

## Idle Scan Step by Step




Fundamentally, an idle scan consists of three steps that are repeated for each port:

1. Probe the zombie's IP ID and record it.
2. Forge a SYN packet from the zombie and send it to the desired port on the target.  
Depending on the port state, the target's reaction may or may not cause the zombie's IP ID to be incremented.
3. Probe the zombie's IP ID again. The target port state is then determined by comparing this new IP ID with the one recorded in step 1.

After this process, the zombie's IP ID should have increased by either one or two. An increase of one indicates that the zombie hasn't sent out any packets, except for its reply to the attacker's probe. This lack of sent packets means that the port is not open (the target must have sent the zombie either a RST packet, which was ignored, or nothing at all). An increase of two indicates that the zombie sent out a packet between the two probes. This extra packet usually means that the port is open (the target presumably sent the zombie a SYN/ACK packet in response to the forged SYN, which induced a RST packet from the zombie). Increases larger than two usually signify a bad zombie host. It might not have predictable IP ID numbers, or might be engaged in communication unrelated to the idle scan.

Even though what happens with a closed port is slightly different from what happens with a filtered port, the attacker measures the same result in both cases, namely, an IP ID increase of 1. Therefore it is not possible for the idle scan to distinguish between closed and filtered ports. When Nmap records an IP ID increase of 1 it marks the port `closed|filtered`.

For those wanting more detail, the following three diagrams show exactly what happens in the three cases of an open, closed, and filtered port. The actors in each are:

 the attacker,  the zombie, and  the target.

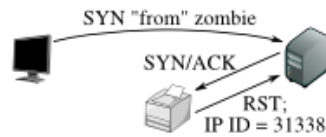
**Figure 5.6. Idle scan of an open port**

Step 1: Probe the zombie's IP ID.



The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID.

Step 2: Forge a SYN packet from the zombie.



The target sends a SYN/ACK in response to the SYN that appears to come from the zombie. The zombie, not expecting it, sends back a RST, incrementing its IP ID in the process.

Step 3: Probe the zombie's IP ID again.



The zombie's IP ID has increased by 2 since step 1, so the port is open!

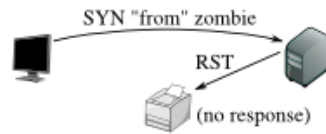
**Figure 5.7. Idle scan of a closed port**

Step 1: Probe the zombie's IP ID.



The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID. This step is always the same.

Step 2: Forge a SYN packet from the zombie.



The target sends a RST (the port is closed) in response to the SYN that appears to come from the zombie. The zombie ignores the unsolicited RST, leaving its IP ID unchanged.

Step 3: Probe the zombie's IP ID again.



The zombie's IP ID has increased by only 1 since step 1, so the port is not open.

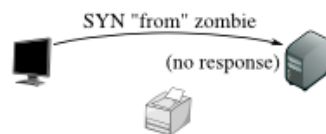
**Figure 5.8. Idle scan of a filtered port**

Step 1: Probe the zombie's IP ID.



Just as in the other two cases, the attacker sends a SYN/ACK to the zombie. The zombie discloses its IP ID.

Step 2: Forge a SYN packet from the zombie.



The target, obstinately filtering its port, ignores the SYN that appears to come from the zombie. The zombie, unaware that anything has happened, does not increment its IP ID.

Step 3: Probe the zombie's IP ID again.



The zombie's IP ID has increased by only 1 since step 1, so the port is not open. From the attacker's point of view this filtered port is indistinguishable from a closed port.

Idle scan is the ultimate stealth scan. Nmap offers decoy scanning (-D) to help users shield their identity, but that (unlike idle scan) still requires an attacker to send some packets to the target from his real IP address in order to get scan results back. One upshot of idle scan is that intrusion detection systems will generally send alerts claiming that the zombie machine has launched a

scan against them. So it can be used to frame some other party for a scan. Keep this possibility in mind when reading alerts from your IDS.

A unique advantage of idle scan is that it can be used to defeat certain packet filtering firewalls and routers. IP source address filtering is a common (though weak) security mechanism for limiting machines that may connect to a sensitive host or network. For example, a company database server might only allow connections from the public web server that accesses it. Or a home user might only allow SSH (interactive login) connections from his work machines.

A more disturbing scenario occurs when some company bigwig demands that network administrators open a firewall hole so he can access internal network resources from his home IP address. This can happen when executives are unwilling or unable to use secure VPN alternatives.

Idle scanning can sometimes be used to map out these trust relationships. The key factor is that idle scan results list open ports from the zombie host's perspective. A normal scan against the aforementioned database server might show no ports open, but performing an idle scan while using the web server's IP as the zombie could expose the trust relationship by showing the database-related service ports as open.

Mapping out these trust relationships can be very useful to attackers for prioritizing targets. The web server discussed above may seem mundane to an attacker until she notices its special database access.

