Securing Network Location Awareness with Authenticated DHCP

Tuomas Aura
Microsoft Research
Cambridge, UK

Michael Roe
Microsoft Research
Cambridge, UK

Steven J. Murdoch
University of Cambridge
Cambridge, UK

Abstract Network location awareness (NLA) enables mobile computers to recognize home, work and public networks and wireless hotspots and to behave differently at different locations. The location information is used to change security settings such as firewall rules. Current NLA mechanisms, however, do not provide authenticated location information on all networks. This paper describes a novel mechanism, based on public-key authentication of DHCP servers, for securing NLA at home networks and wireless hotspots. The main contributions of the paper are the requirements analysis, a naming and authorization scheme for network locations, and the extremely simple protocol design. The mobile computer can remember and recognize previously visited networks securely even when there is no PKI available. This is critical because we do not expect the majority of small networks to obtain public-key certificates. The protocol also allows a network administrator to pool multiple, heterogeneous access links, such as a campus network, to one logical network identity. Another major requirement for the protocol was that it must not leak information about the mobile host's identity or affiliation. The authenticated location information can be used to minimize attack surface on the mobile host by making security-policy exceptions specific to a network location.

I. Introduction

This paper describes a secure network-location-awareness (NLA) mechanism, based around authentication of the dynamic host configuration protocol (DHCP) server. The goal is to provide a ubiquitous mechanism by which a mobile host can securely identify the network to which it is attached. It is intended to be used not only on managed company networks but also at a home, small office, coffee shop, or university campus. The protocol works over the IP layer without needing any special hardware, which leads to both fundamental limitations and advantages.

Our protocol authenticates networks based on their public keys. Networks may have public-key certificates but they are not required. The mobile host remembers networks when it meets them for the first time and securely recognizes them on following visits. It can then store and automatically apply security settings, such as firewall configuration, enabling or disabling network services, and generating or reusing pseudonyms for privacy preserving protocols.

A key insight motivating our work is that many network-security settings are not strict policies but rather safe defaults which users and applications are allowed to and often do override in order to get their work done. Network location awareness enables the computer to remember the modified settings on a per-network basis and to revert to the safe baseline for each new network. Thus, secure NLA enables a new balance between convenience and risk minimization.

This paper's main contributions are the requirements analysis, the idea of recognizing previously visited networks based on their public key, a method for deriving network names and authorization for NLA from X.509 certificates, and the easy-to-implement authentication protocol. We implemented a prototype of the protocol for a Windows 2003 server and Vista client, and use the Windows network location awareness as an example throughout the paper. Any other mobile platform that implements some NLA functionality would benefit equally from the secure location information.

The rest of the paper is organized as follows: section II motivates the work by explaining the need for a new secure location mechanism. Section III outlines our initial design choices. Section IV discusses network naming and authorization in detail. In sections V–VI, we give an overview of the DHCP protocol and show how our authentication extensions fit in. Section VII discusses residual threats, section VIII surveys related work, and section IX concludes the paper.

II. Need for network authentication in NLA

When a Windows computer connects to a new network, it asks the user to determine whether the network is private (work or home) or public. Windows firewall uses this information to select a firewall profile for the network. On following visits to the same network, the same profile is selected automatically. If the computer is a member of a managed domain, there is also a separate profile for the domain network. This is a simple example of how NLA can be used to configure security settings depending on the network location. While the profiles are easy to understand and manage, there is no technical reason why the computer could not store individual settings for each network. Applications and services can also take advantage of NLA and remember settings that are specific to a network location. Such location-aware features will probably increase in sophistication as users and developers get used to them.

Since the location information from NLA is used for changing security settings, we have to ask how reliable this information is. Windows NLA currently identifies the network based on heuristics that take into account various clues including the gateway MAC address, SSID on a wireless link, wireless authentication state, and the presence of a domain

controller on a managed network. Some networks are identified securely: the domain network by authenticating the domain controller, secure wireless LANs with 802.1X or shared-key authentication, and VPN and dialup connections with their specific security mechanisms. Most networks, however, are not strongly authenticated. In particular, wired Ethernet and open wireless access points are identified primarily by the gateway-router MAC address. This includes networks that use the universal access method (UAM), i.e., a web form and password, for access control.

This mechanism is not entirely insecure. Although an attacker could spoof the identity of a private network and cause the mobile computer to use weaker security settings than it should use at a public location, it needs to know the gateway MAC address of private network. The MAC address can be easily discovered by sniffing the local link but that usually requires a visit to the physical location of the access network. For many applications of NLA, this may provide an acceptable level of security. It would, however, be safer to authenticate the network with a stronger mechanisms before applying location-specific security settings.

The reliance on the gateway MAC address causes, however, another limitation of the current NLA mechanism. Unmanaged networks can only be identified at the granularity of a single IP subnet. Some organizations, such as universities and commercial hotspot operators have large, heterogeneous access networks that span multiple IP subnets. Yet, it might be desirable to treat such a network as a single entity and to set the security profile once for the entire university campus or hotspot chain, rather than require the user to configure the settings separately for each IP segment. This calls for a network authentication mechanism that supports secure aggregation of multiple access links to one network identity regardless of the network topology.

Once the location information is uniformly authenticated, it will become much more attractive to use it in security and privacy mechanisms. By adapting the firewall configuration to each network and by enabling and disabling applications and network services, the mobile computer can minimize its attack surface at each location. Disabling unnecessary services and protocols can also limit the amount of information leaked to curious observers about the mobile user and his affiliation.

III. Initial design choices

In this section, we discuss some of our fundamental design choices. We also list high-level requirements for the cryptographic protocol that we will describe in section VI.

A. IP-layer attacker model

Our goal is to ensure that a mobile computer can securely identify the network to which it is attached. That is, we want to prevent an attacker from spoofing the network location. Moreover, we want the protocol to work over the IP layer. This makes it independent of the of the underlying physical network technology but puts some limitations on the kind of security we can achieve.

