Exploiting known vulnerabilities, misconfigurations and weaknesses in native protocols to compromise Windows Active Directory Domains with a focus on traceability and ease of use.

Søren Fritzbøger - s153753
Vejledt af Henrik Tange

20. Januar 2019

Danmarks Tekniske Universitet

Abstract

Abstract here

Table of contents

Abbreviations

AV Antivirus. 5

DC Domain Controller. 6

IDS Intrusion Detection System. 5

LLMNR Link-local Multicast Name resolution. 5–8, 10, 14

NBNS NetBIOS Name Resolution. 5–8

SIEM Security Information and Event Management. 5

WINS Windows Internet Name Service. 8

1 Introduction

1.1 Problem background

When performing a pentest on Windows Active Directory environments, one of the goals is usually to obtain Domain Administrator privileges, usually in the form of a Domain Admin account. There are numerous ways to gain initial foothold in an Active Directory Domain, but the most common one is by exploiting the native protocols NetBIOS Name Resolution (NBNS) and Link-local Multicast Name resolution (LLMNR)[9] to gain crackable hashes which can be bruteforced offline. Hereafter a number of different techniques and exploits can be used to gain additional credentials, but it usually consists of dumping cached credentials on domain joined hosts in one way or another.
As one can easily figure out this is usually a manual job where many different tools are joined together to produce the right result. This usually means it is a trivial and easy job, which requires a lot of time that can be spent on more advanced tasks. This is often done with tools not written by the pentester themselves, which can pose a security risk as the tools can be backdoored or otherwise have security vulnerabilities.
One of the objectives of a pentest is usually to be as silent as possible and not trigger any alerts in any Intrusion Detection System (IDS), Security Information and Event Management (SIEM) or similar systems. The risk of using publicly available tools is therefore also that the tools are highly likely to be detected by Antivirus (AV) and will often trigger unwanted alerts.
Another issue of performing pentests is to document and remember the order of tasks that where done. A pentest usually concludes with a report where the necessary steps are explained to the customer, and this includes in what order tasks were done and which user credentials were used.

1.2 Problem brief

The purpose of this project is to determine whether a proper solution to the aforementioned problem can be found, and to analyze how such a tool can be developed. The problem is split in two, where the first goal is to gain an initial foothold and the second is to gain Domain Administrator privileges. To accomplish this the many techniques and methods will be discussed and evaluated in comparison to each other, and the most valid solution will be implemented in a piece of software that aims to be easy to use and contain a high level of traceability.
The developed piece of software must be able to be easy to use so that an incentive to use it instead of other tools is created. It should be constructed with AV evasion in mind, such that it will not be detected by AVs. Furthermore it must be designed with traceability in mind, such that a clear timeline can be constructed and documented.

1.3 Report structure

1.4 Pentesting Windows Domains

2 Initial foothold

In a Windows Active Directory Domain there are numerous ways of gaining an initial foothold. The following methods are the most used in modern penetration testing of Windows AD environments.

BRUTE User credential bruteforcing

SPRAY Password spraying

EXPL Exploiting known vulnerabilities on unpatched systems

CLEAR Clear text passwords stored on public shares

SPOOF NBNS and/or LLMNR spoofing

All of the above mentioned methods have their weaknesses and strengths, which should be taken into account when choosing the best method or methods to gain initial foothold in the domain. To make an educated guess of which method(s) to pursue further a comparison between the different methods is needed. Table 1 gives a comparison of the different methods, and shows which weaknesses and strengths each methods possess.

Strength	BRUTE	SPRAY	EXPL	CLEAR	SPOOF
Is it automatable?	+	+	-	-	+
Is it fast?	-	-	+	-	-
Account lockout issues?[4]	-	-	+	+	+
Communication with critical systems such as a DC?	-	-	+	+	+
Easy to detect?	-	-	+	+	+
Is it easy to do?	+	+	-	(+)[1]	+
Points	2	2	4	3.5	5

Table 1: Comparison of different methods to gain initial foothold in a Windows AD environment

All of the above mentioned methods are valid and have been used during real penetration tests. Table 1 scores each method according to their pros and cons, and here it is clearly showed that spoofing NBNS and LLMNR is the most optimal way of gaining initial foothold. This corresponds with real life experience where the protocols are enabled by default[7] and not monitored

[1] This method can be very time consuming

correctly.
Now that a method has been chosen, section 2.1 will look further into how spoofing can be done in an automated way.

