Detecting LDAPFragger — A newly released Cobalt Strike Beacon using LDAP for C2 communication (blueteamers approach)

Iveco · Follow
14 min read · Mar 24, 2020

Goals:

- trying to understand how the tool works
- trying to find further detections as blueteamer for an newly released tool
- find an efficient detection rule which can be implemented in large scale environment quickly

After LDAPFragger (introduction blogpost) has been at Github (03–19–2020) I immediately took some views into its capabilities and design. When a new tool for red-teamers has being released, our job as blueteamer is to get up, 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 were known before (by harmj0y) — but are now improved and maybe used in a different way. That could lead to new techniques being born, improving existing tactics. Always try to be one step ahead!*

As the official blog-post at https://research.nccgroup.com/2020/03/19/ldapfragger-bypassing-network-restrictions-using-ldap-attributes/ gave several insight points on how LDAPFragger works, it would be worth testing it in a lab environment myself — as it seem to create some ETW noise — as even the author explained. Any

data or Windows event we can catch up for our SIEM is a good thing, the more light bulbs (alerts/detections) we define, the more we can explain and correlate to later to — in the end even combine this with your own risk-scores, draw your own conclusions on the way of building the final MITRE attack chain for your best detections. So its worth having any many little detections, as small as they shine, in total they may give you different big picture. So lets try to build some detections for LDAPFragger, since at this of writing the story, no public information except from the blog-post about detections for this tool is available.

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 beacons while data is being routed using LDAP-Attributes by LDAP/LDAPS connections to Domain-Controllers within the Active-Directory. This requires any **authenticated user** — which is than able to do LDAP connections and reading/writing/clearing LDAP-AttributeValues back and forth, using the AD as dumpster for C2. The tool requires code execution rights though or an valid compromised useraccount first. But I am still looking for an detection that would look after an initial successful execution, as you never can trust your systems before, always have something for any use-case up and running in your SIEM.

From reading the source-code I immediately saw possible Event IDs to look for:

- **Process Creation** (Sysmon **Event ID 1** or Windows **Event 4688**)
- Needs to do LDAP(S) **network connections** (Sysmon **Event ID 3**)
- **CreateRemoteThread()** (Sysmon **Event ID 8**)
- Data is transferred via NamedPipes (Sysmon **Event ID 17**)
- 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 allow you to quickly imagine first tries to build your rule. **As the first step you should start with simple rules.** You can work on more advanced detection rules (like behaviour-based ones) later — after doing detailed tests. **These tend to be the ultimate goal as bluetamer**, as they are hard to bypass for an attacker. Having multiply mixed-cases of dections ready for the same tools, techniques and procedures could lead to a multiplied, concluded detection or aggregrated resultset. For example take Bloodhound/Sharphound. There are a lot of possible detection methods available for this tool (by process creation, by loaded DLLs, by used parameters in CommandLine), but hardly anyone tells you, that it could also be detected by behaviour. Of course you start with simple rules like Event ID = 1/4688 for tools, PE-Header != TargetImageName, maybe even IMPHASHES, loaded DLL-Sets, but that can be easily bypassed by an attacker. *The solution for an behaviour-based Bloodhound/Sharphound detection rule would be something different.* It is Windows Event ID 4769. Did you ever run a klist.exe after your Bloodhound run? Now you understand, what I mean. Again, clever attackers also know this, they do not do the complete service enumeration, they do it slowly etc. If you understand the tool, and the possibilites you can write multiply rules for the same tool and combined 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 Creation)

Nothing to say here, easily to be bypassed and does not make much sense for an tool like this, it will be specifically used in combination with Cobalt-Strike.
There is an high chance an attacker will do atleast some kind of customizations/obfuscations/blendins.

Detecting LDAPFragger by Sysmon Event ID 3 (Network Connections)

Since the C2 uses LDAP as communication channel the following network connections are done:
Victim → LDAP → Port 389 → DC
Victim → LDAPS → Port 636 → DC

The **Image** field name in the event will reveal the LDAPFrogger process.

