Hacking Wireless

David Goehring,Daniel Martelly,Viet-Tran Nguyen and Evangelos Taratoris
{dggoeh1, martelly,viettran,evtara}@mit.edu

May 14, 2014

Abstract

Wi-fi networks have been used for decades and they are widely used today. However, intrinsic security problems of Wi-fi Networks create possibilities for malevolent manipulation by third parties. Man in the middle attacks are not only possible, but in some cases, they are also very easy to implement. This paper will focus on demonstrating a specific method to manipulate Wi-fi networks. Some background and relative information regarding network attacks will be presented first. Then, a detailed illustration of our own implementation will follow and finally, there will be a section on how to protect oneself against this kind of attacks. A common theme throughout this paper is simplicity of implementation for the average user.

1 Introduction

Our purpose with this paper is to illustrate successful implementation of a man in the middle attack over a Wi-fi network. Using a router and DNS server we manage to show how to gain access to sensitive user information.

Before we explain our own methods, we will present information about man in the middle attacks that are conceptually similar to ours ,but actually differ in implementation. Their strengths will be mentioned, as well as ways to prevent those sort of attacks.

Following this, we will focus on our version of man in the middle attack. We want to illustrate how accesible and easy our methods and implementation are for the average person to use. In this way, we hope to raise awareness for the security issues of Wi-fi networks.

Finally, we want everyone to be able to securely navigate through Wi-fi networks, and therefore we provide some information on how to do this and avoid malicious attacks. If we are allowed a motto for our project, a guiding principle sort to speak, is that we wanted to make all the implementation steps easy enough "for our grandmothers to use."

2 Background

In this section we provide some background regarding specific known security issues of Wi-fi networks in general. Moreover, we give a high level description of the operation of the various network protocols that we will encounter throughout the description of our implementation.

2.1 Security Concerns of Wi-fi networks

Wi-fi networks are widely used today. Indeed, it is hard to imagine being in a city and not having a Wi-fi hotspot within 100 yards. However, despite the fact that they are so common and easy to use, Wi-fi networks exhibit a range of security vulnerabilities. Some of those are due to defective protocol design, and some are due to user mistakes, however usually it is a combination of both. We present some of these security concerns here:

- Some Wi-fi networks are totally unprotected, in the sense that they provide no encryption of the data transferred through the network. However, those cases are almost pathological nowadays as most of the Wi-fi networks do indeed use some sort of security/encryption.

- Even if some kind of encryption is used, it may be very low security and therefore easily malleable. Take for example the protocol WEP(Wired Equivalent Privacy). It utilized the RC4 cipher, which is a stream cipher. It is essential, for such ciphers,that the key is always refreshed, if we want to ensure confidentiality. In order to achieve this, the protocol used an IV of 24 bits. With such a low number of bits for the IV, there is a $50-50$ chance that the IV will be repeated after approximately 5000 packets sent. Therefore a related key attack was possible and indeed ocurred a number of times before WEP was replaced by a new protocol like WPA 2.
- Even when a strong encryption protocol is utilized, it is possible to use packet sniffers such as Wireshack, in order to monitor the traffic that goes through the network.

2.2 DNS

The Domain Name System (DNS) is a naming system for computers that are connected to the Internet or in private networks. It associates hard to remember IP addresses, with easy to remember domain names.[5]

2.2.1 Domain Hierarchy

In order to explain what a domain is and how DNS resolves domains, take www.google.com as an example. In this name the elements "www", "google" and "com" are called labels.

The hierarchy of domains descends from right to left. The leftmost label is the top-level domain (TLD). Each label to the left specifies a subdivision, or subdomain of the domain to the right. For example: the label "google" specifies a subdomain of the "com" domain, and "www" is a sub domain of "google.com". This tree of subdivisions may have up to 127 levels. When a DNS server tries to resolve a name, it starts from the top-level domain and moves to lower level domains progressively until it finds the domain requested or return a "no such domain" error.

2.2.2 Name Resolution in DNS

DNS is maintained by a distributed database system, which uses the client-server model. The nodes of this database are the name servers. Each domain has at least one authoritative DNS server that publishes information about that domain and the name servers of any domains subordinate to it. The top of the hierarchy is served by the root name servers, the servers to query when looking up (resolving) a TLD.

An **authoritative** name server is a name server that gives answers that have been configured by an original source, for example, the domain administrator or by dynamic DNS methods. This contrasts to answers that were obtained via a regular DNS query to another name server. An authoritative-only name server only returns answers to queries about domain names that have been specifically configured by the administrator.

A set of authoritative name servers has to be assigned for every DNS zone. An NS record about addresses of that set must be stored in the parent zone and servers themselves (as self-reference).

When domain names are registered with a domain name registrar, their installation at the domain registry of a top level domain requires the assignment of a primary name server and at least one secondary name server. The requirement of multiple name servers aims to make the domain still functional even if one name server becomes inaccessible or inoperable. The designation of a primary name server is solely determined by the priority given to the domain name registrar. For this purpose, generally only the fully qualified domain name of the name server is required, unless the servers are contained in the registered domain, in which case the corresponding IP address is needed as well.

2.3 HTTP

The HyperText Transfer Protocol(HTTP), is the main protocol used by the World Wide Web and it dictates how messages are created and transmitted. Moreover it controls what actions Web Servers and

browsers should take after they have received a specific command. For example, when you enter a URL in your browser, an HTTP command is sent to the respective server directing it to fetch and transmit the requested web page.

