

Support ▼
[Products](#) ▼ [Solutions](#) ▼ [Why Splunk?](#) ▼ [Resources](#) ▼ [Company](#) ▼


Free Splunk

[Splunk Home](#) [Security](#) [Cloud](#) [DataOps](#) [Machine Learning](#) [Predictive](#) [Observability](#) [Analytics](#) [Dashboards](#) [Cloud](#) [Splunk Live](#) [More](#)

Security APRIL 07, 2022 | 13 MINUTE READ

You Bet Your Lsass: Hunting LSASS Access

By [Splunk Threat Research Team](#)

One of the most commonly used techniques is to dump credentials after gaining initial access. Adversaries will use one of many ways, but most commonly [Mimikatz](#) is used. Whether it be with PowerShell Invoke-Mimikatz, Cobalt Strike's Mimikatz implementation, or a custom version. All of these methods have a commonality: targeting LSASS. The Local Security Authority Subsystem Service (LSASS) is a process in Microsoft Windows operating systems that is responsible for enforcing the security policy on the system. It verifies users logging on to a Windows computer or server, handles password changes, and creates access tokens (per [Wikipedia](#)).

With that, the Splunk Threat Research Team dug into how Mimikatz, and a few other tools found in Atomic Red Team, access credentials via LSASS memory, [T1003.001](#). Part of this process for the Splunk Threat Research Team is to continuously update older analytics to ensure we are providing up to date coverage on latest techniques and behaviors.

To begin, we'll look at our current analytics related to LSASS memory dumping. We will then simulate T1003.001, OS Credential Dumping: LSASS Memory, by using [Mimikatz](#), Cobalt Strike, Atomic Red

Digital Resilience Pays Off

Research reveals every organization suffers from disruption. Investing in critical capabilities enables some to win.


Digital Resilience Pays Off

Download this e-book to learn about the role of Digital Resilience across enterprises.

[Download now](#)

Team [T1003.001](#), and [Invoke-Mimikatz](#). Last, we will update our current analytics or create new ones.

[Splunk Blogs](#)[Security](#)[DevOps](#)[Artificial Intelligence](#)[Platform](#)[Leadership](#)[Partners](#)[.conf](#)[Splunk Life](#)[More ▾](#)

Access LSASS Memory for Dump Creation

Our [first analytic](#) identifies the image load `dbgcore.dll` or `dbghelp.dll` and a `TargetImage` of `lsass.exe`. `Dbgcore.dll` or `dbghelp.dll` are two core Windows debug DLLs that have minidump functions which provide a way for applications to produce crashdump files that contain a useful subset of the entire process context.

This analytic focuses on the `CallTrace` and identifies whether `dbgcore.dll` or `dbghelp.dll` are loaded to dump credentials.

```
`sysmon` EventCode=10 TargetImage=*lsass.exe CallTrace=*dbgcore.
```

```
| stats count min(_time) as firstTime max(_time) as lastTime by  
| rename Computer as dest  
| `security_content_ctime(firstTime)`  
| `security_content_ctime(lastTime)`
```

We found, as we'll show in our simulation, that `dbgcore.dll` and `dbghelp.dll` are no longer utilized with the latest version of Mimikatz or Cobalt Strike. However, it still does capture the more basic utilities that access LSASS memory.

Detect Credential Dumping through LSASS access

This analytic looks for `GrantedAccess` of `0x1010` or `0x1410` against `lsass.exe`. These are common access types and it's probably a good time to understand what they are and the common values.

```
`sysmon` EventCode=10 TargetImage=*lsass.exe (GrantedAccess=0x10  
| stats count min(_time) as firstTime max(_time) as lastTime by  
| rename Computer as dest  
| `security_content_ctime(firstTime)`  
| `security_content_ctime(lastTime)`
```

This query is limited to two `GrantedAccess` rights that are familiar to older versions of Mimikatz, but does not stand the test of time in capturing the latest rights request.

`GrantedAccess` is the requested permissions by the `SourceImage` into the `TargetImage`. Microsoft details out

be combined to create, for example, 0x1400
(PROCESS_QUERY_LIMITED_INFORMATION and
PROCESS_QUERY_INFORMATION). You may notice some

Splunk Blogs Security DevOps Artificial Intelligence Platform Leadership Partners .conf Splunk Life More ▾

