC/NP) 🛠️🪈

Main Concept

Service Control Manager (SCM) ကို အသုံးချပြီး attacker က **remote service creation** လုပ်တဲ့ lateral movement တစ်မျိုးပါ။ **RPC (Remote Procedure Call)** ကနေတဆင့် service အသစ်ထည့်တာ၊ ပြင်တာ၊ start/stop လုပ်တာတွေကို remote ကနေ လုပ်လို့ရပါတယ်။

Full Explanation

Windows ထဲမှာ **service** ဆိုတာ system start တက်တဲ့အချိန်ကတည်းက background မှာ run နေတဲ့ program တွေပါ။ ဥပမာ antivirus service, update service လိုမျိုးတွေပါ။
ဒီ service တွေအားလုံးကို ထိန်းချုပ်ပေးနေတာက **Service Control Manager** ဖြစ်ပြီး၊ အဲ့ဒီ SCM ကို Windows က **remote administration** အတွက်ပါ ဖွင့်ထားပါတယ်။

အဲ့ဒီ remote administration ကို **RPC (Remote Procedure Call)** နည်းလမ်းနဲ့လုပ်ပြီး RPC ကနေတဆင့် service အသစ်ထည့်တာ၊ ပြင်တာ၊ start/stop လုပ်တာတွေကို remote ကနေ လုပ်လို့ရပါတယ်။
ဒါကြောင့် attacker က target machine ပေါ်မှာ **administrator privilege** ရထားပြီဆိုရင် SCM ကို remote ကနေ ချိတ်ပြီး **service အသစ်တစ်ခု install လုပ်နိုင်ပါတယ်**။

Sample Scenario

```
external_table('Winlog')
| where HostName startswith "alice-pc" and EventCode == 7045
```

alice-pc ကိုကြည့်လိုက်တဲ့အခါ
Windows log ထဲမှာ **Event ID 7045** ကိုတွေ့ရပါတယ်။
Event 7045 ဆိုတာ “service အသစ်တစ်ခု install လုပ်လိုက်တယ်” ဆိုတဲ့ event ပါ။
ဒီ event ကိုကြည့်တော့ service နှစ်ခု ထည့်ထားတာကိုတွေ့ရပါတယ်။
UpdaterSvc ဆိုတာက legitimate service လို့ထင်ရနိုင်ပေမယ့်
zgzN ဆိုတဲ့ service name က random စာလုံးတွေဖြစ်ပြီး
အရမ်း suspicious ဖြစ်ပါတယ်။
ဒါကြောင့် zgzN service ကို အဓိကထားပြီး
အဲ့ဒီ event ဖြစ်တဲ့အချိန်နားက log တွေအားလုံးကို timeline လိုက်ကြည့်လိုက်ပါတယ်။

```
let ts1 = datetime("2023-04-01T17:29:27.658Z");
let ts2 = datetime("2023-04-01T17:29:28.434Z");

external_table('Winlog')
| where HostName startswith "alice-pc" and Timestamp between (ts1 .. ts2)
```

Timeline ကို story လိုက်ပြောရရင်—

ပထမဆုံး **Bob account** က
alice-pc ကို **Type 3 logon** နဲ့ authenticate လုပ်ပါတယ်။
Type 3 logon ဆိုတာ network ကနေ ဝင်လာတဲ့ logon ဖြစ်ပြီး
“ဒီ action တွေကို remote ကနေ လုပ်နေတယ်” ဆိုတာကို ပြပါတယ်။
ပြီးတော့ Bob က
`\\*\IPC$` share ထဲက
svcctl နဲ့ **srvsvc** ဆိုတဲ့ named pipe တွေကို access လုပ်ပါတယ်။

> Named pipe ဆိုတာ Windows မှာ
> process နှစ်ခုကြား ဆက်သွယ်ဖို့ သုံးတဲ့ communication channel တစ်မျိုးပါ။
> svcctl pipe က
> Service Control Manager ကို control လုပ်ဖို့ သုံးတဲ့ pipe ဖြစ်ပြီး
> srvsvc က server-related RPC call တွေအတွက် သုံးပါတယ်။
> Remote user တစ်ယောက်က
> svcctl pipe ကို access လုပ်နေပြီဆိုရင်
> **service create/modify လုပ်တော့မယ်** ဆိုတာကို အားနည်းမရှိဘဲ ခန့်မှန်းလို့ရပါတယ်။

