An Alternative Approach to Identifying Stolen Network Clients Using DHCP

by

Christian Roy,

School of Computer Science,

McGill University, Montreal

A thesis submitted to McGill University in partial

fulfillment of the requirements of the degree of

Masters of Science

October, 2005

To Mom and Dad, thanks for everything.

Abstract

Computer theft is a mounting problem despite prevention strategies already in place. The marketplace currently offers post-theft solutions but most depend on software in place on the stolen item. This is a critical vulnerability as it places the theft recovery technology in the hands of the thief.

This thesis is an investigation into the feasibility of abandoning this limitation by moving the theft recovery technology into the server, beyond the hands of the thief.

The contents are divided into two complimentary sections. The first is the creation of a theft detecting DHCP server with acceptable performance levels for production environments. The second involves investigation into the ramifications of widespread adoption of such a server by exploring the countermeasures open to an adversary. Questions of difficulty and reliability of such techniques are explored, in both hardware and software. Also discussed is the possibly of countering such countermeasures.

Sommaire

Le vol d'ordinateur reste un problème en dépi des stratégies de prévention déjà en place. Plusieurs solutions pour retracer un objet volé existent, mais elles dépendent généralement de logiciel installé sur celui-ci. Ceci est une vulnérabilité critique car elle place le logiciel dans les mains du voleur.

Ce mémoire explore la possibilité de dépasser cette limitation en déplaçant le logiciel vers le serveur, hors de la porté du voleur.

Le contenue se divise alors en deux sections. La première couvre la création d'un serveur DHCP permettant de retracer les ordinateurs volés avec niveau de performance adéquate pour un environnent de travail. La seconde couvre les conséquances si un tel serveur entrait en service en explorant les méthodes possibles pour déjouer notre serveur. La difficulté et la fiabilité de ces méthodes sont analysées. Une section finale couvre les possibilités pour contrer ceux qui tenteraient de subvertir les techniques que notre serveur utilise.

Acknowledgements

There are a great many people I would like to thank for making this document possible.

First and foremost I would like to thank Professor Claude Crépeau for providing the original idea and encouraging me to see it to the end. Thank you Professor, for not only making it interesting for me but also for all the occasions when you treated all of us in the labs like more than mere students.

Secondly, I wish to thank the Laboratoire Universitaire Bell for their generosity in funding this thesis. Without them this document would not exist.

The good people at ISC deserve recognition for not only providing the internet with a suite of infrastructure software but also for providing such products in an open source form.

I am also thankful to Ville Aikas, of the University of Washington, for patiently responding to my frequent queries about his DHCP testing suite, which served me well during the evaluation of the prototype and saved me from having to write it myself. To Edwin Groothuis for information and tools related to DHCP. Also to Louis-Philippe Crevier for the LaTex templates. And to Slashdot user 639698 for pointing out the obvious.

Thanks are also offered to the Hutchison Avenue Software Corporation, in general, and Ghyslain Boisvert in particular for too many things to mention — they know why. And to the Intuit corporation for its generous employee compensation.

Thanks also go to Etienne Mineau, Simon Pierre Desrosiers, Geneviève Arboit and Carmen Laht for proofreading.

I wish to thank my examiners Professor Claude Crépeau and Professor Muthucumaru Maheswaran for providing me with their time and comments.

And finally, McGill University for offering its graduate students many employment opportunities.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Theft of Computer Hardware

Since 1995 the Computer Security Institute (CSI)[1] in conjunction with the Federal Bureau of Investigation (FBI) has published the "Computer Crime and Security Survey". The report tracks and analyses the cost of various types of computer crimes perpetrated against American enterprise, government and educational institutions [40]. The 2003 results indicate that "laptop theft" cost American business on the order of 6,830,500 USD [40]. The cost of a single theft goes well beyond the cost of replacing the hardware with estimates placing the value of lost data, lost productivity and technical services at 3,957 USD per incident [9]. Added to these costs is the price of the average portable computer, which currently ranges from 500 to 3,000 USD. With the combined loss reaching into the high four figures, even a single theft can be disastrous to a business or individual. However, many of these losses could be mitigated by having a theft solution in place.

[1] See page 89 for a list of acronyms.

1.2 Current Theft Solutions

There are a number of computer theft protection products on the market. These generally belong to one of two distinct categories. The first category is theft deterrence products. Notable examples include portable computer locks and permanent serial numbers (engraved or affixed). The media fracas, in mid 2004, regarding the vulnerability of pinned tubular cylinder locks, often used in portable computer locks, clearly illustrated that the faith placed in such deterrence is often misplaced [28]. These items may hinder an opportunist but will likely fail to deter a professional thief.

The second category is theft recovery products. This includes any product that actively offers the possibility of recovering the item after it has been lost or stolen. While serial numbers may aid in recovery, they play a passive role at best. Products in this second category use the stolen computer's network connection, be it wired or wireless, or in some cases a modem and a phone line, to communicate with a predetermined recovery service provider.[2] Often these products rely on established protocols such as email or the web but some have deployed proprietary protocols in order to better hide their nature. If a computer with this technology is stolen the owner reports the theft to the recovery service provider. The next time the computer contacts the recovery provider they determine its location from the contact vector and provide the owner and law enforcement with the location data. Here, the inherent weakness is not the dependence on computer interconnectivity but that the recovery technology is placed in the stolen item itself. As the thief has physical access to the stolen item he can, with the appropriate tools and time, disable or remove the recovery technology.

[2]In most cases, however, some products contact the owner directly making them the provider

1.3 Proposed Alternative Approach

With theft deterrence products serving as only a barrier to opportunists and theft recovery products at risk of compromise; then what, if any, solutions are left? Our proposal is the rehabilitation of theft recovery by eliminating the current weakness and placing the recovery technology at the server side, well beyond the thief's grasp.

There is some previous related work in the field of remote computer forensics. Eight years ago Fydor published "Remote OS detection via TCP/IP Stack FingerPrinting" [13]. The techniques for gathering data about a remote computer on a network have been refined since those first steps. However, as of yet, none of them have yielded a way to reliably and uniquely identify a remote computer.

A recent paper by Tadayoshi Kohno, Andre Broido and K.C. Claffy entitled "Remote Physical Device Fingerprinting" outlines a novel approach for obtaining an identifier from a remote computer: using the computer's network clock skew. The paper provides various proofs of reliability of the techniques and proofs of how stable the identifier is in regard to changes in the physical location and operating conditions of the computer. While promising, the paper disclaims the possibility of using clock skews as a unique identifier: "With respect to tracking individual devices, we stress that our techniques do not provide unique serial numbers for devices, but that our skew estimates do provide valuable bits of information that, when combined with other sources of information such as operating system fingerprinting results, can help track individual devices on the Internet" [45]. It may be that with additional work the research outlined by Kohno, Broido and Claffy will yield a unique identifier but currently their results only provide a reliable identifier. We will therefore have to exploit more conventional techniques such as relying on inbuilt unique identifiers or assigned unique identifiers.

This document is therefore the exploration of our attempt at the rehabilitation of theft recovery technology using conventional techniques. Chapter 2 discusses the basic methodology of the design as well as the challenges that this approach imposes. Implementation of a prototype and the difficulties it raises are discussed in Chapter 3. In Chapter 4, we examine the performance of the finished prototype. In Chapter 5, the possible techniques that could be used to thwart this proposal are examined. And finally, in Chapter 6, we discuss the techniques that could be used to defeat the techniques analysed in Chapter 5.

Chapter 2

Design

This chapter examines the general issues raised by moving the theft recovery technology from client to server. Topics covered include the challenges posed by this move, selection of the technology used and finally the mechanism used.

2.1 Challenges

Moving the theft recovery technology from client to server still leaves us firmly in the server-client model. This creates a series of challenges that must be examined before proceeding.

Out of necessity, we assume as little as we can about the client as this is beyond our control. We cannot require custom software, proprietary protocols or unusual hardware. The only assumption we make is that, at some point, the client will enable some form of network communication. Therefore, we need to design our solution around common hardware components and, aiming for maximum operating system

support, common standard protocols.

With these restrictions in mind, we will limit ourselves to the TCP/IP protocol suite and avoid any operating system specific extensions or enhancements.

2.2 Technology Requirements

There are two major requirements for our theft recovery technology prototype system. The first, as mentioned in Section 1.3, is a globally unique identifier for each computer. The second is a protocol on which to build the recovery technology.

2.2.1 Globally Unique Identifier

Without the ability to uniquely identify a computer, we cannot proceed. This is the first and most basic requirement. Unfortunately, we have precious few such identifiers available in network communication. While IP addresses may be globally unique identifiers, they are usually dynamically assigned. The same is true of computer domain names.

One candidate among the few identifiers is the Media Access Control (MAC) address. The standards documents define the MAC address to be a globally unique identifier assigned to each network interface hardware [16, 23]. This means that computers capable of interfacing with a Institute of Electrical and Electronics Engineers (IEEE) 802 style network will have at least one, if not several, MAC addresses, each of which is a globally unique identifier. Today, IEEE 802 style networks encompass the majority of the commonly used networks (Ethernet, Wi-Fi, Token ring...).

A MAC address is 12 digit hexadecimal number, which is commonly written by grouping the digits in pairs. For example, 00:0D:93:63:CF:30. Every Network Interface Card (NIC) that communicates with a IEEE 802 style network will have its own unique MAC address [17]. As all IEEE 802 style MAC addresses share the same address space, the uniqueness of MAC addresses is not restricted to a single type of network [17]. For example, there should never be a Wi-Fi NIC with the same MAC address as an Ethernet NIC.

Given these properties, it seems that the MAC address is an excellent choice for the globally unique identifier.

2.2.2 Client Server Protocol

The second requirement, after the selection of a unique identifier, is the selection of a client-server protocol to work with. There is a plethora of possible choices, from AEP to TFTP. The problem with many of these protocols is that a computer connected to a network is under no obligation to use most of them. Therefore we need a protocol that a computer on a network is almost guaranteed to use. The list of such protocols is limited.

One feature of TCP/IP networks is the requirement that a computer connected to such a network have an IP address. Without such an address the computer cannot truly participate in the network. Though it is possible to have a static IP address, this is only viable in such cases where the computer is permanently connected to a network and only common in institutional contexts. In most cases, IP addresses are dynamically assigned to a computer when it joins a network.

A common mechanism for dynamic IP address assignment is the Dynamic Host

Configuration Protocol (DHCP). Many large Internet Service Providers (ISPs) use the DHCP protocol, though some may continue to use DHCP's ancestor Bootstrap Protocol (BOOTP), these will be fairly rare. ISPs offering some form of Point-to-Point Protocol (PPP) access, be it dial up or Point-to-Point Protocol over Ethernet (PPPoE), do not use DHCP but use Internet Protocol Control Protocol (IPCP) instead.

The majority of modern operating systems include a DHCP client. Microsoft's Windows operating systems have included a DHCP client from 1993 onwards [32, 10]. Similarly, Apple operating systems have also included such a client since mid 1995 [1, 53]. In fact, ease of use, especially for the client has made it practically de rigueur. Almost any product designed for the modern home network, from Tivoes to Xboxes, includes a DHCP client.

The procedure for obtaining an IP address via DHCP is fairly simple. When a computer joins a network it broadcasts a DHCP request message whose format is laid out in Figure 2.1.[1] Upon receiving such a message, the DHCP server offers the client temporary use of an IP address, a lease. This makes DHCP interesting as each time a stolen or lost computer joins a network it will make such a DHCP request because obtaining an IP address is necessary.

The DHCP message, as shown in Figure 2.1, contains a useful piece of information in the "chaddr" field. The Request For Comments (RFC) defining DHCP describes this as the "client hardware address" [35]. Conveniently, this is the client's MAC address that we have chosen as our globally unique identifier.[2]

All of these qualities combine to make DHCP an excellent selection as the client-

[1] See Section A.3 for an expanded explanation of how an IP address is obtained via DHCP.

