Becoming a Hacker – Part 1

By Elite Nabukadnezar

List of Chapters for Part 1:

Document Title: Becoming a Hacker – Part 1
Document version: 0.1
Release date: Wednesday, March 1, 2006

Author: Andronache Laurentiu
Nickname: Elite Nabukadnezar
Location: Bucharest, Romania
Leader of: TAG (The Absolom Group). Contact me if you are a skilled hacker and you want to join the Group.

Website: www.absolom.ro (check it from time to time for the next parts of "Becoming a Hacker")
Document Forum: www.absolom.ro/f (all the discussions related to this document go here)

== Short Introduction ==

I live in Romania so my English is not that good, but I hope you'll understand my tutorials. If you don't I'm sorry, try reading again. "Becoming a Hacker" will be hopefully very interesting for would-be hackers, network admins, and hackers that want to improve themselves. I don't take any responsibility for bla bla bla... you know it. Let's start hacking.

== The OS ==

Are you on Linux? If the answer was "yes" then I might also ask... do you really know how to use it? Can you write a bash script? It doesn't really matter the operating system you're working with, what's important is that you really know how to use it. If you're not that good then improve yourself by reading about it. If you're good, skip this chapter.

In case you don't have Linux installed on your machine choose a Linux distribution and install it. A Unix-like operating system using the Linux kernel[1] is called a "distro" (Linux distribution). You have lots of distros to choose from:

- Fedora Core (http://fedora.redhat.com/)
- Mandriva (http://www.mandriva.com/)
- Debian (http://www.debian.org/)
- Suse (http://www.opensuse.org/)
- Slackware (http://www.slackware.com/)
- Ubuntu (http://www.ubuntulinux.org/)
- Gentoo (http://www.gentoo.org/)
- MEPIS (http://www.mepis.org/) etc.

You can also use other operating systems very similar to Linux like the ones derived from BSD[2] (FreeBSD, OpenBSD and NetBSD) or like those derived from UNIX[3] or its successors (Solaris, HP-UX, IBM AIX etc.).

You can even use Windows if you're too familiar with it and you can't give up programs like Winamp and Windows Media Player. Windows was your choice? Well, then here's what you'll have to do:

1. Install VMware Workstation (www.vmware.com), which is virtualization software that will allow you to run other operating systems from Windows.
2. Install a distro on your VMware machine.
3. Alt-tab to it whenever you need Linux (most of the time).

You can also properly install Linux and then set up a boot menu from which to choose whatever OS you want to boot (recommended if you're a novice).

In case you have root access on a remote UNIX box, you may keep just windows installed on your system and use putty to connect to that server, which you'll use to develop your hacking skills further.

It doesn't really matter what way you will choose, what's important is that you have Linux if you're a starter in hacking. Then learn how to operate it. All the Linux screenshots you'll see in this tutorial are made in Ubuntu; I have it running on VMware.

---

[1] The kernel is the fundamental part of the operating system. Every OS is built on a kernel. It's a piece of software responsible for allocating hardware resources to the applications and many other basic stuff like that

[2] Berkeley Software Distribution refers to a particular version of the Unix operating system that was developed and distributed from the University of California at Berkeley long time ago

[3] An operating system that originated at Bell Labs, 1969. It was a multi-user and multi-tasking operating system, with TCP/IP built-in, the first OS to be written in the C programming language.

== Understanding TCP/IP ==

You can't be a hacker without knowing at least some basic stuff about TCP/IP. I'm gonna help you understand its basics, but this chapter is no substitute for a real book on the subject. I'd recommend "TCP/IP Tutorial and Technical Overview" – By Adolfo Rodriguez, John Gatrell, John Karas and Roland Peschke. You can download it from here:
http://www.absolom.ro/index.php?name=Sections&req=viewarticle&artid=7&page=1

TCP/IP stands for Transmission Control Protocol/Internet Protocol. It is a suite of communication protocols used to interconnect computers/devices on a network, used as a standard for the Internet and most of the private networks. Like most network protocols, TCP/IP is layered. Each layer builds upon a layer below it, adding new functionality. The lowest level protocol is concerned purely with the operation of sending and receiving raw data using available network hardware. At the top are protocols designed specifically for tasks like transferring files and delivering email. In between are levels concerned with things like routing and reliability. The main benefit that a layered protocol stack offers is that if you create a new network application or a new type of hardware, you only need to create a protocol for that application/hardware; you don't need to rewrite the whole stack. TCP/IP is modeled in four layers:

Application Layer: this is provided by the program that uses TCP/IP for communication. Examples: FTP, HTTP, SMTP…

Transport Layer: provides the end-to-end data transfer by delivering data from an application to a remote peer. The most used transport layer protocol is TCP, this being one of the reasons that lead to the entire protocols suite being called TCP/IP. The other transport layer protocol is UDP (User Datagram Protocol). TCP provides connection-oriented reliable data delivery, duplicate data suppression, congestion control and flow control, unlike UDP, which provides connectionless, unreliable but faster service. Applications using UDP will have to implement their own end-to-end integrity, flow control and congestion control, if it is desired. Usually, UDP is used by applications that need a fast transport protocol and can tolerate the loss of some data. For example: audio and video streams.

Network Layer: also called the internet layer or the internetwork layer. The best example here is the Internet Protocol (IP) which is the most important protocol in this layer, again one of the reasons for calling the entire suite of protocols TCP/IP. IP is a connectionless protocol that doesn't provide reliability, flow control, or error recovery. These functions must be provided at a higher or a lower lever. IP provides a routing function that attempts to deliver transmitted messages to their destination. A message unit in an IP network is called an IP datagram. This is the basic unit of information transmitted across TCP/IP networks. Other network layer protocols are ICMP, IGMP, ARP and RARP.

Link layer: also called the network interface layer or the data-link layer, is the interface to the actual network hardware. This interface may or may not provide reliable delivery, and may be packet or stream oriented. In fact, TCP/IP does not specify any protocol here, but can use almost any network

interface available, which illustrates the flexibility of the IP layer. Examples are Ethernet, FDDI[1], X.25 etc

Here's a list with link layer standards:

IEEE[2] 802.2 aka the Logic Link Protocol, used in LANS.

IEEE 802.5 aka the Token Ring LAN.

IEEE 802.3 (Ethernet) has largely replaced all the other standards like token ring or FDDI. Varieties of Ethernet: 10 Mb/s Ethernet, Fast Ethernet (100 Mb/s), Gigabit Ethernet (1000 Mb/s) and 10 Gigabit Ethernet.

FDDI supports data transmission on a fiber network medium at a rate of 100 Mbps. Usually used for joining LANs together.

X.25 is tipically used in the packet-switched networks (PSN) of common carriers, such as the telephone companies.

IEEE 802.11 are the WLAN standards, for example WiFi licensed by the Wi-Fi Alliance.

IEEE 802.16 are BWA (Broadband Wireless Access) standards, for example WiMAX[3].

IEEE 802.15.1 -> Bluetooth.

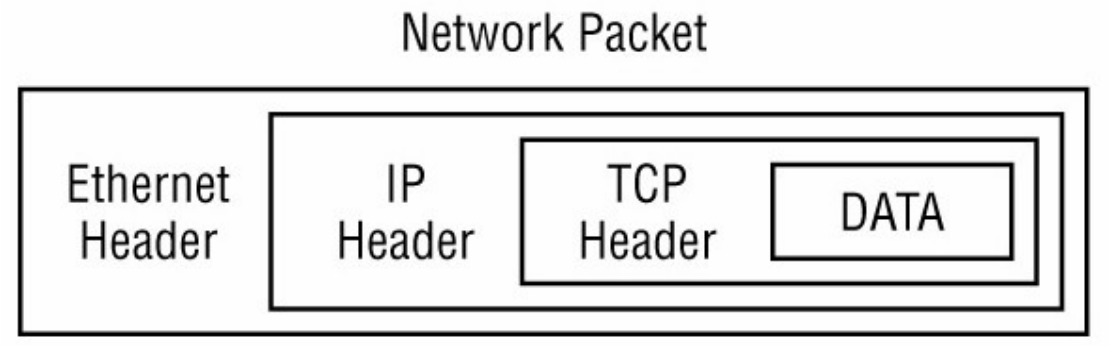
ATM (Asynchronous Transfer Mode) offers high bandwidth (up to 155 Mbps), controlled-delay, fixed-size packet switching and transmission integrating multiple data types (video, voice, data). Uses fixed-size 53 bytes packets also known as "cells" (is often referred to as "cell relay").

SNA (System Network Architecture) is IBM's proprietary networking architecture. Sadly this shit is still used extensively by banks and other financial transaction networks, as well as many government agencies.

PPP (Point to Point Protocol) can connect computers using serial cable, phone line, trunk line, cellular telephone, specialized radio links or fiber optic links. Is used by most of the internet service providers for dial-up access to the internet.

SLIP (Serial Line Internet Protocol) can connect computers over serial ports and modem connections. It has largely been replaced by PPP.

These link layer standards should be more than enough. This is how a network frame (synonym for network packet) will usually look:



[1] Fiber Distributed Data Interface

[2] Institute of Electrical and Electronics Engineers is a professional organization best known for setting transmission system standards.

[3] Worldwide Interoperability for Microwave Access, provides high-throughput broadband connections over long distances

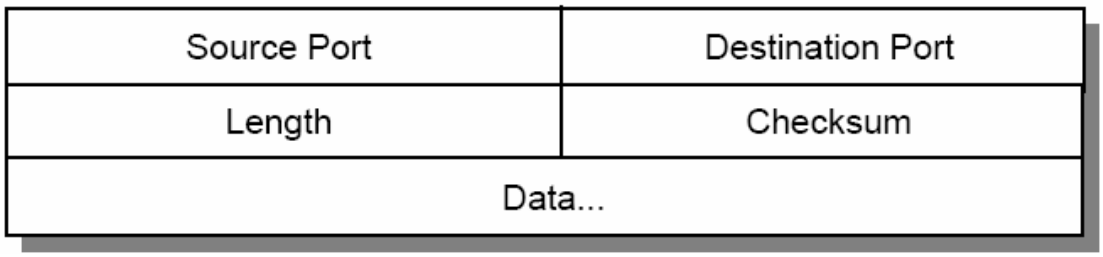
As you can see each layer adds a few bytes to the datagram. The application layer added the data (DATA), the transport layer added a TCP header (in this case the transport layer is TCP), the network layer added an IP header (in this case the network layer is IP) and finally, the link layer added an Ethernet header. If you want to have a better understanding of TCP/IP, install Ethereal.

http://www.ethereal.com/

This is a free network protocol analyzer for UNIX or Windows. It will allow you to capture packets received and sent by your computer in the network. You can interactively browse the captured data, viewing summary and detailed information about each packet, and that will give you a much better understanding of the network data flow. I leave that to you. Instead, I will briefly explain how TCP and UDP work.

UDP requires no connection to be made. It simply adds to the data a header with source port, destination port, length value and checksum. The IP (network layer) will add a header with source address, destination address etc then the link layer will add its header and the packet will be sent. Hopefully the datagram will arrive at its destination. No checking will be made.

This is how the UDP header (+ Data) looks like:



Source Port: Indicates the port of the sending process. It is the port to which replies should be addressed.

Destination Port: Specifies the port of the destination process on the remote host.

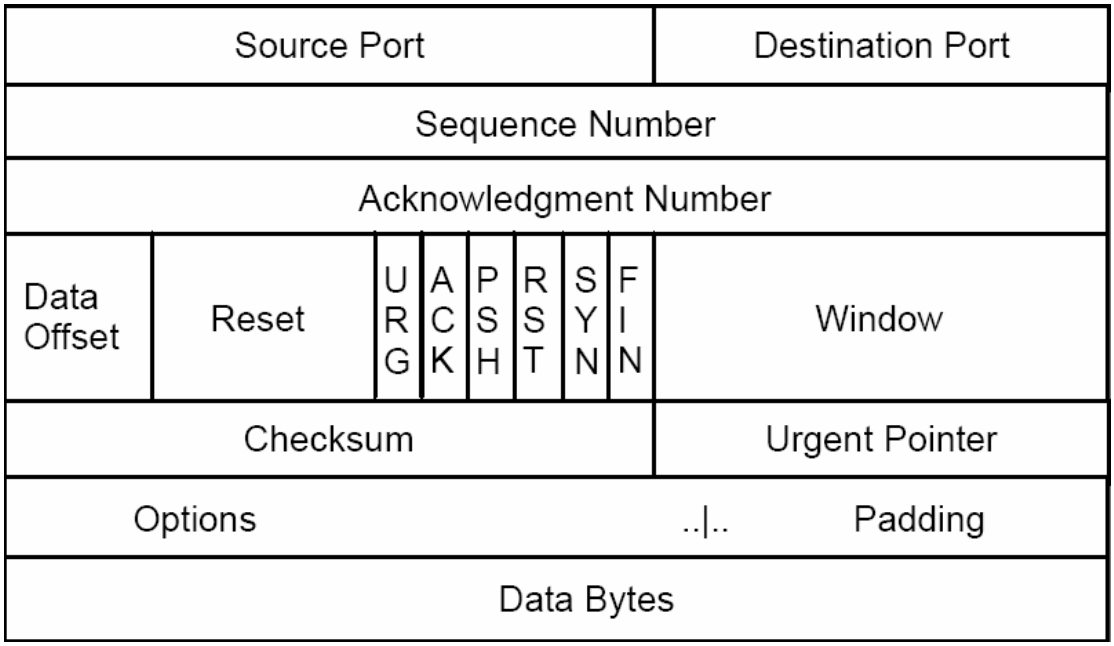
Length: The Length (in bytes) of this user datagram, including the header.

Checksum: The checksum of the data + the UDP header + a “pseudo header” containing IP Source Address, IP Destination Address, Protocol and UDP Length.

Here are some standard application layers that use UDP:

- Trivial File Transfer Protocol (TFTP)
- Domain Name System (DNS)
- Network File System (NFS)
- Remote Procedure Call (RPC)
- Simple Network Management Protocol (SNMP)
- Lightweight Directory Access Protocol (LDAP)

TCP does much more than UDP. It first sets up a connection, and as I said before, it provides reliability, flow control and error correction. TCP is used by the most application layers. This is how a TCP header will look like:



As you can see it's a lot more complicated than the UDP header. A TCP datagram can carry various options. The most important for us are the control bits (also called flags). You can see them in the above picture between the "Reset" and the "Window" fields. All the flags are given 2 possible values (a bit), 0 or 1. If the value is 1 we'll say that the is set. These are the possible flags:

URG: urgent
ACK: acknowledgement
PSH: push
RST: reset
SYN: synchronize
FIN: finish

When the application sends a TCP datagram it also expects an acknowledgement that it was received. If this doesn't come it will send the frame again. The acknowledgement will have the ACK flag set and a sequence number that will allow the server to determine what frames were received.

In TCP, before any data can be transferred, a connection has to be established between the client and the server. This is called a three-way handshake and looks likes this:

Client	SYN ->	Server
Client	<- SYN ACK	Server
Client	ACK ->	Server

First the client will send to the server port (let's say 80) a simple TCP datagram with the SYN flag set. If the server accepts connections on that port (in this case HTTP requests) will respond with a frame with both flags SYN and ACK set. Then the client will send a frame with the ACK flag set and the

connection is established. It may now make requests to the HTTP daemon[1] from that server.

If the server didn't accept TCP connections on that port it'll respond with RST (datagram with the reset flag set).

Client	SYN ->	Server
Client	<- RST	Server

A connection can be terminated with either FIN or RST packets.

Client	FIN ACK ->	Server
Client	<- ACK	Server
Client	<- FIN ACK	Server
Client	ACK ->	Server

or

Client	RST ACK ->	Server

I hope this chapter helped you. Don't forget to read more elaborate tutorials on TCP/IP. I'll get to the non-boring-side of the tutorial now.

[1] Daemon means a program that runs on a server offering a network service (for example HTTPD)

== Becoming a Hacker ==

I'll start this chapter with a famous text which every would-be hacker should read, and every hacker already read. "The Conscience of a Hacker", by Loyd Blankenship aka The Mentor. It appeared in Phrack magazine and it was written on January 8, 1986. Here's the original:

```
                                    ==Phrack Inc.==

                        Volume One, Issue 7, Phile 3 of 10

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
The following was written shortly after my arrest...

                            \/\The Conscience of a Hacker/\/

                                            by

                                        +++The Mentor+++

                                    Written on January 8, 1986
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

        Another one got caught today, it's all over the papers.  "Teenager
Arrested in Computer Crime Scandal", "Hacker Arrested after Bank Tampering"...
        Damn kids.  They're all alike.

        But did you, in your three-piece psychology and 1950's technobrain,
ever take a look behind the eyes of the hacker?  Did you ever wonder what
made him tick, what forces shaped him, what may have molded him?
        I am a hacker, enter my world...
        Mine is a world that begins with school... I'm smarter than most of
the other kids, this crap they teach us bores me...
        Damn underachiever.  They're all alike.

        I'm in junior high or high school.  I've listened to teachers explain
for the fifteenth time how to reduce a fraction.  I understand it.  "No, Ms.
Smith, I didn't show my work.  I did it in my head..."
        Damn kid.  Probably copied it.  They're all alike.

        I made a discovery today.  I found a computer.  Wait a second, this is
cool.  It does what I want it to.  If it makes a mistake, it's because I
screwed it up.  Not because it doesn't like me...
                Or feels threatened by me...
                Or thinks I'm a smart ass...
                Or doesn't like teaching and shouldn't be here...
        Damn kid.  All he does is play games.  They're all alike.

        And then it happened... a door opened to a world... rushing through
the phone line like heroin through an addict's veins, an electronic pulse is
sent out, a refuge from the day-to-day incompetencies is sought... a board is
found.
        "This is it... this is where I belong..."
        I know everyone here... even if I've never met them, never talked to
them, may never hear from them again... I know you all...
        Damn kid.  Tying up the phone line again.  They're all alike...

        You bet your ass we're all alike... we've been spoon-fed baby food at
school when we hungered for steak... the bits of meat that you did let slip
through were pre-chewed and tasteless.  We've been dominated by sadists, or
ignored by the apathetic.  The few that had something to teach found us will-
ing pupils, but those few are like drops of water in the desert.

        This is our world now... the world of the electron and the switch, the
beauty of the baud.  We make use of a service already existing without paying
for what could be dirt-cheap if it wasn't run by profiteering gluttons, and
you call us criminals.  We explore... and you call us criminals.  We seek
after knowledge... and you call us criminals.  We exist without skin color,
without nationality, without religious bias... and you call us criminals.
You build atomic bombs, you wage wars, you murder, cheat, and lie to us
and try to make us believe it's for our own good, yet we're the criminals.
```

```
        Yes, I am a criminal.  My crime is that of curiosity.  My crime is
that of judging people by what they say and think, not what they look like.
My crime is that of outsmarting you, something that you will never forgive me
for.

        I am a hacker, and this is my manifesto.  You may stop this individual,
but you can't stop us all... after all, we're all alike.

                                +++The Mentor+++
```

The text was reprinted in Phrack 14 “in honor and sympathy for the many phreaks and hackers that have been busted recently by the Secret Service”. If you’d like to read more about those times, I’d recommend a great book: Underground, by Suelette Dreyfus & Julian Assange. You can download it from here:
http://www.absolom.ro/index.php?name=Sections&req=viewarticle&artid=4&page=1

Now let’s talk about you becoming a hacker. First you must become paranoid.

