«Секретики» DPAPI или DPAPI для пентестеров / Хабр

habr.com/ru/articles/434514

karelovao

Вторая статья по итогам выступления нашей команды на OFFZONE-2018. На этот раз рассмотрим доклад с MainTrack “Windows DPAPI “Sekretiki” or DPAPI for pentesters”.

Внимание! Очень много буков!

При проведении RedTeam кампаний хочется давать меньше поводов для реакции BlueTeam, но их может быть много. Например, запуск mimikatz для получения пользовательских паролей или сертификатов. Даже если мы сумели «отмазать» его от Касперского, у BlueTeam есть возможность отслеживания с помощью специализированных средств, таких как Sysmon, Microsoft ATA и т.д. В тоже время хотелось бы получить максимум информации со скомпрометированной машины пользователя. В ходе неоднократно проведенных RedTeam кампаний с противодействием настоящим BlueTeam командам мы пришли к выводам, что необходимо в максимально избегать действий, которые могу служить индикаторами компрометации системы. Достигнуть эту цель возможно с помощью использования легальных механизмов и действий, предусмотренных операционной системой для пользователя.

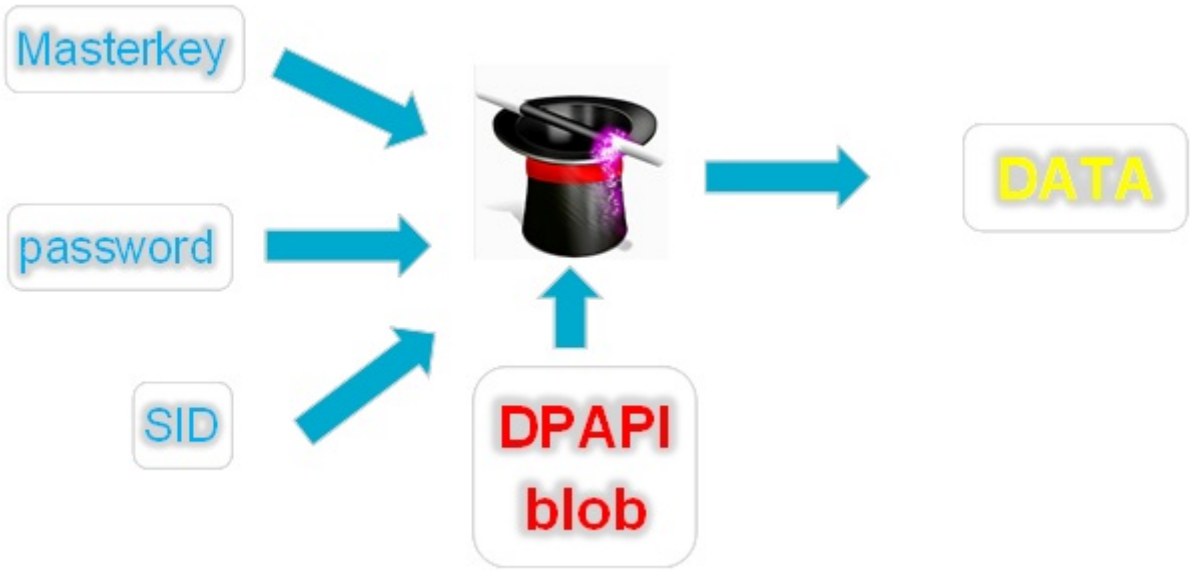
Одним из таких легальных инструментов является механизм DPAPI (Windows Data Protection API), который используется операционной системой и различными приложениями для шифрования чувствительных данных пользователя (прежде всего паролей, криптографических ключей и т.д.) Для конечного пользователя и его приложений DPAPI выглядит предельно просто: есть всего 2 функции – «зашифровать данные» и «расшифровать данные». В данной статье хотелось бы рассмотреть, насколько такой механизм полезен пентестерам при проведении RedTeam кампаний.

Что такое DPAPI? Только кратко и по-русски

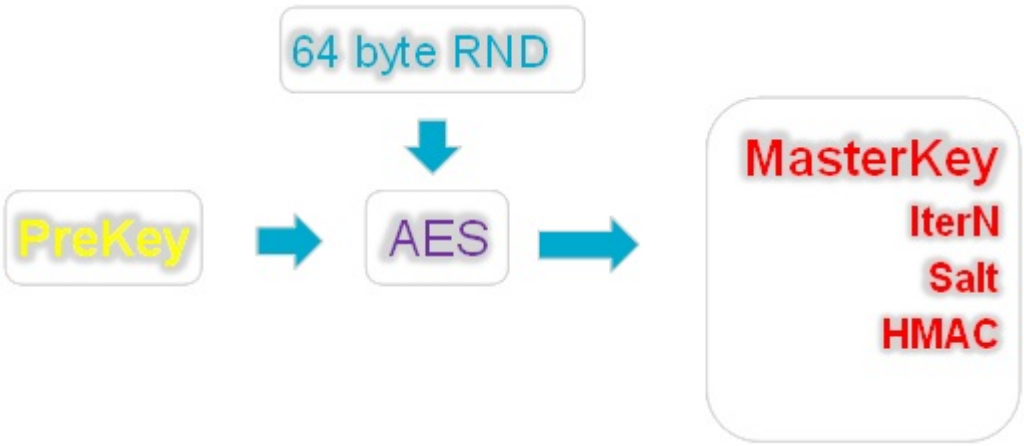
Начиная с 2000 года все ОС Windows стали использовать механизм DPAPI для того, чтобы сохранять пользовательские данные в безопасности.

