# Finding a needle in a haystack - A comparative study of IPv6 scanning methods

Erik Bergenholtz
Dep. of Computer Science and Engineering
Blekinge Institute of Technology, Sweden
Email: ebz@bth.se

Dragos Ilie
Dep. of Computer Science and Engineering
Blekinge Institute of Technology, Sweden
Email: dil@bth.se

Andrew Moss
Dep. of Computer Science and Engineering
Blekinge Institute of Technology, Sweden
Email: awm@bth.se

Emiliano Casalicchio
Dep. of Computer Science and Engineering
Blekinge Institute of Technology, Sweden
Sapienza University of Rome, Italy
Email: emiliano.casalicchio@uniroma1.it

***Abstract*—It has previously been assumed that the size of an IPv6 network would make it impossible to scan the network for vulnerable hosts. Recent work has shown this to be false, and several methods for scanning IPv6 networks have been suggested. However, most of these are based on external information like DNS, or pattern inference which requires large amounts of known IP addresses. In this paper, DeHCP, a novel approach based on delimiting IP ranges with closely clustered hosts, is presented and compared to three previously known scanning methods. The method is shown to work in an experimental setting with results comparable to that of the previously suggested methods, and is also shown to have the advantage of not being limited to a specific protocol or probing method. Finally we show that the scan can be executed across multiple VLANs.**

***Index Terms*—ipv6, ipv6 scanning, cyber scanning, host discovery, penetration testing**

## I. Introduction

Scanning networks for live hosts is an important part of both penetration testing (e.g. [1], [2]) and malware propagation (e.g. [3], [4]). It is therefore important to understand the performance of available scanning methods, both to build scanning resistant networks and to perform security scans. In IPv4, finding most active hosts in a given network is quick, even with naive methods. In IPv6 this task is far from trivial, because the size of a single IPv6 local network is $2^{32}$ times larger than the entire IPv4 address space. As IPv6 becomes more and more widespread [5], understanding how to efficiently scan IPv6 networks for live hosts becomes increasingly more important to security experts.

While heuristics that could potentially make the task of host discovery in IPv6 easier, such as listening to SNMP traffic or sending `ICMPv6 ECHO REQUEST` messages to IPv6 multicast groups (e.g. [3], [6]), are already proposed in the literature, there is a lack performance evaluation studies.

In this work we provide a twofold contribution. First, we propose *Delimiting DHCP (DeHCP)*, a novel approach to limiting the search space of an IPv6 subnet. *DeHCP aims to limit the search space around computers that are closely clustered together in the address space, allowing for a more rigorous sequential scan of that area.* We demonstrate that this will provide a good balance between number of hosts being found and time taken to perform the scan. Second, the performance of *DeHCP* is evaluated and compared with three previously known methods, *Multicast ICMPv6*, *Malformed Multicast ICMPv6* and *Eavesdropping*. These methods were chosen because they have a potentially very high success rate in scanning a LAN.

This study is limited to local-to-local scans[7], as the problem of remote-to-local scans is far more complex because the scan must be performed through a firewall. The study covers three active scanning methods and one passive method for comparison. We are also limiting the scope to non-intrusive methods, i.e. methods that do not attack, disrupt or degrade the normal network operation. This matches what a system administrator defending their network would expect from a penetration tester scanning their network.

The paper is organized as follows. In Section II previously known scanning methods are presented. Related work is discussed in section III. Section IV presents the proposed scanning method, and the performed experiments are described and discussed in Section V. Our conclusions are laid out in Section VI. Future work is suggested in Section VII.

## II. Background

In this section, firstly a taxonomy of IPv6 scanning methods is defined. Then, scanning methods considered in this paper are presented.

### A. Types of scanning

There are multiple types of scanning. A first broad classification is: link and network layer scanning [8]; and scanning methods working at upper layers of the network stack, like DNS (e.g. [1]). In this study we focus on the first group. In this section, the terms *active scanning*, *passive scanning*, *remote-to-local scanning*, and *local-to-local scanning* [7] are defined for link and network layer scanning. Methods of IPv6 scanning that are not in these two layers are not necessarily covered by the taxonomy presented here and are out of the scope of this paper. We observe that active scanning can be

978-1-7281-1244-2/19/$31.00 

either local-to-local or remote-to-local, while passive scanning can only be local-to-local.

