How we detected LockerGoga

Guillaume Bonfante[1,2], Corentin Jannier[2], Jean-Yves Marion[1] and Fabrice Sabatier[3]

[1]LORIA - Université de Lorraine

[2]CYBER-DETECT

[3]CNRS

Abstract

Our objective is to illustrate the uses of the software GORILLE that we develop at CYBER-DETECT. The recent attacks of LockerGoga against Altran in France and Norsk Hydro in Norway illustrate the necessity to have advanced anti-malware defences. GORILLE's basis are morphological analysis. As such, the main features of GORILLE are the following. It is robust with respect to heavy code obfuscations. It applies on dynamic data that can be forged within a virtual environment. Its detection engine is based on behavior recognition.This contribution is an extended version of our Blog's post[1].

`LockerGoga` is a malware that targeted two major companies at the begining of 2019. The first one is Altran in France [4] while the second one is Norsk Hydro [3]. The "success" of these two attacks show the need of new detection techniques. GORILLE is such a tool. It is now developed by CYBER-DETECT following research in morphological analysis at LORIA [7, 8]. The attack in France happened in January and the one in Norway in March. Those attacks should have been stopped. Indeed, the GORILLE engine does the job. It detects LockerGoga and its variants as we will show it.

In a nutshell, GORILLE identifies malicious threats embedded in Linux, MacOS and Windows binary files. For this, GORILLE keeps a collection of malicious behaviors. Each binary file submitted to GORILLE is then scanned and as soon as a set of malicious inter-link behaviors is detected, GORILLE raises an alert. There is no magic behind, just several years of hard work at Loria's Computer Science Lab.

[1]See `http://www.cyber-detect.com/fr-blog.html`.

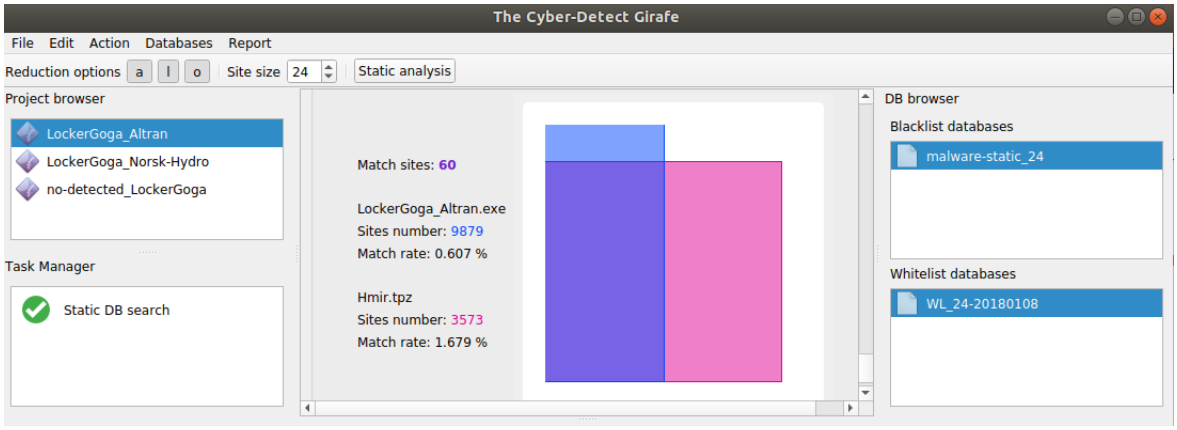
What is a behavior? Hard to answer the question in general. In few words, we built an abstract view of the control flow graph from which we extract some chunks. For instance, for the `Zeus` malware, one of the behavior corresponds to the execution of orders by a Command and Control. This is illustrated by this snippet:

```
static WORD executeScript(LPWSTR scriptText, LPDWORD errorLine)
{
  WORD errorMessageId = 0;
  LPWSTR *lines;
  DWORD linesCount = Str::_splitToStringsW(scriptText, Str::_LengthW(scriptText), &lines, Str::STS_TRIM, 0);
  //*errorLine = (DWORD)-1;

  if(linesCount == (DWORD)-1)
  {
    errorMessageId = CryptedStrings::id_remotescript_error_not_enough_memory;
    WDEBUG0(WDDT_ERROR, "_splitToStringsW failed.");
  }
  else
  {
    //Ïåðåáèðàåì ñòðîêè.
    for(DWORD i = 0; i < linesCount; i++)if(lines[i] != NULL && lines[i][0] != 0)
    {
      LPWSTR *args;
      DWORD argsCount = Str::_getArgumentsW(lines[i], Str::_LengthW(lines[i]), &args, 0);
                                            ^^^^^^^^^^^^^^^
```