Если пропустить всю криптографию, которую мы рассматривали на докладе – для расшифровки зашифрованных через DPAPI данных нам необходимы: мастер-ключ, SID пользователя, хэш пароля пользователя и сам DPAPI блоб (шифрованные DPAPI данные).

В общем виде процесс выглядит так:



Внутри нашей «криптографической шляпы» находится много различных крипто механизмов, которые мы не будем рассматривать в данной статье, дабы не перегружать читателя. Отметим лишь то, что основной частью DPAPI является так называемый Masterkey (мастер-ключ). Если по-простому, мастер-ключ – это 64 байта случайных данных, зашифрованных с помощью prekey, который генерируется из пароля пользователя и его SID.



В генерации prekey так же принимают участие дополнительные параметры: количество итераций (IterN), соль и HMAC, которые могут варьироваться от случая к случаю. Значения этих параметров хранятся вместе с мастер-ключом в одном файле.

Таким образом, зная пароль пользователя, его SID и прочитав из файла мастер-ключа параметры генерации (HMAC, Salt, InterN), мы можем сгенерировать prekey и расшифровать мастер-ключ, т.е. получить те самые случайные 64 байт, которые мы будем использовать для расшифровки DPAPI-блобов.

А если я сменю пароль?

Обычно пароли пользователя меняются с определенной периодичностью. Что будет, если пользователь сменил пароль? Куда делся предыдущий? Ведь для расшифровки мастер-ключа необходимо знать пароль пользователя, а перешифровывать все мастер-ключи пользователя каждый раз – слишком затратное удовольствие. На этот случай у Windows все продумано.

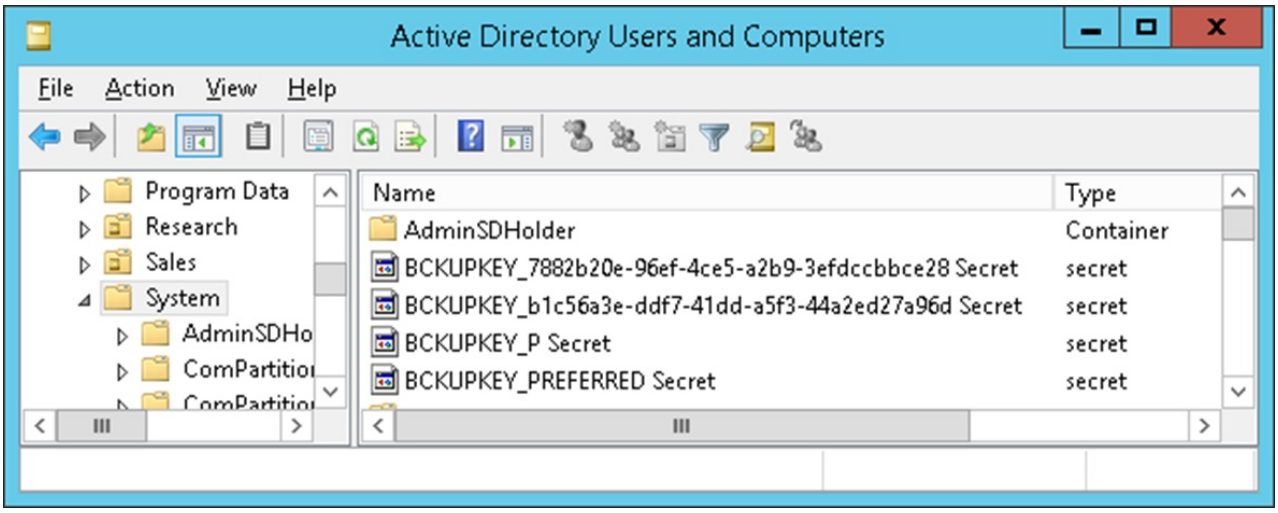
Существует специальный файл (CREDHIST), задача которого хранить все предыдущие пароли пользователя. Он также шифруется текущим паролем пользователя и сохраняется в стек. Если у системы вдруг не получилось расшифровать мастер-ключ, то она поступает следующим образом: используя текущий пароль расшифровывает первую запись в CREDHIST. Полученным паролем пытается снова расшифровать мастер-ключ и так до тех пор, пока не закончатся пароли в цепочке или мастер-ключ не будет расшифрован.

Немного о приватных ключах контроллера домена

Как вы уже догадались, DPAPI применяется для всех пользователей, в том числе и доменных. Для того, чтобы была возможность сбросить пароль пользователю, который его успешно забыл после какой-нибудь пятничной вечеринки, нужен запасной ключ, который будет храниться в надежном месте. По мнению Microsoft, таким надежным местом является контроллер домена.

Суть механизма по расшифровке мастер-ключа после сброса пароля пользователя состоит в следующем: на контроллере домена создается пара RSA-ключей – закрытый и открытый. Закрытый ключ хранится на контроллере домена в базе NTDS и называется BCKUPKEY_xxxx (см. рисунок ниже), а открытый ключ распространяется на все доменные системы и используется для формирования дубликата мастер-ключа при его генерации.

После создания мастер-ключа на доменной машине также создается его дубликат (вернее материала мастер-ключа — его 64-х байт), который хранится вместе с основным мастер-ключом в одном файле и называется Domain Key. При потере основного мастер-ключа, т.е. при сбросе пароля пользователя, система отправляет его дубликат контроллеру домена и просит его расшифровать. Контроллер, авторизовав пользователя, производит расшифровку дубликата и возвращает системе, после чего материал мастер-ключа уже заново шифруется новым паролем.



