Accommodating IPv6 Addresses in Security Visualization Tools*

David Barrera and P.C. van Oorschot
School of Computer Science, Carleton University, Canada

Abstract

Visualization is used by security analysts to help detect patterns and trends in large volumes of network traffic data. With IPv6 slowly being deployed around the world, network intruders are beginning to adapt their tools and techniques to work over IPv6 (vs. IPv4). Many tools for visualizing network activity, while useful for detecting large scale attacks and network behavior anomalies still only support IPv4. In this paper, we explore the current state of IPv6 support in some popular security visualization tools and identify the roadblocks preventing those tools from supporting the new protocol. We propose a filtering technique that helps reduce the occlusion of IPv6 sources on graphs, and enables IPv4 visualization tools to display both IPv4 and IPv6 sources on a single graph. We also suggest using treemaps for visually representing the vast space of remote addresses in IPv6.

Index Terms: K.6.5 [Management of Computing and Information Systems]: Security and Protection (D.4.6, K.4.2)—Unauthorized access (e.g., hacking, phreaking); I.6.8 [Simulation and Modeling]: Types of Simulation—Visual

1 INTRODUCTION

IPv4, the well-known protocol employed for practically every type of network communication, has been used for over 25 years. Unfortunately, its limited address space is projected to be exhausted by August 2012 [9]. With this date fast approaching, system administrators, vendors, ISPs, and end users around the world are slowly adopting IPv6. As IPv6 becomes more widely deployed, adversaries are updating their tools to use it, and benefiting from the fact that IPv6 is often overlooked, disabled, or otherwise unsupported in many network analysis tools. While Internet-scale enumeration of IPv6 hosts is unfeasible, attackers are using advanced features in the new protocol for local network enumeration and reconnaissance.

In security visualization, remote (source) IP addresses are frequently used as one of the displayed fields to discover trends or patterns of activity in groups of hosts. Since IP addresses uniquely identify devices on a network, visualizing their behavior provides better insight into the structure and location of networks. Many security visualization tools have been designed to support IPv4 exclusively, e.g., graphs which enumerate the 32-bit address space (or segments thereof) on axes. Working with the 128-bit addresses of IPv6 is not as simple.

In this paper we review why visualization of remote IP addresses is important, and why IPv6 remote addresses specifically are difficult to graph. We analyze the structure of IPv6 addresses as well as the current IPv6 address allocation trends and use our observations to develop three new techniques for visualizing the IPv6 address space. One technique filters out unused IPv6 address ranges, allowing analysts to visualize the effective IPv6 address space seen generating traffic during a network capture. The second technique allows IPv4-only tools to display dual stack network captures. In the third technique, we propose the use of an existing visualization tool (treemaps) to visualize IPv6 datasets.

The rest of this paper is structured as follows. Section 2 reviews technical details of IPv6 and how it differs from IPv4, and surveys related work. Section 3 considers the general architecture of visualization systems and issues that must be addressed to support IPv6. Section 4 describes the datasets used to demonstrate visualizations in this note. Section 5 proposes three new techniques for visualizing network traffic containing IPv6 addresses. Section 6 presents concluding remarks.

*Version Mar. 22, 2010. Contact author: dbarrera@ccsl.carleton.ca. A preliminary version of this paper appeared as pp. 21-26 in the proceedings of the 6^{th} International Workshop on Visualization for Cyber Security 2009 (VizSec 2009), IEEE Computer Society.

2 Background and Related Work

To better understand the security visualization challenges faced when trying to visualize IPv6, we first review how it differs from IPv4.

2.1 IPv6 Review

IPv6 extends IPv4 from 32 to 128-bit addresses [6] and includes simplified headers that help improve packet processing performance. Although the IPv6 specification has existed for over 10 years, deployment has been relatively slow. An October 2008 study by Google [7] found that IPv6 penetration worldwide is still less than 1% in any country (measured by the number of hosts requesting AAAA DNS records from Google domains). The study also found that 67.9% of IPv6 users worldwide are using 6to4 (a transition mechanism that encapsulates IPv6 packets inside IPv4 packets to help connect "islands" of IPv6 networks) as opposed to native, IPv6-only connectivity. For the United States and Canada, 95% of IPv6 users are using 6to4, presumably because of the small number of ISPs offering native IPv6 in these countries. China and France on the other hand are over 70% native.

It is important to note that low current use of IPv6 on the public Internet does not mean hosts using the protocol should be ignored. Despite the aforementioned statistics, IPv6 penetration is growing everyday and is now enabled by default in many network applications. Hosts are capable of simultaneously using IPv6 for internal network communication, and IPv4 for Internet communication. Thus, even if a host does not have Internet-wide IPv6 connectivity, it might still be exploitable over IPv6 within its subnet.

Transition Mechanisms. To help speed up IPv6 deployment and help ease migration, several RFCs provide methods allowing IPv4 and IPv6 to inter-operate during an initial transitional phase. These transition mechanisms bring new issues to security analysis. For instance, if a network application uses one of the transition mechanisms such as a tunnel (a point to point IPv4 link between an end-user and an ISP through which IPv6 packets are encapsulated) or 6to4, the source and destination IP addresses on those packets may not represent the actual communication endpoints.

