S-ARP: a Secure Address Resolution Protocol[*]

D. Bruschi, A. Ornaghi, E. Rosti
Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano, Italy
E-mail: alor@sikurezza.org, {bruschi, rosti}@dico.unimi.it

Abstract

Tapping into the communication between two hosts on a LAN has become quite simple thanks to tools that can be downloaded from the Internet. Such tools use the Address Resolution Protocol (ARP) poisoning technique, which relies on hosts caching reply messages even though the corresponding requests were never sent. Since no message authentication is provided, any host of the LAN can forge a message containing malicious information.

This paper presents a secure version of ARP that provides protection against ARP poisoning. Each host has a public/private key pair certified by a local trusted party on the LAN, which acts as a Certification Authority. Messages are digitally signed by the sender, thus preventing the injection of spurious and/or spoofed information. As a proof of concept, the proposed solution was implemented on a Linux box. Performance measurements show that PKI based strong authentication is feasible to secure even low level protocols, as long as the overhead for key validity verification is kept small.

1. Introduction

IP over Ethernet networks are the most popular Local Area Networks nowadays. They use ARP, the Address Resolution Protocol, to resolve IP addresses into hardware, or MAC (Medium Access Controllers), addresses [12]. All the hosts in the LAN keep a cache of resolved addresses. ARP resolution is invoked when a new IP address has to be resolved or an entry in the cache expires. The ARP poisoning attack consists of maliciously modifying the association between an IP address and its corresponding MAC address. Various tools available on the Internet [11], [13], [18], allow so called "script kiddies" to perform the sophisticated ARP poisoning attack.

* This work was partially sponsored under the Italian Dept. of Education and Research F.I.R.S.T. project.

Although this is the most popular version, ARP poisoning is not confined to Ethernet networks. Layer 2 switched LANs, 802.11b networks, and cryptographically protected connections are also vulnerable. In [3], various scenarios are described where a wireless attacker poisons two wired victims, a wireless victim and a wired one, or two wireless victims, either through different access points or a single one. As for cryptographically protected networks, the use of cryptography at network layer, e.g., by means of Secure Shell (SSH) [20] or Secure Sockets Layer (SSL) [4], does not protect against ARP poisoning, since such an attack is performed at the layer below.

By performing ARP poisoning, an attacker forces a host to send packets to a MAC address different from the one of the intended destination, which may allow her to eavesdrop on the communication, modify its content (e.g., filtering it, injecting commands or malicious code), hijack the connection. Furthermore, when performed on two different hosts at the same time, ARP poisoning enables an adversary to launch a "man in the middle" (MITM) attack. With MITM attacks traffic between two hosts is redirected through a third one, which acts as the man in the middle, without the two knowing it. The MITM may simply relay the traffic after inspecting it or modify it before resending it. Note that MITM attacks are possible at various layers of the OSI stack. ARP poisoning allows to perform such an attack at data link layer. At network layer, the attack exploits DNS poisoning [5]. The attacker first modifies the DNS tables so as to associate its own IP address with the symbolic names of both victim hosts. Thus, when the victims will query the DNS asking for the each other's IP address, they will receive the attacker's IP address. At this point, all the traffic between the two hosts will first be received by the attacker that will forward it to the respective destination, after possibly modifying it.

In this paper we propose a solution to the ARP poisoning problem based on an extension of the ARP protocol. We introduce a set of functionalities that enable an integrity and authenticity check on the content of ARP replies, using asymmetric cryptography. We call our secure extension to ARP “S-ARP”, Secure ARP. As a proof of concept, S-ARP has been implemented under the Linux operating system and the initial experimental results have shown it is a feasible and effective solution to the ARP poisoning attack, despite its use of asymmetric cryptography. Experimental measurements indicate that S-ARP has a negligible impact on system performance. Note that similar results can be obtained using Secure Link Layer [6]. However, since such a protocol provides a broader spectrum of security services at layer 2 such as traffic confidentiality, it is less efficient than S-ARP. We will discuss SSL in Section 6.

This paper is organized as follows. Section 2 illustrates the problem considered in this paper and recalls how ARP works and why it is vulnerable to poisoning. Section 3 and 4 describe S-ARP and its Linux implementation, respectively. Section 5 presents the results of experimental evaluation on a real system. Section 6 discusses related work. Section 7 summarizes our contributions and concludes the paper.

2. Problem Definition

2.1. Address Resolution Protocol

When an Ethernet frame is sent from one host to another on the same LAN, the 48 bit Ethernet address determines the interface to which the frame is destined. The IP address in the packet is ignored. ARP provides the mapping between the 32 bit IPv4 address and the 48 bit Ethernet address [15], [12]. In the rest of this section we briefly recall how ARP works.

