Network Anonymity Through "MAC Swapping"

A Sayler

December 21, 2011

Abstract

Due to numerous legal challenges, universities and other administrators of large managed networks have been routinely forced to turn over network usage records and match network activities to specific users. Most of these managed networks authenticate and identify users of the network based off of their MAC address, requiring users to register MAC addresses that they may be using and associate them with their user accounts. All of a user's network activity is associated through a user's registered MAC address and the IP address which it has been assigned. MAC addresses, however, are not static, and changing one's MAC address (or assuming the MAC address of an alternate network user) is a trivial operation. This article will discuss some methods of exploiting MAC spoofing to gain anonymity on university, corporate, or similar networks. We will also explore the legal ramifications of using MAC addresses as proof of user identity given the availability of such methods.

Contents

1 Introduction

The rise of the Communication Age, built atop the ubiquitous digital networking technologies of the late 20th century, has redefined anonymity within our society. We now live in a world where one can publish or share their ideas with the planet without needing to reveal or prove their identity.

But how anonymous really is this Internet that we have built? At some point, most of us have to pay our ISP for access to the Net, and thus in most cases must reveal our identify for billing purposes. On public, corporate, or university networks, users are often required to register the devices through which they access the Internet, adding another means of identification.

While anonymity can certainly be abused, the ability to operate and speak anonymously is a fundamental and essential tenet underlying the freedom of information and expression. From DMCA violation enforcement to censorship and monitoring, the ability of users to remain anonymous, or lack thereof, has a profound impact, and one that must not be taken lightly.

Let's dive into what Internet anonymity means and the discussion of a neat trick for helping to obtain it (at least on school, cooperate, and similar registration based networks).

Note: The techniques discussed here are designed to work on school, corporate, or public networks where users connect directly to the network via a NIC. These techniques will not work to gain anonymity on your home cable, DSL connection, or other private connection for reasons that should become obvious below.

1.1 Enemies of Anonymity

Allowing users to remain anonymous makes them far more difficult to control. Thus, there are many groups with a vested interest in eliminating network anonymity. From the RIAA and MPAA and their "Takedown" notices to various governments and corporations, there is no shortage of those who will strive to unmask users of the Internet. Often, these organization will leverage the legal system to force ISPs or other network operators to give up the identities of their users. Due to billing necessities and basic practicality, we often must seed our identities to our ISP, networks admins, or other organizations, and when these organizations can be forced pass this information on to anyone with the right lawyers, maintaining anonymity on the Net can be very difficult.

Still, the ability of network operators to reliably match actions to known user identities is not guaranteed. To see how one might retain their anonymity on the net, we must understand the basics of the network underlying technology.

1.2 Ethernet, IP, and DHCP

Ethernet was developed by Robert Metcalfe at Xerox PARC in the early 1970s. Ethernet embodies the physical and link layers of the TCP/IP network reference stack.

It is by far the most common system for networking computers, both within local network installations and as part of the wider Internet.

Ethernet assigns each physical node on the network a link layer address called a Media Access Control (MAC) address. MAC addresses are 48 bit (6 byte) addresses that are generally assigned to each physical Ethernet interface at the time of it's manufacture [6]. Thus, an Ethernet device normally has a single, permanent MAC address free from the need for any specific user configuration or selection.

Despite this permanent 1:1 intent, most devices allow the user to programmaticly modify their MAC address. Sometimes this is a necessary feature to enable fail-over operation in redundant multi-Ethernet device configurations. Other times, it is the means for enabling Ethernet multicasting and other advanced configurations [6].

While Ethernet is the standard link layer protocol, it is not well suited for inter-network communication. Thus, we use the IP protocol to facilitate Internet communication. IP addresses, unlike MAC addresses, tend to be user or system defined, and are often dynamically allocated.

The DHCP provides a widly used means to automatically assign IP addresses to Ethernet network devices. It does this via a client/server system in which the client identifies itself via its MAC address and requests an IP address. The server then provides the device with a valid IP address based off either a preexisting assignment for the given MAC address or by selecting the next available IP address in an internal pool. Thus, the DHCP system defines a relationship between a devices Ethernet MAC address and it's Internet IP address. The details of DHCP are most recently defined in RFC 2131 [1].

