Detecting ARP Spoofing: An Active Technique

Conference Paper *in* Lecture Notes in Computer Science · December 2005

DOI: 10.1007/11593980_18 · Source: DBLP

Sukumar Nandi
Indian Institute of Technology Guwahati

Detecting ARP Spoofing: An Active Technique

Vivek Ramachandran[1] and Sukumar Nandi[2]

[1] Cisco Systems, Inc., Bangalore India
[2] Indian Institute of Technology, Guwahati, Assam, India

Abstract. The Address Resolution Protocol (ARP) due to its statelessness and lack of an authentication mechanism for verifying the identity of the sender has a long history of being prone to spoofing attacks. ARP spoofing is sometimes the starting point for more sophisticated LAN attacks like denial of service, man in the middle and session hijacking. The current methods of detection use a passive approach, monitoring the ARP traffic and looking for inconsistencies in the Ethernet to IP address mapping. The main drawback of the passive approach is the time lag between learning and detecting spoofing. This sometimes leads to the attack being discovered long after it has been orchestrated. In this paper, we present an active technique to detect ARP spoofing. We inject ARP request and TCP SYN packets into the network to probe for inconsistencies. This technique is faster, intelligent, scalable and more reliable in detecting attacks than the passive methods. It can also additionally detect the real mapping of MAC to IP addresses to a fair degree of accuracy in the event of an actual attack.

1 Introduction

The ARP protocol is one of the most basic but essential protocols for LAN communication. The ARP protocol is used to resolve the MAC address of a host given its IP address. This is done by sending an ARP request packet (broadcasted) on the network. The concerned host now replies back with its MAC address in an ARP reply packet (unicast). In some situations a host might broadcast its own MAC address in a special Gratuitous ARP packet. All hosts maintain an ARP cache where all address mappings learnt from the network (dynamic entries) or configured by the administrator (static entries) are kept. The dynamic entries age out after a fixed interval of time, which varies across operating systems. After the entry ages out it is deleted from the cache and if the host wants to communicate with the same peer, another ARP request is made. The static entries never age out. A more detailed discussion of the ARP protocol is available at [1].

The ARP protocol is stateless. Hosts will cache all ARP replies sent to them even if they had not sent an explicit ARP request for it. Even if a previous unexpired dynamic ARP entry is there in the ARP cache it will be overwritten by a newer ARP reply packet on most operating systems. All hosts blindly cache the ARP

replies they receive, as they have no mechanism to authenticate their peer. This is the root problem, which leads to ARP spoofing.
ARP spoofing is the process of forging ARP packets to be able to impersonate another host on the network. In the most general form of ARP spoofing the attacker sends spoofed ARP responses to the victim periodically. The period between the spoofed responses is much lesser than the ARP cache entry timeout period for the operating system running on the victim host. This will ensure that the victim host would never make an ARP request for the host whose address the attacker is impersonating. Following subsection briefly discuss the current detection and mitigation techniques.

1.1 Current Mitigation and Detection Techniques

Existing ARP spoofing detection techniques are discussed next sequentially.

1.1.1 Secure ARP Protocol (S-ARP)

This has been proposed as a replacement for the ARP protocol in [10]. The S-ARP protocol is definitely a permanent solution to ARP spoofing but the biggest drawback is that we will have to make changes to the network stack of all the hosts. This is not very scalable as going for a stack upgrade across all available operating systems is something both vendors and customers will not be happy about. As S-ARP uses Digital Signature Algorithm (DSA) we have the additional overhead of cryptographic calculations though the authors of the paper have claimed that this overhead is not significant.

1.1.2 Static MAC Entries

Adding static MAC addresses on every host for all other hosts will not allow spoofing but is not a scalable solution at all and managing all these entries is a full time job by itself. This can fail miserably if mobile hosts such as laptops are periodically introduced into the network. Also some operating systems are known to overwrite static ARP entries if they receive Gratuitous ARP packets (GARP).