Address Representation. IPv6 addresses are not written in the IPv4 dotted-decimal notation, but rather as a group of eight 16-bit *hextets* (as opposed to four 8-bit pieces in IPv4). Leading zeroes in each hextet can be omitted, and up to one group of two or more hextets consisting of only zeroes can be expressed as "::". The following examples show different possible notations for an IPv6 address.

```
2001:0DB8:0000:0078:9ABC:0000:0000:0000
2001:0DB8:0:0078:9ABC:0:0:0
2001:DB8:0:78:9ABC::
```

The representation of prefixes in IPv6 is similar to the Classless Inter-Domain Routing (CIDR) notation in IPv4. The prefix length, which specifies how many of the leftmost contiguous bits compromise the prefix, is appended after the IP address. For example, a network block where 64 bits are used for assigning to hosts and 64 remain constant would be expressed as: 2001:DB8:0:78::/64.

2.2 Other Differences from IPv4 and Visualization Implications

The updated IPv6 packet header omits rarely used fields and has been simplified to allow faster processing by network devices. As shown in Figure 1, headers now contain fewer fields, by eliminating the *header length, identification (IPID), flags, fragment offset*, and *header checksum*. The *version, traffic class, payload length, next header* and *hop limit* fields in IPv6 headers are either the same as in IPv4, or used for similar purposes, making them somewhat compatible in both analysis and visualization tools. The *flow label* field is unique to IPv6 and provides a method of tagging all packets belonging to the same end-to-end conversation.

Since IPv6 and IPv4 operate at the Internet layer of the IP stack, there are only differences between protocols at this specific layer. TCP, UDP and any other protocols that run over IP are unchanged, and therefore visualization tools that graph only transport or application layer data generally do not need to be modified to work with IPv6. On the other hand, security visualization tools that display Internet layer data, such as IP addresses or other header fields might require changes to work with IPv6.

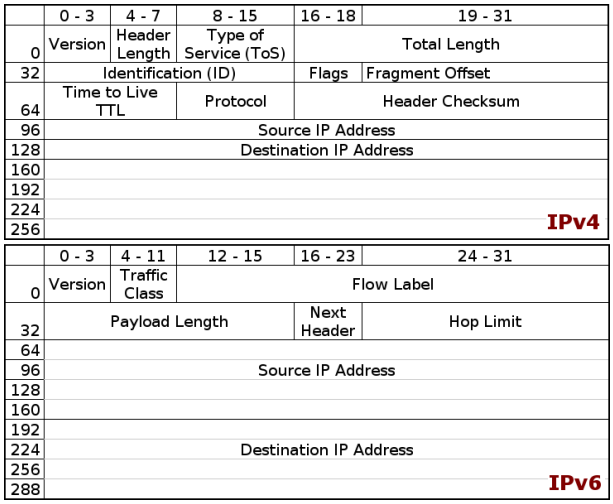


Figure 1: Packet Headers in IPv4 and IPv6

The address space in IPv6 is far too large for brute-force enumeration to be feasible (attackers cannot sequentially probe all IP addresses in IPv6). This means that remote host discovery is much less likely (unless it targets specific, small subnets). However there are some related IPv6 features such as neighbor discovery and multicast which should not be overlooked. Multicast addresses can allow a single device to discover all other IPv6 devices on a local network as described by Moore [14] using *The IPv6 Attack Toolkit* [4] (a collection of network tools for discovering and attacking IPv6 enabled devices).

Another interesting feature of IPv6 is address auto-configuration. Hosts can now assign themselves an IP address without the need for a DHCP server. Using ICMPv6, a server broadcasts the IPv6 prefix which all hosts should prepend to their auto-configured address (this is designed to be collision free since the host's MAC address is used as part of the address). While address auto-configuration helps for network renumbering, it makes monitoring hosts on an IPv6 network a more complicated task.

2.3 Focus on Remote IP Addresses

IP addresses uniquely identify hosts on networks. The vast majority of network analysis tools use IP addresses either to group or classify hosts at the network level. Visualizing IP addresses is useful for detecting similar behavior among groups of hosts as well as mapping the origin of a connection to an approximate geographical location.

It is useful to distinguish between externally initiated connections and internally initiated. In this paper, we focus on the former; the remote hosts in question are *remote addresses*, while those assigned to devices located within a local network perimeter are *local addresses*. Remote and local addresses are also referred to as *source* and *destination* addresses, respectively.

A network administrator should always have local knowledge of the network being defended, that is, which segments of the address space are populated or active. For this reason, monitoring target or local IP addresses in IPv6 is somewhat easier than doing the same for remote addresses. For instance, even if an administrator has 2^{64} addresses available for assignment, in all likelihood only a very small (known) subset of these will be actually used, and the corresponding devices can easily be monitored. Many security visualization tools graph the host segment of the IP address for displaying internal addresses. With this approach, no information is lost from the IP address because the network segment of the address is constant.

Remote IP addresses on the other hand are more difficult to monitor, since it is impossible to know ahead of time which remote hosts will initiate connections, and visualizing a small portion of the address could hide information from the user. It is common to see tools visualizing the entire remote IP address or the network segment of the address, with the drawback of losing the host-level accuracy. In this paper we

focus on visualizing remote IP addresses and attempt to limit the loss of accuracy.

2.4 Related Work

We are not aware of any literature that specifically addresses visualization of IPv6 traffic. The existing literature on security visualization also does not explicitly state that the focus is IPv4 traffic, but typically implicitly assumes that IPv4 is the underlying protocol of the analyzed network traffic. Several visualization tools [11, 12, 19] which provide good insight into remote IP behavior are limited to IPv4 addresses.