At the IP layer, we can verify the logical presence of a network, not its physical proximity. We cannot rely on carefully designed hardware and accurate timing of physical signals and, thus, cannot measure physical distances, as is done in distance bounding protocols (see section VIII). Consequently, we cannot defend against attackers who have access to both the network location which they want to spoof and to the local link of the mobile host. Such attackers could tunnel packets between the two locations and, thus, effectively modify the logical network topology. The security of our protocol depends on physical and logical network boundaries, such as routers and firewalls, to partition the Internet and to limit the locations at which the attacker can spoof, sniff and relay IP packets. In practice, the cost and inconvenience of having to be present at the two locations is sufficient to prevent most attacks. We have opted for this lower level of security in exchange for easy universal deployment that does not require any changes to the physical and link layer technologies.

To state the goal and major assumption of our protocol more precisely:

- We aim to prevent an attacker from spoofing network locations to a mobile host. That is, the host should be able to verify its location reliably in the presence of a malicious attacker.
- We assume that the attacker cannot relay packets in real time (i.e., within seconds) between the mobile computer's real location and the network location that it wants to spoof.

An existing security mechanism in Windows NLA, authentication of a domain controller, already works at the same level of security as our protocol. That is, it depends on a firewall to prevent connections to the domain controller from outside the domain network.

Since our security mechanism works on the logical rather than physical level, it can be analyzed against a Dolev-Yao-style attacker [Dolev83] who has no timing constraints and has access to all communication. One only needs to include in the model the assumption that the attacker cannot relay packets between the two locations.

B. Using DHCP

As already mentioned, we propose to verify the network location by authenticating the dynamic host configuration protocol (DHCP) [Dro97] server on the network. The main reason for this is that DHCP is the one ubiquitous service that is deployed on virtually every wired and wireless network to which client hosts can connect. This allows us to provide secure location information on almost all access networks by modifying just one service.

The main exceptions to the availability of DHCP are cellular data and point-to-point links such as dialup and VPN

connections. These types of networks are typically already strongly authenticated by link-layer security mechanisms, on which we can continue to rely. Nevertheless, some networks like isolated wired LANs with manually configured IP addresses and without any servers or gateways will always remain without secure identification.

An alternative to the use of DHCP would be to disseminate the secure location information via a combination of services available on different networks, such as wireless access points, file servers, Kerberos, RADIUS and AAA servers, Windows domain controllers, and so on. Clearly, it is easier to implement and deploy the security protocol only in one place. For IPv6, authenticating the access router would be a logical choice: the protocol for router discovery could be enhanced to provide secure location information. Most IPv6 links will, however, have a DHCP server to configure parameters such as the local domain suffix [DBV+03]. We decided to concentrate on DHCP and to leave it as future work to extend our secure NLA protocol to work over IPv6 neighbor discovery.

Another reason for focusing on DHCP is that it is relatively easy to fit the required data into the DHCP messages. Indeed, DHCP is specifically intended for configuring hosts with all kinds of information about the network and it readily allows extensions without breaking backward compatibility with old servers and clients that do not support the new features. (More specifically, we can send public-key signatures in non-critical long vendor options.)

A third, and equally important, reason for choosing DHCP is that it is typically the first protocol executed by a mobile host when it enters a new network (apart from the 802.1X authentication on some wireless networks, see section VII). The secure location information should be obtained as early as possible so that the mobile host can use it to make decisions about which other services to use and which protocols to enable on the new network. The later in the network attachment process the mobile authenticates the network, the more trust it must put in the unknown environment before the authentication.

Finally, DHCP is a protocol that is executed entirely on the local link (or adjacent links connected by BOOTP relays) using mostly broadcast messages. It does not require the mobile host to use or know any server identifiers or addresses. Neither does it require the client to have or reveal any identifiers or addresses of itself (the dangers of which will be discussed in section F). Therefore, it is possible to complete the DHCP protocol without the client revealing any information to the network.

C. Public key as a secure network identifier

In this subsection, we argue that public keys as machine-readable network identifiers have the flexibility to support various scenarios from PKI-based authentication of large and heterogeneous networks to the creation of ad-hoc relations with isolated LANs.

The first reason for choosing public-key authentication is that it is more suitable for global operation than existing secret-key protocols. Kerberos, the most common secret-key protocol in IP networks, is rarely used outside closed domains and, thus, is not suitable for roaming users. The current Windows NLA already takes advantage of Kerberos when authenticating the domain network but it would be difficult to extend this to non-domain networks. SIM-based authentication protocols from cellular networks have been adapted for mutual authentication between mobile computers and access networks; however, they require the access networks to have a roaming agreement with the SIM issuer and a connection to its home location register and authentication center (HLR/AuC). With our protocol, a mobile computer should be able to authenticate access networks anywhere in the world, not only a specific operator's network or a limited set of hotspots with roaming agreements.

Another justification for the public-key protocol is that public-key certificates enable offline operation. It is sometimes desirable to authenticate networks that are disconnected from the global Internet. Moreover, some networks require a mobile computer to go through a complex procedure, such as entering a password into a web form or installing VPN software, before granting Internet access. In these situations, public-key credentials enable authentication of the access network before obtaining a connection to the Internet. Our network authentication protocol can be seen as an additional first layer of defense to these more complex security protocols, although interaction between the protocols is not the focus of this paper.

Finally, and perhaps most importantly, public keys can be used to identify networks even when there are no certificates available. A mobile computer can securely recognize networks that it has previously visited based on their public keys. This makes it possible to authenticate a network that has no administration and no certificates, such as a home LAN or an individual wireless hotspot. When the mobile computer returns to the same network, it can recall the configuration and services used there on the previous visit. This fits the model of Windows NLA where the computer asks the user once to determine the status of the network and remembers this choice when the user returns to the same network. Our protocol increases the security of this mechanism for identifying previously visited networks without requiring changes to the user interface.

D. Administrative identifiers

While public keys are practical network identifiers for the internal use of the NLA implementation, users and administrators will want to see a more familiar type of a network name. Windows NLA currently sets the network name to the wireless SSID where one is available but allows users to edit the name. When there is more than one network with the same name, these are numbered. In most situations, our secure NLA protocol does not need to change this naming

scheme: the names allow the user to uniquely identify previously visited networks.

