tenable®

Decrypt Kerberos/NTLM encrypted data in Wireshark

Clément Notin

Staff Research Engineer

Pass the SALT 2023
Lille, France

Bonjour ! 👋

😈 Security researcher, and pentester at heart

🔍 Current focus on identity security, applied in particular to Microsoft Active Directory and Azure AD

@cnotin

@cnotin@infosec.exchange

https://clement.notin.org

Clément Notin

AGENDA

Introduction

How to decrypt Kerberos encrypted traffic?

And what about NTLM?

Conclusion

Questions

AGENDA

Introduction

How to decrypt Kerberos encrypted traffic?

And what about NTLM?

Conclusion

Questions

Microsoft “Active Directory“, you said?

- 🪟 Microsoft solution
- 👯 Directory of users, groups, and devices: LDAP
- ⚙️ Helps manage the assets, and enforce security rules: GPO
- 🔑 Centralized authentication (i.e. SSO) via Kerberos (with extensions) or NTLM
- 👮 AD servers are called Domain Controllers (DCs) and there are normally several
- ↔️ Uses many MS-RPC (Remote Procedure Call) protocols, called “DCE/RPC” in Wireshark

Active Directory

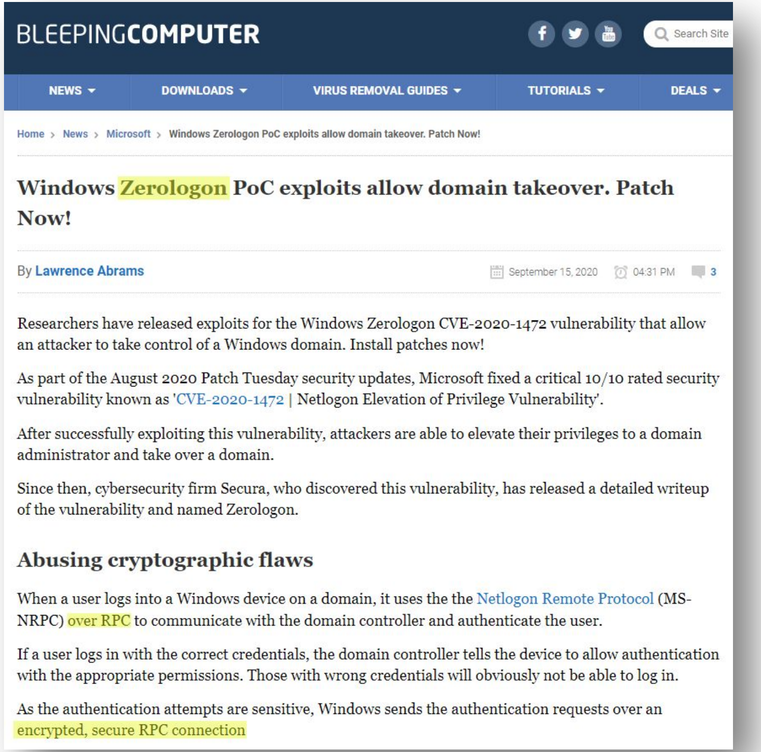
What is the problem this talk will help you solve?

📍 Situation: captured traffic of a Windows box, joined to an Active Directory domain

💪 Can see a lot of traffic: Kerberos, LDAP, SMB, MS-RPC...
with metadata: file names, RPC protocol and function names...

😔 ... but not the payloads: values of the parameters

Why do we need to analyze this RPC traffic?


BLEEPINGCOMPUTER
Search Site
NEWS
DOWNLOADS
VIRUS REMOVAL GUIDES
TUTORIALS
DEALS
Home > News > Microsoft > Windows Zerologon PoC exploits allow domain takeover. Patch Now!

Windows Zerologon PoC exploits allow domain takeover. Patch Now!

By Lawrence Abrams September 15, 2020 04:31 PM 3

Researchers have released exploits for the Windows Zerologon CVE-2020-1472 vulnerability that allow an attacker to take control of a Windows domain. Install patches now!

As part of the August 2020 Patch Tuesday security updates, Microsoft fixed a critical 10/10 rated security vulnerability known as 'CVE-2020-1472 | Netlogon Elevation of Privilege Vulnerability'.

After successfully exploiting this vulnerability, attackers are able to elevate their privileges to a domain administrator and take over a domain.

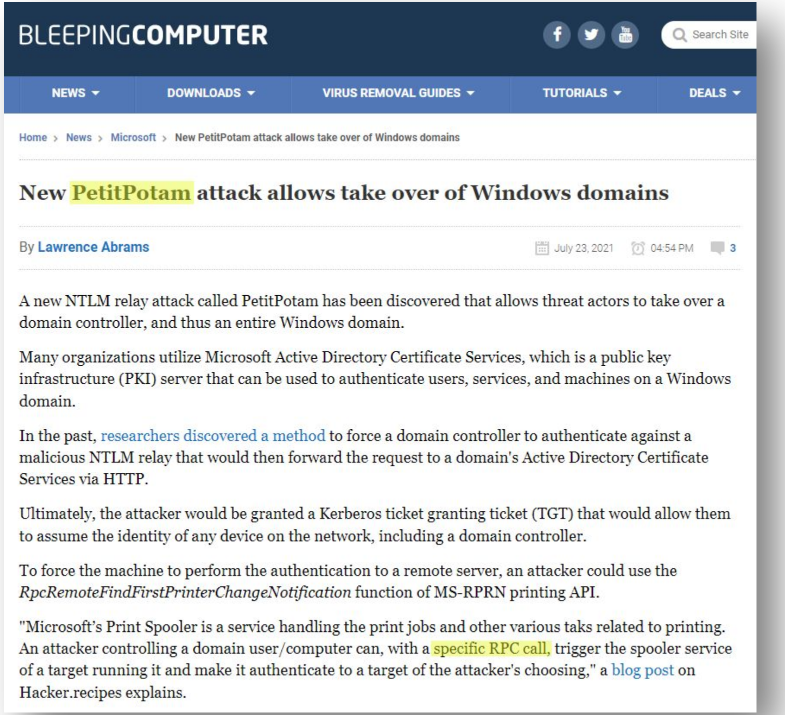
Since then, cybersecurity firm Secura, who discovered this vulnerability, has released a detailed writeup of the vulnerability and named Zerologon.

Abusing cryptographic flaws

When a user logs into a Windows device on a domain, it uses the the Netlogon Remote Protocol (MS-NRPC) over RPC to communicate with the domain controller and authenticate the user.

If a user logs in with the correct credentials, the domain controller tells the device to allow authentication with the appropriate permissions. Those with wrong credentials will obviously not be able to log in.

As the authentication attempts are sensitive, Windows sends the authentication requests over an encrypted, secure RPC connection


BLEEPINGCOMPUTER
Search Site
NEWS
DOWNLOADS
VIRUS REMOVAL GUIDES
TUTORIALS
DEALS
Home > News > Microsoft > New PetitPotam attack allows take over of Windows domains

New PetitPotam attack allows take over of Windows domains

By Lawrence Abrams July 23, 2021 04:54 PM 3

A new NTLM relay attack called PetitPotam has been discovered that allows threat actors to take over a domain controller, and thus an entire Windows domain.

Many organizations utilize Microsoft Active Directory Certificate Services, which is a public key infrastructure (PKI) server that can be used to authenticate users, services, and machines on a Windows domain.

In the past, researchers discovered a method to force a domain controller to authenticate against a malicious NTLM relay that would then forward the request to a domain's Active Directory Certificate Services via HTTP.

Ultimately, the attacker would be granted a Kerberos ticket granting ticket (TGT) that would allow them to assume the identity of any device on the network, including a domain controller.

To force the machine to perform the authentication to a remote server, an attacker could use the *RpcRemoteFindFirstPrinterChangeNotification* function of MS-RPRN printing API.

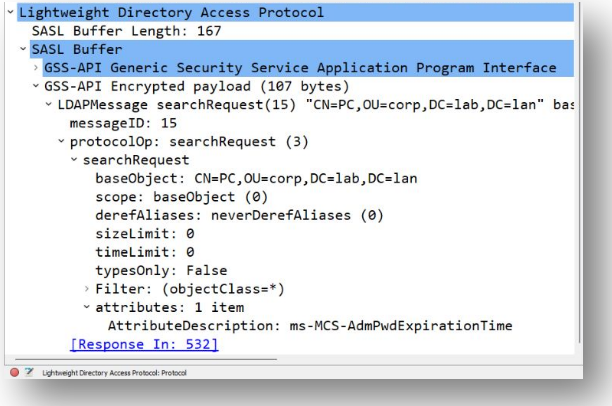
"Microsoft's Print Spooler is a service handling the print jobs and other various taks related to printing. An attacker controlling a domain user/computer can, with a specific RPC call, trigger the spooler service of a target running it and make it authenticate to a target of the attacker's choosing," a blog post on Hacker.recipes explains.

😔 RPC encryption cannot be disabled usually, even in lab environments

Yes we can decrypt it!

🔓 Encrypted layer is decrypted

😌 Underlying dissector can do its work! →



```
⌄ Lightweight Directory Access Protocol
    SASL Buffer Length: 167
  ⌄ SASL Buffer
    › GSS-API Generic Security Service Application Program Interface
    ⌄ GSS-API Encrypted payload (107 bytes)
      ⌄ LDAPMessage searchRequest(15) "CN=PC,OU=corp,DC=lab,DC=lan" bas
          messageID: 15
        ⌄ protocolOp: searchRequest (3)
          ⌄ searchRequest
              baseObject: CN=PC,OU=corp,DC=lab,DC=lan
              scope: baseObject (0)
              derefAliases: neverDerefAliases (0)
              sizeLimit: 0
              timeLimit: 0
              typesOnly: False
            › Filter: (objectClass=*)
            ⌄ attributes: 1 item
                AttributeDescription: ms-MCS-AdmPwdExpirationTime
          [Response In: 532]
```

Lightweight Directory Access Protocol: Protocol

🔭 Similar to the TLS decryption feature

https://wiki.wireshark.org/TLS

😉 I am going to give you a quick overview of how

🤓 Read again the slides, or the blogpost, later to train yourself. Sample PCAPs are provided on the page of this talk on the conference website

AGENDA

Introduction

How to decrypt Kerberos encrypted traffic?

And what about NTLM?

Conclusion

Questions

Road to success

Capture
encrypted traffic

Get
Kerberos keys

Put keys in
keytab file

Give keytab
to Wireshark

Enjoy!

Road to success

Capture encrypted traffic

Get Kerberos keys

Put keys in keytab file

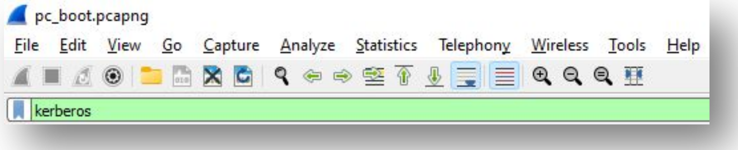
Give keytab to Wireshark

Enjoy!

First look at the capture

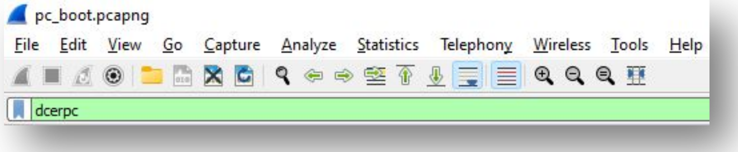
- Open `pc_boot.pcapng` in Wireshark
  - Recorded when the machine was starting to have the most data

- Display filter to see Kerberos and Kerberos-encrypted traffic only:
  - `kerberos`


pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

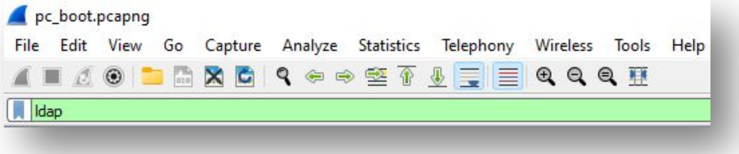
kerberos

- Display filter to see MS-RPC traffic:
  - `dcerpc`


pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dcerpc

- Display filter to see LDAP traffic:
  - `ldap`

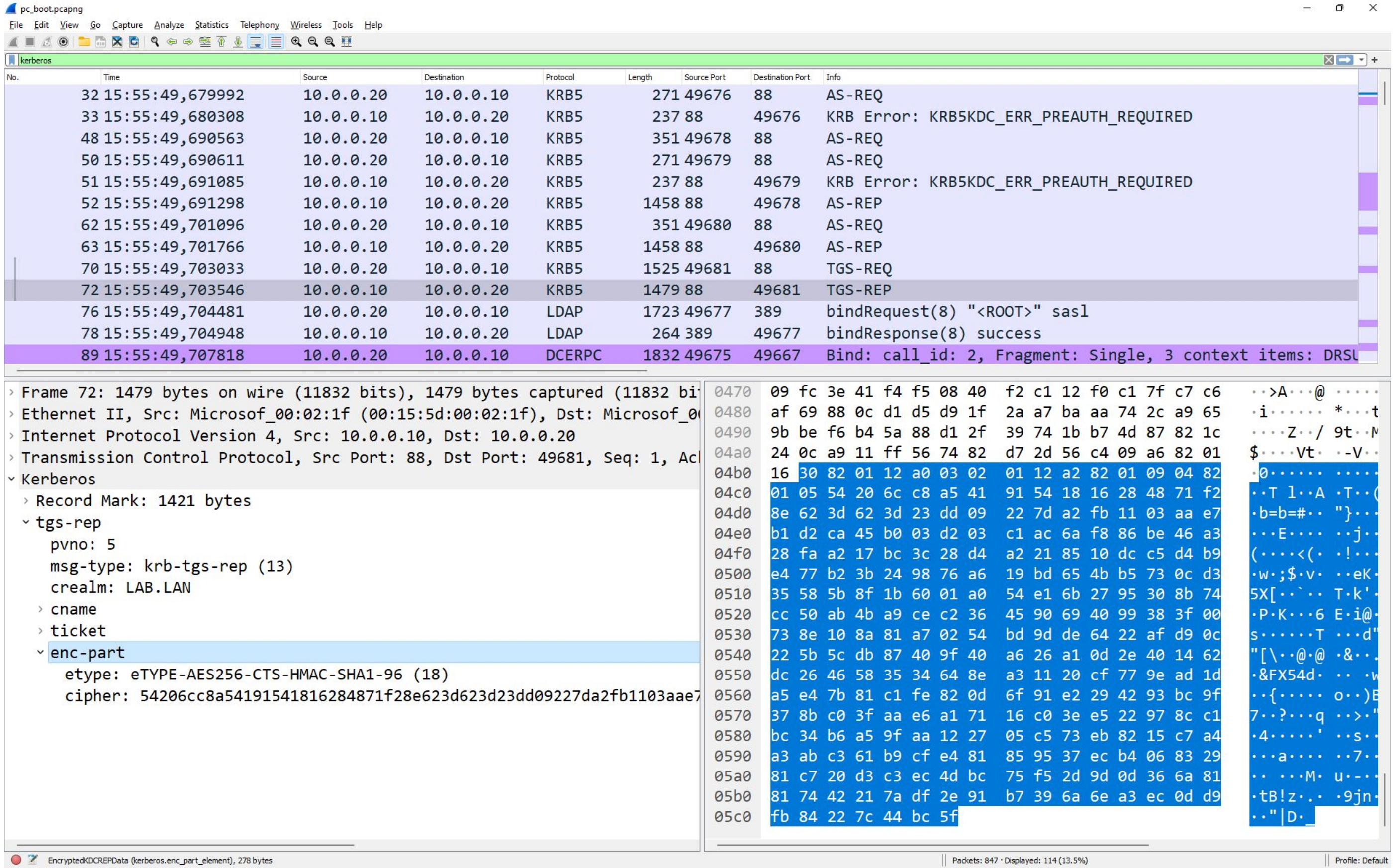

pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ldap

First look at the capture

Kerberos TGS-REP with

🔒 `enc-part`



pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
kerberos

```
No.   Time               Source     Destination Protocol Length Source Port Destination Port Info
   32 15:55:49,679992    10.0.0.20  10.0.0.10   KRB5      271 49676  88     AS-REQ
   33 15:55:49,680308    10.0.0.10  10.0.0.20   KRB5      237 88     49676  KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
   48 15:55:49,690563    10.0.0.20  10.0.0.10   KRB5      351 49678  88     AS-REQ
   50 15:55:49,690611    10.0.0.20  10.0.0.10   KRB5      271 49679  88     AS-REQ
   51 15:55:49,691085    10.0.0.10  10.0.0.20   KRB5      237 88     49679  KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
   52 15:55:49,691298    10.0.0.10  10.0.0.20   KRB5     1458 88     49678  AS-REP
   62 15:55:49,701096    10.0.0.20  10.0.0.10   KRB5      351 49680  88     AS-REQ
   63 15:55:49,701766    10.0.0.10  10.0.0.20   KRB5     1458 88     49680  AS-REP
   70 15:55:49,703033    10.0.0.20  10.0.0.10   KRB5     1525 49681  88     TGS-REQ
   72 15:55:49,703546    10.0.0.10  10.0.0.20   KRB5     1479 88     49681  TGS-REP
   76 15:55:49,704481    10.0.0.20  10.0.0.10   LDAP     1723 49677  389    bindRequest(8) "<ROOT>" sasl
   78 15:55:49,704948    10.0.0.10  10.0.0.20   LDAP      264 389    49677  bindResponse(8) success
   89 15:55:49,707818    10.0.0.20  10.0.0.10   DCERPC   1832 49675  49667  Bind: call_id: 2, Fragment: Single, 3 context items: DRSU
```

```
> Frame 72: 1479 bytes on wire (11832 bits), 1479 bytes captured (11832 bi
> Ethernet II, Src: Microsof_00:02:1f (00:15:5d:00:02:1f), Dst: Microsof_0
> Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.0.20
> Transmission Control Protocol, Src Port: 88, Dst Port: 49681, Seq: 1, Ac
v Kerberos
  > Record Mark: 1421 bytes
  v tgs-rep
      pvno: 5
      msg-type: krb-tgs-rep (13)
      crealm: LAB.LAN
    > cname
    > ticket
    v enc-part
        etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
        cipher: 54206cc8a5419154181628 4871f28e623d623d23dd09227da2fb1103aae7
```

```
0470  09 fc 3e 41 f4 f5 08 40  f2 c1 12 f0 c1 7f c7 c6   ··>A···@ ·····
0480  af 69 88 0c d1 d5 d9 1f  2a a7 ba aa 74 2c a9 65   ·i······ *···t
0490  9b be f6 b4 5a 88 d1 2f  39 74 1b b7 4d 87 82 1c   ····Z··/ 9t··M
04a0  24 0c a9 11 ff 56 74 82  d7 2d 56 c4 09 a6 82 01   $····Vt· ·-V··
04b0  16 30 82 01 12 a0 03 02  01 12 a2 82 01 09 04 82   ·0······ ·····
04c0  01 05 54 20 6c c8 a5 41  91 54 18 16 28 48 71 f2   ··T l··A ·T··(
04d0  8e 62 3d 62 3d 23 dd 09  22 7d a2 fb 11 03 aa e7   ·b=b=#·· "}···
04e0  b1 d2 ca 45 b0 03 d2 03  c1 ac 6a f8 86 be 46 a3   ···E···· ··j··
04f0  28 fa a2 17 bc 3c 28 d4  a2 21 85 10 dc c5 d4 b9   (····<(· ·!···
0500  e4 77 b2 3b 24 98 76 a6  19 bd 65 4b b5 73 0c d3   ·w·;$·v· ··eK·
0510  35 58 5b 8f 1b 60 01 a0  54 e1 6b 27 95 30 8b 74   5X[··`·· T·k'·
0520  cc 50 ab 4b a9 ce c2 36  45 90 69 40 99 38 3f 00   ·P·K···6 E·i@·
0530  73 8e 10 8a 81 a7 02 54  bd 9d de 64 22 af d9 0c   s······T ···d"
0540  22 5b 5c db 87 40 9f 40  a6 26 a1 0d 2e 40 14 62   "[\··@·@ ·&···
0550  dc 26 46 58 35 34 64 8e  a3 11 20 cf 77 9e ad 1d   ·&FX54d· ·· ·w
0560  a5 e4 7b 81 c1 fe 82 0d  6f 91 e2 29 42 93 bc 9f   ··{····· o··)B
0570  37 8b c0 3f aa e6 a1 71  16 c0 3e e5 22 97 8c c1   7··?···q ··>·"
0580  bc 34 b6 a5 9f aa 12 27  05 c5 73 eb 82 15 c7 a4   ·4·····' ··s··
0590  a3 ab c3 61 b9 cf e4 81  85 95 37 ec b4 06 83 29   ···a···· ··7··
05a0  81 c7 20 d3 c3 ec 4d bc  75 f5 2d 9d 0d 36 6a 81   ·· ···M· u·-··
05b0  81 74 42 21 7a df 2e 91  b7 39 6a 6e a3 ec 0d d9   ·tB!z·.· ·9jn·
05c0  fb 84 22 7c 44 bc 5f                               ··"|D·_
```

