Hunting for Zerologon

BI.ZONE · Follow
13 min read · Nov 9, 2020

By Demyan Sokolin, Alexander Bolshakov, Ilyas Igisinov, Vadim Khrykov

CVE-2020–1472, or Zerologon, has already been named one of the most dangerous vulnerabilities discovered in recent years. It allows an attacker to compromise a domain controller's machine account and access the contents of the entire Active Directory database. The only thing the attacker requires to exploit this vulnerability is a network connection to the company's domain controller.

We did our own research on Zerologon and developed various methods to detect its existence: by Windows audit log events, by network traffic and by YARA rules. This article will focus on each of them in detail. The methods described in this article can be used for hunting the Zerologon vulnerability to then subsequently become part of your use-case base.

The Concept of the Zerologon Vulnerability

Zerologon is a vulnerability in the encryption protocol that is used by the Netlogon service. The protocol allows computers to authenticate with a domain controller and update the password for their Active Directory account. It is this feature that makes Zerologon dangerous. More

specifically, the vulnerability allows the attacker to pass themselves off as the domain controller and change his password. The attacker gains access to the domain controller with the highest privileges, and by extension, to the corporate network itself. After changing the password, the attacker can use the domain controller account to develop the attack, for example by performing a DCSync attack (obtaining Active Directory accounts through the replication mechanism).

In September, a Dutch security researcher Tom Tervoort from Secura published a detailed description of the vulnerability. It turned out that Zerologon was caused by a flaw in the cryptographic authentication scheme used by Netlogon Remote Protocol (MS-NRPC). MS-NRPC handshake and authentication involves the use of AES-CFB8 mode (with 8-bit encrypted feedback). This is a version of the AES block cipher and is designed to work with input blocks of 8 bytes instead of the usual 16 bytes (128-bit).

As Tervoort discovered, applying AES-CFB8 cipher to plaintext consisting of only zeros will result in the same plaintext cipher. This is due to an implementation error for 1 of the 256 keys.

Usually, the client computer that wants to communicate with a Netlogon server, such as the Windows domain controller, starts out by sending eight random bytes (what is often called nonce, abbreviation for number used once) to the server.

The process of fixing the vulnerability was divided into two stages. In August 2020, as part of the ‘August Tuesday’, Microsoft released the first update to patch the vulnerability. This update fixes the vulnerability by applying a secure RPC via a secure Netlogon channel. Although the vulnerability has been corrected, Microsoft plans to prohibit the use of unsecured Netlogon connections by default. This will be done in the first quarter of 2021. The update will be released as part of the second phase of remediation.

Zerologon Exploitation Stages

According to the Secura technical document, the Zerologon exploitation process consists of three stages.

1. Sending zero bytes. Instead of sending eight random bytes, the attacker sends eight null bytes. The attacker repeats sending these messages until the server successfully accepts one of them and thus bypasses the authentication process. In the case of Zerologon, an average of 256 attempts to send a single zero byte message are required to connect to the server successfully.

2. Disabling the RPC signing and sealing mechanism. MS-NRPC uses the RPC signing and sealing mechanism for transport layer encryption. Usually, encryption is a mandatory process for data transfer, but in MS-NRPC this mechanism is not mandatory and is managed by the client. The server will not deny a connection to clients who do not request the use of encryption. This means that you can simply disable encryption in the message header. Actually, the second stage is to disable the RPC signing and sealing mechanism so that messages are sent in raw form and the attacker could use the MS-NRPC protocol methods.

3. Changing the account password. The third stage of exploitation of the Zerologon vulnerability is to change the password for the account of the domain controller, which was previously used to establish the connection. Using the NetrServerPasswordSet method in MS-NRPC, the attackers can change the password for the computer account. The simplest way to use this method is to remove the current password or, in other words, set the password to an empty value. It is also possible to change the password by simply sending a request with your preferred new password. After changing the password, the attackers can use the domain controller account to develop their attack, for example, replicate the Active Directory database, as mentioned earlier.

PoC and Exploits

The first PoC was published by Secura at GitHub. The script will try to exploit the Zerologon vulnerability: once the connection is established successfully, it will terminate the work immediately and will not perform any actions via Netlogon.

As for exploits, there have already been many at the time of publishing this article. However, they all behave the same way: they either reset the password or set it to a user-defined value.

There is also an alternative method of exploiting Zerologon. It was discovered by a security researcher Dirk-Jan Mollema, but it must be done in conjunction with other vulnerabilities and deserves a separate article.

Methods of Detection

The aim of our research was to find all possible methods to detect the fact of Zerologon being exploited. We developed and tested the logic of correlation rules, signatures for IPS-\IDS network systems and the YARA rule for detecting artifacts in the memory of the LSASS process.