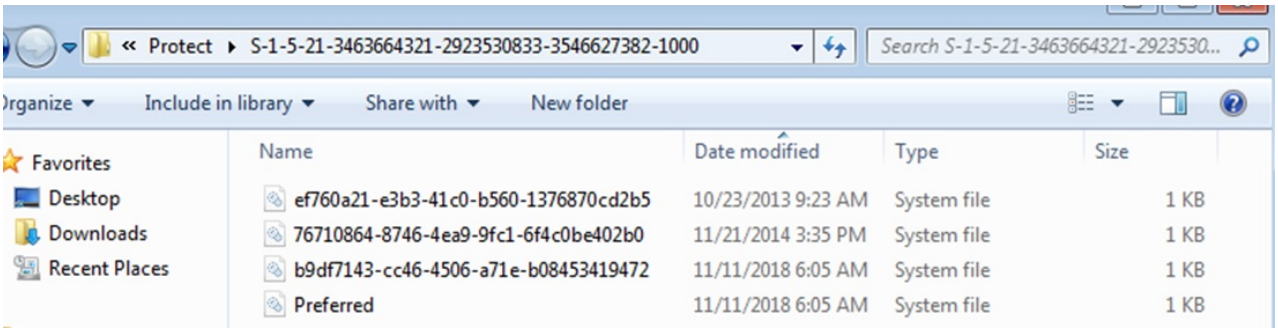
Имея соответствующие привилегии в домене (чаще всего — админские) можно достать эти приватные RSA ключи с контроллера домена через механизм репликации и использовать их при дальнейшей расшифровке мастер-ключей, созданных на доменных машинах. Сделать это можно с помощью mimikatz или DSInternals. Подробнее об этом можно прочитать в mimikatz wiki или блоге DSInternals.

Где хранятся мастер-ключи и какие они бывают?

Мастер-ключ может быть пользовательским и системным, в зависимости от того чьи секреты шифруются. Пользовательский мастер-ключ хранится в профиле пользователя по следующему пути:

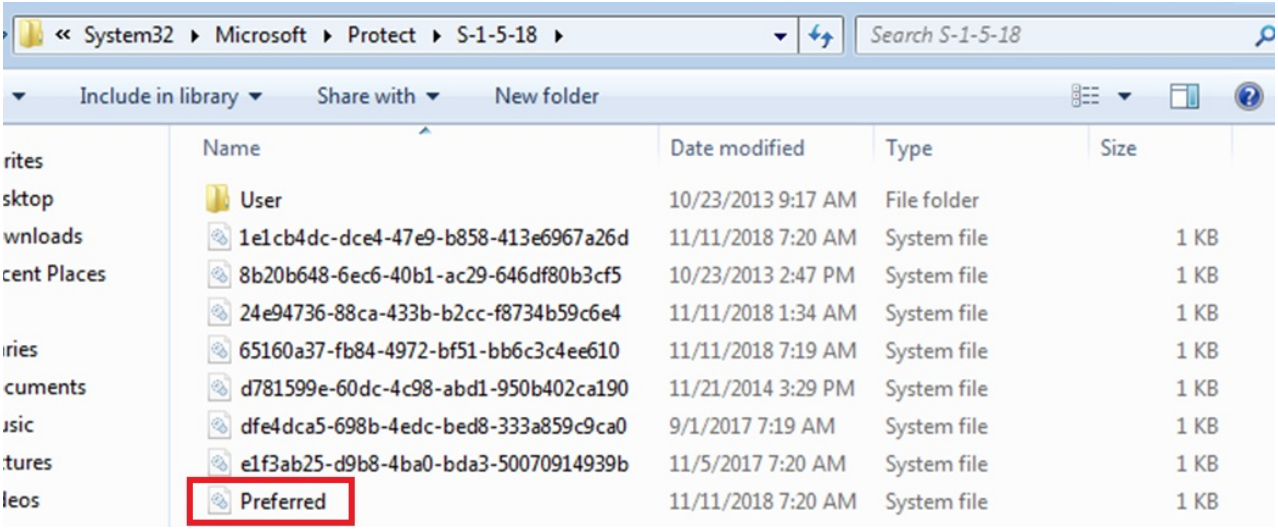
```
Users\%USER%\AppData\Roaming\Microsoft\Protect\%SID%\
```

На всякий случай система хранит все когда-либо используемые пользователем мастер-ключи. Ведь она не знает наперед, каким мастер ключом будет необходимо что-нибудь расшифровать. GUID текущего используемого ключа хранится в файле Preferred.



Системные мастер-ключи хранятся по следующему пути:
`windows\system32\Microsoft\Protect\S-1-5-18\`

Аналогично с пользователем – используется один мастер-ключ, имя которого можно найти в файле Preferred, где хранятся все когда-либо используемые ключи.



Хорошо, а что этот ваш DPAPI может дать пентестеру?