*1) Remote-to-local scanning:* In a remote-to-local scan, the inside of the target network is scanned by a host located on a different network. Being able to perform this type of scan would be devastating to a network, as it would mean that the entire infrastructure is wide open to the outside world.

*2) Local-to-local scanning:* As the name suggests, local-to-local scanning is performed on a local network from a host residing on that same local network. As the host performing the scan is already inside the network, this type of scan is not as severe a security risk as a remote-to-local scan. The methods evaluated in this paper all belong to this category.

*3) Active scanning:* With active scanning, network devices or services are discovered by transmitting *probe* packets on the network and listening for responses to said probes. In the context of host discovery, the *ping sweep* is a good example of a typical active method. *Ping sweep* is a method in which each IP address in an address range is probed with an `ICMP ECHO REQUEST`.

*4) Passive scanning:* Passive network scanning is performed simply by listening to the traffic on the network. The limitations with passive scans are that only hosts that engage in network activities on their own can be discovered with passive methods, and because only traffic that traverses the links on which the scanning host is listening is visible, passive scanning methods are limited to the local-to-local scope.

### B. Multicast ICMPv6

The *Multicast ICMPv6* method is an active local-to-local scan. It utilizes the built-in IPv6 all nodes multicast group, with IP address `ff02::1` [9]. This is a multicast group that all nodes on the local network must be subscribed to, as it is used to perform *Neighbour Discovery* [10] as well as other essential IPv6 functions. A *single* `ICMPv6 ECHO REQUEST` is sent to this address, which reaches all hosts on the local segment. According to RFC4443 [11], these requests should yield an `ICMPv6 ECHO REPLY` from each receiving host.

Adding a Hop-by-Hop IPv6 extension header [12] to the IPv6 header may mitigate ignored `ICMPv6 ECHO REQUEST` messages. According to the RFC, if the `Type` field of this header is set to `0x80` the packet should be discarded by the receiving host and an `ICMPv6 PARAMETER PROBLEM` message should be sent to the originating host. Disabling this behaviour should not be possible, as it is an error detection feature.

Throughout this paper, normal multicast ICMP is referred to as *Multi-Ping* while multicast ICMP with the added Hop-by-Hop header is referred to as *Hop-Ping*.

### C. Eavesdropping

*Eavesdropping* is a passive scanning method that consists of listening to the network and recording the IPv6 source address of all captured packets. To ensure that the methods based on ICMPv6 do not interfere with our results from using this method, all ICMPv6 packets of `ICMPv6 PARAMETER PROBLEM` or `ICMPv6 ECHO REPLY` types are ignored in our eavesdropping experiments.

## III. Related work

We have observed two major branches of solutions to the IPv6 scanning problem. In the first branch, the solutions are based on DNS or similar services, all of which are outside the scope of this paper. The second branch uses common IPv6 address patterns to reduce the search space. We discuss the latter category below, along with miscellaneous methods that fit into neither group.

Pattern based methods infer address patterns from network data. One such method recursively determines a bit-pattern fitting most addresses in the training data, which can be used to generate new addresses. The method was proposed by Ullrich et. al. in [13]. Similarly the *Entropy/IP* system takes a set of sample IPv6 addresses from a target AS network as input, and exposes the underlying addressing scheme using entropy computation and Bayesian Networks. *Entropy/IP* was proposed by Foremski et. al. in [14], and can also be used to predict potentially used addresses. Finding densely populated IPv6 address prefixes can be done using *Multi-Resolution Aggregate Count Ratios*, as shown by Plonka and Berger in [15]. It has also been shown that random scanning combined with simple transformations of IPv6 addresses from known datasets, can quickly identify a large number of active router interfaces. The method is implemented in the tool **Yarrp6**, and is proposed by Beverly et. al. [16].

