Leveraging ARP Gratuitous Response for Wireless IoT Network DoS Attacks: An Exploration (2022)

J. Reger

***Abstract*—The IoT is a novel application of often wireless networks and with its inevitability, comes a serious need to examine the security design philosophies and methodologies being implemented and adhered to by software and hardware developers in the space. These networks, in their current form, are woefully susceptible to wireless attacks given the often-low computational capability of individual nodes as well as the wireless nature that many IoT networks implement out of necessity. The main objective of this project is to evaluate a specific attack that IoT networks are susceptible to. This attack leverages the Address Resolution Protocol and specifically the gratuitous ARP response to execute a Denial-of-Service attack on an individual node on the network. This will be done by constructing a simulated network in Docker, with an array of IoT nodes on the network. The connectivity of the nodes will be tested simply by pinging back and forth between nodes to ensure that connections are working as intended. Then, an unprivileged attacking node will join the network and execute the attack. The attack's success will be evaluated by examining the ARP Caches of other nodes in the network, along with observing the failure of pings being sent to the targeted node, indicating a DoS. After this, a possible solution called static ARP entries will be explored. Its strengths and weaknesses will be discussed.**

I. INTRODUCTION

THE rise of the Internet of Things (IoT) has been extreme. It stands poised to explode in popularity and is already doing so with over 10 billion active devices in 2021 with the market's economic impact measured not in billions but in trillions of USD [1]. IoT is already ubiquitous, present in forms in almost every facet of 21st century life. From the home to the workplace to automated factories and even into the government and miliary, IoT devices continue to shape our lives and the future. The data collected by these devices is equally as diverse. It includes weather data, productivity data, and home data like temperature settings, current fridge stock, and smart doorbell video feeds. While some of this data is relatively low priority and harmless, some is quite sensitive.

For example, if your smart home thermostat data were to be publicly available, a bad actor could evaluate your temperature settings over time to determine when you are usually home and when you are usually away. On top of that, they could likely determine when you leave for vacation and using this information, can launch more intrusive attacks which can threaten your home or personal well-being. Thus, some of the data captured and shared amongst IoT devices is of serious concern. In order to ensure the privacy and safety of users and companies of IoT networks, security is a high priority. However, current IoT network device security is woefully inadequate. This is largely the responsibility of the innate heterogeneity of the IoT along with the computational limitations present in many IoT devices [2]. For that reason, this exploration into one possible attack on an IoT network, one which a wireless IoT network is especially susceptible to, was conducted. Understanding the threats to the current state of IoT networking will help to build a better and more robust suite of defenses in tomorrow's IoT devices.

This work was prepared for Bechir Hamdaoui of the EECS Department of Oregon State University for the course ECE 463, Wireless Communications Networks. *Corresponding author: J. Reger.*

J. Reger is an undergraduate student with the EECS Department of Oregon State University, Corvallis, OR 97330 USA (e-mail: regerjacob@gmail.com)

This attack will leverage a vulnerability of the Address Resolution Protocol (ARP). This protocol is used to map physical addresses (MAC Addresses) to Internet Protocol addresses (IP Address). This is critical because MAC addresses are unique and hard coded, while IP addresses are dynamic. They can be assigned or revoked, even changed and reassigned. A single physical device may have multiple IP addresses. The IP address is software based and provides significant functionality to modern networks, but the MAC addresses are still necessary to ensure that the endpoint is the intended endpoint. ARP provides the connection between the two.

On every node in the network there is an ARP cache, which is a storage of all known MAC-IP mappings. This allows the device to look up a MAC address and determine which IP it needs to send the packet with. If no entry is found, the device will issue what is called an ARP Request. This is a special packet broadcast to the entire subnet requesting the IP of a certain MAC address. If this is found by another node or the node with the requested MAC, that node will issue an ARP Response, a special packet sent back to indicate the correct MAC-IP mapping. However, in order to make networks more dynamic, ARP implemented what is called a gratuitous ARP response, which is an unprompted ARP response, essentially an announcement of oneself with a MAC-IP mapping. While this is very useful for nodes entering a network to easily add themselves to all other ARP caches, it can also be utilized maliciously to attack a network in a variety of ways.

