Adventures in MitM-land

Using MITM to Attack Active Directory Authentication Schemes

About Us

Sagi Sheinfeld (@sagish1233)

- Senior Engineer @CrowdStrike (Former @Preempt)
- Extensive background as a security researcher

Eyal Karni (@eyal_karni)

- Senior Engineer @CrowdStrike (Former @Preempt)
- Previously presented on Black Hat

Yaron Zinar (@YaronZi)

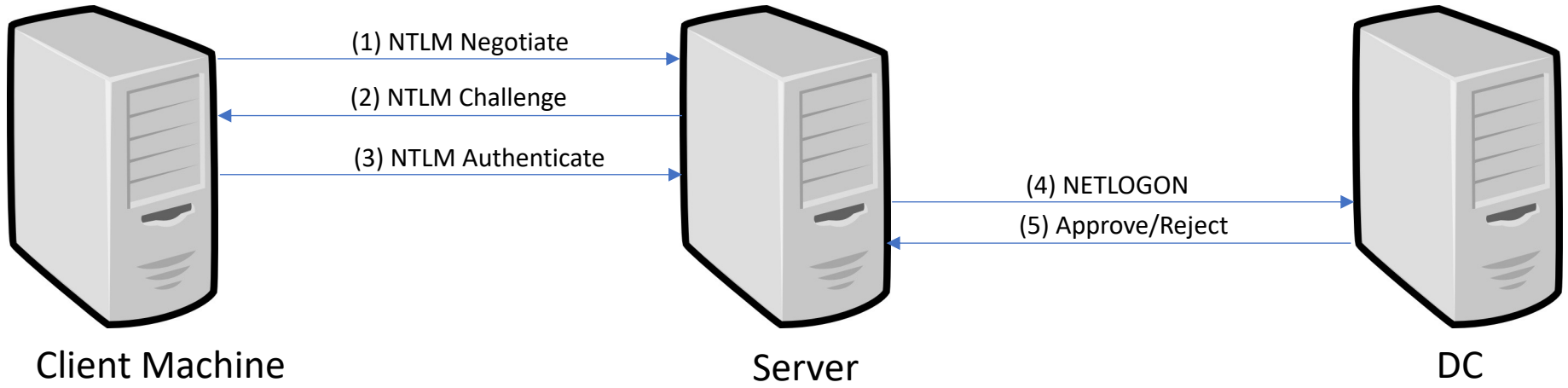
- Senior Manager, Engineering @CrowdStrike
- 2xBlack Hat, 1xDEFCON

Is MITM an Important Technique?

- Sometimes…
  - Works when other techniques fail
  - Often overlooked…

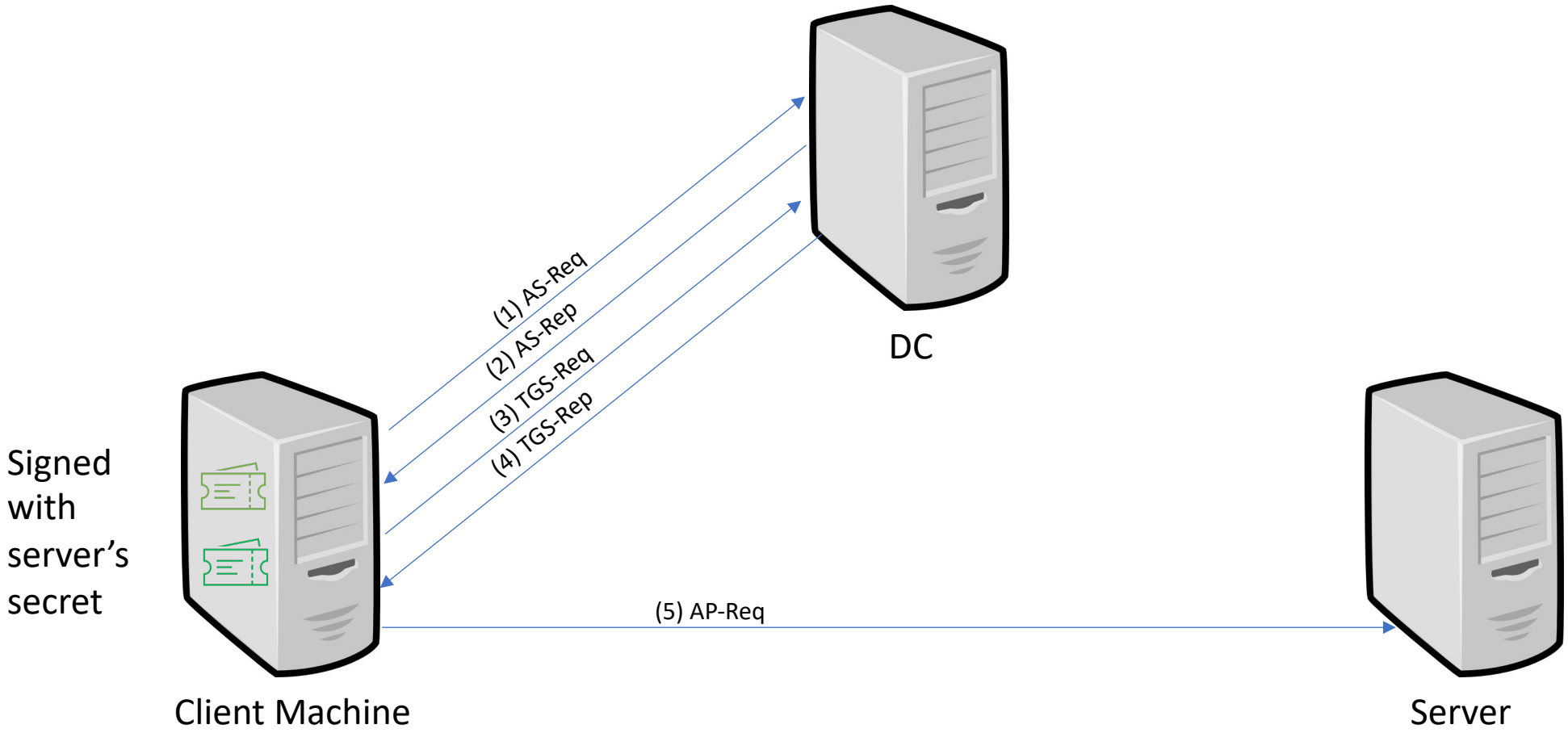
- Active Directory
  - Relatively old protocols
  - Usually don’t use TLS

NTLM Basics



Authentication is *not* bound to the target server!

