Header Steganography

Abusing IPv6 Headers & Prefix Delegation for Data Exfiltration

CackalackyCon 2019

Before We Dive In…

Standard disclaimers apply, especially the one about using this information for illegal shit.

CackalackyCon is open to attendees of all skill levels, so I tried to include information for the beginners just breaking into security/networking all the way up to veterans in the field.

Questions are welcome,
but *please* hold them until the end.

SYN

- New Orleans area native (504/985)
- Worn many hats: Datacenter Engineer, Internet Engineer, CMTS Engineer, Systems & Network Administrator
- Day Job: Sr Cyber Security Analyst
- Work with several animal rescue organizations, certified pilot, and nature photographer
- ISO 8601 Compliant
- CISSP (Don't Judge Me…)

@__wavelength__

darkwavelength@protonmail.ch

Data Exfil…. What?

Data Exfiltration

The unauthorized copying or transfer of data from an environment. Also known as data theft, data exportation or data extrusion.

Motivations

Espionage (Industrial or State-sponsored)

Blackmail / Extortion

Financial Crimes - identity theft for financial or medical fraud

Hacktivism - Wikileaks, Panama Papers, Ashley Madison, etc.

Intelligence gathering - law enforcement, three-letter agencies

Lulz and other, unclear, motives - “The Fappening”, etc.

Penetration testing, CTF, etc.

Data Exfiltration Motivations:
-Espionage: State sponsored activities include government operations to steal information on strategic military information, like troop movements or weapon systems, intel products, communiques, and other information from foreign governments. Industrial espionage activities include foreign corporations attempting to steal intellectual property from domestic competitors to achieve a competitive advantage or copy products
-Financial Crimes: This has become the most common motivation. This includes incidents like the Equifax or Target breach where personal information was stolen in order to open credit and other accounts for financial gain. Medical record theft is used to commit Medicare and Medicaid fraud in the US by receiving services in the stolen identity’s name.
-Intelligence gathering: Differs from espionage activities in that the information is focused on gathering information on people, groups or governments instead of stealing competitive or strategic information. This would include surveillance on terrorist groups like Al Qaeda, ISIS or the Taliban; surveillance of criminal groups such as drug cartels, organized crime syndicates, gangs, etc.; and surveillance of other groups or people who have or are making credible threats to people or property. The information gathered here centers on current operations, future plans, communications channels and other members.

Data exfiltration has existed as long as mankind has had adversarial relationships, long before we had computers and networks to connect them. For example, if you have ever watched old war films or spy movies, operatives would sneak around with microfilm cameras capturing images of documents and plans. This is an example of analog data exfiltration. This requires a threat actor to have physical access to the data they are tasked with stealing and, as such, involves significant risk in the form of possibly being captured, interrogated, imprisoned or killed.

With the rise of digital data, exfiltration techniques evolved. Data is now stored in digital formats, so making copies of data became much easier. With a floppy disk, removable optical media, or, as is common today, USB drive, one merely “drags and drops” the data to the media and walks out the door with a perfect copy. Like the older analog methods, this still requires physical access to where the data is stored and significant risk to the threat actor.

Network-based Methods

The Obvious Protocols

FTP File Transfer Protocol - tcp/20 & tcp/21

SFTP Secure File Transfer Protocol - tcp/22

HTTP HyperText Transfer Protocol - tcp/80

HTTPS Secure HyperText Transfer Protocol - tcp/443

SMTP Simple Mail Transfer Protocol - tcp/25

POP3 Post Office Protocol version 3 - tcp/110

As the world became connected through the Internet, data exfiltration methods evolved again.

Now a threat actor can stay in safety away from their target without the threat of discovery, capture or being killed. Even if they are identified, depending on where they are located, it is unlikely that they would be extradited, especially if they are operatives or allied assets with the government in their current location.

The protocols listed here are the most obvious protocols for a threat actor to transfer data out of their victim's environment. FTP and SFTP are well suited to data transfers since they are designed specifically for that purpose. HTTP and HTTPS are also well suitable to transferring large amounts of data.

SMTP and POP3 can also transfer data. The threat actor could create throw away email accounts on any of the many free email services, send a message to an address they control with data attached to an email and retrieve it at their convenience.

Network-based Methods

More Exotic Exfiltration Methods

DNS

Domain Name Service - udp/53 & tcp/53

Exfiltrated data is placed in the query portion of a DNS request to a domain under the adversary's control, where it is logged and reassembled

ICMP

Internet Control Message Protocol

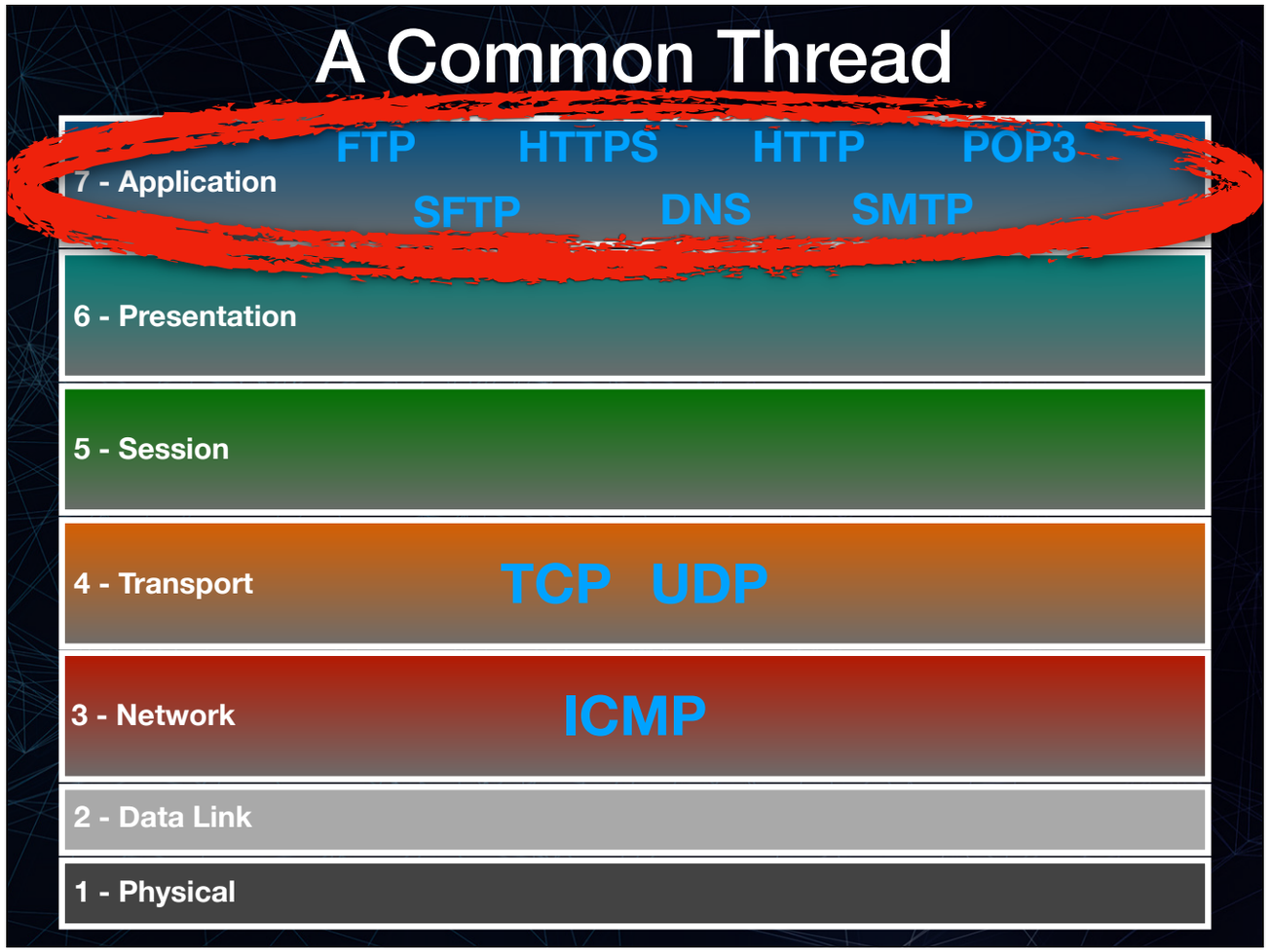
Exfiltrated data is placed in the data field of an ICMP Echo Request.

TCP

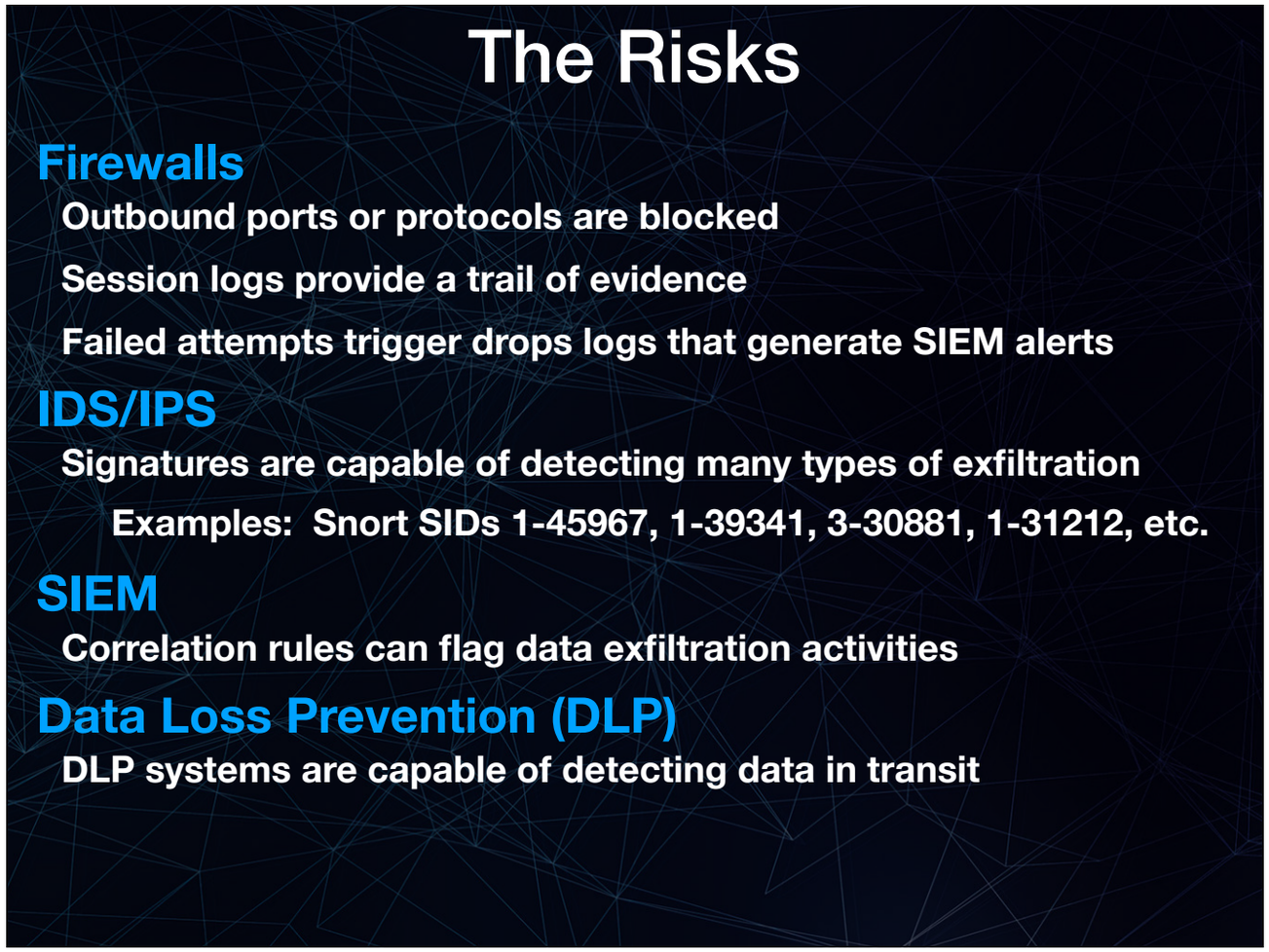
Transmission Control Protocol headers

Exfiltrated data is placed in the port or sequence number field (4 bytes) of a TCP packet.

As ways of detecting data exfiltration improved, threat actors needed to develop new covert channels that were more difficult to detect. This is a list of the most common of those methods with explanations of how they function.



If we look at the protocols discussed, we notice a common thread among them - they cluster in the higher layers of the OSI model. The advantages of using the application layer protocols is that they are designed to carry data well, but using them comes with risks.