When a host needs to send an IP datagram as an Ethernet frame to another host whose MAC address it ignores, it broadcasts a request for the MAC address associated with the IP address of the destination. Every host on the subnet receives the request and checks if the IP address in the request is bound to one of its network interfaces. If this is the case, the host with the matching IP address sends a unicast reply to the sender of the request with the <IP address, MAC address> pair. Every host maintains a table of <IP, MAC> pairs, called *ARP cache*, based on the replies it received, in order to minimize the number of requests sent on the network. No request is made if the <IP, MAC> pair of interest is already present in the cache. ARP cache entries have a typical lifetime of 20 minutes, but some operating systems may reset the expiration time every time they use an entry, thus possibly delaying forever entry refresh [15].

ARP is a stateless protocol, i.e., a reply may be processed even though the corresponding request was never received. When a host receives a reply, it updates the corresponding entry in the cache with the <IP, MAC> pair in the reply. While a cache entry should be updated only if the mapping is already present, some operating systems, e.g., Linux and Windows, cache a reply in any case to optimize performance. Another stateless feature of ARP is the so called *gratuitous ARP*. A gratuitous ARP is a message sent by a host requesting the MAC address for its own IP address. It is sent either by a host that wishes to determine if there is another host on the LAN with the same IP address or by a host announcing that it has changed its MAC address, thus allowing the other hosts to update their caches.

2.2. ARP Poisoning

By forging an ARP reply, an attacker may easily change the <IP,MAC> association contained in a host ARP cache. Since each host presumes its local cache to be trustworthy, the poisoned host will send IP packets encapsulated into Ethernet frames with a bogus MAC address as destination. This way the attacker may receive all the frames originally directed to some other host. If also the cache of the real destination host is poisoned, both communication flows are under the attacker’s control. The attacker realizes a two-way man in the middle, where she can forward the received packets to the correct destination after inspecting and possibly modifying them. The two end points of the connection will not notice the extra hop added by the attacker if the packet TTL is not decremented.

Some operating systems, e.g., Solaris, will not update an entry in the cache if such an entry is not already present when an unsolicited ARP reply is received. Although this might seem an effective precaution against cache poisoning, the attack is still possible. The attacker needs to trick the victim into adding a new entry in the cache first, so that a future (unsolicited) ARP reply can update it. By sending a forged ICMP echo request as if it was from one of the two victims, the attacker has the other victim create a new entry in the cache. When the first victim receives the spoofed ICMP echo request, it replies with an ICMP echo reply, which requires resolving first the IP address of the original ICMP request into an Ethernet address, thus creating an entry in the cache. The attacker can now update it with an unsolicited ARP reply.

ARP poisoning is possible also in switched networks. A layer 2 switch accepts the traffic that comes into each port and directs it only to the port to which the destination host is connected, except for broadcast messages which are sent to all ports. Therefore sniffing is no longer possible by simply configuring the network interface in promiscuous mode. However, it is possible to poison a host cache by sending an unsolicited ARP reply to the host containing the attacker’s MAC address. The same can be done against two hosts at the same time, thus allowing an attacker to intercept all the traffic between those two hosts, without the switch realizing it. Once the attacker has hijacked the packets of a communication, she can modify the payload or even inject new packets in the communication as long as the TCP sequence numbers are adjusted so as to maintain the communication synchronized.

3. Secure ARP

Secure ARP extends ARP with an integrity/authentication scheme for ARP replies, to prevent ARP poisoning attacks. Since S-ARP is built on top of ARP, its specification (as for message exchange, timeout, cache) follows the original one for ARP [12]. In order to maintain compatibility with ARP, an additional header is inserted at the end of the protocol standard messages to carry the authentication information. This way, S-ARP messages can also be processed by hosts that do not implement S-ARP, although in a secure ARP LAN all hosts should run S-ARP. Hosts that run the S-ARP protocol will not accept non authenticated messages unless specified in a list of known hosts. On the contrary, hosts that run the classic ARP protocol will be able to accept even authenticated messages. A mixed LAN is not recommended in a production environment because the part running traditional ARP is still subject to ARP poisoning. Furthermore, the list of hosts not running S-ARP must be given to every secured host that has to communicate with an unsecured one. The interoperability with the insecure ARP protocol is given only for extraordinary events and should be always avoided. It is intended to be used only during the transition phase to a full S-ARP enabled LAN.

3.1. Protocol Overview

S-ARP provides message authentication only. No traffic confidentiality is provided as we believe that such a service should be provided at higher levels in the OSI stack, e.g., by means of IPSec [7] or SSL [4] or specific secure application protocols such as SSH [20]. Furthermore, well configured layer 2 switches operating with S-ARP are sufficient to protect traffic from most of layer 2 attacks[1].

