# BASH Basics

## ****BASH Scripting****

To be a professional hacker – not just a script-kiddie – you must develop some coding/scripting capabilities. Often, we need to automate some commands – at a minimum – or write new original code for such things as rootkits and other exploits. In any case, to become a professional hacker, you need to have some basic scripting/coding skills. I will attempt to get you started here on some basic scripting, but will return to more advanced scripting later in the course.

The more you know in this area, the better you will become. Ideally, the professional hacker has some C, Java, Python, Perl, and Ruby capabilities, but at a very minimum, the professional hacker must be able to write simple BASH scripts. This module is dedicated to writing simple scripts in the Bourne Again Shell or BASH. For those of you with good BASH scripting skills, please bear with us. You may want to skip directly to the exercise for this lesson.

Although the various Linux distributions ship with multiple shells, the BASH shell is contained on nearly all of them and is probably the most popular shell.

In this module, we will develop a simple script to scan a network for the availability of a particular port. I am assuming a basic understanding of the BASH shell and nmap scanning.

## ****I. BASH Scripting Basics****

The BASH shell allows us to automate some tasks and commands by writing a “script”. A script is simply  a stored file with a set of commands that can be run again and again.

As you know, whenever we start a script, we need to tell the operating system what shell or interpreter should execute the file. The operating system checks the initial characters of your script before trying to execute it. If we start our script with the shebang, #!, the terminal then looks immediately after the shebang for the absolute path  to the program that is to execute the following code. So, if we want to use the BASH interpreter, we need to start our script with **#! /bin/bash**. This tells the operating  system we want the following commands executed by the BASH interpreter that can be found at **/bin/bash**.

To help make a BASH shell easier to debug, we can use several options when we call the BASH interpreter.  The **-e** option causes the BASH  interpreter to exit whenever a simple command fails. The **-x** option turns on debugging mode. We probably only want to use the **-x** option during the development of our script as it becomes a bit unwieldy when in a production environment.

I will using the built-in GUI text editor from Kali, Leafpad, for these exercises, but you can use any text editor you please. So, we will start our script with:

#! /bin/bash -e

With this in mind, let’s start with this simple script.



Let’s go through this script line by line.

echo "What is your name?"

This command simply tells the shell to prompt the user for their name. Anything after the echo command will appear in stdout

read name

This command grabs the input from stdin and puts it in a variable called “name”

echo "What module of this training are you in?"

This command tells the shell to display on stdout the words between the ” “.

read module

This command grabs the input from stdin and puts it in a variable called “module”

echo "Welcome " $name " to Module ", $module, "of Hakin9's  Advanced Offensive Security Hacking course!"

Finally, we can take the input from the user and put it into an introduction to the user through stdout.

Now, save this file and change its permissions to allow us to execute it

kali > chmod 755 intro

Now, let’s run our script.

kali > ./intro

When we do so, we should see output similar to this with the exception that the name will vary.

 ****

## ****II. Creating a BASH script scanner****

Now that you have some very  basic scripting skills, let’s move to some slightly more advanced scripting that has some real world application to hacking. Let’s use an example from the world of blackhat hacking.

Presently, there is hacker serving time in U.S. federal prison by the name of Max Butler or Max Vision, as he was known throughout the hacker world. Max was a kind of gray hat hacker. By day, he was an IT security professional in Silicon Valley, CA and by night, he was stealing and selling credit card numbers on the black market. At one time, he ran the world’s largest credit card black market, cardersmarket. Now, Max is serving a 14 year prison term, while at the same time assisting CERT in Pittsburgh.

A few years before Max was caught, he realized that the Aloha Point of Sale (POS) system that is used by many small restaurants, had a technical support capability built into the system. This capability enabled tech support to assist their clients. It was a kind of proprietary RDP. Aloha tech support could access the end user’s system  through port 5505 to provide assistance when the user called.  Max realized if that if he found a system connected to the Internet with the Aloha POS system, you could access the system sysadmin privileges through port 5505. Max was able to enter many of these systems and steal thousands of credit card numbers.

Eventually, Max wanted to find EVERY system that had that port 5505 open so that he could go from stealing thousands of credit card numbers, to stealing millions of credit card numbers. Max decided to write a script that would scan millions of IP addresses in the U.S. looking for systems with port 5505 open. Of course, most systems do NOT have port 5505 open and if they did, it was likely they were running the doomed Aloha POS. He could run this script while at work during the day, come back at night and hack into those systems identified as having port 5505 open, at night.

Let’s write a script that Max may have used to scan for that infamous port 5505 and put those IP addresses into a file. I’m assuming that you have some rudimentary nmap skills, but if not, review your nmap commands before proceeding.

Before we write the script to scan public IPs across the Internet, let’s take on much smaller task to test our script. Instead of scanning the globe, let’s first test our script on our local area network and let’s scan for port 80 to see whether our script actually works. If it does, we can easily edit it to do the much bigger task.

Let’s start here:

#!bin/bash -e

nmap -sT  192.168.1.0/24 -p80 -oG Aloha

This line of our script runs an nmap connect scan (-sT) on the subnet 192.168.1.0 looking for port 80 (-p80) and sends the output (-oG) to a file named Aloha.

cat Aloha | grep open  > Aloha2

This line “cats” the file Aloha and filters for the word “open” and sends the output to Aloha2.

cat Aloha2 | cut -f2 -d ":" | cat -f1 -d "(" > Aloha3

This line is the most important. Here we cat Aloha2 file and then cut the output at the second field (-f2)  when we use the delimiter : (-d”:”) and then cut that output at the first field  (-f1) when we use the delimiter (  (-d “(“)and then send that output to Aloha3.

cat Aloha3

Finally, we display the contents of Aloha3.



Now, if we did everything correct, we should see output like that below. Your output may differ based upon what machines are up on your network and what ports they have open.



Now that we have tested our script on our local network, let’s expand it to look for public IPs across the Internet with port 5505 open. We simply need to edit our script for the IPs it scans and the port it is looking for, like below.

#### https://hakin9.org/wp-content/uploads/2015/02/fffd869959acd6cc606c2b1aaf5c2d13.png

####

Note that I have edited the script to scan 65535 IPs on the subnet 74.129.0.0. scanning for port 5505.

Now, let’s save it and name it alohascript and give ourselves execute permissions. Then run it.

****

It you did everything correctly, you should see output to your screen initially like that above, but eventually, after scanning 65535 IP addresses, if it finds any with port 5505 open, it will list each of those IP addresses.

## ****Exercise for Module 1, Lesson 2:****

Take what you learned in this module on BASH scripting and create a script that will prompt the user for a starting IP address and an ending IP address and the port to scan for (hint: store that info in variables) and output the nmap scan to a file named OffSecScanner.

Test it by looking for SQL Server databases in your range of IP addresses.