Detection using the Netlogon debugging log

Since the vulnerability is exploited in the Netlogon protocol, let us look at the Netlogon event log. By default, only a fraction of the events in the

Windows logs are audited, but you can enable the Netlogon debug mode using the command `nltest /dbflag:0x2080ffff`, which must be run with administrator privileges. The Netlogon service will then save most of the important events to the log file, which can be found at `C:\Windows\debug\netlogon.txt`. The Netlogon service must then be restarted. The screenshot below shows a few interesting lines that were logged by the Netlogon service during the exploitation of the Zerologon vulnerability.

```
#Session opened
10/18 22:33:17 [SESSION] [3116] EVIL: NetrServerAuthenticate entered: DC (10.0.0.1) on account DC$ (Negot: 212fffff)
#Failed attempts to authenticate with zeroed password
10/18 22:33:17 [CRITICAL] [3116] EVIL: NetrServerAuthenticate: Bad password 0 for DC on account DC$
10/18 22:33:17 [CRITICAL] [3116] EVIL: NetrServerAuthenticate: Bad password 0 for DC on account DC$
10/18 22:33:17 [CRITICAL] [3116] EVIL: NetrServerAuthenticate: Bad password 0 for DC on account DC$
...
...
...
10/18 22:33:17 [CRITICAL] [3116] EVIL: NetrServerAuthenticate: Bad password 0 for DC on account DC$
10/18 22:33:17 [CRITICAL] [3116] EVIL: NetrServerAuthenticate: Bad password 0 for DC on account DC$
10/18 22:33:17 [CRITICAL] [3116] EVIL: NetrServerAuthenticate: Bad password 0 for DC on account DC$
10/18 22:33:17 [CRITICAL] [3116] NlTransportLookup: \\DC: Cannot NetSessionEnum 2312
10/18 22:33:17 [SESSION] [3116] EVIL: NetrServerAuthenticate returns Success: DC on account DC$ (Negot: 212fffff)
#Setting password using NetpServerPasswordSet
10/18 22:33:17 [SESSION] [3116] EVIL: NetpServerPasswordSet: Comp=DC Acc=DC$ Entered
10/18 22:33:17 [SESSION] [3116] EVIL: NetpServerPasswordSet: Comp=DC Acc=DC$ Changing password locally
#"d98c1dd4 04b2008f 980980e9 7e42f8ec" is md5 hash of empty password in little-endian format
#md5 hash of empty passwordv in normal format is D41D8CD98F00B204E9800998ECF8427E
10/18 22:33:17 [SESSION] [3116] Setting Password of 'DC$' to: d98c1dd4 04b2008f 980980e9 7e42f8ec  ..............B~
10/18 22:33:17 [SESSION] [3116] EVIL: NetpServerPasswordSet: Comp=DC Acc=DC$ returns 0x0
```

Example of events recorded in the Netlogon debugging log during exploitation of the Zerologon vulnerability

When Netlogon debug mode is configured, every step of the attack is logged and retaining even the MD5 password hash that was set for the domain controller account. The MD5 hash is logged in the little-endian format. However, this method of monitoring is not very convenient from the point of view of gathering events in the SIEM system, and it also requires that the Netlogon debug mode is enabled on all domain controllers.

Detection of artifacts left by exploits

The first stage of the exploitation process is actually brute force. This exploit carries out multiple authentication attempts using Netlogon on a domain controller with a ClientChallenge message consisting of 8 zero bytes. Multiple unsuccessful authentication attempts result in the generation of event 5805 on the domain controller: ‘The session setup from the computer <hostname> failed to authenticate. The following error occurred: Access is denied’ (see example event in the screenshot below).

Additionally, if an incorrect domain controller account name was specified in the exploit, an attempt to authenticate with an incorrect account will generate event 5723 on the domain controller: ‘The session setup from computer <hostname> failed because the security database does not contain a trust account <account> referenced by the specified computer’ (see the screenshot below).

In the case of Zerologon using the mimikatz utility or other exploits running from a host named kali, artifacts remain in the events (the host name with Kali Linux installed changes very rarely, which is why it is considered in the rule). Mimikatz with unchanged source code leaves an artifact as a substring of mimikatz in events 5805 and 5723.

In our research, we also looked at several variations in using exploits and identified cases of exploiting the Zerologon vulnerability from a virtual machine running Kali Linux, where an artifact in the form of a kali substring remained in events 5805 and 5723, specified as the authentication source host.