S-ARP uses asymmetric cryptography. Any S-ARP enabled host is identified by its own IP address and has a public/private key pair. A simple certificate provides the binding between the host identity and its public key. Besides the host public key, the certificate contains the host IP address and the MAC address of the Authoritative Key Distributor (AKD), a trusted host acting as key repository. Each host sends its signed certificate containing the public key and the IP address to the AKD, which inserts the public key and the IP address in a local data base, after the network manager's validation (see Section 3.2). Because of the restricted nature of such a repository, both in terms of number of keys and their exposure to compromise, no revocation lists are kept. In order to avoid replay attacks and to have a common time reference to evaluate expired replies, the AKD also distributes the clock value with which all the other hosts must synchronize.

In S-ARP all reply messages are digitally signed by the sender with the corresponding private key. At the receiving side, the signature is verified using the host public key. If the public key of the sender host is not present in the receiving host key ring or the one in the key ring does not verify the signature, the public key of the sender is requested from the AKD. The AKD sends it to the requesting host in a digitally signed message.

S-ARP adopted the Digital Signature Algorithm (DSA) as the signature algorithm [9]. Such a choice is not a constraint and the signing algorithm could be replaced with any other public key signature scheme. For the sake of efficiency (see Section 5), we use keys of 512 bits. Although 512 bit keys are not considered totally secure, they offer a sufficient degree of security for the type of information they protect in our case, especially if combined with a key refresh policy.

3.2. S-ARP Setup

The first step when setting up a LAN that uses S-ARP is to identify the AKD and distribute through a secure channel its public key and MAC address to all the other hosts. Such an operation may be performed manually when a host is installed on the LAN for the first time. On the other hand, a host that wants to connect to the LAN must first generate a public/private key pair and send its signed certificate to the AKD. Here the correctness of the information provided is verified by the network manager and the host public key together with its IP address is entered in the AKD repository. This operation has to be performed only the first time a host enters the LAN. If a host wants to change its key, it communicates the new key to the AKD by signing the request with the old one. The AKD will update its key and the association is correctly maintained. Section 3.5 explains the protocol behavior when IP addresses are dynamically assigned by a DHCP server. Once connected to the LAN, a host synchronizes its local S-ARP clock with the one received from the AKD.

3.3. Message Format

A S-ARP message is similar to an ARP message, with an additional portion at the end, to maintain compatibility with the original protocol. The additional S-ARP portion comprises a 12 bytes S-ARP header, and a variable length payload, as shown in Figure 1. ARP replies carry the S-ARP

1 Although this is not true for bus networks, such a topology is quickly becoming obsolete, being replaced by layer 2 switched LANs, so we focus on the latter.

header while ARP requests do not change. Future versions of the protocol should consider authenticating ARP requests too as this would speedup the authentication process.

The S-ARP header contains the sender's digital signature, a time-stamp, the type and length of the message. The field "magic" is used to distinguish whether a message carries the S-ARP header. If so, its value is `0x7599e11e`. Since ARP packets are only 42 bytes long and the minimum Ethernet frame length is 60, packets are usually padded with junk[2] and the length of the received packet cannot be used as an indicator of additional parts, such as a S-ARP header. The field "type" distinguishes among five types of messages:

- Signed address resolution (reply only)
- Public key management (request/reply)
- Time synchronization (request/reply).

Signed address resolution messages are exchanged between hosts of the LAN. The other types of messages are exchanged only between a host and the AKD.

The fields "siglen" and "datalen" are the length of the signature and the length of the data in the S-ARP payload, respectively. The field "timestamp" is the value of the local S-ARP clock at the moment of the construction of the packet. Finally, the field "signature" is a SHA-1 hash of the ARP and the S-ARP headers. The resulting 160 bits are signed with DSA[3].

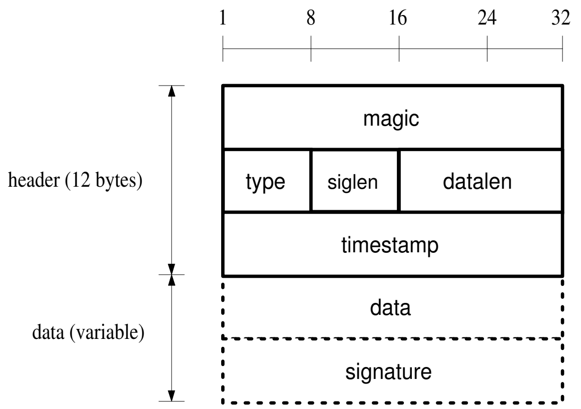

Figure 1. S-ARP packet extension.

2 In order to avoid information leakage [1], the S-ARP additional portion is first written with zeros.

3 The hash is computed with the field "siglen" equal to zero and after the signature has been calculated the field assumes the real length of the signature. This must be remembered during the verification process.