1.1.3 Kernel Based Patches

Kernel based patches such as Anticap[11] and Antidote[12] have made an attempt to protect from ARP spoofing at a individual host level. Anticap[11] does not allow updating of the host ARP cache by an ARP reply that carries a different MAC address then the one already in the cache. This unfortunately makes it drop legal gratuitous ARP replies as well, which is a violation to the ARP protocol specification [1]. Antidote [12] on receiving an ARP reply whose MAC address differs from the previously cached one tries to check if the previously learnt MAC is still alive. If the previously learnt MAC is still alive then the update is rejected and the offending MAC address is added to a list of banned

addresses.
Both the above techniques rely on the fact that the ARP entry in the cache is the legitimate one. This creates a race situation between the attacker and the victim. If the attacker gets his spoofed ARP entry into the hosts cache before the real host can, then the real MAC address is banned. This can only be undone by administrative intervention. Thus we can conclude that wrong learning may cause these tools to fail in detecting ARP spoofing.

1.1.4 Passive Detection

In Passive Detection we sniff the ARP requests/responses on the network and construct a MAC address to IP address mapping database. If we notice a change in any of these mappings in future ARP traffic then we raise an alarm and conclude that an ARP spoofing attack is underway. The most popular tool in this category is ARPWATCH [9].
The main drawback of the passive method is a time lag between learning the address mappings and subsequent attack detection. In a situation where the ARP spoofing began before the detection tool was started for the first time, the tool will learn the forged replies in it's IP to MAC address mapping database. Now only after the victim starts communicating with some other host the inconsistency will be detected and an alarm raised. The attacker may have made his getaway because of this delay. Also a spoofed entry learned as in the above scenario would have to be manually undone by the network administrator. The only solution to this problem is to manually feed the correct address mappings into the database before starting the tool or create an attack free learning traffic. Both of these are unreasonable due to scalability and mobility issues. An ideal example would be mobile hosts e.g. laptops brought in by customers or visitors to a company. This slow learning curve makes it impossible to install passive tools on a large network (1000+ hosts) and expect them to identify attacks instantaneously.
The passive techniques do not have any intelligence and blindly look for a mismatch in the ARP traffic with their learnt database tables. If an ARP spoofing is detected than there is no way of ascertaining if the newly seen address mapping is because of a spoofing attempt or the previously learnt one was actually a spoofed one. Our technique will determine the real MAC to IP mapping during an actual attack to a fair degree of accuracy.
The passive learning technique is also very unreliable. A new address mapping is learnt when ARP traffic is seen from them. Thus a switch ARP Cache table overflow attempt by the generation of random ARP reply packets per second with arbitrary MAC and IP addresses will just result in new stations being discovered instead of being reported as attack traffic. To overcome problems in earlier techniques, we present a new ARP spoofing detection technique. Our technique uses an active approach to detect ARP spoofing. We send out ARP request and TCP SYN packets to probe the authenticity of the ARP traffic we see in the network. The approach is faster, intelligent, scalable and more reliable in detecting attacks than the passive methods. It can also additionally detect

the real mapping of MAC to IP addresses to a fair degree of accuracy in the event of an actual attack. A description of the technique in detail is reported in following sections.

2 The Proposed Active Detection Technique for ARP spoofing

The proposed technique actively interacts with the network to gauge the presence of ARP spoofing attacks. We will henceforth assume the following about the network we desire to protect.

2.1 Assumptions

1. The attacker's computer has a normal network stack. This assumption will hold for most of the attacks as "ready to use" ARP spoofing tools [8] have always been the attacker's most popular choice. If the attacker does use a customized stack then our technique will still detect ARP spoofing but will not be able to predict the correct address mappings anymore. We will discuss performance in the presence of a customized stack in section 2.5.

2. The individual hosts we desire to protect on the network may use a personal firewall but at least one TCP port should be allowed through the firewall. This is to allow our probe packets (TCP SYN packets) to go through. This is a reasonable assumption as even if a firewall is installed some LAN based services such as NETBIOS etc are normally allowed through it for LAN communication.

3. We assume that all devices, which we protect, have a TCP/IP network stack up and running.

2.2 Terminology

We now introduce the terminology used in the rest of this paper.

1. *Threshold interval*: ARP replies to an ARP request must be received within a specified time interval. After this time has elapsed we will consider the ARP request to have "expired". We will call this interval as the "*Threshold Interval*". This will be administratively configurable on any tool using our technique.

2. *Host Database*: This is the mapping of all legitimate IP and MAC pairs on the network verified and learnt by our technique.

The ARP packets consist of the MAC header and the ARP header. Based on the value of the source and destination MAC addresses in the MAC header and as advertised in the ARP header we can divide the all ARP packets into 2 categories.