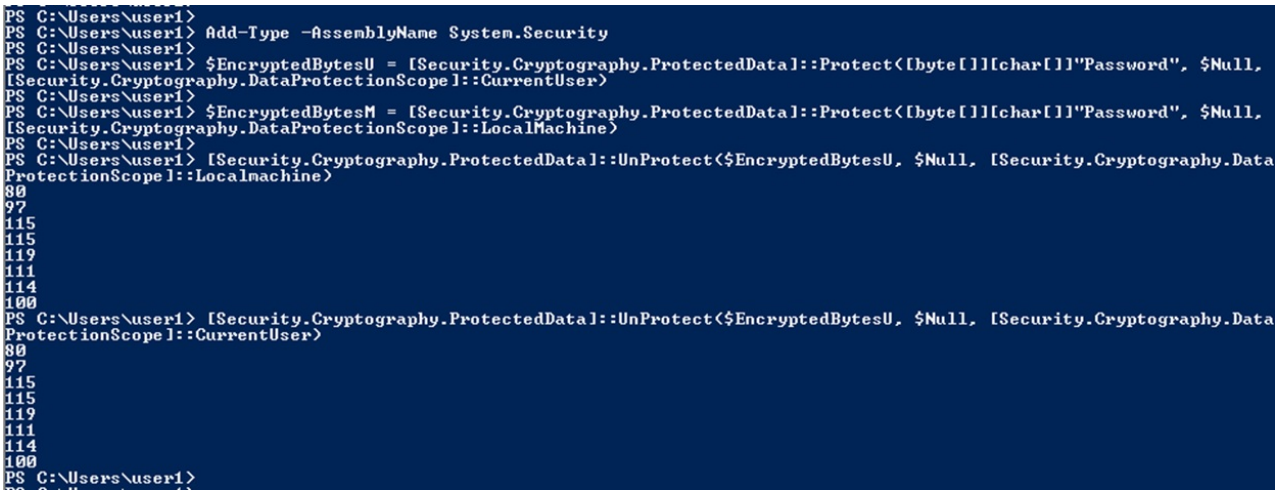
Поскольку DPAPI – легальный и простой механизм, его стараются использовать различные приложения. Потому что это удобно и безопасно. До поры до времени, конечно.

Например, DPAPI используется для шифрования закрытых ключей клиентских и системных сертификатов, WIFI ключей, Chrome (cookie, паролей), DropBox, Skype, RSA SecurID (софтварного приложения, которое генерирует одноразовые ключи). И это далеко не исчерпывающий список.

Задача пентестера – расшифровать нужные блобы и получить пароли, куки и т.д.

Сделать это можно несколькими способами. Так или иначе все они сводятся к двум расшифровкам — онлайн и офлайн. Онлайн расшифровка – это когда на машине пользователя мы просто вызываем системные функции по расшифровке данных и передаем ей на вход DPAPI-блоб, а система уже делает все сама – ищет мастер-ключ, которым был зашифрован блоб, расшифровывает его, используя при этом SID пользователя и хеш пароля, хранящийся в памяти LSASS.

На рисунке ниже приведен пример вызова DPAPI-функций для шифрования и дешифровки на powershell.



Сперва мы шифруем наш секрет (в данном случае слово «Password») через вызов функции [Security.Cryptography.ProtectedData]::Protect(). Причем делаем это два раза — в первом случае используя мастер-ключ пользователя (параметр CurrentUser), а во втором – мастер-ключ системы (параметр LocalMachine). Затем полученные блобы мы можем расшифровать через вызов обратной функции — [Security.Cryptography.ProtectedData]::UnProtect().

При этом в данном случае неважно значение параметра CurrentUser или LocalMachine, т.к. система сама находит мастер-ключ, подходящий для расшифровки блоба, и делает все необходимое. На выходе в обоих случаях мы получаем наш первоначальный секрет – слово «Password» (его побайтовое представление).

При онлайн расшифровке важно понимать, в каком контексте Вы вызываете функцию UnProtect(). Для того чтобы расшифровка прошла удачно необходимо находиться в сессии пользователя или войти в систему под новой сессией. Все дело в хеше пароля, который хранится в памяти LSASS. Если Вы делаете вызов не в сессии пользователя (например, зашли в систему по сети через psexec или meterpreter), то у Вас, соответственно, нет хеша пароля, необходимого для расшифровки мастер-ключа. Он, конечно, есть в соседней сессии, но LSASS вам его не отдаст, т.к. это уже другая сессия хоть и создана она под тем же пользователем. Для удачной онлайн расшифровки вам необходимо либо мигрировать в любой процесс, запущенный вошедшим через GUI пользователем, либо полноценно войти в систему, например, через RDP.

Альтернативой powershell'у для онлайн расшифровки DPAPI-блобов может являться вызов mimiktaz::blob с параметром /unprotect. На входе ему подается бинарный файл с DPAPI-блобом, а на выходе мы получаем расшифрованные данные. Подробнее случаи с использованием mimikatz описаны в блоге HarmJ0y.

Биток упал. Куда девать мою ферму с видюхами?

Ввиду того, что DPAPI мастер-ключи шифруются на пароле пользователя, то можно

попробовать обратный процесс – брутфорс пароля пользователя по его мастер-ключу. Например, мы получили обратный коннект от нашего макроса или DDE из отправленного docx-файла. Мы можем взять мастер-ключ пользователя и восстановить пароль пользователя без какой-либо эскалации привилегий и запусков mimikatz.

Для брутфорса пароля можно использовать Hashcat или JohnTheRipper? Но перед этим необходимо соответствующим скриптом из состава Джона достать параметры для брутфорса:

```
./DPAPImk2john.py –S <sid> -mk <masterkey> -c <domain|local>
```

Затем результат работы скрипта мы уже можем отправлять на нашу ферму с видеокартами и надеяться, что у пользователя слабый пароль, т.к. скорость брутфорса мастер-ключа примерно сопоставима со скоростью перебора WPA2, т.е. совсем уж медленно.

