Detecting and Combating ARP Spoofing

Chai Ming Xuan
National University of Singapore
mingxuan@u.nus.edu

Er Xue Hui
National University of Singapore
xuehuier@u.nus.edu

Wu Wenqi
National University of Singapore
wenqi.wu@u.nus.edu

Zhu Chunqi
National University of Singapore
chunqi@u.nus.edu

ABSTRACT

In light of the 'Sparkle Vulnerability'[1] incident that occurred early in 2016, our team has been particularly interested in preventing Man-In-The-Middle (MITM) attacks, since that is one of its primary attack vectors.

While internet security has generally increased over the past few years, there has been little improvements made to low-level protocols on the Link and Physical layers of the Open Systems Interconnection (OSI) model. Protocols such as Address Resolution Protocol (ARP) are still stateless, making them vulnerable to attacks. While solutions such as Secure ARP (S-ARP) exist, adoption of such solutions remain slow due to their incompatibility with existing protocols. The vulnerability in both ARP and DHCP makes carrying out MITM attacks via ARP spoofing or DHCP spoofing trivial.

Thus, our research focuses on a solution to detect and prevent ARP and DHCP Spoofing while maintaining compatibility with existing protocols. This will greatly limit the MITM attack variants an attacker can consider.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General - Data Communications, Security and Protection

; D.4.6 [**Security and Protection**]: Authentication, Verification

General Terms

Network Security, Intrusion Detection System (IDS), Address Resolution Protocol (ARP), spoofing

[1] `https://sparkle-project.org/documentation/security/`

Keywords

ARP spoofing, ARP Poisoning, network security, attack, MITM, IDS, ARP protection

1. INTRODUCTION

The Address Resolution Protocol (ARP) is an important protocol in computer network communications.
In order for communication to occur between two computers, the computer will need to know two things - the IP and MAC address of the other computer. As IP addresses can be resolved using a DNS lookup, the ARP is used to determine the MAC address of the other computer given an IP address. Without the ARP, it would be difficult for computers to communicate with each other, as an IP-to-MAC address mapping needs to be established.

Traditionally, the ARP cache has been an easy target for ARP attacks like ARP spoofing. Even until today, it is still relatively easy to carry out an ARP spoof, and it remains rather difficult to defend against it.

There is a lack of viable solutions to protect the ARP cache because of the way it is designed. In fact, ARP spoofing is easy to carry out over a wireless network if it is not protected by WPA-Enterprise encryption. Over a Local Area Network (LAN), such attacks require a wiretap if the network uses switches instead of hubs.

Some common attacks that can occur after an ARP Spoof are MITM, Denial-of-Service (DOS) and session hijacking. In such cases, the three basic principles of security - Confidentiality, Integrity and Availability - may be violated. We will be illustrating how an ARP Spoof occurs in Section 2.

Programs which users can use to carry out such an attack include but are not limited to, 'Ettercap' and 'Cain and Abel'. As these programs are open source and documented well, almost anyone can easily carry out ARP Spoofing with some basic technical knowledge.

2. BACKGROUND

In order to understand how ARP spoofing occurs, we will need to cover the basics on how computers communicate with each other. In this section, we will be explaining how computers do so, and introduce the basic notions of how an ARP spoof is carried out.

2.1 Models and Abstractions

In computer networking, the communication protocols are categorised and organised into different layers. There are 2 commonly used models - the TCP/IP model and the OSI model.

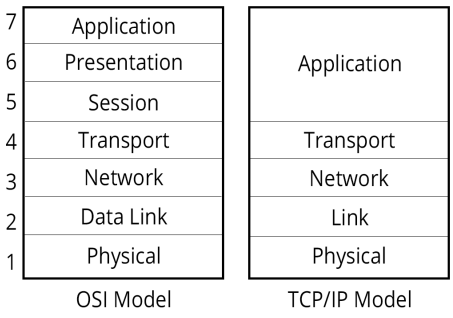
In the TCP/IP Model, there are 5 layers of abstraction:

1. Application Layer
2. Transport Layer
3. Network Layer
4. Link Layer
5. Physical Layer

DHCP is an application layer protocol and ARP is a link layer protocol.