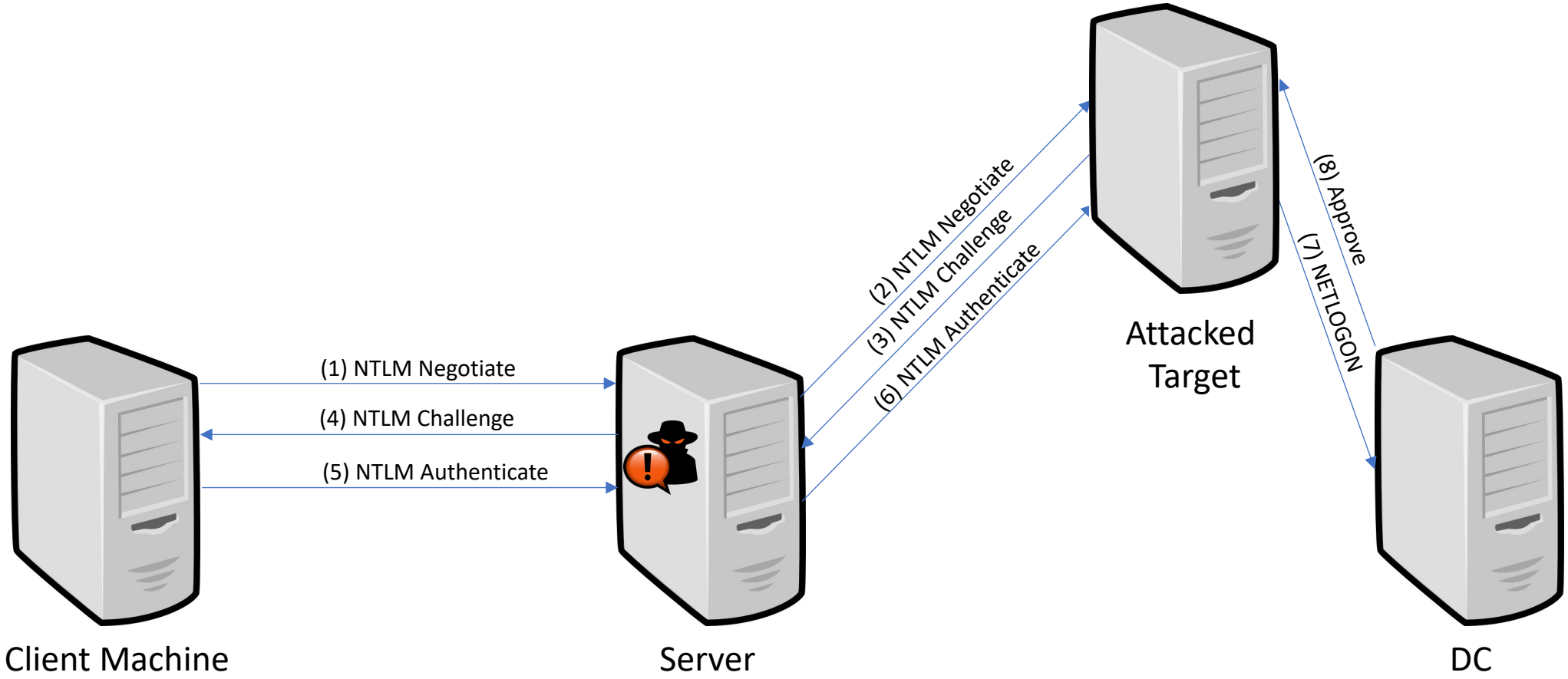
Kerberos Basics



Kerberos vs. NTLM

	NTLM	Kerberos
Proteted from Offline Cracking	X	✓ (except X-roasting)
Can Work w/o Storing Hash in RAM	X	✓
Supports Mutual Authentication	X	✓
Smart Card Support	X	✓
Hashes Contain Salt	X	✓ (except RC4)

NTLM Relay 101



NTLM Relay over DCE/RPC

- **First suggested by Sylvain Heiniger (@sploutchy)**
- Found (at least) one interface (TSCH) with no server signing
- Used NTLM Relay to create a new scheduled task

DCE/RPC Relay Mitigation is Broken

Name	Value	Meaning
RPC_C_AUTHN_LEVEL_DEFAULT	0x00	Same as RPC_C_AUTHN_LEVEL_CONNECT
RPC_C_AUTHN_LEVEL_NONE	0x01	No authentication.
RPC_C_AUTHN_LEVEL_CONNECT	0x02	Authenticates the credentials of the client and server.
RPC_C_AUTHN_LEVEL_CALL	0x03	Same as RPC_C_AUTHN_LEVEL_PKT.
RPC_C_AUTHN_LEVEL_PKT	0x04	Same as RPC_C_AUTHN_LEVEL_CONNECT but also prevents replay attacks.
RPC_C_AUTHN_LEVEL_PKT_INTEGRITY	0x05	Same as RPC_C_AUTHN_LEVEL_PKT but also verifies that none of the data transferred between the client and server has been modified.
RPC_C_AUTHN_LEVEL_PKT_PRIVACY	0x06	Same as RPC_C_AUTHN_LEVEL_PKT_INTEGRITY but also ensures that the data transferred can only be seen unencrypted by the client and the server.

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rpce/425a7c53-c33a-4868-8e5b-2a850d40dc73

Printer Spooler LPE (CVE-2020-1048)

- **Discovered by Peleg Hadar (@peleghd) and Tomer Bar**

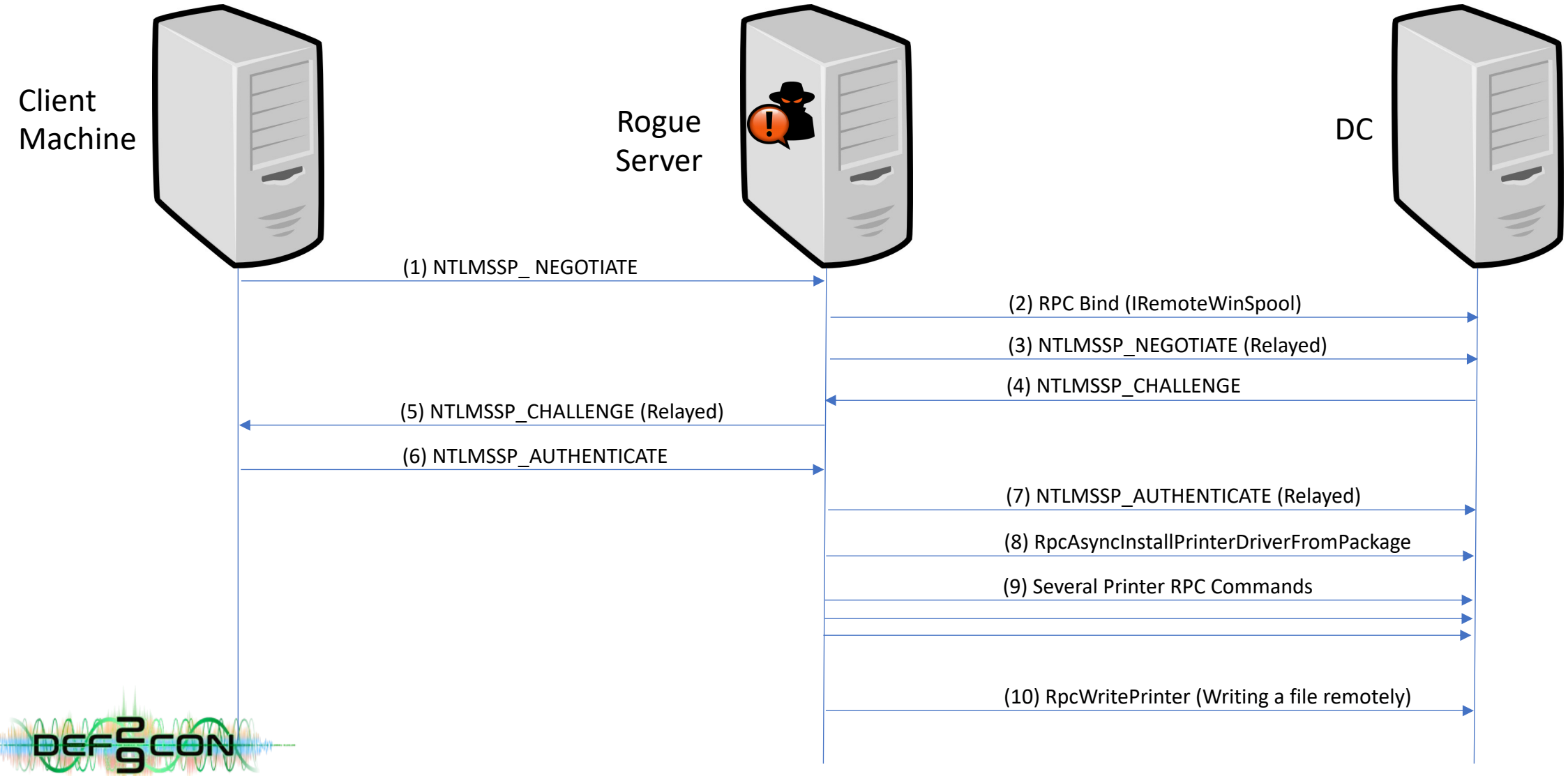
- For printing, you need a driver, and a port
- Any user can install a printer driver (from a pre-existing list)
  - “Generic/ Text” can write anything...
- The port can be a file instead ⇒ **We can write arbitrary files**