EncryptedKDCREPData (kerberos.enc_part_element), 278 bytes

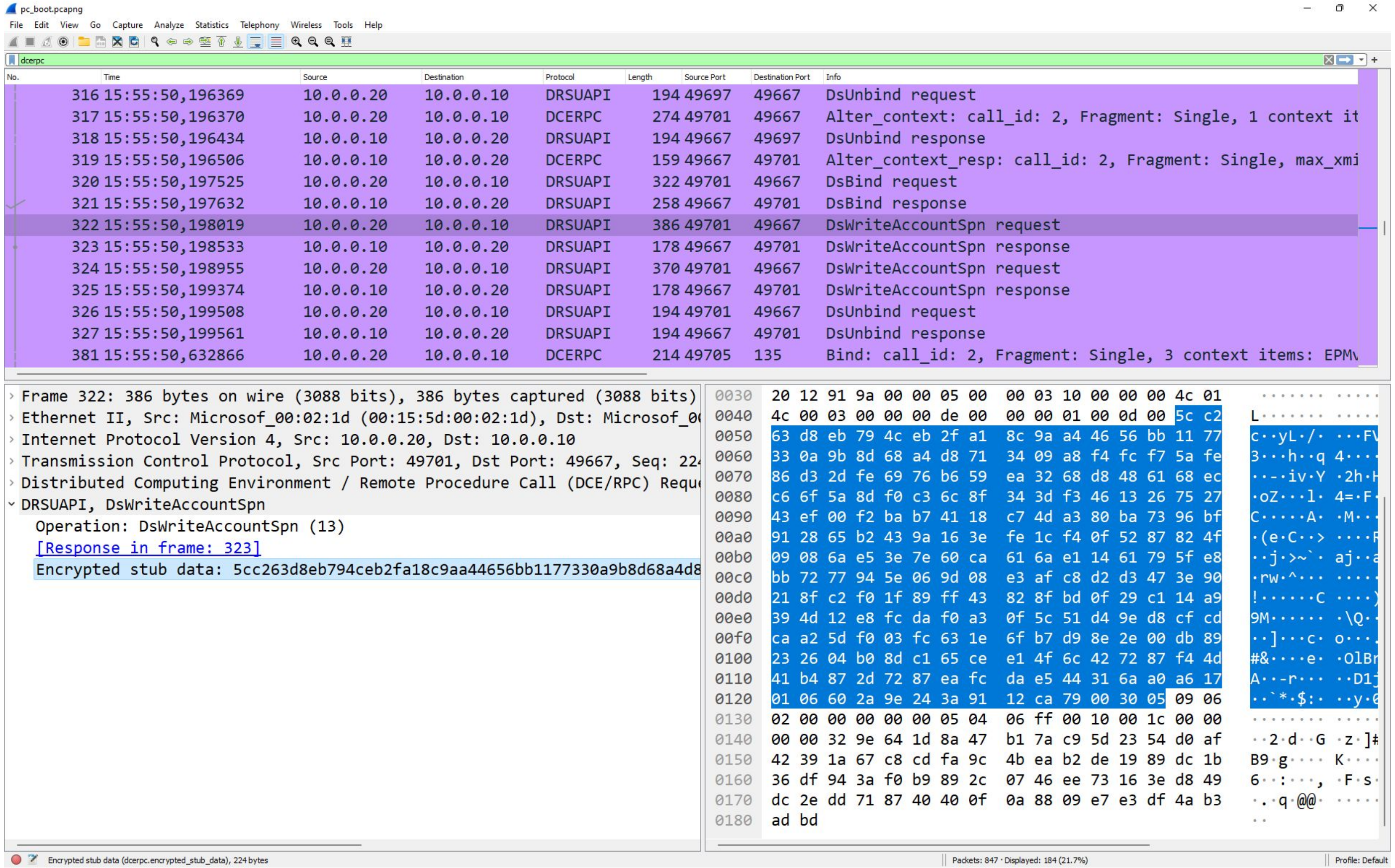
Packets: 847 · Displayed: 114 (13.5%)

Profile: Default

First look at the capture

DRSUAPI
`DsWriteAccountSPN`
`with`

🔒 `encrypted stub data`


pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dcerpc

```
No.  Time               Source     Destination Protocol Length Source Port Destination Port Info
316 15:55:50,196369  10.0.0.20  10.0.0.10  DRSUAPI   194 49697  49667  DsUnbind request
317 15:55:50,196370  10.0.0.20  10.0.0.10  DCERPC    274 49701  49667  Alter_context: call_id: 2, Fragment: Single, 1 context it
318 15:55:50,196434  10.0.0.10  10.0.0.20  DRSUAPI   194 49667  49697  DsUnbind response
319 15:55:50,196506  10.0.0.10  10.0.0.20  DCERPC    159 49667  49701  Alter_context_resp: call_id: 2, Fragment: Single, max_xmi
320 15:55:50,197525  10.0.0.20  10.0.0.10  DRSUAPI   322 49701  49667  DsBind request
321 15:55:50,197632  10.0.0.10  10.0.0.20  DRSUAPI   258 49667  49701  DsBind response
322 15:55:50,198019  10.0.0.20  10.0.0.10  DRSUAPI   386 49701  49667  DsWriteAccountSpn request
323 15:55:50,198533  10.0.0.10  10.0.0.20  DRSUAPI   178 49667  49701  DsWriteAccountSpn response
324 15:55:50,198955  10.0.0.20  10.0.0.10  DRSUAPI   370 49701  49667  DsWriteAccountSpn request
325 15:55:50,199374  10.0.0.10  10.0.0.20  DRSUAPI   178 49667  49701  DsWriteAccountSpn response
326 15:55:50,199508  10.0.0.20  10.0.0.10  DRSUAPI   194 49701  49667  DsUnbind request
327 15:55:50,199561  10.0.0.10  10.0.0.20  DRSUAPI   194 49667  49701  DsUnbind response
381 15:55:50,632866  10.0.0.20  10.0.0.10  DCERPC    214 49705  135    Bind: call_id: 2, Fragment: Single, 3 context items: EPMv
```

```
> Frame 322: 386 bytes on wire (3088 bits), 386 bytes captured (3088 bits)
> Ethernet II, Src: Microsof_00:02:1d (00:15:5d:00:02:1d), Dst: Microsof_0
> Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.0.10
> Transmission Control Protocol, Src Port: 49701, Dst Port: 49667, Seq: 22
> Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Requ
v DRSUAPI, DsWriteAccountSpn
    Operation: DsWriteAccountSpn (13)
    [Response in frame: 323]
    Encrypted stub data: 5cc263d8eb794ceb2fa18c9aa44656bb1177330a9b8d68a4d8
```

```
0030  20 12 91 9a 00 00 05 00  00 03 10 00 00 00 4c 01   ········ ·····
0040  4c 00 03 00 00 00 de 00  00 00 01 00 0d 00 5c c2   L······· ·····
0050  63 d8 eb 79 4c eb 2f a1  8c 9a a4 46 56 bb 11 77   c··yL·/· ···FV
0060  33 0a 9b 8d 68 a4 d8 71  34 09 a8 f4 fc f7 5a fe   3···h··q 4····
0070  86 d3 2d fe 69 76 b6 59  ea 32 68 d8 48 61 68 ec   ··-·iv·Y ·2h·H
0080  c6 6f 5a 8d f0 c3 6c 8f  34 3d f3 46 13 26 75 27   ·oZ···l· 4=·F
0090  43 ef 00 f2 ba b7 41 18  c7 4d a3 80 ba 73 96 bf   C·····A· ·M··
00a0  91 28 65 b2 43 9a 16 3e  fe 1c f4 0f 52 87 82 4f   ·(e·C··> ····R
00b0  09 08 6a e5 3e 7e 60 ca  61 6a e1 14 61 79 5f e8   ··j·>~`· aj··a
00c0  bb 72 77 94 5e 06 9d 08  e3 af c8 d2 d3 47 3e 90   ·rw·^··· ·····
00d0  21 8f c2 f0 1f 89 ff 43  82 8f bd 0f 29 c1 14 a9   !······C ····)
00e0  39 4d 12 e8 fc da f0 a3  0f 5c 51 d4 9e d8 cf cd   9M······ ·\Q··
00f0  ca a2 5d f0 03 fc 63 1e  6f b7 d9 8e 2e 00 db 89   ··]···c· o····
0100  23 26 04 b0 8d c1 65 ce  e1 4f 6c 42 72 87 f4 4d   #&····e· ·OlBr
0110  41 b4 87 2d 72 87 ea fc  da e5 44 31 6a a0 a6 17   A··-r··· ··D1j
0120  01 06 60 2a 9e 24 3a 91  12 ca 79 00 30 05 09 06   ··`*·$:· ··y·
0130  02 00 00 00 00 00 05 04  06 ff 00 10 00 1c 00 00   ········ ·····
0140  00 00 32 9e 64 1d 8a 47  b1 7a c9 5d 23 54 d0 af   ··2·d··G ·z·]#
0150  42 39 1a 67 c8 cd fa 9c  4b ea b2 de 19 89 dc 1b   B9·g···· K····
0160  36 df 94 3a f0 b9 89 2c  07 46 ee 73 16 3e d8 49   6··:···, ·F·s·
0170  dc 2e dd 71 87 40 40 0f  0a 88 09 e7 e3 df 4a b3   ·.·q·@@· ·····
0180  ad bd                                              ··
```

Encrypted stub data (dcerpc.encrypted_stub_data), 224 bytes

Packets: 847 · Displayed: 184 (21.7%)

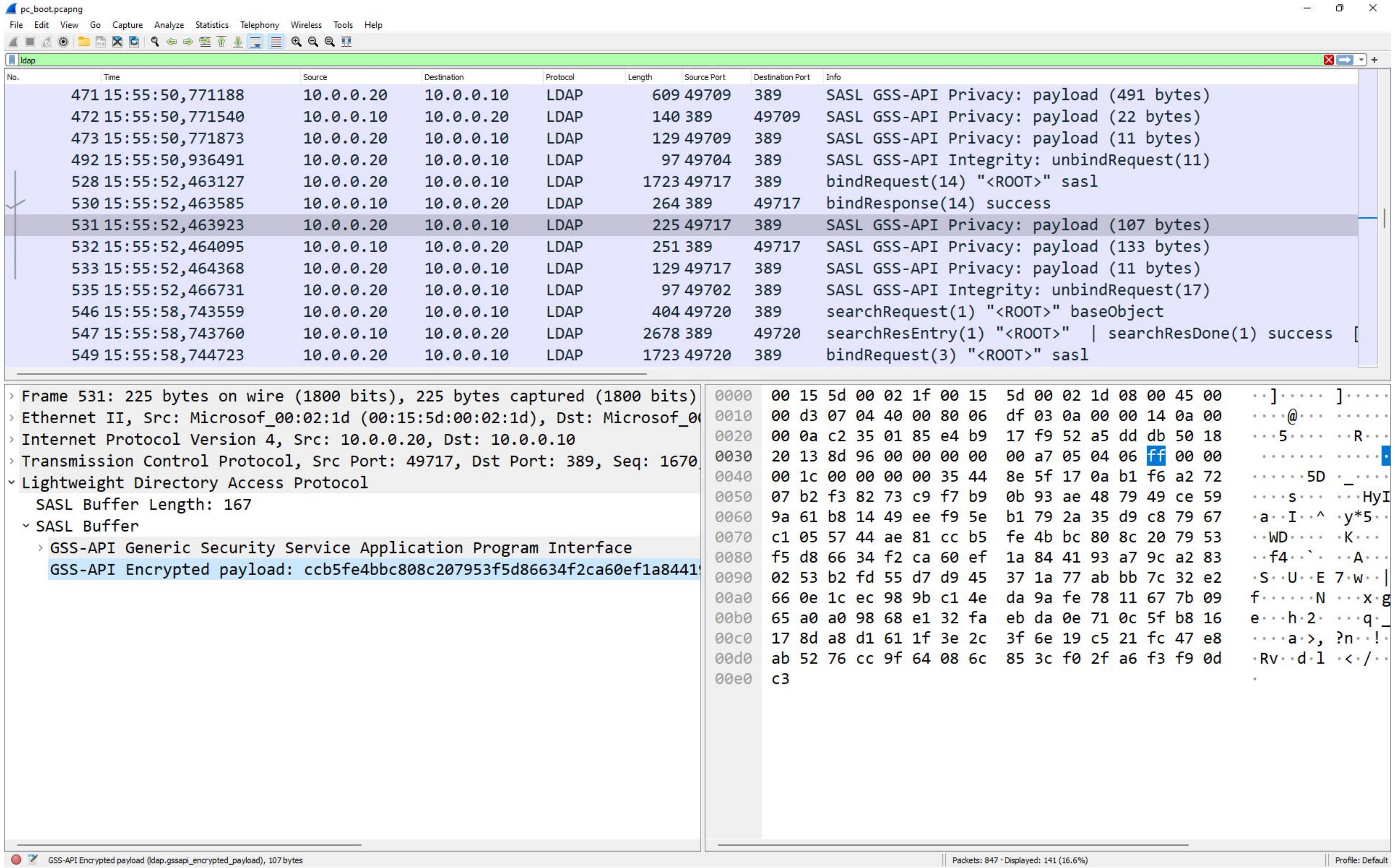
Profile: Default

First look at the capture

LDAP with

🔒 GSS-API

Encrypted payload



pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ldap

No.	Time	Source	Destination	Protocol	Length	Source Port	Destination Port	Info
471	15:55:50,771188	10.0.0.20	10.0.0.10	LDAP	609	49709	389	SASL GSS-API Privacy: payload (491 bytes)
472	15:55:50,771540	10.0.0.10	10.0.0.20	LDAP	140	389	49709	SASL GSS-API Privacy: payload (22 bytes)
473	15:55:50,771873	10.0.0.20	10.0.0.10	LDAP	129	49709	389	SASL GSS-API Privacy: payload (11 bytes)
492	15:55:50,936491	10.0.0.20	10.0.0.10	LDAP	97	49704	389	SASL GSS-API Integrity: unbindRequest(11)
528	15:55:52,463127	10.0.0.20	10.0.0.10	LDAP	1723	49717	389	bindRequest(14) "<ROOT>" sasl
530	15:55:52,463585	10.0.0.10	10.0.0.20	LDAP	264	389	49717	bindResponse(14) success
531	15:55:52,463923	10.0.0.20	10.0.0.10	LDAP	225	49717	389	SASL GSS-API Privacy: payload (107 bytes)
532	15:55:52,464095	10.0.0.10	10.0.0.20	LDAP	251	389	49717	SASL GSS-API Privacy: payload (133 bytes)
533	15:55:52,464368	10.0.0.20	10.0.0.10	LDAP	129	49717	389	SASL GSS-API Privacy: payload (11 bytes)
535	15:55:52,466731	10.0.0.20	10.0.0.10	LDAP	97	49702	389	SASL GSS-API Integrity: unbindRequest(17)
546	15:55:58,743559	10.0.0.20	10.0.0.10	LDAP	404	49720	389	searchRequest(1) "<ROOT>" baseObject
547	15:55:58,743760	10.0.0.10	10.0.0.20	LDAP	2678	389	49720	searchResEntry(1) "<ROOT>" | searchResDone(1) success [
549	15:55:58,744723	10.0.0.20	10.0.0.10	LDAP	1723	49720	389	bindRequest(3) "<ROOT>" sasl

```
> Frame 531: 225 bytes on wire (1800 bits), 225 bytes captured (1800 bits)
> Ethernet II, Src: Microsof_00:02:1d (00:15:5d:00:02:1d), Dst: Microsof_0(
> Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.0.10
> Transmission Control Protocol, Src Port: 49717, Dst Port: 389, Seq: 1670
v Lightweight Directory Access Protocol
    SASL Buffer Length: 167
  v SASL Buffer
    > GSS-API Generic Security Service Application Program Interface
      GSS-API Encrypted payload: ccb5fe4bbc808c207953f5d86634f2ca60ef1a8441
```

```
0000  00 15 5d 00 02 1f 00 15  5d 00 02 1d 08 00 45 00   ··]····· ]·····
0010  00 d3 07 04 40 00 80 06  df 03 0a 00 00 14 0a 00   ····@··· ······
0020  00 0a c2 35 01 85 e4 b9  17 f9 52 a5 dd db 50 18   ···5···· ··R···
0030  20 13 8d 96 00 00 00 00  00 a7 05 04 06 ff 00 00   ········ ······
0040  00 1c 00 00 00 00 35 44  8e 5f 17 0a b1 f6 a2 72   ······5D ·_····
0050  07 b2 f3 82 73 c9 f7 b9  0b 93 ae 48 79 49 ce 59   ····s··· ···HyI
0060  9a 61 b8 14 49 ee f9 5e  b1 79 2a 35 d9 c8 79 67   ·a··I··^ ·y*5··
0070  c1 05 57 44 ae 81 cc b5  fe 4b bc 80 8c 20 79 53   ··WD···· ·K···
0080  f5 d8 66 34 f2 ca 60 ef  1a 84 41 93 a7 9c a2 83   ··f4··`· ··A···
0090  02 53 b2 fd 55 d7 d9 45  37 1a 77 ab bb 7c 32 e2   ·S··U··E 7·w··|
00a0  66 0e 1c ec 98 9b c1 4e  da 9a fe 78 11 67 7b 09   f······N ···x·g
00b0  65 a0 a0 98 68 e1 32 fa  eb da 0e 71 0c 5f b8 16   e···h·2· ···q·_
00c0  17 8d a8 d1 61 1f 3e 2c  3f 6e 19 c5 21 fc 47 e8   ····a·>, ?n··!·
00d0  ab 52 76 cc 9f 64 08 6c  85 3c f0 2f a6 f3 f9 0d   ·Rv··d·l ·<·/··
00e0  c3                                                 ·
```

GSS-API Encrypted payload (ldap.gssapi_encrypted_payload), 107 bytes | Packets: 847 · Displayed: 141 (16.6%) | Profile: Default

Road to success

Capture encrypted traffic

Get Kerberos keys

Put keys in keytab file

Give keytab to Wireshark

Enjoy!

Is it magic? How can it work?

🔑 We need keys ...

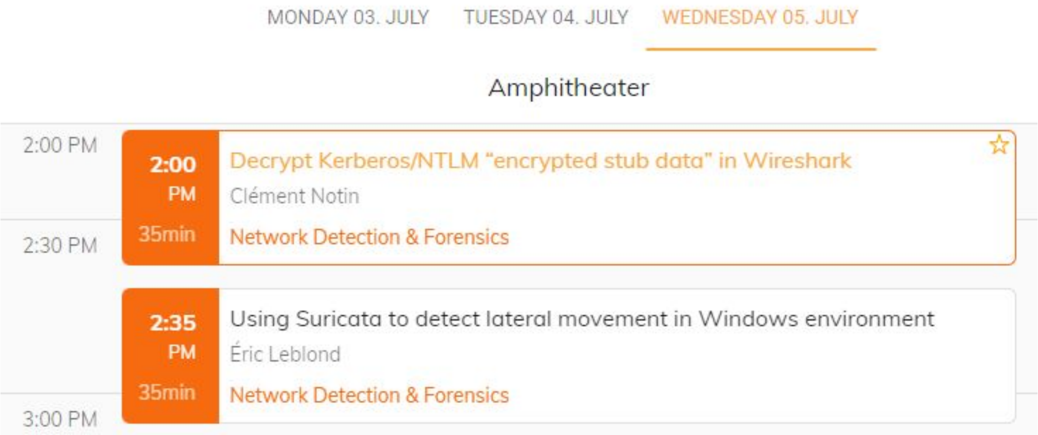
🗝️ ... Kerberos keys!

🔑 🗝️ Different keys

Kerberos 101

🧑‍🎓Stay focused!

Not easy… but it’s worth it since it is useful for the next talk too!



MONDAY 03. JULY TUESDAY 04. JULY WEDNESDAY 05. JULY

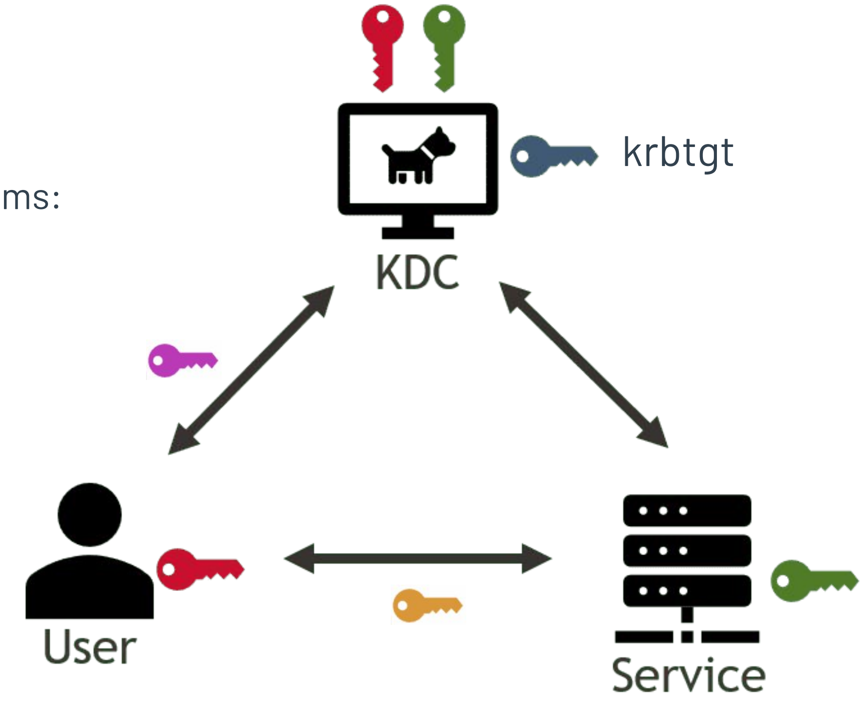
Amphitheater

2:00 PM	2:00 PM 35min	☆ Decrypt Kerberos/NTLM “encrypted stub data” in Wireshark Clément Notin Network Detection & Forensics
2:30 PM	2:35 PM 35min	Using Suricata to detect lateral movement in Windows environment Éric Leblond Network Detection & Forensics
3:00 PM		

Kerberos 101

- Long-term keys
  - Derived from passwords
  - Different keys for different algorithms: DES, RC4, AES128, AES256...
  - 🙏 Wireshark needs

- Session keys
  - Random
  - Short-lived
  - Shared encrypted by long-term keys
  - Used to encrypt the following application traffic



Source: https://www.netspi.com/blog/technical/network-penetration-testing/cve-2020-17049-kerberos-bronze-bit-theory/

How to get the keys?

- Several methods to get the long-term key(s) described on Wireshark wiki: https://wiki.wireshark.org/Kerberos
  - Generate keys from cleartext password, using different tools
  - Get keys from the domain controller database (`ntds.dit`)
  - ...

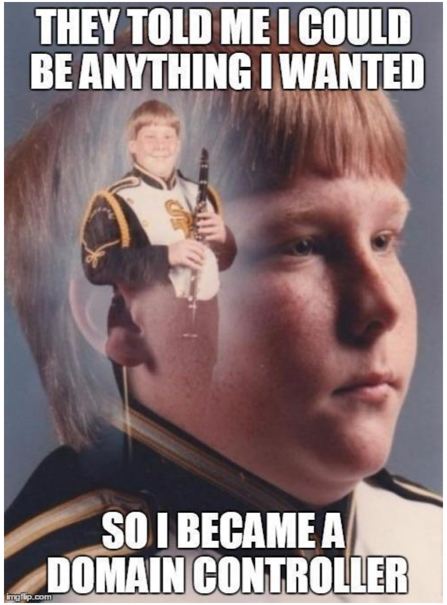
- Or, request the key(s) from a live domain controller: DCSync method
  - Easiest and fastest method!