[2] See Section A.4 for a discussion why the "chaddr" is preferred over the link layer address.

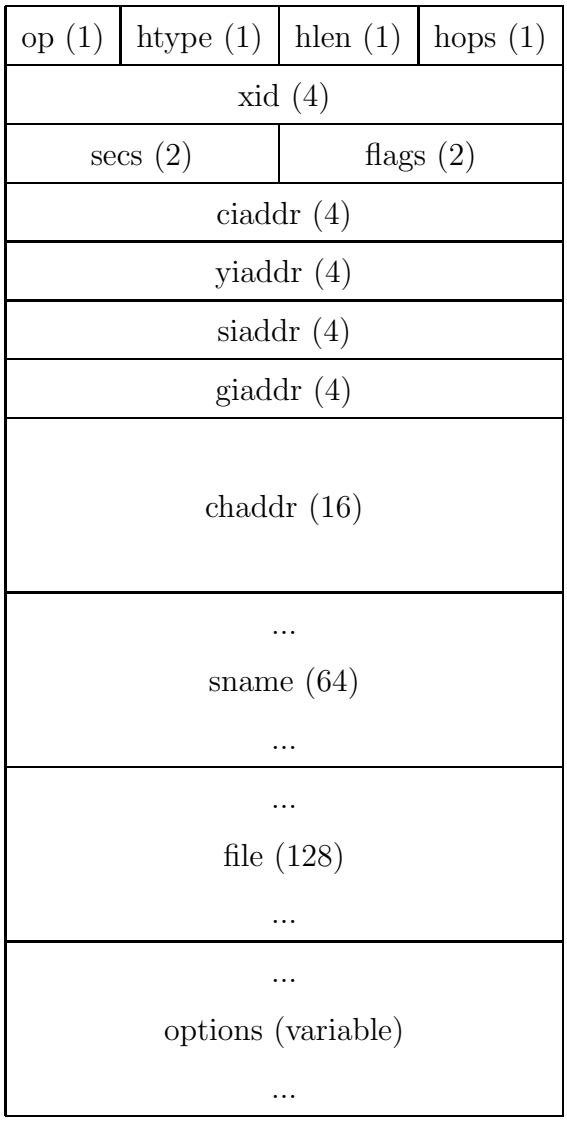


Figure 2.1: Format of a DHCP message [35].

server protocol for our experiment.

2.3 The Blacklist

The chosen mechanism for implementation is a blacklist. An example of this would be lists cataloguing ISPs refusing to control their mail volume. In our case, we can apply the same monitoring device to our problem. A globally distributed list of MAC addresses tied to missing or stolen hardware would be maintained with accompanying information for each entry. Such a blacklist, in combination with a DHCP server monitoring client MAC addresses could be used to detect any clients on the DHCP server's network that report matching MAC addresses.

Detection is, of course, only the first step, beyond this remains the question of tracking down the blacklisted hardware. Is it possible, with only the DHCP data to track down the hardware on the network? This is a question that must be answered by each DHCP server installation.

The frontline in this effort would be ISPs. ISPs are very tight-lipped about their internal workings and thus it is difficult to comment, in a general fashion, on their capabilities. However, thanks to the Canadian Radio-television and Telecommunications Commission (CRTC), we have some insight into Vidéotron's practices, one of Québec's largest ISPs.

Vidéotron, in documents filed with the CRTC working group on high speed internet, specifies its technique for identifying its subscribers. They add an option, "option 82", containing the MAC address of subscribers' cable modems, to the "options" field[3] of subscribers' DHCP traffic [11]. A subscriber's DHCP message goes from the computer to the cable modem to the Cable Modem Termination System (CMTS), where "option 82" is added, and then onto Vidéotron's backbone network [48]. Making this

[3]See Figure 2.1 for the "options" field and Section A.1 for a discussion of the "options" field.

addition at the CMTS renders the process invisible to the subscriber. The result is that each subscriber's DHCP traffic is tagged with not only the computer's MAC address, but also the cable modem's MAC address.

Vidéotron registers new subscribers by linking their cable modem's MAC address to their account information. Therefore, if a Vidéotron subscriber attempted to use a blacklisted computer, then Vidéotron could use the cable modem's MAC address, found in the DHCP message, to track down the subscriber's account information. This data includes the subscriber's name and address, which could be forwarded to the appropriate authorities.

More generally, ISPs often maintain billing in terms of bandwidth usage and thus it is likely they will maintain a mechanism for linking a customer's IP address given by the DHCP server to the customer's billing data. Beyond this, legal requirements in several jurisdictions demand that ISPs be able to identify customers using IP addresses to facilitate law enforcement activities. We can tie the MAC address to the IP address from the DHCP server's logs, which means there is little doubt that ISPs would be able tie it to their customer records.

Chapter 3

Implementation

This chapter discusses the details of the prototype implementation, analysis of the implementation methods, as well as a discussion of some problems encountered with the implementation.

3.1 Implementation Requirements

While it would be possible to implement a blacklisting DHCP server from scratch there are two major reasons why this is not a desirable proposal. The first and most obvious is that such an endeavour is beyond the scope of this work. The second reason is a desire for maximum participation in this venture. The added cost to the participant of migrating from their current DHCP server to a completely new blacklisting DHCP server is likely to be prohibitive. For these reasons, we have chosen to implement this prototype as a modification to one of the most popular DHCP server implementations in the open source community, the Internet Software

Consortium (ISC)'s DHCP server.

Again, to gain the widest acceptance possible, the implementation will need to mirror the design philosophy of the original code. The ISC website claims that its DHCP package provides an implementation "which is designed to be sufficiently general that it can easily be made to work on POSIX-compliant operating systems and also non-POSIX systems like Windows NT and Mac OS" [22]. Once the source code for this package is examined we find generic, mostly, ANSI C code and few system specific API references. Therefore, the blacklisting functionality will have to be ANSI C code and avoid platform specific API references.

Beyond these requirements, the implementation of the blacklist should have minimal impact on the operation of the DHCP server. The vast majority of requests made against the server will be legitimate in nature and should suffer minimal effects. This means that unless the cost of the blacklist on a per connection basis is not minimized, the total impact could be unacceptably high.

The final requirement for the blacklist implementation is scalability. Despite the fact that computer equipment quickly becomes obsolete, an entry on the blacklist may remain in place for several years. In order to accommodate this fact, the blacklist must handle various list sizes and must cope with an expanding list size.

The feature list included in the blacklisting server is as follows:

- The ability to accept or deny DHCP requests from a blacklisted host.

- Support for additional record information for each blacklist entry such as contact information should the server ever see the blacklisted MAC address.

- Checking both the actual MAC address and the supplied client identifier.[1]
- Support for Windows client identifier unspoofing.[2]

3.2 Implementation Method: Hybrid Trie Indexing

The ISC DHCP implementation uses a single thread to process all incoming DHCP messages. This means that should the thread take too much time on a blacklist lookup all subsequent DHCP messages, waiting in the message queue, will go unhandled until the first message is completed. While this may not seem catastrophic, the problem lies in the DHCP standard's timeout mechanism. The RFC document provides a mechanism for a DHCP client to abandon what it considers to be an unanswered message and retry [35]. However, no such mechanism is defined in the RFC for the server to abandon messages that have waited too long in the queue.[3] Any significant delay in searching the blacklist will cause an escalating backlog of DHCP client messages. If the message queue grows too long and each lookup takes too much time there begins to emerge a cascade failure as clients abandon messages in the queue and retransmit, therefore growing the queue and exacerbating the problem. Eventually, most DHCP clients would abandon fail altogether under whatever mechanism is implemented. Of course, this is an unacceptable situation and any implementation of the blacklist must not interfere seriously with the efficiency of each DHCP request. The goal of "minimal cost on a per connection basis" will therefore

[1] See Chapter 6 for a discussion of the use of the client identifier.

[2] See Section 6.1.1 for a discussion of this technique.

[3] Although the RFC defines no such mechanism the ISC server does implement one. Without RFC guidance they must be careful not to conflict with the DHCP client message abandonment algorithm.

be a major hurdle.

However, only the initial DHCP client request causes a blacklist lookup. The client's status, blacklisted for those in the blacklist or whitelisted for those not, is recorded. All further traffic from that DHCP client is no longer checked against the blacklist, unless a new blacklist is obtained. This helps reduce the cost of DHCP traffic and helps further the goal of "minimal cost on a per connection basis".

After several cycles of implementation and performance testing, the solution selected was a hybrid trie. Unlike the standard trie, in a hybrid trie a leaf node does not necessarily indicate a matched string, or in our case a matched MAC address, but merely the possibility of a match. The search will then proceed to secondary storage for further examination. This balancing between memory searches and secondary storage searches allows for tuning the blacklist performance in such a way as to balance memory and secondary storage searches.

3.2.1 Hybrid Trie Indexing Analysis

Analysis of this hybrid trie requires that some terms be defined. The height of the trie will be called h and the alphabet size a. The alphabet size indicates the maximum number of children a trie node can have. As this trie indexes MAC addresses and not regular strings we can be more particular about our trie structure. We could treat all 12 hexadecimal digits of the MAC address as individual letters to be indexed.[4] However, it's easier programmatically to deal with the MAC address as six pairs of hexadecimal digits. Therefore, the maximum value for h will be six, not twelve, and the value of a will be 256, not sixteen.

[4]Recall that a MAC address is a 12 digit hexadecimal number. For example, 00:0D:93:63:CF:30.

Equation 3.1 captures the total upper bound on the complexity for the search time in a hybrid trie.

$$\mathcal{O}(a * h) + \mathcal{O}\left(a^{6-h}\right) \tag{3.1}$$

The first term is the upper bound of the trie search, which is performed in memory. The second term is the search performed on secondary storage. The goal is to maximize the first term and minimize second term. This would suggest the largest value of h possible. However, Equation 3.2 will place restrictions on h.

Equation 3.2 captures the memory requirements of a hybrid trie.

$$\mathcal{O}\left(\sum_{i=0}^{h} a^i\right) = \mathcal{O}\left(\frac{1 - a^{h+1}}{1 - a}\right) \tag{3.2}$$

Here, memory usage is dictated, primarily, by the height of the trie. It is possible to set h to any value, including the maximal value of 6, but too high a value can turn a hybrid trie into a regular trie. This situation wastes much of the advantage of a hybrid trie.

When building the hybrid trie with a blacklist, the goal is to find a value for h such that the second term of Equation 3.1 is minimized while not causing the value of Equation 3.2 to increase beyond utility.

3.3 The Maximum Branching Assumption

The initial implementation of this prototype assumed that the hybrid trie would achieve a level of branching that was equal to the alphabet size a. However, after the original performance testing it was discovered that this assumption is seriously flawed when indexing MAC addresses. To understand this, it is necessary to explain the mechanism of MAC address assignment. Anyone producing networking equipment compatible with IEEE 802 standard networks must obtain what is known as an Organisational Unique Identifier (OUI) prefix [17]. This prefix is three pairs of hexadecimal digits, which prefixes the MAC address of any device produced by the group in question [17]. Three pairs of hexadecimal digits represents 16,777,216 possible combinations. Unfortunately, there are currently only 8479 assigned OUI prefixes.[5] Thus we end up with the following scenario:

h	Actual Maximum Leaf Nodes	Assumed Maximum Leaf Nodes
1	10	256
2	68	65,536
3	8479	16,777,216
4	2,029,056	4,294,967,296

Table 3.1: Actual vs. assumed possible values of trie branching.

Table 3.1 clearly illustrates the sharp division of assumed branching versus actual branching. Thus, when indexing MAC addresses this means that only once the trie height reaches four or more will the branching of the trie even remotely approach the assumed maximum. This will change over time as the IEEE assigns more OUIs but is unlikely to dramatically increase. The lack of branching has proven not to be a fatal flaw for small blacklists, but as the blacklist size increases, it becomes a more

[5] According to the IEEE as of August 2005.

pronounced problem.

