Phishing Detection & Analysis

📌 Phishing Detection & Analysis – Structural Notes (Windows Logs + KQL)

1️⃣ Phishing Basics

အဓိပ္ပါယ် (Burmese)

Phishing ဆိုတာက တိုက်ခိုက်သူ (attacker) တွေက အသုံးပြုသူကို ယုံကြည်သွားအောင် လိမ်လည်ပြီး မလိုလားအပ်တဲ့ လုပ်ရပ်တွေကို လုပ်စေတဲ့ နည်းလမ်းတစ်ခုပါ။ ဥပမာအနေနဲ့ အီးမေးလ်၊ စာတို၊ သို့မဟုတ် ဖိုင်တွေကို တကယ့်အလုပ်လို၊ တကယ့်အဖွဲ့အစည်းလို လုပ်ပြီး ပို့လာတာမျိုးပါ။ အသုံးပြုသူက မသိဘဲ ဖိုင်ဖွင့်လိုက်ရင်၊ လင့်ခ်နှိပ်လိုက်ရင် malware ဝင်သွားနိုင်ပါတယ်။ ဒီနည်းလမ်းက အရမ်းရိုးရှင်းပေမယ့် အတွေ့အကြုံနည်းတဲ့ attacker ကနေ အတွေ့အကြုံမြင့်တဲ့ attacker အထိ အားလုံး အသုံးပြုနေဆဲဖြစ်ပြီး ယနေ့အချိန်ထိ အောင်မြင်နေဆဲဖြစ်ပါတယ်။ Security system တွေ၊ spam filter တွေ၊ user awareness training တွေ တိုးတက်လာပေမယ့် phishing ကတော့ network ထဲကို ဝင်ရောက်ဖို့ အယုံကြည်ရဆုံးနည်းလမ်းတစ်ခုအဖြစ် ရှိနေဆဲပါ။

ဒီ module ကို ပြီးဆုံးတဲ့အချိန်မှာ phishing attack တွေကို detect နဲ့ analyze လုပ်တဲ့အခါ ကြုံတွေ့ရတဲ့ အခက်အခဲတွေကို နားလည်လာပါလိမ့်မယ်။ ဥပမာအနေနဲ့ legitimate file နဲ့ malicious file ကို ဘယ်လို ခွဲခြားရမလဲ၊ Windows logs ထဲမှာ ဘယ် event တွေကို အထူးသတိထားကြည့်ရမလဲ ဆိုတာတွေကို သိလာပါလိမ့်မယ်။ အဲဒီအပြင် malware တစ်ခု run ဖြစ်ပြီးနောက် စက်ထဲမှာ ဘယ်လို behavior တွေ ဖြစ်လာတတ်လဲဆိုတာကိုလည်း လေ့လာရပါလိမ့်မယ်။ ဒါတွေက real malware samples တွေကို အခြေခံပြီး ရှင်းပြထားတာဖြစ်လို့ အလုပ်လုပ်နေတဲ့ environment မှာ တကယ့်အခြေအနေကို နားလည်အောင် ကူညီပေးနိုင်ပါတယ်။

2️⃣ Attacker Perspective

Attacker တစ်ယောက်က phishing attack တစ်ခု လုပ်တဲ့အခါ အရာရာကို လွတ်လပ်စွာ ရွေးချယ်လို့ မရပါဘူး။ သူတို့မှာ အချိန်ကန့်သတ်ချက် ရှိတတ်ပြီး၊ နည်းပညာကျွမ်းကျင်မှု အဆင့်လည်း ကန့်သတ်ချက် ဖြစ်နိုင်ပါတယ်။ ထို့အပြင် အသုံးပြုနိုင်တဲ့ resource တွေ၊ ပိုက်ဆံ၊ infrastructure တွေကလည်း မတူကြပါဘူး။ အရေးကြီးဆုံးကတော့ သူတို့ရဲ့ objective ဖြစ်ပါတယ်။ ဘာကို ရယူချင်လဲ၊ ဘယ်လောက်အထိ လျှို့ဝှက်ရမလဲ ဆိုတာပေါ်မူတည်ပြီး strategy ကို ရွေးချယ်ရပါတယ်။