1.3 MAC Authentication

Many network operators take the MAC/IP relationship a step further by using MAC addresses as a form of client identification. The rational behind this approach is that MAC addresses are normally permanent, whereas IP addresses are assumed to be dynamic. Thus, a user's MAC address can (supposedly) be used as a constant identifier for the user on the network.

In such a system, when a user connects to the network, the network checks the device's MAC address against a table of known MAC addresses for registered users. If a match is found, the network assigns the device an IP address allowing it to communicate on the Internet. If no match is found, the user is normally placed in some form of temporary IP sandbox where no external communication is possible beyond allowing the user to identify themselves to the network operator and register their MAC address.

Figure 1 shows a model for implementing just such a system. Such MAC authentication systems tend to by tightly integrated with the standard DHCP system. They simply add an additional component that validates MAC address before issuing a DHCP response.

Many public, university, and corporate networks use this approach. When a user first accesses the network from a specific device, they are required to provide some from of personal authentication (user credentials, id, etc) before their device is allowed

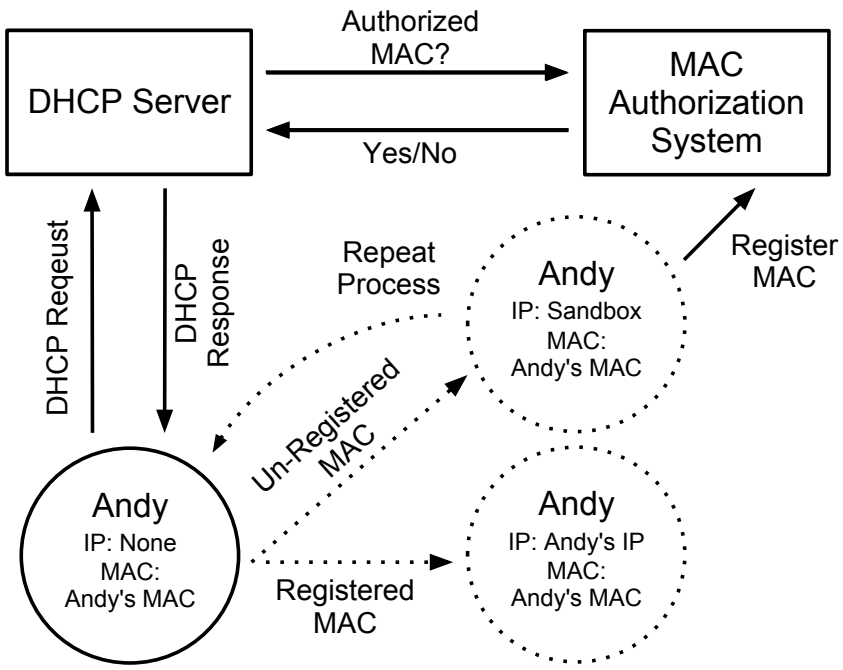

Figure 1: MAC Address Based Network Authentication Model

to connect to the network. I'm sure many of you have encountered the ubiquitous "Please Register" screen when connecting to some form of public network at some point in your life. The DHCP/MAC Authentication system then permanently associates the now registered MAC address with the given user.

MAC address validation and authentication systems like this not only allow the network operator to insure that only paying/authorized users have access to their networks, they also allow the network operator to track network traffic back to specific users. Since the network operator now has a temporal record of which MAC addresses where assigned which IP addresses, and the users to which these MAC/IP combinations belong, the network operator can, theoretically, match any user to their public IP based on the DHCP records and the time the IP was in use.

This, of course, assumes a permanent and 1:1 relationship between users and MAC addresses, which, as we previously mentioned, is not always true.

2 Exploiting the Link Layer

So what happens when we violate the permanent $MAC \Rightarrow User$ relationship that we previously discussed? What can we gain by exploiting the assumption that a MAC address always corresponds to its original user? How easy is it to "steal" another user MAC address and assume their identity on the network?

2.1 MAC Harvesting

We'll start with the latter question first. Since a user's MAC address is present in every Ethernet frame on the local network, harvesting a list of registered MAC address for a given network is relatively easy. Simple sniffing tools like Wireshark or TCPDUMP can lead to large lists of valid MAC addresses.