A few of those risks include:

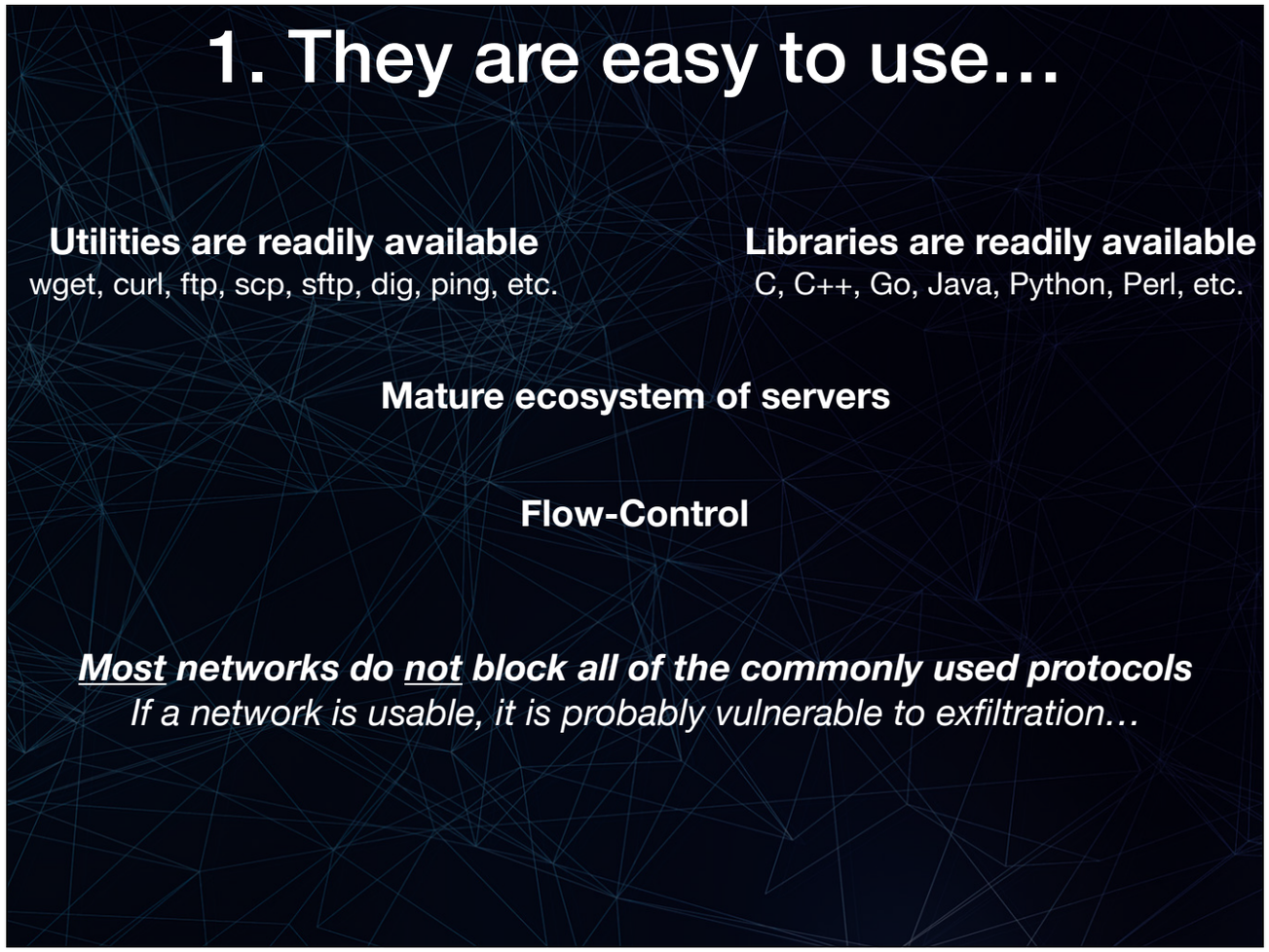
Firewalls: Firewalls provide network and security engineers with the ability to apply policies to their networks by allowing or blocking protocols. When those protocols or ports are blocked, this can interfere with exfiltration. Those failed attempts can trigger packet drop logs that can alert SOC analysts to suspicious activity within the network. Even in cases where those ports and protocols are allowed, session logs can generate a trail of evidence that a SOC can use during incident response.

IDS/IPS: IDS/IPS signatures are capable of detecting many different types of exfiltration activity. The list of Snort signatures shown here are examples of rules capable of detecting data exfiltration.

SIEM: SIEM correlation rules are capable of detecting not only exfiltration traffic, but can also detect the activities leading up to the actual transmission of data. For example, suspicious or unauthorized data access on properly monitored systems can be detected.

Data Loss Prevention: These systems are all capable of monitoring the common protocols, especially HTTP/HTTPS, FTP, SMTP and POP3 for sensitive or proprietary data.

So, why do threat actors rely on the commonly used protocols?



First, for the most part, threat actors are lazy - they will expend the minimum effort required to achieve their objectives.

If we look at most operating systems, they all have tools that are capable of using these protocols to send and receive data. Why write new tools when existing tools can do the job?

If a threat actor is writing their own custom malware, then there are libraries for just about every programming language in use. This reduces the effort required to enable network communication without having to write custom network protocols.

There is a mature ecosystem of servers available for common operating systems that allow threat actors to also receive data at the target. Want to use FTP? There are dozens of server packages available. The same is true for all of the other common protocols.

Flow-control is already built into many of these protocols, meaning threat actors do not have to devise two-way mechanisms to detect and mitigate data loss in transit.

And finally, unless the victim is a high security environment, like the military or a government agency, it is rare for all of the common protocols to be blocked. So if the average user is able to use the network, for example, to browse the web or check Facebook at lunch, then that network is likely vulnerable to exfiltration. In the example provided, the network would be vulnerable to exfiltration using HTTP or HTTPS.

2. Space

In IPv4, further down the OSI model, less space for data…

IPv4 Header

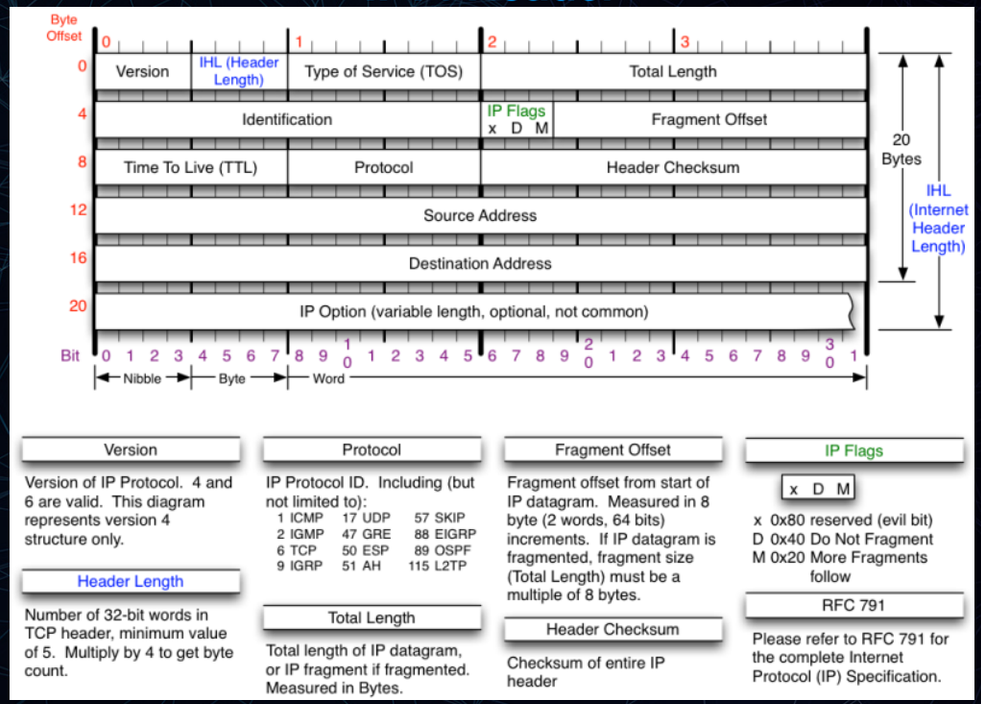


Image Source: nmap.org

Next, as we move down the OSI model, there are less places to put data.

This is the IPv4 header. We have many fields in the header, but few if any places to put data. Many fields have clearly defined formats that prevent arbitrary data from being placed in them, like the version, IHL, IP flags and protocol fields. Other fields hold calculated values that will cause the packet to be dropped if certain protocol checks and verifications fail; these fields include the total length and header checksum fields.

The TTL field is so small that little data could be placed in it and if certain values are used, could cause data loss. This is the field that determines how many routers, or hops, the packet can be forwarded through before it is dropped. So, for example, if the encoded data was a small value, such as 4 or 5, then the packet might not make it to the destination.

Both the source and destination address fields have to be valid for the packet to travel from source to destination, so arbitrary data can not be placed in them either.

And finally, use of the IP Option field is uncommon and may trigger alerts in some monitoring tools.

TCP Header

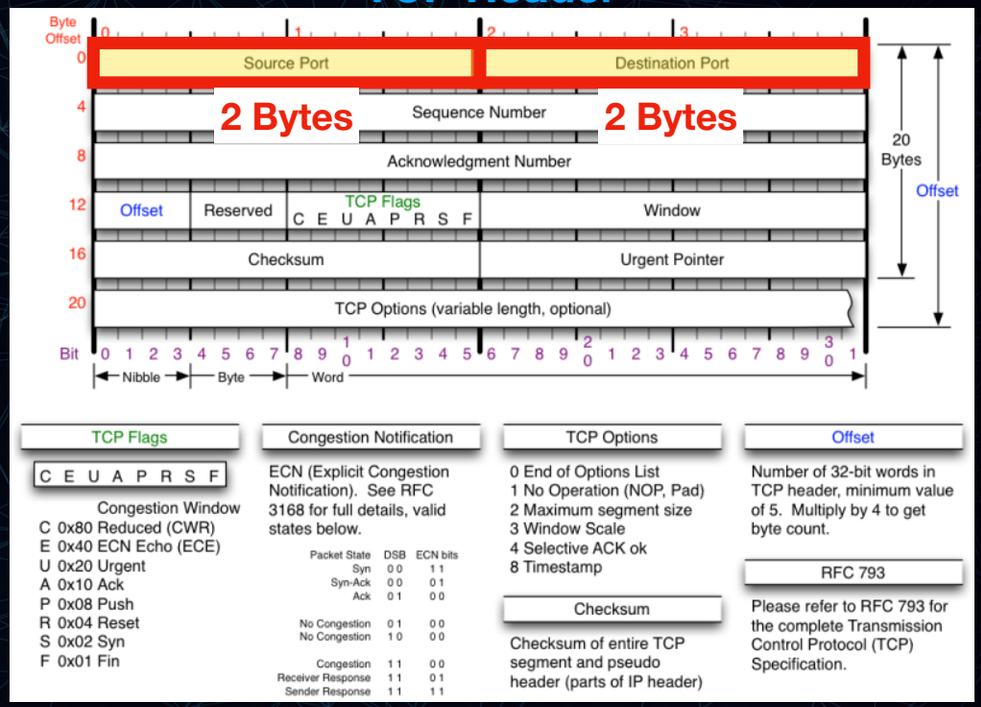


Image Source: nmap.org

The TCP header gives a threat actor some options for embedding data.

The Source and Destination Port fields are capable of holding two bytes of data.

TCP Header

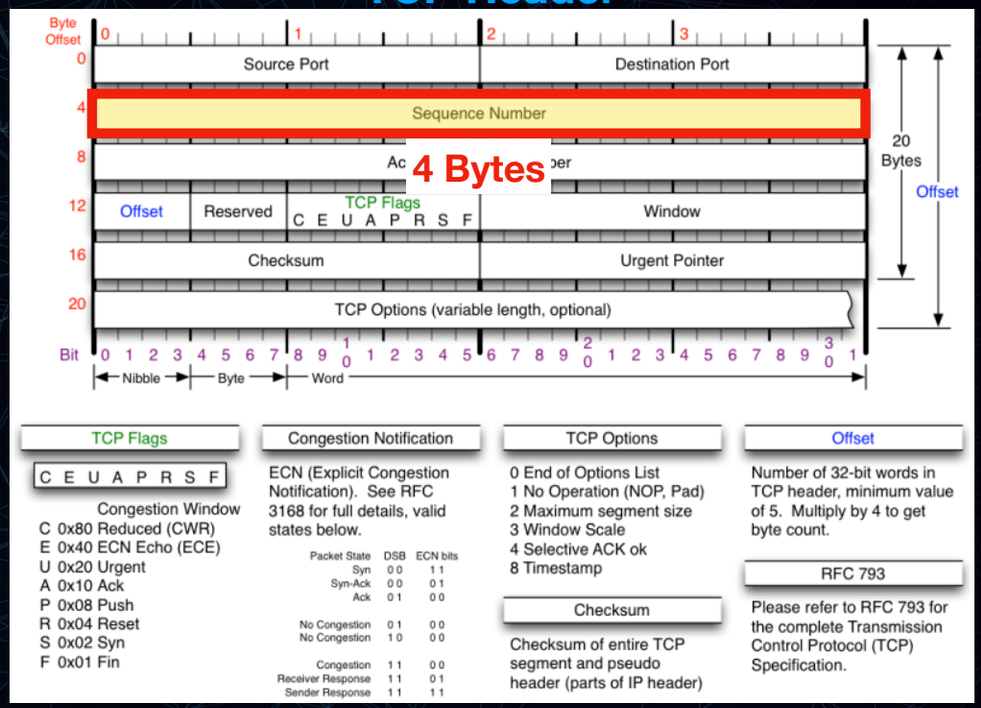


Image Source: nmap.org

The Sequence Number field can hold 4 bytes of data.