Value	Meaning
PROCESS_ALL_ACCESS (0x1ffff)	All possible access rights for a process object.
PROCESS_CREATE_PROCESS (0x0080)	Required to create a process.
PROCESS_CREATE_THREAD (0x0002)	Required to create a thread.
PROCESS_DUP_HANDLE (0x0040)	Required to duplicate a handle using <code>DuplicateHandle</code> .
PROCESS_QUERY_INFORMATION (0x0400)	Required to retrieve certain information about a process, such as its token, exit code, and priority class (see <code>OpenProcessToken</code>).
PROCESS_QUERY_LIMITED_INFORMATION (0x1000)	Required to retrieve certain information about a process (see <code>GetExitCodeProcess</code> , <code>GetPriorityClass</code> , <code>IsProcessInJob</code> , <code>QueryFullProcessImageName</code>). A handle that has the PROCESS_QUERY_INFORMATION access right is automatically granted PROCESS_QUERY_LIMITED_INFORMATION.
PROCESS_SET_INFORMATION (0x0200)	Required to set certain information about a process, such as its priority class (see <code>SetPriorityClass</code>).
PROCESS_SET_QUOTA (0x0100)	Required to set memory limits using <code>SetProcessWorkingSetSize</code> .
PROCESS_SUSPEND_RESUME (0x0800)	Required to suspend or resume a process.
PROCESS_TERMINATE (0x0001)	Required to terminate a process using <code>TerminateProcess</code> .
PROCESS_VM_OPERATION (0x0008)	Required to perform an operation on the address space of a process (see <code>VirtualProtectEx</code> and <code>WriteProcessMemory</code>).
PROCESS_VM_READ (0x0010)	Required to read memory in a process using <code>ReadProcessMemory</code> .
PROCESS_VM_WRITE (0x0020)	Required to write to memory in a process using <code>WriteProcessMemory</code> .
SYNCHRONIZE (0x00100000L)	Required to wait for the process to terminate using the wait functions.

[Skip to main content >](#)

Now that we have a basic understanding of how these

Splunk Blogs

Security

DevOps

Artificial Intelligence

Platform

Leadership

Partners

.conf

Splunk Life

More ▾

Capture Data

To get started with capturing process access event data with Sysmon, we have provided a simple config that identifies TargetImage of lsass.exe. For other EDR products, the name may be similar - [Cross Process Open](#) for Carbon Black, or CrowdStrike Falcon SuspiciousCredentialModuleLoad or LsassHandleFromUnsignedModule (reference Falcon Data Dictionary).

```
<Sysmon schemaversion="4.81">

<!-- Capture all hashes --&gt;

&lt;HashAlgorithms&gt;md5&lt;/HashAlgorithms&gt;

&lt;EventFiltering&gt;

<!-- Event ID 1 == Process Creation. --&gt;

&lt;ProcessCreate onmatch="include"/&gt;

<!-- Event ID 2 == File Creation Time. --&gt;

&lt;FileCreateTime onmatch="include"/&gt;

<!-- Event ID 3 == Network Connection. --&gt;

&lt;NetworkConnect onmatch="include"/&gt;

<!-- Event ID 5 == Process Terminated. --&gt;

&lt;ProcessTerminate onmatch="include"/&gt;

<!-- Event ID 6 == Driver Loaded.--&gt;</pre>
```

```
<DriverLoad onmatch="include"/>
```

[Splunk Blogs](#)[Security](#)[DevOps](#)[Artificial Intelligence](#)[Platform](#)[Leadership](#)[Partners](#)[.conf](#)[Splunk Life](#)[More ▾](#)

```
<ImageLoad onmatch="include"/>
```

```
<!-- Event ID 8 == CreateRemoteThread. -->
```

```
<CreateRemoteThread onmatch="include"/>
```

```
<!-- Event ID 9 == RawAccessRead. -->
```

```
<RawAccessRead onmatch="include"/>
```

```
<!-- Event ID 10 == ProcessAccess. -->
```

```
<ProcessAccess onmatch="include">
```

```
<TargetImage condition="is">C:\Windows\system32\lsass.exe</Targe
```

```
</ProcessAccess>
```

```
<!-- Event ID 11 == FileCreate. -->
```

```
<FileCreate onmatch="include"/>
```

```
<!-- Event ID 12,13,14 == RegObject added/deleted, RegValue Set,
```

```
<RegistryEvent onmatch="include"/>
```

```
<!-- Event ID 15 == FileStream Created. -->
```

```
<FileCreateStreamHash onmatch="include"/>
```

```
<!-- Event ID 17 == PipeEvent. -->
```

```
<PipeEvent onmatch="include"/>
```

</EventFiltering>

[Splunk Blogs](#)[Security](#)[DevOps](#)[Artificial Intelligence](#)[Platform](#)[Leadership](#)[Partners](#)[.conf](#)[Splunk Life](#)[More ▾](#)

The [Sysmon Modular](#) project by [Olaf Hartong](#) has some filtering that may be useful to enhance the configuration. In our testing, we utilized an open Sysmon [configuration](#) and the latest [version](#) of Sysmon.

Now we are ready to simulate.

Simulate

To simulate LSASS Memory Access, we will start with Atomic [Red Team](#) and follow up with Mimikatz, Invoke-Mimikatz, and Cobalt Strike.

Atomic Red Team

For T1003.001, LSASS Memory access, we can run individual tests or all. In this instance, we will download all the prerequisites and then run them all. There are cases where the tests may not complete and may need to be fixed or run manually (this is all based on operating environment variables).

