7 MINUTE
SECURITY

Light Pentest - eBook Edition

A Practical, Step-by-Step Playbook for Internal Network Penetration Testing

By: Brian Johnson / 7 Minute Security / www.7MinSec.com

Version: 1.1, published January 13, 2023

Table of Contents

Change Log

- **January 13, 2023 - v1.1 release!**
  - In the section about snooping on the network with tcpdump, our pal hackern0v1c3 offers some more efficient ways to do packet captures on a timed rotation, and in a way that ideally does not exhaust memory or disk.
  - Our eagle-eyed friend DF noted that in the section about dumping domain controller hashes my **cut** command had a syntax error. It is now fixed, thanks!. In that same section, I added a tip for using hash_combiner to spit out your cracked hashes in a *HASH:PLAIN-TEXT-PASSWORD* format.
  - Created a new section on finding systems with unconstrained delegation and exploiting them.
  - In the section about finding easily pwnable users, I added a second *Get-ADUser* PowerShell query that pulls even more potentially sensitive fields like office phone number, street address and more.
  - Added a section on password spraying under a section about finding easily pwnable passwords.
  - My sharp-retina'd friend *fastchar* noticed that the link to the SANS article in the section about scanning for EternalBlue was borqued. It's now fixed. Grazie!
  - Added a section on relaying credentials with MITM6.
  - *Responder.py* doesn't support *-r* anymore, so I fixed that in the section about abusing insecure protocols.
  - CrackMapExec is awesome at extracting Active Directory credentials, so I added a bit about that in the section about extracting hashes from domain controllers.
- **September 27, 2021 - v1.0 release!**
  - The people rejoice (at least my mom did)!

Upcoming Additions

This section has stuff I want to add and/or know I need to fix in the future, but isn't currently baked into the eBook:

- Add section on dropping .scf files
- Add section on PowerUpSQL
- Add section on SMB share enumeration with PowerView/SharpShares/etc.
- Add section on data exfiltration
- Add section on ~~Cobalt Strike~~ Brute Ratel
- Add command line reference section at the end and import those from BPATTY

Introductory/Boring Stuff

Who are you?

Hi, I'm Brian Johnson. I manage the 7 Minute Security podcast and business, I've been in IT and security for about 20 years, and I really enjoy network pentesting. When I'm not staying up too late doing that, I hang out with my family and sing in a band.

What's this book all about?

This book is a "lets jump in and DO THIS!" style, step-by-step narrative of how we at 7 Minute Security find common vulnerabilities (I like to call it "low-hanging hacker fruit") during internal network pentests.

Who is this book for?

You! Because if you're reading this book right now, you probably bought it. Yay for honesty! If you didn't buy it, then you're clearly a dirty, dirty thief who should go to jail for infinity years. Actually, however you got here, I'm just happy you're enjoying this book (but please buy it :-)).

Seriously though, this book is probably best for people who are already somewhat familiar with networking and working in Windows and Linux operating systems. Additionally, a positive attitude and desire to learn doesn't hurt either!

Who is this book *not* for?

Well, this book is not *not* for anyone in particular, but just know it's designed to be a mile wide and inch deep. So we're *not* going to cover things like:

- **Installing and configuring your penetration test dropbox**. I'll talk at a high level about how my pentest dropbox is setup, but it won't be a step-by-step setup guide (I'm looking at writing this as a separate eBook. Interested? Let me know.)

- **Installing and configuring Kali Linux -** the OffSec folks have a great site dedicated to that already. There are tons of good books on Kali too.

- **Active Directory fundamentals -** Microsoft's site might be a good place to start.

- **Any type of networking fundamentals -** my Google-fu tells me this Microsoft resource might help, though.

Are you responsible if I do something naughty with this info?

Nope, nope, nopey McNopeNope with a side order of nopey dippin' sauce. My lawyers wanted me to make it absolutely clear, so I'll say it one more time in a legal-y way: me (Brian Johnson) and my podcast and business (7 Minute Security) are not in any way, shape or form responsible for anything you do with this information - be it for good or evil. If you get in trouble, use your one phone call to give your mom a ring. In general though, the following advice will help keep you out of bad situations: make sure you always, *always* get written, contractual permission to pentest any network, and always follow the applicable laws of whatever place you're living in.

Are you going to conclude this introduction section of this book with some cheesy thank you to the people that made it possible? Who do you think you are, Stephen flippin' King?