Nakamae et al. [18] attempted to visualize the structure of an IPv6 network. This differs from our approach in several ways. They did not display IPv6 addresses in their visualizations, or analyze IPv6 network traffic, but rather display the connections linking different ISPs (obtained from routing table entries) within a specific region. They do not aim to help visualize the behavior of groups of IPv6 hosts. They focus on representing the IPv6 address hierarchy (see Section 5.3) through an interactive three-dimensional display. Their results produce attractive 3D visualizations rich in information (compared to 2D), with the accompanying trade-off of requiring the user to actively interact with the tool to see information that may be occluded by other parts of the graph. The Cooperative Association for Internet Data Analysis (CAIDA) has also produced visualizations displaying IPv6 autonomous system (AS) interconnections [5].

3 Considerations for Visualizing IPv6 Datasets

Most visualization tools follow these steps to visually represent network traffic captures:

1. *Capture raw data*. Data off the wire is captured by a common network monitoring tool (Tcpdump, Wireshark, NetFlow, etc.).
2. *Parse*. The raw dumps are then parsed to extract information from the data structures used by the monitoring tools. This extracted data is written to text files.
3. *Process and reformat*. Text files are then processed for field selection and modified into the format required by the visualization tool.
4. *Display*. Graphically represent the data.

Data Capturing and Parsing. Popular data *capturing* tools have included support for IPv6 for a few years. These tools are capable of detecting IPv6 in IPv4 encapsulation, detecting the different types of addresses, and understanding the new security enhancements (e.g., IPSec) that are built-in to IPv6. For example, the SiLK tools [3] allow the analyst to display IPv4 flows as IPv6 flows containing 6to4 addresses. SiLK is also being improved to support faster manipulation of IPv6 addresses using high-performance data structures [13]. Since there is very good support for capturing IPv6, this part of the visualization process requires practically no changes. The main point is to make sure that network analysts enable the IPv6 capture feature as it might be disabled by default in some capture tools.

Once data has been saved to a capture file, parsers (generally written in scripting languages, e.g., Perl or Python) are used to read relevant fields from the capture. Parsers work with plain text so they should be able to extract IPv4 and IPv6 without much difficulty. When working with parsers, just as with capture tools, analysts need to make sure the parser is not discarding IPv6 records, since IPv6 addresses are written differently and may cause a poorly written IPv4-only parser to crash or ignore that part of the file.

Data processing and Visualizing. After parsing the file, data for the visualization must be selected. Common network fields that are used in visualization are: source and destination address, the IPID, and source and destination ports. Of these, only the source and destination ports remain unchanged in IPv6. Source and destination addresses have become 128 bits, and the IPID is removed from the header. Visualization tools must not try to extract an IPID (or other non-existent fields) from an IPv6 packet, nor attempt to display an IPv6 address when processing data records from an IPv4 space; existing tools that do so must be modified to prevent them from breaking when processing IPv6 traffic.

Current visualization tools generally don't support IPv6 in the sense that they assume the capture is IPv4 only, and therefore assume 32-bit addresses. For example, the Inetvis [20] cube is hard-coded for IPv4. If a dual stack capture (i.e., containing both IPv4 and IPv6 data) is loaded, the tool will just ignore the IPv6 packets and not display them. NVisionIP [12] and Flovis [19] are other examples of tools where

only 32-bit source addresses are currently supported. Koike et al. [11] also describe a technique similar to NVisionIP in which source and destination addresses are visualized in a 2D matrix; only IPv4 addresses can be displayed.

4 Dataset

Our sample visualizations were generated from two distinct datasets, one containing only IPv6 traffic, and the other containing both IPv4 and IPv6 traffic (sometimes referred to as dual stack). The datasets were obtained from the MAWI Working Group Traffic Archive [2], which is host to a large number of up-to-date traffic captures from several Internet backbone links.

4.1 IPv6-Only Dataset

For the visualizations of Sections 5.1 and 5.3 we used a dataset file from the MAWI Working Group's *Samplepoint-C* (one of their 16 sampling points), which contains 2 million IPv6-only packets, generated June 18, 2008 between 9:00 am and 1:36 pm, and anonymized using a modified version of *tcpdpriv* to comply with WIDE standards. Payloads were removed, but protocols and ports were left unchanged by *tcpdpriv*. The IP addresses were scrambled, with sources sharing the same address prefix preserving this property (two IP addresses A and B on the same subnet X would both be placed in a new subnet Y) throughout the duration of the capture period. The dataset contains 1924 (out of which 1466 generated only TCP and UDP traffic) unique source IP addresses generating traffic at an average 221kbps. Traffic is distributed as TCP (50.3%), UDP (29.6%) and ICMP (19.7%) as well as a small number of other protocols.

4.2 Dual stack Dataset

The dual stack dataset used in Section 5.2 is a capture file from the MAWI Working Group's *Samplepoint-F* which contains about 11.5 million packets captured over a 15 minute window on January 3, 2008. This dataset in particular recorded a considerably higher rate of traffic (averaging 49.95Mbps) than the IPv6-only dataset over a relatively small period of time. Network traffic was also anonymized using the same process as the dataset of Section 4.1. Most of the recorded packets are IPv4 (99.79%), with 354,472 unique hosts and the rest (0.21%) are IPv6, with 3029 unique hosts. Traffic for this dataset is distributed as TCP (63.59%), UDP (28.66%), ICMP (7.01%) and a small number of other protocols.

