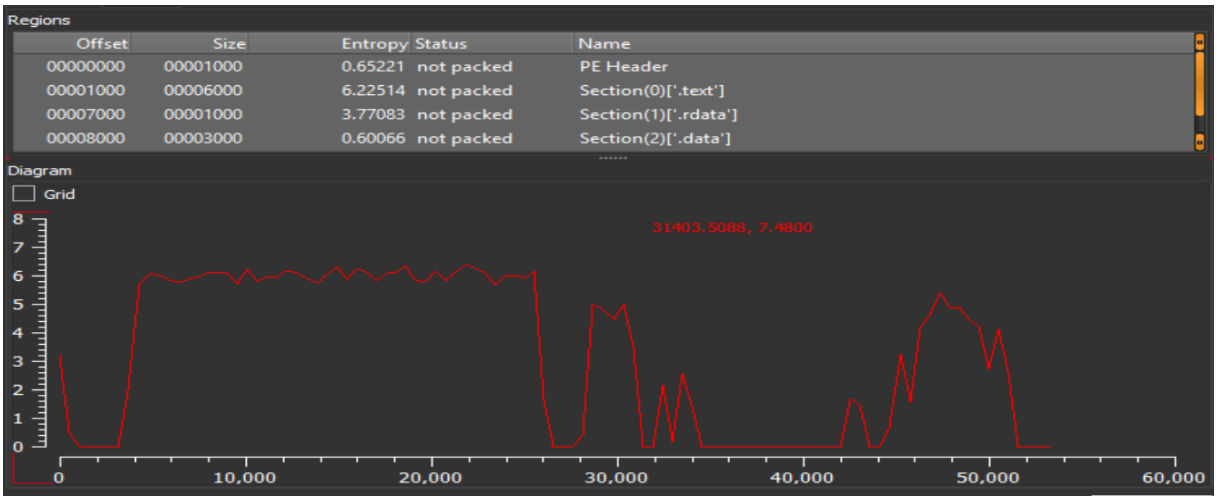
Lab 11-01

Analyze the malware found in Lab11-01.exe.

We are given file named “Lab11-01.exe”.

Malware written in C/C++, entropy 4.49, section sizes seem normal as the strings which are clearly readable. Looks like there is no protection against static analysis.



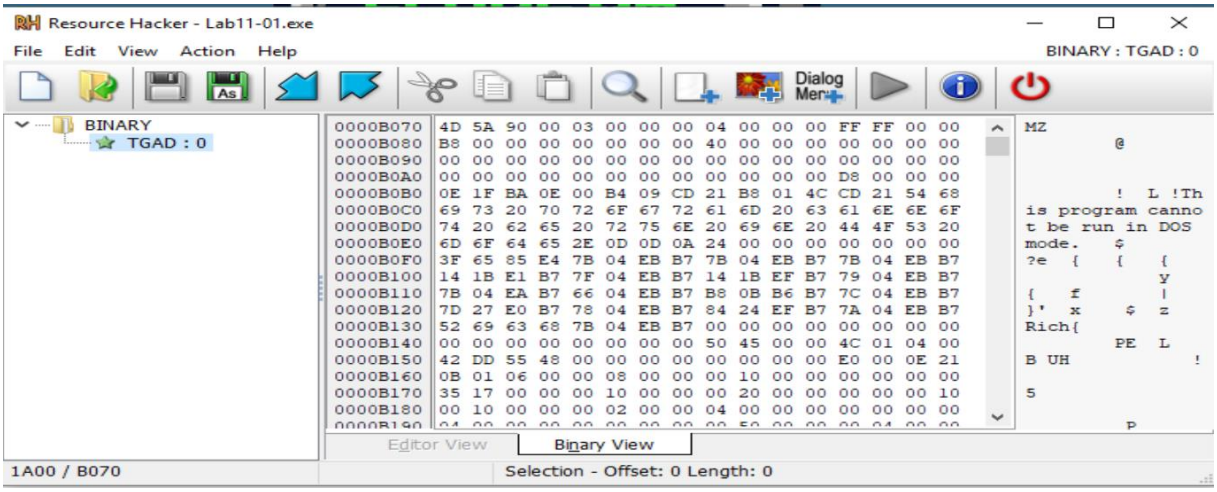
When it comes to string analysis, here are the most interesting results:

- \MSGina
- GinaDLL
- MSGina.dll
- gina.dll
- UN %s DM %s PW %s OLD %s
- %s %s - %s
- ErrorCode:%d ErrorMessage:%s.
- msutil32.sys

Based on the included imported functions it has themes of:

- Registry manipulation
- Resources management
- File operations (read/write)
- Dynamic library loading
- File Memory management

When it comes to new things, to my surprise it looks like the executable holds another file in its resources:



The file characteristics indicate that the file is a dynamic link library (DLL), which might be used by the executable after resolving resources and dropping the file.

