Detecting LDAPFragger — A newly released Cobalt Strike Beacon using LDAP for C2 communication

understand it, test it in a lab and find ways to detect it. Keeping up with knowledge about attack vectors and corresponding detections for it is very important one of your required steps to stay ahead.

> *Understanding attack vectors (used for initial access) and communication channels (used for command and control) according to the MITRE ATT&CK Matrix by hacking tools is one of your fundamental knowledge for building new detection rules. Creating yet unknown or not public available detection rule from scratch will allow you to put up an early light bulb (alert) and catch attackers using even early adopted techniques, or maybe even techniques with — which*

First I went through the sourcecode on Github to understand how it is operating and how it does work. I tend to read the sourcecode first before moving to reading through the explaining blog-post, as this will be give me a rough understanding, about what the tool is doing and how they archieved this (progammaticaly-wise). This requires that you have a good understand of programming skills —a fundamental skillset for blueteamers. In my opinion every blueteamer should have atleast some coding experience. **Reading the blogpost afterwards explains a lot of sentences automatically to you, while technically understand, how it is done.** I will not go into details here, but the main part is that C2 communication is done via Cobalt-Strike

- Read the raw **LDAP- Query** sent to the LDAP-server from the DC
- Needs to **read and write LDAP data** (Windows **Event 5136**), this information is on the domain-controller
- **New Detection possibilities** — A secret gem at the end of the blog (keep reading!)

The more you work with Event IDs (Windows, Sysmon etc.) the more you will be able to straight see these events being thrown just by reading some code. This will

you get a greater visibility. **If you baseline the amount of TGS-requests per each User in your Domain, you know your average amount of Kerberos Ticket requests, you will catch most Bloodhound runs** (note: with default settings). These kind of detection rules are behaviour-based and tend to be a lot of more work (since they need to be especially be designed for your domain/company, amount of users, kerberos ticket-traffic, but play out well in the end. Start easy, work your way up to more complicated rules later. Lets get back to finding something simple for LDAPFragger.

Detecting LDAPFragger by Sysmon Event ID 1/Windows Event 4688 (Process

Sysmon EventID 3 for Port 389 from LDAPFragger.exe

best done by using an EDR which is permanently doing some scanning in the background and checking the threads/process communication/memory regions etc. With Windows Events alone (ETW) we mostly can not do much, if the attacker is good.

Read through all the blogs and tools from

@_xpn_, @hexacorn, @mattifestation, @_RastaMouse

is highly recommend in the area of DLL Injections / .NET / CLR to

Detecting LDAPFragger by Sysmon Event 17 (Named Pipes)

A named pipe will be used for transferring data, this may be important, but this is hard to baseline and most clever attackers today randomize the pipename, so its really hard to make use of. **You will catch some lazy attackers with fixed pipe-names though**, who did not change the name in their beacon or not using a random pipename.

For example:
https://github.com/Neo23x0/sigma/blob/master/rules/windows/sysmon/sys

my first test run of ldapfragger

As you can see, LDAPFragger connected to my Cobalt Strike Team Server

example **how to enable LDAP querry logging on a DC:** https://directoryadmin.blogspot.com/2019/10/hunting-bad-ldap-queries-on-your-dc.html — figure out — its was never meant to be used for detection or security effects. **It will massivly slowdown your complete AD.**

During the start of LDAPFragger you can see **several hard-coded LDAP querys** in the tool which then could be used as an indicator for detection (IOC) or usage of this tool in your SIEM by reading the raw LDAP querys sent to the server. The same approach Microsoft suggested in its blogpost. Though, most of these LDAP filters do not look harmful/malicious. It would

Windows Event 5136 caused by LDAPFragger

To make this event work on your domain-controllers you need to make sure your Advanced Audit Policy is setup correctly:

```
index=wineventlog EventID=5136 AND AttributeValue=”*”
AND AttributeLDAPDisplayName IN (“primaryInternationalISDNNumber”,
“otherFacsimileTelephoneNumber”, “primaryTelexNumber”)
```

LDAP-Attribute-Names, which are valid and commonly used in your environment. This way, even in a small SIEM, you will be able to improve your detection even further (e.g. maybe even you are able to completely whitelist all known LDAP-attributes) — **now you are able to catch any phishy LDAP-attribute which the attacker may try.**

Additionally **we could be possible looking for base64-content within AttributValue settings**, as that would be suspicious and unusual. **Check the picture on top for the suspicious base64 content in the 5136 event.**
Depending on the size of your SIEM you will be able to do this **for all**

This is a more mature detection possibility by leveraging EDRs or Sysmon on endpoints.

I noticed when the LDAPFragger is being executed, Windows will create an "**Active Directory Schema Cache File**" with TargetFilename = "**C:\Users\<User>\AppData\Local\Microsoft\Windows\SchCache\<DomainFQDN>.sch**".

I quickly ran a hunting-query against the production enivornment, to see how many false positives it would create and I was astonished. Not much at

```
index=sysmon EventID=11 SchCache
AND TargetFilename =
“*\\Local\\Microsoft\\Windows\\SchCache\\*.sch”
AND NOT Image IN (“C:\\WINDOWS\\system32\\svchost.exe”,
“C:\\WINDOWS\\system32\\DllHost.exe”,
“C:\\WINDOWS\\system32\\mmc.exe”)
| table _time Computer Image TargetFilename
```

SIGMA-Rule:

I was able to baseline this query well and I was very satisfied with this found, as it seems to be a good thing to share with the InfoSec community and other blueteamers. **Read more about the ADSI cache directory here:** https://docs.microsoft.com/en-us/windows/win32/adsi/adsi-and-uac

Further research seems to harden the fact, that this ETW behaviour comes from the usage of System.DirectoryServices.DirectorySearcher in tools — which is also used by LDAPFragger. This expains, why the ADSI cache file is created. So this “could” be an suspicious behaviour by LDAPFragger for example

Threat Hunting | Threat Intelligence | Windows | Events | Blue Team