Chattering Laptops

Tuomas Aura[1], Janne Lindqvist[2], Michael Roe[1], Anish Mohammed[3]

[1] Microsoft Research, Cambridge, UK
[2] Helsinki University of Technology, Finland
[3] Royal Holloway, University of London, UK

Abstract. Mobile computer users often have a false sense of anonymity when they connect to the Internet at cafes, hotels, airports or other public places. In this paper, we analyze information leaked by mobile computers to the local access link when they are outside their home domain. While most application data can be encrypted, there is no similar protection for signaling messages in the lower layers of the protocol stack. We found that all layers of the protocol stack leak various plaintext identifiers of the user, the computer and their affiliations to the local link, which a casual attacker can observe. This violates the user's sense of privacy and may make the user or computer vulnerable to further attacks. It is, however, not possible to disable the offending protocols because many of them are critical to the mobile user experience. We argue that the most promising solutions to the information leaks are to filter outbound data, in particular name resolution requests, and to disable unnecessary service discovery depending on the network location. This is because most information leaks result from failed attempts by roaming computers to connect to services that are not available in the current access network.

Key words: Privacy, anonymity, mobile computing, wireless networks, network location awareness

1 Introduction

When mobile computer users connect to the Internet at wireless hotspots, cafes, hotel rooms, airport lounges and other public places, they tend to think that nobody can recognize them. Some are aware that sophisticated techniques, such as correlating the appearances of the network interface card's MAC address or other statistically unique information, could be used to trace them. Few know that their computer is openly broadcasting information about them to the local network, including usernames, computer names, and identifiers linkable to their employer, school or home. In this paper we explore the identifiers leaked by mobile computers to the access network.

Our attacker is a passive observer at the same local link, who has no resources for making global observations or skills for sophisticated analysis of the data but who is curious enough to capture network traffic and to see what other network users explicitly tell about themselves. The attacker may be operating the local network or access point, or he may be just another user in the same network.

The user's own computer is not malicious but leaks information accidentally or because of conflicting design goals. We focus on business users whose computers are members of a managed domain.

There is no great drama in being identified in a public place. Most people, however, prefer not to wear a name tag after leaving the office and enjoy the privacy and protection afforded by the relative anonymity. Sometimes, announcing a person's name or affiliation could expose them to further attacks. This vulnerability also applies to computers: a random computer at a cafe is not particularly interesting to a hacker but one belonging to a well-known organization might invite attacks.

Computers perform many tasks automatically without the user's knowing; things just work. The automatic tasks often involve the discovery of network services, which means sending packets to the network. These packets usually identify the service and often also the user. It should be noted that the automatic actions happen by design: most users would probably not want to see any additional dialog windows asking for their permission to go ahead, and disabling the automatic services would destroy the seamless mobility experience that software vendors are hard trying to create.

In this paper, we are mainly interested in identifiers in signaling protocols, packet headers and communication metadata, that is, data that cannot be easily encrypted at the application level. It is often falsely assumed that end-to-end encryption solves all privacy issues apart from traffic analysis. In real networks, not all communication is end-to-end. There are many protocols that are executed with the access network and with global network infrastructure. For example, the DHCP and DNS protocols cannot be protected by encryption. Yet, these protocols reveal all kinds of information about the mobile host. Much work has been done on randomizing the most obvious permanent identifiers (MAC and IP addresses), and, on the attack side, on fingerprinting mobile hosts based on statistical characteristics. In this paper, we consider more explicit user, computer and organization identifiers such as usernames. Clearly, randomized addresses only help privacy if the higher-layer identifier leaks are also controlled, and the statistical attacks matter only if there is no easier way to identify the target.

We use domain-joined Windows XP and Vista laptops as examples throughout the paper because they are common in business use and perform many tasks automatically. Domain members have more identifiers and credentials than typical standalone computers and they tend to access more services. Thus, there is more information that could potentially be leaked.

This paper makes the following contributions: we identify network chatter by mobile computers as a major threat to mobile user privacy, develop a tool for detecting identifier leaks, and use it to examine network traces captured from business laptops. We analyze the causes of the leaks and describe a solution based on network-location awareness. The lessons of this paper could be summarized by saying that using a laptop computer is akin to wearing a name badge that reveals the person's identity and affiliation, and that not telling everyone who you are turns out to be surprisingly hard because there are so many name badges in places that you never knew about. We argue that most of the leaks are caused by unnecessary network chatter, mainly failed attempts at name resolution and server connections, which could be avoided by designing software to be aware of the network locations.

The rest of the paper is organized as follows. We overview related work in Section 2. Section 3 introduces the analysis tools. Section 4 details the sources of identifier leaks. In Section 5, we analyze the findings. Section 6 suggests solutions to the problem and Section 7 concludes the paper.

2 Related work

Information leaks from mobile computers

Information leaks caused by unencrypted network traffic have been noted many times in the literature. There are few systematic studies, however. Kowitz and Cranor [KC05] study how user attitudes change when they are explicitly shown plaintext strings from the network traffic. The strings are mostly application data such as email, instant messages and web searches, but the paper also mentions NetBIOS as one source of information. We see the information leaks as a technical problem rather than as a question of user awareness.

Saponas et al. [SLH07] bring attention to ubiquitous computing devices which can be traced by their unique identifiers or reveal which content the user is downloading. Akritidis et al. [ACL+07] mention RSS subscriptions, plaintext instant messaging, web-browser cookies, and the hostname in the DHCP request (see Section 4.3) as means for identifying mobile users. Pang et al. [PGM+07] suggest confidential discovery of wireless access points.

DNS was originally designed for fixed networks but is increasingly used as a reachability mechanism for roaming hosts. Guha and Francis [GF07] point out that dynamic DNS can be used to query and map a mobile host's location. Broido et al. [BSF06] discuss unnecessary DNS updates for private address ranges, which may also leak information about the host to the foreign access network. In this paper, we discuss more basic operations of DNS and observe that most privacy-compromising data is revealed unnecessarily.