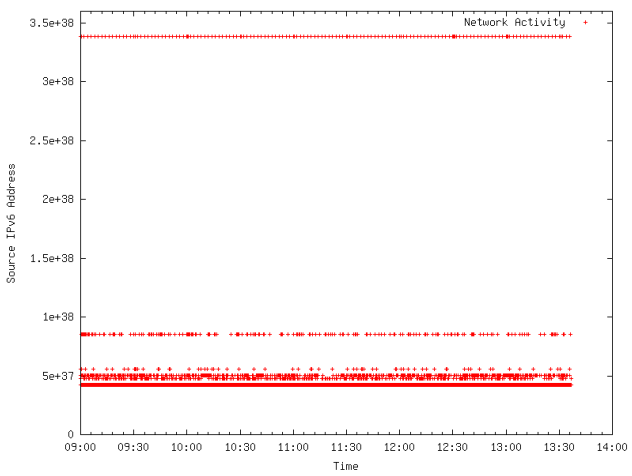
5 Visualization of Remote IP Addresses

To represent IP addresses in IPv6, trying to grow the range of an axis from 2^{32} to 2^{128} does not work. One must rethink how to display information to gain insight on how connections are distributed and where they come from. Additionally, we seek to work with the existing visualization tools with as little disruption as possible, even though such tools might not be ready for IPv6. We provide three ideas in this direction.

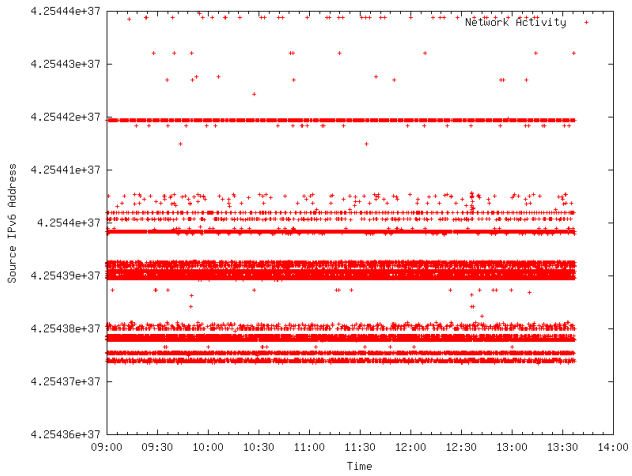
5.1 Whitespace filtering

One of the problems for visualization in IPv6 is that the address space is very sparsely populated. Most of the leftmost 4 bits are reserved by the IETF for future use, which leaves only one prefix (2000::/3) that can be globally routed. To demonstrate how unevenly distributed the remaining addresses are, in Figure 2(a), we plot the dataset of Section 4.1 using the full IPv6 address range on the y axis. Note that we can only distinguish six horizontal lines (corresponding to addresses beginning with 20, 24, 26, 2A, 3F and FE). Figure 2(b) zooms into the bottom line of Figure 2(a); what appeared to be a single address is actually a group of hundreds of addresses (displayed as horizontal lines). Even on a high-resolution display (e.g., 1920x1200), we could potentially still have thousands of remote IP addresses overlapped onto a single row of pixels of the screen (not the case in this example, since the capture contained only 1924 source addresses).

In IPv4, for an analyst to detect patterns and trends using a visualization tool, it is useful to look at neighboring addresses (in the same subnet or class). This can help identify hosts that are collaborating or exhibiting similar behavior. Furthermore, being able to see the behavior of all subnets simultaneously proves to be useful in detecting Internet-wide attacks. IPv6 doesn't immediately add more hosts to



(a) Remote IPv6 hosts seen generating packets during the capture period. At this scale, all points on the scatterplot overlap to the first byte of precision.



(b) 70000x zoom on the *y* axis focusing on the bottom solid line. Significant overlap of points remains on the *y* axis

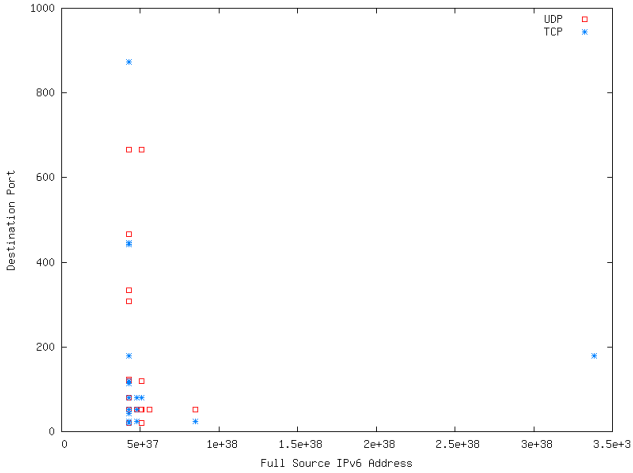
Figure 2: The sparsely populated IPv6 address space

visualizations, just a much wider address range. Removing the unused or unpopulated addresses, i.e., the whitespace (sometimes referred to as *darkspace* in scan detection literature), should restore the same, Internet-wide visualizations as with IPv4.

