Ferhat Elmas
Przemyslaw Pietrzkiewicz
Javier Martín De Valmaseda Castro

Lab-4 DNS

Slightly Harder Reversing

This time, when we disassembled simple, we saw that it was calculating md5 checksum so we get the string and tried to reverse lookup. We had already known the place of the string from previous labs. Check sum was **15b29ffdce66e10527a65bc6d71ad94d** and we tried md5 reverse lookup at http://md5.gromweb.com/. We got **swordfish** as a password and when we enter this password, we got **speakfriendandenter** as a key.

DNS Information Gathering

Version Retrieval

```
$> dig @alfa -c CH -t txt version.bind
```

Here, we set Chaosnet records by -c class parameter, type to txt by -t parameter and requested version.bind at nameserver alfa. Result is as follows:

```
version.bind.   0       CH      TXT     “9.7.3-P1”
```

We can prevent this but this wouldn’t be so secure because actual version can be got even in secured configuration by fpdns tools via this command:

```
$> fpdns -D example.com
```

For fix, we should find version line in named.conf file and edit this line, here we can enter a different string to hide original version but for more enjoy and to play on potential attackers, we can enter wrong version number as if it is forgetten to be changed.

named.conf:

```
options {

        …
        …
        version “9.8.1” // this is final version of named to play on attackers or we can enter an
old version to trick them to try old holes on us that wouldn’t work.
};
```

Listing of the Domain

```
$> dig @alfa example.com AXFR
```

By this command, we can full listing of the domain:
alfa, bravo, charlie, delta, ftp, mysql, ns, smtp, www.

To disable listing with AXFR, we should enter **none** into zone **allow-transfer** field, we can simply enter none since **master has no slaves** nameserver, if we had slaves, we should have entered only their IP addresses into allow-transfer field.

```
zone “example.com” {
        type master;
        file “/var/bind/pri/example.com.zone”;
        allow-query { any; };
        allow-transfer { any; }; // we entered none; instead of any;
};
```

After this change, when we execute above command to get listing, dig says **Transfer failed.**

Insecure Updates

We used **nsupdate** tool to send updates in this way:

```
$>nsupdate
>server 10.10.0.2
>zone example.com
>update add sillytest.example.com 604800 A 10.10.0.6
>send
>quit
$>
```

Since nameserver didn’t require any authentication, we could easily send updates and change the database.

An attacker can take advantage of this, the attacker can insert malicious DNS info, for example directing all attempts to access a particular banking site to a lookalike site under the attacker’s control. In our case, attacker can direct all traffic into example.com and its subdomains to his websites to be a man in the middle.

To fix this updates, we can simply disable updates for the zone by this configuration change,
In global options:

```
options {
        …
        …
        allow-update {
                any;  // change this to none; to globally disable updates for all zones
        };
        …
        ...
};
```

Or in zone block:

```
zone “example.com” {
        type master;
        …
        …
        allow-update { none; }; // add this line to disable updates only for this zone
};
```

After addition, when we execute above command, we get
>update failed: **REFUSED**
as a result.

However, completely disabling updates isn’t a good solution because dynamic update is easier and flexible, we may want to take advantage of it so we should create keys to allow some hosts to update DNS info.

Firstly we create keys for zone (we can also create special keys for users), here algorithm selection is important because some algorithms aren’t compatible with zone parameter, we can choose for example DSA algorithm for zones, HMAC-MD5 for users.
In this command, we used a unrecommended generation, because /dev/urandom continues to generate the key even if sufficient entropy isn’t available. Since we haven’t used vm other than lab assignments, there isn’t sufficient entropy in the system so generation takes so much time, to make faster generation, -r /dev/urandom is favorable. To generate entropy, we could compile kernel in another window :)

```
$>dnssec-keygen -r /dev/urandom -a DSA -b 512 -n ZONE example.com
```

Two files are created, namely:
- Kexample-com.+003+56275.key
- Kexample-com.+003+56275.private

We separated keys from named.conf, so created keys.conf and include it in named.conf:

```
include “keys.conf”;
```

We did because this potentially allows for tighter permissions on the key file, or allowing different groups of administrators read/update access to named.conf or the key file.

keys.conf file contains zone and key pairs:

```
key example.com {
        algorithm DSA; //in our case, but it can RSAMD5, RSASHA1, HMAC-MD5, etc.
        secret ”YrVW9yP6gNMA7VbcU/r2mSIwYnFj/XkCDd6QuqOHE26/ipnrPy+eXrKrUyaFh”;
}
```

In the above, we cropped some part of the secret since it doesn’t fit in to line, important thing is notion. Moreover, secret part is the file with **.key** extension.

And finally we should add this key access into zone in named.conf.
We have already changed related line above to **none** and now we will set it to key:

```
zone “example.com” {
        type master;
        …
        …
        allow-update {
                key example-com;
        };
};
```

Moreover, administrator can give fine grain permission specific host in this syntax:

```
…
update-policy {
        grant <key> <type> <zone> <record-types>;
};
…
```

Another important information is these updates will be written into example.com.zone.jnl, this is the journalling file. Only after some time has passed or a crash, actual example.com.zone file will updated.

Now, for update, we call nsupdate in this manner:

```
$>nsupdate -k Kexample-com.+003+56275.private
```

and then modification commands will be same as above.