ngrep

Abstract

ngrep brings basic power of grep to network traffic. We need ngrep because PCAPs are a binary structure. Although piping the output of strings to grep can work, grep will still not be PCAP aware in the way ngrep is. Rather than simply showing, for example, a line within the PCAP that includes a string match, ngrep understands which packet contains this string and will indicate high level packet details such as source and destination IP addresses, port numbers, and TCP flags.

> *ngrep strives to provide most of GNU grep's common features, applying them to the network layer. ngrep is a pcap-aware tool that will allow you to specify extended regular expressions to match against data payloads of packets. It currently recognizes TCP, UDP and ICMP across Ethernet, PPP, SLIP, FDDI and null interfaces, and understands bpf filter logic in the same fashion as more common packet sniffing tools.*

Source: ngrep man page

```
$ man ngrep
```

Where to Acquire

Available in standard Linux repositories as well as directly from ngrep.sourceforge.net

Examples/Use Case

Note: Some of the examples below presume files and paths that might not match your particular system and tool installation.

Quietly search /pcaps/angler-java.pcap for user agents via the string 'User-Agent'

```
$ ngrep -qI /pcaps/angler-java.pcap "User-Agent"
```

Note: The -q switch for "Quiet" will generally be used in most circumstances and will not be asked called out in subsequent examples.

Search /pcaps/angler-java.pcap for 'User-Agent' in TCP segments destined for port 80

```
$ ngrep -qI /pcaps/angler-java.pcap "User-Agent" "tcp and dst port 80"
```

Note: In the above command line "tcp and dst port 80" is an example of using BPF (Berkely

Packet Filter) expressions. Review the man page for pcap-filter for additional information.

Search /pcaps/styx.pcap for 'User-Agent' in TCP segments destined for traffic *other than* port 80

```
$ ngrep -q -I /pcaps/styx.pcap "User-Agent" "tcp and not dst port 80"
```

Search /pcaps/angler-java.pcap for 'User-Agent' in TCP segments destined for port 80. Have the output honor any linefeeds encountered and wrap text.

```
$ ngrep -q -W byline -I /pcaps/angler-java.pcap "User-Agent" "tcp and dst port 80"
```

Note: To make the output of some traffic, most notably HTTP, easier to read, the -W switch can be set to byline.

Search /pcaps/angler-java.pcap for traffic with the ACK FIN and PUSH flags set

```
$ ngrep -q -I /pcaps/angler-java.pcap "" "tcp[tcpflags]==(tcp-ack|tcp-fin|tcp-push)"
```

Note: When using BPF expressions, a search string, even if blank "" as above, is expected. Otherwise ngrep will treat the BPF expression itself as the regex search pattern.

Sniff on the eth0 interface and look for the string 'HoneyToken' without the quotes being passed in clear text

```
$ sudo ngrep -q -d eth0 "HoneyToken"
```

Note: When using ngrep to bind to an interface, superuser privileges will generally be required.

Sniff on the eth0 interface and look for the string 'HoneyToken' without the quotes being passed in clear text. When found, kill the connection by sending a spoofed TCP RST

```
$ sudo ngrep -q -d eth0 "HoneyToken" -K1
```

Additional Info

Like tcpdump, ngrep allows the use of BPF (Berkely Packet Filter) expressions. See the pcap-filter man page for additional details on bpf filters.

```
$ man pcap-filter
```

A printable PDF version of this cheatsheet is available here:

ngrep

Cheat Sheet Version

Version 1.0