1 The first detection

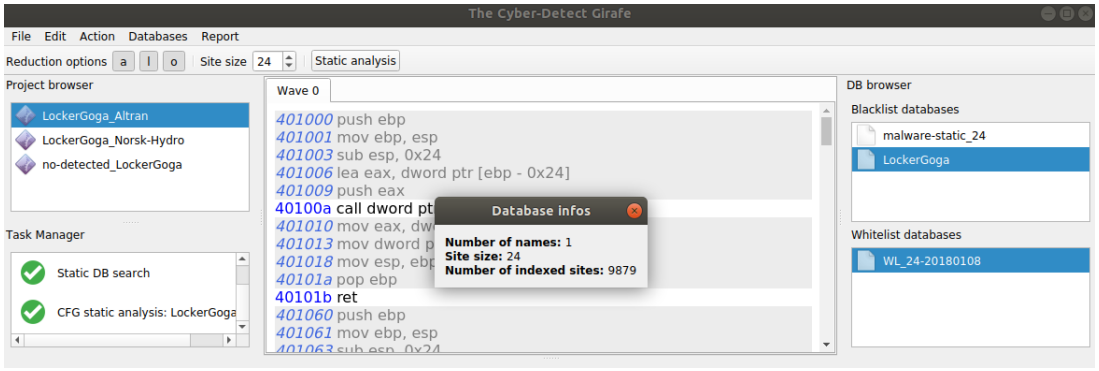
Since Gorille search process is based on a collection of malicious behaviors, the first question which comes in mind is whether or not Gorille is able to detect LockerGoga. Gorille knows 32 812 355 malicious behaviors. And Gorille identifies 60 malicious behaviors in the submitted sample of LockerGoga. Actually, and we did it for fun, we used for the experiments our old malware database dating from 2013. And six years later, it is still up to date! As far as we know, LockerGoga did not exist in 2013. More precisely, as we could see from the version of the libraries code that is statically linked (see Section 2), the sample that we have has been compiled after 2018. But, nevertheless, it reuses some much older components, some of them dating from before 2013. Thus, as a partial conclusion, Gorille doest not need frantic malware database updates. Non solum it sees variants of old attacks, sed etiam it catches new ones whenever they reuse old stuff.



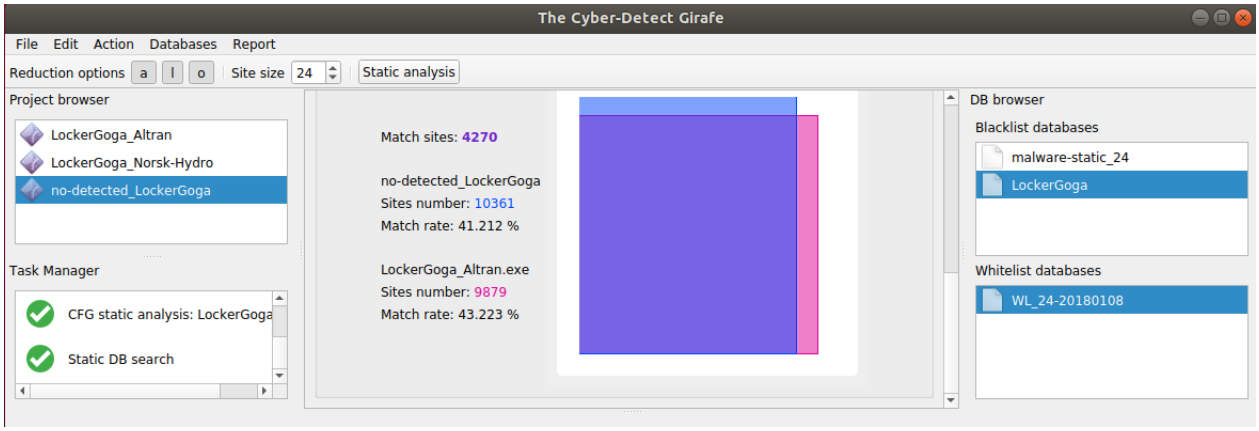
The sample named LockerGoga_Altran [6] corresponds to the malware that attacked Altran in January 25th, 2019. On March 8th, 2019 that is two months later, **MalwareHunterTeam** discovered in [2] that a variant of LockerGoga, that we name here no-detected_LockerGoga was left undetected by all anti-virus products in Virus Total [5].