Anonymity and routing

Anonymity in communications networks has many meanings. Traditionally, the main goal has been end-to-end anonymity, i.e., to hide the client's identity from the servers or peers to which it connects over the Internet. Anonymous routing systems, such as the mix networks introduced by Chaum [Cha81], hide the connection between senders and recipients of messages also from third parties who are assumed to monitor network traffic globally. Onion routing, as described by Syverson et al. [SGR97] extends the idea to hidden servers, i.e., to hiding the recipient from the sender. These mechanisms assume a very strong attacker model and are expensive to implement, yet tend to be fragile against analysis methods that take advantage of the non-ideal characteristics of the underlying technologies. The most common applications for anonymous routing are in content distribution (e.g., Freenet by Clarke et al. [CSWH00]) and anonymous web browsing and censorship resistance (e.g., Tor by Dingledine et al. [DMS04]), where there is a strong incentive for hiding the identities of the communicating parties. Application-specific anonymity systems include remailers and anonymizing web proxies (e.g., Mixmaster by Möller et al. [MCPS03] and Crowds by Reiter and Rubin [RR98]). Simple HTTP proxies and native address translation (NAT) also provide some privacy benefits. The same routing mechanisms can be used for location privacy, i.e., to hide a mobile computer's location from its peers. Mobility protocols, like Mobile IPv6 [JP02], achieve some level of location privacy by routing all packets to and from the mobile via a fixed proxy.

Despite the number and diversity of end-to-end anonymity mechanisms, they share the common goal of hiding the mobile's identity or location from peer nodes over the Internet. Our work differs from this in that we want to protect against observers at the mobile's local link. Our attacker model is also different in the sense that the attacker is assumed to be present only at the access network.

Randomized identifiers

Communications protocols use various kinds of identifiers and addresses that can act as identifiers. For example, the MAC address of a network interface is a globally unique identifier. IPv6 addresses often have the MAC address embedded in their bits in order to guarantee uniqueness [TN98]. A common solution to the issues caused by unique identifiers is to replace them with random, periodically changing values. There is a standard way of generating IPv6 addresses with a pseudo-random number generator [ND01]. Similar randomization has been suggested for the MAC address by Gruteser and Grunwald [GG03b] and many others. The identifier changes have to be carefully timed, with possible silent periods, to maximize anonymity protection and to minimize disruption to communications, which is also noted by Beresford and Stajano [BS03] and Jiang et al. [JWH07]. Clearly, unencrypted higher-level identifiers such as IP addresses have to be changed at the same time as the MAC address. Mobility protocols can be used to guarantee continuity of end-to-end communications over the identifier changes, e.g., as suggested by Lindqvist and Takkinen [LT06]. Since the issues with IP and MAC addresses have already been extensively covered in the literature, we will focus on other identifiers.

The level of anonymity provided by such mechanisms can be measured as the size of the anonymity set or as entropy (see Sweeney [Swe02], Serjantov and Danezis [SD02] or Díaz et al. [DSCP02]), both of which measure the level of uncertainly about the identity of a node. Given the number of users and mobile devices on the Internet, the hope is that the uncertainly will be very large. The academic literature has concentrated on theoretically strong or at least measurable guarantees of anonymity and location privacy. The work presented in this paper differs from the literature in that we consider a rather more elementary goal: not explicitly telling everyone who you are, which turns out to be surprisingly hard.

Host fingerprinting

Fingerprinting of mobile radios based on their non-ideal characteristics is an old military intelligence technique which enables tracing the movements of individual stations. The same kind of analysis has been applied to wireless LAN cards, e.g., by Gerdes et al. [GDMR06]. The analysis of radio signals requires sophisticated hardware and skilled operators, however. A more practical approach is to fingerprint hosts based on their higher-level characteristics such as the MAC-layer capabilities and configuration, which can be combined with network-layer traffic-analysis data for better accuracy (Franklin et al. [FMT06], Greenstein et al. [GGP+07] and Pang et al. [PGG+07]). Some hardware characteristics, such as clock skew and temperature variations (Kohno et al. [KBC05] and Murdoch [Mur06]) can be used to fingerprint hardware remotely. The same techniques could be used to identify devices on the local link. In effect, the hardware and communications fingerprint becomes a unique identifier for the device and user.

Our work differs from the device fingerprinting in that we concentrate on explicit identifiers instead of implicit ones. For example Kohno at al. use the set of peer IP addresses as an implicit identifier that is treated as a set of numbers. We, instead, record the DNS names to which the host connects and look for ones that reveal the client identity or affiliation.

Information flow

One approach to preventing information leaks is to analyze information flow in the system. In the terminology of multi-level security, the user identifiers are high input data, from which information should not leak to the low output, i.e., messages sent to the network. By proving information-flow properties, such as non-interference [GM82], we could be certain that the system does not leak the high data. While such models remain theoretical, there has been progress, e.g., in the language-based proof techniques of Sabelfeld and Myers [SM03].

On the more practical side, we can trace the information flow dynamically in a running system. Most such mechanisms aim to protect system integrity, rather than confidentiality of data. In the Perl programming language, untrusted inputs can be marked as tainted and the tainting is propagated to any values derived from them. Chow et al. [CPG04] use data tainting in a simulated system to analyze the lifetime of confidential data, such as passwords, in the system memory while Zhao et al. [ZCYH05] show evidence that taint propagation can be traced in real time in a production system. Yumerefendi et al. [YMC07] suggest a clever way of detecting data leaks by executing a parallel copy of the process with random bits replacing the confidential data; if the outputs differ, some information is leaking. The same techniques could be used to flag identifiers and other anonymity-compromising data and to detect whether they are being sent to the network. We consider data tainting a potentially useful approach; however, the solutions suggested in this paper are even more practical in nature.