Event 5805. Error of session authentication from host kali. Access denied

Event 5723. Error setting up a session from the host mimikatz due to the absence of an evildc account in the security database

The logic of the first rule for detecting an attempt to exploit a Zerologon vulnerability will therefore be as follows:

(EventID = '5805' OR EventID = '5723') AND (Message contains 'kali' OR Message contains 'mimikatz')

Detection, based on the differences between legitimate password changes and those caused by exploits

We will now look at detection methods based on a different set of events when the computer account password is periodically updated and when it is changed using Zerologon.

By default, the computer account password has a maximum validity period of 30 days. Upon expiry of this period, the password will be changed by the operating system using the Netlogon protocol. This value can be changed using local or group policy:

Computer Configuration → Policies → Windows Settings → Security Settings → Local Policies → Security Options → Domain member: Maximum machine account password age.

When the password of a computer account is changed, several events are generated in the Security event log. The first is an event with EventID 4742 'A computer account was changed', which has a TargetUserName equal to the name of the domain controller account and PasswordLastSet set to the date the password was changed. This event means that the password for the domain controller computer account has been changed.

Event 4742. DC$ account password was changed at 5:46:34 PM

The System event log contains another interesting event with EventID 5823 — ‘The system successfully changed its password on the domain controller. This event is logged when the password for the computer account is changed by the system. It is logged on the computer that changed the password’. This event means that the computer account has been legitimately changed by the system.

Event 5823. The password for the domain controller account was successfully changed by the system at 5:46:34 PM

Thus, two events will be generated when a legitimate change of password is performed for a domain controller account: 5823 and 4742. However, when using Zerologon, event 5823 will not be recorded in the event log.

The logic of the second rule to detect the exploitation Zerologon may look like this:

when both of (**EventID = '4742' AND TargetUserName IN "Domain_Controller_Accounts_List" AND PasswordLastSet !=** '-') and not (**EventID = '5823'**) were detected on the same host within 1 minute.

The rule implemented according to the logic described above will enable the exploitation Zerologon to be detected with high precision. No additional configuration of the audit policy will be required for it to work. The only thing that will be required is to prepare a list of the organisation's domain controllers and place it in the Domain_Controller_Accounts_List.

However, it is possible to look at the process of detecting the exploitation of Zerologon from a different angle. Previously, we wrote that the first stage of exploitation is brute force. Therefore, if both 5805 and 4742 events are detected within one minute on the same domain controller, it will also indicate that Zerologon has been successfully exploited.

The logic of our third rule for detecting the exploitation of Zerologon may look like this:

when both of (**EventID = '4742' AND TargetUserName IN "Domain_Controller_Accounts_List" AND PasswordLastSet !=** '-') and

(**EventID = '5805'**) were detected on the same host within 1 minute.

Detecting exploitation based on network traffic

As we mentioned at the beginning of this article, the first step in exploiting the vulnerability involves multiple attempts to send a ClientChallenge with a pre-set zero key value to circumvent authentication. Practice has shown that in the vast majority of cases at least 10 attempts are required to bypass the authentication mechanism. This anomaly can be easily detected in network traffic. The DCE/RPC protocol is used to send ServerChallenge requests using the NetrServerReqChallenge method and attempts to authenticate using the NetrServerAuthenticate method with a zero ClientChallenge value to the MS-NRPC RPC interface.

The traffic from Mimikatz version 2.2.0–20200916, without DCE/RPC load encryption when bypassing zero key authentication, contains an artifact in the Computer Name variable of the NetrServerReqChallenge method (Opnum 4).

The traffic from Mimikatz version 2.2.0–20200916 when bypassing authentication

The traffic of Mimikatz version 2.2.0–20200918 with DCE/RPC load encryption (using NTLMSSP with RPC_C_AUTHN_LEVEL_PKT_PRIVACY authentication level) does not contain unique artifacts. However, an attack can be detected using the NetrServerReqChallenge (Opnum 4) and NetrServerAuthenticate2 (Opnum 15) repetitive methods in the traffic from a single source within a few seconds' time frame.

Encrypted traffic from Mimikatz version 2.2.0–20200918 when bypassing authentication

With PoC, the NetrServerReqChallenge and NetrServerAuthenticate method pairs are sent in separate TCP sessions. Though, the approach to detecting based on repetitive NetrServerReqChallenge and NetrServerAuthenticate pairs is the same.

In the final phase of the NetrServerPasswordSet2 (Opnum 30) attack with a sequence of 516 zero bytes, an empty password is set for the domain controller computer account.

The traffic of the password reset request using the NetrServerPasswordSet2 method

This makes it possible to detect a Zerologon attack by examining the network traffic through an abnormally large number of single-source DCE/RPC requests with pairs of NetrServerReqChallenge and NetrServerAuthenticate methods in a short period of time. The activity can be more accurately identified under certain conditions based on unique artifacts in traffic rather than statistical anomalies.