Extracting routing information, examining neighbour caches, eavesdropping and multicast `ICMPv6 ECHO REQUEST` messages are examples of miscellaneous methods, and were suggested by Bellovin et. al. [3]. Sending fake router advertisements and using the Hop-by-Hop IPv6 extension header to elicit error messages also fall into this category. Their performance as means for worm propagation was evaluated by Li et. al. [17]. They concluded that in theory, fake router advertisements and multicast ICMP enable faster propagation, while using the Hop-by-Hop extension header provides better coverage. Methods such as listening to SNMP traffic and using local name resolution protocols like mDNS [18] are also possible methods, and are suggested in RFC7707 [6] along with several of the already mentioned methods.

## IV. Delimiting DHCP (DeHCP)

*DeHCP*[1] is a novel local-to-local active scanning method aiming at reducing the address search space. *DeHCP* is based on finding a range of addresses where the host density is likely higher than that of the rest of the network, therefore yielding a large amount of discovered hosts for a relatively small amount of work. To achieve this goal, we use a binary search algorithm to find the upper and lower bounds of a potential DHCP pool ($DHCP_{start}$ and $DHCP_{end}$). The trade off is that the method may miss some live hosts, in particular if they are located outside of a densely populated area. The method treats the

[1] https://github.com/erikbergenholtz/v6scan

**Algorithm 1** Algorithm for finding the lower or upper bound

```
1:  procedure DELIMIT(l, u, w)
2:      host ← (l + u)/2
3:      if l ≥ u then return host
4:      if host responds to probe then
5:          if Looking for lower limit then
6:              return DELIMIT(l, host − 1, w)
7:          else
8:              return DELIMIT(host + 1, u, w)
9:      else
10:         for all tmp in [host − w, host + w] do
11:             if tmp responds to probe then
12:                 if Looking for lower limit then
13:                     return DELIMIT(l, tmp − 1, w)
14:                 else
15:                     return DELIMIT(tmp + 1, u, w)
16:         if Looking for lower limit then
17:             return DELIMIT(host + 1, u, w)
18:         else
19:             return DELIMIT(l, host − 1, w)
```

address space as a range of integers, and so it is protocol independent. This means that the method can delimit DHCP pools in both `IPv4` and `IPv6` networks.

*DeHCP* is an active method that is not constrained to a specific probing technique. In this study, two probing methods were evaluated; a ping (*DeHCP*$_{icmp}$), and a port scan using **nmap** [19] (*DeHCP*$_{nmap}$).

The method essentially performs two scans. First, it scans a small amount of nodes to find the boundaries of the most populated range of addresses. Then, that range is scanned sequentially to find the active hosts. The algorithm for finding the boundaries is shown in Algorithm 1. It takes three parameters `l`, `u` and `w`. `l` is the lower bound of the area currently scanned for the DHCP boundary, `u` is the upper bound of the same area. These two parameters are bounded by the network address space. The window size `w` is used to determine whether or not non-responding addresses are outside the DHCP pool. This process is explained in greater detail below.

A starting point, referred to as a *seed*, must be provided when first running the algorithm. The seed is supplied as either the `l` or `u` argument of the function, depending on whether $DHCP_{start}$ or $DHCP_{end}$ is to be determined. The seed address should be inside of the DHCP pool, otherwise one of the detected boundaries have an unconditional error of $|DHCP_{bound} - seed|$.

The algorithm starts by calculating the middle point between `l` and `u`, called `host` in Algorithm 1. This address is probed to see if it is live. If it answers, it is assumed to be inside the DHCP pool, and consequently the search area is adjusted to move away from the seed address. This is done in the recursive function calls in lines 6 or 8. If the address did not respond to the probe, the range $[host - w, host + w]$ is scanned sequentially for live hosts to determine if the window overlaps the DHCP pool. This is based on the assumption that a host outside the DHCP pool is less likely to have live neighbours than a host inside the DHCP pool. If any of these hosts respond to their probes, the algorithm adjusts the search space in the same way as above. In the case that none of these addresses respond `host` is assumed to be outside the DHCP pool. The search space is then adjusted to move closer to the seed address. At some point, `l` will become larger than `u`. This is the stop condition for the algorithm, and the last value of `host` is returned as the determined boundary of the DHCP pool.