- It is a privileged process, and the access checks are done on the client side ⇒ **We have an LPE**

CVE-2021-1678

- Was found using our RPC scanning tool
- Targeting MS-PAR (IRemoteWinSpool) Interface
  - Interface has only required RPC_C_AUTHN_LEVEL_CONNECT
  - Support remote printer operations
- Works the same as CVE-2020-1048 (just remotely)
  - RpcAsyncInstallPrinterDriverFromPackage (Opnum 62) — Installing “Generic/Text” printer driver
  - RpcAsyncOpenPrinter (Opnum 0)
  - RpcAsyncXcvData (Opnum 33) — Add port
  - RpcAsyncAddPrinter (Opnum 1) — Add a printer with the mentioned driver
  - RpcAsyncStartDocPrinter(Opnum 10) — Start a new document
  - RpcAsyncWritePrinter (Opnum 12) — Write to new document

CVE-2021-1678



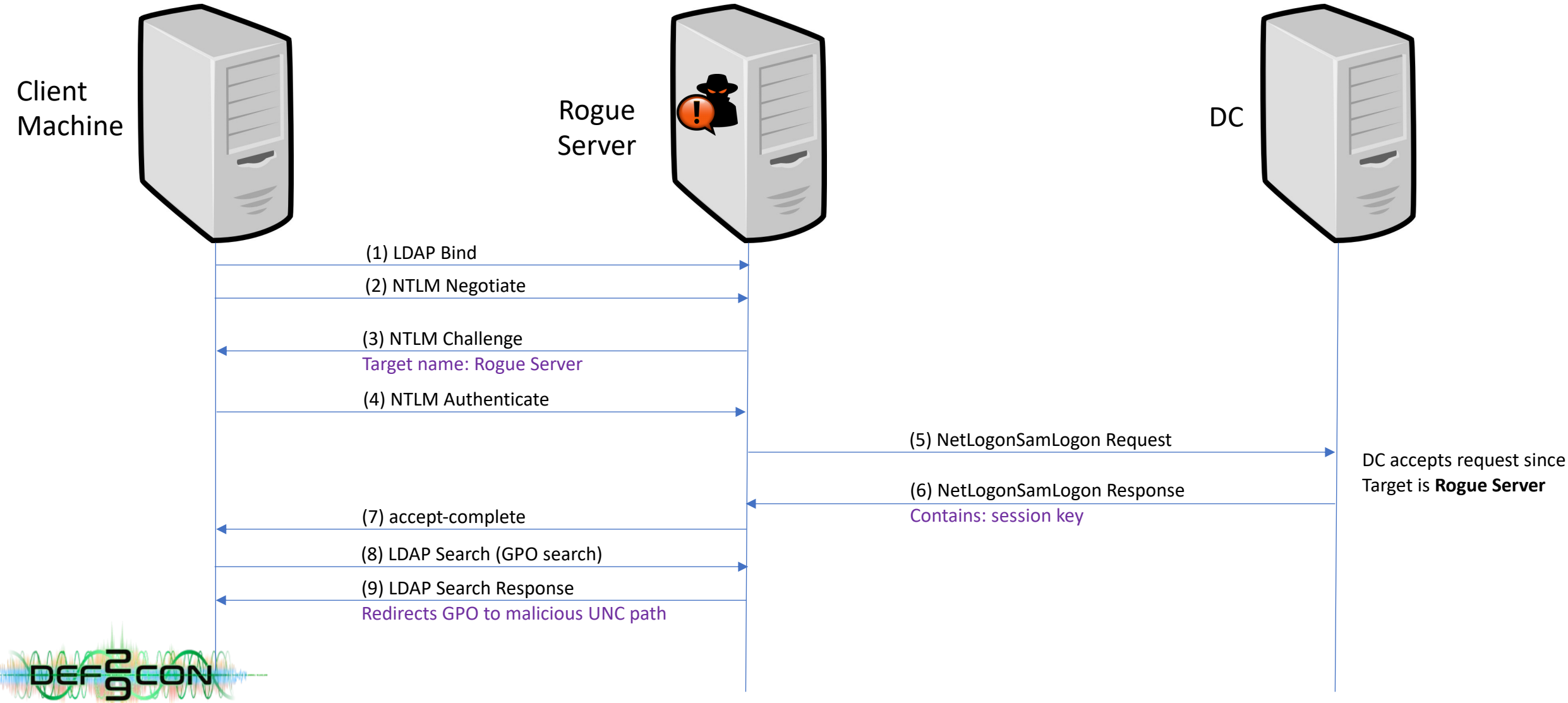
DEMO

MS15-011

- **Initially discovered by Luke Jennings (@jukelennings)**

- Attacking GPO retrieval using MITM
  - Many attack scenarios
  - Both RCE and privilege escalation
  - Some scenarios are still exploitable