1. You shall tell to no one you’ve started hacking, not even to your mother or friends. You may pass the word only if you really thrust your buddies (don’t ever thrust your parents though) and if you’re good enough, that supposedly if FBI was to take your PC from home, they wouldn’t be able to discover anything incriminating on it. They also shouldn’t be able to discover anything incriminating in your house. Don’t buy hacking books and put them in your fucking library. Don’t write passwords on piece of papers etc.
2. You shall change your ISP as often as possible.
3. You shall encrypt your conversations if they’re related to hacking. Instead of Yahoo Messenger, AIM, ICQ or whatever IM program you use, try gaim with the OTR plugin.
For gaim: http://gaim.sourceforge.net/
For OTR (Off-The-Record): http://www.cypherpunks.ca/otr/
Only connect to IRC servers that support SSL (Secure Sockets Layer). Here’s an article about how to add SSL support to MIRC:
http://www.absolom.ro/index.php?name=News&file=article&sid=3
Encrypt you’re e-mails with GnuPG.
http://www.gnupg.org/(en)/download/index.html
Find other anonymity tools here:
http://www.absolom.ro/index.php?name=Sections&req=listarticles&secid=7
4. You shall use encrypted partitions to store your hacking files (even for this tutorial).
http://www.absolom.ro/index.php?name=Sections&req=viewarticle&artid=61&page=1
5. You shall leave no open ports on your computer. You shall use a powerful firewall and antivirus.
6. You shall encrypt your swap file.
http://www.absolom.ro/index.php?name=Sections&req=viewarticle&artid=61&page=1
7. You shall never delete files rather than wipe them.
http://www.absolom.ro/index.php?name=Sections&req=viewarticle&artid=61&page=1
8. You shall never use Telephone, Mobile Phone or VoIP to discuss about hacking.
9. If possible, you shall connect to the internet through a Wi-Fi or Bluetooth hotspot, or using a WLAN gateway you hacked by war driving, or from an

Internet Cafe, or using another's man internet connection and maybe his computer too.
10. You shall have a quick way to destroy your data. Watch The Broken Episode 3 – return of thebroken for a nice one.
http://videos.revision3.com/thebroken/thebroken3.avi
or
http://www.archive.org/download/thebroken3/thebroken3.avi

With time you'll be able to think about more paranoid things like those I just mentioned. Never be too lazy to actually use them.

== WHOIS Databases ==

Let's choose a target for our hacking adventure (you paranoid freak). You may want to make it legal so request approval from the target to try hacking it. You probably won't receive it so try anyway (just joking, don't sue me please).

For the first chapters our target will be NASA Ames Research Center. Our evil thought will be to hack any of their severs accessible from the internet, and then get access to any of their internal networks. If you just wanted to deface their website, the start would have been to open it in your browser.

But defacing is bad and useless and hopefully you knew that already. What we want to do is gather as much information as possible about the internet side of their network, without interfering too much with any of their servers. Our first tool will be the public WHOIS databases. These contain some useful records about each domain, but you can also query them sometimes for registrars, nameservers, network address spaces, etc.

You will use the WHOIS search from websites like these:

Africa Network Information Centre (Africa, portions of the Indian Ocean)
http://www.afrinic.net/cgi-bin/whois

American Registry for Internet Numbers (Canada, United States, islands in the Caribbean Sea and North Atlantic Ocean)
http://ws.arin.net/cgi-bin/whois.pl

Asia Pacific Network Information Centre (Portions of Asia, portions of Oceania)
http://www.apnic.org/search/index.html

Latin American and Caribbean Network Information Centre (Latin America, portions of the Caribbean)
http://lacnic.net/cgi-bin/lacnic/whois

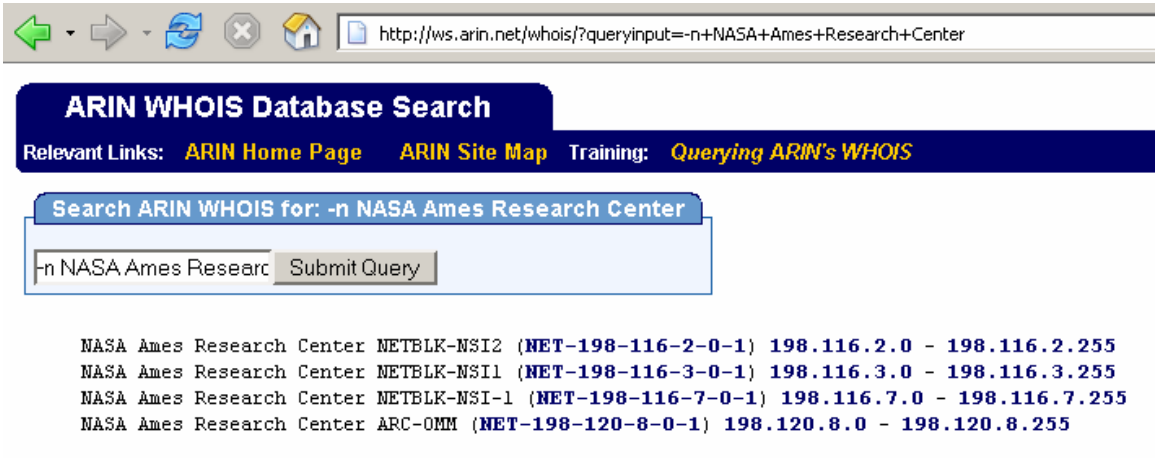
Réseaux IP Européens (Europe, the Middle East, Central Asia)
http://www.ripe.net/perl/whois/

Internet Network Information Centre
http://www.internic.net/whois.html

For NASA Ames Research Center we will query ARIN since the organization is located in USA. There are two easy ways to do it: use the search engine provided by ARIN or use the Linux whois utility. We'll do it both ways.

In the first example I used this query: "-n NASA Ames Research Center". The –n parameter indicates that we only want to see the network

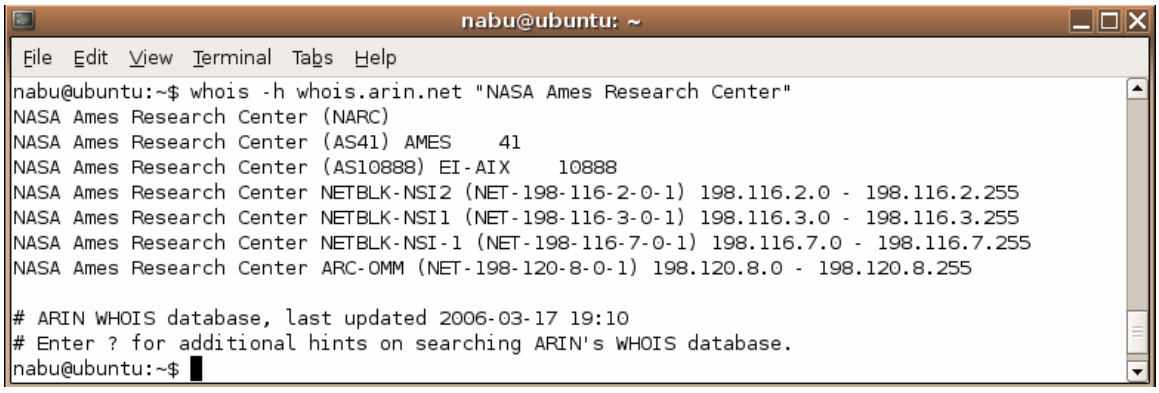
address space record types. This means we only want to know what IP addresses are assigned to NASA ARC.



In this case there are lots of IP addresses organized in 4 net blocks: 198.116.2.*, 198.116.3.*, 198.116.7.* and 198.120.8.*. This means we have about 1016 possible hosts (254*4) to attack. The IPs ending with 0 or 255 don't count because they usually are not assigned to a host, but generally it's a good idea to test them too. Those ending with 0 are reserved for the network address[1] and those ending with 255 for the broadcast address[2].

In Linux we could have found the netblocks with this command:

$ whois –h whois.arin.net "NASA Ames Research Center"



In other UNIX-based operating systems this could have also worked:

$ whois –a "NASA Ames Research Center"

You should always read the manual of a command if you never used it before.

$ man whois

[1] A network address represents the network portion of an IP address.

[2] A broadcast address is an IP address that allows information to be sent to all machines on a given subnet rather to a specific machine.

== Basic Tracerouting and Path Analysis ==

Now we'll try to understand the topology of the internet side of the ARC network with different tools and techniques. We'll start with a program called traceroute in UNIX and tracert in Windows. This is a TCP/IP utility which should allow us to determine the route that IP datagrams take to reach a particular host. How does traceroute work? It's simple. When you execute the traceroute command (example: traceroute 65.161.97.151) your machine sends out, one by one, 3 UDP packets with a TTL (Time-To-Leave) value of 1 and a destination address set to the target host (in our example 65.161.97.151). The first router that will have to forward the UDP datagram to its destination will decrease the TTL value to 0 and therefore drop it. It will send you back an ICMP Time-To-Live Exceeded message (Type 11, Code 0 – TTL exceeded in transit, check Appendix A for all ICMP Types and Codes), with a source address of itself, therefore you will know the IP address of the first hop in the path to the target host. Next, traceroute will send 3 UDP packets with a TTL value of 2, thus the first router you already know will pass it to the next router, which will drop it and send you back an ICMP message Type 11, Code 0 with its source address. You'll have the second router in the path to your target. The traceroute program will continue doing that until it reaches the final destination. Since some NASA routers/firewalls/whatever will drop incoming UDP/outgoing ICMP traffic, we won't have the complete route but we'll try to guess it by other means. This "other means" are actually those that will help us map the network, but we have to study the traditional tools first (sorry). Let's go to http://www.traceroute.org and select a location as close as possible to the servers we're targeting. In this case I chose http://www.traceroute.org/#USA, and then http://voa.his.com/cgi-bin/trace. First we'll request a traceroute that we know it should be successful (because we did it before and it worked). Let's say www.whitehouse.gov.

FROM voa.his.com TO www.whitehouse.gov.

```
traceroute to a1289.g.akamai.net (65.161.97.151), 64 hops max, 44 byte packets
 1  pm28-fe00.his.net (216.194.225.65)  0.801 ms  0.533 ms  0.481 ms
 2  att-kensington-ds3-1.his.net (216.194.224.6)  1.442 ms  1.334 ms  1.296 ms
 3  12.124.234.33 (12.124.234.33)  1.493 ms  1.569 ms 12.124.234.85 (12.124.234.85)
1.557 ms
 4  12.123.9.58 (12.123.9.58)  2.686 ms  2.651 ms  2.242 ms
 5  12.122.82.221 (12.122.82.221)  1.816 ms  1.556 ms  1.598 ms
 6  att-gw.dc.sprint.net (192.205.32.166)  2.599 ms  2.365 ms  2.699 ms
 7  sl-st20-ash-12-0.sprintlink.net (144.232.19.240)  2.666 ms  2.616 ms  2.580 ms
 8  65.161.97.151 (65.161.97.151)  2.632 ms  3.072 ms  2.531 ms
```

Since www.whitehouse.gov is a domain alias for a1289.g.akamai.net (65.161.97.151), we are actually tracerouting to 65.161.97.151. What is a domain alias? Well, it's a way to have separate domains pointing to the same server. I'll help you understand. When you request the White House website in your browser, what you're actually doing is that you connect to 65.161.97.151 on port 80 and request the index page for the virtual host www.whitehouse.gov. Virtual hosting is a method that web servers use to host more than one domain name on the same computer and IP address. An easy way to find aliases and other informations about a domain is the Linux dig command. We won't discuss it now because you can always read its manual if you want more informations about it.

$ man dig

```
nabu@ubuntu:~$ dig www.whitehouse.gov

; <<>> DiG 9.3.1 <<>> www.whitehouse.gov
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54555
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 9, ADDITIONAL: 9

;; QUESTION SECTION:
;www.whitehouse.gov.            IN      A

;; ANSWER SECTION:
www.whitehouse.gov.     7055    IN      CNAME   www.whitehouse.gov.edgesuite.net.
www.whitehouse.gov.edgesuite.net. 155 IN CNAME  a1289.g.akamai.net.
a1289.g.akamai.net.     20      IN      A       80.86.97.88
a1289.g.akamai.net.     20      IN      A       80.86.97.87

;; AUTHORITY SECTION:
g.akamai.net.           1594    IN      NS      n5g.akamai.net.
g.akamai.net.           1594    IN      NS      n6g.akamai.net.
g.akamai.net.           1594    IN      NS      n7g.akamai.net.
g.akamai.net.           1594    IN      NS      n8g.akamai.net.
g.akamai.net.           1594    IN      NS      n0g.akamai.net.
g.akamai.net.           1594    IN      NS      n1g.akamai.net.
g.akamai.net.           1594    IN      NS      n2g.akamai.net.
g.akamai.net.           1594    IN      NS      n3g.akamai.net.
g.akamai.net.           1594    IN      NS      n4g.akamai.net.

;; ADDITIONAL SECTION:
n0g.akamai.net.         235     IN      A       193.231.255.66
n1g.akamai.net.         1757    IN      A       193.231.255.74
n2g.akamai.net.         1599    IN      A       193.231.255.75
n3g.akamai.net.         235     IN      A       193.231.255.66
n4g.akamai.net.         1759    IN      A       193.231.255.74
n5g.akamai.net.         1651    IN      A       193.231.255.75
n6g.akamai.net.         401     IN      A       205.188.162.148
n7g.akamai.net.         235     IN      A       205.188.162.148
n8g.akamai.net.         401     IN      A       205.188.162.148

;; Query time: 210 msec
;; SERVER: 192.168.228.2#53(192.168.228.2)
;; WHEN: Sun Mar 19 20:47:41 2006
;; MSG SIZE  rcvd: 449

nabu@ubuntu:~$
```

Now that we know that http://voa.his.com/cgi-bin/trace works fine, what we want to do is traceroute every single IP from our target network. We'll do it to a few IPs to understand what's happening, then we'll make a script to automate the task. We'll start with 198.116.2.1:

FROM voa.his.com TO 198.116.2.1.

```
traceroute to 198.116.2.1 (198.116.2.1), 64 hops max, 44 byte packets
 1  pm28-fe00.his.net (216.194.225.65)  0.782 ms  0.558 ms  0.490 ms
 2  att-kensington-ds3-1.his.net (216.194.224.6)  1.501 ms  1.343 ms  1.303 ms
 3  12.124.234.33 (12.124.234.33)  1.588 ms 12.124.234.85 (12.124.234.85)  1.624 ms
12.124.234.33 (12.124.234.33)  1.478 ms
 4  12.123.9.58 (12.123.9.58)  63.419 ms  63.246 ms  63.038 ms
 5  tbr1-cl4.sl9mo.ip.att.net (12.122.10.30)  65.568 ms  65.244 ms  65.573 ms
 6  tbr2-cl2.sffca.ip.att.net (12.122.10.42)  65.378 ms  65.252 ms  65.359 ms
 7  gbr2-p40.sffca.ip.att.net (12.122.11.86)  62.948 ms  62.942 ms  63.071 ms
 8  nr1-p360.napsf.ip.att.net (192.205.31.38)  64.064 ms  63.588 ms  63.539 ms
 9  ames1.mae-west.nasa.gov (198.32.136.43)  63.777 ms  63.832 ms  63.724 ms
10  128.161.3.94 (128.161.3.94)  64.268 ms  64.096 ms  63.722 ms
```

```
11  192.150.40.245 (192.150.40.245)  376.877 ms  420.183 ms  435.923 ms
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
Etc. lots of *s
```

As you can see after the hop 11, UDP/ICMP packets are being dropped or some stupid router is sending us ICMP responses with a TTL value to low to reach us, or the PC with IP 198.116.2.1 is switched off. It doesn't really matter if it's on or off on this stage of the attack. Check hop 9. That's an IXP (Internet Exchange Point). We can discover this by searching this IP in the ARIN WHOIS database. It will show it as part of the Exchange Point Blocks. The primary purpose of an IXP is to allow networks to interconnect directly, via the exchange, rather than through one or more 3rd party networks. The advantages of the direct interconnection are numerous, but the primary reasons are cost, latency and bandwidth. We should have expected NASA networks to be interconnected with Internet Exchange Points. This is not surprising at all.

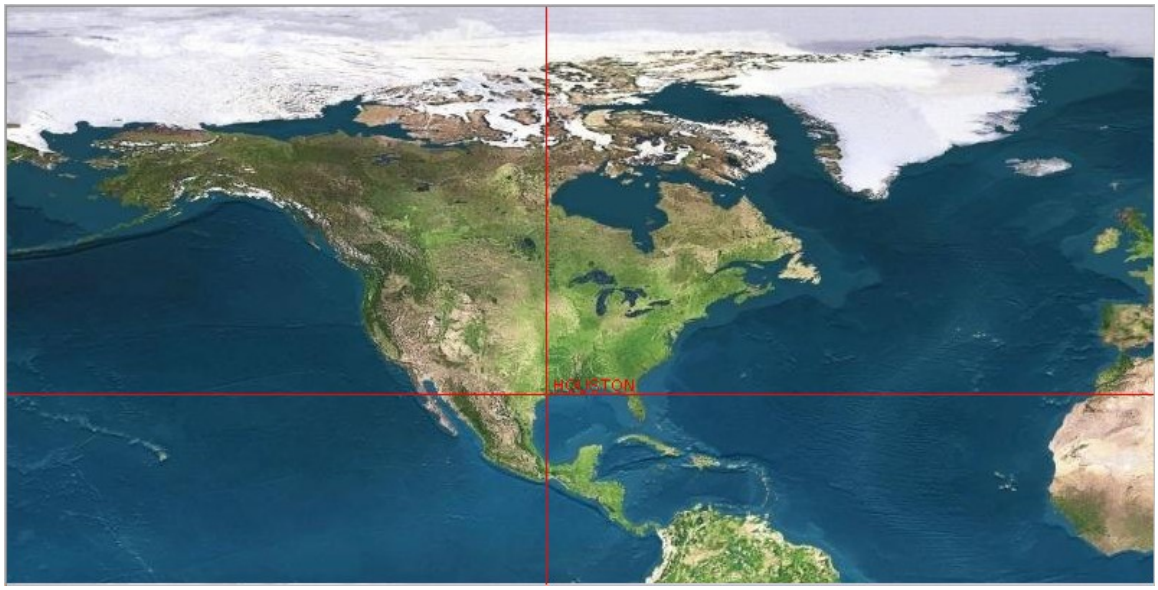
Now check hops 10 and 11. Those IPs don't have a reverse DNS entry. This means that they are not associated with a domain or subdomain (like hop 9 for example). That makes our job a bit harder because we can't easily understand the purpose of those devices or computers in the network. By querying ARIN we discover them as part of the National Aeronautics and Space Administration IP range. That's normal since for getting to them we had to go through a NASA IXP. We won't query ARIN from now on any more, unless we see other IXPs or we don't see an IXP at all. Hops 10 and 11 don't belong to any of the ARC net blocks so they're not that important to us.

