dns-grind User Documentation

pentestmonkey@pentestmonkey.net

14 October 2006

Contents

1 Overview **2**

2 Installation **2**

3 Usage **3**

4 Some Examples **3**
4.1 Bruteforcing Hostnames (A-record Lookups) 3
4.2 Finding Registered Domains (NS-record Lookups) 4
4.3 Finding Interesting Hosts In An IP Range (PTR-record Lookups) 4

1 Overview

dns-grind is a tool for performs lots of DNS queries quickly. In particular:

1. Bruteforce guessing of hostnames within a domain, e.g. if foobar.com doesn't allow zone transfers, you can dns-grind to start guessing hostnames: www.foobar.com, test.foobar.com, ftp.foobar.com, etc.

2. Quickly search a list of potential domain names for those that have name servers, e.g. If you're testing for Foobar Enterprises Ltd, might want to search a list of domain like foobar.ac, foobar.ad, etc.

3. Look for interesting hosts in a list of IP addresses by searching for PTR records - the manual equivalent of 'dig -x 10.0.0.1', 'dig -x 10.0.0.2', etc.

You can instruct dns-grind to only query a specific nameserver you're testing, or to act like a normal DNS client and use the DNS configuration from your OS.

You can pass it a simple list of records to look up or you can give prefixes (e.g. www, test, ftp, ...) and suffixes (foobar.com, foo-bar.com). The examples below should make this a bit clearer.

2 Installation

dns-grind is just a stand alone PERL script, so installation is as simple as copying it to your path. It has only been tested under Linux so far.

It depends on the following PERL modules which you may need to install first:

- Net::DNS
- Socket
- IO::Handle
- IO::Select
- Getopt::Std

If you have PERL installed, you should be able to install the modules from CPAN:

```
# perl -MCPAN -e shell
cpan> install Net::DNS
```

3 Usage

dns-grind should first be passed either a file of records to look up or a combination of prefixes and suffixes. The last parameter should be passed is the DNS query type. Only a few are supported currently.

```
Usage: ./dns-grind.pl -f file | (( -p prefix | -P file ) | ( -s suffix | -S file )) query-type

query-type is one of:
        A
        NS
        MX
        PTR

options are:
        -m n     Maximum number of resolver processes (default: 25)
        -p       Prefix of hostname or domain
        -P file  File of hostname or domain prefixes
        -s       Suffix of hostname or domain
        -S file  File of hostname of domain suffixes
        -f       File of hostnames or domains
        -n host  Nameserver to use (default: determined by OS)
        -d       Debugging output
        -r 0|1   Use recursive queries (default: 1)
        -t n     Wait a maximum of n seconds for reply (default: 5)
        -v       Verbose
        -h       This help message
```

Note the -m option above. Generally speaking you want at least 25 query processes running because DNS lookup can be slow if done sequentially by a low number of processes. Be aware that this tool can stress your local recursive DNS server. I've known the DNS service on ADSL routers to fall over if -m is set too high.

If you want to stress a different DNS server instead, use the -n option.

4 Some Examples

4.1 Bruteforcing Hostnames (A-record Lookups)

In the example below, we use a file of hostname prefixes (with '-P' for prefix option) and a domain, pentestmonkey.net (with the '-s' for suffix option). A single A-record is found.

```
$ cat hostname-prefixes.txt
alpha
backup
cray
...
$ dns-grind.pl -P hostname-prefixes.txt -s pentestmonkey.net A
www.pentestmonkey.net 213.165.240.11
```

NB: Wildcard A-records may ruin your search, but you could always 'grep -v wildcard-ip' as a workaround.

A variant of this scan would be to look for subdomains by replacing hostname-prefixes.txt with subdomain-prefixes.txt (us, uk, hq, intranet, etc.) and searching for NS-records instead of A-records.

4.2 Finding Registered Domains (NS-record Lookups)

The premise for this search is that only registered domains have corresponding NS records. The real-world example below shows that this method kinda works, but needs a little refinement.

We supply a prefix of 'pentestmonkey' with the '-p' option, and a file of potential suffixes [1] with the -S option. Note the captial letter in -S or -P to signify a file option, and lowercase -p or -s for a single prefix or suffix[2].

```
$ cat tlds.txt
ac
ad
ae
...

$ dns-grind.pl -p pentestmonkey -S tlds.txt  ns
pentestmonkey.mp        ns1.sdcdns.mp,ns2.sdcdns.mp
pentestmonkey.vg
pentestmonkey.sh        ns1c.nic.ac,ns2c.nic.ac
pentestmonkey.net       ns0.nl.ev6.net,ns0.uk.ev6.net
pentestmonkey.com       ns0.nl.ev6.net,ns0.uk.ev6.net,ns1.uk.ev6.net
pentestmonkey.ac        ns1c.nic.ac,ns2c.nic.ac
pentestmonkey.io        ns1c.nic.ac,ns2c.nic.ac
pentestmonkey.tm        ns1c.nic.ac,ns2c.nic.ac
pentestmonkey.org       ns0.uk.ev6.net,ns1.uk.ev6.net,ns0.nl.ev6.net
```

After a bit of further investigation we find that there's a wildcard NS record for any domain ending in .mp, .sh, .ac, .io, or .tm. These results are effectively false-positives. As is .vg for which a CNAME is returned when we look up the NS record.

The only registered domains with a prefix of 'pentestmonkey' are therefore .com, .net and .org.

4.3 Finding Interesting Hosts In An IP Range (PTR-record Lookups)

Simply supply a list of IP addresses with the -f option. Below we use the genip[3] tool to generate a list of IPs first.

```
$ genip 10.0.0.0/24 > ips.txt
$ dns-grind.pl -f ips.txt PTR
10.0.0.10 www.example.com
10.0.0.99 manager.example.com
```

[1] This list of top level domains from http://www.iana.org/root-whois/index.html, http://www.icann.org/registries/listing.html and http://www.nominet.org.uk/registrants/legal/rules/

[2] This convention is taken from the way hydra takes its is usernames and password.

[3] http://www.bindshell.net/tools/genip/