Furthermore, every client on a network is required to maintain a list MAC addresses for other devices it has communicated with on the network as part of its ARP table. ARP tables maintain the local listing of MAC to IP address mappings and are a key part of any TCP/IP stack implementation. We can artificially enlarge the size of our ARP table to include the MAC addresses for an arbitrary set of clients on the network by ping sweeping a segment of the IP network using a tool like nmap. Dumping the resulting ARP table entries provides a list of MAC addresses for all reachable clients.

Thus, we see that obtaining a list of registered MAC addresses on a given network is relatively trivial for any user of the network. The user gathering these addresses won't have any knowledge of the MAC to User mapping of the addresses, but they will know that the MAC addresses have been successfully registered since they are active on the public network segment.

2.2 MAC Modification & Spoofing

What about modifying this supposed "permanent" MAC address? That too turns out to be fairly trivial (depending on one's operating system and NIC). There are perfectly legitimate, and often required reasons, for changing a MAC address. Indeed, the Ethernet specification [4] even requires MAC addresses to be changeable.

Changing one's MAC address can generally be done at either the hardware (NIC) or software (OS) level. This is due to the fact that most NIC drivers allow the OS to either pass them a full Ethernet frame complete with a source MAC address already filled in, or to pass them a frame with a blank MAC address to which they insert their own address.

On Linux, changing your MAC address at the OS level is trivial. Simply use the `ifconfig` command with the `hw ether [MAC ADDRESS]` argument. This will modify the MAC address for a specific NIC until the next reboot. Most Linux distributions also provide some means by which you can permanently change your MAC address (so it persists between reboots) through the use of a network interface configuration file.

On Windows, some NIC drivers allow you to set your MAC address via the device properties menu. When this option is not supported, there are numerous third party tools that can be used to change your MAC address.

2.3 Avoiding Detection

Since the whole point of this MAC modding dance is to avoid giving your network operators the ability to track your actions, we should discuss how to undertake such

a process without being detected.

The first place where one risks detection is in the harvesting of a set of MAC addresses. Active network sniffing can often be detected since it requires the user to perform some form of ARP poisoning or other technique that fools the local router into forwarding the client traffic that does not involve her. Passive network sniffing only works on un-switched networks (which, in this day and age, is primarily only wireless networks). And even passive sniffing can often be detected (if less reliably) through the use of anti-sniff products that try to identify sniffers through the extra network latency they cause for the client running them.

Pure ARP based MAC harvesting is completely transparent since the ARP process is a natural part of the TCP/IP model. That said, your ARP table normally only contains devices that you have directly communicated with. This provides a possible means for tracking a spoofed MAC address back to a specific user through the set of all devices with which the user has communicated and from which the user has had the opportunity to harvest MAC addresses. To increase the size of one's ARP table to the point where this becomes infeasible, we often employ techniques like ping sweeping, which can also be detected.

So how does one most readily avoid MAC harvesting detection? Three options seem most tenable:

Offline Wireless Sniffing: Many newer wireless chipsets include a "monitor" sniffing mode where they simply act as wireless radios reporting all traffic they see flying through the air. In this mode, they never actually connect to the wireless network, and thus provide no means to trace their actions through latency or other means. Indeed, there is no record of these devices even having existed as far a the network in concerned.

Long Term ARP Collection: By constantly collecting and logging the MAC address of all devices with which you have ever communicated, one can generate a large MAC collection over a period of time. While this suffers from the same theoretical tracking vulnerability as using this approach in the short term, once one's collection of MAC addresses grows large enough, practical tracking become unlikely, if not impossible.

Cooperative Compilation: What if a group of network users get together and share their MAC addresses with each other in person (or via secure communications)? Now we have a collection of valid MAC addresses with no network based record of these users ever having had access to each others MAC records. More on this later...

Even if we can evade detection on the MAC harvesting front, we must still evade detection on the spoofing front. To do this, we must be careful how we connect to the network with our spoofed address. First a foremost, physical, wired connections in private areas (dorm rooms, offices, etc) are to be avoided. These locations provide a means for tracking traffic back to its physical source, and if that's your desk, your cover is blown, spoofed MAC address or otherwise.

