In the world of penetration testing, one of the most important steps in compromising an environment is gaining authentication. In the realm of Microsoft™ Windows™ operating systems, one of the most common ways to accomplish this is abusing the native and default enabled name resolution protocols – NetBIOS and LLMNR. What are these protocols and why are they enabled in an environment? How do they work? How are attackers using them? What can an attacker do with this access once gained? Let’s explore.

What are NetBIOS and LLMNR?

First, let’s break down these terms. NetBIOS-NS stands for network basic input/output system name service. NetBIOS-NS is often referred to as its base application programming interface, NetBIOS, for short. LLMNR stands for link-local multicast name resolution.

NetBIOS and LLMNR are protocols used to resolve host names on local networks. Their main function is to resolve host names to facilitate communication between hosts on local networks. NetBIOS is generally outdated and can be used to communicate with legacy systems. LLMNR is designed for consumer-grade networks in which a domain name system (DNS) server might not exist. NetBIOS is enabled by default on Microsoft Windows 2000 machines and above (while existing independently of DNS in older versions), and LLMNR is enabled on Microsoft Windows Vista™ machines and above.

A Deeper Dive

On a network using the transmission control protocol (TCP) and internet protocol (IP) stack, which includes most networks today, it is necessary to convert names of resources to IP addresses to connect to these resources. So a network would need to resolve “resource.domain.com” to its IP address of “x.x.x.x” in order to know exactly where to send the traffic. Microsoft Windows clients follow a sequence of methods when attempting to resolve a name to an address, stopping the search when it successfully matches a name to an IP address.

When a computer requests a network resource, it usually follows the hierarchy of queries (using slightly different configurations) specified below to identify the target resource. It stops once the name is identified.

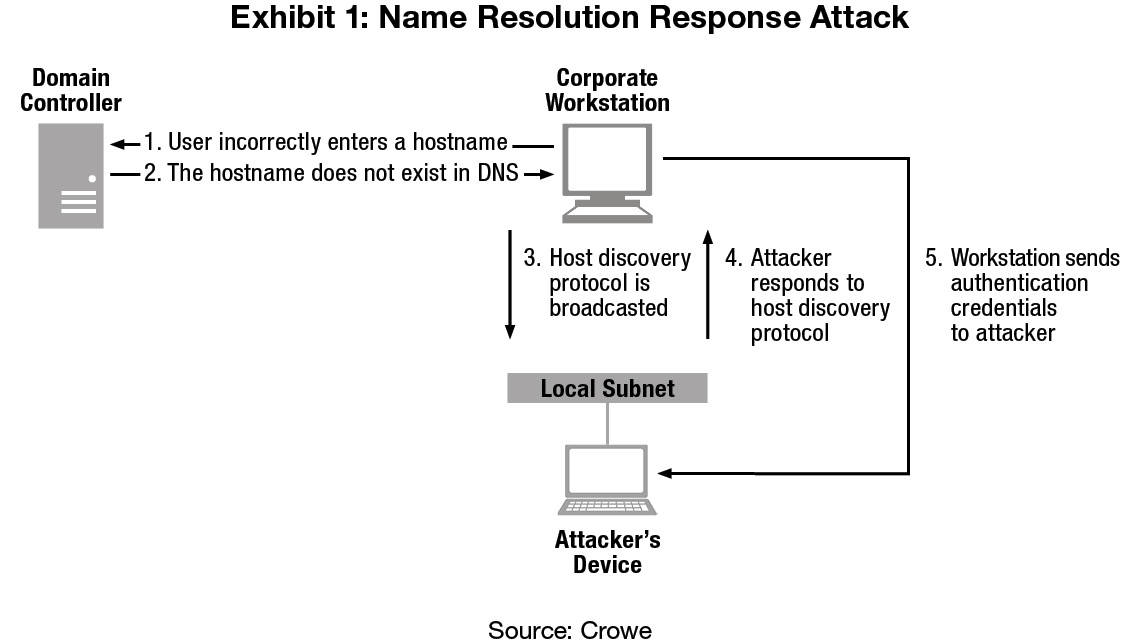
1. Check to confirm whether the request is for the local machine name.
2. Check the local cache of recent successfully resolved names.
3. Search a local hosts file, which is a list of IP addresses and names stored on the local computer. Depending on the device, this file might already be loaded into the local cache.
4. Query a DNS server, if so configured.
5. If LLMNR is enabled, broadcast LLMNR queries across the local subnet to ask its peers for resolution.
6. If NetBIOS is enabled, attempt NetBIOS name resolution via broadcasting NetBIOS-NS queries to the local subnet if the name is not in the local NetBIOS cache. This step might use a Windows Internet Name Service (WINS) server if so configured, as well as the LAN manager hosts (LMHOSTS) file.