Let's suppose that hop 11 was the last one and it was the IP we were trying to traceroute. Let's have a better understanding of its location by watching a map. Go to http://www.ip2location.com/free.asp and input the 3 addresses from the last 3 successful hops in the IP Address(es) box you'll see there. Then press the button "Find Location" and it will show you something like this:

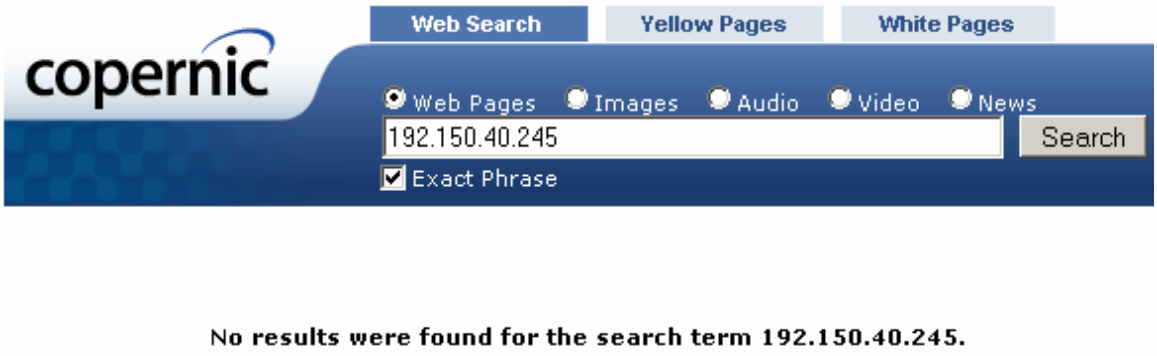
IP Address	Country (Short)	Country (Full)	Flag	Region	City	ISP	Map
198.32.136.43	US	UNITED STATES		TEXAS	WEBSTER	EXCHANGE POINT BLOCKS	MAP IT!
128.161.3.94	US	UNITED STATES		ALABAMA	HUNTSVILLE	NATIONAL AERONAUTICS AND SPACE ADMINISTRATION	MAP IT!
192.150.40.245	US	UNITED STATES		TEXAS	HOUSTON	NATIONAL AERONAUTICS AND SPACE ADMINISTRATION	MAP IT!

By pressing MAP on the last IP we'll know that our target computer is located near that point, in Houston, Texas.

If the real trace would have been complete we could have known the exact position of the real IP we were tracerouting (198.116.2.1), but this will happen rarely in a government address space, especially a NASA network. Putting each target IP on a map will always help us understand the structure of the internet side of a network in a better way.



If by using http://www.copernic.com/ and searching for the two IPs from hops 10 and 11 we discover nothing interesting about them, we should not try any more, since as I said they're not part of the ARC address space anyway. But keep in mind that search engines may help you when you least expect it.



The advantage of www.copernic.com is that it displays results from more search engines for a single query.

Next we'll traceroute 198.116.3.1, 198.116.7.1 and 198.120.8.1 and I'll delete the unimportant hops.

```
traceroute to 198.116.3.1 (198.116.3.1), 64 hops max, 44 byte packets
9  ames1.mae-west.nasa.gov (198.32.136.43)  63.622 ms  63.965 ms  162.181 ms
10  128.161.3.94 (128.161.3.94)  63.945 ms  63.881 ms  64.057 ms
11  128.161.1.94 (128.161.1.94)  70.668 ms  70.463 ms  70.607 ms
12  * * *
13  * * *

traceroute to 198.116.7.1 (198.116.7.1), 64 hops max, 44 byte packets
9  ames1.mae-west.nasa.gov (198.32.136.43)  64.223 ms  65.599 ms  64.507 ms
10  128.161.3.94 (128.161.3.94)  64.540 ms  64.801 ms  64.546 ms
11  128.161.21.54 (128.161.21.54)  65.855 ms  66.596 ms  65.809 ms
12  * * *
13  * * *

traceroute to 198.120.8.1 (198.120.8.1), 64 hops max, 44 byte packets
9  ames1.mae-west.nasa.gov (198.32.136.43)  64.621 ms  64.557 ms  64.784 ms
10  128.161.3.94 (128.161.3.94)  64.209 ms  64.224 ms  64.444 ms
11  * * *
12  * * *
```

We only see two new IPs discovered, both in US, not far away from each other, one in Huntsville, Alabama and the other in Houston, Texas (again). We're not doing a great job with this traces but we'll do an auto-trace script anyway. This might be helpful for you in the future, with other, less firewalled, networks. It will help you traceroute an IP range and when finished you'll be hopefully able to extract some interesting informations about the network you're targeting, from its logs. If might not help you too much with the ARC network but don't worry. We're just at the beginning anyway.

How do you analyze the results of a traceroute? Let's look at an example. We'll presume that our target was www.software_for_shoes.com and that the company had 3 IP addresses assigned: 198.1.5.55, 198.1.5.56, 198.1.5.57 and 198.1.5.58

After tracerouting to each of the 4 IPs we could discover something like this:

Traceroute from whatever host to 198.1.5.55

```
…
6  att-gw.dc.sprint.net (192.205.32.166)
7  firewall.software_for_shoes.com (198.1.5.55)
```

Traceroute from whatever host to 198.1.5.56

```
…
6  att-gw.dc.sprint.net (192.205.32.166)
7  firewall.software_for_shoes.com (198.1.5.55)
8  www.software_for_shoes.com (198.1.5.56)
```

Traceroute from whatever host to 198.1.5.57

```
…
6  att-gw.dc.sprint.net (192.205.32.166)
7  firewall.software_for_shoes.com (198.1.5.55)
8  smtp.software_for_shoes.com (198.1.5.57)
```

Traceroute from whatever host to 198.1.5.58

```
…
6  att-gw.dc.sprint.net (192.205.32.166)
7  firewall.software_for_shoes.com (198.1.5.55)
8  ns.software_for_shoes.com (198.1.5.57)
```

So now it's easy to understand that we shouldn't try to hack 198.1.5.55 since it's a firewall device/host, which protects the web server, the SMTP server and the name server. We can see that each connection to these 2 daemons will be filtered by 198.1.5.55, since our datagrams are routed through the firewall.

"att-gw.dc.sprint.net" is the router of the internet provider for "Software for Shoes", so it's of no interest for us. We can recognize the ISP as Sprintlink (sprint.net). The naming of the server also gives away informations about it:

dc is a city code – Washington DC

gw – gateway router

att –the fabricant of the router (AT&T)

I hope you understood how useful tracerouting can be in some hacking scenarios. Not let's have a look at the basic tools that can be used for tracerouting. In Windows the program is called tracert. In most distros you'll have traceroute. In Ubuntu Linux I don't have it so I'll have to install it first. I already have tracepath installed, but that's like the stupid version of traceroute so we won't use it.

$ sudo apt-get install traceroute

Once you have traceroute installed you should read its manual.

$ man traceroute

Using it is easy. Most of the time you'll do it like this:

$ traceroute www.vodafone.com

Or

$ traceroute 195.233.125.5

Here's another example of use:

$ traceroute –w 10 –q 2 –m 40 –v www.vodafone.com &>traceroute_www.vodafone.com &

-w 10 means that we change the timeout interval from the default of 5 into 10 seconds. The probe has more time to come back. If it doesn't come back "*" will be printed. By default each probe is sent on each route 3 times. With -q 2 we only send it twice. –m 40 means that after 40 hops without reaching the destination, the program will quit. –v means verbose mode. www.vodafone.com is our target. We are saving the results of the traceroute in a file called traceroute_www.vodafone.com and the entire tracing process is being done in the background (&). We can check the progress anytime with the cat command:

$ cat traceroute_www.vodafone.com

You can read about tracepath too. There are just a few rows. You'll probably observe that you have tracepath6 and traceroute6 installed too. We won't use those because they're for IPv6[1]. Play for a few minutes with the traceroute command until you truly understand it. When done continue reading… we'll be making our auto-tracerouting script.

Create a directory called mapping and inside it a file called trace.pl with the following content:

```
#!/usr/bin/perl
@tag=split(/-/,@ARGV[0]);
@ip1=split(/\./,@tag[0]);
@ip2=split(/\./,@tag[1]);
for ($a=@ip1[0]; $a<1+@ip2[0]; $a++) {
for ($b=@ip1[1]; $b<1+@ip2[1]; $b++) {
for ($c=@ip1[2]; $c<1+@ip2[2]; $c++) {
for ($d=@ip1[3]; $d<1+@ip2[3]; $d++) {
print "$a.$b.$c.$d\n";
system "lynx -dump http://voa.his.com/cgi-bin/trace?$a.$b.$c.$d &>trace_$a.$b.$c.$d &";
}}}}
```

Now there are different ways to execute it. One:

$ perl trace.pl 198.116.2.1-198.116.2.20

Or you could make it executable…

$ chmod +x trace.pl

and run it with this command:

$ ./trace.pl 198.116.2.1-198.116.2.20

Or you could copy it in a directory situated in your PATH, and run it from any directory without putting "./" in front of it.

[1] The Internet Protocol version 6 is a network layer standard which governs the addressing and routing of data packets through a network. This version of the internet protocol is destined to replace version 4, currently in use in most of the networks. It is improved upon version 4 in two main areas: much larger addresses (128 bits, allowing for exponentially more hosts on the Internet) and extensibility. The base IPv6 header is 40 bytes.

$ echo $PATH

Now choose a directory (for example “/usr/bin”) and copy it:

$ cp trace.pl /usr/bin

And run it like this:

$ trace.pl 198.116.2.1-198.116.2.20

What I’ve just said are basic Linux stuff. If you’ve read them with interest then please stop reading this tutorial for a few hours and learn how to use the Linux console.

http://www.lowfatlinux.com/linux-basics.html

If you want our script to work you must have perl installed. Most distros have it. Lynx is needed too. On my Ubuntu I didn’t have it installed so…

$ sudo apt-get install lynx

You probably understood what the script does. If not, take a perl tutorial and learn at least the basics. Perl is an interpreted programming language. Its main advantage is that you won’t have to compile programs made in perl. You will simply run them with the perl interpreter.

http://www.google.com/search?hl=en&q=perl+tutorial

Our perl script (trace.pl) takes each IP in the given range, and uses an online traceroute utility for tracerouting to that IP. It writes the result in a file called trace_IP. So by doing

$ ./trace.pl 198.116.2.1-198.116.2.20

we’ll soon end up with lots of files like these: trace_198.116.2.1, trace_198.116.2.1, trace_198.116.2.2, etc. These tracings are background processes and we don’t want to flood the server with trace requests so we’ll take a 20 IP range at a time.

First:

$ ./trace.pl 198.116.2.1-198.116.2.20

After a while we’ll use the ps aux command to see if the tracerouting is over. If it is, we’ll start with the next IPs.

$ ./trace.pl 198.116.2.21-198.116.2.40

And so on until you finish tracerouting your entire target network. When it’s over you’ll analyze the resulted files. I used these IP ranges as an example. Don’t use our script for any of the ARC class C networks. We won’t map them with traceroute.

Another simple way to traceroute to a certain target is using route servers[1]. With their help we can also determine which networks are routed to the internet and which are not. These servers have thousands of routes. You simply connect to them using telnet, and then type “help” or “?” to be shown the available commands.

Here is an example where I connect to route-server.bti.net.ph, I do a traceroute, a ping and then I use the route server as a jump box for connecting to another server.

[1] These servers run one or more network layer routing protocols and provide users the ability to query for routing descriptions

```
nabu@altos:~$ telnet route-server.bti.net.ph
Trying 203.115.134.70...
Connected to route-server.bti.net.ph.
Escape character is '^]'.
C
******************************************************
     Welcome to the Bayan Telecommunications Inc.
                 Route View Server

          TELNET to route-server.bti.net.ph

  For comments email: IP Operations ip-ops@skyinet.net

******************************************************


route-server.bti.net.ph>traceroute ip www.whitehouse.gov
Translating "www.whitehouse.gov"...domain server (202.78.97.2) [OK]

Type escape sequence to abort.
Tracing the route to a1289.g.akamai.net (60.254.131.10)

  1 ip-ops-lan-gw.bti.net.ph (202.78.98.1) [AS 6648] 0 msec 4 msec 0 msec
  2 202.78.97.215 [AS 6648] 4 msec 4 msec 0 msec
  3 Vlan111.msfc1.QBY-Quezon.teleglobe.net (64.86.127.13) [AS 6453] 4 msec 0 msec 4 msec
  4 if-2-0-0.bb1.QBY-Quezon.teleglobe.net (64.86.127.134) [AS 6453] 0 msec 0 msec 4 msec
  5 if-2-2.core1.HK2-HongKong.teleglobe.net (216.6.95.81) [AS 6453] 36 msec 36 msec 40 msec
  6 ix-0-0.core1.HK2-HongKong.teleglobe.net (216.6.95.54) [AS 6453] 40 msec 40 msec 40 msec
  7 a1289.g.akamai.net (60.254.131.10) [AS 20940] 40 msec 40 msec 40 msec
route-server.bti.net.ph>ping ip 216.239.59.104

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 216.239.59.104, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 360/365/372 ms
route-server.bti.net.ph>telnet sdf-eu.org
Trying sdf-eu.org (192.94.73.35)... Open
Kerberos:       No default realm defined for Kerberos!

sdf-eu.org (ttyp7)
if new, login 'new' ..

login: bossacct
Password:
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004
    The NetBSD Foundation, Inc.  All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
    The Regents of the University of California.  All rights reserved.

You have new mail.
$
```

Here's another example where I connect to server.ip.att.net and I check the route table for our ARC networks.

```
nabu@altos:~$ telnet route-server.ip.att.net
Trying 12.0.1.28...
Connected to route-server.cbbtier3.att.net.
Escape character is '^]'.
CCC
############## route-server.ip.att.net ###############
#########  AT&T IP Services Route Monitor  ############

This router maintains peerings with customer-facing routers
throughout the AT&T IP Services Backbone:

 12.123.21.243   Atlanta, GA            12.123.133.124  Austin, TX
 12.123.41.250   Cambridge, MA          12.123.5.240    Chicago,IL
 12.123.17.244   Dallas, TX             12.123.139.124  Detroit, MI
 12.123.37.250   Denver, CO             12.123.134.124  Houston, TX
 12.123.29.249   Los Angeles, CA        12.123.1.236    New York, NY
 12.123.33.249   Orlando,FL             12.123.137.124  Philadelphia, PA
 12.123.142.124  Phoenix, AZ            12.123.145.124  San Diego, CA
 12.123.13.241   San Francisco, CA      12.123.25.245   St. Louis, MO
 12.123.45.252   Seattle, WA            12.123.9.241    Washington, DC

This router has the global routing table view from each of the above
routers, providing a glimpse to the Internet routing table from the
AT&T network's perspective.

*** Please Note:

Ping and traceroute delay figures measured with this box are unreliable,
due to the high CPU load this box experiences when complicated "show" commands
are being executed.

For questions about this route-server, send email to: jayb@att.com

#################### route-server.ip.att.net ####################

route-server>show ip route | include 198.116
B    198.116.25.0/24 [20/0] via 12.123.1.236, 5w3d
B    216.198.116.0/24 [20/0] via 12.123.1.236, 7w0d
B    198.116.0.0/14 [20/0] via 12.123.1.236, 2w5d
route-server>show ip route | include 198.120
B    216.198.120.0/24 [20/0] via 12.123.1.236, 4w6d
B    198.120.0.0/14 [20/0] via 12.123.1.236, 2w5d
route-server>
```

Here are some route servers to play with. Don't waste too much time on the subject.

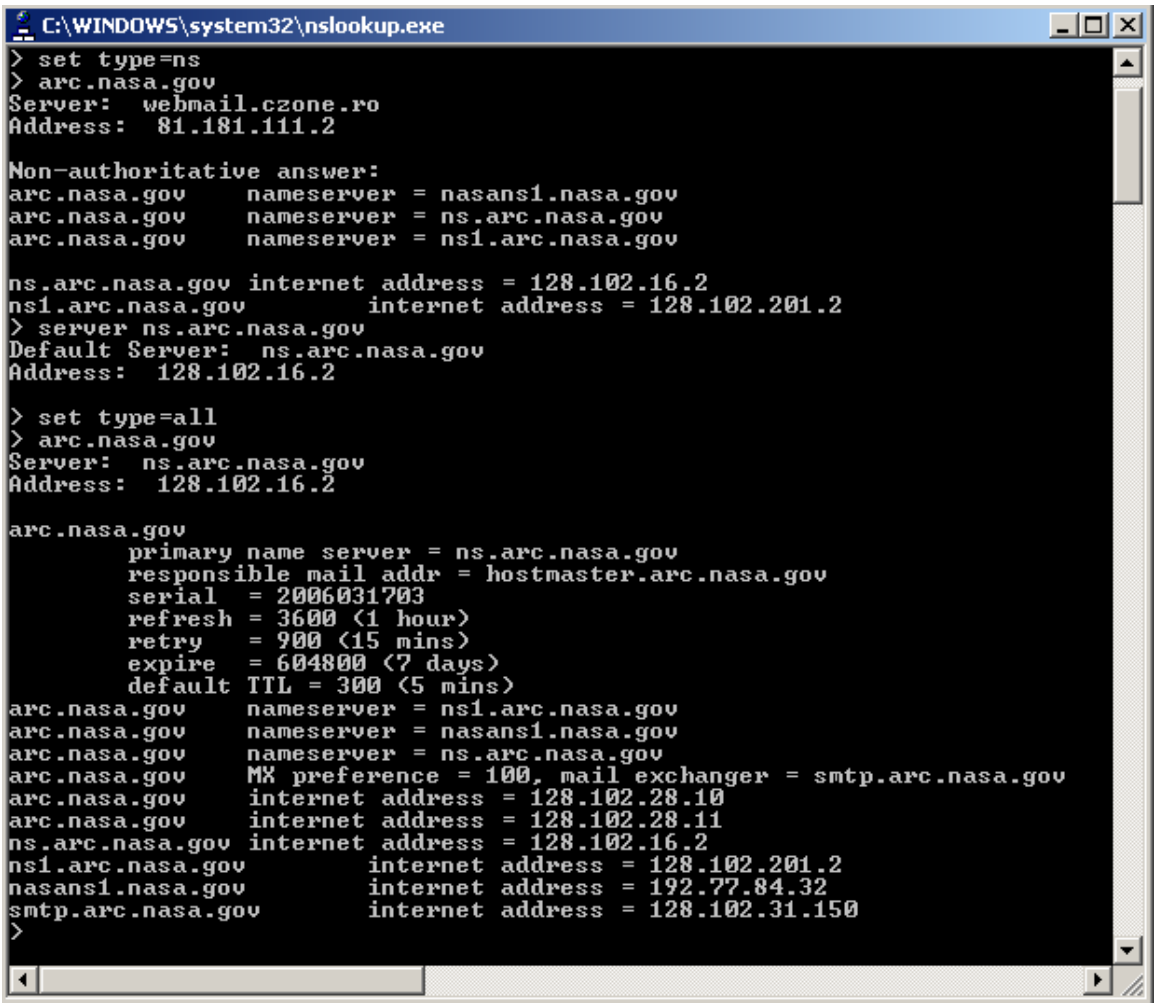
For IPv4:
route-server.belwue.de
route-views.on.bb.telus.com
route-views.ab.bb.telus.com
route-server.cerf.net
route-server.ip.tiscali.net
route-server.gblx.net
route-server.eu.gblx.net
route-server.savvis.net
public-route-server.is.co.za
route-server.twtelecom.net
route-server.as5388.net
route-server.opentransit.net
tpr-route-server.saix.net
route-views.routeviews.org

route-views2.routeviews.org
route-views3.routeviews.org
route-views.eqix.routeviews.org
route-views.isc.routeviews.org
route-views.kixp.routeviews.org
route-views.linx.routeviews.org
route-views.wide.routeviews.org
route-server.gt.ca
route-server.bti.net.ph
route-server.as6667.net
routeserver.sunrise.ch
route-server.he.net
route-server.ip.att.net
route-views.optus.net.au
route-server.wcg.net
route-server.colt.net
route-views.bmcag.net
route-server.manilaix.net.ph
route-server.host.net
route-server.central.allstream.com
route-server.east.allstream.com
route-server.west.allstream.com
route-server.mainz-kom.net
lg.sp.ptt.br
lg.rsix.tche.br
lg.ptt.ansp.br