II. Problem Statement

For this project, the goal is to simulate a DoS attack on an IoT network by leveraging this gratuitous ARP response. In order to simulate this, the network simulation tool Docker was used. Docker is a powerful containerization platform that enables a user to virtualize a network and many nodes on that network. For this project, a bridge network named NetOne was created with:

```
$ docker network create --subnet 10.2.6.0/24
NetOne --driver bridge
```

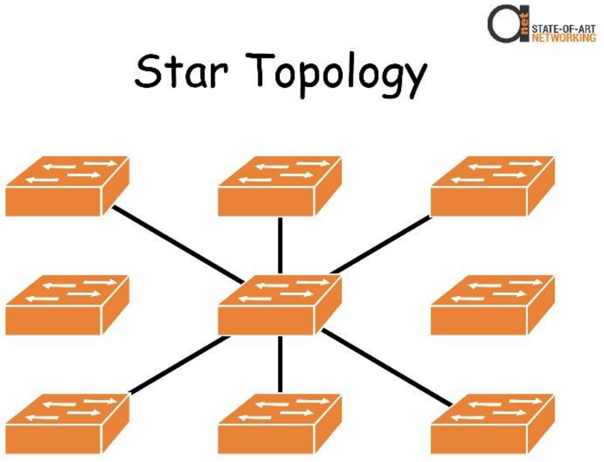


Fig. 1. Star Network Topology diagram [3].

The bridge network driver used by docker is essentially a star network, which is diagramed in Figure 1 above. Here all nodes on the network must route their traffic through a hub or host seen in the center. In the docker bridge network, that is the user's device. However, because we will not be implementing any administrative actions on the docker network, it better resembles a mesh network seen in Figure 2 below where communications are peer to peer where all nodes are also neighbors. This is a network topology often seen in IoT networks, though star networks are also common in the IoT.

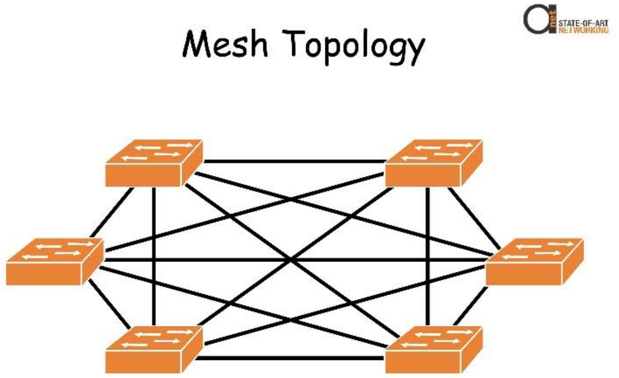
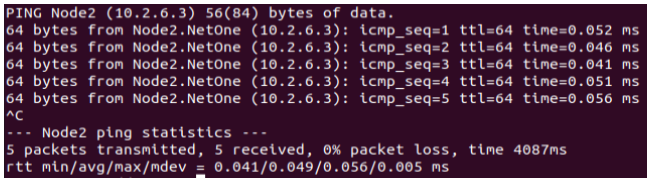


Fig. 2. Mesh Network Topology diagram [3].

Then, eight containers were created and connected to the NetOne network. All eight containers were unprivileged and represent IoT devices connected to this network. One of these will be used as the attacking node, though it receives no special treatment as any other node, highlighting how vulnerable the network is to this attack. These containers were run using the following docker command:

```
$ docker run -it --network NetOne --name Node1
--ip 10.2.6.2 ubuntu
```

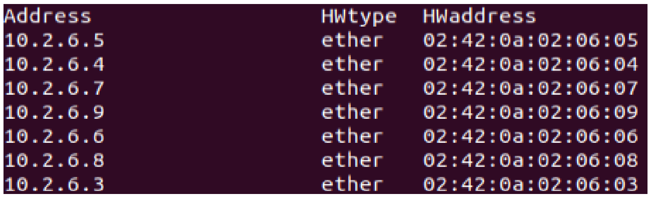
This command was repeated for each of the eight nodes with the names changing accordingly and the IP addresses incrementing as well i.e. Node1 - 10.2.6.2, Node2 10.2.6.3 and so on. In this network, the Ubuntu device running the docker network also acts as a gateway for the network, though it will be hands-off in terms of network administration and nodes will be allowed to freely communicate with each other as they would in an IoT network. To test this network, each node pinged the other nodes to simulate IoT network traffic being exchanged between nodes. The results of one such ping can be seen below in Figure 3.



```
PING Node2 (10.2.6.3) 56(84) bytes of data.
64 bytes from Node2.NetOne (10.2.6.3): icmp_seq=1 ttl=64 time=0.052 ms
64 bytes from Node2.NetOne (10.2.6.3): icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from Node2.NetOne (10.2.6.3): icmp_seq=3 ttl=64 time=0.041 ms
64 bytes from Node2.NetOne (10.2.6.3): icmp_seq=4 ttl=64 time=0.051 ms
64 bytes from Node2.NetOne (10.2.6.3): icmp_seq=5 ttl=64 time=0.056 ms
^C
--- Node2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4087ms
rtt min/avg/max/mdev = 0.041/0.049/0.056/0.005 ms
```

Fig. 3. This is the resulting ping output when pinging Node2 from Node3. All packets are sent and received correctly as expected with 0% packet loss.