In the OSI Model, there are 7 layers of abstraction:

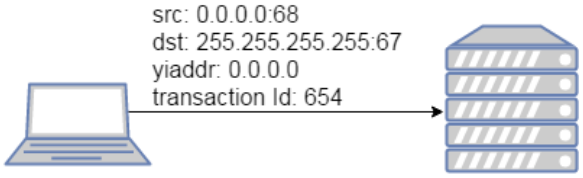
7. Application Layer
6. Presentation Layer
5. Session Layer
4. Transport Layer
3. Network Layer
2. Data Link Layer
1. Physical Layer



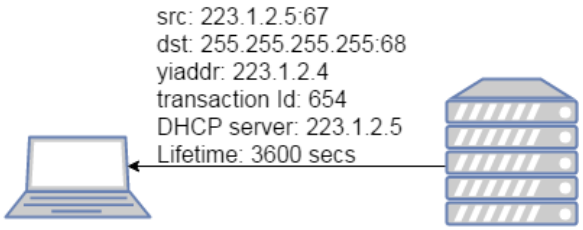
2.2 Computer Communications

In a typical modern computer, all IP addresses are resolved dynamically through the use of the Dynamic Host Configuration Protocol (DHCP). This is carried out in the following steps[3]:

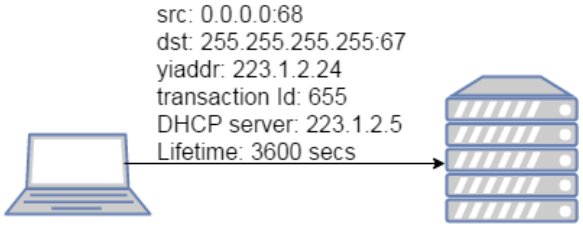
1. DHCP Discover:
The new host broadcasts a DHCP discover message, where 255.255.255.255 is a typical broadcast address.



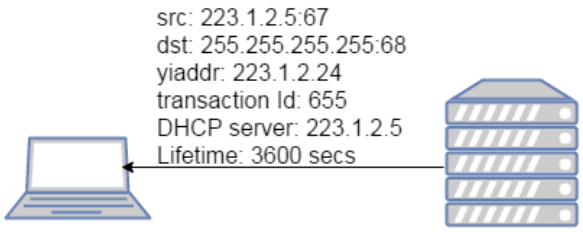
2. DHCP Offer:
The DHCP server responds with a DHCP offer.



3. DHCP Request:
The host selects from the offers and sends a DHCP request.

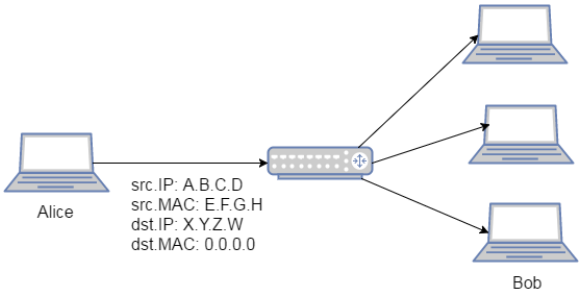


4. DHCP ACK:
The DHCP Server confirms the requested parameters.

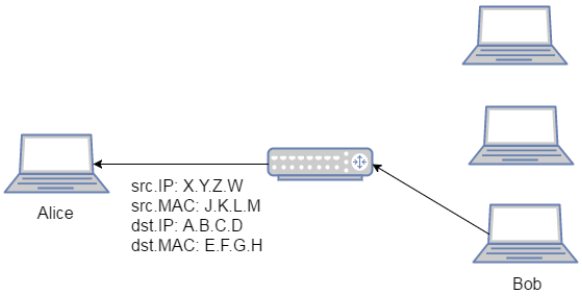


After that, assuming everyone has already had their IPs assigned to them via DHCP, suppose the the user Alice wishes to send some information to Bob. The following steps are then carried out:

1. ARP Request: Alice's computer sends out an ARP request to find out which MAC address has Bob's IP.



2. When Bob's computer receives the request, he sends back an ARP response packet.



3. Alice gets the response and stores the corresponding IP-to-MAC entry into the ARP cache.

2.3 Poisoning the ARP Cache

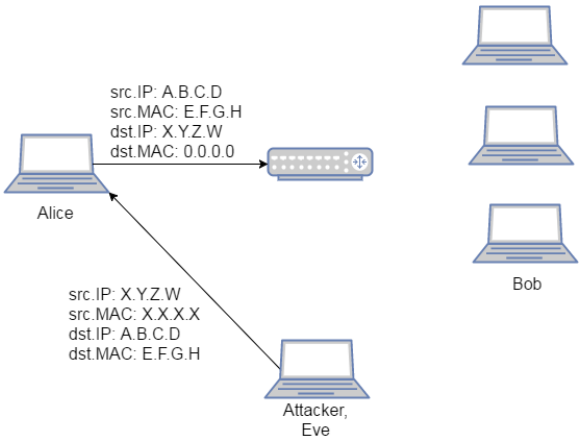
For ARP spoofing to work, the attacker typically has to be in the same network as his victim[4]. Sniffing is carried out to first determine the victim's IP address on the network. This can be done using a network sniffer such as Wireshark.

There are 3 common ways to poison the ARP cache. The first is to send a broadcast request, the second is to send multiple responses, and the third is an unsolicited response. In this paper, we will only cover the second method[6].

If an attacker, Eve, wishes to carry out ARP spoofing, this is typically what happens:

1. Alice's computer sends out an ARP request to find out what is Bob's MAC address.

2. Before Bob's computer can reply, the attacker, Eve, sends a spam of packets to Alice's computer, claiming to be Bob. The ARP cache then becomes poisoned.

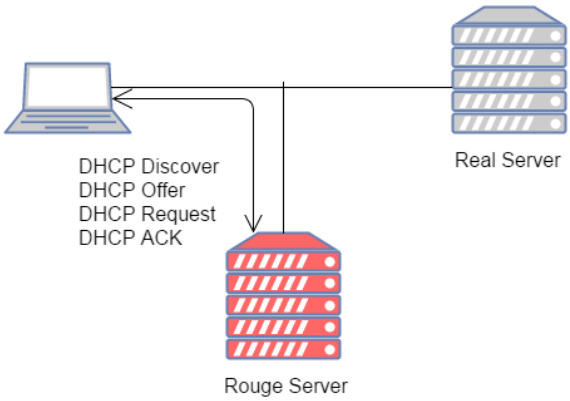


In this case, there is a race condition between Eve and Bob to send a response to Alice. If Eve succeeds in responding to Alice before Bob can do so, Alice's ARP cache becomes successfully poisoned.

In addition, to keep Alice's ARP cache poisoned, Eve will continue to send out fake ARP packets at regular intervals, just before Alice's ARP cache flushes the spoofed entry. This is because ARP caches have different time to live(TTL) timings, which it is dependent on the type/model of host or router. The TTL can range from 60 seconds to 20 minutes.

2.4 DHCP Spoofing

In DHCP spoofing, there is a race condition between the rouge server and the legitimate server to respond to a user's DHCP Discover broadcast. Should the rouge server respond to the user first, the user then becomes subjected to an MITM attack.



The solution to this problem is to implement DHCP snooping.

3. CURRENT SOLUTIONS AND MITIGATIONS

There are currently many solutions in the market to combat ARP Spoofing. However, many of these solutions are described to have major issues. From [5], [1] and [2], these are the summaries and drawbacks of current implementations in the market:

1. Use of Cryptographic Techniques
Examples: S-ARP

This method suggests that the ARP protocol be redesigned with cryptographic measures in place. However, implementing such cryptographic measures severely degrades the runtime performance of the ARP protocol.

