Detecting Credential Stealing Attacks Through Active In-Network Defense

By Chintan Shah on Sep 22, 2021

Executive Summary

Today, enterprises tend to use multiple layers of security defenses, ranging from perimeter defense on network entry points to host based security solutions deployed at the end user's machines to counter the ever-increasing threats. This includes inline traffic filtering and management security solutions deployed at access and distribution layers in the network, as well as out of band solutions like NAC, SIEM or User Behavior Analysis to provide identity-based network access and gain more visibility into the user's access to critical network resources. However, layered security defenses face the major and recurring challenge of detecting newer exploitation techniques as they heavily rely on known behaviors. Additionally, yet another significant challenge facing the enterprise network is detecting post-exploitation activities, after perimeter security is compromised.

Post initial compromise, to be able to execute meaningful attacks, attackers would need to steal credentials to move laterally inside the network, access critical network assets and eventually exfiltrate data. They will use several sophisticated techniques to perform internal reconnaissance and remote code execution on critical resources, which range from using legitimate operating system tools to discover network assets to using novel code execution techniques on the target. Consequently, differentiating between the legitimate and malicious use of Windows' internal tools and services becomes a high priority for enterprise networks.

To tackle this long-standing problem of detecting lateral movement, enterprise networks must formulate active in-network defense strategies to effectively prevent attackers from accessing critical network resources. Network Deception is one such defensive approach which could potentially prove to be an effective solution to detect credential theft attackers. Detecting credential stealing attacks with deception essentially requires building the necessary infrastructure by placing the decoy systems within the same network as production assets and configuring them with decoy contents to lure the attackers towards the decoy machines and services. Accurately configuring and tuning the deceptive network can deflect the attacker's lateral movement path towards the deceptive services, consequently allowing the attackers to engage with the deceptive network, helping enterprises protect production assets.

MITRE Shield, a knowledge base maintained by MITRE for active defense techniques highlights many of the methods in adversary engagement. Some of the techniques described by MITRE Shield Matrix with respect to network deception are as below:

MITRE Shield	Description	ATT&CK Technique
Decoy Account – DTE0010	A decoy account is created for defensive or deceptive purposes. The decoy account can be used to make a system, service, or software look more realistic or to entice an action	Account Discovery, Reconnaissance
Decoy Credentials – DTE0012	Seed a target system with credentials (such as username/password, browser tokens, and other forms of authentication data)	Credential Access, Privilege Escalation
Decoy Diversity – DTE0013	deployment of decoy systems with varying Operating Systems and software configurations	Reconnaissance
Decoy Network – DTE0014	Multiple computing resources that can be used for defensive or deceptive purposes	Initial Access
Decoy Personna – DTE0015	Used to establish background information about a user. In order to have the adversary believe they are operating against real targets	Initial Access, Discovery, Reconnaissance
Decoy System – DTE0017	Computing resources presented to the adversary in support of active defense	Reconnaissance

Over the course of this paper, we will discuss some of the widely adapted credential theft attacks executed by adversaries after the initial compromise and then move on to discuss defense techniques against the above MITRE Shield attacks and how to use them effectively to detect deceptive credential usage in the network.

Network Deception – An Active in-network defensive approach

- Most of the targeted attacks involve stealing credentials from the system at a certain point in time as attackers would use them to pivot to other systems in the network. Some of the credential stealing techniques like Golden Ticket attacks have been found to be used in multiple ransomwares armed with lateral movement capabilities.
- Active in-network defense strategies described by the MITRE Shield matrix are significant and play a critical role in detecting credential abuse in the network.
- Network Deception uses these active defense techniques to build the deceptive network infrastructure which could potentially lead to redirecting an attacker's lateral movement path and engaging them to the decoy services without touching the critical production systems.
- It involves placing decoy systems, decoy credentials and decoy contents all throughout the production network essentially converting it into a trap, playing a crucial role in mitigating the attacks.

McAfee Protection

- McAfee MVISION Endpoint Security has the capabilities to protect against credential theft attacks like credential extraction from LSASS process memory via ATP rule 511. More details on configuring policies and a demo are available here.
- McAfee MVISION Endpoint Detection and Response (EDR) has the capabilities to detect credential access from tools like

Mimikatz.

- With McAfee MVISION EDR and ENS integration with Attivo's network and endpoint deception sensor, McAfee can manage its agents and receive alerts for detections in ePO and EDR.