Sometimes, however, the operating system or applications may have a security policy (e.g., a Windows group policy) that is specified based on human-readable network names. In that case, the names must be globally unique and there must be a secure way of assigning them to networks. A wireless SSID is not suitable for this purpose because there is no effective mechanism for assigning globally unique SSIDs or for verifying their ownership.

For these reasons, we suggest using a DNS suffix as the network name whenever a globally unique name is needed. DNS names have an effective global allocation scheme. They often correspond to natural administrative boundaries and, because of the hierarchical structure, allow variable granularity for the network naming. As we will explain in detail is section IV, the binding between DNS suffixes and certificates can be secured using public-key certificates.

E. Authentication and authorization

In this paper, we use frequently the word *authentication*. Since this word has many meanings, it requires some further explanation. It is important to understand the difference between the abstract idea of identifying a location securely and the concrete mechanism of authenticating the DHCP server.

Our ultimate goal in this paper is to identify and authenticate the network location. We do this in two ways. First, we recognize networks and ensure that the same network name, when encountered again, either refers to the same network or, if it does not, is treated as a new network. Second, where certificates issued by a trusted certification authority (CA) exist, it is possible to authenticate the globally unique network name. In practice, we believe that the first type of authentication is sufficient to prevent most location-spoofing attacks, and that it is unnecessary for most applications and users to consider the difference between the two types of authentication. The optional certification of networks allows the system administrators to manage the users' security and assign security profiles to networks, e.g., via Windows group policy. This frees the user from making any judgments over security settings when encountering the specific networks.

Since we are, on the abstract level, not interested in the identity of the DHCP server but rather the location that it advertises, the critical question is not who the server is but whether it is *authorized* to advertise the specific location. The authorization may arise from two sources. (Such sources of authority are often called *roots of trust*.) In our system, *a* DHCP server that first advertises a network name gains automatically authority over that name. This control is limited to the namespace of the particular client machine. Another possible source of authorization is a certificate chain that starts from a trusted certification authority (CA). If a conflict arises between the names from the these two sources, we may have to ask the user to resolve it. Given the uniqueness of the DNS names, we do not expect such conflicts to occur often between honest access networks. For SSIDs, conflicts could be more common.

On the protocol level, we are authenticating the DHCP server, which is *entity authentication*. Although we implement this by signing messages sent by the server, it is important to keep in mind that the goal is not data-origin authentication. It does not even matter very much which particular message is signed. As a side effect of signing the DHCP responses, we may get some security for the data in the messages, i.e., the various host configuration parameters. That is, however, not the goal of the current work and anyone who relies on such data authentication should assess separately its suitability for the particular application that they have in mind.

F. Privacy of mobile users

While verifying the location, we want to protect the privacy of the mobile user. The client should be able to remain anonymous towards the access network and all other nodes there at least until it has verified the identity of the network and, possibly, throughout the network access. This means that the secure NLA protocol should *not* require mutual authentication and it should not leak information about the user to the access network. We will approach this problem by explaining first why some existing security mechanisms are not sufficiently privacy friendly even though they in some other respects might be acceptable mechanisms for secure NLA.

In corporate wireless LANs, the client and network typically authenticate each other with the 802.1X [EA03] protocol and public-key certificates. The information from the server certificate can be used to secure location awareness. 802.1X, however, is intended as an access control mechanism for closed networks and, thus, puts emphasis on authenticating the client. Since 802.1X allows the addition of new authentication methods, e.g., EAP-TLS for authentication with TLS/SSL certificates, EAP-SIM for authentication with a GSM SIM, and PEAP for password-based user authentication. Of these, the PEAP method protects the client's privacy by first authenticating the server and then performing the client authentication inside the established secure channel. This hides the client identity from both passive observers and rogue access points. If we had been interested in securing NLA only in wireless LANs and not in wired networks, we might have chosen to define a new EAP method with server-only authentication.

When a Windows machine attaches to its domain network, it connects to the domain controller (DC) and performs a Kerberos authentication. Windows NLA uses this authentication as a secure mechanism for verifying that the machine is connected to the domain network. Authenticating the domain controller gives the same level of security as the DHCP-based protocol presented in this paper; i.e., the attacker has to have an accomplice inside the domain network in order to spoof the location to a mobile machine that is roaming

elsewhere. The limitations of the DC authentication are that it works only for one administrative domain per machine and, thus, cannot be used to authenticate home or visited networks, and that attempting connection to the domain controller leaks information about the mobile computer's domain name (because it involves resolving the DNS names _kerberos._udp. *DomainName* and _ldap._tcp.dc._msdcs. *DomainName*). This is sufficient information to find out the organization to which the mobile computer belongs. An active attacker could also spoof responses to these DNS requests, which would cause the client to send a Kerberos authentication request (KRB_AS_REQ) to the supposed domain controller. The authentication request contains the client host name as well as its domain suffix (i.e., Kerberos realm) in plaintext. In conclusion, the DC authentication, just like 802.1X, does not give any anonymity for the mobile computer.

The DHCP-based protocol described in this paper has the advantage of not requiring client identification at all. Moreover, the clients broadcasts its DHCP messages on the access link. Thus, the client does not need to know a specific server name or try to resolve it to an IP address. This means that the client can authenticate the network location without revealing anything to the network about itself or its affiliations.

It is well-known that an Ethernet adapter can be tracked based on its hardware (MAC) address [EP02]. An IPv6 address may similarly contain parts that allow correlation between appearances of the same host. These problems can be solved by randomizing the hardware address and IPv6 address periodically [LT06][ND01]. For full privacy protection, our protocol should be used together with such address randomization. Nevertheless, even if the MAC address remains a unique identifier, hiding the higher-layer identifiers makes identification of mobile computers significantly less practical for local attackers, such as individual users.

G. Protocol requirements

We summarize the design goals for our protocol:

- increase reliability of NLA by authenticating the DHCP server on the access network;
- recognize previously visited networks securely based on their public key;
- if a network has a globally unique name and is certified, authenticate it using a PKI;
- protect user privacy by not revealing the user identity, affiliation, or the list of user's preferred networks to either passive or active adversaries;
- allow the client to roam in networks that do not support the new protocol without compromising security of NLA at networks that do;
- allow future use of the authentication mechanism for other security purposes, such as preventing general spoofing attacks against DHCP.