Sysmon EventID 3 for Port 389 from LDAPFragger.exe

The problem is here, yeah guess… **It might not be so unusual for workstations or servers to do some LDAP/LDAPS communication.** In my opinion this is no good idea and does not make much sense to follow this approach further (huge noise, a lot to be baselined, low value).

Detecting LDAPFragger by Sysmon Event ID 8 (CreateRemoteThread)

I noticed LDAPFragger makes use of **CreateRemoteThread()** and **CreateProcess**() to hollow the Windows **notepad.exe (in default, unmodified source-code).** We would be able to detect CreateRemoteThread() calls by Event ID 8, but we will not know the **TargetImage** in the event. The same is for the other injection method, which is done to **notepad.exe** on default, but can of course be **easily changed.**
CreateRemoteThread() detection rules are very often used and very common. But the problem is, that this is nothing unusual in the Windows world. So detecting a malicious thread from a normal thread is a really a huge problem, if you don't know how the process flow of this machine is normally — or you don't have any additionaly heuristic running. Detecting injections is 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 unmanaged processes injections / spoofing many process information — they permanently come up with new ideas, methods and techniques. Mindblowing stuff — must read!

I did not find a satisfying solution to detect LDAPFragger using Sysmon EID 8. A good cheatsheet for Sysmon to detect Win32 API Calls is can be found here. Original credits go here https://github.com/jsecurity101/Windows-API-to-Sysmon-Events/blob/master/README.md

It would be worth digging further into this:

an attacker not reading the sourcecode may fall into default detection traps

Blueteamers always need to find more sophisticated solutions, to even catch these attackers, **which are really good and even improve tools further and modify their sourcecode especially for the targeted operation.**

It was crazy how many attackers you were able to spot using some default pipenames from sourcecodes of Cobalt-Strike SMB beacons (using default PipeName msagent_*). Always think about the lazy attackers and the APT actors. Its worth having different set of rules available to catch multiply stages of the same usecase. But thats an own story on itself.

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/sysmon_mal_namedpipes.yml

and another good one is here: https://github.com/Neo23x0/sigma/blob/master/rules/windows/sysmon/sysmon_apt_turla_namedpipes.yml

These are good starting point if you want to dig further into this area (detecting beacons by named pipes). **So with hard work, or a good baseline of known pipe names in your company, also Sysmon Event 17 for Pipe created could be used for detection.**

But:

my first test run of ldapfragger

As you see here LDAPFragger connected to my Cobalt Strike Team Server and used a **random pipename. Grabbing windows pipenames and baselining these in your company is a own medium story on its own.** I did use the default LDAPfragger and CS-Teamserver version without huge modifications, so in my opinioning detecting LDAPfragger using Pipes Created (Sysmon Event 17) is no good solution, if you not specialized on this area before (different story on its own).

Detecting LDAPFragger by reading the raw LDAP- Query sent to the LDAP-server from the DC

The best would be as blue-teamer, to be able to see LDAP-Querries / LDAP search filters sent to the LDAP server. But this will not work on huge companys, because the LDAP server event generation would be too noisy. So it is mostly disabled. Microsoft Defender ATP could help with this, see this blogpost: https://techcommunity.microsoft.com/t5/microsoft-defender-atp/hunting-for-reconnaissance-activities-using-ldap-search-filters/ba-p/824726 but some may not be able to do so. To activate LDAP- querry logging you have to enable the debug mode for LDAP on the DC, see here for 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 require some effort to detect this tool behaviour using its LDAP querries alone. If you enable LDAP querries logging, take a look at the sourcecode. A lot of hard-coded LDAP-querries can be found in the tool, this could be — in combination — reveal for sure particular patterns and allow for alerting. Getting LDAP logs especially will help you with all the detections on the Microsoft blogpost e.g. detect Bloodhound/Sharphound **because they have some really, obvious ugly hard-coded LDAP querries/LDAP-filters**), but that requires you to have all that logging enabled. There is also recon via LDAP (DNS names etc.) available for attackers. LDAP gets more and more interesting today. **Most companies don't have these logs cause the performance impact is huge.** Therefore I did not find this as a good solution.