Lateral Movement – Introduction

Lateral movement refers to the tools and techniques used by attackers to progressively expand their foothold within an enterprise network after gaining initial access. As shown in the figure below, lateral movement activity comprises of several stages starting from credential theft, target enumeration and discovery, privilege escalation, gaining access to network resources and eventually remote code execution on the target before exfiltrating data to accomplish a successful attack. Once inside the network, attackers will deploy a range of techniques at each stage of lateral movement to achieve their end goal. One of the primary challenges an attacker will face while moving laterally inside a network is to hide their activities in plain sight by generating a minimum volume of legitimate looking logs to be able to remain undetected. To achieve this, an attacker might choose to embed the tool within a malicious executable or use the operating system's internal legitimate tools and services to perform its lateral movement operations, consequently making this network traffic harder to distinguish.

As per the Verizon DBIR report 2020, over 80% of data breaches involve credential theft attacks. Credential theft is one of the primary tasks attackers need to perform post-exploitation and after gaining initial control of the target machine. It will usually be the first step towards lateral movement strategies which will allow attackers to elevate their privileges and acquire access to other network resources. As indicated earlier, attackers have long been abusing Windows legitimate features like SMB, RPC over SMB, Windows Management Instrumentation, Windows Remote Management, and many other features to perform lateral movement activities. **Figure 1** below highlights where lateral movement falls within the attack chain and its different stages. To remain stealthier, these activities would span a period ranging from many weeks to months.

Figure 1 – Stages of Lateral movement

To be able to distinguish between the admissible and malicious use of these inbuilt services, it is extremely critical for organizations to deploy advanced Threat Detection solutions. Over the course of this blog, we will discuss various credential theft techniques used by adversaries during lateral movement. We will also discuss an approach that can be used to effectively detect these techniques inside the network.

Credential Theft Attacks

Attackers use a variety of tools and techniques to execute credential theft attacks. Many of these tools are open source and readily available on the internet. Operating systems like Windows implement Single Sign On (SSO) functionality, which require the user's credentials to be stored in memory, thereby allowing the OS to seamlessly access network resource without repeatedly asking the user to re-enter those credentials. Additionally, user credentials are stored in memory in a variety of formats like NTLM hashes, reversibly encrypted plaintext, Kerberos tickets, PINs, etc., which can be used to authenticate to services depending upon the supported authentication mechanism. These credentials can be acquired by attackers from memory by parsing appropriate credential storage structures or using the Windows credential enumeration APIs. Consequently, these attacks pose major security concerns, especially in the domain environment if the attacker gains access to privileged credentials which can then be reused to access critical network resources. In the following sections, we discuss some of the widely adapted credential stealing techniques used by malware, with respect to the Windows operating system. Similar credential stealing techniques can also be used with other operating systems as well.

Stealing Credentials from LSASS Process Memory

The Local Security Authority Subsystem Service (LSASS) process manages and stores the credentials of all the users with active Windows sessions. These credentials stored in the LSASS process memory will allow users to access other network resource such as files shares, email servers and other remote services without asking them for the credentials again. LSASS process memory stores the credentials in many formats including reversibly encrypted plaintext, NTLM hashes, Kerberos Tickets (Ticket

Granting Tickets, etc.). These credentials are generated and stored in the memory of the LSASS process when a user initiates the interactive logon to the machine such as console logon or RDP, runs a scheduled task or uses remote administration tools. The encryption and decryption of credentials is done using **LsaProtectMemory** and **LsaUnProtectMemory** respectively and hence a decryption tool using these APIs will be able to decrypt LSASS memory buffers and extract them. However, malware would need to execute with local administrator privileges and enable "**SeDebugPrivilege**" on the current process to be able access the LSASS process memory.

Below is a code snapshot from one of the famous credential harvesting tools, Mimikatz, enabling the required privileges on the calling thread before dumping the credentials.

Figure 2 – Checking for required privileges

We can see that the NTLM hash of the user's credentials is revealed, and this can be brute forced offline as shown below. Many Windows services, such as SMB, support NTLM authentication and NTLM hashes can be used directly for authentication eliminating the need for the clear text passwords.

Figure 3 – Cracking NTLM Hashes offline

Attackers avoid using freely available tools like Mimikatz directly on the target machine to harvest credentials since they are easily detected by AVs. Instead, they use recompiled clones of it with minimal functionality to avoid noise. Below is one such instance where malware embeds recompiled Mimikatz code with the minimal required functionality.

