Sharpening Techniques with Impacket

redteamrecipe.com/sharpening-techniques-with-impacketrtc0025

Reza Rashidi

DumpNTLMInfo

Techniques Used

`DumpNTLMInfo.py` is likely a Python script that utilizes Impacket's capabilities to interact with network protocols. The primary technique it might use is the extraction or dumping of NTLM (NT LAN Manager) information from a specified target. NTLM is a suite of Microsoft security protocols intended to provide authentication, integrity, and confidentiality to users.

URL

URL: DumpNTLMInfo.py on GitHub

Description

Assuming the typical functionality of Impacket scripts, `DumpNTLMInfo.py` would be a tool for extracting NTLM authentication details from a target system. This could include gathering NTLM hashes, which are often a target for attackers due to their potential use in pass-the-hash attacks. The script might interact with services like SMB (Server Message Block) or others that utilize NTLM for authentication.

Get-GPPPassword

Techniques Used

The script likely uses techniques to retrieve and decrypt passwords stored in Group Policy Preferences files. These files are often used by system administrators to configure settings on Windows machines across a network. Historically, they have been known to store credentials in an encrypted form, but the encryption key was publicly disclosed by Microsoft, making it possible to decrypt these passwords.

URL

URL: Get-GPPPassword.py on GitHub

Description

`Get-GPPPassword.py` is probably designed to automate the process of finding, retrieving, and decrypting passwords stored in GPP files. This script would typically target XML files that contain these encrypted passwords, such as `Groups.xml`, `Services.xml`, `Scheduledtasks.xml`, `Datasources.xml`, and `Printers.xml`. Once it locates these files, it uses the known decryption key to reveal the plaintext passwords.

GetADUsers

Techniques Used

`GetADUsers.py` likely employs techniques to query Active Directory for user account information. This can include retrieving details such as usernames, account settings, and other attributes stored in AD. The script may use LDAP (Lightweight Directory Access Protocol) queries or similar methods to extract this information from the AD database.

URL

URL: GetADUsers.py on GitHub

Description

The primary function of `GetADUsers.py` is probably to enumerate user accounts in an Active Directory environment. This can be particularly useful in penetration testing and security auditing scenarios, where understanding the user landscape of a target network is crucial.

GetNPUsers

Techniques Used

`GetNPUsers.py` likely utilizes techniques to exploit the Kerberos protocol in Active Directory environments. Specifically, it targets accounts where the Kerberos pre-authentication is not required. By doing so, the script can request a Kerberos Ticket

Granting Service (TGS) ticket for any user without needing their password. This ticket can then be used to attempt offline cracking of the user's password.

URL

URL: GetNPUsers.py on GitHub

Description

The primary function of `GetNPUsers.py` is to enumerate and exploit AD user accounts that are vulnerable due to the 'Do not require Kerberos preauthentication' setting. This setting, if enabled, means that the account does not require the initial proof of knowledge of the password to start the Kerberos authentication process.

The script typically works by:

- Enumerating users in the AD who have the 'Do not require Kerberos preauthentication' setting enabled.
- Requesting TGS tickets from the Kerberos Key Distribution Center (KDC) for these users.
- These tickets are encrypted with the user's password hash, which can then be extracted and subjected to offline password cracking techniques.

GetUserSPNs

The `GetUserSPNs.py` script from the Impacket suite is designed to interact with Service Principal Names (SPNs) in Active Directory (AD) environments. Here's an overview based on the typical functionality of such scripts:

Techniques Used

`GetUserSPNs.py` likely employs techniques to enumerate and request service tickets for accounts that have registered SPNs in an AD environment. In Kerberos, SPNs are used to uniquely identify a service instance. When a service ticket is requested for an SPN, it is encrypted with the service account's password. This script can be used to request these tickets, which can then be subjected to offline password cracking attempts.

URL

URL: GetUserSPNs.py on GitHub

Description

The primary function of `GetUserSPNs.py` is to identify and exploit potential vulnerabilities associated with service accounts in Active Directory. The script:

- Enumerates user accounts with registered SPNs.
- Requests Kerberos tickets for these SPNs.