အဲ့ဒီနောက် Bob က
BBolhYzk.exe ဆိုတဲ့ executable ကို
ADMIN$ share ကို အသုံးပြုပြီး
`C:\Windows` folder ထဲ upload လုပ်ပါတယ်။

ADMIN$ ဆိုတာက
administrator privilege ရှိသူတွေ အတွက်သာ access ရတဲ့ hidden share ဖြစ်ပါတယ်။
ပြီးတော့ attacker က
zgzN ဆိုတဲ့ service အသစ်ကို install လုပ်ပြီး
service path ကို
`%systemroot%\BBoIhYzk.exe`
လို့ သတ်မှတ်လိုက်ပါတယ်။

အဲ့ဒီအချိန်မှာ Windows ရဲ့
services.exe process က
zgzN service ကို start လုပ်ပေးပါတယ်။
services.exe က SYSTEM privilege နဲ့ run နေတာဖြစ်တဲ့အတွက်
BBoIhYzk.exe ကို SYSTEM user နဲ့ spawn လုပ်သွားပါတယ်။
ဒီနေရာမှာ attacker က
SYSTEM privilege ကို ရပြီးသားဖြစ်သွားပါပြီ။

BBoIhYzk.exe run ဖြစ်လာတဲ့အချိန်
named pipe တစ်ခုကို create လုပ်ပါတယ်—

`\RemCom_communication`

ဒီ name ကိုကြည့်တာနဲ့
ဒီ tool က **RemCom** ဆိုတဲ့ PsExec-like tool ဖြစ်တယ်ဆိုတာကို သိနိုင်ပါတယ်။
RemCom က
PsExec ကို mimic လုပ်ထားတဲ့ custom tool တစ်မျိုးဖြစ်ပြီး
remote command execution အတွက် သုံးပါတယ်။
နောက်ထပ်

- `\RemCom_stdin*`
- `\RemCom_stdout*`
- `\RemCom_stderr*`
  ဆိုတဲ့ pipes တွေကိုလည်း create လုပ်ပြီး
  attacker နဲ့ target ကြား command input/output ကို handle လုပ်ပါတယ်။

အဲ့ဒီနောက် attacker က
remote ကနေ named pipe ကို connect လုပ်ပြီး
cmd.exe ကို spawn လုပ်ခိုင်းကာ
interactive shell တစ်ခုလို သုံးနိုင်ပါတယ်။

ဒီ attack ကို အကျဉ်းချုပ်ပြောရရင်—

Bob က network ကနေ authenticate လုပ်တယ်
svcctl / srvsvc pipe တွေကို access လုပ်တယ်
ADMIN$ ကနေ executable upload လုပ်တယ်
Event 7045 နဲ့ service install ဖြစ်တယ်

services.exe က SYSTEM နဲ့ payload ကို run ပေးတယ်
named pipe ကနေ remote shell ကို control လုပ်တယ်

ဒီ sequence ကို တွေ့ရင်
RPC over Named Pipes (RPC/NP) + remote service creation attack ဖြစ်တယ်လို့ အတိအကျ ပြောလို့ရပါတယ်။

အရမ်းအရေးကြီးတဲ့ security insight တစ်ခုရှိပါတယ်။

RemCom (နဲ့ impacket psexec.py) က
named pipe တွေကို create လုပ်တဲ့အချိန်
null DACL နဲ့ create လုပ်ထားပါတယ်။
null DACL ဆိုတာ
“ဘယ် user မဆို authentication မလိုဘဲ pipe ကို connect လုပ်လို့ရတယ်”
ဆိုတဲ့ အဓိပ္ပါယ်ပါ။
ဒါကြောင့်
RemCom service run နေတဲ့ machine ပေါ်မှာ
အခြား attacker တစ်ယောက်က
ကိုယ်ပိုင် client တစ်ခုရေးပြီး
already-open RemCom pipe ကို connect လုပ်လိုက်ရင်
SYSTEM privilege ကို hijack လုပ်နိုင်ပါတယ်။
ဒါဟာ offensive tool တွေအတွင်းက
မသိကြသေးတဲ့ vulnerability တစ်ခုဖြစ်ပြီး
အထူးသဖြင့် impacket psexec.py မှာ တွေ့ရပါတယ်။