DCSync to get the keys

- Domain Controllers (DCs) have a synchronization protocol
- If we are Domain Admins, or spoof the identity of a DC, we can request secret attributes containing NTLM hashes & Kerberos keys

- Tool: mimikatz ➡️ https://github.com/gentilkiwi/mimikatz
⚠️ Hack tool: not a virus but enough to trigger your antivirus. Use it at your own discretion and preferably in a lab.

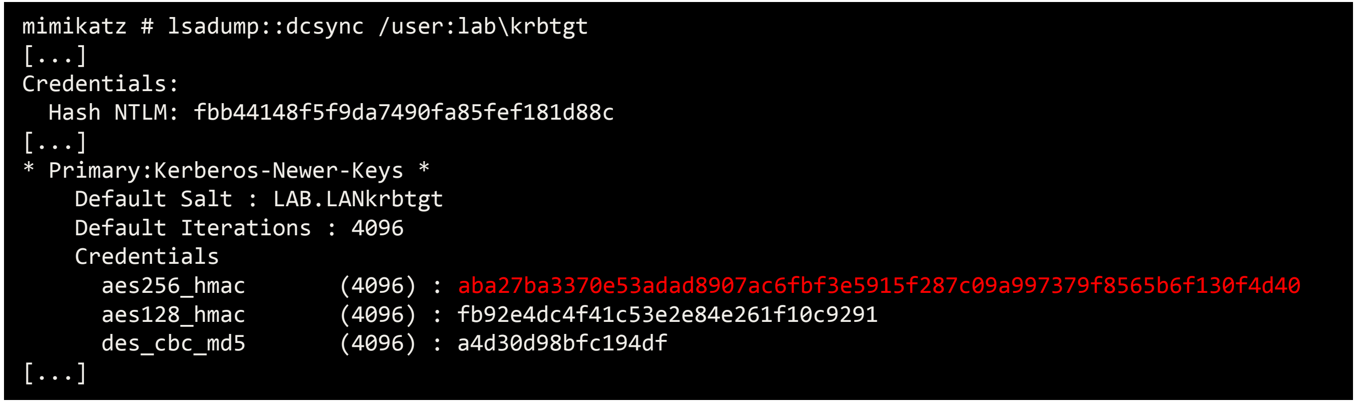
- Method described in ➡️ https://adsecurity.org/?p=1729


THEY TOLD ME I COULD BE ANYTHING I WANTED
SO I BECAME A DOMAIN CONTROLLER
imgflip.com

Source: https://twitter.com/gentilkiwi/status/1000032166884061185

DCSync to get the keys

Use mimikatz to get the AES256 keys of the master “`krbtgt`” account:



```
mimikatz # lsadump::dcsync /user:lab\krbtgt
[...]
Credentials:
  Hash NTLM: fbb44148f5f9da7490fa85fef181d88c
[...]
* Primary:Kerberos-Newer-Keys *
    Default Salt : LAB.LANkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : aba27ba3370e53adad8907ac6fbf3e5915f287c09a997379f8565b6f130f4d40
      aes128_hmac       (4096) : fb92e4dc4f41c53e2e84e261f10c9291
      des_cbc_md5       (4096) : a4d30d98bfc194df
[...]
```

`mimikatz dcsync krbtgt.txt`

Road to success

Capture encrypted traffic

Get Kerberos keys

Put keys in keytab file

Give keytab to Wireshark

Enjoy!

How to give the keys to Wireshark?

- 📄 keytab file
- Used a lot for Kerberos in Linux world
- Contains usernames and long-term Kerberos keys

How to fill the keytab?

- Several methods are available (e.g. `ktutil` on Linux)
- 🐍 I like to use this Python script
  ➡️ https://github.com/dirkjanm/forest-trust-tools/blob/master/keytab.py
  - Dependency on the `impacket` library:
    ⚠️ Hack tool: not a virus but enough to trigger your antivirus. Use it at your own discretion and preferably in a lab.

- User name and domain name do not seem to matter
- Only the `krbtgt` key is necessary usually, but we can provide as many keys as we have (especially for analysis of the Kerberos protocol itself)
- Ensure to select the right algorithm ID

Write keys to keytab

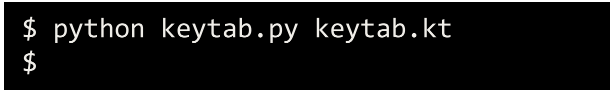
- Modify `keytab.py` around line 112 to add the AES 256 key (`keytype=18`)

```
[…]
# Add your own keys here!
# Keys are tuples in the form (keytype, 'hexencodedkey')
# Common keytypes for Windows:
# 23: RC4
# 18: AES-256
# 17: AES-128
# Wireshark takes any number of keys in the keytab, so feel free to add
# krbtgt keys, service keys, trust keys etc
keys = [
    (23, 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'),
    (18, 'aba27ba3370e53adad8907ac6fbf3e5915f287c09a997379f8565b6f130f4d40'), # krbtgt
    (17, 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'),
    (18, 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'),
    (23, 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa')
]
[…]
```

`keytab.py`

Write keys to keytab

- Run the script
  - In case of error due to impacket dependency: install impacket ➡️ https://github.com/SecureAuthCorp/impacket#quick-start
  - `python3 -m pip install impacket`



```
$ python keytab.py keytab.kt
$
```

Road to success

Capture encrypted traffic

Get Kerberos keys

Put keys in keytab file

Give keytab to Wireshark

Enjoy!

Provide the keytab to Wireshark

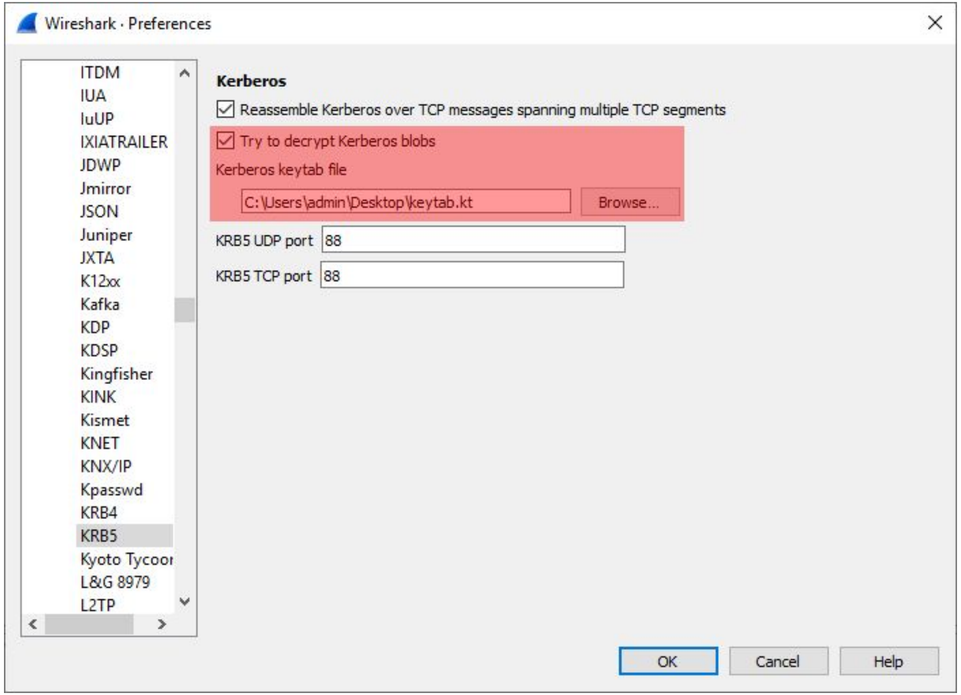
⚙️ Open Preferences

- ➡️ Protocols
- ➡️ KRB5 (Kerberos v5)

✅ Check

“Try to decrypt Kerberos blobs”

🔍 Browse to the location of the keytab file



Wireshark · Preferences

Kerberos

☑ Reassemble Kerberos over TCP messages spanning multiple TCP segments

☑ Try to decrypt Kerberos blobs

Kerberos keytab file: C:\Users\admin\Desktop\keytab.kt

KRB5 UDP port: 88

KRB5 TCP port: 88

♻️ If you modify the keytab (e.g. to add keys), and want to see changes:
➡️ easiest is to restart Wireshark

Is it working?

- Blue 🟦 = 🔓 : decryption successful
  - Display filter: `kerberos.decrypted_keytype`

- Yellow 🟨 = 🔒 : decryption failed
  - Display filter: `kerberos.missing_keytype`
  - Likely because of missing key, or its value for the selected algorithm was not provided

Road to success

Capture encrypted traffic

Get Kerberos keys

Put keys in keytab file

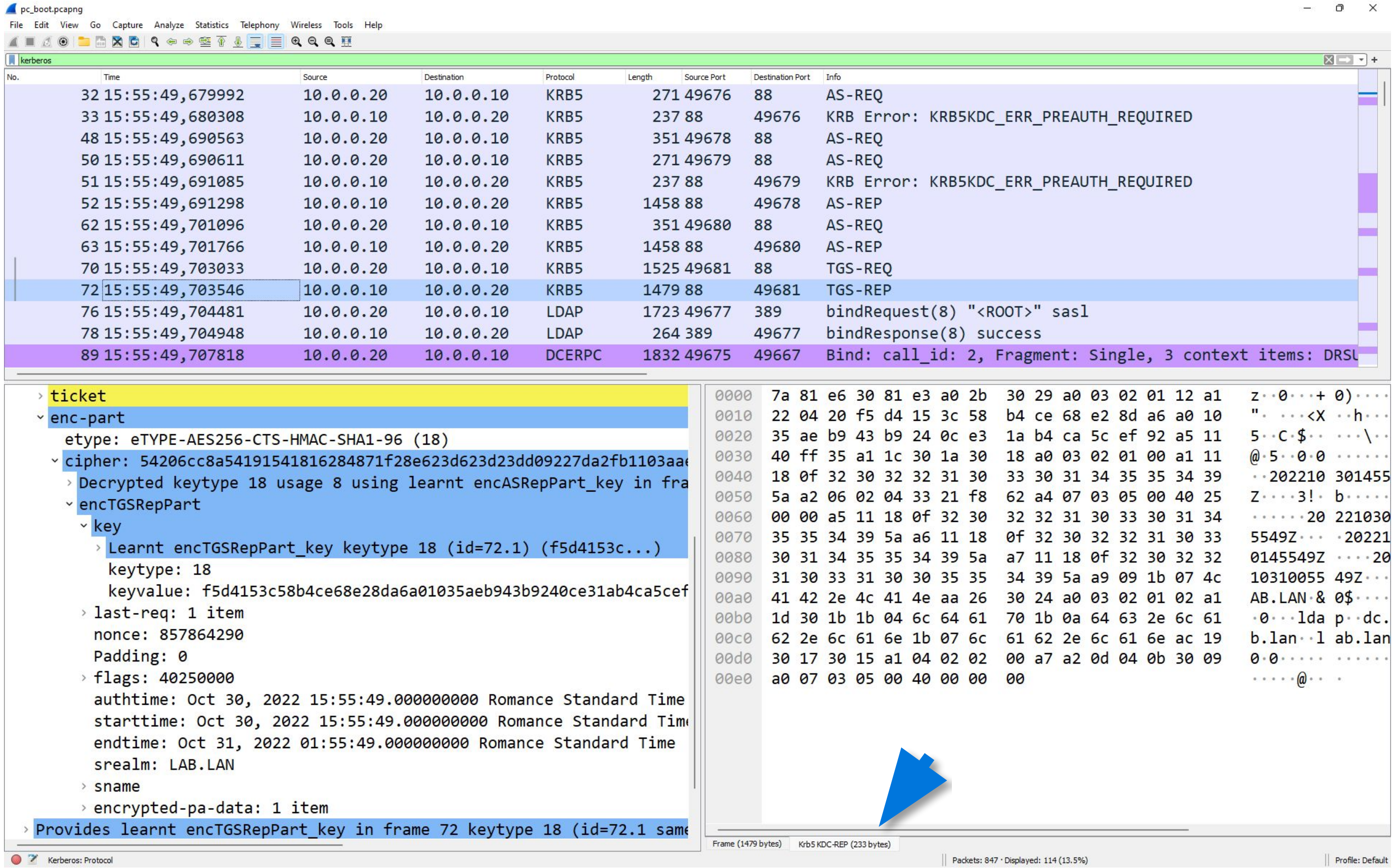
Give keytab to Wireshark

Enjoy!

New look at decrypted capture

Kerberos TGS-REP with

🔓 enc-part



```
kerberos
```

```
No.  Time                Source      Destination  Protocol  Length  Source Port  Destination Port  Info
  32 15:55:49,679992     10.0.0.20   10.0.0.10    KRB5       271    49676        88                AS-REQ
  33 15:55:49,680308     10.0.0.10   10.0.0.20    KRB5       237    88           49676             KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
  48 15:55:49,690563     10.0.0.20   10.0.0.10    KRB5       351    49678        88                AS-REQ
  50 15:55:49,690611     10.0.0.20   10.0.0.10    KRB5       271    49679        88                AS-REQ
  51 15:55:49,691085     10.0.0.10   10.0.0.20    KRB5       237    88           49679             KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
  52 15:55:49,691298     10.0.0.10   10.0.0.20    KRB5      1458    88           49678             AS-REP
  62 15:55:49,701096     10.0.0.20   10.0.0.10    KRB5       351    49680        88                AS-REQ
  63 15:55:49,701766     10.0.0.10   10.0.0.20    KRB5      1458    88           49680             AS-REP
  70 15:55:49,703033     10.0.0.20   10.0.0.10    KRB5      1525    49681        88                TGS-REQ
  72 15:55:49,703546     10.0.0.10   10.0.0.20    KRB5      1479    88           49681             TGS-REP
  76 15:55:49,704481     10.0.0.20   10.0.0.10    LDAP      1723    49677        389               bindRequest(8) "<ROOT>" sasl
  78 15:55:49,704948     10.0.0.10   10.0.0.20    LDAP       264    389          49677             bindResponse(8) success
  89 15:55:49,707818     10.0.0.20   10.0.0.10    DCERPC    1832    49675        49667             Bind: call_id: 2, Fragment: Single, 3 context items: DRSL
```

```
  > ticket
  v enc-part
      etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
    v cipher: 54206cc8a54191541816284871f28e623d623d23dd09227da2fb1103aae
      > Decrypted keytype 18 usage 8 using learnt encASRepPart_key in fra
      v encTGSRepPart
        v key
          > Learnt encTGSRepPart_key keytype 18 (id=72.1) (f5d4153c...)
            keytype: 18
            keyvalue: f5d4153c58b4ce68e28da6a01035aeb943b9240ce31ab4ca5cef
        > last-req: 1 item
          nonce: 857864290
          Padding: 0
        > flags: 40250000
          authtime: Oct 30, 2022 15:55:49.000000000 Romance Standard Time
          starttime: Oct 30, 2022 15:55:49.000000000 Romance Standard Time
          endtime: Oct 31, 2022 01:55:49.000000000 Romance Standard Time
          srealm: LAB.LAN
        > sname
        > encrypted-pa-data: 1 item
> Provides learnt encTGSRepPart_key in frame 72 keytype 18 (id=72.1 same
```

```
0000  7a 81 e6 30 81 e3 a0 2b  30 29 a0 03 02 01 12 a1   z··0···+ 0)····
0010  22 04 20 f5 d4 15 3c 58  b4 ce 68 e2 8d a6 a0 10   "· ···<X ··h···
0020  35 ae b9 43 b9 24 0c e3  1a b4 ca 5c ef 92 a5 11   5··C·$·· ···\··
0030  40 ff 35 a1 1c 30 1a 30  18 a0 03 02 01 00 a1 11   @·5··0·0 ·······
0040  18 0f 32 30 32 32 31 30  33 30 31 34 35 35 34 39   ··202210 301455
0050  5a a2 06 02 04 33 21 f8  62 a4 07 03 05 00 40 25   Z····3!· b·····
0060  00 00 a5 11 18 0f 32 30  32 32 31 30 33 30 31 34   ······20 221030
0070  35 35 34 39 5a a6 11 18  0f 32 30 32 32 31 30 33   5549Z··· ·20221
0080  30 31 34 35 35 34 39 5a  a7 11 18 0f 32 30 32 32   0145549Z ····20
0090  31 30 33 31 30 30 35 35  34 39 5a a9 09 1b 07 4c   10310055 49Z···
00a0  41 42 2e 4c 41 4e aa 26  30 24 a0 03 02 01 02 a1   AB.LAN·& 0$····
00b0  1d 30 1b 1b 04 6c 64 61  70 1b 0a 64 63 2e 6c 61   ·0···lda p··dc.
00c0  62 2e 6c 61 6e 1b 07 6c  61 62 2e 6c 61 6e ac 19   b.lan··l ab.lan
00d0  30 17 30 15 a1 04 02 02  00 a7 a2 0d 04 0b 30 09   0·0····· ·······
00e0  a0 07 03 05 00 40 00 00  00                        ·····@·· ·
```

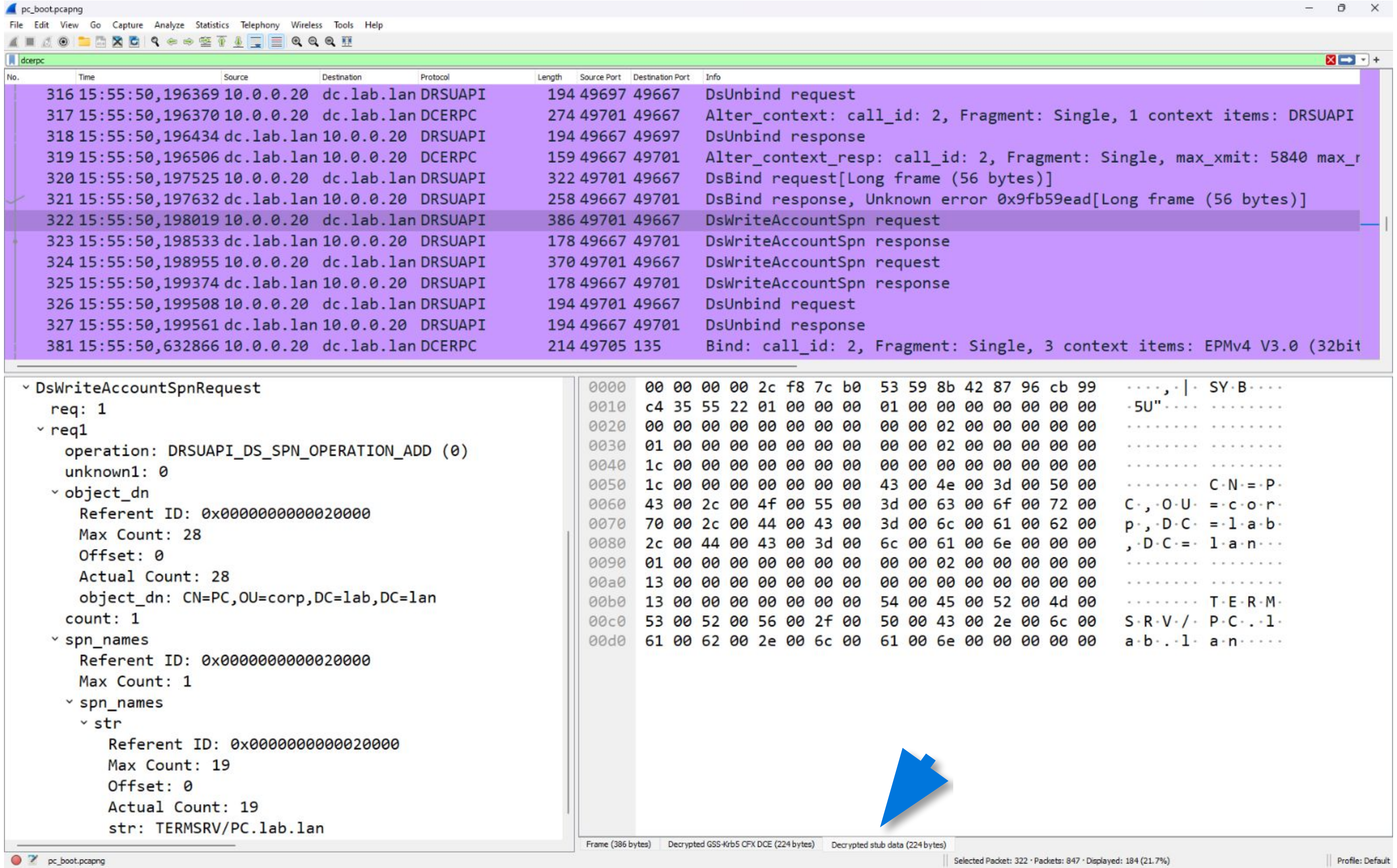
Frame (1479 bytes) | Krb5 KDC-REP (233 bytes)