3.4. Message Authentication

Every host maintains a ring of the public keys and corresponding IP addresses previously requested to the AKD. When a host receives a S-ARP reply, it searches the sender IP address and its corresponding public key in its ring. If it finds such an entry, it uses the content to verify the signature, otherwise it sends a request to the AKD for the certificate. A request to the AKD is sent also in case the key in the local ring does not verify the signature, since it may no longer be valid[4]. In this case the packet is enqueued in a "pending replies list". The AKD sends a signed reply with the requested public key and the current time-stamp. Upon receiving the reply from the AKD, the host resynchronizes the local clock with the time-stamp, if necessary, stores the public key in its ring and verifies the signature. In case the old key were no longer valid, if the new key received from the AKD is the same as the one in the cache, the reply is considered invalid and is dropped. If the key has indeed changed, the host updates its cache and verifies the signature with the new key.

If the time-stamp in the S-ARP reply is too old, the reply is discarded to avoid replay attacks. Since hosts are not synchronized exactly, an acceptable difference between the time-stamp and the local clock is in the range of $30s$. Such a value is arbitrary and can be set by the network administrator, provided it is not so large to allow an attacker to launch a replay attack. Without the use of time-stamps, an attacker could successfully perform a poisoning attack even with S-ARP, in the following scenario. An attacker stores a sniffed S-ARP reply from victim 192.168.0.1 with MAC address 01:01:01:01:01:01. The attacker waits until the victim is off-line and cannot reply to ARP requests. At this point, the attacker changes its own MAC address to 01:01:01:01:01:01 and sends the stored S-ARP reply when requested.

3.5. Key Management

S-ARP hosts are identified by the IP address as it appears in the host certificate. Since particular care must be taken when dealing with dynamically assigned IP addresses, we consider key management in networks with statically or dynamically assigned IP addresses separately.

In the next sections we will use the following notation:

3.5.1. Static Networks In networks with statically assigned IP addresses, keys are bound to IP addresses when they are generated and then inserted in the AKD repository. Therefore, when a generic host i broadcasts a

4 S-ARP public keys do not have an explicit expiration date. They are changed either periodically by the system administrator or upon request in case of compromise.

AKD	Authoritative Key Distributor
S-DHCP	S-ARP enabled DHCP server
$\mathbf{H}_i$	Generic host i
Rq(a)	Request for object a
Rp(a)	Reply carrying object a
SHA(x)	SHA-1 hash of message x
T	Local S-ARP Time-stamp
N	Nonce
$\mathbf{A}_H$	Host H's IP address
$\mathbf{M}_H$	Host H's MAC address
$\mathbf{P}_H$	Host H's Public Key
$\mathbf{S}_H\mathbf{(x)}$	Message x digitally signed by host H

regular ARP request to find host j's MAC address, assuming j's key is not in i's cache, H_j replies with a signed message containing its own MAC address and the local S-ARP clock. Upon receiving host j's reply, H_i contacts the AKD to request j's key. The nonce N in host i's key request prevents replay attacks that could desynchronize its S-ARP clock. The AKD's signed reply includes the requested key, the nonce N and the time-stamp T, which host i will use to update its local S-ARP clock T_i. The sequence of messages exchanged is summarized below.

$\mathbf{H}_i \rightarrow \mathbf{all}$:	$\mathrm{Rq(M}_j)$
$\mathbf{H}_j \rightarrow \mathbf{H}_i$:	$\mathrm{S}_{H_j}(\mathrm{Rp(M}_j) \mid\mid \mathrm{T}_j)$
$\mathbf{H}_i \rightarrow \mathbf{AKD}$:	$\mathrm{Rq(P}_{H_j}) \mid\mid \mathrm{N}$
$\mathbf{AKD} \rightarrow \mathbf{H}_i$:	$\mathrm{S}_{AKD}(\mathrm{Rp(P}_{H_j}) \mid\mid \mathrm{N} \mid\mid \mathrm{T})$

Note that an attacker cannot produce a valid signature for an IP address other than its own. This is because the public key used for verifying the host's signatures has been released by the AKD, which first has verified the correctness of the information contained in the certificate submitted by the host and then released such an information in digitally signed messages. Thus an attacker can no longer send spoofed ARP replies to redirect traffic through its adapter. However, an attacker could still announce a false MAC address for its adapter, whether such an address be some other host's or a non-existing one. In the former case, the victim host would receive both its legitimate traffic and additional traffic originally directed to the attacker, thus possibly suffering a denial of service. In the latter case, all the traffic towards the attacker would be dropped, thus isolating the attacker.

3.5.2. Dynamic Networks In a S-ARP network where a DHCP server dynamically assigns IP addresses to the hosts, keys cannot be bound to IP addresses at generation time. Such a binding is dynamic and is renewed every time a host is assigned a new IP address. This implies that the DHCP server has to talk to the S-ARP server, thus requiring a customized version of the DHCP server. We implemented it and called it S-DHCP.