1. *Inconsistent Header ARP packets*: The MAC addresses in the MAC and ARP header differ i.e. Source MAC address in MAC header! = Source MAC address in ARP header (in ARP requests/responses) and/or Destination MAC address in MAC header! = Destination address in ARP header (only for ARP replies).

2. *Consistent Header ARP packets*: These are the compliment of the *Inconsistent Header ARP packets*. The MAC addresses in the MAC and ARP headers match in these packets.

Note that *Inconsistent Header ARP packets* are guaranteed spoofed packets, as such an anomaly is only possible in attack traffic. Based on the above classification we can further bunch the *Consistent Header ARP packets* into three groups:

1. *Full ARP Cycle*: An ARP request and it's corresponding ARP replies seen within the *threshold interval*.

2. *Request Half Cycle*: An ARP request for which no replies are sent as seen within the *threshold time*.

3. *Response Half Cycle:* An ARP reply generated without an ARP request.

These three categories form the basis of our input to the ARP spoofing detection mechanism. The following subsection discusses the Architecture of the proposed technique in detail.

2.3 Architecture

Please refer to Figure 1 for the architecture discussion. We have adopted a modularized approach and have divided our spoof detection into the following modules:

1. *ARP Sniffer module*: This sniffs all ARP traffic from the network.

2. *MAC - ARP header anomaly detector module*: This module classifies the ARP traffic into *Inconsistent Header ARP packets* and *Consistent Header ARP packets*.

3. *Known Traffic Filter module*: This filters all the traffic, which is already learnt. It will either drop the packet if the Ip to MAC mapping is coherent with the learnt Host Database or raise an alarm if there are any contradictions. All the new ARP packets with unknown addresses are sent to the *Spoof Detection Engine* for verification.

4. *Spoof Detection Engine module:* This is the main detection engine. We feed the *Consistent Header ARP packets* to it as input. The design of this module will be discussed in Section 2.4.

5. *Add to Database Module*: Legitimate ARP entries verified by the *Spoof Detection Engine* are added to the *Host Database* by this module.

6. *Spoof Alarm Module*: This module raises an alarm on detection of ARP spoofing by sending a mail, SMS etc to the administrator.

As shown in Figure 1, the *ARP Sniffer* module sniffs all the ARP traffic in its LAN segment and passes it to the *MAC – ARP Header Anomaly Detector*. This module passes the entire *Consistent Header ARP packets* to the *Known Traffic Filter* module. The entire *Inconsistent Header ARP packets* are sent to the Spoof Alarm. This is done because the *Inconsistent Header ARP packets* are all spoofed packets as discussed earlier. The *Known Traffic Filter* module will remove all traffic coherent with the already learnt addresses by consulting the *Host Database*. If there is a contradiction in the ARP traffic for already learnt addresses then it raises a *Spoof Alarm*. All new ARP traffic is passed to the *Spoof Detection Engine*.

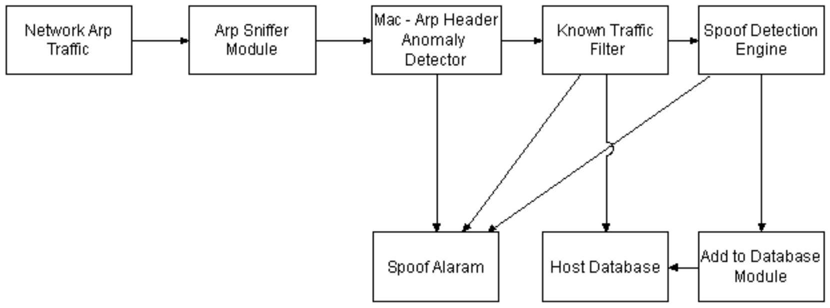

Fig. 1. Inter-relation between various Modules used by the ARP Spoof Detection Algorithm

The *Spoof Detection Engine* applies our detection algorithm to detect ARP spoofing. The newly seen Consistent *Header ARP packets* are input to this module. The engine now internally bunches these packets into the three categories discussed in Section 2.2 namely *Full ARP Cycle, Request* and *Response Half Cycle*packets. The detection algorithm applied by the engine will be discussed in the section 2.4. After applying the detection algorithm the *Spoof Detection engine* either sends the ARP entry to the *Add to Database* module or the *Spoof Alarm* module. The *Add to Database* module will add these verified MAC and IP address mapping to the *Host Database*. The *spoof detection engine* is discussed in detail next.