Note that the NetBIOS name of a computer and its host name can be different. However, in most cases, the NetBIOS name is either the same or a truncated version of the full host name. NetBIOS name resolution to an IP can happen via broadcast communications, using a WINS server, or using the LMHOSTS file.

The Attack

The main issue with these protocols is the inherent trust the victim computer assumes with other devices within its segment of the network. If the computer cannot identify the resource it’s looking for among the first four steps listed above, our favorite local name resolution protocols come into play. The best example of this is when a user mistypes the name of a resource or requests a resource that is no longer reachable.

Naturally, this scenario occurs often in user segments of the network, and this is where the attacker enters. Once the attacker notices that these resources are being requested on the network via either LLMNR or NetBIOS-NS, nothing is stopping the attacker from responding to the victim computer and is essentially telling it, “Yes! I’m that network resource!” In fact, the attacker’s response goes further – it tells the victim computer, “Not only am I that network resource, but you should authenticate to me!” Sure enough, the victim computer responds to the attacker with a hashed version of its network credentials. Exhibit 1 details the exploitation steps associated with the vulnerable network discovery protocols:

****

The usual goal here is for the attacker to obtain the user credentials from the victim machine. In the end, to get the actual password in “cleartext” such that it is usable to gain authentication to the network, the hashed passwords must be cracked in their NetNTLMv2 format.

Cracking passwords can be achieved via a hybrid dictionary attack, which takes a significant amount of computing, depending on the strength of the password. Briefly described, the password cracking via a hybrid dictionary attack entails taking a large list of potential passwords (including common passwords and dictionary words) and a set of rules for transforming those passwords (such as combining them, swapping out zeroes for O’s, or adding numbers and special characters to the end). Once these passwords and rules are available, then they are hashed with the same one-way algorithm as the password and then compared to see if they are the same.

Alternatively, attackers can perform the same attack simply by trying every possible combination of all possible characters. Called a “brute force” attack, this method generally takes much longer, depending on the length of the password.

Another possible attack vector is for the attacker to relay the credentials to another system in the environment in which those credentials are valid. This method is similar to the previously described method, except for instead of simply saving the credentials, the attacker aims them at another, second system. If the account’s credentials are valid on the second system, then the attacker can successfully gain access to that system. This method allows an attacker to pivot around an environment, and it can be repeated until access is gained to all reachable systems upon which the relayed credentials are valid.

To recap, the conditions necessary for such attacks to occur include the following:

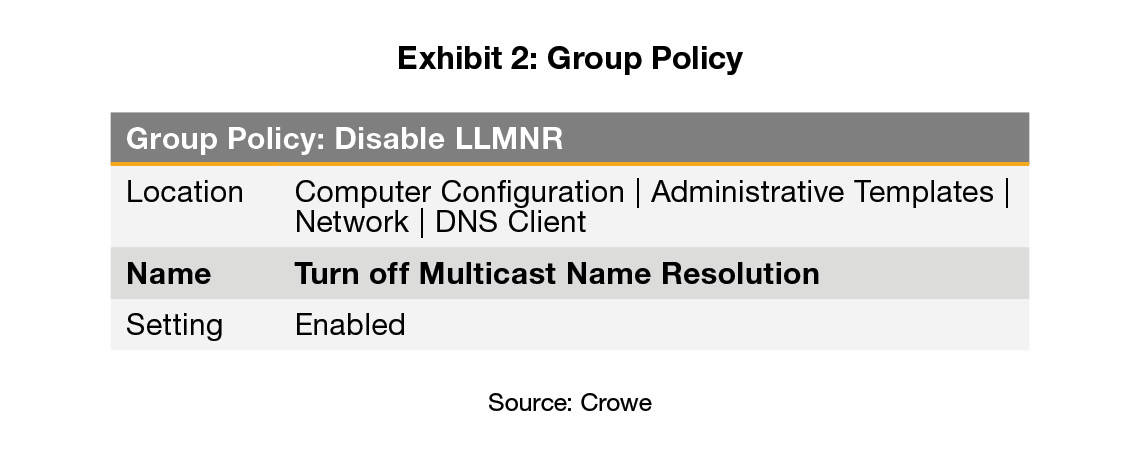
* NetBIOS or LLMNR must be enabled on the victim computer. Additionally, the firewall on the victim computer must allow this traffic to the machine, which by default uses ports UDP 137, UDP 138, TCP 139, TCP 5355, and UDP 5355.
* The attacker system must be on the same network segment (local subnet) as the victim computer. In essence, the attacker computer must be able to “see” the traffic of the victim computer.
* Once the username and hashed password are captured, the attacker must be able to “crack” the hashed password or be able to reuse those credentials on another system.

