Modules

Modules

- `crypto` : This modules deals with the Microsoft Crypto Magic world.
- `dpapi` : The Data Protection Application Programming Interface module. Consider this as an opsec safe option (for now) for getting credentials.
- `event` : this module deals with the Windows Event logs (to clear footprints after compromise).
- `kerberos` : This module deals with the Greek Mythology's three headed Hades dog without the help of Hercules.
- `lsadump` : this module contains some well known functionalities of Mimikatz such as DCSync, DCShadow, dumping of SAM and LSA Secrets.
- `misc` : The miscellaneous module contains functionalities such as PetitPotam, PrintNightmare RPC Print Spooler and others.
- `net` : some functionalities in this module are similar to the Windows **net** commands. Enumerating sessions and servers configured with different types of Kerberos delegations is also included.
- `privilege` : This module deals with the Windows privileges. It includes the favorite debug privilege which holds the keys to LSASS.
- `process` : This module deal with Windows processes. It can also be used for process injection and parent process spoofing.
- `rpc` : The Remote Procedure Call module of Mimikatz. It can also be used for controlling Mimikatz remotely.
- `sekurlsa` : The most beloved module of Mimikatz. Even Benjamin has mentioned in the past that one day people will discover that Mimikatz is more than `sekurlsa::logonpasswords`. Hope we made some effort on this Benjamin.
- `service` : This module can interact with Windows services plus installing the `mimikatzsvc` service.
- `sid` : This module deals with the Security Identifier.
-

- `standard` : This module contains some general functionalities which are not related to exploitation.
- `token` : This module deals with the Windows tokens (who does not really like elevating to `NT AUTHORITY\ SYSTEM`).
- `ts` : This module deals with the Terminal Services. It can be an alternative for getting clear-text passwords.
- `vault` : This module dumps passwords saved in the Windows Vault.

Commands

crypto

- `crypto::capi` patches CryptoAPI layer for easy export (Experimental ⚠️)
- `crypto::certificates` lists or exports certificates
- `crypto::certtohw` tries to export a software CA to a crypto (virtual) hardware
- `crypto::cng` patches the CNG (Cryptography API: Next Generation) service for easy export (Experimental ⚠️)
- `crypto::extract` extracts keys from the CAPI RSA/AES provider (Experimental ⚠️)
- `crypto::hash` hashes a password in the main formats (NT, DCC1, DCC2, LM, MD5, SHA1, SHA2) with the username being an optional value
- `crypto::keys` lists or exports key containers
- `crypto::providers` lists cryptographic providers
- `crypto::sc` lists smartcard/token reader(s) on, or deported to, the system. When the CSP (Cryptographic Service Provider) is available, it tries to list keys on the smartcard
- `crypto::scauth` it creates a authentication certificate (smartcard like) from a CA
- `crypto::stores` lists cryptographic stores
- `crypto::system` it describes a Windows System Certificate
- `crypto::tpminfo` displays information for the Microsoft's TPM Platform Crypto Provider

dpapi

- `dpapi::blob` describes a DPAPI blob and unprotects/decrypts it with API or Masterkey
- `dpapi::cache` displays the credential cache of the DPAPI module
- `dpapi::capi` decrypts a CryptoAPI private key file
- `dpapi::chrome` dumps stored credentials and cookies from Chrome
- `dpapi::cloudapkd` is undocumented at the moment
- `dpapi::cloudapreg` dumps azure credentials by querying the following registry location
- `dpapi::cng` decrypts a given CNG private key file
- `dpapi::create` creates a DPAPI Masterkey file from raw key and metadata
- `dpapi::cred` decrypts DPAPI saved credential such as RDP, Scheduled tasks, etc (cf. dumping DPAPI secrets)
- `dpapi::credhist` describes a Credhist file
- `dpapi::luna` decrypts Safenet LunaHSM KSP
- `dpapi::masterkey` describes a Masterkey file and unprotects each Masterkey (key depending). In other words, it can decrypt and request masterkeys from active directory
- `dpapi::protect` protects data via a DPAPI call
- `dpapi::ps` decrypts PowerShell credentials (PSCredentials or SecureString)
- `dpapi::rdg` decrypts Remote Desktop Gateway saved passwords
- `dpapi::sccm` is used to decrypt saved SCCM credentials
- `dpapi::ssh` extracts OpenSSH private keys
- `dpapi::tpm` decrypts TPM PCP key file (Microsoft's TPM Platform Crypto Provider (PCP))
- `dpapi::vault` decrypts DPAPI vault credentials from the Credential Store
- `dpapi::wifi` decrypts saved Wi-Fi passwords
- `dpapi::wwman` decrypts Wwan credentials