We assume that, if an organization deploys a secure DHCP server, dynamic IP addresses can be assigned only to well known machines that have been enrolled in the system and authorized in some way. What type of connection, to which sub-net, and other details regarding what a host may or may not do are part of the authorization profile associated with the host, as defined by the security policy of the organization. Part of the enrollment procedure is the generation by the host of the public-private key pair and the corresponding certificate. At this stage, the IP field of such a certificate is empty. To complete the enrollment procedure, the AKD manually inserts the certificate with the null IP address and the corresponding public key in its own key repository, using a secure channel. Note that this procedure is performed only once, before the host ever enters the system. Later on, if the host wanted to change its key, it could just send a key_exchange packet to the AKD.

When host H joins the network, it requests an IP address to the S-DHCP server. In order to allow the S-DHCP server and the AKD to identify it, H appends the signed SHA-1 digest of its public key P_H to the IP request to the S-DHCP server. Before assigning an IP address to H, the S-DHCP server contacts the AKD to verify whether H is authorized to be added to the LAN, i.e., if H's key is in the AKD repository and it is valid, and to inform the AKD of the IP address the host will be assigned. The message is signed by the S-DHCP server and comprises the original signed digest from H and the proposed IP address. The AKD searches its database for the given public key and replies to S-DHCP with an ACK or a NACK. The message exchange sequence in case of a positive response from the AKD is summarized below.

$\mathbf{H} \rightarrow \mathbf{S\text{-}DHCP}$:	DHCP request $\mid\mid \mathrm{S}_H(\mathrm{SHA(P}_H))$
$\mathbf{S\text{-}DHCP} \rightarrow \mathbf{AKD}$:	$\mathrm{S}_{S-DHCP}(\mathrm{S}_H(\mathrm{SHA(P}_H)) \mid\mid \mathrm{A}_H)$
$\mathbf{AKD} \rightarrow \mathbf{S\text{-}DHCP}$:	$\mathrm{S}_{AKD}(\mathrm{ACK})$
$\mathbf{S\text{-}DHCP} \rightarrow \mathbf{H}$:	$\mathrm{S}_{S-DHCP}(\text{DHCP reply} \mid\mid \mathrm{A}_H)$

If the response from the AKD is positive, the S-DHCP server proceeds with the assignment of the new IP address to H, while the AKD updates H's entry in the repository binding H's new IP address to H's key. If the response from the AKD is negative, the S-DHCP server will not release a new IP to the host and the host will not be able to join the LAN. Every time the S-DHCP releases a new IP address to a host for an expired lease or a new request, it will contact the AKD to inform it of the new association. The S-DHCP will release the renewal to the host and meanwhile will contact the AKD to inform it of the renewal. The procedure is the same as for a new assignment. From the AKD point of view there is no difference between manually inserted association or S-DHCP automatic association, so a mixed network with static and dynamically assigned IP addresses is managed correctly.

4. Implementation

S-ARP has been implemented under the Linux operating system and is available for download at URL : [10]. The prototype was implemented as a proof of concept and it is not intended to be a final and fully functional daemon to be used in large or production environments. It is composed of two parts: a kernel patch and a user-space daemon, as illustrated in Figure 2. The kernel patch removes the ARP packet from the incoming packet list through the `dev_remove_pack()` function. This way the kernel will not parse any ARP packets and will drop them. Note that the patch does not affect the way the kernel tries to resolve Ethernet addresses, since it continues to send ARP request as usual. It will only not process the replies. Since in the current version of S-ARP requests are not signed, it is possible to use the simple old ARP implementation for the requests and leave reply verification to a userland daemon. Such a daemon captures S-ARP packets through a link layer socket, verifies the signature and add the ARP entry in the system cache via a netlink socket. The daemon can act as AKD or as a generic host depending upon the command line parameter passed to the protocol at launch time. It is also responsible for the communications with the AKD for key management.

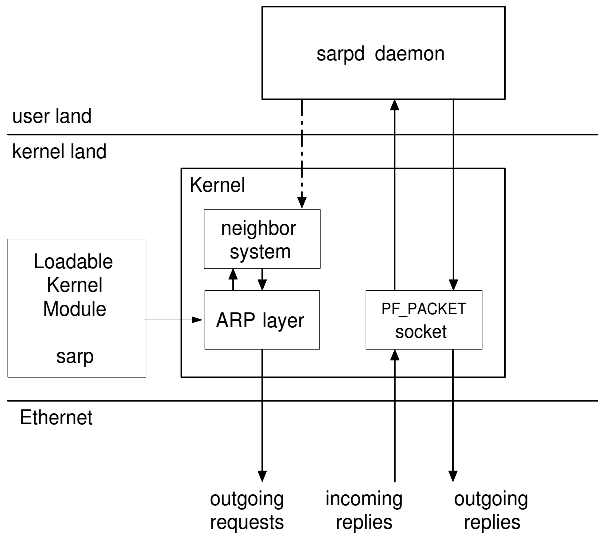

Figure 2. The structure of S-ARP.