🎯 Spear Phishing

ပစ်မှတ်အနည်းငယ်သာ ရှိတဲ့ attacker၊ အထူးသဖြင့် spear-phishing လုပ်တဲ့ attacker တွေဟာ ပစ်မှတ်ထားနိုင်တဲ့ လူအရေအတွက်ကနည်းတာကြောင့် ကန့်သတ်ချက်တွေ ရှိတတ်ပါတယ်။ တခါတရံမှာ သူတို့ရဲ့ objective က အလွန်အမင်း လျှို့ဝှက်ဖို့ လိုအပ်တာကြောင့် မထင်ရှားအောင် လုပ်ရပါမယ်။ ဒါကြောင့် သူတို့က အောင်မြင်နိုင်ချေကို မြှင့်တင်ဖို့အတွက် ပစ်မှတ်နဲ့ အရင် ယုံကြည်မှု တည်ဆောက်ဖို့ ကြိုးစားတတ်ပါတယ်။ ပို့တဲ့ message ထဲမှာလည်း ပစ်မှတ်နဲ့ ဆက်စပ်တဲ့ context တွေ ထည့်သွင်းပြီး သဘာဝကျအောင်

ပြုလုပ်ပါတယ်။ Infrastructure ပိုင်းမှာလည်း တစ်ခါတရံ ပစ်မှတ်အတွက် သီးသန့် domain၊ server တွေကို အသုံးပြုပြီး အရမ်းသန့်ရှင်းအောင် ပြင်ဆင်ထားတတ်ပါတယ်။

🌐 Mass Phishing

တစ်ဖက်မှာတော့ ပစ်မှတ်အများကြီးကို ရည်ရွယ်ထားတဲ့ attacker၊ အထူးသဖြင့် mass phishing လုပ်တဲ့သူတွေဟာ ငွေကြေးရရှိဖို့ ရည်ရွယ်ချက်ရှိတာများပါတယ်။ ဒါမှမဟုတ် အစိုးရအဖွဲ့အစည်းများ၊ ကုမ္ပဏီကြီးများလို ပစ်မှတ်အလားအလာ အရမ်းများတဲ့ အရာတွေကို ရည်ရွယ်ထားတာလည်း ဖြစ်နိုင်ပါတယ်။ ဒီလို attacker တွေက အောင်မြင်နိုင်ချေကို မြှင့်တင်ဖို့အတွက် ပစ်မှတ်စာရင်းကို အများဆုံးဖြန့်ကျက်ပါတယ်။ Payload ပိုင်းမှာတော့ အရမ်းပြည့်စုံစရာ မလိုဘဲ၊ ပုံမှန် antivirus နဲ့ email security control တွေကို ကျော်နိုင်လောက်ရင် “လုံလောက်ပြီ” လို့ သတ်မှတ်တတ်ပါတယ်။ Infrastructure ကိုလည်း security measure တွေ တိုးတက်လာတာနဲ့အမျှ အမြဲပြောင်းလဲနေပြီး domain၊ IP တွေကို မကြာခဏ လဲလှယ်တတ်ပါတယ်။

3️⃣ Mark-of-the-Web (MotW)

Background

၂၀၂၃ ခုနှစ်မတိုင်ခင်ကာလအထိ phishing attack တွေမှာ အများဆုံးအသုံးပြုခဲ့တဲ့ initial access နည်းလမ်းက macro phishing ဖြစ်ပါတယ်။ Macro phishing ဆိုတာက Microsoft Word၊ Excel စတဲ့ document ဖိုင်တွေထဲမှာ macro code တွေ ထည့်ပြီး victim ကို ဖွင့်အောင်လုပ်တာပါ။ User က “Enable Content” သို့မဟုတ် “Enable Macros” ကို နှိပ်လိုက်တာနဲ့ malware code က run သွားပါတယ်။ အဲ့ဒီအချိန်တုန်းက Microsoft က internet ကနေ download လုပ်လာတဲ့ macro ပါတဲ့ document တွေကို Protected View နဲ့ ဖွင့်ပေးတာပဲ လုပ်ခဲ့ပါတယ်။ Protected View က user ကို နောက်ထပ် click တစ်ချက် လိုအပ်စေတာကြောင့် attacker အတွက် မသင့်တော်ပေမယ့် လုံးဝ မဖြစ်နိုင်တော့တဲ့ အဆင့် မဟုတ်ခဲ့ပါဘူး။