The chosen solution to this branching limitation is to index the blacklist of MAC addresses backwards. The pairs of the MAC address are not reversed but simply indexed backwards. Thus if a forward index would be 00:0D:93:63:CF:30 then the backwards index would be 30:CF:63:93:0D:00. Manufacturers must exhaust 95% of all possible suffixes of an assigned OUI prefix before requesting another [17]. We can therefore assume that the suffix of a MAC address will be random and thus, will conform to the maximum branching assumption.

Chapter 4

Performance

This chapter covers the analysis of the performance of the prototype including success conditions, testing conditions and results.

4.1 Success Parameters

How do we define success in this endeavour? The stated goal of "minimal cost on a per connection basis" needs to be measured in some way. An article by Bruce Bahlmann suggests several appropriate metrics for DHCP servers [5]. His suggestions, adapted for our particular use are:

- Transaction Rate: The number of transactions handled on a per second basis. Transaction here is taken to mean any request-response pair of DHCP message. This includes both DHCPDISCOVER - DHCPOFFER pairs and DHCPREQUEST - DHCPACK pairs.[1]

[1]See Section A.2 for a discussion of message types and Section A.3 for an expanded explanation

- Average Transaction Time: The average time taken between the server's reception of DHCP message and the server sending a DHCP response is measured here. Again, this includes both DHCPDISCOVER - DHCPOFFER pairs and DHCPREQUEST - DHCPACK pairs.

- Average Lease Request Time: Obtaining an IP address, a lease, from a DHCP server is a two-step process. The average time taken during the first step, the time between the server's reception of DHCPDISCOVER message and the server sending a DHCPOFFER response, is measured here.[2]

- Average Lease Acquisition Time: This is the entire time taken up in the obtaining of a lease on an IP address. This is the average time between the server's reception of a DHCPDISCOVER message and to the server's sending of a DHCPACK message.

- Percentage of Failed Transactions: Percentage of transactions that have failed to complete for whatever reason. Transaction here is taken to mean any request-response pair of DHCP message. This includes both DHCPDISCOVER - DHCPOFFER pairs and DHCPREQUEST - DHCPACK pairs. Here, we divide the number of failed requests by the number of requests made, both failed and successful. Generally, the reason for such failures is a timeout. The client gives up waiting on the server, and tries again.

With these metrics we can determine the effects the blacklist lookup is having on the DHCP server.

of how an IP address is obtained via DHCP.

[2]The second step of obtaining a DHCP lease, the time between a DHCPREQUEST message and a DHCPACK message, is not measured, as it does not perform a blacklist lookup, and so should not vary from the original server.

4.2 Testing Environment

Obtaining and configuring thousands of clients was, of course, not possible. Instead, a DHCP client simulator was used to create the illusion of thousands of clients [49]. In our case, we simulated 5000 clients. This number was chosen as it represents 20 Class C subnets.[3] This was the maximum number of subnets available with the networking equipment in use.

The computer chosen to host the DHCP server was a Cobalt Qube 2. This machine is best described as underpowered, being a 250 MHz MIPS processor [7]. The quantity of DHCP traffic on a network is likely to be much smaller than, for example, HyperText Transfer Protocol (HTTP) traffic. Therefore the machines hosting DHCP servers are likely to be much less powerful. The Qube is meant to represent, and exaggerate, this fact in our tests.

4.3 Testing Requirements

In order to test effectively we will need to run the blacklisting DHCP server with a sizable blacklist. The procedure for generating such a blacklist is fairly straightforward. Randomly select one of the assigned OUI[4] prefixes and generate the remaining three pairs of hexadecimal digits at random. The question is, how large should this test blacklist be?

It is incredibly difficult to obtain any sort of meaningful statistics on theft of

[3] A Class C subnet is a network with IP addresses that vary only in the last quad. e.g. 201.68.34.1-255

[4] When the testing was performed, a slightly different list, including some OUI prefixes seen in the wild but not on the official OUI list, was used.

computer hardware. We mentioned the yearly CSI/FBI survey in Section 1.1 but unfortunately, the problem with this publication is that it gives numbers in terms of monetary loss and not unit loss.

The only group we are aware of currently publishing statistics on the number of computer thefts is Safeware Insurance. Safeware Insurance specializes in computer insurance and publishes a yearly report of computer loss statistics for the United States of America. These numbers are extrapolated from actual customers' insurance claims [43].

The original test value for the number of supported records was three year's worth of data for the United States of America. Three year was chosen as a reasonable mark at which the deprecation of the computer makes it almost beyond worth recovering [29]. According to the Safeware's 1996 data, there were 265,000 notebook computer thefts [41]. Rounding up for simplicity we ended with 900,000 records.

The 2001 survey released by Safeware Insurance placed the number of notebook computer thefts at 591,000 [42]. Approximately three years worth of data places us in the range of 1,500,000, which is our second blacklist test size.

This data makes it clear that the implementation must scale fairly well in order to accommodate the likely increasing number of stolen computers in coming years. However, with the depreciation rate of the average computer and various statutes of limitation on theft it is unlikely that a computer's record would remain on the blacklist permanently [29, 8, 47, 4]. With this data in mind we've chosen a fifteen-year upper bound for record retention and have therefore selected the figure of 9,000,000 blacklisted records as our stress testing bound.

4.4 Performance Testing

The first step in the test procedure is an iterative one. A DHCP client simulator is used to create thousands clients which begin making DHCP requests against the server with a pause p between each request. Initially p is set to a large number, something that would generate one transaction per minute.[5] The test continues until all clients are served. The value of p is then reduced and the clients again make their requests. This continues until p is found that places a maximum load on the server without increasing the number of failed transactions to unacceptable levels.

Once the proper value of p is determined then the second step in the test procedure begins. As in step one, a DHCP client simulator is used to create thousands clients who begin making DHCP requests against the server with the pause p between them. This continues until the clients are all served.

This second test is then repeated until five data sets are obtained. After five data sets are gathered, then the metrics outlined in Section 4.1 are calculated then averaged. On occasions where the metrics calculated with five data sets are suspect, for example besting the values of the baseline, then twenty data sets are obtained. This is how the metrics found in the following sections are compiled.

In some cases it takes more than five iterations of the test to obtain five data sets, or twenty as the case may be. This occurs when the server exhibits too high a failure rate. The value of p is chosen to place the maximum load on the DHCP server without causing the server to drop too many connections on average. This is, at best, a precarious position for the server. Any small delay, for whatever reason, be it network congestion or background activity on the server machine, can cause

[5] Guided by experience the initial value for p is rarely set that high.

failures to build up to an unacceptable number on some tests. This means that some data sets are discarded.

4.4.1 Establishing Baselines

A baseline performance test against which to compare the blacklisting DHCP server is required. This baseline was generated by running the unmodified ISC DHCP server trough the procedure outlined in Section 4.4. The baseline data generated is found in Table 4.1.

Metric	Value
Transactions Rate (per s)	39.061
Transaction Time (μs)	347,342.227
Lease Request Time (μs)	538,401.190
Lease Acquisition Time (μs)	5,955,793.465
% of Failed Transactions	0.017

Table 4.1: Baseline metric values for unmodified DHCP server.

4.4.2 Phase 1: Base Case

As mentioned in Section 4.3, the size of our initial test blacklist was chosen to be 900,000 records. Using the procedure outlined in Section 4.4, we generated the following results.

Forward Indexing

The original forward indexing implementation generated the results found in Table 4.2.

Metric	Trie Height (h)			
	1	2	3	4
Transactions Rate (per s)	-	9.942	39.021	39.056
Transaction Time (μs)	-	631,620.671	347,252.834	350,040.386
Lease Request Time (μs)	-	883,567.920	542,226.191	541,102.495
Lease Acquisition Time (μs)	-	6,526,178.288	5,956,636.346	5,948,880.033
% of Failed Transactions	-	0.050	0.052	0.034

Table 4.2: Forward indexing, hybrid trie, metric values for blacklist size 900,000.

There is no data for a trie height of one because of the mentioned trie branching restriction in Section 3.3. With so few nodes in the trie, the performance drops to unacceptable levels, with transaction rates falling to below 1 per second.

The poor showing with a trie height of two compared to the baseline in Table 4.1 is also the result of the branching limits of forward indexing. A trie height of two results in an average of 10,000 records at each leaf node. Searching this many records on secondary storage is prohibitive and results in the degraded performance observed.

Increasing the trie height to three causes the number of records found at leaf nodes to drop by two orders of magnitude, compared to the previous height, leaving the number of records at close to 100. This helps to explain why the increase in performance is so startling between the trie height of two and three. Here we have the unusual situation where the Transaction Time for the trie height of three is

actually less than our baseline metric. Bruce Bahlmann, in the article suggesting our test metrics, explains that the lower value of the Transaction Time is essentially tied to the lower value of the Transaction Rate this way: "[transaction time] will gradually increase as the server and the network becomes increasingly taxed" [5]. A higher Transaction Rate signals that the network is becoming "increasingly taxed" which in turn causes a higher Transaction Time. Since the baseline has a higher Transaction Rate this is perhaps why it has a greater Transaction Time than in this case.

At this point we have matched the performance level found in the baseline test, as outlined in Table 4.1.

There is little improvement in extending the trie height beyond three. With this change, we are merely expending more memory, as illustrated by Equation 3.2, with little gain. In this case however the Lease Acquisition Time is better than the one found in the baseline. A further discussion of this result is found in the next section.

Backwards Indexing

The solution to the branching problem, backwards indexing, generated the results found in Table 4.3.

The performance with a reverse indexing trie height of one provided acceptable levels of performance. Some may regard the lower Transaction Time combined with the lower Transaction Rate observed with a trie height of one as puzzling compared to other trie heights having higher Transaction Times with higher Transaction Rates but the explanation is the same as with a forward indexing trie of height three.

Unlike the forward indexing implementation, increasing the trie height to two

Metric	Trie Height (h)			
	1	2	3	4
Transactions Rate (per s)	32.674	39.034	39.034	39.036
Transaction Time (μs)	343,701.121	349,764.871	347,636.327	349,251.576
Lease Request Time (μs)	546,079.735	541,268.783	540,209.988	541,072.043
Lease Acquisition Time (μs)	5,940,454.860	5,948,086.449	5,929,105.547	5,926,241.145
% of Failed Transactions	0.024	0.033	0.039	0.029

Table 4.3: Backwards indexing, hybrid trie, metric values for blacklist size 900,000.

gives us performance numbers roughly equivalent to those found during the baseline in Table 4.1.

There is little improvement in extending the trie height beyond two and again, as with forward indexing, we are merely extending the memory usage without providing any remarkable performance benefit.

Here with a backwards indexing trie of height two, three and four the Lease Acquisition Times are again lower than with the baseline value. To further compound the oddity of the situation we have the following:

- Trie heights of two and four have similar Transaction Times and Lease Request Times while having dissimilar Lease Acquisition Times.
- Trie heights of three and found have similar Lease Acquisition Times yet have different Transaction Times and Lease Request Times.

Combined with the value of the Lease Acquisition Time found with a forward indexing trie height of four we have a puzzling situation. The solution to this puzzle lies in the nature of the metric. Recall that the Lease Acquisition Time measure the time between the “ server’s reception of a DHCPDISCOVER message and to the server’s

sending of a DHCPACK message". This is a pair of DHCP message exchanges. First a DHCPDISCOVER - DHCPOFFER pair is exchanged which begins the Lease Acquisition Time. After an interval of time the DHCPREQUEST - DHCPACK pair is then exchanged which ends the Lease Acquisition Time. During that interval the number of intervening DHCP requests by other clients can cause the value of the Lease Acquisition Time to fluctuate, as evidenced by the odd lower values we have observed. Given that the Lease Acquisition Time is prone to fluctuation, dependent on the network, it seems that it does not serve as an excellent metric for our testing purposes.