2.4 The Spoof Detection Engine

The *Spoof Detection Engine* is the heart of the whole system. The three different ARP Cycle packets as discussed in Section 2.2 are treated in slightly different ways by the *Spoof Detection Engine* to detect an attempted spoofing. The *Spoof Detection Engine* works based on the following Rules:

Rule A: "The network interface card of a host will accept packets sent to its MAC address, Broadcast address and subscribed multicast addresses. It will pass on these packets to the IP layer. The IP layer will only accept IP packets addressed to its IP address(s) and will silently discard the rest of the packets. If the accepted packet is a TCP packet it is passed on to the TCP layer. If a TCP SYN packet is received then the host will either respond back with a TCP SYN/ACK packet if the destination port is open or with a TCP RST packet if the port is closed".

Rule B: "The attacker can spoof ARP packets impersonating a host but he can never stop the real host from replying to ARP requests (or any other packet) sent to it. The valid assumption here is that the real host is up on the network."

It should be noted that these rules have been derived from the correct behavior that a host's network stack should exhibit when it receives a packet. To exemplify Rule A, let a host have MAC address = X and IP address = Y. If this host receives a packet with destination MAC address = X and destination IP address = Z then even though the network interface card would accept the packet as the destination MAC address matches, the host's network stack will silently discard this packet as the destination IP address does not match, without sending any error messages back to the source of the packet.

Based on Rule A, we can conceive of two types of probe packets from a host's network stack point of view which we will use to detect ARP spoofing.
a. *Right MAC – Wrong IP packet:* The destination MAC address in the packet is of the host but the IP address is invalid and does not correspond to any of the host's addresses. The destination host will silently drop this packet.
b. *Right MAC – Right IP packet:* The destination MAC address and IP addresses pairs are of the host's and its network stack accepts it.
We will henceforth assume that the attacker is using an unmodified network stack. The performance of our technique in the presence of a modified network stack will be evaluated in Section 2.5. Based on the above observation we will construct our own packets based on *Rule A* and send them on the network. We will use the address information in the ARP response packet sent by the host whose authenticity is to be verified. We will use the MAC and IP addresses used in the ARP response packet to construct a TCP SYN packet i.e. the destination MAC and IP in the TCP SYN packet will be the source MAC and IP address advertised in the ARP response packet and the source MAC and IP in the TCP SYN packet would be of the host running the *Spoof Detection Engine.* The TCP destination port will be chosen based on the presence/absence of packet filtering

firewalls on the network hosts. If there is a firewall installed on the hosts we will choose the “allowed TCP port” (as in section 2.1) and if no firewalls are there then we can choose any TCP port. The rest of the header values in the TCP SYN packet will be set as usual.

When a TCP SYN packet as constructed above is sent to the source of the ARP reply packet, the host’s response will be based on Rule A. If the ARP response was from the real host its IP stack will respond back with either a TCP RST packet (If the destination port is closed) or a TCP SYN/ACK packet (if the destination port is open).

If the ARP response had been from a malicious host then its network stack would silently discard the TCP SYN packet in accordance with Rule A. Thus based on the fact that the *Spoof Detection Engine* does/does not receive any TCP packets in return to the SYN packet it sent, it can judge the authenticity of the received ARP response packet.

We will now discuss how Rules A and B can be used together to detect ARP spoofing attempts in a network. Please refer to Figure 2 for a diagrammatic representation of the algorithm in the form of a flow chart. As we had mentioned earlier the ARP packets are classified into the 3 cycles namely *Full ARP Cycle, Request* and*Response Half Cycles* and then fed as input to the *Spoof Detection Engine.* We will now discuss the application of the above discussed technique to these 3 Cycles to detect ARP spoofing.

2.4.1 Full ARP Cycle A *Full ARP Cycle* will consist of an ARP request and one or more responses. We will send TCP SYN packet(s) constructed using the MAC and IP address information in the ARP reply packet(s) to the source host(s) as mentioned previously in Section 2.4. Based on *Rule A* only the real host will reply back with either a TCP SYN/ACK or RST packet. We will add this entry into our *Host Database* as a legitimate MAC to IP address mapping. All other ARP replies which were part of the recorded *Full ARP Cycle* are spoofed replies and the module will raise a *Spoof Alarm* for their addresses.