Figure 4 – Credential extraction tool embedded inside malicious executable

Detection can also be avoided by using several "living off the land' mechanisms, available in many post-exploitation frameworks, to execute the credential harvesting tools directly from memory using Reflective PE injection, where the binary is never written to the disk. Yet another approach is to dump the LSASS process memory using process dumping tools, exfiltrate the dump and extract the credentials offline. Microsoft has documented multiple ways to configure additional LSASS process protection which can prevent credentials being compromised.

Stealing Credentials from Security Accounts Manager (SAM) Database

The SAM database is a file on a local hard drive that stores the credentials for all local accounts on the Windows computer. NT hashes for all the accounts on the local machine, including the local administrator credential hash, are stored in the SAM database. The SAM database file is in %SystemRoot%system32/config and the hashes of the credentials are within the registry HKLM\SAM. Attackers need to acquire elevated privileges to be able to access the credentials from the SAM database. The example below demonstrates how the credentials from the SAM database can be revealed through a simple Meterpreter session.

Figure 5 – Dumping SAM database

Stealing Credentials from Windows Credential Manager (CredMan)

Windows Credential Manager stores the Web and SMB/RDP credentials of users if they choose to save them on the Windows machine, thereby preventing the authentication mechanism from asking for those passwords again on subsequent logins. These credentials are encrypted with Windows Data Protection APIs (DPAPI) **CryptProtectData,** either using the current user's logon session or a generated master key, and then saved on the local hard drive. Consequently, any process running in the context of

the logged in user will be able to decrypt the credentials using **CryptUnProtectData** DPAPI. In the domain environment, these credentials can be used by attackers to pivot to other systems in the network. Data Protection APIs provide the cryptographic functionalities that can be used to securely store credentials and keys. These APIs are used by several other Windows components like browsers (IE/Chrome), certificates and many other applications as well. Below is one example of how credential dumping tools like Mimikatz can be used to dump stored Chrome credentials.

Figure 6 – Dumping browser credentials

DPAPI can be abused in multiple ways. In the Active Directory domain joined environment, if other users have logged into the compromised machine, provided a malware is running with escalated privileges, it can extract other user's master keys from the LSASS memory which can then be used to decrypt their secrets. Below is a screenshot of how the master key can be extracted by using the credential dumping tool.

Figure 7 – Extracting DPAPI Master Key

Malware also tends to use multiple variants of credential enumeration APIs available within Windows. These APIs can extract credentials from Windows Credential Manager. Below is one instance of the malware using CredEnumerateW API to retrieve credentials and then search for terminal services passwords which It would use to pivot to other systems.

Figure 8 – Extracting credentials using Windows API

Stealing Service Account Credentials Through Kerberoasting

In the domain joined environment, the Kerberos protocol has a significant role to play with respect to authentication and requesting access to services and applications. It provides Single-Sign-On functionality for accessing multiple shared resources within the enterprise network. The Kerberos authentication mechanism in Active Directory involves multiple requests and responses like Ticket Granting Ticket (TGT) and Ticket Granting Service (TGS) supported by a Key Distribution Server (KDC), usually a Domain Controller. Upon successful authentication, a user will be able to access the respective services.

Attackers gaining access to a system joined in the domain would usually look for high value assets like Active Directory Controller, Database server, SharePoint server, Web Server, etc., and these services are registered in the domain with the specific Service Principal Name (SPN) values, which is a unique identifier of the Service Account in the domain. These SPN values are used by Kerberos to map the instance with the logon account allowing the client to authenticate to the respective service. Well known SPN values are listed out here. Once the attacker is authenticated with any domain user credentials and has information about the SPN values of the services within the domain, they can initiate the Kerberos Ticket Granting Service request (TGS – REQ) to the Key Distribution Server with the specified SPN value. Details on how the SPN values are registered and used in Kerberos authentication is documented here. TGS response from the KDC will have the Kerberos Ticket encrypted with the hash of the service account. This ticket can be extracted from the memory and can be brute forced offline to acquire service account credentials, allowing a domain user to gain admin level access to the service.

