Investigating Alerts

Alert တစ်ခုရလာတဲ့အချိန် ဘယ်ကနေ စပြီး စုံစမ်းရမလဲ

Security Engineer၊ SOC Analyst တစ်ယောက်အနေနဲ့ SIEM ကနေ alert တစ်ခုထွက်လာတဲ့အခါ အရမ်းများသောအခါ “ဒီ alert က အန္တရာယ်ရှိလား၊ မရှိလား” ဆိုတာကို ချက်ချင်း မဆုံးဖြတ်နိုင်ပါဘူး။ Data တွေအများကြီး log တွေ၊ event တွေကို ရှာဖွေကြည့်နေရင်း အချိန်ကုန်သွားပေမယ့် နောက်ဆုံးမှာ network က တကယ် compromise ဖြစ်ထားလား၊ false positive လား ဆိုတာ မသေချာသေးတဲ့ အခြေအနေကို လူအများကြီး ကြုံဖူးကြပါတယ်။

ဒီစာမှာတော့ အဲ့ဒီလို အခြေအနေတွေကို ဘယ်လို ကိုင်တွယ်ရမလဲဆိုတာကို လက်တွေ့ဥပမာတွေဖြင့် ပြသပေးထားပါတယ်။ SIEM ထဲမှာ alert တစ်ခုတွေ့ရင် တစ်ချက်ကြည့်ပြီး ဆုံးဖြတ်တာမဟုတ်ဘဲ၊ ဘယ်လို အဆင့်ဆင့် စုံစမ်းသွားရမလဲ၊ ဘယ် log တွေကို ဆက်ကြည့်ရမလဲ၊ အချက်အလက်တွေကို ဘယ်လို ချိတ်ဆက်စဉ်းစားရမလဲ ဆိုတာကို “investigative process” အနေနဲ့ ရှင်းပြထားပါတယ်။ ဒါဟာ “ငါတို့က တာဝန်ယူပြီး စနစ်တကျ စုံစမ်းထားပါတယ်” ဆိုတဲ့ due diligence ကို ပြသနိုင်ဖို့အတွက်လည်း အရေးကြီးပါတယ်။ အချို့အဖွဲ့အစည်းတွေမှာ tool မစုံလင်တာ၊ log မပြည့်စုံတာတွေ ရှိနိုင်ပေမယ့် အဲ့ဒီအခြေအနေထဲမှာတောင် ဘယ်လို သဘောတရားနဲ့ စုံစမ်းရမလဲဆိုတာကို ဒီ module က သင်ပေးပါတယ်။

နောက်တစ်ခု အရေးကြီးတာက Windows processes တွေအကြောင်းကို အထူးအလေးထားပြီး ပြောထားတာပါ။ Windows system ထဲမှာ process တွေက ပုံမှန်အလုပ်လုပ်နေတဲ့ program တွေလည်း ဖြစ်နိုင်သလို၊ attacker တွေက အဲ့ဒီ process တွေကို အသုံးချပြီး malicious code တွေကို run စေတတ်ပါတယ်။ ဥပမာအားဖြင့် system မှာ ပုံမှန်ရှိနေတဲ့ process တစ်ခုကို ကြည့်လိုက်ရင် “ဒါက Windows ရဲ့ ပုံမှန် process ပဲ” လို့ ထင်ရနိုင်ပေမယ့် attacker က process injection လုပ်ထားတာ၊ parent-child relationship မမှန်တာ၊ unusual command line နဲ့ run နေတာမျိုးတွေ ဖြစ်နိုင်ပါတယ်။ ဒီလိုအရာတွေကြောင့် Windows processes တွေက analyst တွေအတွက် မသေချာမှု အများဆုံး ဖြစ်စေတဲ့ source တစ်ခု ဖြစ်လာပါတယ်။

Investigation Basics

Alert တစ်ခုရလာတဲ့အခါ ပထမဆုံးလုပ်ရမယ့်အရာက အချက်အလက်ကို ချက်ချင်း “ကောင်းတယ်၊ ဆိုးတယ်” လို့ မဆုံးဖြတ်ဘဲ **မေးခွန်းကြီးငါးခုကို စဉ်းစားခြင်းပါ။** ဒီမေးခွန်းတွေက who, what, when, where, why ဆိုတဲ့ မေးခွန်းတွေဖြစ်ပါတယ်။ အဓိပ္ပါယ်က suspicious ဖြစ်နိုင်တဲ့ action ကို

- **ဘယ်သူက လုပ်ခဲ့တာလဲ၊**
- **ဘာလုပ်တာလဲ၊**
- **ဘယ်အချိန်မှာ လုပ်တာလဲ၊**
- **ဘယ် system သို့မဟုတ် ဘယ်နေရာမှာ လုပ်တာလဲ၊**
- **အရေးကြီးဆုံးကတော့ အဲ့ဒီလုပ်ရပ်က ဘာကြောင့် suspicious ဖြစ်ရတာလဲ** ဆိုတာပါ။

ဒီမေးခွန်းတွေက investigation တစ်ခုရဲ့ အခြေခံအုတ်မြစ်ဖြစ်ပါတယ်။

What & Why

ဒီထဲမှာ what နဲ့ why ဆိုတဲ့ မေးခွန်းနှစ်ခုက အတွေ့အကြုံနဲ့ နည်းပညာပိုင်း နားလည်မှု အများဆုံးလိုအပ်ပါတယ်။ “What” ကို ဖြေဖို့ဆိုရင်

- alert ကို ဘယ်လို detection logic နဲ့ ထုတ်ထားလဲ၊
- ဘယ် artifact တွေက trigger ဖြစ်စေခဲ့လဲဆိုတာကို အောက်ခြေအဆင့်အထိ နားလည်ဖို့လိုပါတယ်။

Alert တစ်ခု true positive ဖြစ်တယ်ဆိုတာက “ဒီ event က detection rule နဲ့ ကိုက်ညီတယ်” လို့သာ အဓိပ္ပါယ်ရပါတယ်။ *ဒါက မဖြစ်မနေ **malicious** ဖြစ်နေတယ်ဆိုတာ မဟုတ်သေးပါဘူး။* Detection နဲ့ ကိုက်ညီတာကို သေချာအောင် စစ်ပြီးမှ နောက်တစ်ဆင့်အနေနဲ့ “ဒါက တကယ် အန္တရာယ်ရှိတဲ့ လုပ်ရပ်လား” ဆိုတာကို ဆက်စဉ်းစားရပါမယ်။

“Why” ကို ဖြေဖို့ဆိုရင် attacker တစ်ယောက်က ဘယ်လို လုပ်ရပ်တွေကို မကောင်းတဲ့ရည်ရွယ်ချက်နဲ့ လုပ်တတ်လဲဆိုတာကို အနည်းဆုံး သိထားရပါမယ်။ အန္တရာယ်ရှိတယ်လို့ ခေါ်တဲ့ action က အမှန်တကယ် ဘယ်လို ပုံစံနဲ့ ဖြစ်တတ်လဲ၊ attacker က ဘယ်အခြေအနေမှာ ဒီလိုလုပ်မလဲ ဆိုတာကို စဉ်းစားရပါတယ်။ ဒီလို စဉ်းစားခြင်းက threat model တစ်ခု တည်ဆောက်နေတာနဲ့ တူပါတယ်။

Example

ဥပမာအနေနဲ့ SAMR enumeration alert တစ်ခု ရလာတယ်လို့ ယူပါစို့။

> **SAMR** (**Security Account Manager Remote**) ဆိုတာ Windows domain ထဲမှာ user နဲ့ group အချက်အလက်တွေကို မေးမြန်းနိုင်တဲ့ protocol တစ်ခုပါ။

Attacker တွေက domain ထဲမှာ user တွေကို စူးစမ်းချင်တဲ့အချိန် **SAMR enumeration** ကို အသုံးပြုတတ်ပါတယ်။ Alert က EDR လို security product ကနေ လာရင် underlying artifact တွေကို analyst မမြင်ရနိုင်ပါဘူး။ ဒါပေမယ့် artifact တွေကို ရနိုင်တယ်ဆိုရင် SAMR connection တကယ်ဖြစ်ခဲ့လား၊ အဲ့ဒီ activity က enumeration ပုံစံနဲ့ ကိုက်ညီလားဆိုတာကို စစ်ဆေးရပါမယ်။

ဒီအချက်အလက်တွေကို အသုံးပြုပြီး “ဒီ alert မှာ ပြောထားတဲ့အရာက သဘာဝကျလား” ဆိုတဲ့ threat model ကို စဉ်းစားရပါတယ်။ Attacker တစ်ယောက်က ဘာကြောင့် SAMR enumeration လုပ်မလဲ၊ ပုံမှန်အားဖြင့် ဘယ်လိုနည်းလမ်းနဲ့ လုပ်မလဲ၊ Alert ထဲက အချက်အလက်တွေက SAMR enumeration အကြောင်း သိထားတဲ့အချက်တွေနဲ့ ကိုက်ညီလားဆိုတာတွေကို မေးရပါမယ်။

Close Investigation

အချို့အချိန်တွေမှာ ဒီ W မေးခွန်းတွေကို ပထမအကြိမ်ပဲ ဖြေကြည့်လိုက်တာနဲ့ investigation ကို ပိတ်လို့ရနိုင်ပါတယ်။ ဥပမာအားဖြင့် alert ထဲက activity က ပုံမှန် administrative activity တစ်ခုသာ ဖြစ်နေတယ်ဆိုရင် “ဒါက malicious မဟုတ်နိုင်ပါဘူး” လို့ due diligence ပြုပြီး incident ကို close လုပ်နိုင်ပါတယ်။ Threat model နဲ့ မကိုက်ညီတဲ့ activity ဆိုရင် အချိန်ထပ်မကုန်ဘဲ ရပ်တန့်သင့်ပါတယ်။

စာထဲမှာ ဥပမာပေးထားသလို Tier 0 account၊ အဓိပ္ပါယ်က Domain Admin လို အမြင့်ဆုံးအဆင့် account တစ်ခုက SAMR enumeration လုပ်နေတာဆိုရင် သီအိုရီအရ attacker ဖြစ်နိုင်ပေမယ့်

လက်တွေ့မှာတော့ ပုံမှန် admin အလုပ်လုပ်နေတာနဲ့ ပိုကိုက်ညီပါတယ်။ ဒီလိုဆုံးဖြတ်ချက်တွေက

- **environment ကို ဘယ်လောက်နားလည်လဲ၊**
- **threat ကို ဘယ်လောက်သိလဲ၊**
- **alert ရဲ့ context ကို ဘယ်လောက် ဖတ်နိုင်လဲ**ဆိုတဲ့အပေါ် မူတည်ပါတယ်။

Investigation ဆက်လုပ်သွားတဲ့အခါ အသစ်ရလာတဲ့ အချက်အလက်တိုင်းက ကိုယ့်ရဲ့ သီအိုရီကို အားပေးနိုင်သလို ဖျက်သိမ်းနိုင်ပါတယ်။ အဓိကရည်ရွယ်ချက်က **incident ကို escalate လုပ်ရမလား၊ de-escalate လုပ်ရမလား ဆိုတာကို အမြန်ဆုံး ဆုံးဖြတ်နိုင်ဖို့ပါ။** ဒါကြောင့် အချိန်အနည်းဆုံးနဲ့ ဆုံးဖြတ်ချက်ရစေမယ့် အချက်အလက်တွေကို ဦးစားပေး စုဆောင်းရပါမယ်။ ဥပမာအားဖြင့် ဒီ user သို့မဟုတ် ဒီ process က အရင်က SAMR enumeration လုပ်ဖူးလား၊ ဒီလိုအလုပ်လုပ်တတ်တဲ့ အခြား user တွေ ရှိလားဆိုတာတွေကို စစ်ကြည့်နိုင်ပါတယ်။

Can't Decide

တချို့အချိန်တွေမှာ တစ်ကြိမ်၊ နှစ်ကြိမ်လောက် စုံစမ်းပြီးတောင် ခိုင်မာတဲ့ ဆုံးဖြတ်ချက် မရနိုင်ပါဘူး။ ဒီလိုဖြစ်ရင် **“ဘာကြောင့် မဆုံးဖြတ်နိုင်တာလဲ”** ဆိုတာကို မှတ်တမ်းတင်ထားဖို့ အရေးကြီးပါတယ်။ အများဆုံးကြုံရတဲ့ အကြောင်းရင်းတွေက

- telemetry မရှိတာ၊
- log မသိမ်းထားတာ၊
- enrichment မကောင်းတာ၊
- analysis လုပ်ဖို့ tool မလုံလောက်တာတွေပါ။