ဒါပေမယ့် Microsoft က ၂၀၂၃ မတိုင်ခင်ကာလမှာ အရေးကြီးတဲ့ ဆုံးဖြတ်ချက်တစ်ခု ချခဲ့ပါတယ်။ Internet ကနေ လာတဲ့ document တွေအတွက် macro ကို default အနေနဲ့ disable လုပ်လိုက်တာပါ။ ဒီဆုံးဖြတ်ချက်ကြောင့် mass phishing လုပ်နေတဲ့ attacker တွေအတွက် macro phishing က တော်တော်လေး အလုပ်မဖြစ်တော့ပါဘူး။ ဒီအခြေအနေကြောင့် attacker တွေဟာ သူတို့ရဲ့နည်းလမ်းတွေကို ပြောင်းလဲအောင် လိုက်လျောညီထွေ ပြုလုပ်ရပါပြီ။

Motw

Mark-of-the-Web ဆိုတာက file တစ်ခုကို internet ကနေ download လုပ်လိုက်တဲ့အခါ Windows က အလိုအလျောက် ထည့်ပေးတဲ့ **alternate data stream** တစ်ခုပါ။ အထူးသဖြင့် **Zone.Identifier လို့ ခေါ်တဲ့ ADS ထဲမှာ ZoneId ဆိုတဲ့ value ပါပြီး ZoneId က 3 ဖြစ်နေရင် “ဒီဖိုင်က internet ကနေ လာတာ”** ဆိုတာကို Windows သိနိုင်ပါတယ်။ ဒီ attribute ရှိနေတဲ့ ဖိုင်တွေကို Windows Defender နဲ့ security product တွေက ပိုပြီး သေချာစစ်ဆေးပါတယ်။

အရင်ကတည်းက MotW ပါတဲ့ payload တွေဟာ MotW မပါတဲ့ payload တွေထက် အောင်မြင်နိုင်ချေနည်းခဲ့ပါတယ်။ ဒါပေမယ့် Microsoft က macro ကို ပိတ်လိုက်ပြီးနောက်မှာတော့ MotW ပါတဲ့ payload

တွေရဲ့ အောင်မြင်နိုင်ချေက အလွန်အမင်း ကျဆင်းသွားပါတယ်။ ဒီအခြေအနေကြောင့် mass phishing လုပ်နေတဲ့ attacker တွေဟာ နည်းလမ်းအသစ် နှစ်မျိုးကို အဓိက အသုံးပြုလာကြပါတယ်။

Marco Payload

နည်းလမ်းတစ်ခုကတော့ ရိုးရိုး macro payload ကိုပဲ အသုံးပြုပေမယ့် Mark-of-the-Web ကို ရှောင်လွှဲနိုင်တဲ့ delivery နည်းလမ်းနဲ့ ပို့တာပါ။ ဆိုလိုတာက file ကို internet download လုပ်ထားတဲ့အတိုင်း မမြင်ရအောင် ဖန်တီးပြီး Windows က MotW မထည့်အောင် လုပ်တာဖြစ်ပါတယ်။ နောက်တစ်နည်းလမ်းကတော့ macro ကို လုံးဝမသုံးတော့ဘဲ alternative payload တွေကို အသုံးပြုတာပါ။

ISO, VHD

MotW ကို ရှောင်လွှဲဖို့ အသုံးများတဲ့ နည်းလမ်းတွေထဲမှာ zip ဖိုင်တွေ ပါဝင်ပါတယ်။ Windows က zip ဖိုင်တွေကို MotW apply လုပ်တဲ့ပုံစံက မညီညာတဲ့အတွက် တချို့အခြေအနေတွေမှာ zip ထဲက file တွေက MotW မပါတော့ပါဘူး။ ISO ဖိုင်နဲ့ VHD ဖိုင်တွေကိုလည်း အသုံးများပါတယ်။ ဒီဖိုင်တွေက disk image အမျိုးအစားဖြစ်ပြီး Windows မှာ mount လုပ်လိုက်ရင် local disk တစ်ခုလို ဖြစ်သွားတဲ့အတွက် MotW က မထိရောက်တော့ပါဘူး။

LoLBaS

Macro အစား အသုံးများလာတဲ့ alternative payload တွေကတော့ ရိုးရိုး executable ဖိုင်တွေ ဖြစ်နိုင်သလို Windows ထဲမှာ အရင်ကတည်းက ပါလာတဲ့ legitimate tool တွေကို အသုံးချတဲ့ living-off-the-land binaries and scripts ဖြစ်နိုင်ပါတယ်။ ဒီ tool တွေကို LoLBaS သို့မဟုတ် LoLBins လို့ ခေါ်ကြပါတယ်။ Attacker တွေက ဒီ payload တွေကို MotW bypass နည်းလမ်းနဲ့ပဲ ပို့နိုင်သလို တချို့အခါ MotW ပါတဲ့အတိုင်းပဲ ပို့တာလည်း တွေ့ရပါတယ်။