2. Passive Detection
Examples: Agnitum Outpost Firewall
Such methods only detect ARP spoofing but are unable to do anything to correct it. In addition, this method assumes that initial setting of ARP entries are correct, as the initial database is used for comparing with an incoming ARP packet to check if it has been spoofed.

3. Kernel-based patches
Examples: Anticap, Antidote
A patch is given to fix the OS Kernel. However, this may cause compatibility issues.

4. Making ARP entries static
In this method, ARP entries are configured to be static. In this case, DHCP will not be used to resolve local IP addresses for a computer. However, it will not be very user-friendly for standard office workers to configure, and will be very difficult for network administrators to manage IP-to-MAC mappings in a huge company.

4. GOALS

In taking up this project, we hoped to achieve the following goals:

1. Provide users with a means to actively combat ARP spoofing
2. Make any network more secure.
3. Provide a GUI for users to see what is happening in their network in real-time.

5. PROOF OF CONCEPT

5.1 Implementation System

Operating System:
OSX El Capitan, Arch-Linux [Manjaro 15.12 Capella]

We created a software product called 'Spoof Alert' to monitor the computer's network traffic and alert the user should there be any suspicious packets that hint at ARP or DHCP spoofing. Spoof Alert is implemented using Python 2.7 with scapy and tkinter as the main library dependencies.

5.2 Assumptions

There are many variants of ARP spoofing. In order to deliver a focused solution, we have made certain assumptions about our environment. Most of these assumptions are still likely hold in actual cases.

1. Attacker's computer has a normal network stack.

2. All computers on the network have at least one TCP port open, since our solution requires computers to reply to TCP/SYN packets.

3. All computers on the network have a TCP/IP network stack up and running.

4. The attacker will only ARP spoof an existing IP address. He will not spoof IP addresses which have not yet been issued by the DHCP server.

5. At the start up of our application, the ARP cache of all computers are valid and no attacks should be taking place.

5.3 Overall Architecture

Spoof Alert's architecture shown in Figure 5.1.

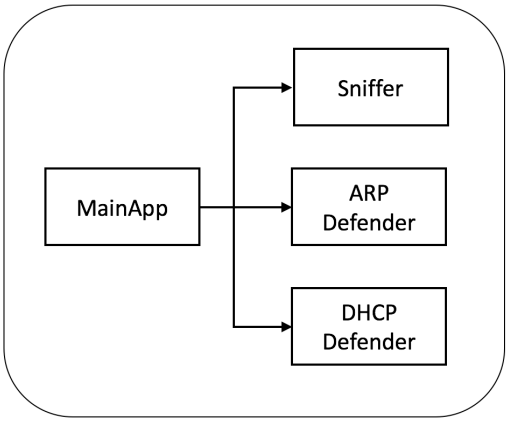


Figure 5.1: High level view of Spoof Alert's architecture.

The sniffer runs on a separate thread from the main application and sniffs and filters out ARP and DHCP packets which are present in the network. Any packets the sniffer captures will be passed to a defender module (currently we have two such modules implemented), depending on the type of the packet. The defender module does a thorough inspection of any received packet (further elaborated in section 5.5).

In future implementations, additional defender modules can be added to defend against a wider range of MITM attacks.

5.4 The Attacker

As we wanted to focus solely on developing the solution, we used Ettercap to test the performance of our software instead of writing our own attacker.

5.5 The Defender Module

As of now, the ARP Defender and DHCP Defender modules have been implemented for Spoof Alert.

5.5.1 ARP Defender module

Upon receiving an ARP packet that is captured by the sniffer, the ARP Defender first checks that the packet's ARP header MAC and IP addresses for sender are consistent with that of its Ethernet headers. If there is a mismatch in headers, we conclude that the packet is invalid, and an ARP spoofing attack is potentially taking place.

If the headers match, we proceed to check if the packet is consistent with the entries in our ARP table, i.e. there is an entry in the ARP table which has the same sender IP and MAC addresses as the packet. If so, the packet is deemed

to be a valid one, with the assumption that our ARP table has not been poisoned.