A disadvantage to idle scanning is that it takes far longer than most other scan types. Despite the optimized algorithms described in [the section called “Idle Scan Implementation Algorithms”](#), A 15-second SYN scan could take 15 minutes or more as an idle scan. Another issue is that you must be able to spoof packets as if they are coming from the zombie and have them reach the target machine. Many ISPs (particularly dialup and residential broadband providers) now implement egress filtering to prevent this sort of packet spoofing. Higher end providers (such as colocation and T1 services) are much less likely to do this. If this filtering is in effect, Nmap will print a quick error message for every zombie you try. If changing ISPs is not an option, you might try using another IP on the same ISP network. Sometimes the filtering only blocks spoofing of IP addresses that are *outside* the range used by customers. Another challenge with idle scan is that you must find a working zombie host, as described in the next section.

## **Finding a Working Idle Scan Zombie Host**

The first step in executing an IP ID idle scan is to find an appropriate zombie. It needs to assign IP ID packets incrementally on a global (rather than per-host it communicates with) basis. It should be idle (hence the scan name), as extraneous traffic will bump up its IP ID sequence, confusing the scan logic. The lower the latency between the attacker and the zombie, and between the zombie and the target, the faster the scan will proceed.

When an idle scan is attempted, Nmap tests the proposed zombie and reports any problems with it. If one doesn't work, try another. Enough Internet hosts are vulnerable that zombie candidates

aren't hard to find. Since the hosts need to be idle, choosing a well-known host such as `www.yahoo.com` or `google.com` will almost never work.

A common approach is to simply execute a Nmap ping scan of some network. You could use Nmap's random IP selection mode (`-iR`), but that is likely to result in far away zombies with substantial latency. Choosing a network near your source address, or near the target, produces better results. You can try an idle scan using each available host from the ping scan results until you find one that works. As usual, it is best to ask permission before using someone's machines for unexpected purposes such as idle scanning.

We didn't just choose a printer icon to represent a zombie in our illustrations to be funny—simple network devices often make great zombies because they are commonly both underused (idle) and built with simple network stacks which are vulnerable to IP ID traffic detection.

Performing a port scan and OS identification (`-O`) on the zombie candidate network rather than just a ping scan helps in selecting a good zombie. As long as verbose mode (`-v`) is enabled, OS detection will usually determine the IP ID sequence generation method and print a line such as “IP ID Sequence Generation: Incremental”. If the type is given as `Incremental` or `Broken little-endian incremental`, the machine is a good zombie candidate. That is still no guarantee that it will work, as Solaris and some other systems create a new IP ID sequence for each host they communicate with. The host could also be too busy. OS detection and the open port list can also help in identifying systems that are likely to be idle.

Another approach to identifying zombie candidates is to run the `ipidseq` NSE script against a host. This script probes a host to classify its IP ID generation method, then prints the IP ID classification much like the OS detection does. Like most NSE scripts, `ipidseq.nse` can be run against many hosts in parallel, making it another good choice when scanning entire networks looking for suitable hosts.

While identifying a suitable zombie takes some initial work, you can keep re-using the good ones.

## Executing an Idle Scan

Once a suitable zombie has been found, performing a scan is easy. Simply specify the zombie hostname to the `-sI` option and Nmap does the rest. [Example 5.19](#) shows an example of Eret scanning the Recording Industry Association of America by bouncing an idle scan off an Adobe machine named Kiosk.

### Example 5.19. An idle scan against the RIAA

```
# nmap -Pn -p- -sI kiosk.adobe.com www.riaa.com
```

```
Starting Nmap ( http://nmap.org )
Idlescan using zombie kiosk.adobe.com (192.150.13.111:80);
Class: Incremental
```

```
Nmap scan report for 208.225.90.120
(The 65522 ports scanned but not shown below are in state:
closed)
```

Port	State	Service
21/tcp	open	ftp
25/tcp	open	smtp
80/tcp	open	http
111/tcp	open	sunrpc
135/tcp	open	loc-srv
443/tcp	open	https
1027/tcp	open	IIS
1030/tcp	open	iad1
2306/tcp	open	unknown
5631/tcp	open	pcanywheredata
7937/tcp	open	unknown
7938/tcp	open	unknown
36890/tcp	open	unknown

```
Nmap done: 1 IP address (1 host up) scanned in 2594.47 seconds
```

From the scan above, we learn that the RIAA is not very security conscious (note the open PC Anywhere, portmapper, and Legato nsrexec ports). Since they apparently have no firewall, it is unlikely that they have an IDS. But if they do, it will show kiosk.adobe.com as the scan culprit. The `-Pn` option prevents Nmap from sending an initial ping packet to the RIAA machine. That would have disclosed Ereet's true address. The scan took a long time because `-p-` was specified to scan all 65K ports. Don't try to use kiosk for your scans, as it has already been removed.

By default, Nmap forges probes to the target from the source port 80 of the zombie. You can choose a different port by appending a colon and port number to the zombie name (e.g. `-sI kiosk.adobe.com:113`). The chosen port must not be filtered from the attacker or the target. A SYN scan of the zombie should show the port in the `open` or `closed` state.

## Idle Scan Implementation Algorithms

While [the section called "Idle Scan Step by Step"](#) describes idle scan at the fundamental level, the Nmap implementation is far more complex. Key differences are parallelism for quick execution and redundancy to reduce false positives.