TCP Header

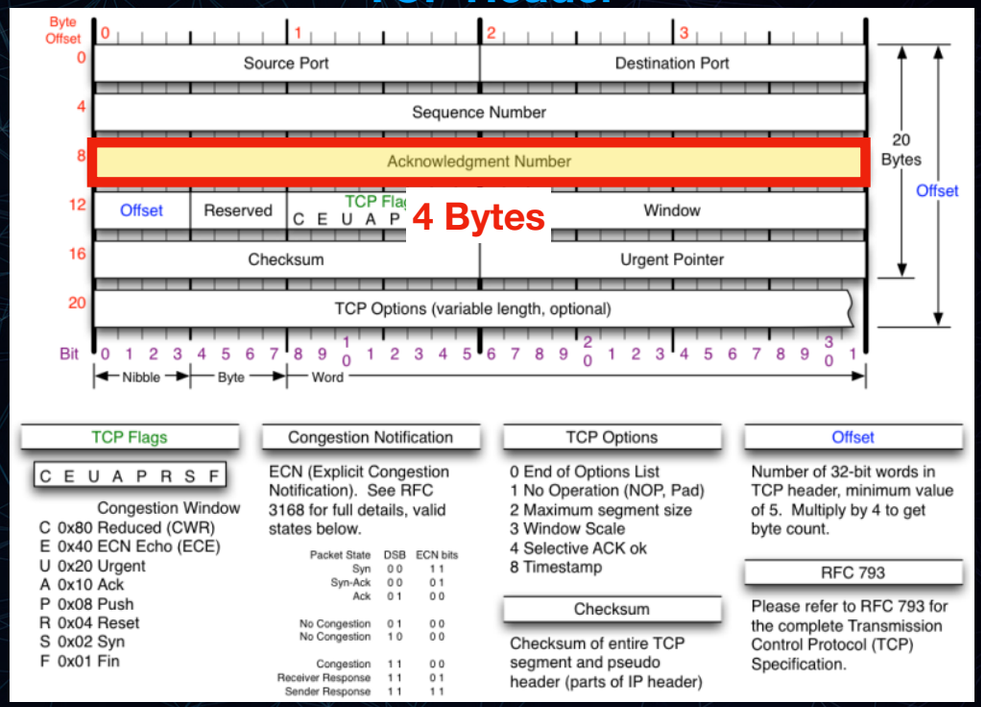


Image Source: nmap.org

And the Acknowledgement Number field is also capable of holding 4 bytes of data.

TCP Header

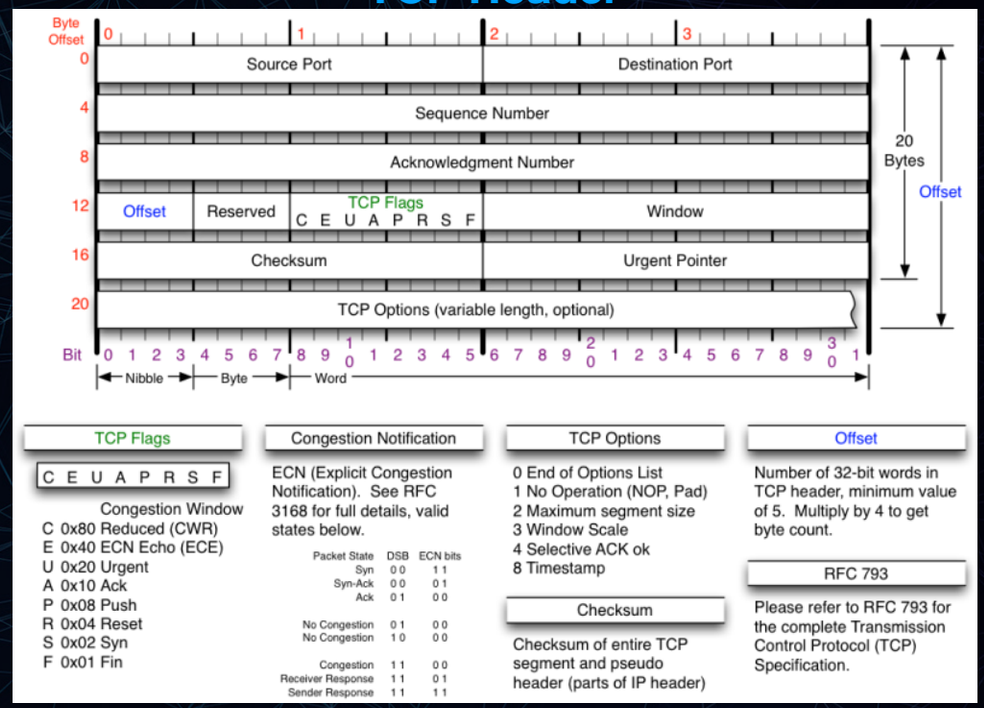


Image Source: nmap.org

The Offset, TCP Flags and Checksum fields can not carry arbitrary data. The Window field could carry arbitrary data, but using this field would be problematic because the value in this field directly affects the operation of TCP at both ends of the connection. The Urgent Pointer field could also carry arbitrary data, but the use of this field is uncommon.

UDP Header

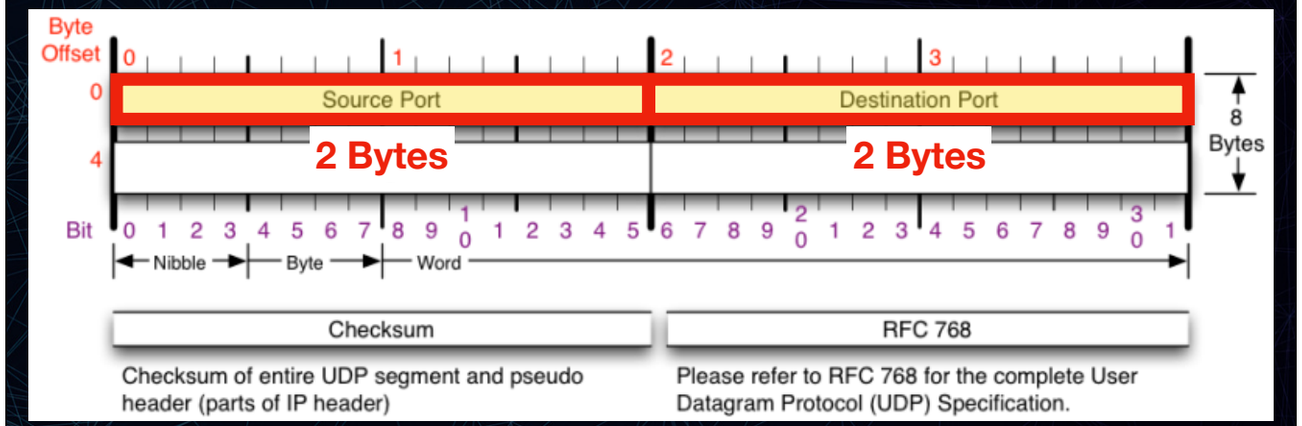


Image Source: nmap.org

Like the TCP header, the UDP header allows for 2 bytes of data in the Source and Destination Port fields.

UDP Header

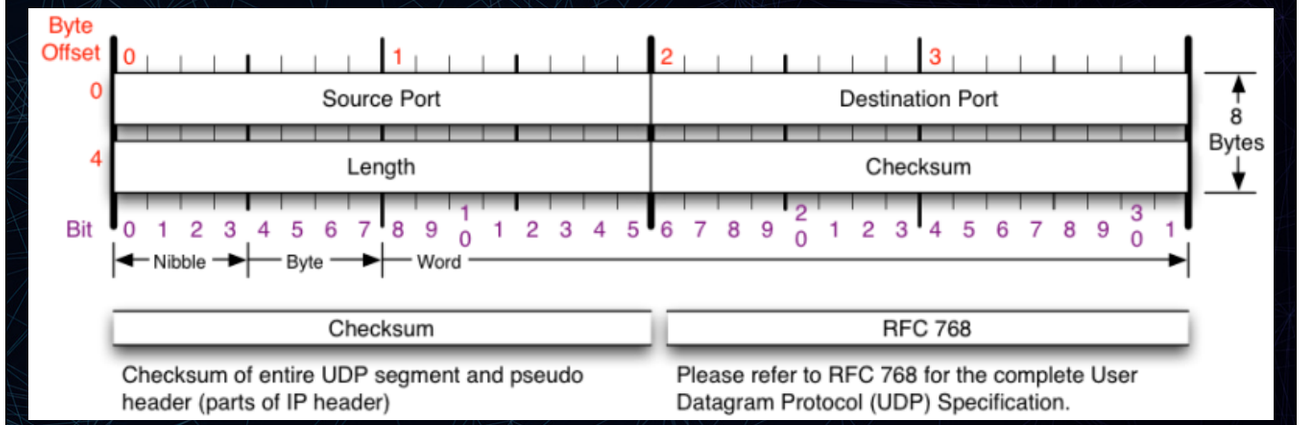
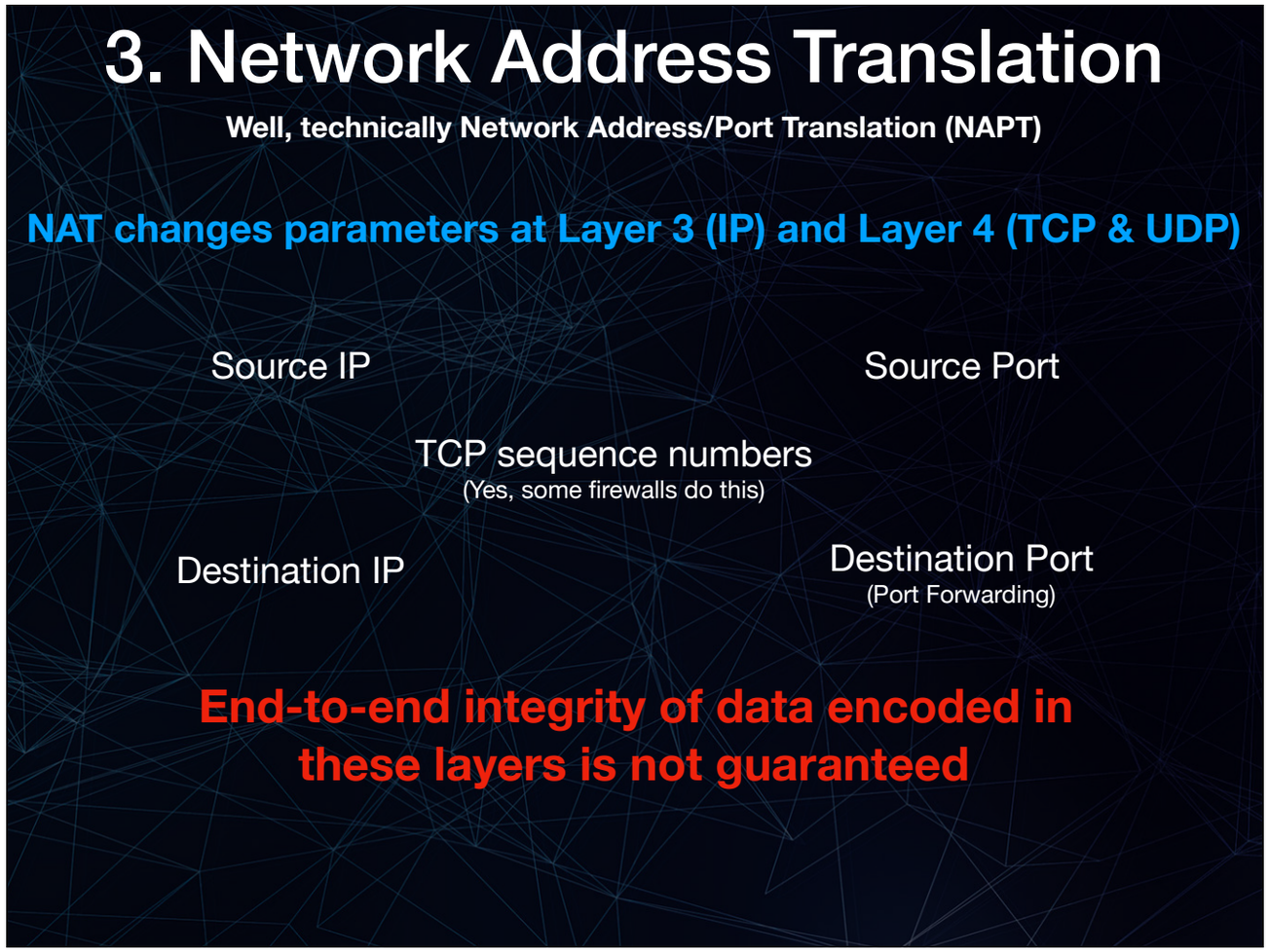


Image Source: nmap.org

The Length and Checksum fields are both calculated fields and can not carry arbitrary data.



The third reason that threat actors rely on protocols in the high layers of the OSI model is Network Address Translation or NAT. Well, technically, Network Address and Port Translation. As you will see shortly, NAT operates by changing parameters in both Layer 3 and Layer 4. For example, the source and destination IPs can change. Source and destination ports can also be changed by NAT. Even TCP sequence numbers can change as packets flow through some firewalls.

For this reason, the integrity of data secreted away in any of these fields can not be guaranteed end-to-end.

So why do we tolerate networks changing data in transit?

Some Historical Context

IPv4 was formalized in 1981,
but, by the early ‘90s, it was clear there was a problem.

Address space exhaustion!

32 bits of address space is 4,294,967,296 addresses, but…

Addresses reserved for Multicast:	268,435,424
Addresses reserved for “Future Use”:	268,435,424
Addresses tied up in “Legacy” /8 assignments:	~300,000,000
Addresses reserved for the “Loopback”:	16,777,216
Addresses reserved for “Private Addressing”:	~26,200,000

Leaving ~3.4 Billion addresses…

To understand that, we need some historical context.

When IPv4 was formalized in 1981, there were only several thousand machines connected to what would become the Internet, but in the 1990s the Internet grew faster than expected and a major problem was on the horizon. With thousands of hosts connecting monthly, even daily, it was clear that the Internet would run out of IP addresses in the future.