If the packet is inconsistent with the ARP table, Spoof Alert will perform an active check by sending a TCP SYN packet to the sender's IP address as stated in the packet. Assuming that there are spoofers in the network, they will not have the IP address as stated in the TCP SYN packets. The spoofers therefore will not respond to the TCP SYN packet as the packet will be discarded by their network stack. Only the real host with the legitimate IP address will respond to the TCP SYN.

In the absence of replies after the TCP SYN packet is sent out, the ARP packets previously received will be classified as a spoof packet. A summary of the ARP defenders process flow is shown in Figure 5.2.

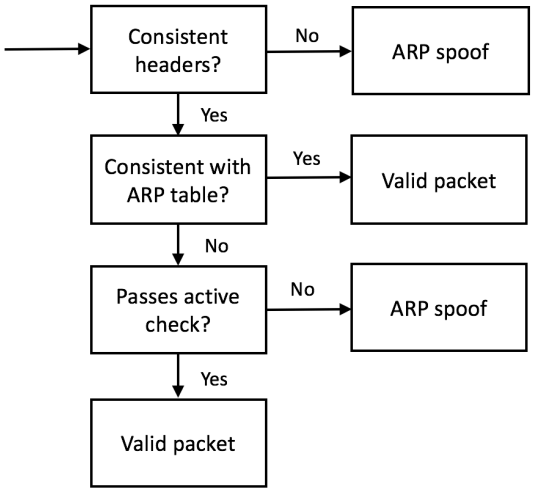


Figure 5.2: Process flow of the ARP defender on receiving a packet.

5.5.2 *DHCP Defender module*

The DHCP Defender seeks to detect the presence of rogue DHCP servers in the network. While DHCP snooping allows the sending of DHCP offers by servers connected to a trusted port, this is only possible in a switched network.

To allow detection of rogue servers in a wireless environment, Spoof Alert requires the valid DHCP servers in the network to have their IP and MAC address added its white-list. This requires the DHCP servers to have reserved static IP addresses in the network. Any computer that is not on the white-list but sends a DHCP offer will be treated as a rogue DHCP server.

5.6 The GUI

Spoof Alert's view is split into several tabs. Below are some screenshots of our early prototype.

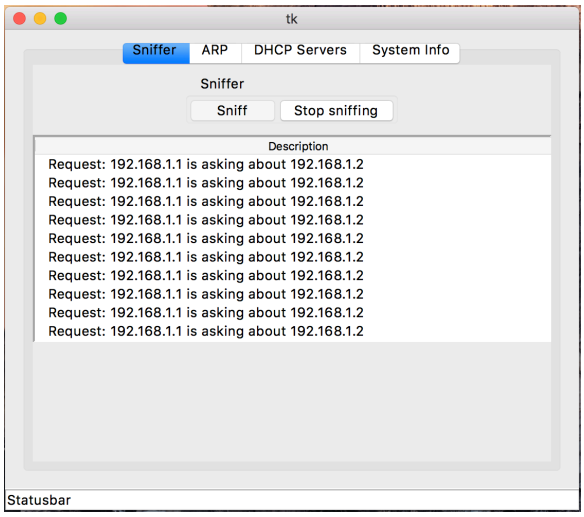


Figure 5.3: In the Sniffer tab, the user can start the sniffer for Spoof Alert. All packets captured by the sniffer will be displayed here. Upon the detection of a suspicious packet, the user will be prompted by the application.

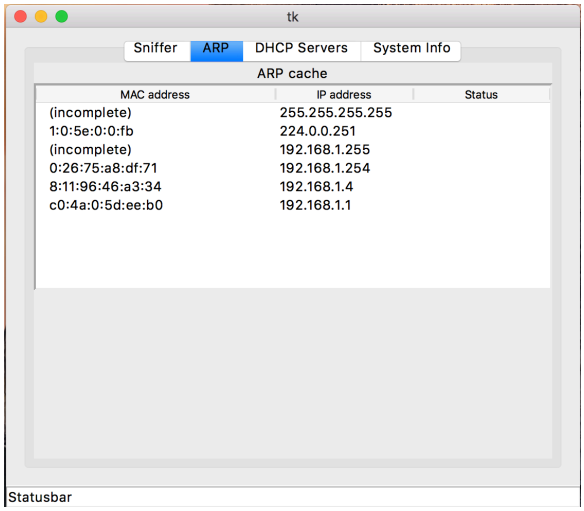


Figure 5.4: In the ARP tab, the user can view the current entries in his computers ARP cache.