- Retrieves these tickets, which are encrypted with the account's password hash.

addcomputer

Techniques Used

`addcomputer.py` likely employs techniques to automate the process of adding a new computer account to an Active Directory domain. This can involve using LDAP (Lightweight Directory Access Protocol) or other AD protocols to communicate with the domain controller and create a new computer account. The script might also handle setting or modifying various attributes associated with the computer account.

URL

URL: addcomputer.py on GitHub

Description

The primary function of `addcomputer.py` is to facilitate the addition of computer accounts to an AD domain. This can be particularly useful for system administrators and IT professionals who need to automate the process of joining multiple computers to a domain.

Key features of the script might include:

- Creating a new computer account in a specified organizational unit (OU).
- Setting specific attributes for the computer account, such as its name, description, and security settings.
- Automating the process of joining a computer to a domain in bulk operations.

atexec

echniques Used

`atexec.py` is likely a Python script designed to execute commands on a remote machine in a Windows environment. This script probably uses the AT protocol, a legacy scheduling system, to remotely execute commands. It might interact with the Windows Task Scheduler service to schedule tasks (commands) to be executed on a remote machine.

URL

URL: atexec.py on GitHub

Description

The primary function of `atexec.py` would be to facilitate remote command execution on Windows systems. This can be particularly useful for system administrators, IT professionals, and penetration testers for various purposes, such as:

- Remotely managing systems by executing necessary commands or scripts.
- Testing the security of networked systems by assessing how they handle unsolicited task scheduling and command execution.
- Automating tasks across multiple machines in a networked environment.

changepasswd

Techniques Used

`changepasswd.py` likely employs techniques to interact with AD services for password management. This can involve using network protocols such as LDAP (Lightweight Directory Access Protocol) or SMB (Server Message Block) to communicate with AD controllers and execute password change requests. The script might handle authentication, encryption, and the necessary protocol interactions to securely change a user's password.

URL

URL: changepasswd.py on GitHub

Description

The primary function of `changepasswd.py` is to facilitate the changing of passwords for user accounts in an AD domain. This can be particularly useful for system administrators and IT professionals who need to manage user credentials, either for routine password updates or in response to security incidents.

Key features of the script might include:

- Automating the process of changing passwords for multiple accounts.
- Providing a secure and efficient way to update credentials in a networked environment.
- Ensuring compliance with organizational password policies and security standards.

dcomexec

Techniques Used

`dcomexec.py` likely employs techniques to interact with Windows systems using the DCOM protocol. DCOM is a Microsoft technology for communication among software components distributed across networked computers. The script probably uses this protocol to remotely execute commands or scripts on a target machine, leveraging DCOM's ability to communicate across different network segments.

URL

URL: dcomexec.py on GitHub

Description

The primary function of `dcomexec.py` is to facilitate remote command execution on Windows systems using DCOM. This can be particularly useful for system administrators, IT professionals, and penetration testers for various purposes, such as:

- Remotely managing systems by executing necessary commands or scripts.
- Testing the security of networked systems by assessing how they handle remote execution requests.
- Automating tasks across multiple machines in a networked environment.

describeTicket

Techniques Used

`describeTicket.py` likely employs techniques to parse and interpret Kerberos tickets. Kerberos is a network authentication protocol commonly used in Windows Active Directory environments. The script probably analyzes the tickets to extract and display information such as the ticket's encryption type, validity period, and the identities of the principal and target server.

URL

URL: describeTicket.py on GitHub

Description

The primary function of `describeTicket.py` is to provide a detailed description of Kerberos tickets. This can be particularly useful for security professionals and system administrators for purposes such as:

- Analyzing and debugging Kerberos authentication issues.
- Understanding the security characteristics of Kerberos tickets, such as the encryption types used.
- Auditing and monitoring Kerberos ticket usage within a network environment.

dpapi

Techniques Used

`dpapi.py` likely employs techniques to interact with the Windows Data Protection API, which is used to provide encryption and decryption capabilities for protecting data such as passwords and keys. The script might handle tasks like decrypting DPAPI-protected data,

which can be crucial in forensic analysis or during security assessments to access protected information.