## V. Experiments and Analysis

To evaluate and compare the methods considered in this paper, experiments were performed in an emulated network. These experiments were preceded by a simulation of the *DeHCP* method, and complemented by evaluation in a production environment.

### A. Simulation of DeHCP

Simulation was used as a means to determine which window size $w$, in the *DeHCP* method, gives the most accurate outcome. In the simulation, the network was modelled as a binary vector $a = \{a_1, ..., a_{10^6}\}$, where $a_i = 1$ represents a responding host and $a_i = 0$ represents a host not responding. These values were uniformly randomized according to the two density parameters explained below. The range $i \in [450000, 550000]$ was defined as the DHCP pool, and was placed in the middle of the network to reduce bias in the results. The following parameters controlled the simulation:

1) *Outer density* ($\rho_{out}$) - Density in percent of responding hosts outside the DHCP pool, where $\rho_{out} \in \{0.001, 0.01, 0.1, 1\}$.
2) *Inner density* ($\rho_{in}$) - Density in percent of responding hosts inside the DHCP pool, where $\rho_{in} \in \{0.001, 0.01, 0.1, 1, 5, 10, ..., 100 | \rho_{in} \geq \rho_{out}\}$
3) *Window* - The amount of IP addresses scanned above and below an IP that does not respond to probes $w$, where $w \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 50, 500, 5000\}$
4) *Seed placement* - The placement of the seed address in relation to the DHCP pool. The evaluated seed addresses were each quarter into the DHCP pool.

The results of the simulations were analysed, and used as a basis for choosing the appropriate window size when running the experiments on the production and emulated network. The difference between the determined and actual boundary of the DHCP pool $\Delta DHCP$ was recorded in each simulation, and each set of parameters was simulated 100 times.

The sum $|\Delta DHCP_{start}| + |\Delta DHCP_{end}|$ of absolute differences was calculated for each simulation. The boxes in Figure 1 shows the the middle $50\%$ of simulation outcomes. From this we can see that smaller window sizes induce smaller errors, and that the error is smallest for a window size $w = 4$. This value also has a very low median of $\tilde{w}_4 = 14$. For this reason, $w = 4$ was chosen for the other experiments.

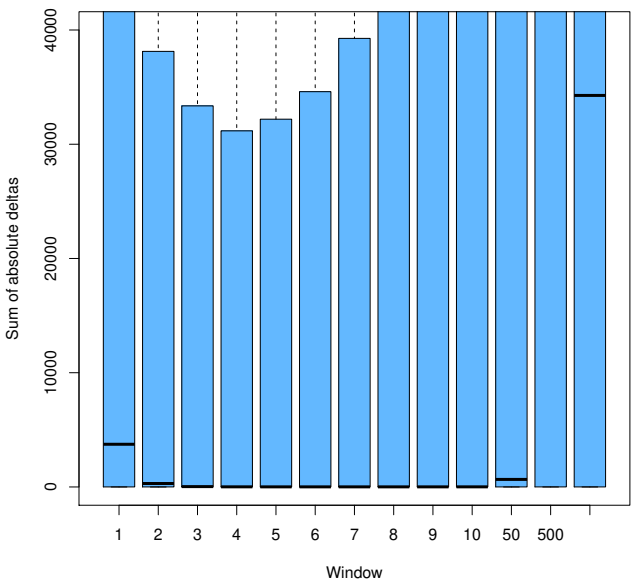


Fig. 1: Error for the tested window sizes over $\rho_{out} \in \{0.001\%, 0.01\%, 0.1\%, 1\% | \rho_{in} \geq \rho_{out}\}$, and $DHCP_{start} \leq seed \leq DHCP_{end}$.