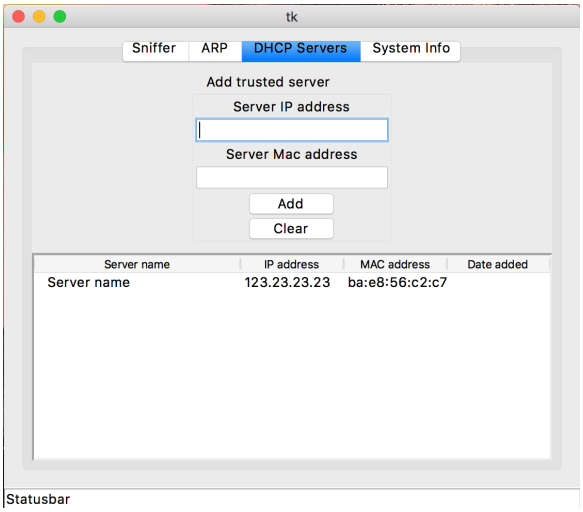


Figure 5.5: In the DHCP servers tab, the user can add or remove DHCP servers from the white-list. If users wish to remove servers, they can only clear the entire list of servers.

5.7 Python Code

The code for our implementation can be found at `https://github.com/cs3235group3/app`

6. OUR SOLUTION VERSUS CURRENT SOLUTIONS

In contrast to current available solutions, we tried to adopt an active method of detecting ARP spoofing, and avoid the weaknesses of current solutions.

1. Secure ARP
Secure ARP (S-ARP) is an extension to the current ARP that allows authentication of ARP replies. However, this requires an additional header field to be added to the ARP packet and the network has to be modified to run authentication checks of S-ARP packets in order for this protocol to be supported.
Spoof Alert can be run from multiple client machines in the network and does not require any additional support.

2. arpwatch
arpwatch is a computer software tool used to monitor ARP packets in a network. Upon detecting that a MAC-IP address pairing has changed, arpwatch will send an email to the administrator as an alert to possible ARP spoof attack. This may result in false positives in an environment where IP addresses are assigned dynamically.
Upon detecting that a pairing has changed, Spoof Alert does not sound off the alarm immediately but sends a TCP SYN packet to the sender. If a response is given, it can be assumed that the packet is valid.

3. Static ARP
Static ARP will prevent all variants of ARP spoofing.
However, this does not scale well in an large environment and gratuitous packets may not be supported by the network.

7. PERFORMANCE

We tested the performance of our implementation using the Python library Scapy. The results are presented in Figure 7.1.

Experiment Parameters	Outcome
ARP response packet with inconsistent sender IP and MAC addresses in ARP and Ethernet headers	ARP spoof alert
ARP response packet with consistent headers that is consistent with the ARP table	Valid
ARP response packet with consistent headers that is inconsistent with the ARP table	ARP spoof alert
DHCP offer packet from white-listed computer	Valid
DHCP offer packet from non-white listed computer	DHCP spoof alert

Figure 7.1: Results from various experiments

8. LIMITATIONS AND FUTURE WORK

Our project has several issues which can be improved on.

8.1 Limitations

Currently, our tool only sniffs the network but does not prevent the addition of spoofed ARP pairings to the computers local ARP cache. If the attacker sets up a server on his attacking machine that responds to TCP SYN packets, it will bypass our detection system.