> ဒီလို အခြေအနေတွေကို မှတ်တမ်းတင်ထားခြင်းက နောက်ပိုင်း telemetry ပိုကောင်းအောင် တိုးချဲ့ဖို့ အကြောင်းပြချက်ကောင်း ဖြစ်လာပါတယ်။ Telemetry ကောင်းရင် ဆုံးဖြတ်ချက်လည်း မြန်လာပါတယ်။

Conclusion

နောက်ဆုံးအနေနဲ့ investigation ဆိုတာ network event တွေက အမြဲပြောင်းလဲနေတဲ့အတွက် rigid step-by-step process တစ်ခုကို အမြဲတမ်း မသုံးသင့်ပါဘူး။ ဒီလို flexibility ကြောင့် analyst တွေအတွက် မသေချာမှုနဲ့ စိတ်ပူမှုတွေ ဖြစ်လာနိုင်ပါတယ်။ အဲ့ဒီကို ဖြေရှင်းနိုင်တဲ့ နည်းက due diligence ကို ပြသနိုင်အောင်

- alert ကို သေချာနားလည်ခြင်း၊
- true positive ဖြစ်မဖြစ် စစ်ဆေးခြင်း၊
- threat model နဲ့ ကိုက်ညီတဲ့ သီအိုရီတွေ တည်ဆောက်ခြင်း၊
- အဲ့ဒီသီအိုရီတွေကို အတည်ပြု သို့မဟုတ် ဖျက်သိမ်းဖို့ အချက်အလက်တွေ စုဆောင်းခြင်း၊
- နောက်ဆုံးမှာ ကိုယ့်ရဲ့ တွေ့ရှိချက်တွေကို မှတ်တမ်းတင်ခြင်း ဖြစ်ပါတယ်။
  ဒီလိုလုပ်နိုင်ရင် investigation တစ်ခုကို တာဝန်ယူပြီး စနစ်တကျ လုပ်ခဲ့တယ်ဆိုတာကို ပြသနိုင်ပါတယ်။

SAMR Alert - Understand

အရင်ဆုံး EDR (**Endpoint Detection and Response** system) ကနေ alert တစ်ခု ရလာပါတယ်။

- Title: SAMR Enumeration
- Time: 2024-01-30T01:38:32.728Z
- Host: DC
- Source User: WIN11-20$
- Source Host: WIN11-20
- Description: WIN11-20$ enumerated a large number of domain users with the SAMR protocol.

Source host က WIN11-20 workstation ဖြစ်ပြီး အဲ့ဒီ computer account က Domain Controller ကို SAMR protocol သုံးပြီး domain user အများကြီးကို enumerate လုပ်ခဲ့တယ်လို့ description မှာ ဖော်ပြထားပါတယ်။

ဒီအချက်အလက်တွေကို ကြည့်လိုက်ရင် who, when, where ဆိုတဲ့ မေးခွန်းတွေကို အဖြေတချို့ ရပြီးသားပါ။

- ဘယ်သူက လုပ်တာလဲဆိုရင် WIN11-20$ computer account ဖြစ်ပြီး၊
- ဘယ်အချိန်မှာလဲဆိုရင် သတ်မှတ်ထားတဲ့ timestamp မှာ ဖြစ်ပါတယ်။
- ဘယ်နေရာမှာလဲဆိုရင် WIN11-20 workstation ကနေ DC ကို ချိတ်ဆက်ပြီး လုပ်ထားတာဖြစ်ပါတယ်။

ဒါပေမယ့် analyst အနေနဲ့ အရေးကြီးဆုံးဖြစ်တဲ့ **"ဘာလုပ်တာလဲ"** နဲ့ **"ဘာကြောင့် suspicious ဖြစ်တာလဲ"** ဆိုတဲ့ what နဲ့ why ကိုတော့ အပြည့်အဝ နားမလည်သေးပါဘူး။

SAMR

ဒီအဆင့်မှာ investigation ကို ဆက်လုပ်ဖို့ဆိုရင် SAMR ဆိုတာ ဘာလဲ၊ ဘာအတွက် အသုံးပြုတာလဲဆိုတာကို အရင်ဆုံး နားလည်ရပါမယ်။ SAMR ဆိုတာ Security Account Manager Remote protocol ဖြစ်ပြီး Windows domain environment ထဲမှာ user, group နဲ့ computer account တွေကို remote ကနေ စီမံခန့်ခွဲဖို့ အသုံးပြုတဲ့ protocol ဖြစ်ပါတယ်။ System administrator တွေက

- domain user list ကို ကြည့်တာ၊
- group membership ကို ပြောင်းတာ၊
- user attribute တွေကို ပြင်တာ၊
- password ပြောင်းတာတွေကို SAMR သုံးပြီး လုပ်တတ်ကြပါတယ်။

ဒါကြောင့် SAMR ကို သုံးတယ်ဆိုတာကိုသာ ကြည့်ပြီး ချက်ချင်း "အန္တရာယ်ရှိတယ်" လို့ မဆုံးဖြတ်နိုင်ပါဘူး။

ဒါပေမယ့် ဒီ alert ထဲမှာ အရေးကြီးတဲ့ စကားလုံးတစ်ခု ရှိပါတယ်။ အဲ့ဒါက **“enumerated a large number of domain users”** ဆိုတာပါ။ အဓိပ္ပါယ်က ပုံမှန်ထက် များစွာသော user တွေကို enumerate လုပ်ခဲ့တယ်ဆိုတာကို EDR က သတိပေးနေတာပါ။ Telemetry က opaque ဖြစ်လို့ analyst က underlying detail အားလုံးကို မမြင်ရပေမယ့် anomaly ဖြစ်တယ်ဆိုတာကိုတော့ alert က ပြောပြနေပါတယ်။

ဒီအချက်အလက်ပေါ် အခြေခံပြီး နောက်တစ်ဆင့်အနေနဲ့ research လုပ်လိုက်တဲ့အခါ SAMR ကို SharpHound, Impacket, Nmap လို popular reconnaissance tools တွေက အသုံးပြုတယ်ဆိုတာကို တွေ့ရပါတယ်။ SharpHound က BloodHound အတွက် data စုဖို့ သုံးတဲ့ tool ဖြစ်ပြီး domain structure နဲ့ privilege relationship တွေကို ရှာဖွေရာမှာ အသုံးများပါတယ်။ Impacket နဲ့ Nmap လည်း attacker တွေက network နဲ့ domain ကို စူးစမ်းဖို့ အသုံးချတတ်ကြတဲ့ tool တွေဖြစ်ပါတယ်။ ဒီလို tool တွေက SAMR ကို သုံးပြီး domain user, group အချက်အလက်တွေကို ရယူတတ်ကြပါတယ်။

ဒီအချိန်မှာ analyst က ကိုယ့် environment ကို ပြန်စဉ်းစားရပါမယ်။ Win11-20 ဆိုတာ workstation တစ်လုံးဖြစ်ပြီး Domain Controller မဟုတ်ပါဘူး။ ပုံမှန်အားဖြင့် workstation တစ်လုံးက computer account အသုံးပြုပြီး domain user အများကြီးကို enumerate လုပ်နေတာက သဘာဝမကျပါဘူး။ ဒီလို အချက်တွေကို ချိတ်ဆက်စဉ်းစားလိုက်တဲ့အခါ “ဒါက attacker behavior ဖြစ်နိုင်တယ်” ဆိုတဲ့ threat model တစ်ခုကို တည်ဆောက်နိုင်ပါတယ်။

Conclusion

ဒီ threat model အပေါ် အခြေခံပြီး ပထမဆုံး theory တစ်ခုကို analyst က ဆွဲထုတ်နိုင်ပါတယ်။ အဲ့ဒီ theory က Win11-20 workstation ကို attacker က compromise လုပ်ထားပြီး computer account WIN11-20$ ကို အသုံးပြုပြီး domain reconnaissance လုပ်နေတယ်ဆိုတာပါ။ ရည်ရွယ်ချက်က domain အတွင်းမှာ ဘယ် user တွေရှိလဲ၊ ဘယ် account တွေကို target လုပ်နိုင်လဲဆိုတာ သိဖို့ဖြစ်ပြီး နောက်တစ်ဆင့် privilege escalation လုပ်ဖို့ ပြင်ဆင်နေတာဖြစ်နိုင်ပါတယ်။

SAMR Alert - Validate

ပထမဆုံး နားလည်ရမယ့်အချက်က **ဒီ alert မှာ individual SAMR operation တစ်ခုချင်းစီကို analyst မမြင်ရဘူး**ဆိုတာပါ။ အဓိပ္ပါယ်က user ကို ဘယ်လို sequence နဲ့ enumerate လုပ်ခဲ့လဲ၊ user ဘယ်လောက်ကို query လုပ်ခဲ့လဲဆိုတဲ့ အသေးစိတ် telemetry ကို မရရှိနိုင်ပါဘူး။ ဒါကြောင့် alert ကို “ပြည့်ပြည့်စုံစုံ validate” လုပ်ဖို့ ခက်ခဲပါတယ်။ အကယ်၍ EDR က artifact တွေ ပေးနိုင်တယ်ဆိုရင် enumerated user အရေအတွက်ကို စစ်ပြီး တကယ် enumeration ပုံစံလား၊ privileged group ကိုပဲ ရွေးပြီး query လုပ်ထားတာလားဆိုတာကို စစ်နိုင်ပါမယ်။ ဒါပေမယ့် အဲ့ဒီလိုမရတဲ့အခြေအနေမှာ **host-level artifact** တွေကို အားကိုးရပါမယ်။

SAMR Message Transport

ဒီနေရာမှာ SAMR ဘယ်လို communicate လုပ်လဲဆိုတာ နားလည်ရပါမယ်။ Client တစ်ခုက SAMR ကို သုံးချင်ရင် RPC ကို အသုံးပြုပြီး ဆက်သွယ်ပါတယ်။ အဲ့ဒီ RPC က SMB ပေါ်ကနေလည်း ဖြစ်နိုင်သလို TCP ပေါ်ကနေလည်း ဖြစ်နိုင်ပါတယ်။

> RPC over SMB ဖြစ်ရင် port 445 ကို အသုံးပြုပါတယ်။

> RPC over TCP ဖြစ်ရင် အရင်ဆုံး port 135 ကို သုံးပြီး နောက်မှာ dynamic port တစ်ခုကို ချိတ်ဆက်ပါတယ်။

Dynamic port ဆိုတာ 49152 ကနေ 65535 အတွင်းရှိတဲ့ port တွေပါ။

Lab မှာ Impacket သို့မဟုတ် SharpHound ကို အသုံးပြုပြီး စမ်းကြည့်ရင် ဒီ tool တွေက RPC over SMB ကို သုံးတာကို တွေ့ရပါလိမ့်မယ်။ ဒါကြောင့် attacker က ဒီ tool တွေကို အသုံးပြုခဲ့တယ်ဆိုရင် source host က Domain Controller ကို port 445 နဲ့ ဆက်သွယ်ထားတဲ့ artifact တွေကို တွေ့ရမယ်လို့ ခန့်မှန်းနိုင်ပါတယ်။ ဒါဟာ threat model နဲ့ ကိုက်ညီတဲ့ validation logic ဖြစ်ပါတယ်။

ဒီ activity ကို source host ဖြစ်တဲ့ Win11-20 မှာ isolate လုပ်ဖို့ Windows event နဲ့ Sysmon event တွေကို အသုံးပြုပါတယ်။

> Windows Event ID 5145 က Detailed File Share access event ဖြစ်ပြီး SMB pipe access တွေကို ပြသပေးပါတယ်။ SAMR ကို သုံးရင် \pipe\samr ကို access လုပ်ရပါတယ်။

> Sysmon Event ID 3 က NetworkConnect event ဖြစ်ပြီး network connection အကြောင်းကို ပြသပေးပါတယ်။ Port 445 ကို ချိတ်ဆက်ထားတာကို ဒီ event နဲ့ တွေ့နိုင်ပါတယ်။

အဲ့ဒီ logic ကို အခြေခံပြီး Kusto query တစ်ခုကို သုံးပါတယ်။ ဒီ query က alert ထဲမှာ ဖော်ပြထားတဲ့ အချိန်အနီးအနားမှာ \pipe\samr ကို access လုပ်ထားတဲ့ event တွေကို ရှာပြီး၊ အဲ့ဒီ event နဲ့ အချိန်အနည်းငယ်အတွင်း Domain Controller ကို port 445 နဲ့ ဆက်သွယ်ထားတဲ့ network connection event ကို join လုပ်ပါတယ်။ Join လုပ်ခြင်းရဲ့ အကျိုးကျေးဇူးက event နှစ်ခုကို တစ်ခုတည်းလို မြင်ရပြီး source IP၊ timing နဲ့ behavior ကို တစ်ပြိုင်နက် နားလည်နိုင်တာပါ။

```
external_table('Winlog')
| where Timestamp between (datetime(2024-01-30T01:38:32.000Z) ..
datetime(2024-01-30T01:38:34.000Z))
  and EventCode == 5145
  and HostName contains "dc"
  and RelativeTargetName == "samr"
| project Start = Timestamp, HostName, EventCode, RelativeTargetName
| join kind=inner (
    external_table('Winlog')
    | where EventCode == 3 and DestinationPort == 445
    | project End = Timestamp, HostName, EventCode, DestinationPort,
SourceIp
) on HostName
| where (End - Start) between (0sec .. 2sec)
```