IPv4 allowed for approximately 4.3 billion systems to be connected to the Internet because the address space is 32 bits wide. However, due to a number of special use reservations, including the Multicast reservations, legacy /8 network assignments and private addressing, the useable pool of Internet addresses was only about 3.4 billion addresses.

Seems like a lot…

Earth's Population in 1995 - 5.7 Billion

Earth's Population in 2006 - 6.7 Billion

Number of connected devices in 2009 - 4.8 Billion (est.)

Cell Phones in the world in 2015 - 4.5 Billion

Number of connected devices in 2020 - 25-75 Billion (est.)

Which seems like a lot of addresses, but if we look at several statistics, we realize it was not.

For example, in 1995 there were already more people on Earth than IP addresses. If everyone had one device connected to the Internet, then 3.4 billion addresses would be insufficient.

By 2009, with an estimated 4.8 billion Internet connected devices, we had exceeded the total capacity of IPv4.

And by 2020, with estimates as high as 75 billion devices on the global Internet, there is no longer any doubt that IPv4 would be insufficient to deal with the long term growth of the Internet.

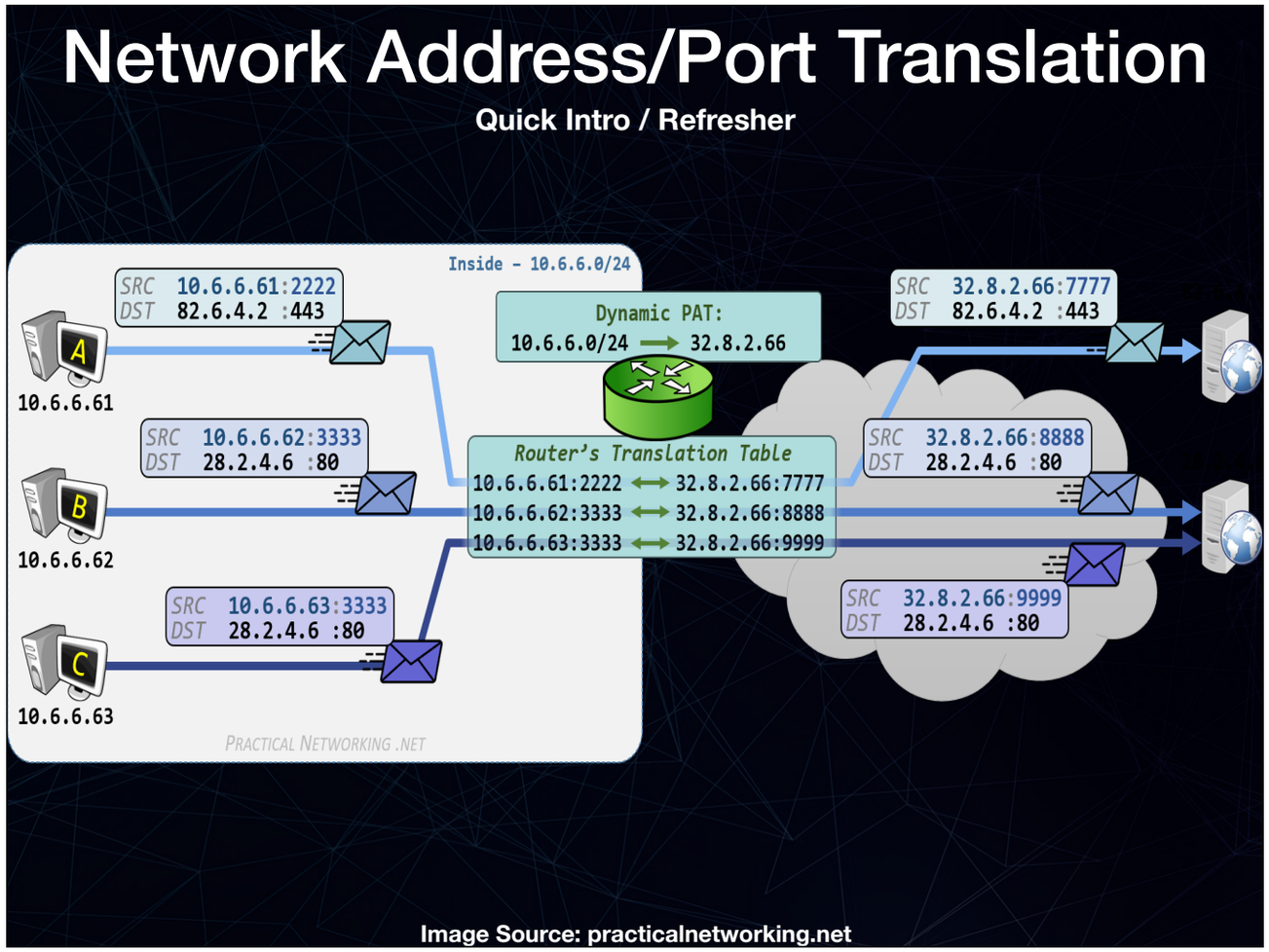


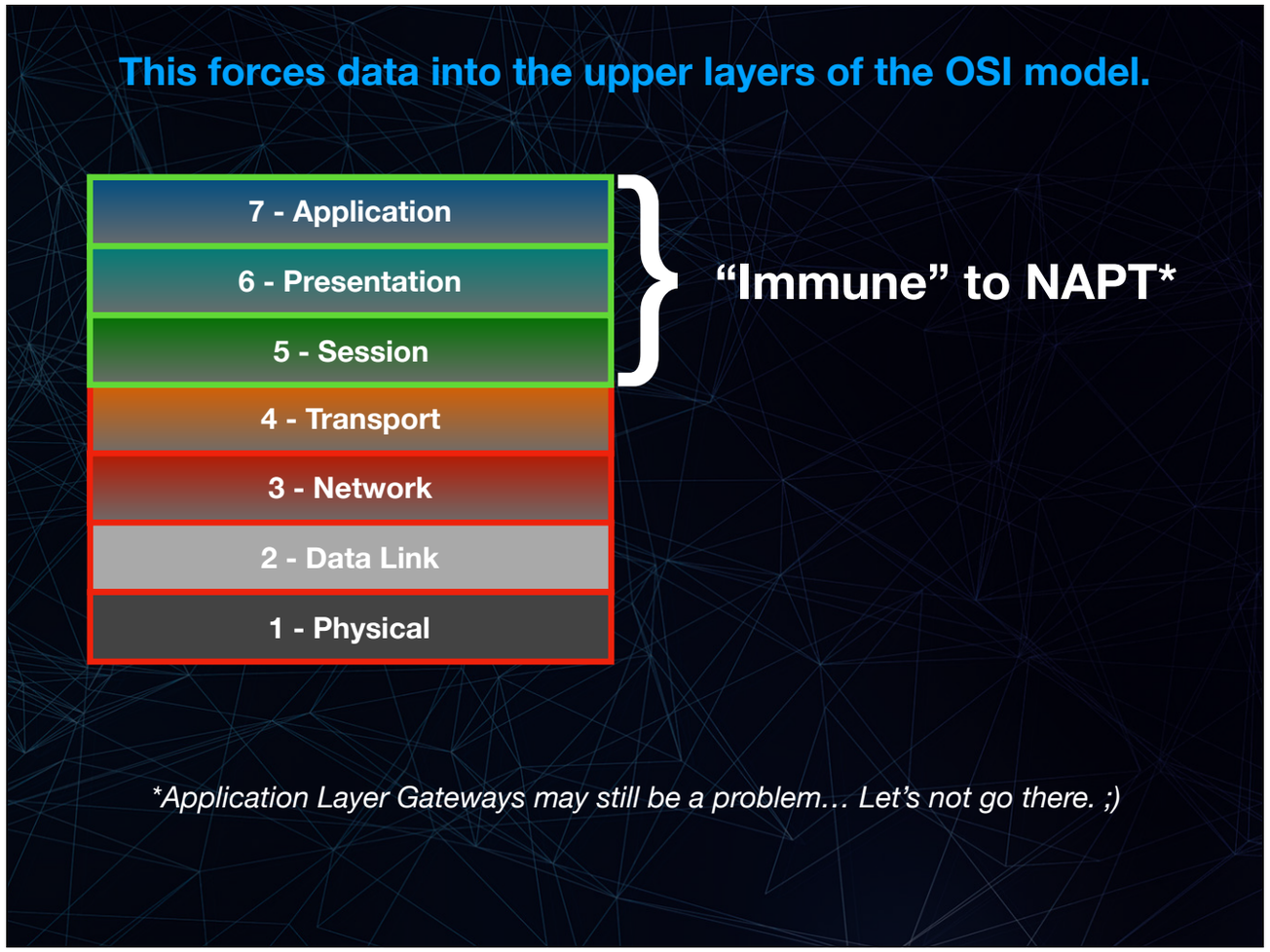
Image Source: practicalnetworking.net

So we needed a work around and NAPT was developed. How does NAPT work?

As we can see in the diagram, hosts A, B and C are all using one of the familiar private IP subnets in the 10-network, in this case 10.6.6.0/24. They are connected to their default gateway which has a single globally routable address of 32.8.2.66. As packets leave the network, the source address is translated from a private address to this public address. When packets return, the destination address is translated back to the private address. To allow multiple systems to using the same public address, the port is used to track which system is initiating the connection, as indicated by the "Router's Translation Table." This allows the router to not only track multiple system sharing the public address, but multiple conversations from those systems to other systems on the Internet.

In this diagram, we can see that not only does the source/destination address change, but the port numbers can also change. Note that Host A's packet has a source port of 2222, but when it leaves the router, the source port is 7777. When return traffic comes back destined for port 7777, the router knows to which system that packet should be sent on the internal network. That packet is then sent to the correct host which is listening on port 2222.

This illustrates why encoding data in Layer 3 and Layer 4 fields results in data loss at the network perimeter when NAT is used, which is most networks.



And so, threat actors are forced to secret data away in the upper layers of the OSI model since they are “immune” to the effects of NAT.

There are Application Layer Gateways and Server Load Balancers that can mangle data in the upper layers as well, but they are out of the scope of this presentation.

So, what if we lived in a world without NAT?

Well, we do live in a world without NAT (or soon will be). Introducing IPv5… I mean, IPv6.

Hacker jeopardy knowledge, the v4 and v6 do not refer to versions technically, but to the IP Protocol Number. IPv4 is protocol number 4. In between the finalization of IPv4 and IPv6, a protocol called the Internet Steam Protocol was assigned protocol number 5. This is why there was the jump from IPv4, over 5, to IPv6.

The Wikipedia article linked in the slide discusses the Internet Stream Protocol.

History of IPv6

1990 - Initiation of the effort to define the replacement for IPv4

1992 - The first proposals for IPv4 replacement start appearing

1995 - RFC 1883 defines the first draft of an IPv6 “standard”

1998 - RFC 2460 supersedes RFC 1883

2017 - RFC 8200 supersedes RFC 2460

The history of IPv6 begins in 1990 when the effort to replace IPv4 was initiated. Within two years, the first experimental protocol work was underway and in 1995 IPv6 was formalized in RFC 1883. Two additional RFCs were drafted, the latest being published in 2017 with RFC 8200.

Some of the changes in IPv6

(Not an exhaustive list…)

WARNING: Just a few walls of text ahead…

Changes from IPv4

1. Address space quadrupled to 128 bits from 32 bits

IPv4: 4,294,967,296 addresses

IPv6: 2^{128} or 3.4028237×10^{38} addresses

340,282,366,920,938,463,463,374,607,431,768,211,456

Just for comparison…

Galaxies in the observable universe: 1.0×10^{10}

Stars in the average galaxy: 1.0×10^{11}

Stars in the universe: 1.0×10^{21} or 1 billion trillion

1,000,000,000,000,000,000,000 stars

Number of grains of sand on earth (est.): 7.5×10^{18}

7,500,000,000,000,000,000 grains

If we look at the grains of sand comparison, we would be able to assign each grain of sand its own /64 subnet and still have address space left over.

Changes from IPv4

2. Address format changed

IPv4: decimal digits in a “dotted quad” format

Examples: 10.100.1.1, 4.2.2.2, 204.251.13.1, etc

IPv6: eight, colon-separated hexadecimal groups

Examples:

fe80::289:1f1:a010:8b38

2a03:2880:f111:83:face:b00c::25de

2607:f8b0:4002:808::200e

2600:1402:f000:79e::1aca

Note that none of the example IPv6 addresses contains all eight hex groups. IPv6 allows you to shorten an address by abbreviating contiguous zeros with double colons, but only one time in an address.

For example, in the address fe80::289:1f1:a010:8b38, there are double colons and only five hexadecimal groups shown. This means that between the two colons, there were three consecutive groups of zero. If there was a second group of consecutive zeros, they would have to be written out, however, those can be truncated down to a single zero. An example of that would be fe80::289:0:0:8b38.

Changes from IPv4

3. Subnetting changes

IPv6: like IPv4, subnets work the same…

But the /64 bit subnet is special. This is the smallest subnet recommended when hosts (PCs, phones, etc.) are attached to it.

Why?

Key IPv6 features expect host-attached networks are configured as /64:

Extended Unique Identifier (EUI-64) - allows host to automatically configure valid IPv6 address using NIC MAC address (v6 equiv. of RFC 3927)

Neighbor Discovery Protocol (NDP)

Router Advertisement (RA) - router advertises local subnet to hosts allowing them to use EUI-64 to automatically configure an address

Neighbor Discovery (ND) - neighboring systems advertise their link-local addresses to each other (v6 “equivalent” of ARP and RARP)

Like IPv4, IPv6 subnetting works the same, just with larger subnet masks. There is one important distinction in that the /64 subnet is special. In IPv4, you could attach as many or as few devices as you wanted to a network and adjust the subnet mask accordingly. In IPv6, the /64 subnet is the shortest recommended subnet when hosts - laptops, desktop, tablets, phones, streaming devices, etc. - are connected to it.