To download [Invoke-Atomicredteam](#) and the Atomic Tests, run the following

```
[Net.ServicePointManager]::SecurityProtocol =  
  
[Net.SecurityProtocolType]::Tls12  
  
IEX (IWR 'https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/Invoke-AtomicTest.ps1')  
  
Install-AtomicRedTeam -getAtoms -force  
  
Install prerequisites  
  
Some Atomic tests have prerequisites and it is very simple to get them installed.  
  
Invoke-AtomicTest T1003.001 -GetPrereqs
```

```
PS C:\Users\Administrator> Invoke-AtomicTest T1003.001 -TestNumbers 1
PathToAtomsFolder = C:\AtomicRedTeam\atoms

Attempting to satisfy prereq: Windows Credential Editor
Attempting to satisfy prereq: Windows Credential Editor must exist on disk at specified location (C:\AtomicRedTeam\atoms\T1003.001\bin\wincred.exe)
GetProcessTokenForFile: File C:\AtomicRedTeam\atoms\T1003.001\bin\wincred.exe Memory using Process
GetProcessTokenForFile: File C:\AtomicRedTeam\atoms\T1003.001\bin\wincred.exe must exist on disk as specified location (C:\AtomicRedTeam\atoms\T1003.001\bin\wincred.exe)
Prereq already met: PreReq tool PreReq Systemcall must exist on disk at specified location (C:\AtomicRedTeam\atoms\T1003.001\bin\prereq.exe)
PreReq Defined
File C:\AtomicRedTeam\atoms\T1003.001\bin\prereq.exe Dump LSASS.exe Memory using Direct system calls and API unhooking
Attempting to satisfy prereq: Dump executable must exist on disk at specified location (C:\AtomicRedTeam\atoms\T1003.001\bin\liverdpfank\luewarr.exe)
```

[Splunk Blogs](#) [Security](#) [DevOps](#) [Artificial Intelligence](#) [Platform](#) [Leadership](#) [Partners](#) [.conf](#) [Splunk Life](#) [More ▾](#)

```
Attempting to satisfy prereq: Direct Dumper must be installed
Attempting to satisfy prereq: PreReq must be installed and part of PATH
Path is not recognized as an internal or external command,
operable program or batch file.
GetProcessTokenForFile: File C:\AtomicRedTeam\atoms\T1003.001\bin\liverdpfank\luewarr.exe Memory using Process
GetProcessTokenForFile: File C:\AtomicRedTeam\atoms\T1003.001\bin\liverdpfank\luewarr.exe must exist on disk at specified location (C:\AtomicRedTeam\atoms\T1003.001\bin\liverdpfank\luewarr.exe)
PreReq already met: PreReq tool PreReq Systemcall must exist on disk at specified location (C:\AtomicRedTeam\atoms\T1003.001\bin\prereq.exe)
PreReq Defined
File C:\AtomicRedTeam\atoms\T1003.001\bin\liverdpfank\luewarr.exe Dump LSASS.exe Memory using Direct system calls and API unhooking
Attempting to satisfy prereq: Computer must have Microsoft .Net 4.0 Framework installed
.NET 4 must be installed manually
.NET 4.0 Framework 4.0 executable can be found here: https://github.com/Scobu/RedTeam-Tools/blob/main/createdup.exe
GetProcessTokenForFile: File C:\AtomicRedTeam\atoms\T1003.001\bin\liverdpfank\luewarr.exe Memory using Process
File C:\AtomicRedTeam\atoms\T1003.001\bin\liverdpfank\luewarr.exe Dump LSASS.exe Memory using Direct system calls and API unhooking
PreReq already met: Computer must have hardware ease
```

Now we will invoke T1003.001.

Invoke-AtomicTest T1003.001

```
PS C:\Users\Administrator> Invoke-AtomicTest T1003.001
PathToAtomsFolder = C:\AtomicRedTeam\atoms

Executing test: T1003.001-3 windows Credential Editor
Windows Credential Editor - (c) 2010-2013 Amilia Security - by Hernan Ochoa (hernan@amiliasecurity.com)
Use -h for Help.

Executing test: T1003.001-2 Dump LSASS.exe Memory using ProcDump
ProcDump V0.11 - Sysinternals process dump utility
Copyright (C) 2009-2021 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

Executing test: T1003.001-3 Dump LSASS.exe Memory using comsvcs.dll
Done executing test: T1003.001-3 Dump LSASS.exe Memory using comsvcs.dll
#> [21:14:58] Error: Cannot remove item C:\Users\Administrator\AppData\Local\Temp\part.out.txt: The process cannot access the file 'C:\Users\Administrator\AppData\Local\Temp\part.out.txt' because it is being used by another process.
At C:\AtomicRedTeam\invoke-atomicredteam\Public\Invoke-AtomicTest.ps1:304 char:3
    + Remove-Item $outputfilename
    + CategoryInfo          : WriteObject: (C:\Users\Administrator\AppData\Local\Temp\part.out.txt):String) [Remove-Item], IOException
    + FullyQualifiedErrorId : RemoveItemFailed,Microsoft.PowerShell.Commands.RemoveItemCommand
Executing test: T1003.001-4 Dump LSASS.exe Memory using direct system calls and API unhooking
Done executing test: T1003.001-4 Dump LSASS.exe Memory using direct system calls and API unhooking
```

Before we hop into Splunk, let's run the other two simulations.

