Cache poisoning in S-ARP and Modifications

Omkant Pandey
Crypto Group
Institute of Technology
Banaras Hindu University, India
omkant.pandey@cse04.itbhu.org

Vipul Goyal
OSP Global
Mumbai
India
vipulg@cpan.org

Abstract

The Address Resolution Protocol (ARP) was designed to convert IP address of a host into its Machine address (MAC). However, no attention was given to security and authentication leading to the problem of: ARP-cache poisoning; which in fact provides platform for several other attacks also. In an attempt to resolve the arp-cache poisoning problem, S-ARP (a Secure ARP) was proposed recently. S-ARP authenticates each ARP reply by requiring the original host's digital signatures in it.

Though S-ARP is a simple solution and is expected to be secure; it is not. This paper describes two scenarios in which cache-poisoning is possible in S-ARP. Later we propose improvements in S-ARP so as to ensure that these attacks are eliminated.

1 Introduction

Local Area Networks running TCP/IP or UDP/IP are the most popular networks these days. Each host on such a network is assigned an IP address (32 bits). Hosts also posses a network interface card (NIC) having a unique physical address (48 bits) also called MAC address. For the final delivery of any packet destined to some host, its MAC must be known to the Ethernet protocol. Thus address resolution protocol is used to resolve an IP address into MAC address. Resolved addresses are kept in a cache so as to avoid unnecessary work for already resolved addresses every time they are needed. Resolution is invoked only for expired or absent entries in the cache, otherwise cache entries are used. The ARP-Poisoning attack involves modifying the cache-contents so as to receive the data intended to someone else (victim). Even a newbie can launch sophisticated poisoning attack using easily available tools and tutorials on internet, [1], [2], [3], [4] and [5].

Cache poisoning is more serious than it seems. After poisoning the cache, attacker is in fact able to prepare a platform for other attacks also. By poisoning cache of a host, he is able to receive all traffic which originally is intended to someone else. This data can be inspected, changed or modified in anyway before being resend to the original receiver. If cache of the other party is also poisoned at the same time in a way attacker wants, he can launch a man-in-the middle (MiM) attack. Further, cache poisoning can be used to launch a Denial-of-Service (DoS) attack, [5].

The story does not end here. ARP poisoning is possible in Layer 2 switched LANs and 802.11b networks also. Even cryptographically protected connections are vulnerable because this protection is provided at network layer by means of Secure Shell (SSH) [6] or Secure Socket Layer (SSL) [7], where as ARP poisoning aims at layer below it.

Recently, a Secure Address Resolution Protocol abbreviated as S-ARP [8], was proposed to address the problem of cache poisoning. S-ARP uses digital signatures for message authentication. However, even with this cache poisoning is possible. This paper describes how cache can be poisoned even when S-ARP is used.

The organization of this paper is as follows. Section 2 describes working of ARP and the problem of cache poisoning. Section 3 describes S-ARP and in Section 4 we show how cache can be poisoned in S-ARP. Section 5 describes modifications in S-ARP to avoid poisoning and Section 6 concludes the paper and describes scope for future work.

2 Classical ARP and Cache-Poisoning

We first recall how ARP works and then pay attention to how cache is poisoned when ARP is in use.

2.1 Address Resolution Protocol

Hosts and applications in a network work with domain names which are converted to IP address by a DNS server [9]. But once packets containing application data arrive on the local Ethernet network of the host, packets can be transmitted only if the MAC address buried in the NIC of the destination host is known to the switch. Switch maintains mapping of physical ports to the hardware address. The Ethernet protocol uses this information to send data. Thus a conversion is needed from IP addresses to MAC and vice versa. ARP [10], [11] provides a dynamic mapping from an IP address to the corresponding hardware address. RARP does reverse of ARP (a dynamic mapping from MAC to IP).