Note that not only we have detected spoofing but also have successfully detected the MAC to IP address mapping of the true host on the network, as only the true host’s network stack replies to TCP SYN probes as per *Rule A*.

2.4.2 Request Half Cycle A *Request half cycle* might arise when either the destination host is down on the network. If the source IP of the ARP request packet is unknown and not in our *Host Database* then we will send a TCP SYN packet constructed as mentioned in Section 2.4 from the source MAC and IP address information advertised in the ARP request packet. If we get a TCP SYN/ACK or RST packet in response then the host is authentic else we raise a *Spoof Alarm.* As an alternative way of detecting spoofing we also send an ARP

Request packet to the sender of the *Request Half Cycle* and we will raise a *Spoof Alarm* if we do not get the same MAC address in the ARP Response packet from the host in return. We will use both these mechanisms simultaneously to detect spoofing. The latter method will come in handy when the attacker uses a customized stack which we will discuss in Section 2.5. Figure 1 only contains the TCP method flow for simplicity.

2.4.3 Response Half Cycle A *Response Half Cycle* could arise because of two situations:

1. It is an ARP spoofing attempt by a malicious attacker. This is one of the most common ways of orchestrating an ARP spoofing by sending periodic spoofed ARP response packets to the victims so that the spoofed address entry never expires in the victim's ARP cache.

2. We may have missed the ARP request. This may happen if the detection tool just came online after the ARP request was sent and so we could only sniff the ARP response. Another remote possibility is we missed a packet because of a huge number of packets coming in and inadequate buffer space in the input queue.

To probe the authenticity of the sender of the ARP response we first send an ARP request packet corresponding to the ARP response packet i.e. the destination IP address in the constructed ARP request = the source IP address of the received ARP response and the source IP address of the constructed ARP request = *Spoof Detection Engine's* host's IP address. The Source MAC address of the constructed ARP request = *Spoof Detection Engine's* host's MAC address and the destination MAC address will be the broadcast address.

By *Rule B* even if an attacker is spoofing ARP packets on the network he cannot stop the real host from replying to an ARP request sent to it. As the destination MAC address of an ARP request is the broadcast address so every host will receive it. Thus when we send the above packet out to the network there could be two possible responses:

1. *One or more ARP responses:* We will now consider our ARP request and these ARP responses as a single Full ARP Cycle. This Full ARP Cycle will now be dealt with as in Section 2.4.1 to detect spoofing.

2. *No ARP responses:* If we do not receive any ARP responses than most probably the real host is down and the ARP responses we see are by an impersonating attacker. To detect this we send a TCP SYN packet constructed

as in Section 2.4 based on the information in the received ARP response packet. We will find that the impersonating host will not reply to this TCP packet as its network stack will discard it according to *Rule A* and we will raise a Spoof Alarm.

Thus we have successfully shown how we can detect ARP spoofing attacks in a network using our active injection technique. Till now we have assumed that the attacker is using a normal network stack and orchestrates all these attacks with ready to use tools such as ARP-SK [8]. We will now discuss the performance of our technique in the presence of a customized network stack used by the attacker.

2.5 Attacker Uses a Customized Stack:

Let us assume that the attacker is aware of our proposed method and has customized his network stack to reply to the TCP SYN packets and ARP request packets destined for the real host, he desires to impersonate. Even in such a scenario we will be able to detect ARP spoofing successfully using *Rule B*. The only limitation now would be that we would not be able to detect the real MAC and IP address as in the previous case.

Almost all ARP spoofing techniques continuously send spoofed ARP response packets to the victims. This is done so that the victim never needs to raise an ARP request, as the ARP cache entry for the host who's MAC is being spoofed never ages out. But if we send an ARP request on the network, requesting for the MAC address of the host (whose address is being spoofed) the host will reply with an ARP response packet (Rule B). Now we will have a MAC address mismatch for the same IP as the spoofed replies sent by the attacker previously will carry a different MAC address.