2.4 TLS/SSL

Transport Layer Security (TLS) (and its predecessor Secure Sockets Layer (SSL)) is a cryptographic protocol that ensures security of communication over the internet. It uses asymmetric cryptography, and in general, provides the users with confidentiality, message integrity and message authentication through MAC protocols. It is a protocol used widely in many applications and therefore most of the security of those applications depends heavily on the security of the protocol itself.

First, it is important that both the client and the server have agreed to use TLS. Then a handshaking procedure is used.

The handshake procedure consists of the following steps[1]:

- The SSL or TLS client sends a "client hello" message that lists cryptographic information such as the SSL or TLS version and, in the client's order of preference, the CipherSuites supported by the client.
- The SSL or TLS server responds with a "server hello" message that contains the CipherSuite chosen by the server from the list provided by the client, the session ID, and another random byte string. The server also sends its digital certificate. If the server requires a digital certificate for client authentication, the server sends a "client certificate request" that includes a list of the types of certificates supported and the Distinguished Names of acceptable Certification Authorities (CAs).
- The SSL or TLS client verifies the server's digital certificate.
- The SSL or TLS client sends the random byte string that enables both the client and the server to compute the secret key to be used for encrypting subsequent message data. The random byte string itself is encrypted with the server's public key.
- If the SSL or TLS server sent a "client certificate request", the client sends a random byte string encrypted with the client's private key, together with the client's digital certificate, or a "no digital certificate alert".
- The SSL or TLS server verifies the client's certificate.
- The SSL or TLS client sends the server a "finished" message, which is encrypted with the secret key, indicating that the client part of the handshake is complete.
- The SSL or TLS server sends the client a "finished" message, which is encrypted with the secret key, indicating that the server part of the handshake is complete.
- For the duration of the SSL or TLS session, the server and client can now exchange messages that are symmetrically encrypted with the shared secret key.

When the TLS handshake is completed, the session begins with symmetric session keys established in the handshake. The client and the server use the session keys to encrypt and decrypt the data they send to each other and to validate its integrity.

2.4.1 HTTPS

HTTPS is in essence HTTP with an additional security layer build on top using SSL/TLS. HTTPS creates a secure channel over an insecure network. Assuming that we are utilizing a public key certificate that can be trusted, HTTPS gives protection against man-in-the-middle attacks and eavesdroppers.

Web browsers have certificate authorities pre-installed in their software. Therefore they can tell whether an HTTPS website should be trusted or not. Certificate authorities are in this way being trusted by web browser creators to provide valid certificates. All of the following must be true for a user to trust an HTTPS website:

- The user trusts that the browser software correctly implements HTTPS and that the certificate authorities have been correctly pre-installed.

- The user trusts the certificate authority to vouch only for legitimate websites. The website provides a valid certificate, which means it was signed by a trusted authority.
- The certificate correctly identifies the website (e.g., when the browser visits "https://google.com", the received certificate is properly for "Google Inc." and not some other entity).
- Either the intervening hops on the Internet are trustworthy, or the user trusts that the protocol's encryption layer (TLS/SSL) is sufficiently secure against eavesdroppers.

HTTPS is especially important over unencrypted networks (such as WiFi),since (as we have mentioned above) someone on the same local network can "packet sniff" and discover sensitive information.

The security of TLS/SSL makes it is impractical to implement an HTTPS spoofing attack. Therefore, we must find another way to present the user with a similar using website without using the HTTPS protocol.

3 Related work

There are many ways to perform Man-in-the-middle attack. This section will outline two methods namely ARP poisoning and Evil twin attack which are different from ours.

3.1 ARP poisoning

ARP poisoning is a type of attack where the Media Access Control (MAC) address is changed by the attacker.

3.1.1 Overview of ARP

ARP stands for Address Resolution Protocol which is a protocol to associate a network layer address (i.e IP address) to a link layer address (i.e MAC address). A typical communication is as follows (Figure 1 depicts the above procedure visually):

- Machine A in a local network wants to contact machine B in the same local network
- Using DNS, machine A determines the IP address of machine B
- It will looks in its ARP Cache table if there is a MAC address for machine B already present in the table
- If not, machine A will broadcast an ARP requests to every machine in the same network with message like "Which machine has this IP address xxx.yyy.zzz.kk? Here is my IP and MAC"
- All of the machine in the network will ignore such request except for the machine B which has that IP address.
- Machine B will send a respond back to A (using A IP and MAC in the ARP request) with its MAC address.
- Machine A updates its ARP cache table with B information and starts communicating from there.

3.1.2 Vulnerability

ARP built upon the idea of simplicity and efficiency without any sort of authentication. In the above procedure, a malicous machine C sends a reply to machine A claiming that it has the requested MAC address for the associated IP address, machine A will blindly accept the ARP request from C and update its table. To add insult to injury, some machines will accept an ARP response even if it never sent a request in the first place. This protocol creates a security hole in which a malicous machine C can craft its own ARP responses to target machine A and associate any IP address (i.e the default gateway) to any MAC address (i.e its MAC address) to establish a man in the middle attack among other things. Even if machine A refreshs its cache, machine C can continue its attack by periodically sending an ARP response. On the bright side, this type of attack only works on a LAN network, so attacker needs to have physical access to the LAN network for this attack to work.