Invoke-mimikatz

For invoke-Mimikatz, we utilized Atomic Red Team T1059.001 test number 1. This uses the 2019 version of Mimikatz. Roberto Rodriguez called out the differences in his [blog](#) from 2017 as well, in that older versions request different permissions. Upon successful execution, it will invoke Mimikatz in memory and dump credentials.

```
PS C:\Users\Administrator\Downloads\mimikatz_trunk\x64> Invoke-AtomicTest T1059.001 -TestNumbers 1
PathToAtomsFolder = C:\AtomicRedTeam\atoms

Executing test: T1059.001-1 Mimikatz
Done executing test: T1059.001-1 Mimikatz

#####
# mimikatz 2.2.0 (x64) #18362 Oct 30 2019 13:01:25
## ^ ##
##   /** Benjamin DELIBERT genitikiwi ( benjamin@genitikiwi.com )
##   /** Vincent LE TOUX ( vincent.letoux@gmail.com ) */
## v ##
##   > http://pingcastle.com / http://mysmartlogon.com ***
mimikatz(powershell) # sekurisa::logonpasswords

Authentication Id : 0 ; 48954 (00000000:0000bf3a)
Session           : Interactive from 1
User Name         : 
Domain           : 
Logon Server     : (null)
Logon Time       : 1/5/2022 8:19:12 PM
SID              : S-1-5-90-0-1

msv :
  [00000003] Primary
  * Username  : WIN-DC-137$ 
  * Domain   : ATTACKRANGE
  * Password  : (null)
kerberos :
  [null] 
  * Username  : WIN-DC-137$ 
  * Domain   : attackrange.local
  * Password  : b0 bd 55 69 13 21 a1 b6 c6 10 76 52 5a 5e 4c 3d e6 43 22 e9 45 b2 53 95 48 55
  3c c3 17 18 25 d9 a3 24 ee 73 00 f9 bf b6 65 ea 09 87 a6 80 11 2f 95 b3 b4 54 bb c8 20 71 3a 71 0c f
  e 59 24 58 9f 99 34 52 9b 04 6e ae 5e 81 16 10 33 4b 35 20 1c f1 b8 30 9f e6 d3 fe 37 8d fb 24
```

Mimikatz

[Skip to main content ▾](#)

other products are turned off to avoid any issues.

We will run the following variations

[Splunk Blogs](#) [Security](#) DevOps Artificial Intelligence Platform Leadership Partners .conf Splunk Life More ▾

```

PS C:\Users\Administrator\Downloads\mimikatz_trunk\x64> .\mimikatz.exe SEKURLSA::Krbtgt exit
#####
# ## mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
# ## ^##, "A La Vie, A L'Amour" - (oe.oe)
# ## < /> /* Benoît DELPY gentilkiwi ( benjamin@gentilkiwi.com )
# ## > https://blog.gentilkiwi.com/mimikatz
# ## ^## V ## Vincent LE TOUX
# ## > https://pingcastle.com / https://mysmartlogon.com ===
#####
mimikatz(commandline) # sekurlsa::logonpasswords
Privilage-'20' Ok

mimikatz(commandline) # sekurlsa::logonpasswords
Authentication Id : 0 48994 (00000000:0000bf3a)
Logon Type : 3 Interactive from 1
User Name : DWF-1
  * Domain : ATTACK RANGE
  * User : DWF-1 Manager
Logon Server : (null)
Logon Time : 1/3/2022 8:19:12 PM
SID : S-1-5-21-1024-31455-50-0-1
SID Hash : 3c c3 17 8c 2d d0 a3 24 e3 00 f9 bf b6 69 ea 09 87 a6 80 11 2f 95 b3 b4 54 bb c8 20 71 3a 71 0c fe 5f 14 e3 25 ae 63 1a 85

mimikatz(commandline) # [0000000] Principe
  * Username : WIN-DC-1375
  * Domain : ATTACK RANGE
  * User : WIN-DC-1375
  * SID : S-1-5-21-1024-31455-50-0-1
  * SHA1 : 7d87aae427abb73554bb4208dad58e9eb5300f2a
  * tskipg : 1
  * vuid : 1
  * Username : WIN-DC-1375
  * Domain : ATTACK RANGE
  * Password : (null)
kerberos {
  * Username : WIN-DC-1375
  * Domain : attackrange_local
  * Password : 20 bd 10 35 20 35 bb c6 10 76 52 5a 5e 4c 3d e6 43 22 a9 45 ad 2b 53 95 45 55 ad xc ce aa 38 fe 63 1a 85
  * Flags : 0x00000000000000000000000000000000
}
3c c3 17 8c 2d d0 a3 24 e3 00 f9 bf b6 69 ea 09 87 a6 80 11 2f 95 b3 b4 54 bb c8 20 71 3a 71 0c fe 5f 14 e3 25 ae 63 1a 85

mimikatz(commandline) # SEKURLSA::Krbtgt
Current krbtgt: 5 credentials
  * rc4_hmac_nt : 5ffcf4fdbb82aa7dd2882d4e98dbf934e
  * rc4_hmac_old : 5ffcf4fdbb82aa7dd2882d4e98dbf934e
  * rc4_md4 : 5ffcf4fdbb82aa7dd2882d4e98dbf934e
  * aes256_hmac : e7c188f7ea617d50e505f060e77dec23ba0dcc99f5cf16c121ff5e9f7ed96
  * aes128_hmac : c95e878dc9e8966d77deb3407d460156

mimikatz(commandline) # exit
Bye!

