Software 4 (Domain Name Service)

Reza Naeemi

Documentation

The spoofcheck software is modeled to mimic the professor's provided executable of the same name, with the exception that mine is named myspoofcheck to avoid conflict with file names. The intent of the software is to allow a server to check the integrity of a client connection by looking for IP Address address spoofing. It performs this integrity check by using DNS to authenticate a client that has established a connection to the server. The spoofcheck implementation references the advertised IP Address address against the client's official host name, aliases, and registered IP Address addresses.

Note that spoofcheck is only compatible with IPv4 connections. Usage of spoofcheck is as a hosting server only. The binary can be compiled and executed on any Linux based OSs. Usage is ./spoofcheck [port number]

A client must only establish a TCP connection with the spoofcheck server as no other protocol is used. After establishing a connection, the spoofcheck server checks if the client is spoofing the IP address or if it is, actually, an honest connection, then it immediately ends the connection.

Note that client connections are subject to be reported as spoofing if no DNS mapping exists for a client IP Address, even if connections are actually authentic. For this reason, spoofcheck is not a robust method of checking for actual IP Address spoofing and should be viewed as just an educational example for Linux DNS system calls in the C++ programming language.

The spoofcheck server is persistent and will remain running until manually shut down by killing the parent process. Multiple connections can be established through threaded operation on the supplied port number while the server is running.

Execution Console Output

I ran tests using Telnet through the Command Prompt from the uw2-140 windows computer lab, while also remotely being logged into uw1-320-00, through an SSH connection on my own computer. My version of the spoofcheck server was running on uw1-320-00 and I performed a telnet connection from the localhost, uw1-320-00, and through Putty directly from a Windows based machine in the uw2-140 lab.

```
client addr = 127.0.0.1 port number = 44827
official hostname: localhost.localdomain
alias: localhost
ip address: 127.0.0.1 ... hit!
an honest client

client addr = 69.91.202.149 port number = 51434
official hostname: uw2-140-c0yb7y1.uwb.edu
alias: none
ip address: 69.91.202.149 ... hit!
an honest client

client addr = 127.0.0.1 port number = 44828
official hostname: localhost.localdomain
alias: localhost
ip address: 127.0.0.1 ... hit!
an honest client
```

```
client addr = 69.91.198.151 port number = 33490
official hostname: uw1-320-00.uwb.edu
alias: uw1-320-00
ip address: 69.91.198.151 ... hit!
an honest client

client addr = 69.91.198.166 port number = 58087
official hostname: uw1-320-15.uwb.edu
alias: uw1-320-15
ip address: 69.91.198.166 ... hit!
an honest client

client addr = 69.91.198.166 port number = 58088
official hostname: uw1-320-15.uwb.edu
alias: uw1-320-15
ip address: 69.91.198.166 ... hit!
an honest client
```

The following is what I got when sitting at the North Juanita Startbucks, connected to the Google Starbucks WAP with an Comcast Business ISP connection:

```
client addr = 50.248.211.51 port = 52952
official hostname: 50-248-211-51-static.hfc.comcastbusiness.net
alias: none
ip address: 50.248.211.51 ... hit!
an honest client
```

For both Linux computers and the Windows based machines, a DHCP obtained IP Address is used, and proper DNS forwarding was set up for those IP Addresses. As a result, the connections were successful using spoofcheck. The saved console outputs has been included in this Program 4 Lab submission.

After some searching, I was lucky enough to find a Windows computer at the UWB campus that did not have DNS forwarding properly setup, so I was able obtain the output for it:

```
client addr = 172.56.42.170 port number = 44368
gethostbyaddr error for the client( 172.56.42.170): 1
a spoofing client

client addr = 172.56.42.215 port number = 21965
gethostbyaddr error for the client( 172.56.42.215): 1
a spoofing client
```

In all cases, my developed version of the spoofcheck software mimicked the professor's provided version.

Analysis

To establish a connection, the client must simply initiate a TCP handshake with the listening spoofcheck server communicating on the same port number. Once connection has been established with the server, flow of control is left exclusively to the spoofcheck server. Server-initiated TCP disconnection occurs immediately after the spoofcheck server completes the verification process on the host-name-to-address mapping of the client. Since the spoofcheck server does not require or except the client to initiate disconnection, the connection is closed on

the server end. This is important to ensure that connections are closed and do not end up simply timing out as a result of the client process being terminated.

Clients using DHCP obtained IP addresses are no more subject to errors in spoofcheck authentication than clients with statically assigned IP addresses. Due to the nature of how DHCP works, the DHCP server is typically closely tied to a DNS server, where proper forward DNS resolution is a requirement for spoofcheck to function properly. Therefore, the method that is used to obtain an IP Address is irrelevant so as long as the proper DNS records exist for spoofcheck to function. The actual process used to obtain the IP Address is transparent to TCP.

Client verification while connecting through NAT may in some cases not work, depending on the network configuration. The most obvious failure is when DNS is not properly configured on the NAT WAN address. A pure NAT operating on IP Address alone may not correctly handle traffic where as soon as the protocol stack is traversed, basic protocols such as TCP and UDP can break unless NAT takes the proper action beyond the network layer, however devices operating behind a NAT are otherwise virtually indistinguishable from the other legitimate hosts.

In the case of FTP, it uses separate connections from control traffic and for data traffic (Active mode) where the host making the request identifies the corresponding data connection by its network layer and transport number layer addresses. If the host making the request lies behind a simple NAT firewall, the translation of the IP Address and/or TCP port makes the information received by the server invalid.

For spoofcheck, this type of bi-directional communication does not exist and is not possible without some type of matching client. Because the TCP hostent data structure is built from what the server observes using DNS resolution and not from TCP headers that a client provides, the resultant FQDN returned by the DNS server is the official hostname obtained by gethostbyaddr or gethostbyname functions. Spoofcheck is therefore not affected by NAT where the publically accessible IP Address has the proper matching forward DNS resolution.