Homework #5 Solution

Contact TAs: vegetable@csie.ntu.edu.tw

Network Administration

1. DHCP (10%)

When you try to renew the ip your computer get from DHCP server, you may always get the same IP. What may be the reason?

Answer: DHCP server may reserve IP addresses for each MAC address. As long as you use the same MAC address, you will get the same IP from the DHCP server.

2. DNS (40%)

(a) (5%) Why should DNS be distributed? Please name three disadvantages if a single server handles all domain name translation services.
Answer:

- A single server lacks redundancy. If the server is down, then nobody can access DNS service.
- A single server may cause high latency for cilents that are far from the server.
- Holding all the DNS records on a single server is hard to maintain and easily attacked by others.

(b) (5%) What is DNS cache? Why is it helpful? How it works? Simply describe it.
Answer: DNS cache is the temporary storage of information about previous DNS lookups on a operating system or web browser. Caching a recent DNS lookup makes resolving URL to its corresponding IP much more efficient. DNS cache can occur at not only our local machine but also resolver, root server and so on. Therefore even if we can not find record in our local DNS cache, the resolver may have what we want, avoiding the need to go through the time-consuming complete DNS lookup process.

(c) (10%) dig.sh is a bash script that can do type A DNS query and print out the response using hexdump. Try the command `./dig.sh www.csie.ntu.edu.tw` on CSIE workstation and point out where the IP of www.csie.ntu.edu.tw is in the response.
Answer:

```
00000000  00 00 81 80 00 01 00 01  00 03 00 03 03 77 77 77  |.............www|
00000010  04 63 73 69 65 03 6e 74  75 03 65 64 75 02 74 77  |.csie.ntu.edu.tw|
00000020  00 00 01 00 01 c0 0c 00  01 00 01 00 00 01 15 00  |................|
00000030  04 8c 70 1e 1a c0 10 00  02 00 01 00 00 01 15 00  |..p.............|
00000040  09 06 63 73 6d 61 6e 32  c0 10 c0 10 00 02 00 01  |..csman2........|
00000050  00 00 01 15 00 08 05 6e  74 75 6e 73 c0 15 c0 10  |.......ntuns....|
00000060  00 02 00 01 00 00 01 15  00 08 05 63 73 6d 61 6e  |...........csman|
00000070  c0 10 c0 6a 00 01 00 01  00 00 00 89 00 04 8c 70  |...j...........p|
00000080  1e 0d c0 56 00 01 00 01  00 00 e0 9f 00 04 8c 70  |...V...........p|
00000090  03 10 c0 41 00 01 00 01  00 00 00 89 00 04 8c 70  |...A...........p|
000000a0  1e 0e                                            |..|
000000a2
```

(d) (10%) While doing query in the above problem, you may also see "csman", "csman2", "ntuns" in the response. There are only part of the domain name of the name servers because of compression. Please write down how to decompress and get the full domain name of the name servers and what may happen if DNS server doesn't compress large responses. (hint: DNS uses UDP by default)

Answer:

Entire domain names or the end of a domain name is replaced with a pointer to a prior occurrence of the same name. Pointers are two bytes long, with highest two bits set to 1, and following bits is an offset from the start of the message.

For example, " csman" is followed by binary value 1100,0000,0001,0000. It is a pointer to offset 10000, which is 16 in decimal format. We can find out that the pointer points to `.csie.ntu.edu.tw`, and the entire domain name is `csman.csie.ntu.edu.tw`.

With IPv4 standard, only packets with length equal or less than 576 bytes are guaranteed to be reassembled if fragmented in transition. Also, fragmented packets have higher chance to be dropped while transmitting using UDP. Thus, larger responses may be truncated or transmitted using TCP, which requires three way handshake and is slower than UDP. Therefore, we can conclude that it is better to reduce length of responses and keep them transmitted using UDP.

(e) (10%) Please describe 3 types of DNS attacks. How they occur? How to prevent(or detect) them? (hint: DNS cache and DNS server which does not restrict query clients may be abused.)

Answer:

- Cache poisoning attack: If the DNS server does not validate DNS response to ensure that they are from an authoritative source, the DNS server may receive and cache the incorrect record from attackers. We can use secure DNS (DNSSEC) which incorporate cryptographic signatures to prevent this attack.
- DNS amplification attack: Amplification attack occurs by sending small queries that result in large responses, and DNS query can exactly do such thing. The attacker sends UDP packets with spoofed IP address which points to the IP address of the victim to a DNS resolver. These UDP packets make requests to DNS resolver, often including arguments such as "ANY"to receive the largest response. Then the network infrastructure of the victim becomes overwhelmed by those large responses from DNS server. One way to prevent this attack is to reduce the number of open DNS resolvers. Restricting a DNS resolver so that it will only respond to queries from trusted sources makes the server have less possibility to be exploited by attackers.
- DNS Tunneling: DNS Tunneling exploits common and useful DNS protocol. By encoding the data of other programs or protocols in DNS queries and responses, attackers can hide and transmit their malicious program. Such hidden data often occurs in TXT, NULL type DNS record. It is worthwhile to note that an important technique of this attack is to change the domain name in each query to elude the DNS cache. We can block very large UDP and TCP packets, or block TXT and NULL type record to prevent this attack.