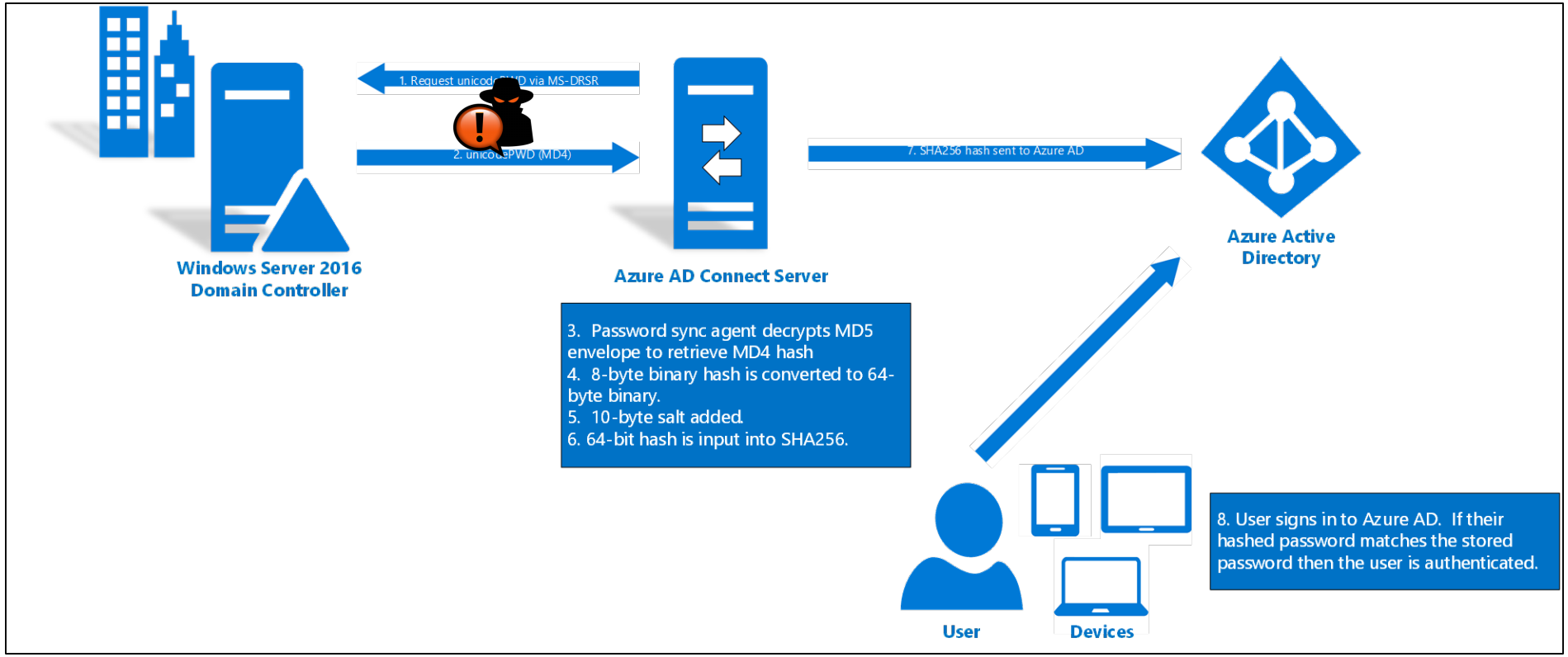
MS15-011 Explained



MS Fixes for MS15-011

- GPO retreival can no longer operate with NTLM
  - Registry Key
- Hardened UNC Paths
  - Configuration to block NTLM usage in SMB
  - Defaults
    - *\SYSVOL
    - *\NETLOGON

Azure AD Connect



https://docs.microsoft.com/en-us/azure/active-directory/hybrid/how-to-connect-password-hash-synchronization

Azure AD Connect MITM Attack

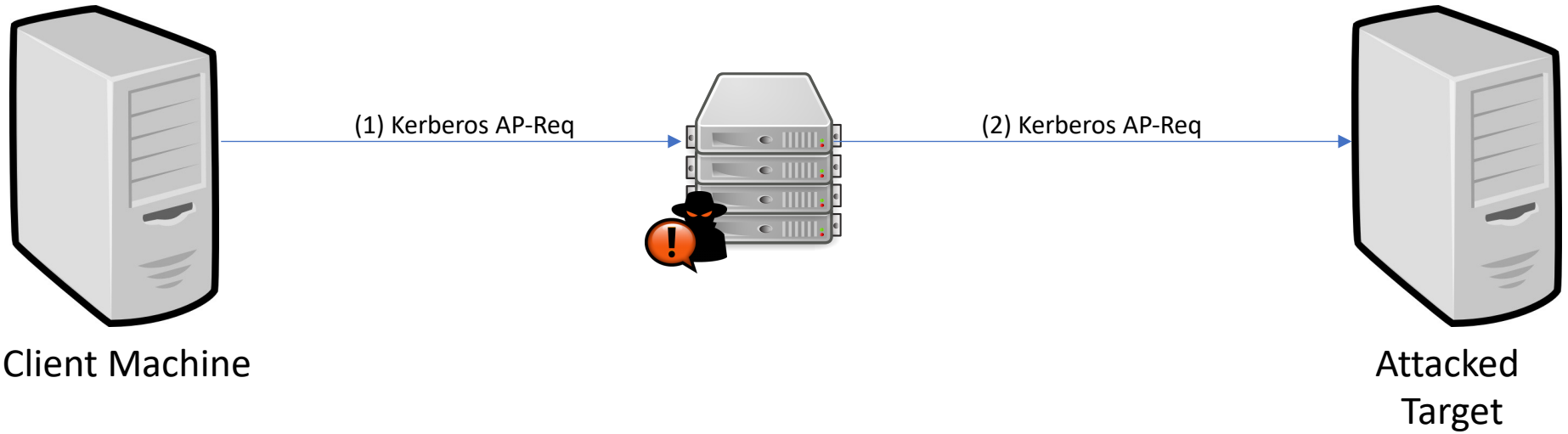
- MITM Between Azure AD Connect and DC

- Attack Steps:
  - Establish a full MITM , make Kerberos fail while allowing LDAP to pass to the DC
  - Wait for domain replication in NTLM
  - Inject new change MD4 password for an account of your choice
  - Log in to Azure AD with injected password

DEMO

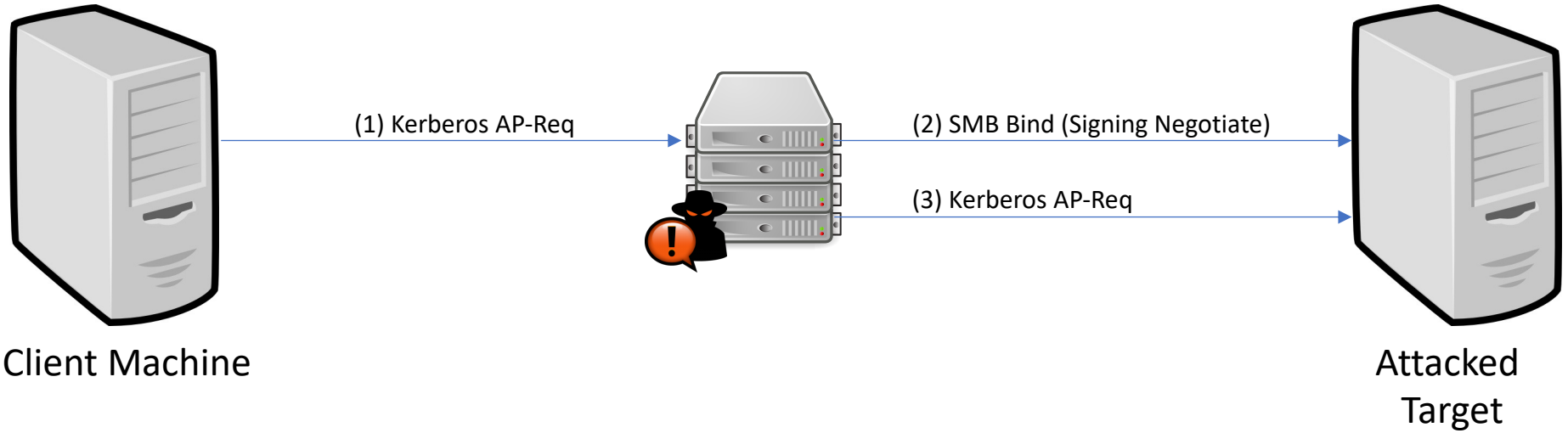
Kerberos Relay

- Same as NTLM Relay (actually much easier)
  - Just to the original target server



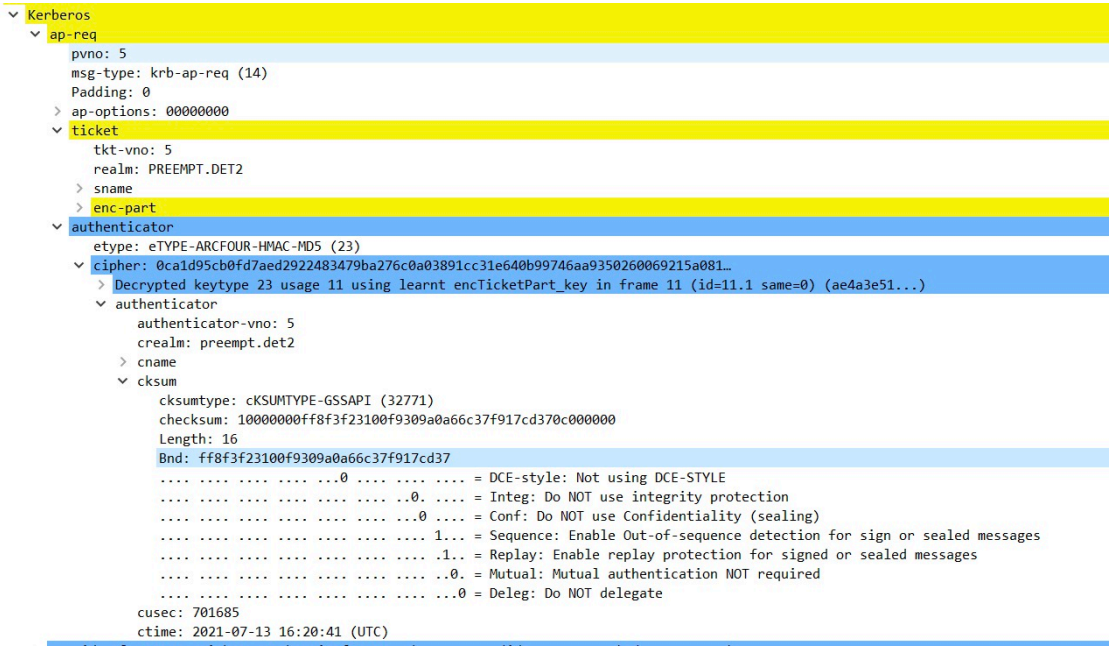
Kerberos Relay – Cont.

