Onelogon: Taking over Active Directory Accounts via Netlogon

Alexander Neff
Ruhr University Bochum
onelogon@aneff.io

Tobias Holl
Ruhr University Bochum
tobias.holl@rub.de

Kevin Borgolte
Ruhr University Bochum
kevin.borgolte@rub.de

Abstract

Microsoft's Active Directory (AD) is a critical component of the IT infrastructure of numerous enterprises. Thus, security vulnerabilities in AD can have dire consequences for the security posture of an organization's IT infrastructure. At the core of the AD architecture is the Netlogon Remote Protocol, which is used to manage computer accounts, delegate authentication requests, and various other management tasks.

In 2020, Tervoort identified Zerologon, a critical vulnerability in Netlogon, which allowed attackers to fully compromise an AD management domain [37]. In turn, Microsoft released two patches: one aiming to rectify the cryptographic vulnerability that enabled the attack, and the second one to ensure that all Netlogon communication is signed and sealed.

In this paper, we analyze these patches and show that they are insufficient to mitigate the underlying vulnerabilities. We show that the cryptographic patch can be bypassed by an unprivileged attacker within the AD domain. We introduce the *Onelogon* attack with two distinct variants for varying attacker capabilities, both of which allow an attacker to take over a vulnerable AD account in approximately 30 minutes. If this AD account belongs to a Domain Controller, an attacker can leverage *Onelogon* to fully compromise the AD domain.

With the goal of mitigating the attacks, we identified their underlying root cause: the incorrect use of AES-CFB8 encryption. Both the earlier Zerologon attack and our new attack exploit how Netlogon incorrectly uses AES-CFB8. Finally, we provide and compare various mitigation and detection approaches for Microsoft and AD operators to prevent account takeover attacks and authentication bypasses in the short term and fundamentally. Unfortunately, addressing the underlying root cause requires a backward-incompatible change to Netlogon: reimplementing AES-CFB8 correctly.

We disclosed this issue to Microsoft and CERT-Bund, but do not expect any fixes to be forthcoming. In the meantime, we advise users to apply the mitigation and detection strategies outlined in this paper.

1 Introduction

Microsoft's Active Directory (AD) is the most widely used directory service for managing identities and access in enterprise environments [14]. While it is mostly known for the administration of user and computer accounts, access permissions, and authentication, it can manage many more components, including certificates and software updates. This makes the security of an AD deployment crucial to the security of an organization's IT infrastructure.

Across an AD domain, network services can use different protocols for authentication [24]. Two of these protocols, Kerberos and NT LAN Manager (NTLM), have been extensively studied [2; 4; 7; 29; 31; 32], but research has largely ignored the other supported protocols.

The Netlogon Remote Protocol is one of these protocols. It is a remote procedure call (RPC) protocol that provides services related to pass-through authentication, computer account management, and domain trust services. For example, this includes managing any authentication within the network, the integration of all devices connected to Active Directory, access to network shares, etc.

Previously, Tervoort discovered the Zerologon vulnerability (CVE-2020-1472; CVSS score: 10.0, critical), which exploits a cryptographic flaw in Netlogon's authentication mechanism [37]. It allows an attacker to impersonate arbitrary computers in the AD domain, including the Domain Controller itself, which is the server responsible for managing the entire AD domain. It also allows the attacker to reset the password of the Domain Controller's computer account to an empty string. The result is then the *immediate and complete compromise of the domain*. Microsoft addressed the Zerologon vulnerability with two patches: the first patch rejects any connections attempting to exploit the underlying cryptographic vulnerability, and the second patch enforces the signing and sealing (encryption) on all RPC connections.

In this paper, we show that, while Microsoft addressed the Zerologon vulnerability specifically, it did not, in fact, address the flaw stemming from incorrect cryptographic use.

We make the following technical contributions:

- We show that the first patch (terminating connections that attempt exploitation), is inadequate in certain circumstances and only prevents Zerologon, but not other Zerologon-like attacks.
- We introduce such a new attack, *Onelogon*.
- We show that our *Onelogon* attack is indeed practical and allows an attacker to compromise a vulnerable AD domain in less than 33 minutes, and that vulnerable setups exist in the wild.
- We discuss the required mitigations to fix the underlying issue and we discuss short-term, interim solutions until such a proper patch is released and widely rolled out.

Our artifact and implementation are available as open source at `softsec.link/woot26.onelogon`.

2 Background

As Active Directory and the Netlogon protocol are rather uncommon targets in academia, we briefly introduce the required background for our *Onelogon* attack.

2.1 Netlogon Remote Protocol

The Netlogon Remote Protocol (NRPC) is an RPC protocol developed by Microsoft that is used for various tasks in an Active Directory environment. Today, NRPC is primarily used to provide pass-through authentication for NTLM (a deprecated, but still widely-used challenge-response authentication protocol), but it also serves a number of other use cases [20], including the discovery of various domain components. Before the Directory Replication Service (DRS) Remote Protocol, NRPC was also the primary protocol for domain replication between Domain Controllers [19].

Authentication. After the underlying TCP connection is successfully established, Netlogon first establishes a secure channel to authenticate the later RPC calls and protect some sensitive information (e.g., passwords) included in those calls. To do so, the client and server negotiate a shared session key [20, §§ 3.1.4, 3.5.4.4].

Both parties generate a random 8-byte *challenge*, which they exchange with each other. From the challenges and a password hash[1] of the account that is trying to establish the secure channel, both sides compute the session key. Each party then proves ownership of that session key by deriving its *Netlogon credential* from the challenge that it sent before with the computed session key and sending it to the other party,

[1]This is typically the NT hash, possession of which is generally considered as equivalent to possession of the password in an AD domain.

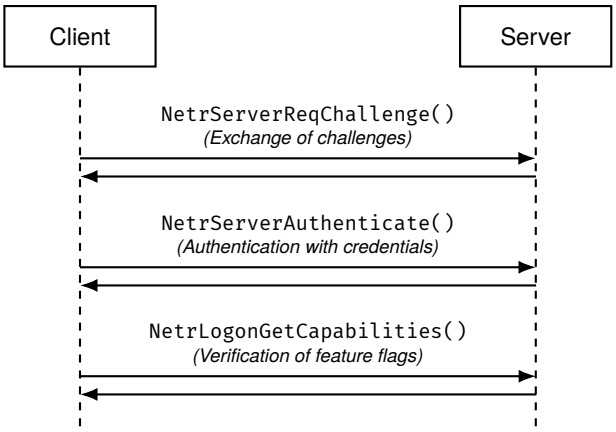

Figure 1: The Netlogon authentication flow is generally a three-step process [20, § 3.1.4.1]. Client and server first exchange random challenges, then authenticate with credentials generated from these challenges, and finally verify that no downgrade occurred during the negotiation.

which verifies it against its own computation of the same value. From this point forward, the RPC calls and responses that are sent over the NRPC connection can be encrypted and signed with the session key. The client credential is also used later to authenticate RPC calls. Figure 1 shows this process in detail.

The exact method by which the session key and credentials are derived depends on a set of feature flags that the client transmits alongside the authentication request. To prevent downgrade attacks, the client can later verify the feature flags the server received over the secure channel and terminate the connection if they do not match [20, § 3.1.4.1]. In this paper, for the threat model of our *Onelogon* attack, we assume that the latest method (AES encryption) is used rather than older RC4-based approaches. This is the default configuration on modern Windows systems.

The server identifies sessions by an arbitrary client-provided string. Usually, this is the client's NetBIOS computer name.