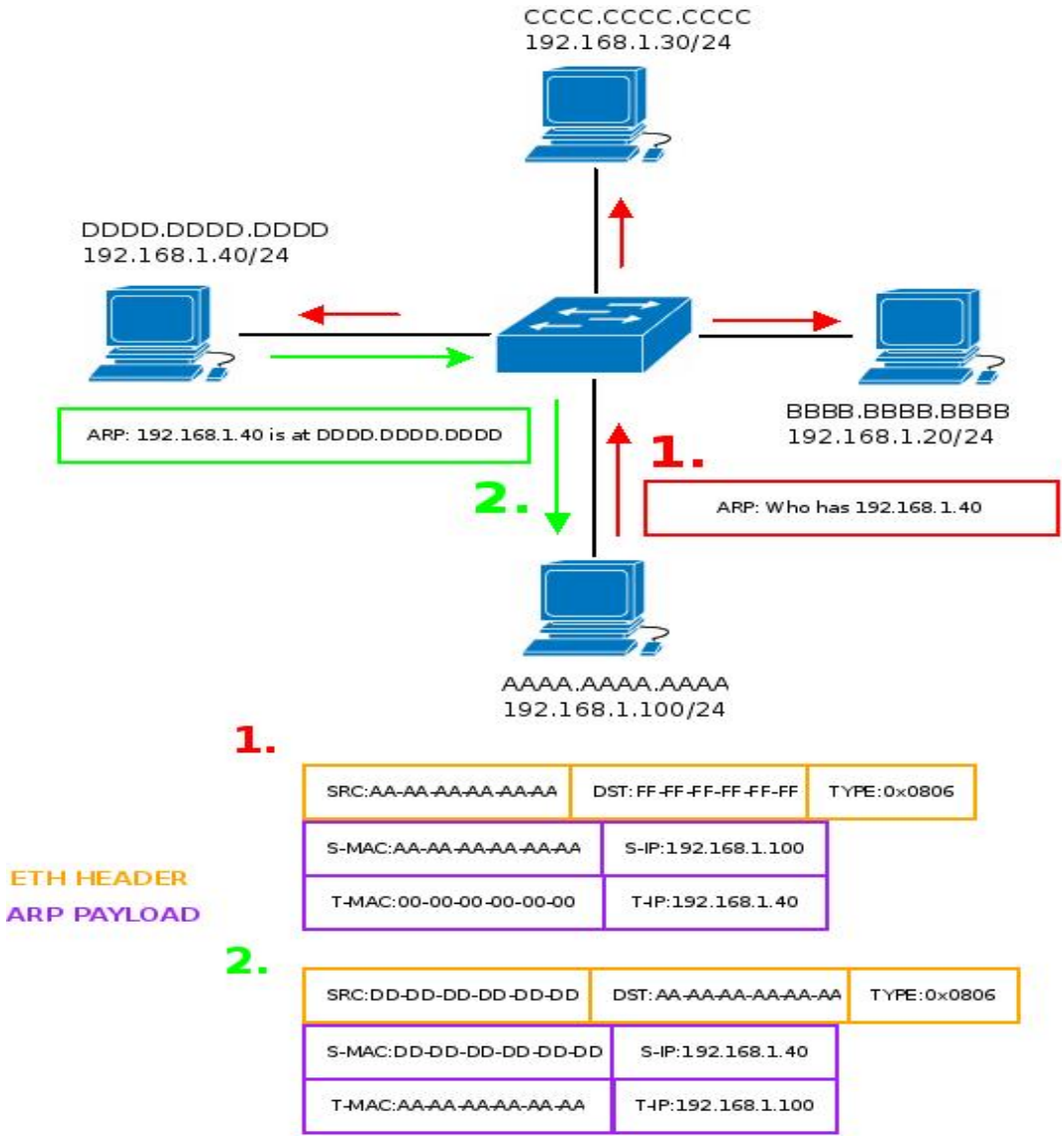

Figure 1: Machine A sends the ARP request (1) to ask for the MAC address of the IP address 192.168.1.40. Machine D replies with its MAC address. The content of the ARP request and response are summarized in the yellow and purple table.

3.1.3 Damage

Among the problems that ARP cache poisoning can cause to the victim machine, following are the two major ones:

- **DOS (Denial of Service)**: If machine C decides to associate every other machines in the network including the default gateway with a non existent MAC addresses, it can block machine A from communicating with the network. Any packet machine A sends out will be delivered to a non existent MAC addresses and eventually gets dropped.
- **Man in the Middle**: If machine C wants to intercept the traffic between machine A to any machine in the network, it can associate those IPs that machine A tries to communicate to with its MAC address. Now machine C can act as the man in the middle and start tampering with machine A traffic.

3.1.4 Prevention

There are many proposed solutions for this including:

- Using software such as ArpWatch which will inform victim about changes in ARP mappings.
- Statically configure ARP table to prevent any automatic update
- Using advanced switches which will discard malicous ARP packets.

3.2 Evil-twin attack

Evil twin attack is another type of Man in the Middle in which an attacker can broadcast a wifi signal (either by using software like softAP or router) with the same SSID as the legitimate one on the premise.

3.2.1 Overview of wireless connection

A typical communication between a 802.11 station (i.e a laptop) and a 802.11 AP can be seen in Figure 2. 802.11 can listen for AP beacon/probe response actively or passively. ESSID (Extended Service Set ID) is the name given to any group of APs providing wireless access to the same upstream network, such as a corporate network or the Internet. The 802.11 station user will intentionally choose an AP to connect, or the 802.11 station will automatically send the best AP (based on signal strength) to ebstablish a connection. An AP using WEP or WPA can optionally challenge the station to prove it knows a shared key. But in many WLANs, the AP just returns an Authenticate Response. After the connection is ebstablished, both parties can start communicating until either one sends a Deauthenticate or Disassociate packet [14].