OneNote

၂၀၂၃ ခုနှစ်အစောပိုင်းမှာ အလွန်အမင်း လူကြိုက်များလာတဲ့ နည်းလမ်းတစ်ခုကတော့ OneNote phishing ဖြစ်ပါတယ်။ OneNote ရဲ့ .one notebook ဖိုင်တွေက attachment တွေကို လက်ခံနိုင်ပြီး user က attachment ကို click လုပ်လိုက်တဲ့အခါ operating system က တိုက်ရိုက် execute လုပ်ပေးနိုင်ပါတယ်။ OneNote က Microsoft ရဲ့ တရားဝင် product ဖြစ်တဲ့အတွက် user တွေက သံသယမဖြစ်ဘဲ ဖွင့်တတ်ကြတာက attacker တွေအတွက် အခွင့်အလမ်းကြီးတစ်ခု ဖြစ်လာပါတယ်။

OneNote ဖိုင်တွေထဲမှာ ထည့်လေ့ရှိတဲ့ attachment တွေကတော့ Windows မှာ default handler ရှိပြီးသား executable အမျိုးအစားတွေဖြစ်ပါတယ်။ ဥပမာအားဖြင့် .cmd နဲ့ .bat ဖိုင်တွေကို cmd.exe က run ပေးပြီး .hta ဖိုင်ကို mshta.exe က run ပေးပါတယ်။ .js၊ .vbs၊ .wsf လို script ဖိုင်တွေကို wscript.exe က execute လုပ်ပေးပြီး .chm ဖိုင်ကို hh.exe က ဖွင့်ပေးပါတယ်။ ဒီလို legitimate Windows binary တွေကို အသုံးချထားတာကြောင့် security tool တွေအနေနဲ့ detect လုပ်ဖို့ ပိုပြီး ခက်ခဲလာပါတယ်။

Attacker Evasion

Some common attachments for OneNote files are:

File Extension	Executable
.cmd, .bat	cmd.exe
.hta	mshta.exe

|.wsf, .js, .vbs, .jse |wscript.exe|
|.chm |hh.exe |

4️⃣ Test : OneNote Attack Enumeration

How many OneNote phishing attacks did Microfocus experience?

```
external_table('Winlog')
| where ParentProcessName endswith "onenote.exe"
| summarize count() by ParentProcessName
```

5️⃣ OneNote Phishing (.one)

Why popular?

အသုံးပြုသူဘက်က ကြည့်ရင် OneNote phishing ဟာ macro phishing နဲ့ တော်တော်ဆင်တူပါတယ်။ နှစ်မျိုးလုံးမှာ document တစ်ခုကို သဘာဝကျကျ စီစဉ်ထားပြီး အသုံးပြုသူကို “တစ်ခုခု click လုပ်စေချင်တာ” ကို အဓိကထားပါတယ်။ Macro phishing မှာတော့ Word သို့မဟုတ် Excel document ကို ဖွင့်လိုက်တဲ့အခါ “Enable Content” သို့မဟုတ် “Enable Macros” ကို နှိပ်လိုက်ရင် VBA script က run သွားအောင် ပြုလုပ်ထားပါတယ်။ ဒီလို macro ပါတဲ့ document တွေဟာ internet ကနေ download လုပ်ထားရင် Mark-of-the-Web ပါနေတတ်ပြီး user ကို အပို click တစ်ချက် လိုအပ်အောင် လှုံ့ဆော်တတ်ပါတယ်။

OneNote phishing မှာလည်း စိတ်ကူးအတူတူပဲ ဖြစ်ပါတယ်။ OneNote notebook ဖိုင်ကို အသုံးပြုပြီး user ကို လှည့်စားကာ executable attachment ကို click လုပ်အောင် ပြုလုပ်ထားပါတယ်။ OneNote စာမျက်နှာထဲမှာ “Click here to view document”၊ “Double click to open” စတဲ့ စာသားတွေ၊ ပုံတွေ ထည့်ပြီး အလုပ်ဆိုင်ရာ ဖိုင်လို၊ invoice လို သဘောထားရအောင် ဖန်တီးထားတာများပါတယ်။ User က သံသယမဖြစ်ဘဲ attachment ကို နှိပ်လိုက်တာနဲ့ operating system က အဲ့ဒီ executable ကို run ပေးလိုက်ပါတယ်။