Privacy policy and preferences

Another approach to privacy is not to discuss the technology but the policies. In addition to legal frameworks, there are several technical policy frame-

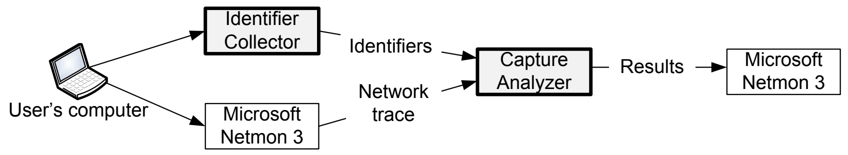

Fig. 1. Data flow between the analysis tool components

works for location privacy [Zug03][CJBMM04][GG03a][Pet02] and languages for expressing preferences on the disclosure of personally identifiable information [Cra02][AHK03]. We do not explicitly discuss privacy policies or user preferences in this paper. In Section 6, however, we suggest an implicit privacy preference mechanism that interprets any networking functionally explicitly enabled by the user as a policy decision.

3 Tool for analyzing network chatter

We initially became concerned over information leaks to the access network when looking at network traces. As we started to comb through them for previously unknown issues, it soon became apparent that a more systematic approach was needed. For this reason, we developed a tool for detecting leaked identifiers in network traces. The tool is defensive in the sense that it can only be used to analyze leaks from one's own computer. This limitation allows us to find offending user, machine and organization identifiers in places that have not been previously catalogued. We have previously used a similar tool to detect user identifiers in electronic documents [AKR06].

The general structure of the tool is shown in Figure 1. It consists of two modules: Identifier Collector and Capture Analyzer. The Netmon network monitor is used for recording network traffic and for viewing the discovered information leaks in their context.

3.1 Collecting personal identifiers

The Identifier Collector gathers the user's personal identifiers, which will then be used as search strings in the capture file analysis. It finds identifiers from the local computer and from the active directory (AD), which is a directory service for Windows computers. The identifiers include the username, machine name, NetBIOS group, domain name, globally unique identifiers (GUIDs), names of various domain-specific services, as well as less obvious identifiers such as postal address and telephone number. An alternative would be to let the user type in the sensitive identifiers but we wanted the tool to be as automatic as possible. In addition to improving usability for non-expert users, automation makes the results reproducible.

3.2 Capture-file analysis

The Capture Analyzer module searches for the identifiers in a network capture file. We use two different search algorithms for this purpose. The main difficulties were that the format of the captured packets is complex, variable and sometimes unknown, and that we would like to detect information leaks in any protocol layer or data field.

Data formats

One approach to the search would be to parse the packets in the same way as in network monitoring software such as Ethereal, Wireshark or Netmon and then search each data field separately, taking into account its data format, for the offending identifiers. A limitation of this approach is that we might miss some data fields that are not correctly identified by the parser. For this reason, we decided to search through the raw packet data using algorithms that can handle a large number of data and text encodings. We parse the packets and pinpoint any suspicious protocol fields only after detecting identifiers in the raw packet data. The tool offers two different tradeoffs between speed and completeness of the search.

Simple string search

The Aho-Corasick algorithm [Aho75] performs a fast text search for multiple search strings. We encode textual search strings, such as a username, with a number of common string encodings: ASCII, Unicode UTF8, UTF16 in big- and little-endian byte order, and UTF32. The search is case-insensitive and ignores accents and common character variations. Thus, for example, the character values aàáâãäåAÀÁÂÃÄÅ are all considered matches for each other. Short strings are also encoded as NetBIOS identifiers, which have their own peculiar format. Additionally, we look for copies of the MAC address outside the Ethernet header and for Windows GUIDs of the user, computer and domain. Binary identifiers are treated as special cases based on their specific characteristics. The number of encodings has been tuned to keep the tool speed acceptable for fast interactive use.

Regular expression search

The second string search algorithm aims to perform a more complete search than the simple string search above, which was optimized for speed and not assurance. With a slower search algorithm we wanted to detect any identifiers that may be missed by the simple string search. Regular expressions provide flexibility to support multiple layers of data encoding. We start with a simple tree-shaped expression constructed from the original search strings and expand this by replacing each character in the expression with its different encodings. This is done recursively for multiple layers of encodings: upper and cases; accents and other character variations; URL, XML and C escapes and numeric representations; and, finally, Unicode and other character encodings. The resulting regular expression is large but represents an even larger number of multi-layer encodings of the search strings. Figure 2 shows a simplified example of how the original character: N

upper and lower case, accents: (n|ñ|N|Ñ)

various escape notations:

```
(n|(((\x?)|%|(&#?)| |-)0*((156)|(110)|(6e));?)|ñ|(((\x?)|%|(&#?)| |-)0*((361)|(241)|(f1));?)|N|(((\x?)|%|(&#?)| |-
)0*((116)|(78)|(4e));?)|Ñ|(((\x?)|%|(&#?)| |-)0*((321)|(209)|(d1));?))
```

ASCII, UTF-8, little and big-Endian UTF-16:

```
((00*6e)|((((00*78)?00*5c)|(00*25)|((00*23)?00*26)|(00*20)|(00*2d))(00*30)*((00*3100*3500*36)|(00*3100*3100*
30)|(00*3600*65))(00*3b)?)|(00*(3f|(c3b1)|f1))|((((00*78)?00*5c)|(00*25)|((00*23)?00*26)|(00*20)|(00*2d))(00*30)
*((00*3300*3600*31)|(00*3200*3400*31)|(00*6600*31))(00*3b)?)|(00*4e)|((((00*78)?00*5c)|(00*25)|((00*23)?00*
26)|(00*20)|(00*2d))(00*30)*((00*3100*3100*36)|(00*3700*38)|(00*3400*65))(00*3b)?)|(00*(3f|(c391)|d1))|((((00*
78)?00*5c)|(00*25)|((00*23)?00*26)|(00*20)|(00*2d))(00*30)*((00*3300*3200*31)|(00*3200*3000*39)|(00*6400*31))
(00*3b)?))
```

Fig. 2. Regular expressions for encodings of 'N'

regular expression for one character of a search string is constructed. Because of the expression size, the search is done with a non-deterministic automaton, which means it consumes a lot of memory. We tuned the number of encoding layers and their complexity to keep the memory consumption for even a large set of identifiers below 1GB. The aim was to keep the search time under an hour for large datasets.