Kerberos: Protocol | Packets: 847 · Displayed: 114 (13.5%) | Profile: Default

New look at decrypted capture

DRSUAPI
`DsWriteAccountSPN`
`with`

🔓 `decrypted stub data`


pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dcerpc

```
No.  Time             Source      Destination Protocol Length Source Port Destination Port Info
316 15:55:50,196369 10.0.0.20   dc.lab.lan DRSUAPI   194 49697 49667  DsUnbind request
317 15:55:50,196370 10.0.0.20   dc.lab.lan DCERPC    274 49701 49667  Alter_context: call_id: 2, Fragment: Single, 1 context items: DRSUAPI
318 15:55:50,196434 dc.lab.lan 10.0.0.20   DRSUAPI   194 49667 49697  DsUnbind response
319 15:55:50,196506 dc.lab.lan 10.0.0.20   DCERPC    159 49667 49701  Alter_context_resp: call_id: 2, Fragment: Single, max_xmit: 5840 max_r
320 15:55:50,197525 10.0.0.20   dc.lab.lan DRSUAPI   322 49701 49667  DsBind request[Long frame (56 bytes)]
321 15:55:50,197632 dc.lab.lan 10.0.0.20   DRSUAPI   258 49667 49701  DsBind response, Unknown error 0x9fb59ead[Long frame (56 bytes)]
322 15:55:50,198019 10.0.0.20   dc.lab.lan DRSUAPI   386 49701 49667  DsWriteAccountSpn request
323 15:55:50,198533 dc.lab.lan 10.0.0.20   DRSUAPI   178 49667 49701  DsWriteAccountSpn response
324 15:55:50,198955 10.0.0.20   dc.lab.lan DRSUAPI   370 49701 49667  DsWriteAccountSpn request
325 15:55:50,199374 dc.lab.lan 10.0.0.20   DRSUAPI   178 49667 49701  DsWriteAccountSpn response
326 15:55:50,199508 10.0.0.20   dc.lab.lan DRSUAPI   194 49701 49667  DsUnbind request
327 15:55:50,199561 dc.lab.lan 10.0.0.20   DRSUAPI   194 49667 49701  DsUnbind response
381 15:55:50,632866 10.0.0.20   dc.lab.lan DCERPC    214 49705 135    Bind: call_id: 2, Fragment: Single, 3 context items: EPMv4 V3.0 (32bit
```

```
˅ DsWriteAccountSpnRequest
    req: 1
  ˅ req1
      operation: DRSUAPI_DS_SPN_OPERATION_ADD (0)
      unknown1: 0
    ˅ object_dn
        Referent ID: 0x0000000000020000
        Max Count: 28
        Offset: 0
        Actual Count: 28
        object_dn: CN=PC,OU=corp,DC=lab,DC=lan
      count: 1
    ˅ spn_names
        Referent ID: 0x0000000000020000
        Max Count: 1
      ˅ spn_names
        ˅ str
            Referent ID: 0x0000000000020000
            Max Count: 19
            Offset: 0
            Actual Count: 19
            str: TERMSRV/PC.lab.lan
```

```
0000  00 00 00 00 2c f8 7c b0  53 59 8b 42 87 96 cb 99   ····,·|· SY·B····
0010  c4 35 55 22 01 00 00 00  01 00 00 00 00 00 00 00   ·5U"···· ········
0020  00 00 00 00 00 00 00 00  00 00 02 00 00 00 00 00   ········ ········
0030  01 00 00 00 00 00 00 00  00 00 02 00 00 00 00 00   ········ ········
0040  1c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  1c 00 00 00 00 00 00 00  43 00 4e 00 3d 00 50 00   ········ C·N·=·P·
0060  43 00 2c 00 4f 00 55 00  3d 00 63 00 6f 00 72 00   C·,·O·U· =·c·o·r·
0070  70 00 2c 00 44 00 43 00  3d 00 6c 00 61 00 62 00   p·,·D·C· =·l·a·b·
0080  2c 00 44 00 43 00 3d 00  6c 00 61 00 6e 00 00 00   ,·D·C·=· l·a·n···
0090  01 00 00 00 00 00 00 00  00 00 02 00 00 00 00 00   ········ ········
00a0  13 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00b0  13 00 00 00 00 00 00 00  54 00 45 00 52 00 4d 00   ········ T·E·R·M·
00c0  53 00 52 00 56 00 2f 00  50 00 43 00 2e 00 6c 00   S·R·V·/· P·C·.·l·
00d0  61 00 62 00 2e 00 6c 00  61 00 6e 00 00 00 00 00   a·b·.·l· a·n·····
```

Frame (386 bytes)
Decrypted GSS-Krb5 CFX DCE (224 bytes)
Decrypted stub data (224 bytes)

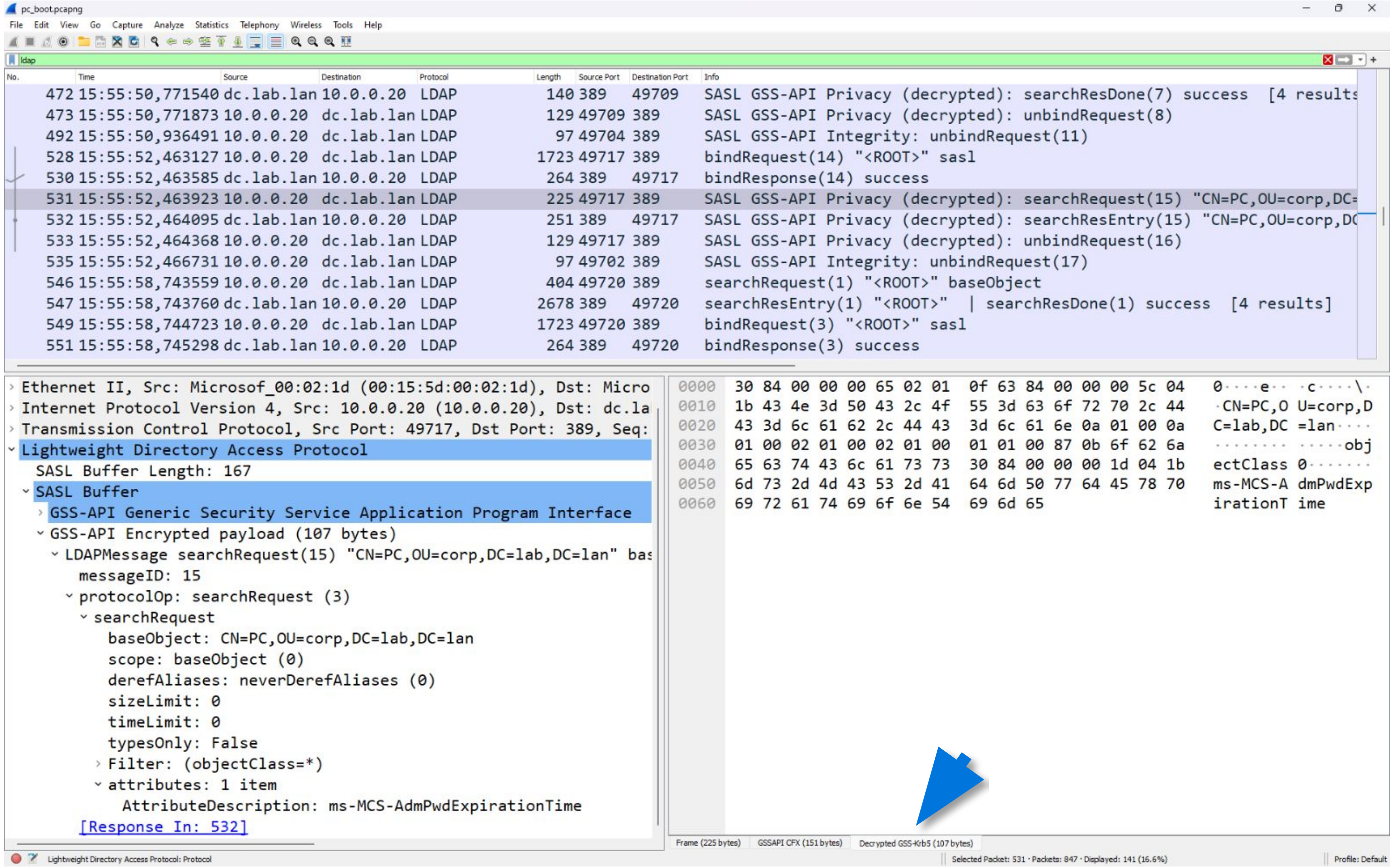
pc_boot.pcapng

Selected Packet: 322 · Packets: 847 · Displayed: 184 (21.7%)

Profile: Default

New look at decrypted capture

LDAP with
🔓 Decrypted
GSS-Krb5


pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ldap
Frame (225 bytes)
GSSAPI CFX (151 bytes)
Decrypted GSS-Krb5 (107 bytes)

```
No.  Time             Source      Destination Protocol Length Source Port Destination Port Info
 472 15:55:50,771540 dc.lab.lan 10.0.0.20  LDAP      140 389   49709  SASL GSS-API Privacy (decrypted): searchResDone(7) success  [4 results
 473 15:55:50,771873 10.0.0.20  dc.lab.lan LDAP      129 49709 389    SASL GSS-API Privacy (decrypted): unbindRequest(8)
 492 15:55:50,936491 10.0.0.20  dc.lab.lan LDAP       97 49704 389    SASL GSS-API Integrity: unbindRequest(11)
 528 15:55:52,463127 10.0.0.20  dc.lab.lan LDAP     1723 49717 389    bindRequest(14) "<ROOT>" sasl
 530 15:55:52,463585 dc.lab.lan 10.0.0.20  LDAP      264 389   49717  bindResponse(14) success
 531 15:55:52,463923 10.0.0.20  dc.lab.lan LDAP      225 49717 389    SASL GSS-API Privacy (decrypted): searchRequest(15) "CN=PC,OU=corp,DC=
 532 15:55:52,464095 dc.lab.lan 10.0.0.20  LDAP      251 389   49717  SASL GSS-API Privacy (decrypted): searchResEntry(15) "CN=PC,OU=corp,DC
 533 15:55:52,464368 10.0.0.20  dc.lab.lan LDAP      129 49717 389    SASL GSS-API Privacy (decrypted): unbindRequest(16)
 535 15:55:52,466731 10.0.0.20  dc.lab.lan LDAP       97 49702 389    SASL GSS-API Integrity: unbindRequest(17)
 546 15:55:58,743559 10.0.0.20  dc.lab.lan LDAP      404 49720 389    searchRequest(1) "<ROOT>" baseObject
 547 15:55:58,743760 dc.lab.lan 10.0.0.20  LDAP     2678 389   49720  searchResEntry(1) "<ROOT>"  | searchResDone(1) success  [4 results]
 549 15:55:58,744723 10.0.0.20  dc.lab.lan LDAP     1723 49720 389    bindRequest(3) "<ROOT>" sasl
 551 15:55:58,745298 dc.lab.lan 10.0.0.20  LDAP      264 389   49720  bindResponse(3) success
```

```
> Ethernet II, Src: Microsof_00:02:1d (00:15:5d:00:02:1d), Dst: Micro
> Internet Protocol Version 4, Src: 10.0.0.20 (10.0.0.20), Dst: dc.la
> Transmission Control Protocol, Src Port: 49717, Dst Port: 389, Seq:
v Lightweight Directory Access Protocol
    SASL Buffer Length: 167
  v SASL Buffer
    > GSS-API Generic Security Service Application Program Interface
    v GSS-API Encrypted payload (107 bytes)
      v LDAPMessage searchRequest(15) "CN=PC,OU=corp,DC=lab,DC=lan" bas
          messageID: 15
        v protocolOp: searchRequest (3)
          v searchRequest
              baseObject: CN=PC,OU=corp,DC=lab,DC=lan
              scope: baseObject (0)
              derefAliases: neverDerefAliases (0)
              sizeLimit: 0
              timeLimit: 0
              typesOnly: False
            > Filter: (objectClass=*)
            v attributes: 1 item
                AttributeDescription: ms-MCS-AdmPwdExpirationTime
          [Response In: 532]
```

```
0000  30 84 00 00 00 65 02 01  0f 63 84 00 00 00 5c 04   0····e·· ·c····\·
0010  1b 43 4e 3d 50 43 2c 4f  55 3d 63 6f 72 70 2c 44   ·CN=PC,O U=corp,D
0020  43 3d 6c 61 62 2c 44 43  3d 6c 61 6e 0a 01 00 0a   C=lab,DC =lan····
0030  01 00 02 01 00 02 01 00  01 01 00 87 0b 6f 62 6a   ········ ·····obj
0040  65 63 74 43 6c 61 73 73  30 84 00 00 00 1d 04 1b   ectClass 0·······
0050  6d 73 2d 4d 43 53 2d 41  64 6d 50 77 64 45 78 70   ms-MCS-A dmPwdExp
0060  69 72 61 74 69 6f 6e 54  69 6d 65                  irationT ime
```

```
Lightweight Directory Access Protocol: Protocol    Selected Packet: 531 · Packets: 847 · Displayed: 141 (16.6%)    Profile: Default
```

AGENDA

Introduction

How to decrypt Kerberos encrypted traffic?

And what about NTLM?

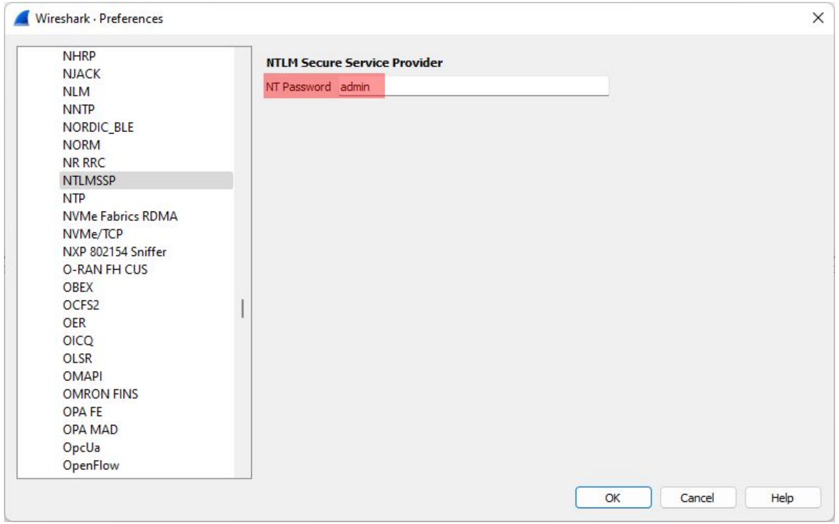
Conclusion

Questions

We can decrypt NTLM traffic too!

⚙️ Open Preferences
➡️ Protocols
➡️ NTLMSSP

⌨️ Type the cleartext password in the "NT Password" field



Wireshark · Preferences

NTLM Secure Service Provider

NT Password admin

😔 Limitations:

- need the cleartext password
- must be ASCII (incompatible with machine account passwords)
- can provide only one at a time

NTLM LDAP capture

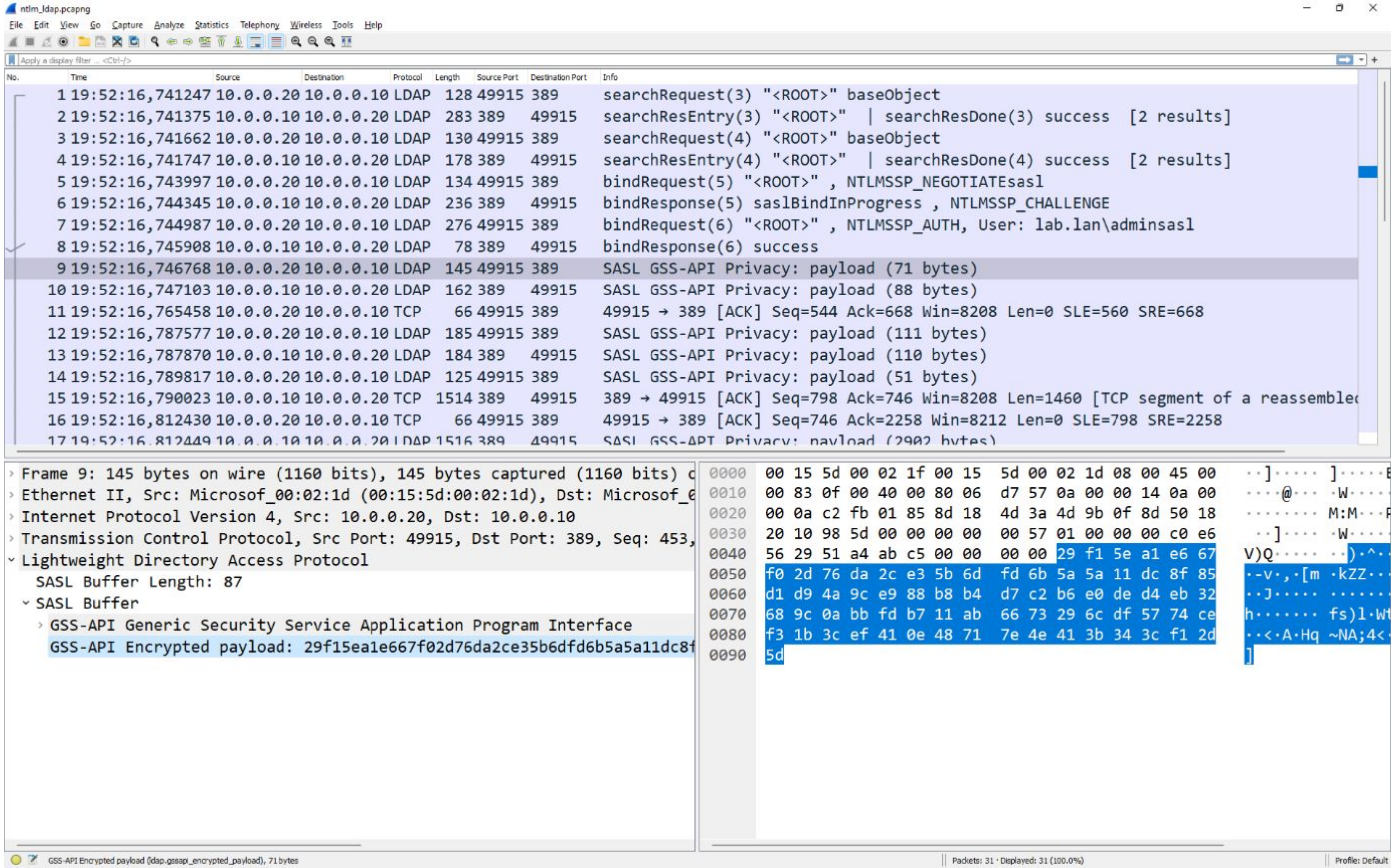
- Open `ntlm_ldap.pcapng` in Wireshark
- Get a first look
- Then provide the NT password: “`admin`”

New look at decrypted capture

LDAP with

🔒 `GSS-API`

`Encrypted payload`


ntlm_ldap.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No. Time            Source    Destination Protocol Length Source Port Destination Port Info
  1 19:52:16,741247 10.0.0.20 10.0.0.10 LDAP  128 49915 389   searchRequest(3) "<ROOT>" baseObject
  2 19:52:16,741375 10.0.0.10 10.0.0.20 LDAP  283 389   49915 searchResEntry(3) "<ROOT>"  | searchResDone(3) success  [2 results]
  3 19:52:16,741662 10.0.0.20 10.0.0.10 LDAP  130 49915 389   searchRequest(4) "<ROOT>" baseObject
  4 19:52:16,741747 10.0.0.10 10.0.0.20 LDAP  178 389   49915 searchResEntry(4) "<ROOT>"  | searchResDone(4) success  [2 results]
  5 19:52:16,743997 10.0.0.20 10.0.0.10 LDAP  134 49915 389   bindRequest(5) "<ROOT>" , NTLMSSP_NEGOTIATEsasl
  6 19:52:16,744345 10.0.0.10 10.0.0.20 LDAP  236 389   49915 bindResponse(5) saslBindInProgress , NTLMSSP_CHALLENGE
  7 19:52:16,744987 10.0.0.20 10.0.0.10 LDAP  276 49915 389   bindRequest(6) "<ROOT>" , NTLMSSP_AUTH, User: lab.lan\adminsasl
  8 19:52:16,745908 10.0.0.10 10.0.0.20 LDAP   78 389   49915 bindResponse(6) success
  9 19:52:16,746768 10.0.0.20 10.0.0.10 LDAP  145 49915 389   SASL GSS-API Privacy: payload (71 bytes)
 10 19:52:16,747103 10.0.0.10 10.0.0.20 LDAP  162 389   49915 SASL GSS-API Privacy: payload (88 bytes)
 11 19:52:16,765458 10.0.0.20 10.0.0.10 TCP    66 49915 389   49915 → 389 [ACK] Seq=544 Ack=668 Win=8208 Len=0 SLE=560 SRE=668
 12 19:52:16,787577 10.0.0.20 10.0.0.10 LDAP  185 49915 389   SASL GSS-API Privacy: payload (111 bytes)
 13 19:52:16,787870 10.0.0.10 10.0.0.20 LDAP  184 389   49915 SASL GSS-API Privacy: payload (110 bytes)
 14 19:52:16,789817 10.0.0.20 10.0.0.10 LDAP  125 49915 389   SASL GSS-API Privacy: payload (51 bytes)
 15 19:52:16,790023 10.0.0.10 10.0.0.20 TCP  1514 389   49915 389 → 49915 [ACK] Seq=798 Ack=746 Win=8208 Len=1460 [TCP segment of a reassemblec
 16 19:52:16,812430 10.0.0.20 10.0.0.10 TCP    66 49915 389   49915 → 389 [ACK] Seq=746 Ack=2258 Win=8212 Len=0 SLE=798 SRE=2258
 17 19:52:16,812449 10.0.0.10 10.0.0.20 LDAP 1516 389   49915 SASL GSS-API Privacy: payload (2902 bytes)
```

