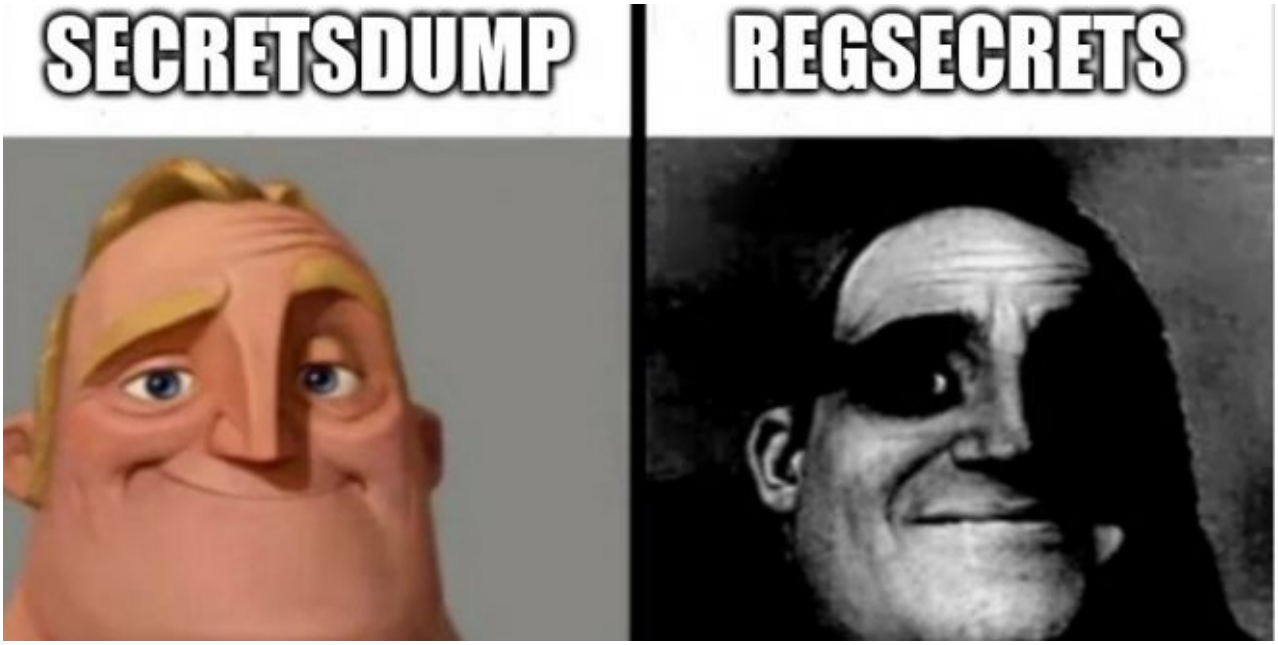
LSA Secrets: revisiting secretsdump

synacktiv.com/publications/lsa-secrets-revisiting-secretsdump



Rédigé par Julien Egloff - 20/02/2025 - dans Pentest - Téléchargement
When doing Windows or Active Directory security assessments, retrieving secrets stored on a compromised host constitutes a key step to move laterally within the network or increase one's privileges. The infamous `secretsdump.py` script from the impacket suite is a well-known tool to extract various sensitive secrets from a machine, including user hashes, the base secret for the DPAPI encryption mechanism, service accounts cleartext credentials, and more.

As years passed, security products began to effectively detect and block the execution of this script, which led us to have a closer look at the inner workings of secretsdump and devise a new version that is currently less prone to detection.

Revisiting secretsdump

As stated in the introduction, `secretsdump.py` allows retrieving specific secrets in the registry from a targeted computer, but it also allows performing DCSync operations when targeting a domain controller. This blogpost will focus only on the remote registry part, without using the recently added vssadmin approach.

Currently, the following secrets are retrieved by `secretsdump.py`:

- SAM hive content: contains local user NT hashes.
- MsCache: contains domain user hashes if the registry key CachedLogonsCount is higher than zero.

- LSA secrets:
  - Computer password if enrolled in an Active Directory domain.
  - DPAPI local keys (dpapi_userkey and dpapi_machinekey).
  - Passwords of accounts used for non-interactive services.