Using a user land daemon was chosen not to burden the kernel with a time consuming task such as the verification of a digital signature. In particular, since the kernel (as of 2.4.x) is not preemptible, if the signature verification were left to it, no other task could execute until the verification had terminated. With the introduction of crypto API and kernel preemption in the upcoming 2.6 kernel, the current implementation could be revisited and compared with a kernel one, for the best performance.

5. Experimental Evaluation

In order to measure the overhead introduced by S-ARP, a test bed comprising three PC's connected through a 10 Mbit/sec hub was set up. A 1.0 GHz AMD Athlon 4 computer with 256 MB RAM running Gentoo Linux 1.4, kernel 2.4.20, acted as the AKD. Two 1.6 GHz Intel Pentium 4 computers with 128 MB RAM, running Debian Linux 3.0, kernel 2.4.18, acted as generic network hosts. Note that there is no difference in the implementations of ARP in the two distributions and kernel versions of the Linux operating system running on the test machines. We conducted two sets of measurements. We first measured the signature operation in isolation and then we indirectly measured the impact of S-ARP on address resolution.

5.1. Signature Performance

From a performance point of view, S-ARP execution time is dominated by signature verification and signature generation. Since the time required by signature verification depends upon the length of the key, which is a critical parameter of the protection level the key offers against cryptoanalytic attacks, the bit length of the public keys should strike a balance between these two factors. Signature creation is time consuming mostly due to the exponential calculation. However, some factors of such a calculation can be computed separately because they do not depend upon the message to be signed, thus significantly improving the execution time [14]. Unfortunately, nothing similar can be done for signature verification.

We ran 1000 tests to measure the time to generate a signature with pre-computation of the exponential factors and 1000 tests to measure the time to verify a signature for 512 bit and 1024 bit keys. The results are reported in Table 1. As the table shows, once the exponential factors have been computed, the time to generate the signature is independent of the key length. Furthermore, the time required to verify a signature is about 20-25% larger than the total time required by the complete generation of a signature.

key len.	operation	min	max	mean	std. dev.
512 bit	exp. fact.	923	1082	982.47	16.91
	sig. gen.	32	58	33.45	1.53
	sig. verif.	1133	1255	1201.46	15.45
1024 bit	exp. fact.	2565	2819	2721.67	38.05
	sig. gen.	34	59	35.36	1.50
	sig. verif.	3204	3458	3346.24	38.07

Table 1. Execution times in μsec for signature operations for different key lengths. Averages were obtained on 1000 tests.

5.2. ICMP Performance

We measured the performance of S-ARP indirectly, by means of ICMP messages. A set of `ping` commands were repeated, with no parameters, both with and without S-ARP. `ping` provides the roundtrip delay of an ICMP echo request from a host to another, which can be used as an indirect measure of the cost of address resolution. The first time an ICMP echo request/reply is sent, if the destination MAC address is unknown, an instance of ARP is executed.

`ping` returns the roundtrip delay for each ICMP message sent by the pinging computer, which for the first message includes the time for address resolution. It is therefore possible to estimate the impact of (S-)ARP in the execution time of ICMP. We identified the performance of the the baseline case when the system ran the original ARP. The average delay of the first echo reply, i.e., the one that requires Ethernet address resolution, is 0.705 msec, with and average standard deviation equal to 0.049. All the experiments were performed with “cold” caches, i.e., after flushing their content.

Two sets of experiments were performed. In the first scenario the two host have never communicated before, therefore they do not have each other’s public keys and request them to the AKD. Such a scenario is burdened with the highest overhead, but it occurs only the first time a new MAC address is needed, since keys are stored in cache after the first request. All subsequent requests will find the keys in cache, thus speeding up the execution. This is the second scenario considered, and it characterizes the average operating case of S-ARP. Measurements in this case include only the time required by signature verifications and creation.

5.2.1. Cold Key Caches When two hosts exchanging ICMP echo request/reply do not have each other’s key in their local cache, they have to request them to the AKD. In this case the authentication process requires 4 signature verifications and as many signature generations, which are irrelevant compared to the former if the exponential factors have been computed separately during an idle period, as shown in Table 1.

Table 2 summarizes the results for the measured roundtrip delays of ICMP echo requests for 512 bits and 1024 bits keys for 20 repetitions as yielded by the `ping` command[5] Although in both case the time is non negligible, we should remember that it occurs only the very first time, so it does not hurt performance in the average case. As the table shows, the roundtrip delay increases more than linearly as the key size increases, thus the importance to choose an appropriate size for the keys. For the sake of comparison, the table also reports the results of the same test performed with the classic ARP protocol. As expected, the cost of security is paid in performance degradation. However, such a cost is acceptable when the frequency of ARP traffic is taken into consideration.

key len.	min	max	mean	std. dev.
512 bit	17.7	18.1	17.86	0.12
1024 bit	48.0	48.8	48.49	0.22
classic ARP	0.6	0.8	0.70	0.05

Table 2. Roundtrip delay in μsec for ICMP echo request messages with cold key caches for different key lengths.