```
> Frame 9: 145 bytes on wire (1160 bits), 145 bytes captured (1160 bits) c
> Ethernet II, Src: Microsof_00:02:1d (00:15:5d:00:02:1d), Dst: Microsof_0
> Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.0.10
> Transmission Control Protocol, Src Port: 49915, Dst Port: 389, Seq: 453,
v Lightweight Directory Access Protocol
    SASL Buffer Length: 87
  v SASL Buffer
    > GSS-API Generic Security Service Application Program Interface
      GSS-API Encrypted payload: 29f15ea1e667f02d76da2ce35b6dfd6b5a5a11dc8f
```

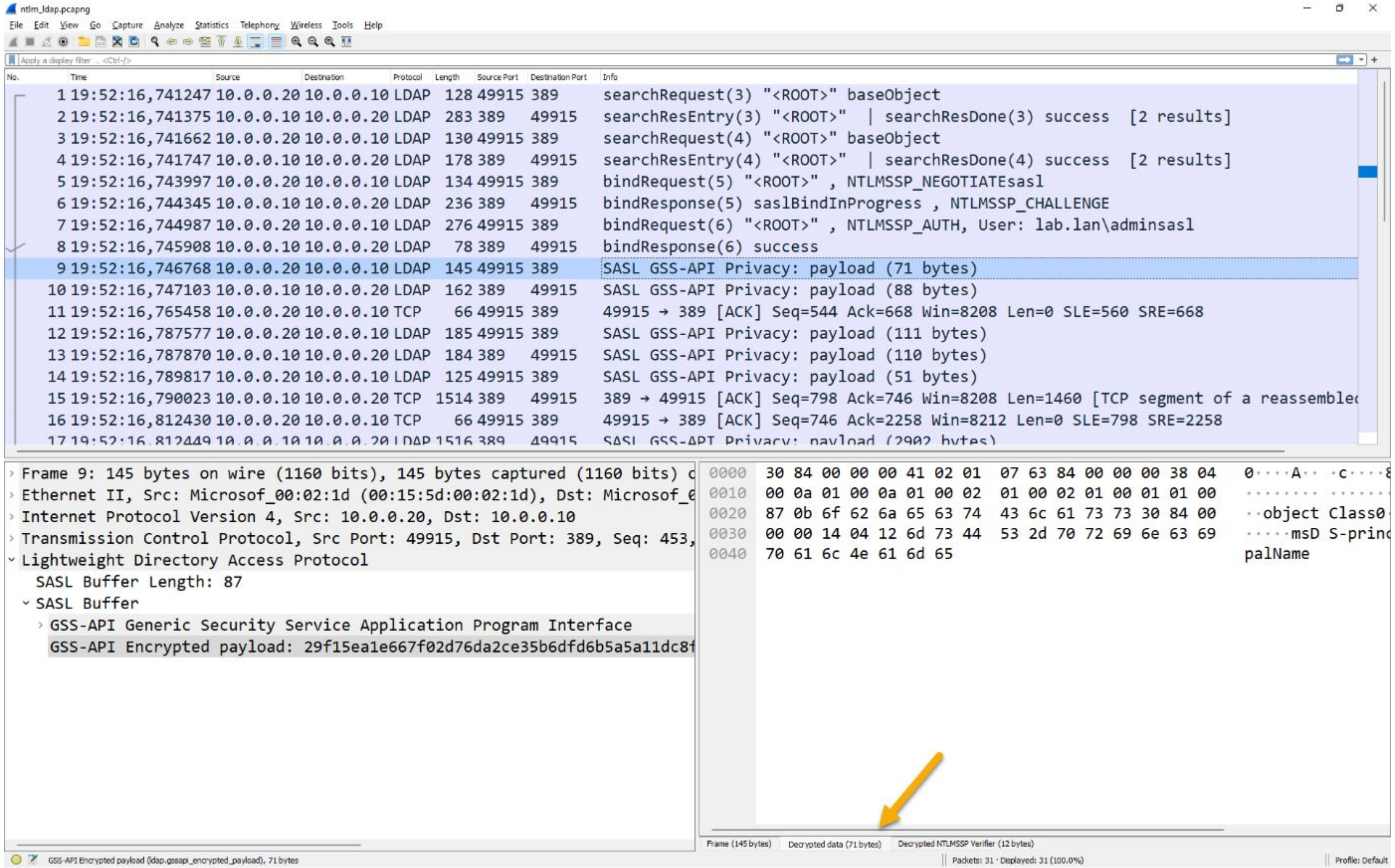
```
0000  00 15 5d 00 02 1f 00 15  5d 00 02 1d 08 00 45 00   ··]····· ]·····E
0010  00 83 0f 00 40 00 80 06  d7 57 0a 00 00 14 0a 00   ····@··· ·W·····
0020  00 0a c2 fb 01 85 8d 18  4d 3a 4d 9b 0f 8d 50 18   ········ M:M···P
0030  20 10 98 5d 00 00 00 00  00 57 01 00 00 00 c0 e6    ··]···· ·W·····
0040  56 29 51 a4 ab c5 00 00  00 00 29 f1 5e a1 e6 67   V)Q····· ··)·^··
0050  f0 2d 76 da 2c e3 5b 6d  fd 6b 5a 5a 11 dc 8f 85   ·-v·,·[m ·kZZ···
0060  d1 d9 4a 9c e9 88 b8 b4  d7 c2 b6 e0 de d4 eb 32   ··J····· ·······
0070  68 9c 0a bb fd b7 11 ab  66 73 29 6c df 57 74 ce   h······· fs)l·Wt
0080  f3 1b 3c ef 41 0e 48 71  7e 4e 41 3b 34 3c f1 2d   ··<·A·Hq ~NA;4<·
0090  5d                                                 ]
```

GSS-API Encrypted payload (ldap.gssapi_encrypted_payload), 71 bytes | Packets: 31 · Displayed: 31 (100.0%) | Profile: Default

New look at decrypted capture

LDAP with
🔓 `Decrypted data`

➡️ Notice the tab at the bottom
Unfortunately the LDAP dissector does not seem to use this data


ntlm_ldap.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time            Source    Destination Protocol Length Source Port Destination Port Info
   1 19:52:16,741247 10.0.0.20 10.0.0.10 LDAP  128 49915 389   searchRequest(3) "<ROOT>" baseObject
   2 19:52:16,741375 10.0.0.10 10.0.0.20 LDAP  283 389   49915 searchResEntry(3) "<ROOT>"  | searchResDone(3) success  [2 results]
   3 19:52:16,741662 10.0.0.20 10.0.0.10 LDAP  130 49915 389   searchRequest(4) "<ROOT>" baseObject
   4 19:52:16,741747 10.0.0.10 10.0.0.20 LDAP  178 389   49915 searchResEntry(4) "<ROOT>"  | searchResDone(4) success  [2 results]
   5 19:52:16,743997 10.0.0.20 10.0.0.10 LDAP  134 49915 389   bindRequest(5) "<ROOT>" , NTLMSSP_NEGOTIATEsasl
   6 19:52:16,744345 10.0.0.10 10.0.0.20 LDAP  236 389   49915 bindResponse(5) saslBindInProgress , NTLMSSP_CHALLENGE
   7 19:52:16,744987 10.0.0.20 10.0.0.10 LDAP  276 49915 389   bindRequest(6) "<ROOT>" , NTLMSSP_AUTH, User: lab.lan\adminsasl
   8 19:52:16,745908 10.0.0.10 10.0.0.20 LDAP   78 389   49915 bindResponse(6) success
   9 19:52:16,746768 10.0.0.20 10.0.0.10 LDAP  145 49915 389   SASL GSS-API Privacy: payload (71 bytes)
  10 19:52:16,747103 10.0.0.10 10.0.0.20 LDAP  162 389   49915 SASL GSS-API Privacy: payload (88 bytes)
  11 19:52:16,765458 10.0.0.20 10.0.0.10 TCP    66 49915 389   49915 → 389 [ACK] Seq=544 Ack=668 Win=8208 Len=0 SLE=560 SRE=668
  12 19:52:16,787577 10.0.0.20 10.0.0.10 LDAP  185 49915 389   SASL GSS-API Privacy: payload (111 bytes)
  13 19:52:16,787870 10.0.0.10 10.0.0.20 LDAP  184 389   49915 SASL GSS-API Privacy: payload (110 bytes)
  14 19:52:16,789817 10.0.0.20 10.0.0.10 LDAP  125 49915 389   SASL GSS-API Privacy: payload (51 bytes)
  15 19:52:16,790023 10.0.0.10 10.0.0.20 TCP  1514 389   49915 389 → 49915 [ACK] Seq=798 Ack=746 Win=8208 Len=1460 [TCP segment of a reassemblec
  16 19:52:16,812430 10.0.0.20 10.0.0.10 TCP    66 49915 389   49915 → 389 [ACK] Seq=746 Ack=2258 Win=8212 Len=0 SLE=798 SRE=2258
  17 19:52:16,812449 10.0.0.10 10.0.0.20 LDAP 1516 389   49915 SASL GSS-API Privacy: payload (2902 bytes)
```

```
> Frame 9: 145 bytes on wire (1160 bits), 145 bytes captured (1160 bits) c
> Ethernet II, Src: Microsof_00:02:1d (00:15:5d:00:02:1d), Dst: Microsof_0
> Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.0.10
> Transmission Control Protocol, Src Port: 49915, Dst Port: 389, Seq: 453,
v Lightweight Directory Access Protocol
    SASL Buffer Length: 87
  v SASL Buffer
    > GSS-API Generic Security Service Application Program Interface
      GSS-API Encrypted payload: 29f15ea1e667f02d76da2ce35b6dfd6b5a5a11dc8f
```

```
0000  30 84 00 00 00 41 02 01  07 63 84 00 00 00 38 04   0····A·· ·c····8
0010  00 0a 01 00 0a 01 00 02  01 00 02 01 00 01 01 00   ········ ·······
0020  87 0b 6f 62 6a 65 63 74  43 6c 61 73 73 30 84 00   ··object Class0
0030  00 00 14 04 12 6d 73 44  53 2d 70 72 69 6e 63 69   ·····msD S-princ
0040  70 61 6c 4e 61 6d 65                               palName
```

Frame (145 bytes)
Decrypted data (71 bytes)
Decrypted NTLMSSP Verifier (12 bytes)

GSS-API Encrypted payload (ldap.gssapi_encrypted_payload), 71 bytes

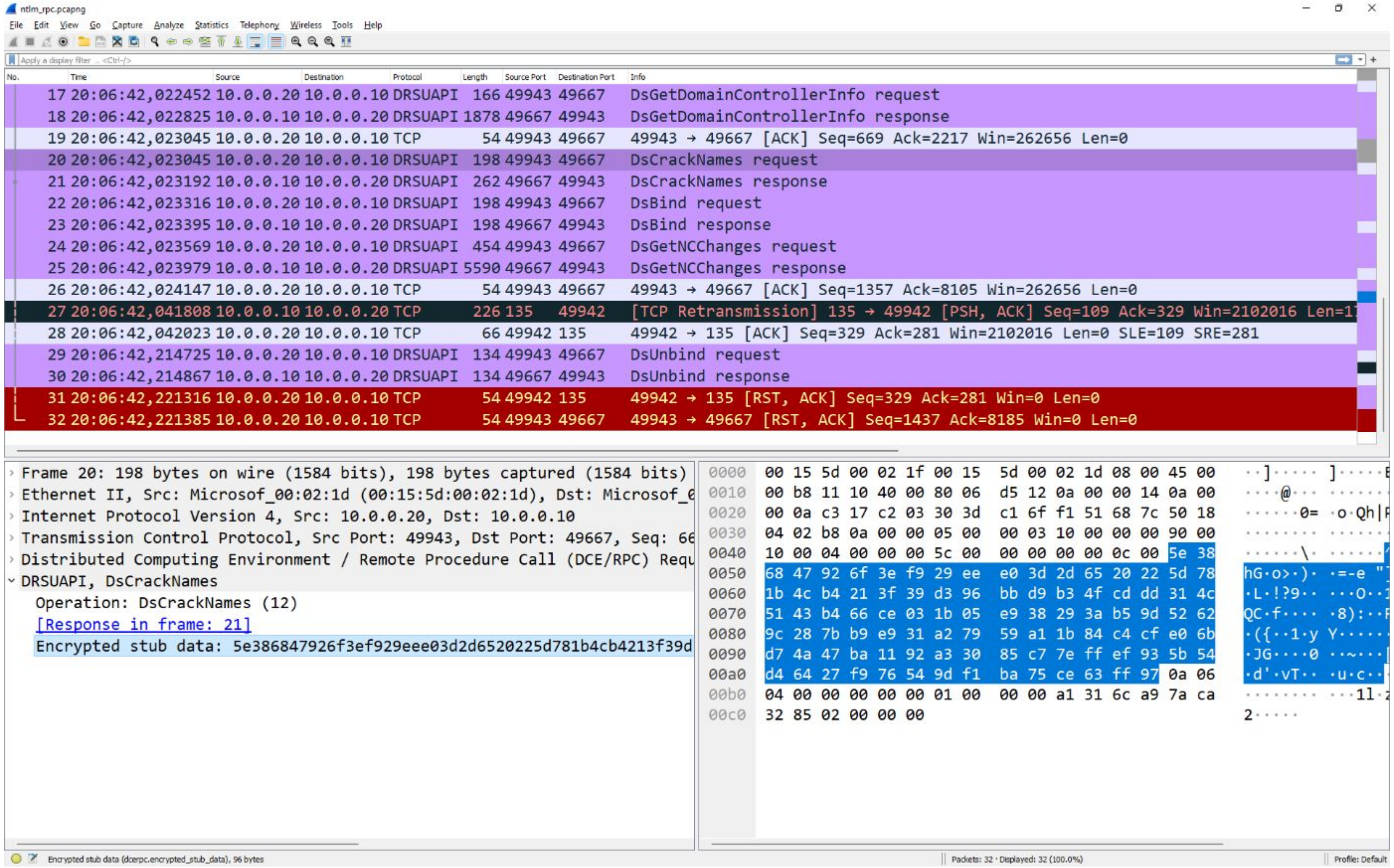
NTLM RPC capture

- Open `ntlm_rpc.pcapng` in Wireshark
- Get a first look
- Then provide the NT password: “`admin`”

New look at decrypted capture

DRSUAPI `DsCrackNames` with

🔒 `encrypted stub data`



ntlm_rpc.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time             Source    Destination Protocol Length Source Port Destination Port Info
  17 20:06:42,022452 10.0.0.20 10.0.0.10 DRSUAPI  166 49943 49667  DsGetDomainControllerInfo request
  18 20:06:42,022825 10.0.0.10 10.0.0.20 DRSUAPI 1878 49667 49943  DsGetDomainControllerInfo response
  19 20:06:42,023045 10.0.0.20 10.0.0.10 TCP       54 49943 49667  49943 → 49667 [ACK] Seq=669 Ack=2217 Win=262656 Len=0
  20 20:06:42,023045 10.0.0.20 10.0.0.10 DRSUAPI  198 49943 49667  DsCrackNames request
  21 20:06:42,023192 10.0.0.10 10.0.0.20 DRSUAPI  262 49667 49943  DsCrackNames response
  22 20:06:42,023316 10.0.0.20 10.0.0.10 DRSUAPI  198 49943 49667  DsBind request
  23 20:06:42,023395 10.0.0.10 10.0.0.20 DRSUAPI  198 49667 49943  DsBind response
  24 20:06:42,023569 10.0.0.20 10.0.0.10 DRSUAPI  454 49943 49667  DsGetNCChanges request
  25 20:06:42,023979 10.0.0.10 10.0.0.20 DRSUAPI 5590 49667 49943  DsGetNCChanges response
  26 20:06:42,024147 10.0.0.20 10.0.0.10 TCP       54 49943 49667  49943 → 49667 [ACK] Seq=1357 Ack=8105 Win=262656 Len=0
  27 20:06:42,041808 10.0.0.10 10.0.0.20 TCP      226 135   49942  [TCP Retransmission] 135 → 49942 [PSH, ACK] Seq=109 Ack=329 Win=2102016 Len=1
  28 20:06:42,042023 10.0.0.20 10.0.0.10 TCP       66 49942 135    49942 → 135 [ACK] Seq=329 Ack=281 Win=2102016 Len=0 SLE=109 SRE=281
  29 20:06:42,214725 10.0.0.20 10.0.0.10 DRSUAPI  134 49943 49667  DsUnbind request
  30 20:06:42,214867 10.0.0.10 10.0.0.20 DRSUAPI  134 49667 49943  DsUnbind response
  31 20:06:42,221316 10.0.0.20 10.0.0.10 TCP       54 49942 135    49942 → 135 [RST, ACK] Seq=329 Ack=281 Win=0 Len=0
  32 20:06:42,221385 10.0.0.20 10.0.0.10 TCP       54 49943 49667  49943 → 49667 [RST, ACK] Seq=1437 Ack=8185 Win=0 Len=0
```

```
> Frame 20: 198 bytes on wire (1584 bits), 198 bytes captured (1584 bits)
> Ethernet II, Src: Microsof_00:02:1d (00:15:5d:00:02:1d), Dst: Microsof_0
> Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.0.10
> Transmission Control Protocol, Src Port: 49943, Dst Port: 49667, Seq: 66
> Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Requ
v DRSUAPI, DsCrackNames
    Operation: DsCrackNames (12)
    [Response in frame: 21]
    Encrypted stub data: 5e386847926f3ef929eee03d2d6520225d781b4cb4213f39d
```

```
0000  00 15 5d 00 02 1f 00 15  5d 00 02 1d 08 00 45 00   ··]·····  ]·····E
0010  00 b8 11 10 40 00 80 06  d5 12 0a 00 00 14 0a 00   ····@···  ······
0020  00 0a c3 17 c2 03 30 3d  c1 6f f1 51 68 7c 50 18   ······0=  ·o·Qh|P
0030  04 02 b8 0a 00 00 05 00  00 03 10 00 00 00 90 00   ········  ······
0040  10 00 04 00 00 00 5c 00  00 00 00 00 0c 00 5e 38   ······\·  ······^
0050  68 47 92 6f 3e f9 29 ee  e0 3d 2d 65 20 22 5d 78   hG·o>·)·  ·=-e "]
0060  1b 4c b4 21 3f 39 d3 96  bb d9 b3 4f cd dd 31 4c   ·L·!?9··  ···O··1
0070  51 43 b4 66 ce 03 1b 05  e9 38 29 3a b5 9d 52 62   QC·f····  ·8):··R
0080  9c 28 7b b9 e9 31 a2 79  59 a1 1b 84 c4 cf e0 6b   ·({··1·y  Y······
0090  d7 4a 47 ba 11 92 a3 30  85 c7 7e ff ef 93 5b 54   ·JG····0  ··~···[
00a0  d4 64 27 f9 76 54 9d f1  ba 75 ce 63 ff 97 0a 06   ·d'·vT··  ·u·c··
00b0  04 00 00 00 00 00 01 00  00 00 a1 31 6c a9 7a ca   ········  ···1l·z
00c0  32 85 02 00 00 00                                  2·····
```

Encrypted stub data (dcerpc.encrypted_stub_data), 96 bytes

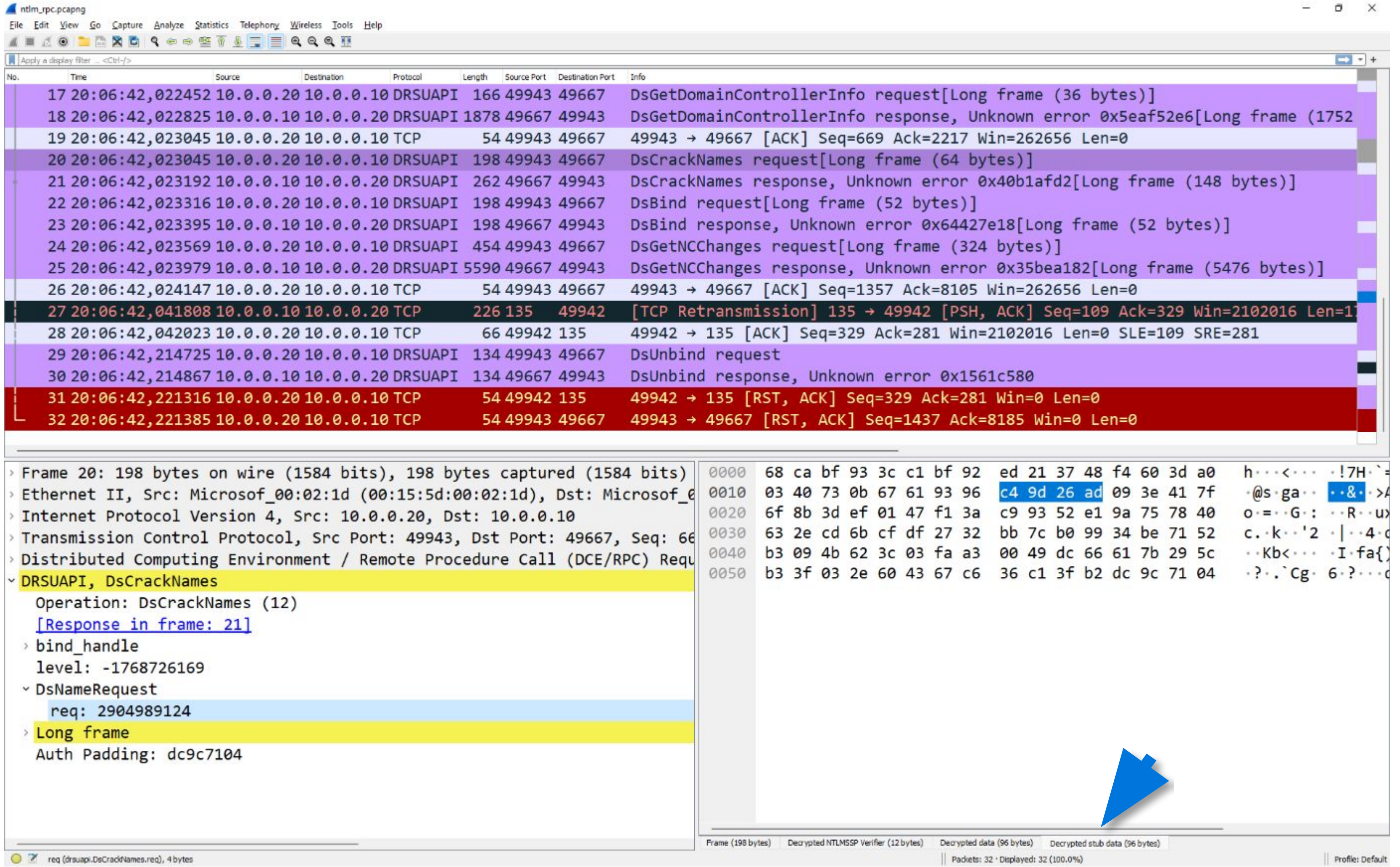
Packets: 32 · Displayed: 32 (100.0%)

Profile: Default

New look at decrypted capture

DRSUAPI `DsCrackNames` with 🔓 `decrypted stub data`

The DRSUAPI dissector uses this data… but there seems to be a bug since no data makes sense 😢



ntlm_rpc.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time             Source     Destination Protocol Length Source Port Destination Port Info
   17 20:06:42,022452 10.0.0.20 10.0.0.10 DRSUAPI  166 49943 49667   DsGetDomainControllerInfo request[Long frame (36 bytes)]
   18 20:06:42,022825 10.0.0.10 10.0.0.20 DRSUAPI 1878 49667 49943   DsGetDomainControllerInfo response, Unknown error 0x5eaf52e6[Long frame (1752
   19 20:06:42,023045 10.0.0.20 10.0.0.10 TCP       54 49943 49667   49943 → 49667 [ACK] Seq=669 Ack=2217 Win=262656 Len=0
   20 20:06:42,023045 10.0.0.20 10.0.0.10 DRSUAPI  198 49943 49667   DsCrackNames request[Long frame (64 bytes)]
   21 20:06:42,023192 10.0.0.10 10.0.0.20 DRSUAPI  262 49667 49943   DsCrackNames response, Unknown error 0x40b1afd2[Long frame (148 bytes)]
   22 20:06:42,023316 10.0.0.20 10.0.0.10 DRSUAPI  198 49943 49667   DsBind request[Long frame (52 bytes)]
   23 20:06:42,023395 10.0.0.10 10.0.0.20 DRSUAPI  198 49667 49943   DsBind response, Unknown error 0x64427e18[Long frame (52 bytes)]
   24 20:06:42,023569 10.0.0.20 10.0.0.10 DRSUAPI  454 49943 49667   DsGetNCChanges request[Long frame (324 bytes)]
   25 20:06:42,023979 10.0.0.10 10.0.0.20 DRSUAPI 5590 49667 49943   DsGetNCChanges response, Unknown error 0x35bea182[Long frame (5476 bytes)]
   26 20:06:42,024147 10.0.0.20 10.0.0.10 TCP       54 49943 49667   49943 → 49667 [ACK] Seq=1357 Ack=8105 Win=262656 Len=0
   27 20:06:42,041808 10.0.0.10 10.0.0.20 TCP      226 135   49942   [TCP Retransmission] 135 → 49942 [PSH, ACK] Seq=109 Ack=329 Win=2102016 Len=1
   28 20:06:42,042023 10.0.0.20 10.0.0.10 TCP       66 49942 135     49942 → 135 [ACK] Seq=329 Ack=281 Win=2102016 Len=0 SLE=109 SRE=281
   29 20:06:42,214725 10.0.0.20 10.0.0.10 DRSUAPI  134 49943 49667   DsUnbind request
   30 20:06:42,214867 10.0.0.10 10.0.0.20 DRSUAPI  134 49667 49943   DsUnbind response, Unknown error 0x1561c580
   31 20:06:42,221316 10.0.0.20 10.0.0.10 TCP       54 49942 135     49942 → 135 [RST, ACK] Seq=329 Ack=281 Win=0 Len=0
   32 20:06:42,221385 10.0.0.20 10.0.0.10 TCP       54 49943 49667   49943 → 49667 [RST, ACK] Seq=1437 Ack=8185 Win=0 Len=0
```

```
> Frame 20: 198 bytes on wire (1584 bits), 198 bytes captured (1584 bits)
> Ethernet II, Src: Microsof_00:02:1d (00:15:5d:00:02:1d), Dst: Microsof_0
> Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.0.10
> Transmission Control Protocol, Src Port: 49943, Dst Port: 49667, Seq: 66
> Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Requ
v DRSUAPI, DsCrackNames
    Operation: DsCrackNames (12)
    [Response in frame: 21]
  > bind_handle
    level: -1768726169
  v DsNameRequest
      req: 2904989124
  > Long frame
    Auth Padding: dc9c7104
```

```
0000  68 ca bf 93 3c c1 bf 92  ed 21 37 48 f4 60 3d a0   h···<···  ·!7H·`=
0010  03 40 73 0b 67 61 93 96  c4 9d 26 ad 09 3e 41 7f   ·@s·ga··  ··&··>A
0020  6f 8b 3d ef 01 47 f1 3a  c9 93 52 e1 9a 75 78 40   o·=··G·:  ··R··ux
0030  63 2e cd 6b cf df 27 32  bb 7c b0 99 34 be 71 52   c.·k··'2  ·|··4·q
0040  b3 09 4b 62 3c 03 fa a3  00 49 dc 66 61 7b 29 5c   ··Kb<···  ·I·fa{)
0050  b3 3f 03 2e 60 43 67 c6  36 c1 3f b2 dc 9c 71 04   ·?·.`Cg·  6·?···q
```