Then the ARP cache can be observed from one of the nodes, in this case Node1. Figure 4 shows this ARP cache. As you can see, it correctly maps each IP address to the corresponding MAC address, and therefore is working as intended.



```
Address                  HWtype  HWaddress
10.2.6.5                 ether   02:42:0a:02:06:05
10.2.6.4                 ether   02:42:0a:02:06:04
10.2.6.7                 ether   02:42:0a:02:06:07
10.2.6.9                 ether   02:42:0a:02:06:09
10.2.6.6                 ether   02:42:0a:02:06:06
10.2.6.8                 ether   02:42:0a:02:06:08
10.2.6.3                 ether   02:42:0a:02:06:03
```

Fig. 4. Node1 ARP cache before attack.

For this attack, Node3 will be used as the attacking/compromised node. There is nothing unique or special about Node3, it exists on the same level as every other node on the network. In a real IoT network, this node may be a device of the attacker, or often may be a compromised device that was once a benign member of the network but has become a platform from which the attacker launches their attack. After this, a possible solution will be explored. This is a method called static ARP table entries. These are entries into the ARP cache from Figure 4 that are qualified as static. This means they cannot be changed using an ARP response, but instead must be manually changed by the device itself.

III. Solution Approach

Before static ARP table entries can be explored, first the attack must be conducted. In order to execute this attack, a python script was written which used the Scapy packet manipulation tool to handle the network manipulation. First, the script asks the attacker if they would like to conduct a DoS attack or reset the network to its condition before the DoS attack. This second option was critical for testing the attack often without needing to reset the network. If the attacker selected the DoS attack they would be prompted to enter an IP address that they wish to DoS.

Next, the script iterates the IP addresses in the network and forges for each a new ARP response packet with the destination set to the IP, and the source IP set to the target IP address. This makes the packet appear as if it is coming from the target, when it is in fact coming from the attacker. By default, Scapy attaches the senders MAC as the source MAC address. By pairing the target IP with the sender's (attacker's) MAC address, this packet will effectively map the target IP address to the attacker's MAC address in every node's ARP cache. That way, any time a node on the network attempts to send data to the target, it will instead be routed to the attacker at which point it is dropped, effectively cutting off connection to the target. This can be seen in Figure 5. Here the attacker, Host C, has sent a gratuitous ARP instructing nodes to route traffic headed for IP 10.10.10.20 to the MAC address cc:cc:cc:cc:cc:cc, which is the MAC of the attacker. Now all traffic intended for Host B will instead be routed to Host C. In our network this means all traffic from any of the other nodes bound for the target node will instead be routed to Node3 (the attacker).

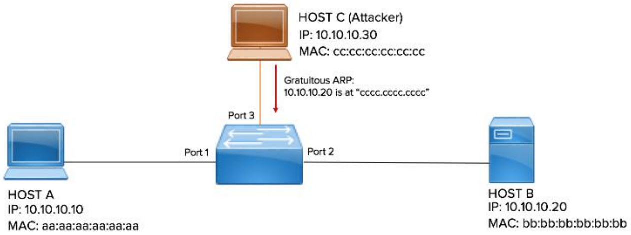
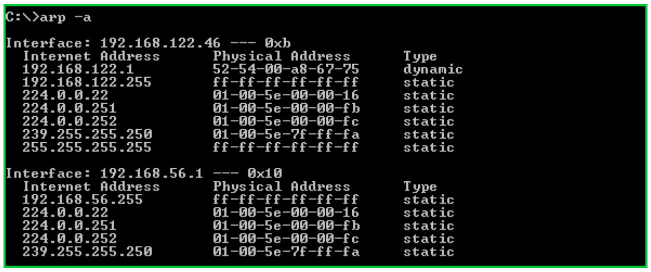


Fig. 5. Gratuitous ARP Poisoning diagram [4].

The script continues to emit these forged gratuitous ARP responses continuously so that any attempt by the target node to reassert its MAC-IP mapping would be immediately drowned out. This can be cancelled by the attacker using a keyboard interrupt but will otherwise continue indefinitely. An alternative form of a DoS attack using a similar method would be to route all traffic in the entire network through the target node which would overload the node's reception capabilities and force a restart, shutoff, or battery depletion.