As we see, GORILLE detects 60 malicious behaviors in the yet undetected sample of LockerGoga, which are identical to the previous identified ones! The technological advance of GORILLE allows to stop variants of unknown threats.

Actually, we can play with GORILLE a little bit more. Indeed, GORILLE is able to learn the specific malicious functionalities of LockerGoga by itself. First, we built the "LockerGoga" specific database which contains the 9879 behaviors (also mentioned as sites) in LockerGoga, not only bad ones. Indeed, LockerGoga incorporates, as usual in any software, third party libraries coming from Microsoft and open source libraries. Collectively, third party libraries also denote behaviors.



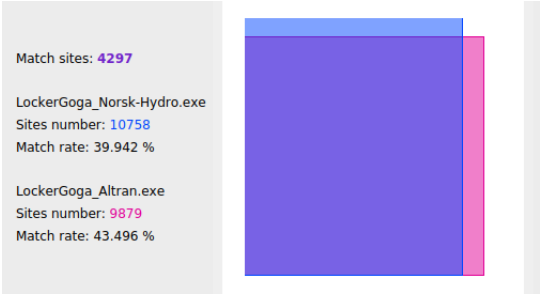
That said, we can compare all behaviours of LockerGoga_Altran, which were involved in Altran incident and the no-detected_LockerGoga. of MalwareHunter. There are 4270 common behaviours, roughly the half, that are common between bot samples.



Then, using our tool `binsim` from the GORILLE suite, we can synchronize both codes, that is to find the precise correspondence between functions of LockerGoga_Altran and no-detected_LockerGoga.

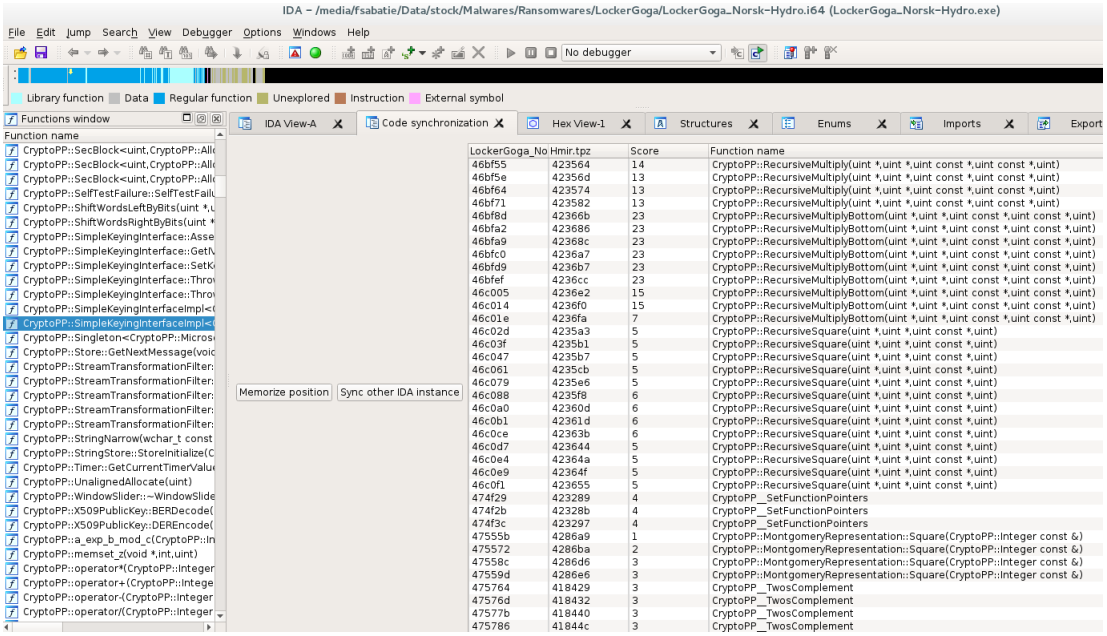
2 Other forms of the attack

After Norsk Hydro attack, we wanted to recognize the malware. Since, the detection of this version involves exactly behaviors of the Altran's attack, this is a strong clue that there is a relationship between the two samples. After comparison with the LockerGoga database, no doubt that both are very close.