Yes and no. But writing this book absolutely wouldn't have been possible without a lot of encouragement, love and support from so many, so it's only appropriate I use this digital paper to thank them:

- God.
- My wife, Aimee (a.k.a. "Mrs. 7MS") who always encourages me to go for the next job, certification or opportunity. I would never be running my own company today if it wasn't for her.
- My boys, Cameron and Atticus, who listen patiently to my security stories each day and then beg me to talk about *anything* else.
- Podcast listeners and my supportive pals on Slack and social media. Thanks a million.
- Countless security engineers and researchers who have created blogs, tools, podcasts, videos and more to share their wealth of knowledge for the benefit of the community, and/or have mentored me personally. These folks include - but are *not* limited to:
    - Joe "The Machine" Skeen (@Gh0sthax)
    - Joe "Champagne Schwarzenegger" Klein (@UrbanMongoose)
    - Black Hills Information Security (John and the gang!)
    - hackern0v1c3
    - Ken Nevers
    - Paul Wilch
    - Xoke

Enough blabbering, are we ready to start pentesting yet?

Hang tight...we're almost there. Just a few things before we get started:

First you'll need something to pentest with

As I mentioned earlier, I'm considering writing a standalone e-book on getting your pentest dropbox installed and configured, but these four podcast episodes should help you get a jump start:

- DIY Pentest Dropbox Tips - Part 1
- DIY Pentest Dropbox Tips - Part 2
- DIY Pentest Dropbox Tips - Part 3
- DIY Pentest Dropbox Tips - Part 4

We also have a few episodes about building pentest dropboxes on our YouTube channel.

At the core, though, we run VMWare ESXi on Intel NUCs like this one, and configure two primary virtual machines:

- A Windows 10 Pro VM with 2 processors and 4 gigs of RAM and 30 gigs of disk space
    - We also attach a secondary virtual disk with a few gigs of space and provision it with Bitlocker To Go.

- A Kali Linux VM with 2 processors and 4 gigs of RAM and 30 gigs of disk space. When you install the OS, choose the option to encrypt the entire drive.

Note that in both VMs we encourage having at least one drive or partition that is fully encrypted. Why? Well, we want to make sure that these dropboxes are as safe and secure as possible when they go to and from our client sites. If you complete a pentest remotely and gathered sensitive data in the process, it's a good idea to delete that *before* the client ships the device back to you. But even if you forget to do that, it's nice to know that any drives with sensitive information are fully encrypted should they fall into the wrong hands.

We assume compromise

7 Minute Security does almost all of our penetration tests under an "assume compromise" narrative. We believe that given enough time/energy/resources/money, an attacker *will* find their way past your perimeter and onto your internal network - be that through phishing, vishing or plugging directly into your network.

Some see this as "cheating" and insist that all penetration tests should start from the perimeter, requiring the tester to create C2 infrastructure and phishing campaigns, as well as write custom malware, to be successful. We agree - that type of test offers a *ton* of value. However, most of our customers do not have the budget for this type of engagement. So we essentially start our tests by plugging our dropbox into their network with no other information except perhaps a basic Active Directory account and the subnets we're allowed to attack. From there, we set out to achieve goals agreed upon with the client, such as:

- Capturing/cracking passwords
- Escalating privileges
- Exploiting systems
- Locating and exfiltrating sensitive information

Section 1: Get a Lay of the Land

Do we have a method to our madness?

Ok so you've got your shiny pentest box fired up, you've remoted into it, your connection speed is nice, you have snacks and caffeine ready, and you're ready to start testing pens! So where do we start? Port scans? Launch a bunch of scripts we found off shady parts of the internet? Hope and pray the client has XP machines in their environment so we can pwn them with MS08-067? Launch a vulnerability scanner on the network and call the report done? DDOS the network just to show how cool we are?

While those are all approaches I've seen pentest firms do in the past, I recommend following a methodology like the Penetration Testing Execution Standard (PTES). While it's a little long in the tooth, at a high level I think their approach makes a lot of sense. Boiled down and simplified, we want to:

- Learn what hosts, ports and services are active on the network
- Find software, hardware and network protocols that are vulnerable to attack
- Leverage those weakness to exploit systems, capture/crack passwords and escalate privileges
- Locate the "crown jewels" of the test (this may be Domain Admin access, a specific file share, a database full of healthcare information, etc.)
- Setup back doors so we can persist in the network as much as possible
- Exfiltrate data (I recommend simulating bogus data rather than real, sensitive data)

Personally, when I start a fresh pentest I operate under the assumption that the client has a SIEM or other security monitoring tools and are potentially going to "see" *everything* I do. While that's rarely the case, I don't advise beginning your test by shotgunning nmap scans around the network or running Nessus all over the place. There's a ton you can learn about the network without making a lot of "noise."