5.2.2. Cached Keys In this case there are two fewer verification operations, i.e., those on the AKD messages, so we expect it to be less time consuming. The public key of the two hosts are already in the respective key caches. This is the most common scenario. Two hosts have exchanged their keys in a previous communication, so when they communicate again they only need to verify each other’s signatures on the S-ARP replies. The AKD is not contacted in this case.

Table 3 summarizes the results for the measured roundtrip delays of ICMP echo requests for 512 bits and 1024 bits keys for 20 repetitions as yielded by the `ping` command. As the table shows, the time is almost half the time measured with cold caches, thus showing an acceptable overhead.

5 Caches are flushed after each execution of the `ping` command, in order to make sure they are cold on both machines.

key len.	min	max	mean	std. dev.
512 bit	8.8	9.3	8.96	0.13
1024 bit	23.6	24.4	24.00	0.20
classic ARP	0.4	0.5	0.46	0.05

Table 3. Roundtrip delay in μsec for ICMP echo request messages with cached keys for different key lengths.

6. Related Work

6.1. Defenses Against ARP Poisoning

A possible defence against ARP poisoning is using static entries in the ARP cache. Static entries cannot be updated by ARP replies and can be changed only manually by the system administrator. Such an approach however is not viable for networks with hundreds of hosts because those entries must be inserted manually on each host. Automating such a solution via a network script is not recommendable since it relies on higher levels of the ISO/OSI stack. Relying on higher levels when the data link layer has not been secured yet may be dangerous because the protocol used to exchange the list can be hijacked using ARP poisoning before the list is distributed. Even worse, some operating system (such as Windows) may accept dynamic updates even if an entry is set as static, thus making static Ethernet routing useless [19].

"Port security" is another mechanism for tackling the problem. It is a feature present in many modern switches that allows the switch to recognize only one MAC address on a physical port. This is often suggested as an effective protection against ARP poisoning, but it is not. If the attacker does not spoof its own MAC address, it can poison the two victims' cache without letting the switch interfere with the poisoning process.

Besides static cache entries and port security, the only other defense that will not modify ARP behaviour is detection. IDS and personal firewalls usually notice the ARP switch and warn the user that the entry in the cache is changed. As it often happens in the computer security domain, the decision is left to the user and his/her awareness. Given the particularly sophisticated level of operation in this case, we doubt the average user will take the proper actions.

Some kernel patches exist that try to defend against ARP poisoning. "Anticap" [2] does not update the ARP cache when an ARP reply carries a different MAC address for a given IP from then one already in cache and will issue a kernel alert that someone is trying to poison the ARP cache. Such a solution is against ARP definition itself, since it drops legal gratuitous ARP. "Antidote" [16] is more sophisticated. When a new ARP replies announcing a change in a <IP, MAC> pair is received, it tries to discover if the previous MAC address is still alive. If the previous MAC address replies to the request, the update is rejected and the new MAC address is added to a list of "banned" addresses. In [17] a solution that implements two distinct queues, for requested addresses and received replies, is proposed. The system discards a reply if the corresponding request was never sent, i.e., is not in the queue, and in the received queue an IP address associated with a different Ethernet address is already present.

All these solutions have the same problem. If the malicious ARP reply is sent before the real one is put in the cache, for a real request, the victim caches the wrong reply and discards the real one. A race condition exists between the attacker and the victim. When the first ARP request is broadcast, both the victim and the attacker receive the message. The first one who replies will take over the other forever. Furthermore, the attacker could also spoof an ICMP echo request message and immediately send after it a false ARP reply. When the victim receives the ICMP echo request, it performs an ARP request, but the false reply is already in its queue of received packet, so it accepts it a the valid one. If Antidote is installed, a host can spoof the sender MAC address and force a host to ban another host.

Solutions such as a centralized ARP cache or a DHCP server broadcasting ARP information, as they are deployed in IP over ATM networks [8], have not been considered as the attacker could spoof the source of the broadcast and poison the whole LAN. A digitally signed or MAC-ed broadcast packet would not be vulnerable to spoofing, yet broadcasting ARP tables could generate large traffic on the LAN. Since an entry for each host needs to be broadcast, on large networks this will generate considerable traffic and every host would have to store the entire ARP table even if it might not be needed at the moment. The main problem with centralized ARP cache is that if a host goes down, the central server will not notice the event. Thus, when a host that wishes to communicate with the one currently down asks for ARP information to the central server, it will receive the information even if the host is down. At this point an attacker could impersonate the offline host using its MAC address and receive all the packets sent to it.