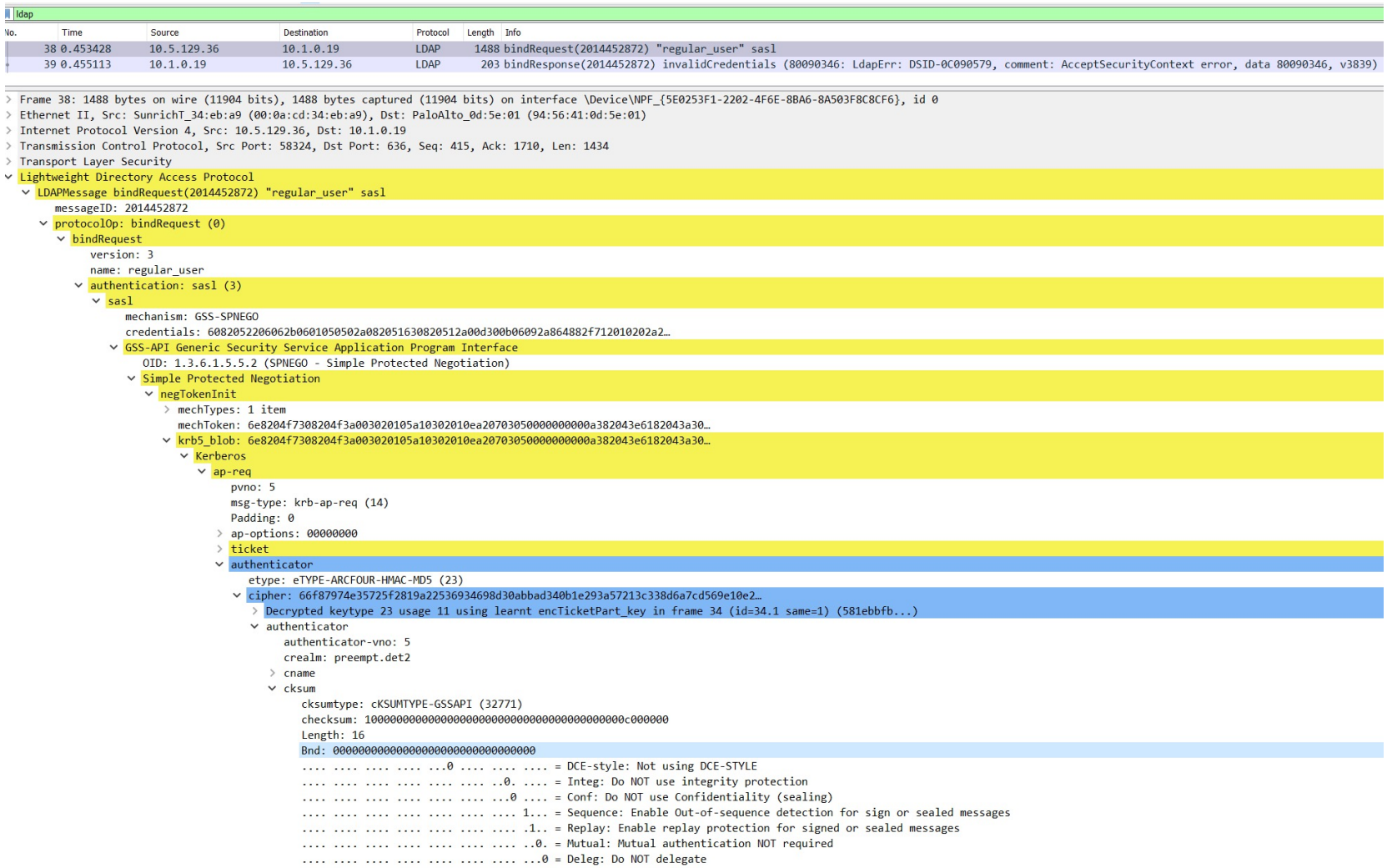
- SMB Relay
  - Works the same as with NTLM
  - Attacker can negotiate no signing if server signing is not required (default)