Defender အနေနဲ့

environment ထဲမှာ
\RemCom* နဲ့စတဲ့ named pipes တွေကို
အထူးသတိထား monitor လုပ်သင့်ပါတယ်။
Red team / pentester engagement တွေမှာလည်း
ဖြစ်နိုင်ရင်
wmiexec.py လို
named pipe exposure မရှိတဲ့ tool တွေကို
အသုံးပြုဖို့ တောင်းဆိုတာက ပိုလုံခြုံပါတယ်။
ဒီအပိုင်းကို နားလည်သွားရင်
service-based lateral movement ကို
log ထဲကနေ confidence အပြည့်နဲ့ ဖော်ထုတ်နိုင်ပါပြီ။

---

7) RPC/TCP service creation (stealthier) 🌫️ ➡️ 🌐

Main Concept:

RPC/TCP နဲ့ service creation လုပ်တဲ့ lateral movement ဟာ artifact နည်းပြီး stealthy ဖြစ်ပါတယ်။ ဒါပေမယ့် log မရှိတာ မဟုတ်ပါဘူး။ Log တွေဟာ ပိုပြီး “အနည်းငယ် ခွဲခြားစဉ်းစားရတဲ့” log တွေ ဖြစ်သွားတာပါ။ Defender အနေနဲ့ environment ကို နားလည်ပြီး authentication, file share, service creation, process execution, network connection တွေကို timeline အဖြစ် ပေါင်းကြည့်နိုင်ရင် ဒီလို stealthy attack တွေကိုပါ ဖော်ထုတ်နိုင်ပါတယ်။

Full Explanation

Background

အရင်တုန်းက RemCom လို tool တွေဟာ **RPC over Named Pipes (RPC/NP)** ကို အသုံးများခဲ့ပါတယ်။ အဲ့ဒီနည်းလမ်းမှာ svcctl နဲ့ srvsvc လို named pipe တွေကို access လုပ်ရတဲ့အတွက် log ထဲမှာ artifact တွေ များများကျန်ခဲ့ပါတယ်။ ဒါကြောင့် detection လုပ်ဖို့ အလွယ်တကူ ဖြစ်ပါတယ်။

RPC/TCP ကို သုံးတဲ့ service creation ကတော့ အဲ့ဒီ pipe artifact တွေ မသုံးတော့ပါဘူး။ Service Control Manager ကို **TCP-based RPC** နဲ့ တိုက်ရိုက် ချိတ်ဆက်လိုက်တာဖြစ်လို့ svcctl pipe access, RemCom pipe creation လို obvious indicator တွေ မပေါ်တော့ပါဘူး။ ဒါကြောင့် offensive tool အများစုဟာ default အနေနဲ့ RPC/TCP ကို သုံးလာကြပါတယ်။

Sample Scenario

UpdaterSvc service creation ကို timeline လိုက်ကြည့်ရင် အခြေအနေကို သေချာမြင်နိုင်ပါတယ်။

ပထမဆုံး Bob account က alice-pc ကို **logon type 3** နဲ့ authenticate လုပ်ပါတယ်။ ဒီ logon type က network ကနေ remote access လုပ်နေတယ်ဆိုတာကို ပြပါတယ်။ ဒါပေမယ့် ဒီ logon တစ်ခုတည်းနဲ့တော့ ဘာလုပ်မလဲဆိုတာကို မသိနိုင်သေးပါဘူး။ File copy လုပ်မလား၊ WMI သုံးမလား၊ scheduled task လုပ်မလားဆိုတာ မခန့်မှန်းနိုင်ပါဘူး။