For IPv6:
route-server.ip.tiscali.net
route-server.opentransit.net
route-views3.routeviews.org
route-views6.routeviews.org
route-views.eqix.routeviews.org
route-views.linx.routeviews.org
route-views.wide.routeviews.org
route-server.he.net
grh.sixxs.net
route-server.host.net

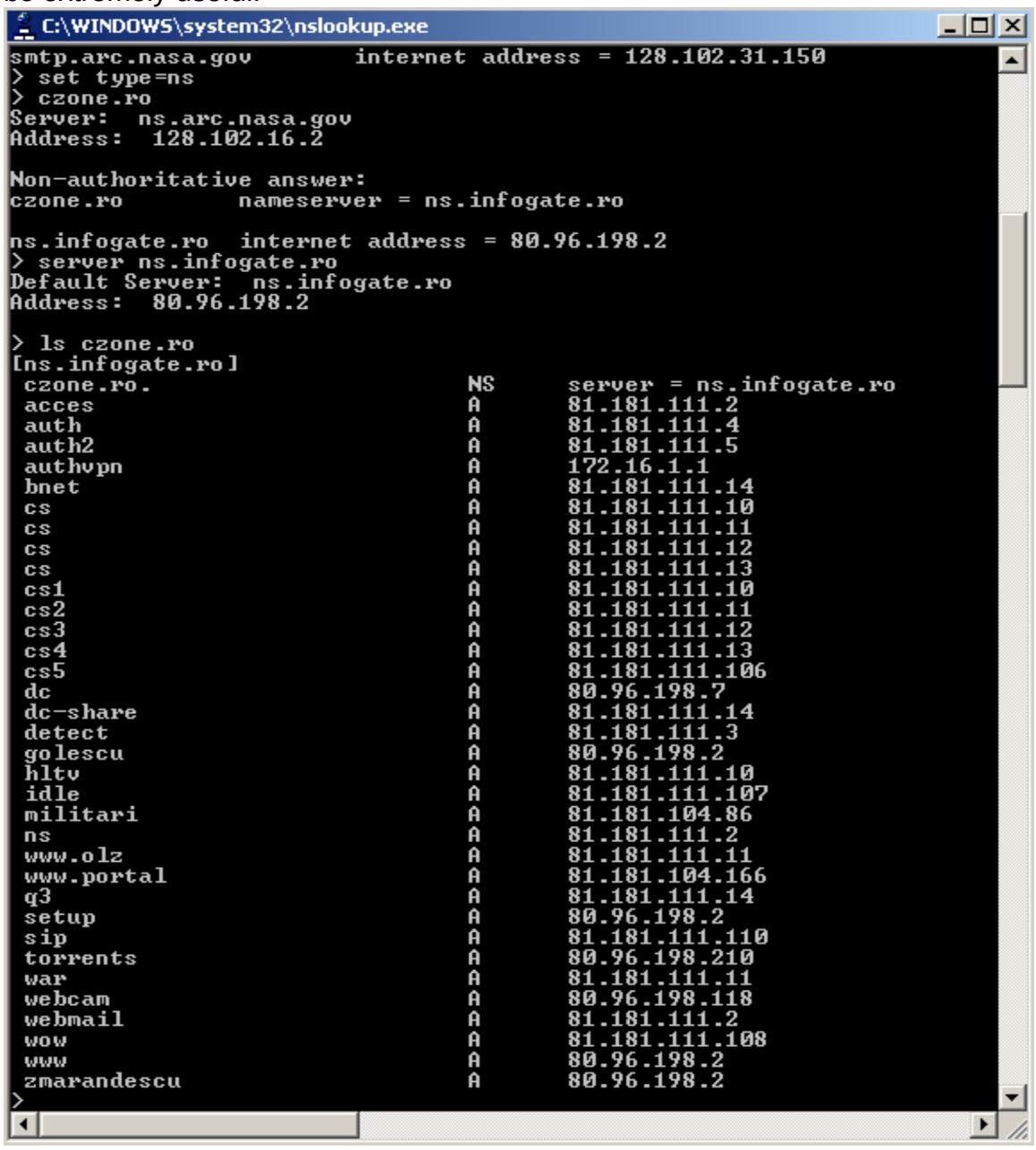
== Mapping with DNS and Geolocation ==

Let's learn some new basic commands. The first one is nslookup. You have it in all operating systems, but I'll use the windows version because it has more implementations. This program is for querying Internet domain name servers (DNS) (like we did with dig). This means it will connect usually on port 53 to whatever server you tell it to connect, and then you can use it in interactive mode to query that name server for informations about various hosts and domains. You can also use it with arguments, if you only want it to respond to a single query. But we don't want that. Type nslookup in your console and hit enter. On the first two lines it will probably show you the DNS it got connected to. That's your primary DNS. You can change the name servers that you're using by editing this file: /etc/resolv.conf (in Linux). The order in which you have them written in it is very important. Most of the people will put local name servers first, because you don't want reverse lookup to be laggy (delayed because of the distance between you and the server). What you should do next is type "?" to be shown the help of the nslookup command. If it's not implemented read the help with man nslookup. Once you've made an idea about how to use it, continue reading the tutorial. I will enter a few commands, then I'll make a screen shot and explain you what I did. Sec…



First I told to the primary DNS that I only want to see the name servers of the domain that I'm about to input (set type=ns). The domain was actually a subdomain… arc.nasa.gov. The primary DNS responded with the 3 nameservers used by arc.nasa.gov. I connected to one of them (server ns.arc.nasa.gov) and made it show me all its records (set type=all) related to arc.nasa.gov. Strangely none of the IPs we got were in the ARC address space. Usually at least the mail server is, but this is NASA… everything is bothersome. Anyway… supposedly we wanted to intercept e-mails sent and received by people having addresses like marvin_christensen@arc.nasa.gov… then we'd probably want to hack the mail exchanger smtp.arc.nasa.gov. Not an easy job, but nothing is impossible.

What we'll be doing next is called a DNS zone transfer, and there is no hacker alive to not have done this at least once in his hacking adventures. It only works when the DNS is configured by an idiot, but when it works it can be extremely useful.

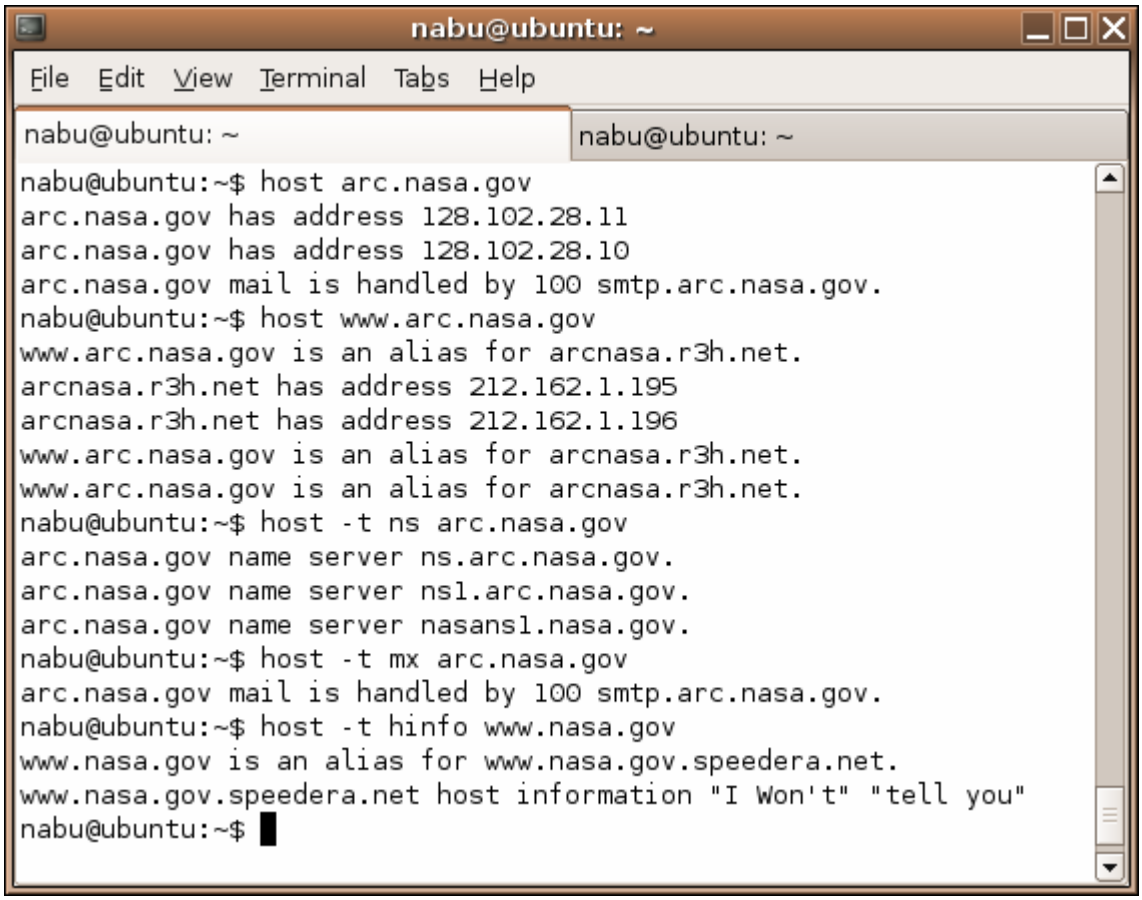


Here we've connected to the DNS server for the domain czone.ro, and then made it show us all its subdomains. C-zone is an internet provider. If we were evil and this was our ISP, we could have started to analyze each subdomain. Very soon we would have observed that http://auth.czone.ro is used for administrators' login, we would have hacked it with (let's say) SQL injection then we'd have paid for our internet service without actually paying, if you know what I mean…

Anyway… I hope you understood what could be the use of nslookup when hacking. I'll present you another tool.

$ man host

Windows doesn't have "host", but you'll find it in any UNIX-based OS. Basically this is the modern version of nslookup and it kind of does the same things. Let's look at some examples.



I won't explain what I did… it's just some basic stuff. You'll understand them because I already told you about nslookup and you'll observe the similitudes. What's new is the HINFO DNS query type, which means host information. You could have done the same thing with nslookup. The website www.nasa.gov is hosted by Speedera Networks. When you see a website hosted by Speedera Networks or by Akamai Technologies you should have a second thought about hacking it if you're lame. As you can see the Speedera technicians have a sense of humor since the host information is "I Won't" "tell you". What exactly are they not telling us? The system architecture and the operating system.

Let's make another script by modifying trace.pl

$ cp trace.pl reverse.pl

$ vi reverse.pl

Here's reverse.pl:

```
#!/usr/bin/perl
@tag=split(/-/,@ARGV[0]);
@ip1=split(/\./,@tag[0]);
@ip2=split(/\./,@tag[1]);
for ($a=@ip1[0]; $a<1+@ip2[0]; $a++) {
for ($b=@ip1[1]; $b<1+@ip2[1]; $b++) {
for ($c=@ip1[2]; $c<1+@ip2[2]; $c++) {
for ($d=@ip1[3]; $d<1+@ip2[3]; $d++) {
$mwe = `host $a.$b.$c.$d`;
chop ($mwe);
chop ($mwe);
If (substr($mwe, 0, 4) ne 'Host") {
$mwe =~ s/.*pointer //g;
print "$a.$b.$c.$d = $mwe\n";
}
}}}}
```

And run it like this:

$ perl reverse.pl 198.116.2.0-198.116.2.255 > reverse_ARC

$ perl reverse.pl 198.116.3.0-198.116.3.255 >> reverse_ARC

$ perl reverse.pl 198.116.7.0-198.116.7.255 >> reverse_ARC

$ perl reverse.pl 198.120.8.0-198.120.8.255 >> reverse_ARC

It will make us a list with all the IPs in the ARC address space that have associated a domain, and print that domain. These lines:

If (substr($mwe, 0, 4) ne 'Host") {

$mwe =~ s/.*pointer //g;

are meant to filter the responses, so that the IPs which are not reversible will not appear in the output, and the output will be in the form ip=domain. The two chops remove "." from the end of the query response. If the program doesn't work it's probably because of the filtering, so you'll have to modify it yourself. Learn perl first. I made the script in Ubuntu and it worked fine. When all the reversing is done this is how the reverse_ARC file should look like:

```
198.116.2.4=sisyphus.nsi.nasa.gov
198.116.2.5=nsiss2.nsi.nasa.gov
198.116.2.80=ws0.nsi.nasa.gov
198.116.2.81=ws1.nsi.nasa.gov
198.116.2.82=ws2.nsi.nasa.gov
198.116.2.83=ws3.nsi.nasa.gov
198.116.2.84=ws4.nsi.nasa.gov
198.116.2.90=mcfpv.nsi.nasa.gov
198.116.2.91=mchplj4.nsi.nasa.gov
198.116.2.100=lupine.nsi.nasa.gov
198.116.2.101=dcqmgate.nsi.nasa.gov
198.116.2.103=mikespb.nsi.nasa.gov
198.116.2.104=fugitive.nsi.nasa.gov
198.116.2.105=camelot.nsi.nasa.gov
198.116.2.106=avalon.nsi.nasa.gov
198.116.2.107=grmiller.nsi.nasa.gov
198.116.2.108=jeeves.nsi.nasa.gov
198.116.2.110=phoenix.nsi.nasa.gov
198.116.2.120=whitman.nsi.nasa.gov
```

198.116.2.211=msdoze.nsi.nasa.gov
198.116.2.217=mcn-fbsd.nsi.nasa.gov
198.116.2.218=mcn-ms.nsi.nasa.gov
198.116.2.219=mcn-test1.nsi.nasa.gov
198.116.2.220=mcn-test2.nsi.nasa.gov
198.116.3.21=biz.arc.nasa.gov
198.116.3.22=slim.arc.nasa.gov
198.116.3.23=jonah.arc.nasa.gov
198.116.3.24=aviation-server.arc.nasa.gov
198.116.3.25=mobydick.arc.nasa.gov
198.116.3.26=shamu.arc.nasa.gov
198.116.3.27=workbench.arc.nasa.gov
198.116.3.28=patient-safety.arc.nasa.gov
198.116.3.29=patientdb.arc.nasa.gov
198.116.3.30=psrs-backup.arc.nasa.gov
198.116.3.31=psrs-server.arc.nasa.gov
198.116.3.32=dbtest.arc.nasa.gov
198.116.3.33=titan-server.arc.nasa.gov
198.116.3.34=humphrey.arc.nasa.gov
198.116.3.35=perilog01.arc.nasa.gov
198.116.3.36=perilog02.arc.nasa.gov
198.116.3.37=perilog03.arc.nasa.gov
198.116.3.51=shos-dell.arc.nasa.gov
198.116.3.52=sues-dell.arc.nasa.gov
198.116.3.53=lisas-dell.arc.nasa.gov
198.116.3.54=freeman.arc.nasa.gov
198.116.3.55=harveys-dell.arc.nasa.gov
198.116.3.56=jimmys-dell.arc.nasa.gov
198.116.3.57=andrews-dell.arc.nasa.gov
198.116.3.58=psrs-travel.arc.nasa.gov
198.116.3.59=eds-dell.arc.nasa.gov
198.116.3.60=donnas-dell.arc.nasa.gov
198.116.3.61=barbs-laptop.arc.nasa.gov
198.116.3.62=chucks-laptop.arc.nasa.gov
198.116.3.63=admin-dell.arc.nasa.gov
198.116.3.64=jorges-dell.arc.nasa.gov
198.116.3.65=elizas-dell.arc.nasa.gov
198.116.3.66=pearly-gates.arc.nasa.gov
198.116.3.67=calvins-dell.arc.nasa.gov
198.116.3.68=dons-dell.arc.nasa.gov
198.116.3.69=elisas-laptop.arc.nasa.gov
198.116.3.70=dans-dell.arc.nasa.gov
198.116.3.71=pauls-dell.arc.nasa.gov
198.116.3.72=freds-dell.arc.nasa.gov
198.116.3.73=steves-dell.arc.nasa.gov
198.116.3.76=mikes-dell.arc.nasa.gov
198.116.3.78=jeppview.arc.nasa.gov
198.116.3.79=dons-g4.arc.nasa.gov
198.116.3.80=marianas-g4.arc.nasa.gov
198.116.3.81=calvins-g4.arc.nasa.gov
198.116.3.82=teds-g4.arc.nasa.gov
198.116.3.83=chucks-g4.arc.nasa.gov
198.116.3.84=daves-g4.arc.nasa.gov
198.116.3.85=dans-toy.arc.nasa.gov
198.116.3.86=jeffs-g4.arc.nasa.gov
198.116.3.87=lindas-g4.arc.nasa.gov
198.116.7.1=atcnsifw.lmsal.com
198.116.7.5=cat2950-b206lmsal-6a1-a.lmsal.com
198.116.7.6=mail.lmsal.com
198.116.7.7=cat4006-b252lmsal-109-a.lmsal.com
198.116.7.8=www.lmsal.com
198.116.7.10=dns.lmsal.com
198.116.7.11=sagnet.lmsal.com
198.116.7.12=sagnetp1.lmsal.com
198.116.7.13=sagnetp2.lmsal.com
198.116.7.14=sagnetp3.lmsal.com
198.116.7.15=sagnetp4.lmsal.com
198.116.7.16=sagnetp5.lmsal.com
198.116.7.17=sagnetp6.lmsal.com
198.116.7.18=sagnetp7.lmsal.com
198.116.7.19=sagnetp8.lmsal.com
198.116.7.20=dhcp20.lmsal.com