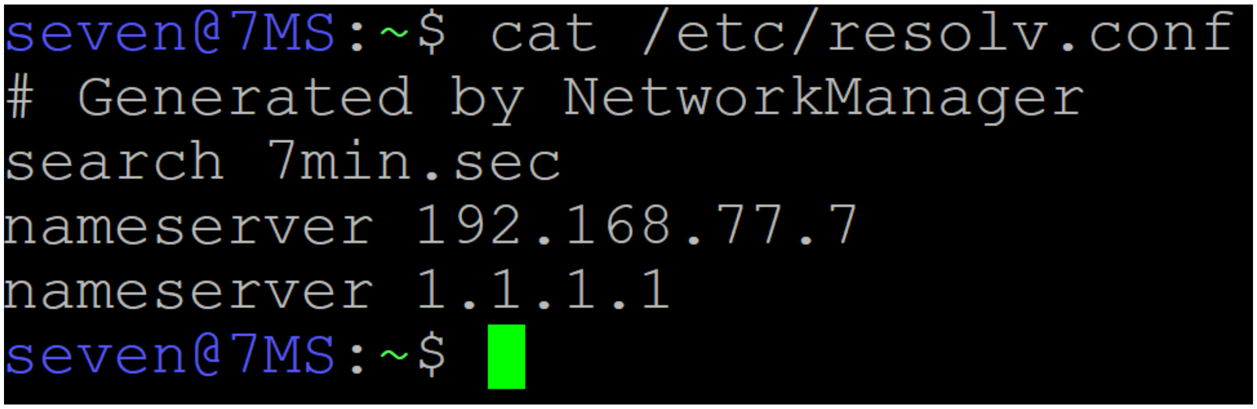
Find the domain controllers

One of the first things I want to know about the environment is where the domain controllers are. Domain controllers will likely play an important part of actions we do later, so it's nice to know what subnets they live in, because if they're in the 192.168.1.x and 192.168.2.x networks, there's a pretty good chance that *other* important systems live there as well - like databases, file and mail servers.

We've got a couple of options for finding the DCs. From your Kali box, you can drop to a Terminal window and type:

```
cat /etc/resolv.conf
```

The output could look something like this:



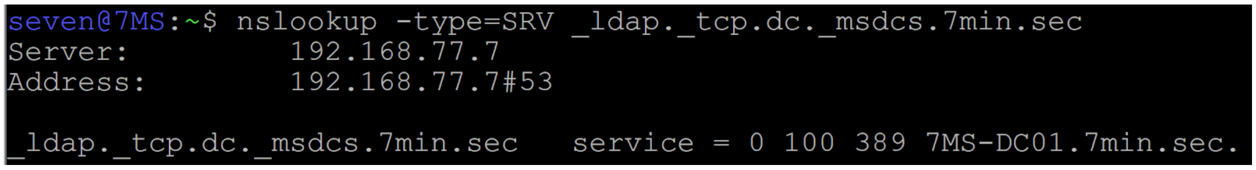
```
seven@7MS:~$ cat /etc/resolv.conf
# Generated by NetworkManager
search 7min.sec
nameserver 192.168.77.7
nameserver 1.1.1.1
seven@7MS:~$
```

It's important to note that the *nameserver* entries here are not *always* going to be domain controllers. From the output above, we see one internal DNS server of *192.168.77.7* that is probably a domain controller, and the second entry, *1.1.1.1* is the upstream resolver. We'll do a bit of nmap scanning in a bit to ensure the *192.168.77.7* endpoint is a DC, but for now just make note of these addresses.

To get a more definite picture of where our domain controllers are, we can run the following:

```
nslookup -type=SRV _ldap._tcp.dc._msdcs.7min.sec
```

My output looks like this:



```
seven@7MS:~$ nslookup -type=SRV _ldap._tcp.dc._msdcs.7min.sec
Server:         192.168.77.7
Address:        192.168.77.7#53

_ldap._tcp.dc._msdcs.7min.sec   service = 0 100 389 7MS-DC01.7min.sec.
```

In my test lab here, I've got just the one domain controller at *192.168.77.7*. In the "real world" you'll probably see at least two entries. Make note of these host names and IPs. You'll need them later.