3.2.2 Vulnerability

Evil twin attacks happens in various forms, but the vulnerability many times come from humans. This include vendors such as an airport, coffee shop, or book store which provide its customers free wifi access without any network encryption password (WEP or WPA). This mostly comes from the inconvinience of establishing the pre-sharing password. Evil-twin attacks also exploit the common desires of general public who want to access free wifi. Therefore, an attacker with a machine capable of broadcasting wifi signal can establish himself as a legit AP on the premise and trick the user to connect to it. Some hardwares will automically connect to a network in its preference list based on signal strength of the AP. The attacker can even force the victim computer to connect to his AP even when the victim is already connected to a legit AP by launching mutliple DOS. [2].

3.2.3 Damage

Similarly to the ARP poisoning, when successfully trap the user to connect to the its AP, the attacker has a platform to perform man in the middle attack. It can present the users with various phishing sites to harvest credentials, DNS hijacking to redirect the users to its servers, responses to user requests with offensive information among other things.

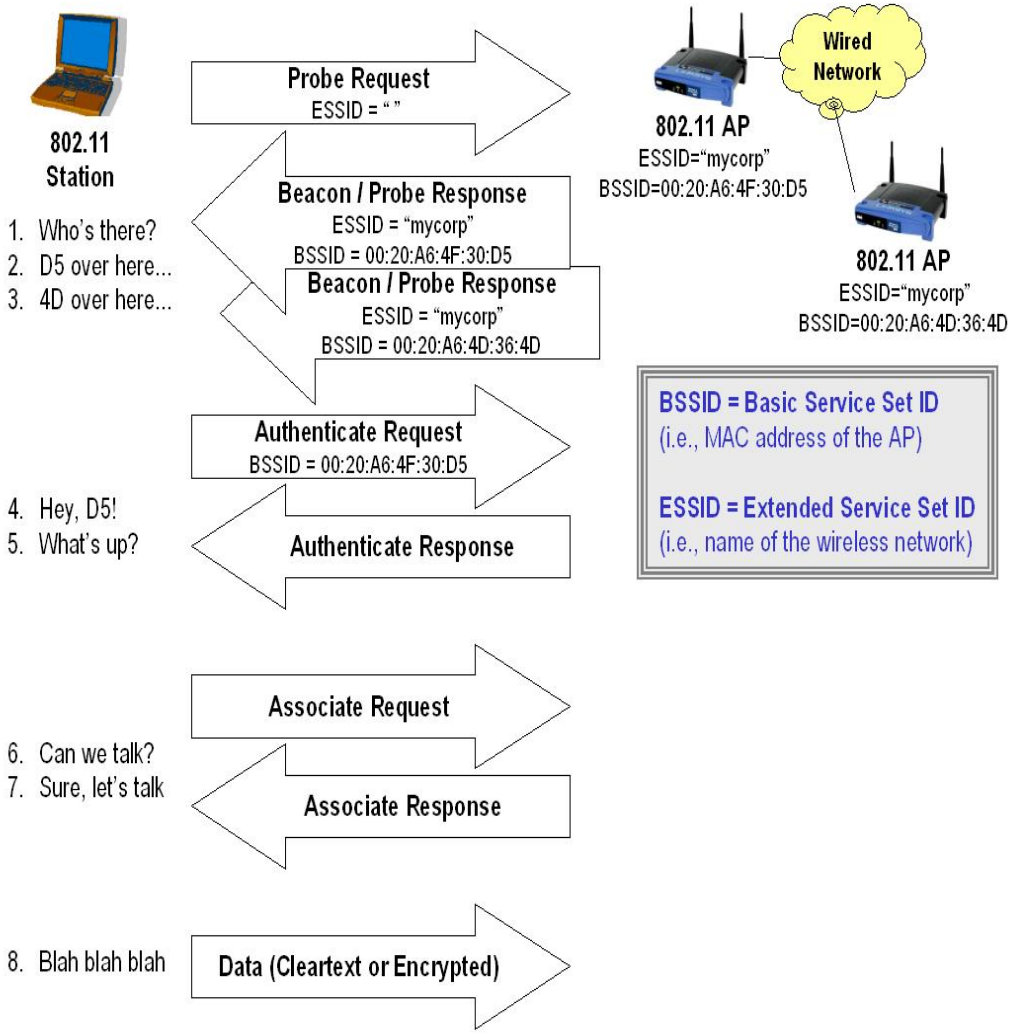

Figure 2: 802.11 station starts off by sending a probe request to search for available Wifi network. The 802.11 APs will send the probe responses identify itself by an SSID. If the 802.11 station decides to connect, the authentication process will proceed.

3.2.4 Prevention

There are no absolute tool/mechanism to completely eliminate evil twin attack. Following are some suggestions for mitigation:

- Configure the wireless settings of the OS to connect only to preferred networks and only upon requests.
- Avoid connecting to insecure networks and browsing sensitive/personal information such as banking.
- Use external software such as AirDefense Personal to warn user of unusual wireless activity.

4 Methodology

4.1 Overview

There are three main components we need control over for our version of a man in the middle attack: a router, a DNS server, and a content delivery server. Our router will force DNS requests to go to our DNS server. The DNS server will respond with the IP address of our content server when a victim asks for a target website of our choosing (see Figure 3). With the victim connects to our content server, the server will deliver content which looks similar to the target website but is modified (see Figure 4).

Links to resources on how to setup these systems are referenced at the end of this paper.