Kerberoasting is a well-documented attack technique listed in MITRE ATT&CK and it essentially abuses the Kerberos authentication allowing adversaries to request the TGS Tickets for the valid service accounts and brute force the ticket offline to extract the plain text credentials of the service accounts, consequently enabling them to elevate their privileges from domain user to domain admin. As an initial step to this lateral movement technique, the attacker would perform an internal reconnaissance to gain information about the services registered in the domain and get SPN values. A simple PowerShell command after importing the Active Directory PowerShell module, as shown below, can initiate the LDAP query to get information about all the user accounts from the Domain Controller with the SPN value set.

Figure 9 – PowerShell command to generate LDAP query

Attackers can specifically choose to scan the domain for MSSQL service with the registered SPN value used for Kerberos authentication. PowerShell scripts like GetUserSPNs can scan all the user SPNs in the domain or MSSQL service registered in the domain with Discover-PSMSSQLServers or Invoke-Kerberoast scripts. Following is an example output from the script:

Figure 10 – Kerberoasting PowerShell script output

Once an attacker has the SPN value of the SQL service, a Kerberos Ticket Granting Service Ticket request (TGS-REQ) can be initiated to the domain controller with the SPN value. This can be done by a couple of PowerShell commands generating KRB-TGS-REQ as shown below:

Figure 11 – Kerberos TGS request

Consequently, the Domain Controller sends the TGS-RESP with the ticket of the service account which will be cached in the memory and can be extracted by dumping tools like Mimikatz as a .kirbi document. This can be brute forced offline by tgsrespcrack, allowing the attacker to gain unrestricted access to the service with elevated privileges.

Stealing Credentials from Active Directory Domain Service (ntdis.dit) File

As indicted earlier, once an attacker has penetrated the domain network, it will be natural to progress towards targeting critical assets, such as the Active Directory controller. The Active Directory Database Services AD DS Ntds.dit file is one of the most overlooked attack vectors in the domain environment but can have significant impact if the attacker is able to gain the domain administrative rights leading to complete domain compromise.

The Ntds.dit file is the authoritative store of credentials for all the users in the domain joined environment, storing all the information about the users, groups and memberships, including credentials (NT Hashes) of all the users in the domain with historical passwords and user's DPAPI backup master keys. An Attacker with domain admin rights can gain access to the Domain Controller's file system and acquire credentials like hashes, Kerberos tickets and other reversibly encrypted passwords of all the users joined in the domain by dumping and exfiltrating the Ntds.dit file. These credentials can then be used by the attacker to further access resources by using attack techniques like PTH within the network since the credentials used across other shared resource could be same.

Multiple techniques can be used to dump the Ntds.dit file from the Domain Controller locally as well as remotely and extract the NTLM hashes/DPAPI backup keys for all the domain joined users. One of the techniques is to use the Volume Shadow Copy Service using the **vssadmin** command line utility and then extract the Ntds.dit file from the volume shadow copy as shown below.

Figure 12 – Dumping Volume shadow copy for C drive

Sensitive data on Active Directory is encrypted with the Boot Key (Syskey) stored in the SYSTEM registry hive and dumping the SYSTEM registry hive is a prerequisite as well to be able to extract all the credentials.

Publicly available Active Directory auditing frameworks like DSInternals provide PowerShell cmdlets to extract the Syskey from the SYSTEM registry hive and extract all the credentials from the Ntds.dit file.

Ntds.dit can also give access to the powerful service account within the Active Directory Domain, KRBTGT (Key Distribution Centre Service account). Acquiring the NTLM hash of this account can enable the attacker to execute a Golden Ticket attack leading to complete domain compromise with unrestricted access to any service on the domain joined system.

Stealing Credentials Through a DCSync Attack – From Domain user to Domain Admin

A DCSync attack is a method of credential acquisition which allows an attacker to impersonate the Domain Controller and can consequently replicate all the Active Directory objects to the impersonating client remotely, without requiring the user to logon to the DC or dumping the Ntds.dit file. By impersonating the Domain Controller, the attacker could acquire the NTLM hash of the KRBTGT service account, enabling them to gain access to all the shared resources and applications in the domain joined environment. To be able to execute this credential stealing technique, an attacker would have to compromise the user account with the required permissions, specifically DS-Replication-Get-Changes and DS-Replication-Get-Changes-All, as shown below.