2.1 Spoofing

To understand how spoofing of LLMNR and NBNS works we first need to elaborate how spoofing can lead to a credential compromise. To do this we need to understand how name resolution works in Windows, regardless of the protocol used. Windows follows a sequence of steps in order to resolve a host name.[5] The steps are the following:

1. The client checks to see if the name queried is its own
2. The client searches local Hosts file
3. The client queries the DNS server
4. If enabled, Name resolution is done (LLMNR and NBNS)

This is illustrated on figure 1 where it is also illustrated how spoofing fits into the sequence. As it is shown, the Attacker will listen to multicast packets send on the local subnet and answer to any Name Resolution packets. If successful, the *Client* will register *Server1* to have the IP address of *Attacker*, and thereby sending all packets intended for *Server1* to *Attacker*.

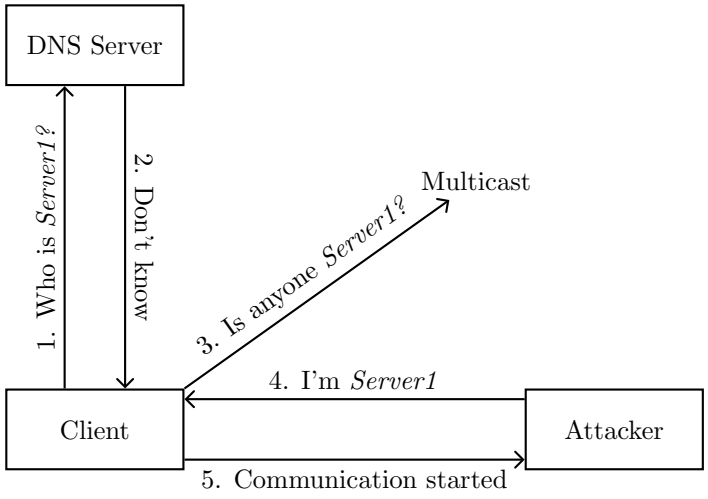

Figure 1: How an attacker spoofs the Name Resolution protocols in order to receive traffic intended for other servers

In Windows two different Name Resolution protocols are used and operate by default on all machines. LLMNR and NBNS work side by side unless specifically

turned off, which it is not by default. In section 2.1.2 and 2.1.2 the protocols are analyzed in detail and the attacks are described in detail.
After successfully spoofing a host name and getting traffic redirected to our machine, we need to be able to use that traffic for a malicious purpose. The most common purpose is to gather credentials from the client by having rouge servers running on the attacker. The technique and methods behind this is explained in section 2.2

2.1.1 NetBIOS Name Resolution (NBNS)

In Windows NBNS is implemented in the Windows Internet Name Service (WINS) which is a legacy service used to map host names to IP addresses. In newer versions of Windows it has no use, but it is kept for backward compatibility purposes. The NetBIOS RFC specification, RFC 1001[2], contains much more than Name Resolution, but for spoofing purposes we only need to look at Name Resolution. RFC 1002[3] contains detailed technical specification as to how Name Resolution is implemented in NetBIOS.

2.1.2 Link-local Multicast Name resolution (LLMNR)

LLMNR is the newest protocol for name resolution where DNS name resolution is not possible[1]. LLMNR works in the same way as NBNS in such that a name query is sent to the link-scope multicast address(es), and a responder can hereafter respond to the packet and claim itself as the host that was requested. The sequence of events with LLMNR according to RFC 4795[1] is the following

1. An LLMNR sender sends a LLMNR query to the link-scope multicast address(es). This is a LLMNR packet containing a Question Section
2. A responder responds to this query by sending an UDP packet. This is a LLMNR packet containing a Question Section and a Resource Record
3. The sender process the responders packet

This is all well if we assume that the network itself is not already compromised. In case a malicious host is existent on the network, and the host is listening on the link-scope multicast address there is nothing from stopping this host in responding maliciously to the packets. LLMNR is made to follow the DNS specification, so in order to spoof it we need to know how DNS packets look like. This can be found in RFC 1035[8].

LLMNR packets There exists two different LLMNR packet types. A **request** and a **response**. The **request** contains the *header* and a *question section*. The **response** contains a *header*, a *question section* and a *resource record*. Format details of these types can be seen in figure 2.

This report will not explain the packet details in full, but will focus on the parts necessary to spoof a LLMNR response. The most important fields of the Question Section and the Resource Record is explained in list 1 and 2.