4.2 Raspberry Pi and Router Setup

A Raspberry Pi Model B was used for both the router setup and DNS Server hosting. Given more time, it would be possible to run the content server on the Raspberry Pi as well. The Raspberry Pi is a low power Linux based computer which currently sells for $35 in the US. We decided to use a Raspberry Pi since there is extensive community support for this particular hardware and its low price. The particular distribution of Linux used is called Raspbian which is based on Debian.

4.2.1 Router Setup

The Raspberry Pi was made to behave as an internet router by using a wireless adapter which forwards packets to the ethernet cable interface. The wireless adapter used is a $10 USB device from a company called Edimax. To convert the Raspberry Pi into an access point a package called hostapd was used. The version of this package used is specific to the USB adapter used.

Another package called isc-dhcp-server was installed to behave as the Dynamic Host Configuration Protocol (DHCP) server. The job of this server is to assign temporary IP addresses to devices which connect to it and distributes basic network parameters including DNS parameters.

At this point, computers will be able to connect to the Raspberry Pi but internet packets will not be forwarded to the ethernet cable. The last step is to forward packets from the wireless interface to the ethernet interface by modifying the IP tables.

After following these instructions, the Raspberry Pi behaves as a normal internet router. However, the owner has control over some important parameters including the name of the network, the wifi password, and the default DNS servers to use. To attempt a man in the middle attack, a person can set the name and password of the network to something that people are likely to connect to. For example at a coffee shop, the name of the network can contain the name of the coffee shop. The defaults DNS servers used should be ones that redirect targeted name requests to a compromised content server.

4.3 DNS Server

The DNS server is also run through the Raspberry Pi using a package called bind which is currently on version 9. After setting up its basic functionality, users can set up their own zones. When a DNS request is sent to the bind server, it will first check if the address is in one of its own zones before forwarding the request to other known DNS servers. This makes it easy to override the standard response for a given URL and replace it with one of your own.

When a user uses the compromised DNS server, the browser does not have a standard way of verifying that the DNS response is the correct one. Therefore the URL displayed in the web browser will show

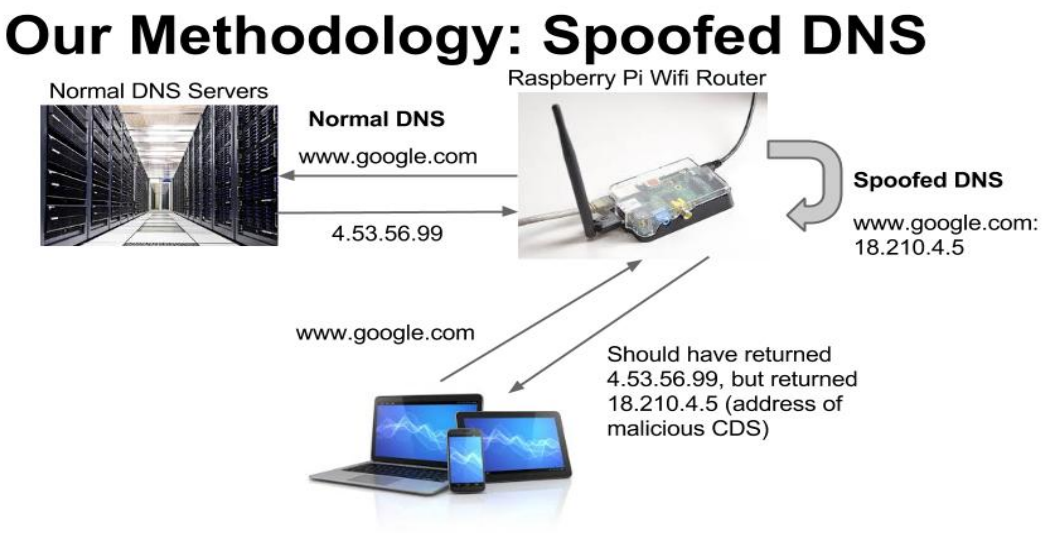

Figure 3: Block Diagram Outlining our DNS Spoofing Attack

the name that was typed despite being connected to a different server. In a man in the middle attack, the zone would map URL to the IP address of a server we control.

4.4 Content Delivery/Manipulation Server

4.4.1 Overview

The purpose of the Content Delivery Server (CDS) is to not only fetch the web pages requested by a victim of our Man in the Middle attack but also to modify those webpages in various ways so that the implementer of our attack can either cause the vicitim to become frustrated (via prank manipulations) or clandestinely collect information about the victim such as login information or all key press data (see Figure 4). Our server comes with a simple UI that allows the implementer of the attack to spin up an instance of our content server and employ certain attacks/pranks via that interface. This was a crucial design decision that makes our system usable by anyone with the appropriate .jar file, meaning all grandma has to do to start hacking is double click (run the file assuming that the Java Runtime Environment is installed).

4.4.2 Outline of CDS System and User Interface

After running the .jar file, the CDS GUI (see Figure 5) is displayed. Here the user first enters the Target URLs (domains that the system should intercept and manipulate web pages for). These target urls should be semi-colon separated so that the system can parse them correctly.

Next, the user can specify a plethora of different page manipulations that can be used in different combinations to implement a wide variety of attacks. The types of options our CDS supports are as follows:

- **Swap Image** - Replaces every image on the retrieved webpage with a particular image that is specified by a link to that image (put a link to the image to be displayed in the Swap Image textbox). This attack can cause frustration and startle or frighten the victim if the desired image is obnoxious or scary enough. Note a simple way to retrieve an image link is to do a google search for the image, right click the desired image, and click Copy Image URL.