Здесь стоит дополнительно заметить, что в случае, когда мастер-ключ генерируется на доменной Windows 10, то к генерации prekey добавляется еще 10000 раундов алгоритма PBKDF2. Но еще хуже то, что ни Hashcat, ни JohnTheRipper об этом не знают (по крайней мере на момент написания данной статьи), а это значит, что они не смогут сбрутить пароль от такого мастер-ключа.

«Забрать все что плохо лежит, а потом уже разбираться...»

Как мы уже отмечали ранее – выполнение подозрительных действий на машине пользователя может спровоцировать дополнительный интерес к нам со стороны Blueteam команды, а это соответственно чревато тем, что весь RedTeam на этом и закончится. В качестве примера можно привести запуск powershell на компьютере бухгалтера или секретаря с последующим расследованием инцидента. Дабы не вызывать излишних подозрений, лучше использовать оффлайн способы расшифровки DPAPI-блобов. Для этого необходимо сначала забрать с машины все необходимое, а именно:

- Мастер-ключи пользователя;
- Мастер-ключи системы;
- Файл CREDHIST (если это не доменная машина);
- Пароль пользователя (или его sha1/ntlm хеш);
- SID пользователя;
- DPAPI-блобы, которые мы хотим расшифровать.

Для расшифровки в оффлайн-режиме нам никак не обойтись без специализированного инструментария. Такими инструментами могут быть:

- Mimikatz;
- Impacket (начиная с 18-й версии в нем есть функционал по DPAPI);
- Фреймворк dpapick.

Именно о фреймворке dpapick мы поговорим подробнее.

Сам python фреймворк dpapick был сделан исследователем Жан-Мишелем Пикодом еще в 2014-м году и представляет собой реализацию механизмов DPAPI на питоновских крипто-библиотеках. Использование питона, равно как и структура фреймворка позволяет с достаточной легкостью адаптировать его под различные механизмы DPAPI. В своей изначальной версии dpapick не умел использовать domain backup key для расшифровки мастер-ключей, а также в нем отсутствовали механизмы по расшифровке мастер-ключей, созданных на Windows 10 в режиме доменной машины.

После исправления данных недостатков и расширению функционала по расшифровке DPAPI-блобов, dpapick превратился в достаточно хороший инструмент по оффлайн расшифровке DPAPI. Ниже, в качестве примеров – мы покажем варианты использования этого фреймворка для расшифровки зашифрованных через DPAPI-пользовательских данных.

Chrome – забираем и расшифровываем куки и пароли

Куки Chrome хранятся в файле `%localappdata%\Google\Chrome\User Data\Default\Cookies`
Данные о логинах – в файле `%localappdata%\Google\Chrome\User Data\Default\Login Data`

Оба файла представляют из себя sqlite3 БД, в которых чувствительные данные хранятся в виде DPAPI-блобов. В составе dpapick присутствует готовый диссектор (парсер) этих данных (examples/chrome.py). Для успешной расшифровки ему необходим указать каталог с мастер-ключами, sid пользователя, его пароль или местоположение приватного ключа контроллер-домена а так же sqlite3 файл от Chrome (cookie или login data).

Оффлайн расшифровка Chrome cookie с помощью пароля пользователя

```
./chrome.py --cookie <cookiefile> --sid <SID> --password <..> --masterkey
<masterkeydir>
```

Оффлайн расшифровка Chrome cookie с помощью хеша от пароля пользователя

```
./chrome.py --cookie <cookiefile> --sid <SID> --hash <..> --masterkey
<masterkeydir>
```

Оффлайн расшифровка Chrome паролей с помощью приватного ключа с контроллера домена

```
./chrome.py --chrome <login file>  --pkey <rsa-priv.pem> --masterkey
<masterkeydir>
```

DPAPI для клиентских сертификатов

Клиентские сертификаты используются много где – для генерации OTP, EFS или аутентификации в VPN, Web-приложениях и т.д.

Сами сертификаты публичного ключа хранятся в профиле пользователя: `%APPDATA%\Microsoft\SystemCertificates\My\Certificates\`
А приватные ключи, с помощью которых собственно и производится подпись или иные криптографические операции зашифрованы через DPAPI и расположены так же в профиле пользователя по пути:

```
%APPDATA%\Roaming\Microsoft\Crypto\RSA\<SID>\
```

Для успешной расшифровки приватных ключей сертификатов и последующего воссоздания PFX-файлов нам необходимы помимо вышеперечисленных файлов еще мастер-ключи пользователя, а также его SID и пароль (либо приватный RSA-ключ с контроллера).

С помощью Dpapick и пароля пользователя расшифруем это:

```
./efs.py --certificates <cert dir> --rsakyes <RSA dir> --sid <..> --password <..>
--masterkey <masterkeydir>
```

Опциональный параметр rsaout в скриншоте дает возможность дополнительно экспортировать расшифрованные RSA ключи в PEM-формате. Результатом работы скрипта является воссозданный PFX-файл без пароля, который можно уже импортировать к себе и использовать по назначению. Если в вышеуказанных каталогах присутствуют несколько сертификатов и приватных ключей, то dpapick попытается расшифровать каждый из них и сделать несколько pfx-файлов.

Те же действия можно выполнить, используя доменный приватный ключ для расшифровки мастер-ключа, указав соответствующий параметр:

```
./efs.py --certificates <cert dir> --rsakyes <RSA dir> --masterkey <masterkeydir>
--pkey <domain bkp key>
```

Еще немного о «фишечках» домена

Говоря о домене Active Directory стоит упомянуть еще такую замечательную фишку как Credentials Roaming – функцию домена, когда мастер-ключи, зашифрованные пароли и сертификаты «путешествуют» за пользователем по всему домену Active Directory. Они не привязаны к конкретной машине и «приедут» на тот компьютер, на котором доменный пользователь залогинится.