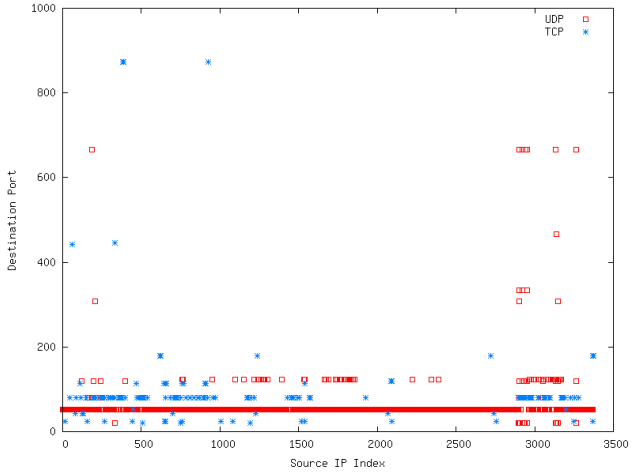
Eliminating the whitespace is easily done as follows. During the network capture, build a list of unique remote IPv6 addresses observed. To preserve relative placement (so that if address A numerically follows B, this relation is retained in the new list) of addresses, sort the resulting list by numerical value . Finally, when displaying data, display the index of the IP address in the list instead of the full address.

As an optional step, subnet information can be preserved in the graphs, by deliberately inserting gaps. We insert 2 null elements into the previously mentioned IPv6 address list between each populated 64-bit subnet. The end result is a list where all addresses in the same /64 subnet are contiguous, and all /64 subnets are separated by a small space. This optional step may help the user identify subnets visually rather than trying to otherwise validate findings in the original raw dataset.

To save the analyst from changing his workflow by having to look-up the address that matches a given index, visualization tools could be modified to display the address when mousing over an index.



(a) Full address space

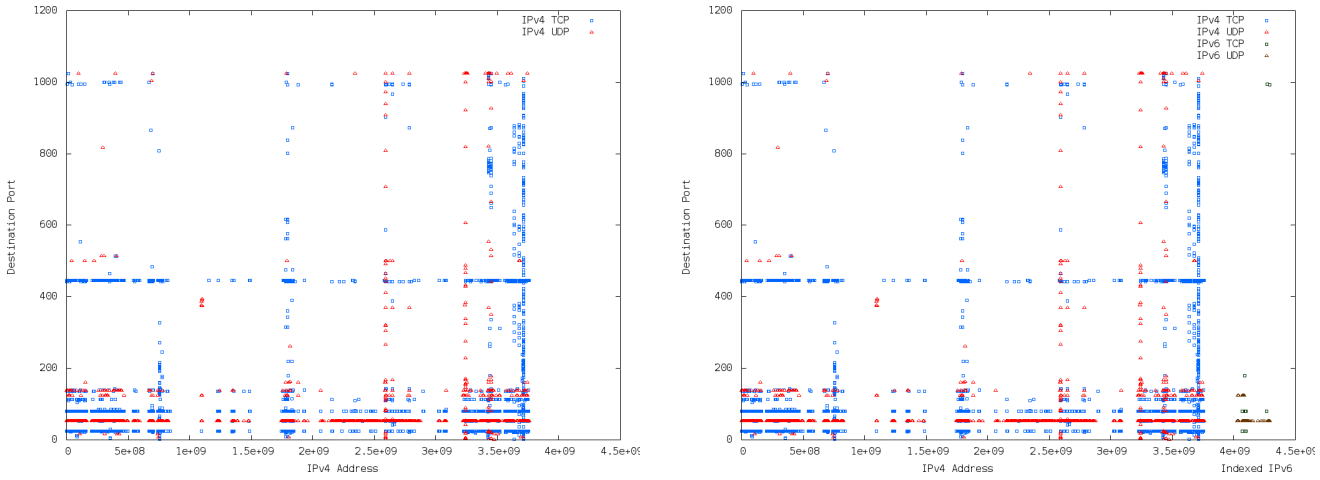


(b) Using index value in place of absolute IP address

Figure 3: Filtering out unpopulated address space

Figure 3(a) presents our first dataset as an example IPv6-only capture being visualized with no filtering at all. The *x* axis displays the 128-bit addresses as an integer value. Due to the uneven distribution of addresses, as well as the huge IPv6 address space, the populated addresses all overlap on the left side of the scatterplot. In Figure 3(b), unused address space has been eliminated, and each remote IPv6 address has been indexed. The resulting scatterplot is able to display over 1400 remote addresses and what destination ports each one targets during the capture period. This filtered scatterplot reveals that traffic to UDP port 53 (DNS) was frequent throughout the entire capture period, and that no single host attempted connections to a large number of destination ports (noted by the absence of vertical lines in Figure 3(b)). Note that the horizontal axis in Figure 3(b) displays the *address index* rather than the full address.

5.2 Mapping IPv6 into Unused IPv4 Address Space



(a) Only IPv4 Traffic

(b) Mapping IPv6 addresses into IPv4

Figure 4: Mapping indexed IPv6 addresses into the IPv4 address space

The technique of Section 5.1 works exclusively with IPv6 datasets. Because IPv6 and IPv4 can co-exist on a network, it is possible for a network capture to include both IPv6 and IPv4 packets. Furthermore, while migration from IPv4 to IPv6 takes place, it is likely that many networks will see both types of traffic. This section presents a technique that builds upon that of Section 5.1 to allow dual stack datasets to be displayed on IPv4-only visualizations (or limited to a tool assuming a 32 bit address space for display).

Due to the currently low deployment of IPv6, for a given dual stack network it is reasonable to assume that there will be more IPv4 traffic than IPv6 recorded at the network border. This may vary if traffic capturing is done internally (where most traffic could conceivably be IPv6), but that scenario is beyond the scope of this paper. Mapping the currently small amount of IPv6 traffic into unused (i.e., reserved for private or future use) portions of the IPv4 address space provides a visualization strategy which is capable of displaying both IPv6 and IPv4 traffic, while maintaining compatibility with current IPv4-only tools.