6.2. Secure Link Layer

The only kernel patch which assures mutual authentication between the requester and the replier even on the first message is Secure Link Layer [6]. SLL provides authenticated and encrypted communication between any two hosts on the same LAN. SLL requires a Certification Authority (CA) to generate SLL certificates for all legitimate hosts on the network.

SLL handles authentication and session key exchange before any messages are transferred from one host to another. Elliptic curve cryptography algorithms are used for both operations. SLL defines three authentication messages that hosts send each other to perform mutual authentication and session key exchange. After authentication, the payload data field of all Ethernet frames sent between two hosts is encrypted with Rijndael using a 128-bit key and 128-bit long blocks.

Such a mechanism is too complex for our intent. Mutual authentication between two hosts is sufficient for avoiding ARP poisoning. Encrypting ARP replies does not yield any additional security since the association between IP and MAC addresses should be public. Furthermore, SLL also maintains all the cryptographic keys in kernel-space. Note that the amount of memory required could be considerable in case of class B networks. Since it is not recommended to use kernel memory with information that could be as well managed in user space, such as keys, a "light" version of SSL with no payload encryption would still have a considerable performance impact. Therefore we decide to design a new protocol that could be implemented in user-space.

7. Conclusions and Future Work

The paper presents a feasible solution to the problem of ARP poisoning attacks. The cause of ARP poisoning is the lack of message authentication, so that any host in the LAN is able to spoof messages pretending to be someone else. We propose an authentication scheme for ARP replies using public key cryptography, which extends ARP to S-ARP. Adding strong authentication to ARP messages resolves the problem, thus denying any attempt of ARP poisoning.

Future work includes porting S-ARP to other platforms so as to allow interoperability. Better kernel integration will be implemented since the upcoming Linux kernel (2.6.0) will be fully preemptible. Once the implementation of cryptographic routine will be moved to kernel space, even S-ARP request will be signed and the receiver will cache the information on the request, thus speeding up the whole authentication process.

When firewall and gateway appliances will be equipped with cryptographic co-processors, the implementation of S-ARP on embedded systems could be considered. Another issue concerns the elimination of the single point of failure represented by the AKD.

References

[1] AtStake.com. Etherleak: Ethernet frame padding information leakage. http://www.atstake.com/research/advisories/2003/a010603-1.txt, 2003.

[2] M. Barnaba. anticap. http://cvs.antifork.org/cvsweb.cgi/anticap, 2003.

[3] B. Fleck. Wireless access points and arp poisoning. http://www.cigitallabs.com/resources/papers/download/arp poison.pdf.

[4] A. O. Freier, P. Karlton, and P. C. Kocher. The secure socket layer protocol v3.0. http://wp.netscape.com/eng/ssl3/draft302.txt, 1996.

[5] A. Householder and B. King. Securing an internet name server. http://www.cert.org/archive/pdf/dns.pdf, 2002.

[6] F. Hunleth. Secure link layer. http://www.cs.wustl.edu/∼fhunleth/projects/projects.html.

[7] S. Kent and R. Atkinson. Security architecture for the Internet Protocol. RFC 2401, 1998.

[8] M. Laubach. Classical IP and ARP over ATM. RFC 1577, 1994.

[9] NIST. Digital signature standard (dss). Technical Report FIPS PUB 186, National Institute of Standards and Technology, http://www.itl.nist.gov/fipspubs/fip186.htm, 1994.

[10] A. Ornaghi. S-arp: a secure arp. http://security.dico.unimi.it/en/doctools/tools.html, 2003.

[11] A. Ornaghi and M. Valleri. A multipurpose sniffer for switched LANs. http://ettercap.sf.net.

[12] D. C. Plummer. An ethernet address resolution protocol. RFC 826, 1982.

[13] D. Song. A suite for man in the middle attacks. http://www.monkey.org/∼dugsong/dsniff.

[14] W. Stallings. *Criptography and Network Security*. Prentice Hall, ISBN 0-13-869017-0, 1998.

[15] R. W. Stevens. *TCP/IP Illustrated, vol 1*. Addison Wesley, ISBN 0-201-63346-9, 2001.

[16] I. Teterin. Antidote. http://online.securityfocus.com/archive/1/299929.

[17] M. V. Tripunitara and P. Dutta. A middleware approach to asynchronous and backward compatible detection and prevention of arp cache poisoning. In *Proc. 15th Annual Computer Security Application Conference (ACSAC)*, pages 303–309, 1999.

[18] R. Wagner. Address resolution protocol spoofing and man-in-the-middle attacks. http://rr.sans.org/threats/address.php, 2001.

[19] S. Whalen. An introduction to arp spoofing. http://packetstormsecurity.nl/papers/protocols/intro_to_arp_spoofing.pdf, 2001.

[20] T. Ylonen. Ssh: Secure login connections over the internet. In *Proc. of the Sixth Usenix Unix Security Symposium*, pages 37–42, 1996.