ဒီ query ကို ကြည့်လိုက်ရင် အစပိုင်းမှာ Domain Controller ပေါ်က SAMR pipe ကို access လုပ်ထားတဲ့ event ကို ရှာပါတယ်။ အချိန်က alert ထဲမှာပါတဲ့ timestamp အနီးအနား ဖြစ်ရပါမယ်။ နောက်ပိုင်းမှာ port 445 ကို ချိတ်ဆက်ထားတဲ့ network connection event တွေကို ရှာပြီး HostName ကို အခြေခံပြီး join လုပ်ပါတယ်။ နောက်ဆုံးမှာ event နှစ်ခုရဲ့ အချိန်က ၂ စက္ကန့်အတွင်း ဖြစ်ရပါမယ်ဆိုတဲ့ condition ကို

ထည့်ထားပါတယ်။ အဓိပ္ပါယ်က SAMR pipe ကို access လုပ်တဲ့အချိန်နဲ့ SMB connection ဖြစ်တဲ့အချိန်က နီးစပ်ရမယ်ဆိုတာပါ။

ဒီ query ကို မြင်ရင် ကြောက်စရာကြီးလို ထင်နိုင်ပေမယ့် အမှန်တကယ်တော့ SIEM investigation မှာ အရမ်းအသုံးများတဲ့ pattern ပါ။ Suspicious event တစ်ခုကို ရှာပြီး အဲ့ဒီ event နဲ့ အချိန်အနည်းငယ် အတွင်း ဖြစ်ပွားတဲ့ network activity ကို ဆက်စပ်ကြည့်နေတာပဲ ဖြစ်ပါတယ်။

Start	Hosti	Evt	RelativeTargetNan		End	HostName1	Eventl	DestinationPort	Image
2024-01-30 01:38:32.728	dc	5145	samr	1	2024-01-30 01:38:33.666	Win11-20	3	445	System
2024-01-30 01:38:32.728	dc	5145	samr	1	2024-01-30 01:38:33.666	Win11-20	3	445	C:\Windows\System32\winlogon.exe
2024-01-30 01:38:32.728	dc	5145	samr	1	2024-01-30 01:38:33.666	Win11-20	3	445	C:\Windows\System32\winlogon.exe

အရေးကြီးတဲ့အချက်က validation က theoretical knowledge တစ်ခုတည်းနဲ့ မလုံလောက်ဘူးဆိုတာပါ။ Attack က လက်တွေ့မှာ ဘယ်လိုပုံစံနဲ့ ဖြစ်လဲဆိုတာကို သိထားရင် ဘာ artifact တွေ ပေါ်လာမလဲဆိုတာကို ခန့်မှန်းနိုင်ပါတယ်။ Research က အထောက်အကူဖြစ်ပေမယ့် lab မှာ SharpHound နဲ့ Impacket ကို ကိုယ်တိုင် run ကြည့်ဖူးတာက ပိုပြီး အထောက်အကူဖြစ်ပါတယ်။ ဒီ case မှာလည်း theory အတိုင်း Win11-20$ က SMB ကို သုံးပြီး \pipe\samr ကို access လုပ်ထားတာကို တွေ့ရပါတယ်။

Test : SAMR Alert - Gather and Analyze

Which additional events on DC support our assessment that an attacker performed reconnaissance with the Win11-20$ account?

အရင်ဆုံး စဉ်းစားရမယ့်အချက်က validation အဆင့်မှာ ကျွန်တော်တို့ သိလာပြီးသားအရာပါ။ Win11-20$ workstation account က Domain Controller ကို SMB နဲ့ ချိတ်ဆက်ပြီး \pipe\samr ကို access လုပ်ထားတယ်ဆိုတာကို network နဲ့ file share artifact တွေကနေ အတည်ပြုနိုင်ခဲ့ပါတယ်။ ဒါပေမယ့် reconnaissance လုပ်တယ်ဆိုတဲ့ သီအိုရီကို ပိုမိုခိုင်မာအောင် လုပ်ချင်ရင် **“SAMR ကို access လုပ်တဲ့ အချိန်အနီးအနားမှာ DC ပေါ်မှာ ဘာတွေ ထပ်ဖြစ်သွားသေးလဲ”** ဆိုတာကို ကြည့်ရပါမယ်။ ဒီလို context ကို ကြည့်တာပဲ Gather and Analyze အဆင့်ရဲ့ အဓိကအလုပ်ပါ။

ဒီနေရာမှာ analyst က 5145 event တစ်ခုတည်းကိုပဲ မကြည့်တော့ပါဘူး။ အဲ့ဒီ event ဖြစ်တဲ့အချိန် အနီးအနားမှာ Domain Controller ပေါ်မှာ ဖြစ်ပေါ် ခဲ့တဲ့ event အမျိုးမျိုးကို ဆက်စပ်ကြည့်ပါတယ်။ အကြောင်းရင်းက attacker ရဲ့ reconnaissance ဟာ အလုပ်တစ်ခုပဲ မလုပ်ဘဲ ဆက်တိုက်လုပ်တတ်လို့ပါ။ User enumeration ပြီးရင် privileged group တွေကိုပါ စူးစမ်းတတ်ပါတယ်။

စာထဲမှာ ပြထားတဲ့ query က ပထမအပိုင်းမှာ SAMR connection ကို အရင် isolate လုပ်ထားပါတယ်။ အရင်တုန်းက query နဲ့ ဆင်တူပြီး alert timestamp အနီးအနားမှာ Domain Controller ပေါ်က \pipe\samr ကို access လုပ်တဲ့ 5145 event နဲ့ port 445 ကို ချိတ်ဆက်ထားတဲ့ network connection event ကို ဆက်စပ်ထားပါတယ်။ ဒီအပိုင်းကို variable တစ်ခုထဲမှာ သိမ်းထားတာဖြစ်ပြီး analyst အနေနဲ့ အဲ့ဒီအပိုင်းကို နည်းနည်းကြည့်ရင် ရပါတယ်။

အရေးကြီးတာက ဒုတိယအပိုင်းပါ။ ဒုတိယအပိုင်းမှာ အဲ့ဒီ SAMR connection ဖြစ်တဲ့ HostName ကို အခြေခံပြီး Domain Controller ပေါ်မှာ ဖြစ်ခဲ့တဲ့ event အမျိုးမျိုးကို ထပ် join လုပ်ပါတယ်။ EventCode ကန့်သတ်မထားဘဲ Timestamp, EventAction, SubjectUserName, TargetUserName လို context အချက်အလက်တွေကို ယူပါတယ်။ နောက်ဆုံးမှာ SAMR connection ဖြစ်တဲ့အချိန်နဲ့ အခြား event တွေ ဖြစ်တဲ့အချိန်က နီးစပ်နေရပါမယ်ဆိုတဲ့ condition ကို ထည့်ထားပါတယ်။ အဓိပ္ပါယ်က “SAMR access လုပ်နေတဲ့ အချိန်အနီးမှာ DC ပေါ်မှာ ဘာတွေ ဖြစ်သွားသေးလဲ” ဆိုတာကို ကြည့်ချင်တာပါ။

```
let samr_connections = external_table('Winlog')
| where Timestamp between (datetime(2024-01-30T01:38:32.000Z) ..
datetime(2024-01-30T01:38:34.000Z))
  and EventCode == 5145
  and HostName contains "dc"
  and RelativeTargetName == "samr"
| extend key = 1
| project Start = Timestamp, HostName, EventCode, RelativeTargetName, key
| join kind=inner (
    external_table('Winlog')
    | where EventCode == 3
      and DestinationPort == 445
      and HostName != "dc"
    | extend key = 1
    | project End = Timestamp, HostName, EventCode, DestinationPort, Image,
key
) on key
| where (End - Start) between (0sec .. 2sec)
| take 1;

samr_connections
| join kind=inner (
    external_table('Winlog')
    | project CtxTimestamp = Timestamp, HostName, EventCode, EventAction,
SubjectUserName, TargetUserName
) on HostName
| where (End - CtxTimestamp) between (0sec .. 2sec)
```

ဒီ query ကို အဓိပ္ပါယ်ဖွင့်ပြရရင် analyst က “SAMR access ဖြစ်တဲ့ အချိန်နဲ့ တစ်စက္ကန့် နှစ်စက္ကန့်အတွင်း Domain Controller ပေါ်မှာ ဘယ် action တွေ ထပ်ဖြစ်ခဲ့လဲ” ဆိုတာကို မြင်ချင်တာပါ။ အဲ့ဒီ အချက်အလက်တွေကို စုစည်းပြီးကြည့်လိုက်တဲ့အခါ Win11-20$ account က DC ပေါ်မှာ privileged local group တွေကိုပါ enumerate လုပ်ထားတာကို တွေ့ရပါတယ်။

ဒီတွေ့ရှိချက်က အရမ်းအရေးကြီးပါတယ်။ ပုံမှန် administrative activity အများစုက user list တစ်ခုပဲ ကြည့်ပြီး ရပ်တတ်ပါတယ်။ ဒါပေမယ့် attacker တွေက privilege escalation လုပ်ဖို့ စဉ်းစားနေတဲ့အချိန် domain admin group၊ enterprise admin group လို privileged group တွေကို အရင်ဆုံး စူးစမ်းတတ်ပါတယ်။ Win11-20$ workstation account က privileged local group တွေကို enumerate လုပ်နေတာက “ဒီဟာ reconnaissance အတွက် လုပ်နေတယ်” ဆိုတဲ့ threat model နဲ့ အလွန်ကိုက်ညီပါတယ်။

ဒါကြောင့် Gather and Analyze အဆင့်မှာ analyst က သီအိုရီကို အတည်ပြုနိုင်တဲ့ သက်သေကို ထပ်ရလာပါတယ်။ SAMR ကို သုံးပြီး user enumeration လုပ်ထားတာသာမက privileged group enumeration ကိုပါ ဆက်လုပ်ထားတာကြောင့် “Win11-20$ account ကို attacker က compromise လုပ်ပြီး domain reconnaissance လုပ်နေတယ်” ဆိုတဲ့ assessment က ပိုမိုခိုင်မာလာပါတယ်။ ဒီအဆင့်အထိ ရောက်လာတဲ့အခါ incident ကို escalate လုပ်သင့်တယ်ဆိုတဲ့ ဆုံးဖြတ်ချက်ကို ယုံကြည်စိတ်ချစွာ ချနိုင်လာပါတယ်။

SAMR Alert - Strengthening Our Assessment

Since winlogon.exe enumerating users and groups like this is anomalous, we would be justified in escalating our investigation. In other words, you have everything you need to move to the next phase or tier of your organizations incident response process. If you don't have the experience to make this determination, you can demonstrate that the activity is sufficiently anomalous with more information gathering and analysis. We can ask the questions:

- Do any other instances of winlogon.exe enumerate users and groups using SAMR?
- Do similar users enumerate users and groups using SAMR?

Which user other than WIN11-20$ enumerated users with the SAMR protocol?