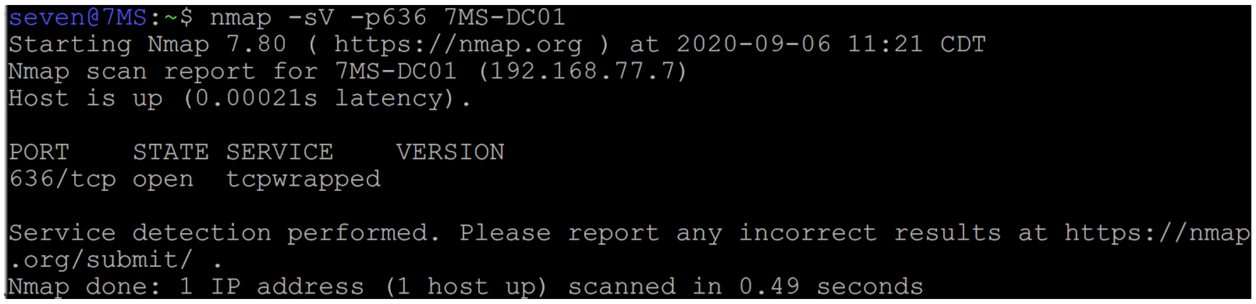
If your output is a *big* list with a ton of DCs, then you can pipe your *nslookup* command with an *awk* command to get a clean list of IPs!

```
nslookup -type=SRV _ldap._tcp.dc._msdcs.7min.sec | awk '{ print $7 }' | sed
's/\.$//g' | sed '/^$/d'
```

Great. Now that we know where the DCs are, there's another important bit of information we'll want to know about them - specifically, whether LDAPS is configured properly. We'll talk more about LDAPS a bit later in this book, but for now lets scan the LDAPS port (TCP 636) with nmap and add the *-sV* flag to do a version scan:

```
nmap -sV -p636 7MS-DC01
```

My output says:

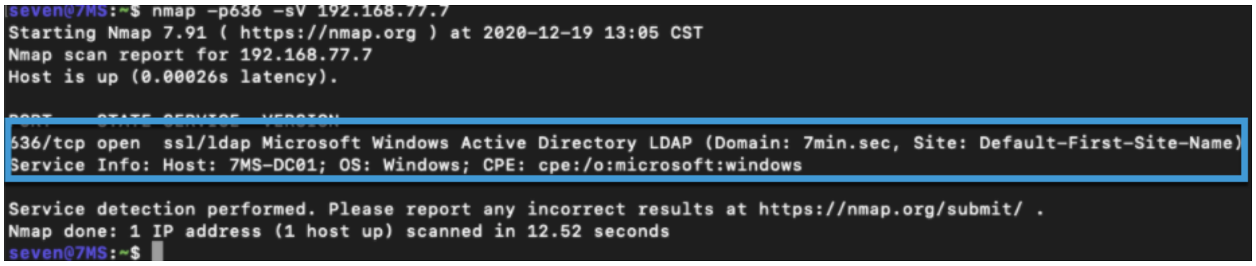


```
seven@7MS:~$ nmap -sV -p636 7MS-DC01
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-06 11:21 CDT
Nmap scan report for 7MS-DC01 (192.168.77.7)
Host is up (0.00021s latency).

PORT    STATE SERVICE    VERSION
636/tcp open  tcpwrapped

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.49 seconds
```

Note that the service shows as *tcpwrapped*. I learned from this great blog post that "*...if you just get a "tcpwrapped" response, LDAPS is not properly configured*." Why is that a big deal? Because if LDAPS is *not* properly configured, certain relay attacks will fail because they rely on the ability to relay HTTPS to LDAPS - which is only possible if LDAPS *is* properly configured.

Note: in case you're interested, I was able to get LDAPS properly working in my lab environment by following this article to get OpenSSL installed and then I followed this gist and this blog to get the LDAPS install itself complete.

When the domain controllers are properly configured for LDAPS, the nmap scan result will look more like this:



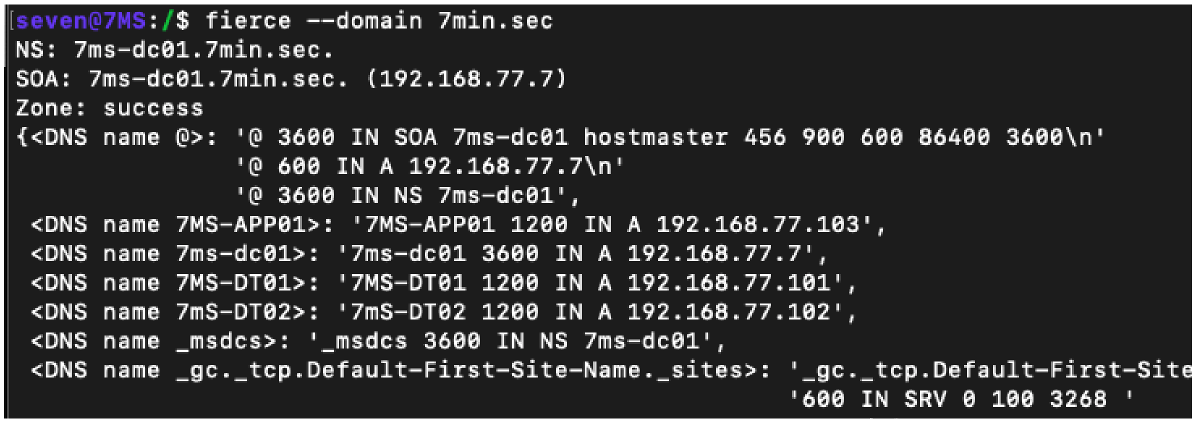
```
seven@7MS:~$ nmap -p636 -sV 192.168.77.7
Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-19 13:05 CST
Nmap scan report for 192.168.77.7
Host is up (0.00026s latency).

PORT    STATE SERVICE  VERSION
636/tcp open  ssl/ldap Microsoft Windows Active Directory LDAP (Domain: 7min.sec, Site: Default-First-Site-Name)
Service Info: Host: 7MS-DC01; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.52 seconds
seven@7MS:~$
```

Force a DNS zone transfer

Now that we know where the domain controllers are (again, in my case there's just the one), I like to see if any of them are misconfigured to allow zone transfers. With a zone transfer, your Kali box is basically asking the DNS server "Hey pal, is it cool if you give me a record of ALL the hosts you know about, complete with their IP addresses?" If the server is misconfigured, BLAMO! You get that information, which can be incredibly helpful because you'll essentially have a "map" of the network and can often find valuable targets without doing any other recon.

There are several ways to request a domain transfer from a DNS server, but I kind of like the fierce tool since it's easy to use and built right into Kali. We'll check if 7min.sec is misconfigured to allow a domain transfer like so:

```
fierce --domain 7min.sec
```



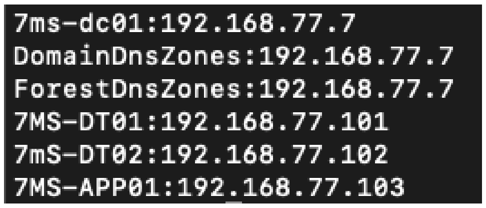
```
seven@7MS:/$ fierce --domain 7min.sec
NS: 7ms-dc01.7min.sec.
SOA: 7ms-dc01.7min.sec. (192.168.77.7)
Zone: success
{<DNS name @>: '@ 3600 IN SOA 7ms-dc01 hostmaster 456 900 600 86400 3600\n'
               '@ 600 IN A 192.168.77.7\n'
               '@ 3600 IN NS 7ms-dc01',
 <DNS name 7MS-APP01>: '7MS-APP01 1200 IN A 192.168.77.103',
 <DNS name 7ms-dc01>: '7ms-dc01 3600 IN A 192.168.77.7',
 <DNS name 7MS-DT01>: '7MS-DT01 1200 IN A 192.168.77.101',
 <DNS name 7mS-DT02>: '7mS-DT02 1200 IN A 192.168.77.102',
 <DNS name _msdcs>: '_msdcs 3600 IN NS 7ms-dc01',
 <DNS name _gc._tcp.Default-First-Site-Name._sites>: '_gc._tcp.Default-First-Site
                                                      '600 IN SRV 0 100 3268 '
```

As you see above, the 7MS domain controller *7MS-DC01* has coughed up all DNS records it knows about. In a "real" network of a larger size you'd see a *ton* of records and quickly be able to see interesting machines based on their names, which could help you zero in on valuable targets to exploit. In that big of a network, it might help to first dump the DNS records to a text file:

```
fierce --domain 7min.sec > /tmp/dns.txt
```

Then use some *cat* and *awk* commands to make the export a little cleaner:

```
cat /tmp/dns.txt| awk '{ print $3, $8 }' | tr -d ' <>,}\'\' | sort -t . -k
3,3n -k 4,4n
```



```
7ms-dc01:192.168.77.7
DomainDnsZones:192.168.77.7
ForestDnsZones:192.168.77.7
7MS-DT01:192.168.77.101
7mS-DT02:192.168.77.102
7MS-APP01:192.168.77.103
```

Ahhh, a nice clean list of hostnames and IPs!