URL

URL: dpapi.py on GitHub

Description

The primary function of `dpapi.py` is to facilitate the decryption and analysis of data protected by the Windows Data Protection API. This can be particularly useful for security professionals and system administrators for purposes such as:

- Decrypting data that has been encrypted using DPAPI, which can include passwords, keys, and other sensitive information.
- Conducting forensic analysis on Windows systems to recover protected data.
- Auditing and testing the security of systems that use DPAPI for data protection.

esentutl

Techniques Used

`esentutl.py` likely employs techniques to interact with ESE database files, also known as JET Blue databases. These databases are used by various Windows services and applications, including Active Directory, Exchange, and Windows Update. The script might handle tasks like reading, extracting, or manipulating data within ESE database files, which can be crucial in forensic analysis or during security assessments.

URL

URL: esentutl.py on GitHub

Description

The primary function of `esentutl.py` is to facilitate the interaction with ESE database files. This can be particularly useful for security professionals and system administrators for purposes such as:

- Extracting information from ESE databases for forensic analysis.
- Auditing and testing the integrity and security of data stored in ESE databases.
- Understanding the structure and contents of ESE databases for troubleshooting or analysis.

findDelegation

Techniques Used

`findDelegation.py` likely employs techniques to query Active Directory for delegation settings on user and computer accounts. Delegation in AD refers to the ability of one account to act on behalf of another. The script might use LDAP (Lightweight Directory Access Protocol) queries or similar methods to identify accounts with delegation permissions, which can be a significant aspect in assessing the security posture of an AD environment.

URL

URL: findDelegation.py on GitHub

Description

The primary function of `findDelegation.py` is to enumerate and report on delegation settings within an Active Directory domain. This can be particularly useful for security professionals conducting authorized assessments of AD environments.

Key features of the script might include:

- Identifying user and computer accounts with unconstrained, constrained, or resource-based constrained delegation settings.
- Helping in the assessment of potential security risks associated with improper delegation configurations.
- Assisting in auditing and compliance checks related to AD delegation policies.

getArch

Techniques Used

`getArch.py` likely employs techniques to remotely identify the architecture (32-bit or 64-bit) of a Windows operating system on a networked computer. This can involve using network protocols such as SMB (Server Message Block) to interact with the remote system and gather information about its architecture. This kind of information is crucial for tailoring further attacks or assessments to the specific environment of the target system.

URL

URL: getArch.py on GitHub

Description

The primary function of `getArch.py` is to remotely determine the architecture of a Windows machine in a network. This can be particularly useful for IT professionals, system administrators, and security researchers for purposes such as:

- Preparing for deployment of software or patches that are architecture-specific.

- Conducting security assessments or penetration tests where knowing the target's architecture is crucial for exploiting certain vulnerabilities.
- Automating inventory management in a networked environment to keep track of different system architectures.

getPac

Techniques Used

`getPac.py` likely employs techniques to interact with and analyze the PAC, which is a part of the Kerberos ticket in AD environments. The PAC contains user authorization information, such as group memberships and user rights. The script might use network protocols and authentication mechanisms to request and retrieve PAC data from a domain controller, which can be crucial for understanding user privileges and roles in an AD domain.

URL

URL: getPac.py on GitHub

Description

The primary function of `getPac.py` is to facilitate the retrieval and analysis of PAC data within Kerberos tickets in an AD domain. This can be particularly useful for security professionals conducting authorized assessments of AD environments.

Key features of the script might include:

- Extracting PAC data from Kerberos tickets to analyze user privileges and group memberships.
- Assisting in auditing and compliance checks related to user rights and access controls in AD.
- Helping in forensic analysis and security investigations by providing detailed information about user authorizations.

getST

Techniques Used

`getST.py` likely employs techniques to interact with the Kerberos protocol in AD environments. The script might be used to request Service Tickets for specific services within the domain. This process involves authenticating to the Kerberos Key Distribution Center (KDC) and requesting tickets that grant access to specific services. These tickets can be crucial for understanding access controls and permissions within an AD domain.

URL

URL: getST.py on GitHub

Description

The primary function of `getST.py` is to facilitate the acquisition of Kerberos Service Tickets for different services in an AD domain. This can be particularly useful for security professionals conducting authorized assessments of AD environments.