ဒီအချိန်အထိ ကျွန်တော်တို့ သိလာတဲ့အချက်တွေကို ပြန်ချုပ်ကြည့်ရင် Win11-20$ workstation account က Domain Controller ပေါ်မှာ SAMR ကို သုံးပြီး user တွေကို enumerate လုပ်ထားတယ်၊ privileged group တွေကိုပါ ဆက်ပြီး enumerate လုပ်ထားတယ်၊ အဲ့ဒီ activity တွေကို လုပ်နေတဲ့ process က winlogon.exe ဖြစ်တယ်ဆိုတာကို တွေ့ထားပြီးပါပြီ။ winlogon.exe ဆိုတာ Windows login process ဖြစ်ပြီး ပုံမှန်အားဖြင့် user login, session management လို အလုပ်တွေကိုပဲ လုပ်ရမယ့် process ပါ။ Domain user နဲ့ group တွေကို အစုလိုက် enumerate လုပ်နေတာက ဒီ process အတွက် သဘာဝမကျတဲ့ behavior ဖြစ်ပါတယ်။

ဒါကြောင့် “winlogon.exe က SAMR သုံးပြီး ဒီလို enumeration လုပ်နေတာက anomalous ဖြစ်တယ်” ဆိုတဲ့ အချက်အပေါ် အခြေခံပြီး investigation ကို escalate လုပ်ဖို့ သင့်တော်ပြီလို့ ဆုံးဖြတ်နိုင်ပါတယ်။ Escalate လုပ်တယ်ဆိုတာ SOC Tier 2 သို့မဟုတ် Incident Response team ကို လွှဲပေးတာ၊ containment နဲ့ response phase ကို စတင်တာမျိုးတွေကို ဆိုလိုပါတယ်။ Analyst အနေနဲ့ ဒီ ဆုံးဖြတ်ချက်ကို ချနိုင်ဖို့ အတွေ့အကြုံရှိရင် ဒီအဆင့်မှာတင် လုံလောက်ပါတယ်။

ဒါပေမယ့် ကိုယ့်အတွေ့အကြုံ မလုံလောက်သေးဘူးလို့ ခံစားရရင်၊ ဒါမှမဟုတ် “ဒါက တကယ်ပုံမှန်မဟုတ်ဘူး” ဆိုတာကို ပိုပြီး သက်သေပြချင်ရင် ထပ်မံ အချက်အလက်စုဆောင်းပြီး analysis လုပ်နိုင်ပါတယ်။ ဒီအချိန်မှာ analyst က မေးသင့်တဲ့ မေးခွန်းတွေက “ဒီလို behavior ကို အခြား system တွေမှာလည်း တွေ့ရလား” ဆိုတဲ့ ပုံစံပါ။ အထူးသဖြင့် winlogon.exe ကသာ ဒီလိုလုပ်နေတာလား၊ တခြား host တွေမှာ winlogon.exe က SAMR enumeration လုပ်နေတာ ရှိလားဆိုတာကို ကြည့်ရပါမယ်။

တစ်ချိန်တည်းမှာပဲ “Win11-20$ လို workstation account မဟုတ်တဲ့ user တွေက SAMR enumeration လုပ်နေတာ ရှိလား” ဆိုတာကိုလည်း စစ်ကြည့်ရပါမယ်။ ပုံမှန်အားဖြင့် Domain Admin account သို့မဟုတ် management server တွေကသာ SAMR ကို သုံးပြီး enumeration လုပ်တာမျိုး ရှိ

နိုင်ပါတယ်။ ဒါကြောင့် environment တစ်ခုလုံးမှာ SAMR enumeration ကို ဘယ် user တွေ လုပ်နေလဲဆိုတာကို ကြည့်ခြင်းက anomaly ကို ပိုပြီး ပြတ်သားစေပါတယ်။

ဒီအတွက် SIEM ထဲမှာ simple search တစ်ခု လုပ်လို့ရပါတယ်။ Event ID 5145 ဖြစ်ပြီး RelativeTargetName က samr ဖြစ်တဲ့ event တွေကို စုစည်းကြည့်တာပါ။ အောက်က query က Win11-20$ ကို ချန်ထားပြီး SAMR ကို access လုပ်ထားတဲ့ အခြား user တွေကို ရှာဖို့ အသုံးပြုထားတာ ဖြစ်ပါတယ်။

```
external_table('Winlog')
| where EventCode == 5145 and SubjectUserName !startswith "win11-20"
| summarize count() by SubjectUserName
```

ဒီ query ရဲ့ အဓိပ္ပါယ်က Domain Controller ပေါ်မှာ SAMR pipe ကို access လုပ်ထားတဲ့ user တွေကို စုစည်းပြီး ဘယ် account က ဘယ်လောက်ကြိမ် လုပ်ထားလဲဆိုတာကို ကြည့်တာပါ။ Environment က ကြီးမားရင် ဒီလို aggregation က အရမ်းအရေးကြီးပါတယ်။ ဘာကြောင့်လဲဆိုတော့ ပုံမှန် behavior ဖြစ်တဲ့ account တွေက အမြဲပေါ်လာပြီး၊ anomalous ဖြစ်တဲ့ account တွေက ချက်ချင်း ထင်ရှားလာလို့ပါ။

Investigating Processes

Investigation လုပ်တဲ့အခါ process တွေက အလွန်အမင်း မရှင်းလင်းမှုကို ဖြစ်စေပါတယ်။ Analyst တွေ မှားယွင်းတဲ့ assessment တွေ ချမိတာရဲ့ အကြီးဆုံးအကြောင်းရင်းတွေထဲမှာ process ကို မမှန်ကန်စွာ နားလည်ခြင်းက ပါဝင်ပါတယ်။ ဒါကြောင့် ဒီသင်ခန်းစာမှာ process အကြောင်းကို သီးသန့်အပိုင်းတစ်ခုအနေနဲ့ ဆွေးနွေးထားတာပါ။ အကြောင်းရင်းက attacker အများစုဟာ နောက်ထပ် attack တွေ လုပ်ဖို့အတွက် target system ထဲမှာ beacon သို့မဟုတ် implant တစ်ခုခုကို အရင်ဆုံး ချထားတတ်ကြလို့ပါ။ အဲ့ဒီ beacon သို့မဟုတ် implant တွေဟာ မဖြစ်မနေ process တစ်ခုရဲ့ အတွင်းမှာပဲ run ရပါတယ်။

နောက်တစ်ချက်က အခုခေတ်မှာ EDR က နေရာတိုင်းလိုလိုမှာ ရှိနေပြီဖြစ်တဲ့အတွက် SOC analyst တွေ အနေနဲ့ process နဲ့ ပတ်သက်တဲ့ alert တွေကို စုံစမ်းရတာ အလွန်အမင်း ပုံမှန်အလုပ်တစ်ခု ဖြစ်လာပါတယ်။ ဒါပေမယ့် beginner analyst တွေအတွက်တော့ process တစ်ခုကို ဘယ်အချိန်မှာ နက်နက်ရှိုင်းရှိုင်း တူးဖော်စုံစမ်းသင့်လဲ၊ ဘယ်အချိန်မှာ “ဒါပုံမှန်ပါပဲ” လို့ ချန်ထားပြီး ဆက်မလုပ်သင့်တော့ဘူးလဲ ဆိုတာကို ခွဲခြားဖို့ အရမ်းခက်ခဲပါတယ်။

Attacker တစ်ယောက်က user space ထဲမှာ malicious code ကို run ချင်ရင် process တစ်ခုအတွင်းမှာပဲ run ရပါမယ်။ ဒါပေမယ့် security အမြင်နဲ့ကြည့်ရင် process တွေက အလွန် flexible ဖြစ်ပြီး အလွန် complex ဖြစ်ပါတယ်။ Process တစ်ခုဟာ thread အများကြီးနဲ့ အလုပ်လုပ်တတ်ပြီး အဲ့ဒီ thread တွေက မူလ executable file နဲ့ တိုက်ရိုက် မဆိုင်တဲ့ code တွေ ဖြစ်နိုင်ပါတယ်။ ဒါကြောင့် analyst အနေနဲ့ process name, code signing certificate, hash တို့ကိုသာ ကြည့်ပြီး “ဒါ safe ပါ” ဒါမှမဟုတ် “ဒါ malicious ပါ” လို့ ဆုံးဖြတ်တာက အလွန်အန္တရာယ်ရှိပါတယ်။

ပိုပြီးခက်ခဲသွားစေတာက process တွေရဲ့ အပြုအမူတွေပါ။ Process တစ်ခုက သူ့ရဲ့ parent process ကို မှန်မှန်မပြဘဲ spoof လုပ်ထားနိုင်ပါတယ်။ Process ကို user တစ်ယောက်က စတင်ခဲ့ပေမယ့် network resource တွေနဲ့ ဆက်သွယ်တဲ့အခါ အခြား user context နဲ့ ဆက်သွယ်နေတာမျိုးလည်း ဖြစ်နိုင်ပါတယ်။ Process တစ်ခုက အခြား process တွေရဲ့ memory ကို ဖတ်နိုင်၊ ရေးနိုင်ပါတယ်။ ဒါ့အပြင် ကိုယ်တိုင်

မဟုတ်ဘဲ အခြား process ထဲမှာ thread အသစ်တွေကို စတင်စေတဲ့ injection လုပ်ရပ်တွေကိုပါ လုပ်နိုင်ပါတယ်။

ဒီလိုအရာတွေကြောင့် အတွေ့အကြုံနည်းတဲ့ analyst တွေအတွက် process investigation က အလွန်အမင်း စိတ်ရှုပ်ထွေးစေပါတယ်။ Threat တွေက disk ပေါ်မှာ malicious exe တစ်ခုတည်းနဲ့ပဲ run နေမယ်ဆိုရင် VirusTotal လို service ထဲ ထည့်ပြီး စစ်လိုက်ရုံနဲ့ အလုပ်ပြီးသွားနိုင်ပါလိမ့်မယ်။ ဒါပေမယ့် လက်တွေ့မှာ attack အများစုက ဒီလောက် မရိုးရှင်းပါဘူး။ Process-based alert အများစုက SOP တစ်ခုကို တိတိကျကျ လိုက်လုပ်ရုံနဲ့ မဖြေရှင်းနိုင်တဲ့ sophisticated investigation ကို လိုအပ်ပါတယ်။

Alert တစ်ခုကို ပထမဆုံးရလာတဲ့အချိန်မှာ analyst က အခြေခံ information gathering နဲ့ analysis ကို လုပ်ပါတယ်။ Who, what, when, where, why ဆိုတဲ့ မေးခွန်းတွေကနေ process အတွက် လိုအပ်တဲ့ အချက်အလက်တွေကို ဆွဲထုတ်နိုင်ပါတယ်။ ဥပမာအားဖြင့် ဒီ process ကို

- ဘယ် user context နဲ့ run လုပ်ထားလဲ၊
- integrity level ဘယ်လောက်ရှိလဲဆိုတာကို သိရပါမယ်။
- Parent process က ဘာလဲ၊
- PPID ကို spoof လုပ်ထားတာလားဆိုတာကို စစ်ရပါမယ်။
- Process စတင်ခဲ့တဲ့ အချိန်၊
- command line argument တွေ၊
- process နဲ့ parent process နှစ်ခုလုံးရဲ့ run ပုံစံကို ကြည့်ရပါမယ်။
- Process က ဘယ် host ပေါ်မှာ run နေတာလဲ၊
- ဘယ် activity ကို suspicious လို့ သတ်မှတ်ထားတာလဲ ဆိုတာတွေကိုပါ သိထားရပါမယ်။

ဒီအခြေခံအချက်အလက်တွေကို ရလာတဲ့အခါ analyst က “ဒီ activity က threat model တစ်ခုနဲ့ ကိုက်ညီလား” ဆိုတာကို စဉ်းစားနိုင်လာပါတယ်။ Threat model ရှိလား၊ မရှိလားဆိုတာက analyst ရဲ့ ပထမဆုံး theory တွေကို ဆုံးဖြတ်ပေးပါတယ်။ အဲ့ဒီနောက်မှာပဲ

- “ဒီ context မှာ benign ဖြစ်နိုင်လား”
- “ဒီ user သို့မဟုတ် ဒီ process က အရင်က ဒီလိုလုပ်ဖူးလား”
- “တူညီတဲ့ user သို့မဟုတ် process တွေက ဒီလိုလုပ်ဖူးလား”
- ဆိုတဲ့ မေးခွန်းတွေကို ဆက်ပြီး မေးလာရပါတယ်။ ဒီမေးခွန်းတွေက investigation ကို ဘယ်ဘက်ကို ဦးတည်ရမလဲဆိုတာ လမ်းညွှန်ပေးပါတယ်။

Process investigation မှာ timelining ဆိုတာ အလွန်အစွမ်းထက်တဲ့ နည်းလမ်းတစ်ခုပါ။ SAMR ဥပမာမှာလို WIN11-20$ ရဲ့ activity ကို Domain Controller ပေါ်မှာ timeline ဆွဲပြီး enumeration theory ကို ထောက်ခံခဲ့တာကို သတိရနိုင်ပါတယ်။ Process အတွက်ဆိုရင်

- DNS request တွေ၊
- internal နဲ့ external network connection တွေ၊
- child process တွေ၊
- injection event တွေ၊
- file နဲ့ registry access တွေ၊
- module load တွေ၊
- named pipe အသုံးပြုမှုတွေ၊

- share access နဲ့ credential access တွေကို timeline အနေနဲ့ ကြည့်နိုင်ပါတယ်။

ဒါတွေက အကုန်မဟုတ်သေးပေမယ့် EDR နဲ့ OS log တွေက ပုံမှန်အားဖြင့် ဒီလို telemetry အချို့ကိုတော့ ပေးနိုင်ရပါမယ်။

အကယ်၍ ဒီလို telemetry တွေကို မရနိုင်ဘူးဆိုရင် raw host log တွေကို ဈေးအနိမ့်ဆုံး storage တစ်ခုထဲ ကိုပဲဖြစ်ဖြစ် စုထားဖို့ အချိန်နဲ့ အားထုတ်မှု သုံးရတာက တန်ဖိုးရှိပါတယ်။ Investigation လုပ်တဲ့အခါ “အချက်အလက် မရှိလို့ မဆုံးဖြတ်နိုင်ဘူး” ဆိုတာက analyst အတွက် အားနည်းချက်တစ်ခု ဖြစ်လာနိုင်လို့ပါ။