При включении этой «фишечки» все сертификаты, которые импортирует пользователь, равно как и все его приватные ключи и пароли — улетают в AD и хранятся в соответствующих атрибутах учетной записи: msPKIAccountCrdentailas и msPKIDPAPIMasterKeys.

Посмотреть, как это выглядит внутри AD можно, например, через ldapsearch:

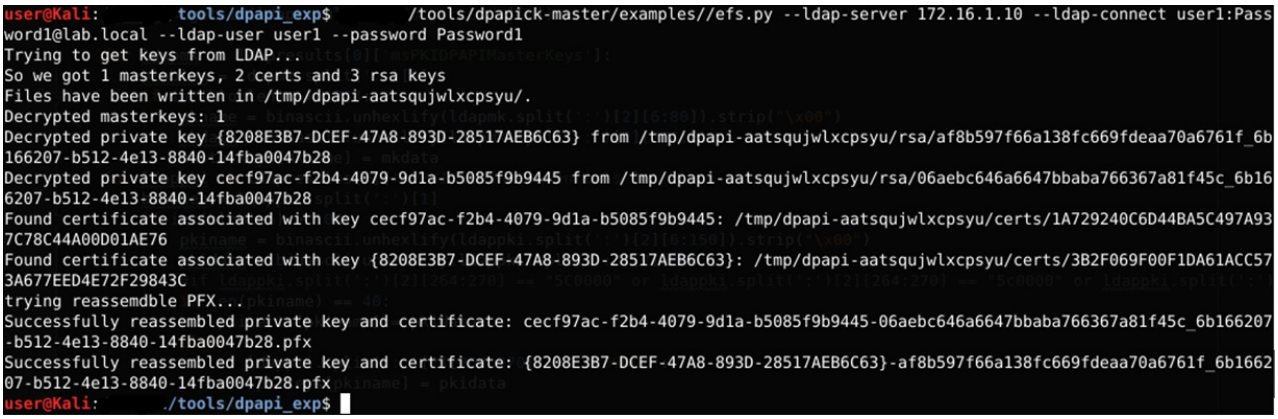
```
ldapsearch -x -h dc1.lab.local -D “user1@lab.local" -s sub "samAccountname=user1"
ldapsearch -x -h dc1.lab.local -D "admin@lab.local" -s sub
"samAccountname=anyuser"
```

По умолчанию пользователь может получить DPAPI атрибуты только для своей учетной записи. Но с повышенными привилегиями это можно сделать для любой учетной записи, в том числе и учетной записи компьютера.

Credential Roaming очень удобная технология не только для админов, но и для пентестеров. Получив доступ к домен контроллеру через ldap можно слить все сертификаты пользователя, его мастер-ключи и зашифрованные через DPAPI пароли (например пароли для подключения к сетевым дискам).

И почему бы не добавить подобный функционал в состав dpapick, подумали мы – и научили его автоматизированному изъятию сертификатов с контроллера домена через ldap, их расшифровки и формированию pfx-файлов.

```
./efs.py –ldap-server <..> --ldap-connect admin:Passw0rd@lab.local --ldap-user
user1 --password Password1
./efs.py –ldap-server <..> --ldap-connect admin:Passw0rd@lab.local --ldap-user
user1 --pkey <rsa-priv.pem>
```



Для выполнения скрипту необходимо указать домен-контроллер в качестве ldap-сервера, реквизиты подключения к нему, имя учетной записи, для которой мы получаем сертификаты и пароль для расшифровки мастер-ключа (или приватный backup ключ контроллера).

Dropbox. Угнать за 60 секунд…

Dropbox – еще один пример использования DPAPI для хранения пользовательских секретов. Токены авторизации для dropbox хранятся в файлах:

```
c:\users\<username>\Appdata\Local\Dropbox\instance1\config.dbx
c:\users\<username>\Appdata\Local\Dropbox\instance_db\instanse.dbx
```

Это зашифрованные sqlite3 базы, содержащие данные для подключения. Для шифрования применяется симметричный ключ, который в свою очередь шифруется через DPAPI и хранится в реестре:

```
HKCU\SOFTWARE\Dropbox\ks
HKCU\SOFTWARE\Dropbox\ks1
```

Таким образом, общий порядок угона dropbox следующий:

1. забираем с компьютера два файла базы данных;
2. получаем ключи из реестра и расшифровываем их с помощью dpapick;
3. зашифровываем с помощью DPAPI полученные ключи уже на своей машине и кладем в реестр;
4. на своей машине заменяем файлы базы данных и запускаем Dropbox.