- **Force YouTube Video to Play** - Adds a fixed iframe to the retrieved webpage that automatically starts playing the specified YouTube video. This provides the same benefits as the **Swap Image** option except now there is also audio and since the iframe is fixed the victim won't be able to remove the YouTube video from view even by scrolling down the page. Put a link to the YouTube video to be played in the Force YouTube Video to Play textbox to enable this option. To get the

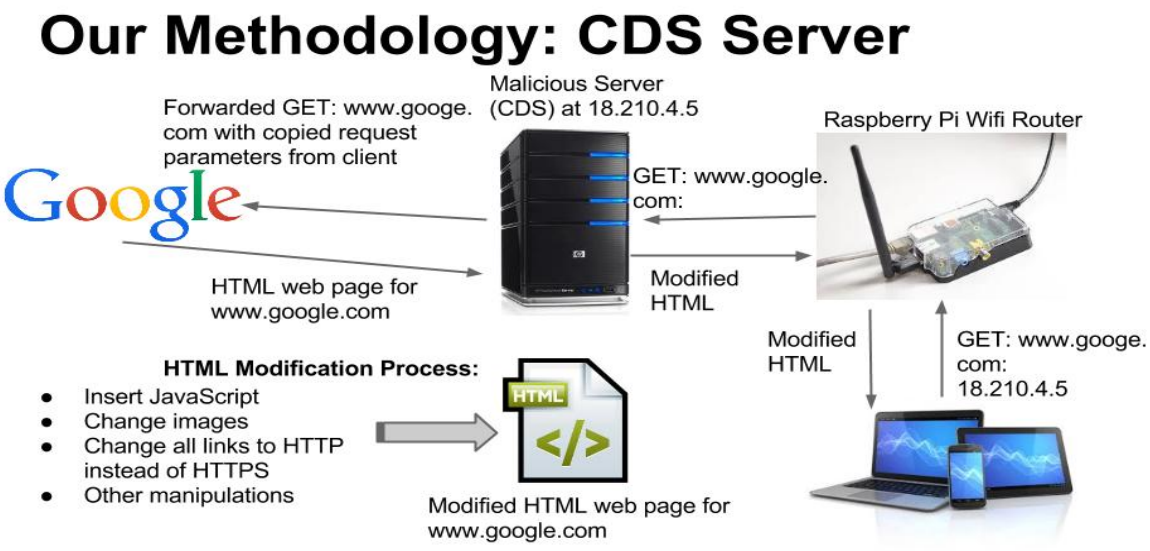

Figure 4: Block diagram outlining how our CDS works where google.com is the site being attacked

YouTube video link simply search for the desired video on YouTube, click the Share button on the YouTube video page, click the Embed button, and copy the src address.

- **Popup Message** - This causes an alert box with the specified message to be displayed to the user on every button click and disables the button's original functionality. This can cause immense frustration as the victim will be plagued by the user's message and won't be be able to interact with simple web forms properly (i.e. not be able to login to GMail). To enable this option simply enter a message in the Popup Message textbox.
- **Keylogger** - Attaches a keylogger to every input field on the retrieved webpage. The keylogger is essentially some javascript that is attached to the jQuery keypress event for that input field. Once the event is triggered the javascript packages the input field's name, id, and contents into a JSON object which it then sends to our CDS `/log` endpoint (since all domains point to our CDS, the correct handler will be invoked) via an AJAX request, thus covertly logging keystrokes. Once the server processes this log request it writes the sent data to a file called `log.txt` in the directory where the .jar file is located. The structure of a `KeyLogger` log entry is as follows (however in the log the data is horizontally concatentated): The first line is a timestamp indicating when the

Fri May 02 19:36:08 EDT 2014
Key Stroke Detected
Remote IP: 0:0:0:0:0:0:0:1
Webpage: https://bankofamerica.com/
Field Name: password_field
Data: password

keypress event took place. The second line just indicates the type of log entry it is. The third line indicates the IP address of the client that requested the webpage that they keyloggers are attached to. The fourth line specifies the webpage the victim was on when the keypress event happened. The fifth and sixth lines specify the name and value of the input field, respectively, in this case the field was a password field and the keylogger picked up password data.

This attack is meant for clandestine data collection and can be used in combination with the previous options but it won't be as effective because the previous options expose to the victim that the fact that the webpage has been manipulated in an unathorized way. To enable this option simply check the checkbox.

- **Save Credentials** - Saves victim login information. This is done by attaching some javascript to the `onsubmit` event of forms that have "login" or "sign" in their ids or names (this is a simple

Figure 5: Screen Shot of the User Interface for our Content Delivery Server

heuristic that tries to target login forms specifically). When the form is submitted the entire contents of the form is sent to the `/log` endpoint (since all domains point to our CDS, the correct handler will be invoked) on the CDS via AJAX and is logged along with the keylogger data. A `SaveCredentials` log entrie looks identical to `KeyLogger` log entry except the entry type is Save Credentials and the data is actually the string version of an entire form (you can easily discern the form's fields and corresponding values).

This attack is meant for clandestine data collection and can be used in combination with the previous options but it won't be as effective because the previous options expose to the victim that the fact that the webpage has been manipulated in an unathorized way. To enable this option simply check the checkbox.

Now that the user has specified all the page manipulation options, the attack can be initiated by simply clicking `Submit`. If the user changes any options, clicking `Submit` again will restart the server with the updated parameters.