The mapping technique begins with indexing the dataset, assigning a sequential, unique ID to each IPv6 address seen during the capture period (as described in Section 5.1). IPv4 addresses are left unmodified during this step. Next, an unused IPv4 address range is selected. For this step good candidate networks include those specified as private in RFC 1918 [16], or the entire Class E which is currently reserved for future use. [10] range (240.0.0.0 - 255.255.255.255). Using, for example, RFC 1918's $10.*.*.*$ network will provide 3 full octets (2^{24} addresses) for mapping in IPv6 addresses, allowing up to 16,777,216 IPv6 addresses to fit in that range without overlap. Similarly, if the $192.168.*.*$ range were selected 65,536 (=2^{16}) IPv6 addresses could be mapped. In our example, we select the reserved Class E network for our mapping since all three RFC 1918 address ranges were in use in the dataset (due to the anonymization process). Using the Class E range has the added benefit of concentrating all IPv6 traffic at the end of the IPv4 address range, rather than embedding it between used ranges.

It also offers 28 bits of unused IPv4 space. Finally each indexed IPv6 address is translated (mapped) into its corresponding IPv4 address depending on the range selected. In our example, the first and smallest IPv6 address becomes 240.0.0.0. The visualization is then generated using the existing IPv4-based tool, displaying both IPv4 addresses and mapped IPv6 addresses.

Figure 4 shows the resulting visualization displaying the traffic from the 15 minute dataset. In Figure 4(a), only IPv4 TCP and UDP connections are plotted. The integer source IP on the x-axis ranges from 0 to about 4.2 billion ($=2^{32}$), but there is a large section of unused addresses to the right of the graph. In Figure 4(b), the IPv6 traffic is included as well, making use of the free space, thus displaying all TCP and UDP traffic from the dataset of Section 4.2. We can see in Figure 4(b) that some trends, such as a constant stream of UDP connections to destination port 53 (DNS), are common to both the IPv4 and IPv6 portions of the graph. We also note that there is no HTTPS traffic (TCP port 443) over IPv6, which is common in IPv4, and practically no traffic to destination ports greater than 200 was recorded in the IPv6 address space.

This approach has several benefits. Current IPv4 visualization tools don't have to be modified to display dual stack captures after they are parsed using this technique. This allows forward compatibility and some breathing room while newer IPv6 visualization techniques are developed. Also, analysts can use a single visualization which displays all traffic, rather than doubling their workload by analyzing IPv4 and IPv6 data separately. When all traffic is on a single display, correlating activity of hosts using different IP versions is more straightforward.

Because this approach moves one address space into another, analysts must be informed regarding this approach, or otherwise educated to differentiate between the two sections (the IPv4 range and the IPv6 range) of the graph. Failure to understand how the space is divided may otherwise lead to inaccurate interpretation of graphs. Finally, as IPv6 gains traction and IPv4 is phased out, our approach will need to be modified to support a larger number of IPv6 addresses. This is discussed in more detail in Section 5.4

5.3 Using IPv6 Address Hierarchy with Treemaps

IPv6 address allocation standards dictate that addresses must be assigned in a hierarchical manner, due to the extremely large address space. Assigning IPv6 addresses otherwise (e.g., in an ad hoc way) would lead to huge routing tables on backbone Internet routers.

RFC 3587 [8] specifies a global unicast format for aggregating IPv6 addresses in order to keep global routing tables efficient. In this format bits of the address are grouped together forming a three level hierarchy: the global routing prefix, the subnet ID and the interface ID. Moving down the hierarchy (i.e., reading more bits of the IPv6 address from left to right), reveals more about the IP address. The global routing prefix might tell us what regional registrar owns the address block, while the subnet ID might tell us what country and ISP own the sub-block. Finally the interface identifier will uniquely identify a host or endpoint.

One visualization tool that can represent hierarchy effectively is a treemap [17]. Treemaps have been used in the past for representing hard drive space (by using the file directory structure as hierarchy). One implementation of the treemap algorithm is a Java application written by the HCI Lab at the University of Maryland [15]. The application reads in a text file in a specific format and outputs a treemap that can be navigated with dynamic tree height selection, node filtering and different layout types. We used this tool to illustrate how treemaps can be used to visualize IPv6 remote addresses.

Figure 5 shows an example treemap created from the dataset of Section 4.1 (IPv6-only). The treemap represents all remote IPv6 addresses seen during the 4.5 hour capture period, the number of packets matching each port and their protocol. The size of each rectangle is proportional to the number of packets, and the color represents the protocol. The treemap is hierarchical (using each hextet of the remote IPv6 address as a level) so viewing the first 3 hextets would display only the global routing prefix, and viewing the first 4 would display the subnet ID as well. The user can zoom in as far as the last hextet of the address, with the drawback of a more cluttered treemap. Rectangles that are too small to display a label can be expanded to full screen when clicked. Mousing over any rectangle gives a popup label displaying the packet count.