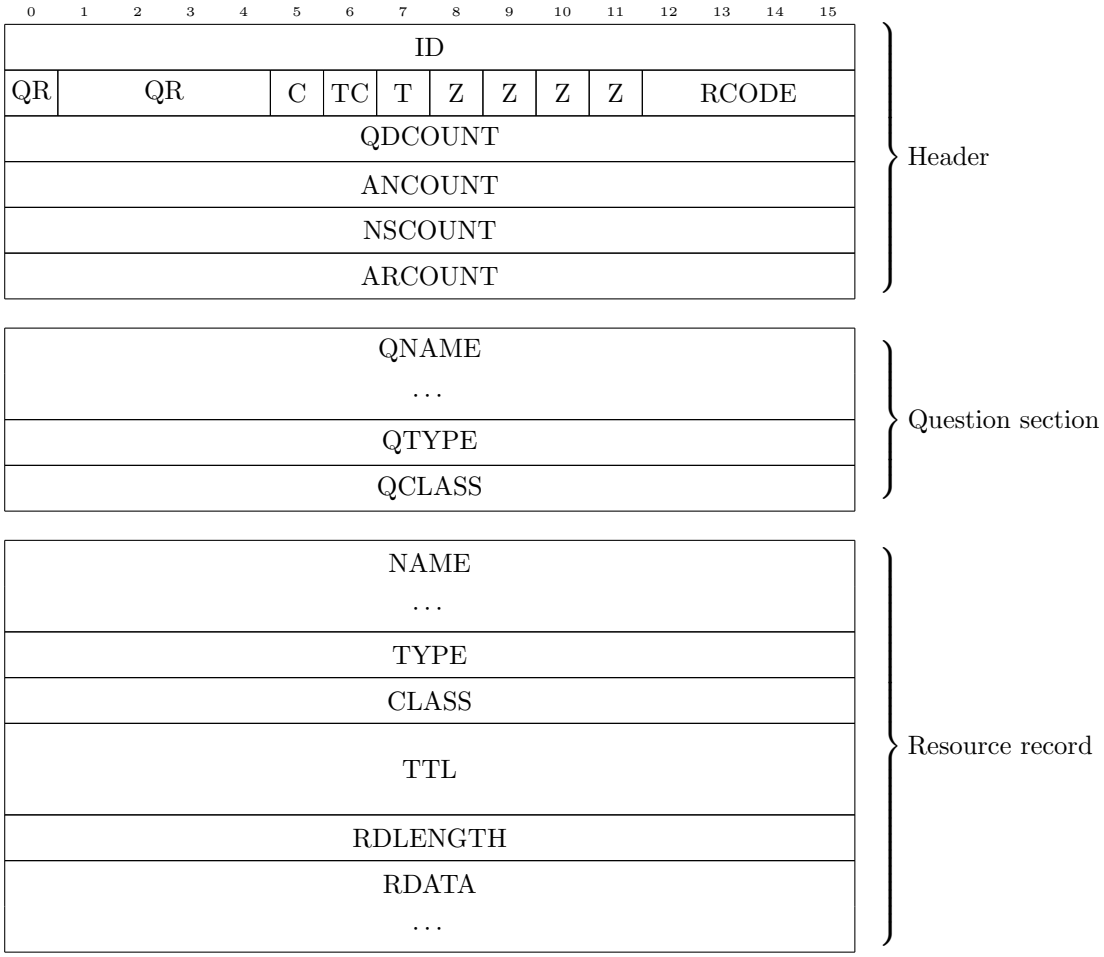

Figure 2: Link-local Multicast Name resolution (LLMNR) packet specification[1][8]

QNAME A domain name in the following format: A length octet followed by that number of octets

QTYPE Two octet code which specify the query type. Usually $0x0001$ for Host address(IP)

QCLASS Two octet code which specify the query class. Usually $0x0001$ for Internet (IN)

List 1: Question Section

NAME See QNAME of *Question Section*

Type See TYPE of *Question Section*

CLASS See QCLASS of *Question Section*

TTL a 32 bit unsigned integer which specify Time To Live in minutes

RDLENGTH a 32 bit unsigned integer which specify the number of octets in RDATA

RDATA A variable length string of octets. Usually an IP address

List 2: Resource record

So to answer a LLMNR packet we need to create a *Resource Record* to match the *Question section* sent out by a client. *Name, Type and Class* should match the request, *TTL* should be set to an arbitrary time in minutes (for example 30 - $0x0000001e$ in bytes), *RDLENGTH* should be 4 ($0x0004$) and RDATA should be our own IP Address.

```
0000   f2 75 00 00 00 01 00 00 00 00 00 00 07 73 65 72  .u...........ser
0010   76 65 72 31 00 00 01 00 01                       ver1.....
```

Figure 3: LLMNR request

```
0000   f2 75 80 00 00 01 00 01 00 00 00 00 07 73 65 72  .u...........ser
0010   76 65 72 31 00 00 01 00 01 07 73 65 72 76 65 72  ver1......server
0020   31 00 00 01 00 01 00 00 00 1e 00 04 c0 a8 00 02  1............@.b
```

Figure 4: Spoofed LLMNR response

Figure 3 and 4 shows how the response should look for a request for *server1*. After sending this response we would have succeeded in spoofing the LLMNR protocol to send traffic to our host.

