

ARP & ICMP WEAKNESSES: IMPACT & NETWORK PERFORMANCE ANALYSIS OF A NOVEL ATTACK STRATEGY

Ashish Anand

ashish.anand@titsbhiwani.org

Ashish Jain

ashish.btech@gmail.com

Dept. of Computer Science

Maharishi Dayanand University (Haryana), India

Abstract – After the ARP and IP were drafted, a subtle weakness in the Address Resolution Protocol was discovered. Unlike TCP, ARP relies on raw sockets and like UDP; ARP provides no means to establish the authenticity of the source of incoming packets. Although this problem can be resolved in case of UDP packets by considering alternate approaches such as DNS replies being sent over TCP rather than UDP using the DNSSEC architecture so that false DNS replies may not be accepted by a host; ARP is still prone to similar attacks. This paper identifies known weaknesses of the ARP and analyses the impact of a network flooding utility developed by us, the underlying ideology of which is this very weakness of the ARP. The purpose of our implementation is to extend what conventional tools can do, by incorporating a network flooding module in it, and to simulate a flooded network where hosts are forced to broadcast outgoing packets to the entire network. In some network conditions, the gateway may also be brought into broadcast mode, leading to undesired results. Various attack strategies are considered and the network performance during these attacks is measured. We also reveal a strategy by which ICMP replies are received by a host trying to PING a destination, but the host fails to recognize these replies. Such a weakness in the ICMP can lead to erroneous network management.

Keywords – Network Security, ARP & ICMP Protocols, Performance Analysis

I. INTRODUCTION

When two hosts wish to communicate over a switched network for the first time, mere knowledge of the destination IP address is not sufficient. It is essential to establish the physical location of the destination host, which is denoted by a unique hardware address that binds each host's IP address to an individual port on the Layer 2 switch (obviously assuming that

physical connections are static). This initial discovery is facilitated by the Address Resolution Protocol that communicates at the MAC level, rather than at the IP level. Once each host has identified the physical location of other hosts under the switch using the Address Resolution Protocol, they may then communicate using the Internet Protocol.

After the ARP and IP were drafted [1] a subtle weakness in the Address Resolution Protocol was discovered [2]. It was realized that unlike TCP (connection oriented), the ARP relies on raw sockets and somewhat parallel to the UDP, ARP provides no means to establish the authenticity of the source of incoming packets. Although this problem can be resolved in case of UDP packets by considering alternate approaches such as DNS replies being sent over TCP rather than UDP using the DNSSEC [3] architecture so that false DNS replies may not be accepted by a host; the ARP is still prone to similar attacks. At the time of this writing, there exists no means to identify the authenticity of originating ARP packets under a Layer 2 switched network. Needless to mention, various attack strategies have been devised making use of this weakness in the Address Resolution Protocol.

This paper identifies known weaknesses of the Address Resolution Protocol and analyses the impact of a network flooding utility developed by us, the underlying ideology of which is this very weakness of the Address Resolution Protocol. The purpose of our implementation *dudaARP* is to extend what conventional tools such as *Ettercap*, *ARPSpoof*, *DSniff* etc. [4] can do, by incorporating a network flooding module in it, and to simulate a flooded network where hosts are forced to broadcast outgoing packets to the entire network. Depending on the network topology, type of hardware and subnet configuration, the gateway may also be brought into broadcast mode. Various attack strategies are considered and the network performance during these attacks is studied.

IV. OBSERVATIONS

The aftermath of our attack strategy on the HTTP/TCP and ICMP protocols was found to exhibit the following characteristics. Here we assume *A* to be the attacker, *B* the client and *C* the gateway/server. Their IP addresses and MAC addresses are denoted by *A_IP*, *B_IP*, *C_IP* and *A_MAC*, *B_MAC*, *C_MAC* respectively. Broadcast MAC (*FF:FF:FF:FF:FF:FF*) is denoted by *BDCT_MAC*.