4.4.3 Phase 2: Midrange

This is a midrange test of performance using a larger blacklist size of 1,500,000. Following the procedure found in Section 4.4 with a randomly generated blacklist we obtain the following results.

Forward Indexing

Forward indexing of the midrange blacklist generated the results found in Table 4.4.

In this case, there is no data for both trie heights of one and two as performance in these cases is unacceptable. This is again due to the branching restriction discussed in Section 3.3.

With a trie height of three, we achieve performance similar to that found during the baseline test in Table 4.1. Extending the trie height to four does little beyond using more memory and does not remarkably improve performance.

Metric	Trie Height (h)			
	1	2	3	4
Transactions Rate (per s)	-	-	39.049	39.051
Transaction Time (μs)	-	-	349296.664	350093.035
Lease Request Time (μs)	-	-	540790.631	539816.378
Lease Acquisition Time (μs)	-	-	5972910.136	5959859.791
% of Failed Transactions	-	-	0.037	0.027

Table 4.4: Forward indexing, hybrid trie, for blacklist size 1,500,000.

Backwards Indexing

Backwards indexing of the midrange blacklist generated the results found in Table 4.5.

Metric	Trie Height (h)			
	1	2	3	4
Transactions Rate (per s)	28.103	39.053	39.036	-
Transaction Time (μs)	352,250.202	347,835.085	349,442.480	-
Lease Request Time (μs)	563,871.886	539,648.110	540,276.719	-
Lease Acquisition Time (μs)	5,922,291.588	5,970,578.997	5,969,034.806	-
% of Failed Transactions	0.042	0.033	0.031	-

Table 4.5: Backwards indexing, hybrid trie, for blacklist size 1,500,000.

Here the performance with a reverse indexing trie height of one, while low, is still impressive.

Again, as with a blacklist of size 900,000, we obtain performance matching the baseline metrics with a trie height of two or more.

There is no data for a trie height of four simply because such a trie height generates 1,499,738 records which represents 99.98 percent of the blacklist's records. With this many records in memory we have squandered the advantage of a hybrid trie, by essentially, transforming it into a regular trie, which would violate the goal stated in Section 3.2.1.

4.4.4 Phase 3: Stress Test

Unlike the previous set of tests, here we have a significantly larger blacklist size of 9,000,000. This is a stress test and is meant to check the system for its ability to scale.

Forward Indexing

Forward indexing of the stress test sized blacklist generated the results found in Table 4.6.

	Trie Height (h)			
Metric	1	2	3	4
Transactions Rate (per s)	-	-	28.098	32.651
Transaction Time (μs)	-	-	332,845.608	339,511.928
Lease Request Time (μs)	-	-	545,072.807	548,471.909
Lease Acquisition Time (μs)	-	-	5,883,402.460	6,056,957.512
% of Failed Transactions	-	-	0.036	0.060

Table 4.6: Forward indexing, hybrid trie, for blacklist size 9,000,000.

As with the previous forward indexing test, found in Section 4.4.3, there is no data for trie heights of one or two simply because of unacceptable performance.

Unlike the previous forward indexing tests, neither trie heights of three or four are able to duplicate the performance found in the baseline tests. Though, in the case of a trie height of four, the performance is competitive.

Backwards Indexing

Backwards indexing of the stress test sized blacklist generated the results found in Table 4.7.

	Trie Height (h)			
Metric	1	2	3	4
Transactions Rate (per s)	-	32.659	28.096	-
Transaction Time (μs)	-	340,280.849	335,621.349	-
Lease Request Time (μs)	-	545,960.626	549,796.964	-
Lease Acquisition Time (μs)	-	5,968,388.707	6,022,713.630	-
% of Failed Transactions	-	0.028	0.044	-

Table 4.7: Backwards indexing, hybrid trie, for blacklist size 9,000,000.

There is no data for the trie height of one because of poor performance. There is no data for trie height of four for reasons similar to those of the previous backwards indexing test in Section 4.4.3.

Here it is important to explain why performance decreases as trie height increases. In this case, once the trie height reaches three we obtain a trie containing nearly 7,000,000 leaf nodes. This is approximately 77 percent of the total number of records. At this point, the memory expense of the hybrid trie begins to parallel that of a regular trie. The time required to page such memory begins to exert a prohibitive cost on search times.

As with forward indexing, we do not obtain values identical to the baseline test.

4.5 Conclusions

The first conclusion we may extract from the data is that in a deployment of this prototype, the selection of a trie of height two with a reverse index is likely to yield the best performance levels. Installations that do not wish to tweak performance should generally rely on this configuration.

Section 3.2 explains that only the first DHCP request by a client causes a blacklist lookup. In the tests we have performed, the server has a fresh blacklist and therefore every client's first request causes a blacklist lookup. This situation is not likely to occur in a production environment unless the network experiences a high client turnover. Networks with a fairly stable client population will cause few blacklist lookups since the clients will long since have been whitelisted.

It was mentioned, in the description of the testing procedure, that the test was designed to place a significant, continuous heavy load on the DHCP server. It is unlikely that a network's DHCP server would be under a similar load barring some mass network oddity given the nature of the protocol. Sites wishing to minimize memory usage may be able to do so by lowering the trie height or using a forward indexing trie. These will likely result in lower performance but this may be acceptable if the server is also configured to offer IP address leases with longer lease times. This would cause less frequent renewals of leases and thus decrease DHCP traffic.

The inability of the DHCP server to maintain ideal transaction rates during the stress test, while unfortunate, is not totally unexpected. The blacklist size was originally chosen to be exceedingly difficult. It is unlikely that records would accumulate

for fifteen years. The statute of limitations on theft crimes varies by jurisdiction but generally ranges between 5 to 6 years, though some jurisdictions have no such limitations [8, 47, 4]. The obsolescence time for computer equipment is often ridiculously short [29]. Both of these factors combine to make it unlikely that records would be retained for such a long timeframe.

Chapter 5

Countermeasures

In this chapter, we consider the countermeasures available to anyone trying to defeat a blacklisting DHCP server, as described in earlier chapters. Given the open nature of the DHCP standard and the number of open source implementation based servers, we must assume that wide scale deployment of these ideas could not be kept secret for any length of time. As is often the case, "security through obscurity" is not an option.

Section 5.1 covers the countermeasures available at the hardware level. In Section 5.2, we will address the countermeasures available at the level of the operating system and drivers. Finally, in Section 5.3, we will discuss networking solutions that mask the MAC addresses.

5.1 The Hardware

5.1.1 NIC Replacement

The simplest option in dealing with a blacklisted MAC address is to discard it by replacing the Network Interface Card (NIC). The question of cost, difficulty and practicality however must all enter into the equation.

Desktop PCs

The desktop PC is the easiest computer in which to perform NIC replacement. Up until fairly recently, the NIC was found inserted into the computer using whatever expansion slot system the motherboard supported. The current trend in motherboard construction is to provide basic functionality such as NICs permanently integrated into the motherboard's circuitry. In both cases the solution to a blacklisted NIC is straightforward: a new NIC must be obtained and placed into an available expansion slot. In the case of motherboards with a non-integrated blacklisted NIC then the old NIC should be removed and the new one inserted. This case guarantees the availability of a slot for the new hardware. The case of motherboards with an integrated NIC may pose a problem, as there is no guarantee of an open expansion slot. Only in cases where no slot is available would this procedure be impractical.

Currently, the price point for a new Peripheral Component Interconnect (PCI)[1] NIC, wireless or wired, from a reputable manufacturer is reaching the sub 20 CAD range. Even Gigabit Ethernet[2] hardware is reaching reasonable price levels. This

[1]Currently the most common expansion slot type is PCI although PCI express is slowly replacing it.

[2]Gigabit Ethernet is the likely successor to the overwhelmingly popular Ethernet type network

makes the question of price almost certainly moot.

The final question is one of difficulty. For those unfamiliar with the ins and outs of replacing computer hardware the task may be daunting but for anyone with a modicum of practical computing experience it should be straightforward and require little time.

Integrated PCs

The term integrated PC is used to mean laptops, notebooks, sub-notebooks, tablet PCs and even Personal Digital Assistants (PDAs). The common denominator in all of these is that in most cases the entire unit is integrated into a single small package. The drawback of such small sizes is the integration. In order to accommodate the small form factor, most components are placed directly onto the motherboard and few of these PCs include on-motherboard expansion slots. The only solution in these cases is to rely on Universal Serial Bus (USB), Personal Computer Memory Card International Association (PCMCIA) or flash-card devices to supply an alternate format NIC as any blacklisted NIC is likely irremovable due to integration. In some of the smaller devices, none of these options are available either for lack of the proper expansion slots or lack of appropriate expansion hardware. Even if the equipment is available, it is likely that the expansion slot is already in use. The practicality of the replacement approach with integrated PCs is much lower than in desktops.

The current price point for USB or PCMCIA cards is roughly equivalent to that of the PCI format. As was already mentioned, this cost is no barrier to use. Flash-card based alternatives are, however, significantly more expensive reaching easily into the 100 CAN range.

and as such represents the next generation in wired networks.

Most would agree that the difficulty in using USB, PCMCIA or flash based cards is fairly low. All were designed for ease of use by the end user.

5.1.2 NIC Reprogramming

The majority of NIC manufacturers currently use a type of chip known as Electronically Erasable Programmable Read-Only Memory (EEPROM)[3] chip to store the MAC address of a NIC [24]. As the name of this type of chip indicates, it is possible to erase the data stored on the chip and replace it with new data. The implication is that it is possible to permanently alter a NIC's blacklisted MAC address with a new unlisted MAC address.

There are, in fact, two different techniques for writing new data onto a NIC's EEPROM. The first is described in "MAC Address Cloning", which although dated, is serviceable. It involves the physical removal of the EEPROM chip from the NIC and reprogramming it using an EEPROM programmer, which is, at best, a tricky operation as in many cases it requires that the chip be desoldered from the circuit board [24]. In addition, reprogramming the chip may prove difficult, as the data format may need to be reverse engineered [24]. However, some manufacturers are kind enough to offer this information on their websites [18]. This places this technique outside the ability of everyone but the electronics hardware expert.

The second technique is to rely on the NIC itself to provide the facilities for reprogramming the EEPROM. MAC addresses are not the only data stored on a NIC's EEPROM chip and thus manufacturers occasionally supply both the circuitry on the NIC and software required to write the EEPROM chip's memory [19]. The danger here is that an unsuccessful attempt to write a new MAC address may render

[3]Historically, ROM chips could only be written to once, hence the term Read-Only Memory.

the NIC useless. This may not be disastrous on a desktop PC but it could render an integrated PC all but incapable of network access. Unlike the previous technique, here, anyone familiar with software can accomplish the task assuming a third party provides the software. Building the software from scratch is a non-trivial task.

This solution, unlike hardware replacement, requires a higher level of technical savvy and bears a sharp penalty for failure.

5.2 The Software

In this section we will cover the variety of software solutions open to someone wishing to mask a blacklisted MAC address. As these solutions are operating system dependant, we will cover the range of most frequently used networking operating systems. The Google Zeitgeist[4] provides us with a snapshot of operating systems found on the web [15]. Though this may not be completely representative of computers connected to the internet — servers for instance rarely make connections to Google — it does serve as a reasonably accurate barometer of end user systems. The zeitgeist indicates that the Linux operating system has a 1% market share, the Mac a 3% market share and the various Windows operating systems dominate with 91% market share. We will therefore endeavour to cover these operating systems and their mechanisms for altering MAC addresses, if any.

[4]The Google Zeitgeist tracked statistics concerning Google users, including operating system type.