Remediation

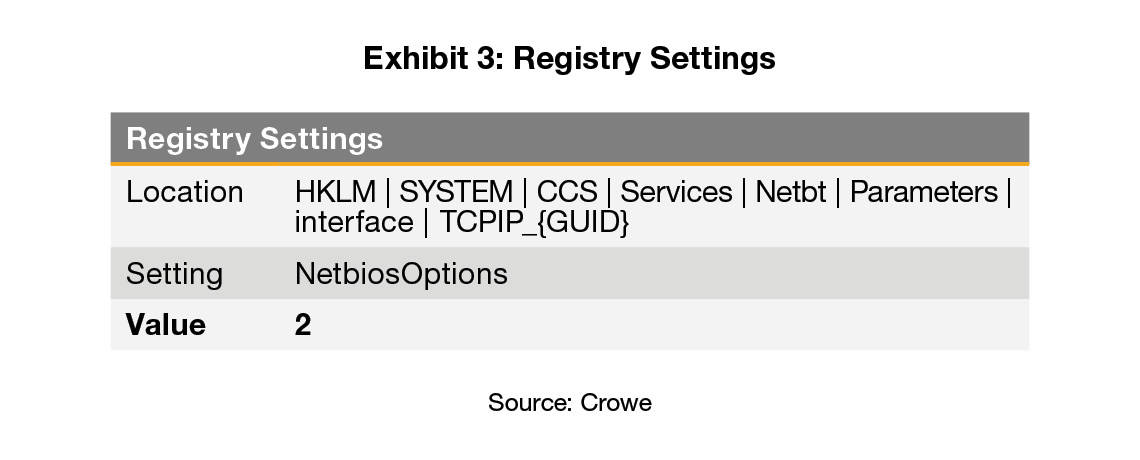
Now that we understand the grave implications of leaving these protocols enabled, how do we get these off the network?

Since most enterprise networks use dynamic host configuration protocol (DHCP), NetBIOS name resolution can be disabled through DHCP server options that apply to each network interface card (NIC) that requests an IP address, such as the network adapter on each computer. The Microsoft support site [describes](http://support.microsoft.com/kb/313314) the detailed steps to accomplish this on each DHCP server.

As for LLMNR, it can be disabled through the Group Policy. The appropriate configuration is to set “Turn off Multicast Name Resolution” to “Enabled” as follows:

****

As an alternative to disabling NetBIOS through the DHCP server(s), one can also create a one-time logon script to run on each Windows device by configuring the following registry key value “NetbiosOptions” to “2” as follows:

****

In addition to disabling NetBIOS on the NIC of each computer and through DHCP and disabling LLMNR, the outbound NetBIOS and LLMNR traffic should be restricted on the host firewall of each system by blocking the NetBIOS protocol and TCP port 139 as well as the LLMNR UDP port 5355. This step can prevent any NetBIOS or LLMNR traffic from accessing or leaving the computer, even when the device is taken out of the corporate network and connected to less secure public networks.

Lastly, in terms of the third condition of this attack, if user passwords are too strong for attackers to crack, then they cannot do much locally with the NetNTLMv2 hashed credentials they have obtained via these protocols. According to previous posts on this blog discussing [password expirations](https://www.crowe.com/cybersecurity-watch/nist-password-expirations/) and password policies, once a password reaches a length of around 16 characters and does not consist of solely common dictionary words, it becomes exponentially harder to crack via the methods described in this post. As for the possible relay of credentials, server message block (SMB) signing should be put in place where possible to prevent the relay of credentials in the NetNTLMv2 format across the network. For more details about the protections offered by SMB signing, check out this [post](https://blogs.technet.microsoft.com/josebda/2010/12/01/the-basics-of-smb-signing-covering-both-smb1-and-smb2/) on Microsoft’s TechNet blog.

Although the options discussed here are not the sole answer to addressing the vulnerability of name resolution protocols, they’re a good layer to add to your security controls and strengthen your environment, even once these overly-generous protocols are incapacitated.

The Takeaway

Many features of the Microsoft Windows operating system are designed for ease of use rather than security. The NetBIOS and LLMNR name resolution protocols enabled by default are just another example of features designed to make connectivity easy for end users but that also open the door for attackers on your network. Evaluating these default configurations and disabling those that are not absolutely necessary to create a secure computing baseline is a critical step to any workstation deployment in an enterprise environment. A great starting point would be the examples of baselines provided by the [Center for Internet Security](https://www.cisecurity.org/cis-benchmarks/" \o "Center for Internet Security" \t "_blank), the [National Institute of Standards and Technology](https://nvd.nist.gov/ncp/repository), the [U.S. National Security Agency](https://www.iad.gov/iad/library/ia-guidance/security-configuration/operating-systems/index.cfm) (you will need to install the Department of Defense's root certificate to view the site securely), and, of course, [Microsoft](https://blogs.technet.microsoft.com/secguide/). Hopefully this post has highlighted at least one strategy that brings you one step closer to the secure baseline that’s right for you and your organization.