We will now discuss our performance for a customized network stack in the light of the ARP Cycles:

1. *Full ARP Cycle*: If spoofing is on then, both the Attacker and the real host will reply to the original ARP request and we can detect a conflict in the MAC address for the same IP. Also if we send a TCP SYN packet to the source address of both the ARP replies than we will receive two TCP packets in response as the attacker's customized stack replies as well along with the real host. This makes it very easy to detect spoofing.

2. *Request Half Cycle*: As outlined in Section 2.4.2 we will try to authenticate the sender of the request by sending an ARP request packet back to the sender and check the reply(s) for spoofing. If the source MAC addresses in the ARP reply packet received for the injected ARP request does not match

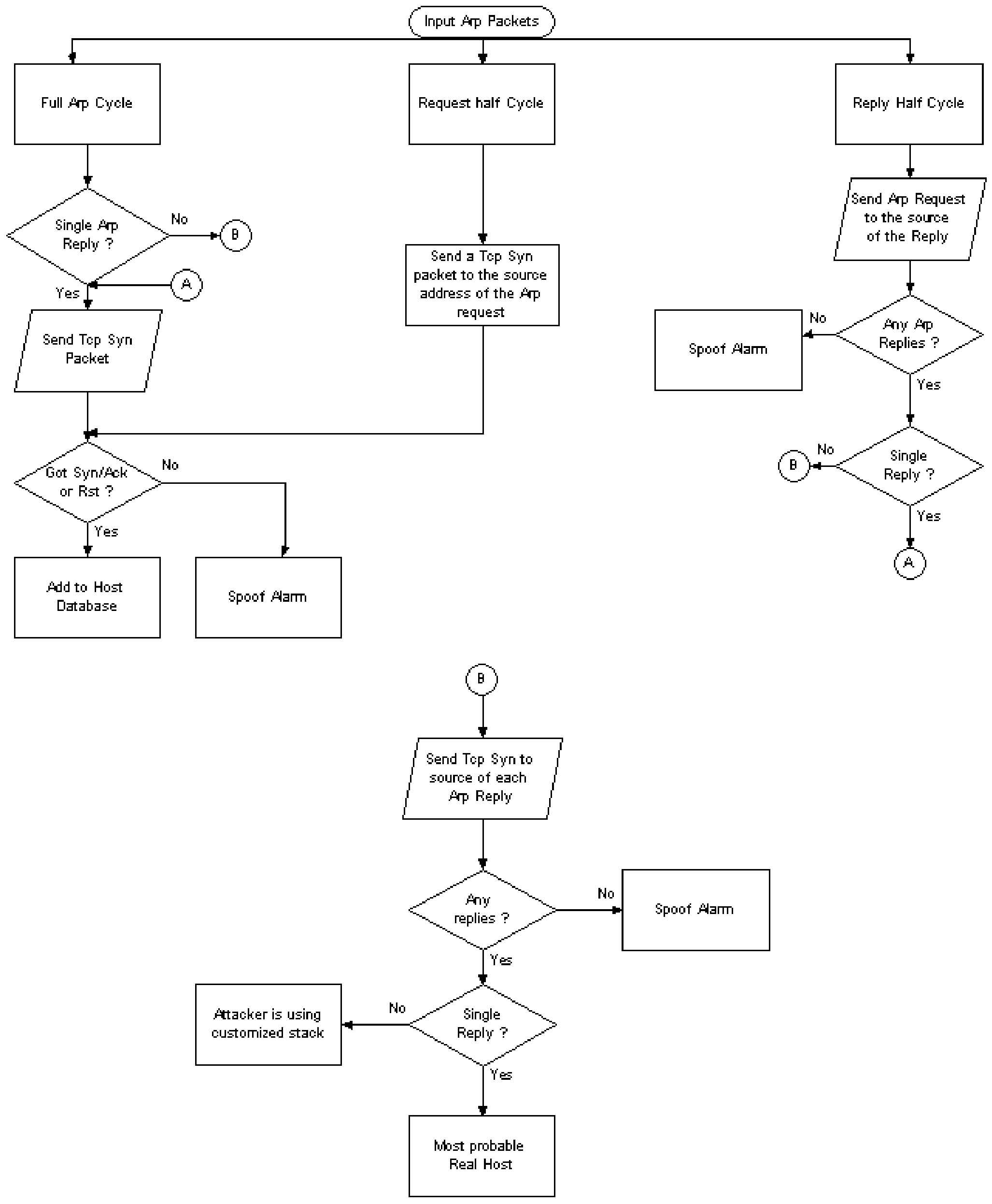

Fig. 2. Flow Chart Representation of the Spoof Detection Engine

the MAC address in the ARP *Request Half Cycle* packet then we will raise a *Spoof Alarm*.