Parallelizing idle scan is trickier than with other scan techniques due to indirect method of deducing port states. If Nmap sends probes to many ports on the target and then checks the new IP ID value of the zombie, the number of IP ID increments will expose how many target ports are open, but not which ones. This isn't actually a major problem, as the vast majority of ports in a large scan will be `closed` | `filtered`. Since only open ports cause the IP ID value to increment, Nmap will see no intervening increments and can mark the whole group of ports as `closed` | `filtered`. Nmap can scan groups of up to 100 ports in parallel. If Nmap probes a group then finds that the zombie IP ID has increased  $<N>$  times, there must be  $<N>$  open ports

among that group. Nmap then finds the open ports with a binary search. It splits the group into two and separately sends probes to each. If a subgroup shows zero open ports, that group's ports are all marked `closed` | `filtered`. If a subgroup shows one or more open ports, it is divided again and the process continues until those ports are identified. While this technique adds complexity, it can reduce scan times by an order of magnitude over scanning just one port at a time.

Reliability is another major idle scanning concern. If the zombie host sends packets to any unrelated machines during the scan, its IP ID increments. This causes Nmap to think it has found an open port. Fortunately, parallel scanning helps here too. If Nmap scans 100 ports in a group and the IP ID increase signals two open ports, Nmap splits the group into two fifty-port subgroups. When Nmap does an IP ID scan on both subgroups, the total zombie IP ID increase better be two again! Otherwise, Nmap will detect the inconsistency and rescan the groups. It also modifies group size and scan timing based on the detected reliability rate of the zombie. If Nmap detects too many inconsistent results, it will quit and ask the user to provide a better zombie.

Sometimes a packet trace is the best way to understand complex algorithms and techniques such as these. Once again, the Nmap `--packet-trace` makes these trivial to produce when desired. The remainder of this section provides an annotated packet trace of an actual seven port idle scan. The IP addresses have been changed to `Attacker`, `Zombie`, and `Target` and some irrelevant aspects of the trace lines (such as TCP window size) have been removed for clarity.

```
Attacker# nmap -sI Zombie -Pn -p20-25,110 -r --packet-trace -v
Target
Starting Nmap ( http://nmap.org )
```

`-Pn` is necessary for stealth, otherwise ping packets would be sent to the target from Attacker's real address. Version scanning would also expose the true address, and so `-sV` is *not* specified. The `-r` option (turns off port randomization) is only used to make this example easier to follow.

Nmap firsts tests Zombie's IP ID sequence generation by sending six SYN/ACK packets to it and analyzing the responses. This helps Nmap immediately weed out bad zombies. It is also necessary because some systems (usually Microsoft Windows machines, though not all Windows boxes do this) increment the IP ID by 256 for each packet sent rather than by one. This happens on little-endian machines when they don't convert the IP ID to network byte order (big-endian). Nmap uses these initial probes to detect and work around this problem.

```
SENT (0.0060s) TCP Attacker:51824 > Zombie:80 SA id=35996
SENT (0.0900s) TCP Attacker:51825 > Zombie:80 SA id=25914
SENT (0.1800s) TCP Attacker:51826 > Zombie:80 SA id=39591
RCVD (0.1550s) TCP Zombie:80 > Attacker:51824 R id=15669
SENT (0.2700s) TCP Attacker:51827 > Zombie:80 SA id=43604
RCVD (0.2380s) TCP Zombie:80 > Attacker:51825 R id=15670
SENT (0.3600s) TCP Attacker:51828 > Zombie:80 SA id=34186
RCVD (0.3280s) TCP Zombie:80 > Attacker:51826 R id=15671
SENT (0.4510s) TCP Attacker:51829 > Zombie:80 SA id=27949
RCVD (0.4190s) TCP Zombie:80 > Attacker:51827 R id=15672
```

```
RCVD (0.5090s) TCP Zombie:80 > Attacker:51828 R id=15673
RCVD (0.5990s) TCP Zombie:80 > Attacker:51829 R id=15674
Idlescan using zombie Zombie (Zombie:80); Class: Incremental
```

This test demonstrates that the zombie is working fine. Every IP ID was an increase of one over the previous one. So the system appears to be idle and vulnerable to IP ID traffic detection. These promising results are still subject to the next test, in which Nmap spoofs four packets to Zombie as if they are coming from Target. Then it probes the zombie to ensure that the IP ID increased. If it hasn't, then it is likely that either the attacker's ISP is blocking the spoofed packets or the zombie uses a separate IP ID sequence counter for each host it communicates with. Both are common occurrences, so Nmap always performs this test. The last-known Zombie IP ID was 15674, as shown above.

```
SENT (0.5990s) TCP Target:51823 > Zombie:80 SA id=1390
SENT (0.6510s) TCP Target:51823 > Zombie:80 SA id=24025
SENT (0.7110s) TCP Target:51823 > Zombie:80 SA id=15046
SENT (0.7710s) TCP Target:51823 > Zombie:80 SA id=48658
SENT (1.0800s) TCP Attacker:51987 > Zombie:80 SA id=27659
RCVD (1.2290s) TCP Zombie:80 > Attacker:51987 R id=15679
```