### B. Emulated network

The scanning methods were first evaluated in an emulated network. The scans were performed over six hours, during which the *DeHCP* method was restarted once per hour since it does not run indefinitely. Both *DeHCP*$_{icmp}$ and *DeHCP*$_{nmap}$ were evaluated. A host is considered online by the *DeHCP*$_{nmap}$ scan if any of the probed ports reply with a `TCP/ACK`, or if **nmap** reports a MAC address. This is to avoid false positives caused by a router responding with `TCP/RST` if it knows the scanned host does not exist.

The emulated network was constructed to enable evaluation of the *DeCHP* method on a network using DHCPv6 to assign IPv6 addresses. The network had a simple layout, with 14 physical computers running Ubuntu 16.04 LTS. One of the computers was the designated router and DHCPv6 server, while the other 13 machines acted as hosts for 32 virtual machines each. In total, there were 429 active clients on the network, which all had their IPv6 addresses assigned via DHCPv6 by the router, from a DHCPv6 pool of 512 addresses. This DHCPv6 pool was situated in the address space so that the *DeHCP* scan would not find the address of the router, as this is a guaranteed hit which would bias the results because the scans were performed from the router.

Including the router there were 430 IPv6 addresses assigned. The scans were performed from the router, and an IPv6 address in the middle of the DHCPv6 address pool was used to seed the *DeHCP* method. Therefore, the emulation effectively emulated a scenario in which an external attacker gained access to the edge router, and performed the scan from there.

All virtual machines were configured the in same way. They ran Ubuntu Server 18.04 LTS, and each virtual machine ran a `TCP/SYN` scan of the router machine using **nmap**[2] with a randomized delay of 5 to 90 minutes between scans.

TABLE I: Results of scanning the emulated `/64` network

| Method | Type | Global (F/T) |
|---|---|---|
| *Eavesdrop* | Passive | 430/430 |
| *Multi-Ping* | Active | 430/430 |
| *Hop-Ping* | Active | 430/430 |
| *DeHCPv6*$_{icmp}$ | Active | 429/430 |
| *DeHCPv6*$_{nmap}$ | Active | 429/430 |

*1) Analysis:* The results from the experiments performed in the emulated network can be found in Table I, where the results are presented on the form *#Found/#Total* active hosts. In the table, we can see that all methods found all hosts that they were expected to find. The *DeHCP* methods did not find the router address, which is expected as the network was laid out so that the guaranteed answer from the scanning host would not bias the result, as discussed previously.

The *Multi-Ping* and *Hop-Ping* methods found all used addresses on the network within a few seconds, and because of this it would seem that these methods are clearly better than the others. However, it should be noted that the network was almost homogeneous, i.e. all hosts (with the exception of the physical machines) were configured exactly the same, meaning that if one of the hosts is configured to answer all machines are configured to answer. As shown in Section V-C2 neither of these methods reach close to this level of performance in a non-homogeneous network.

The *DeHCPv6*$_{icmp}$ and *DeHCPv6*$_{nmap}$ methods finished in approximately 20 and 21 minutes respectively, i.e. one host was discovered approximately every 3 seconds on average. To achieve this, both methods issued 1625 probes. This includes both determining the limits of the DHCP pool and sequentially scanning the entire determined range.

Another noteworthy point is that in the emulated network all virtual machines were rigged to port scan the designated router at least once per 90 minutes. Because the experiment was ran for six hours, this guaranteed that all hosts transmitted packets at least four times during the experiment. As all hosts are on the same network segment the *Eavesdropping* method was guaranteed to find all hosts. As seen in section V-C2 this is not realistic.

### C. Production network experiments

Following the experiments in the emulated setting, the scans were evaluated in two production networks belonging to Blekinge Institute of Technology; the campus wired network as well as the campus wireless network. In these experiments, the scanning methods were ran from a computer on the target network. The experiments were performed over 24 hours during business days to cover both periods of high and low network activity, which impacts the number of live nodes. The experiment on the production networks were performed in the same way as the experiment in the emulated setting, with the

[2] `nmap -6 -n -F -sS`

TABLE II: Results of scanning the BTH IPv4 networks

| | | Wired | Wireless |
|---|---|---|---|
| Method | Type | Global (F/T) | Global (F/T) |
| *DeHCPv4*$_{icmp}$ | Active | 122/1449 | 33/226 |
| *DeHCPv4*$_{nmap}$ | Active | 119/1449 | 51/226 |

exception for the longer duration. During the experiments, for each method we collected discovered IP addresses and timestamps for the discovery.