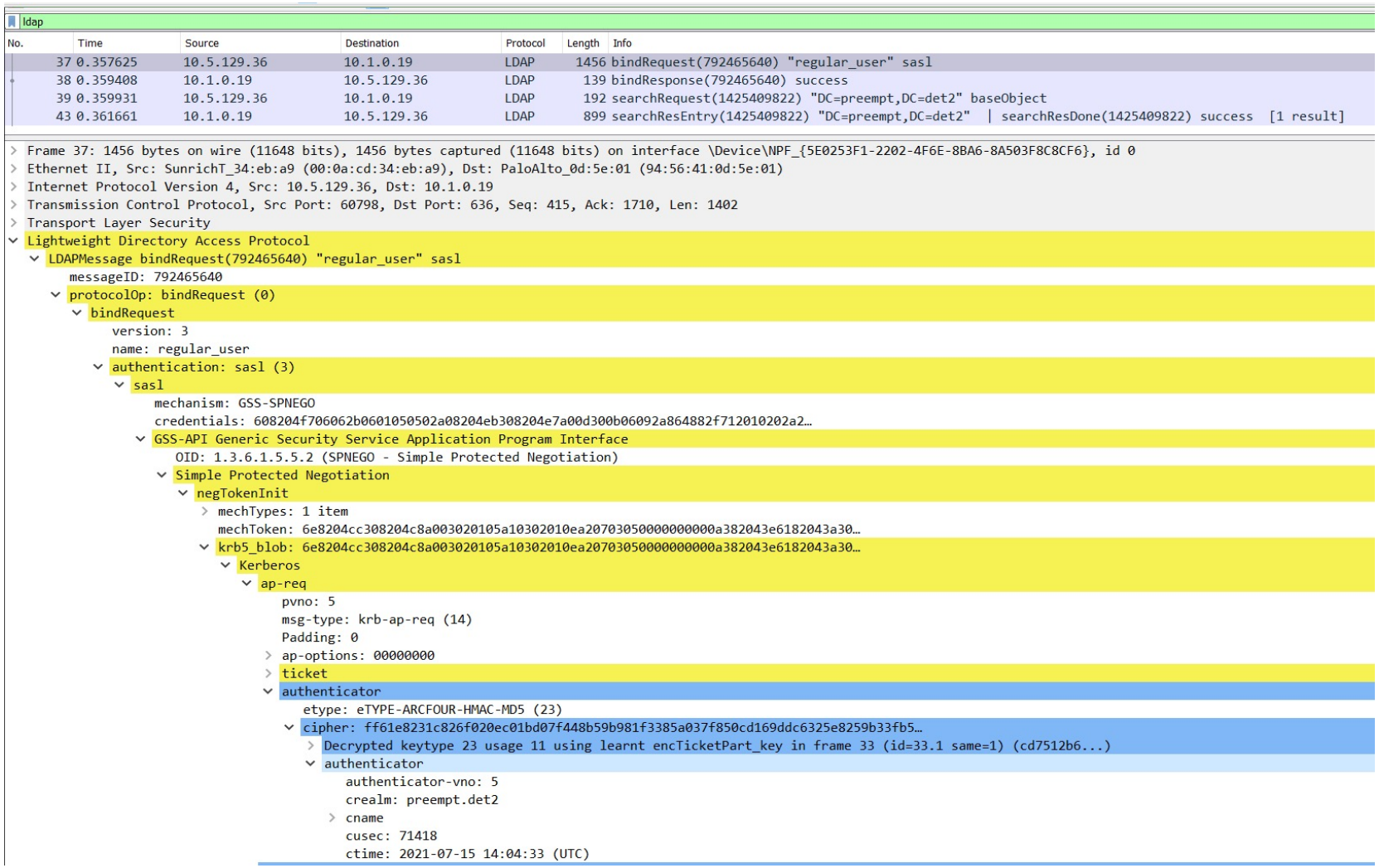


Kerberos Relay over TLS

- Relay protection in TLS channels
  - Extended Protection for Authentication
- Important examples of such protection
  - LDAPS (called Channel Bindings)
  - AD FS
  - IIS
- Can this be bypassed?
  - NTLM (check out our DEFCON 2019 talk ☺)
  - Kerberos
    - AP-Req contained signed certificate thumbprint inside checksum field
    - What happens when Kerberos client has no checksum field?

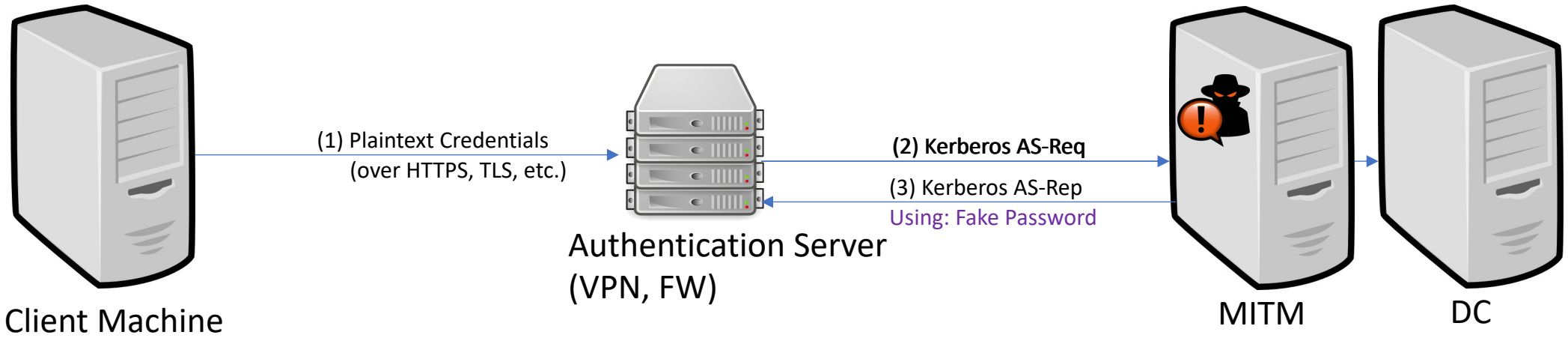



ldap
No. Time Source Destination Protocol Length Info
38 0.453428 10.5.129.36 10.1.0.19 LDAP 1488 bindRequest(2014452872) "regular_user" sasl
39 0.455113 10.1.0.19 10.5.129.36 LDAP 203 bindResponse(2014452872) invalidCredentials (80090346: LdapErr: DSID-0C090579, comment: AcceptSecurityContext error, data 80090346, v3839)
Frame 38: 1488 bytes on wire (11904 bits), 1488 bytes captured (11904 bits) on interface \Device\NPF_{5E0253F1-2202-4F6E-8BA6-8A503F8C8CF6}, id 0
Ethernet II, Src: SunrichT_34:eb:a9 (00:0a:cd:34:eb:a9), Dst: PaloAlto_0d:5e:01 (94:56:41:0d:5e:01)
Internet Protocol Version 4, Src: 10.5.129.36, Dst: 10.1.0.19
Transmission Control Protocol, Src Port: 58324, Dst Port: 636, Seq: 415, Ack: 1710, Len: 1434
Transport Layer Security
Lightweight Directory Access Protocol
LDAPMessage bindRequest(2014452872) "regular_user" sasl
messageID: 2014452872
protocolOp: bindRequest (0)
bindRequest
version: 3
name: regular_user
authentication: sasl (3)
sasl
mechanism: GSS-SPNEGO
credentials: 6082052200062b0601050502a082051630820512a00d300b06092a864882f712010202a2…
GSS-API Generic Security Service Application Program Interface
OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation)
Simple Protected Negotiation
negTokenInit
mechTypes: 1 item
mechToken: 6e8204f7308204f3a003020105a10302010ea20703050000000000a382043e6182043a30…
krb5_blob: 6e8204f7308204f3a003020105a10302010ea20703050000000000a382043e6182043a30…
Kerberos
ap-req
pvno: 5
msg-type: krb-ap-req (14)
Padding: 0
ap-options: 00000000
ticket
authenticator
etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
cipher: 66f87974e35725f2819a22536934698d30abbad340b1e293a57213c338d6a7cd569e10e2…
Decrypted keytype 23 usage 11 using learnt encTicketPart_key in frame 34 (id=34.1 same=1) (581ebbfb...)
authenticator
authenticator-vno: 5
crealm: preempt.det2
cname
cksum
cksumtype: cKSUMTYPE-GSSAPI (32771)
checksum: 10000000000000000000000000000000000000000000c000000
Length: 16
Bnd: 00000000000000000000000000000000
....0 .... = DCE-style: Not using DCE-STYLE
....0. .... = Integ: Do NOT use integrity protection
....0 .... = Conf: Do NOT use Confidentiality (sealing)
.... 1... = Sequence: Enable Out-of-sequence detection for sign or sealed messages
.... .1.. = Replay: Enable replay protection for signed or sealed messages
....0. = Mutual: Mutual authentication NOT required
....0 = Deleg: Do NOT delegate