3. *Reply Half Cycle*: If a customized stack is used we will get multiple replies to the ARP request we send as in Section 2.4.3. Also when we send out TCP SYN packets to the sources of the ARP request we will get multiple TCP (SYN/ACK or RST) packets in return with different MAC addresses. This is enough to conclude that a spoofing is going on.

Note that though we can detect ARP spoofing even in the presence of an attacker aware of our methods and using a customized stack we cannot predict the correct MAC to IP address mapping. This is the only limitation of our method in the presence of a customized stack.

3 Comparison with Passive Techniques:

Our technique is clearly much faster and reliable than the passive detection techniques. This technique can be used in a large network and it will immediately detect ARP spoofing attacks even if the attack had begun before the tool using our technique was operational. This is because the time lag between learning and detection is very less as we probe the authenticity of hosts as soon as we see ARP traffic from them. Also our technique verifies the authenticity of the ARP traffic on the network and does not blindly add newly seen traffic to its database. Even in the event of an actual attack our technique can detect the correct IP to MAC address mapping of the real host in the absence of the attacker using a customized network stack. If the attacker uses a customized stack, which replies to our probes we are still able to detect ARP spoofing but will not be able to predict the real MAC to IP address mapping. So even in our worst-case scenario (in the presence of a customized stack) our performance is still better than using a Passive detection technique.

4 Conclusion:

This paper proposed an active technique to detect ARP spoofing. We have shown that our technique is much faster, intelligent and scalable compared to passive detection techniques. Our technique also detects the correct MAC to IP address mapping during an actual attack. In presence of a customized stack, our detection algorithm will still detect ARP spoofing, though it is not be able to infer the correct address mapping. As we are using an active method to probe the authenticity of ARP traffic on a per packet basis, the time lag between learning new addresses and detecting spoofing is minimum. The network overhead due to the packet injection by us is fairly minimal as we send one ARP request and one TCP SYN packet per host on the network and then infer their authenticity based on the replies to our packets. Also as these packets are sent out only once for every host, for the entire lifetime of the tool, it makes it very scalable even for large networks.

References

1. D Plummer, “An Ethernet Address Resolution Protocol”, RFC-826, USC Information Science Institute, California, November 1982. http://www.ietf.org/rfc/rfc0826.txt
2. Stevens, W. Richard. “TCP/IP Illustrated, Volume 1. The Protocols”. Addison Wesley Longman, Inc, 1994. ISBN: 0201633469.
3. R.Wagner, “Address Resolution Protocol Spoofing and Man in the Middle Attacks” http://rr.sans.org/threats/address.php,2001.
4. A. Ornaghi and M. Valleri, “A multipurpose sniffer for switched LANs” http://ettercap.sf.net.
5. AtStake.com. Etherleak: Ethernet frame padding information leakage. http://www.atstake.com/research/advisories/2003/a010603-1.txt, 2003.
6. Althes. “The IP Smart spoofing”, InterOp Paris 2002. http://www.althes.fr/ressources/avis/smartspoofing.htm
7. Yuri Volobuev. “Redir games with ARP and ICMP”. http://lists.insecure.org/lists/bugtraq/1997/Sep/0059.html
8. Fredric Raynal, Eric Detoisien, Cedric Blancher, “ARP-SK: a swiss knife tool for ARP”. http://www.ARP-sk.org/
9. Lawrence Berkeley National Laboratory , “ARPWATCH tool”: ARP Spoofing Detector. ftp://ftp.ee.lbl.gov/ARPwatch.tar.gz
10. Danilo Bruschi, Alberto Ornaghi, Emilia Rosti , “S-ARP: a Secure Adderess Resolution Protocol” 19th Annual Computer Security Applications Conference, 2003, www.acsac.org/2003/papers/111.pdf
11. M. Barnaba, “Anticap” http://cvs.antifork.org/cvsweb.cgi/anticap, 2003
12. I. Teterin , “Antidote” http://online.securityfocus.com/archive/1/299929