*1) BTH network topology:* The production networks used for the experiments were the wired and wireless university networks of Blekinge Institute of Technology. The networks use both IPv4 and IPv6, where the IPv4 addresses are assigned via DHCP [20], and the IPv6 addresses are autoconfigured. We scanned the IPv4 address space of these networks to show that the *DeHCP* scan is not dependent on any particular protocol. We also scanned the IPv6 network, to observe how the method behaves in a network with autoconfigured addresses. Because the addresses are uniformly distributed over the entire address space, and not clustered closely together, we expect that the method will find a very low number of hosts in this setting. The seed address for the *DeHCP* method is set to the IP addresses of the hosts from which the scan is performed, both in the case of IPv4 and IPv6.

The wired BTH network is divided into multiple smaller VLANs that have their own DHCP pools, each with 45 hosts on average. The IPv6 network is a full `/48` segment, while the IPv4 network segment for the whole BTH network is a `/19`, and is populated by approximately 2200 hosts. The wireless IPv4 network is a `/22` segment, and has a full `/64` IPv6 subnet dedicated to it. In the wired network, the *DeHCP* scans on the IPv6 wired network searched the entire `/48` address space for a DHCP pool, while the whole `/19` address space was searched in the IPv4 network. This means that multiple VLANs, and therefore DHCP pools, were covered by the scan. In this situation *DeHCP* is expected to find the convex hull of all DHCP pools. The IPv4 subnet of the wireless network is not covered by the `/19`, and so this is scanned on its own together with its `/64` IPv6 subnet. To estimate the state of the network a script that dumps the ARP table and IPv6 neighbour cache on request was installed on a central router. It should be noted that these values contain stale entries as well, which may inflate the total number of hosts on the network. In Tables II and III the number of unique IP addresses fetched from the router is denoted by *T* (Total), while the number of unique IP addresses found by the evaluated methods is the *F* (Found) value.

*2) Analysis:* The results of scanning the IPv4 networks can be found in Table II. We see that the scan found addresses from several different VLANs, as each VLAN is only populated by 45 computers on average. The results also show very clearly that the method works in an IPv4 setting, and so is independent of the protocol used on the network.

The results of scanning the IPv6 networks are shown in Table III. In this setting the addresses were distributed uniformly in the entire address space due to autoconfiguration, which lead to an extremely low host denisty ($\rho \propto 1 \times 10^{-16}\%$). Because the *DeHCP* scanning method attempts to find densely clustered IP addresses in the search space, as expected it found a very low number of hosts in these experiments, regardless of probing method.

Speaking in the favour of the method is the scanning time; upon examining the logs, we found that the entire delimiting scan took at most 1617 seconds. The range $[host-w, host+w]$ was scanned in each step of the binary search, as the scan did not find any hosts other than the scanning host. This means that in this case, when the delimiting scan was at its slowest, this process finished in just under 30 minutes for a network segment containing $2^{80}$ addresses, and used no more than 1548 probes in the process.

We can also observe from Table III that the *Eavesdropping* method found the largest amount of live IPv6 addresses of the evaluated scanning methods, while still only finding a small number of addresses in the wired network. The reason for the low number of discovered hosts, even with the *Eavesdropping* method, is that the experiment was conducted from a computer placed in the network of a single computer lab. As the experiment was not carried out while the lab was in use, the *Eavesdropping* method is only expected to find automatically generated traffic, such as router solicitations and advertisements.

The *Multi-Ping* almost reached the performance of the *Eavesdropping* method, but found mostly one kind of IP address per experiment. This is the expected outcome because of [21], as the ICMPv6 messages originated from the global IPv6 address. Note that the method finds some link-local IPv6 addresses when using a global address as the packet source. This is likely because the recipient of the `ICMPv6 ECHO REQUEST` did not have a global IPv6 address assigned, but had IPv6 enabled.