Credential Computation. If client and server negotiated AES support, they compute the client and server credentials from the respective challenge by encrypting it with the session key using AES in 8-bit CFB mode. That is, they encrypt each byte separately by XORing it with the first byte of the AES encryption of the previous 16 bytes of ciphertext (prepended with the 16-byte IV, if not enough ciphertext is available yet). For Netlogon authentication, *this IV is always zero* [20, § 3.1.4.4.1], which enabled the Zerologon vulnerability (see Section 2.2). Figure 2 illustrates this mode of operation.

Remote Procedure Calls. In all subsequent RPC calls that require a secure channel, except `NetrLogonSamLogonEx`, the client must include an *authenticator* as proof of identity [20, § 3.1.4.5].

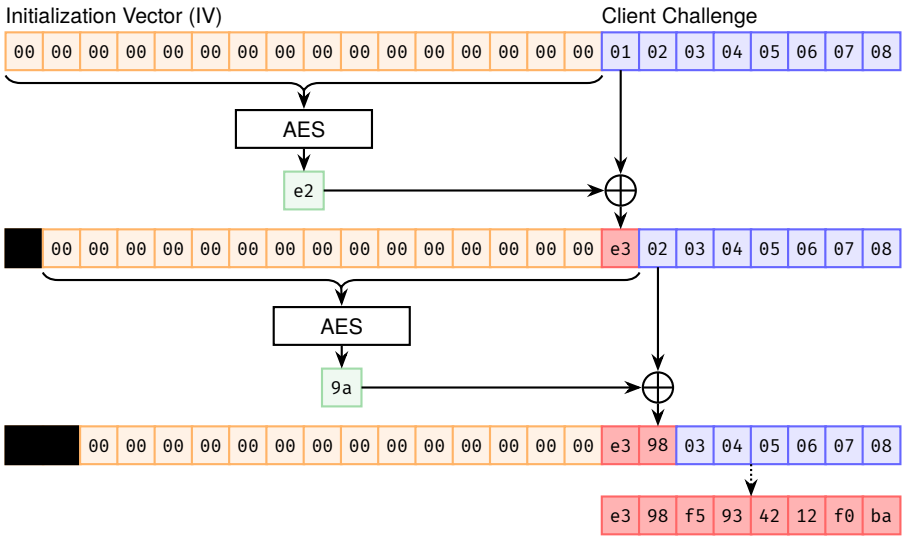

Figure 2: AES in 8-bit CFB mode with zeroed IV as in Netlogon [37, Figure 2].

This authenticator contains a 32-bit Unix timestamp, and a credential that is obtained by re-encrypting the sum of the current client credential and the timestamp with the session key as for the original credential computation. The server verifies the authenticator, increments the client credential by one, and computes its own *return authenticator*, which it includes in the response. The incremented client credential then becomes the new client credential for the next RPC call.

If client and server additionally negotiate the "secure RPC" feature flag, then all RPC messages must at least be signed and may also be encrypted using the session key.

2.2 Zerologon

In 2020, Tervoort disclosed Zerologon (CVE-2020-1472), a critical vulnerability in Netlogon that allows an attacker to reset the password of any domain computer [37]. It is caused by a flaw in the cryptographic design of the Netlogon protocol. It only assumes that an attacker has access to the internal network and can establish a TCP connection to the Domain Controller, but does not require any other privileges.

2.2.1 Attack Chain

An attacker exploits Zerologon to craft a fake client credential, disable secure RPC, and then spoof an RPC call to clear the password of the computer account of the Domain Controller itself, which they can then leverage to escalate their own privileges and compromise the entire AD domain.

Crafting a Fake Client Credential. The attack exploits how Netlogon uses AES-CFB8 to implement authentication. Normally, the server computes the client credential by encrypting the client challenge (chosen by the attacker) with the session key (unknown to the attacker) with AES-CFB8. Because the session key is unknown, the client credential should be unpredictable. However, Netlogon uses an all-zero initialization vector (IV) for this encryption.[2] Thus, the initial input to the AES block cipher always consists of 16 zeroes. There is then a $\frac{1}{256}$ chance that this all-zero input will encrypt to a ciphertext that starts with a zero byte, and thus the first plaintext byte will encrypt to itself (see Figure 2). If that plaintext byte, which is attacker-controlled, is zero, the next input to AES will also be entirely zeroes, and so on. With probability $\frac{1}{256}$, a plaintext of all zeroes will encrypt to a ciphertext that is also all zeroes [37].

The attacker can then simply create new sessions until they encounter an exploitable session key and authenticate successfully with an all-zero client credential. They can achieve this by requesting a new server challenge, which changes the session key. In practice, this process only takes a few seconds [37].

Disabling Signing and Sealing. As long as signing and sealing (encryption) are enabled, the attacker cannot make RPC calls using the fake client credential, because they do not know the session key. This is a setting that is negotiated via request flags during the authentication phase. At the time Zerologon was disclosed, servers did not enforce RPC signing and sealing in the Netlogon protocol. Since in this attack the attacker directly attempts authentication against the server, they can simply not set this flag in the request and proceed with unencrypted and unsigned RPC messages [37].

[2] This contradicts NIST's recommendations for all variants of AES-CFB, which state that the IV *must* be unpredictable for any given message [8].

Obtaining the Authenticator. The RPC calls remain authenticated via credential-derived authenticators (see Section 2.1). However, this derivation uses the same cryptographic method as the initial computation of the client credential. As long as the initial client credential is all zeroes, which the attacker has already achieved, and the authenticator timestamp is also zero, which the server does not validate, the input into AES-CFB8 is again all zeroes, and thus we can obtain an all-zero authenticator.

Changing the Computer Account Password. With both a valid client credential and a valid authenticator, the attacker can now call RPC functions that require a secure channel.

The `NetrServerPasswordSet2` function is of particular interest, as it allows changing the password of any computer account by providing a new password (in the clear) [20, § 3.5.4.4.6]. However, the password-related functions in Netlogon encrypt the object containing the password buffer with the session key (again, when using AES-CFB8, with an all-zero IV). Unfortunately, while the Netlogon specification requests the password to be padded with random data (with the password at the end, followed by its length in bytes), the attacker can zero out the request structure entirely. Thus, the entire structure can again encrypt to all-zeroes, allowing the attacker to clear the password of any computer account without needing to know the session key [37].

Escalating Privileges. Importantly, the attack is not limited to regular computer accounts, but also works against the computer account of the Domain Controller itself. Once the Domain Controller's account has been successfully compromised, the attacker can use its privileges to escalate to domain administrator privileges, for example, by replicating the Active Directory database to extract NT and LM hashes (that essentially serve as passwords) and AES keys used for Kerberos authentication using the DCSync attack [10; 37]. Thus, an attacker can impersonate any user or computer in the domain, including domain administrators.

2.2.2 Microsoft's Three-Step Mitigation

The Zerologon vulnerability (CVE-2020-1472) was assigned a CVSSv3 base score of 10.0 (critical) by NIST [30]. Microsoft instead assigned a base score of 5.5 (medium) [17]. Microsoft mitigated Zerologon in three phases, providing organizations with guidance on how to detect and address it [18].

Initial Deployment. In a security update released on August 11, 2020, Microsoft modified the Netlogon protocol to mitigate the vulnerability and add protections to the RPC layer. We discuss the technical details of these changes in Section 4. The update also added logging capabilities for non-compliant devices and enabled secure RPC by default for all Windows-based clients [18].