A. HTTP/TCP Packets

Normally, when *B* wishes to communicate with *C*, after *A* has acquired *B*'s MAC address, a packet with these parameters is sent to *C*:

Source IP: *B_IP*
Source MAC: *B_MAC*
Destination IP: *C_IP*
Destination MAC: *C_MAC*

The conventional three-way handshake between *B* and *C* takes place only if *C* receives packets from *B* destined to the $\{C_IP - C_MAC\}$ binding only. However, during a single host flood, packets destined to *C* are forced to have the following parameters:

Source IP: *B_IP*
Source MAC: *B_MAC*
Destination IP: *C_IP*
Destination MAC: *BDCT_MAC*

Thus, we observe that *C* will not acknowledge such a packet and the intended handshake will not take place. Instead, the same packet will be retransmitted from *B* to *C* over and over again, effectively resulting in a denial of service attack on *B*. By continuously sending out ARP packets to *B* from *A* to poison *B*'s ARP cache, at a rate faster than the TTL of the cache (operating system specific), it is possible to cause a permanent denial of service attack on *B*, assuming no intrusion detection systems or other defense mechanisms are incorporated.

However, by incorporating conventional ARP spoofing techniques [7], it is possible to make the three-way handshake between *B* and *C* successful, so that the packets originating from *B*, destined to *C*, and vice versa, may be monitored by making them pass through *A*, and continuous communication between *B* and *C* would still be possible. Two packets with the following parameters must be crafted by *A*:

ARP Packet 1

Source IP: *B_IP*
Source MAC: *B_MAC*
Destination IP: *C_IP*
Destination MAC: *C_MAC*
Contents: *B_IP* is at *A_MAC*

ARP Packet 2

Source IP: *C_IP*
Source MAC: *C_MAC*
Destination IP: *B_IP*
Destination MAC: *B_MAC*
Contents: *C_IP* is at *A_MAC*

Note that 'IP Forwarding' must be enabled at *A*. If not, denial of service would result on both *B* and *C*. This however, is an operating system implementation specific issue, and is beyond the scope of this paper.

We extend this approach of crafting ARP packets to making both the source and destination believe that the other is physically located at *BDCT_MAC*. The network bandwidth is then monitored. We are interested in initial TCP handshake packet behavior and HTTP packet behavior. Shown below are our observations on an isolated LAN where *A* sends these ARP packets to *B* and *C*.

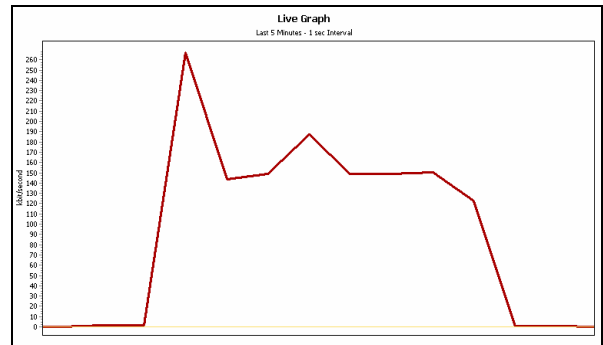


Figure 4: Normal HTTP Behavior

These figures show the bandwidth consumption of the interface being monitored with respect to time. The sharp rise indicates the HTTP GET request being answered [Figure 4] and subsequent bandwidth is consumed during downloading a webpage. The initial negligible rise before the sudden peak is the three-way TCP handshake [Figure 5]. Evidently, only a fraction of the bandwidth is consumed during this handshake, as compared the rest of the bandwidth utilization occurring while downloading a webpage.

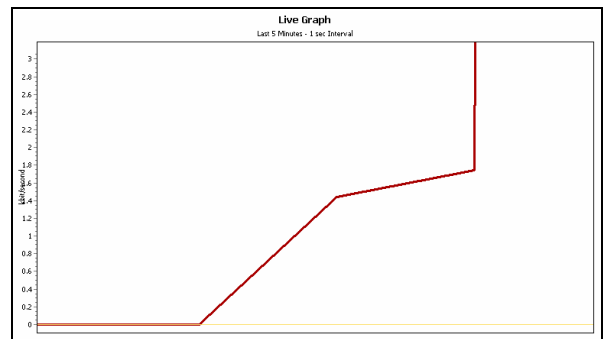


Figure 5: TCP Handshake
(Zoomed Initial Phase of Fig. 4)