event

- `event::clear` clears a specified event log
- `event::drop` patches event services to avoid new events (⚠️ experimental)

kerberos

- `kerberos::ask` can be used to obtain Service Tickets. The Windows native command is `klist get`
- `kerberos::clist` lists tickets in MIT/Heimdall ccache format. It can be useful with other tools (i.e. ones that support Pass the Cache)
- `kerberos::golden` can be used to forge golden and silver tickets. It can also be used for forging inter-realm trust keys
- `kerberos::hash` computes the different types of Kerberos keys for a given password
- `kerberos::list` has a similar functionality to `klist` command without requiring elevated privileges. Unlike `sekurlsa::tickets`, this module does not interact with LSASS
- `kerberos::ptc` can be used to pass the cache. This is similar to `kerberos::ptt` that does pass the ticket but is different in the sense that the ticket used is a `.ccache` ticket instead of a `.kirbi` one
- `kerberos::ptt` is used for passing the ticket by injecting one or may Kerberos tickets in the current session. The ticket can either be a TGT (Ticket-Granting Ticket) or an ST (Service Ticket)
- `kerberos::purge` purges all kerberos tickets similar to `klist purge`
- `kerberos::tgt` retrieves a TGT (Ticket-Granting Ticket) for the current user

lsadump

- `lsadump::backupkeys` dumps the DPAPI backup keys from the Domain Controller (cf. dumping DPAPI secrets)
- `lsadump::cache` can be used to enumerate Domain Cached Credentials from registry. It does so by acquiring the `SysKey` to decrypt `NL$KM` (binary protected value) and then `MSCache(v1/v2)`
- `lsadump::changentlm` can be used to change the password of a user
- `lsadump::dcshadow` TODO
- `lsadump::dcsync` can be used to do a DCSync and retrieve domain secrets. This command uses the Directory Replication Service Remote protocol (MS-DRSR) to request from a domain controller to synchronize a specified entry
- `lsadump::lsa` extracts hashes from memory by asking the LSA server. The `patch` or `inject` takes place on the fly
-

- `lsadump::mbc` dumps the Machine Bound Certificate. Devices on which Credential Guard is enabled are using Machine Bound Certificates
- `lsadump::netsync` can be used to act as a Domain Controller on a target by doing a Silver Ticket. It then leverages the Netlogon to request the RC4 key (i.e. NT hash) of the target computer account
- `lsadump::packages` lists the available Windows authentication mechanisms
- `lsadump::postzerologon` is a procedure to update AD domain password and its local stored password remotely mimic `netdom resetpwd`
- `lsadump::RpData` can retrieve private data (*at the time of writing, Nov 1st 2021, we have no idea what this does or refers to* 🤷)
- `lsadump::sam` dumps the local Security Account Manager (SAM) NT hashes (cf. SAM secrets dump)
- `lsadump::secrets` can be used to dump LSA secrets from the registries. It retrieves the `SysKey` to decrypt `Secrets` entries
- `lsadump::setntlm` can be used to perform a password reset without knowing the user's current password. It can be useful during an active directory Access Control (ACL) abuse scenario
- `lsadump::trust` can be used for dumping the forest trust keys. Forest trust keys can be leveraged for forging inter-realm trust tickets. Since most of the EDRs are paying attention to the KRBTGT hash, this is a stealthy way to compromise forest trusts
- `lsadump::zerologon` detects and exploits the ZeroLogon vulnerability

misc

- `misc::aadcookie` can be used to dump the Azure Panel's session cookie from `login.microsoftonline.com`
- `misc::clip` monitors clipboard. `CTRL+C` stops the monitoring
- `misc::cmd` launches the command prompt
- `misc::compress` performs a self compression of mimikatz
- `misc::detours` is experimental and it tries to enumerate all modules with Detours-like hooks
- `misc::efs` is Mimikatz's implementation of the MS-EFSR abuse (PetitPotam), an authentication coercion technique
- `misc::lock` locks the screen. It can come in handy with `misc::memssp`
- `misc::memssp` patches LSASS by injecting a new Security Support Provider (a DLL is registered)
-