အဲ့ဒီနောက် Bob က **53we.exe** ဆိုတဲ့ executable ကို target system ကို upload လုပ်ပါတယ်။ ဒီ file upload က ADMIN$ သို့မဟုတ် C$ share ကနေ လာတာဖြစ်နိုင်ပြီး၊ Windows Security log မှာ **5145 event** (Detailed File Share) အဖြစ် ကျန်ပါတယ်။ File name က random ဖြစ်နေတာက legitimate admin tool ထက် malware ဖြစ်နိုင်ခြေ ပိုမြင့်စေပါတယ်။

ပြီးတော့ Bob က နောက်တစ်ခါ authenticate ထပ်လုပ်ပါတယ်။ ဒီ authenticate ကို များသောအားဖြင့် RPC/TCP session အသစ်တစ်ခု ဖန်တီးဖို့ သုံးပါတယ်။ ဒီနေရာမှာ svcctl pipe access လို artifact မရှိတဲ့အတွက် attack လုပ်နေတယ်ဆိုတာကို ချက်ချင်း မမြင်နိုင်ပါဘူး။

အဲ့ဒီ authenticate အပြီးမှာ Bob က **UpdaterSvc** ဆိုတဲ့ service ကို create လုပ်ပါတယ်။ ဒီအချိန်မှာ Windows log ထဲမှာ **Event ID 7045** ထွက်လာပြီး service အသစ်တစ်ခု install လုပ်လိုက်တယ်ဆိုတာကို ပြပါတယ်။ Service name က UpdaterSvc ဖြစ်နေတာကြောင့် တော်တော် legitimate လို မြင်ရနိုင်ပြီး zgzN လို random name ထက် detection ခက်ပါတယ်။

Service install ဖြစ်ပြီးတာနဲ့ Windows ရဲ့ **services.exe** process က service ကို start လုပ်ပေးပြီး **53we.exe ကို SYSTEM privilege နဲ့ spawn လုပ်ပါတယ်**။ ဒီ point မှာ attacker က fully privileged code execution ကို ရထားပြီးသားဖြစ်ပါတယ်။

53we.exe run ဖြစ်ပြီးချင်းမှာတော့ RemCom လို named pipe interaction မရှိဘဲ **တိုက်ရိုက် C2 server ကို network connection ထွက်သွားပါတယ်**။ ဒီ network connection က attacker-controlled infrastructure နဲ့ ချိတ်ဆက်ဖို့အတွက်ဖြစ်ပြီး Sysmon Event ID 3 လို network connect log တွေမှာသာ မြင်ရပါမယ်။

Detection

ဒီ attack ကို RemCom example လို မျက်လုံးထဲထွက်နေအောင် မမြင်ရတဲ့ အကြောင်းရင်းက artifact တွေ နည်းသွားလို့ပါ။ svcctl pipe access မရှိ၊ RemCom named pipe မရှိ၊ interactive shell pattern မရှိပါဘူး။ Log ထဲမှာ ကျန်တာတွေက logon type 3, file upload, service creation, SYSTEM process spawn, outbound network connection ဆိုတဲ့ **generic event တွေ** ပဲ ဖြစ်ပါတယ်။

ဒါကြောင့် detection လုပ်ချင်ရင် single event ကို မကြည့်ဘဲ **context ကို ပေါင်းစည်းပြီး တွေးရပါမယ်**။ Service install event (7045) တစ်ခုကို ကြည့်တဲ့အခါ “ဒီ service ကို ဘယ်သူ create လုပ်လဲ၊ အချိန်နားမှာ network logon ရှိလား၊ service path က unusual လား၊ recently upload လုပ်ထားတဲ့ file ကို run လုပ်နေတာလား” ဆိုတာတွေကို ဆက်စပ်စဉ်းစားရပါမယ်။

ဒီလို scenario တွေမှာ အသုံးဝင်တဲ့ analysis technique တစ်ခုက **least occurrence** ဖြစ်ပါတယ်။ Environment ထဲမှာ UpdaterSvc လို service name မကြာခဏ တွေ့ရလား၊ SYSTEM က C:\Windows\53we.exe လို random executable ကို run လုပ်တာ မကြာခဏ ဖြစ်လားဆိုတာကို historical data နဲ့ နှိုင်းယှဉ်ရင် anomalous ဖြစ်နေမလားဆိုတာ သိနိုင်ပါတယ်။

Test : Beacon Upload

Which share did the attacker upload 53we.exe to?