6️⃣ OneNote Detection Logic (Key Insight)

Defender Advantage

Marco Phishing :

defender ဘက်ကနေ ကြည့်ရင် macro phishing နဲ့ OneNote phishing ကြားက ကွာခြားချက်က အရမ်းကြီးပါတယ်။ Macro phishing မှာ VBA script က winword.exe process အတွင်းမှာ တိုက်ရိုက် run ဖြစ်သွားပါတယ်။ VBA ရဲ့ စွမ်းဆောင်ရည်က အရမ်းကျယ်ပြန့်တဲ့အတွက် attacker တွေက defense evasion နည်းလမ်း အမျိုးမျိုးကို အသုံးပြုနိုင်ပါတယ်။ File system ကို manipulate လုပ်တာ၊ registry ကို ပြောင်းတာ၊ memory ထဲမှာသာ payload ကို run လုပ်တာ စတဲ့ အရာတွေကို လုပ်နိုင်ပါတယ်။

ဒါကြောင့် macro phishing ကို detect လုပ်ဖို့ defender တွေအနေနဲ့ network traffic နဲ့ DNS data တွေကို အများကြီး အားကိုးရပါတယ်။ ဒါပေမယ့် ဒီ data တွေထဲမှာလည်း domain-fronting၊ အတု domain name တွေ၊ domain မသုံးဘဲ IP address တိုက်ရိုက်သုံးတာ စတဲ့ မသေချာမှုတွေ အများကြီး ပါဝင်နေပါတယ်။ ထို့အပြင် process အလိုက် network connection နဲ့ DNS lookup ကို log မလုပ်ထားရင် spear-phishing လို အလွန်စနစ်တကျ လုပ်ထားတဲ့ attack တွေကို နားမလည်နိုင်တော့ပါဘူး။ ဒီလိုအခြေအနေမှာ defender ဘက်က အားနည်းနေတတ်ပြီး attacker က အသာစီး ရနေတတ်ပါတယ်။

OneNote Phishing :

အပြန်အလှန်အားဖြင့် OneNote phishing ကတော့ defender အတွက် အများကြီး လွယ်ကူပါတယ်။ OneNote phishing payload တစ်ခုက အလုပ်စတင်ဖို့အတွက် မဖြစ်မနေ child process တစ်ခုကို spawn လုပ်ရပါတယ်။ ဆိုလိုတာက onenote.exe ကနေ cmd.exe၊ mshta.exe၊ wscript.exe စတဲ့ process တစ်ခုခုကို အသစ်ဖွင့်ရပါမယ်။ ဒီ behavior က အရမ်းထင်ရှားပြီး Windows logs မှာလည်း အလွယ်တကူ တွေ့နိုင်ပါတယ်။

ဒါကြောင့် defender တွေအနေနဲ့ OneNote process ကနေ child process တွေ ဘာတွေ ထွက်လာတတ်လဲဆိုတာကို baseline လုပ်ထားရုံနဲ့ မလုံလောက်တဲ့အရာတွေကို အလွယ်တကူ ဖော်ထုတ်နိုင်ပါတယ်။ Allow-list ထဲမှာ ပါသင့်တဲ့ OneNote child process တွေက အရမ်းနည်းပါးပါတယ်။ အမှန်တကယ် လုပ်ငန်းသုံး OneNote notebook တစ်ခုမှာ .hta၊ .wsf လို executable script ဖိုင်တွေကို embed လုပ်ပြီး ပို့လာတာက အလွန်ရှားပါးပါတယ်။

🔍 Detect OneNote Phishing – KQL `

- Find the Compromise User

```
external_table('Winlog')
| where EventCode == 4688
| where ParentProcessName endswith "onenote.exe"
| where NewProcessName has_any (
    "cmd.exe",
    "mshta.exe",
    "wscript.exe",
    "cscript.exe",
    "powershell.exe",
    "hh.exe"
)
| project Timestamp, SubjectUserName, NewProcessName, CommandLine
```