More erratic bandwidth utilization is observed once the attack is in progress. When B tries to connect to the web server on C , it must first establish the three-way TCP handshake. Thus, B sends an initial TCP $\{syn\}$ packet [8] to C_IP , which is meant to reach the location C_MAC , but actually gets destined to $BDCT_MAC$ due to our attack conditions. When this packet arrives at C_IP , it fails to get recognized and the host C does not reply back with the expected $\{syn, ack\}$ packet. A timeout occurs and the initial $\{syn\}$ packet is retransmitted by B destined to C (but $BDCT_MAC$ rather than C_MAC as long as the attack is in progress). This process occurs a number of times [Figure 6], as long as B keeps trying to connect to C while the attack is in progress.

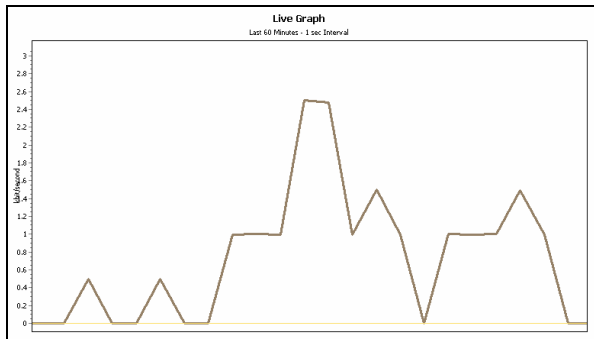


Figure 6: Zoomed Initial $\{syn\}$ Packet Retransmission

From the following overlapped comparison [Figure 7], it follows that while the attack is in progress, multiple $\{syn\}$ packets are broadcast across the network. While a successful handshake would not use up more than three packets, the attack conditions make the victimized host generate multiple packets during the same time duration. The number of these multiple packets generated across the network by a victimized host may be generalized by the expression:

$$N_{packets} = N_{hosts} * N_{attempts}$$

where,

N_{hosts} denotes the number of online hosts under the network segment that receive the broadcast packet.

$N_{attempts}$ denotes the number of connection retries before terminating the attempt.

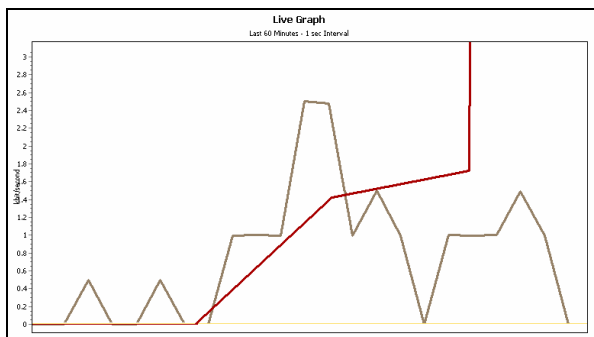


Figure 7: Successful Handshake vs. Retransmitted $\{syn\}$ Packet Bandwidth Utilization

Incase of conventional Windows systems, $N_{attempts}$ is usually equal to three. Thus, the number of unwanted packets generated across the network is directly proportional to the number of online hosts that receive the broadcast. Such a condition can lead to network congestion when the number of online hosts is large.

B. ICMP/PING Packets

Normally, when host B tries to PING host C [9, 10], an ICMP Request packet is sent to B 's IP address. Similarly, in normal conditions, host C returns an ICMP Reply packet back to host B . It is a well known fact that ARP is a non-routable protocol. Thus, limiting this discussion to a network segment under a single router/LAN, ARP related issues come into the play. The following paragraphs talk about our observations about the ICMP/PING tool behavior under Microsoft Windows and Redhat Enterprise Linux.

In our attack strategy, we define three players, i.e. the victim, gateway and attacker. In all cases, the attacker is the Linux host on which our implementation is running. We then perform a series of PING attempts from the victim to the gateway and vice versa, considering all combinations of the operating systems being tested. The first attack strategy (single host broadcast) is where the victim is made to believe that the MAC address of the gateway is $BDCT_MAC$ after which, the PING operation is performed from both locations, while in the second attack strategy (host-cum-gateway broadcast attack), before the PING operations, we poison the ARP cache of both the victim as well as the gateway, making each one think that the other host's MAC address is $BDCT_MAC$.