The reason for this is that several IPv6 specific protocols operate on the assumption that end-user devices are on no smaller than a /64 subnet. Extended Unique Identifier, or EUI-64, is one such protocol. Like the RFC 3927 protocol, which generates 169.254.x.x addresses on a IPv4 network when there is no DHCP server, EUI-64 allows IPv6-enabled hosts to auto-configure an address using their NICs MAC address. Because MAC addresses are supposed to be globally unique, there is supposed to be no chance of an address collision.

The Network Discovery Protocols also expect the subnet length to be no shorter than a /64. The Router Advertisement protocol is used by IPv6 enabled routers to advertise the subnet configured on a network segment. The Neighbor Discovery protocol is the equivalent of ARP and RARP protocols in IPv4.

Changes from IPv4

4. Prefix Delegation (PD)

IPv4: no equivalent feature; not enough addresses

Customers assigned a single IPv4 address (NAPT)

IPv6: scaleable way to assign subnets to customers

1. Customer assigned /64 subnet for their network
2. Customer assigned < /64 subnet for their network

Cable Provider X - Assigns /60 to residential customers

Equivalent to *16 x /64 networks*

Cable Provider Y - Assigns /56 to residential customers

Equivalent to *256 x /64 networks*

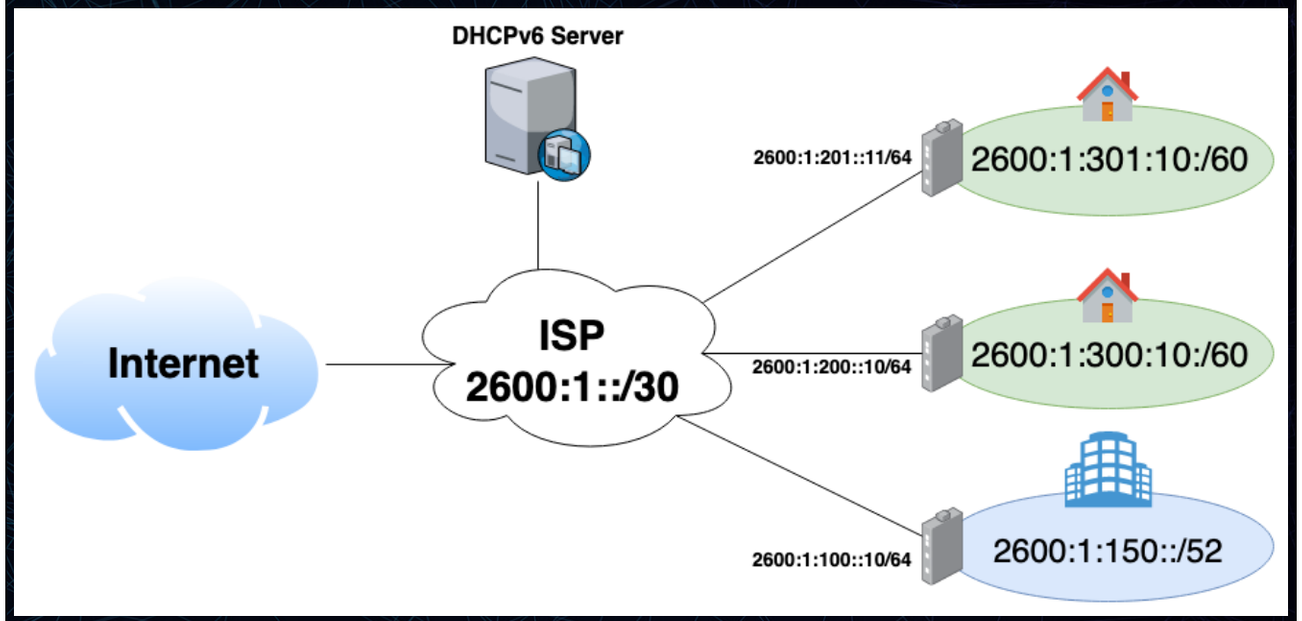
Prefix Delegation, or PD, has no matching protocol in IPv4. DHCP in an IPv4 network hands out IPs. Originally designed to be used within networks to give hosts an address when they connected, it was later used in service provider networks to hand out addresses to subscribers. DHCP was only designed to assign a single address to a host in an IPv4 network since a host only needs one address to operate.

Because IPv6 requires a minimum subnet length of /64 for subnets with end-user devices attached, /64 subnets need to be issued. PD provides the means to do that. In PD, the router will request a subnet from the DHCPv6 server, which will issue a subnet to the router. The router can then assign a /64 subnet or subnets from that delegated prefix to the various networks behind it where the customer's devices reside.

Providers can assign either a /64 subnet per customer or a shorter prefix to enable multiple subnets to be used in the customers network for different VLANs.

Prefix Delegation Illustrated

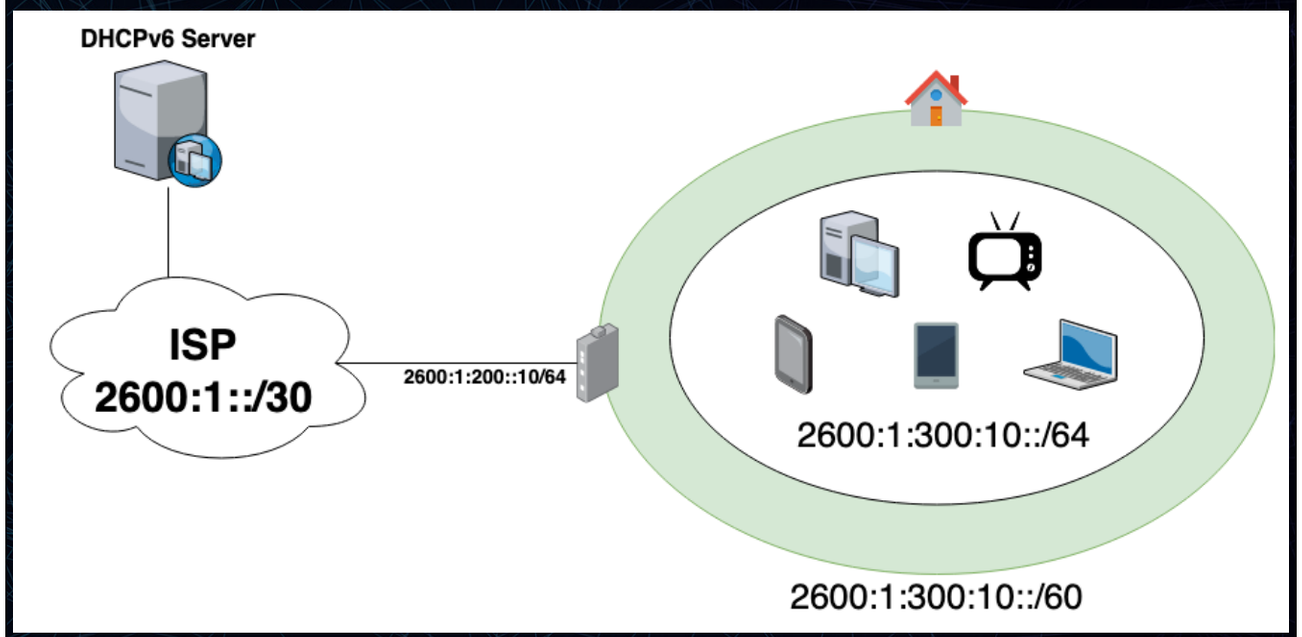
Overview



Here is a graphical example of Prefix Delegation at work. The ISP has a /30 prefix it has been assigned. Out of that network, smaller prefix delegations are assigned to customers to use in their networks. The residential customers are each assigned a /60 subnet. The business customer has a /52 prefix delegation assigned to it.

Prefix Delegation Illustrated

Typical Site



In a typical residential environment, most customers use only one /64 out of a PD assignment, so there is dormant IPv6 space in those locations.

I would be remiss…

Bear with me for just one moment…

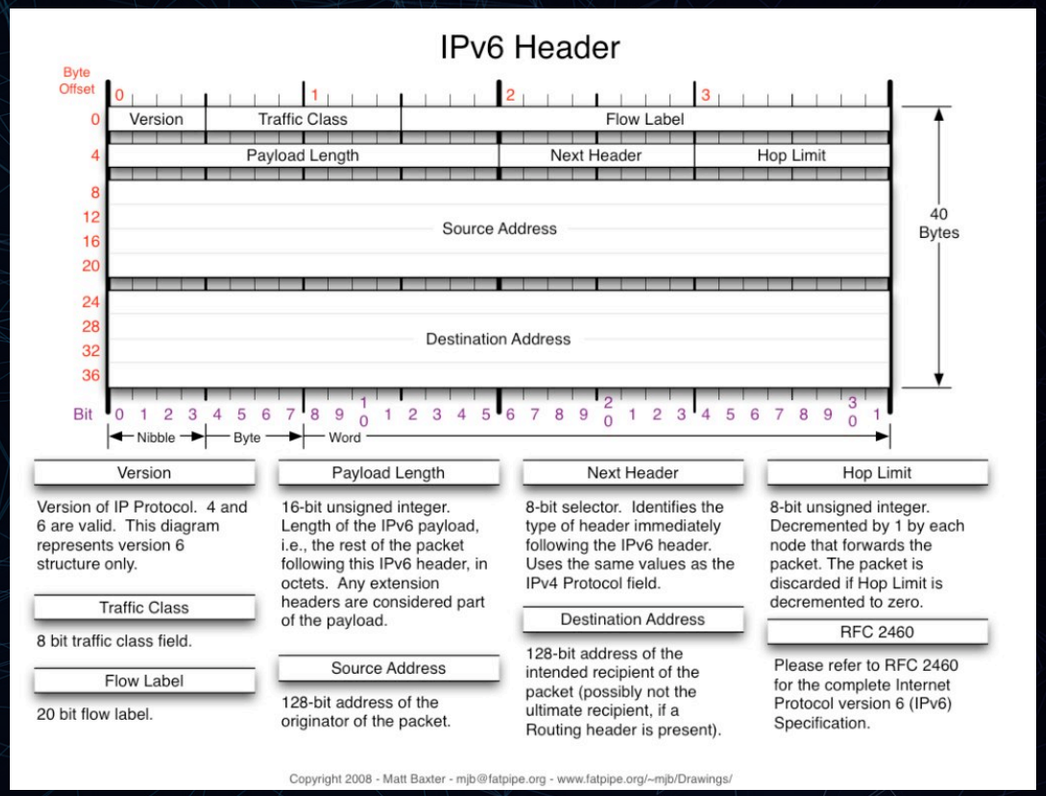
At this point, I have to mention that I often hear that since IPv6 lacks NAT, it is less secure than IPv4.

This statement is false. NAT is not and never was a security feature and is strictly an address conservation feature.
If you are relying on NAT for security, you are doing it wrong. Because IPv6 addresses are globally routable, this forces everyone to secure their networks properly through the use of access control lists, firewall inspection features, etc.

Now there is nothing about IPv6 that prevents the use of the previously discussed data exfiltration methods. All of them will still work.

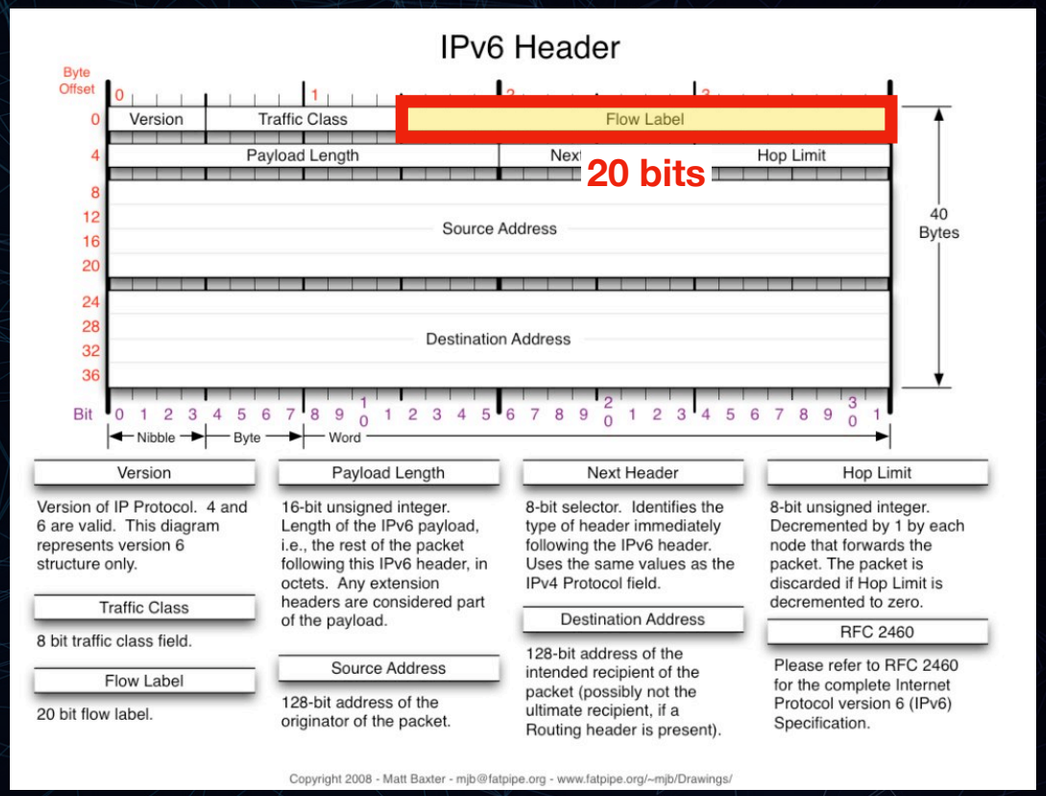
However, due to the changes in IPv6, threat actors have new methods available to them for data exfiltration in an IPv6 environment.