Find and Address. Microsoft then introduced additional event IDs to allow administrators to identify devices that do not support or do not yet use secure RPC. They encouraged customers to request updates for third-party devices to enable the use of secure RPC, and they introduced a group policy that would exempt devices or accounts from upcoming changes that would enforce signing and sealing on a protocol level [18].

Enforcement Mode. Starting on February 9, 2021, Microsoft began rolling out "enforcement mode." In this mode, all Domain Controllers enforce secure RPC for all devices except those specifically allowed by the ACL in the aforementioned group policy [18].

3 Related Work

Besides Zerologon (Section 2.2) and work closely related to it, there exists little publicly available analysis of Netlogon.

Other Vulnerabilities. Solino discovered a vulnerability in the authentication mechanism that allows an attacker to mount an NTLM relay attack against a server requiring SMB signing [35]. Tervoort identified a downgrade attack in Netlogon that allows an active on-path attacker to perform arbitrary RPC calls by forcing the client and server to use unencrypted and unsigned communication [36]. Kabibo introduced an approach to enumerate domain and trust information, as well as users and computers, if anonymous access to the Netlogon RPC pipe was allowed [12; 13].

Zerologon-based Relay Attack. Mollema found an alternative method of launching the Zerologon attack by relaying authentication to another service [26]. This attack leverages the pass-through authentication mechanisms of Netlogon, which servers use to validate NTLM logon information when users attempt to authenticate.

Detecting and Mitigating Zerologon. Practitioners have described the Zerologon vulnerability, its mitigations, and indicators of compromise at length [5; 6; 28; 34]. On the academic side, Bezzateev, Fomicheva, and Zhemelev extend the common detection methods based on the Windows Event Log by using Netlogon's debug log [3]. When enabled, events like RPC calls or authentication attempts are logged [15], which facilitates detecting and blocking Zerologon attacks in real-time, but the additional logging overhead can be significant. Similarly, Myllyla and Costin propose a detection method that uses Sysmon [23] to log network connections [27]. He et al. focus their analysis on Microsoft's three-step mitigation for the vulnerability [11].

So far, no work has analyzed Microsoft's patch to the cryptographic issue involved in Zerologon in depth.

4 Onelogon

Next, we examine the technical aspects of Microsoft's mitigations against the Zerologon attack, and how they can be bypassed in theory and practice.

4.1 Secure RPC Enforcement

To mitigate the Zerologon attack, Microsoft began enforcing the use of secure RPC connections, that is, the RPC messages are additionally signed and encrypted with the session key. This protection mechanism is an additional layer on top of the authenticator-based identity proof (Section 2.1) and the encryption of sensitive data, such as plaintext passwords (Section 2.2.1).

While the encryption still does not follow cryptographic best practices (AES-CFB8 is again used with a fixed IV, but it is non-zero at least), messages are signed using HMAC-SHA256, which prevents Zerologon-style attacks that do not obtain the session key directly [20]. It is also possible to use an older cryptographic scheme based on RC4 and HMAC-MD5, which we did not analyze for this paper.

However, because secure RPC connections are not supported by all devices, especially third-party appliances, Microsoft allows disabling secure RPC enforcement via the *Domain Controller: Allow vulnerable Netlogon secure channel connections* group policy.

This setting is widely used in practice; its only purpose is to exclude some or all accounts from secure RPC enforcement. We obtained empirical data from SpecterOps[3] (a security company active in Active Directory and enterprise network security), who collected the group policy settings from the domain controllers of 44 enterprise customers. In late 2025, five years after the Zerologon mitigations were released, 23% of companies still have active deployments in which this policy is active.[4] While a detailed statistical analysis is beyond the scope of this paper, this shows that, for many real-world environments, the only defense against Zerologon is Microsoft's cryptographic patch, which we describe next.

4.2 Vulnerable Challenges

With revision 37.0 of the Netlogon protocol specification, Microsoft applied a cryptographic patch that aims to mitigate Zerologon [16]. It attempts to counter Zerologon's core attack

```
bool NlIsChallengeVulnerable(char *challenge) {
    for (int i = 1; i < 5; ++i)
        if (challenge[i] != challenge[0])
            return false;
    return true;
}
```

Listing 1: Microsoft's check for Netlogon challenges and authenticators used in a Zerologon attack, as implemented in `netlogon.dll`.

primitive: that, for certain session keys, AES-CFB8 as used by Netlogon will encrypt a plaintext that consists of only zeroes to an all-zero ciphertext.

The patch (Listing 1) enforces that the client challenge and all authenticators used for RPC calls contain at least two distinct byte values in the first five bytes. Otherwise, the challenge or authenticator are considered "vulnerable" and the connection is rejected outright.

This patch is insufficient: It does not fully mitigate all Zerologon-style attacks and it does not address the underlying cryptographic problem.

In particular, it is still possible to craft a fake client credential that bypasses the patch if we encounter a session key that produces a key stream byte of zero not only for an all-zeroes input but also for subsequent inputs to the AES block cipher that occur during encryption of the challenge.

A challenge that passes Microsoft's security check will, during encryption with AES-CFB8, produce at least four different inputs to the AES block cipher (all zeroes initially, but then different inputs to encrypt the final three bytes of the challenge). To obtain a challenge that encrypts to itself while bypassing the patch, the resulting key stream byte must be zero for all four of these inputs. The probability of this occurring is only 2^{-32}, which — without additional improvement — requires a number of attempts too large for a practical attacker (e.g., a naïve implementation of this attack with individual NRPC connections would require several years of bruteforcing against the Domain Controller). However, this intuition gives rise to a more sophisticated attack, *Onelogon*, which we show is feasible in practice.

4.3 The Global Challenge List

To make our attack practical, we need to first examine how Netlogon handles incoming connections in detail.

The Netlogon server stores the challenges for pending connections in a linked list, the *global challenge list* `NlGlobalChallengeList`, where they are identified by the client's computer name (which is attacker-controlled). Every authentication attempt against any of these challenges requires the server to look it up in the list, that is, iterating through it, skipping entries with mismatched computer names, and

[3] https//specterops.io/

[4] For reasons of client confidentiality, we only received aggregate statistics from SpecterOps. Collecting data from enterprises that choose to engage a security company likely biases this sample toward environments that are generally more security-conscious. Thus, it is possible that this setting is even more prevalent in practice (i.e., more AD domains are vulnerable).

comparing the credentials in the request with those stored in the list entries. Unsurprisingly, this causes a significant slowdown for connection and authentication attempts when the list is full.

Fortunately for us however, it also means that we do not need to send an authentication attempt for every challenge that we store on the server, because the service will always test our authentication attempts against *all* pending sessions. Since the server challenge will also be randomized and thus different on every connection attempt, we can send the same challenge multiple times in a row and still obtain different session keys for every authentication request.

To prevent resource exhaustion, Windows caps the list at 100,000 entries and any further connection attempts will replace a randomly selected entry in the list. Periodically, the Netlogon server also removes all list entries for challenges older than two minutes. Finally, when a connection is successfully authenticated, all entries for that computer name are removed from the list.

A fast exploit attempt therefore must either a) manually clear the challenge list (which requires successful authentication), or b) remain below 100,000 open connection attempts. Otherwise, we need to wait for the pending sessions to expire before we can attempt additional connections, which is too slow for practical purposes.