In rest of the paper, we will discuss in detail how these requirements influenced the protocol design.

IV. IDENTIFYING AND NAMING NETWORKS

This section explains in detail how our protocol recognizes networks and binds names to them. We start with the simplest case of uncertified small networks with only one DHCP server, which are typical at home or in coffee shops. We then consider certified network names, used in closed domains and commercial access networks. Combining ideas from these two cases allows us to finally tackle the most difficult network type: large, uncertified and heterogeneous networks with multiple DHCP servers, such as university campus infrastructure.

A. Uncertified small networks

Most access networks are set up by individuals and small businesses who do not want to go through the trouble and expense of obtaining a certificate from a commercial certification authority (CA). They often do not have the resources and expertise to set up their own CA, which would require them not only to deploy server certificates to the DHCP servers but also to distribute the root certificate to all potential clients.

In these situations, the client will not be able to rely on a trusted authority to verify the network identity. Instead, we want the client to learn the network key when it comes to the network for the first time and then recognize the network as the same one on the following visits. This way, the client machine can memorize security settings for the network, such as the firewall profile. On the following visits, the client can verify the network identity using DHCP authentication and automatically enable the previous security settings. This improves the security of NLA in wired and open wireless access networks, which can currently be spoofed by anyone who knows the gateway MAC address.

Implementing the secure recognition of previously visited small networks is straightforward: the DHCP server on the network presents a self-signed X.509 server certificate to the client. The client stores this certificate (or a hash of the public key) with the network settings and, on the following visits, requires the network to authenticate itself using the same key. If the key changes, the network is treated as a new one. The user-readable name of the network can be taken from the certificate (we'll discuss the names in certificates below) but other names such as the wireless SSID can be used as well. A limitation of this scenario is that it is does not scale to larger networks with multiple DHCP servers. If there are several DHCP servers on the network, they would have to share the same secret signature key. Otherwise, the client will treat the networks as separate.

B. Certified network names

Network operators with high security requirements and sufficient resources may want to certify their DHCP servers in

order to give the networks authenticated, globally unique names. Large closed organizations can issue their own certificates while small organizations and those operating open access networks tend to rely on well-known commercial CAs like Verisign or CyberTrust.

For many network operators, the cost of issuing or purchasing new certificates specifically for network authentication may be too high. For this reason, we would like to give network administrators the option of reusing existing server certificates, including SSL server certificates, for the DHCP-server authentication.

On the conceptual level, implementing the authentication is straightforward: the DHCP server presents a certificate chain to the client. The client verifies that the certificate chain starts from a trusted root CA and ends in a server certificate issued to the DHCP server. (Because of space limitations in DHCP messages, which must fit into one unfragmented IP packet on the local link, the chain should typically be just a single certificate or at most a few certificates long.) The client then takes the network name from the server certificate.

There is a problem with the names in existing X.509 certificates, though. As explained earlier, we want to use DNS domain suffixes as network names. For example, a network could be called *contoso.com*. The server certificate issued to the DHCP server, on the other hand, is an X.509 end-entity certificate that specifies a fully qualified domain name (FQDN). Thus, the DHCP server's certificate will specify a host, not a suffix. That could, for example, be *dhcpserver.sales.contoso.com*. We could extend the certificate standard by specifying a new name type for network names (in the SubjectAltName field). The problem with that approach is that it may take years to standardize such extensions, to deploy support for them in X.509 certificate implementations, and to gain acceptance of commercial CAs. Hence, it is desirable to use the existing name types if at all possible.

Our solution is to take as the network name the local DNS suffix advertised by the DHCP server. *The client will verify that the advertised DNS suffix is a suffix of the server FQDN in the server certificate.* The suffix may be the entire FQDN, just the last segment like *com*, or something in between. This simple policy may sound surprising and requires further explanation.

First, consider the server *dhcpserver.sales.contoso.com.* The organization may decide to name its networks by the department (*sales.contoso.com*) or treat them as one large network (*contoso.com*). Our policy for naming allows any departmental DHCP server to use the longer and more accurate sub-domain name or the shorter and coarser company-wide name. In some sense, we are defining the semantics of hierarchical network names as membership: servers in sub-domains are also members of the organization as a whole and may represent it.

Second, consider what happens if a rogue server *dhcpserver.sales.contoso.com* claims plain *com* as its network name, or *dhcpserver.contoso.co.uk* claims *co.uk*. Obviously, no honest server would do that. A malicious server, on the other hand, could name its network *com* or *co.uk* and impersonate any other network that uses the same name. But that does not matter because no honest network uses such names. It is important to note that a DHCP server or network that assumes the name *com* will not have any authority over longer names such as *contoso.com*. Any server that selects too short a name for its network will only compromise the security of its own network, not anyone else's. The same applies to rogue departmental servers that try to hijack an unused organizational name.

Another way to think about the semantics of the names is that when a DHCP server chooses the name for its network, it authorizes others to be (or to pretend to be) the same network. The longer name it chooses, the fewer other servers, if any, are allowed to advertise the same network name. The shorter name it chooses, the more other servers it is permitting to be (or to pretend to be) on the same network.

This scenario scales to arbitrarily large networks, including corporate intranets distributed across multiple continents. A limitations is the need for a common trusted root CA between the servers and clients.

C. Uncertified large networks

The scenarios discussed above allow the mobile client to name and remember small uncertified networks and to authenticate globally unique names of certified networks. By combining the solutions from these two scenarios, we can also support secure NLA in large uncertified networks that have multiple DHCP servers.

The idea that the client accepts any certificate or certificate chain even if it does not recognize the root CA. The client then remembers the root CA and the local DNS suffix advertised by the DHCP server.

When the client connects to the same network again, it will require the network to authenticate itself using a certificate chain that starts from the same root key as on the first visit. The previous security settings will be applied only after successful authentication. If either the root CA or the advertised network name differ from the previous visit, the network is treated as a new one.

As in the case of small uncertified networks, the user-readable network name can be anything, although it probably makes sense to use the advertised DNS suffix as the default name. If multiple networks have the same name, these can be numbered (e.g., *contoso.com* and *contoso.com 2*). These names are local to the client machine and can be used to refer to only local security settings, not to any policy that uses globally unique names.