5.2.1 Linux Operating System

The Linux operating system integrates support for changing a NIC's MAC address. This offers a solution to the blacklisted MAC address problem. The method used to accomplish this is described in the man page for the `ifconfig` command, which reads as follows: "Set the hardware [MAC] address of this [network] interface [card], if the device driver supports this operation" [12]. With confirmation that the operating system itself supports this option, it only remains to verify if the NIC's device driver supports the functionality for the NIC in question.

The weaknesses of this solution are twofold. The first is that, unlike the hardware solutions we covered in Section 5.1, it is not permanent. Each time the power is cycled on the computer, the `ifconfig` command must be re-run otherwise the true MAC address will be used. This is not a fatal problem, a bit of clever scripting and the change can be made each time the computer boots up. The second weakness is the very nature of the change. The bit of clever scripting may, for a great variety of reasons, be wiped out. An operating system reinstallation or upgrade could easily remove the change and return the computer to its blacklisted MAC address.

Linux is widely regarded as user-unfriendly and often demands a fairly in-depth set of technical skills and know-how to operate on a daily basis. It is not unreasonable to claim that enabling this functionality on a permanent basis should be well within the grasp of most Linux users given the general day-to-day demands of using such an operating system.

5.2.2 Apple OS X

FreeBSD is "the primary reference platform for [OS X's] kernel development" [3]. This places FreeBSD at the very core of OS X and makes it unsurprising that OS X supports the modification of a NIC's MAC address, as this functionality is also found in FreeBSD. The `ifconfig` command, the same as the Linux command, provides this functionality for both FreeBSD and OS X, but with slightly different syntax [6].

Support for MAC address spoofing from the operating system is only half the equation; the device driver for the NIC also needs to support the functionality. The drivers for the integrated Ethernet NICs found on recent Apple computers, in most cases, support the spoofing of its MAC address. In contrast, drivers for the Apple Airport Extreme wireless card do not support the spoofing of its MAC address. Whatever the reason for its omission, the lack of this functionality has been remedied by third party developers. The first fix involves patching the network driver for the Airport Extreme card to enable the functionality with `ifconfig` [44]. The second fix involves compiling a new kernel for OS X [37].

When the network driver does allow for spoofing the MAC address, experimentation has determined that OS X 10.3 and the new OS X 10.4 have slightly different behaviours. In the case of 10.3, a spoofed MAC address only affects the link layer of the packets but DHCP packets contain the unspoofed value in the "chaddr" field.[5] With 10.4 however, a spoofed MAC address is used in both the link layer and in the DHCP "chaddr" field. Given the updated behaviour in 10.4, we can assume that the behaviour found in 10.3 was a bug and not a feature.

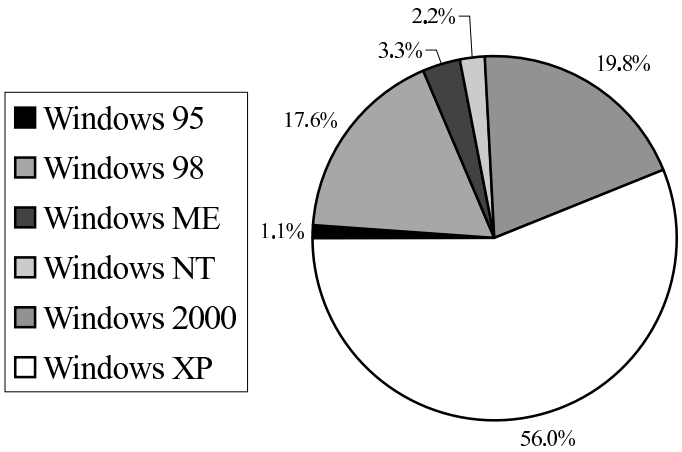
As the mechanism for spoofing is identical to the one used in Linux, it shares the same weakness: fragility. Beyond this, the fact is that because most Apple users

[5]For a discussion of the link layer and the "chaddr" field see Section A.4

purchase their computer in order to avoid the pains of Windows, never mind those of the command line, it seems less likely that the average Apple user would be able to use such a solution. Perhaps in the future this feature will be integrated into the Graphical User Interface (GUI).

5.2.3 Windows

The earlier mentioned 91% market share is the total sum figure for all of the many versions of Windows currently in use on the internet. In Figure 5.1, we can see that there is a wide spectrum of Windows operating systems still in use on the web.



Clockwise, from Windows 95 to Windows XP

Figure 5.1: Windows operating systems found on the internet [15].

A Short History of Windows

The history of the Windows operating system is fairly convoluted. There are dozens of branches, splits, false starts and dead-end development paths [10]. A very crude simplification of this history is to claim that there are two strains of Windows oper-

ating systems. The first strain originated in the Disk Operating System (DOS) and includes Windows 1.0, 2.0, 3.x, 95, 98, ME. This is Microsoft's legacy strain and it is more or less a dead development branch. The second and currently active strain originated with the development of Windows NT by David Cutler and his team [14]. This second strain was an attempt by Microsoft to create an operating system that was legacy free, genuinely robust, networked and multi-user [14]. This strain includes the Windows NT 3, NT 4, 2000, XP, the forthcoming Vista, the projected Blackthorn and future Microsoft desktop and server operating systems.

As this document only covers the major operating systems still seen in the wild on the internet, we do not cover the legacy operating system families such as the Windows x.x series or the Windows NT 3.x series. Both of these families are many years out of date, passed their "end-of-life" dates and almost impossible to obtain for research [30].

Windows 95, Windows 98 First Edition, Second Editions and Windows ME

All four of these operating systems offer the functionality to spoof a NIC's MAC address, although this functionality is not available through the Windows GUI. It can only be found buried in the Windows registry.

The Windows registry is used to store the vast majority of configuration information in the Windows operating system. Its structure vaguely resembles that of a DOS file structure. What would be a directory is known as a key. A key may have subkeys much like a directory may have subdirectories. A key may also contain a set of keyword-value pairs much like a directory may have files. However, despite it's resemblance to a directory structure, the registry is not available from the file browser

or the command line. A separate program must be used to access and modify the registry. This is because the registry is not actually located in a single file but spread across several files, depending on which version of Windows operating system [54].

The procedure found here for spoofing the MAC address is a summary of the procedure provided by Kyle Lai [26]. In Windows 95 et al., each NIC installed has it own subkey under the following registry key:

```
HKLM\System\CurrentControlSet\Services\Class\Net
```

To spoof the MAC address of a particular NIC, a new keyword-value pair must be created in the NIC's subkey. This new keyword-value pair should be named "NetworkAddress", its type should be of "String Value" and its value should be whatever the spoofed MAC address value is desired, as expressed in hexadecimal notation, for example "000090A9C5FF". As with many Windows configuration updates, a reboot is necessary to complete the procedure.

None of the operating systems in this series were built with security in mind, thus there is no way of restricting user access to the registry. This makes the addition or deletion of registry values simply a question of access. Anyone with physical access to a machine running any of these operating systems can spoof the MAC address of any NIC attached to that operating system.

Windows NT 4

Windows NT 4 also offers the ability to spoof a NIC's MAC address and it too is unavailable from the Windows GUI.

The procedure found here is, again, a summary of the procedure provided by Kyle

Lai [27]. It is virtually identical to that used in with Windows 95 et al. although with a different registry key:

```
HKLM\SYSTEM\CurrentControlSet\Services\[Adaptor]\Parameters
```

Unlike the previous case there is no subkey for each NIC. In this case each NIC is assigned an identifier, which must be substituted in place of "`[Adaptor]`" in the registry key path. The value of this identifier can be obtained by using the `net config rdr` command.

In contrast to Windows 95 et al., Windows NT 4 was designed with multi-user facilities and security in mind. With these features, it is possible that someone who can boot up the computer would be unable to modify the registry and thus unable to spoof the MAC address of a computer's NICs. What one needs, beyond being able to boot the operating system, is access to an Administrator account if one wishes to spoof a MAC address, because only Administrator accounts can edit the registry.

Windows 2000, Windows XP

As with their predecessor Windows NT, Windows 2000 and XP also offer the ability to spoof a NIC's MAC address.

The procedure found here is, once more, a summary of the procedure provided by Kyle Lai [25]. It is similar to the previous procedures but with two slight differences. The first difference is the registry key:

```
HKLM\SYSTEM\CurrentControlSet\Control\Class\4D36E972-E325-11CE-BFC1-08002BE10318
```

Like Windows 95 et al., each NIC has its own four-digit subkey in which a new

keyword-value pair must be added to spoof the MAC address. The new keyword-value pair should, as previously, be named "NetworkAddress" and its value should be the desired spoofed value. The second difference is that the type of the keyword-value pair should be "REG_SZ".

Unlike previous versions of Windows, there is, in some cases, a GUI for spoofing a NIC's MAC address. The functionality is included in some driver software at the manufacturer's discretion [25]. This GUI functionality is simply an automation of the above-described procedure. When it is used, it alters the registry in identical fashion.

As upgrades to Windows NT 4, both 2000 and XP maintain the security policies vis-à-vis the registry. This means that even with boot access to the operating system, anyone wishing to spoof a MAC address would still need access to an Administrator account. This applies to both techniques mentioned above.

Windows Upgrades and Spoofing

Microsoft's various Windows operating systems often offer to upgrade from one version to another. This begs the question, how stable is the mechanism for spoofing MAC addresses through such upgrades? Table 5.1 contains the results of testing the various mechanisms for MAC address spoofing against permitted Windows upgrade path.

Table 5.1 illustrates that when upgrading within a strain the spoofing generally survives an upgrade without issue. If the upgrade crosses from one strain to another, however, the spoofing is lost. The exception being going from Windows NT 4 to either Windows 2000 or Windows XP. This is not surprising given that the mechanism for

To Windows / From Windows	98	98SE	ME	NT 4	2000	XP
95	Preserved	Preserved	Preserved	N\A	Lost	N\A
98	N\A	Preserved	Preserved	N\A	Lost	Lost
98SE	N\A	N\A	Preserved	N\A	Lost	Lost
NT 4	N\A	N\A	N\A	N\A	Lost	Lost
2000	N\A	N\A	N\A	N\A	N\A	Preserved

Table 5.1: Effect of upgrading Windows on MAC address spoofing.

spoofing varies in NT 4 as compared to 2000 and its inheritor XP, despite their sharing the same strain. Why did this happen and is it likely to reoccur with Vista or Blackthorn? The probable reason for the change between NT 4 and 2000 was that Windows 2000 was the first step in the integration of both operating system strains into a single codebase and, as such, while still being the inheritor of NT 4, it carries many modifications that result from the integration of the home operating system codebase [36]. It seems unlikely that Vista or Blackthorn will suffer from such changes now that the integration of the home operating system strain is complete.

Registry Stability and Security

The registry is in constant flux in a Windows environment with keys being added, read, modified and deleted at all times. Given that the mechanism for spoofing a MAC address is registry dependent we need to examine how stable the change is.

For all Windows operating systems the location in the registry that enables spoofing of MAC addresses ties together the model of the NIC card and the PCI slot location into a unique registry path. This means that moving the card from one PCI slot to another will result in the card transmitting its un-spoofed MAC address unless

care is taken to re-spoof the value for the newly created registry unique path based on the card type and its new PCI slot.

Certain troubleshooting procedures involve deleting a NIC's key from the registry. This would also delete the keyword-value pair "NetworkAddress" containing any spoofed value. When Windows automatically rebuilds the NIC's key the keyword-value pair would not be regenerated, thus destroying the spoofing. Some early driver installation programs used similar behaviour but this is mostly a thing of the past. Most drivers now follow the proper installation procedure.

5.3 NAT Technology

Network Address Translation (NAT) technology was developed as a stopgap solution to the ever-increasing demand for IP addresses [33]. The rather clever solution, laid out in the defining RFC, was to place a device that would provide a "stub" network, a small network of non-globally unique IP addresses valid only within this small "stub". If any network traffic had to leave the "stub", then it would be translated into a globally unique IP address [33].