System Administration

In Arch Linux, we usually use `pacman` to manage packages. For questions 1-3, briefly describe your answers. For questions 4-9, write down your commands based on Arch Linux. For question 10, you have to submit a tarball, we will talk about it later. You can assume that package `pacman` and `pacutils` are pre-installed.

1. (5%) What are the differences between rolling release and fixed release Linux distributions?

   Fixed

   - **Description**: every version will be released and maintained periodically; usually only minor updates will be performed in the same version number.
   - **Release cycle**: take Debian for example: in Debian, there are three stages of a package: unstable, testing, stable. Upstream developers upload their new versions to unstable repo. After an unstable package gains maturity, it will be migrated to testing. If the package survives in testing without serious problems, the maintainers of Debian will migrate it to stable and the new Debian release will include the package. For packages in stable, only security updates may be performed.
   - **Versioning**: yes
   - **Packages**: usually older
   - **Stability**: higher
   - **Upgrading**: more difficult. Prone to failure due to larger differences between the two versions, sometimes needs re-installation.
   - **Example**: Ubuntu, CentOS.

   rolling

   - **Description**: strictly speaking, there's no "versions" in such sort of distribution: new software packages are constantly put into production not long after their release and without thoroughly testing their stability.
   - **Release cycle**: in rolling releases distros, new packages are shipped into production without thorough testing; since there are no discrete versions of the distro, the software repo is simply updated incrementally (i.e. replace the packages with newer ones if there's any)
   - **Versioning**: no
   - **Packages**: usually the latest
   - **Stability**: lower
   - **Upgrading**: easier. Just upgrade all the packages with the package manager.
   - **Example**: Arch, Gentoo.

2. (10%) As you may know, the Linux workstations in our department currently run Arch Linux, and many users rely on them to have their jobs/homework/projects done. Based on your understandings on rolling/fixed release models from the previous question, what are the pros and cons of using a rolling release Linux distribution (such as Arch) as workstations, rather than a fixed release distribution (such as Ubuntu/CentOS)? Which one do you think is more suitable for workstations, and why?

   (5%) Pros:

   - suitable for developers for using/testing new features when new software packages release

- softwares packages can be updated more frequently, no need to wait for a long time to use new versions
- full system upgrade can be done more easily, sometimes even without requiring to temporarily make services off-line.

Cons:

- potentially more unstable and more buggy

(5%) personal opinion on workstation application - criteria: full mark (5%) will be given as long as the argument is complete and appropriate: the statements cannot be found any discrepancy with actual facts or logically contradictory and the argument as a whole is expressed lucidly without equivocality or impertinent contexts. Minor points may be deducted (3-4%) if the argument has some perceivable flaws; more points may be deducted (1-2%) if the argument fails to support one's opinion. Zero (0%) will be given if one answers nothing, or merely something unrelated.

3. (5%) What are the differences between `pacman -Syu` and `pacman -Sy`, and why the latter is not recommended?

   `pacman -Syu` will download a copy of the package database from the server(s) defined in pacman.conf and upgrade all packages that are out-of-date, while `pacman -Sy` will only perform the first part. Use `pacman -Sy` followed by `pacman -S <package>` will cause partial upgrades, which are not supported in Arch Linux, due to the natural of rolling release.

4. (5%) How to search for a package, say `vim`, either from (a) installed packages or from (b) package databases?

   (a) pacman -Qs vim
   (b) pacman -Ss vim

5. (2.5%) How to list all dependencies of an installed package, say `firefox`?

   ```
   pactree firefox
   ```

6. (2.5%) How to find which package a file in the file system belongs to, say `/etc/resolv.conf`?

   ```
   pacman -Qo /etc/resolv.conf
   ```

7. (2.5%) How to remove a package, say `emacs`, as well as its dependencies which are not required by any other installed packages?

   ```
   pacman -Rs emacs
   ```

8. (2.5%) How to list all orphan packages?

   ```
   pacman -Qdt
   ```

9. (5%) In our in-class lab, we created a package `sudo-oasis`, which depends on `sudo`. Please write down the commands you used to generate a gpg key and sign the `sudo-oasis` package.

   Use `gpg --gen-key` to generate key, and Use `makepkg --sign --key <key> sudo-oasis` to sign package.

10. (10%) Write a script `10.sh` that meet the following criteria.

    - Create a local repository at `/repo`
    - Define it as `nasa-repo` in the config file(s) of `pacman`
    - Add the signed `sudo-oasis` package into `nasa-repo`

- Add the key to pacman keyring so that we can sync with it.

For this question, you must submit a tarball named `10.tar`, which should contains at least four files: `10.sh`, the signed `sudo-oasis` package, the signature file for the package, and the corresponding public key. We will run following code on this clean VM (username: nasa, password: nasa) to test your script.

The following is the sample code of `10.sh`.

```
#!/bin/sh

cat <<EOF > /etc/pacman.d/nasa-repo
[nasa-repo]
Server = file:///repo
EOF

echo "Include = /etc/pacman.d/nasa-repo" >> /etc/pacman.conf

install -d /repo
cp sudo-oasis-0.1-1-any.pkg.tar.xz.sig sudo-oasis-0.1-1-any.pkg.tar.xz /repo
repo-add /repo/nasa-repo.db.tar /repo/sudo-oasis-0.1-1-any.pkg.tar.xz

pacman-key --add public.key
pacman-key --lsign-key <key>
```