7️⃣ Re-Phished – Same User Hit Twice

Goal

`.one` payload နဲ့ user တစ်ယောက် **၂ ကြိမ် phished** ခံရတာကိုရှာ

```
external_table('Winlog')
| where EventCode == 4688
| where ParentProcessName endswith "onenote.exe"
| summarize count() by SubjectUserName
| where count_ >= 2
```

📌 **Result → Username (domain မပါ)**

8️⃣ Indicator Extraction from OneNote Payload

Scenario

- item.one click
- wscript.exe
- Invoke-WebRequest
- External IP download
- DLL execute via rundll32

🔍 Pivot from item.one – KQL

```
external_table('Winlog')
| where EventCode == 4688
| where CommandLine contains "item.one"
| project payloadTime = Timestamp, TargetUserName, payloadCommand =
CommandLine
| join kind=inner (
    external_table('Winlog')
    | where EventCode == 4688
    | project Timestamp, TargetUserName, ParentProcessName, CommandLine
) on TargetUserName
| where (payloadTime - Timestamp) between (-2s .. 2s)
```

ဒီမှာ ပေးထားတဲ့ KQL query က item.one ပါတဲ့ CommandLine ကို အခြေခံပြီး 4688 process creation event တွေကို ရှာပါတယ်။ ပထမပိုင်းမှာ item.one ကို execute လုပ်တဲ့အချိန်ကို payloadTime အနေနဲ့ သတ်မှတ်ပြီး user name နဲ့ command line ကို ထုတ်ထားပါတယ်။ နောက်ပိုင်းမှာတော့ တူညီတဲ့ user name နဲ့ ဖြစ်ပေါ် ခဲ့တဲ့ အခြား process creation event တွေကို join လုပ်ပြီး payloadTime ရဲ့ အရှေ့နောက် ၂ စက္ကန့်အတွင်း ဖြစ်ပေါ် ခဲ့တဲ့ event တွေကိုသာ ရွေးချယ်ထားပါတယ်။ ဒီလိုလုပ်ခြင်းအားဖြင့် item.one ကို click လုပ်တဲ့အချိန်နဲ့ ဆက်စပ်နေတဲ့ child process တွေကို တစ်စုတစ်စည်းတည်း မြင်နိုင်ပါတယ်။

ဒီ result ကို ကြည့်လိုက်ရင် attacker ရဲ့ တကယ့် malicious activity ကို ထင်ရှားစွာ မြင်ရပါတယ်။ wscript.exe ကနေ Invoke-WebRequest ကို အသုံးပြုပြီး external IP address တစ်ခုဖြစ်တဲ့ 165.22.160.25 ကနေ “160223” ဆိုတဲ့ file ကို download လုပ်ထားတာကို တွေ့ရပါတယ်။ အဲ့ဒီ file ကို %tmp% folder ထဲမှာ adeP1F.dll ဆိုတဲ့ နာမည်နဲ့ save လုပ်ထားပြီးနောက် rundll32 ကို အသုံးပြုပြီး execute လုပ်ထားပါတယ်။ ဒီအချက်တွေက network indicator၊ file indicator နဲ့ execution behavior ကို တစ်ပြိုင်နက်တည်း ပြသနေပါတယ်။

ဒီအခြေအနေမှာ IoC အနေနဲ့ အသုံးပြုနိုင်တဲ့ အရာတွေက IP address တစ်ခုတည်းမကပါဘူး။ Download လုပ်ထားတဲ့ adeP1F.dll ဆိုတဲ့ file name ကိုလည်း indicator အနေနဲ့ သုံးနိုင်ပါတယ်။ ထို့အပြင် အဲ့ဒီ file ရဲ့ hash ကို ရယူထားနိုင်ရင် incident scoping၊ log pivoting နဲ့ alert rule ဖန်တီးတဲ့အခါ အရမ်းအသုံးဝင်ပါတယ်။ ဒီလို indicator တွေကို စုထားခြင်းက အခြား system တွေမှာ အလားတူ attack ဖြစ်ပြီးသားလား၊ နောက်ထပ် infection ရှိလားဆိုတာကို စစ်ဆေးဖို့ အရေးကြီးပါတယ်။