Currently most personal ISPs only offer a single IP address or, in some cases, demand a surcharge for each additional IP address. This has led to an increase in the use of routers at home. Given this restriction, most of these routers incorporate NAT technology, as it is a cheap and easy solution for several home computers to share a single home internet connection.

The "stub" network must have a mechanism for assigning IP addresses to each of its members. This often comes in the form of an internal DHCP server. The router itself has its own MAC address used on the external connection, which is what

the outside world sees. Thus any computer with a blacklisted MAC address that is connected to a NAT device will not be reported to the wider network. The wider network will only see the NAT device's MAC address.

This makes these pieces of home networking equipment a simple defence against MAC monitoring. Unless the computer is connected to the internet directly, the blacklisted MAC address will never leave the "stub" network.

These home NAT devices are unlikely to be able to support the solution proposed in this document. There is however a solution to this limitation. The device could be updated to forward any new MAC address it receives to the ISP's DHCP server thus opening it up for examination. The problems here are likely to be of a legal, ethical and public relations nature. Consumers are presumably not going to be happy that a piece of equipment they have purchased is "ratting them out" to their ISPs. The media incident over the Pentium III and it's unique identifier is illustration enough that consumer's do not enjoy corporations intruding into their privacy.[6]

[6]For those that don't recall the Pentium III chip introduced a "processor serial number". Fears at the time were quite high as is illustrated by the FAQ Intel released [20]. Complaints against Intel were filed in the USA [46].

Chapter 6

Counter-Countermeasures

The examination in Chapter 5 of countermeasures to the proposed blacklisting system leaves us with the conclusion that, with moderate difficulty and technical knowledge, it may be possible for someone to hide their true MAC address. How then do we counter this situation? In this chapter we discuss some approaches that can be used.

6.1 DHCP Client Identifier

The introduction of the DHCP protocol provided the designers with the opportunity to correct flaws and fix omissions that had been exposed during its previous incarnation as the BOOTP. One of the changes was renaming BOOTP "vendor extensions" to DHCP options [35]. During this transformation, several new types of DHCP options were created, including the "client identifier" option [35]. It serves as a unique identifier instead of the "chaddr" field, which drops its dual role and returns to simply being the client's MAC address [35, 34].

The format of the "client identifier" option, as defined in RFC 2132, is illustrated in Figure 6.1.

Code	Length	Type	Client Identifier			
61	n	t	i_1	i_2	...	i_{n-1}

Figure 6.1: DHCP option 61, the client identifier [34].

The comments on this option in RFC 2132 make it clear that it may carry a MAC address or it may include some arbitrary data. If the MAC address is transmitted, the type byte is set to the appropriate hardware type and if arbitrary data is transmitted, the type byte must be set to 0 [21, 34]. The only other restriction the "client identifier" is that, like the "chaddr", the value must be unique on the subnetwork [35].

The "client identifier" option enables another opportunity to track a stolen computer. Though a clever thief may change the MAC address of the computer, it is possible that the thief would overlook the "client identifier". To this end, if the "client identifier" were explicitly set to a unique identifier it too could be tracked in the blacklist. In order to avoid duplicate entries it is suggested that the value used be the MAC address.

6.1.1 Windows and the Client Identifier

Within the set of Windows operating systems we are discussing, all include a "client identifier" in DHCP requests.[1] This "client identifier" is simply a copy of the NIC's MAC address value. Unfortunately for us, by default if the NIC's MAC address is spoofed, this value is used for the "client identifier" as well.

The ability to set custom values for the "client identifier" was only introduced in

[1]See Figure 5.1 for list of Windows operating systems under discussion.

the business strain of Windows operating systems starting with Windows NT 4 and continuing on with Windows 2000 and Windows XP. The procedure, for Windows NT and Windows 2000—XP, is found in Microsoft article KB172408, and is summarized in the following sections [31].

Windows NT 4

The procedure in article KB172408 is much like that for spoofing a NIC's MAC address. It simply relies on a slightly different registry key:

```
HKLM\SYSTEM\CurrentControlSet\Services\[Adaptor]\Parameters\Tcpip
```

As in the procedure found in Section 5.2.3, the NIC's value for "`[Adaptor]`" must be substituted. Once the key is found, a new keyword-value pair must be created with the name "DhcpClientIdentifier" with the desired "client identifier" as the value.

The keyword-value is specified in the article to be of type "REG_DWORD" which would limit the range of "client identifier" from 00:00:00:00 to FF:FF:FF:FF. As MAC addresses are six hexadecimal pairs, this limitation to four pairs makes this feature less than useful.

However, our experimentation has shown that the keyword-value pair type "REG_BINARY" can be used without issue, despite lack of official Microsoft support. The advantage of this key type is that it supports arbitrarily large values, for example six hexadecimal pairs.

Windows 2000, Windows XP

Mirroring the change in key location for spoofing MAC addresses in the upgrade from Windows NT 4 to Windows 2000, article KB172408 also specifies a new key location for the "client identifier":

```
HKLM\System\CurrentControlSet\Services\TcpIp\Parameters\Interfaces\NICGUID
```

The rest of the procedure laid is identical to that used with the Windows NT "client identifier" spoofing found above.

Again, our experiments have also shown that, as with Windows NT, using the type "REG_BINARY" provides for arbitrarily large values for the spoofed client identifier despite lack of mention of this in the official Microsoft document.

The only further note to add is that article KB172408 does not include Windows XP on its "Applies To" list. The review date for the article is June 3, 2003 while Windows XP was released on September 25, 2001 [10]. It seems likely that the omission of Windows XP on the list of supported operating systems is simply an oversight.

Windows Upgrades and Client Identifier

Unlike what can be observed in Table 5.1, the situation vis-à-vis upgrades and a custom "client identifier" is fairly simple: when upgrading from Windows NT to either Windows 2000 or Windows XP a custom "client identifier" is lost. This is simple to explain given the distinct registry key NT uses as compared to 2000 and XP. An upgrade from 2000 to XP, however, does preserve a custom "client identifier". Predicting if future releases of Windows will continue to support this feature is difficult

to gauge because, as mentioned, Windows XP does not officially support the feature.

Windows Custom Client Identifiers

Following the requirements of the RFC defining DHCP, the Windows DHCP client uses a type byte[2] of zero if a custom client identifier is used. This means that a blacklisting DHCP server will need to be clever in handing "client identifiers" values. Some may be MAC addresses with a type byte of zero and some may be arbitrary values.

6.1.2 Apple OS X and the Client Identifier

The Google Zeitgeist tells us that 91% of operating systems found on the internet are Window variants of some sort. This level of homogeneity has engendered a philosophy that favours catering only to Windows operating system. Some ISPs only support Windows machines officially and have come to rely on the default Windows behaviour of including the "client identifier" even if the RFC states that the inclusion is optional.

Thanks to the policies of these ISPs, it's often the case that Apple's operating systems need to duplicate Windows behaviour. To this end we find that from at least OS 8.5 onwards, it is possible to supply a custom "client identifier" [2]. Consistent with Apple's long commitment to ease of use and well-designed user interface, there is a simple user interface element for setting the "client identifier" on Apple's operating systems.

[2]See Section 6.1 for a discussion of the type byte.

6.1.3 Linux Operating System

The vast majority of Linux distributions currently use the ISC implementation of the DHCP client. The ISC reference DHCP client has, from its introduction in version 2.0, always included the ability to specify any arbitrary value for the "client identifier", including arbitrary values for the type byte, a fact obtained by studying the source code of the initial release.

An alternative DHCP client named `pump` is used in some Linux distributions and is offered as an alternative to the usual ISC implementation. Currently, the `pump` implementation does not include the "client identifier" by default, however it does offer the ability to duplicate Windows DHCP client behaviour by including a "client identifier" that matches the MAC address but there is no way to include a custom "client identifier" [38].

A less popular choice but still common enough is `dhcpcd`; which as the name implies is a DHCP client daemon. The daemon command line options include the ability to specify a custom "client identifier" [55].

Chapter 7

Conclusion

In this document we have proposed the "rehabilitation" of the theft recovery solution. To this end a prototype system has been developed and successfully tested thus fulfilling our stated goal.

Unfortunately the success remains only valid with version four of the Internet Protocol (IP), which is currently in use over the majority of the internet. However, the upcoming switch to IP version six will include an upgrade to the MAC address space [52]. This will expand the size of a MAC address from twelve hexadecimal digits to sixteen hexadecimal digits [52]. This expanded address space unfortunately places an extra strain on the system as designed. Further research would be required to ensure continuing viability.

Unaddressed remain the logistical requirements of this proposal. First among these would be the maintenance and distribution of a MAC address blacklist. Second would be ISP participation: mandatory or voluntary? How this could be handled at a local, regional and national level? These questions are outside the scope of this

document as they are entirely dependant on policy decisions and not technical merits. However, they remain an issue for any real world adoption of the idea.

The majority of countermeasures we have discussed have a varying level of permanence attached to them thus making them imperfect. Still, they have eroded the historically static nature of the MAC address granting it a certain malleability. This malleability unfortunately places into jeopardy the "rehabilitation" we have attempted. There is however little that can be done about this troubling problem. Beyond this there is the issue of the many machines hiding behind NAT technology though as IP version six is implemented it may be that the numbers of such machines will decrease. NAT has an inherent number of limitations and is often employed in order to avoid paying for an additional IP. With IP version six providing for 3.4×10^{38} addresses it may be that the days of NAT are numbered [51]. Regardless, we have succeeded in removing the theft recovery technology from the client but dependence on standard hardware and software has its price.

In short, while we have proven the techniques and ideas proposed as plausible, there is some doubt whether they would yield satisfying results if deployed on the internet.

Appendix A

DHCP Primer

This section is a simplification of what is found in RFC 2131[1] which is the current reference describing the basics of DHCP.[2]

A.1 What is DHCP?

DHCP is a protocol that a client can use to discover the appropriate network configuration without user intervention and more importantly without manual preconfiguration. At its core, it serves as a method for a client to obtain the minimum information necessary in order to create a fully functional network connection; usually this means an IP address. There are some exceptions where this is not the case, however, they are beyond the scope of this discussion.

DHCP is an extension of BOOTP and was designed to be backwards compatible in

[1] A good source for RFCs is `http://www.rfc-editor.org/`.

[2] As of July 2005.

addition to supporting a host of new features. Therefore, as was BOOTP, DHCP is a client-server protocol, which uses the connectionless User Datagram Protocol (UDP) transport protocol. DHCP has a single message format, which is shown in Section 2.2.2, but also reprinted here, in Figure A.1, for ease of reference.

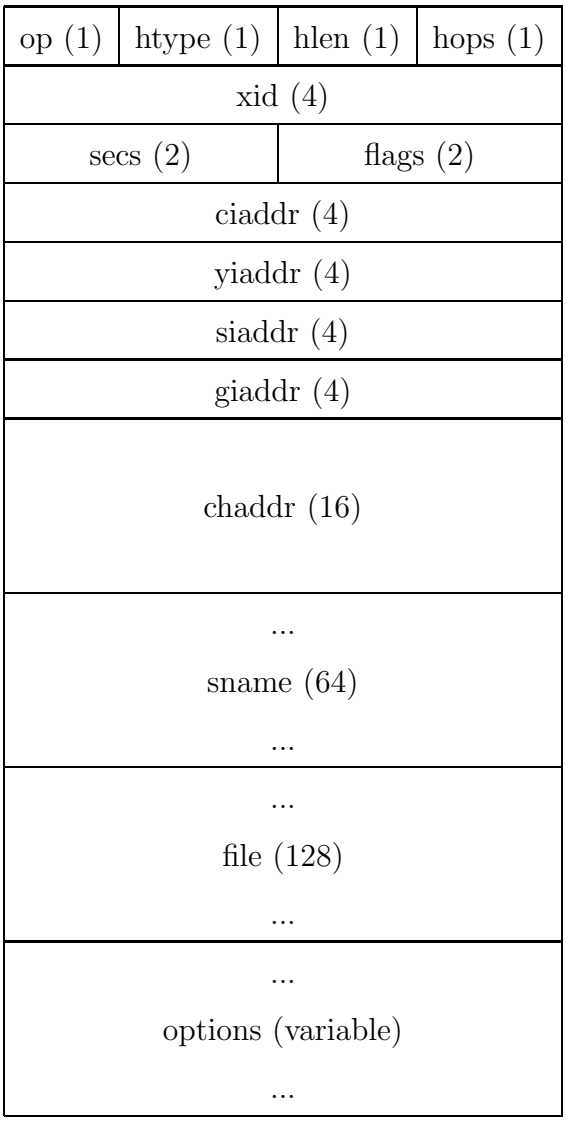


Figure A.1: Format of a DHCP message [35].