Key features of the script might include:

- Requesting and obtaining Service Tickets for specified services or users.
- Assisting in auditing and compliance checks related to Kerberos-based authentication and access controls in AD.
- Helping in forensic analysis and security investigations by providing detailed information about service access and permissions.

getTGT

Techniques Used

`getTGT.py` likely employs techniques to interact with the Kerberos protocol in AD environments. The script might be used to request Ticket Granting Tickets, which are essential for Kerberos-based authentication. This process involves authenticating to the Kerberos Key Distribution Center (KDC) and requesting a ticket that can be used to obtain service tickets for various services within the domain.

URL

URL: getTGT.py on GitHub

Description

The primary function of `getTGT.py` is to facilitate the acquisition of Kerberos Ticket Granting Tickets in an AD domain. This can be particularly useful for security professionals conducting authorized assessments of AD environments.

Key features of the script might include:

- Requesting and obtaining TGTs for specified users.
- Assisting in auditing and compliance checks related to Kerberos-based authentication in AD.
- Helping in forensic analysis and security investigations by providing detailed information about authentication processes and access permissions.

goldenPac

Techniques Used

`goldenPac.py` likely employs techniques related to the creation and use of a Golden Ticket. A Golden Ticket is a forged Ticket Granting Ticket (TGT) in Kerberos, which can be used to gain unauthorized access to any service in an AD domain. The script might use vulnerabilities or misconfigurations in the AD environment to create a TGT that is trusted by the Key Distribution Center (KDC), allowing broad access across the network.

URL

URL: goldenPac.py on GitHub

Description

The primary function of `goldenPac.py` is to exploit the Kerberos protocol in an AD domain by creating a Golden Ticket. This can be particularly useful (and dangerous) for security professionals conducting authorized assessments of AD environments or for malicious actors.

Key features of the script might include:

- Creating a forged TGT that provides extensive access rights within an AD domain.
- Bypassing standard authentication and authorization processes in AD.
- Potentially maintaining persistent and undetected access to a network.

karmaSMB

Techniques Used

`karmaSMB.py` likely employs techniques to exploit weaknesses in the SMB protocol, which is used for network file sharing in Windows environments. The script might simulate an SMB server to intercept and manipulate SMB traffic, potentially allowing unauthorized access to network resources or sensitive information.

URL

URL: karmaSMB.py on GitHub

Description

The primary function of `karmaSMB.py` is to exploit the SMB protocol for various purposes such as unauthorized access, data interception, or network reconnaissance. This can be particularly useful (and dangerous) for security professionals conducting authorized assessments or for malicious actors.

Key features of the script might include:

- Setting up a rogue SMB server to intercept SMB requests.
- Exploiting vulnerabilities in SMB clients or servers to gain unauthorized access or information.

- Assisting in penetration testing and red team exercises to identify and mitigate SMB-related vulnerabilities.

keylistattack

Techniques Used

Given the name `keylistattack.py`, the script likely involves techniques related to cryptographic keys or credentials. It might be designed to exploit vulnerabilities related to key management or authentication processes in networked environments. The script could potentially be used for tasks like extracting, decrypting, or exploiting keys or credentials transmitted over a network.

URL

URL: keylistattack.py on GitHub

Description

Without specific details, it's challenging to provide an accurate description of `keylistattack.py`. However, in general, a script with this name in the Impacket suite would be used for security testing and network analysis purposes, potentially focusing on:

- Identifying and exploiting vulnerabilities related to key management or authentication in network protocols.
- Extracting or decrypting keys or credentials transmitted over the network.
- Conducting penetration testing or security assessments to identify and mitigate potential security risks related to cryptographic keys or credentials.

kintercept

Techniques Used

Given the name `kintercept.py`, the script likely involves techniques related to network interception or manipulation. It might be designed to capture, analyze, or modify network traffic, potentially focusing on Kerberos protocol traffic or other authentication-related data in a networked environment.

URL

URL: kintercept.py on GitHub

Description

Without specific details, it's challenging to provide an accurate description of `kintercept.py`. However, in general, a script with this name in the Impacket suite would be used for security testing and network analysis purposes, potentially focusing on:

- Intercepting and analyzing network traffic, particularly focusing on authentication protocols like Kerberos.
- Exploiting vulnerabilities related to network traffic interception or manipulation.
- Conducting penetration testing or security assessments to identify and mitigate potential security risks related to network traffic.

lookupsid

Techniques Used

`lookupsid.py` likely employs techniques to enumerate and resolve SIDs (Security Identifiers) in a Windows network. SIDs are unique identifiers for user and group accounts in Windows. The script might use network protocols such as SMB (Server Message Block) or RPC (Remote Procedure Call) to query Windows systems or domain controllers to resolve SIDs to their corresponding account names.