4.4 Attacking With a Computer Account

For now, we assume that an attacker has successfully obtained credentials to an arbitrary but otherwise unprivileged computer account to be able to clear their connections from the global challenge list (see Section 4.3). In practice, obtaining credentials for an arbitrary computer account is easier than obtaining credentials for a specific high-privilege account (e.g., a Domain Controller), since compromising any single domain-joined host (e.g., any laptop or desktop computer of a user, or any low-privileged server) yields credentials for its corresponding computer account. However, we will also introduce a variant of our attack that does not have this prerequisite in Section 4.5.

We observe that the final challenge byte is never used as part of the input to the AES block cipher (it is only encrypted with the key stream byte generated from the previous 16 bytes of IV and data), and therefore does not influence the generated keystream used for encryption. We can leverage this behavior to reduce the 32-bit brute-force attack (Section 4.2) to 2^{24} connection attempts. Figure 3 illustrates this attack.

Consider, without loss of generality, the client challenge `00 00 00 00 55 66 77 00`. Assume that the self-encrypting property we relied on previously still holds for the all-zero input and for the first two non-zero inputs to the block cipher. Then, the AES-CFB8 encryption of the challenge (i.e., the client credential) takes the form `00 00 00 00 55 66 77 XX` with an unknown byte in the last position. Because the challenge list is not cleared on unsuccessful authentications, an attacker can seed the challenge list with 100,000 possible session keys and then simply try all 256 possible client credentials to detect if there is a challenge vulnerable to *Onelogon*. If this is not the case, the attacker flushes the challenge list by successfully authenticating with the same computer name, and retries the attack. To obtain the resulting RPC authenticator for a timestamp of zero, the protocol then re-encrypts the last byte (`XX`) with the same keystream byte as before. Since this is just an XOR operation, that is, its own inverse, the authenticator is simply the same as the original client challenge.

Therefore, an attacker does not actually need to brute-force all 32 bits. Instead, it is sufficient to brute-force 24 bits, and try the 256 possible client credentials for authentication afterwards.

4.5 A General Attack

We can eliminate our previous assumption of a computer account and generalize our *Onelogon* attack by shifting complexity away from the number of pending connections to less than 100,000 and toward the number of authentication attempts that we need to make. We can do so by developing a meet-in-the-middle approach to successfully compute a valid credential and authenticator, shown in Figure 4.

Fundamentally, we apply the core idea from before to the 7^{th} byte of the credential instead of the final byte. Again, this means that the authenticator's 7^{th} byte will be identical to that of the original challenge. However, this time, the keystream byte for the final authenticator byte will differ from that used to obtain the credential, and will thus be unknown to the attacker.

Consider, without loss of generality, a client challenge in the form `00 00 00 00 55 66 00 00`. If our AES-CFB8 self-encryption primitive holds for the first *two* block cipher inputs, we obtain the corresponding client credential `00 00 00 00 55 66 XX YY`. Intuitively, obtaining a session key that fulfils this requirement has a probability of $256^{-2} = 2^{-16}$, which is expected to occur at least once if we fill the global challenge list with our connection attempts.

The attacker can now brute-force the last two bytes of the credential (`XX YY`) until authentication succeeds. However, the correct authenticator is still unknown to the attacker. As we have shown previously, the encryption "cancels out" for byte 7, but it does not for the final byte, that is, the attacker knows only that the authenticator has the form `00 00 00 00 55 66 00 ZZ`. Luckily, RPC connections are tolerant to incorrect authenticators (the request is simply ignored), so the attacker can then just test all possible authenticators to make the RPC call.

This meet-in-the-middle attack variant of *Onelogon* does not require a computer account to reset the global challenge list because it does not exceed 100,000 challenges. Its success is only limited by the speed of which an attacker can perform authentication attempts within the two minute timeout window for challenges.

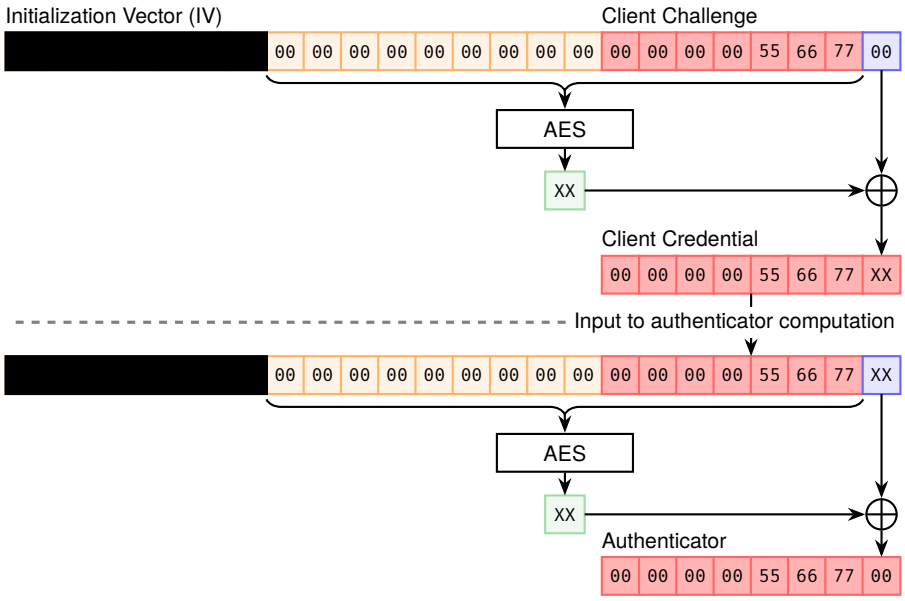

Figure 3: Computing a fake client credential and authenticator for the attack described in Section 4.4.

4.6 Full Attack Chain

For both attack scenarios of *Onelogon*, we can follow an attack chain similar to that of Zerologon (see Section 2.2.1).

First, the attacker needs to decide on a target account that is vulnerable to *Onelogon*, that is, an account for which secure RPC enforcement is disabled (see Section 4.1). To enumerate the accounts that are exempt from this enforcement, the attacker can parse group policy objects (GPO) that are found on the `SYSVOL` network share of the Domain Controller [22]. If a GPO defines the registry key `VulnerableChannelAllowList`, the Discretionary Access Control List (DACL) for that key can be used to extract the exempted machine accounts.

In theory, every account with either one of the `WORKSTATION_TRUST_ACCOUNT` or `SERVER_TRUST_ACCOUNT` flags can be used to authenticate over Netlogon. This includes managed service accounts (MSAs), but as the exemption policy is mainly designed to allow legacy systems to connect, it is unlikely that MSAs are exempted from the enforcement. Because of the flexibility of Windows' DACLs, administrators can, in addition to accounts, also exempt entities like user groups or security principals from secure RPC enforcement. Any account that is a member of an exempted entity can be targeted by the attacker. Moreover, if a group like `Authenticated Users` or `Everyone` is exempted, the attacker can compromise any computer account in the domain.

The attacker then uses one of the two *Onelogon* attack variants (Section 4.4 or Section 4.5) to perform an RPC call as the target user. As for Zerologon, the attacker uses a zero timestamp in the authenticator, and makes a `NetrServerPasswordSet2` RPC call [37].

Because we know that the session key will still encrypt all-zero plaintexts to all-zero ciphertexts, we can again provide an all-zero buffer as the argument (see Section 2.2.1). This sets the password length to zero, and thus clears the password for the target account. The attacker has now full control over the account with all of its privileges. For example, if the targeted account was the Domain Controller's computer account, the attacker has fully compromised the entire AD domain.

If the attacker is able to edit the registry key (or group policy) directly, for example, because access was delegated to lesser-privileged users, then the attacker can directly exempt a Domain Controller's computer account and use *Onelogon* to take full control of the AD domain.