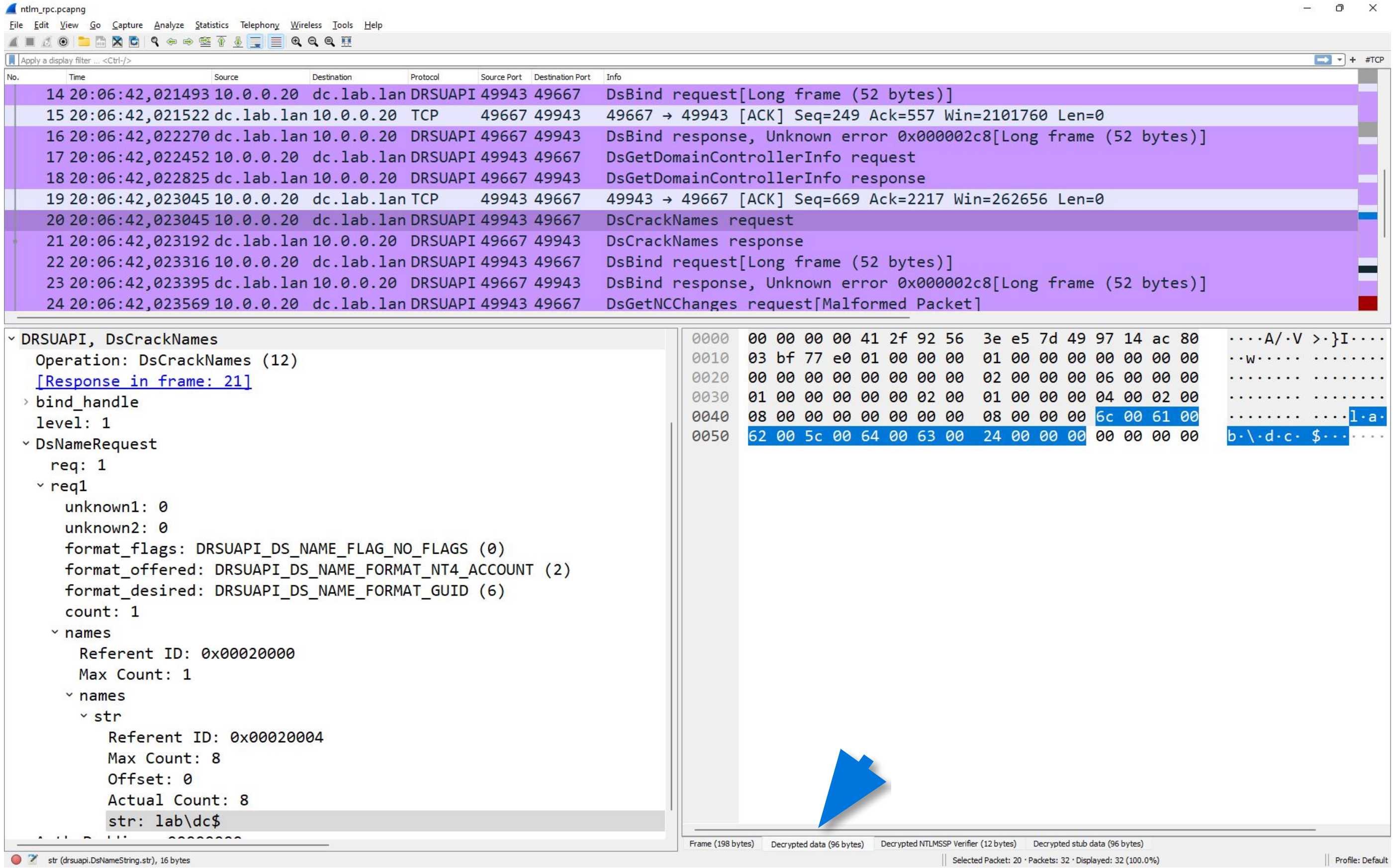
Frame (198 bytes)
Decrypted NTLMSSP Verifier (12 bytes)
Decrypted data (96 bytes)
Decrypted stub data (96 bytes)
req (drsuapi.DsCrackNames.req), 4 bytes
Packets: 32 · Displayed: 32 (100.0%)
Profile: Default

New look at decrypted capture

DRSUAPI `DsCrackNames` with 🔓 `decrypted stub data`

Actually… 😉
I [fixed](#) this bug! 🎉

➡️ upgrade to v4.0.6 (or v3.6.14 backport)


ntlm_rpc.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
#TCP

```
No.  Time            Source      Destination Protocol Source Port Destination Port Info
14 20:06:42,021493 10.0.0.20  dc.lab.lan DRSUAPI 49943 49667 DsBind request[Long frame (52 bytes)]
15 20:06:42,021522 dc.lab.lan 10.0.0.20  TCP     49667 49943 49667 → 49943 [ACK] Seq=249 Ack=557 Win=2101760 Len=0
16 20:06:42,022270 dc.lab.lan 10.0.0.20  DRSUAPI 49667 49943 DsBind response, Unknown error 0x000002c8[Long frame (52 bytes)]
17 20:06:42,022452 10.0.0.20  dc.lab.lan DRSUAPI 49943 49667 DsGetDomainControllerInfo request
18 20:06:42,022825 dc.lab.lan 10.0.0.20  DRSUAPI 49667 49943 DsGetDomainControllerInfo response
19 20:06:42,023045 10.0.0.20  dc.lab.lan TCP     49943 49667 49943 → 49667 [ACK] Seq=669 Ack=2217 Win=262656 Len=0
20 20:06:42,023045 10.0.0.20  dc.lab.lan DRSUAPI 49943 49667 DsCrackNames request
21 20:06:42,023192 dc.lab.lan 10.0.0.20  DRSUAPI 49667 49943 DsCrackNames response
22 20:06:42,023316 10.0.0.20  dc.lab.lan DRSUAPI 49943 49667 DsBind request[Long frame (52 bytes)]
23 20:06:42,023395 dc.lab.lan 10.0.0.20  DRSUAPI 49667 49943 DsBind response, Unknown error 0x000002c8[Long frame (52 bytes)]
24 20:06:42,023569 10.0.0.20  dc.lab.lan DRSUAPI 49943 49667 DsGetNCChanges request[Malformed Packet]
```

```
˅ DRSUAPI, DsCrackNames
    Operation: DsCrackNames (12)
    [Response in frame: 21]
  › bind_handle
    level: 1
  ˅ DsNameRequest
      req: 1
    ˅ req1
        unknown1: 0
        unknown2: 0
        format_flags: DRSUAPI_DS_NAME_FLAG_NO_FLAGS (0)
        format_offered: DRSUAPI_DS_NAME_FORMAT_NT4_ACCOUNT (2)
        format_desired: DRSUAPI_DS_NAME_FORMAT_GUID (6)
        count: 1
      ˅ names
          Referent ID: 0x00020000
          Max Count: 1
        ˅ names
          ˅ str
              Referent ID: 0x00020004
              Max Count: 8
              Offset: 0
              Actual Count: 8
              str: lab\dc$
    Auth Padding: 00000000
```

```
0000  00 00 00 00 41 2f 92 56  3e e5 7d 49 97 14 ac 80   ····A/·V >·}I····
0010  03 bf 77 e0 01 00 00 00  01 00 00 00 00 00 00 00   ··w····· ········
0020  00 00 00 00 00 00 00 00  02 00 00 00 06 00 00 00   ········ ········
0030  01 00 00 00 00 00 02 00  01 00 00 00 04 00 02 00   ········ ········
0040  08 00 00 00 00 00 00 00  08 00 00 00 6c 00 61 00   ········ ····l·a·
0050  62 00 5c 00 64 00 63 00  24 00 00 00 00 00 00 00   b·\·d·c· $·······
```

Frame (198 bytes)
Decrypted data (96 bytes)
Decrypted NTLMSSP Verifier (12 bytes)
Decrypted stub data (96 bytes)

str (drsuapi.DsNameString.str), 16 bytes

Selected Packet: 20 · Packets: 32 · Displayed: 32 (100.0%)

Profile: Default

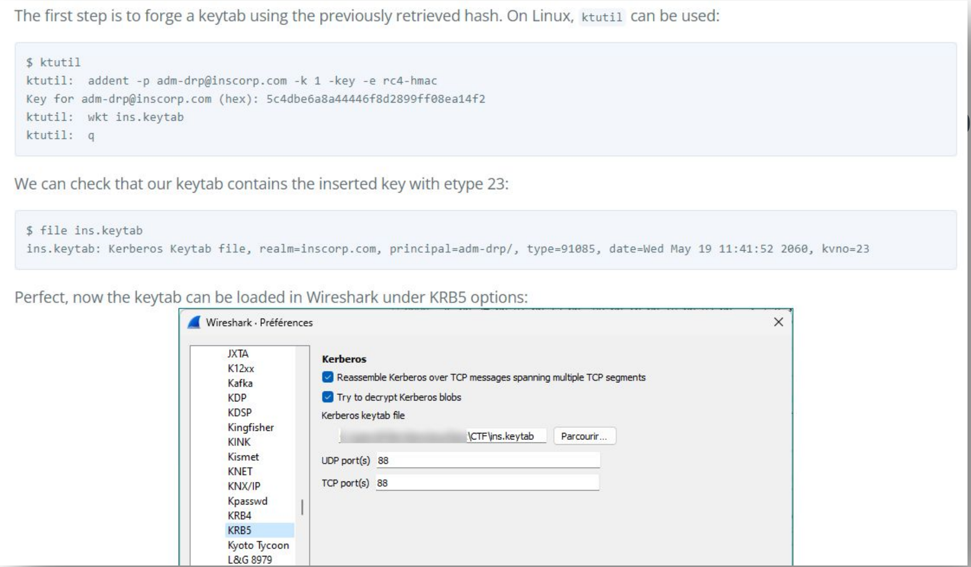
Just the NTLM hash?

🤔 What if I just have the NTLM hash instead of the password?

➡️ Put the NT hash in a keytab (too!) file with **`keytype=23`** (RC4 == NT hash) and configure it like previously

(yes we're configuring Kerberos options to decrypt NTLM… 🤷‍♂️)

ted!



The first step is to forge a keytab using the previously retrieved hash. On Linux, `ktutil` can be used:

```
$ ktutil
ktutil:  addent -p adm-drp@inscorp.com -k 1 -key -e rc4-hmac
Key for adm-drp@inscorp.com (hex): 5c4dbe6a8a44446f8d2899ff08ea14f2
ktutil:  wkt ins.keytab
ktutil:  q
```

We can check that our keytab contains the inserted key with etype 23:

```
$ file ins.keytab
ins.keytab: Kerberos Keytab file, realm=inscorp.com, principal=adm-drp/, type=91085, date=Wed May 19 11:41:52 2060, kvno=23
```

Perfect, now the keytab can be loaded in Wireshark under KRB5 options:

Kerberos

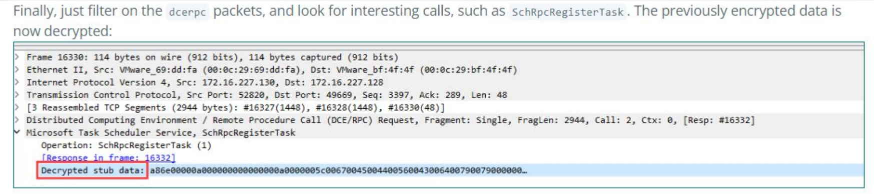
☑ Reassemble Kerberos over TCP messages spanning multiple TCP segments

☑ Try to decrypt Kerberos blobs

Kerberos keytab file: [redacted]\CTF\ins.keytab

UDP port(s): 88

TCP port(s): 88



Finally, just filter on the `dcerpc` packets, and look for interesting calls, such as `SchRpcRegisterTask`. The previously encrypted data is now decrypted:

```
> Frame 16330: 114 bytes on wire (912 bits), 114 bytes captured (912 bits)
> Ethernet II, Src: VMware_69:dd:fa (00:0c:29:69:dd:fa), Dst: VMware_bf:4f:4f (00:0c:29:bf:4f:4f)
> Internet Protocol Version 4, Src: 172.16.227.130, Dst: 172.16.227.128
> Transmission Control Protocol, Src Port: 52820, Dst Port: 49669, Seq: 3397, Ack: 289, Len: 48
> [3 Reassembled TCP Segments (2944 bytes): #16327(1448), #16328(1448), #16330(48)]
> Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Single, FragLen: 2944, Call: 2, Ctx: 0, [Resp: #16332]
v Microsoft Task Scheduler Service, SchRpcRegisterTask
    Operation: SchRpcRegisterTask (1)
    [Response in frame: 16332]
    Decrypted stub data: a86e00000a000000000000000a0000005c00670045004400560043006400790079000000…
```

Source: https://tipi-hack.github.io/2023/01/22/insomnihack-teaser-autopsy.html#the-dcerpc-decryption-problem

AGENDA

Introduction

How to decrypt Kerberos encrypted traffic?

And what about NTLM?

Conclusion

Questions

Recap

Capture encrypted traffic

Get Kerberos keys

Put keys in keytab file

Give keytab to Wireshark

Enjoy!

Recap

🚀 I know it was quick...

📄 So you can refer later to the slides or the blogpost:
https://medium.com/tenable-techblog/decrypt-encrypted-stub-data-in-wireshark-deb132c076e7

📜 Wireshark wiki on:

Kerberos
https://wiki.wireshark.org/Kerberos

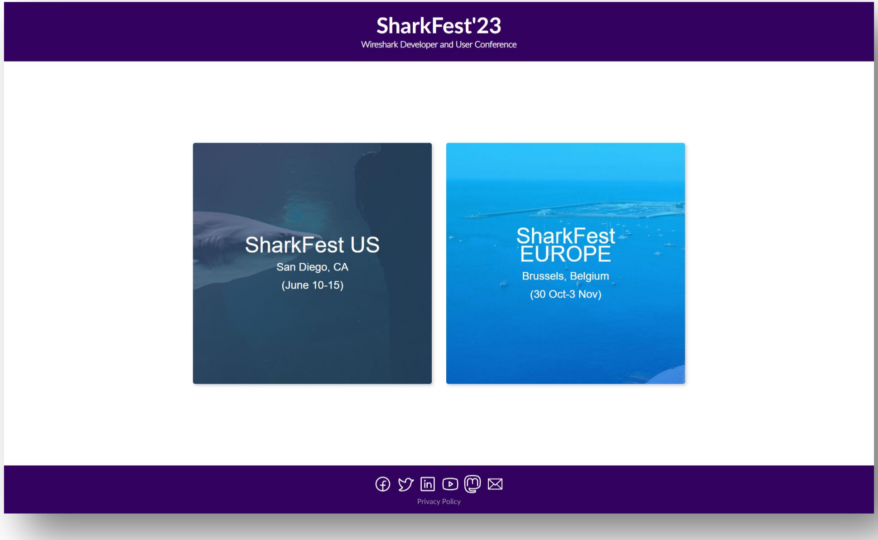
NTLM
https://wiki.wireshark.org/NTLMSSP

Thanks

👏 Many thanks to Ronnie Sahlberg for making me this discover this feature and for having implemented it!

🙇‍♂️ For the Wireshark community who built this incredible tool
and who encouraged me to talk about this feature at SharkFest Europe 2022

🧑‍🎓 You should attend SharkFest!
https://sharkfest.wireshark.org/



SharkFest'23
Wireshark Developer and User Conference

SharkFest US
San Diego, CA
(June 10-15)

SharkFest EUROPE
Brussels, Belgium
(30 Oct-3 Nov)

Privacy Policy

AGENDA

Introduction

How to decrypt Kerberos encrypted traffic?

And what about NTLM?

Conclusion

Questions

Questions?

Blogpost ➡️

@cnotin

@cnotin@infosec.exchange

https://clement.notin.org

Clément Notin

AGENDA

Introduction

How to decrypt Kerberos encrypted traffic?

And what about NTLM?

Conclusion

Questions

Bonus

Who is this talk for?

🧪

Security researchers who work on encrypted Microsoft traffic, especially Active Directory

🔍

Security analysts who need to analyze suspicious traffic

⚙️

Developers who need to work with the underlying encrypted protocols

🧐

Curious folks!

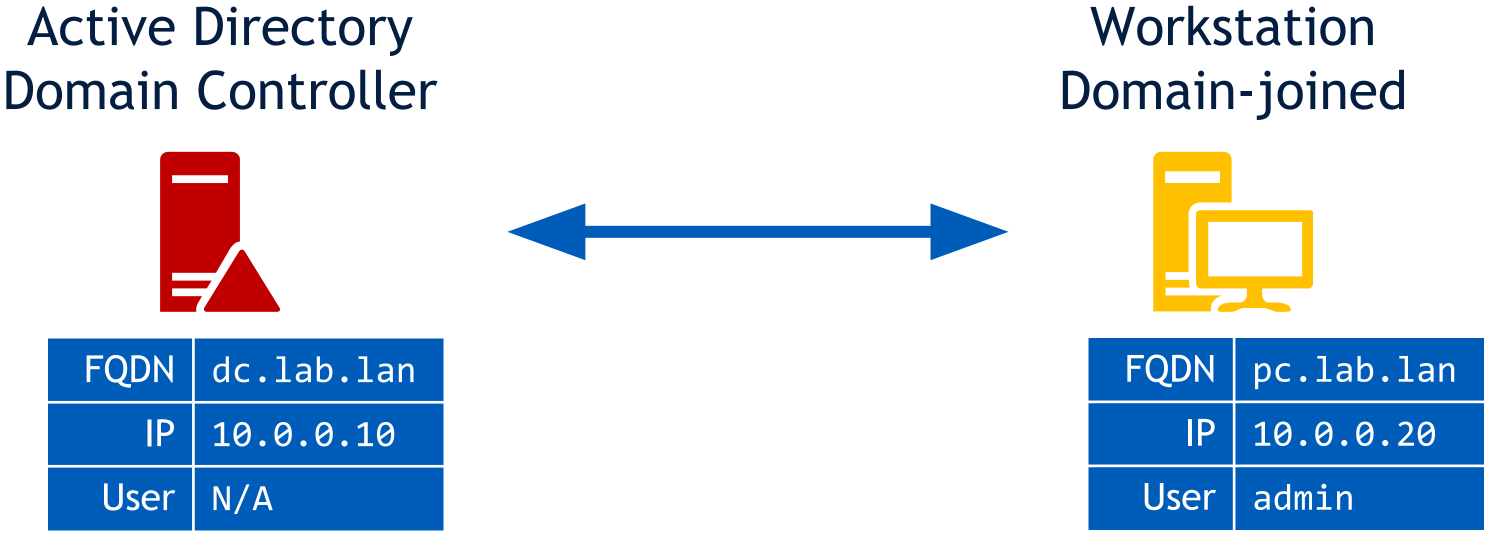
Microsoft "Active Directory“, you said?

- Uses many (previously) proprietary and specific protocols:
  - SMB (previously known as CIFS)
  - MS-RPC, based on the standard “DCE/RPC” as seen in Wireshark
    - [MS-SAMR] Security Account Manager (SAM) Remote Protocol
    - [MS-NRPC] Netlogon Remote Protocol
    - [MS-LSAD] Local Security Authority (Domain Policy) Remote Protocol
    - [MS-GPOL] Group Policy: Core Protocol
    - ...
  - Now published via open specifications

- Other open protocols:
  - LDAP
  - ...