In order to find MAC address of a host from its IP address, a client first broadcasts a message sayings “If anyone owns IP address *a.b.c.d*, please send your MAC address back to me”. It also includes its own IP address and MAC address in this request so that the destination host later will not need an ARP request for this host. Let us call the requesting client as *S* and destination client as *D*. *D* first stores/updates the IP-to-MAC mapping of requesting client in its ARP-cache (and so does other hosts on the network who see the broadcasted ARP request as they do not waste a chance of updating their entries). Now *D* sends a unicast reply to *S* mentioning its IP-to-MAC mapping. This mapping is stored by *S* in its ARP-cache. Now, whenever *S* requires conversion of IP to MAC, it checks out its cache before it broadcasts an ARP-request. Entries in ARP-cache must be refreshed every 20 minutes even if entry is in use [10], but most Berkley derived implementations do not follow this and restart the timer for an entry whenever a reference is made to that entry [11], [12], which as we later see, should not be done. When a host broadcasts an ARP-request for finding its own MAC address, it is called *gratuitous ARP*. Gratuitous serves two purposes – first, when a host is added to a network, gratuitous lets it know if some host already exists with IP address assigned to it; Second, when the NIC of a host is replaced, gratuitous lets other hosts know about it and let them update their respective caches. Gratuitous is also used in backup file servers [13]. Important point is that, ARP-replies (including gratuitous) are implicitly trusted and this makes ARP vulnerable to attacks.

Top

2.2 Poisoning the cache

Attackers send forged ARP-replies which can easily be created using tools like LIBNET [14] and fool the system. Suppose *S* sends an ARP-request to find *D's* MAC. In the forged ARP-reply, an attacker will put his own MAC-address as source-MAC and *D*'s IP address in source-IP address. Since ARP-replies are implicitly trusted, *S,* puts a wrong entry *D's IP address to Attacker's MAC* in his ARP-cache. Thus, attacker is able to deceive *S* and will receive all data meant for *D* until this cache-entry is correctly updated. This is called cache-poisoning. Thus, In a LAN, a unique IP address has a unique MAC address bounded to it. If a host's cache on this LAN contains wrong *<IP, MAC>* mapping in the cache, its cache is said to be poisoned. Formally:

Definition 1: *The ARP cache of a host is said to be poisoned if it contains an entry corresponding to a wrong <IP, MAC> mapping, i.e. the IP address IP and the Mac address MAC in the mapping do not belong to the same machine.*

As we discussed before, entries should be refreshed despite being in use after each 20 minutes. Timer should not be started whenever a reference is made because once cache is poisoned, attacker might continuously communicate by managing a reply from *S* within every 20 minutes and hence keeping the cache always poisoned. [5] Describes how easily a host's cache can be poisoned. Plethora of ARP attacks such as sniffing, broadcasting, DoS, connection hijacking and cloning all described in [15] can be launched after poisoning. Tools like ARPOISON [1] and ETTERCAP [2] make this task further easy. If attacker poisons the destination host's cache also, he/she is able to launch a two-way MiM thus controlling the communication between the two hosts where he /she can forward the data packets to the original destination after inspection and/or possibly modification. Attacker will cleverly not decrement the TTL so two end points of communication are unable to notice the extra hop added by the attacker.

ARP poisoning is possible in switched networks also where though sniffing is not possible just by configuring network interface in promiscuous mode [8], but it is possible to poison a host cache by sending an unsolicited ARP reply to the host containing attacker's MAC address. This when done against two hosts at same time, again allows the attacker to monitor and control communication between the two hosts where anything can be done with packets including injection of new packets while maintaining the communication synchronized by adjusting TCP sequence numbers.

3 S-ARP: Secure ARP

S-ARP [8] was recently proposed as a solution to prevent ARP cache poisoning and is worth considering. S-ARP does not trust ARP replied. Rather, it requires the sender to put her digital signatures on each reply (containing nonce and timestamp to avoid replay attacks). S-ARP uses asymmetric cryptography. Any S-ARP host is identified by its IP address and has a public/private key pair certified by a local trusted certification authority (CA). A simple certificate provides the binding between the host identity and its public key. Besides public key, certificate contains the host IP address and MAC address of Authoritative Key Distributor (AKD), a trusted host acting as key repository. All reply messages are digitally signed by the sender with the corresponding private key. At the receiving end, before an updating/addition of an entry, signature is verified by the public key of the corresponding

host. If public key is not in the local ring of the receiver host, it asks AKD for the public key. If signature is not verified, then also public key is requested by the AKD because the cached public key might no longer be valid (changed because of some key-updating policy or in case of a compromise). If verification succeeds, the received information is believed and corresponding action (update/add) is executed. Otherwise, reply is just rejected. Note that, all hosts are synchronized using a local S-ARP clock.

