Verifying Packet Level Response v0.11

Table of Contents

1 - Summary

The guide explains how to verify packet level responses. The tools and methods used in this guide have proven helpful to me when trying to verify packet level responses on a system. This paper is aimed toward people that perform penetration tests and vulnerability assessments. Keep in mind this guide shows only one method of verifying packet level responses. For security-minded individuals out there reading this guide, feel free to contact me with your preferred method(s) of verifying packet level responses.

2 – TCP sequence number predictability

I will first try and identify the predictability of the TCP sequence numbers. `Hping` will be used to collect TCP sequence numbers generated by the target host. `Hping` works by sending x number of SYN packets to the open port specified, and then prints out the TCP sequence numbers generated by the target host along with the differences. The first column below shows the sequence number while the second column shows the difference between the current and last sequence number. From the output you can determine the predictability of the TCP sequence numbers. In this case the sequence numbers are truly random.

```
# hping -c 5 -n -p 22 -Q -S 10.10.1.10
HPING 10.10.1.10 (fxp0 10.10.1.10): S set, 40 headers + 0 data bytes
 803495237 +803495237
 471937929 +3963409987
1381238037 +909300108
1965887675 +584649638
 615916430 +2944996050

--- 10.10.1.10 hping statistic ---
5 packets tramitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.8/0.8/1.0 ms
```

3 – IP identification number predictability

`Hping` will be used to identify the predictability of the IP identification numbers. `Hping` works by sending x number of SYN packets to the open port specified, and then prints out the responding packets generated by the target host. The id value represents the identification number from each packet. From the output you can determine the predictability of the IP identification numbers. In this case the identification numbers are randomized.

```
# hping -c 5 -n -p 22 -S 10.10.1.10
HPING 10.10.1.10 (fxp0 10.10.1.10): S set, 40 headers + 0 data bytes
len=46 ip=10.10.1.10 ttl=64 DF id=49308 sport=22 flags=SA seq=0 win=65535 rtt=1.0 ms
len=46 ip=10.10.1.10 ttl=64 DF id=55816 sport=22 flags=SA seq=1 win=65535 rtt=0.8 ms
len=46 ip=10.10.1.10 ttl=64 DF id=39757 sport=22 flags=SA seq=2 win=65535 rtt=0.8 ms
len=46 ip=10.10.1.10 ttl=64 DF id=32864 sport=22 flags=SA seq=3 win=65535 rtt=0.8 ms
len=46 ip=10.10.1.10 ttl=64 DF id=48250 sport=22 flags=SA seq=4 win=65535 rtt=0.8 ms

--- 10.10.1.10 hping statistic ---
5 packets tramitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.8/0.8/1.0 ms
```

4 – TCP timestamp sampling/System up-time

`Hping` will be used to identify the TCP timestamp update frequency, and the system up-time. `Hping` works by sending x number of SYN packets to the open port specified, and then prints out the responding packets generated by the target host. These responding packets also include the timestamp information. `Hping` tries to determine the system up-time from these packets. In this case the timestamp values seem to increase by two every second, and the system up-time seems to misleading.

```
# hping -c 5 -n -p 22 --tcp-timestamp -S 10.10.1.10
HPING 10.10.1.10 (fxp0 10.10.1.10): S set, 40 headers + 0 data bytes
len=56 ip=10.10.1.10 ttl=64 DF id=62962 sport=22 flags=SA seq=0 win=65535 rtt=1.0 ms
  TCP timestamp: tcpts=1003409716

len=56 ip=10.10.1.10 ttl=64 DF id=52145 sport=22 flags=SA seq=1 win=65535 rtt=0.8 ms
  TCP timestamp: tcpts=1003409718
  HZ seems hz=2
  System uptime seems: 5806 days, 18 hours, 27 minutes, 39 seconds

len=56 ip=10.10.1.10 ttl=64 DF id=59657 sport=22 flags=SA seq=2 win=65535 rtt=0.8 ms
  TCP timestamp: tcpts=1003409720
  HZ seems hz=2
  System uptime seems: 5806 days, 18 hours, 27 minutes, 40 seconds

len=56 ip=10.10.1.10 ttl=64 DF id=40769 sport=22 flags=SA seq=3 win=65535 rtt=0.8 ms
  TCP timestamp: tcpts=1003409722
  HZ seems hz=2
  System uptime seems: 5806 days, 18 hours, 27 minutes, 41 seconds

len=56 ip=10.10.1.10 ttl=64 DF id=54206 sport=22 flags=SA seq=4 win=65535 rtt=0.8 ms
  TCP timestamp: tcpts=1003409725
  HZ seems hz=2
  System uptime seems: 5806 days, 18 hours, 27 minutes, 42 seconds


--- 10.10.1.10 hping statistic ---
5 packets tramitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.8/0.8/1.0 ms
```

5 – Overall verification

The last step is to perform an overall verification of the packet level responses. `Nmap` will be used to do this because of the fact that the above tests will all be performed in the os detection function. This is done by sending SYN packets to the open port specified, and analyzing the responses. The output below shows the level of predictability for TCP sequence numbers, IP identification numbers, and if available the system up-time.

```
# nmap -sS -PS80 -O -v -p 22,80 -n 10.10.1.10

Starting nmap 3.46 ( http://www.insecure.org/nmap/ ) at 2003-10-01 10:20 CDT
Host 10.10.1.10 appears to be up ... good.
Initiating SYN Stealth Scan against 10.10.1.10 at 10:20
Adding open port 22/tcp
The SYN Stealth Scan took 0 seconds to scan 2 ports.
For OSScan assuming that port 22 is open and port 80 is closed and neither are firewalled
Interesting ports on 10.10.1.10:
PORT   STATE  SERVICE
22/tcp open   ssh
80/tcp closed http
Device type: general purpose
```

```
Running: OpenBSD 3.X
OS details: OpenBSD 3.0 or 3.3
TCP Sequence Prediction: Class=truly random
                         Difficulty=9999999 (Good luck!)
IPID Sequence Generation: Randomized

Nmap run completed -- 1 IP address (1 host up) scanned in 8.247 seconds
```

6 – Links

Hping - http://www.hping.org/

Nmap - http://www.insecure.org/nmap/