```

Alright, that finishes our easy tests for Mimikatz.

Mimikatz and Cobalt Strike

Similarly, run the same commands within a session using Cobalt Strike. The behavior we will look for here is similar to most [Cobalt Strike](#) behavior we've identified in the past, a spawned process, default of rundll32.exe, with no command-line arguments, with a process access event targeting LSASS.exe.

```
beacon> logonpasswords
[*] Tasked beacon to run mimikatz's sekurlsa::logonpasswords command
[+] host called home, sent: 296058 bytes
[+] received output:
```

[Splunk Blogs](#)[Security](#)[DevOps](#)[Artificial Intelligence](#)[Platform](#)[Leadership](#)[Partners](#)[.conf](#)[Splunk Life](#)[More ▾](#)

```
Domain      : Window Manager
Logon Server : (null)
Logon Time   : 1/18/2022 6:45:04 PM
SID         : S-1-5-90-0-1

msv :
[00000003] Primary
* Username : WIN-DC-MHAAG-AT$
* Domain  : ATTACKRANGE
* NTLM     : fc3470242c5db5880bd63bb169e471d5
* SHA1     : c0c7e88ab57d47cd1e513b29734d5b749f4d704d

tspkg :
wdigest :
* Username : WIN-DC-MHAAG-AT$
* Domain  : ATTACKRANGE
* Password : (null)

kerberos :
* Username : WIN-DC-MHAAG-AT$
* Domain  : attackrange.local
* Password : 62 26 2f 24 01 5d a4 a6 50 f5 f8 39 d8 d7 d9 ec
aa 74 a5 44 b6 df 77 44 89 00 a5 4d e2 67 f0 a0 b2 a5 bc d6 b5 28 2c 0
7a cb 80 91 69 55 f4 ab ee e3 a8 78 e6 2b c3 eF 7a 70 d0 4c 56 6F 80 a
db 21 e5 88 92 c9 bb 06 1e ba 93 f7 46

ssp :
[WIN-DC-MHAAG-AT] administrator *4600
beacon>
```

Notes on testing

Typically, our process is to simulate 1 test at a time and validate coverage. Iterate over each test and modify our query as needed. For brevity, the blog skips the thorough process and highlights a faster process.

Continuous Improvement

Access LSASS Memory for Dump Creation

For our first analytic that focuses on CallTrace image load `dbgcore.dll` or `dbghelp.dll`, we found that over time Mimikatz moved away from these two DLLs (`dbgcore.dll` or `dbghelp.dll`). The main DLL used by Mimikatz is now `ntdll.dll`. `Ntdll.dll` is a native Windows binary that provides similar native API paths to perform credential dumping. For example in the `sekurlsa` module there are many `ntdll` exported api's, but what stands out is `RtlCopyMemory` which is used to execute the module related to credential dumping.

After simulating the behavior we needed, we get some results

```
| stats count min(_time) as firstTime max(_time) as lastTime by
```

Splunk Blogs Security DevOps Artificial Intelligence Platform Leadership Partners .conf Splunk Life More ▾

```
| rename Computer as dest | `security_content_ctime(firstTime)
```

dest	Terphinge	TerphProcessID	SourceIndex	SourceProcessID	count
won-011-137-attorange.local	C:\Windows\system32\calc.exe	644	C:\Windows\system32\cmd.exe	644	5524
won-011-137-attorange.local	C:\Windows\system32\cmd.exe	644	C:\Windows\system32\cmd.exe	644	3095
won-011-137-attorange.local	C:\Windows\system32\cmd.exe	644	C:\Windows\system32\cmd.exe	644	3092
won-011-137-attorange.local	C:\Windows\system32\cmd.exe	644	C:\Windows\system32\cmd.exe	644	3108
won-011-137-attorange.local	C:\Windows\system32\cmd.exe	644	C:\Windows\system32\cmd.exe	644	1588

In our screenshot, we see some utilities that still use `dbgcore.dll` or `dbghelp.dll` when credential dumping occurs. However, we do not see Mimikatz.exe and a few other utilities using `dbgcore.dll` or `dbghelp.dll`. Based on CallTraces, we see a pattern of `ntdll.dll` being used by various credential dumping utilities. If we add `ntdll.dll` to our current query we get the following results:

We now have a list of processes (source) targeting lsass.exe. Sometimes legitimate applications will request access to lsass.exe for credential access, say to authenticate with AzureCLI or a software deployment application. What will differentiate these? This will be environment dependent based on roles and access associates may have, based on process hierarchy, or GrantedAccess. As we will dig into next, filtering may be much easier once we combine GrantedAccess with CallTrace.

Now we add GrantedAccess to our query to identify any patterns

We can see the permissions being requested by the SourceImage and we will begin looking at what those values mean next.

Detect Credential Dumping through LSASS access

Now our second analytic is focused on GrantedAccess, which we explored earlier what the values are. Now that simulation is complete we can begin digging in.