There are three fields in Figure A.1 that are of keen interest to us in this document. Anyone wishing further information about other fields should consult the RFC. The

first field of interest is the "op" field, which denotes the type of BOOTP message we are interpreting. Either a BOOTREQUEST or a BOOTREPLY. This is a holdover from BOOTP and is kept for backwards compatibility reasons.

The second field of interest is the "chaddr" field, which contains the DHCP client's MAC address. MAC addresses are discussed in Section 2.2.1. A discussion on the necessity of this field can be found in Section A.4.

The third field of interest is the "options" field. There is no specified size for this field in Figure A.1. The reason for this lack of specificity is that the "options" field is actually composed of several option fields each containing separate information. These DHCP options are all identified by an option number between 0 and 255. The option number is also the first byte of the option when it is encoded in the DHCP message. Some option fields are, despite the name, required while others are not.

In this document, there are two "option" fields mentioned so far. The first is option 61, which is illustrated in Figure 6.1 and discussed at length in Section 6.1. The second example is option 82, an option, added by some ISPs in order to better identify their clients. This option is mentioned briefly in Section 2.3.

A.2 DHCP Message Types

DHCP was designed to be backwards compatible with BOOTP and thus, instead of adding new message types in the "op" field to separate BOOTP message from DHCP message, the designers chose to use an option field instead. Option 53, the DHCP Message Type option, was born.

As illustrated in Figure A.2, there are nine possible values for Option 53, though

Code	Length	Type
53	1	1-9

Figure A.2: DHCP option 53, DHCP message type [34].

only a few are germane to our discussion. The subset we will be discussing is:

- DHCPDISCOVER: A message broadcast by a client attempting to "locate available [DHCP] servers" [35].
- DHCPREQUEST: A message sent by a client attempting to obtain an IP address lease from a particular DHCP server.
- DHCPOFFER: A message from the server destined to the client offering a lease on an IP address. It is sent in response to a DHCPDISCOVER message.
- DHCPACK: A message from the server destined to the client indicating that all is well with its request. Sent in response to a DHCPREQUEST.

These message types make up the basics of the DHCP protocol and will be referenced in most of the examples that follow.

A.3 Obtaining a DHCP Lease

The process of obtaining a lease on an IP address is illustrated in Figure A.3. We have chosen to illustrate the simplest case, which is when a client has no IP address and is in the process of requesting a lease. As we can see, this process has two steps, where each step includes a request by the client and a response by the server.

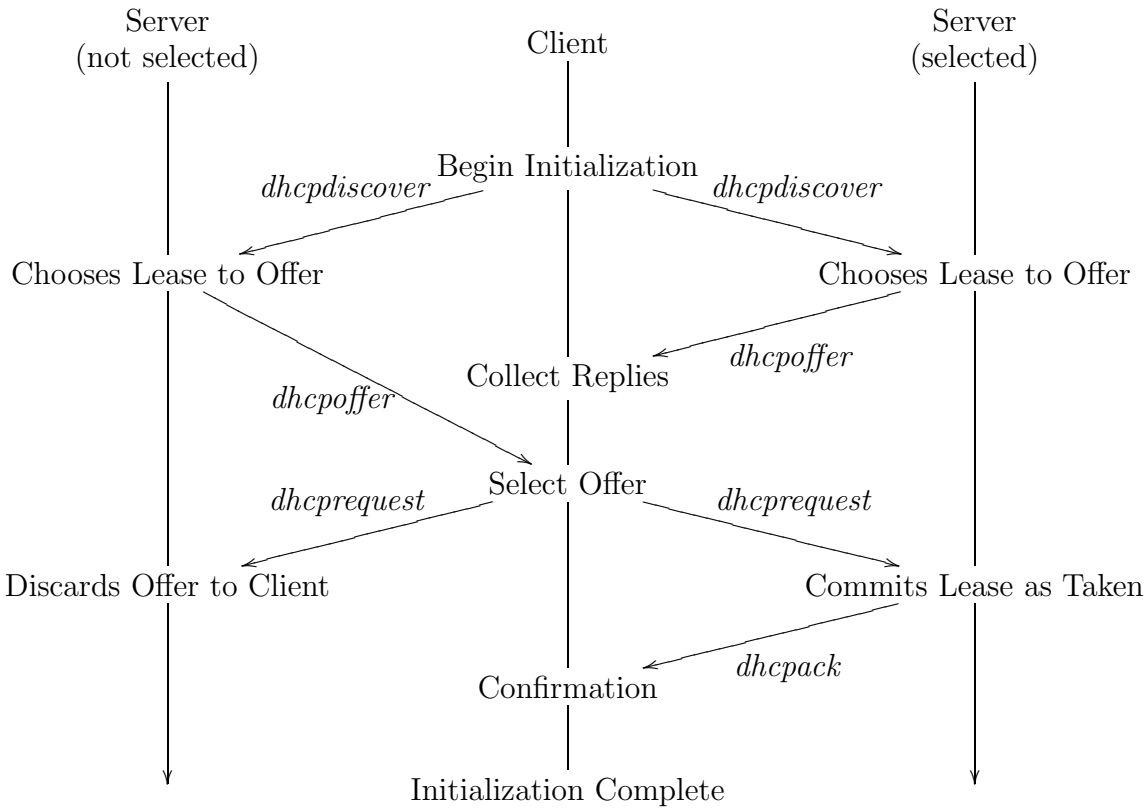


Figure A.3: Obtaining A DHCP lease [35].

Not shown is that all messages send in Figure A.3 are broadcast messages. Broadcasting a message means that the messages are sent to no particular IP address and everyone on the network can receive it.[3] The client must do this as it has no idea what the IP address of the DHCP server is. The server must do this because the client does not yet have an IP address.

[3]Here here is a dual meaning to the term broadcast. It means the message is broadcast with destination 255.255.255.255, a broadcast IP address. It also means that the message is broadcast with destination FF:FF:FF:FF:FF:FF, an Ethernet link layer broadcast address. This is a simplification of the true behaviour, which is explained in RFC 2131.

A.4 On the Necessity of the "chaddr" Field

Some question the necessity of using the "chaddr" field in the DHCP message to match against the blacklist as suggested in Section 2.2.2. They argue that the MAC address, the hardware address, would be available in the link layer, the hardware layer. This objection requires a detour into the workings of the TCP/IP protocol suite.

There are four layers to the TCP/IP network stack. The topmost is the application layer, then the transport layer, then comes the network layer and finally the link layer [50]. In our cases the layers are: DHCP, UDP, IP and for the sake of this discussion we will use Ethernet as our link layer. As a DHCP message descends down the layers of the TCP/IP protocol suite, the message is wrapped in various layers; thus the DHCP message is wrapped in a UDP message, which is in turn wrapped in an IP message, which is finally wrapped in an Ethernet frame [50].

An Ethernet frame contains both a destination and a source address link layer addresses, which are MAC addresses [50]. This means that a DHCP server, upon reception of an Ethernet frame, would have the MAC address of the sender available as the link layer source address.

There is, however, a situation where the link layer source address is not that of the DHCP client. This is where a DHCP relay is in use. In Section A.3 we mention that all the messages exchanged between he client and the server were broadcast messages. This means that the server and client must share an environment where they can hear each other's broadcast messages. If it is not the case, then a DHCP relay must be inserted between the client and the server in order to solve this problem.

In Figure A.4, the client is on subnet A, a subsection of the network, and the

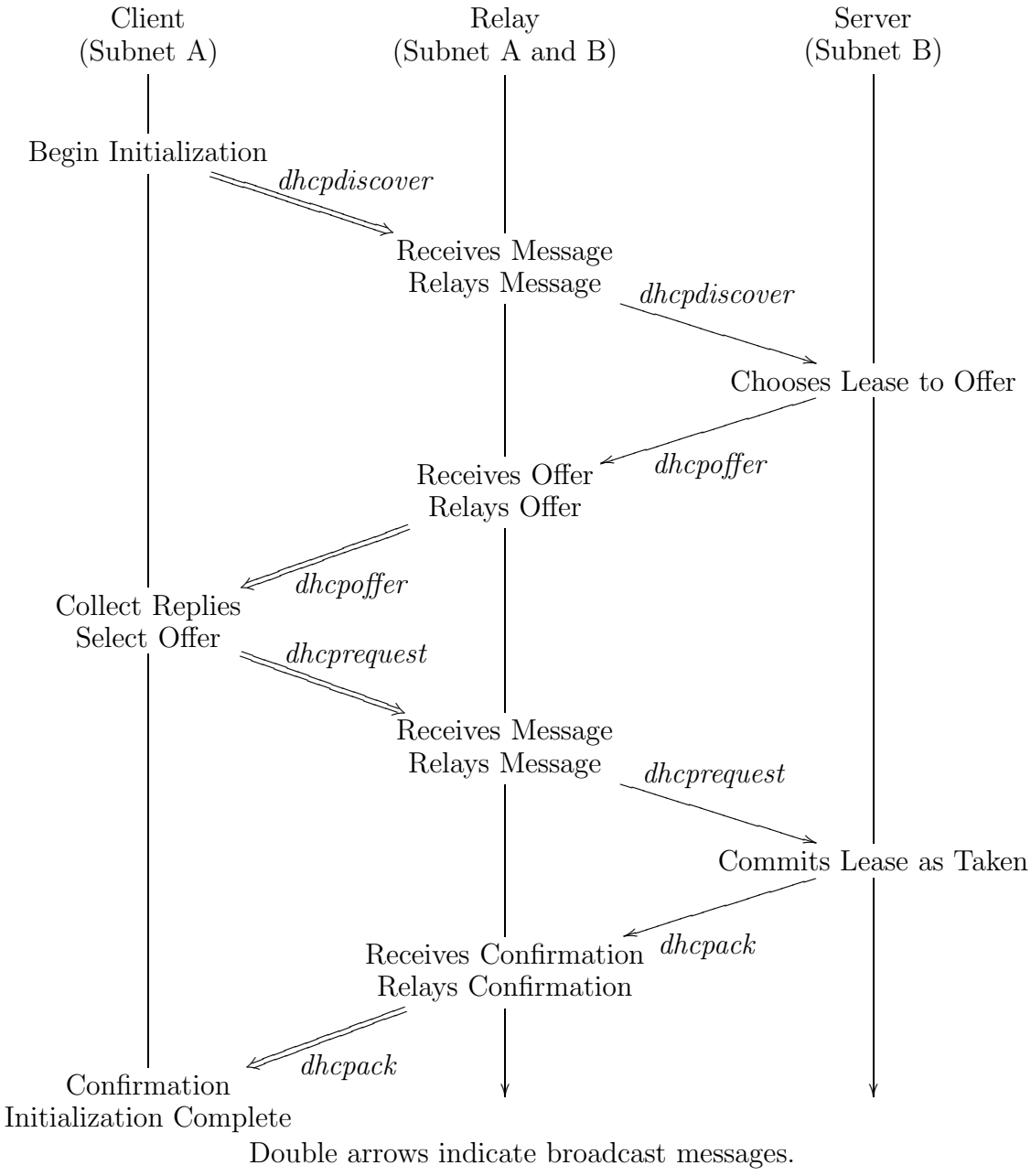


Figure A.4: Obtaining A DHCP lease with a relay [35].

server is on subnet B, a separate subsection of the network. Broadcast traffic is confined to individual subnets. The relay is, of course, on both subnets as it must act as a bridge between the two.