You can test this system on localhost (no DNS server or Raspberry Pi required) by doing the following:

- Run the .jar file and use the UI to launch the server
- Open up a browser and type `http://localhost:8080/?url=google.com` to use the CDS server to connect to google.com/
- The value of the `url` GET parameter is extracted by the CDS and uses that as the url of the webpage to retrieve.

4.4.3 Implementation Details

Main Server Module

Our CDS was built in Java and uses the Jetty API and framework to handle HTTTP requests. The server keeps track of all clients that made requests to as identified by IP address so that the system can quickly identify the same client and store cookies sent to/from the client (victim). This is important for websites that need to identify users to generate webpages customized for a particular user. The main server is located in `EasyMIMServer.java` in the `server` package. The server is made up of a Servelet class that has two functions `doGet()` and `doPut()` that handle GET and PUT requests, respectively. Once a request is made to the CDS, the CDS analyzes the request and does the following:

- If this is a new client it adds the client information to the `sessions` hashmap keyed on client IP address.

- If the request is a `/log` request then pass the request on to the `Logger` module.
- If the request is a normal HTTP request then pass the request and client information (after adding the client to the session if it is a new client) on to the `WebsiteProcessor` module.

WebsiteProcessor Module
The websiteProcessor takes an HTTP request made to the CDS does the following:

- Extracts the intended URL the victim wants to reach
- Makes a series of requests to the intended URL using a variety of protocols starting with HTTPS until a valid response is received (no HTTP error codes). The reason for this is that some websites like Facebook won't load over HTTP while others won't load over HTTPS. In turn, to get the maximum coverage we try fetching the webpage with all protocols.
- Takes the received HTML webpage and passes it on to the `processHTML()` function that parses the HTML using JSoup (an HTML parser that provides jQuery like functionality) and does the following manipulations:
  - Append a reference to the jQuery library to the Head of the document just in case it isn't already there. Without jQuery the javascript inserted by the CDS wouldn't work.
  - Attempts to set the appropriate site icon in the header.
  - Modifies the src attribute of all image and input elements from relative paths to absolute paths (this is primarily used to get the software to work on localhost for testing, in the real scenario this is uncessary) based on the domain of the fetched webpage.
  - Fetches a `WebsiteHandler` for the domain of the webpage being loaded (if one exists, these will be described in the following section) and passes the parsed HTML object to the corresponding WebsiteHandler for processing.
  - After this the webpage is modified in the ways specified by the options in the UI.
- Now the modified webpage is returned to the client.

WebsiteHandlers
Because websites differ in the way that they structure their HTML, some webpages need further alteration just to make them look normal. This is where the `WebsiteHandler` modules come into play. They are website specific and process the retrieved HTML in a way specific to that domain so that the page looks right to the client (i.e. the javascript runs correctly and the images look right). Once a `WebsiteHandler` module is developed to handle webpages from a particular domain the module is registered with the `WebsiteProcessor` by adding the `WebsiteHandler` to the static attribute hashmap `domainToHandler` in the `WebsiteProcessor` keyed on the domain for the `WebsiteHandler`. The `WebsiteProcessor` checks for a `WebsiteHandler` when it processes a webpage from a particular domain. Note that you can also do this for subdomains.

Logger Module
The `Logger` module is used to log the `KeyLogger` and `SaveCredentials` data sent by the client-side keylogging javascript in the modified HTML page returned by the CDS. The data is appened to a `log.txt` file in the same directory as the .jar.

Bridging the Gap From the UI to the Server
Once the user hits `Submit` in the main GUI, the user specified data is extracted from the GUI and placed in a `EasyMIMConfig` object that is then passed to the constructor of the server that then uses those settings in setting up the server.

4.4.4 Demo

We have a demo of the CDS server and UI which we dubbed EasyMIM (Easy Man in the Middle) located here: `https://www.youtube.com/watch?v=WBuCj5uTXos`.Our source code is also located here: `https://github.com/sciencewiz1/EasyMIM`.

5 Relevance to Society

We believe this research is particularly relevant at the moment and we would like to enumerate some of the ways that this research is relevant to programmers as well as laymen.

5.1 Ease of Attack

5.1.1 For a Programmer

For somebody who is familiar with programming and working with computers this would be a simple and inexpensive attack to execute. A programmer could make this attack work on any Operating System of their choice by downloading and installing the equivalent packages mentioned in the methodology section.

5.1.2 For a Layman

Although, there's currently no easy to use hardware or software which can accomplish this kind of the man in the middle attack out of the box, it is easy to conceive how to make such a product. A possible business model would be to buy Raspberry Pis install all the packages and software we developed and distribute these with a markup. With the appropriate instructions, users would plug in their Raspberry Pi to ethernet, an HDMI monitor, and a keyboard. The default boot up would display the UI we have developed making it easy to setup. Once the initial settings are set up, users could then leave their Raspberry Pi in any location which has an ethernet cable and a power source to begin gathering private information. From the safety and comfort of their own homes, this data would be forwarded to their home computer.

5.2 Protection Against this Attack

Realizing how cheap and easy it can be to execute this kind of attack should be an indicator that web developers and the common person need to be educated in protection against these kinds of attacks. Some of the following strategies may seem simple yet they can still be very effective.