3.3 Integration with Netmon 3

The Capture Analyzer works together with the Microsoft Netmon 3 network-monitoring software. It takes as input a network capture file and a list of identifiers and produces a Netmon filter that lists the matching packets. This is loaded into Netmon for detailed manual analysis of the information leaks in the capture. Currently, the tool can only be used for offline analysis.

3.4 Discussion of completeness

It is rather difficult to assess the completeness of the search. We have enhanced the tool to detect all classes of identifier leaks that we initially knew about, suspected, or found by manual methods. The tool has some specific limitations, however. It cannot find intentionally obfuscated information, which falls outside our attacker model, and it cannot search encrypted data. Currently, we do not search through compressed data or other encodings that do not respect byte boundaries (e.g., Base64 and uuencode). Fortunately, such encodings are rarely used in signaling messages below the application layer.

Since we look at the capture files as raw bytes and do not parse the packets, we cannot detect data that spans across multiple packets. The most likely reason for this to occur is when an identifier has been split into two TCP segments and, thus, is non-contiguous in the packet capture data. In the future, we may enhance the tool to support TCP segmentation. Fragmented IP datagrams could pose a similar problem. We initially planned to implement defragmentation at the IP layer but failed to do this because the sample data did not contain any fragmented packets.

Some false positives are produced by the case- and accent-insensitive search, which means that 3- or 4-character names have some accidental matches in binary files (e.g., “tÏÑå” would match “Tina”. The number of false positives was acceptably small for the purposes of research. If the same algorithm is used for routine monitoring, it would be easy filter recurring false positives. Another class of false positives arises if the user is affiliated with an organization whose name occurs frequently in network traffic (e.g., Google) or is a common word (e.g., Time).

Active attacks are entirely beyond the scope of this paper. They are, however, not as difficult to implement as one might first believe. In fact, it may be easier for the attacker to induce the mobile computer into executing a specific protocol, such as IKE or DHCP, than to sit passively on the network and wait for events to occur spontaneously. We plan to continue this work in the direction of active-attack analysis.

4 Information leaks to local link

This section reports findings from the analysis of network traces collected at various locations using domain-joined computers running Windows XP and Vista and a range of client software that is commonly found on business laptops. The analysis was done with the search tools described in the previous section.

4.1 DNS

The domain name system (DNS) is a directory service that resolves human-readable host names into IP addresses. The literature (see Section 2) already considers the privacy issues created by dynamic updates to the DNS. We look at a more basic operation: name resolution.

DNS queries

Connecting to online services may reveal information about the client. We will discuss examples of such services in the following sections. However, before connecting to almost any service, the client will resolve the DNS name of the server using the local DNS in the access network. Consequently, the easiest way to track the activities of a mobile computer is to record its DNS requests. For example, if the computer connects to a VPN gateway of its organization (e.g., vpn-gw.contoso.com), a look at a DNS log is sufficient to identify the company. The user may not be aware that many such queries happen automatically, without an explicit user action.

Resolving private names

Many organizations use a private IP address range (e.g., 10.0.0.0/8) for their internal network and a private DNS zone (e.g., *.private.contoso.com or *.contoso.local) to name the computers on the private network. The private names can only be resolved by the local DNS server at the intranet and are not visible from outside. DNS resolvers on client computers do not, however, know when they are in the intranet. Thus, a mobile computer may try to resolve a private name when it is roaming outside the private network. The name resolution will fail but the DNS request reveals the name of a server and organization.

Default suffixes

Since human users prefer to type short computer names (e.g., hobbit) rather than fully qualified domain names (FQDN) (e.g., hobbit.sales. contoso.com.), the resolver automatically appends default suffixes to the name. For example, when resolving hobbit, the computer typically queries for hobbit.sales.contoso.com and hobbit.contoso.com, in that order. Originally, there was a security reason for trying the longer name and, thus, the more local name space first: it prevented users from accidentally entering their password to a prompt presented by a more remote host than the one they intended to access.

The default DNS suffix for stationary computers used to be configured either manually or by DHCP. Mobile computers may have two possible suffixes: a primary suffix from their home domain and a connection-specific suffix obtained from DHCP at the access network. We are concerned about the primary suffix because it reveals the mobile host's affiliation. When the computer tries to resolve any DNS name, such as google.com, it will start by querying for google.com.sales.contoso.com. This means that any DNS query will leak the mobile's default domain suffix to the access link and to the local DNS server.

4.2 Other name resolution protocols

NetBIOS over TCP (NBT) provides another name service, which is mainly used in closed Windows domains (or workgroups) at workplaces and homes. Computers broadcast name queries to the local network and answer them directly or via WINS proxy.

NetBIOS lookup

Similar to DNS, NetBIOS name lookups reveal to the access network the names of the services to which the user or computer is connecting. The names are broadcast to the local link and, thus, can be heard by any computer on the same access link, even on a switched wire network. Unlike the hierarchical DNS names, NetBIOS names are not globally unique. For this reason, the protocol is rarely needed when roaming outside the user's workplace or home, yet it is typically enabled everywhere.

WINS registration

When a computer connects to a network, it may also try to register its NetBIOS name and group in the WINS server. Since it does not know whether a server exists in this network, the registration is attempted regardless of the location. The registration attempt reveals the computer name (e.g., hobbit), which is typically the same as the first part of the FQDN, and the computer's Windows domain or workgroup (e.g., sales or contoso). Again, this information is broadcast to the access link.