URL

URL: lookupsid.py on GitHub

Description

The primary function of `lookupsid.py` is to enumerate and resolve SIDs in a Windows network environment. This can be particularly useful for IT professionals, system administrators, and security researchers for purposes such as:

- Identifying user and group accounts in a Windows network based on their SIDs.
- Conducting security assessments or penetration tests where resolving SIDs to account names is necessary for understanding network permissions and roles.
- Auditing and compliance checks to ensure proper account management and security configurations.

machine_role

Techniques Used

Given the name `machine_role.py`, the script likely involves techniques related to network reconnaissance and system identification. It might be designed to query networked machines to determine their roles, such as whether they are domain controllers, file servers, or workstations. This information is often crucial in network management and security assessments.

URL

URL: machine_role.py on GitHub

Description

Without specific details, it's challenging to provide an accurate description of `machine_role.py`. However, in general, a script with this name in the Impacket suite would be used for purposes like:

- Identifying the roles and functions of machines in a networked environment.
- Assisting in network mapping and reconnaissance during security assessments or penetration testing.
- Gathering information that can be used to tailor further security testing or administrative actions to the specific network architecture.

mimikatz

Techniques Used

`mimikatz.py` likely employs techniques to interface with or replicate some functionalities of Mimikatz. This might include extracting credentials from memory, performing pass-the-hash attacks, or other activities related to credential theft and manipulation on Windows systems.

URL

URL: mimikatz.py on GitHub

Description

The primary function of `mimikatz.py` is to assist in security testing and penetration testing activities by providing capabilities similar to those of Mimikatz. This can include:

- Extracting plaintext passwords, hashes, and Kerberos tickets from memory.
- Performing pass-the-hash or pass-the-ticket attacks.
- Assisting in security assessments by identifying and exploiting weaknesses in Windows authentication mechanisms.

mqtt_check

Techniques Used

Given the name `mqtt_check.py`, the script likely involves techniques related to the MQTT protocol, which is a lightweight messaging protocol used for small sensors and mobile devices. It might be designed to test the security of MQTT brokers, check for vulnerabilities, or perform reconnaissance on MQTT-based communication in a networked environment.

URL

URL: mqtt_check.py on GitHub

Description

Without specific details, it's challenging to provide an accurate description of `mqtt_check.py`. However, in general, a script with this name in the Impacket suite would be used for purposes like:

- Testing the security and configuration of MQTT brokers.
- Identifying vulnerabilities or misconfigurations in MQTT-based communication systems.
- Gathering information for network reconnaissance in environments where MQTT is used.

mssqlclient

Techniques Used

`mssqlclient.py` likely involves techniques for connecting to, querying, and potentially exploiting Microsoft SQL Server databases. This might include running SQL queries, executing commands, or exploiting SQL Server features for various purposes, including both legitimate database management and security testing.

URL

URL: mssqlclient.py on GitHub

Description

Without specific details, it's challenging to provide an accurate description of `mssqlclient.py`. However, in general, a script with this name in the Impacket suite would be used for purposes like:

- Connecting to Microsoft SQL Server databases for querying and database management.
- Executing SQL commands or stored procedures.
- Potentially exploiting SQL Server vulnerabilities or misconfigurations for security testing and penetration testing purposes.

mssqlinstance

Techniques Used

`mssqlinstance.py` likely involves techniques for discovering and interacting with Microsoft SQL Server instances on a network. This might include identifying active SQL Server instances, determining their versions, and potentially gathering other relevant information about the database environment.

URL

URL: mssqlinstance.py on GitHub

Description

Without specific details, it's challenging to provide an accurate description of `mssqlinstance.py`. However, in general, a script with this name in the Impacket suite would be used for purposes like:

- Discovering Microsoft SQL Server instances on a network.
- Enumerating information about these instances, such as version numbers, available databases, and configuration details.
- Potentially assisting in the initial stages of a security assessment or penetration test by mapping out SQL Server resources within a target network.

net

Techniques Used

`net.py` likely involves techniques for network communication, enumeration, and possibly exploitation. This might include tasks like scanning for open ports, identifying network services, and potentially exploiting vulnerabilities in network protocols or services.