With this configuration, in all the messages received by the DHCP server, the “source address” field in the link layer message would be the MAC address of the DHCP relay, and not that of the DHCP client. However, the “chaddr” field of the messages received by the DHCP server would contain the MAC address of the client.

This is why we use the “chaddr” field of the DHCP message instead of the “source address” field of the link layer message.

Bibliography

[1] Apple Computing. Open Transport Versions. http://developer.apple.com/qa/nw/nw64.html.

[2] Apple Computing. Technical Note TN1142. http://developer.apple.com/technotes/tn/tn1142.html.

[3] Apple Computing. The Evolution of Darwin. http://developer.apple.com/darwin/history.html.

[4] Arizona State Legislature. Time Limitations. In *Arizona Revised Statutes*, title 13, chapter 107, section B.1. Arizona State, 1956. 13 A.R.S §107.B.1.

[5] Bruce Bahlmann. Birds-Eye.Net Carrier-Class DHCP Testing Setup. http://www.birds-eye.net/technical_archive/carrier_class_dhcp_testing.htm.

[6] BSD Group. Mac OS X Man Pages : ifconfig BSD System Manager's Manual. http://developer.apple.com/documentation/Darwin/Reference/ManPages/man8/ifconfig.8.html.

[7] Cobalt Networks, Inc. Cobalt Qube 2 User Manual. http://www.sun.com/hardware/serverappliances/pdfs/discontinued/manual.qube2.pdf.

[8] Connecticut State Legislature. Limitation of Prosecutions. In *General Statutes of Connecticut*, title 54 chapter 966, section 53-193.(b). Connecticut State, 2001. 54 Conn. 996 §53.B.1.

[9] David M. Smith Ph.D. The Cost of Lost Data. *Graziadio Business Report*, 6(3), 2003. `http://gbr.pepperdine.edu/033/dataloss.html`.

[10] Éric Lévénez. Windows History. `http://www.levenez.com/windows/history.html`.

[11] Frans Vandendries of Vidéotron Communications inc. DHCP Server Information Requirements. Technical report, CRTC, 2001.

[12] Fred N. van Kempen and Alan Cox and Phil Blundell and Andi Kleen. ifconfig(8).

[13] Fyodor. Remote OS detection via TCP/IP Stack FingerPrinting. `http://www.nmap.org/nmap/nmap-fingerprinting-article.html`, October 1998.

[14] G. Pascal Zachary. *Show-Stopper!: The Breakneck Race to Create Windows NT and the Next Generation at Microsoft*. Free Press, 1994. ISBN 0029356717.

[15] Google Inc. Google Zeitgeist. `http://www.google.com/press/zeitgeist/zeitgeist-jun04.html`.

[16] IEEE 802 Working Group. *IEEE Standards for Local and Metropolitan Area Networks: Overview and Architechture*. IEEE, 2001.

[17] IEEE Registration Authority. Registration Authority Frequently Asked Questions. `http://standards.ieee.org/faqs/OUI.html`.

[18] Intel Corporation. 82547GI(EI)/82541GI(EI)/82541ER EEPROM Map and Programming Information Guide. `http://www.intel.com/design/network/applnots/ap446.pdf`.

[19] Intel Corporation. ERUPDATE - Intel 82559ER Fast Ethernet PCI Controller EEPROM Utility. http://www.intel.com/design/network/drivers/erupdate.txt.

[20] Intel Corporation. Intel®Pentium®III Processor, Processor Serial Number Questions and Answers. http://support.intel.com/support/processors/pentiumiii/sb/CS-007579.htm.

[21] Internet Assigned Number Authority. Address Resolution Protocol Parameters. Technical report, Internet Corporation for Assigned Names and Numbers, 2005.

[22] ISC. ISC Dynamic Host Configuration Protocol. http://www.isc.org/index.pl?/sw/dhcp/.

[23] JTC 1/SC 6. *ISO/IEC 15802-1: Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Common specifications – Part 1: Media Access Control (MAC) service definition.* ISO/IEC, 1995.

[24] kingpin@l0pht.com. MAC Address Cloning. http://www.zone-h.org/download/file=4197.

[25] Kyle Lai CISSP CISA of KLC Consulting Inc. Spoof MAC address on Windows 2000, XP, 2003 Server. http://www.klcconsulting.net/Change_MAC_w2k.htm.

[26] Kyle Lai CISSP CISA of KLC Consulting Inc. Spoof MAC address on Windows 98/ME. http://www.klcconsulting.net/Change_MAC_w98.htm.

[27] Kyle Lai CISSP CISA of KLC Consulting Inc. Spoof MAC address on Windows NT. http://www.klcconsulting.net/Change_MAC_wnt.htm.

[28] Marc Weber Tobias. Updated Security Alert: Notebook Computer Locking Devices. http://www.security.org/dial-90/sl-new.htm.

[29] Mark E. Doms and Wendy E. Dunn and Stephen D. Oliner and Daniel E. Sichel. How Fast Do Personal Computers Depreciate? Concepts and New Estimates. Working Paper, US Federal Reserve, November 2003.

[30] Microsoft. Windows Product Family - Product Lifecycle Dates. `http://support.microsoft.com/default.aspx?scid=fh;[ln];LifeWin`.

[31] Microsoft. Custom DHCP Client Identifiers for Windows NT. `http://support.microsoft.com/kb/172408/EN-US/`, June 2003. Article ID 172408.

[32] Microsoft. DHCP (Dynamic Host Configuration Protocol) Basics. `http://support.microsoft.com/kb/169289/EN-US/`, June 2003. Article ID 169289.

[33] Network Working Group. The IP Network Address Translator (NAT). RFC 1631, Internet Engineering Steering Group, 1994.

[34] Network Working Group. DHCP Options and BOOTP Vendor Extensions. RFC 2132, Internet Engineering Steering Group, 1997.

[35] Network Working Group. Dynamic Host Configuration Protocol. RFC 2131, Internet Engineering Steering Group, 1997.

[36] Paul Thurrott. Microsoft outlines future of Windows NT and Windows 95. `http://www.winnetmag.com/Article/ArticleID/16867/16867.html`.

[37] Peter Bartoli. MAC Spoofing on the Mac. `http://slagheap.net/etherspoof/`.

[38] Red Hat, Inc. pump(8).

[39] RFC Editor. RFC Copyrights. `http://www.rfc-editor.org/copyright.23Jan01.html`.

[40] Robert Richardson. Computer Crime and Security Survey. Technical report, CSI/FBI, 2003.

[41] Safeware Inc. Safeware 1996 Loss Study. Technical report, Safeware Insurance Agency, 1996.

[42] Safeware Inc. Safeware 2001 Loss Study. Technical report, Safeware Insurance Agency, 2001.

[43] Safeware Inc. Safeware 2002 Loss Statistics Notebooks and Desktops. Technical report, Safeware Insurance Agency, 2002.

[44] Stefan Esser. Spoofing the MAC address on Airport Extreme cards. `http://wishlist.suspekt.org/`.

[45] Tadayoshi Kohno and Andre Broido and K.C. Claffy. Remote Physical Device Fingerprinting . *IEEE Symposium on Security and Privacy 2005 and IEEE Transactions on Dependable and Secure Computing*, 2005. There are three versions of this paper in circulation. We reference the full version found here: `http://www.cse.ucsd.edu/users/tkohno/papers/PDF/`.

[46] The Center for Democracy and Technology. Intel Pentium III Processor Serial Number. `http://www.cdt.org/privacy/issues/pentium3/`.

[47] Vermont State Legislature. Limitation of Prosecutions for Certain Felonies. In *Vermont Statutes*, title 13, chapter 151, section 4501.(b). Vermont State, 1959. 13 V.S.A. §4501.(b).

[48] Vidéotron Communications inc. TPIA Phase 3 Testing Report. Technical report, CRTC, 2000.

[49] Ville Aikas. DHCP Client Simulator. `http://staff.washington.edu/~aikasevj/dhcp_client/`.

[50] W. Richard Stevens. *TCP/IP Illustrated, Volume 1.* Addison-Wesley Professional, 1993. ISBN 0201633469.

[51] Wikipedia. IPv6. http://en.wikipedia.org/wiki/Ipv6.

[52] Wikipedia. MAC address. http://en.wikipedia.org/wiki/MAC_address.

[53] Wikipedia. System 7 (Macintosh). http://en.wikipedia.org/wiki/System_7_(Macintosh).

[54] Wikipedia. Windows registry. http://en.wikipedia.org/wiki/Windowsregistry.

[55] Yoichi Hariguchi and Sergei Viznyuk. dhcpcd(8).

Acronyms

AEP AppleTalk Echo Protocol

ANSI American National Standards Institute

API Application Program Interface

BOOTP Bootstrap Protocol

CMTS Cable Modem Termination System

CRTC Canadian Radio-television and Telecommunications Commission

CSI Computer Security Institute

DHCP Dynamic Host Configuration Protocol

DOS Disk Operating System

EEPROM Electronically Erasable Programmable Read-Only Memory

FBI Federal Bureau of Investigation

GUI Graphical User Interface

HKLM HKEY_LOCAL_MACHINE

HTTP HyperText Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

IP Internet Protocol

IPCP Internet Protocol Control Protocol

ISC Internet Software Consortium

ISP Internet Service Provider

MAC Media Access Control

MHz megahertz

MIPS Microprocessor without Interlocked Pipeline Stages

NAT Network Address Translation

NIC Network Interface Card

OUI Organisational Unique Identifier

PC Personal Computer

PCI Peripheral Component Interconnect

PCMCIA Personal Computer Memory Card International Association

PDA Personal Digital Assistant

POSIX Portable Operating System Interface

PPP Point-to-Point Protocol

PPPoE Point-to-Point Protocol over Ethernet

RFC Request For Comments

ROM Read-Only Memory

TCP Transmission Control Protocol

TFTP Trivial File Transfer Protocol

UDP User Datagram Protocol

USB Universal Serial Bus

Appendix B

RFC Copyrights

The following text, with an edit providing a link to the copyright notice in place of reproduction, discusses the copyrights attached to RFC and is authored by the RFC Editor [39]:

To protect the integrity of RFC publication, all RFCs are with published an ISOC copyright statement. The text of the standard copyright statement can be found at ftp://ftp.rfc-editor.org/ in-notes/rfc-editor/copyright.23Jan01

This copyright notice was designed to ensure that Request For Comments documents will have the widest possible distribution. The following general guidelines control the reproduction and modification of RFCs.

1. Copying and distributing an entire RFC without any changes:

    1a. The copying and free redistribution are generally encouraged.

    1b. The inclusion of such RFC copies in other documents and collections that are distributed for a fee is also encouraged. However, in this case it is a

courtesy (i) to ask the RFC author and (ii) provide the RFC author with a copy of the final document or collection.

Anyone can take some RFC, put them in a book, copyright the book, and sell it. This in no way inhibits anyone else from doing the same thing, or inhibits any other distribution of the RFCs.

2. Copying and distributing the whole RFC with changes in format, font, etcetera:

   2a. The same as case 1, with the addition that a note should be made of the reformatting.

3. Copying and distributing portions of an RFC:

   3a. As with any material excerpted from another source, proper credit and citations must be provided.

4. Translating RFCs into other languages:

   4a. Since wide distribution of RFCs is very desirable, translation into other languages is also desirable. The same requirements and courtesies should be followed in distributing RFCs in translation as would be followed when distributing RFCs in the original language.

The RFC Editor
Last revised: 23 January 2001