The four spoofed packets coupled with the probe from Attacker caused the Zombie to increase its IP ID from 15674 to 15679. Perfect! Now the real scanning begins. Remember that 15679 is the latest Zombie IP ID.

```
Initiating Idlescan against Target
```

```
SENT (1.2290s) TCP Zombie:80 > Target:20 S id=13200
SENT (1.2290s) TCP Zombie:80 > Target:21 S id=3737
SENT (1.2290s) TCP Zombie:80 > Target:22 S id=65290
SENT (1.2290s) TCP Zombie:80 > Target:23 S id=10516
SENT (1.4610s) TCP Attacker:52050 > Zombie:80 SA id=33202
RCVD (1.6090s) TCP Zombie:80 > Attacker:52050 R id=15680
```

Nmap probes ports 20-23. Then it probes Zombie and finds that the new IP ID is 15680, only one higher than the previous value of 15679. There were no IP ID increments in between those two known packets, meaning ports 20-23 are probably closed|filtered. It is also possible that a SYN/ACK from a Target port has simply not arrived yet. In that case, Zombie has not responded with a RST and thus its IP ID has not incremented. To ensure accuracy, Nmap will try these ports again later.

```
SENT (1.8510s) TCP Attacker:51986 > Zombie:80 SA id=49278
RCVD (1.9990s) TCP Zombie:80 > Attacker:51986 R id=15681
```

Nmap probes again because four tenths of a second has gone by since the last probe it sent. The Zombie (if not truly idle) could have communicated with other hosts during this period, which would cause inaccuracies later if not detected here. Fortunately, that has not happened: the next IP ID is 15681 as expected.

```
SENT (2.0000s) TCP Zombie:80 > Target:24 S id=23928
SENT (2.0000s) TCP Zombie:80 > Target:25 S id=50425
SENT (2.0000s) TCP Zombie:80 > Target:110 S id=14207
```

```
SENT (2.2300s) TCP Attacker:52026 > Zombie:80 SA id=26941
RCVD (2.3800s) TCP Zombie:80 > Attacker:52026 R id=15684
```

Nmap probes ports 24, 25, and 110 then queries the Zombie IP ID. It has jumped from 15681 to 15684. It skipped 15682 and 15683, meaning that two of those three ports are likely open. Nmap cannot tell which two are open, and it could also be a false positive. So Nmap drills down deeper, dividing the scan into subgroups.

```
SENT (2.6210s) TCP Attacker:51867 > Zombie:80 SA id=18869
RCVD (2.7690s) TCP Zombie:80 > Attacker:51867 R id=15685
SENT (2.7690s) TCP Zombie:80 > Target:24 S id=30023
SENT (2.7690s) TCP Zombie:80 > Target:25 S id=47253
SENT (3.0000s) TCP Attacker:51979 > Zombie:80 SA id=12077
RCVD (3.1480s) TCP Zombie:80 > Attacker:51979 R id=15687
```

The first subgroup is ports 24 and 25. The IP ID jumps from 15685 to 15687, meaning that one of these two ports is most likely open. Nmap tries the divide and conquer approach again, probing each port separately.

```
SENT (3.3910s) TCP Attacker:51826 > Zombie:80 SA id=32515
RCVD (3.5390s) TCP Zombie:80 > Attacker:51826 R id=15688
SENT (3.5390s) TCP Zombie:80 > Target:24 S id=47868
SENT (3.7710s) TCP Attacker:52012 > Zombie:80 SA id=14042
RCVD (3.9190s) TCP Zombie:80 > Attacker:52012 R id=15689
```

A port 24 probe shows no jump in the IP ID. So that port is not open. From the results so far, Nmap has tentatively determined:

- Ports 20-23 are `closed|filtered`
- Two of the ports 24, 25, and 110 are open
- One of the ports 24 and 25 are open
- Port 24 is `closed|filtered`

Stare at this puzzle long enough and you'll find only one solution: ports 25 and 110 are open while the other five are `closed|filtered`. Using this logic, Nmap could cease scanning and print results now. It used to do so, but that produced too many false positive open ports when the Zombie wasn't truly idle. So Nmap continues scanning to verify its results:

```
SENT (4.1600s) TCP Attacker:51858 > Zombie:80 SA id=6225
RCVD (4.3080s) TCP Zombie:80 > Attacker:51858 R id=15690
SENT (4.3080s) TCP Zombie:80 > Target:25 S id=35713
SENT (4.5410s) TCP Attacker:51856 > Zombie:80 SA id=28118
RCVD (4.6890s) TCP Zombie:80 > Attacker:51856 R id=15692
```



```
Discovered open port 25/tcp on Target
SENT (4.6900s) TCP Zombie:80 > Target:110 S id=9943
SENT (4.9210s) TCP Attacker:51836 > Zombie:80 SA id=62254
RCVD (5.0690s) TCP Zombie:80 > Attacker:51836 R id=15694
Discovered open port 110/tcp on Target
```