The treemap algorithm has generated five large rectangles (marked with black borders in Figure 5) corresponding to the first 16 bits of the IPv6 address: 2001, 2A01, 2620 and 2A02 which are IANA assigned prefixes; and FE80 which is a link local block. The analyst can zoom in to the tree by double clicking on any of the labels, at which point the treemap will be redrawn using the selected node as the root of

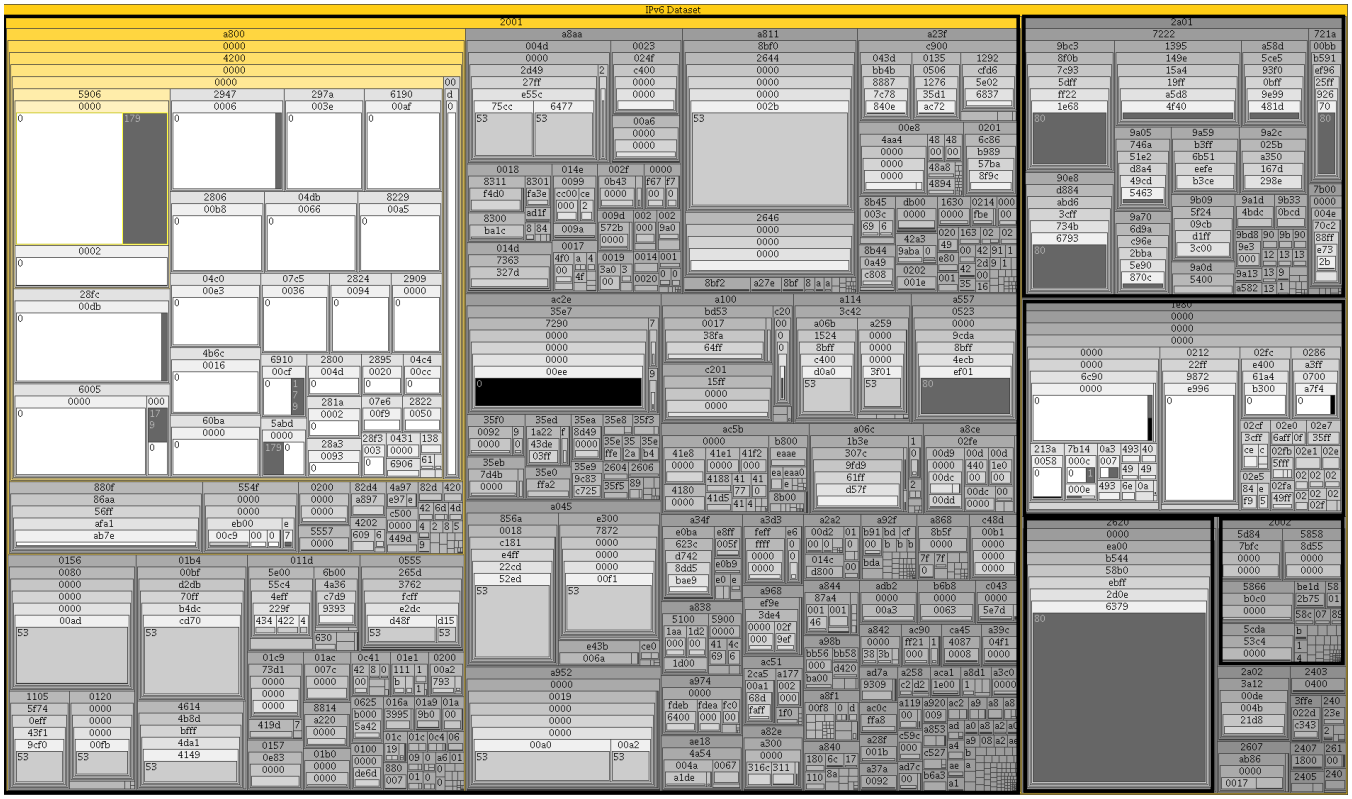


Figure 5: A treemap representing IPv6 source address, destination port and packet count (ICMP (white), TCP (dark grey), UDP (light grey), everything else (black)). Best viewed in color. Full resolution version available at http://bit.ly/vizsec1

the tree. The tree is also searchable which might allow analysts to look for specific prefixes or network blocks. Correlating this data with the list of IPv6 unicast assignments [1] by IANA reveals geographical location of nodes.

One of the benefits of treemaps is that the user can quickly find sources behaving similarly by looking at the contents of nodes. For instance, in Figure 5 to the left, we notice that the highlighted hosts (mostly white nodes in the treemap) were seen sending mostly ICMPv6 traffic during the capture period. Some of these hosts were also seen sending TCP traffic on port 179, corresponding to the Border Gateway Protocol (BGP). These hosts are therefore probably routers or monitoring devices (or devices being monitored).

Another example of using treemaps for detecting specific types of hosts is presented in Figure 6. In this treemap all hosts from our dataset except those seen generating UDP traffic have been hidden. The resulting treemap displays 3 nodes with a distinctive destination port layout. All three nodes show the most common destination port contacted was 32769 UDP (the second ephemeral port available by default on Unix-based operating systems) followed by 53 UDP and then many other high numbered UDP ports. At a first glance, the contents of their nodes would suggest the three hosts behave similarly. This intuition is confirmed by searching for these IPv6 addresses using the SiLK tools, revealing that all three hosts only responded to requests on port 53 during the 4.5 hour capture.

5.4 Limitations

The three previously described techniques have advantages and drawbacks. The first method (removing unused addresses) will work well with datasets that have a small number of IPv6 hosts, and therefore has questionable scalability. The datasets of Sections 4.1 and 4.2 contained only 1924 and 3029 source IPv6 addresses, respectively, which generated uncluttered visualizations. In IPv4, we know that there can be at most 2^{32} remote addresses (actually far less due to reserved addresses) and that network address translation (NAT) devices may hide a large number of hosts behind a single IP address. In IPv6, each Internet enabled device can have its own address, potentially saturating

Figure 6: Finding DNS servers. Best viewed in color. Full resolution version available at http://bit.ly/vizsec2

visualizations created with this technique.