The *Hop-Ping* method performed very poorly. In fact, the only IP addresses that were found were those of the hosts performing the scan. Upon further investigation we found that the `ICMPv6 ECHO REQUEST` messages did not reach the other hosts on the network, but were dropped in transit.

## VI. Conclusions

The *DeHCP* simulation shows very promising results. With an outer density that is below 1% the error in finding the DHCP boundaries is small, with a median of 14 for a window size of $w = 4$. In reality, even a $0.001\%$ network density (i.e. the smallest density used in the simulation) outside a DHCP pool is highly unrealistic, as $0.001\%$ an entire `/64` IPv6 subnet is $1.8 \times 10^{14}$ hosts. Because of this, the *DeHCP* method is likely to perform well in a real life network using DHCPv6, as the network density will never be anywhere close to $0.001\%$ in such a setting. This is confirmed by the very positive results in the emulated network.

It should also be noted that, while discussions about the *DeHCP* method in this paper has largely revolved around the network utilizing DHCPv6, the main requirement for it to

TABLE III: Results of scanning the BTH /48 network with global source addresses

| | | BTH wired network | | BTH wireless network | |
|---|---|---|---|---|---|
| Method | Type | Link-local (F/T) | Global (F/T) | Link-local (F/T) | Global (F/T) |
| *Eavesdrop* | Passive | 29/1290 | 37/4483 | 247/249 | 235/486 |
| *Multi-Ping* | Active | 5/1290 | 40/4483 | 2/249 | 83/486 |
| *Hop-Ping* | Active | 0/1290 | 1/4483 | 0/249 | 1/486 |
| $DeHCPv6_{icmp}$ | Active | 0/1290 | 1/4483 | 0/249 | 1/486 |
| $DeHCPv6_{nmap}$ | Active | 0/1290 | 0/4483 | 0/249 | 1/486 |

function is that an address range of higher host density exists on the network. This means that the method could potentially discover ranges of servers with static IP addresses that are closely clustered together in the address space.

Finally, the *Eavesdropping* and *Multi-Ping* methods both performed as expected, and while the *Hop-Ping* method did not perform well in the real setting it was shown in the emulated network that the method works.

## VII. Future work

This work considers only a specific method of exploiting density in the address space to reduce scanning costs while still discovering a large amount of active hosts. Our future work will use density estimation methods and consider a more general network topology, where hosts are not necessarily clustered together in the address space.

## Acknowledgements

We would like to thank Björn Mattsson for allowing us to run our experiments on the BTH production network, for providing a script that showed the state of the network, and for helping when parts of the network needed to be reconfigured.

This project has received funding from *Swedish Agency for Economic and Regional Growth (Tillväxtverket)* under the umbrella of European Structural and Investment Funds (ESI) covered by grant agreement No **20201213** (Test Arena Blekinge). The project is run in collaboration with Blue Science Park in Karlskrona, Sweden.

## References

[1] C. Ottow, F. E. van Vliet, P. de Boer, and A. Pras, "The impact of ipv6 on penetration testing," in Information and Communication Technologies - 18th EUNICE/ IFIP WG 6.2, 6.6 International Conference, EUNICE 2012, Budapest, Hungary, August 29-31, 2012. Proceedings, ser. Lecture Notes in Computer Science, R. Szabó and A. Vidács, Eds., vol. 7479. Springer, 2012, pp. 88–99. [Online]. Available: https://doi.org/10.1007/978-3-642-32808-4_9

[2] H. Rafiee, C. Mueller, L. Niemeier, J. Streek, C. Sterz, and C. Meinel, "A flexible framework for detecting ipv6 vulnerabilities," in The 6th International Conference on Security of Information and Networks, SIN '13, Aksaray, Turkey, November 26-28, 2013, A. Elçi, M. S. Gaur, M. A. Orgun, and O. B. Makarevich, Eds. ACM, 2013, pp. 196–202. [Online]. Available: http://doi.acm.org/10.1145/2523514.2527001

[3] S. M. Bellovin, B. Cheswick, and A. Keromytis, "Worm propagation strategies in an ipv6 internet," LOGIN: The USENIX Magazine, vol. 31, no. 1, pp. 70–76, 2006.