- `misc::mflt` identifies Windows minifilters inside mimikatz, without using **fltmc.exe**. It can also assist in fingerprinting security products, by altitude too (Gathers details on loaded drivers, including driver altitude)
- `misc::ncroutemon` displays Juniper network connect (without route monitoring)
- `misc::ngcsign` can be used to dump the NGC key (Windows Hello keys) signed with the symmetric pop key.
- `misc::printnightmare` can be used to exploit the PrintNightMare vulnerability in both [MS-RPRN RpcAddPrinterDriverEx] and [MS-PAR AddPrinterDriverEx]. The bug was discovered by Zhiniang Peng (@edwardzpeng) & Xuefeng Li (@lxf02942370)
- `misc::regedit` launches the registry editor
- `misc::sccm` decrypts the password field in the `SC_UserAccount` table in the SCCM database
- `misc::shadowcopies` is used to list the available shadow copies on the system
- `misc::skeleton` injects a "Skeleton Key" into the LSASS process on the domain controller
- `misc::spooler` is Mimikat's implementation of the MS-RPRN abuse (PrinterBug), an authentication coercion technique
- `misc::taskmgr` launches the task manager
- `misc::wp` sets up a wallpaper
- `misc::xor` performs XOR decoding/encoding on a provided file with `0x42` default key

net

- `net::alias` displays more information about the local group memberships including Remote Desktop Users, Distributed COM Users, etc
- `net::deleg` checks for the following types of Kerberos delegations
- `net::group` displays the local groups
- `net::if` displays the available local IP addresses and the hostname
- `net::serverinfo` displays information about the logged in server
- `net::session` displays the active sessions through NetSessionEnum() Win32 API function
- `net::share` displays the available shares
- `net::stats` displays when the target was booted
- `net::tod` displays the current time

- `net::trust` displays information for the active directory forest trust(s)
- `net::user` displays the local users
- `net::wsession` displays the active sessions through NetWkstaUserEnum() Win32 API function

privilege

- `privilege::backup` requests the backup privilege (`SeBackupPrivilege`)
- `privilege::debug` requests the debug privilege (`SeDebugPrivilege`)
- `privilege::driver` requests the load driver privilege (`SeLoadDriverPrivilege`)
- `privilege::id` requests a privilege by its `id`
- `privilege::name` requests a privilege by its name
- `privilege::restore` requests the restore privilege (`SeRestorePrivilege`)
- `privilege::security` requests the security privilege (`SeSecurityPrivilege`)
- `privilege::sysenv` requests the system environment privilege (`SeSystemEnvironmentPrivilege`)
- `privilege::tcb` requests the tcb privilege (`SeTcbPrivilege`)

process

- `process::exports` lists all the exported functions from the DLLs each running process is using. If a**** `/pid` is not specified, then exports for `mimikatz.exe` will be displayed
- `process::imports` lists all the imported functions from the DLLs each running process is using. If a**** `/pid` is not specified, then imports for `mimikatz.exe` will be displayed
- `process::list` lists all the running processes. It uses the NtQuerySystemInformation Windows Native API function
- `process::resume` resumes a suspended process by using the NtResumeProcess Windows Native API function
- `process::run` creates a process by using the CreateProcessAsUser Win32 API function. The CreateEnvironmentBlock is also utilized
- `process::runp` runs a subprocess under a parent process (Default parent process is `LSASS.exe`). It can also be used for lateral movement and process spoofing
-

- `process::start` starts a process by using the CreateProcess Win32 API function. The `PID` of the process is also displayed
- `process::stop` terminates a process by using the NtTerminateProcess Windows Native API function. The Win32 API equal one is TerminateProcess
- `process::suspend` suspends a process by using the NtSuspendProcess Windows Native API function

rpc

- `rpc::close` closes remote RPC sessions
- `rpc::connect` connects to an RPC endpoint
- `rpc::enum` enumerates RPC endpoints on a system
- `rpc::server` starts an RPC server

sekurlsa

- `sekurlsa::backupkeys` lists the preferred Backup Master keys
- `sekurlsa::bootkey` sets the SecureKernel Boot Key and attempts to decrypt LSA Isolated credentials
- `sekurlsa::cloudap` lists Azure (Primary Refresh Token) credentials based on the following research: Digging further into the Primary Refresh Token. According to Benjamin:
- `sekurlsa::credman` lists Credentials Manager by targeting the Microsoft Local Security Authority Server DLL (lsasrv.dll)
- `sekurlsa::dpapi` lists DPAPI cached masterkeys
- `sekurlsa::dpapisystem` lists the `DPAPI_SYSTEM` secret key
- `sekurlsa::ekeys` lists Kerberos encryption keys
- `sekurlsa::kerberos` lists Kerberos credentials
- `sekurlsa::krbtgt` retrieves the krbtgt RC4 (i.e. NT hash), AES128 and AES256 hashes
- `sekurlsa::livessp` lists LiveSSP credentials. According to Microsoft, the LiveSSP provider is included by default in Windows 8 and later and is included in the Office 365 Sign-in Assistant
- `sekurlsa::logonpasswords` lists all available provider credentials. This usually shows recently logged on user and computer credentials
-