နောက်ဆုံးမှာ attacker ရဲ့ process tree ကို ကြည့်လိုက်ရင် attack တစ်ခုလုံးရဲ့ လမ်းကြောင်းကို တစ်ချက်တည်းနဲ့ နားလည်နိုင်ပါတယ်။ OneNote က စပြီး script engine ကို ဖွင့်တယ်၊ script engine က network ကို ဆက်သွယ်တယ်၊ file ကို download လုပ်တယ်၊ နောက်ဆုံး DLL ကို execute လုပ်တယ် ဆိုတဲ့ chain က log တွေထဲမှာ အပြည့်အစုံ ထင်ရှားနေပါတယ်။ ဒီလို process tree ကို နားလည်ထားနိုင်ခြင်းက phishing incident တစ်ခုကို forensic အမြင်နဲ့ တိတိကျကျ ဖော်ထုတ်နိုင်ဖို့ အခြေခံအရေးကြီးဆုံး အချက်ဖြစ်ပါတယ်။

🧠 Extracted IoCs

- 🌐 IP address (e.g. `165.22.160.25`)
- 📁 File name (`adeP1F.dll`)
- 🧩 LOLBins (`wscript.exe`, `rundll32.exe`)

9️⃣ Process Tree Forensics (“Walk the Dog”)

Explanation

User တစ်ယောက်က .one document ကို ဖွင့်ပြီး attachment ကို ချက်ချင်း မနှိပ်ဘဲ နာရီဝက်၊ တစ်နာရီလောက် အကြာမှ click လုပ်တာလည်း ဖြစ်နိုင်ပါတယ်။ တချို့ attacker တွေကလည်း payload ထဲမှာ intentional delay ထည့်ထားပြီး ခဏစောင့်ပြီးမှ နောက်ထပ် stage ကို download လုပ်စေတတ်ပါတယ်။ ဒီလိုအခြေအနေတွေမှာ event တစ်ခုတည်းနဲ့ မဆုံးဖြတ်ဘဲ process ID ကို အခြေခံပြီး ဆက်တိုက် လိုက်ရှာရပါတယ်။ ဒါကို forensic သဘောထားနဲ့ “walk the dog” လို့ ခေါ်လေ့ရှိပြီး ProcessId နဲ့ NewProcessId ကို အစားထိုးသုံးပြီး search ကို ထပ်ထပ်လုပ်သွားတာ ဖြစ်ပါတယ်။

- Payload click ≠ immediate execution
- Delay / user hesitation possible
- Use **ProcessId** ↔ **NewProcessId** to walk tree

📌 DFIR mindset > alert-only mindset

🔟 ISO / VHD Phishing

Why dangerous?

Spear-phishing လို တိကျသေချာပြီး သေသပ်စွာ လုပ်ထားတဲ့ attack တွေမှာ threat intel တစ်ခုတည်းနဲ့ မလုံလောက်ပါဘူး။ rpruitt လို user name ကို indicator အဖြစ် မသုံးနိုင်ရင် ဒီ event တစ်ခုတည်းနဲ့ operational alert တစ်ခု တည်ဆောက်ဖို့ အရမ်းခက်ပါတယ်။

ဒီဥပမာမှာ attacker က .vhd ဖိုင်တစ်ခုကို အသုံးပြုထားပါတယ်။ အဲဒီ .vhd ထဲမှာ .lnk shortcut ဖိုင်တစ်ခု ပါပြီး၊ အဲဒီ .lnk က hidden folder တစ်ခုဖြစ်တဲ့ sandalwood ထဲက annotates.cmd ကို ညွှန်ပြထားပါ တယ်။ User က .vhd ကို double-click လုပ်လိုက်တာနဲ့ virtual drive တစ်ခုလို mount ဖြစ်သွားပြီး root folder ထဲက .lnk ကို click လုပ်တဲ့အခါ annotates.cmd ကို run လုပ်သွားပါတယ်။ Command line ကို ကြည့်လိုက်ရင် relative path ကို သုံးထားတာကြောင့် **4688 process creation event တစ်ခုတည်းနဲ့ "ဒါက .vhd ထဲကလာတာ" လို့ ဆက်စပ်ဖော်ထုတ်ရခက်ပါတယ်။** Analyst အနေနဲ့ "E: drive ထဲမှာ sandalwood\annotates.cmd ရှိနေတယ်" ဆိုတာကို သိရင် အများကြီး အထောက်အကူရမှာပါ။