The first thing I notice is the presence of DllMain information. This suggests that the file hidden in the resources is indeed a .dll file. After resolving the resource with Resource Hacker, I inspected it in IDA. The .dll file is intended to mimic the original behavior of a Windows library called “msgina.dll”.

The DLL entry point involves finding the original library and obtaining a handle to it.

The remaining functions redirect calls to the original ones, except for one: **WlxLoggedOutSAS**. This function is invoked when the user logs out.

After the redirect, the malware saves the data into a newly created file, “msutil32.sys,” in the system directory. The logged data follows the format "UN %s DM %s PW %s OLD %s".

```
.text:100014F6          mov     ecx, [esi+4]
.text:100014F9          push    edx
.text:100014FA          push    ecx
.text:100014FB          push    eax             ; Args
.text:100014FC          push    offset aUnSDmSPwSOldS ; "UN %s DM %s PW %s OLD %s"
.text:10001501          push    0               ; dwMessageId
.text:10001503          call    sub_10001570
.text:10001508          add     esp, 18h
```

Below is a documented structure from Microsoft that should identify what each shortcut (UN, DM, PW, OLD) means.

typedef struct _WLX_MPR_NOTIFY_INFO {

PWSTR pszUserName;

PWSTR pszDomain;

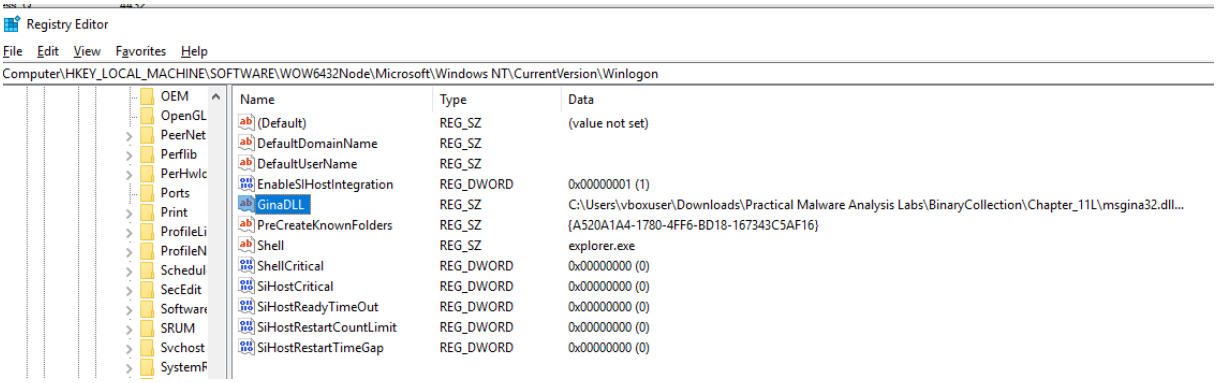
PWSTR pszPassword;

PWSTR pszOldPassword;}

Let's inspect the original file "Lab11-01.exe" more deeper.

The first code logic is intended to get a handle to its own file, find the resources "TGAD", allocate the data using VirtualAlloc and resolve it under a name "msgina32.dll" at its own working directory, and output "DR" at console.

Then it tries to access a registry key under **HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL** to set a value to a path to newly resolved resource PE file "msgina32.dll".



By doing that, it achieves persistence on the OS, because each time the Winlogon catches an event, it's going to talk to infected msgina32.dll first. After success, it prints another mysterious line "RI" to the console.

Questions

1. What does the malware drop to disk?

The malware drops a file “msgina32.dll” from resources under its own directory.

The resource is called TGAD.

2. How does the malware achieve persistence?

Malware achieves persistence by saving its fake msgina32.dll file path to HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL so each time the WINLOGON process gets triggered, it calls the malware dll that redirects the call to the original one.

That way whenever an WINLOGON events happens, the dll will run.

3. How does the malware steal user credentials?

The malware steals user credentials by exploiting the built-in GINA (Graphical Identification and Authentication) interface, which was used in older versions of Windows such as NT 3.51, NT 4.0, 2000, XP, and Server 2003.

It does this by inserting its own infected DLL into the system registry, effectively redirecting function calls. The malicious DLL mimics the original GINA DLL to avoid disrupting the process.

4. What does the malware do with stolen credentials?

When the WlxLoggedOutSAS function is called, the malware intercepts and steals the provided credentials, then saves them in a file named msutil32.sys.

5. How can you use this malware to get user credentials from your test environment?

As deducted earlier, the malware saves stolen data only when WlxLoggedOutSAS gets called, so to see it’s behavior we would have to log out from the OS, log back in, then check the newly created file msutil32.sys hidden at system directory.