Probes of ports 25 and 110 show that they are open, as we deduced previously.

```
SENT (5.0690s) TCP Zombie:80 > Target:20 S id=8168
SENT (5.0690s) TCP Zombie:80 > Target:21 S id=36717
SENT (5.0690s) TCP Zombie:80 > Target:22 S id=4063
SENT (5.0690s) TCP Zombie:80 > Target:23 S id=54771
SENT (5.3200s) TCP Attacker:51962 > Zombie:80 SA id=38763
RCVD (5.4690s) TCP Zombie:80 > Attacker:51962 R id=15695
SENT (5.7910s) TCP Attacker:51887 > Zombie:80 SA id=61034
RCVD (5.9390s) TCP Zombie:80 > Attacker:51887 R id=15696
```

Just to be sure, Nmap tries ports 20-23 again. A Zombie IP ID query shows no sequence jump. On the off chance that a SYN/ACK from Target to Zombie came in late, Nmap tries another IP ID query. This again shows no open ports. Nmap is now sufficiently confident with its results to print them.

```
The Idlescan took 5 seconds to scan 7 ports.
Nmap scan report for Target
PORT      STATE      SERVICE
20/tcp    closed|filtered ftp-data
21/tcp    closed|filtered ftp
22/tcp    closed|filtered ssh
23/tcp    closed|filtered telnet
24/tcp    closed|filtered priv-mail
25/tcp    open      smtp
110/tcp   open      pop3
```

```
Nmap finished: 1 IP address (1 host up) scanned in 5.949 seconds
For complete details on the Nmap idle scan implementation, read idle_scan.cc from the Nmap source code distribution.
```

<https://nmap.org/book/idlescan.html>

---

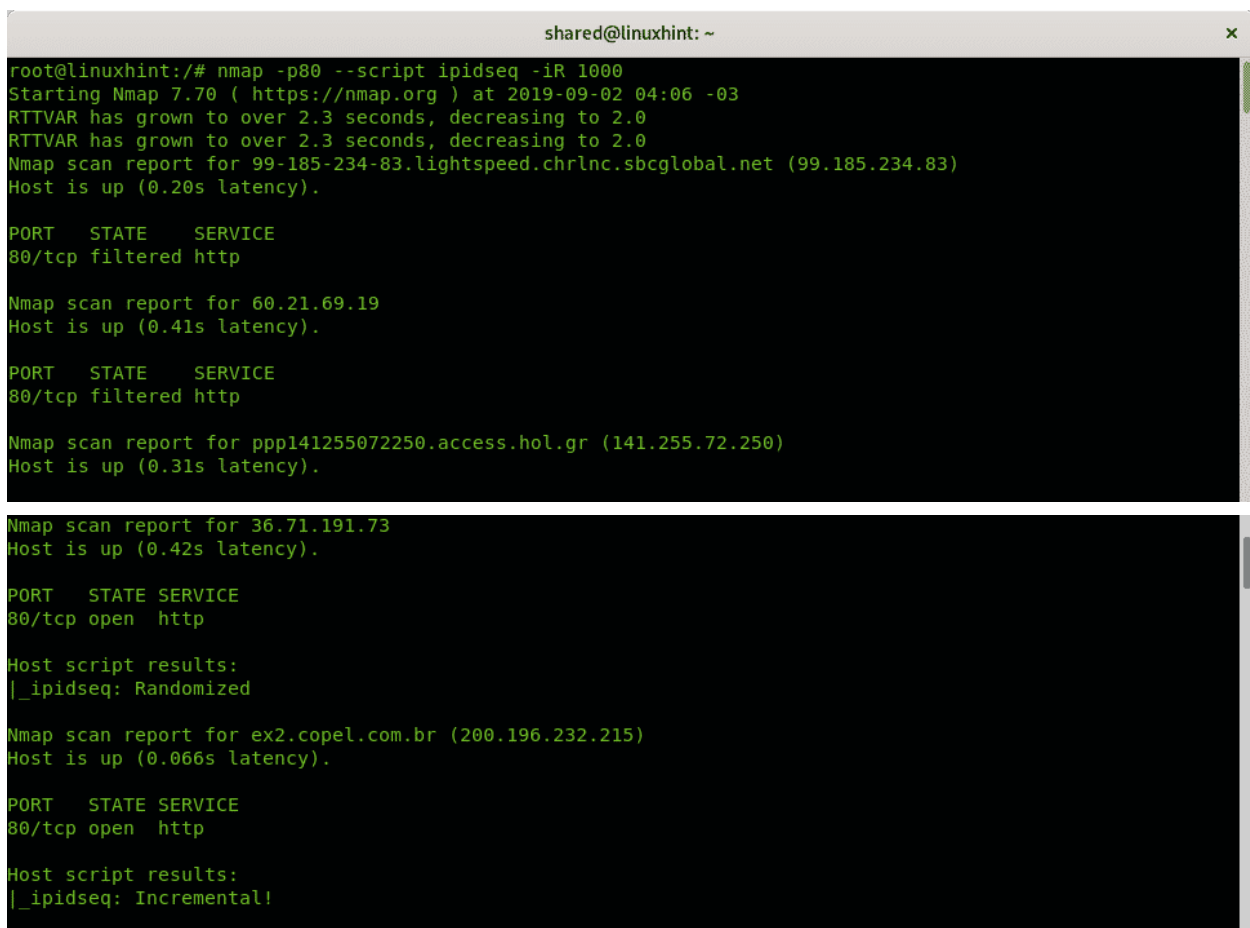
## Finding a zombie device

Nmap NSE (Nmap Scripting Engine) provides the script `IPIDSEQ` to detect vulnerable zombie devices. In the following example the script is used to scan port 80 of random 1000 targets to look for vulnerable hosts,