Figure 13 – User with privileges

Once the attacker compromises the user account with the required privileges, Pass-The-Hash attacks can be executed to spawn a command shell with the forged logon session. Credential dumping tools like Mimikatz do this by enumerating all the user logon sessions and replacing the user credentials with the stolen usernames and NTLM hashes provided, in the current logon session. Behind the scenes, this is executed by duplicating the current process's access token, replacing the user credentials pointed by duplicated access token and subsequently using the modified access token to start a new process with the stolen credentials which will be used for network authentication. This is as shown below for example user **"DCPrivUser".**

Figure 14 – Pass-the-Hash attack

Further, as indicated below, any subsequent NTLM authentication from the logon session will use the stolen credentials to authenticate to domain joined systems like the Active Directory Controller.

Attackers can now initiate the AD user objects Replication request to the Domain Controller using Directory Replication Services Remote Protocol (DRSUAPI). DRSUAPI is the RPC protocol used for replication of AD objects. With DCERPC bind request to DRSUAPI, an RPC call to **DSGetNCChanges** will replicate all the user AD objects to the impersonating client. Attackers would usually target the KRBTGT account since acquiring the NTLM hash of this account will enable them to execute a Golden Ticket attack resulting in unrestricted access to domain services and applications.

Figure 15 – DCSync Attack

As indicated earlier, with the NTLM hash of the KRBTGT account, adversaries can initiate a Golden Ticket attack (Pass-the-Ticket) by injecting the forged Kerberos tickets into the current session which can be used to authenticate to any service with the client that supports pass the ticket (for instance, sqlcmd.exe connection to DB server, PsExec, etc.)

Figure 16 – Golden ticket with forged Kerberos ticket

Detecting Credential Stealing Attacks with Network Deception

The credential theft techniques we discussed in the previous sections are just the tip of the iceberg. Adversaries can use many other sophisticated credential stealing techniques to take advantage of system misconfigurations and legitimate administrative tools and protocols and, at the same time, remain undetected for a longer period. With many other event management

solutions with SIEMs, used in conjunction with other network security solutions, it becomes a challenge for administrators to distinguish malicious use of legitimate tools and services from lateral movement. Perimeter solutions have their limitations in terms of visibility once the attacker crosses the network boundary and is inside the domain environment. It is extremely critical for organizations to protect and monitor critical network assets like the Domain Controller, Database server, Exchange Servers, build systems and other applications or services, as compromising these systems will result in significant damages. Therefore, enterprise networks must deploy a solution to detect credential stealing attacks as they can be used to pivot to other systems on the network and move laterally once an attacker establishes an attack path to a high value target. If the deployment of a solution within the critical zones of the network can detect the use of stolen credentials before adversaries can reach their target, the critical assets could still be prevented from being compromised.

Network Deception is one such deployment within the domain environment where, using the MITRE Shield techniques like decoy systems and network, decoy credentials, decoy accounts, decoy contents, could potentially help detect lateral movement early in the adversary's attack path to the target asset and at the same time, report significantly low false detection rates. The idea of deception originates from the decades old honeypot systems but, unlike those, relies more on forging trust and giving adversaries what they are looking for. With its inbuilt proactiveness it is configured to lure attackers towards deceptive systems. As shown in the figure below, Network Deception consists of authentic looking decoy systems placed within the domain network, specifically in the network where the critical assets are placed. These decoy systems (could be virtual machines) are the full-fledged OS with configured applications or services and could be replicating the crucial services like Domain Controller, Exchange or DB server and other decoy machines that could lead to those systems. The image below highlights the key foundational aspects of the Network Deception

Figure 17 – Network Deception

Key Aspects of Network Deception

As visualized in the figure above, Network Deception comprises the following key basic facts with respect to the deployment in the domain joined environment:

- As a part of deployment, decoy/deceptive machines are planted within the network alongside production systems and critical assets. These decoy systems could be real systems or virtual systems with production grade operating systems with the required setup to make them blend well with real systems.
- As one of the key aspects, deceptive machines are configured to lure attackers towards the decoy services instead of the production services, thereby deflecting or misleading the attacker's lateral movement path to the target asset.
- Many of the decoy machines could replicate critical services like Domain Controller, DB servers, Exchange/SharePoint servers and other critical services or applications within the data center.
- Any legitimate domain user should not be generating traffic to or communicating with the configured decoy machines unless there are some misconfigurations in the network, which need to be corrected.