Wireless networks seem to be the more robust choice for a successful undetected spoofing attempt. But even here we must be careful. If one suddenly changes their

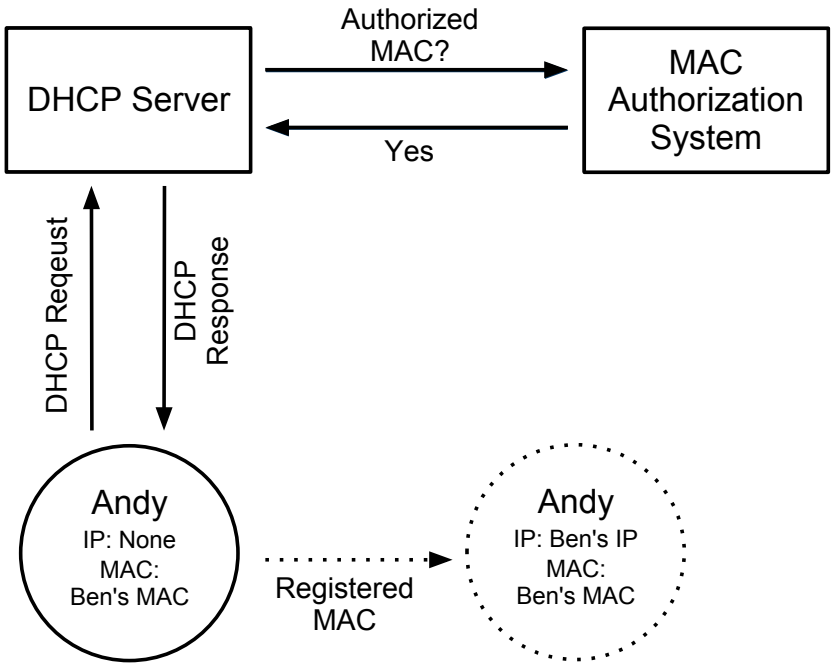

Figure 2: Result of MAC spoofing in a DHCP/MAC Authorization system

MAC address and then reconnects to a small wireless network to which they were previously connected, they risk exposure by temporal correlation. I.e. "Client A disappears from the network and a moment later Client B Appears" is a behavior that could be correlated over time to lead back to the spoofing user. Thus, only spoofing on large wireless networks and allowing some downtime between connecting to the network from ones real vs one's spoofed MAC address are desirable actions.

Finally, what happens if we try to spoof a MAC address that is already in use on the network? In Ethernet and DHCP land, that's generally an implementation specific behavior. Often it will result in a broken network connection for both the actual MAC holder and the spoofing party. It is also an obvious red flag that spoofing is occurring. Thus, it behooves us to insure the MAC address that we are assuming is not already in use on the network at the time they wish to use it, and to insure that this remains true through our entire use of the address. Remember, our goal is operation anonymity, not a DOS attack.

2.4 Results & Consequences

So what does the ability to harvest and modify MAC addresses buy us? By itself, not much. Indeed, one's MAC address is rarely present at the endpoint of a packet sent over the Internet since MAC addresses are part of the local Ethernet network. They get blown away and replaced each time a packet traverses to a separate segment of the IP network (i.e. The Internet).

Where a new MAC address buys us ground is in the fact that under DHCP/MAC

Authentication systems, changing our MAC address also changes our IP address, and thus the user to which all of our network interactions point. Figure 2 shows the result of assuming another user's MAC address (i.e. Andy assumes Ben's MAC) in such a system.

Now, as far as the network operator is concerned, any action Andy takes will be attributed to Ben. Thus, we have gained a form on anonymity through our use of MAC spoofing. Our network actions are no longer associated with us. By frequently changing the user whose MAC address we have assumed, we can increase this level of anonymity.

While this technique provides a form of anonymity, it is also a form of impersonation. In situations where we have not obtained another user's permission to use her identify, we are treading on what is probably unethical (legal or otherwise) ground. Anonymity at the expense of others is not our goal. We will address this issue in the next section.

3 Building an Anonymous Network

How do we leverage MAC spoofing to gain anonymity without treading on the rights of other network users? The key is cooperation with other users. Each network user in the DHCP/MAC Authentication paradigm is required to register his or her MAC address once. Once registered, the user's MAC address has free access on the network. There is no compelling reason or benefit to retaining your own MAC address after you have registered it if you have access to another registered MAC address. How can we exploit this fact?