LLMNR

The link-local multicast name resolution (LLMNR) protocol is also intended for the local link. Unlike NetBIOS, it also works over IPv6. The queries are sent as link-scope multicast. Although we have not observed this protocol in actual use, Windows Vista laptops sometimes send spontaneous LLMNR requests for their own name in order to detect possible name conflicts.

4.3 DHCP

The dynamic host configuration protocol (DHCP) is used to configure a host with network-specific parameters such as an IP address and the local DNS suffix. It is often the first protocol executed when a computer attaches to a network. A typical execution consists of two request-response pairs: the client broadcasts a DISCOVER message and receives one or more OFFERs from servers. It then sends a REQUEST for one of the offers and the chosen server responds with an ACK. The main purpose of the protocol is to transfer configuration information from the server to the client, i.e., to the mobile host. Thus, the client does not necessarily need to reveal anything about itself. In practice, however, clients do tell quite a lot.

Host identification

The DHCP protocol allows the client to identify itself by sending its hostname in the DISCOVER message. This enables the network to select host-specific parameters such as a permanently assigned IP address. Hostnames are unique only to a specific domain and, thus, have little significance to a DHCP server at a foreign network. In principle, the identifier could be simply left out while roaming. Unfortunately, when the client sends the DISCOVER message, it may not yet know whether it is connected to the domain network or roaming elsewhere.

DNS registration

The DHCP client may want to register its new address in the DNS. The client itself can connect to the a dynamic DNS server at its home organization to update the forward record, i.e., the mapping from name to IP address. The DHCP server, on the other hand, is responsible for updating the backward record from the newly allocated IP address to the DNS name. Within a Windows domain, the DHCP server may also update the forward record on the client's behalf. Either way, the DHCP server needs to know the client DNS name. For this reason, a Windows client sends its FQDN to the DHCP server in the REQUEST message. It does this regardless of the network location and, thus, reveals the host name and domain suffix to the access network.

4.4 Domain controller

The domain controller (DC) is the authentication and directory server for a particular Windows domain. It implements a version of the LDAP directory-access protocol. When a client is configured to be a member of a domain, it always tries to find the domain controller of the network. It performs a DNS query for a service resource to find the domain controller, e.g., LDAP._TCP.dc._msdcs.sales.contoso.com. If the DC is found, the client knows it is on the intranet and starts sending LDAP queries to the controller. On the other hand, if the DC is not found, the client (Netlogon service) may try to use a cached IP address to send the queries. For stationary computers and ones that move in the intranet, this improves reliability in case of DNS failures. For roaming computers, however, the attempts to connect to the DC will fail anyway.

Both the domain name and the cached IP address will reveal the mobile's affiliation. The IP address will be cached only for 15 minutes. We found, however, that if the computer was put into a sleep-saving mode at the intranet and resumed later in a foreign network, the cached addresses were still used for several minutes.

4.5 File shares and printers

Operating systems such as Windows try to improve the user's roaming experience by discovering previously used network services and setting them up for quick access. This can, however, result in unnecessary network chatter and failed connection attempts.

Mounted network drives

The user in Windows can assign a drive letter to a network share so that it appears as a local disk (e.g., map \\contoso-srv-2\alice\ as the Z: drive). These shares are automatically mounted when the user logs in or connects to a network. In order to find out whether the share is available, the client needs to probe the server. The attempt to resolve the server's DNS or NetBIOS name can be seen by anyone observing network traffic at the access link. Although we tested only Windows shares, automounted NFS volumes would presumably cause similar privacy issues.

Shortcuts to network shares

Shortcuts to network file shares on other machines may cause similar attempts at name resolution. These are usually accessed only after a user action but it is not always obvious to the user which actions trigger the network access. For example, right-clicking a shortcut may cause an attempt to connect to the server.

Printers

Windows saves information on all printers that have ever been configured for use, unless they are explicitly deleted. A roaming user may sometimes use a local network printer at the access network. These printers accumulate into the printer list on the client computer, which many users never clean. When the user views the list of printers, the computer automatically tries to connect to the printers and shows which ones are online. These connection attempts may reveal not only the user's organization but where the user has been roaming. For example, one of our test laptops readily revealed to the network that it had been printing in three cities on different continents.

4.6 IKE and Kerberos

IKE with GSSAPI authentication

One of the most surprising sources of information leaks is the Internet key exchange (IKE) protocol. IKE is designed to protect the participants' identities against sniffing (in the main mode, which is implemented by Windows). This protection is achieved by first performing an unauthenticated Diffie-Hellman key exchange and by encrypting the following authentication with the session key. The identity protection is considered one of the main security features of IKE.

The standard IKE authentication methods are based on shared keys and public-key certificates. Windows extends this with Kerberos authentication using the GSSAPI [PS01]. The client requests Kerberos authentication in the first message it sends to the server. The client then obtains a Kerberos ticket from the authentication center (AC) and uses this ticket for authentication in IKE.

The most obvious information leak happens because the GSSAPI authentication method sends the client computer name and domain to the server in the first IKE message (in the SA payload). This may not appear to be a privacy issue because Kerberos authentication is used only in the intranet and, thus, the data should never be sent when the computer is roaming in a foreign access network. In reality, the leak does sometimes occur when the client has just moved from the intranet to a foreign access network (usually via sleep mode) and applications still attempt connections to intranet servers based on previously resolved IP addresses.

Kerberos ticket request and ticket

Windows clients also sometimes attempt to connect the Kerberos server while roaming. The ticket request contains the client computer name in plaintext. Since this is a rare occurrence, we were not able to establish the exact cause of the request. Since the Kerberos server is usually not reachable from outside the intranet, the client will not receive a ticket. If it did, the ticket would further reveal the name of the server for which the ticket is intended.