ldap
No. Time Source Destination Protocol Length Info
37 0.357625 10.5.129.36 10.1.0.19 LDAP 1456 bindRequest(792465640) "regular_user" sasl
38 0.359408 10.1.0.19 10.5.129.36 LDAP 139 bindResponse(792465640) success
39 0.359931 10.5.129.36 10.1.0.19 LDAP 192 searchRequest(1425409822) "DC=preempt,DC=det2" baseObject
43 0.361661 10.1.0.19 10.5.129.36 LDAP 899 searchResEntry(1425409822) "DC=preempt,DC=det2" | searchResDone(1425409822) success [1 result]
Frame 37: 1456 bytes on wire (11648 bits), 1456 bytes captured (11648 bits) on interface \Device\NPF_{5E0253F1-2202-4F6E-8BA6-8A503F8C8CF6}, id 0
Ethernet II, Src: SunrichT_34:eb:a9 (00:0a:cd:34:eb:a9), Dst: PaloAlto_0d:5e:01 (94:56:41:0d:5e:01)
Internet Protocol Version 4, Src: 10.5.129.36, Dst: 10.1.0.19
Transmission Control Protocol, Src Port: 60798, Dst Port: 636, Seq: 415, Ack: 1710, Len: 1402
Transport Layer Security
Lightweight Directory Access Protocol
LDAPMessage bindRequest(792465640) "regular_user" sasl
messageID: 792465640
protocolOp: bindRequest (0)
bindRequest
version: 3
name: regular_user
authentication: sasl (3)
sasl
mechanism: GSS-SPNEGO
credentials: 608204f706062b0601050502a08204eb308204e7a00d300b06092a864882f712010202a2…
GSS-API Generic Security Service Application Program Interface
OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation)
Simple Protected Negotiation
negTokenInit
mechTypes: 1 item
mechToken: 6e8204cc308204c8a003020105a10302010ea20703050000000000a382043e6182043a30…
krb5_blob: 6e8204cc308204c8a003020105a10302010ea20703050000000000a382043e6182043a30…
Kerberos
ap-req
pvno: 5
msg-type: krb-ap-req (14)
Padding: 0
ap-options: 00000000
ticket
authenticator
etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
cipher: ff61e8231c826f020ec01bd07f448b59b981f3385a037f850cd169ddc6325e8259b33fb5…
Decrypted keytype 23 usage 11 using learnt encTicketPart_key in frame 33 (id=33.1 same=1) (cd7512b6...)
authenticator
authenticator-vno: 5
crealm: preempt.det2
cname
cusec: 71418
ctime: 2021-07-15 14:04:33 (UTC)

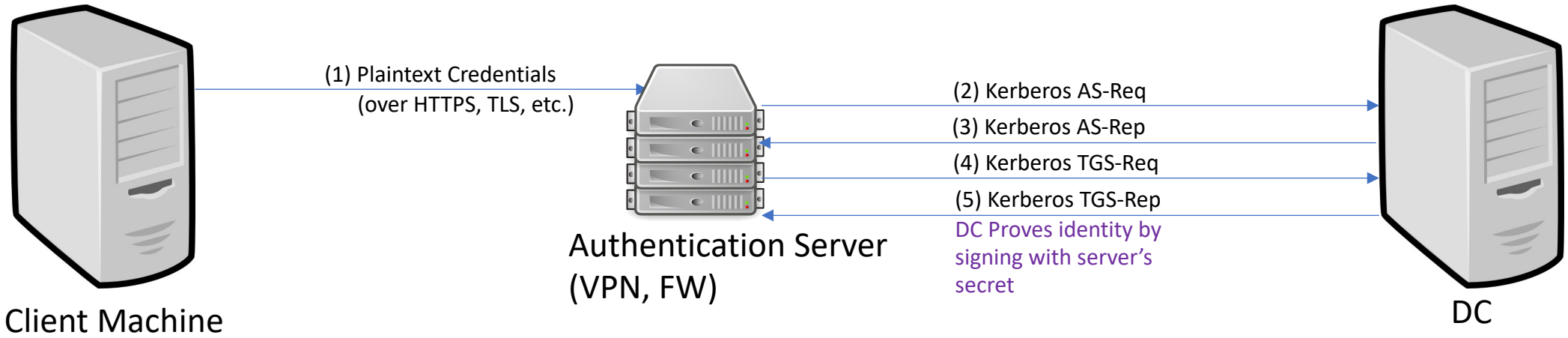
KDC Spoofing

- KDC Spoofing
  - Old Technique
  - Using MITM for authentication bypass
  - Typically exists in VPNs, FWs

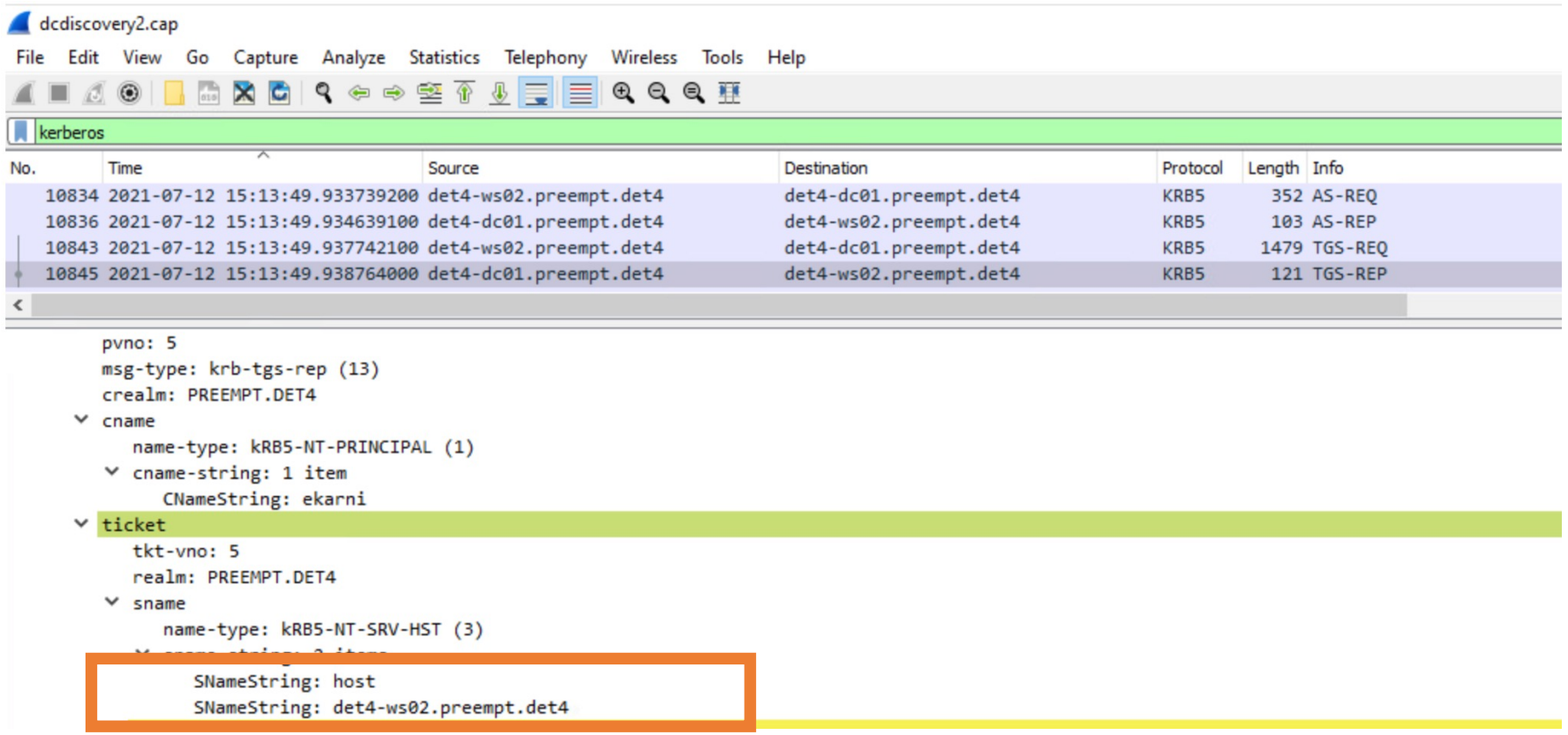


KDC Spoofing Protection

- **Very old technique**
- Protection
  - Create a computer account for authentication server
  - Create a TGS ticket to self using TGT