4.7 Feasibility

We previously discussed in Section 4.1 that AD environment configurations that exempt some or all connections from secure RPC enforcement (and thus *potentially vulnerable to Onelogon*) occur frequently (≈23% of AD domains). Another, differently-scoped scan in 2026 for a different set of SpecterOps clients, revealed AD domains that are *directly vulnerable to Onelogon* at six of 270 enterprises (≈2.2%).

Next, we show that the attacks are also truly practical to perform. We implemented prototypes of our attacks in Python 3.13 using the Impacket library [9]. We measured the performance of our prototypes against a fully patched and up-to-date Domain Controller running the latest Windows Server 2025 Standard Evaluation Build 26100. The attacker machine ran Kali Linux 2025.4. We used VMware Workstation Pro 25H2 to virtualize both the attacker and target machines on

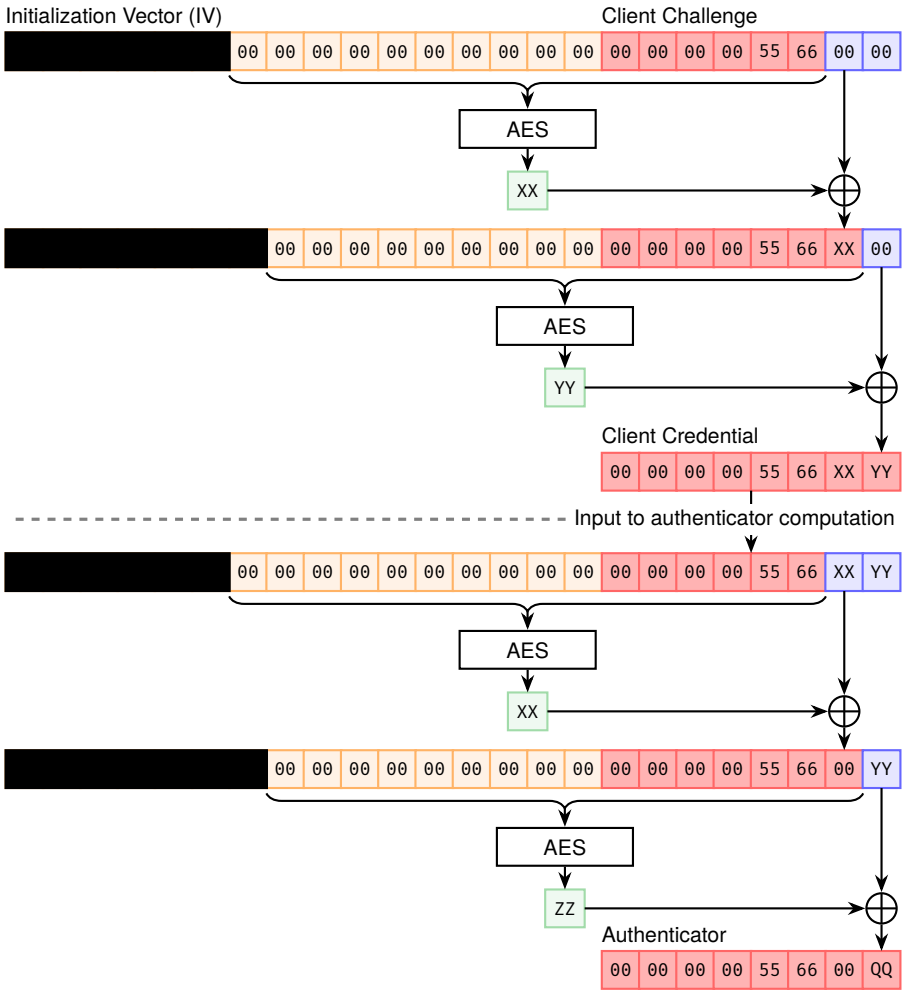

Figure 4: A meet-in-the-middle attack on the computation of the client credential and the RPC authenticator allows a practical attack on the Netlogon protocol.

a machine with an AMD Ryzen 9 7950X 16-core processor and 64 GB of DDR5 RAM.

We then measured the average time per authentication attempt made for each attack, and then calculate the expected average attack time for each of the attack scenarios. Note that the actual attack time is highly variable, because it is entirely up to chance when we encounter a vulnerable session key. However, we can anecdotally confirm that the attack time in practice tends to match these estimates. Table 1 shows an overview of these results.

We did not attempt to evaluate the naïve implementation of the full 32-bit brute-force attack with a full connection setup and teardown on every attempt. With an empty challenge list (the best possible scenario), a challenge request followed by an authentication attempt took roughly 100 ms; performing the expected number of 2^{31} requests would take 6.8 years to complete, which is prohibitively long.

For completeness, we also implemented more intelligent variants of the naïve brute-force. If an attacker relies on the global challenge list to perform only one authentication attempt per 100,000 challenges, performances increases drastically to around a month of brute-forcing, though the attacker then needs to wait two minutes for the list to be cleared. It is also possible to continue sending challenges, which improves performance somewhat, but the random replacement in the challenge list makes it difficult to estimate an expected attack time. The average time to send 100,000 new challenges increases from 3.164 s to 42.002 s when the list is full; in the best case, if we only ever replace old challenges (which is unrealistic), the expected attack time would drop to 10.44 days. By leveraging another compromised computer account to flush the challenge list, the cost of the brute-force attack can be further reduced.

The *Onelogon* attacks we introduced in this paper are those that are practical. Interestingly, our 24-bit brute-force attack (Section 4.4) and our meet-in-the-middle approach (Section 4.5) are similar in performance. Both compromise the target account with an expected attack time of 32.44 min and

Attack	Account	Time per cycle (s)	Expected # of cycles	Expected attack time
32-bit brute-force waiting for the timeout	✘	120.000	$\frac{2^{31}}{100000} \approx 21474.8$	29.83 days
32-bit brute-force with account	✔	3.164	$\frac{2^{31}}{100000} \approx 21474.8$	18.87 h
24-bit brute-force (Section 4.4)	✔	23.200	$\frac{2^{23}}{100000} \approx 83.9$	32.44 min
Meet-in-the-middle (Section 4.5)	✘	120.000	$\frac{2^{15}}{1778} \approx 18.4$	36.84 min

Table 1: Expected average attack times for *Onelogon*. The *time per cycle* refers the to the time taken for one round of filling the challenge list and running all authentication attempts. Note that the the performance of the meet-in-the-middle attack is instead bounded by the number of authentication requests that can be performed before the challenges expire. At 65.68 ms per authentication attempt, with 116.364 s remaining until timeout after placing 100,000 challenges, this translates to 1778 authentication attempts per cycle.

36.84 min respectively, thus the 24-bit attack being marginally faster, but the meet-in-the-middle approach eschewing the account requirement.

Finally, additional parallelization of the attacks would only lead to marginal performance gains because Netlogon serializes most operations, including authentication and modifications of the challenge list, behind a global lock.

5 Discussion

Addressing the *Onelogon* attacks, like Zerologon, is a non-trivial, complex endeavor because real-world AD deployments require some of the features that make the attacks possible in the first place. A combination of multiple approaches, mitigations and also active detection techniques, is required to comprehensively defend against *Onelogon*. In the following, we discuss defenses against *Onelogon*, some of which administrators of affected network can deploy themselves and others that require Microsoft to release patches. We have shared these defenses with Microsoft as part of our responsible disclosure.