Detecting artifacts in the LSASS address space

Since during the exploitation of Zerologon authentication is performed on the domain controller using the MS-NRPC function `hNetrServerAuthenticate2 (hNetrServerAuthenticate3)`, the Local Security Authority Subsystem Service (LSASS) is used in the authentication process. When calling the authentication server, an artifact — a structure containing the parameters transferred to the `hNetrServerAuthenticate2 (hNetrServerAuthenticate3)` function — is sent to the `lsass.exe` process address space (see figure below).

Example of a call to the hNetrServerAuthenticate3 function with the arguments transferred to it

Fragment of the lsass process address space dump with artifacts after Zerologon was exploited

A fragment of the source code of one of the exploits and a fragment of the dump of the address space of the lsass process with the artifacts left behind can be seen in the relationship. The arguments transferred to the function have remained in memory and the overall data structure has been preserved. If you look at the end, the 4 bytes (0x212fffff) are flags — Netlogon Negotiable Options. There are 8 zero bytes in front of it, representing zero ciphertext. The DC host name is then written in front of them three times in the same form and sequence as the arguments were passed to the function. A byte is also visible in memory, meaning the type of secure channel Netlogon ServerSecureChannel (06).

However, sometimes for unspecified reasons this and other bytes may take different values that are not related to the above description. Artifacts in the address space of the `lsass.exe` process are stored in a segment that does not involve long-term data storage and can be deleted at any time after they appear. Our research has shown that in most cases, artifacts will remain in the `lsass.exe` process memory for no longer than half an hour after which they are overwritten with other data.

The main marker that allows us to find this section of the address space and use it for further checks are Netlogon Negotiable Options flags. They represent a 32-bit number that is formed in binary form from a set of different parameters.

According to the Secura technical paper, as well as other research into this vulnerability, the second stage of exploiting Zerologon is to disable the RPC mechanism by disabling the desired bit in the flag. In all the exploits and studies, we found that the flag value 0x212fffff is used to disable this mechanism. However, our research has shown that the only parameter influencing the success of exploitation is the 25th bit, which is responsible for enabling AES-CFB8 support ‘Supports Advanced Encryption Standard (AES) encryption (128 bit in 8-bit CFB mode) and SHA2 hashing’. Only this

flag is required to successfully exploit the vulnerability, the others can be set to 1 or 0, this will not affect the result. Changing several bits in the flag thus results in the loss of the anchor looked for in the lsass address space in the form of bytes ff ff 2f 21, which contain flags that are used by all the exploits tested during the survey.

In addition to bypassing the detection logic of YARA rules, which will be described below, the use of some Netlogon Negotiable Options flags results in the lsass address space leaving no artifacts that can be used to identify the exploitation of Zerologon. One of these flags is 0x312fffff. Our research confirms the hypothesis that the flags which make it possible to disable the RPC signing and sealing mechanism are not actually designed for this purpose.

During the research, we analysed various YARA rules (including those developed by specialists from Cynet) aimed at detecting traces of Zerologon in the system's memory. We found that existing rules do not take into account the various anomalies associated with modifying some significant bytes in the address space, and we decided to implement a new YARA rule that takes these anomalies into account.

Keep in mind, that exploitation is possible with almost any combination of Netlogon Negotiable Options flags, but either way, it is the flags that serve as anchors for detecting artifacts in the memory. That is why we have adopted the following restriction: to use in our YARA rule the common combination of flags 0x212fffff. During the development process, we tested various exploits, including the zerologon module of the mimikatz utility. The resulting rule has been tested on Windows Server 2012R2 and Windows Server 2016.

The YARA rule developed during the survey is presented below.

Conclusions

The logic of the rules on Windows audit events can be used as hypotheses when conducting Zerologon hunting exercises. Specific patterns in traffic will be useful for developing network IDS signatures. With the YARA rule and the appropriate tools, you can periodically scan the lsass.exe process memory on domain controllers.

Of course, all of the above approaches can be used in tandem, which will make it possible not only to detect cases of Zerologon past and present exploitation, but also to increase the speed of incident classification.

Reports of corporate information systems being encrypted during Zerologon attacks are becoming more and more common. Therefore, it should be remembered that the ability to detect Zerologon exploitation is not enough for protection. Installing security updates and patches from Microsoft will help to significantly reduce the risks.

Zerologon Microsoft Vulnerability Secops Cybersecurity

-- 1

Written by BI.ZONE

Follow

175 Followers

BI.ZONE: an expert in digital risks management. We help organizations around the world to develop their businesses safely in the digital age