4.7 TLS/SSL

Plaintext certificates

The TLS handshake protocol sends the certificates unencrypted over the network. Usually, only the server is authenticated and only the server certificate is sent. This means that if the client connects to a secure web server of its own organization, the name of the organization will appear on the wire. Sometimes, TLS is used also for client authentication. This may happen, for example, when the client application is not a web browser but a web-service client, an email client or a TLS-VPN client. In that case, the plaintext client certificate and name are seen on the network, which allows easy and reliable identification of the client.

EAP-TLS

Secure 802.11 wireless LANs do not leak much information to those who are not authorized to join the network. However, wireless networks in managed domains may use certificates and the EAP-TLS protocol for client authentication. The TLS handshake in EAP-TLS reveals the client identity to anyone listening, even to those who themselves are not authorized to access the network. This problem has been addressed by a recent privacy enhancement to the EAP-TLS protocol [SAH08].

4.8 Application metadata

It is clear that plaintext access to email, web pages, search engines and other online services leaks confidential data. For example, we found unencrypted instant messaging (IM) clients sending not only the messages themselves but also the username, real name, gender, birth date, post code, buddy list and block list over the network. Unencrypted SIP signaling for IM or VoIP also reveals the user name and possibly who his contacts are. All this information could, however, be protected from sniffing by encrypting the messages between the client and the server.

A particularly interesting case is the iTunes music-player software, which discovers other iTunes users nearby. It does this by broadcasting advertisements to the local link, which contain the username and computer name. (The protocol is Apple Bonjour, which is based a proposal for multicast DNS [CK06]). This allows the users to listen and purchase the same music. There is no obvious way to encrypt this communication as the aim is to communicate with new people without configuring a security association.

5 Discussion of the leaks

In the information leaks discovered above, essentially the same data is revealed again and again:

– user identifiers (username, GUID, email address, real name),
– computer name, and
– user affiliation (DNS suffix, domain or workgroup, servers accessed).

One way to understand the consequences of such data leaks is to compare carrying a mobile computer to wearing a name badge or an RFID tag that broadcasts the name and affiliation of the person carrying it. Although the user's real name is not sent to the network as frequently as other identifiers, an email address or username and domain are usually sufficient to discover the user's personal web page or other information about the user. Although being identified is not very dangerous in itself, it may expose the user to unwanted attention from other people, and it may put the computer to a higher risk of attacks by hackers on the same access link.

Broadcast links, such as wireless access points, are the most opportune places for the casual observer. Most public-access wireless links are unencrypted or use the same shared key between all stations. On a switched wire Ethernet, a casual observer can only see broadcast packets. These comprise mainly DHCP DISCOVER and REQUEST messages and NetBIOS name lookups and registrations. Together, these packets may reveal the computer name and the user's organization but usually not the username. A typical situation where one can see many such broadcast packets is a wire network at a hotel or airport lounge.

Few public wireless access points currently use link-layer encryption. Those that do will block out unauthorized users but only for the purpose of charging. They will still let in mutually distrusting users who have paid paid for the access. Most access points now support per-client encryption keys between the AP and the client. The resulting level of privacy is similar to switched wire Ethernet.

It may seem that preventing a computer from sending a name or an identifier to the network is a simple task. If it were a question of one identifier sent by one piece of software, this would indeed be the case. What makes the problem difficult is that there so many protocols, at all layers of the network stack, and so many applications are sending so many different identifiers at different times. Naturally, all these protocols serve some purpose and cannot be simply disabled without causing inconvenience to the user.

The current practice in software engineering is to build the protocols, applications and services to be independent of each other and let each one perform its own discovery procedure. As a result, there is no single product or manufacturer in control of all the data that is sent to the network by a mobile computer. In this sense, Internet-enabled appliances are in a more reasonable position to protect the user privacy while it is almost impossible to know or control what data is sent to the network from a fully-fledged computer.

Most of the information leaks occur because of failed service discovery attempts. That is, the roaming computer is trying to find and access network services that are not accessible on the foreign network or it is trying to execute protocols that are only used between computers that belong to the same domain. The computer does this because it doesn't know which network it is on or which services are available there. In most cases, the client receives a "non-existent domain" response to the DNS requests, no response to the NetBIOS lookups, or finds that the server IP address is unreachable. These failed queries and connection attempts constitute unnecessary network chatter. If the client had some way of knowing whether the service can be accessed from the current location, it would not need to send out any of those packets. That is the reasoning behind the solution we introduce in Section 6.

Some of the identifier leaks are caused by public Internet services such as instant messaging, VoIP and various toolbars. The obvious solution is to encrypt the data between the client and server in a way that protects the client identity; for example, authenticate the client inside an TLS/SSL tunnel or deploy IPsec in identity-protecting mode. The technology exists and its deployment is simply a business issue.

A slightly more subtle problem is created by services that are operated by the mobile computer's home organization and are accessible from the Internet. These include email servers, web-mail interfaces, and VPN gateways. Encryption hides the identity of the user and computer but cannot mask their affiliation with the server. The identity of the organization could be obscured a little by hard-wiring the server IP addresses to the client or by using nondescript DNS names. A more robust solution is an anonymous routing mechanism such as Tor (see Section 2) at the cost of relatively poor real-time performance.

6 Preventing unnecessary chatter

In this section, we describe a strategy for preventing the unnecessary network chatter. The basic idea is to identify the access networks and to attempt connection to a service only on the networks where the service exists.