5.1 Mitigations

Randomize the AES-CFB8 IV. The core flaw that enables both the Zerologon and *Onelogon* attacks is that the IV used for the AES-CFB8 encryption is hardcoded to an all-zero value [20, see § 3.1.4.4.1]. This results in all-zero inputs to the block cipher being used to derive a constant (for Zerologon) or near-constant (for *Onelogon*) key stream.

To conform with NIST recommendations, the IV used for any CFB mode variant should be unique and unpredictable for each plaintext [8]. If, as is commonly done incorrectly in other protocols, the IV is fully determined by the sender and transmitted alongside the message, a malicious client can still set it to contain only zeroes (and, in fact, this would also enable similar attacks with non-zero values). Instead, it is possible to derive sufficiently unpredictable IVs from a nonce (e.g., a message sequence number) by encrypting them with the session key, which the attacker is not able to recover in any of the attacks we describe [8].

However, modifying the Netlogon protocol to initialize the IV properly would be a backwards-incompatible change. Such a change would require a transition period and would still leave devices without support for the new protocol version vulnerable to attacks.

Reject Malicious Credentials. In both Zerologon and *Onelogon*, the goal is to find a session key that produces a ciphertext that is similar to the plaintext input. As a short-term mitigation that is backward-compatible, we can detect and reject connections that use such session keys based on a similarity metric (e.g., the number of matching bytes) between the plaintext and ciphertext.

Instead of looking (only) for sequences of identical bytes in the challenge itself, we can also, for example, reject connections where the first five bytes of the client challenge and client credential match. Like the original Zerologon patch, this would result in a false positive rate of 2^{-40} while rejecting all corresponding Zerologon and *Onelogon* challenges. A similar check can be applied to the computation of authenticators from the credential.

However, such a mitigation is not perfect and does not address the root cause itself. To bypass this mitigation, the attacker would need to further generalize our general *Onelogon* attack (Section 4.5) to cancel out the change to the fifth byte while obtaining the authenticator. The attacker then needs to brute-force the last three bytes of the authenticator, instead of only the last byte, which significantly increases the cost and time needed for exploitation. By cleverly combining this approach with appropriate timeouts and retries, it provides a strong probabilistic defense against Zerologon and *Onelogon*, just like stack canaries and address space layout randomization do for memory corruption vulnerabilities.

Restrict the Authenticator Timestamp. Once a secure channel has been successfully established by the attacker, an authenticator is required to prove authenticity for subsequent

RPC calls (see Section 2.1). Currently, the Domain Controller accepts every timestamp, including a value of zero (00:00 UTC, January 1, 1970), which allows an attacker to reuse the client credential as the authenticator.

In practice, clearly some level of flexibility is necessary to account for clock drift and imperfect synchronization between clocks, but accepting arbitrary 32-bit values appears unnecessarily lenient to us. It remains unclear to us why arbitrary values are supported for the authenticator timestamp — many other Active Directory features have significantly less tolerance for clock disparities.

As the timestamp is added to the first four bytes of the client credential, this mitigation would force an attacker to at least brute-force the last four bytes of the authenticator's credential.

Prevent Authenticator Brute-Forcing. In Section 4.5, we reduce the overall brute-force complexity of the attack by shifting it to the authenticator computation. Similarly, both of the previous mitigations would force an attacker to brute-force additional bytes in the authenticator.

Legitimate clients with an established secure channel should know both the session key and the client credential that was used for the initial Netlogon authentication. If the Domain Controller receives an authenticator that does not match the expected value, then either an outdated secure channel is used, or an attacker is attempting to brute-force the authenticator. In either case, we should not only reject the RPC call, but also terminate the secure channel. Legitimate clients can re-establish the secure channel, but attackers will no longer be able to brute-force the authenticator.

Ensure RPC Signing and Sealing. An attacker who has established a secure channel using *Onelogon* still requires the session key to sign subsequent RPC packets. If secure RPC is enforced, this means that an attacker cannot use that channel to make any RPC calls, and the channel is useless.

Unfortunately, Microsoft was forced to allow deployments to exempt certain legacy clients from secure RPC enforcement because such clients are widely deployed. Since uses of this exemption remain even five years after the initial Zerologon publication (see Section 4.1), it is evident that such legacy setups remain active today.

We highly recommend network administrators to audit both the GPO and the corresponding `VulnerableChannelAllowList` key on all Domain Controllers to ensure that no machine accounts are listed. If some legacy clients remain that are not capable of secure RPC, they should be isolated to a separate Active Directory forest to limit the impact of *Onelogon*. Moreover, it is crucial to audit and restrict write privileges for the `VulnerableChannelAllowList` key, to prevent attackers from enabling the vulnerability for specific target accounts.

5.2 Detection

Comprehensively addressing the *Onelogon* attacks properly is non-trivial: three of the four mitigations that we presented require Microsoft to make changes to Netlogon. Operators and administrators are left with only one mitigation that they can deploy themselves, namely ensuring proper RPC signing and sealing, but this might be incompatible with their AD deployment. If, in practice, we might be unable to patch the vulnerability, then it is crucial for us to at least be able to detect exploitation attempts and stop exploitation. Therefore, in the following, we discuss multiple ways of *detecting*, rather than preventing, *Onelogon* attacks within a network.

Windows Event Log. In the past, Microsoft introduced several event IDs to log potential exploitation attempts of the Zerologon vulnerability [18].

Specifically, event IDs 5827 and 5828 are logged when a non-exempted account attempts to establish a secure channel without secure RPC and this is rejected. If, on the other hand, such a connection attempt is permitted, and an RPC call is made, the Netlogon service will log events with the IDs 5830 (for machine accounts) or 5831 (for trust accounts) [17].

For all attacks that we presented, brute-forcing the session key will almost certainly lead to multiple failed authentication attempts (unless we guess the session key correctly on the first try, *Onelogon* will fail at least one authentication attempt). For the first unsuccessful authentication attempt, we observe an event with ID 5805 in the event log. After the secure channel is successfully established, the attack described in Section 4.4 will additionally log one event with ID 5830 or 5831, depending on the target account. Our meet-in-the-middle attack from Section 4.5 has to brute-force a single byte of the authenticator; every failed attempt at this last step will also result in a logged event.

Therefore, if operators are actively monitoring for these events, they can detect Zerologon and *Onelogon* attacks. This enables them to quarantine and isolate the host launching the attacks, stopping the exploitation attempts before the attacker can move laterally, escalate privileges, or compromise the domain with these attacks.

Netlogon Debug Logs. Microsoft also provides the capability to enable debug logging for the Netlogon service on Domain Controllers [21]. When enabled, the Netlogon server logs detailed information about Netlogon RPC calls including authentication attempts. All three *Onelogon* attack variants rely on repeated authentication attempts to find a vulnerable session key.

Initially, an attacker populates the global challenge list with 100,000 pending sessions via `NetrServerReqChallenge` calls. Once the list reaches 5,000 entries, every subsequent call to `NetrServerReqChallenge` causes the Netlogon server to write a "critical" debug log entry, which indicate that the

session list is starting to fill up and that sessions will be removed after two minutes. Additionally, it also emits a critical debug log entry for every failed authentication attempt that lists which account failed to authenticate.

Bezzateev, Fomicheva, and Zhemelev sketch a detection mechanism for Zerologon that relies on the Netlogon debug log in their short paper [3]. While debug logs commonly have significant runtime overhead and operators might not be inclined to enable them, such a mechanism can be used to detect ongoing *Onelogon* exploitation attempts. Administrators who are unable to enforce secure RPC signing and sealing because of legacy devices in their network might find the overhead of debug logs more palatable than the cost of having to replace all devices that do not support RPC signing and sealing.