V. ATTACK RESULTS

Note that under networks equipped with intrusion detection systems [11], such an attack will not be successful since such network activity can be easily detected. However, the purpose of our strategy was different from that of conventional ARP spoofing, and thus we ignore this fact.

A. Single Host Broadcast Attack

When we poison the Windows ARP [Table 1.a] cache and try to PING the Linux host, the ICMP Request is received at the gateway and the ICMP Reply is issued and successfully received back and recognized by the victim, thus making the attack unsuccessful. In spite of having poisoned the Windows cache, the attack was unsuccessful.

Once the Windows host has been poisoned [Table 1.b], the gateway would obviously be able to PING the victim as its cache has not yet been poisoned. This is not an interesting case from the attacker's point of view.

When we poison the Linux ARP cache and try to PING the Windows host [Table 1.a], the ICMP Request is received at the gateway, but this is because the gateway MAC is a

Table 1: ICMP/PING behavior under MS Windows & Redhat Enterprise Linux

Victim	Gateway	Victim-->Gateway PING		
		Request Received	Replied By Destination	Reply Recognized
Windows	Linux	Y	Y	Y
Linux	Windows	Y	N	N

[A]

Single Host Broadcast Attack

Victim	Gateway	Gateway-->Victim PING		
		Request Received	Replied By Destination	Reply Recognized
Windows	Linux	Y	Y	Y
Linux	Windows	Y	Y	N

[B]

Victim	Gateway	Victim-->Gateway PING		
		Request Received	Replied By Destination	Reply Recognized
Windows	Linux	Y	Y	N
Linux	Windows	Y	N	N

[C]

Host-cum-Gateway Broadcast Attack

Victim	Gateway	Gateway-->Victim PING		
		Request Received	Replied By Destination	Reply Recognized
Windows	Linux	Y	N	N
Linux	Windows	Y	Y	N

[D]

subset of BDCT_MAC and hence besides the rest of the network, the Windows host also receives this ICMP Request packet. However, the Windows host does not issue an ICMP Reply packet back to the victim. Needless to mention, the Linux host is forced to believe that the Windows gateway is down.

Once the Linux host has been poisoned [Table 1.b], the Windows gateway does not recognize ICMP Reply packets arriving from the victim and is forced to believe that the victim host is down.

B. Host-cum-Gateway Broadcast Attack

When the ARP cache of both hosts is poisoned [Table 1.c], they are forced to believe that the other host is located at BDCT_MAC and this leads to interesting results. The Windows host sends an ICMP Request to BDCT_MAC, thus it also reaches the Linux host. The gateway issues an ICMP Reply packet and sends it out to BDCT_MAC but this packet is not recognized by the victim even though it physically arrives at the victim. Thus, the victim displays a *Request Timed Out* error even though the destination host is up.

When the ARP cache of both hosts is poisoned [Table 1.c], and the Linux host is considered to be the victim while the Windows host is considered to be the gateway and the victim tries to PING the gateway, the ICMP Request packet is received at the gateway, but it does not send back an ICMP Reply packet to the victim. Thus, the victim is forced to again believe, that the gateway host is down, in spite of its being alive.

The reversed case of ‘Gateway → Victim PING’ [Table 1.d] is actually the same as the last two cases discussed in the case of a host-cum-gateway broadcast attack. In both cases, the destination appears down, when it is actually up and alive.

C. Network Error Detection under Windows

In case of the host-cum-gateway broadcast attack, when we victimize the Linux host, the Windows host detects a network error and reports that another system on the network is using the same IP address as that of the Windows host.

D. Inference: Single Host Broadcast Attack

The attack is feasible only under a particular network segment or router since we are manipulating ARP packets, which are not routed outside the home network. An active host seems to be down when it does not reply to ICMP Request packets even after receiving them. This situation occurs since the destination host *C* receives an ICMP Request packet addressed to {C_IP, BDCT_MAC} rather than the expected {C_IP, C_MAC} combination. A host will also appear to be down when ICMP Reply packets are not sent back to the source in spite of having received ICMP Request packets by the destination. This occurs in case of the host-cum-gateway broadcast attack.