3.1 Cooperative Spoofing

The answer is, "by trading MAC addresses with other registered users". The more, the merrier. And it's best if you don't even know with whom you are trading. This turns the MAC Authentication paradigm on its head. The network operators can still require users to identify themselves to gain their initial network access, but if the users then jumble their MAC to User associations, this initial identification can no longer lead to future association.

In it's simplest implementation, such a system could be built through real-word "MAC swapping parties". Such parties would involve a group of authorized network users gathering together, each with their single authorized, and traceable MAC address in hand. All users at the party then throw their MAC addresses into a hat and take turns drawing new MAC addresses. Now each user has the means to access the network without their actions being traced back to them. Indeed, they don't even know who specifically their actions trace back to. They can now use the network secure in the fact that they have a sound alibi that breaks any MAC to User associations the network operator may try to assert. Figure 3 illustrates this concept.

Want more anonymity? Increase the number of people at the party. Hold parties each week. Swap early and swap often. Welcome to the great MAC to User

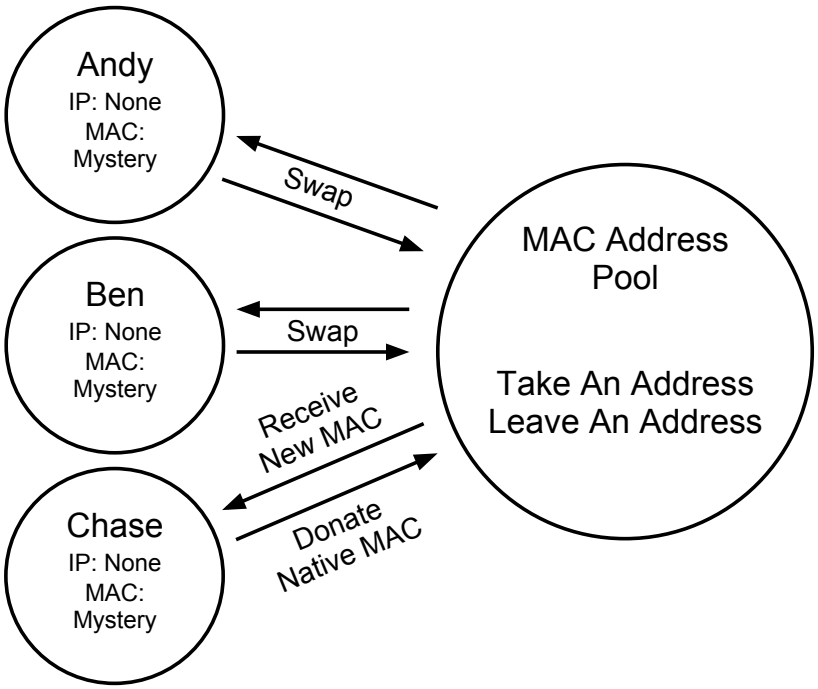

Figure 3: A "MAC Swapping Party"

randomization system.

While such a system is certainly effective in subverting the operator's ability to associate MAC addresses, with specific users, it has some impractical consequences. Namely, it's difficult to organize a group of people to meet frequently and perform a swap. It also ignores the fact that changing one's MAC address is a process that, while simple, isn't widely understood by the average user. Can we automate this process to make it simpler to join?

3.2 Automating the System

Imagining a system that automates the swapping of MAC addresses is not difficult. While there are many considerations in implementing such a system, conceptually, the process is the same as a physical "swapping party".

A good automated MAC swapping system would involve some piece of software that users could install on their computer. This software would record the users current MAC address, and then report this MAC address to a central or distributed MAC pool. In return, the software would receive a new address from the pool. The software could be configured to perform this swap at each boot.

The software will need to employ some form of encryption to avoid revealing which MAC address was volunteered to the pool. The pool would also need at least a few spare MAC addresses to insure a free MAC address is always available for each swap. Obtaining such addresses, however, is not difficult since most MAC Authenti-

cation systems have some means for allowing users to register arbitrary device MAC addresses (for your iPod, Kindle, etc). This means a few users would just have to register a handful of "fake" MAC addresses and volunteer these to the pool to create a small buffer.