URL

URL: net.py on GitHub

Description

Without specific details, it's challenging to provide an accurate description of `net.py`. However, in general, a script with this name in the Impacket suite would be used for purposes like:

- Conducting network reconnaissance to identify active hosts and services.
- Enumerating network resources and configurations.
- Potentially assisting in penetration testing or security assessments by identifying vulnerabilities or misconfigurations in network services.

netview

Techniques Used

`netview.py` likely involves techniques for viewing and analyzing network structures and devices. This might include tasks like enumerating network shares, users, and devices, as well as potentially identifying network configurations and vulnerabilities.

URL

URL: netview.py on GitHub

Description

Without specific details, it's challenging to provide an accurate description of `netview.py`. However, in general, a script with this name in the Impacket suite would be used for purposes like:

- Enumerating and analyzing network resources such as shares, users, and devices.
- Assisting in network mapping and reconnaissance during security assessments or penetration testing.
- Identifying potential vulnerabilities or misconfigurations in network setups.

nmapAnswerMachine

Techniques used:

- NMAP scripting - the script is designed to work with NMAP to scan ports and respond to requests
- Port scanning - it scans TCP ports on the target system to find open ones
- Crafted responses - it sends crafted responses to hide open ports and make the system appear offline

URL:
https://github.com/fortra/impacket/blob/master/examples/nmapAnswerMachine.py

Description: This Python script works as an "answer machine" for NMAP port scans. When NMAP scans ports on the system running this script, it will send crafted responses to make those ports appear closed. This hides open ports from the NMAP scan to make the system seem offline and prevent fingerprinting. The script listens on the local interface for SYN scan probes from NMAP and sends back spoofed RST packets in response. It does this for all TCP ports except 110, 139, and 445 which are left open to mimic a Windows machine.

ntfs-read

Techniques used:

- Impersonation - the script impersonates a domain user to gain access
- SMB protocol - communicates with the target using the SMB protocol
- Mounting drives - mounts the target NTFS volume locally to read files

URL: **https://github.com/fortra/impacket/blob/master/examples/ntfs-read.py**

Description: This Python script allows reading files from NTFS drives by leveraging SMB connections and user impersonation. It takes a username, password, target, and optionally a domain as input. Using this credential information, it impersonates the user to mount the target NTFS drive over SMB. By default it will mount the C drive. Once

mounted, it can list paths and read any accessible files on the remote NTFS volume just as if it were a local drive. This allows directory enumeration and file access via the SMB protocol by essentially masquerading as an authenticated domain user to the target.

ntlmrelayx

Techniques used:

- NTLM relay - intercepts NTLM authentication and relays it to other services
- SMB relay - relays NTLM auth to SMB shares and executes commands
- HTTP server - contains an HTTP server for NTLM relay over HTTP
- SMB client - connects to SMB shares for file manipulation

URL: **https://github.com/fortra/impacket/blob/master/examples/ntlmrelayx.py**

Description: ntlmrelayx.py is a tool that performs NTLM relay attacks by intercepting authentication requests to one service and relaying them to either SMB shares or an HTTP server. It listens for incoming NTLM authentication attempts, captures the challenge/response, and replays it to authenticate as the user to different services. This can be used to relay authentication to SMB servers to enumerate shares and read/write files. It can also relay NTLM to an HTTP server under the attacker's control to execute commands. Additionally it contains an HTTP server specifically for NTLM relay. The tool can relay credentials to access services the user is authorized for but attackers are not. This allows escalation of privileges and lateral movement.

ping

Techniques used:

- ICMP - uses ICMP echo requests and replies to ping hosts
- Raw sockets - creates raw sockets to send and receive ICMP packets
- Privilege escalation - must run as root/admin to create raw sockets

URL: **https://github.com/fortra/impacket/blob/master/examples/ping.py**

Description: This Python script allows pinging hosts using ICMP echo requests and replies. It creates raw ICMP sockets which requires admin privileges. After creating the raw sockets, it can then send ICMP echo request packets to the target hosts and listen for ICMP echo reply packets back. Based on the replies, it calculates and prints out statistics showing how long it takes to receive replies from the target. This allows basic network connectivity testing by leveraging raw ICMP socket communication.