Abusing the IPv6 Header



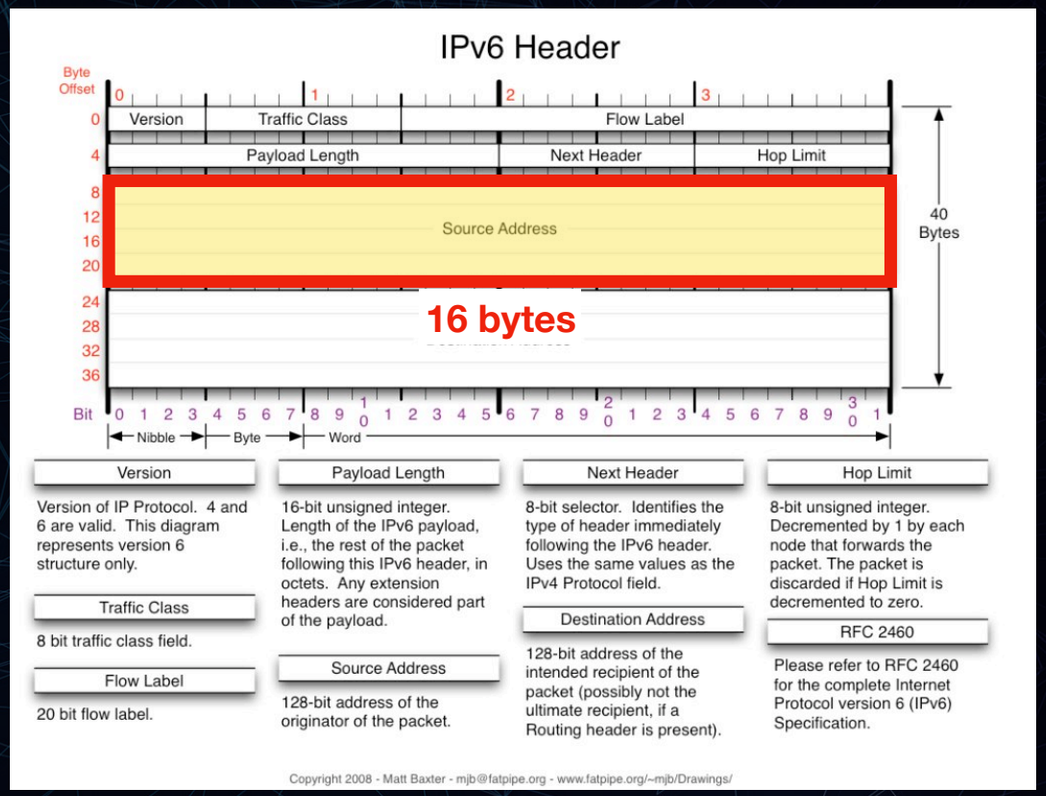
This is the IPv6 header. It is significantly different from the IPv4 header, not only in that the address fields are 128 bits, but that the number of fields has been reduced to streamline it. But, this new header format does offer new places into which a threat actor could secret away data.

Abusing the IPv6 Header



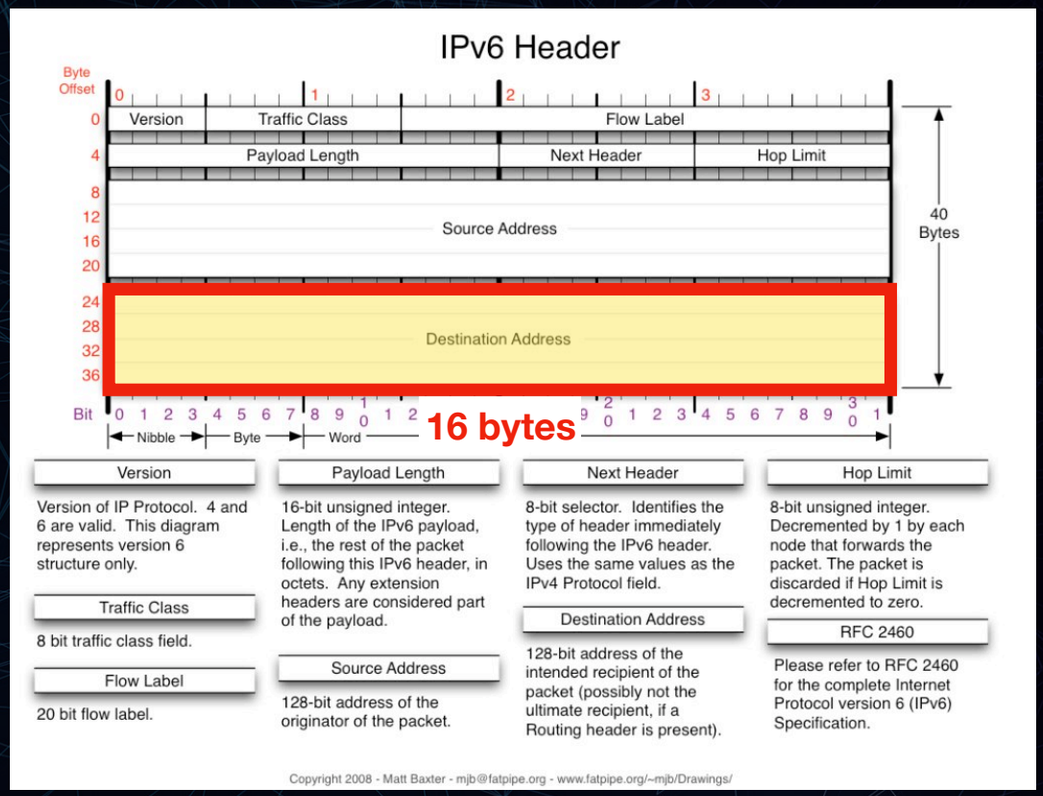
For example, this field is the Flow Label field and it is 20 bits wide. This field is used by IPv6 to mark related packets with a pseudo-random value so that routers and firewalls will treat those packets the same. For example, the traffic between a client to a web server could be tagged with a flow label so that all those packets would be treated the same by QoS and other network policies.

Abusing the IPv6 Header

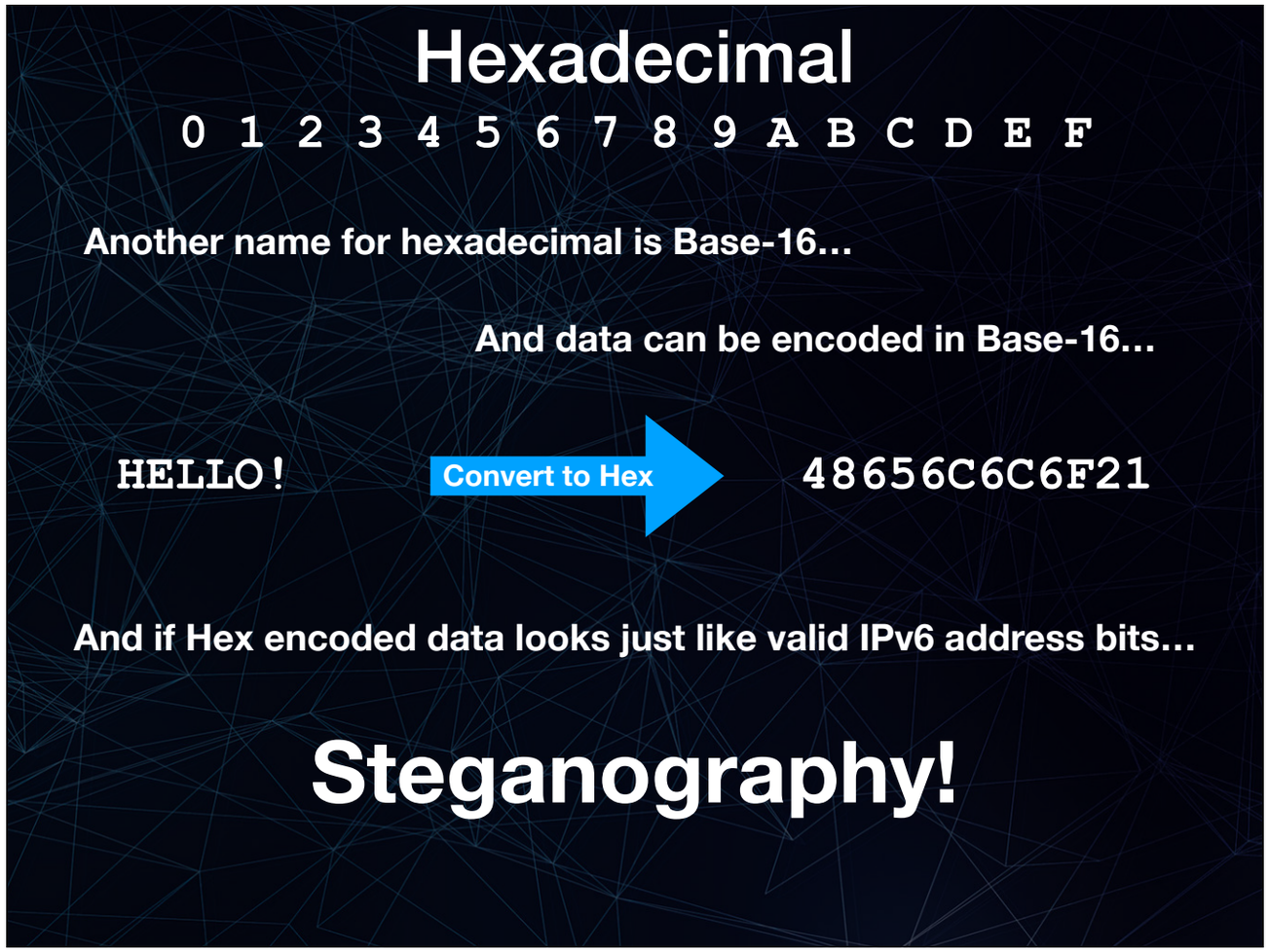


This field is the Source Address. It is 128 bits wide or 16 bytes.

Abusing the IPv6 Header



And this is the Destination Address. Just like the Source Address field, it is 128 bits, or 16 bytes, wide.



Next, the IPv6 address format uses hexadecimal digits for address information. Another name for hexadecimal is Base-16 and data can be encoded in Base-16 format.

In this example, the greeting “Hello!” has been encoded in Base-16 resulting in the string “48656C6C6F21”. This string looks a lot like IPv6 address bits missing the colons. If you group the hexadecimal digits correctly, then this Base-16 encoded data will look just like an IPv6 address. By hiding data using this technique, a threat actor will achieve steganography within an IPv6 address.

Steganography?

Noun

The art or practice of concealing a message, image, or file within another message, image, or file

2001:DB8::4D65:6574:4031:3050

2001:DB8::0D1C:4FE3:230A:EFA3

2001:DB8::5245:4345:4956:4544

2001:DB8::56DF:DEAD:BEEF:0AB3

If you are unfamiliar with steganography, it is embedding information or messages into other information or data. For example, data can be hidden inside of images by encoding data on a pixel's least significant bits. Doing so subtly changes the colors of the pixels, but in a manner that is imperceptible to the human eye. Data can be hidden in visible text by using "every Nth letter" techniques or by making subtle changes to the text layout, spacing or kerning. Other techniques include hiding ultrasonic sounds in audio tracks or embedding data into video streams.

In this case, the message "Meet@10P", once converted to hex, fits into the IPv6 address and appears to be nothing more than a valid address. In fact, show the address to any network engineer or security professional and they are not likely to recognize data is hidden in the address.

Looking at the three IPv6 addresses at the bottom of the slide, only one of them contains a message. Prima facia, the one with "DEAD BEEF" appears to be it, but it is not. You can embed plain text words in IPv6 addresses using the letters from the hex digits, but that is not where the message is hidden.

Steganography?