This scheme allows the network operator to issue certificates to the DHCP servers in order to bind them into one logical network location. It is up to the network operator to decide the granularity of the locations (i.e., DNS suffixes) it advertises. For example, a university can certify all DHCP servers on its campus and set them to advertise the same DNS

suffix. It is a major functional improvement over the current unauthenticated NLA mechanism that we can securely aggregate multiple access links into one network identity regardless of the physical network topology.

D. Combined scenario

As the reader may have noticed, the three scenarios outlined above are actually special cases of one uniform network identification mechanism: When the client computer attaches to a network, it receives a certificate chain (often of length one) from the DHCP server. The server also advertises a local DNS suffix. The client verifies that the certificate chain is valid, except that it does not require the chain start from a trusted root CA. It does check all other aspects of the certificate chain, including any name constraints that limit the authority of sub-CAs to subsets of the DNS name space. It also requires the advertised DNS suffix to be a suffix of the server name in the server certificate. The client then uses the combination of the root CA public key and the advertised suffix as the machine-readable identifier of the network. If it has visited the same network previously and has stored permanent security settings for this network, it may configure them automatically. On the other hand, if either the root CA or the advertised suffix differ from previously visited networks, the client treats the current network as a new one.

The network name presented to the user may be the DNS suffix or some other identifier like the SSID of a wireless network. However, if the client has security policies that refer to globally unique network names, it uses the advertised DNS suffix to look up the policy and applies the policy only after verifying that the certificate chains starts from a trusted CA (i.e., the network is "certified"). For the benefit of expert users, the client user interface should give some indication of whether the name presented to the user is globally unique (certified by a trusted CA) or local to the particular computer. Most users, however, will not need to know the difference. Any name conflicts between certified, uncertified but securely recognized, and completely unauthenticated legacy networks can be solved by appending a number to the user-readable network name and treating them as different networks.

V. DHCP OVERVIEW

In this section, we give an overview of the dynamic host configuration protocol (DHCP) with focus on the features that are relevant to our protocol design. Figure 1 shows a typical protocol execution when a mobile computer first enters an access network. Only selected data fields are shown and the fields that reveal the mobile identity or affiliation (at the IP layer or above) have been highlighted.

The execution consists of two request-response pairs, which are sent in UDP packets with the client port 68 and server port 67. The requests are broadcast on the local link with no source address (source 0.0.0.0, destination 255.255.255.255). The responses are sent as unicast from the server IP address to the offered client IP address. (Clients may request broadcast responses.) The messages are linked together by the TransactionID, chosen by the client before it sends the first message. The server identifies the client by the ClientHardwareAddress, which must be unique to each link. The client may include in its messages another identifier field called ClientID, which contains a unique client identifier. Windows sets this field equal to the client hardware address, which is why we won't discuss it further.

		Message	Fields
Client	Broadcast	DHCPDISCOVER	TransactionID, ClientHardwareAddress, **HostName (harrys-laptop)**
Server	Client	DHCPOFFER	TransactionID, ClientHardwareAddress, YourIP, ServerIdentifier
Client	Broadcast	DHCPREQUEST	TransactionID, ClientHardwareAddress, **RequestedIPAddress,** ServerIdentifier, **HostName (harrys-laptop),** **FQDN (harrys-laptop.** **example.org)**
Server	Client	DHCPACK	TransactionID, ClientHardwareAddress, **YourIP,** ServerIdentifier, **Domain (contoso.com)**

Figure 1. Typical DHCP protocol execution

The reason for including the client hardware address in the message content in addition to the packet header is that it is possible for the server to be located outside the access link. In that case there is a BOOTP relay agent on the access link that forwards requests to the server and responses back to the client. We have omitted from the figure most fields related to the relay agent mechanism. It does not affect the design of our protocol apart from the fact that the granularity of secure location awareness will be the set of network links covered by one DHCP server and its relay agents, rather than the individual access link. Routers typically have the capability of acting as BOOTP relays between adjacent links.

The first message exchange consists of the DHCPDISCOVER request from the client and the DHCPOFFER response from one or more servers. The client typically broadcasts its host name, which enables the servers to select host-specific parameters. The offer always contains an IP address and often other host parameters. The client may receive multiple offers. It chooses one offer from one server and broadcasts the DHCPREQUEST, which is interpreted as a request by the chosen server and as a rejection message by all other servers. The request repeats the hostname and may also contain the client's fully-qualified domain name (FQDN) so that the DHCP server can update the client's DNS entry with the new IP address. With the DHCPACK response, the server commits the requested address to the client. This

		Message	Authentication fields
Client	Broadcast	DHCPDISCOVER	Nonce1
Server	Client	DHCPOFFER	Domain suffix, Nonce1, Signature, Certificates
Client	Broadcast	DHCPREQUEST	Nonce2
Server	Client	DHCPACK	Domain suffix, Nonce2, Signature, Certificates

Figure 2. Authenticating DHCP responses

acknowledgement may contain the local domain suffix of the network. The server has no obligation to reserve the address for the client until it sends the acknowledgement. If the address is no longer available when the server receives the request, it will respond with a DHCPNACK message, after which the client may reinitiate the protocol.

There are two kinds of data fields in the DHCP messages that are of particular interest to us: The domain suffix in the acknowledgement is the location identifier that we want to authenticate. We also need to worry about the host name and FQDN in the discovery and request messages because they reveal the client identity and affiliation.

The server typically offers the client only a limited-length lease to the IP address. The client may later renew this lease by executing a shorter protocol with just the request and acknowledgement messages. At that point, the client has already executed the initial four-message protocol on the same link. Sometimes, the client might mistakenly believe it is on the same link when its point of attachment to the network has, in fact, changed. In that case, the client might start the protocol by sending a request to the new link. This request will inevitably leak the client's old IP address to the new network. Since this only happens on rare occasions when the hardware-based or link-layer detection of network attachment fails, we do not try to protect against the information leakage.

In IPv6, stateless autoconfiguration has replaced DHCP as the primary mechanism for address allocation. Nevertheless, DHCP is still needed for configuring other parameters local to the access network, such as the DNS server address. The DHCPv6 protocol execution typically consists of only two messages where the client requests and receives specific parameters.