3.3 Results & Consequences

By making the MAC swapping process simple and automatic, we can drastically increase the number of users participating in the system. This, in turn, leads to greater anonymity. Thus we can create an anonymous network under the MAC Authentication paradigm by destroying the MAC to user associations on which it relies.

But are there downsides? Maybe. In a MAC swapping system we are trading the right to be the sole user of our native MAC address for some level of anonymity through randomization. This means that while our actions won't trace to us, they may trace to another user. Or for that matter, another user's actions may trace back to us. While revealing our MAC swapping involvement should provide a reasonable doubt that the other user's actions are not our own, and thus avoid us taking the blame for such actions in a court of law, it may still lead to short term headaches.

Furthermore, if the network operator decided to ban and punish instances of MAC swapping (legally or otherwise), revealing that you have swapped MAC addresses might get you in trouble even if it avoids you getting blamed for another user's actions.

Obviously we hope that network operators do not chose this course of action. Our system does not violate the basic goal of MAC Authentication: insuring only authorized user can access the network. It only breaks the secondary result of MAC Authentication, the ability to trace user actions back to users. None the less, crackdowns will occur.

The best defense against such a crackdown is in numbers. MAC Swapping can be seen as a form of network activism. Essentially, it represents civil network disobedience. While a small group of MAC swappers could probably be punished or banned from the network, an entire university campus can not. If enough people participate in such a system and demand their right to anonymity, cracking down on all such users becomes very difficult, both practically and politically.

4 Legal Ramifications

Where does large scale MAC swapping leave us legally? Both as users and network operators?

The power of MAC spoofing lies only partially in the ability of one to assume another's network identify. Its power also lies in its ability *to provide a reasonable doubt that a given MAC address does and always has corresponded to a single given user*. By assuming another's MAC address, we can avoid our actions being traced back to us. By claiming that another user may have assumed our MAC address, we

can claim that our supposed actions were not our own. This one-two punch combo leads to a fairly robust legal defense and enough ambiguity to provide reasonable anonymity.

Thus MAC Swapping provides not only a technological exploit to remaining anonymous, it provides a legal defense to attempts to identify network based off their registered MAC addresses. If enough people start participating in large scale MAC swapping systems, we can all reasonably claim that activity matched to our MAC address is not our own, whether it actually is or not.

5 Conclusion

Currently, many network registration systems assume that a user's MAC address is permanent and static. These systems assume that a given MAC address always matches it's original user, and tie the associated IP address back to this same user. This system can be used to identify the user behind a specific IP address and to track a users actions on the network.

Violating this MAC to User assumption, however, is not difficult. Trivial techniques exist for both MAC harvesting and spoofing. These techniques allow us to assume the "identity" of another user on the network by assuming that user's MAC address. This creates anonymity at the expense of violating another user's rights.

We can maintain this anonymous benefit without the unethical side effects by creating an opt-in MAC swapping system. In such a system, a group of users voluntarily swaps MAC addresses in such a manner as to insure that no one knows who's MAC address each user obtained. This scrambles the presumed MAC to User mapping and allows the group of users to operate anonymously with no means to trace an IP back to a specific user in the group.

University and corporate campuses are one of the primary users of these MAC Authentication systems. If this process were automated and carried out on a large scale network, an entire campus could operate anonymously, effectively subverting the ability of the network operators to trace IPs back to users. This network based civil disobedience in demand of anonymity leads to a paradigm shift in the legal approach to identifying users on the network: If IP addresses are no longer good matches to specific users, many of the tenets of the IP identification legal process fall apart. There is no longer any basis to assuming that an IP addresses can be traced back to a single user.

In our ever more interconnect world, network anonymity is an important right. MAC Swapping provides an ethical, practical, and simple means towards gaining network anonymity on MAC Authenticated networks. While it does not guard against all forms of inadvertently availing oneself on the network, it does provide a sound legal and technological basis for preventing network operators from identifying their users. Now we just need to build such a system and see what happens...

6 References

[1] Droms, R.: Bucknell University. Network Working Group. "RFC2131: Dynamic Host Configuration Protocol". March 1997. `http://tools.ietf.org/html/rfc2131`.