Следует знать, что для вышеуказанных веток реестра установлены специальные разрешения. Их может читать только пользователь. Ни администратор, ни система их прочитать не могут. Таким образом, если обращаться к реестру от имени другого пользователя (пусть даже и администратора), то необходимо сначала установить соответствующие разрешения на указанные ветки реестра. Например, так (powershell):

```
$Sid="S-1-5-21-3463664321-2923530833-3546627382-1000";
$key=[Microsoft.Win32.Registry]::USERS.OpenSubKey("$sid\SOFTWARE\Dropbox\ks",
[Microsoft.Win32.RegistryKeyPermissionCheck]::ReadWriteSubTree,
[System.Security.AccessControl.RegistryRights]::ChangePermissions);
$acl = $key.GetAccessControl();
$rule = New-Object System.Security.AccessControl.RegistryAccessRule
("administrator","FullControl","Allow");
 $acl.SetAccessRule($rule);
$key.SetAccessControl($acl);
$key_path = "REGISTRY::HKEY_USERS\$Sid\SOFTWARE\Dropbox\ks";
(Get-ItemProperty -Path $key_path -Name Client).Client;
```

Ключи ks и ks1 содержат заголовок (8 байт) версии dbx перед DPAPI-блобом и md5 HMAC DPAPI-блоба (последние 16 байт). Сам DPAPI-блоб начинается с 9-го байта 0x01000000D0… Эти байты нужно скопировать в формате base64 в файл, который затем расшифровать через dpapick:

```
./filegeneric.py --sid <..> --password <..> --masterkey <..> --base64file <..>
```

Затем, на своей машине необходимо зашифровать полученные на прошлом этапе ключи уже нашими master-key и положить результат в соответствующие ветки реестра.

Для зашифровки удобнее всего применить powershell:

```
$hdata="4efebbdf394d4003317fc5c357beac4b";
[Byte[]] $dv0_entropy =
0xd1,0x14,0xa5,0x52,0x12,0x65,0x5f,0x74,0xbd,0x77,0x2e,0x37,0xe6,0x4a,0xee,0x9b;
$data = ($hdata -split "(?<=\G\w{2})(?=\w{2})" | %{ [Convert]::ToByte( $_, 16 )
});
Add-Type -AssemblyName System.Security;
$dk1 = [system.security.cryptography.protecteddata]::Protect($data,$dv0_entropy,
[System.Security.Cryptography.DataProtectionScope]::CurrentUser);
$pr=([System.BitConverter]::ToString($dk1));$pr
$OBJ_hmac = New-Object System.Security.Cryptography.HMACMD5
$hmac = $OBJ_hmac.ComputeHash($dk1)
$pr=([System.BitConverter]::ToString($hmac));$pr
```

В данном случае — hdata — ключ который получили на этапе расшифровки. dv0_entropy — константа энтропии, используемая DBOX в DPAPI. К получившемуся блобу необходимо спереди приписать заголовок 8 байт 0x00000000F6000000, а сзади — HMACMD5+0x00
После этого можно писать данные в соответствующие ключи реестра.

DPAPI и RSA SecurID

RSA SecurID – клиентская программа, которая используется для генерации одноразового пароля, разработанная компанией RSA.

Является достаточно популярной штукой для больших компаний и также использует DPAPI, только немного сложнее. В данном случае, инженеры RSA решили заморочиться и применили более сложные схемы DPAPI.

Данные токенов хранятся в файле `%LOCALAPPDATA%\RSA\SecurIDStorage`, который является sqlite3 базой. В базе к каждому токену записан его зашифрованный EnTokenSid (параметры начальной инициализации алгоритма генерации кодов). EnTokenSid формируется на основе DBKey, SID токена и SID пользователя, а DBKey уже формируется путем DPAPI расшифровки DBKeyEnc в следующей последовательности:
`DBKeyEnc = DPAPI(CurrenUser, DPAPI(LocalSystem(DBKey))`

Т.е. сначала DB Key шифруется системным мастер-ключом, а потом получившийся DPAPI-блоб еще раз шифруется уже пользовательским мастер-ключом.

Так же в базе присутствует CryptoCheckSum от CheckSum:
CryptoCheckSum = DPAPI blob(CurrenUser)

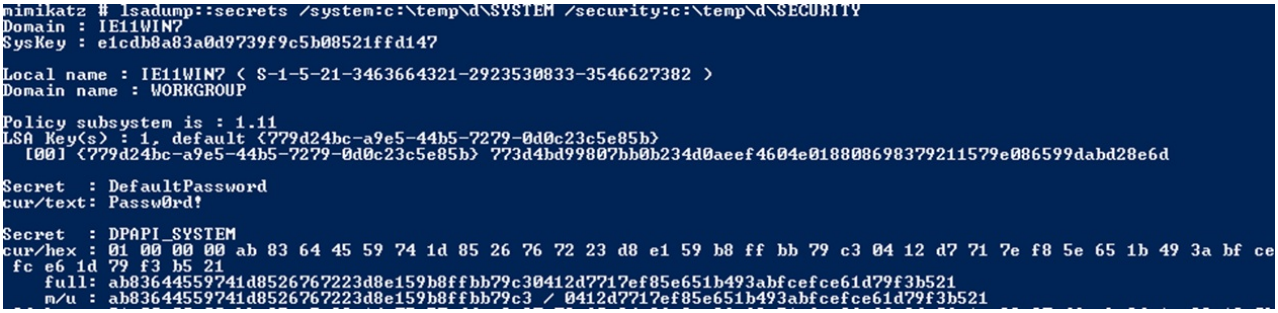
Таким образом, для того, чтобы слитый SecurIDStorage заработал на вашей машине необходимо:

1. Ввиду того, что SID пользователя участвует в формировании EncTokenSid, необходимо установить на своей виртуальной машине текущему пользователю SID в такое же значение, что и SID пользователя, у которого взята база SecurIDStorage. В этом нам поможет утилита NewSid от SysInternals;
2. Расшифровать DBKeyEnc используя мастер-ключ пользователя и его пароль или доменный приватный ключ (в случае, если машина доменная);
3. Расшифровать результат предыдущей расшифровки, используя уже системный мастер-ключ и значение параметра DPAPI_SYSTEM;
4. Расшифровать CryptoCheckSum, используя мастер-ключ пользователя
5. Зашифровать полученные значения DBKey и CheckSum в обратном порядке уже на своей виртуальной машине;
6. В некоторых версия SecurID потребуется так же установить размер HDD виртуальной машины таким же, как и размер HDD исходной машины, т.к. программа сверяет его при запуске.