VI. DHCP AUTHENTICATION PROTOCOL

Figure 2 shows the authentication information that we add to the DHCP messages. The principle is simple: each request contains a client nonce (i.e., a new random number chosen by the client), and each response contains a copy of the client nonce, the server's signature, and the server's certificate chain, which is often just a single self-signed certificate. For network identification, the server must send the local domain suffix, i.e., the location identifier, also in the offer.

It is important to note that the main contribution of this paper is not the protocol itself but rather the careful consideration of the requirements and design details that lead to the choice of such a simple mechanism.

A. What to authenticate

As explained in section IV, the DHCP protocol execution typically consists of two request-response message exchanges: discover-offer and request-acknowledgement. For entity authentication, it suffices to authenticate either one of the responses. It is, however, not straightforward to decide which one.

Our solution, which we will motivate below, works as follows: The client decides which responses need authentication and sends a nonce in the request to indicate this. The server automatically signs the response whenever the request contains a nonce. A simple client will ask the server to sign all responses while a smarter client can leave the nonce out whenever it does not need the security.

The client should always ask the server to sign the offer for privacy reasons. Otherwise, an active attacker could send offers that appear to come from various DHCP servers and observe which of the offers the client prefers. This could be used to discover the client's preferred networks and, thus, its affiliation.

In a 4-message protocol execution, it may be sufficient to authenticate just the offer. This is the case if the only purpose of the authentication is to secure NLA. On the other hand, if the goal is also to authenticate the host-configuration parameters obtained from the DHCP server, then it is necessary to sign also the acknowledgement. Moreover, requiring a signature on all messages when the client knows that the server supports authenticated DHCP may help to protect against various denial-of-service attacks.

In the two-message request-acknowledgement protocol (for requesting a previously allocated address) the client should always ask the server to sign the response in order to detect changes in network attachment in a secure way.

To summarize, the simplest and safest policy for the client is to request a signature for all DHCP responses. In the full 4-message protocol, the client may choose to leave the signature out from the second response. That signature is not required for securing NLA but it may be needed for authentication of other information in the message.

B. Freshness

We use nonces to prevent replay attacks. Since the DHCP protocol consists of request-response pairs and only the responses are authenticated, the client should send a fresh nonce in each request and the server should copy it into the following signed response.

The nonces should be freshly generated random or pseudo-random numbers. They should be unpredictable and must not be repeated with more than negligible probability. Thus, the nonces need to be about 128 bits long. It is not a good idea to use sequence numbers instead of random nonces. The first

reason is that it is easy for the client to accidentally repeat the same sequence after state loss. The second, more important, reason is that the sequence numbers could be used to correlate the appearances of the same mobile computer in different networks.

An alternative to nonces would be to use time stamps, but mobile devices and embedded DHCP servers may not have a real-time clock. (Home routers that implement a DHCP server usually have a clock but it is rarely used for anything other than time stamps in log files and might not be set to the correct time.) There would be no particular advantage to time stamps either. The usual arguments for timestamps are that they save one message (half round-trip time) in the protocol execution and that they can be used to for authenticating broadcast messages. Neither argument is valid in our protocol: the nonces do not add any messages and the responses are unicast.

C. Privacy protection

The fact that our protocol performs only server authentication makes it relatively easy to hide the client identity from both passive and active adversaries in the access network. This would be much more difficult if the protocol required the client to have a certificate or some other type of credential. We also want to hide the client's preference of networks, except the unavoidable disclosure of the network to which the client actually connects. It is not obvious how to do this and we need to find a suitable trade-off between privacy and functionality.

In order to better understand the privacy issues in the DHCP protocol, we need to carefully consider what kind of information the client may leak in the protocol messages and whether we can avoid it. As noted in section IV, the client reveals its hostname and domain suffix in the HostName and FQDN fields. This information should not be sent by clients that wish to hide their identity. Typically, not much functionality is lost by removing the host name from the messages. In theory, the client name could be used by the DHCP server to give the client its old IP address and to configure client-specific host parameters. In practice, most DHCP servers offer the old IP address based on the client's hardware address. The FQDN in the request, on the other hand, is used by many DHCP servers to update the DNS records (either both A and PTR records or only the backward PTR record). For this reason, the client may want to send the FQDN in the request. At that point, it has already authenticated the network as its domain network based on the signed offer. This behavior could be configured by the domain administrators, e.g., via Windows group policy.

Additionally, the client sends its MAC address in the ClientHardwareAddress and optional ClientID fields. We do not address the problems with the MAC address in the paper because reasonable solutions based on address randomization can be found in the literature. Changing MAC address naturally means that the client will not be offered the same IP address as on its previous visit to the same network. This may be an inconvenience for expert users, mainly to those who would like to configure manually TCP or UDP port forwarding at the NAT in a home network. The average user will not notice any difference because all commonly used protocols are designed to cope with changing client IP addresses.

The client may also reveal a previous IP address by requesting it in the RequestedIPAddress field. For a simple client, or one that did not detect the change in network attachment, this may be the last IP address used. For a client that supports network location awareness, it is more likely to be the last IP address used in the same network. A privacy-conscious client should always execute the full 4-message protocol after changes in network attachment rather than reveal its old IP address.

Various DHCP extensions may leak information about host parameters. At minimum, the client must reveal which parameters it wishes to learn from the server. This information alone is not sufficient to identify the individual client but it could be used together with other observations to profile clients. Therefore, mobile computers whose users are interested in anonymity should only request the most common DHCP parameters and not any ones specific to their organization.

A less obvious channel for leaking information is created by the choices that the client makes during the protocol execution. In particular, the client's preferred networks may be revealed by the fact that the client chooses one offer from many. An active attacker could spoof offers with different domain suffixes to find out which one the client prefers. We protect against this threat by signing the offers. The attacker could also replay offers from various networks and observe which offer the client chooses. This is prevented by the nonce in the response as long as the client cannot relay messages in real time.

Finally, we need to consider the contents of the new data fields added by the security protocol. Fortunately, the only new information sent by the client is the random nonce, which does not reveal any information beyond the fact that the client supports authenticated DHCP.