Complementary to this weakness is the other (already known and documented) ICMP weakness which enables an attacker to make the source believe that the destination host is down, by forging the IP address of the destination. In other words, this already known weakness makes the ICMP vulnerable by relying on IP spoofing, while this paper talks about using ARP spoofing to achieve similar aftermaths.

E. Inference: Host-cum-Gateway Broadcast Attack

Under this attack strategy, we are completely able to compromise Linux and Windows gateways that receive PING requests from Windows and Linux hosts respectively. In other words, the Windows host was successfully victimized since it *did* receive ICMP Reply packets from the gateway but failed to recognize them and displayed the *Request Timed Out* error, even when the Linux gateway

was up and alive. This scenario suggests that the Linux host is not wise enough since it accepts BDCT_MAC as the destination MAC and sends back ICMP Reply packets to the source. It also suggests that the Windows host is not intelligent enough in the sense that it does not recognize ICMP Reply packets if they are destined to BDCT_MAC, which is different from its own MAC (BDCT_MAC \neq B_MAC).

VI. FUTURE WORK

The eight cases described in the previous section involved one operating system trying to PING another, both being different from one another. We intend to extend our attack strategy between similar operating systems (Windows \rightarrow Windows and Linux \rightarrow Linux PING) and compile these results and inferences.

The network error detected on the Windows host is expected to be generated on multiple Windows hosts by simply poisoning the cache of a single or multiple hosts using the host-cum-gateway broadcast attack module of our implementation. This observation needs to be verified upon implementation.

VII. CONCLUSION

In case of the host-cum-gateway broadcast attack, the PING utility under Windows proves to be unintelligent since it does not recognize ICMP Reply packets unless they are destined to the Windows host MAC address. This is a very controversial situation since one may argue that if it were to accept any other MAC address, other factors may be compromised. However, the whole debate undoubtedly does give rise to a surfacing weakness of the Internet Control Message Protocol itself.

The Linux host may be considered unintelligent as it replies to ICMP Request packets even if they are not destined to the Linux host's MAC address. If a Windows host receives ICMP Request packets destined to anything other than its own MAC, it does not issue ICMP Reply packets. Though this sounds intelligent, it goes without saying that this very host would appear to be down even when it is up and alive, making our attack strategy successful.

REFERENCES

- [1] D. C. Plummer, Nov. 1982. RFC826: *Networking Group: An Ethernet Address Resolution Protocol: Converting Network Protocol Addresses to 48-bit Ethernet Addresses for Transmission on Ethernet Hardware*
- [2] D. Bruschi, A. Ornaghi, E. Rosti, Dec. 2003. SARP: A Secure Address Resolution Protocol – *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC '03)* [Pg. 66]
- [3] S. Cheung, K. N. Levitt, Jun. 2000. A Formal Specification Based Approach for Protecting the DNS – *Proceedings of the IEEE International Conference on Dependable Systems and Networks*, New York, USA [Pg. 641]
- [4] Ettercap Documentation and Tutorials Available: [ettercap.sourceforge.net]
- [5] H. Altunbasak, et. al., Nov. 2004. Addressing the Weak Link between Layer 2 and Layer 3 in the Internet Architecture – *Proceedings of the 29th IEEE International Conference on Local Computer Networks (LCN'04)* [Pg. 417 – 418]
- [6] K. Daley, R. Larson, J. Dawkins, Aug. 2002. A Structural Framework for Modeling Multi-Stage Network Attacks – *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02)* [Pg. 5]
- [7] M. Dornseif, T. Holz, C. Klein, 2004. *NoSeBrEaK: Attacking Honeynets – Proceedings of the 5th IEEE Information Assurance Workshop, 2004*
- [8] M. Zalewski, 2001. Strange Attractors and TCP/IP Sequence Number Analysis Available: [www.bindview.com]
- [9] J. Postel, Sept. 1981. RFC792: *Networking Group: Internet Control Message Protocol: DARPA Internet Program Protocol Specification*
- [10] J. White, Sept. 2000. RFC2925: *Distributed Management Working Group (The Internet Society): Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations*
- [11] R. Dodge, D. J. Ragsdale, C. Reynolds, 2003. Organization and Training of a Cyber Security Team – *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics, 2003*