Network Traffic Analysis. Brute-forcing session keys and authenticators over the network also requires sending a large amount of RPC calls to the Domain Controller. At an average size of 130 B per RPC call, the 24-bit bruteforce variant described in Section 4.4 will result in approximately 1 GB of network traffic to the Domain Controller. Monitoring network traffic, for example, through an network intrusion detection or prevention system, for unusually high RPC call counts and traffic flow can also help detect ongoing *Onelogon* attacks.

5.3 Future Work

At its core, the *Onelogon* attack that we introduced in this paper relies on the same flawed AES-CFB8 implementation that the Zerologon vulnerability exploited. Without fundamentally fixing the implementation of how Netlogon is incorrectly using the AES-CFB8 encryption mode and following cryptographic best practices (NIST recommendations), new variants of the attacks and other vulnerabilities might be discovered in Netlogon in the future. Currently, only the Netlogon Security Support Provider (SSP) protects against authentication bypasses in the Netlogon protocol. This protection only remains effective as long as no new vulnerabilities are found in the session key establishment process or the signing and sealing of RPC packets. Consequently, future work should closely examine the Netlogon SSP to identify potential weaknesses.

Samba also implements the Netlogon protocol following the MS-NRPC specification [38]. Since Samba was also found vulnerable to Zerologon [1], it is likely that Samba is also vulnerable to the *Onelogon* attack. However, we have not empirically validated that this is the case, as Microsoft's implementation is most widely used. Future work should investigate if other Netlogon implementation are also vulnerable.

Mollema combined the original Zerologon attack with a relay attack to compromise AD domains without resetting any machine account passwords [26]. This attack exploits the pass-through authentication mechanisms of the Netlogon protocol to request the session key of a relayed NTLM connection. The *Onelogon* attack achieves the same goal of establishing a secure channel with any computer account. The attack by Mollema could be adapted to the *Onelogon* attack to bypass authentication and obtain the session key. Unlike the attacks we presented in this paper, the relay attack would grant administrative privileges directly to the targeted computer instead of compromising its computer account. Due to the overall complexity of the *Onelogon* attack and the higher time requirements, a different approach is likely needed to work around the time constraints of the relay attack, which future research should study.

Finally, fuzzing the Netlogon protocol implementation could also reveal new vulnerabilities. In particular, fuzzing RPC functions that should require an established secure channel could expose new vulnerabilities that allow privilege escalation from a normal computer account to a Domain Controller or even across trust boundaries. Here, we want to point out CVE-2026-41089, a stack-based buffer overflow in Netlogon discovered after the submission of this paper [25].

6 Conclusion

In this paper, we introduced *Onelogon*, a new Zerologon-like attack that circumvents Microsoft's mitigations in certain, relatively common Active Directory (AD) configurations. *Onelogon* relies on a cryptographic flaw in the Netlogon protocol's authentication mechanism that was not sufficiently addressed by Microsoft's Zerologon patches.

We describe two variants of *Onelogon* that both allow an attacker to take over AD accounts and, in some cases, the entire domain in less than 40 minutes. We show how Microsoft and operators of AD domains can defend against these attacks, make multiple practical recommendations as to address the vulnerability at its root cause and probabilistic to retain backward-compatibility, and discuss how administrators can detect ongoing and successful attacks in practice.

Onelogon is, unfortunately, yet another case where the support for legacy systems ultimately compromises the security posture of an entire network. To help administrators and operators secure their systems, we make our implementation to detect vulnerable AD domains publicly available with the publication of this paper.

Disclosure

We responsibly disclosed our findings to Microsoft in December 2025 (see Ethical Considerations). Unfortunately, Microsoft has so far argued that any use of Netlogon RPC without secure RPC is "documented vulnerable" [18] and has not been willing to provide additional patches or guidance. Therefore, to provide practitioners sufficient time to address the *Onelogon* vulnerability themselves, we additionally disclosed the issue to CERT-Bund [33]; we do not expect patches to be forthcoming.

Acknowledgments

This research is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA – 390781972, the ICANN Grant Program, and the Internet Society Foundation, as well as the Vienna Science and Technology Fund (WWTF) and the City of Vienna [10.47379/ICT19056]. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the respective funding agencies.

Ethical Considerations

We disclosed the vulnerability and the *Onelogon* attacks that we described in this paper to Microsoft in December 2025, with the goal of engaging in a coordinated disclosure process to minimize impact for operators of AD domains.

Unfortunately, Microsoft has decided not to release a new patch and is instead arguing that the documentation for the required group policy options already includes a warning that exempted accounts might be "at risk" [18]. We disagree with this assessment and approach. The concept of "at risk" indicates that there is *assumed to be a non-zero probability* that a vulnerability *may* exist and that operators *should* take steps to minimize *potential* risk. However, it does not indicate that there is a *specific* and *confirmed* vulnerability that can be actively exploited, especially not within less than 40 minutes, and that operators *must* take steps to mitigate a *concrete* risk. Due to the relatively common use of the group policy and ease of exploitation, we believe that Microsoft's approach puts a considerable number of AD deployments in danger.

To nevertheless provide stakeholders sufficient time to react and mitigate the concrete risk that *Onelogon* poses, we additionally disclosed *Onelogon* to CERT-Bund [33]. With the release of the paper, we will also release our prototype implementation to allow administrators to check for vulnerable configurations in their AD deployments themselves. Finally, with the goal of preventing that this vulnerability is exploited, we have also already discussed detection mechanisms that Microsoft can implement if they chose to and others that operators can apply independently (see Section 5.1), and we also provided clear indicators of attempt and successful attacks (see Section 5.2).

References

[1] A. Bartlett and D. Bagnall. *Unauthenticated domain takeover via netlogon ("ZeroLogon")*. 2020. URL: https://www.samba.org/samba/security/CVE-2020-1472.html (visited on 11/26/2025).

[2] M. Bécard and G. André. *NTLM reflection is dead, long live NTLM reflection! – An in-depth analysis of CVE-2025-33073*. June 11, 2025. URL: https://www.synacktiv.com/en/publications/ntlm-reflection-is-dead-long-live-ntlm-reflection-an-in-depth-analysis-of-cve-2025 (visited on 12/02/2025).

[3] S. V. Bezzateev, S. G. Fomicheva, and G. A. Zhemelev. "Agent-based ZeroLogon Vulnerability Detection." In: *2021 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF)*. 2021. DOI: 10.1109/WECONF51603.2021.9470548.

[4] F. Butler, I. Cervesato, A. D. Jaggard, A. Scedrov, and C. Walstad. "Formal analysis of Kerberos 5." In: *Theoretical Computer Science* 367.1 (2006). Automated Reasoning for Security Protocol Analysis, pp. 57–87. DOI: 10.1016/j.tcs.2006.08.040.

[5] F. Cardoso. *What Is Zerologon?* 2020. URL: https://www.trendmicro.com/en/what-is/zerologon.html (visited on 11/05/2025).

[6] N. Corporation. *Zerologon Vulnerability Explained: Risks, Exploits and Mitigation*. 2020. URL: https://netwrix.com/en/cybersecurity-glossary/cyber-security-attacks/zerologon-vulnerability/ (visited on 11/05/2025).

[7] J. De Ciercq. "Deflecting Active Directory Attacks." In: *ISSE 2006 — Securing Electronic Busines Processes: Highlights of the Information Security Solutions Europe 2006 Conference*. 2006, pp. 168–175. DOI: 10.1007/978-3-8348-9195-2_18.