The base query looks like this with some simulated data:

```
| stats count min(_time) as firstTime max(_time) as lastTime by  
  
| rename Computer as dest | `security_content_ctime(firstTime)`
```

Index	Sourcefile	SourcefileID	Searchtext	SearchtextID	Typeoffset	TypeoffsetID	Eventcode	EventcodeID	Groupname	Groupid
source-127-137-177-234	C:\Windows\System32\cmd.exe	114	C:\Windows\System32\cmd.exe	644	18	0x1010	1	0x1010	cmd	1
source-127-137-177-234	C:\Windows\System32\cmd.exe	114	C:\Windows\System32\cmd.exe	644	18	0x1010	1	0x1010	cmd	1
source-127-137-177-234	C:\Windows\System32\cmd.exe	279	C:\Windows\System32\cmd.exe	644	18	0x1010	1	0x1010	cmd	1
source-127-137-177-234	C:\Windows\System32\cmd.exe	344	C:\Windows\System32\cmd.exe	644	18	0x1010	1	0x1010	cmd	1
source-127-137-177-234	C:\Windows\System32\cmd.exe	543	C:\Windows\System32\cmd.exe	644	18	0x1010	1	0x1010	cmd	1
source-127-137-177-234	C:\Windows\System32\cmd.exe	553	C:\Windows\System32\cmd.exe	644	18	0x1010	1	0x1010	cmd	1
source-127-137-177-234	C:\Windows\System32\cmd.exe	563	C:\Windows\System32\cmd.exe	644	18	0x1010	1	0x1010	cmd	1
source-127-137-177-234	C:\Windows\System32\cmd.exe	583	C:\Windows\System32\cmd.exe	644	18	0x1010	1	0x1010	cmd	1

With all the simulation it was easy to spot the patterns between CallTrace and GrantedAccess, so we created a table to showcase these values:



Splunk Blogs

Security

DevOps

Artificial Intelligence

Platform

Leadership

Partners

.conf

Splunk Life

More ▾

0x1fffff	createdump.exe	C:\Windows\SYSTEM32\ntdll.dll+a6144 C:\Windows\SYSTEM32\ntdll.dll+6c
0x1010	Invoke-mimikatz	C:\Windows\SYSTEM32\ntdll.dll+a6144 C:\Windows\System32\KERNELBAS
0x1438	mimikatz.exe	C:\Windows\SYSTEM32\ntdll.dll+a6144 C:\Windows\System32\KERNELBAS
0x1410	PasswordHashesView.exe	C:\Windows\SYSTEM32\ntdll.dll+a6144 C:\Windows\System32\KERNELBAS\x64\PasswordHashesView.exe+cf2d C:\Users\Administrator\Downloads\nirs

With all this testing, our updated Sysmon query combines the two analytics we set out to enhance by focusing on specific GrantedAccess rights and CallTrace DLLs. Will this catch everything? Probably not, but it will at least catch the majority of tools used and allow us to filter out known good in environments and focus on the rare.

```
`sysmon` EventCode=10 TargetImage=*lsass.exe GrantedAccess IN ("
```

This will require some filtering as common system processes will access lsass.exe with GrantedAccess of 0x1400 and 0x1010.

TargetImage	GrantedAccess	TargetProcess	SourceImage	SourceProcess	TargetName	Count
C:\Windows\system2\lsass.exe	0x1010	420 C:\Windows\system2\lsass.exe!{powershell.exe}		1112 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	1
C:\Windows\system2\lsass.exe	0x1010	620 C:\Users\ADMINI~1\Temp\crashdump.vbs.vbs		6208 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	1
C:\Windows\system2\lsass.exe	0x1010	620 C:\Users\ADMINI~1\Temp\crashdump.vbs.vbs		4044 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	1
C:\Windows\system2\lsass.exe	0x1010	620 C:\Users\ADMINI~1\Temp\crashdump.vbs.vbs		5844 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	1
C:\Windows\system2\lsass.exe	0x1010	620 C:\Windows\system2\lsass.exe!{powershell.exe}		4722 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	1
C:\Windows\system2\lsass.exe	0xFFFFF	620 C:\Users\ADMINI~1\Temp\crashdump.vbs.vbs		4244 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	1
C:\Windows\system2\lsass.exe	0xFFFFF	620 C:\Windows\system2\lsass.exe!{powershell.exe}		4420 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	1
C:\Windows\system2\lsass.exe	0xFFFFF	620 C:\Windows\Temp\crashdump.vbs.vbs		6420 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	1
C:\Windows\system2\lsass.exe	0x1010	620 C:\Windows\Temp\crashdump.vbs.vbs		6340 NT AUTHORITY\NETWORK SERVICE	NT AUTHORITY\SYSTEM	2
C:\Windows\system2\lsass.exe	0xFFFFF	620 C:\Windows\system2\lsass.exe!{powershell.exe}		1120 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	2
C:\Windows\system2\lsass.exe	0xFFFFF	620 C:\Windows\system2\lsass.exe!{powershell.exe}		2452 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	2
C:\Windows\system2\lsass.exe	0xFFFFF	620 C:\Windows\system2\lsass.exe!{powershell.exe}		6208 ATTACKER&SUBJEST&TAKE	NT AUTHORITY\SYSTEM	4

For Mimikatz and the various items that come with it, whenever it does make contact with LSASS.exe the results are mostly the same.