Basic Decoy Network Setup

Since credential theft plays an important role in a successful targeted attack, deception essentially focuses on planting fake credentials on the production and decoy endpoints at multiple places within the OS and monitoring the use of these credentials to pivot to other systems. With respect to the network setup, the following are the key aspects, however this list is not exhaustive, and much more could be added:

- **Replicating critical network assets and services with decoy machines**: Replicating critical network services like Active Directory, DB services, etc., will make more sense since these are the most targeted systems in the network. The decoy Active Directory should be configured with deceptive AD objects (users, groups, SPNs, etc.). with deceptive contents for other replicated services.
- **Planting authentic looking decoy machines in the production network:** As indicated earlier, these decoy machines could be real or virtual machines with the production grade OS placed alongside production systems in the critical infrastructure

to blend in well. These decoy machines should be joined to the decoy AD and configured with deceptive user accounts to monitor successful logon attempts to the systems.

- **Injecting deceptive credentials on production endpoints**: Production endpoints should be injected with deceptive credentials at multiple places like LSASS process memory, Credential Manager, browser credentials, etc., to increase the possibility of these credentials being picked up and used to pivot to decoy systems in the network. These endpoints could be public facing machines or their replicas as well.
- **Decoy Machine runs client applications pointing to decoy services:** Decoy machines may run the client with deceptive credentials and configured to point to the decoy services. These could be DB/FTP/Email clients and any other replicated decoy services.
- **Mark decoy systems as "NO LANDING ZONE":** One of the key deployment aspects of deception is to mark all the decoy systems and services as "NO LANDING ZONE", essentially meaning no legitimate domain users should be accessing decoys and any attempts to access these systems should be closely monitored.

Some of the other setup required for effective deployment of deception is as summarized below:

Figure 18 – Deceptive network setup – Basic requirements

Basic Decoy Systems Setup

To detect the use of deceptive credentials, setting up decoy machines is an essential part of the solution as well. Primarily, decoy machines should enable the access attackers are looking to have during the lateral movement phase. Decoys should also be configured to enable relevant auditing services to be able to generate events. For instance, the following enables the account logon events to be audited:

Decoy machines must be setup to run the log collector agent that can collect the access logs generated and forward them to the correlation server. However, in the domain joined environment, it is also essential to tune the decoy machines to forward only the relevant logs to the correlation server to minimize false positives.

The below highlights some of the auditing required to be enabled on the decoy systems for effective correlation.

Figure 19 – Basic decoy setup

Illustrating and Achieving Network Deception

The following sections describe some examples of how deception can be achieved in the domain network, along with a visualization of how credential theft can be detected.

Network Deception – Example 1: Injecting NETONLY credentials into LSASS process memory

LSASS process memory is one of the prime targets for attackers, as well as malware armed with lateral movement capabilities since it caches a variety of credentials. Credential extraction from the LSASS process requires opening a read handle to the process itself which is closely monitored by EDR products but there are stealthier ways around it.

One of the primary tasks towards achieving credential-based deception is to stage the deceptive credentials in LSASS process memory. This can be accomplished on the production and decoy systems by executing a trivial credential injection code which uses the **CreateProcessWithLogonW** Windows API with the specified crafted credentials. **CreateProcessWithLogonW** creates the new logon session using the caller process access token and spawns the process specified as a parameter in the security

context of the specified deceptive credentials and it will be staged in the LSASS memory until the process runs in the background. The below shows the example code calling the API with the specified credentials which is also visible when credentials are extracted with Mimikatz.

Figure 20 – Injecting credentials into LSASS memory

One of the parameters to **CreateProcessWithLogonW** is "**dwLogonFlags**" which should be specified as **LOGON_NETCREDENTIALS_ONLY** as shown in the code above. This ensures the specified credentials are used only on the network and not for local logons. Additionally, NETONLY credentials used to create a logon session are not validated by the system. Below is a code snapshot from credential extraction tool Mimikatz, using a similar approach to forge a logon session and replacing the credentials with the supplied ones while executing Pass-the-Hash attacks.

Figure 21 – Mimikatz code for PTH attack

Network Deception – Example 2: Configure deceptive hostnames for decoy VMs

Attackers or malware moving laterally inside the network might do a recon for interesting hostnames via nbtstat/nbtscan. To deflect the lateral movement path, decoy systems can be configured with real looking hostnames that match the production systems. These hostnames will then be visible on NetBIOS scans as shown below.