ping6

Techniques used:

- ICMPv6 - uses ICMPv6 echo requests and replies to ping IPv6 hosts

- Raw sockets - creates raw sockets to send and receive ICMPv6 packets
- Privilege escalation - must run as root/admin to create raw sockets

URL: **https://github.com/fortra/impacket/blob/master/examples/ping6.py**

Description: This Python script allows pinging IPv6 hosts using ICMPv6 echo requests and replies. It creates raw ICMPv6 sockets which requires admin privileges. After creating the raw sockets, it can then send ICMPv6 echo request packets to the target IPv6 hosts and listen for ICMPv6 echo reply packets back. Based on the replies, it calculates and prints out statistics showing how long it takes to receive replies from the target. This allows basic IPv6 network connectivity testing by leveraging raw ICMPv6 socket communication.

psexec

Techniques used:

- PSEXEC - uses Windows sysinternals psexec functionality
- SMB connections - communicates with the target using SMB
- Command execution - executes commands on the remote host
- Lateral movement - gains access to remote machines on a network

URL: **https://github.com/fortra/impacket/blob/master/examples/psexec.py**

Description:

This Python script utilizes Windows sysinternals psexec to execute commands on remote Windows machines through SMB connections. It allows passing usernames and passwords/hashes to authenticate and connect to the remote host's C$ admin share through SMB. From there is uses the psexec functionality to copy and execute commands in the context of the user on the remote host. This allows lateral movement and command execution on other machines with valid credentials. It also supports passing hash values if credentials are captured. The script handles all the SMB communication and authentication to leverage the Windows psexec methods for executing processes on other machines on the network.

raiseChild

Techniques used:

- Process control - injects code into a new child process
- Code injection - injects DLL into subprocess memory space
- Privilege escalation - elevates privileges by injecting into parent process

URL: **https://github.com/fortra/impacket/blob/master/examples/raiseChild.py**

Description: This Python script allows spawning new processes and injecting arbitrary DLLs into their memory. It leverages the Windows API to create a new suspended child process of the parent python.exe process. It then writes the bytes of the specified DLL into the new process's memory space. Finally, it resumes execution which causes the DLL to execute within the child process. By spawning from the parent Python process it inherits the parent's level of privileges, allowing escalation.

rbcd

Techniques used:

- DLL injection - injects DLL into lsass.exe process
- Credential dumping - dumps credentials from memory with injected DLL
- LSASS access - leverages access to LSASS process to steal creds

URL: **https://github.com/fortra/impacket/blob/master/examples/rbcd.py**

Description: This Python script provides the capability to perform credential dumping by injecting a DLL into the lsass.exe process. It leverages access to lsass in order to load a custom RBC DLL which can then access memory and dump credentials. The script uses Windows API calls to open the lsass process, allocate memory, write the encoded DLL, and create a remote thread to call it. This gives execution inside lsass, allowing the DLL to extract credentials from memory. These credentials can then be used for lateral movement, pivoting, and escalation.

rdp_check

Techniques used:

- RDP Protocol - Connects via RDP protocol
- Brute force - Tries credentials against target RDP services
- Remote Desktop - Tests connectivity and login to RDP services

URL: **https://github.com/fortra/impacket/blob/master/examples/rdp_check.py**

Description: This Python script tests RDP services for valid account credentials. It takes a target, username file, password file and optional domain and attempts to connect to the target's RDP service using found account/password combinations. For each user/pass pair found in the files, it tries to authenticate to verify a valid combination. If successful, it returns that the credentials are valid for RDP access to that target. This allows quickly brute forcing and checking remote desktop credentials.

rpcdump

Techniques used:

- RPC inspection - Extracts info from RPC services

- SMB client - Connects to target RPC services over SMB
- Discovery - Enumerates and gathers data for recon