VII. RESIDUAL VULNERABILITIES

In this section, we briefly summarize the known vulnerabilities of the DHCP-based secure location awareness protocol.

The main limitation of the protocol is that an attacker who is present on both the local link of a DHCP server and on the local link of the mobile client can relay messages between the two locations and convince the client that it is on the same link as the DHCP server. For company and home networks, we depend on firewalls to prevent the attacker from accessing the DHCP server. Physical access controls and wireless LAN security also make it difficult for an attacker to insert its own node into the intranet. Wireless hotspots and other public access networks, on the other hand, give relatively easy access

to the attacker. For these networks, the cost of maintaining physical presence at the local link of the DHCP server is relatively low. Fortunately, the consequences of a successful spoofing attack are low as well, because the mobile client will have relatively strict security settings in these networks.

In another form of the tunneling attack, the attacker relays DHCP messages to and from a large number of popular networks and tries to find out which one the client prefers. The cost of this attacks is fairly high compared to the minor compromise of privacy that results. Nevertheless, client implementations may want to protect against this type of attack by receiving only some small maximum number of offers after sending a discovery message and by ignoring the rest.

In secure wireless networks, the 802.1X authentication is executed before DHCP. This may affect client anonymity because the client certificate in EAP-TLS authentication will reveal the client identity before the client has authenticated the network. An attacker could pretend to be a secure wireless access point and wait for clients that try to connect to it. It can then request the client certificate in the TLS handshake. While this is not exactly a vulnerability in our protocol, it is a way of circumventing the privacy protection. There are two immediately available solutions, both somewhat unsatisfactory: the first is to disable automatic connection to certificate-authenticated wireless networks so that users can use their judgment before connecting; the second is to mandate the use of the privacy-protecting PEAP authentication method where a secure tunnel is established to the server before client authentication. Ultimately, one would hope for a more general privacy-protection mechanism for wireless LAN authentication.

The fact that we allow the DHCP server to use any server certificate, such as one issued to an SSL server, also needs discussion. This saves certificate provisioning costs but it also allows any server within the same organization to impersonate the DHCP server. This might appear a major shortcoming. It is, however, a necessary compromise between security and ease of deployment. The same kind compromise is already widely accepted in corporate wireless LAN authentication where any host with an SSL server certificate can pretend to be a RADIUS server in the EAP-TLS method. For organizations that require stronger network-name authentication, we suggest adding a new certificate field (X.509 extended key usage or application policy) to indicate that the certificate is specifically intended for network authentication (both DHCP-based and 802.1X with EAP-TLS). All clients should treat networks that specify the extended key usage as different from those that do not, even if the two networks otherwise appear the same (i.e., have the same root CA and name). This prevents the use of generic server certificates for impersonating networks that use the special certificates. Moreover, high-security clients may be configured not to use names from generic server certificates to access security policies that are specified in terms of globally unique names. This still allows the client to use such certificates for network aggregation, similar to certificates issued by an unknown root CA (see section IV.C).

Finally, our protocol depends on the security of the DHCP server. In home networks, the server is usually located in the gateway device, which combines the functionality of a NAT, router and firewall. These devices are notoriously vulnerable to compromise because they are typically purchased based on price rather than security; are left with the default configuration; and there is currently no effective way of deploying security patches to their firmware. This may enable an attacker to take control of the DHCP server either by connecting from the local network using a default password or by exploiting a software vulnerability remotely from the Internet. Thus, we have to assume that, in many networks without professional administration, the DHCP server will be compromised. The consequences to the secure NLA protocol are, however, mitigated by the fact that if the attacker already has a permanent presence at the home network of the mobile client, the attacker gains little from being able to spoof that location to the same client in other access networks. It would be simpler for the attacker to mount any attacks when the mobile computer is at home.

VIII. Related work

There is a standard for authenticated DHCP [DA01]. Currently, the only authentication method is a symmetric MAC with a preshared key, which is practical only for small, closed organizations. The specification could be extended to support public-key signatures. RSA signatures and certificates do not, however, fit into the authentication option and we found it simpler to send them in a long DHCP vendor option.

Wireless LAN security protocols [EA03] may seem like an alternative to DHCP authentication. They, too, have the limitation of requiring a pre-established relation between the client and the network. Public wireless access networks are typically unauthenticated or use the universal access method (UAM). UAM hijacks HTTP requests and redirects the browser onto a secure web page, whose certificate the mobile user could, in theory, inspect for network identification. The user involvement makes UAM unsuitable for universal network authentication.

Location awareness and the related privacy discussions usually refer to geographic locations, such as GPS coordinates or place names (see, e.g., [Leo98]). There have some been attempts at securing the location services, e.g., [DM96][Kuh04]. Even when the location information is obtained from communications networks, as it is for example in the E911 and GSM location services, the ultimate aim is to obtain physical coordinates. Yet, to computers, the logical network location is far more important. Operating systems for mobile computers naturally try to adapt to the network conditions and increase usability by remembering settings for each network. Windows NLA is only the latest step in this direction. A key development is the realization that an IP

address or an address prefix does not represent a unique logical location on the network and more complex heuristics are needed that combine several observed identifiers to a unique network identity. Our protocol removes much of the inherent uncertainly of such heuristics by allowing the access network itself to determine the right granularity of location information for that particular network and by using public keys as authenticated network identifiers.

Any protocol that tries to determine the relative location of itself and another entity by sending messages between them is vulnerable to tunneling or "wormhole" attacks [HPJ03] where the attacker relays messages between the client and a distant location. Some applications, such as door keys and other physical access-control tokens, are particularly vulnerable to tunneling attacks. The only secure mechanism for detecting such attacks is to measure the roundtrip time between the devices, but the implementation will require careful consideration of the physical signals and communications hardware [HK05]. Our protocol explicitly ignores the tunneling attacks in order to enable universal deployment to IP networks without significant new hardware requirements.