Let us dive into `secretsdump.py`'s current technical implementation:

- The `RemoteRegistry` service is checked. If its not enabled or not started, it is enabled and started.
- The `bootkey` is computed by performing registry queries on the `SYSTEM\CurrentControlSet\Control\Lsa\[JD|Skew1|GBG|Data]` keys.
- The `SAM` and `SECURITY` hives are saved inside `C:\Windows\Temp` with a randomly generated name of 8 characters with the `.tmp` extension.
- Hives content is read using SMB and the `ADMIN$` administrative share. All keys and secrets are decrypted and displayed.
- Temporary hives are deleted and the `RemoteRegistry` is reset to its initial state.

While its execution may leave forensic traces (partially described in this article), the following heuristics are easily picked up by security products, and will raise alerts:

- Interactions with the `svcctl` named pipe to perform action on services, especially the `RemoteRegistry` service required by secretsdump.
- Connecting to the `ADMIN$` share.
- Saving base registry hives like `SAM` or `SECURITY`.
- Writing files to a temporary folder with a fixed size and a specific extension.

More importantly, current EDR-like products recognize and often block saving base hives.

What if only interacting with the `RemoteRegistry` was enough to perform secrets extraction? This would also avoid writing anything on the disk, leaving fewer forensics traces.

There is one problem to circumvent, the `SAM` and `SECURITY` hives are not accessible with a Local Administrator account, `SYSTEM` identity is required:

```
$ reg.py Administrator:Password123@172.16.203.128 query -keyName HKLM\\SAM\\SAM
Impacket v0.13.0.dev0+20250206.100953.075f2b10 - Copyright Fortra, LLC and its
affiliated companies

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[-] DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

However, as such a restriction is not applied when saving hives, there might be a way to query arbitrary keys in the same manner.

Fortunately, the Microsoft documentation is well written and the `BaseRegOpenKey` method contains a `dwOptions` argument that accepts a `REG_OPTION_BACKUP_RESTORE` value. Opening a key with this intent will bypass any ACL check, therefore, allowing to dump secrets while only interacting with the `svcctl` named pipe and the `RemoteRegistry` service.

A pull request has been opened on the original impacket repository with a script named `regsecrets.py` which implements the aforementioned method. When retrieving secrets, the following steps are performed:

- The `RemoteRegistry` service is checked. If its not enabled or not started, it is enabled and started.
- The `bootkey` is computed by performing registry queries on the `SYSTEM\CurrentControlSet\Control\Lsa\[JD|Skew1|GBG|Data]` keys.
- Secrets are computed using registry queries only.
- The `RemoteRegistry` is reset to its initial state.

This script was executed in multiple recent engagements and successfully circumvented the detection and / or blocking actions performed by security products.

Bonus: dpapidump.py

Moreover, a script from a pull request from 2021 that was never merged was modified and added into our pull request. The original script performs the following steps:

- Connect to the WMI interface over DCOM to perform queries and retrieve encrypted secrets related to SCCM. It first connects to the RPC port mapper before connecting to another port and then connects to a high dynamic port. However, depending on the firewall configuration, the second port might not be reachable.
- Connect to the `C$` share, retrieve machine DPAPI masterkeys in `Windows\System32\Microsoft\Protect\S-1-5-18\User` and credentials secrets in `Windows\System32\config\systemprofile\AppData\Local\Microsoft\Credentials`. Credentials include scheduled task user and password for non-interactive tasks.
- Retrieve the necessary `dpapi_userkey` if it was not provided using secretsdump.
- Decrypt the retrieved secrets and print them.
- Close all connections.

If the `dpapi_userkey` is provided, this script does not perform actions that have a high risk of raising security alerts. When performing post-compromise secrets collections, running in succession `regsecrets.py` then `dpapidump.py` while providing the `dpapi_userkey` minimizes the risk of raising security alerts while enhancing the scope of recovered secrets. Amongst them, SCCM information are retrieved, including the popular Network Access Accounts credentials (NAA) if SCCM is deployed with this account.

A modified version of this script, called `dpapidump.py` is included in the PR with multiple changes:

- It uses the technique described in the first part of this article to retrieve local machine DPAPI keys if not provided in the CLI.
- The collections of information related to SCCM in the WMI database has been enhanced to match those retrieved by sccmhunter.
- Various small bug fixes and code improvements.
- Disable the retrieval of the credentials file related to `virtualapp/didlogical`, this credential has always the same file name and does not contain useful information security wise.

Conclusion

In conclusion, security products should rely on generic heuristics rather than specific ones tied to public tooling. Moreover, reading the documentation and being curious often lead to interesting discoveries.

If you are curious and liked this article, you might be interested in reading more about Windows secrets extraction.