URL: **https://github.com/fortra/impacket/blob/master/examples/rpcdump.py**

Description: This Python script dumps information from remote RPC services by connecting over SMB. It extracts details from the RPC backend that can be used for reconnaissance and discovery. By passing target host details and optional credentials, rpcdump.py communicates with the RPCSS service to enumerate available RPC endpoints. It then probes and dumps information about the RPC backends such as UUIDs, annotated UUIDs, protocol sequences, and supported transfers. This reveals system info and can identify potential lateral movement attacks.

samrdump

Techniques used:

- SAMR access - Queries SAMR for user info
- Hash dumping - Extracts NT and LM hashes from SAMR
- SMB client - Connects to target over SMB
- Credential access - Retrieves hashes for cracking

URL: **https://github.com/fortra/impacket/blob/master/examples/samrdump.py**

Description:

This Python script extracts user account details and password hashes from the Security Account Manager Remote (SAMR) service. It connects to the target over SMB and then queries the SAMR functions to enumerate users and extract their NT and LM hashes. By default it will extract hashes for all accessible users, or can dump just a single specified user's hashes. The hashes can then be cracked to attain account passwords for lateral movement and privilege escalation.

smbclient

Techniques used:

- SMB client - connects to SMB shares as a client
- File access - access, copy, delete remote files over SMB
- Lateral movement - pivot through SMB shares
- Discovery - enumerate SMB shares/files

URL: **https://github.com/fortra/impacket/blob/master/examples/smbclient.py**

Description: smbclient.py provides SMB client functionality to interact with remote SMB shares and files. It connects to target Windows SMB servers using valid credentials or anonymously. It can then list shares, enumerate directories, upload/download files from

the remote server. It provides an interface similar to an smbclient shell allowing file manipulation and discovery. Since itpivot through file shares using stolen credentials or via anonymous access.

smbexec

Techniques used:

- Command execution - executes commands over SMB
- SMB client - connects to target's ADMIN$ share
- Lateral movement - access other systems on the network

URL: **https://github.com/fortra/impacket/blob/master/examples/smbexec.py**

Description: This Python script executes commands on remote Windows hosts utilizing SMB connections. It connects to the remote admin share (ADMIN$) over SMB using valid credentials. It then creates a service on the remote host via the Service Control Manager. That service is started by the script pointing to cmd.exe with any specified commands. Output from cmd is returned over SMB and printed/saved. This allows execution of arbitrary OS commands on remote hosts the valid credentials have access to.

smbpasswd

Techniques used:

- SMB client - Connects to target over SMB
- SAMR access - Queries SAMR to change passwords
- Credential access - Changes account passwords

URL: **https://github.com/fortra/impacket/blob/master/examples/smbpasswd.py**

Description: This Python script allows changing local account passwords on remote machines over SMB by accessing the SAMR database. It takes target details and authenticates over SMB to connect to the remote SAMR service. User credentials and a new password are provided to reset and update the passwords in the SAM database for those accounts. This results in the account passwords being updated without needing local access on the Windows machine.

smbrelayx

Techniques used:

- SMB relay - Intercepts SMB auth and relays to hit target
- NTLM relay - Captures NTLM messages and replays them
- Credential theft - Steals NTLM hashes passing by
- Privilege escalation - Abuses relayed creds for escalation

URL: **https://github.com/fortra/impacket/blob/master/examples/smbrelayx.py**

Description: smbrelayx.py performs SMB and NTLM authentication relay attacks by intercepting authentication attempts and relaying them to access other resources. It listens for incoming SMB connection attempts that contain NTLM messages. It captures challenge/response NTLM info and uses it to connect and authenticate to other servers that the user has access to. This could give elevated access by abusing relayed credentials. It can also capture NTLM hashes over these connections through manipulation of the SMB/NTLM traffic that it controls.

smbserver

Techniques used:

- SMB server - Implements an SMB server
- File sharing - Shares folders over SMB
- Credential capture - Logs authentication attempts
- Proxy - Man in the middle SMB connections

URL: **https://github.com/fortra/impacket/blob/master/examples/smbserver.py**

Description:

smbserver.py sets up a fake SMB server that can capture credentials and interact with Windows SMB client connections. It implements the SMB protocol to simulate an SMB share that clients can connect to. It allows sharing a local folder over SMB when clients connect. The server logs all authentication attempts and file activity. This allows man-in-the-middle attacks to intercept legit SMB connections by running as a proxy, capturing SMB credentials in the process.

##