Literature on location privacy is mainly concerned with hiding the location of the mobile host from the peers with which it communicates. For example, onion routing [Cha81][SGR97] can prevent the server from observing the client location and vice versa. Since IP addresses are both locations and identifiers, this is usually stated in terms of anonymity: the communicating peers cannot find out each other's real identity. The location-privacy requirements of our protocol are somewhat different: we want the mobile host and user to remain anonymous towards the access network. Clearly, the access network knows where the mobile is. Our protocol does not help it to find the mobile's name or affiliation or to correlate multiple appearances of the same mobile at the same or different access networks.

The authentication of uncertified networks in our protocol is similar to the leap-of-faith authentication used in the secure shell (SSH) [Ylo96] or opportunistic IPsec [CM02], where the client learns the server's public key when connecting to it for the first time and, thereafter, associates that public key with the server name. There is a subtle difference, though, to the advantage of our protocol. The SSH client receives the server name from the user and tries to associate it with one correct computer and public key. Our DHCP client, on the other hand, encounters a network and a public key and decides to call them with the name suggested by the DHCP server. Only after that, the name is presented to the user. Thus, the client computer, not the user, is the authority on network naming, meaning that the computer cannot really be wrong no matter what names it decides to use for a network. The blind leap of faith happens only if the user or an application decides to use the locally assigned name to apply a security policy that refers to globally unique names. As long as the network names are used in the internal world of the client computer and its user, for example, to recall security settings saved on a previous visit to the same access network, it doesn't matter very much what the actual values of the names are.

With the emergence of various cryptographically authenticable identifiers, such as the cryptographically generated IPv6 addresses [Aur03] and HIP identifiers [NYW03], which do not require certification, the idea of opportunistic security associations is moving towards mainstream networking applications. One example of opportunistic authentication in network attachment is the quick network access protocol (Quick NAP) [AET+06], which is a clean-slate redesign of the current complex network attachment procedure for wireless networks. Although focused on access control and authorization of clients for network access, the protocol allows ad-hoc relations between a client and an access network, which are identified by the hashes of their public keys. Our protocol continues this trend towards using public keys as identifiers and enabling security without certified names wherever possible.

IX. Conclusion

In this paper, we describe an authentication protocol for DHCP servers. It is intended for securing network-location information distributed by the server. The protocol allows both PKI-based server authentication and secure identification of previously visited networks that have no certificates. We are careful to protect client anonymity at every step of the process. The protocol uses standard DHCP extension mechanisms. In addition to the authentication protocol itself, major contributions of the paper are a careful requirements analysis, the idea of recognizing network locations based on their public key, the use of a local PKI to aggregate multiple access networks into a single logical identity, and a novel way of binding network names to X.509-certified DHCP servers.

References

[AET+06] Jari Arkko, Pasi Eronen, Hannes Tschofenig, Seppo Heikkinen, and Anand Prasad. Quick NAP--secure and efficient network access protocol. In Proc. 6th International Workshop on Applications and Services in Wireless Networks (ASWN 2006), pages 163-170, Berlin, Germany, May 2006.

[Aur03] Tuomas Aura. Cryptographically generated addresses (CGA). In Proc. 6th Information Security Conference (ISC'03), volume 2851 of LNCS, pages 29-43, Bristol, UK, October 2003. Springer.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 24(2):84-88, February 1981.

[CM02] Claude Castelluccia and Gabriel Montenegro. IPv6 opportunistic encryption. Technical Report 4568, INRIA, October 2002.

[DM96] Dorothy E. Denning and Peter F. MacDoran. Location-based authentication: Grounding cyberspace for better security. Elsevier Computer Fraud & Security, pages 12-16, February 1996.

[DY83] D. Dolev and A. Yao. On the security of public-key protocols. Communications of the ACM, 29(8):198-208, August 1983.

[Dro97] Ralph Droms. Dynamic host configuration protocol. RFC 2131, IETF, March 1997.

[DA01] Ralph Droms and Bill Arbaugh. Authentication for DHCP messages. RFC 3118, IETF, June 2001.

DBV+03] Ralph Droms, Jim Bound, Bernie Volz, Ted Lemon, Charles E. Perkins, and Mike Carney. Dynamic host configuration protocol for IPv6 (DHCPv6). RFC 3315, IETF, July 2003.

[EA03] Jon Edney and William A. Arbaugh. Real 802.11 Security: Wi-Fi Protected Access and 802.11i. Addison-Wesley, 2003.

[EP02] Alberto Escudero-Pasqual. Privacy in the next generation Internet: data protection in the context of the European Union. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, December 2002.

[HK05] Gerhard P. Hancke and Markus G. Kuhn. An RFID distance bounding protocol. In Proc. IEEE SecureComm 2005, Athens, Greece, September 2005. IEEE Communications Society.

[HPJ03] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet leashes: A defense against wormhole attacks in wireless networks. In Proc. IEEE Infocomm 2003, April 2003.

[ITU97] International Telecommunication Union. ITU-T recommendation X.509 (1997 E): Information technology—open systems interconnection—the directory: Authentication framework, June 1997.

[Kuh04] Markus G. Kuhn. An asymmetric security mechanism for navigation signals. In Proc. 6th International Workshop on Information Hiding, volume 3200 of LNCS, pages 239-252, Toronto, Canada, May 2004. Springer.

[Leo98] Ulf Leonhardt. Supporting Location-Awareness in Open Distributed Systems. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, May 1998.

[LT06] Janne Lindqvist and Laura Takkinen. Privacy management for secure mobility. In Proc. Workshop on Privacy in the Electronic Society (WPES 2006), Alexandria, VA, USA, October 2006.

[ND01] Thomas Narten and Richard Draves. Privacy extensions for stateless address autoconfiguration in IPv6. RFC 3041, IETF Network Working Group, January 2001.

[NYW03] Pekka Nikander, Jukka Ylitalo, and Jorma Wall. Integrating security, mobility, and multi-homing in a HIP way. In Proc. Network and Distributed Systems Security Symposium (NDSS'03), pages 87-99, San Diego, CA USA, February 2003.

[SGR97] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In Proc. 1997 IEEE Symposium on Security and Privacy, pages 44-54, Oakland, CA USA, May 1997. IEEE Computer Society Press.

[Ylo96] Tatu Ylönen. SSH—secure login connections over the Internet. In Proc. 6th USENIX Security Symposium, pages 37–42, San Jose, CA USA, July 1996. USENIX Association.