**Introduction**

Many people when attempting to connect to a secure website, such as a banking website, simply enter the well-known web address such as *www.chase.com*. What actually occurs between the web browser and the network is the IP address of *http://www.chase.com* is found and a request is sent. Since the server wants to use a secure version of the web site it then sends an http redirect message back to the client telling it to instead contact *https://www.chase.com*. This inherent redirect is subject to a number of different attacks; however this work focuses on the possibility of intercepting the redirect packet and instead servicing the initial http request with a mock site. We will show that an adversary can pose as a public access point and perform such an attack. If more time were given various solutions would also be explored that would mitigate the possibility of such attacks.

**Related Work**

There are many works that can be found about phishing schemes that perform similar attacks. Most of these use techniques such as ARP cache or DNS cache poisoning to allow all network traffic to be routed through them, the most notable of these is *sslstrip.* These tools however are mostly focused on changing the redirect message to contain a homograph-similar site name (such as *www.cha5e.com* etc.).

Other works discuss the correctness of even allowing an http redirect to point to an https citing security gaps such as the ones used by the aforementioned tools.

**Adversary model**

This attack is demonstrated as a type of Man In The Middle attack. An adversary can provide a public access point and allow anyone to connect to the internet via his connection. Being the first hop in their connection all packets are available to sniff and alter as desired. The adversary would use a filter to forward all normal traffic but would keep a listing of sites that it has doctored and replace all redirects to https versions of these sites with the doctored ones. This would present the victim with a website that appears to be the normal https site they are used to but would actually only be a normal http site with the correct address that they initially typed (as opposed to a homograph-similar address). Since many people are unaware of the difference between http and https the adversary would be able to acquire sensitive information from the victim such as usernames and passwords.

**Methodology**

We were unable to implement any solution to this problem but thought of a few ways one would go about implementing one.

One idea is to build a table into browsers that contains lists of sites that should always resolve using the https version instead of http. This list could be maintained through browser updates and other methods but may end up causing too much upkeep overhead or become unsynchronized with the actual state of the internet.

Some people purpose the idea that the redirect message should be exchanged for an error message that prompts the user to manually contact the server via the https address. This may thwart many problems similar to the one in this paper but also creates a less user friendly method.

Ideally there would be a way to make it the networks responsibility so it could be done without bothering either the client or the server.

**Implementations/Experimentation**

**Conclusion**

**References**