Noun

The art or practice of concealing a message, image, or file within another message, image, or file

`Meet@10P` → **Convert to Hex** → `4D65657440313050`

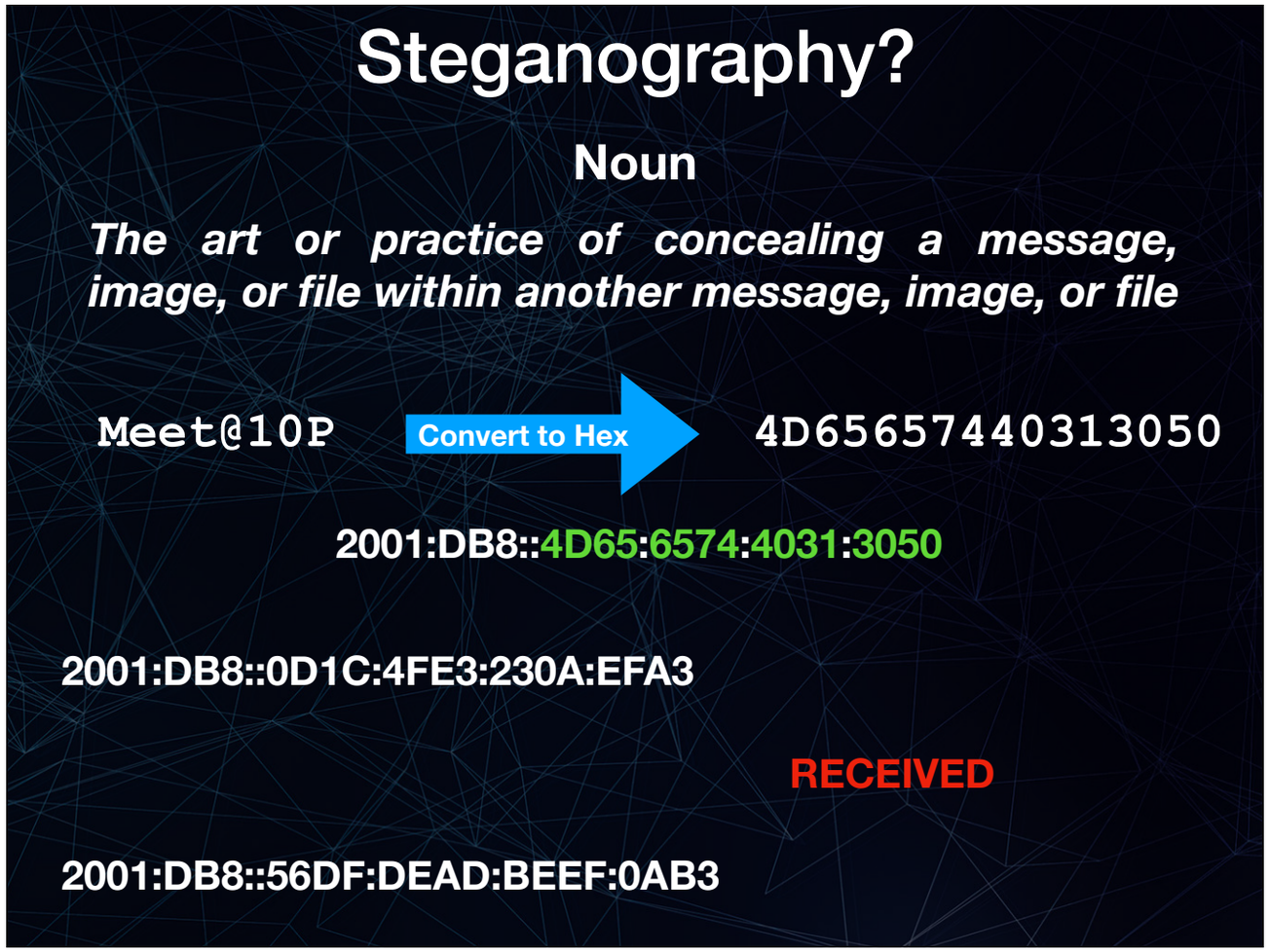
2001:DB8::4D65:6574:4031:3050

2001:DB8::0D1C:4FE3:230A:EFA3

2001:DB8::5245:4345:4956:4544

2001:DB8::56DF:DEAD:BEEF:0AB3

The message is hidden in the second address, which looks no different than the other addresses.



When we decode the data hidden in that address, we get the message “RECEIVED” acknowledging receipt of the meeting time.

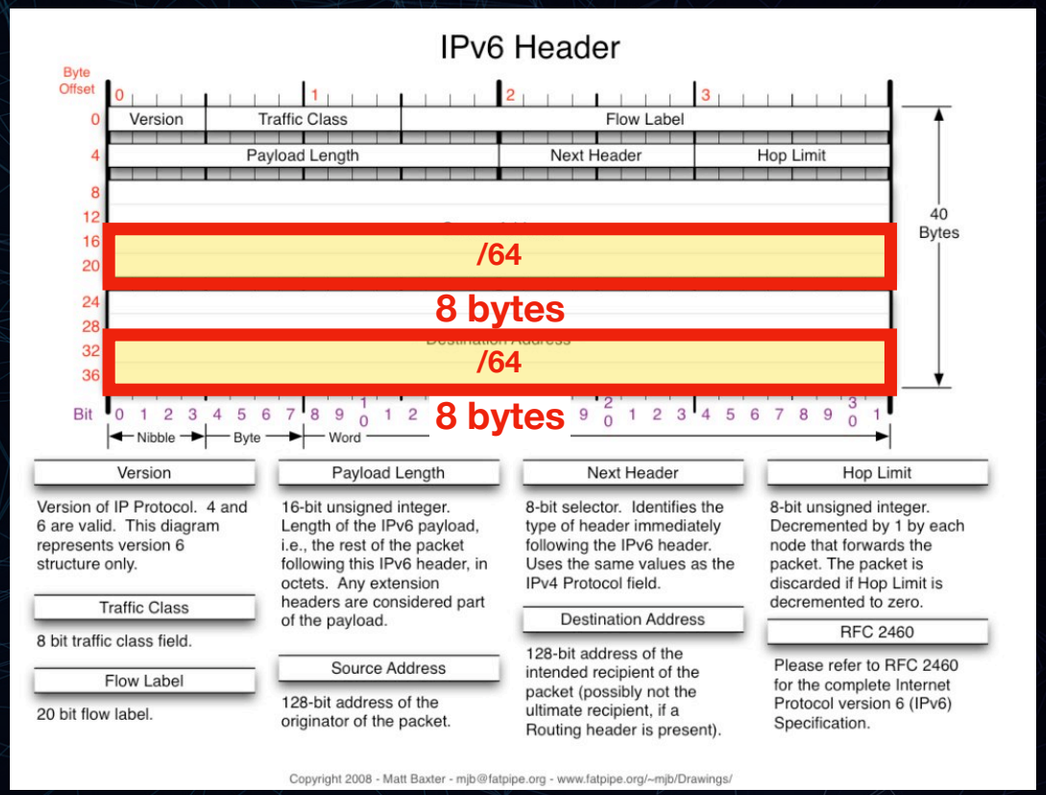
Steganography

One key consideration with steganography is the amount of data that can be hidden within a given set of data without being detected.

The IPv6 minimum recommended subnet size dictates the amount of data that can be concealed in the Layer 3 header fields.

64 bits = 8 bytes / packet

IPv6 Subnetting



IPv6 Subnetting

So, 8 bytes of data fits into the host portion of a v6 address…

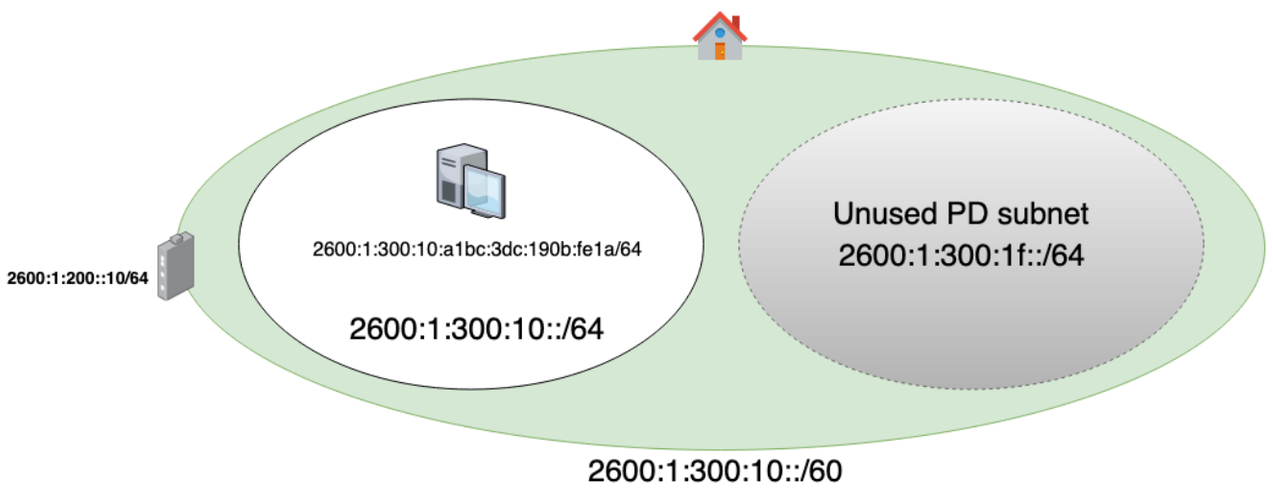
But where do you send it?

If only you had a spare /64 IPv6 subnet…

Or multiple subnets…

Abusing Prefix Delegation

The Setup

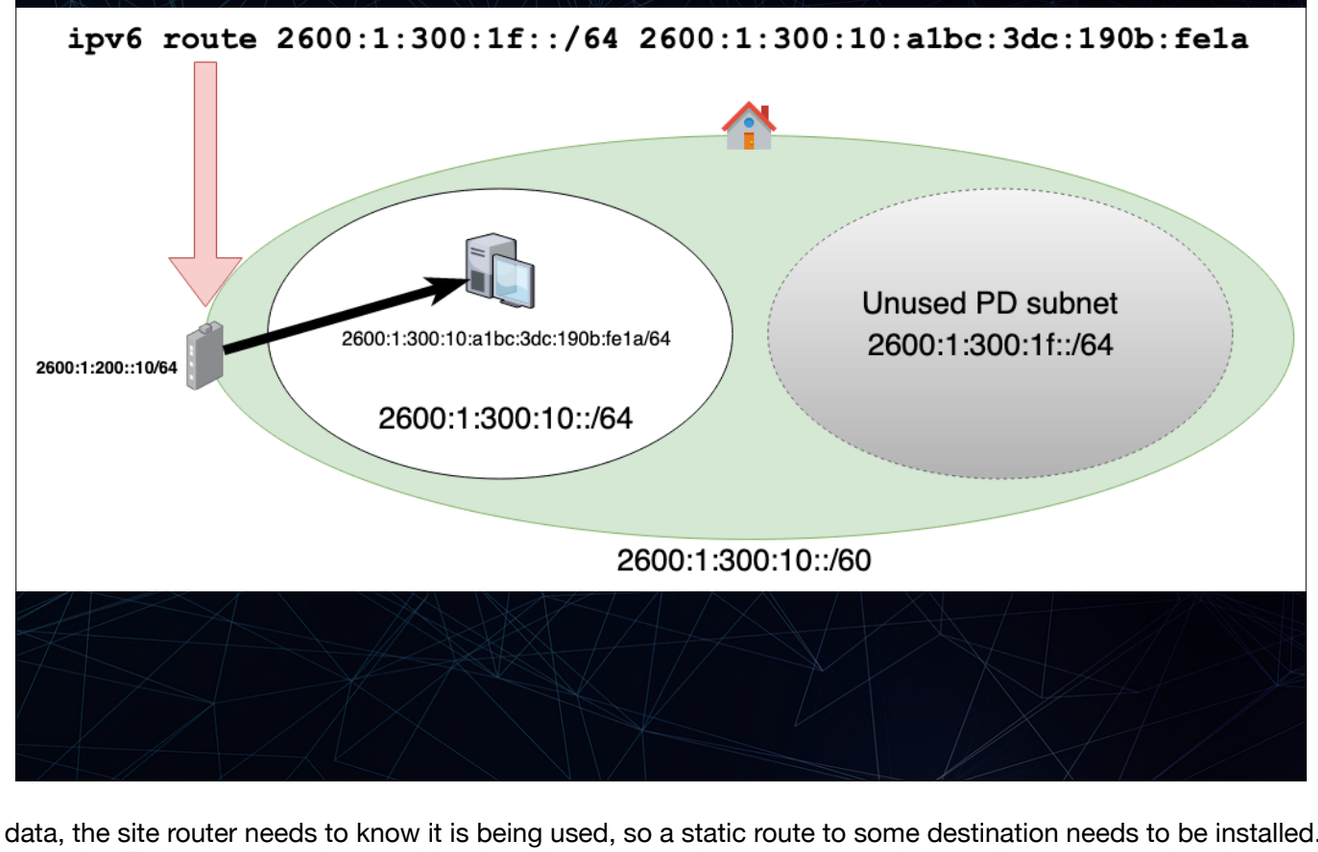


This is the point where Prefix Delegation comes into the picture.

This is what a setup with the provider assigning a /60 to a customer looks like. Again, most customers use only one of the available /64 subnets in the PD assignment, so threat actors could have a number of /64 subnets available to use. In this case, one /64 has been selected from the site's larger prefix delegation subnet.