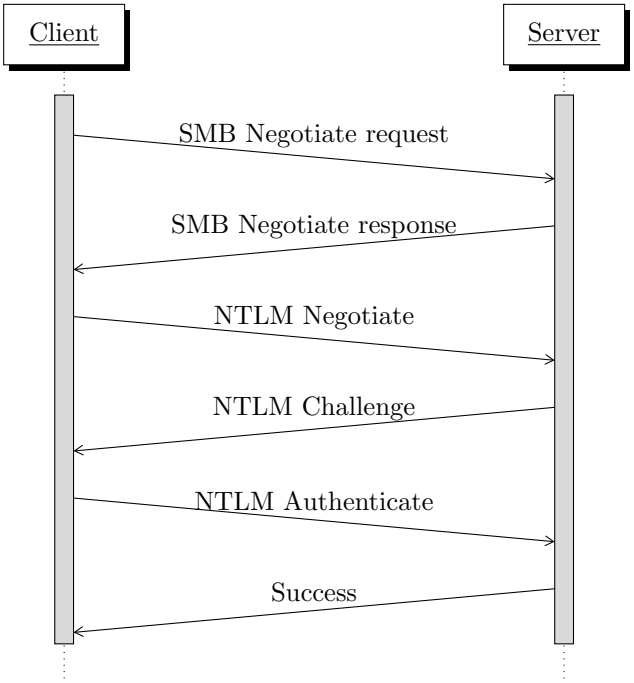

Figure 5: SMB NTLM authentication[6]

2.2 Credential acquiring

2.2.1 Credential types

LM
NT
NTLM
NetNTLMv1
NetNTLMv2

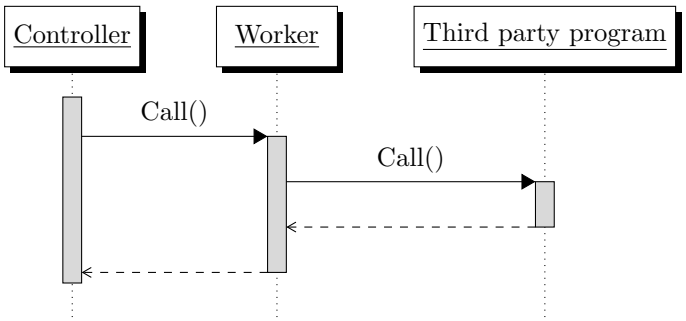

Figure 6: Test figure

List of Figures

References

[1] et al. Aboba. *Link-Local Multicast Name Resolution (LLMNR)*. RFC 4795. IETF. URL: https://tools.ietf.org/html/rfc1002.

[2] NetBIOS Working Group. *Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods*. RFC 1002. IETF. URL: https://tools.ietf.org/html/rfc1001.

[3] NetBIOS Working Group. *Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifiations*. RFC 1002. IETF. URL: https://tools.ietf.org/html/rfc1002.

[4] Microsoft. *Account lockout threshold*. URL: https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/account-lockout-threshold (visited on 11/17/2018).

[5] Microsoft. *Microsoft TCP/IP Host Name Resolution Order*. URL: https://support.microsoft.com/en-us/help/172218/microsoft-tcp-ip-host-name-resolution-order (visited on 11/20/2018).

[6] Microsoft. *NTLM Over Server Message Block (SMB)*. URL: https://msdn.microsoft.com/en-us/library/cc669093.aspx (visited on 11/15/2018).

[7] The Cable Guy - Microsoft. *Link-Local Multicast Name Resolution*. URL: https://docs.microsoft.com/en-us/previous-versions//bb878128(v=technet.10) (visited on 11/20/2018).

[8] Mockapetris. *Domain Names - Implementation and specifiation*. RFC 1035. IETF. URL: https://tools.ietf.org/html/rfc1035.

[9] Georgia Weidman. *Scenario-based pen-testing: From zero to domain admin with no missing patches required*. URL: https://www.computerworld.com/article/2843632/security0/scenario-based-pen-testing-from-zero-to-domain-admin-with-no-missing-patches-required.html (visited on 11/15/2018).