5.2.1 For Developers

Our attack is a variation of the evil-twin type of attack. Instead of broadcasting an insecure wireless signal to attract users, we can broadcast the signal with similar SSID to the legitimate one; this will lead to an evil-twin attack. For developers and hardware vendors, there are a few suggestion that could help users avoid this type of attack:

- The 802.11 station can establish a context-leasing protocol in which the station will record the SSID along with its signal strength/location using the information from the probe responses. This assumes that the attacker is mobile and will try to associate the malicous wifi with different legitimate sources. Therefore, if the 802.11 station can record the above metric and notify the user of any location changes of the AP, the user will be aware of this suspicous behavior [9]. This can only mitigate the cases when the 802.11 station tries to reconnect to an already known network (i.e library, university, coffee shop). This can't prevent if the user attempt to connect to the insecure network for the first time.
- Implementing PKI on the wireless network such that AP should include a signature of the SSID (signed it with their private key) and their public key as part of the probe responses so that client can verify it [9].
- Hardware vendor should notify clients with potential risks for certain products that might be vulnerable to this type of attack and improve their products to cope with attacks.
- Browser vendors should make active notifications if the users are visiting non http website or insecure website. Study shows that active notification can mitigate many phishing related attacks [6].

5.2.2 For the General Population

There are several ways that the average person can protect themselves from this kind of attack. They only require diligence and attention to certain details on the part of the user.

- **Only connect to known secured wireless networks.** There is no easy way to verify which wireless hotspots are honest or not so it is best to connect to hotspots which you believe belong to honest people. One must also make sure that the network is secured with a strong password otherwise it is easy for a dishonest router can use the same name as an honest router.

- **Always use the 's' in https on websites that have sensitive information.** In most browsers there is a lock next to the URL when in https mode. The 's' in https stands for secure and is a way to make sure all the data sent back and forth from a server is encrypted. Using public key cryptography as described in the background, the browser is able to verify that a particular URL actually belongs to the server that it is talking to. Although there have been successful attacks on https they are much more involved, harder to pull off, and once detected are quickly shut down.
- **For those who are truly concerned use a VPN (Virtual Private Network).** A computer using a VPN sends internet requests in encrypted form to a server that has been set up by the user. The VPN will then forward the plaintext requests to the appropriate destination and forward the responses back to the user. Using a VPN server properly will allow a person to use any wireless hotspot of their choice without having to worry about data being stolen.

6 Conclusion

We started our paper by presenting the details of some man-in-the-middle attacks that were already known to the general public.

Using a router, a DNS server and a "content delivery server", we showed how to implement our own man-in-the-middle attack over a Wi-fi network. We used a Rasberry Pi for the router setup and the DNS Server Hosting, and (with time permitting) we would be able to use the Rasberry Pie to run the content server as well.

We essentially utilized the methods of Spoofed DNS and Content Delivery Server to fool the user into believing that what he sees on his screen is what he actually requested. Meanwhile, all of the practical implementation is relatively straightforward to follow and replicate. Therefore, we achieved the main goal of our project, which was ease of implementation.

Finally, we showed that informed and cautious users can make sure that they are not being manipulated by malicious third parties when using a Wi-fi network.

We believe that by our project we raised awareness to some important security issues regarding Wi-fi networks and we hope that in the future more attention will be drawn on dealing with the inherent flaws of the design of such networks.

7 Supplementary Materials

Here are links to the source code and demo video:

Source Code: `https://github.com/sciencewiz1/EasyMIM`

Demo Video: `https://www.youtube.com/watch?v=WBuCj5uTXos`

8 References

References

[1] *An overview of the SSL or TLS handshake.* URL: `http://www-01.ibm.com/support/knowledgecenter/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660_.htm?lang=en` (visited on 03/24/2014).

[2] J. Bellardo and S. Savage. "802.11 denial-of-service attacks: Real vulnerabilities and practical solutions". In: *USENIX Security* (2003).

[3] Thomas H. Cormen et al. *Introduction to Algorithms.* 2nd. McGraw-Hill Higher Education, 2001. ISBN: 0070131511.

[4] Oracle Corp. *Java^{TM} Platform, Standard Edition 7 API Specification.* URL: `http://docs.oracle.com/javase/7/docs/api/` (visited on 04/23/2014).

[5] *Domain Name System, Wikipedia.* URL: `http://en.wikipedia.org/wiki/Domain_Name_System` (visited on 04/25/2014).

[6] Serge Egelman, Lorrie Faith Cranor, and Jason Hong. "You've been Warned: An Empirical Study of the Effectiveness of Web Browser Phising Warnings". In: *CMU Research Showcase* (2008).

[7] The Eclipse Foundation. *Eclipse IDE.* URL: `https://www.eclipse.org/` (visited on 04/23/2014).

[8] Inc GitHub. *GitHub*. URL: https://github.com/ (visited on 04/23/2014).

[9] Harold Gonzales et al. "Practical Defenses for Evil Twin Attacks in 802.11". In: *IEEE Globecom Communications and Information Security Symposium* (2010).

[10] Jonathan Hedley. *JSoup API and Documentation*. URL: http://jsoup.org/ (visited on 04/23/2014).

[11] M. FRANS KAASHOEK von Hicks. *Principles of Computer System Design An Introduction*. 1st. Morgan Kaufmann, 2009.

[12] *How-To: Turn a Raspberry Pi into a WiFi router*. URL: http://raspberrypihq.com/how-to-turn-a-raspberry-pi-into-a-wifi-router (visited on 03/31/2014).

[13] Webtide LLC. *Jetty API and Documentation*. URL: http://www.eclipse.org/jetty/ (visited on 04/23/2014).

[14] Lisa Phifer. "Anotomy of a Wireless "Evil Twin" Attack". In: *Watch Guard* (2005).