198.116.7.21=dhcp21.lmsal.com
198.116.7.22=dhcp22.lmsal.com
198.116.7.23=dhcp23.lmsal.com
198.116.7.24=dhcp24.lmsal.com
198.116.7.25=dhcp25.lmsal.com
198.116.7.26=dhcp26.lmsal.com
198.116.7.27=dhcp27.lmsal.com
198.116.7.28=dhcp28.lmsal.com
198.116.7.29=dhcp29.lmsal.com
198.116.7.30=dhcp30.lmsal.com
198.116.7.31=dhcp31.lmsal.com
198.116.7.32=dhcp32.lmsal.com
198.116.7.33=dhcp33.lmsal.com
198.116.7.34=dhcp34.lmsal.com
198.116.7.35=dhcp35.lmsal.com
198.116.7.36=dhcp36.lmsal.com
198.116.7.37=dhcp37.lmsal.com
198.116.7.38=dhcp38.lmsal.com
198.116.7.39=dhcp39.lmsal.com
198.116.7.40=dhcp40.lmsal.com
198.116.7.60=baldrick.lmsal.com
198.116.7.61=deploy2.lmsal.com
198.116.7.62=sundog.lmsal.com
198.116.7.63=star.lmsal.com
198.116.7.64=flare.lmsal.com
198.116.7.65=codonics.lmsal.com
198.116.7.66=tqcm-pc2.lmsal.com
198.116.7.67=sunlab1.lmsal.com
198.116.7.68=wap4.lmsal.com
198.116.7.69=sxifm3b.lmsal.com
198.116.7.70=sswmac.lmsal.com
198.116.7.71=whitney.lmsal.com
198.116.7.72=visitorptr.lmsal.com
198.116.7.73=helicity.lmsal.com
198.116.7.74=xuvpc1.lmsal.com
198.116.7.75=xuvpc2.lmsal.com
198.116.7.76=nlfff.lmsal.com
198.116.7.77=monsws.lmsal.com
198.116.7.78=evolve.lmsal.com
198.116.7.79=sunbeam.lmsal.com
198.116.7.80=stellar.lmsal.com
198.116.7.81=hesp.lmsal.com
198.116.7.82=zorak.lmsal.com
198.116.7.83=ln17ps.lmsal.com
198.116.7.84=atahualpa.lmsal.com
198.116.7.86=eve.lmsal.com
198.116.7.87=tracer.lmsal.com
198.116.7.88=vestige.lmsal.com
198.116.7.89=onyx.lmsal.com
198.116.7.90=gpibenet2.lmsal.com
198.116.7.91=rogerc.lmsal.com
198.116.7.92=dimming.lmsal.com
198.116.7.93=bernina.lmsal.com
198.116.7.94=sxiccd1.lmsal.com
198.116.7.95=r204hp81502.lmsal.com
198.116.7.96=talk1.lmsal.com
198.116.7.97=mrclean.lmsal.com
198.116.7.98=gpibenet1.lmsal.com
198.116.7.99=capella.lmsal.com
198.116.7.100=castor.lmsal.com
198.116.7.101=hessi.lmsal.com
198.116.7.102=brahe.lmsal.com
198.116.7.103=sgidemo.lmsal.com
198.116.7.104=r204hpcolor.lmsal.com
198.116.7.105=sgiconsole.lmsal.com
198.116.7.106=r224hp4050.lmsal.com
198.116.7.107=fpp.lmsal.com
198.116.7.108=secchi.lmsal.com
198.116.7.109=xrt.lmsal.com
198.116.7.110=dept.lmsal.com
198.116.7.111=kokuten.lmsal.com
198.116.7.112=sxipc1.lmsal.com

198.116.7.113=sundown.lmsal.com
198.116.7.114=faculae.lmsal.com
198.116.7.115=r204hp8150.lmsal.com
198.116.7.116=r244hp8150.lmsal.com
198.116.7.117=cdburn.lmsal.com
198.116.7.118=printserver.lmsal.com
198.116.7.119=corona.lmsal.com
198.116.7.120=helios.lmsal.com
198.116.7.121=bigben.lmsal.com
198.116.7.122=grid1.lmsal.com
198.116.7.124=nutflush.lmsal.com
198.116.7.125=remnant.lmsal.com
198.116.7.126=r204hp5500.lmsal.com
198.116.7.127=nova.lmsal.com
198.116.7.128=hmi.lmsal.com
198.116.7.129=sis.lmsal.com
198.116.7.130=fppccd1.lmsal.com
198.116.7.131=fppccd2.lmsal.com
198.116.7.132=mini.lmsal.com
198.116.7.133=mhd.lmsal.com
198.116.7.134=sag.lmsal.com
198.116.7.135=nomad.lmsal.com
198.116.7.136=aia.lmsal.com
198.116.7.137=stereo.lmsal.com
198.116.7.138=jpmac.lmsal.com
198.116.7.139=diapason.lmsal.com
198.116.7.140=alanmac.lmsal.com
198.116.7.141=gpibws.lmsal.com
198.116.7.142=r211hp4000.lmsal.com
198.116.7.143=r211hp4500.lmsal.com
198.116.7.144=gpibenet6.lmsal.com
198.116.7.145=gpibenet7.lmsal.com
198.116.7.146=r110hp4050.lmsal.com
198.116.7.147=r1n11hp5.lmsal.com
198.116.7.148=wap1.lmsal.com
198.116.7.149=sunshine.lmsal.com
198.116.7.150=kuna.lmsal.com
198.116.7.160=sxifm4a.lmsal.com
198.116.7.161=xuvpc.lmsal.com
198.116.7.162=web1.lmsal.com
198.116.7.163=r142hp5500.lmsal.com
198.116.7.164=imai.lmsal.com
198.116.7.165=artemis.lmsal.com
198.116.7.167=sunspot.lmsal.com
198.116.7.168=ccdlab1.lmsal.com
198.116.7.169=ccdlab2.lmsal.com
198.116.7.170=monspc.lmsal.com
198.116.7.171=sxipc4.lmsal.com
198.116.7.173=schiff.lmsal.com
198.116.7.174=socrates.lmsal.com
198.116.7.175=caravalho.lmsal.com
198.116.7.176=ssw.lmsal.com
198.116.7.177=visitor.lmsal.com
198.116.7.178=sxi.lmsal.com
198.116.7.179=towplane.lmsal.com
198.116.7.180=secchienet.lmsal.com
198.116.7.181=alang5.lmsal.com
198.116.7.182=cosec.lmsal.com
198.116.7.183=ismgw.lmsal.com
198.116.7.184=adrpc.lmsal.com
198.116.7.185=ccdcam.lmsal.com
198.116.7.186=hmiccd1.lmsal.com
198.116.7.187=hmiccd2.lmsal.com
198.116.7.188=mfbd.lmsal.com
198.116.7.189=beast.lmsal.com
198.116.7.200=backupa1.lmsal.com
198.116.7.201=gpibenet5.lmsal.com
198.116.7.202=pore.lmsal.com
198.116.7.203=cpa.lmsal.com
198.116.7.204=michelson.lmsal.com
198.116.7.205=canopy.lmsal.com
198.116.7.206=nice.lmsal.com

198.116.7.207=deploy1.lmsal.com
198.116.7.208=mdisim.lmsal.com
198.116.7.209=java1.lmsal.com
198.116.7.210=rose.lmsal.com
198.116.7.211=darrel.lmsal.com
198.116.7.212=roadie.lmsal.com
198.116.7.213=dakota.lmsal.com
198.116.7.214=theory.lmsal.com
198.116.7.215=sxiem1.lmsal.com
198.116.7.216=web1a.lmsal.com
198.116.7.217=soaring.lmsal.com
198.116.7.218=shimmer.lmsal.com
198.116.7.219=tarbell.lmsal.com
198.116.7.220=shing.lmsal.com
198.116.7.221=plage.lmsal.com
198.116.7.222=horus.lmsal.com
198.116.7.223=sxiem2.lmsal.com
198.116.7.224=hyades.lmsal.com
198.116.7.225=faze.lmsal.com
198.116.7.226=jiba.lmsal.com
198.116.7.227=cree.lmsal.com
198.116.7.228=composer.lmsal.com
198.116.7.229=wolfson.lmsal.com
198.116.7.230=hmigse1.lmsal.com
198.116.7.231=shadow.lmsal.com
198.116.7.232=wap3.lmsal.com
198.116.7.233=wap2.lmsal.com
198.116.7.234=solserv.lmsal.com
198.116.7.236=talofa.lmsal.com
198.116.7.237=tqcm-pc1.lmsal.com
198.116.7.238=redsox.lmsal.com
198.116.7.239=gpibenet3.lmsal.com
198.116.7.240=moonrise.lmsal.com
198.116.7.241=shawnee.lmsal.com
198.116.7.243=GoBlue.lmsal.com
198.116.7.244=heman.lmsal.com
198.116.7.245=xuv.lmsal.com
198.116.7.246=sxt1.lmsal.com
198.116.7.247=lotus.lmsal.com
198.116.7.248=shine.lmsal.com
198.116.7.249=notung.lmsal.com
198.116.7.250=gpibenet4.lmsal.com
198.116.7.251=gpibenet_em2.lmsal.com
198.116.7.252=hikari.lmsal.com
198.116.7.253=ccdlab.lmsal.com
198.120.8.1=ellis-2nd-fl.arc.nasa.gov
198.120.8.30=bdavidson.arc.nasa.gov
198.120.8.32=jmorrell.arc.nasa.gov
198.120.8.33=mtucker.arc.nasa.gov
198.120.8.34=dvaldez.arc.nasa.gov
198.120.8.35=rchin.arc.nasa.gov
198.120.8.37=ommbray.arc.nasa.gov
198.120.8.38=kryton.arc.nasa.gov
198.120.8.39=thumper.arc.nasa.gov
198.120.8.40=skutter.arc.nasa.gov

Pretty neat eh? Choosing the first target from such a big list is not easy. But the advantage is that there are so many possible targets, that a few of them will be vulnerable for sure. We'll consider all these IPs as possible targets since ARC wouldn't have assigned a domain to an IP unless it was something there… What could that "something" be? Here's an incomplete list:

- UNIX-based Server
- NT Server
- Netware Server
- Mac Server
- WAP (Wireless Access Point)

- Firewall
- Router (forwards packets between networks)
- PBX (Private Branch Exchange – this is a telephone switching system)
- Switch
- Hub
- Terminal Server (device that allows dial-up connections to enter a network)
- Bridge (device connecting 2 networks)
- Videoconferencing Server / Webcam
- VoIP Phone / VoIP Gateway / POTS<->VoIP Adapter
- Load Balancer (distributes traffic in an optimal manner)
- Power Controller (provides reset capability to unattended equipment)
- UPS Network Management Card (manages the UPSs protecting servers and network equipment)
- Printer / Print Server
- PDA (Personal Digital Assistant – strangely you can find shit like that connected to the internet)
- Game Console
- Fileserver (a set of hard drives that make up the majority of disk space available in a network)
- Web Proxy
- VPN device (Virtual Private Network – offers a secure way to connect to a remote network as if you where actually in that network)
- Encryption Accelerator (encrypts IP packets so you don't have to encrypt them yourself. This way your processor does less thinking.)

Now let's find host information for each IP from the reverse_ARC list. The list must be in this form for our script to work:

ip1=domain1
ip2=domain2

Here's the script that will do the hinfo querying for us. We'll name it hinfo.pl:

```
#!/usr/bin/perl
$file=@ARGV[0];
open(DAT, $file) || die("Could not open file!");
@content=<DAT>;
close(DAT);
foreach $both (@content)
{
@tag=split(/=/,$both);
$ip=@tag[0];
$domain=@tag[1];
chop($domain);
$hinfo = `host –t hinfo $domain`;
chop($hinfo);
if (($hinfo eq "") || (substr($hinfo, 0, 4) eq "Host")) {}
else
{
print "$ip=$domain";
$hinfo =~ s/.*host information //g;
print " ($hinfo)\n";
}}
```

Let's run it:

$ perl hinfo.pl reverse_ARC > reverse_ARC_with_HINFO

The resulting file should look like this:

```
nabu@ubuntu:~/mapping$ ./hinfo.pl reverse_ARC &> reverse_ARC_with_HINFO
nabu@ubuntu:~/mapping$ cat reverse_ARC_with_HINFO
198.116.2.4=sisyphus.nsi.nasa.gov ("PC386" "Linux")
198.116.2.5=nsiss2.nsi.nasa.gov ("Sun Sparc 2" "N/A")
198.116.2.80=ws0.nsi.nasa.gov ("Temporary" "N/A")
198.116.2.81=ws1.nsi.nasa.gov ("Temporary" "N/A")
198.116.2.82=ws2.nsi.nasa.gov ("Temporary" "N/A")
198.116.2.83=ws3.nsi.nasa.gov ("Temporary" "N/A")
198.116.2.90=mcfpv.nsi.nasa.gov ("Shiva Fastpath V" "N/A")
198.116.2.91=mchplj4.nsi.nasa.gov ("HP Laserjet4si" "N/A")
198.116.2.100=lupine.nsi.nasa.gov ("Sun-4/60" "SunOS")
198.116.2.101=dcqmgate.nsi.nasa.gov ("Mac" "MACOS")
198.116.2.105=camelot.nsi.nasa.gov ("Temporary" "N/A")
198.116.2.110=phoenix.nsi.nasa.gov ("Quadra 700" "MacOS")
nabu@ubuntu:~/mapping$
```

Yap it's sad… they only gave us informations about a few of their hosts… and probably inaccurate informations. We'll thank them for what we have and we'll go further. Let's find the location of each IP from the reverse_ARC list. If we wanted to do it manually we had lots of methods to choose from. I've already shown you one a few pages before:

1. www.ip2location.com/free.asp

This uses the IP2Location software which is very expensive. The disadvantage when using their geolocation free service is that you can only use it 20 times per day. If you change proxy from time to time you'll be able to use it 2000 times per day, but that's illegal (who cares).

2. The most accurate method but the one that most of the time won't work is host –t loc. Try it in your console:

$ host –t loc yahoo.com

This works.

```
nabu@altos:~$ host -t loc yahoo.com
yahoo.com location 37 23 30.900 N 121 59 19.000 W 7.00m 100m 100m 2m
```

Now try:

$ host –t loc sisyphus.arc.nasa.gov

No response.

3. Using the GeoSelect software is also pretty accurate. Go to:
http://www.geobytes.com/IpLocator.htm
and input the IP address for sisyphus.arc.nasa.gov. It will give you a very detailed response, and it will also pinpoint the location on a map.

You can do the same by imputing the IP address at the end of this URL:
http://www.geobytes.com/IpLocator.htm?GetLocation&&ipaddress=

The same proxy thing can be done here if they block your access.

4. The NetGeo Database. Old. Inaccurate. You don't wanna use it. They should win the contest "Most Stupid Database". I only told you about NetGeo because I want you to make fun of them.
http://netgeo.caida.org/perl/netgeo.cgi?target=198.116.2.4

Let's make a script that will be given an IP range and will create files with the IPs in that range, followed by the location. For our script to work you'll have to install the MaxMind GeoLite City database and the perl API for it. You can find a guide that will help you to do it on the MaxMind website (www.maxmind.com). If I remember correctly all the steps, that's how I did it on my Ubuntu Linux:

$ mkdir other
$ cd other
$ wget http://www.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
$ gunzip GeoLiteCity.dat.gz
$ sudo mkdir /usr/local/share/GeoIP
$ sudo mv GeoLiteCity.dat /usr/local/share/GeoIP/
$ wget http://www.maxmind.com/download/geoip/api/c/GeoIP-1.3.14.tar.gz
$ tar xvfz GeoIP-1.3.14.tar.gz
$ cd GeoIP-1.3.14
$ ./configure (gave an error so I had to…)
$ sudo apt-get install zlib1g-dev
$ ./configure (now it worked)
$ make
$ sudo make install
$ cd ..
$ wget http://www.maxmind.com/download/geoip/api/perl/Geo-IP-1.27.tar.gz
$ tar xvfz Geo-IP-1.27.tar.gz
$ cd Geo-IP-1.27
$ perl Makefile.PL
$ make
$ sudo make install
$ cd ../..
$ vi geolocate.pl

And this is geolocate.pl:

```
#!/usr/bin/perl
use Geo::IP;
@tag=split(/-/,@ARGV[0]);
@ip1=split(/\./,@tag[0]);
@ip2=split(/\./,@tag[1]);
for ($a=@ip1[0]; $a<1+@ip2[0]; $a++) {
for ($b=@ip1[1]; $b<1+@ip2[1]; $b++) {
for ($c=@ip1[2]; $c<1+@ip2[2]; $c++) {
for ($d=@ip1[3]; $d<1+@ip2[3]; $d++) {
my $gi = Geo:IP->open("/usr/local/share/GeoIP/GeoIPCity.dat", GEOIP_STANDARD);
my $r = $gi->record_by_name("$a.$b.$c.$d");
$file=$r->city;
open(DAT,">>$file") || die("Cannot open file");
print DAT "$a.$b.$c.$d=".$r->country_name.", ".$r->city."\n";
close(DAT);
}}}}
```

While this script will give fast and accurate results for most of the IP ranges, it won't work that well for ARC. So we won't use it. Instead we'll use methods 1 and 3. We'll discover that the first block (198.116.2.*) is located in Houston, Texas, the second (198.116.3.*) in Cupertino, California, the third (198.116.7.*) in Palo Alto, California, which is really close with Cupertino and the last one (198.120.8.*) in Arab, Alabama.

Next we'll make a script that will try do detect if our target domain has any other subdomains, maybe located in another internet block. How will it work? Well… the concept is quite simple actually. First it will ask Google with a query like this "site:target.com –site:www.target.com" (check Appendix B for Google Advanced Operators), and then will bruteforce our main DNS server using a file called subdomains.txt. You will need a Google key for it to work. Get one from here:
http://www.google.com/apis/

$ vi subdomains.pl

```
#!/usr/bin/perl
use SOAP::Lite;
if ($#ARGV<0){die "perl subdomains.pl domainname\ne.g. perl subdomains.pl lmsal.com\n";}
my $company = $ARGV[0];

####### You may want to edit these lines: ################
$key   = "qhHRGPdQFHIf/xiifDWsnhxshXmdHtLV";
my $service = SOAP::Lite->service("http://diva.nhki.net/blog/lib/MT/GoogleSearch.wsdl");
my $numloops=40;          #number of pages - max 100
$already_known_subdomain = "www";
$timeout = 5;
#########################################################################


my $query = "site:$company -site:$already_known_subdomain.$company";
push @allsites,DoGoogle($key,$query,$company);

## Remove duplicates
@allsites=dedupe(@allsites);
print STDOUT "\n---------------\nSubdomains:\n---------------\n";
foreach $site (@allsites){
      print STDOUT "$site\n";
      ## Uncomment this if… but you know why.
       ## system "host -W $timeout $site";
      $subzero = "subs_$company";
      open(DAT,">>$subzero") || die("Cannot open file.");
      print DAT "$site\n";
      close(DAT);

}

print "\n---------------\nBruteforcing DNS:\n---------------\n";
foreach $subdomain (`cat subdomains.txt`)
            {
            chop($subdomain);
            $subdomain = $subdomain.".".$company;
            $matrix = `host -W $timeout $subdomain`;
            if (substr($matrix, 0, 4) ne 'Host')
            {
                  print "$matrix\n";
                  $dns_bruteforce_results = "dnsbf_$company";
                  open(DAT,">>$dns_bruteforce_results") || die("Cannot open file.");
                  print DAT "$matrix\n";
                  close(DAT);
            }
            }
```

```
###############------subs-------############
sub dedupe{
	my (@keywords) = @_;
	my %hash = ();
	foreach (@keywords) {
		$_ =~ tr/[A-Z]/[a-z]/;
		chomp;
		if (length($_)>1){$hash{$_} = $_;}
	}
	return keys %hash;
}

sub parseURL{
		my ($site,$company)=@_;
		if (length($site)>0){
			if ($site =~ /:\/\/([\.\w]+)[\:\/]/){
				my $mined=$1;
				if ($mined =~/$company/){
					return $mined;
				}
			}
		}
		return "";
}

sub DoGoogle{
		my ($GoogleKey,$GoogleQuery,$company)=@_;
		my @GoogleDomains="";
	for ($j=0; $j<$numloops; $j++){
		print STDERR "$j ";
		my $results = $service
		    -> doGoogleSearch($GoogleKey,$GoogleQuery,(10*$j),10,"true","","true","","latin1","latin1");

		my $re=(@{$results->{resultElements}});
		foreach my $results(@{$results->{resultElements}}){
			my $site=$results->{URL};
				my $dnsname=parseURL($site,$company);
				if (length($dnsname)>0){
			push @GoogleDomains,$dnsname;
				}
		}
		if ($re !=10){last;}
	}
		return @GoogleDomains;
}
```

For our script to work you'll have to install some perl libraries: LWP (http://www.linpro.no/lwp/), SOAP-Lite (http://soaplite.com/download.html) and URI (http://www.cpan.org/modules/by-module/URI/URI-1.35.tar.gz). Also modify the $key variable so it would use your own Google key. Otherwise it might not work. The script uses subdomains.txt as the source for the subdomains that will try to detect. Here's the subdomains.txt file that I used, I've pasted it here on 5 columns so it would not occupy that much space. You can download it from (www.absolom.ro/downloads/tutorials/subdomains.txt).