*vulnerable hosts are classified* as **Incremental** or **little-endian incremental**. Additional examples of NSE use, despite unrelated to Idle Scan are described and shown at [How to scan for services and vulnerabilities with Nmap](#) and [Using nmap scripts: Nmap banner grab](#).

IPIDSEQ example to randomly find zombie candidates:

“nmap -p80 --script ipidseq -iR 1000” ,, or “nmap --script ipidseq 192.168.1.0-255”



```
shared@linuxhint: ~
root@linuxhint:/# nmap -p80 --script ipidseq -iR 1000
Starting Nmap 7.70 ( https://nmap.org ) at 2019-09-02 04:06 -03
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
Nmap scan report for 99-185-234-83.lightspeed.chrlnc.sbcglobal.net (99.185.234.83)
Host is up (0.20s latency).

PORT      STATE      SERVICE
80/tcp    filtered  http

Nmap scan report for 60.21.69.19
Host is up (0.41s latency).

PORT      STATE      SERVICE
80/tcp    filtered  http

Nmap scan report for ppp141255072250.access.hol.gr (141.255.72.250)
Host is up (0.31s latency).

Nmap scan report for 36.71.191.73
Host is up (0.42s latency).

PORT      STATE      SERVICE
80/tcp    open      http

Host script results:
|_ipidseq: Randomized

Nmap scan report for ex2.copel.com.br (200.196.232.215)
Host is up (0.066s latency).

PORT      STATE      SERVICE
80/tcp    open      http

Host script results:
|_ipidseq: Incremental!
```

```
Nmap scan report for mtocontrata.scm.uam.es (150.244.32.53)
Host is up (0.29s latency).
```

```
PORT      STATE SERVICE
80/tcp    open  http
```

```
Host script results:
|_ipidseq: Incremental!
```

```
Nmap scan report for 74.85.211.10
Host is up (0.18s latency).
```

```
PORT      STATE SERVICE
80/tcp    filtered http
```

```
Nmap scan report for p5B22E140.dip0.t-ipconnect.de (91.34.225.64)
Host is up (0.27s latency).
```

```
PORT      STATE SERVICE
80/tcp    closed http
```

```
Host script results:
|_ipidseq: Incremental!
```

```
Nmap scan report for adsl-ull-147-38.51-151.wind.it (151.51.38.147)
Host is up (0.29s latency).
```

```
PORT      STATE SERVICE
```

```
Nmap scan report for 175.212.48.77
Host is up (0.34s latency).
```

```
PORT      STATE SERVICE
80/tcp    closed http
```

```
Host script results:
|_ipidseq: Incremental!
```

```
Nmap scan report for 95.59.31.121
Host is up (0.33s latency).
```

```
PORT      STATE SERVICE
80/tcp    filtered http
```

```
Nmap scan report for 2.87.27.93.rev.sfr.net (93.27.87.2)
Host is up (0.29s latency).
```

```
PORT      STATE SERVICE
```

```
Host is up (0.27s latency).
```

```
PORT      STATE SERVICE
80/tcp    closed http
```

```
Host script results:
|_ipidseq: Incremental!
```

```
Nmap scan report for 43.229.244.38
Host is up (0.29s latency).
```

```
PORT      STATE SERVICE
80/tcp    open  http
```

```
Host script results:
|_ipidseq: Incremental!
```

```
Nmap done: 1001 IP addresses (126 hosts up) scanned in 188.18 seconds
root@linuxhint:/#
```

As you can see several vulnerable zombie candidate hosts were found **BUT** they are all false positive. The most difficult step when carrying out an Idle scan is to find a vulnerable zombie device, it is difficult due to many reasons:

- Many ISP block this type of scan.
- Most of Operating Systems assign IP ID randomly
- Well configured firewalls and honeypots may return false positive.

In such cases when trying to execute the Idle scan you'll get the following error:

***“...cannot be used because it has not returned any of our probes – perhaps it is down or firewalled.  
QUITTING!”***

If you are lucky in this step you will find an old Windows system, an old IP camera system or an old network printer, this last example is recommended by the Nmap book.

When looking for vulnerable zombies you may want to exceed Nmap and implement additional tools like **Shodan** and **faster scanners**. You can also run random scans detecting versions to find a possible vulnerable system.