ဒီလို timeline တွေကို ကြည့်ရတာက တကယ်ပဲ စိတ်ပင်ပန်းစရာ ဖြစ်ပါတယ်။ ဒါပေမယ့် အရေးကြီးတာက အချက်အလက်အားလုံးကို တစ်ပြိုင်နက် နားလည်ဖို့ မလိုပါဘူး။ Analyst ရဲ့ အဓိက ရည်ရွယ်ချက်က incident ကို escalate လုပ်မလား၊ de-escalate လုပ်မလား ဆိုတာကို ဆုံးဖြတ်ခြင်းပါ။ ဒါကြောင့် dataset ကြီးထဲမှာ ဘာကို ဦးစားပေးကြည့်မလဲဆိုတာကို ကိုယ့်ရဲ့ theory ကို အတည်ပြု သို့မဟုတ် ဖျက်သိမ်းနိုင်မယ့် အချက်အလက်တွေကို အခြေခံပြီး ရွေးချယ်ရပါမယ်။

Process တစ်ခုက အချိန်တိုင်း ဘာလုပ်နေလဲဆိုတာကို အပြည့်အစုံ နားလည်ဖို့ ကြိုးစားတာက investigation ကို ပိုပြီး နှေးကွေးစေပြီး မလိုအပ်တဲ့ rabbit hole တွေထဲ ဝင်သွားစေနိုင်ပါတယ်။ အဲ့ဒီလိုဖြစ်သွားရင် အချိန်ကုန်ပြီး အဓိကဖြစ်ရပ်ကို နားလည်မှုက ပိုပြီး လျော့သွားတတ်ပါတယ်။ ဒါကြောင့် process investigation မှာ အရေးကြီးဆုံးက “ရည်ရွယ်ချက်ကို မမေ့ဘဲ လိုအပ်တာကိုသာ ဦးစားပေးကြည့်ခြင်း” ဖြစ်ပါတယ်။

NPP Alert - Understand

EDR ကနေ alert တစ်ခု ရလာပါတယ်။

- Title: Network Service Discovery
- Time: 2024-01-30T02:01:32.869Z
- Target Host: DC
- Source User: alice
- Source Process: notepad++.exe
- Source Host: WIN11-20
- Description: User alice performed network service discovery against DC

ဒီ alert ကို ဖတ်လိုက်တာနဲ့ analyst အနေနဲ့ who, what, when, where ဆိုတဲ့ မေးခွန်းတွေကို တော်တော်လေး လွယ်လွယ်ကူကူ ဖြေနိုင်ပါတယ်။

- **who** - aceresponder\alice
- **what** - Scanned a large number of ports on the DC
- **when** - 2024-01-30T02:01:32.869Z
- **where** - On Win11-20 in the context of notepad++.exe

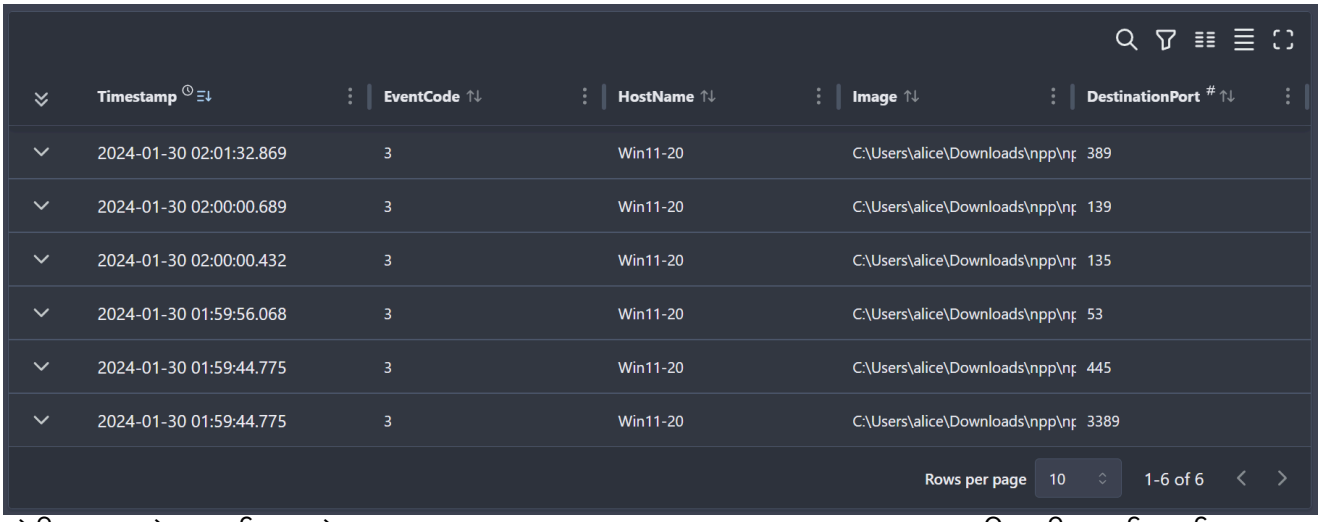
ဒီ alert ကို ဖတ်လိုက်တာနဲ့ analyst အနေနဲ့ who, what, when, where ဆိုတဲ့ မေးခွန်းတွေကို တော်တော်လေး လွယ်လွယ်ကူကူ ဖြေနိုင်ပါတယ်။ Activity ကို လုပ်ခဲ့တဲ့သူက aceresponder\alice ဖြစ်ပါတယ်။

What ဆိုရင် Domain Controller ပေါ်မှာ port အများကြီးကို scan လုပ်ပြီး ဘယ် service တွေ run နေလဲဆိုတာကို စူးစမ်းခဲ့တာဖြစ်ပါတယ်။ When ဆိုရင် alert ထဲမှာပါတဲ့ timestamp ဖြစ်ပြီး Where ဆိုရင် WIN11-20 workstation ပေါ်မှာ notepad++.exe process အတွင်းကနေ လုပ်ခဲ့တာဖြစ်ပါတယ်။

NPP Alert - Validate

ပထမဆုံး analyst လုပ်တာက notepad++.exe process က တကယ် network activity လုပ်ထားလားဆိုတာကို စစ်ကြည့်တာပါ။ Sysmon EventCode 3 (NetworkConnect) ကို သုံးပြီး alert ဖြစ်တဲ့ အချိန်အနီးတဝိုက်မှာ notepad++.exe ကနေ network connection တွေ ရှိလားဆိုတာကို query ဆွဲပါတယ်။ Query ရဲ့အဓိက အကြောင်းအရာက alert မတိုင်ခင် နည်းနည်းနဲ့ alert ပြီးပြီးချင်း အချိန်အတွင်း notepad++.exe image ပါတဲ့ network connection event တွေကို ရှာတာပါ။

```
external_table('Winlog')
| where EventCode == 3 and Timestamp between (datetime(2024-01-
30T01:55:00.000Z) .. datetime(2024-01-30T02:05:00.000Z)) and Image contains
"notepad++.exe"
| project Timestamp, EventCode, HostName, Image, DestinationPort
```



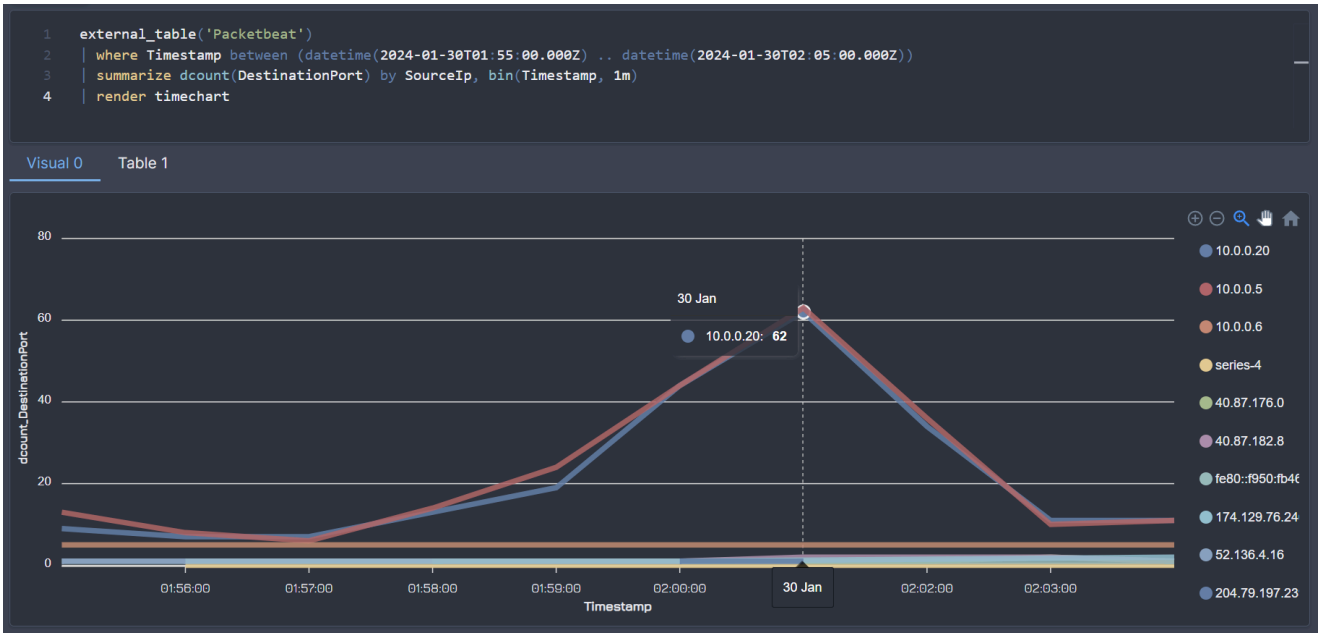
Timestamp	EventCode	HostName	Image	DestinationPort
2024-01-30 02:01:32.869	3	Win11-20	C:\Users\alice\Downloads\npp\nr	389
2024-01-30 02:00:00.689	3	Win11-20	C:\Users\alice\Downloads\npp\nr	139
2024-01-30 02:00:00.432	3	Win11-20	C:\Users\alice\Downloads\npp\nr	135
2024-01-30 01:59:56.068	3	Win11-20	C:\Users\alice\Downloads\npp\nr	53
2024-01-30 01:59:44.775	3	Win11-20	C:\Users\alice\Downloads\npp\nr	445
2024-01-30 01:59:44.775	3	Win11-20	C:\Users\alice\Downloads\npp\nr	3389

Rows per page 10 1-6 of 6

အဲ့ဒီ query ရဲ့ရလဒ်အနေနဲ့ notepad++.exe ကနေ unique destination port ၆ ခုကို ဆက်သွယ်ထားတာတွေ့ရပါတယ်။ ဒီအချက်က အလွန်အရေးကြီးပါတယ်။ Notepad++ ဆိုတာ ပုံမှန်အားဖြင့် local file edit လုပ်ဖို့ သုံးတဲ့ application ဖြစ်ပြီး network port အများကြီးကို လိုက်ပြီး connect လုပ်နေဖို့ မလိုအပ်ပါဘူး။ ဒါကြောင့် “probably not something notepad++ should be doing” လို့ ပြောထားတာက security context အရ အလွန်မှန်ပါတယ်။ ဒီအချက်တစ်ခုတည်းနဲ့တင် alert ဟာ false positive မဖြစ်နိုင်ဘူးဆိုတဲ့ အရိပ်အယောင်ကို ရရှိပြီးသားဖြစ်ပါတယ်။

ဒါပေမယ့် analyst က ဒီနေရာမှာပဲ မရပ်ပါဘူး။ “Notepad++ က ပုံမှန်အားဖြင့် network မသုံးဘူး” ဆိုတာက common sense ဖြစ်ပေမယ့် environment-specific baseline မရှိရင် ခိုင်မာမှုနည်းနိုင်ပါတယ်။ ဥပမာအားဖြင့် ဒီ organization ထဲမှာ Notepad++ ကို plugin တွေနဲ့ automation tool အဖြစ် သုံးနေတယ်ဆိုရင် network activity ရှိနိုင်ပါတယ်။ ဒါကြောင့် notepad++.exe ရဲ့ instance အခြားတွေကို စစ်ကြည့်မယ်ဆိုရင် reference အနည်းငယ်ပဲ ရနိုင်တယ်၊ အထူးသဖြင့် environment ထဲမှာ Notepad++ မများရင် ပိုပြီး မခိုင်မာပါဘူး။