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 the hosts using ARP only. The S-ARP packet extension is shown in figure 3.1.

We briefly describe the meaning of each of the fields: - “magic” is used to distinguish whether a message carries the S-ARP header. If so, its value is 0x7599e11e. “type” indicates one of the following messages:

- Signed address resolution (rep. only)
- Public key management (req. / rep.)
- Time synchronization (req. / rep.)

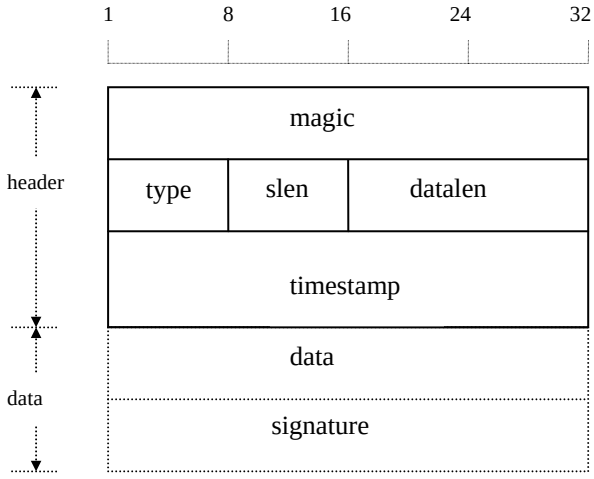

Fig. 3.1 S-ARP packet extension

Fields “slen” and “datalen” indicate length of signature and data. “timestamp” is the value of 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.

Key management is also a part of S-ARP. However it is not required for the purpose of this paper and the details can be found in the original paper [8].

4 Cache Poisoning in S-ARP

There are two possible cases in which S-ARP fails to avoid cache-poisoning: when a host replies to an arp-request, and then goes down in less than 20 minutes and when the attacker

provides wrong mapping for its own IP address so as to send its own traffic (possibly junk) to the victim leading to the possibility of a Denial of Service attack. Here we describe how it happens:

- This kind of cache poisoning is possible when the victim computer shuts down. Recall that entries in S-ARP are refreshed every 20 minutes (just as in classical ARP). Suppose a host H_2 sends an ARP request asking for the MAC address of host H_1. Host H_1 sends a signed reply leading to host H_2 updating/creating an entry mapping H_1's IP address to its MAC address. Just after time t (<20 minutes) after this event, host H_1 goes down. The attacker, which was waiting for this event, quickly changes its MAC address to what was H_1's MAC address. In this way all hosts (which stored a mapping for H_1) like H_2 now contain a poisoned entry (according to Definition 1) which map H_1's IP address to the attacker's MAC address. Thus, at this point the attacker can successfully receive all the messages targeted to H_1 for a time T=20-t minutes. Note that usually this time is more since the hosts also permit some time delay to take care of the clock synchronization errors.

- Since every system has full authority over its own IP address, an attacker can provide false information about its MAC. Suppose some host issues a request to know the attacker's MAC address. The attacker now supplies wrong information and claiming that its machine address is MAC_H (which in reality is machine address of the victim host H). This results in cache poisoning in accordance with Definition 1. With this, the attacker will be able to send a lot of junk traffic to victim host and hence possibly launching a kind of not-so-strong DoS attack. This attack can be made serious if the attacker owns multiple IP addresses. Further it will be hard for the victim to detect the attacker as the attacker need not send any direct traffic to the victim.

In the next section, we propose a few medications to S-ARP to eliminate these attacks.

5 Modifications

The attacks described above are restricted in the sense that the first scenario cannot last for more than $(20 - t)$ minutes and the host can avoid it if $t = 20$ minutes; i.e. it waits for upto 20 minutes before going down. In the second scenario, the victim receives extra traffic meant for only one machine (or a limited number of machines) and hence it is hard to launch a serious DoS attack. Yet, the important thing is that the first scene requires the host to stay for 20 minutes since the last ARP reply was sent to anyone on the LAN. This may not be desirable or even possible in some cases e.g. due to power backup limitations, a system learns upon a power cut that it will have to go down with a few minutes. Further, in the second case, although the attacker cannot launch a serious DoS attack in most environments, it can at least significantly slow down the victim system and/or consume a significant portion of its network connection bandwidth by junk traffic.
Thus, we consider it desirable to eliminate these attacks and modify S-ARP accordingly. In order to realize this, we propose the understated changes in S-ARP:

- So that hosts are not forced to wait for $(20 - t)$ minutes before going down; we introduce a final condition in S-ARP. Just before a host becomes offline (either it shuts down or it stops communication by shutting down its network interface because it's IP/MAC address is being changed), it broadcasts a digitally signed warning to get its entry, if any, deleted from the caches of each host on the network. Thus, if a host receives a warning from some other host, it checks its cache and deletes the entry for that host if it is found after properly verifying the

signatures. This modification allows the hosts to go down without waiting unnecessarily. Note that if an active attacker interferes to destroy this process; he can still poison the cache for maximum of *(20 – t)* minutes after which all entries get invalidated by themselves.

- Finally, to ensure that the second kind of attack is not possible; each host not only verifies digital signature in the arp reply but also ensures that there is one-one mapping in the arp-cache i.e. no two IP addresses are mapped to same MAC address in the cache. If so happens; the entry is not updated and the event is logged/reported to the system administrator. Note that the attacker is easily identified by the system administrator in this scenario as she is a known member of the network having a valid public key, IP address and a certificate binding the two. The attacker cannot deny the attempt of ARP cache poisoning as she has sent a fraudulent digitally signed <IP, MAC> mapping.

The first change does not put any additional burden on S-ARP and its running cost remains exactly the same. This requirement now changes the total number of messages distinguished by the "type" field. In addition to previous messages, this field now also recognizes following message:

- Signed request for entry deletion

The second change can be realized in practice either by maintaining a separate file having the cache entries sorted according to the Mac addresses or by searching for Mac address duplicity in the usual cache file. The first option should be preferred in most environments as it offers a better performance and can also efficiently serve the purpose of RARP (Reverse Address Resolution Protocol).

6 Conclusion and Future Work

We describe two scenarios in which cache poisoning is possible despite the use of S-ARP. We propose some modifications in the original protocol so as to ensure that no poisoning is possible. Proposed changes can be easily realized in practice with minimal changes in implementation and performance of S-ARP.

Future work includes measuring the performance of S-ARP with these changes included to substantiate the claim that the performance is not affected. Performance comparison can be shown by considering different probability models to model the going-offline trend among the hosts. When firewalls and gateways will be equipped with cryptographic co-processors, implementation of S-ARP on embedded systems can be considered. Other intelligent countermeasures can be devised which perform better than those described in this paper.

References

[1] ARPOISON, A tool for ARP-cache poisoning, available at link: http://web.syr.edu/~sabuer/arpoison/

[2] Ornaghi, A., and Valleri, M., ETTERCAP, A multipurpose sniffer for switched LANs Link: http://ettercap.sf.net.

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

Top

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

[5] Hacking UNIX 2003, a tutorial for performing various attacks including ARP poisoning attack, on UNIX systems. http://www.duho.cjb.net

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

[7] Freier, A.O., Karlton, P., and Kocher, P.C., The secure socket layer protocol v3.0. http://wp.netscape.com/eng/ssl3/draft302.txt, 2002.

[8] Bruschi, D., Ornaghi, A., Rosti, E., S-ARP: a Secure Address Resolution Protocol, in *Proceedings of 19th Annual Computer Security Applications Conference (ACSAC)*, 2003.

[9] Albitz, P., and Liu, C., DNS and BIND. O'Reilly and Associates, Sebastopol, California, 1992

[10] Plummer, D., "RFC-826: An Ethernet Address Resolution Protocol" IETF, Nov. 1982

[11] Stevens, R. W., "TCP-IP Illustrated vol. 1, The-Protocols". Printed by Pearson Education Asia, 2003.

[12] Comer, D. E., "Internetworking with TCP-IP" Vol. 1, 2001

[13] Bhide, A., Elnozahy, E. N., Morgan, S. P. "A Highly Available Network File Server," Proceedings of the 1991 Winter USENIX Conference, pp. 199-205, Dallas, Tex.

[14] LIBNET, A tool for creating packets. Link: http://www.packetfactory.net/libnet

[15] Whalen, S. H., "An Introduction to ARP Spoofing", 2003