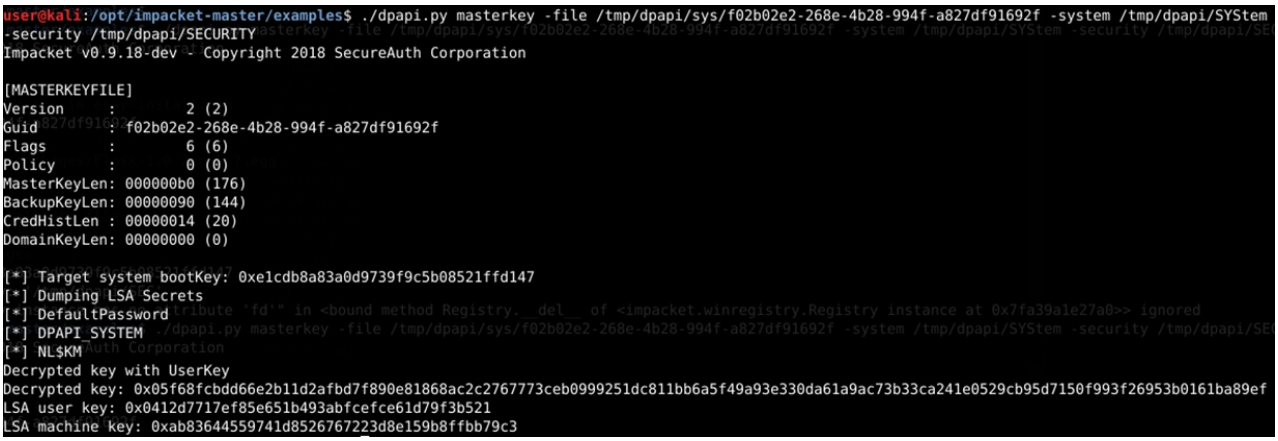
Как уже сказано выше, для расшифровки DBKeyEnc помимо мастер-ключа пользователя нам потребуется еще системный мастер-ключ, а также значение DPAPI_SYSTEM, с помощью которого производится расшифровка системных мастер-ключей. DAPPI_SYSTEM это фактически уже сформированный prekey, участвующий в формировании системных мастер-ключей. Получить его можно из памяти LSASS (через mimikatz или проанализировав дамп процесса) или из соответствующих веток реестра (HKLM\SYSTEM, HKLM\SECURITY), сдампив их и проанализировав тем же Impacket.

Затем полученный DPAPI_SYSTEM мы можем использовать для расшифровки необходимых блобов с помощью dpapick (парсер — examples/filegeneric.py), как показано на следующих скриншотах:

1) Получение DPAPI_SYSTEM через mimikatz offline



2) Получение DPAPI_SYSTEM через Impacket offline



3) Расшифровка DPAPIck с пользовательскими и системными мастер-ключами

Шпаргалка

Чтобы Вы не забыли места конкретных данных – вынесем их в отдельный раздел:

Пользовательские мастер-ключи

```
%APPDATA%\Microsoft\Protect\<SID>\*
```

Системные мастер-ключи

```
Windows\System32\Microsoft\Protect\*
```

DPAPI_SYSTEM

```
LSASecrets – online
SYSTEM, SECURITY (reg save …, system\backup, etc)
```

Пользовательские сертификаты

```
%APPDATA%\Microsoft\SystemCertificates\My\Certificates\
%APPDATA%\Microsoft\Crypto\RSA\<SID>\
```

Системные сертификаты

```
HKLM:\SOFTWARE\Microsoft\SystemCertificates\MY\Certificates\*
C:\Programdata\Microsoft\Crypto\RSA\MachineKeys\
```

Chrome

```
%localappdata%\Google\Chrome\User Data\Default\Cookies
%localappdata%\Google\Chrome\User Data\Default\Login Data
```

DropBox

```
HKCU\SOFTWARE\Dropbox\ks
HKCU\SOFTWARE\Dropbox\ks1
%APPDATA%\Local\Dropbox\instance1\config.dbx
%APPDATA%\Local\Dropbox\instance_db\instanse.dbx
```

Rsa SecurId

```
%LOCALAPPDATA%\RSA\SecurIDStorage
```

Небольшое заключение

DPAPI – шикарная вещь – главное, понимать, как ее можно использовать при проведении пентестов и RedTeam исследований.

В данной статье мы рассмотрели всего несколько примеров, где может применяться расшифровка DPAPI. На самом деле сфера применения гораздо шире. Например, мы не рассмотрели RDP (*.rdg), Icloud (pList file), Skype(*.xml), ключи для подключения к Wi-Fi. Везде применяется DPAPI и реализованы соответствующие парсеры в составе фреймворка dpapick.

Доработанная нами версия dpapick размещена на нашем GitHub. Призываем использовать данный инструмент для расшифровки DPAPI и будем признательны за дальнейшую доработку dpapick.

А какую-то интересную информацию можно найти в нашем канале в телеграм. Рассказываем про ИБ глазами RedTeam.

P.S. Спасибо организаторам OFFZONE-2018 за классную конференцию!

P.P.S. Вторая часть статьи тут