အဲ့ဒီကြောင့် ပိုကောင်းတဲ့ validate လုပ်နည်းတစ်ခုအနေနဲ့ analyst က **netflow dataset** ကို အသုံးချပါတယ်။ Netflow ရဲ့ အားသာချက်က process-level မဟုတ်ပေမယ့် network behavior ကို ပိုကျယ်ကျယ်ပြန့်ပြန့် မြင်နိုင်တာပါ။ အထူးသဖြင့် successful connection တွေတင်မက failed connection တွေပါ ပါဝင်တာကြောင့် scanning behavior ကို ဖော်ထုတ်ဖို့ အလွန်အသုံးဝင်ပါတယ်။



Analyst က source IP တစ်ခုက destination port မျိုးစုံကို အချိန်တိုအတွင်း ဘယ်လောက် connect လုပ်ထားလဲဆိုတာကို visualization လုပ်ပါတယ်။ အဲ့ဒီ graph ကို ကြည့်လိုက်တဲ့အခါ 10.0.0.20 (WIN11-20) ကနေ destination port မျိုးစုံကို ဆက်သွယ်ထားတဲ့ spike ကြီးတစ်ခုကို တွေ့ရပါတယ်။ ဒီလို pattern က network scanning ရဲ့ classic behavior ဖြစ်ပါတယ်။ ပုံမှန် user activity မှာ ဒီလောက် port မျိုးစုံကို အချိန်တိုအတွင်း connect လုပ်တာ မဖြစ်တတ်ပါဘူး။

ဒီနေရာမှာ analyst က အရေးကြီးတဲ့ reasoning တစ်ခုကို သုံးပါတယ်။ Netflow event တွေကို notepad++.exe process နဲ့ ၁၀၀% တိုက်ရိုက် ချိတ်မရပါဘူး။ Process name မပါဘူး၊ IP-level data ပဲ ဖြစ်ပါတယ်။ ဒါပေမယ့် အချိန်၊ source host၊ alert timing နဲ့ behavior pattern တွေ အားလုံးကို ချိတ်ဆက်စဉ်းစားလိုက်တဲ့အခါ “ဒီ scanning traffic ဟာ notepad++.exe alert နဲ့ ဆက်စပ်နေတာဖြစ်နိုင်ချေ အလွန်မြင့်တယ်” လို့ inference လုပ်နိုင်ပါတယ်။ Cybersecurity investigation မှာ ဒီလို correlation နဲ့ inference က အလွန်အရေးကြီးပါတယ်။

NPP Alert - Theory Refinement

အခုအချိန်မှာ analyst အနေနဲ့ notepad++.exe process က suspicious network scanning လုပ်ထားတယ်ဆိုတာကို အတော်လေး ယုံကြည်နိုင်လာပြီ ဖြစ်ပါတယ်။ ဒါပေမယ့် “suspicious” လို့ သိရုံနဲ့ မပြီးသေးပါဘူး။ နောက်တစ်ဆင့်မှာ အရေးကြီးတာက **ဘယ်လိုနည်းနဲ့ attacker က notepad++.exe အတွင်းမှာ malicious code ကို execute လုပ်နိုင်ခဲ့သလဲ** ဆိုတာကို ပိုပြီး တိတိကျကျ စဉ်းစားဖို့ ဖြစ်ပါတယ်။ ဒါကိုပဲ theory refinement လို့ ခေါ်ပါတယ်။

Theory refinement မလုပ်ခင် အရင်ဆုံး analyst က “အမှန်တကယ် ဖြစ်ပျက်ခဲ့တာတွေ” ကို သေချာအောင် လုပ်ရပါမယ်။ ဒါကြောင့် suspicious network connection event တွေထဲမှာ ပါလာတဲ့ **Process ID (PID)** ကို သုံးပြီး process creation event ကို ပြန်ရှာပါတယ်။ Windows logs မှာ EventCode 1

(Process Create) ကို ကြည့်လိုက်ရင် notepad++.exe ကို ဘယ်အချိန်မှာ၊ ဘယ်သူက၊ ဘယ် process က စတင်ပေးခဲ့လဲဆိုတာကို သိနိုင်ပါတယ်။

```
external_table('Winlog')
| where EventCode == 1 and ProcessId == 5188
```

Query ရဲ့ ရလဒ်အရ notepad++.exe ရဲ့ parent process က explorer.exe ဖြစ်နေပါတယ်။ ဒီအချက်က အလွန်အရေးကြီးပါတယ်။ Explorer.exe ဆိုတာ Windows File Explorer ဖြစ်ပြီး user က icon ကို double-click လုပ်တဲ့အခါ process အသစ်တွေကို စတင်ပေးတတ်ပါတယ်။ ဒါကြောင့် ဒီအခြေအနေမှာ “alice က Explorer window ထဲကနေ Notepad++ ကို ပုံမှန်လို click လုပ်ပြီး ဖွင့်ခဲ့တာ ဖြစ်နိုင်တယ်” လို့ အလွန်တန်ဖိုးရှိတဲ့ inference တစ်ခုကို လုပ်နိုင်ပါတယ်။ ဆိုလိုတာက malware dropper တစ်ခုက background မှာ silent execution လုပ်လိုက်တာမျိုး မဟုတ်နိုင်ဘူးဆိုတဲ့ possibility ကို လျော့ချပေးလိုက်တာပါ။

Signature Verification

Signed file, valid signature

File Version Information

Copyright	Copyleft 1998-2022 by Don HO
Product	Notepad++
Description	Notepad++
Original Name	notepad++.exe
Internal Name	notepad++.exe
File Version	8.54
Date signed	2023-06-17 23:46:00 UTC

Signers

+ Notepad++
+ DigiCert Trusted G4 Code Signing RSA4096 SHA384 2021 CA1
+ DigiCert Trusted Root G4
+ DigiCert

နောက်တစ်ဆင့်အနေနဲ့ analyst က notepad++.exe ရဲ့ file hash ကို VirusTotal လိုမျိုး service တွေမှာ စစ်ကြည့်ပါတယ်။ ဒီအဆင့်ရဲ့ ရည်ရွယ်ချက်က “Executable ကိုယ်တိုင်က malware လား၊ trojanized version လား” ဆိုတာကို စစ်ဆေးဖို့ပါ။ စစ်ကြည့်လိုက်တဲ့အခါ notepad++.exe က signed ဖြစ်ပြီး clean ဖြစ်တယ်ဆိုတာ သိရပါတယ်။ အဲ့ဒီလိုဆိုရင် fake Notepad++ binary တစ်ခုကို attacker က drop လုပ်ပြီး user ကို run ခိုင်းလိုက်တယ်ဆိုတဲ့ theory ကို လုံးဝနီးပါး ဖယ်ရှားနိုင်ပါတယ်။

ဒီနေရာမှာ အရေးကြီးတဲ့ သဘောတရားတစ်ခုရှိပါတယ်။ File က clean ဖြစ်တယ်၊ signature လည်းမှန်တယ်ဆိုတာက “notepad++.exe က အပြစ်မရှိဘူး” လို့ မဆိုလိုပါဘူး။ အဲဒါက “Executable ကိုယ်တိုင်က malware မဟုတ်ဘူး” လို့ပဲ ဆိုလိုတာပါ။ Attacker က legitimate process ကို abuse လုပ်နိုင်တဲ့ နည်းလမ်းတွေ အများကြီး ရှိပါတယ်။ ဒီအချက်ကို နားမလည်တဲ့ beginner analyst တွေက “VirusTotal clean = false positive” လို့ အမြန်ဆုံး ဆုံးဖြတ်ပြီး incident ကို close လုပ်တတ်ကြပါတယ်။ ဒီ module က အဲဒီအမှားကို ရှောင်နိုင်အောင် သင်ပေးချင်တာပါ။

အခုအချိန်မှာ analyst က အခြေခံအချက်အလက်တွေကို သုံးပြီး “attacker ဘယ်လိုနည်းနဲ့ notepad++.exe အတွင်းမှာ malicious behavior ကို လုပ်နိုင်ခဲ့မလဲ” ဆိုတာကို စဉ်းစားရပါမယ်။ Process security အကြောင်းကို နားလည်ထားရင် theory အချို့ကို ဆောက်နိုင်ပါတယ်။

ပထမဆုံး ဖြစ်နိုင်ချေက attacker က notepad++ ကို **client-side exploit** နဲ့ exploit လုပ်ခဲ့တာ ဖြစ်နိုင်ပါတယ်။ ဥပမာအားဖြင့် malicious file တစ်ခုကို notepad++ နဲ့ ဖွင့်လိုက်တဲ့အခါ vulnerability တစ်ခုကြောင့် arbitrary code execution ဖြစ်သွားတာမျိုးပါ။ ဒီလိုဆိုရင် process name က legit ဖြစ်နေပြီး behavior ကတော့ malicious ဖြစ်လာနိုင်ပါတယ်။

ဒုတိယ ဖြစ်နိုင်ချေက attacker က legitimate notepad++.exe process ထဲကို **code injection** လုပ်ခဲ့တာ ဖြစ်နိုင်ပါတယ်။ ဒီအခြေအနေမှာ notepad++ ကို user က ပုံမှန်ဖွင့်ထားပြီးသား ဖြစ်နိုင်ပါတယ်။ Attacker ရဲ့ implant က memory injection, thread injection စတဲ့ technique တွေကို သုံးပြီး notepad++.exe process အတွင်းမှာ malicious thread တစ်ခု run နေတာ ဖြစ်နိုင်ပါတယ်။ ဒီလိုဆိုရင် hash, signature, process name အားလုံး legit ဖြစ်နေပေမယ့် network scanning လို malicious activity တွေကို လုပ်နိုင်ပါတယ်။

တတိယ ဖြစ်နိုင်ချေက attacker က notepad++ ရဲ့ **legitimate execution flow ကို hijack** လုပ်ခဲ့တာ ဖြစ်နိုင်ပါတယ်။ ဥပမာ DLL search order hijacking, plugin abuse, configuration abuse စတာတွေ ဖြစ်နိုင်ပါတယ်။ Notepad++ က plugin system ရှိတဲ့ application ဖြစ်တဲ့အတွက် attacker က malicious plugin တစ်ခုကို load လုပ်စေခဲ့ရင် process behavior က လုံးဝ ပြောင်းလဲသွားနိုင်ပါတယ်။

ဒီအဆင့်မှာ အရေးကြီးဆုံး သင်ခန်းစာက “အဖြေကို ချက်ချင်း မဆုံးဖြတ်ဘဲ possibility တွေကို လျော့ချသွားတာ” ပါ။ Legitimate executable တစ်ခုက malicious behavior ပြုလုပ်နေတယ်ဆိုရင် executable ကိုယ်တိုင်ကို မကြည့်ဘဲ **process context, execution method, memory behavior** စတဲ့ အချက်တွေကို နက်နက်ရှိုင်းရှိုင်း ဆက်လက် စုံစမ်းရမယ်ဆိုတဲ့ mindset ကို ဒီ theory refinement အပိုင်းက သင်ပေးနေပါတယ်။

NPP Alert - Gather and Analyze

ဒီအပိုင်းမှာတော့ **NPP Alert – Gather and Analyze** ကို ဆက်လက်ရှင်းပြပါမယ်။ အခုအချိန်မှာ analyst အနေနဲ့ theory refinement အဆင့်မှာ ဖြစ်နိုင်ချေရှိတဲ့ theory သုံးခုကို ဆောက်ပြီးသားဖြစ်ပါတယ်။ ဒီ Gather and Analyze အဆင့်ရဲ့ ရည်ရွယ်ချက်က အဲဒီ theory တွေကို အချက်အလက်နဲ့ တစ်ခုချင်းစီ စစ်ပြီး ဖယ်ရှားသွားခြင်း ဖြစ်ပါတယ်။ ဒီနေရာမှာ analyst က အလွယ်ဆုံး၊ စစ်ရတာ အချိန်နည်းဆုံး ဖြစ်တဲ့ **low-hanging fruit** ကို အရင်ဆုံး ကိုင်တွယ်ပါတယ်။ အဲဒါကတော့ theory #1 ဖြစ်တဲ့ “attacker က notepad++ ကို client-side exploit နဲ့ exploit လုပ်ခဲ့တာ ဖြစ်နိုင်တယ်” ဆိုတဲ့အယူအဆပါ။

Analyst က ဒီ theory ကို စစ်ဖို့အတွက် notepad++ ရဲ့ ပုံမှန် behavior ကို ကိုယ်တိုင် စမ်းကြည့်ပါတယ်။ File တစ်ခုကို notepad++ နဲ့ ဖွင့်လိုက်တဲ့အခါ Windows မှာ process creation event တစ်ခု ဖြစ်ပြီး command line arguments ထဲမှာ ဖွင့်လိုက်တဲ့ file path ကို ထည့်သွင်းထားတာကို တွေ့ရပါတယ်။ ဥပမာ အားဖြင့် C:\Users\alice\Documents\test.txt လို file ကို ဖွင့်လိုက်ရင် notepad++.exe ရဲ့ command line ထဲမှာ အဲဒီ file path က ထင်ရှားစွာ ပါလာပါတယ်။