Splunk Blogs Security DevOps Artificial Intelligence Platform Leadership Partners .conf Splunk Life More ▾

As noted in our table or [Call Trace and GrantedAccess](#), dependent upon what is being executed with the utility (MimiKatz for example) the access will be different. This was also noted by Roberto Rodriguez [here](#) and Carlos Perez [here](#).

Does this catch every variation of Mimikatz out there?
Most likely not. However, it will be a great start to identify uncommon GrantedAccess rights to Lsass.exe. This may be expanded upon or converted to other utilities to assist with detecting suspicious LSASS access.

Additional Observations

During our simulations we identified behaviors that may assist teams in identifying suspicious SourceUser accessing LSASS. Typically, we will see source NT AUTHORITY\SYSTEM and TargetUser NT AUTHORITY\SYSTEM for normal system process behavior. However, seeing source ATTACKRANGE\administrator and Target NT AUTHORITY\SYSTEM is suspicious.

TargetImage #	GrantedAccess #	TargetProcessId #	SourceImage #	SourceProcessId #	SourceUser #	TargetUser #	Op
C:\Windows\system32\lusaas.exe	0x1010	628	C:\Windows\system32\run11212.exe	4324	ATTORANGE\administrator	N/A	N/A
C:\Windows\system32\lusaas.exe	0x1010	628	C:\Windows\system32\run11212.exe	6464	NT AUTHORITY\SYSTEM	NT AUTHORITY\SYSTEM	N/A

What if an adversary is already elevated? SourceUser will not be a user account, but NT AUTHORITY\SYSTEM. This may be a bit more difficult to detect, but it's worth a hunt.

With all this data we hope you found this informative and understand a bit of our continuous improvement for our content.



[Skip to main content](#)

Windows Hunting System Account Targeting Lsass

Splunk Blogs Security DevOps Artificial Intelligence Platform Leadership Partners .conf Splunk Life More ▾

EventIDCode=18 TargetImage=Lsass.exe						
stats count,firstTime as firstTime,mostTime as lastTime by Computer, TargetImage, GrantedAccess, SourceImage, SourceProcessId, SourceUser, TargetUser						
security_content_ctime(firstTime)						
security_content_ctime(lastTime)						
29 events (Partial results) at 12/22 0:00:00.000 PM to 12/22 6:46:02.000 PM No Event Sampling *						
Events [29] Patterns Status (0) Visualization						
20 Per Page ▾ Formatted Previous ▾						
dest_id	TargetImage	GrantedAccess	SourceImage	SourceProcessId	SourceUser	TargetUser
win-0tq-0tqg-attack	C:\Windows\system32\lsass.exe	0x100	C:\Windows\System32\user32.dll	3708	ATTORNEY-CLIENT-Administrator	NT AUTHORITY\SYSTEM
range						
193.attacksrange.local						
win-0tq-0tqg-attack	C:\Windows\system32\lsass.exe	0x100	C:\Windows\System32\user32.dll	7408	ATTORNEY-CLIENT-Administrator	NT AUTHORITY\SYSTEM
range						
193.attacksrange.local						
win-0tq-0tqg-attack	C:\Windows\system32\lsass.exe	0x100	C:\Windows\System32\user32.dll	8044	ATTORNEY-CLIENT-Administrator	NT AUTHORITY\SYSTEM
range						
193.attacksrange.local						
win-0tq-0tqg-attack	C:\Windows\system32\lsass.exe	0x100	C:\Windows\System32\user32.dll	876	NT AUTHORITY\SYSTEM	NT AUTHORITY\SYSTEM
range						
193.attacksrange.local						
win-0tq-0tqg-attack	C:\Windows\system32\lsass.exe	0x100	C:\Windows\System32\user32.dll	2408	ATTORNEY-CLIENT-Administrator	NT AUTHORITY\SYSTEM
range						
193.attacksrange.local						

Windows Non-System Account Targeting Lsass

The following analytic identifies non SYSTEM accounts requesting access to lsass.exe.

EventIDCode=18 TargetImage=Lsass.exe						
stats count,mostTime as firstTime,mostTime as lastTime by Computer, TargetImage, GrantedAccess, SourceImage, SourceProcessId, SourceUser, TargetUser						
security_content_ctime(firstTime)						
security_content_ctime(lastTime)						
0 of 29320 events matched No Event Sampling *						
Events [0] Patterns Status (0) Visualization						
20 Per Page ▾ Formatted Previous ▾						
dest_id	TargetImage	GrantedAccess	SourceImage	SourceProcessId	SourceUser	TargetUser
win-0tq-0tqg-attack	C:\Windows\system32\lsass.exe	0x100	C:\Windows\System32\user32.dll	3708	ATTORNEY-CLIENT-Administrator	NT AUTHORITY\SYSTEM
range						
193.attacksrange.local						
win-0tq-0tqg-attack	C:\Windows\system32\lsass.exe	0x100	C:\Windows\System32\user32.dll	7408	ATTORNEY-CLIENT-Administrator	NT AUTHORITY\SYSTEM
range						
193.attacksrange.local						
win-0tq-0tqg-attack	C:\Windows\system32\lsass.exe	0x100	C:\Windows\System32\user32.dll	8044	ATTORNEY-CLIENT-Administrator	NT AUTHORITY\SYSTEM
range						
193.attacksrange.local						
win-0tq-0tqg-attack	C:\Windows\system32\lsass.exe	0x100	C:\Windows\System32\user32.dll	2408	ATTORNEY-CLIENT-Administrator	NT AUTHORITY\SYSTEM
range						
193.attacksrange.local						