[For more information, please check this —>https://linuxhint.com/nmap\\_idle\\_scan\\_tutorial/](https://linuxhint.com/nmap_idle_scan_tutorial/)

## Fight Against Stealth Idle Scan

### **How can I protect my IP address?**

You are typically assigned a dynamic (changes periodically) IP address by your ISP. However, you can sign up for a static (never changes) address if you wanted to run a web server from your house. There is some debate as to which is safer, though it is arguably more secure to have a dynamic IP address because “there isn’t a constant target for the attacker,” says Steven Burn, Lead Malware Intelligence Analyst at Malwarebytes.

It all boils down to the steps you take to secure your IP address. Here’s how you can keep your IP address out of the hands of criminals.

**1. Use a Virtual Private Network (VPN):** This protects your data online by the use of encryption and proxy tunneling. It hides your IP address and redirects your traffic through a separate server, making it much safer for you online. VPN services are without question the

best practice for hiding your IP address, says Burn. They can be found online with monthly service charges, however, the price might not be worth it for every home user. There are free VPN services out there, but don't expect them to be fast enough for any streaming or gaming.

In addition, you could run a proxy, which acts as an additional hub through which Internet requests are processed, all while hiding your IP address. It can determine legitimate over non-legitimate requests.

**2. Update your router and firewall rules:** Your router forwards data between networks, and your firewall prevents unauthorized access. Make sure you change the administrative password on your router, since default passwords are frequently used by attackers to break into your network. Each default password provided by your ISP is the same and can be easily searched online. Also, set your firewall rules to not allow any ping requests from the Internet. This makes sure unauthorized visitors won't get through.

**3. Change privacy settings on instant messaging applications:** Only allow direct connections from contacts and don't accept calls or messages from people you don't know. Changing your settings to private makes it harder to find your IP address because people who don't know you cannot connect with you.

**4. Update your antivirus solution and add security layers:** Making sure you are caught up on all of your security software's updates ensures you're protected from threats. Adding additional security on top of your antivirus further protects you. For instance, [Malwarebytes Anti-Malware](#) blocks malicious URLs originating from phishing emails in addition to blocking bad sites you might encounter.

Protecting your IP address is one aspect of protecting your identity. Securing it through these steps is one more way to stay safe

against the wide variety of attack vectors cybercriminals are using today.

<https://blog.malwarebytes.com/101/2016/07/how-to-protect-your-ip-address/>

## Defenses (in another word)

Fortunately, there are several defenses which can be deployed to prevent most IPID-related attacks:

### **Network Administrators:**

- Configure your firewalls/border routers to deny incoming packets with bogus source addresses (eg. that appear to come from machines within your network, reserved IPs like 10.X.X.X or 192.168.X.X, localhost IPs 127.X.X.X, etc. Any good firewall guide should provide more detailed guidance on these essential rules.
- Stateful firewall rules can also help against these sorts of attacks -- make sure your firewall offers this feature and that it is enabled.
- Try to run operating systems with less predictable IP ID sequences, such as recent versions of OpenBSD, Solaris, or Linux. While these operating systems are immune from becoming zombies with the current version of Nmap, they may not stop all IPID-related attacks. Further investigation is needed.
- Implement egress filtering to prevent spoofed packets from leaving your network. This prevents your employees/students from launching some of these attacks.

### **Internet Service Providers (ISPs)**

- The most important protection ISPs can offer is utilizing egress filtering to prevent spoofed packets from leaving your networks. This prevents users from executing many nasty attacks in addition to stopping Idle scanning. Besides helping the Internet, egress filtering can save you from substantial costs investigating IP spoofing attacks.

## Common Defense against Idle scan

Don't use a public host in front of your firewall that uses a predictable IPID sequence.

Solaris and Linux are two operating systems that are not vulnerable to this type of behavior.







## Methods of Intrusion Detection

There are three ways to detect intruders in different categories:

**Signature based Detection Policy:** In this technique predefine signature matches the network traffic it sees against a list of attack signatures in a packet. The main drawback of this policy is that, it look predefine signatures and therefore it can miss newly developed attacks which may contain malicious activity. Properly tuned signature detection IDS might be high on false negative, but can be low on false positive.

**Anomaly based Detection Policy:** It learns what “normal” traffic for your network looks like and it has no predefine signature for match the packet trace and will then alert you when it sees something abnormal. In this technique some time anything new or something different might have the chance of being labelled as abnormal traffic, so properly tuned anomaly detection IDS might be high on false positive, but low on false negative.

**Hybrid Detection Policy:** This technique is a combination of anomaly and signature and takes the best features of both the techniques used for detection purpose. This combined approach gives existence of single IDS for monitoring the attacks in network

## Categories of Intrusion Detection System

There are three categories to detect intruders-

**Host Intrusion Detection System (HIDS):** A host-based intrusion detection system (HIDS) is a system that monitors a computer system on which it is installed to detect an intrusion, and responds by logging the activity and notifying the designated authority.. Host

based intrusion detection systems has ability to log analysis, integrity management checking, detection of root kit and alerting.