[2] Electronic Frontier Foundation. "INTERNET SERVICE PROVIDER SAFE HARBORS AND EXPEDITED SUBPOENA PROCESS IN THE U.S. DIGITAL MILLENNIUM COPYRIGHT ACT AND RECENT BILATERAL FREE TRADE AGREEMENT". `https://www.eff.org/files/filenode/FTAA/ISP_june05.pdf`.

[3] Electronic Frontier Foundation. "Unsafe Harbors: Abusive DMCA Subpoenas and Takedown Demands". September 2003. `https://www.eff.org/wp/unsafe-harbors-abusive-dmca- subpoenas-and-takedown-demands`.

[4] IEEE. *IEEE 802.3: CSMA/CD (Ethernet) ACCESS METHOD*. 2008. Accessed 05/12/11 at 1330. `http://standards.ieee.org/about/get/802/802.3.html`.

[5] Mitchell, Bradley. "The MAC Address". Accessed 05/12/2011 at 1340. `http://compnetworking.about.com/od/networkprotocolsip/l/aa062202a.htm`.

[6] Wireshark. "Ethernet (IEEE 802.3)". 2011-04-25 22:24:29 Edit. Accessed 05/12/11 at 1335. `http://wiki.wireshark.org/Ethernet`.

A MAC Harvesting and Spoofing Tutorial

This section lays out a basic tutorial for harvesting a collection of MAC addresses on a network and assuming another client's MAC address. This technique will employ ping sweeping, which, as mentioned in this article, is traceable. I recommend you only employ these techniques on a network on which you have the right to experiment. Furthermore, utilizing another user's MAC address without their permission is often unethical and in some cases illegal. Don't be evil.

The MAC spoofing techniques discussed here would also work in a MAC swapping scenario where no harvesting is necessary. All techniques discussed here were undertaken on a Linux system and 802.11 Wireless network.

1. Identify where you are on the network through `ifconfig`. You are most interested in your IP address and subnet.

2. Find and ping all active devices on your network subnet using the information from the previous step. A command like `nmap -sP -n 192.168.1.0/24` will perform this step for an computer on the 192.168.1.0 network with a subnet mask of 255.255.255.0.

3. Flush the ARP cache to record the MAC addresses of all active devices on the network. This can be done using the `arp -n -H ether` command. Pipe the output from this command to a file for easy searching later.

4. You must now wait for a device to drop off the network. Wait a few minutes and then run the `nmap` command from the previous step a second time. It may be helpful to pipe the outputs from both calls to `nmap` to files for easy comparison.

5. Compare the output to the previous `nmap` output. If an address appears in the first nmap listing but not the second, it's a good indicator that the device is no longer on the network. Note any such addresses. If no clients have left, wait a bit more and try again (or put your haxor skilz to good use and write a little script to test regularly and automatically).

6. You now must locate the (presumably) available MAC address for one of the clients who has left the network. To do this, search the previously dumped ARP table for the IP address in question. The `grep` command is your friend.

7. Presumably, you now have harvested a usable MAC address. To assume this user's identity on the network, we must spoof this MAC address.

8. The first step to spoofing the acquired MAC address is to power down your wireless network interface, generally `wlan0`. This can be accomplished via the `sudo ifconfig wlan0 down` command.

9. Once powered down, use ifconfig to replace your native MAC address with the acquired MAC address. The following command accomplishes this:
   `sudo ifconfig wlan0 hw ether <MAC ADDRESS>`.

10. Finally, you will power your wireless interface back up and attempt to connect to the network. This can be done via the `sudo ifconfig wlan0 up` command.

11. Once connected, you can verify that you have successfully switched MAC, and thus also IP, addresses. Calling `ifconfig` will verify this. Note that you will often be assigned the same IP address as the former user of your acquired MAC address. This is a byproduct of the DHCP operation on many networks.

12. This change is not permanent, but will instead only exist until the next reboot. If you were swapping MAC addresses as opposed to harvesting one, you would now make your MAC changes permanent by adding them to the appropriate interface configuration file. On Linux, this is distribution dependent, but a quick search of the Interwebs can provide you with the necessary steps.

13. Welcome to your new life as another user!