Real-time visualizations can be slow to implement with the first and second methods since they require a list to be sorted each time a new IP address is added. The treemap method of Section 5.3 is faster to update real-time, but requires the user to interact with the tool to reveal hosts with low volumes of traffic (e.g., low and slow scans).

The transition mechanisms described in Section 2.1 will also require the analyst to further investigate points of interest, since source and destination IP addresses might not represent the end device, but rather a tunnel endpoint. Address ranges reserved for transition mechanisms also appear to be more densely populated.

As IPv6 becomes more widely deployed, techniques like that described in Section 5.2 will not scale well. While it is impossible to predict when these techniques might become obsolete, they can be further improved by allowing a many-to-one mapping of IPv6 to IPv4 addresses, and using additional complementary tools to display whether a point on the graph represents a single address or a set. Such methods will allow existing visualization techniques to continue to be used while the transition from IPv4 to IPv6 takes place.

6 Conclusion and Future Work

As IPv6 infrastructure is deployed, attackers are able to use IPv6, and have enabled support for it in their applications. Current visualization techniques fail to offer methods for displaying the source of connections if IPv6 is being used. This allows attackers the opportunity to remain undetected. We have proposed three new techniques that allow analysts using security visualization tools to see IPv6 network traffic, without requiring a major rewrite of the existing tools. While these methods seem to give good results on sample datasets, their practical utility once IPv6 deployment becomes mainstream remains unclear. A main objective has been to raise awareness of the need for security visualization tools to support IPv6 datasets. Future work includes finding new visualizations techniques for IPv6 traffic and improving the current proposed techniques.

ACKNOWLEDGEMENTS

We thank the anonymous referees for their helpful comments. The second author acknowledges NSERC funding under a Discovery Grant and as Canada Research Chair in Internet Authentication and Computer Security. Partial funding from NSERC ISSNet is also acknowledged.

REFERENCES

[1] IANA IPv6 Unicast Address Assignments. http://www.iana.org/assignments/ipv6-unicast-address-assignments Last updated 2008-05-13.

[2] Measurement and Analysis on the WIDE Internet (MAWI) Working Group Traffic Archive. http://tracer.csl.sony.co.jp/mawi/.

[3] SiLK Tools. http://tools.netsa.cert.org/silk/.

[4] The IPv6 Attack Toolikit. http://freeworld.thc.org/thc-ipv6/.

[5] CAIDA. Visualizing IPv6 AS-level Internet Topology 2008. http://www.caida.org/research/topology/as_core_network/ipv6.xml, Accessed April 20, 2009.

[6] S. Deering and R. Hinden. RFC2460 - Internet Protocol, Version 6 (IPv6) Specification. http://www.faqs.org/rfcs/rfc2460.html, Dec. 1998.

[7] S. H. Gunderson. Measuring the current state of IPv6 for ordinary users. http://bit.ly/QTEXA, October, 2008.

[8] R. Hinden, S. Deering, and E. Nordmark. RFC 3587 - IPv6 Global Unicast Address Format. http://www.faqs.org/rfcs/rfc3587.html, August 2003.

[9] G. Huston. IPv4 Address Report. Retrieved on November 15, 2009 from http://www.potaroo.net/tools/ipv4/index.html.

[10] S. Kirkpatrick, M. Stahl, and M. Recker. RFC 1918 - Address Allocation for Private Internets. http://www.faqs.org/rfcs/rfc1166.html, July 1990.

[11] H. Koike, K. Ohno, and K. Koizumi. Visualizing cyber attacks using IP matrix. In *IEEE Workshop on Visualization for Computer Security, 2005. (VizSEC 05)*, pages 91–98, Oct. 2005.

[12] K. Lakkaraju, W. Yurcik, and A. J. Lee. NVisionIP: Netflow visualizations of system state for security situational awareness. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pages 65–72, New York, NY, USA, 2004.

[13] J. McHugh. Data Structures for IPv6 Network Traffic Analysis Using Sets and Bags. In *Proc. of FloCon 2009*, Scottsdale, AZ, USA, 2009.

[14] H. Moore. Uninformed.org - Exploiting Tomorrow's Internet Today: Penetration Testing with IPv6. http://www.uninformed.org/?v=10&a=3, October 2008.

[15] U. of Maryland Human Computer Interaction Lab. Treemap: Home page. http://www.cs.umd.edu/hcil/treemap/.

[16] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, and E. Lear. RFC 1918 - Address Allocation for Private Internets. http://www.faqs.org/rfcs/rfc1918.html, February 1996.

[17] B. Shneiderman and M. Wattenberg. Ordered treemap layouts. In *Proceedings of IEEE Symposium on Information Visualization, 2001. INFOVIS 2001*, pages 73–78, 2001.

[18] J. M. Shu Nakamae, Yuji Sekiya. A Study Into a Visualization of an IPv6 Network. In *Proceedings of the 9th Annual Conference of the Internet Society INET'99*, 1999.

[19] T. Taylor, D. Paterson, J. Glanfield, C. Gates, S. Brooks, and J. McHugh. FloVis: Flow Visualization System. In *Conference For Homeland Security. CATCH '09. Cybersecurity Applications & Technology*, pages 186–198, March 2009.

[20] J.-P. van Riel and B. Irwin. InetVis, a visual tool for network telescope traffic analysis. In *Afrigaph '06: Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 85–89, New York, NY, USA, 2006. ACM.