[8] M. Dworkin. *SP 800-38A. Recommendation for Block Cipher Modes of Operation Methods and Techniques*. Dec. 1, 2001. DOI: 10.6028/NIST.SP.800-38A.

[9] Fortra et al. *NRPC implementation in impacket*. 2025. URL: https://github.com/fortra/impacket/blob/master/impacket/dcerpc/v5/nrpc.py (visited on 05/27/2025).

[10] Fortra et al. *secretsdump.py script from impacket*. 2025. URL: https://github.com/fortra/impacket/blob/master/impacket/examples/secretsdump.py (visited on 11/04/2025).

[11] H. He, J. Self, K. French, S. Bhunia, M. Salman, and P. A. Regis. "Zerologon Explored: In-Depth Analysis and Mitigation Strategies for Microsoft's Critical Vulnerability." In: *2024 IEEE 24th International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW)*. 2024. DOI: 10.1109/CCGridW63211.2024.00025.

[12] H. Kabibo. *A journey into forgotten Null Session and MS-RPC interfaces*. 2024. URL: https://securelist.com/no-auth-domain-information-enumeration/112629/ (visited on 10/16/2025).

[13] H. Kabibo. *NauthRPC*. 2024. URL: https://github.com/sud0Ru/NauthNRPC (visited on 10/16/2025).

[14] G. McDonald, P. Papadopoulos, N. Pitropakis, J. Ahmad, and W. J. Buchanan. "Ransomware: Analysing the Impact on Windows Active Directory Domain Services." In: *Sensors* 22.3 (2022). DOI: 10.3390/s22030953.

[15] Microsoft. *Enabling debug logging for the Netlogon service*. 2006. URL: https://learn.microsoft.com/en-us/troubleshoot/windows-client/windows-security/enable-debug-logging-netlogon-service (visited on 11/07/2025).

[16] Microsoft. *[MS-NRPC]: Netlogon Remote Protocol*. Diff. between Rev. 36.0 and Rev. 37.0. Aug. 26, 2020. URL: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/ (visited on 11/07/2025).

[17] Microsoft. *CVE-2020-1472: Netlogon Elevation of Privilege Vulnerability*. Aug. 11, 2020. URL: https://msrc.microsoft.com/update-guide/en-US/advisory/CVE-2020-1472 (visited on 12/09/2025).

[18] Microsoft. *How to manage the changes in Netlogon secure channel connections associated with CVE-2020-1472*. 2020. URL: https://support.microsoft.com/en-us/topic/how-to-manage-the-changes-in-netlogon-secure-channel-connections-associated-with-cve-2020-1472-f7e8cc17-0309-1d6a-304e-5ba73cd1a11e (visited on 11/07/2025).

[19] Microsoft. *[MS-DRSR]: Directory Replication Service (DRS) Remote Protocol*. Rev. 44.0. Apr. 23, 2024. URL: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-drsr/ (visited on 11/05/2025).

[20] Microsoft. *[MS-NRPC]: Netlogon Remote Protocol*. Rev. 47.0. Sept. 9, 2025. URL: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/ (visited on 11/05/2025).

[21] Microsoft. *Enabling debug logging for the Netlogon service*. 2025. URL: https://learn.microsoft.com/en-us/troubleshoot/windows-client/windows-security/enable-debug-logging-netlogon-service (visited on 11/24/2025).

[22] Microsoft. *How to rebuild the SYSVOL tree and its content in a domain*. 2025. URL: https://learn.microsoft.com/en-us/troubleshoot/windows-server/group-policy/rebuild-sysvol-tree-and-content-in-a-domain (visited on 11/11/2025).

[23] Microsoft. *Sysmon*. 2025. URL: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon (visited on 11/07/2025).

[24] Microsoft. *Windows authentication overview*. 2025. URL: https://learn.microsoft.com/en-us/windows-server/security/windows-authentication/windows-authentication-overview (visited on 11/29/2025).

[25] Microsoft. *CVE-2026-41089 - Security Update Guide - Microsoft - Windows Netlogon Remote Code Execution Vulnerability*. May 12, 2026. URL: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-41089 (visited on 06/09/2026).

[26] D.-J. Mollema. *A different way of abusing Zerologon (CVE-2020-1472)*. 2020. URL: https://dirkjanm.io/a-different-way-of-abusing-zerologon/ (visited on 11/25/2025).

[27] J. Myllyla and A. Costin. "Reducing the Time to Detect Cyber Attacks : Combining Attack Simulation With Detection Logic." In: *FRUCT'29 : Proceedings of the 29th Conference of Open Innovations Association FRUCT*. 2021. URL: https://old.fruct.org/publications/acm29/files/Myl.pdf (visited on 11/07/2025).

[28] S. Narang. *CVE-2020-1472: Microsoft Finalizes Patch for Zerologon to Enable Enforcement Mode by Default*. 2021. URL: https://de.tenable.com/blog/cve-2020-1472-microsoft-finalizes-patch-for-zerologon-to-enable-enforcement-mode-by-default (visited on 11/05/2025).

[29] B. Neuman and T. Ts'o. "Kerberos: an authentication service for computer networks." In: *IEEE Communications Magazine* 32.9 (1994), pp. 33–38. DOI: 10.1109/35.312841.

[30] NIST. *CVE-2020-1472 Detail*. Aug. 17, 2020. URL: https://nvd.nist.gov/vuln/detail/cve-2020-1472 (visited on 12/09/2025).

[31] S. H. Qatinah and I. A. Al-Baltah. "Kerberos Protocol: Security Attacks and Solution." In: *2024 1st International Conference on Emerging Technologies for Dependable Internet of Things (ICETI)*. 2024. DOI: 10.1109/ICETI63946.2024.10777133.

[32] E. Shamir. *The Renaissance of NTLM Relay Attacks: Everything You Need to Know*. 2025. URL: https://specterops.io/wp-content/uploads/sites/3/2025/04/SPO_NTLM_WhitePaper_Updated.pdf (visited on 12/02/2025).

[33] B. für Sicherheit in der Informationstechnik. *BSI - CERT-Bund*. Apr. 9, 2026. URL: https://www.bsi.bund.de/EN/CERT-Bund (visited on 06/09/2026).

[34] M. Simakov and Y. Zinar. *Zerologon (CVE-2020-1472): An Unauthenticated Privilege Escalation to Full Domain Privileges*. 2020. URL: https://www.crowdstrike.com/en-us/blog/cve-2020-1472-zerologon-security-advisory/ (visited on 11/05/2025).

[35] A. Solino. *Windows Pass-Through Authentication Methods Improper Validation*. 2015. URL: https://www.coresecurity.com/core-labs/advisories/windows-pass-through-authentication-methods-improper-validation (visited on 11/25/2025).

[36] T. Tervoort. *Taking over Windows Systems with a Netlogon Man-in-the-Middle Attack (CVE-2019-1424)*. Nov. 15, 2019. URL: https://cybersecurity.bureauveritas.com/blog/cve-2019-1424 (visited on 12/02/2025).

[37] T. Tervoort. *Zerologon: Unauthenticated domain controller compromise by subverting Netlogon cryptography*. 2020. URL: https://www.secura.com/uploads/whitepapers/Zerologon.pdf (visited on 05/23/2025).

[38] A. Tridgell and Samba Team. *Samba NRPC implementation*. 2025. URL: https://gitlab.com/samba-team/samba (visited on 11/26/2025).