3com	aleader	antemium	arudius	augustux
4all	alixe	antomic	asian	aurora
64studio	alt	apache	asianlinux	aurox
64_studio	altos	apodio	asianux	austrumi
a	amarok	arabbix	aslinux	auth
abuledu	amber	arabian	asp	b
accelerator	ankur	arch	asplinux	b2d
access	ankurbangla	arche	astaro	back
adios	annvix	archeos	asterisk	backtrack
admelix	annyung	archie	athene	backup
agnula	anonym	ares	atmission	base
aix	anonymos	ark	atomix	bastion

bayanihan
bck
bearops
beatrix
beehive
beernix
belenix
berry
biadix
big
biglinux
bind
biobrew
bioknoppix
blackpanther
blackrhino
blag
blin
blue
bluepoint
bluewall
bonzai
border
boten
box
breezy
bridge
brlspeak
bsd
buffalo
burapha
business
byo
byzantine
byzantineos
c
caixa
caixamagica
caos
catix
ccux
cd
cdlinux
censornet
cent
centos
chains
chinese2000
cisco
clarkconnect
cle
cluster
clusterix
clusterknoppix
cobind
cocreate
college
condorux
conectiva
console
content
cool
core
corel
corporate
cosix
coyote
cpubuilders
crux
cs
cvp
d
damnsmall
danix
dapper
darkstar
data
database
dc
dchub
deadcd
debian
debxpde
deepwater
defender
definity
deli
demolinux
demudi
desk
desktop
desktopbsd
dev
devil
dizinha
dnalinux
dns
domino
dominoserver
download
dragonfly
dragonflybsd
dream
dreamlinux
dynasoft
dyne
dynebolic
e
eadem
eagle
e-bus
e-business
edu
edubuntu
educd
eduknoppix
edulinux
ehad
elearnix
elive
elx
email
e-mail
engarde
eridani
erposs
esafe
e-safe
esware
euronode
evilentity
evinux
external
extranet
ezplanet
f
famelix
feather
featherweight
fedora
fermi
fileserver
filter
finger
finnix
fire
firebox
firewall
flash
flonix
foresight
fork
fox
foxdesktop
freebsd
freedows
freeduc
freeducsup
freenas
freepia
freesbie
frenzy
front
frugalware
ftosx
ftp
fw
fw-1
fwall
fwe
fwi
g
game
gate
gatekeeper
gateway
gauntlet
geexbox
gelecek
genieos
gentoo
gentooth
gentoox
geo
geolivre
gibraltar
ging
gnix
gnoppix
gnox
gnu
gnulinex
gnustep
goblinx
gobolinux
grafpup
grml
group
gtw
guadalinex
gulicbsd
gulic-bsd
gw
h
h3knix
haansoft
hakin9
hancom
happy
happymac
hard
haydar
hedinux
helix
help
heretix
hikarunix
hispafuentes
hiweed
hklpg
hoary
holon
honeywall
hop
how-tux
hp
hpjet
hpsecure
hpux
hp-ux
http
https
hub
hubworx
i
ibm
ibox
icepack
idms
ids
ignalum
immunix
impi
in
indlinux
info
insert
inside
install
internal
internet
intranet
ipchains
ipcop
ipfw
irix
islack
j
jamd
jblinux
jet
jolinux
jollix
julex
jusix
k
k12ltsp
kaella
kalango
kanotix
karamad
kate
kateos
kdemar
kernel
kinneret
klax
klustrix
km
kmlinux
knopils
knoppel
knopperdisk
knoppix
knoppix64
knoppixmame
knoppixstd
knoppmyth
knosciences
komodo
kondara
kore
kororaa
krud
kubuntu
kurumin
l
lab
lamppix
lan
las
laser5
lba
lbalinux
lfs
lg3d
lgis
libranet
liis
linare
lineox
linespa
linnex
linnexos
linpus
linspire
linux
linux4all
linuxconsole
linuxeducd
linuxin
linuxinstall
linuxo
linuxplus
linuxppc
linuxxp
list
litrix
livux
llgp
lliurex
lnxbbc
local
loco
lonix
lorma
lotus
lotusdomino
lotusnotes
lotusserver
lrs
luinux
luit
luminux
lunar
lycoris
m
m0n0wall
mac
madeinlinux
magic
mail
mailfeed
mailgate
mailgateway
mailgroup
mailhost
maillist
mailmarshall
mailpop
mailrelay
main
management
mandows
mandrake
mandriva
max
mayix
mcnlive
media
mediainlinux
medialab
medialinux
mepis
merdeka
miko
mimesweeper
miracle
miros
mizi
mockup
molinux
momonga
monoppix
monowall
morphix
movix
ms
msc
msclinux
msproxy
mssql
mumi
munjoy
muriqui
murix
musix
mutagenix
mx
my
myah
myahos
mylinux
mysql
n
nameserver
nas
nasgaia
nature
natures

navyn
navynos
neat
nepalinux
net
netbsd
netware
network
netwosix
news
newsdesk
newsfeed
newsgroup
newsroom
newsserver
nexenta
niigata
nitix
nix
n-ix
nntp
no
node
nonux
nordisknoppix
notes
noteserver
notesserver
novell
ns
ns1
ns2
ns3
nst
nt
ntalk
nubuntu
nuxone
o
odc
oeone
office
ogoknoppix
olivebsd
omoikane
onebase
onet
open
openbsd
opendesktop
openlab
openlx
openna
openwall
oralux
os
out
outside
overclockix
p
paipix
panther
parallelknoppix
pardus
parsix
path
pbx
pcbsd
pc-bsd
pclinuxos
pda
peachtree
penguinsleuth
pentoo
pequelin
pfsense
phaeronix
phayoune
phlak
phpsol
piebox
pilot
pingo
pingwinek
pix
plamo
planb
plan-b
pld
plus
pocketlinux
polarbear
pop
pop3
pophost
popmail
popserver
poseidon
power
ppc
pqui
print
printer
printspool
private
progeny
project
projectdev
protector
proxy
proxyserver
ptux
public
pud
puppy
q
qilinux
qpop
quantian
r
radio
raptor
rays
read
redcreek
redflag
redhat
redoffice
redwall
remote
resala
rescue
rhino
rip
rock
rockscluster
root
roslims
route
router
rpath
rpmlive
rr4
rr64
rsh
rsync
rubix
runt
s
sale
salvare
sam
samba
santafe
scanner
schillix
sci
scientific
scilinux
sco
screen
screening
secret
secure
securepoint
seek
sense
sentinel
sentinix
sentry
sentryfirewall
server
sftp
shabdix
sharing
shark
shell
site
skole
skolelinux
slack
slackintosh
slackware
slamd64
slampp
slavix
slax
slix
slotech
slo-tech
slynux
smail
smap
smartpeer
sme
smeserver
smoothwall
smtp
smtpgateway
smtpgw
snappix
sniffer
snofrix
snort
soft
sol
solaris
sonic
sorcerer
sot
source
sourcemage
soyombo
speak
spectra
sphinx
sphinxos
spool
sql
squid
squiggle
squiggleos
ssh
stampede
startcom
station
storm
stream
stress
stresslinux
stux
stx
sulix
sun
sunjds
sunos
suse
switch
symphony
t
t2
tablix
talinux
talk
tao
taprobane
tech
telnet
temp
terminal
tfm
thepacketmaster
thinstation
thiz
tilix
tiny
tinysofa
tle
tmp
topologi
topologilinux
tpm
transfer
trend
trendmicro
triance
trianceos
trinity
troppix
trustix
truva
trx
tugux
tumix
tupiserver
tuquito
turbo
turbolinux
turkix
tux
tv
u
ubuntu
ufficiozero
uhu
uhu-linux
ultima
underground
united
unix
unseen
ups
user
userlinux
uso
ututo
v
vector
video
videolinux
vine
virtual
virux
vista
vlan
vlos
vm
vmware
vnc
vnlinux
voip
voltalinux
voodoo
vpn
w
wall
wan
wap
ware
warty
wazobia
web
webmail
webproxy
webserver
webswitch
whax
whitebox
who
whoppix
wienux
win
win2000
win2003
win2k
win31
win95
win98
win9x
winbi
windows
winme
winnt
winserver
winxp
wolvix
womp
work
workstation
wow
write
ww
www
x
xandros
xarnoppix
xenoppix
xevian
xfer
xfld
xos
xp
xteam
y
yellow
yellowdog
yes
yoper
z
zen
zenix
zenwalk
zerahstar
zeus
zonecd
zopix

I'll test the script with arc.nasa.gov and lmsal.com. It will print on the screen its discoveries, and will also create these files: subs_arc.nasa.gov and dnsbf_arc.nasa.gov, respectively subs_lmsal.com and dnsbf_lmsal.com.
Let's see the output for arc.nasa.gov:

```
nabu@altos:~/mapping$ ./subdomains.pl arc.nasa.gov
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
---------------
Subdomains:
---------------
issworkshop.arc.nasa.gov
phenomorph.arc.nasa.gov
aefs.arc.nasa.gov
lunar.arc.nasa.gov
grants.arc.nasa.gov
naccenter.arc.nasa.gov
risk.arc.nasa.gov
vams.arc.nasa.gov
ssrl.arc.nasa.gov
nai.arc.nasa.gov
aeronautics.arc.nasa.gov
robotics.arc.nasa.gov
bocachica.arc.nasa.gov
casc.arc.nasa.gov
virtualskies.arc.nasa.gov
uarc.arc.nasa.gov
ecocast.arc.nasa.gov
technology.arc.nasa.gov
clickworkers.arc.nasa.gov
researchpark.arc.nasa.gov
lstworkshop.arc.nasa.gov
externalrelations.arc.nasa.gov
biovis.arc.nasa.gov
safecopter.arc.nasa.gov
pso.arc.nasa.gov
postdoc.arc.nasa.gov
opensource.arc.nasa.gov
colorusage.arc.nasa.gov
gec2003.arc.nasa.gov
impact.arc.nasa.gov
sail.arc.nasa.gov
infotech.arc.nasa.gov
hdcp.arc.nasa.gov
cfo.arc.nasa.gov
surf.arc.nasa.gov
exp.arc.nasa.gov
x500.arc.nasa.gov
astroventure.arc.nasa.gov
appliedit.arc.nasa.gov
wingsovermars.arc.nasa.gov
ficworkproducts.arc.nasa.gov
apms.arc.nasa.gov
worldwind.arc.nasa.gov
moffetthistoric.arc.nasa.gov
topweb.arc.nasa.gov
academy.arc.nasa.gov
rotored.arc.nasa.gov
vision.arc.nasa.gov
space.arc.nasa.gov
cgbr.arc.nasa.gov
hci.arc.nasa.gov
ffc.arc.nasa.gov
quest.arc.nasa.gov
nstars.arc.nasa.gov
pbagroup.arc.nasa.gov
hamradio.arc.nasa.gov
ebpi.arc.nasa.gov
isse.arc.nasa.gov
is.arc.nasa.gov
```

mail.arc.nasa.gov
pdars.arc.nasa.gov
nai1.arc.nasa.gov
marte.arc.nasa.gov
astrobionics.arc.nasa.gov
ieeenano2003.arc.nasa.gov
www.simlabs.arc.nasa.gov
environment.arc.nasa.gov
sky.arc.nasa.gov
ails.arc.nasa.gov
erc.arc.nasa.gov
geo.arc.nasa.gov
reentry.arc.nasa.gov
psa.arc.nasa.gov
halfdome.arc.nasa.gov
asm.arc.nasa.gov
robosphere.arc.nasa.gov
web99.arc.nasa.gov
ecs.arc.nasa.gov
asrs.arc.nasa.gov
mas.arc.nasa.gov
generations.arc.nasa.gov
isis.arc.nasa.gov
labcam.arc.nasa.gov
asapdata.arc.nasa.gov
webtads.arc.nasa.gov
history.arc.nasa.gov
astrobiology.arc.nasa.gov
questdb.arc.nasa.gov
ic.arc.nasa.gov
astrobiotech.arc.nasa.gov
abscicon2004.arc.nasa.gov
learn.arc.nasa.gov
procure.arc.nasa.gov
dart.arc.nasa.gov
aerospace.arc.nasa.gov
probews2.arc.nasa.gov
lis.arc.nasa.gov
leonid.arc.nasa.gov
jetstream.arc.nasa.gov
cca.arc.nasa.gov
vathena.arc.nasa.gov
encounter.arc.nasa.gov
abscicon.arc.nasa.gov
nx.arc.nasa.gov
eo.arc.nasa.gov
marsoweb.arc.nasa.gov

Bruteforcing DNS:
---
aix.arc.nasa.gov has address 128.102.37.17
aix.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

amber.arc.nasa.gov has address 143.232.122.83
amber.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

apache.arc.nasa.gov has address 143.232.73.90
apache.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

asp.arc.nasa.gov has address 143.232.68.239
asp.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

aurora.arc.nasa.gov is an alias for eos.arc.nasa.gov.
eos.arc.nasa.gov has address 128.102.119.101
aurora.arc.nasa.gov is an alias for eos.arc.nasa.gov.
aurora.arc.nasa.gov is an alias for eos.arc.nasa.gov.
eos.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

b.arc.nasa.gov has address 143.232.64.83
b.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

backup.arc.nasa.gov has address 128.102.152.48
backup.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

box.arc.nasa.gov has address 143.232.68.73
box.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

coyote.arc.nasa.gov has address 128.102.173.38
coyote.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

darkstar.arc.nasa.gov has address 143.232.124.73
darkstar.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

domino.arc.nasa.gov has address 128.102.119.176
domino.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

dragonfly.arc.nasa.gov has address 143.232.165.52
dragonfly.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

eagle.arc.nasa.gov has address 143.232.207.76
eagle.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

email.arc.nasa.gov has address 143.232.64.51
email.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

feather.arc.nasa.gov has address 128.102.147.31
feather.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

fedora.arc.nasa.gov has address 128.102.148.44
fedora.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

finger.arc.nasa.gov is an alias for mailstar1.arc.nasa.gov.
mailstar1.arc.nasa.gov has address 128.102.4.37
finger.arc.nasa.gov is an alias for mailstar1.arc.nasa.gov.
finger.arc.nasa.gov is an alias for mailstar1.arc.nasa.gov.
mailstar1.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

fire.arc.nasa.gov has address 143.232.73.113
fire.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

frenzy.arc.nasa.gov has address 143.232.74.245
frenzy.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

gate.arc.nasa.gov has address 128.102.102.51
gate.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

geo.arc.nasa.gov has address 128.102.142.10
geo.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

helix.arc.nasa.gov has address 143.232.117.32
helix.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

help.arc.nasa.gov has address 128.102.105.43
help.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

ids.arc.nasa.gov has address 143.232.64.133
ids.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

in.arc.nasa.gov has address 192.203.230.200
in.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

kate.arc.nasa.gov has address 143.232.74.92
kate.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

list.arc.nasa.gov is an alias for lists.arc.nasa.gov.
lists.arc.nasa.gov has address 128.102.2.23
list.arc.nasa.gov is an alias for lists.arc.nasa.gov.
list.arc.nasa.gov is an alias for lists.arc.nasa.gov.
lists.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

lunar.arc.nasa.gov has address 128.102.2.141
lunar.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

mail.arc.nasa.gov has address 128.102.31.140
mail.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

max.arc.nasa.gov has address 143.232.119.151
max.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

news.arc.nasa.gov has address 129.99.34.69
news.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

ns.arc.nasa.gov has address 128.102.16.2
ns.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

ns1.arc.nasa.gov has address 128.102.201.2
ns1.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

ns2.arc.nasa.gov has address 143.232.252.34
ns2.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

ns3.arc.nasa.gov has address 128.102.0.34
ns3.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

open.arc.nasa.gov is an alias for q12.arc.nasa.gov.
q12.arc.nasa.gov has address 128.102.205.80
open.arc.nasa.gov is an alias for q12.arc.nasa.gov.
open.arc.nasa.gov is an alias for q12.arc.nasa.gov.
q12.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

out.arc.nasa.gov has address 198.32.136.5
out.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

panther.arc.nasa.gov has address 143.232.67.62
panther.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

pilot.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

pop3.arc.nasa.gov is an alias for securemail.arc.nasa.gov.
securemail.arc.nasa.gov has address 128.102.31.141
pop3.arc.nasa.gov is an alias for securemail.arc.nasa.gov.
pop3.arc.nasa.gov is an alias for securemail.arc.nasa.gov.
securemail.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

q.arc.nasa.gov has address 128.102.205.74
q.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

raptor.arc.nasa.gov has address 128.102.174.241
raptor.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

rock.arc.nasa.gov has address 143.232.155.105
rock.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

sam.arc.nasa.gov has address 143.232.165.66
sam.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

scanner.arc.nasa.gov has address 128.102.122.114
scanner.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

shark.arc.nasa.gov has address 143.232.74.61
shark.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

smtp.arc.nasa.gov has address 128.102.31.150
smtp.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

snort.arc.nasa.gov has address 143.232.190.38
snort.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

sonic.arc.nasa.gov has address 143.232.67.209
sonic.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

spectra.arc.nasa.gov has address 143.232.121.94
spectra.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

squid.arc.nasa.gov has address 128.102.119.16

squid.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

storm.arc.nasa.gov has address 143.232.124.65
storm.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

sun.arc.nasa.gov has address 143.232.64.39
sun.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

t2.arc.nasa.gov has address 143.232.136.80
t2.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

tao.arc.nasa.gov has address 143.232.68.74
tao.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

trinity.arc.nasa.gov has address 128.102.173.20
trinity.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

united.arc.nasa.gov has address 143.232.128.249
united.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

unix.arc.nasa.gov has address 143.232.156.36
unix.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

virtual.arc.nasa.gov is an alias for biovis.arc.nasa.gov.
biovis.arc.nasa.gov has address 128.102.37.205
virtual.arc.nasa.gov is an alias for biovis.arc.nasa.gov.
virtual.arc.nasa.gov is an alias for biovis.arc.nasa.gov.
biovis.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

voodoo.arc.nasa.gov has address 128.102.119.197
voodoo.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

vpn.arc.nasa.gov is an alias for sounion.arc.nasa.gov.
sounion.arc.nasa.gov has address 128.102.10.115
vpn.arc.nasa.gov is an alias for sounion.arc.nasa.gov.
vpn.arc.nasa.gov is an alias for sounion.arc.nasa.gov.
sounion.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

wap.arc.nasa.gov is an alias for server-mpo.arc.nasa.gov.
server-mpo.arc.nasa.gov has address 128.102.195.63
wap.arc.nasa.gov is an alias for server-mpo.arc.nasa.gov.
wap.arc.nasa.gov is an alias for server-mpo.arc.nasa.gov.
server-mpo.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

web.arc.nasa.gov has address 128.102.2.147
web.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

webserver.arc.nasa.gov has address 143.232.110.23
webserver.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

win.arc.nasa.gov has address 143.232.66.88
win.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

windows.arc.nasa.gov has address 128.102.2.59
windows.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

work.arc.nasa.gov has address 128.102.147.61
work.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

wow.arc.nasa.gov has address 143.232.64.129
wow.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

www.arc.nasa.gov is an alias for arcnasa.r3h.net.
arcnasa.r3h.net has address 212.162.1.195
arcnasa.r3h.net has address 212.162.1.196
www.arc.nasa.gov is an alias for arcnasa.r3h.net.
www.arc.nasa.gov is an alias for arcnasa.r3h.net.