Figure 22 – Deceptive host names pointing to decoy machines

These decoy systems can also run the relevant client applications pointing to the decoy services, with authentication directed to the decoy Domain Controller in the network. Detection of this attack path happens much earlier, however the decoy network setup keeps the adversaries engaged, helping admins to study their Tools and Techniques.

Figure 23 – Decoy machines running clients pointing to decoy services

A similar deception setup can also be done for the browsers where saved credentials can point to the decoy applications and services within the domain. For instance, Chrome saves the credentials in the SQLite format on the disk which can be decrypted using DPAPI as discussed earlier sections. The below examples demonstrate deceptive browser credentials which can lure adversaries towards the decoy services.

Figure 24 – Inserting deceptive browser credentials

In addition to some of the techniques discussed above, and many others highlighted in the previous sections, setting up deception involves much more advanced configuration of decoy systems to minimize false positives and needs to be tuned to the environment to accurately identify malicious activities. Deception can also be configured to address multiple other phases of lateral movement activity including reconnaissance and target discovery, essentially redirecting the adversaries and giving them a path to the target. Below is a high-level visualization of how the decoy network can look like the domain environment.

Figure 25 – Deception network setup

On the occasion where one of the domain-joined or public facing systems is compromised, authentication would be attempted to other domain joined systems in the network. If an authentication is attempted and any of the decoy systems are accessed

and logged on, the use of these planted deceptive credentials should be a red flag and something which must be investigated. The visualization below shows the flow and an event being sent to an administrator on accessing one of the decoy systems.

Figure 26 – Deceptive credentials usage for authentication in the domain

One such example event of successfully logging on to the decoy system is as shown below:

Figure 27 – Alert send to administrator on using deceptive credentials

MITRE ATT&CK Techniques:

Credential theft attacks discussed here are mapped by MITRE as below:

Technique ID	Technique Name	Description
T1003.001	LSASS Process Memory	Attackers may attempt to access LSASS process memory to extract credentials as it stores a variety of credentials. Administrative privileges are required to access the process memory.
T1003.002	SAM Database	Accessing credentials from SAM database requires SYSTEM level privileges. Stores credentials for all the local user accounts on the machine.
T1003.003	NTDS.dit file	Contains credentials for all the domain users. File is present on the DC and domain admin privileges are required to access this file.
T1003.006	DCSync	Attacker can extract the credentials from the DC by impersonating the domain controller and use DRSUAPI protocol to replicate credentials from DC.
T1558.001	Golden Ticket	Attackers acquiring credentials for KRBTGT account can forge the Kerberos ticket called Golden Ticket, allowing them to get unrestricted access to any system in the domain
T1558.002	Silver Ticket	Allows attacker to get admin level access to the service accounts by abusing Kerberos authentication
T1558.003	Kerberoasting	Allows attackers to extract the Kerberos tickets for service accounts from memory and brute force offline to get credentials

Conclusion

As credential theft attacks play a significant role in an attacker's lateral movement, so as in-network defense for the defenders. With attackers' lateral movement tactics evolving and getting more stealthier, defenders will have to adapt to innovative ways of defending the critical network assets. In–network defense strategies like Deception could prove to be a promising and forward-looking approach towards detecting and mitigating data theft attacks. Strategic planting of decoy systems within the production network, inserting decoy credentials and decoy contents on calculative selection of endpoints and decoy systems and accurately setting up the logging and correlation via SIEMs for monitoring the use of decoy contents, could certainly detect and mitigate the attacks early in the lateral movement life cycle.

Endpoint solutions like User Entity Behavior Analytics (UEBA) and Endpoint Detection and Response (EDR) could also play a significant role in building the deception infrastructure. For instance, one of the ways UEBA solutions could prove useful is to baseline user behavior and monitor access to credential stores on the system. UEBA/EDR could raise the red flag on injection of forged Kerberos tickets in the memory. This can provide user level visibility to a greater extent when integrated with SIEM, playing a crucial role in mitigating credential theft attacks.

About the Author

Chintan Shah

Chintan Shah is currently working as a Security Researcher with McAfee Intrusion Prevention System team and holds broad experience in the network security industry. He primarily focuses on Exploit and vulnerability research, building Threat Intelligence frameworks, Reverse engineering techniques and malware analysis. Chintan had researched and uncovered multiple targeted and espionage attacks in the past ...

Read more posts from Chintan Shah ›

‹ Previous Article

Categories: McAfee Enterprise ATR