- Open-source implementation: Samba-AD SAMBA Active Directory

Lab setup for all the examples



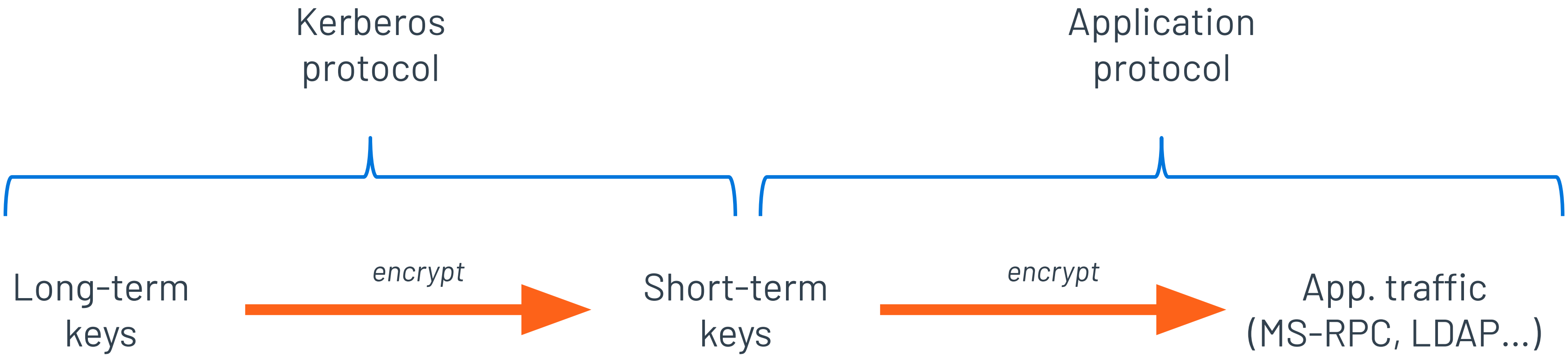
Active Directory
Domain Controller

FQDN	dc.lab.lan
IP	10.0.0.10
User	N/A

Workstation
Domain-joined

FQDN	pc.lab.lan
IP	10.0.0.20
User	admin

Kerberos 101



Kerberos protocol

Application protocol

Long-term keys

encrypt

Short-term keys

encrypt

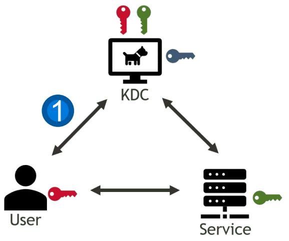
App. traffic (MS-RPC, LDAP...)

🙏 **Wireshark needs**

🙈 **Wireshark can get**

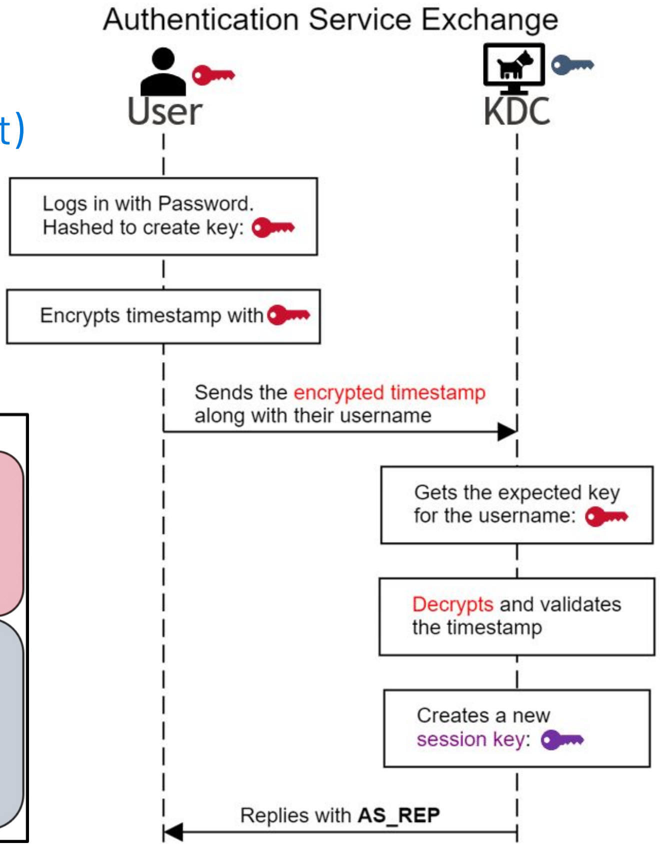
🔓 **Wireshark decrypts**

Kerberos 101



User receives from KDC:

- TGT (Ticket Granting Ticket)
  - Incl. Session key



Logs in with Password. Hashed to create key:

Encrypts timestamp with

Sends the encrypted timestamp along with their username

Gets the expected key for the username:

Decrypts and validates the timestamp

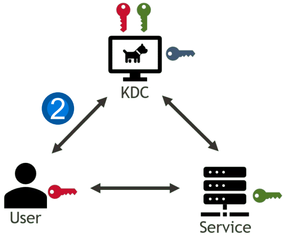
Creates a new session key:

Replies with **AS_REP**

- “cname”: “User”
- “enc-part”: Data encrypted with User’s key
  - Session key:
  - Flags:
    - Forwardable: 1
  - “sname”: “krbtgt”
- Ticket-Granting Ticket: TGT encrypted with KDC’s key
  - Session key:
  - Flags:
    - Forwardable: 1
  - “cname”: “User”
  - PAC: User’s Authorization Data (e.g. Groups)

Source: https://www.netspi.com/blog/technical/network-penetration-testing/cve-2020-17049-kerberos-bronze-bit-theory/

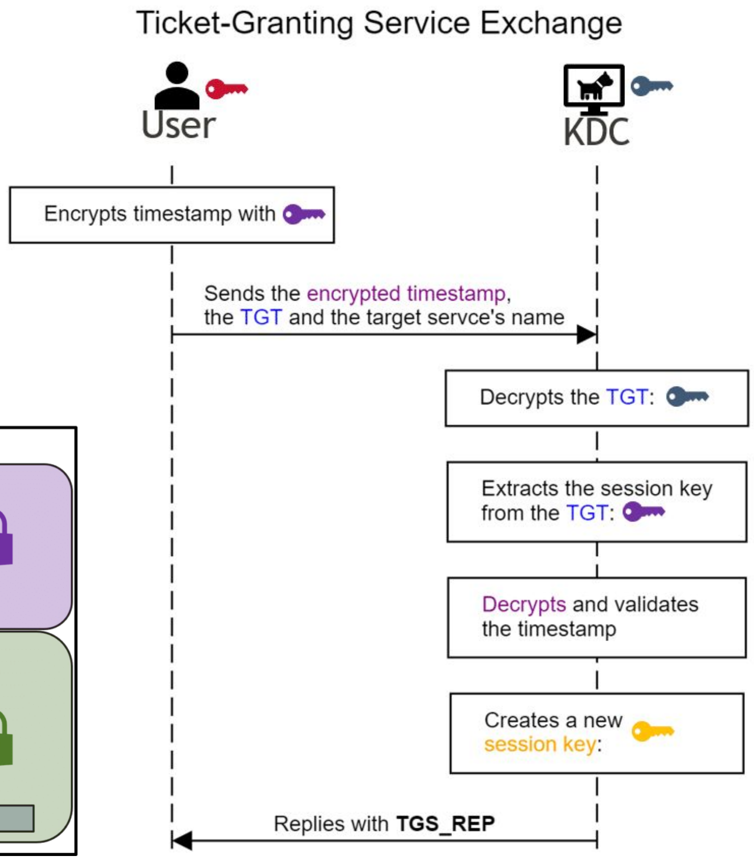
Kerberos 101



User receives from KDC:

- Service Ticket
  - Incl. Session key

- “cname”: “User”
- “enc-part”: Data encrypted with logon session key
  - Session key:
  - Flags:
    - Forwardable: 1
  - “sname”: “Service”
- Service Ticket: Encrypted with Service’s key
  - Session key:
  - Flags:
    - Forwardable: 1
  - “cname”: “User”
  - PAC: User’s Authorization Data (e.g. Groups)



Ticket-Granting Service Exchange

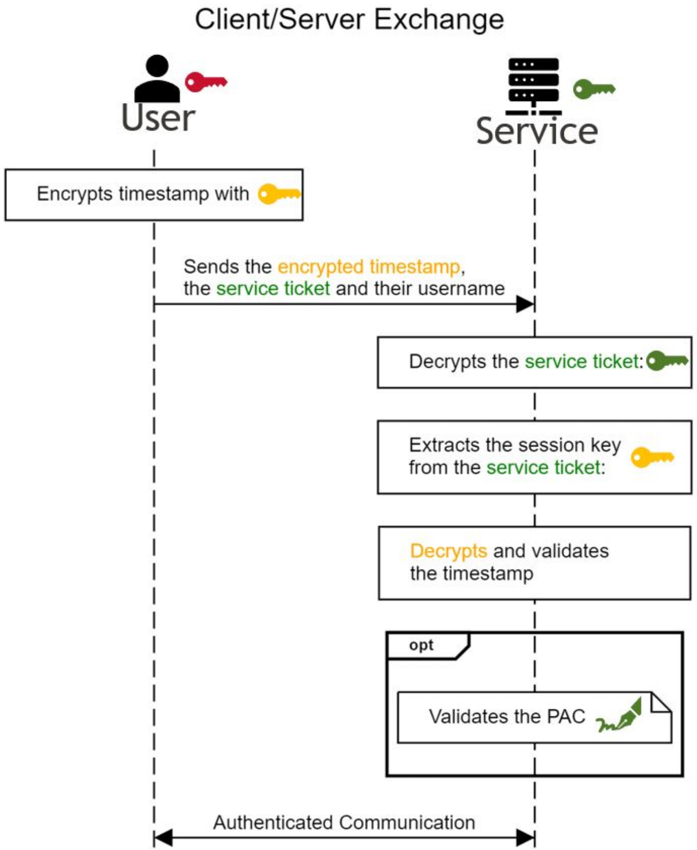
User
KDC
Encrypts timestamp with
Sends the encrypted timestamp, the TGT and the target servce's name
Decrypts the TGT:
Extracts the session key from the TGT:
Decrypts and validates the timestamp
Creates a new session key:
Replies with **TGS_REP**

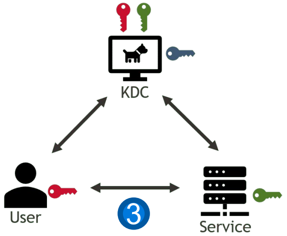
Source: https://www.netspi.com/blog/technical/network-penetration-testing/cve-2020-17049-kerberos-bronze-bit-theory/

Kerberos 101

Users sends to Service:

- Service Ticket
  - Incl. Session key


Client/Server Exchange
User
Service
Encrypts timestamp with
Sends the encrypted timestamp, the service ticket and their username
Decrypts the service ticket:
Extracts the session key from the service ticket:
Decrypts and validates the timestamp
opt
Validates the PAC
Authenticated Communication

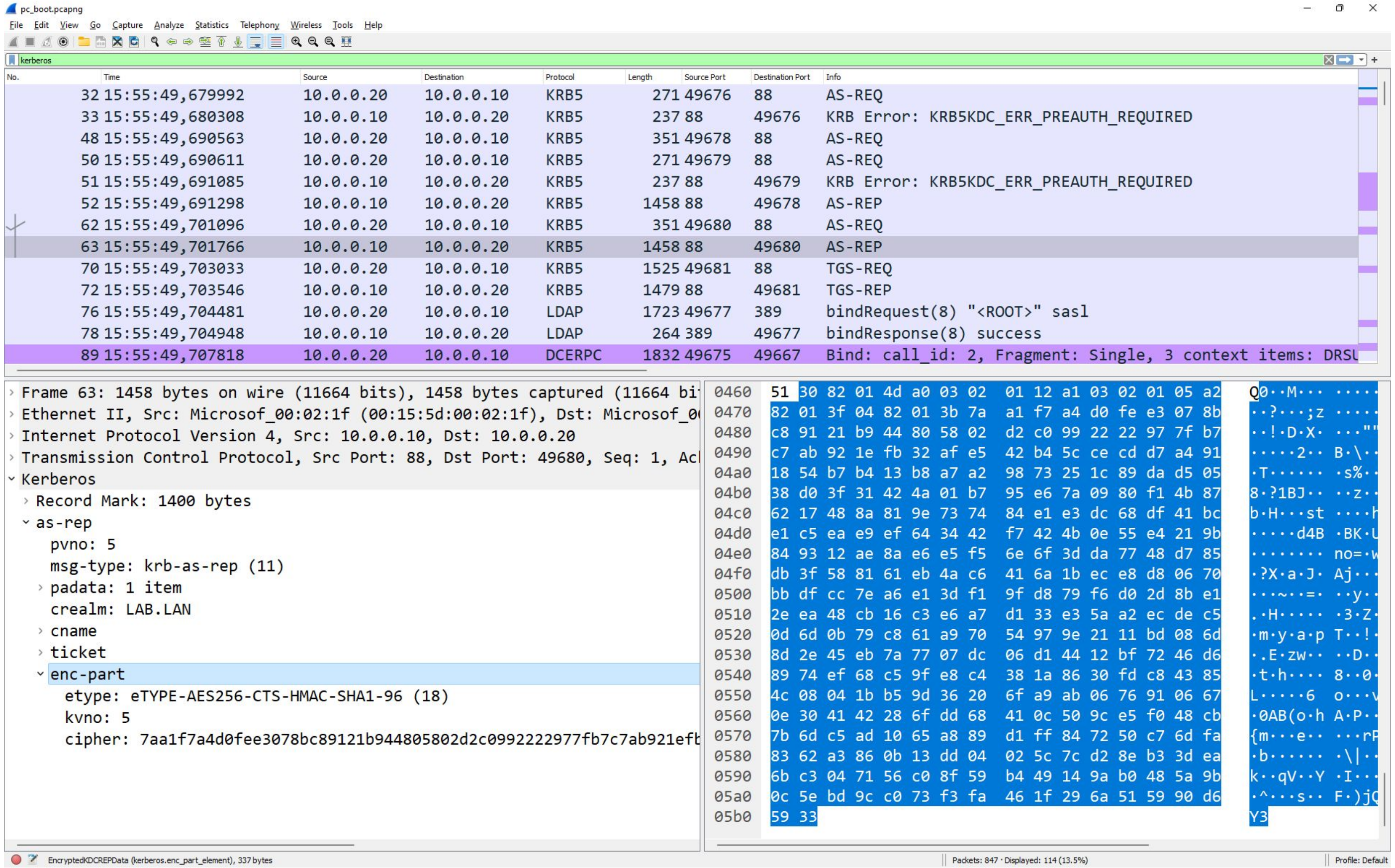


Source: *https://www.netspi.com/blog/technical/network-penetration-testing/cve-2020-17049-kerberos-bronze-bit-theory/*

First look at the capture

Kerberos AS-REP with

🔒 `enc-part`



pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
kerberos

```
No.  Time               Source     Destination Protocol Length Source Port Destination Port Info
  32 15:55:49,679992    10.0.0.20  10.0.0.10   KRB5      271 49676   88     AS-REQ
  33 15:55:49,680308    10.0.0.10  10.0.0.20   KRB5      237 88      49676  KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
  48 15:55:49,690563    10.0.0.20  10.0.0.10   KRB5      351 49678   88     AS-REQ
  50 15:55:49,690611    10.0.0.20  10.0.0.10   KRB5      271 49679   88     AS-REQ
  51 15:55:49,691085    10.0.0.10  10.0.0.20   KRB5      237 88      49679  KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
  52 15:55:49,691298    10.0.0.10  10.0.0.20   KRB5     1458 88      49678  AS-REP
  62 15:55:49,701096    10.0.0.20  10.0.0.10   KRB5      351 49680   88     AS-REQ
  63 15:55:49,701766    10.0.0.10  10.0.0.20   KRB5     1458 88      49680  AS-REP
  70 15:55:49,703033    10.0.0.20  10.0.0.10   KRB5     1525 49681   88     TGS-REQ
  72 15:55:49,703546    10.0.0.10  10.0.0.20   KRB5     1479 88      49681  TGS-REP
  76 15:55:49,704481    10.0.0.20  10.0.0.10   LDAP     1723 49677   389    bindRequest(8) "<ROOT>" sasl
  78 15:55:49,704948    10.0.0.10  10.0.0.20   LDAP      264 389     49677  bindResponse(8) success
  89 15:55:49,707818    10.0.0.20  10.0.0.10   DCERPC   1832 49675   49667  Bind: call_id: 2, Fragment: Single, 3 context items: DRSU
```

```
> Frame 63: 1458 bytes on wire (11664 bits), 1458 bytes captured (11664 bi
> Ethernet II, Src: Microsof_00:02:1f (00:15:5d:00:02:1f), Dst: Microsof_0
> Internet Protocol Version 4, Src: 10.0.0.10, Dst: 10.0.0.20
> Transmission Control Protocol, Src Port: 88, Dst Port: 49680, Seq: 1, Ac
v Kerberos
  > Record Mark: 1400 bytes
  v as-rep
      pvno: 5
      msg-type: krb-as-rep (11)
    > padata: 1 item
      crealm: LAB.LAN
    > cname
    > ticket
    v enc-part
        etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
        kvno: 5
        cipher: 7aa1f7a4d0fee3078bc89121b944805802d2c0992222977fb7c7ab921eft
```

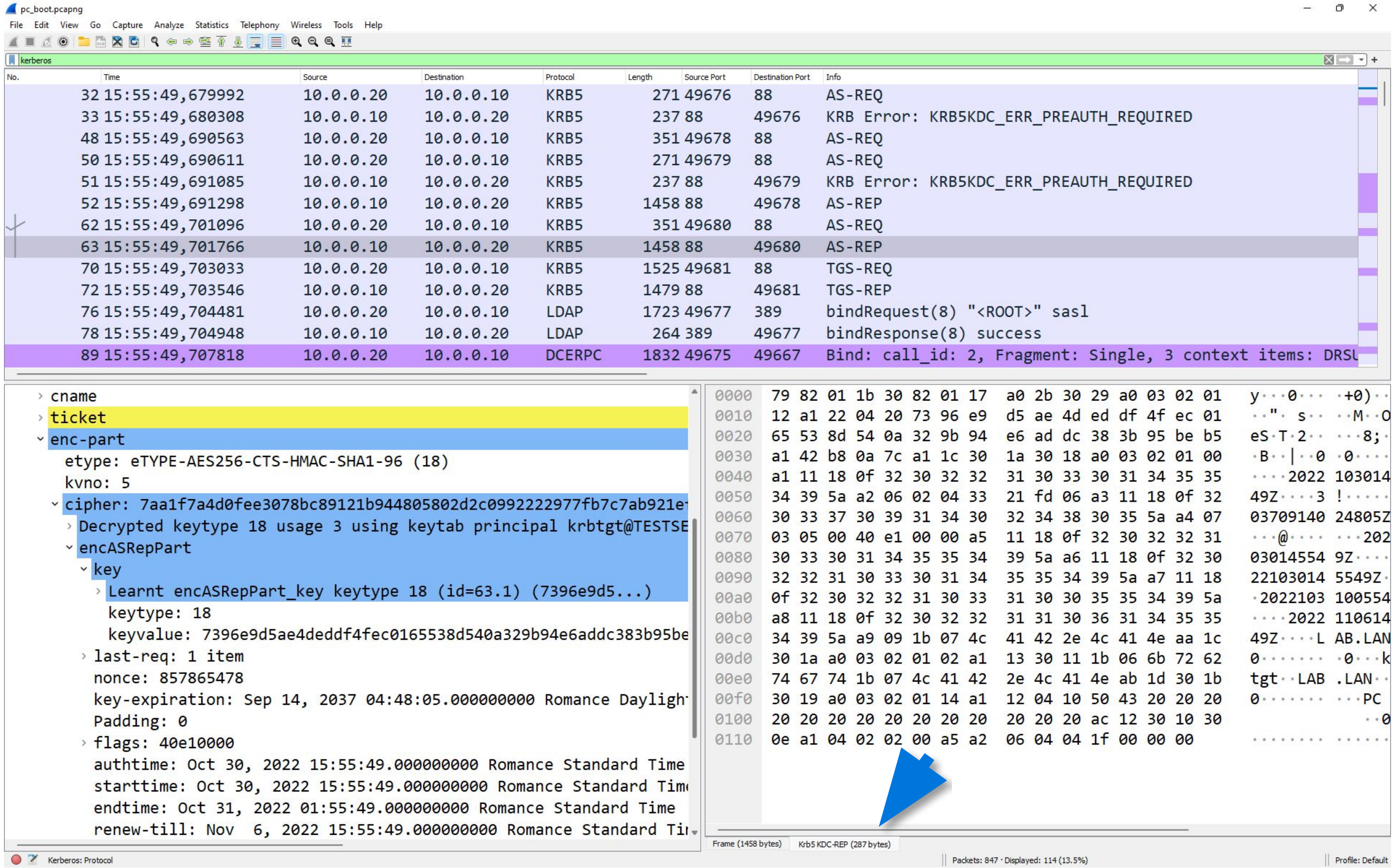
```
0460  51 30 82 01 4d a0 03 02  01 12 a1 03 02 01 05 a2   Q0··M··· ·····
0470  82 01 3f 04 82 01 3b 7a  a1 f7 a4 d0 fe e3 07 8b   ··?···;z ·····
0480  c8 91 21 b9 44 80 58 02  d2 c0 99 22 22 97 7f b7   ··!·D·X· ···""
0490  c7 ab 92 1e fb 32 af e5  42 b4 5c ce cd d7 a4 91   ·····2·· B·\··
04a0  18 54 b7 b4 13 b8 a7 a2  98 73 25 1c 89 da d5 05   ·T······ ·s%··
04b0  38 d0 3f 31 42 4a 01 b7  95 e6 7a 09 80 f1 4b 87   8·?1BJ·· ··z··
04c0  62 17 48 8a 81 9e 73 74  84 e1 e3 dc 68 df 41 bc   b·H···st ····h
04d0  e1 c5 ea e9 ef 64 34 42  f7 42 4b 0e 55 e4 21 9b   ·····d4B ·BK·U
04e0  84 93 12 ae 8a e6 e5 f5  6e 6f 3d da 77 48 d7 85   ········ no=·w
04f0  db 3f 58 81 61 eb 4a c6  41 6a 1b ec e8 d8 06 70   ·?X·a·J· Aj···
0500  bb df cc 7e a6 e1 3d f1  9f d8 79 f6 d0 2d 8b e1   ···~··=· ··y··
0510  2e ea 48 cb 16 c3 e6 a7  d1 33 e3 5a a2 ec de c5   .·H····· ·3·Z·
0520  0d 6d 0b 79 c8 61 a9 70  54 97 9e 21 11 bd 08 6d   ·m·y·a·p T··!·
0530  8d 2e 45 eb 7a 77 07 dc  06 d1 44 12 bf 72 46 d6   ·.E·zw·· ··D··
0540  89 74 ef 68 c5 9f e8 c4  38 1a 86 30 fd c8 43 85   ·t·h···· 8··0·
0550  4c 08 04 1b b5 9d 36 20  6f a9 ab 06 76 91 06 67   L·····6  o···v
0560  0e 30 41 42 28 6f dd 68  41 0c 50 9c e5 f0 48 cb   ·0AB(o·h A·P··
0570  7b 6d c5 ad 10 65 a8 89  d1 ff 84 72 50 c7 6d fa   {m···e·· ···rP
0580  83 62 a3 86 0b 13 dd 04  02 5c 7c d2 8e b3 3d ea   ·b······ ·\|··
0590  6b c3 04 71 56 c0 8f 59  b4 49 14 9a b0 48 5a 9b   k··qV··Y ·I···
05a0  0c 5e bd 9c c0 73 f3 fa  46 1f 29 6a 51 59 90 d6   ·^···s·· F·)jQ
05b0  59 33                                              Y3
```