[4] C. C. Zou, D. Towsley, and W. Gong, "On the performance of internet worm scanning strategies," Performance Evaluation, vol. 63, no. 7, pp. 700 – 723, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166531605001112

[5] "Google ipv6," https://www.google.com/intl/en/ipv6/statistics.html, accessed: 2018-06-12.

[6] F. Gont and T. Chown, "Network reconnaissance in ipv6 networks," RFC Editor, RFC 7707, March 2016.

[7] E. Bou-Harb, M. Debbabi, and C. Assi, "Cyber scanning: A comprehensive survey," IEEE Communications Surveys and Tutorials, vol. 16, no. 3, pp. 1496–1519, 2014. [Online]. Available: https://doi.org/10.1109/SURV.2013.102913.00020

[8] "Information technology – open systems interconnection – basic reference model: The basic model," International Organization for Standardization, Standard, June 1996.

[9] "Ipv6 multicast address space registry," accessed: 2018-06-14. [Online]. Available: https://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml

[10] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," RFC Editor, RFC 4861, September 2007, http://www.rfc-editor.org/rfc/rfc4861.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4861.txt

[11] A. Conta, S. Deering, and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," RFC Editor, RFC 4443, March 2006, http://www.rfc-editor.org/rfc/rfc4443.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4443.txt

[12] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC Editor, STD 86, July 2017.

[13] J. Ullrich, P. Kieseberg, K. Krombholz, and E. R. Weippl, "On reconnaissance with ipv6: A pattern-based scanning approach," in 10th International Conference on Availability, Reliability and Security, ARES 2015, Toulouse, France, August 24-27, 2015. IEEE Computer Society, 2015, pp. 186–192. [Online]. Available: https://doi.org/10.1109/ARES.2015.48

[14] P. Foremski, D. Plonka, and A. W. Berger, "Entropy/ip: Uncovering structure in ipv6 addresses," in Proceedings of the 2016 ACM on Internet Measurement Conference, IMC 2016, Santa Monica, CA, USA, November 14-16, 2016, P. Gill, J. S. Heidemann, J. W. Byers, and R. Govindan, Eds. ACM, 2016, pp. 167–181. [Online]. Available: http://dl.acm.org/citation.cfm?id=2987445

[15] D. Plonka and A. W. Berger, "Temporal and spatial classification of active ipv6 addresses," in Proceedings of the 2015 ACM Internet Measurement Conference, IMC 2015, Tokyo, Japan, October 28-30, 2015, K. Cho, K. Fukuda, V. S. Pai, and N. Spring, Eds. ACM, 2015, pp. 509–522. [Online]. Available: http://doi.acm.org/10.1145/2815675.2815678

[16] R. Beverly, R. Durairajan, D. Plonka, and J. P. Rohrer, "In the ip of the beholder: Strategies for active ipv6 topology discovery," 2018.

[17] J. Li, F. Su, Z. Lin, and Y. Ma, "The research and analysis of worm scanning strategies in ipv6 network," in 13th Asia-Pacific Network Operations and Management Symposium, APNOMS 2011, Taipei, Taiwan, September 21-23, 2011. IEEE, 2011, pp. 1–4. [Online]. Available: https://doi.org/10.1109/APNOMS.2011.6076995

[18] S. Cheshire and M. Krochmal, "Multicast DNS," RFC Editor, RFC 6762, February 2013, http://www.rfc-editor.org/rfc/rfc6762.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6762.txt

[19] G. Lyon, "Nmap security scanner," https://nmap.org/, accessed: 2018-05-24.

[20] R. Droms, "Dynamic Host Configuration Protocol," RFC Editor, RFC 2131, March 1997, http://www.rfc-editor.org/rfc/rfc2131.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2131.txt

[21] D. Thaler, R. Draves, A. Matsumoto, and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)," RFC Editor, RFC 6724, September 2012, http://www.rfc-editor.org/rfc/rfc6724.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6724.txt