yellow.arc.nasa.gov has address 128.102.192.161
yellow.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.

zen.arc.nasa.gov is an alias for cca.arc.nasa.gov.

```
cca.arc.nasa.gov has address 128.102.37.209
zen.arc.nasa.gov is an alias for cca.arc.nasa.gov.
zen.arc.nasa.gov is an alias for cca.arc.nasa.gov.
cca.arc.nasa.gov mail is handled by 100 smtp.arc.nasa.gov.
```

Jesus Christ… they are giving us so much informations that I'm beginning to think they want to be hacked. We have hosts like snort.arc.nasa.gov, squid.arc.nasa.gov, aix.arc.nasa.gov, sun.arc.nasa.gov etc. etc. etc. lots of them give us too much info about the host itself. If for example we had a 0day exploit for squid (that's an open source proxy daemon) we'd surely know which host is vulnerable. Let's check our script for lmsal.com and then this chapter is over.

```
nabu@altos:~/mapping$ ./subdomains.pl lmsal.com
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
--------------
Subdomains:
--------------
sdo.lmsal.com
blackadder.lmsal.com
ana.lmsal.com
java1.lmsal.com
zorak.lmsal.com
canopy.lmsal.com
shadow.lmsal.com

--------------
Bruteforcing DNS:
--------------
dns.lmsal.com has address 198.116.7.10

ftp.lmsal.com is an alias for sunbeam.lmsal.com.
sunbeam.lmsal.com has address 198.116.7.79
ftp.lmsal.com is an alias for sunbeam.lmsal.com.
ftp.lmsal.com is an alias for sunbeam.lmsal.com.

lotus.lmsal.com has address 198.116.7.247

mail.lmsal.com has address 198.116.7.6

www.lmsal.com has address 198.116.7.8
```

== Basic Host Enumeration and (Basic) Techniques to Bypass Firewalls and Intrusion Detection Systems ==

OK. Now let's chose a more specific target from the ARC address space. Let's say LMSAL, the Lockheed Martin Solar and Astrophysics Laboratory. It has hosts all over block 198.116.7.0/24, which as we established earlier is located in Palo Alto.
http://www.lmsal.com/maps.htm

What we want to do is find out exactly what computers are powered on in this netblock, and how are they interconnected, so that we can begin hacking them. Trying to scan for open ports a computer that doesn't exist will never give results and will trigger an IDS (Intrusion Detection System) alarm because we sent to many useless packets. NASA => Ames Research Center => the Lockheed Martin Solar and Astrophysics Laboratory… we're talking about a firewalled network which should have a great protection. You don't wanna make foolish steps. We have to send as less packets as possible and make it all efficient. For this, we'll install tcptraceroute
http://michael.toren.net/code/tcptraceroute/ and hping3
http://wiki.hping.org/download

Tcptraceroute is a traceroute implementation using TCP datagrams. By sending out TCP SYN packets instead of UDP packets, it is able to bypass the most common firewall filters. If the host that we are tcptracerouting is not listening for incoming connections, it will respond with a RST packet indicating that the port is closed. Since most firewalls will drop our packet if it's not headed to an open port, we'll have to use open ports and existent hosts if we want a complete traceroute. In networks that are not firewalled you can use tcptraceroute however you like.

Hping3 is a packet generator and analyzer for the TCP/IP protocol. It also has TCL scripting capabilities but don't start learning TCL unless you have a lot spare time to waste. If you want to learn a programming language, learn something serious: C++ or if you want to be a software developer in the future try Java or better C# (C sharp)/Visual .NET.

Anyway, once we've installed these tools we'll take our list of target hosts and start discovering which is how.

198.116.7.0
198.116.7.1=atcnsifw.lmsal.com
198.116.7.2
198.116.7.3
198.116.7.4
198.116.7.5=cat2950-b206lmsal-6a1-a.lmsal.com
198.116.7.6=mail.lmsal.com
198.116.7.7=cat4006-b252lmsal-109-a.lmsal.com
198.116.7.8=www.lmsal.com
198.116.7.9
198.116.7.10=dns.lmsal.com
198.116.7.11=sagnet.lmsal.com
198.116.7.12=sagnetp1.lmsal.com
198.116.7.13=sagnetp2.lmsal.com
198.116.7.14=sagnetp3.lmsal.com
198.116.7.15=sagnetp4.lmsal.com
198.116.7.16=sagnetp5.lmsal.com
198.116.7.17=sagnetp6.lmsal.com
198.116.7.18=sagnetp7.lmsal.com
198.116.7.19=sagnetp8.lmsal.com
198.116.7.20=dhcp20.lmsal.com
198.116.7.21=dhcp21.lmsal.com
198.116.7.22=dhcp22.lmsal.com
198.116.7.23=dhcp23.lmsal.com
198.116.7.24=dhcp24.lmsal.com
198.116.7.25=dhcp25.lmsal.com
198.116.7.26=dhcp26.lmsal.com
198.116.7.27=dhcp27.lmsal.com
198.116.7.28=dhcp28.lmsal.com
198.116.7.29=dhcp29.lmsal.com
198.116.7.30=dhcp30.lmsal.com
198.116.7.31=dhcp31.lmsal.com
198.116.7.32=dhcp32.lmsal.com
198.116.7.33=dhcp33.lmsal.com
198.116.7.34=dhcp34.lmsal.com
198.116.7.35=dhcp35.lmsal.com
198.116.7.36=dhcp36.lmsal.com
198.116.7.37=dhcp37.lmsal.com

198.116.7.38=dhcp38.lmsal.com
198.116.7.39=dhcp39.lmsal.com
198.116.7.40=dhcp40.lmsal.com
198.116.7.41-198.116.7.59
198.116.7.60=baldrick.lmsal.com
198.116.7.61=deploy2.lmsal.com
198.116.7.62=sundog.lmsal.com
198.116.7.63=star.lmsal.com
198.116.7.64=flare.lmsal.com
198.116.7.65=codonics.lmsal.com
198.116.7.66=tqcm-pc2.lmsal.com
198.116.7.67=sunlab1.lmsal.com
198.116.7.68=wap4.lmsal.com
198.116.7.69=sxifm3b.lmsal.com
198.116.7.70=sswmac.lmsal.com
198.116.7.71=whitney.lmsal.com
198.116.7.72=visitorptr.lmsal.com
198.116.7.73=helicity.lmsal.com
198.116.7.74=xuvpc1.lmsal.com
198.116.7.75=xuvpc2.lmsal.com
198.116.7.76=nlfff.lmsal.com
198.116.7.77=monsws.lmsal.com
198.116.7.78=evolve.lmsal.com
198.116.7.79=sunbeam.lmsal.com
198.116.7.80=stellar.lmsal.com
198.116.7.81=hesp.lmsal.com
198.116.7.82=zorak.lmsal.com
198.116.7.83=ln17ps.lmsal.com
198.116.7.84=atahualpa.lmsal.com
198.116.7.85
198.116.7.86=eve.lmsal.com
198.116.7.87=tracer.lmsal.com
198.116.7.88=vestige.lmsal.com
198.116.7.89=onyx.lmsal.com
198.116.7.90=gpibenet2.lmsal.com
198.116.7.91=rogerc.lmsal.com
198.116.7.92=dimming.lmsal.com
198.116.7.93=bernina.lmsal.com
198.116.7.94=sxiccd1.lmsal.com
198.116.7.95=r204hp81502.lmsal.com
198.116.7.96=talk1.lmsal.com
198.116.7.97=mrclean.lmsal.com
198.116.7.98=gpibenet1.lmsal.com
198.116.7.99=capella.lmsal.com
198.116.7.100=castor.lmsal.com
198.116.7.101=hessi.lmsal.com
198.116.7.102=brahe.lmsal.com
198.116.7.103=sgidemo.lmsal.com
198.116.7.104=r204hpcolor.lmsal.com
198.116.7.105=sgiconsole.lmsal.com
198.116.7.106=r224hp4050.lmsal.com
198.116.7.107=fpp.lmsal.com
198.116.7.108=secchi.lmsal.com
198.116.7.109=xrt.lmsal.com
198.116.7.110=dept.lmsal.com
198.116.7.111=kokuten.lmsal.com
198.116.7.112=sxipc1.lmsal.com
198.116.7.113=sundown.lmsal.com
198.116.7.114=faculae.lmsal.com
198.116.7.115=r204hp8150.lmsal.com
198.116.7.116=r244hp8150.lmsal.com
198.116.7.117=cdburn.lmsal.com
198.116.7.118=printserver.lmsal.com
198.116.7.119=corona.lmsal.com
198.116.7.120=helios.lmsal.com
198.116.7.121=bigben.lmsal.com
198.116.7.122=grid1.lmsal.com
198.116.7.123
198.116.7.124=nutflush.lmsal.com
198.116.7.125=remnant.lmsal.com
198.116.7.126=r204hp5500.lmsal.com
198.116.7.127=nova.lmsal.com
198.116.7.128=hmi.lmsal.com
198.116.7.129=sis.lmsal.com
198.116.7.130=fppccd1.lmsal.com
198.116.7.131=fppccd2.lmsal.com
198.116.7.132=mini.lmsal.com
198.116.7.133=mhd.lmsal.com
198.116.7.134=sag.lmsal.com
198.116.7.135=nomad.lmsal.com
198.116.7.136=aia.lmsal.com
198.116.7.137=stereo.lmsal.com
198.116.7.138=jpmac.lmsal.com
198.116.7.139=diapason.lmsal.com
198.116.7.140=alanmac.lmsal.com
198.116.7.141=gpibws.lmsal.com
198.116.7.142=r211hp4000.lmsal.com
198.116.7.143=r211hp4500.lmsal.com
198.116.7.144=gpibenet6.lmsal.com
198.116.7.145=gpibenet7.lmsal.com
198.116.7.146=r110hp4050.lmsal.com
198.116.7.147=r1n11hp5.lmsal.com
198.116.7.148=wap1.lmsal.com
198.116.7.149=sunshine.lmsal.com
198.116.7.150=kuna.lmsal.com
198.116.7.151-198.116.7.159
198.116.7.160=sxifm4a.lmsal.com
198.116.7.161=xuvpc.lmsal.com
198.116.7.162=web1.lmsal.com
198.116.7.163=r142hp5500.lmsal.com
198.116.7.164=imai.lmsal.com
198.116.7.165=artemis.lmsal.com
198.116.7.166
198.116.7.167=sunspot.lmsal.com
198.116.7.168=ccdlab1.lmsal.com
198.116.7.169=ccdlab2.lmsal.com
198.116.7.170=monspc.lmsal.com
198.116.7.171=sxipc4.lmsal.com
198.116.7.172
198.116.7.173=schiff.lmsal.com
198.116.7.174=socrates.lmsal.com
198.116.7.175=caravalho.lmsal.com
198.116.7.176=ssw.lmsal.com
198.116.7.177=visitor.lmsal.com
198.116.7.178=sxi.lmsal.com
198.116.7.179=towplane.lmsal.com
198.116.7.180=secchienet.lmsal.com
198.116.7.181=alang5.lmsal.com
198.116.7.182=cosec.lmsal.com
198.116.7.183=ismgw.lmsal.com
198.116.7.184=adrpc.lmsal.com
198.116.7.185=ccdcam.lmsal.com
198.116.7.186=hmiccd1.lmsal.com
198.116.7.187=hmiccd2.lmsal.com
198.116.7.188=mfbd.lmsal.com
198.116.7.189=beast.lmsal.com
198.116.7.190-198.116.7.199
198.116.7.200=backupa1.lmsal.com
198.116.7.201=gpibenet5.lmsal.com
198.116.7.202=pore.lmsal.com
198.116.7.203=cpa.lmsal.com
198.116.7.204=michelson.lmsal.com
198.116.7.205=canopy.lmsal.com
198.116.7.206=nice.lmsal.com
198.116.7.207=deploy1.lmsal.com
198.116.7.208=mdisim.lmsal.com
198.116.7.209=java1.lmsal.com
198.116.7.210=rose.lmsal.com
198.116.7.211=darrel.lmsal.com
198.116.7.212=roadie.lmsal.com
198.116.7.213=dakota.lmsal.com
198.116.7.214=theory.lmsal.com
198.116.7.215=sxiem1.lmsal.com
198.116.7.216=web1a.lmsal.com

198.116.7.217=soaring.lmsal.com
198.116.7.218=shimmer.lmsal.com
198.116.7.219=tarbell.lmsal.com
198.116.7.220=shing.lmsal.com
198.116.7.221=plage.lmsal.com
198.116.7.222=horus.lmsal.com
198.116.7.223=sxiem2.lmsal.com
198.116.7.224=hyades.lmsal.com
198.116.7.225=faze.lmsal.com
198.116.7.226=jiba.lmsal.com
198.116.7.227=cree.lmsal.com
198.116.7.228=composer.lmsal.com
198.116.7.229=wolfson.lmsal.com
198.116.7.230=hmigse1.lmsal.com
198.116.7.231=shadow.lmsal.com
198.116.7.232=wap3.lmsal.com
198.116.7.233=wap2.lmsal.com
198.116.7.234=solserv.lmsal.com
198.116.7.235
198.116.7.236=talofa.lmsal.com
198.116.7.237=tqcm-pc1.lmsal.com
198.116.7.238=redsox.lmsal.com
198.116.7.239=gpibenet3.lmsal.com
198.116.7.240=moonrise.lmsal.com
198.116.7.241=shawnee.lmsal.com
198.116.7.242
198.116.7.243=GoBlue.lmsal.com
198.116.7.244=heman.lmsal.com
198.116.7.245=xuv.lmsal.com
198.116.7.246=sxt1.lmsal.com
198.116.7.247=lotus.lmsal.com
198.116.7.248=shine.lmsal.com
198.116.7.249=notung.lmsal.com
198.116.7.250=gpibenet4.lmsal.com
198.116.7.251=gpibenet_em2.lmsal.com
198.116.7.252=hikari.lmsal.com
198.116.7.253=ccdlab.lmsal.com
198.116.7.254
198.116.7.255

The most elementary discovery tool is ping. This sends an ICMP echo_request packet and listens for an ICMP echo_reply. If it comes then the host we've pinged is up. If it doesn't come our ICMP echo_request has probably been dropped by a firewall or the host is down. Usually it's a good idea to ping the broadcast address or the network address of a network. Some hosts might respond. In our case we won't do it, but this would have been the commands:

$ ping –b –c 1 198.116.7.0
$ ping –b –c 1 198.116.7.255

Let's send a simple ICMP echo_request with hping:

```
root@altos:/home/nabu# hping -V -c 1 -C 8 216.239.59.104
using eth0, addr: 192.168.228.128, MTU: 1500
HPING 216.239.59.104 (eth0 216.239.59.104): icmp mode set, 28 headers + 0 data bytes
len=46 ip=216.239.59.104 ttl=128 id=21856 tos=0 iplen=28
icmp_seq=0 rtt=109.9 ms

--- 216.239.59.104 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 109.9/109.9/109.9 ms
root@altos:/home/nabu#
```

-V makes our output verbose, –c 1 means we're sending only one packet and finally –C 8 means we're sending ICMP type 8 (that's echo_request). I've surrounded the response of 216.239.59.104 (www.google.com) with a red rectangle.

But ICMP type 8 (echo request), isn't the only type of ICMP request. We can expect answers from other ICMP packets as well. These are: timestamp request, information request and address mask request.

Let's try to discover if nasa.gov is powered on with these 3 type of packets.

First we'll use an ICMP type 13 request (timestamp request):

```
root@altos:/home/nabu# hping -c 1 -C 13 nasa.gov
HPING nasa.gov (eth0 198.116.144.49): icmp mode set, 28 headers + 0 data bytes
len=46 ip=198.116.144.49 ttl=1 id=11658 icmp_seq=0 rtt=173.8 ms
ICMP timestamp: Originate=35182675 Receive=83520207 Transmit=83520207
ICMP timestamp RTT tsrtt=175


--- nasa.gov hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 173.8/173.8/173.8 ms
root@altos:/home/nabu#
```

It worked. Host On. Now let's try and ICMP type 17 request (address mask request):

```
root@altos:/home/nabu# hping -c 1 -C 17 nasa.gov
HPING nasa.gov (eth0 198.116.144.49): icmp mode set, 28 headers + 0 data bytes

--- nasa.gov hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@altos:/home/nabu#
```

It didn't work… no surprise. But keep in mind that this "discovery request" usually works with routers. We could also try to send an ICMP type 15 request (information request), but we can't do it with hping because it doesn't have it implemented (I hope it's on the "to do" list). If you really want to try that too, get the icmpenum program.
http://www.nmrc.org/project/misc/icmpenum-1.1.1.tgz
or sing (Send ICMP Nasty Garbage)
http://sourceforge.net/project/showfiles.php?group_id=6994

```
root@altos:/home/nabu# sing -c 1 -info nasa.gov
SINGing to nasa.gov (198.116.144.49): 8 data bytes

--- nasa.gov sing statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
root@altos:/home/nabu#
```

All these ICMP requests can also be sent to the broadcast or the network address. Hopefully some hosts will respond.

Let's get back to our LMSAL hosts list. I hope you've read the manual of both commands (hping and tcptraceroute). If so, let's start our host discovery with the first hosts:

198.116.7.0
198.116.7.1=atcnsifw.lmsal.com
198.116.7.2
198.116.7.3
198.116.7.4
198.116.7.5=cat2950-b206lmsal-6a1-a.lmsal.com
198.116.7.6=mail.lmsal.com
198.116.7.7=cat4006-b252lmsal-109-a.lmsal.com
198.116.7.8=www.lmsal.com
198.116.7.9
198.116.7.10=dns.lmsal.com

We see 198.116.7.6=mail.lmsal.com… it surely has a SMTP daemon on port 25, 198.116.7.8= www.lmsal.com it must have a web daemon on port 80 and 198.116.7.10= dns.lmsal.com should listen for DNS requests on 53. Let's test our assumptions.