8.2 Future Work: GUI

1. A sleeker and more minimalistic UI will be implemented.

2. Support for removal of a single DHCP server entry from white-list.

3. Support the add-on of custom defender modules as well as allowing the user to control the packets to capture.

4. Come up with a less intrusive way of alerting the user of possible spoof attacks in the network.

5. Prevent the addition of spoofed ARP pairings to the computers local ARP cache.

6. Implementation of a Graph interface for the user to monitor the ARP packet flow for his computer.

7. Various UI Bug fixes

8.3 Future Work: Backend Code

1. No Support for Network Encrypted with WPA-Enterprise:
Our solution will not work on a network that uses WPA-Enterprise level of encryption. This is because the structure of WPA-Enterprise is such that each user can only see his incoming or outgoing network connections.

2. Assumes the User is on an Insecure Network:
Our solution assumes that the user does not have any form of defence installed on his computer. (eg. no firewall that can prevent ARP spoofing, a network that does not use any enterprise level encryption, etc.) However, it is more likely

that users have at least some kind of basic protection system on his computer.

3. Tested Only on Wireless Networks:
The attacks have only been tested to work for various users across the wireless network which uses no encryption, WEP, WPA, and WPA2. While it might work on a network that uses a hub, it might require a wiretap for a switched network.

We hope to improve our solution for a more varied set of systems in the future. Furthermore, if we are able to tell who our attacker is, we hope to implement a more aggressive method to counter their ARP attacks, such as spoofing them back.

9. CONCLUSION

Through this project, we have learnt that ARP Spoofing is not easy to defend against. Even though there are solutions on the market that have been out for a while, uptake of these market solutions have been slow because of various reasons, such as incompatibility with existing work flows.

In fact, it might be difficult to get people to improve and change low level protocols, such as DHCP and ARP, to use new standards. This is because such low level protocols are primarily designed for efficiency over security, and also because there is a lack of emphasis on such low level protocols. From Kaspersky's Real-Time Cyber Threat Map[2], we can see that network attacks are usually not among the top problems detected by Kaspersky's products. Thus, it makes sense that less emphasis will be placed on securing low level network protocols such as the ARP.

As we have realised through the development of our Python code, unless one is using a network that has been encrypted by WPA-Enterprise, it is otherwise easy to fall prey to MITM attacks.

Furthermore, as active detection of ARP spoofing is not widely implemented yet, our solution may not be entirely foolproof. Thus, we will need to put in more work and testing to make the solution more viable for a wider variety of situations.

10. ACKNOWLEDGEMENTS

First and foremost, we would like to thank Prof. Hugh Anderson for his guidance and patience with us throughout the semester. Our initial project was to investigate the hacking of Hello Barbie. However, we eventually decided to change the topic for various reasons, and he very kindly allowed us to do so. This is despite the topic being changed quite late in the semester, and on top of it, was self-proposed.

Our thanks also goes out to him for loaning us the Hello Barbie even though we eventually dropped the project.

In addition, our appreciation also goes out to our friends who have given us technical advice, as well as loaned us various items for our project, such as a DLink Dir-825 router which runs on DD-RWT, so that we could work on the project in school.

[2] `https://cybermap.kaspersky.com/`

11. REFERENCES

[1] N. Behboodian and S. A. Razak. Arp poisoning attack detection and protection in wlan via client web browser. pages 20–24. International Conference on Emerging Trends in Computer and Image Processing 2011, December 2011.

[2] G. Kaur and J. Malhotra. A review of latest techniques to secure wireless networks against arp poisoning attacks. *International Journal of the Future Generation Communication and Networking*, 8(2):225–232, 2015.

[3] J. F. Kurose and K. W. Ross. *Computer Networking - A Top-Down Approach*. Pearson, USA, 2013.

[4] A. Ornaghi and M. Valleri. Man in the middle attacks. Blackhat Conference, Europe, 2003.

[5] V. Ramachandran and S. Nandi. Detecting arp spoofing: An active technique. pages 1–13. Cisco Systems, Inc. Bangalore India, Indian Institute of Technology, Guwahati, Assam, India.

[6] R. Wagner. Address resolution protocol spoofing and man-in-the-middle attacks. SANS Institute, August 2001.