Some laptops (e.g., those with Mac OS X) have for some time allowed the user to configure network profiles and select them manually. Windows Vista implements a network-location-awareness (NLA) service that identifies the access network automatically, without user interaction. Since this mechanism is not yet widely known, we explain it in some detail. NLA creates a fingerprint of the access network, which is a set of parameters associated with the network. NLA then computes a network identifier as a cryptographic hash of the fingerprint. Applications and operating-system components can query NLA for the network identifier and use it as a database key to store and retrieve any information related to networks. On the first visit to the network, the network identifier is just a random-looking number. On the following visits, it can be used to recognize the network. Windows Vista currently uses NLA to remember the choice of a firewall profile for each access network, so that the user is asked only once at each network (and not asked at all for the intranet).

The choice of parameters in the NLA fingerprint varies by network type; for the purposes of this paper, it suffices to think of the security profile for authenticated networks and the gateway MAC address for others. Although the gateway MAC address can be spoofed, the casual attackers considered in this paper would not know the address value. For this purpose, we have proposed an enhancement to the NLA mechanism would authenticate the DHCP server on the network and use the server public key as the network fingerprint [ARM07]. This kind of authentication would enable us to authenticate any previously visited network without a PKI, which is exactly what is needed for the chatter-limiting mechanisms explained below.

Given the NLA mechanism, the next step is to disable and enable service discovery protocols depending on the network location. For this purpose, we propose the following policy: *When client software stores information about an online service for the purpose of connecting to it later, it must also store the NLA network identifiers of the access links where the service is known to be accessible. Automatic connection attempts to the service are only allowed on those networks.*

In a sense, when the user or administrator decides to access a service on a network, he is making a policy decision to enable the same service always on the same network. The rule applies both to applications and operating-system components that act as service clients. Some default policies should also apply:

- The active directory and Kerberos server should only be accessed on the intranet.
- NetBIOS should disabled by default and enabled separately for each network if needed.
- The default DNS suffix should be disabled outside the domain network.
- Network file shares may be accessed automatically and printers probed for availability only in the network where they were originally configured, or if the user explicitly request connection on another network.
- IKE with GSSAPI authentication should only take place in the intranet.

As a result, there should be no failed attempts at name resolution or failed connections to servers when the computer is on the wrong access network. While we believe this is the right approach in the long term, it requires changes to all the different service clients and applications that send data to the network. It can be argued that this requires a culture change to the way network client software is designed, which we see as necessary.

Another way to control network chatter is to filter outbound traffic from the computer at a host firewall. This would enable us to implement immediately some of the policies mentioned above, such as disabling specific DNS or NetBIOS queries. Packet filtering is a temporary emergency measure, however, because it is typically done at very coarse granularity, such as disabling access to all files shares instead of access to specific ones. The practicality of deep packet inspection for this purpose remains to be tested as there are potential issues with the firewall performance. Another problem with using firewalls is that dropping packets may cause unpredictable failure of applications.

Finally, some of the information leaks described in section 4 are transitory in the sense that they occur only when the mobile computer has just moved from the intranet or home to a public access point. This happens because software caches state data, such as IP addresses, and uses them even after moving to a new network. These problems can be solved by detecting when the mobile computer has disconnected from a network and by discarding any state data that may be stale after the event. The same should be done after the computer has been in a sleep mode and possibly moved to a new location.

7 Conclusion

In this paper, we analyzed identifier leaks from mobile computers to the access link. We discovered that the username, computer name and organizational identifiers such as the domain suffix are sent unencrypted to the network by a large number of different protocols and applications. This is a breach of privacy and may expose users and their computers to unnecessary risks or embarrassment.

The privacy concerns could discourage people from using new communications technology to its full potential. We suggest a solution based on network location awareness (NLA). Client software should remember the networks on which it has been configured to access each service. It should not try to automatically discover the service at other locations. This solution requires changes both to application clients and to many parts of the network stack; in effect, we are proposing a change of culture in the way service discovery in network network-enabled software is implemented.

References

[ACL+07] P. Akritidis, W.Y. Chin, V.T. Lam, S. Sidiroglou, and K.G. Anagnostakis. Proximity breeds danger: Emerging threats in metro-area wireless networks. In *Proceedings of 16th USENIX Security Symposium*, Boston, MA, USA, August 2007. USENIX Association.

[Aho75] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: an aid to bibliographic search. Communications of the ACM, 18(6):333–340, June 1975.

[AHK03] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. Enterprise privacy authorization language (EPAL 1.2). Research Report RZ 3485, IBM, March 2003.

[AKR06] Tuomas Aura, Thomas A. Kuhn, and Michael Roe. Scanning electronic documents for personally identifiable information. In *Proceedings of 5th ACM Workshop on Privacy in the Electronic Society (WPES'06)*, Alexandria, VA, USA, October 2006. ACM.

[ARM07] Tuomas Aura, Michael Roe, and Steven J. Murdoch. Securing network location awareness with authenticated DHCP. In *Proceedings of 3rd International Conference on Security and Privacy in Communication Networks (SecureComm 2007)*, Nice, France, September 2007. IEEE Press.

[BS03] Alastair R. Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, January–March 2003.

[BSF06] Andre Broido, Hao Shang, Marina Fomenkov, Young Hyun, and KC Claffy. The Windows of private DNS updates. *Computer Communication Review (ACM SIGCOMM)*, 36(3):93–98, July 2006.

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

[CK06] Stuart Cheshire and Marc Krochmal. Multicast DNS. Internet-Draft draft-cheshire-dnsext-multicastdns-06, IETF, August 2006. Expired.

[CPG04] Jim Chow, Ben Pfaff, Tal Garfinkel, Kevin Christopher, and Mendel Rosenblum. Understanding data lifetime via whole system simulation. In *Proceedings of 13th Usenix Security Symposium*, pages 321–336, San Diego, CA, USA, August 2004. USENIX.

[CSWH00] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 46–66, Berkeley, CA, USA, July 2000. Springer.

[Cra02] Lorrie Faith Cranor. *Web Privacy with P3P.* O'Reilly, September 2002.