Name	Technique ID	Tactic	Description
Windows Hunting System Account Targeting Lsass	T1003.001	Credential Access	Identifies all processes requesting access into Lsass.exe
Windows Non-System Account Targeting Lsass	T1003.001	Credential Access	Identifies non SYSTEM accounts requesting access to Lsass.exe.
Windows Possible Credential Dumping	T1003.001	Credential Access	The following analytic is an enhanced version of two previous analytics that identifies common

[Skip to main content >](#)

permission requests
and CallTrace DLLs in
order to detect

Splunk Blogs Security DevOps Artificial Intelligence Platform Leadership Partners .conf Splunk Life More ▾

References

- https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service
- <https://docs.microsoft.com/en-us/windows/win32/api/minidumpapiset/nf-minidumpapiset-minidumpwritedump>
- https://cyberwardog.blogspot.com/2017/03/chronicles-of-threat-hunter-hunting-for_22.html
- <https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1>
- <https://docs.microsoft.com/en-us/windows/win32/procthread/process-security-and-access-rights>
- <https://github.com/trustedsec/SysmonCommunity-Guide/blob/master/chapters/process-access.md>

Splunk Threat Research Team

The Splunk Threat Research Team is an active part of a customer's overall defense strategy by enhancing Splunk security offerings with verified research and security content such as use cases, detection searches, and playbooks. We help security teams around the globe strengthen operations by providing tactical guidance and insights to detect, investigate and respond against the latest threats. The Splunk Threat Research Team focuses on understanding how threats, actors, and vulnerabilities work, and the team replicates attacks which are stored as datasets in the [Attack Data repository](#).

Our goal is to provide security teams with research they can leverage in their day to day operations and to become the industry standard for SIEM detections. We are a team of industry-recognized experts who are encouraged to improve the security industry by sharing our work with the community via conference talks, open-sourcing projects, and writing white papers or blogs. You will also find us presenting our research at conferences such as Defcon, Blackhat, RSA, and many more.

Read more [Splunk Security Content](#).

[Security](#) 4 MIN READ

Here's What's New in ESCU: July 2018

Find out what's new in the July 2018 releases of...

[Security](#) 3 MIN READ

Selecting the Right Skills for Your SA&O Project (Part 2 of 2)

This article provides a high-level overview of th...

[Security](#) 6 MIN READ

Hunting for Malicious PowerShell using Script Block Logging

The Splunk Threat Research Team recentl...

About Splunk

The Splunk platform removes the barriers between data and action, empowering observability, IT and security teams to ensure their organizations are secure, resilient and innovative.

Founded in 2003, Splunk is a global company — with over 7,500 employees, Splunkers have received over 1,020 patents to date and availability in 21 regions around the world — and offers an open, extensible data platform that supports shared data across any environment so that all teams in an organization can get end-to-end visibility, with context, for every interaction and business process. Build a strong data foundation with Splunk.

[Learn more about Splunk >](#)



Subscribe to our blog

Splunk Blogs [Security](#) [DevOps](#) [Artificial Intelligence](#) [Platform](#) [Leadership](#) [Partners](#) [.conf](#) [Splunk Life](#) More ▾

Splunk straight to your inbox.

[Follow @Splunk >](#)

[Follow @Splunk >](#)

[Sign Up Now](#)

COMPANY	PRODUCTS	LEARN	CONTACT SPLUNK
About Splunk	Free Trials & Downloads	OpenTelemetry: An Introduction	Contact Sales >
Careers	Pricing	Red Team vs Blue Team	Contact Support >
Global Impact	View All Products	What is Multimodal AI?	User Reviews
How Splunk Compares		An Introduction to Distributed Systems	
SPLUNK SITES			
Leadership	.conf	Data Lake vs Data Warehouse	
Newsroom	Documentation	What is Business Impact Analysis?	Gartner Peer Insights™
Partners	Investor Relations	Risk Management Frameworks Explained	PeerSpot
Perspectives by Splunk	Training & Certification	CVE: Common Vulnerabilities and Exposures	TrustRadius
Splunk Policy Positions	T-Shirt Store	What are DORA Metrics?	SPLUNK MOBILE
Splunk Protects	Videos	View All Articles	
Splunk Ventures			
Supplier Central	View All Resources		
Why Splunk?			



© 2005 - 2024 Splunk LLC All rights reserved.

[Legal](#) [Patents](#) [Privacy](#) [Sitemap](#) [Website Terms of Use](#)