The explored solution to this is called static ARP table entries. An example of some of these can be seen in the below image, Figure 6. These are ARP cache entries that do not expire and cannot be changed via an ARP response. They are hard coded into the device and are a permanent mapping of an IP address to a MAC address. They can only be modified manually on the device itself. These are normally used for ARP entries that the user does not want to ever expire temporally. However, they can also be utilized in order to prevent all forms of ARP poisoning attacks including the explored DoS attack in this paper.



```
C:\>arp -a

Interface: 192.168.122.46 --- 0xb
  Internet Address      Physical Address      Type
  192.168.122.1         52-54-00-a8-67-75     dynamic
  192.168.122.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0x10
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
```

Fig. 6. Static ARP Cache Entries [5].

Static ARP table entries are probably the most outright effective solution against ARP poisoning. Short of gaining administrative access to the device itself, there's really no way to attack it that we know of. It entirely defends against this style of attack. The downside however comes from laboriousness of properly implementing total static ARP table entries. Having a completely static ARP cache also makes the network significantly less dynamic. Nodes cannot freely enter and leave the network, because in order to do so, every node in the network must update its ARP cache manually. This is an extremely laborious process for the network administrator and is often outright preventative in some networks.

IV. Performance Evaluation

To evaluate the project, first an examination of the attack and its effect on the network. Then, that will be followed by an evaluation of the effect of static ARP entries on the ARP cache when faced with the same attack.

A. The Attack

First the attack was executed. The script was run from the attacking node, Node3. Next, the ARP cache of the other nodes on the network was viewed. Because the gratuitous ARP responses with forged sources were being constantly sent out over the network, the ARP caches of these nodes read that the hardware MAC address of the target IP address, 10.2.6.3, had changed since the last look at the ARP table in Figure 4. It now read the same exact MAC address as the attacking node, Node3.

This is because the forged ARP response packet was linking the attackers MAC address to the target IP address, so that any traffic intended for the target would instead be routed to the attacker. At this point, an attacker could form another spoofed connection with the target and launch a man in the middle attack, but in this case the attacker will just drop all traffic and use this method to conduct a DoS on the target.

Not only has the ARP cache changed for Node1, but it has also changed for all other nodes on the network. This means that any and all traffic on the network will be rerouted to the attacker to be dropped, though if the attacker wanted to, they could selectively DoS. This may make the attack more difficult to detect. While it is a very good indication that the ARP caches have changed, in order to fully test, actual packet communication must be simulated.

In order to do that, multiple different nodes attempted to ping the target node. When the pings were attempted, they would always and inevitably time out. This indicates that while the packets had been transmitted, no responses were coming back. This is because the packets were being transmitted to the attacker instead of to the intended target node. They were then dropped, so no responses were received by the pinging node. This results in 100% packet loss by all nodes to the target node, a perfect DoS.

The target node is still fully capable of transmitting data to other nodes but is incapable of receiving any itself. Through some modifications, the attacking script could also reroute the entire ARP cache of the target node so that no outgoing packets from the target node reach their intended destinations either, however it is clear the possible damaging effects of this vulnerability.

This is only further exacerbated by the wireless nature of real IoT networks. They are even more susceptible to such attacks, because nodes can enter a network simply by entering the area and can begin transmitting packets without physically wiring into the network. Propagating such packets through a wireless network would be even easier, and this attack shows just how at-risk IoT devices are in their current state. This is because ARP control packets remain unencrypted so even if the IoT device utilizes encryption for its data transmission, it is still at risk for this DoS attack.

B. The Solution

The proposed solution, as read above, is static ARP table entries. These table entries were manually entered into the nodes of the network to statically map MAC addresses to IP addresses for all other nodes on the network. This is a laborious process. However, when the attack was again conducted, none of the ARP caches changed at all. This indicates that the solution was probably a success. The target IP address still mapped to the correct MAC address, which indicates that the traffic should be routed without issue.

In order to test this, just like before, pings were used to simulate packet communications between the nodes. When the target node was pinged by any of the other nodes on the network, they received every ECHO they requested, with a packet loss of 0%. This shows that on the network, no DoS has occurred on the target node, it is fully working.

Therefore, we have evidence to suggest that static ARP table entries represent a complete defense of an IoT network against ARP poisoning-based attacks. However, there are some significant downsides to this implementation. Current Wi-Fi networks often have a limited IP address range, so IP addresses may be reused for new nodes entering the network. Any time this occurred, like a certain device needed to move to a new IP address, each other device would need to have its ARP cache manually updated by a network administrator.