[CJBMM04] Jorge R. Cuellar, Jr. John B. Morris, Deirdre K. Mulligan, Jon Peterson, and James M. Polk. Geopriv requirements. RFC 3693, IETF, February 2004.

[DSCP02] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*, volume 2482 of *LNCS*, San Francisco, CA, USA, April 2003. Springer.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, USA, August 2004. USENIX Association.

[FMT06] Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jamie Van Randwyk, and Douglas Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *15th Proceedings of USENIX Security Symposium*, pages 167–178, Vancouver, B.C., Canada, July 2006. USENIX Association.

[GDMR06] Ryan Gerdes, Thomas Daniels, Mani Mina, and Steve Russell. Device identification via analog signal fingerprinting: A matched filter approach. In *Proceedings of 13th Annual Network and Distributed System Security Symposium (NDSS 2006)*, San Diego, CA, USA, February 2006. Internet Society.

[GM82] Joseph A. Goguen and Jose Meseguer. Security policies and security models. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, pages 11–20, Los Alamitos, CA, USA, April 1982. IEEE Computer Society Press.

[GG03a] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of MobiSys 2003: The First International Conference on Mobile Systems, Applications, and Services*, pages 31–42, San Francisco, CA, USA, May 2003. USENIX Association.

[GG03b] Marco Gruteser and Dirk Grunwald. Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. In *Proceedings of 1st ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH)*, pages 46–55, 2003.

[GF07] Saikat Guha and Paul Francis. Identity trail: Covert surveillance using DNS. In *Proceedings of 7th International Symposium on Privacy Enhancing Technologies (PET 2007)*, volume 4776 of *LNCS*, Ottawa, Canada, June 2007. Springer.

[GGP+07] Ben Greenstein, Ramakrishna Gummadi, Jeffrey Pang, Mike Y. Chen, Tadayoshi Kohno, Srinivasan Seshan, and David Wetherall. Can Ferris Bueller still have his day off? Protecting privacy in the wireless era. In *Proceedings of 11th Workshop on Hot Topics in Operating Systems (HotOS XI)*, San Diego, CA, USA, May 2007. USENIX Association.

[JWH07] Tao Jiang, Helen J. Wang, and Yih-Chun Hu. Preserving location privacy in wireless LANs. In *Proceedings of 5th International Conference on Mobile Systems, Applications, and Services (MobiSys 2007)*, pages 246–257, San Juan, Puerto Rico, USA, June 2007. ACM Press.

[JP02] David B. Johnson and Charles Perkins. Mobility support in IPv6. RFC 3775, IETF, June 2004.

[KBC05] Tadayoshi Kohno, Andre Broido, and KC Claffy. Remote physical device fingerprinting. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2005. IEEE Computer Society.

[KC05] Braden Kowitz and Lorrie Cranor. Peripheral privacy notifications for wireless networks. In *Proceedings of Workshop on Privacy in Electronic Society (WPES'05)*, pages 90–96, Alexandria, VA, USA, November 2005. ACM Press.

[Law03] George Lawton. Instant messaging puts on a business suit. *Computer*, 36(3):14–16, March 2003.

[LT06] Janne Lindqvist and Laura Takkinen. Privacy management for secure mobility. In *Proceedings of Workshop on Privacy in Electronic Society (WPES'06)*, pages 63–66, Alexandria, VA, USA, October 2006. ACM Press.

[MCPS03] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. Internet-Draft draft-moeller-v2-01, IETF, July 2003. Expired.

[Mur06] Steven J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *Proceedings of ACM Conference on Computer and Communications Security (CCS'06)*, pages 27–36, Alexandria, VA USA, November 2006. ACM Press.

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

[PGM+07] Jeffrey Pang, Ben Greenstein, Damon McCoy, Srinivasan Seshan, and David Wetherall. Tryst: The case for confidential service discovery. In *Proceedings of the 6th Workshop on Hot Topics in Networks (HotNets-VI)*, Atlanta, CA, USA, November 2007. ACM Press.

[PGG+07] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan, and David Wetherall. 802.11 user fingerprinting. In *Proceedings of 13th Annual International Conference on Mobile Computing and Networking (MobiCom '07)*, Montreal, QC, Canada, September 2007. ACM Press.

[Pet02] Jon Peterson. A privacy mechanism for the session initiation protocol (SIP). RFC 3323, IETF, November 2002.

[PS01] Derrell Piper and Brian Swander. A GSS-API authentication method for IKE. Internet-Draft draft-ietf-ipsec-isakmp-gss-auth-07, IETF, July 2001. Expired.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[SM03] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, January 2003.

[SLH07] T. Scott Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal, and Tadayoshi Kohno. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *Proceedings of 16th USENIX Security Symposium*, Boston, MA, USA, August 2007. USENIX Association.

[SD02] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*, volume 2482 of *LNCS*, San Francisco, CA, USA, April 2002. Springer.

[SAH08] Dan Simon, Bernard Aboba, and Ryan Hurst. The EAP-TLS authentication protocol. RFC 5216, IETF, March 2008.

[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.

[Swe02] Latanya Sweeney. k-Anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[TN98] Susan Thomson and Thomas Narten. IPv6 stateless address autoconfiguration. RFC 2462, IETF, December 1998.

[YMC07] Aydan R. Yumerefendi, Benjamin Mickle, and Landon P. Cox. TightLip: Keeping applications from spilling the beans. In *Proceedings of 4th USENIX Symposium on Networked Systems Design & Implementation*, pages 159–172, Cambridge, MA, USA, April 2007. USENIX Association.

[ZCYH05] Qin Zhao, Winnie W. Cheng, Bei Yu, and Scott Hiroshige. DOG: Efficient information flow tracing and program monitoring with dynamic binary rewriting. Technical report, MIT, 2005.

[Zug03] Alf Zugenmaier. *Anonymity for Users of Mobile Devices through Location Addressing.* PhD thesis, University of Freiburg, Freiburg, Germany, 2003.