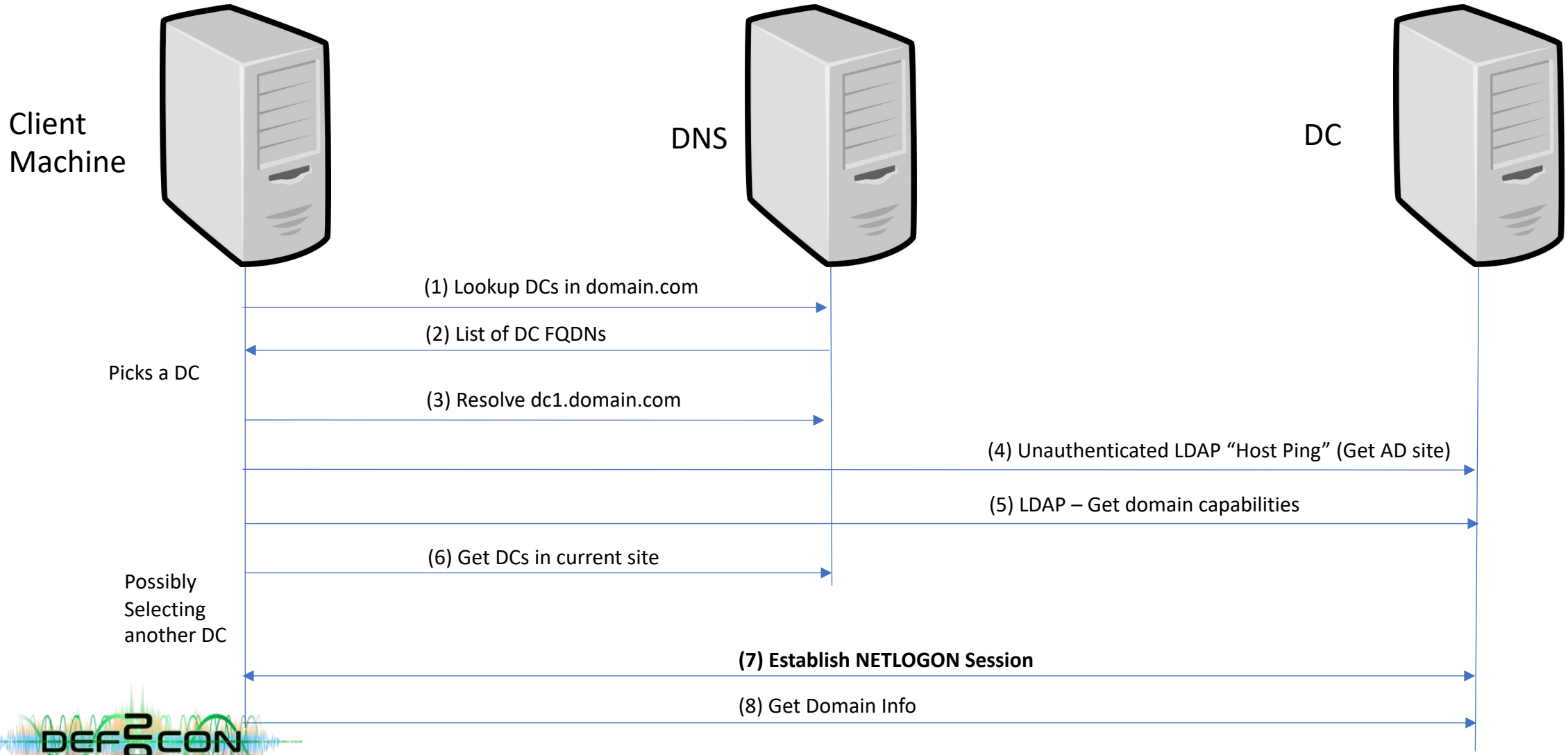
Kerberos Injection



Kerberos Injection

- So, we cannot manipulate TGT and TGS, what now?

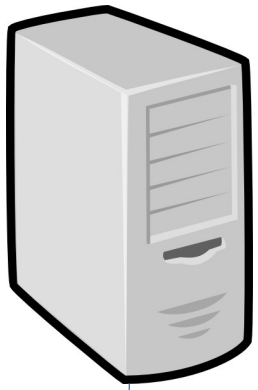
DC Selection Process



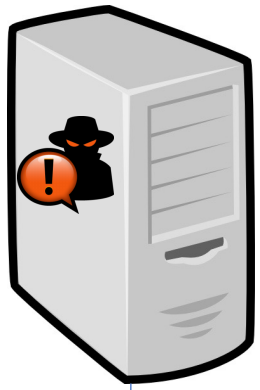
Kerberos Injection

- We can intervene in the DC selection process:
  - Client choose a DC using combination of DNS and LDAP queries
  - Our MITM relays AS-REQ and TGS-REQ (to self) to a real DC
  - MITM is able to serve subsequent DCE/RPC and LDAP requests
    - (As long as NETLOGON secure channel is not required)

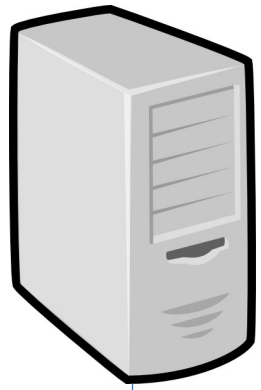
Kerberos Injection



Client Machine



Rogue Server



DC

(1) DC Selection Process

(2) Injected DC
Using: Rogue Server

(3) Kerberos AS-Req

(4) Kerberos AS-Req

(6) Kerberos AS-Rep

(5) Kerberos AS-Rep

(7) Kerberos TGS-Req

(8) Kerberos TGS-Req

(10) Kerberos TGS-Rep
Signing with client secret

(9) Kerberos TGS-Rep
Signing with client secret

(11) Kerberos TGS-Req (LDAP)

(12) Kerberos TGS-Req

(14) Kerberos TGS-Rep
Signing with Rogue server's secret

(13) Kerberos TGS-Rep
Signing with Rogue server's secret

(15) LDAP Search

(16) LDAP Search Response
Malicious injected data

Kerberos Injection – Attack Scenario

- A service that:
  - Uses Kerberos (the usual case…)
  - Ingests data from DC without certificate/netlogon validation (the usual case…)
  - Does not have a fixed DC configured (the usual case…)
  - MITM between the server and the DNS

- The attack:
  - Use MITM to redirect to the Rogue DC
  - Client requests ticket to rogue server (SPN needs to be registered!)
  - Modify responses to the ingested data

Kerberos Injection – How to Mitigate?

- Authenticate DC
  - Establish a NETLOGON channel
  - Use LDAPS with certificate validation
  - Use Kerberos Armoring (we have not tested this…)

- Windows GPO is still safe…

DEMO

Responsible Disclosure

- IRemoteWinSpool NTLM Relay
  - Microsoft fixed issue under CVE-2021-1678
  - Regarding other vulnerable interfaces: "*Regarding other DCE/RPC interfaces for potential exploitation, **If youl find other exploitable DCE/RPC interfaces, please submit these separately**. Doing so will allow us to investigate each one individually*."
- Azure AD
  - MS Acknowledged the issue and replied: "*Thank you for reaching out. **MitM requirement requires another vulnerability to be exploited to achieve a successful MitM, or a compromised connection**, or some level of privileges. We also strongly recommend to treat AD Connect server as a domain controller, following hardened security practices*"
- Channel Bindings
  - MS Acknowledged the issue and replied: : "*Microsoft has decided that it will not be fixing this vulnerability in the current version and we are closing this case.*"
- Kerberos Injection
  - A few vendors are working on fixing their Kerberos clients – expect updates soon

Closing Remarks

- MITM is not a security boundary (at least for Microsoft)

- More Technically:
  - Securing Protocols from MITM is hard
  - Kerberos is not validating DC identity properly
  - GSS-API does not guarentee protection from MITM

Tips for Defenders

- Network Hardening
  - Enable server/client signing
  - Regularily patch software
  - Treat critical servers (e.g., AAD Connect) the same as DC
- Kerberos Injection
  - Monitor suspiciously registered SPNs
- Microsoft Recommendation: Avoid being MITM'd… :P