Sysmon ကို အသုံးပြုထားတဲ့ environment မှာဆိုရင် Sysmon Event ID 1 က process creation အချိန်မှာ image path ကို ပိုမိုအသေးစိတ် ပေးနိုင်တာကြောင့် ဒီလို association ကို လုပ်ရ ပိုလွယ်ပါ တယ်။ ဒါက defense တစ်ခုကို တည်ဆောက်ဖို့ အလွန်ကောင်းတဲ့ အချက်ဖြစ်ပေမယ့် environment ကြီးမားလာရင်၊ unusual workload များလာရင် ဒီ field ကို အခြေခံတဲ့ detection တွေရဲ့ ယုံကြည်နိုင်မှုက လျော့ကျလာတတ်ပါတယ်။

ဒီနေရာမှာ အရေးကြီးတဲ့ အချက်တစ်ခု ထပ်ပေါ်လာပါတယ်။ Windows က virtual drive တစ်ခု mount လုပ်တဲ့အခါကို Microsoft-Windows-VHDMP-Operational log ထဲမှာ Event ID 12 အနေနဲ့ မှတ်တမ်း တင်ပေးပါတယ်။ ဆိုလိုတာက user တစ်ယောက်က .vhd သို့မဟုတ် .iso ဖိုင်ကို mount လုပ်လိုက်ပြီဆိုရင် process creation မတိုင်ခင်တောင် independent event တစ်ခုအနေနဲ့ log ထဲမှာ ကျန်နေပါတယ်။

ဒီ EID 12 event က အခြေအနေတော်တော်များများမှာ ကိုယ်တိုင်ရပ်တည်နိုင်တဲ့ indicator တစ်ခု ဖြစ်နိုင် ပါတယ်။ Environment ပေါ်မူတည်ပြီး Downloads, Documents လို folder တွေကနေ .vhd သို့မဟုတ် .iso mount လုပ်တာကို regex သုံးပြီး detect လုပ်နိုင်ပါတယ်။ ဒါမှမဟုတ် ဒီ event တွေကို အချိန်အလိုက် hunt လုပ်ပြီး file path နဲ့ file name ကို စုစည်းမယ်ဆိုရင် စိတ်ဝင်စားစရာ aggregation တွေကို ရနိုင်ပါ တယ်။

ဒါပေမယ့် ပြဿနာက EID 12 နဲ့ 4688 process creation event ကြားမှာ တိုက်ရိုက် link မရှိပါဘူး။ ဒီလို loosely linked event နှစ်ခုကို ချိတ်ဆက်ဖို့အတွက် အချိန်နဲ့ workstation ကို အခြေခံပြီး grouping လုပ်ရ ပါတယ်။ SIEM တချို့မှာ subsearch သုံးပြီး alert တစ်ခုကို သေသပ်စွာ တည်ဆောက်နိုင်ပေမယ့် အများ အားဖြင့် ဒီလို case တွေမှာ လူတစ်ယောက်က နောက်ဆုံး အမြင်နဲ့ စစ်ဆေးပေးရတာ မဖြစ်မနေ လိုအပ်ပါ တယ်။

Kusto (KQL) မှာ ဒီလို sequential event တွေကို ချိတ်ဖို့ join ကို အသုံးပြုပါတယ်။ အောက်က query က virtual drive mount (EID 12) ဖြစ်ပြီးနောက် ၃ စက္ကန့်အတွင်း ဖြစ်ပေါ်လာတဲ့ 4688 event တွေကို workstation အလိုက် ဆက်စပ်ပြပေးပါတယ်။ ဒီ result က ".vhd mount လုပ်ပြီး နောက်ဘာ process တွေ run ဖြစ်သွားလဲ" ဆိုတာကို တစ်နေရာတည်းမှာ မြင်စေပါတယ်။

Key Logs

Event	Meaning
4688	Process Creation
12 (VHDMP)	ISO / VHD mounted

🔍 Detect VHD / ISO Mount → Execution Chain

```
external_table('Winlog')
| where EventCode == 12
| project mountTime = Timestamp, HostName, mountMessage = Message
| join kind=inner (
    external_table('Winlog')
    | where EventCode == 4688
    | project Timestamp, HostName, TargetUserName, ParentProcessName,
CommandLine
) on HostName
| where (Timestamp - mountTime) between (0s .. 3s)
```

🧠 Final Takeaway (Important)

> **OneNote phishing is noisy at the process level**
> **ISO/VHD phishing is noisy at the mount level**
>
> Defender အနေနဲ့
> ❌ “Is this file malicious?” မမေးဘူး
> ✅ “Is this behavior normal?” ကို မေးရမယ်