Let us add few words on the retro-engineering of LockerGoga. We learnt from [1] that LockerGoga is using CryptoPP, a cryptographic functions library. Let's go. First, we learn CryptoPP and then, we use the function

matching engine of GORILLE to simplify the IDA view[2]. Functions within LockerGoga are automatically labeled with CryptoPP library's names.



3 Dynamic analysis

Up to this point, everything was done statically. But, GORILLE can take benefit of our dynamic analysis framework. It is based on DYNAMORIO[3] with special effort to make it transparent to anti-virtualization techniques. And LockerGoga use some of them. There are suspicious `cpuid` instructions. But, our tool also observed a "SystemKernelDebuggerInformation" that is clearly a protection. There is also call to "OutputDebugString", but that one serves for other purposes.

Second point, our tool did not see any self-modification tricks. Thus the static analysis of the file is sufficient. Again, that information saves so much time for the retro-engineer. Nevertheless, the execution of the malware is obfuscated. The main process launch some sub-processes that serve to hide/encrypt data. For instance we can read within the execution trace the following call:

[2]Here we use the famous disassembler IDA from Hexrays, but any other tool could be used.

[3]A dynamic analysis tool. See `https://github.com/DynamoRIO`.

```
0x00a1e492 call [0x7692103d]  WINAPI CreateProcessW(
_In_ [0x0044e00c] "C:\Windows\system32\cmd.exe"
_In_Out_ [0x0044e010] 0x004d7f18
_In_ [0x0044e014] 0x00000000
_In_ [0x0044e018] 0x00000000
_In_ [0x0044e01c] FALSE
_In_ [0x0044e020] 0x00000000
_In_ [0x0044e024] 0x00000000
_In_ [0x0044e028] NULL
_In_ [0x0044e02c] 0x0044e09c
_Out_ [0x0044e030] 0x0044e110
)
Return TRUE
```

Yes, we follow sub-processes and threads. Our tool reveals these hidden behaviors.

4 Conclusion

All right, Gorille sees LockerGoga. Does it mean it will discover every malware? No, of course not. But, it clearly sees (some) malware that others don't see. We think that a panel of detecting engine using different technologies is much stronger than a simple anti-virus software and want to contribute to this aim.

What are the limit of the system? Theoretically, malware recognition is undecidable. In practice, with our dynamic tool which runs within Windows, it is not that easy to escape Gorille. We still need to work on other Operating Systems such as Linux or MacOS and naturally Android and other IOT systems.

References

[1] https://labsblog.f-secure.com/2019/03/27/analysis-of-lockergoga-ransomware/.

[2] https://twitter.com/malwrhunterteam/status/1104082562216062978.

[3] https://www.bbc.com/news/technology-47624207.

[4] https://www.bleepingcomputer.com/news/security/new-lockergoga-ransomware-allegedly-used-in-altran-attack/.

[5] https://www.virustotal.com/en/file/eda26a1cd80aac1c42cdbba9af813d9c4bc81f6052080bc33435d1e076e75aa

[6] https://www.virustotal.com/#/file/bdf36127817413f625d2625d3133760af724d6ad2410bea7297ddc116abc268

[7] G. Bonfante, J. Fernandez, J.-Y. Marion, B. Rouxel, F. Sabatier, and A. Thierry. Codisasm: Medium scale concatic disassembly of self-modifying binaries with overlapping instructions. In *22nd ACM Conference on Computer and Communications Security*, 2015.

[8] G. Bonfante, J.-Y. Marion, and F. Sabatier. gorille sniffs code similarities, the case study of qwerty versus regin. In *Malcon 2015*, 2015.