Answer with the winlog.event_data.ShareName value as it appears in the logs.

```
external_table('Winlog')
| where HostName startswith "alice-pc"
| where EventCode == 5145
| where Message contains "53we.exe"
| project Timestamp, User, ShareName, Message
```

The answer : *\C$

---

8) SCShell (hijack existing service ImagePath) — even stealthier 🕵️‍♂️🔁

Main Concept:

အရင်မှာ ကျွန်တော်တို့ ကြည့်ခဲ့တဲ့ service-based lateral movement တွေက service **အသစ်တစ်ခု install** လုပ်ပါတယ်။ အဲ့ဒီလိုဆိုရင် Windows log ထဲမှာ Event ID 7045 (service installed) ဆိုတာက အရမ်းကောင်းတဲ့ indicator ဖြစ်ပါတယ်။ Defender တွေအတွက် “အို၊ service အသစ် ထည့်ထားတယ်” လို့ ချက်ချင်းသိနိုင်ပါတယ်။

ဒါပေမယ့် attacker အချို့က ဒီ indicator ကိုရှောင်ချင်ပါတယ်။ “service အသစ် မထည့်ဘဲ SYSTEM privilege နဲ့ code run လုပ်လို့ရမလား” ဆိုပြီး စဉ်းစားလာကြပါတယ်။ အဲ့ဒီအဖြေက **SCShell** ပါ။

Full Explanation

SCShell ဆိုတာ service အသစ်တစ်ခု မထည့်ပါဘူး။ အစား **ရှိပြီးသား legitimate service တစ်ခုကို ခဏလောက် hijack လုပ်တာ** ပါ။
Windows service တစ်ခုစီမှာ
“ဒီ service start ဖြစ်ရင် ဘယ် executable ကို run မလဲ”
ဆိုတာကို registry ထဲက **ImagePath** value နဲ့ သတ်မှတ်ထားပါတယ်။

ဒီ **registry path** က—

```
HKLM\System\CurrentControlSet\Services\<service name>\ImagePath
```

ဒီ ImagePath ကို ပြောင်းလိုက်ရင်
service ကို start လုပ်တဲ့အချိန်
Windows က **အသစ်ပြောင်းထားတဲ့ executable ကို SYSTEM privilege နဲ့ run ပေးပါလိမ့်မယ်**။
ဒါကြောင့် SCShell ကို **even stealthier** လို့ ခေါ်တာပါ။
Indicator အနေနဲ့ မမြင်သာတဲ့ artifact တွေကိုသာ ချန်ထားပါတယ်။

Detection

ဒီ attack မှာ defender အတွက် အရေးကြီးဆုံး clue က
registry value modification ပါ။
အထူးသဖြင့်—

```
HKLM\System\CurrentControlSet\Services\<service name>\ImagePath
```

ဒီ value ကို

- **ဘယ်အချိန်မှာ ပြောင်းသလဲ**
- **ဘာတန်ဖိုးကနေ ဘာတန်ဖိုးကို ပြောင်းသလဲ**
- **ပြီးရင် မကြာခင် ပြန်ပြောင်းထားလား**

ဒီ sequence ကို စောင့်ကြည့်နိုင်ရင်
SCShell attack ကို ဖော်ထုတ်နိုင်ပါတယ်။

Defender Note

အထူးသဖြင့် suspicious ဖြစ်တာက
ImagePath ကို random executable (`C:\Windows\53we.exe` လို) နဲ့ ပြောင်းပြီး
စက္ကန့်အနည်းငယ်အတွင်း
မူလ path (`svchost.exe -k netsvcs` လို) ကို ပြန်ထားတာပါ။

ဒီလို “ပြောင်း → start → ပြန်ပြောင်း” pattern ကို
legitimate admin activity မှာ အရမ်းရှားပါတယ်။

ဒါကြောင့် author က
“ImagePath registry values ကို အမြဲစောင့်ကြည့်ထားတာက ကောင်းပါတယ်”
လို့ ပြောထားတာပါ။