`sekurlsa::minidump` can be used against a dumped LSASS process file and it does not require administrative privileges. It's considered as an "offline" dump

- `sekurlsa::msv` dumps and lists the NT hash (and other secrets) by targeting the MSV1_0 Authentication Package
- `sekurlsa::process` switches (or reinits) to LSASS process context. It can be used after `sekurlsa::minidump`
- `sekurlsa::pth` performs Pass-the-Hash, Pass-the-Key and Over-Pass-the-Hash. Upon successful authentication, a program is run (n.b. defaulted to `cme.exe`)
- `sekurlsa::ssp` lists Security Support Provider (SSP) credentials
- `sekurlsa::tickets` lists Kerberos tickets belonging to all authenticated users on the target server/workstation. Unlike `kerberos::list` , sekurlsa uses memory reading and is not subject to key export restrictions. Sekurlsa can also access tickets of others sessions (users)
- `sekurlsa::trust` retrieves the forest trust keys
- `sekurlsa::tspkg` lists TsPkg credentials. This credentials provider is used for Terminal Server Authentication
- `sekurlsa::wdigest` lists WDigest credentials. According to Microsoft, WDigest.dll was introduced in the Windows XP operating system

service

- `service::-` removes the `mimikatzsvc` service
- `service::+` installs the `mimikatzsvc` service by issuing `rpc::server service::me exit`
- `service::preshutdown` pre-shuts down a specified service by sending a `SERVICE_CONTROL_PRESHUTDOWN` signal
- `service::remove` removes the specified service (It must be used with caution)
- `service::resume` resumes a specified service, after successful suspending, by sending a `SERVICE_CONTROL_CONTINUE` signal
- `service::shutdown` shuts down a specified service by sending a `SERVICE_CONTROL_SHUTDOWN` signal
- `service::start` starts a service
- `service::stop` stops a specified service by sending a `SERVICE_CONTROL_STOP` signal
- `service::suspend` suspends the specified service. It sends a `SERVICE_CONTROL_PAUSE` signal

sid

- `sid::add` adds a SID to `sIDHistory` of an object
- `sid::clear` clears the `sIDHistory` of a target object
- `sid::lookup` looks up an object by its SID or name
- `sid::modify` modifies an object's SID
- `sid::patch` patchs the NTDS (NT Directory Services). It's useful when running `id::modify` or `sid::add`
- `sid::query` queries an object by its SID or name

standard

- `standard::answer` or `answer` provides an answer to The Ultimate Question of Life, the Universe, and Everything!
- `standard::base64` or `base64` switches file input/output to base64
- `standard::cd` or `cd` can change or display the current directory. The changed directory is used for saving files
- `standard::cls` or `cls` clears the screen
- `standard::coffee` or `coffee` is the most important command of all
- `standard::exit` or `exit` quits Mimikatz after clearing routines
- `standard::hostname` or `hostname` displays system local hostname
- `standard::localtime` or `localtime` displays system local date and time
- `standard::log` or `log` logs mimikatz input/output to a file
- `standard::sleep` or `sleep` make Mimikatz sleep an amount of milliseconds
- `standard::version` or `version` displays the version in use of Mimikatz

token

-

`token::elevate` can be used to impersonate a token. By default it will elevate permissions to `NT AUTHORITY\SYSTEM`

- `token::list` lists all tokens on the system
- `token::revert` reverts to the previous token
- `token::run` executes a process with its token
- `token::whoami` displays the current token

ts

- `ts::logonpasswords` extracts clear text credentials from RDP running sessions (server side)
- `ts::mstsc` extracts cleartext credentials from the mstsc process (client side)
- `ts::multirdp` enables multiple RDP connections on the target server
- `ts::remote` performs RDP takeover/hijacking of active sessions
- `ts::sessions` lists the current RDP sessions. It comes in handy for RDP hijacking

vault

- `vault::cred` enumerates vault credentials
- `vault::list` lists saved credentials in the Windows Vault such as scheduled tasks, RDP, Internet Explorer for the current user

Previous
General 🛠️

Next
crypto

Last updated 2 years ago