ဒီ observation က အလွန်အရေးကြီးပါတယ်။ အခု incident မှာ analyst က process creation event ကို ပြန်ကြည့်တဲ့အခါ notepad++.exe ရဲ့ command line arguments ထဲမှာ suspicious file path တစ်ခုမှ မပါဝင်တာကို သိထားပြီးသား ဖြစ်ပါတယ်။ အဲဒါကြောင့် “attacker က Alice ကို malicious file တစ်ခု ပို့ပြီး notepad++ နဲ့ ဖွင့်ခိုင်းလိုက်တာ” ဆိုတဲ့ social engineering + client-side exploit scenario က လုံးဝမကိုက်ညီတော့ပါဘူး။

စာအရေးအသားထဲမှာ “There are ways to open files in notepad++ without passing the file name on the command line” လို့ ပြောထားတာကိုလည်း သတိထားရပါမယ်။ အမှန်တကယ်တော့ advanced technique တွေ သုံးရင် file path ကို command line မှာ မပြဘဲ file ကို load လုပ်နိုင်တာတွေ ရှိပါတယ်။ ဒါပေမယ့် SOC investigation အတွက် probability ကို စဉ်းစားရပါတယ်။ Attacker တစ်ယောက်အနေနဲ့ social engineering နဲ့ client-side exploit ကို ရွေးချယ်မယ်ဆိုရင် ပို့ရိုးရှင်းတဲ့ နည်းလမ်းတွေကို သုံးတတ်ပြီး ဒီလို edge case technique တွေကို အသုံးပြုနိုင်ချေ နည်းပါတယ်။

ဒါကြောင့် analyst က ဒီအဆင့်မှာ theory #1 ကို “ဖြစ်နိုင်ချေ အလွန်နည်းပါး” လို့ သတ်မှတ်ပြီး လက်ရှိ investigation မှာ အဓိက focus မထားတော့ပါဘူး။ ဒီလိုလုပ်ခြင်းရဲ့ အဓိကအကျိုးကျေးဇူးက investigation ကို မလိုအပ်တဲ့ direction တွေကို မဆွဲဆောင်သွားဘဲ အချိန်နဲ့ အင်အားကို ထိရောက်စွာ အသုံးချနိုင်ခြင်း ဖြစ်ပါတယ်။

Test : NPP Alert - Timelining File Creation

Investigate the history of `C:\Users\alice\Downloads\npp\npp\notepad++.exe` which file was dropped to disk alongside notepad++.exe?

Analyst က အခု investigation မှာ notepad++.exe ကို Alice ရဲ့ Downloads directory အောက်မှာ `npp\npp\notepad++.exe` ဆိုတဲ့ path နဲ့ တွေ့ထားပြီးသား ဖြစ်ပါတယ်။ Legitimate software installer တစ်ခုဆိုရင် file တွေကို တစ်ခုတည်းမဟုတ်ဘဲ supporting DLL တွေ၊ resource file တွေနဲ့ အတူတူ drop လုပ်တတ်ပါတယ်။ အဲ့ဒီလိုပဲ attacker က legitimate application ကို အသုံးချမယ်ဆိုရင် သူ့ malicious code ကို DLL တစ်ခုအနေနဲ့ အတူ drop လုပ်ပြီး execution flow ကို hijack လုပ်နိုင်ပါတယ်။ ဒါကြောင့် file creation history ကို timeline နဲ့ ကြည့်တာက အလွန်အသုံးဝင်ပါတယ်။

ဒီအတွက် analyst က Sysmon EventCode 11 ကို သုံးပါတယ်။ EventCode 11 ဆိုတာ file creation event ဖြစ်ပြီး ဘယ် process က ဘယ် file ကို ဘယ်အချိန်မှာ disk ပေါ်ရေးခဲ့လဲဆိုတာကို ပြပါတယ်။ Query မှာ `alice\Downloads\npp\npp` path ပါတဲ့ TargetFilename တွေကို filter လုပ်ထားတာကြောင့် notepad++ နဲ့ ဆက်စပ်တဲ့ file creation event တွေကိုပဲ မြင်ရပါတယ်။

```
external_table('Winlog')
| where EventCode == 11 and TargetFilename contains
```

```
"alice\\Downloads\\npp\\npp"
| project Timestamp, Image, TargetFilename
```

Query ရဲ့ရလဒ်ကို ကြည့်လိုက်တဲ့အခါ အလွန်အရေးကြီးတဲ့ observation တစ်ခုကို တွေ့ရပါတယ်။ Explorer.exe က notepad++.exe ကို disk ပေါ်ရေးတဲ့ အချိန်နီးနီးမှာပဲ **dbghelp.dll** ဆိုတဲ့ DLL file ကိုလည်း အတူတူ disk ပေါ်ရေးထားတာကို တွေ့ရပါတယ်။ Timestamp တွေကလည်း အလွန်နီးစပ်နေပါတယ်။ ဒီလို behavior က “user က zip file တစ်ခုကို extract လုပ်လိုက်တယ်” ဒါမှမဟုတ် “installer archive ကို unpack လုပ်လိုက်တယ်” ဆိုတဲ့ scenario နဲ့ အလွန်ကို ကိုက်ညီပါတယ်။

ဒါပေမယ့် security point of view ကနေ ကြည့်ရင် ဒီ observation က ပိုပြီး အရေးကြီးပါတယ်။ dbghelp.dll ဆိုတာ Windows debugging library တစ်ခုဖြစ်ပြီး DLL search order hijacking မှာ မကြာခဏ အသုံးချခံရတတ်တဲ့ DLL အမျိုးအစားပါ။ Legitimate executable တစ်ခုက current working directory ထဲမှာ DLL ကို အရင်ရှာတဲ့ behavior ရှိရင် attacker က malicious dbghelp.dll ကို executable နဲ့အတူ drop လုပ်ပြီး code execution ကို hijack လုပ်နိုင်ပါတယ်။

NPP Alert - Gather and Analyze 2

အရင်ဆုံး **theory #2** ကို စကြည့်ပါမယ်။ Theory #2 က attacker က legitimate notepad++.exe process ထဲကို **code injection** လုပ်ခဲ့တာ ဖြစ်နိုင်တယ်ဆိုတဲ့ အယူအဆပါ။ Code injection ဆိုရင် Windows မှာ အများဆုံးတွေ့ရတဲ့ technique က CreateRemoteThread ကို သုံးတဲ့ နည်းလမ်းပါ။ **Sysmon Event ID 8 က CreateRemoteThread event** ဖြစ်ပြီး “process တစ်ခုက အခြား process ထဲမှာ thread အသစ်တစ်ခု ဖန်တီးလိုက်ပြီ” ဆိုတဲ့ အချိန်ကို မှတ်တမ်းတင်ထားပါတယ်။

Analyst က ဒီ event ကို သုံးပြီး TargetProcessId က notepad++.exe (PID 5188) ဖြစ်တဲ့ injection event ရှိလားဆိုတာကို စစ်ကြည့်ပါတယ်။ Query ရဲ့ အဓိကရည်ရွယ်ချက်က “notepad++.exe ထဲကို အခြား process က thread inject လုပ်ခဲ့တဲ့ evidence ရှိလား” ဆိုတာကို ရှာဖို့ပါ။ ရလဒ်အနေနဲ့ ဘာမှ မတွေ့ရပါဘူး။

ဒီနေရာမှာ analyst က အလွန်အရေးကြီးတဲ့ professional judgment တစ်ခုကို ပြုလုပ်ပါတယ်။ **“Injection ကို CreateRemoteThread မသုံးဘဲ လုပ်နိုင်တာတွေ ရှိတယ်”** ဆိုတာကို သူက သိထားပါတယ်။ ဒါပေမယ့် SOC investigation ရဲ့ လက်တွေ့ဘဝမှာ CreateRemoteThread ကို သုံးတဲ့ technique တွေက အများဆုံးဖြစ်ပါတယ်။ ထို့ကြောင့် Sysmon Event 8 မတွေ့ဘူးဆိုတာနဲ့တင် theory #2 ကို ယာယီဘေးဖယ်ထားနိုင်ပါတယ်။ ဒါကို **“table this theory for now”** လို့ ပြောထားတာပါ။ နောက်ပိုင်းမှာ အခြား evidence မတွေ့ရရင် ပြန်လာကြည့်နိုင်အောင် theory ကို အပြီးအပိုင် ဖျက်မပစ်ဘဲ ထားထားတာက စနစ်တကျ investigation လုပ်တဲ့ နည်းလမ်းပါ။

```
external_table('Winlog')
| where EventCode == 8 and TargetProcessId == 5188
```

နောက်တစ်ဆင့်မှာ analyst က **theory #3** ကို အာရုံစိုက်ပါတယ်။ Theory #3 က execution hijacking ဖြစ်ပြီး theory #1 နဲ့ #2 ထက် ပိုကျယ်ပြန့်ပါတယ်။ Execution hijacking ဆိုတာ DLL search order hijacking, side-loading, plugin abuse စတဲ့ နည်းလမ်းများစွာ ပါဝင်နိုင်ပါတယ်။ ဒီအချိန်မှာ analyst က

threat model ကို ပြန်စဉ်းစားပါတယ်။ **“Attacker ရဲ့ code က ဘယ်လိုနည်းနဲ့ system ထဲကို ဝင်လာနိုင်လဲ”** ဆိုတဲ့ မေးခွန်းပါ။

Notepad++.exe က Alice ရဲ့ Downloads folder ထဲမှာ ရှိနေတဲ့အချက်ကို ထပ်မံစဉ်းစားလိုက်တဲ့အခါ Alice ကိုယ်တိုင် download လုပ်ထားတာဖြစ်နိုင်ခြေ အလွန်မြင့်ပါတယ်။ ဒီလိုဆိုရင် attacker က phishing link၊ fake download page စတာတွေကနေ archive တစ်ခုကို Alice ကို download လုပ်ခိုင်းပြီး extract လုပ်စေခဲ့တာ ဖြစ်နိုင်ပါတယ်။ ဒီ scenario ကို နောက်ဆုံး file creation timeline နဲ့ ချိတ်ကြည့်လိုက်တဲ့အခါ notepad++.exe နဲ့ dbghelp.dll ကို တစ်ပြိုင်နက် drop လုပ်ထားတာကို တွေ့ထားပြီးသား ဖြစ်ပါတယ်။ ဒီ proximity က ပိုပြီး မသက်ဆိုးစရာ ဖြစ်လာပါတယ်။

ဒီအချက်တွေကို အခြေခံပြီး analyst က theory ကို ထပ်မံ refine လုပ်ပါတယ်။ “Attacker က DLL search order hijacking ကို အသုံးပြုပြီး legitimate, signed notepad++.exe process အတွင်းမှာ malicious code ကို execute လုပ်ခဲ့တယ်” ဆိုတဲ့ theory ပါ။ အထူးသဖြင့် notepad++ ရဲ့ ဒီ version က dbghelp.dll ကို relative path နဲ့ load လုပ်တဲ့ behavior ရှိနေတဲ့အတွက် attacker ရဲ့ malicious dbghelp.dll က အရင်ဆုံး load ဖြစ်ပြီး execution flow ကို hijack လုပ်နိုင်ပါတယ်။

ဒီအဆင့်မှာ အမှန်ဆိုရင် analyst က host ကို isolate လုပ်ပြီး dbghelp.dll sample ကို upload လုပ်ကာ reverse engineering လုပ်နိုင်ပါတယ်။ ဒါပေမယ့် ဒီ module ရဲ့ သင်ကြားရေးအတွက် “DLL က malicious ဟုတ်မဟုတ် သေချာမသိသေးဘူး” ဆိုတဲ့ အခြေအနေကို ဆက်ယူပါတယ်။ အဲဒီလိုဆိုရင် analyst က additional evidence ကို ရှာပါတယ်။

အဲဒီအတွက် Sysmon Event ID 7 ကို သုံးပါတယ်။ Event ID 7 က module load event ဖြစ်ပြီး process တစ်ခုထဲကို DLL တစ်ခု load လုပ်တဲ့အချိန်ကို မှတ်တမ်းတင်ပါတယ်။ Query က dbghelp.dll load ဖြစ်တဲ့ event တွေကို စုစည်းပြီး signature status ကို ကြည့်ပါတယ်။ ရလဒ်ကို ကြည့်လိုက်တဲ့အခါ system ထဲမှာ load ဖြစ်နေတဲ့ dbghelp.dll အားလုံးနီးပါးက signed ဖြစ်ပြီး legitimate ဖြစ်ပါတယ်။ Notepad++ ရဲ့ installed version က load လုပ်တဲ့ dbghelp.dll ကလည်း signed ဖြစ်ပြီး path က C:\Windows\System32 ဖြစ်ပါတယ်။

```
external_table('Winlog')
| where EventCode == 7 and ImageLoaded endswith "dbghelp.dll"
| project Timestamp, EventCode, Image, Signed, SignatureStatus
```

ဒါနဲ့ ဆန့်ကျင်ဘက်အနေနဲ့ Alice ရဲ့ Downloads folder ထဲက dbghelp.dll က signed မဟုတ်ဘူး၊ system path မှာ မရှိဘူးဆိုတဲ့ အချက်က အလွန်ကို ပြင်းထန်တဲ့ indicator ဖြစ်ပါတယ်။ Legitimate Windows DLL တစ်ခုက user Downloads folder ထဲမှာ unsigned အနေနဲ့ load ဖြစ်နေတယ်ဆိုတာဟာ ပုံမှန်လုံးဝ မဖြစ်နိုင်ပါဘူး။

Test : Indicators

Timeline network connections from the tainted notepad++.exe process. What is the attacker's IP?