Detecting LDAPFragger by Event ID 5136 — A directory service object was modified

So I was going to look further for **Event ID 5136** as this was also correctly mentioned by the author in the blogpost when introducing LDAPFragger. This sounds like a reliable Windows event.

When the attackers does not change these attributes, we can use these as IOCs for LDAPFragger:

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:

Advanced Audit Policy: DS Access
Subcategory: Audit Directory Service Changes
With State: **Success**
Event ID: 5136 A directory service object was modified
OSSEM Event Template
ETW: Microsoft-Windows-Security-Auditing
Channel: Security
Note: Windows **Server** will have this audit policy enabled by default (success only), while when you install the Security Baselines from Microsoft this will be set to failures only. **You need to have the success events of 5136.** So there is a chance that you will be able to do this already on your domain-controllers within your production environment.

Checking the **DC Winevent-Log** for **EventID 5136** and a set **AttributeValue**, matching any of the **LDAPAttributeNames** listed in the blog on the top, **seems to be a valid solution.**

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

example SIGMA rule for LDAPfragger using EID 5136

Attackers would be able to encounter this detection, by using different attribute names for the initial discovery process of which attribute to chose for initial data exchange. Most companies though will not make make use of these default LDAP attributes (as the author mentioned) — so this detection will work on a large scale to find lazy attackers atleast — when they did not change the sourcecode or thought of different, non-used LDAP attributes which could be used for initial exchange of data. Note: Already some further LDAP attributes have been mentioned on Twitter, which could be better used, and more stealthy in some environments.

So again you can go one step further and create a **list of valid AttributeLDAPDisplayName** used in your company the past 90 days (e.g. search backwards). From this result-set create a whitelist of acceptable 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 LDAPAttributes** (e.g. make use of Splunk shanon). This requires large processing power and Splunk knowledge.

I recommend setting up this up as alert, when your company is not using these LDAP-Attributes, or even make use of the whitelisting approach for LDAP-Attribute-Names (look backwards, what is known and valid).

> After some more digging I noticed something different, that lead me to writing this story at all.

Detecting LDAPFragger by a secret gem — the ADSI cache

After doing some splunking and data-sciene with the data-set created by the tool **I noticed one event**, worth taking a further look into.

Sysmon Event ID 11 (FileCreate) will be thrown when a file is being created and you are monitoring the path accordingly to your sysmon-config.

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 all, and I ran this search 365 days backwards. It seems to be a great indicator! we will learn about this directory a bit more later. You will get noise by Microsoft tools like “C:\\WINDOWS\\system32\\svchost.exe”, “C:\\WINDOWS\\system32\\DllHost.exe”, “C:\\WINDOWS\\system32\\mmc.exe” — this is acceptable, but why is it and is it notable? I noticed it also catches a lot of powershell.exe users, who seem to be remotely working with the Active-Directory. So we need to find out, why this is happing and what is going on here. But when removing the Windows-Noise and the known powershells, all left was 1 True-Postitive for LDAPFragger.exe. To make use of Sysmon Event 11 you must make sure the path mentioned is in your scope, so events are generated.

two examples sysmon config for E11 to catch AD-schema cache files

Splunk-Query:

```
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:

example SIGMA rule for LDAPfragger using EID 11 (edit: selection2 should actually be Image, sry)

Will reveal **LDAPFragger.exe usages** and people using different other tools like powershell, **working with ADSI operations.**

catching LDAPFragger via Event ID 11 from Sysmon

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.

Final conclusion for blueteamers:

1. **Windows Event ID 5136** will help you with lazy attackers, who will not further invest money and ressources to improve the toolset. It is recommend, to set up an alerting on the 3 default LDAP-attributes mentioned in the blog, which the tool will use by default. Further improve this alert by using the whitelisting approach of known LDAPAttributes for example.
2. Setting up a second detection for **Sysmon Event ID 11** in folder C:\Users\<Username>\AppData\Local\Microsoft\Windows\SchCache\<DomainFQDN>.sch and using a whitelist seems to give another good IOC of the tool LDAPFragger.exe possible being used

A story written by xknow_infosec (@xknow_infosec) — 24.03.2020
https://twitter.com/xknow_infosec

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

 --

Written by Iveco

67 Followers

Follow