EncryptedKDCREPData (kerberos.enc_part_element), 337 bytes | Packets: 847 · Displayed: 114 (13.5%) | Profile: Default

New look at decrypted capture

Kerberos AS-REP with

🔓 enc-part



pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

kerberos

```
No.  Time              Source     Destination Protocol Length Source Port Destination Port Info
  32 15:55:49,679992   10.0.0.20  10.0.0.10   KRB5      271 49676   88      AS-REQ
  33 15:55:49,680308   10.0.0.10  10.0.0.20   KRB5      237 88      49676   KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
  48 15:55:49,690563   10.0.0.20  10.0.0.10   KRB5      351 49678   88      AS-REQ
  50 15:55:49,690611   10.0.0.20  10.0.0.10   KRB5      271 49679   88      AS-REQ
  51 15:55:49,691085   10.0.0.10  10.0.0.20   KRB5      237 88      49679   KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
  52 15:55:49,691298   10.0.0.10  10.0.0.20   KRB5     1458 88      49678   AS-REP
  62 15:55:49,701096   10.0.0.20  10.0.0.10   KRB5      351 49680   88      AS-REQ
  63 15:55:49,701766   10.0.0.10  10.0.0.20   KRB5     1458 88      49680   AS-REP
  70 15:55:49,703033   10.0.0.20  10.0.0.10   KRB5     1525 49681   88      TGS-REQ
  72 15:55:49,703546   10.0.0.10  10.0.0.20   KRB5     1479 88      49681   TGS-REP
  76 15:55:49,704481   10.0.0.20  10.0.0.10   LDAP     1723 49677   389     bindRequest(8) "<ROOT>" sasl
  78 15:55:49,704948   10.0.0.10  10.0.0.20   LDAP      264 389     49677   bindResponse(8) success
  89 15:55:49,707818   10.0.0.20  10.0.0.10   DCERPC   1832 49675   49667   Bind: call_id: 2, Fragment: Single, 3 context items: DRSU
```

```
  > cname
  > ticket
  v enc-part
      etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
      kvno: 5
    v cipher: 7aa1f7a4d0fee3078bc89121b944805802d2c0992222977fb7c7ab921e
      > Decrypted keytype 18 usage 3 using keytab principal krbtgt@TESTSE
      v encASRepPart
        v key
          > Learnt encASRepPart_key keytype 18 (id=63.1) (7396e9d5...)
            keytype: 18
            keyvalue: 7396e9d5ae4deddf4fec0165538d540a329b94e6addc383b95be
        > last-req: 1 item
          nonce: 857865478
          key-expiration: Sep 14, 2037 04:48:05.000000000 Romance Dayligh
          Padding: 0
        > flags: 40e10000
          authtime: Oct 30, 2022 15:55:49.000000000 Romance Standard Time
          starttime: Oct 30, 2022 15:55:49.000000000 Romance Standard Tim
          endtime: Oct 31, 2022 01:55:49.000000000 Romance Standard Time
          renew-till: Nov  6, 2022 15:55:49.000000000 Romance Standard Ti
```

```
0000  79 82 01 1b 30 82 01 17  a0 2b 30 29 a0 03 02 01   y···0··· ·+0)··
0010  12 a1 22 04 20 73 96 e9  d5 ae 4d ed df 4f ec 01   ··"· s·· ··M··O
0020  65 53 8d 54 0a 32 9b 94  e6 ad dc 38 3b 95 be b5   eS·T·2·· ···8;·
0030  a1 42 b8 0a 7c a1 1c 30  1a 30 18 a0 03 02 01 00   ·B··|··0 ·0····
0040  a1 11 18 0f 32 30 32 32  31 30 33 30 31 34 35 35   ····2022 103014
0050  34 39 5a a2 06 02 04 33  21 fd 06 a3 11 18 0f 32   49Z····3 !·····
0060  30 33 37 30 39 31 34 30  32 34 38 30 35 5a a4 07   03709140 24805Z
0070  03 05 00 40 e1 00 00 a5  11 18 0f 32 30 32 32 31   ···@···· ···202
0080  30 33 30 31 34 35 35 34  39 5a a6 11 18 0f 32 30   03014554 9Z····
0090  32 32 31 30 33 30 31 34  35 35 34 39 5a a7 11 18   22103014 5549Z·
00a0  0f 32 30 32 32 31 30 33  31 30 30 35 35 34 39 5a   ·2022103 100554
00b0  a8 11 18 0f 32 30 32 32  31 31 30 36 31 34 35 35   ····2022 110614
00c0  34 39 5a a9 09 1b 07 4c  41 42 2e 4c 41 4e aa 1c   49Z····L AB.LAN
00d0  30 1a a0 03 02 01 02 a1  13 30 11 1b 06 6b 72 62   0······· ·0···k
00e0  74 67 74 1b 07 4c 41 42  2e 4c 41 4e ab 1d 30 1b   tgt··LAB .LAN··
00f0  30 19 a0 03 02 01 14 a1  12 04 10 50 43 20 20 20   0······· ···PC
0100  20 20 20 20 20 20 20 20  20 20 20 ac 12 30 10 30            ··0
0110  0e a1 04 02 02 00 a5 a2  06 04 04 1f 00 00 00      ········ ······
```

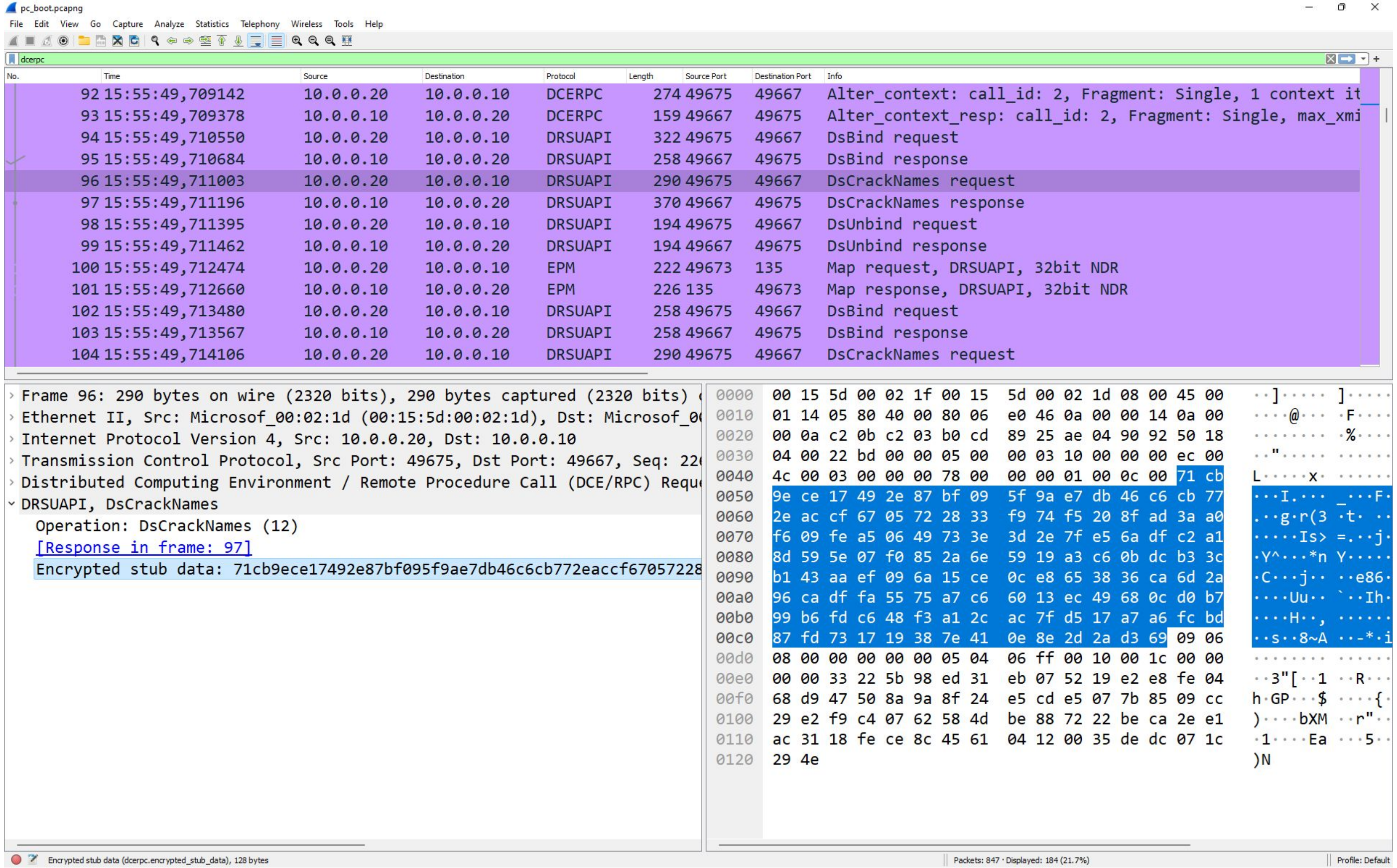
Frame (1458 bytes)   Krb5 KDC-REP (287 bytes)

Kerberos: Protocol   Packets: 847 · Displayed: 114 (13.5%)   Profile: Default

First look at the capture

DRSUAPI `DsCrackNames` with

🔒 `encrypted stub data`



pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
dcerpc

No.  Time             Source     Destination Protocol Length Source Port Destination Port Info
  92 15:55:49,709142  10.0.0.20  10.0.0.10   DCERPC      274 49675       49667            Alter_context: call_id: 2, Fragment: Single, 1 context it
  93 15:55:49,709378  10.0.0.10  10.0.0.20   DCERPC      159 49667       49675            Alter_context_resp: call_id: 2, Fragment: Single, max_xmi
  94 15:55:49,710550  10.0.0.20  10.0.0.10   DRSUAPI     322 49675       49667            DsBind request
  95 15:55:49,710684  10.0.0.10  10.0.0.20   DRSUAPI     258 49667       49675            DsBind response
  96 15:55:49,711003  10.0.0.20  10.0.0.10   DRSUAPI     290 49675       49667            DsCrackNames request
  97 15:55:49,711196  10.0.0.10  10.0.0.20   DRSUAPI     370 49667       49675            DsCrackNames response
  98 15:55:49,711395  10.0.0.20  10.0.0.10   DRSUAPI     194 49675       49667            DsUnbind request
  99 15:55:49,711462  10.0.0.10  10.0.0.20   DRSUAPI     194 49667       49675            DsUnbind response
 100 15:55:49,712474  10.0.0.20  10.0.0.10   EPM         222 49673       135              Map request, DRSUAPI, 32bit NDR
 101 15:55:49,712660  10.0.0.10  10.0.0.20   EPM         226 135         49673            Map response, DRSUAPI, 32bit NDR
 102 15:55:49,713480  10.0.0.20  10.0.0.10   DRSUAPI     258 49675       49667            DsBind request
 103 15:55:49,713567  10.0.0.10  10.0.0.20   DRSUAPI     258 49667       49675            DsBind response
 104 15:55:49,714106  10.0.0.20  10.0.0.10   DRSUAPI     290 49675       49667            DsCrackNames request
```

```
> Frame 96: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits) (
> Ethernet II, Src: Microsof_00:02:1d (00:15:5d:00:02:1d), Dst: Microsof_0(
> Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.0.10
> Transmission Control Protocol, Src Port: 49675, Dst Port: 49667, Seq: 22(
> Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Reque
v DRSUAPI, DsCrackNames
    Operation: DsCrackNames (12)
    [Response in frame: 97]
    Encrypted stub data: 71cb9ece17492e87bf095f9ae7db46c6cb772eaccf67057228
```

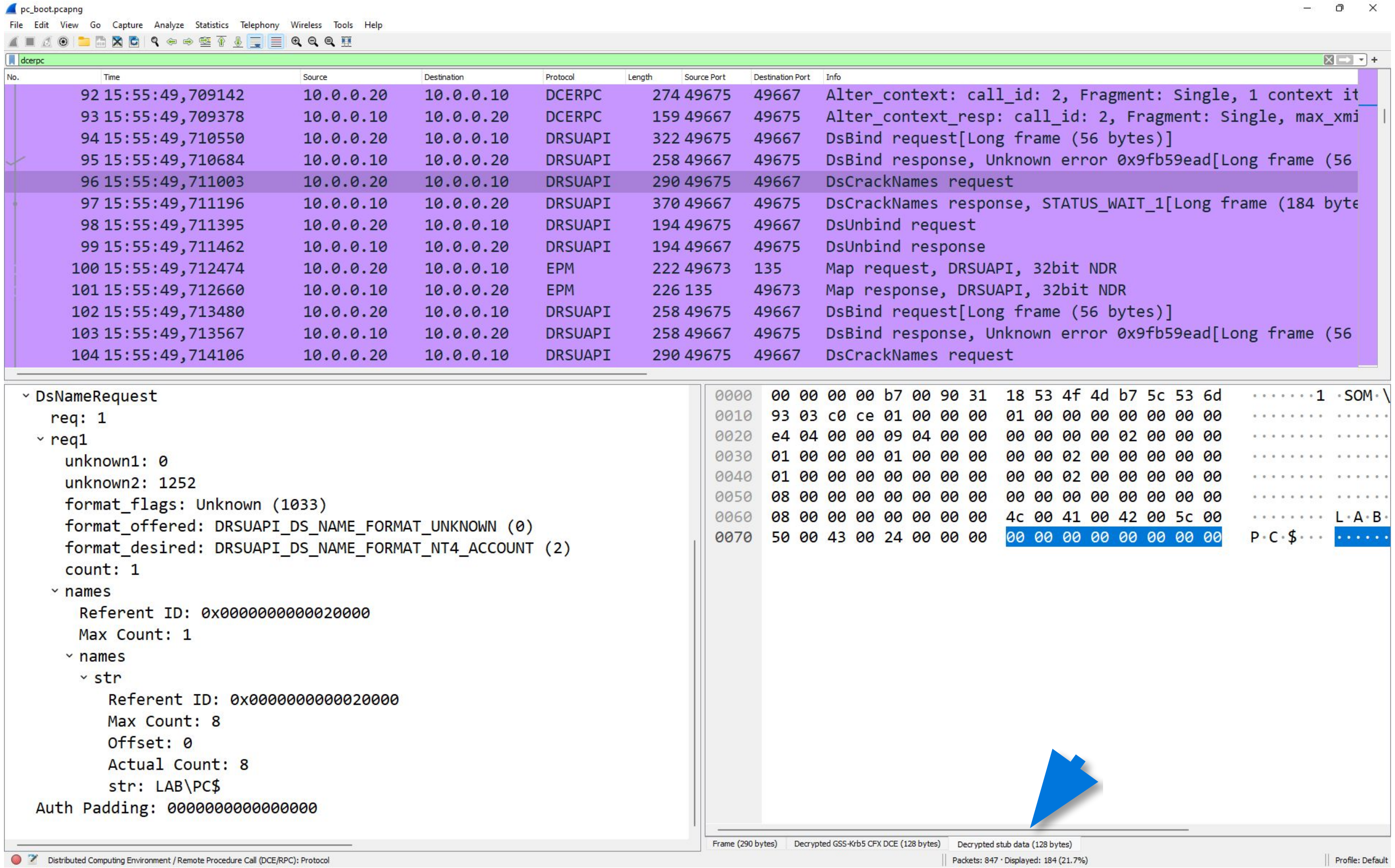
```
0000  00 15 5d 00 02 1f 00 15  5d 00 02 1d 08 00 45 00   ··]····· ]·····
0010  01 14 05 80 40 00 80 06  e0 46 0a 00 00 14 0a 00   ····@··· ·F····
0020  00 0a c2 0b c2 03 b0 cd  89 25 ae 04 90 92 50 18   ········ ·%····
0030  04 00 22 bd 00 00 05 00  00 03 10 00 00 00 ec 00   ··"····· ·······
0040  4c 00 03 00 00 00 78 00  00 00 01 00 0c 00 71 cb   L·····x· ·······
0050  9e ce 17 49 2e 87 bf 09  5f 9a e7 db 46 c6 cb 77   ···I.··· _···F·
0060  2e ac cf 67 05 72 28 33  f9 74 f5 20 8f ad 3a a0   .··g·r(3 ·t· ··
0070  f6 09 fe a5 06 49 73 3e  3d 2e 7f e5 6a df c2 a1   ·····Is> =.··j·
0080  8d 59 5e 07 f0 85 2a 6e  59 19 a3 c6 0b dc b3 3c   ·Y^···*n Y·····
0090  b1 43 aa ef 09 6a 15 ce  0c e8 65 38 36 ca 6d 2a   ·C···j·· ··e86·
00a0  96 ca df fa 55 75 a7 c6  60 13 ec 49 68 0c d0 b7   ····Uu·· `··Ih·
00b0  99 b6 fd c6 48 f3 a1 2c  ac 7f d5 17 a7 a6 fc bd   ····H··, ······
00c0  87 fd 73 17 19 38 7e 41  0e 8e 2d 2a d3 69 09 06   ··s··8~A ··-*·i
00d0  08 00 00 00 00 00 05 04  06 ff 00 10 00 1c 00 00   ········ ·······
00e0  00 00 33 22 5b 98 ed 31  eb 07 52 19 e2 e8 fe 04   ··3"[··1 ··R···
00f0  68 d9 47 50 8a 9a 8f 24  e5 cd e5 07 7b 85 09 cc   h·GP···$ ····{·
0100  29 e2 f9 c4 07 62 58 4d  be 88 72 22 be ca 2e e1   )····bXM ··r"··
0110  ac 31 18 fe ce 8c 45 61  04 12 00 35 de dc 07 1c   ·1····Ea ···5··
0120  29 4e                                              )N
```

Encrypted stub data (dcerpc.encrypted_stub_data), 128 bytes | Packets: 847 · Displayed: 184 (21.7%) | Profile: Default

New look at decrypted capture

DRSUAPI `DsCrackNames` with

🔓 `decrypted stub data`



pc_boot.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dcerpc

```
No.  Time             Source     Destination Protocol Length Source Port Destination Port Info
  92 15:55:49,709142  10.0.0.20  10.0.0.10   DCERPC      274 49675  49667  Alter_context: call_id: 2, Fragment: Single, 1 context it
  93 15:55:49,709378  10.0.0.10  10.0.0.20   DCERPC      159 49667  49675  Alter_context_resp: call_id: 2, Fragment: Single, max_xmi
  94 15:55:49,710550  10.0.0.20  10.0.0.10   DRSUAPI     322 49675  49667  DsBind request[Long frame (56 bytes)]
  95 15:55:49,710684  10.0.0.10  10.0.0.20   DRSUAPI     258 49667  49675  DsBind response, Unknown error 0x9fb59ead[Long frame (56
  96 15:55:49,711003  10.0.0.20  10.0.0.10   DRSUAPI     290 49675  49667  DsCrackNames request
  97 15:55:49,711196  10.0.0.10  10.0.0.20   DRSUAPI     370 49667  49675  DsCrackNames response, STATUS_WAIT_1[Long frame (184 byte
  98 15:55:49,711395  10.0.0.20  10.0.0.10   DRSUAPI     194 49675  49667  DsUnbind request
  99 15:55:49,711462  10.0.0.10  10.0.0.20   DRSUAPI     194 49667  49675  DsUnbind response
 100 15:55:49,712474  10.0.0.20  10.0.0.10   EPM         222 49673  135    Map request, DRSUAPI, 32bit NDR
 101 15:55:49,712660  10.0.0.10  10.0.0.20   EPM         226 135    49673  Map response, DRSUAPI, 32bit NDR
 102 15:55:49,713480  10.0.0.20  10.0.0.10   DRSUAPI     258 49675  49667  DsBind request[Long frame (56 bytes)]
 103 15:55:49,713567  10.0.0.10  10.0.0.20   DRSUAPI     258 49667  49675  DsBind response, Unknown error 0x9fb59ead[Long frame (56
 104 15:55:49,714106  10.0.0.20  10.0.0.10   DRSUAPI     290 49675  49667  DsCrackNames request
```

```
⌄ DsNameRequest
    req: 1
  ⌄ req1
      unknown1: 0
      unknown2: 1252
      format_flags: Unknown (1033)
      format_offered: DRSUAPI_DS_NAME_FORMAT_UNKNOWN (0)
      format_desired: DRSUAPI_DS_NAME_FORMAT_NT4_ACCOUNT (2)
      count: 1
    ⌄ names
        Referent ID: 0x0000000000020000
        Max Count: 1
      ⌄ names
        ⌄ str
            Referent ID: 0x0000000000020000
            Max Count: 8
            Offset: 0
            Actual Count: 8
            str: LAB\PC$
  Auth Padding: 0000000000000000
```

```
0000  00 00 00 00 b7 00 90 31  18 53 4f 4d b7 5c 53 6d   ·······1 ·SOM·\
0010  93 03 c0 ce 01 00 00 00  01 00 00 00 00 00 00 00   ········ ······
0020  e4 04 00 00 09 04 00 00  00 00 00 00 02 00 00 00   ········ ······
0030  01 00 00 00 01 00 00 00  00 00 02 00 00 00 00 00   ········ ······
0040  01 00 00 00 00 00 00 00  00 00 02 00 00 00 00 00   ········ ······
0050  08 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ······
0060  08 00 00 00 00 00 00 00  4c 00 41 00 42 00 5c 00   ········ L·A·B·
0070  50 00 43 00 24 00 00 00  00 00 00 00 00 00 00 00   P·C·$··· ······
```

Frame (290 bytes)
Decrypted GSS-Krb5 CFX DCE (128 bytes)
Decrypted stub data (128 bytes)
Distributed Computing Environment / Remote Procedure Call (DCE/RPC): Protocol
Packets: 847 · Displayed: 184 (21.7%)
Profile: Default