ဒီ Indicators အဆင့်ရဲ့ အဓိကရည်ရွယ်ချက်က **attacker နဲ့ ဆက်သွယ်နေတဲ့ network information** ကို ရယူဖို့ ဖြစ်ပါတယ်။ အထူးသဖြင့် malware သို့မဟုတ် attacker-controlled code တစ်ခု run နေရင်

အများဆုံးတွေ့ရတာက command-and-control (C2) server နဲ့ ဆက်သွယ်တာပါ။ ဒါကြောင့် analyst က tainted notepad++.exe process ကနေ **ဘယ် IP ကို network connection လုပ်ထားလဲ** ဆိုတာကို timeline နဲ့ ကြည့်ပါတယ်။

```
external_table('Winlog')
| where EventCode == 3 and Image endswith "notepad++.exe"
| project Timestamp, EventCode, Image, DestinationIp, DestinationPort
```

ဒီအတွက် Sysmon EventCode 3 (NetworkConnect) ကို အသုံးပြုပြီး Image က notepad++.exe ဖြစ်တဲ့ event တွေကို စုစည်းပါတယ်။ Query ရဲ့ ရည်ရွယ်ချက်က notepad++.exe process က ဘယ် အချိန်မှာ၊ ဘယ် IP ကို၊ ဘယ် port နဲ့ ဆက်သွယ်ခဲ့လဲဆိုတာကို တိတိကျကျ မြင်ရဖို့ပါ။ Internal network connection တွေက legitimate traffic ဖြစ်နိုင်ပေမယ့် **external network connection** တွေကတော့ အလွန်ကို အရေးကြီးတဲ့ signal ဖြစ်ပါတယ်။

2024-01-30 01:53:20.941	3	C:\Users\alice\Downloads\npp\npp\not	10.0.0.5	389
2024-01-30 01:42:41.675	3	C:\Users\alice\Downloads\npp\npp\not	10.0.0.5	135
2024-01-30 01:42:41.675	3	C:\Users\alice\Downloads\npp\npp\not	10.0.0.5	49669
2024-01-30 01:42:41.424	3	C:\Users\alice\Downloads\npp\npp\not	174.129.76.246	443
2024-01-30 01:42:40.129	3	C:\Users\alice\Downloads\npp\npp\not	174.129.76.246	8080
2024-01-30 01:42:39.832	3	C:\Users\alice\Downloads\npp\npp\not	174.129.76.246	80

ရလဒ်ကို timeline နဲ့ ကြည့်လိုက်တဲ့အခါ အရေးကြီးတဲ့ pattern တစ်ခုကို တွေ့ရပါတယ်။ notepad++.exe ကို execute လုပ်ပြီး မကြာခင်အချိန်အတွင်းမှာပဲ external IP တစ်ခုကို network connection လုပ်ထားတာကို တွေ့ရပါတယ်။ ဒီလို behavior က user application ပုံမှန် behavior နဲ့ လုံးဝမကိုက်ညီပါဘူး။ Notepad++ က code editor တစ်ခုသာ ဖြစ်ပြီး execute လုပ်လိုက်တာနဲ့ အင်တာနက်ပေါ်ကို automatic ဆက်သွယ်ဖို့ မလိုအပ်ပါဘူး။

ဒါကြောင့် analyst က ဒီ external IP ကို **attacker ရဲ့ C2 IP address ဖြစ်နိုင်ချေ အလွန်မြင့်တယ်** လို့ inference လုပ်နိုင်ပါတယ်။ အထူးသဖြင့် ဒီ connection က tainted process execute လုပ်ပြီးချင်း ဖြစ်လာတာ၊ DLL hijacking theory နဲ့လည်း ကိုက်ညီတာကြောင့် ဒီ IP address ဟာ malware communication endpoint ဖြစ်နိုင်ချေ အလွန်ခိုင်မာပါတယ်။

Test : Back to Winlogon

Which process injected into winlogon.exe?

အရင်ဆုံး context ကို ပြန်ကြည့်ရအောင်။ winlogon.exe ဆိုတာ Windows ရဲ့ core system process တစ်ခုဖြစ်ပြီး user logon, authentication, credential handling တွေနဲ့ တိုက်ရိုက်ဆက်နွယ်နေပါတယ်။ ပုံမှန်အားဖြင့် winlogon.exe ထဲကို user-level application တစ်ခုက thread injection လုပ်တာဆိုရင် အလွန်ကို မသင်္ကာစရာ ဖြစ်ပါတယ်။ ဒါကြောင့် “ဘယ် process က winlogon.exe ထဲကို injection လုပ်ခဲ့လဲ” ဆိုတာကို သိနိုင်ရင် attacker ရဲ့ execution chain ကို နားလည်နိုင်ပါတယ်။

ဒီမေးခွန်းကို ဖြေဖို့ analyst က အရင်က အသုံးပြုခဲ့ပြီးသား technique ကိုပဲ ပြန်သုံးပါတယ်။

> Sysmon EventCode 8 က CreateRemoteThread event ဖြစ်ပြီး process တစ်ခုက အခြား process ထဲကို thread အသစ် ဖန်တီးလိုက်တဲ့အခါ log ထုတ်ပေးပါတယ်။

```
external_table('Winlog')
| where EventCode == 8 and TargetImage endswith "winlogon.exe"
```

Query မှာ TargetImage ကို winlogon.exe နဲ့ filter လုပ်ထားတာကြောင့် “winlogon.exe ကို target လုပ်ပြီး injection လုပ်ထားတဲ့ event” တွေကိုသာ မြင်ရပါတယ်။

```
CreateRemoteThread detected:
RuleName: -
UtcTime: 2024-01-30 00:17:31.437
SourceProcessGuid: {c9af28fc-3a86-65b8-b305-000000000d00}
SourceProcessId: 9896
SourceImage: C:\Windows\System32\conhost.exe
TargetProcessGuid: {c9af28fc-e5f2-65b7-0a00-000000000d00}
TargetProcessId: 776
TargetImage: C:\Windows\System32\winlogon.exe
NewThreadId: 4824
StartAddress: 0x00000225FE980000
StartModule: -
StartFunction: -
SourceUser: NT AUTHORITY\SYSTEM
TargetUser: NT AUTHORITY\SYSTEM
```

Query ရဲ့ရလဒ်ကို ကြည့်လိုက်ရင် winlogon.exe ထဲကို injection လုပ်ထားတဲ့ **source process** ကို တိတိကျကျ သိနိုင်ပါတယ်။

Test : Winlogon SAMR Enumeration

Which tool did the attacker use to enumerate SAMR from winlogon.exe?

အရင်ဆုံး SharpHound အကြောင်းကို နည်းနည်း ပြန်နားလည်ရအောင်။ SharpHound ဆိုတာ BloodHound tool ရဲ့ data collection component ဖြစ်ပြီး Active Directory environment ထဲမှာ reconnaissance လုပ်ဖို့ attackers နဲ့ red team တွေက အများဆုံး သုံးတဲ့ tool တစ်ခုပါ။ Default SharpHound invocation ကို run လုပ်လိုက်ရင် domain users, groups, sessions, ACLs စတဲ့ အချက်အလက်တွေကို စုစည်းပြီး **disk ပေါ်မှာ file အများအပြားအနေနဲ့ သိမ်းဆည်းပါတယ်**။ အများအားဖြင့် .json files တွေ၊ အဲဒီ data တွေကို စုထားတဲ့ .zip archive တစ်ခု၊ နဲ့ execution အတွက် အသုံးပြုတဲ့ cache .bin file တစ်ခုကို ဖန်တီးပါတယ်။ အထူးသဖြင့် SharpHound က cache file ကို `C:\Windows\System32` ထဲမှာ သိမ်းတတ်တာက အလွန် distinctive behavior ဖြစ်ပါတယ်။

```
external_table('Winlog')
| where ProcessId == 776 and EventCode == 11
| project Timestamp, EventCode, Image, TargetFilename
```

အခု investigation ထဲကို ပြန်ကြည့်ရင် analyst က အရင်က winlogon.exe process ကို **injected ဖြစ်ထားတဲ့ malicious process** အဖြစ် သတ်မှတ်ပြီးသား ဖြစ်ပါတယ်။ ProcessId 776 က အဲဒီ compromised winlogon.exe process ကို ကိုယ်စားပြုပါတယ်။ ဒါကြောင့် analyst က “ဒီ winlogon process က SAMR enumeration alert ဖြစ်တဲ့အချိန်နောက်မှာ ဘာ file တွေကို disk ပေါ်ရေးခဲ့လဲ” ဆိုတာကို timeline နဲ့ စစ်ကြည့်ပါတယ်။ ဒီအတွက် Sysmon EventCode 11 (FileCreate) ကို သုံးပြီး ProcessId 776 နဲ့ filter လုပ်ထားတာပါ။

Timestamp	EventCode	Image	TargetFilename
2024-01-30 01:38:33.397	11	C:\Windows\system32\winlogon.exe	C:\Windows\System32\YzlhZji4ZmMtMTU5OC00MjFjLWIzOGUtNjlhODdjNmYwNzMw.bin
2024-01-30 01:38:33.294	11	C:\Windows\system32\winlogon.exe	C:\Windows\System32\20240129203832_BloodHound.zip
2024-01-30 01:38:32.981	11	C:\Windows\system32\winlogon.exe	C:\Windows\System32\20240129203832_computers.json
2024-01-30 01:38:32.980	11	C:\Windows\system32\winlogon.exe	C:\Windows\System32\20240129203832_domains.json
2024-01-30 01:38:32.980	11	C:\Windows\system32\winlogon.exe	C:\Windows\System32\20240129203832_ous.json
2024-01-30 01:38:32.559	11	C:\Windows\system32\winlogon.exe	C:\Windows\System32\20240129203832_users.json
2024-01-30 01:38:32.559	11	C:\Windows\system32\winlogon.exe	C:\Windows\System32\20240129203832_groups.json
2024-01-30 01:38:32.558	11	C:\Windows\system32\winlogon.exe	C:\Windows\System32\20240129203832_gpos.json
2024-01-30 01:38:32.325	11	C:\Windows\system32\winlogon.exe	C:\Windows\System32\20240129203832_containers.json
2024-01-30 01:37:44.769	11	C:\Windows\system32\winlogon.exe	C:\Windows\System32\%LOCALAPPDATA%\Microsoft\Windows\SchCache

Query ရဲ့ ရလဒ်ကို ကြည့်လိုက်တဲ့အခါ SAMR alert ဖြစ်ပြီး မကြာခင်အချိန်အတွင်းမှာပဲ **.json file, .zip file နဲ့ .bin file** တွေကို disk ပေါ်မှာ ဖန်တီးထားတာကို တွေ့ရပါတယ်။ ဒီ timing က အလွန်အရေးကြီးပါတယ်။ Alert မဖြစ်ခင် အချိန်အများကြီး မဟုတ်ဘဲ enumeration ဖြစ်ပြီးချင်းမှာ reconnaissance result files တွေကို ဖန်တီးထားတာဖြစ်ပါတယ်။ ဒီ behavior က SharpHound ရဲ့ default execution pattern နဲ့ တစ်ထပ်တည်းကိုက်ညီပါတယ်။

ဒီအချက်က ဘာကိုဆိုလိုလဲဆိုရင် အရင်က SAMR enumeration alert ကို “suspicious” လို့ပဲ သတ်မှတ်ထားရာကနေ အခုအချိန်မှာတော့ **အတည်ပြုထားတဲ့ malicious reconnaissance activity** ဖြစ်သွားပါပြီ။ Winlogon.exe ဆိုတဲ့ highly privileged system process ထဲမှာ inject လုပ်ထားတဲ့ attacker code က SharpHound ကို run ပြီး domain reconnaissance လုပ်ခဲ့တာကို file-level artifact တွေနဲ့ ထောက်ခံနိုင်သွားပါပြီ။