**Network Intrusion Detection System (NIDS):** A network-based IDS (NIDS) analyzes packets coming across a network connection for data that look like its part of an attack. NIDS analyze network traffic for attacks, using signature or anomaly detection (or both). Its network interface card (NIC) runs in promiscuous mode, which means that it captures all network traffic that goes by its NIC, not just the traffic destined for the IDS system itself.

**Distributed Intrusion Detection System(DIDS):** Distributed Intrusion Detection System (DIDS) is a combination of NIDS sensors and HIDS sensors, or both, distributed across your organization, and they reporting to a central correlation system. Attacks are logged either periodically or continuously that generated on the sensors to the server station where they can be stored in a central database.

## Overview of Snort Tool

Snort is an open source and freely available network based intrusion detection and prevention system (available at [http:// www.snort.org/snort-downloads?](http://www.snort.org/snort-downloads?)). It can analyze the packet in real-time traffic on any network. It analyzes protocol and also has the ability to detect different type of attacks in network. Intrusion detection process of snort based on rule, basically snort rule written by user apply to checks against packet.

Snort can be configured to run in following three modes:

- **Sniffer mode** simply reads the packets of the network and display packet detail to user in a console (screen).
- **Packet Logger mode** used to logs the packets to disk in given format.

- **Network Intrusion Detection System (NIDS) mode** used to performs detection and analysis of packet in real time network traffic and generate alert if any suspicious activity found. This is the most complex and configurable mode.

**Components of Snort:** Snort is basically divided into five components and these components work together for every packet to detect particular attacks and to generate alert and output in appropriate format from the detection system. A Snort-based Intrusion Detection System consists of the following major components shown below:

**Components of Snort Packet decoder:** The packet decoder takes the Layer 2 data sent over from the packet capture library and takes it apart. First it decodes the Data Link frame (such as Ethernet, Token Ring, or 802.11), then the IP protocol, then the TCP or UDP packet. When finished decoding, Snort has all the protocols information in all the right places for further processing.

**Preprocessors:** These are components or plug-ins which performed after packet decoder and it can be mixed with Snort to modify or arrange data packets to specific protocol before those packets reach to the detection engine execute some operation to find out if the packet is being used by an intruder and send it specific task.

**Detection engine:** The detection engine is the heart of Snort. It takes information from the packet decoder and preprocessors and operates on it at the transport and application layers, comparing what's in the packet to information in its rules-based detection plug-in. These rules contain signatures for attacks.

**Logging and Alerting System:** Logging and alerting system working depends on the output phase of the detection engine. It used to log the activity and generate the alert based on the previous component of snort.

**Output Modules:** Plug-ins of output modules are performed various operations for the output generated by the logging and alerting system of Snort. Output modules are very useful when we show the attacks in web based user interface using third party tools like BASE, Snorby and SGUIL.

**Snort Rules:** Most intruder activity has some sort of signature like viruses. We create Snort rules to detect intruders using information about these signatures. Signatures may be present in the payload or in the header parts of a packet. Snort's network intrusion detection system is based on rules and these rules are based on intruder signatures. These rules can be used to check various parts of a data packet and rules looks like as follows. alert ip any any [?] any any (msg:"snort bad rule";content:hello; sid:10000099;) rule header rule option

**Idle Stealth Port Scan -** In order to find out which services are running on a specific host and can be attacked without revealing his own IP address .The idle stealth scan (or zombie scan) is a stealthy port scanning technique which allows an attacker to scan a target machine ports without the need of sending a single IP packet containing his own IP address directly to target. Instead he uses the IP address of a third host (zombie). The zombie host also known as idle host for port scans. Effectively, an idle port scan consists of three steps that are repeated for each port:

- Internet Protocol Identification (IP ID): In IPv4, the Identification (ID) field is a 16-bit value that is unique for every datagram for a given source address, destination address, and protocol, such that it does not repeat within the maximum datagram lifetime [RFC6864]. The IPv4 ID field was originally intended for fragmentation and reassembly.

**Attacker Zombie Target Step 1.** Probe the zombie's IP ID and records it. The attacker sends a SYN/ACK to zombie. The zombie not expecting SYN/ACK, send back a RST, disclosing it IP ID. Step 2. Forge a SYN packet from the zombie. The target sends a SYN/ACK to zombie in response to the SYN that appear to come from the zombie. The not expecting it, send back a RST, incrementing it IP ID in the process. Step 3. Probe the zombie IP ID again. The zombie IP ID has increased by 2 since step 1, so the port is open. Idle Stealth Port Scan for Open Port.

**Step 1. Probe the zombie's** IP ID and records it. The attacker sends a SYN/ACK to zombie. The zombie not expecting SYN/ACK, send back a RST, disclosing it IP ID. Step 2. Forge a SYN packet from the zombie. The target sends a RST (the port is closed) in response to the SYN that appear to come from the zombie host. The zombie ignores the unsolicited RST, leaving its IP ID unchanged. Step 3. Probe the zombie IP ID again. The zombie IP ID has increased by only 1 since step 1, so the port is not open.

For more information, please visit —><https://www.slideshare.net/skpatel91/detection-of-idle-stealth-port-scan-attack-in-network-intrusion-detection-system-using-snort>