First we'll send a SYN packet for port 25 to mail.lmsal.com. It should respond with SYN ACK if it allows connections, and RST if not. In both cases we'll know the host is alive because it sent us a response back. If we would have sent our packet to another port, it probably would have been dropped by the firewall. Why? Because it surely has a rule like that:

Allow traffic (TCP, port 25) to mail.lmsal.com
Drop any other shit

```
root@altos:/home/nabu# hping -V -c 1 -S -p 25 198.116.7.6
using eth0, addr: 192.168.228.128, MTU: 1500
HPING 198.116.7.6 (eth0 198.116.7.6): S set, 40 headers + 0 data bytes
len=46 ip=198.116.7.6 ttl=128 id=22239 tos=0 iplen=44
sport=25 flags=SA seq=0 win=64240 rtt=174.6 ms
seq=2044355617 ack=1069115710 sum=789e urp=0


--- 198.116.7.6 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 174.6/174.6/174.6 ms
root@altos:/home/nabu#
```

-p 25 means destination port 25 and –S means that we're sending a TCP datagram with the SYN flag set. As we can see, 198.116.7.6=mail.lmsal.com sent us a response with flags SA (SYN ACK), so the SMTP service is available, the port is open, the host is on.

198.116.7.6=mail.lmsal.com ON

The same basic technique works with the other two hosts. For 198.116.7.8=www.lmsal.com we modify the command so that it will send to port 80 (-p 80), and for 198.116.7.10=dns.lmsal.com on port 53 (-p 53).

198.116.7.8=www.lmsal.com ON
198.116.7.10=dns.lmsal.com ON

Probably a better rule on the firewall for dns.lmsal.com would have been:

Allow traffic (UDP, port 53) to dns.lmsal.com
Drop any other shit

But since it responded to our TCP (SYN) datagram it probably has something like this:

Allow traffic (UDP and TCP, port 53) to dns.lmsal.com
Drop any other shit

Tcptracerouting to www.lmsal.com and dns.lmsal.com doesn't give us any other useful information except that they're all behind 192.67.126.2. You can test it yourself. In the best case it could have shown us other LMSAL hosts which were routing our packets and that would have meant they were alive.

```
root@altos:/home/nabu# tcptraceroute -f 6 -q 1 -w 10 198.116.7.6 25
Selected device eth0, address 86.107.80.114, port 1036 for outgoing packets
Tracing the path to 198.116.7.6 on TCP port 25 (smtp), 30 hops max
 6  de-fra01a-rd2-pos-5-0.aorta.net (213.46.179.6)  22.144 ms
 7  us-ewr01a-rd1-pos-4-0.aorta.net (213.46.160.146)  69.779 ms
 8  ge-6-4-210.car2.Newark1.Level3.net (4.78.20.129)  79.432 ms
 9  ae-2-54.bbr2.Newark1.Level3.net (4.68.99.97)  74.385 ms
10  ae-0-0.bbr1.Washington1.Level3.net (64.159.0.229)  75.144 ms
11  ae-11-53.car1.Washington1.Level3.net (4.68.121.82)  86.155 ms
12  4.79.228.22  82.733 ms
13  128.161.3.14  84.606 ms
14  192.150.40.253  216.661 ms
15  128.161.21.54  202.896 ms
16  192.67.126.2  204.922 ms
17  mail.lmsal.com (198.116.7.6) [open]  126.873 ms

root@altos:/home/nabu# tcptraceroute -f 6 -q 1 -w 10 www.lmsal.com 80
Selected device eth0, address 86.107.80.114, port 1037 for outgoing packets
Tracing the path to www.lmsal.com (198.116.7.8) on TCP port 80 (www), 30 hops max
 6  de-fra01a-rd2-pos-5-0.aorta.net (213.46.179.6)  18.266 ms
 7  us-ewr01a-rd1-pos-4-0.aorta.net (213.46.160.146)  85.084 ms
 8  ge-6-4-210.car2.Newark1.Level3.net (4.78.20.129)  80.467 ms
 9  ae-2-52.bbr2.Newark1.Level3.net (4.68.99.33)  82.405 ms
10  ae-0-0.bbr1.Washington1.Level3.net (64.159.0.229)  78.123 ms
11  ae-11-51.car1.Washington1.Level3.net (4.68.121.18)  80.748 ms
12  4.79.228.22  103.624 ms
13  128.161.3.14  122.620 ms
14  192.150.40.253  229.713 ms
15  128.161.21.54  245.017 ms
16  192.67.126.2  245.679 ms
17  www.lmsal.com (198.116.7.8) [open]  227.716 ms


root@altos:/home/nabu# tcptraceroute -f 6 -q 1 -w 10 dns.lmsal.com 53
Selected device eth0, address 86.107.80.114, port 1038 for outgoing packets
Tracing the path to dns.lmsal.com (198.116.7.10) on TCP port 53 (domain), 30 hops max
 6  de-fra01a-rd2-pos-5-0.aorta.net (213.46.179.6)  33.890 ms
 7  us-ewr01a-rd1-pos-4-0.aorta.net (213.46.160.146)  139.690 ms
 8  ge-6-4-210.car2.Newark1.Level3.net (4.78.20.129)  165.961 ms
 9  ae-2-52.bbr2.Newark1.Level3.net (4.68.99.33)  160.731 ms
10  as-3-0.bbr2.Washington1.Level3.net (4.68.128.206)  124.287 ms
11  ae-21-54.car1.Washington1.Level3.net (4.68.121.114)  117.777 ms
12  4.79.228.22  112.334 ms
13  128.161.3.14  112.788 ms
14  192.150.40.253  151.845 ms
15  128.161.21.54  153.563 ms
16  192.67.126.2  155.356 ms
17  dns.lmsal.com (198.116.7.10) [open]  156.602 ms
```

How can we proof that 192.67.126.2 is the firewall for all this hosts? Well, easy if it's configured by an idiot. We'll send a SYN packet with TTL (time to leave) 16 to, let's say, dns.lmsal.com. Since TTL is only 16 it will only get till 192.67.126.2, which is hop 16 in the above trace. Now… if this wasn't a firewall it would send us back an ICMP Type 1 Code 0 packet (TTL exceeded in transit), whatever the port we were sending to. If this was a firewall, it would send back ICMP Type 1 Code 0 only if our packet wasn't already dropped. But remember… it would drop it if it was not destined for an open port. Check this:

```
root@altos:/home/nabu# hping -c 1 -S -t 16 -p 25 mail.lmsal.com
HPING mail.lmsal.com (eth0 198.116.7.6): S set, 40 headers + 0 data bytes
TTL 0 during transit from ip=192.67.126.2 name=UNKNOWN

--- mail.lmsal.com hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@altos:/home/nabu# hping -c 1 -S -t 16 -p 31337 mail.lmsal.com
HPING mail.lmsal.com (eth0 198.116.7.6): S set, 40 headers + 0 data bytes

--- mail.lmsal.com hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@altos:/home/nabu#
```

With the –t 16 option we set the TTL value to 16. Since in the second case (when we chose a random port) 192.67.126.2 did not respond any more, this is clearly the firewall for our hosts. I hope you understood what I did; it's very easy if you pay attention. We can also use this technique to discover open ports on their hosts using their firewall! So the firewall, which should have stopped us... is actually helping. We thank NASA for its security experts since 192.67.126.2 is on one of their netblocks. Check a port discovery:

```
root@altos:/home/nabu# hping --scan 1-1024 -S -t 16 mail.lmsal.com
Scanning mail.lmsal.com (198.116.7.6), port 1-1024
1024 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
  21:                       1    68 47189   (ICMP  11   0 from 192.67.126.2)
  20:                       1    68 47190   (ICMP  11   0 from 192.67.126.2)
  22:                       1    68 47191   (ICMP  11   0 from 192.67.126.2)
  25:                       1    68 47192   (ICMP  11   0 from 192.67.126.2)
  80:                       1    68 47193   (ICMP  11   0 from 192.67.126.2)
 110:                       1    68 47197   (ICMP  11   0 from 192.67.126.2)
 113:                       1    68 47198   (ICMP  11   0 from 192.67.126.2)
 143:                       1    68 47203   (ICMP  11   0 from 192.67.126.2)
 443:                       1    68 47222   (ICMP  11   0 from 192.67.126.2)
 674:                       1    68 47234   (ICMP  11   0 from 192.67.126.2)
 993:                       1    68 47257   (ICMP  11   0 from 192.67.126.2)
All replies received. Done.
```

Funny, eh? Their firewall just "scanned" a host for us, between ports 1 and 1024 (--scan 1-1024). What we actually did was simply discovering the firewall rules (or the ACL - access control rules) related to mail.lmsal.com and ports 1-1024. This method is called firewalking. There's a tool that automates that, called firewalk, but there's no need to download it since you can use hping for it.

I hope you understood these simple notions, let's continue checking ON/OFF status from the beginning of the list. 198.116.7.0. This is rarely assigned to a real host as I said a few chapters before.

```
root@altos:/home/nabu# hping -c 1 -S -p 22 198.116.7.0
HPING 198.116.7.0 (eth0 198.116.7.0): S set, 40 headers + 0 data bytes

--- 198.116.7.0 hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@altos:/home/nabu# hping -c 1 -A -p 22 198.116.7.0
HPING 198.116.7.0 (eth0 198.116.7.0): A set, 40 headers + 0 data bytes
len=46 ip=198.116.7.0 ttl=128 id=22590 sport=22 flags=R seq=0 win=32767 rtt=12.0 ms

--- 198.116.7.0 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 12.0/12.0/12.0 ms
root@altos:/home/nabu#
```

First we sent to 198.116.7.0 a SYN on port 22. It did not give any response, nor RST or SYN ACK. So if it really was a host there, then the firewall was configured to drop our packets because they were destined for a closed port. But, as we've seen before, some people aren't able to properly configure a firewall, although they do it for a government organization. If a host receives an ACK packet, as if we were already connected to them, it will respond with a RST. Firewalls should be configured to not allow ACK packets entering the network, unless they're part of a three-way handshake or a connection has already been established by that source address with the destination address (stateful[1] firewalls can do that). Luckily for us, everything is fucked up in LMSAL's protection. We sent an ACK packet to 198.116.7.0 and the host responded with RST. Host on.

198.116.7.0 ON

Now check 198.116.7.1= atcnsifw.lmsal.com. By the name this could actually be another firewall (atcnsi**fw**, atc = Applied Technology Center?, nsi = NASA Science Internet? or Network Solutions Incorporated?). We don't know what's protecting if it's really a firewall, but we also don't care right now. Let's send an ACK to it.

```
root@altos:/home/nabu# hping -c 1 -A -p 22 198.116.7.1
HPING 198.116.7.1 (eth0 198.116.7.1): A set, 40 headers + 0 data bytes
len=46 ip=198.116.7.1 ttl=128 id=22620 sport=22 flags=R seq=0 win=32767 rtt=9.6 ms

--- 198.116.7.1 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 9.6/9.6/9.6 ms
root@altos:/home/nabu#
```

Again, it worked. Nice job till now.

198.116.7.1=atcnsifw.lmsal.com ON

If we wanted to tcptraceroute to the last two IPs we could have used the –A parameter, which would have set the ACK flag on outgoing packets, therefore making the firewall "welcome us in the network".

I'm having a premonition that this methods will work with the rest of the hosts. As homework, make a script that will scan all the remaining hosts

[1] There are two main types of firewalls: Stateless firewalls treat each network packet (frame) in isolation. Stateful firewalls will monitor network connections and treat each packet as part of a connection.

from the LMSAL netblock in a random order. It should display something like this:

```
ON 198.116.7.0
ON 198.116.7.1=atcnsifw.lmsal.com
ON 198.116.7.2
ON 198.116.7.3
ON 198.116.7.4
ON 198.116.7.5=cat2950-b206lmsal-6a1-a.lmsal.com
ON 198.116.7.6=mail.lmsal.com
ON 198.116.7.7=cat4006-b252lmsal-109-a.lmsal.com
…
OFF whatever=whatever
…
```

It should also use a delay of 20 seconds before sending each packet, because we don't want it triggering an IDS alarm.

If you want tips in making the script or there's something you didn't understand… post your question on the document's forum (www.absolom.ro/f). Also any comments, ideas etc are welcome.

== What Have We Learned and What's Coming ==

It took me about 20 hours in total to write this tutorial, I hope you enjoyed it. There are some scripts presented in it. These are not the scripts that I use in my hacking adventures; they are merely given because I wanted you to make an idea about what type of scripts you could create. Start from this scratches and make your own shit, because there's a lot of space for improvements. Be inventive.

We have learned to understand "enemy networks" and I gave you lots of basic techniques to do that. I have shown you vulnerabilities in some NASA networks and they might patch them, therefore it would be a good idea to try these techniques on other networks. Since you probably are a beginner try them on fucked up networks. You can find lots of these with Google. Search for terms like "shoe store" or "porn download". While waiting for the next parts of the tutorial make sure you've understood everything in this one, cause things will get more complicated (also much more interesting). Remember that if you won't practice what you've learned you won't fully understand it.

In the next part I guess we'll learn advanced mapping, some port discovery techniques, service detection and maybe OS fingerprinting too. This will help us have a complete image of the target network, with the services it's running, so we can begin exploiting them.

Good luck.

== Appendix A – ICMP Types & Codes ==

ICMP TYPE NUMBERS

The Internet Control Message Protocol (ICMP) has many messages that are identified by a "type" field.

Type	Name	Reference
0	Echo Reply	[RFC792]
1	Unassigned	[JBP]
2	Unassigned	[JBP]
3	Destination Unreachable	[RFC792]
4	Source Quench	[RFC792]
5	Redirect	[RFC792]
6	Alternate Host Address	[JBP]
7	Unassigned	[JBP]
8	Echo	[RFC792]
9	Router Advertisement	[RFC1256]
10	Router Solicitation	[RFC1256]
11	Time Exceeded	[RFC792]
12	Parameter Problem	[RFC792]
13	Timestamp	[RFC792]
14	Timestamp Reply	[RFC792]
15	Information Request	[RFC792]
16	Information Reply	[RFC792]
17	Address Mask Request	[RFC950]
18	Address Mask Reply	[RFC950]
19	Reserved (for Security)	[Solo]
20-29	Reserved (for Robustness Experiment)	[ZSu]
30	Traceroute	[RFC1393]
31	Datagram Conversion Error	[RFC1475]
32	Mobile Host Redirect	[David Johnson]
33	IPv6 Where-Are-You	[Bill Simpson]
34	IPv6 I-Am-Here	[Bill Simpson]
35	Mobile Registration Request	[Bill Simpson]
36	Mobile Registration Reply	[Bill Simpson]
37	Domain Name Request	[RFC1788]
38	Domain Name Reply	[RFC1788]
39	SKIP	[Markson]
40	Photuris	[RFC2521]
41	ICMP messages utilized by experimental mobility protocols such as Seamoby	[RFC4065]
42-255	Reserved	[JBP]

Many of these ICMP types have a "code" field. Here we list the types again with their assigned code fields.

Type	Name	Reference
0	Echo Reply	[RFC792]
	Codes 0 No Code	
1	Unassigned	[JBP]
2	Unassigned	[JBP]
3	Destination Unreachable	[RFC792]

Codes
0 Net Unreachable
1 Host Unreachable
2 Protocol Unreachable
3 Port Unreachable
4 Fragmentation Needed and Don't Fragment was Set
5 Source Route Failed
6 Destination Network Unknown
7 Destination Host Unknown
8 Source Host Isolated
9 Communication with Destination Network is Administratively Prohibited
10 Communication with Destination Host is Administratively Prohibited
11 Destination Network Unreachable for Type of Service
12 Destination Host Unreachable for Type of Service
13 Communication Administratively Prohibited [RFC1812]
14 Host Precedence Violation [RFC1812]
15 Precedence cutoff in effect [RFC1812]

4 Source Quench [RFC792]

Codes
0 No Code

5 Redirect [RFC792]

Codes
0 Redirect Datagram for the Network (or subnet)
1 Redirect Datagram for the Host
2 Redirect Datagram for the Type of Service and Network
3 Redirect Datagram for the Type of Service and Host

6 Alternate Host Address [JBP]

Codes
0 Alternate Address for Host

7 Unassigned [JBP]

8 Echo [RFC792]

Codes
0 No Code

9 Router Advertisement [RFC1256]

Codes
0 Normal router advertisement
16 Does not route common traffic [RFC2002]

10 Router Selection [RFC1256]

Codes
0 No Code

11 Time Exceeded [RFC792]

Codes
0 Time to Live exceeded in Transit
1 Fragment Reassembly Time Exceeded

12 Parameter Problem [RFC792]

Codes
0 Pointer indicates the error
1 Missing a Required Option [RFC1108]
2 Bad Length

13 Timestamp [RFC792]

Codes
0 No Code

14 Timestamp Reply [RFC792]

Codes
0 No Code

15 Information Request [RFC792]

Codes
0 No Code

16 Information Reply [RFC792]

Codes
0 No Code

17 Address Mask Request [RFC950]

Codes
0 No Code

18 Address Mask Reply [RFC950]

Codes
0 No Code

19 Reserved (for Security) [Solo]

20-29 Reserved (for Robustness Experiment) [ZSu]

30 Traceroute [RFC1393]

31 Datagram Conversion Error [RFC1475]

32 Mobile Host Redirect [David Johnson]

33 IPv6 Where-Are-You [Bill Simpson]

34 IPv6 I-Am-Here [Bill Simpson]

35 Mobile Registration Request [Bill Simpson]

36 Mobile Registration Reply [Bill Simpson]

39	SKIP	[Markson]
40	Photuris	[RFC2521]

Codes
0 = Bad SPI
1 = Authentication Failed
2 = Decompression Failed
3 = Decryption Failed
4 = Need Authentication
5 = Need Authorization

The list was ripped from:
http://www.iana.org/assignments/icmp-parameters

== Appendix B – Google Advanced Operators ==

Google ignores common words and characters such as “where” and “how”. **+where + is Romania** –> will include “where” and “is” in the search, although they are common words. Google searches are not case sensitive.

“where is Romania” –> will search for this exact phrase

where is Romania –Dracula –> will search for pages containing the first 3 words, but not the last one

~trojan source code –> will search for pages containing the word “trojan” or words related to it (backdoor, orifice, etc.) and will search only on those pages also containing the words “source” and “code”

where is Romania cache:www.google.com –> will search for the first 3 words in Google’s cache

link:www.absolom.ro –> will search for all the pages that contain a link to www.absolom.ro

related:www.absolom.ro –> will search for all the pages similar to www.absolom.ro

define:hacker –> will provide definitions to the word “hacker”, gathered from various online sources

site:absolom.ro Nabukadnezar –> will search for “Nabukadnezar” on the website absolom.ro

allintitle:The Absolom Group –> will search websites that have all 3 words in the title

intitle:Absolom Nabukadnezar –> will search for pages having “Absolom” in their title and “Nabukadnezar” anywhere on the page

allinurl:packet storm –> will search for pages having an URL containing “packet” and “storm”

inurl:absolom Nabukadnezar -> will search for an URL containing “Absolom” and going to a page that contains “Nabukadnezar”

For examples on how these operators can be combined to find interesting things on the internet, check:
http://johnny.ihackstuff.com/index.php?module=prodreviews