This would be prohibitively difficult for many IoT networks but may possibly be a seriously robust solution for certain static IoT networks that have the same nodes and are not designed to be connected to by any other node. This appears to be the most valid use-case for this solution. Further work may be done to refine the idea of static ARP table entries or perhaps construct a bit of protection on the ARP cache in order to protect against an ARP poisoning attack without sacrificing the plasticity of the network. However, substantial changes to the ARP protocol would certainly need to be made. Maintaining backwards compatibility would be a critical component of any such future work as the number of IoT devices currently in use is extreme.

V. Conclusion

This project began by identifying a weakness of modern IoT networks, security. Reasoning for the importance of this weakness, the data sensitivity, was then identified. A specific attack that IoT networks are vulnerable was then identified, the gratuitous ARP response DoS attack. To test this attack and any possible solutions, the IoT network environment was simulated through the use of the Docker tool.

Docker simulates the network as wired, though because ARP functions identically whether wired or wireless, I was sure that for this use-case, this model of simulation as sufficient. Each node on the network was simulated as a Docker container running a barebones version of Ubuntu. Each was given a specific IP address and was introduced to every other node on the network to ensure that the ARP caches were populated.

Then the attack was conducted. The ARP caches of all nodes on the network were poisoned and all traffic intended for the target node was rerouted to the attacking node where it was dropped. This was reflected in the changed caches of each node on the network, indicating that all traffic bound for the target IP address would be sent to the attacker's hardware MAC address. Then, a solution was devised. Static ARP table entries represented a method which may fully alleviate the threat of ARP poisoning-based attacks on an IoT network. It was manually implemented on the simulated network and resulted in a perfect defense against the attack.

Through this project, I have learned about the workings of ARP as a protocol, why it is necessary, and how it can be exploited. I learned one such exploit. I learned more about how to simulate a network using Docker and learned more about packet manipulation through Scapy. In order to launch the attack, I needed to learn about the structure of an ARP packet, how to indicate its identity as a response packet, and how to populate the relevant variables in the packet to achieve the desired result. I learned a great deal about how IoT and other Wi-Fi based networks manage their traffic. I then learned one possible solution to the problem of traffic rerouting using static ARP table entries. I learned the benefits and drawbacks of such an approach, and why it is not used widely in its current form.

The greatest challenge I faced when executing this project was getting the attacking script to work correctly. In order to do this, it needs to iterate the network IP addresses sending ARP responses to everyone in the entire network. Learning how to properly craft an ARP response along with bug fixing and general ease of use improvements to the script took the longest. Even though the script itself was not all that complicated, my relative ineptitude with Scapy and even python more generally made this the most challenging part. However it also represents where I feel I have learned the most. I feel comfortable understanding network ARP routing now, I understand the structure of an ARP packet, and I feel much more comfortable leveraging a tool like Scapy to test malicious attacks on a network.

References

Basic format for periodicals:

[1] B. Jovanovic, "Internet of Things statistics for 2022 – Taking Things Apart", *DataProt*, May 13, 2022, Accessed June 2, 2022, [Online] Available: https://dataprot.net/statistics/iot-statistics/

[2] R. Kollolu. "A Review on Wide variety and Heterogeneity of IoT Platforms", *The Int. Journal of Anal. And Exper. Modal Analysis*, vol. XII, no. 1, pp. 3753-3760. January 29, 2020.

[3] "Network Topologies: Star, Mesh & Hybrid". Ad-net.com. https://www.ad-net.com.tw/network-topologies-star-mesh-hybrid/ (accessed June 2, 2022).

[4] "Dynamic ARP Inspection". Documentation.meraki.com. https://documentation.meraki.com/MS/Other_Topics/Dynamic_ARP_Inspection (accessed June 2, 2022).

[5] I. Baydan. "Display, Add, and Remove Arp Information with Windows Arp Command". POFTUT.com. (accessed June 2, 2022).

[6] W. Gao et al., "ARP Poisoning Prevention in Internet of Things," *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, 2018, pp. 733-736, doi: 10.1109/ITME.2018.00166.

[7] R. Grimmick, "ARP Poisoning: What is it and How to Prevent ARP Spoofing Attacks,". *Varonis*. 2021. https://www.varonis.com/blog/arp-poisoning

[8] E. Harmoush. "Gratuitous ARP" *Practical Networking*. 2021. https://www.practicalnetworking.net/series/arp/gratuitous-arp/