Abusing Prefix Delegation

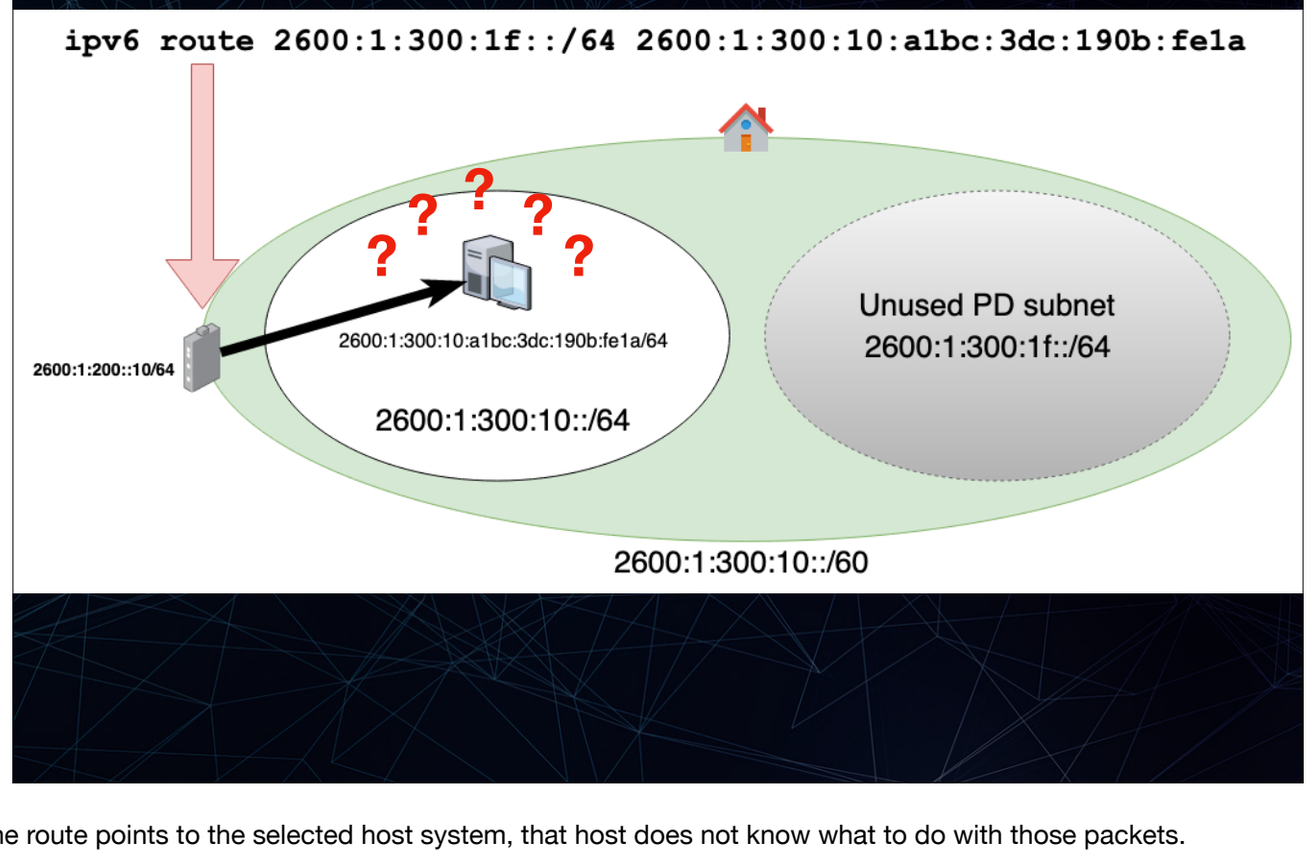
The Setup



In order to use the unused /64 subnet to receive data, the site router needs to know it is being used, so a static route to some destination needs to be installed. In this example, we are using a Cisco router to point the unused subnet to an arbitrary host to receive it. The router must be capable of configuring a static route to one of the prefix delegated /64 subnets, something that may not be supported on lower end devices.

Abusing Prefix Delegation

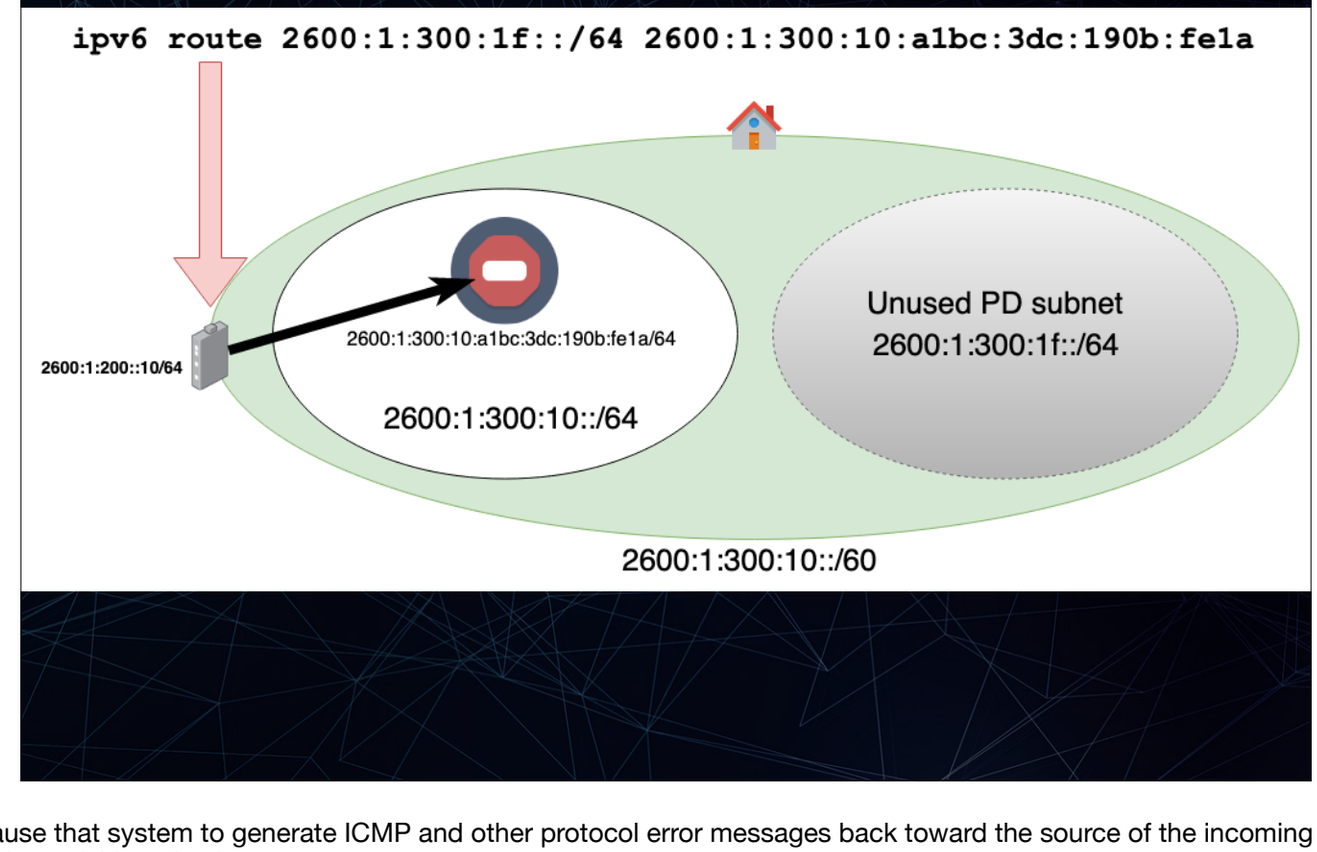
The Setup



The problem that crops up is that even though the route points to the selected host system, that host does not know what to do with those packets.

Abusing Prefix Delegation

The Setup



So they are dropped at that system. This will cause that system to generate ICMP and other protocol error messages back toward the source of the incoming packets. This is called “backscatter” and it can be detected by firewalls and IDS/IPS, exposing the activities of a threat actor.

Abusing Prefix Delegation

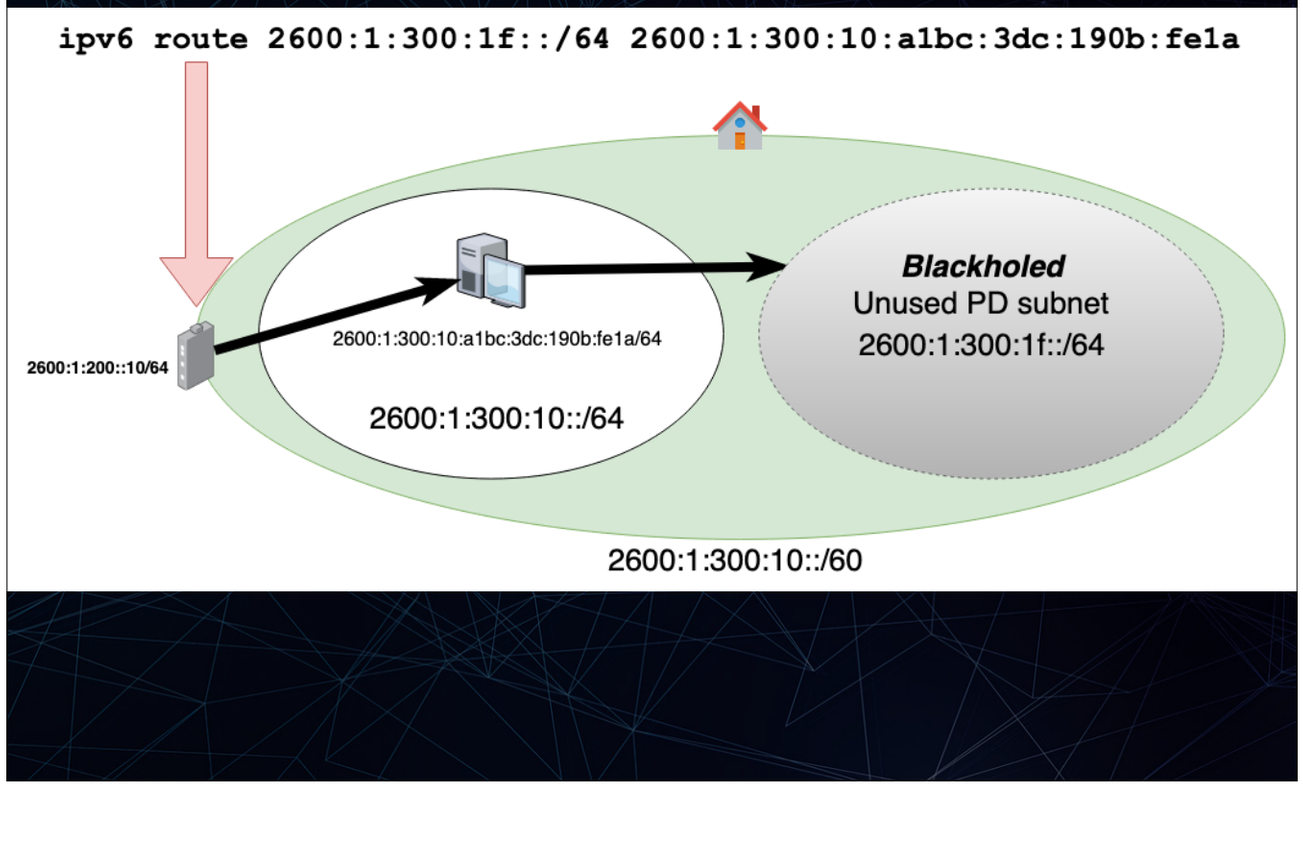
The Setup

```
me@test-vm:~$ ip -6 route
me@test-vm:~$ sudo ip -6 route add blackhole 2600:1:300:1f::/64
me@test-vm:~$ ip -6 route
blackhole 2600:1:300:1f::/64 dev lo  metric 1024  error -22 pref medium
```

In order to eliminate that backscatter, we configure a blackhole route on that host. This results in the packets to the target subnet being accepted by the system and then silently dropped into the ether.

Abusing Prefix Delegation

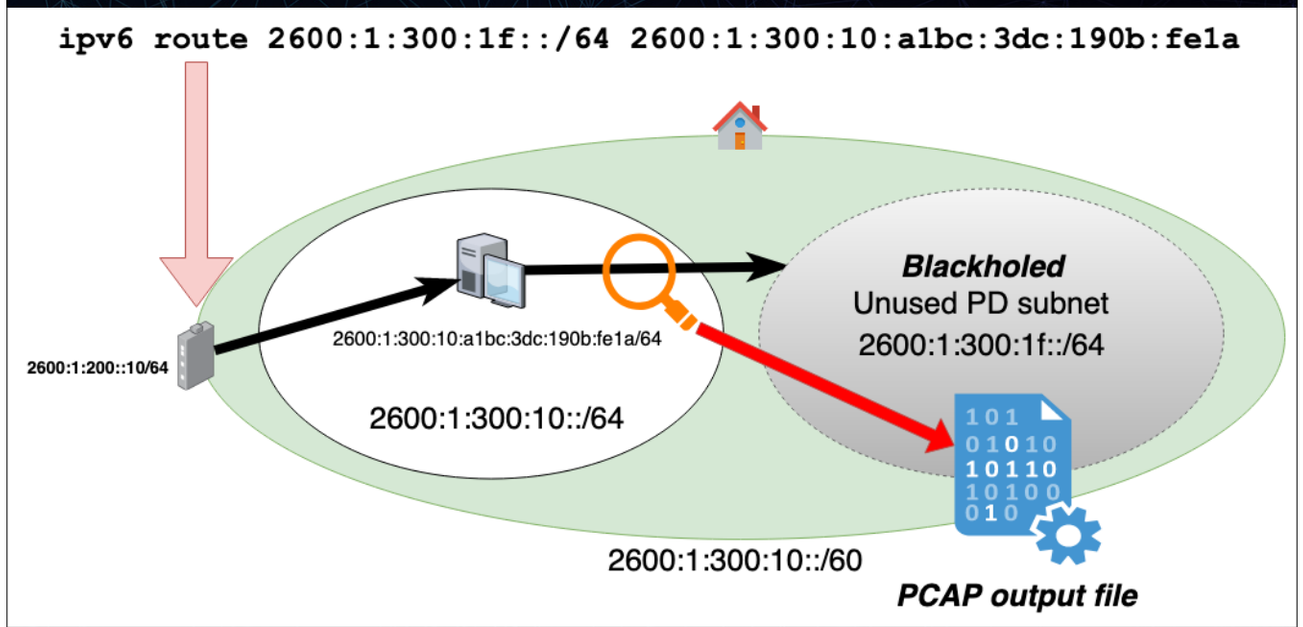
The Setup



So, logically, this is what has been setup.

Abusing Prefix Delegation

The Setup



Now, even though the target host is dropping all of the packets to the target subnet, on ingress they can be sniffed using a tool such as tcpdump or Wireshark. In order to prevent contamination of the data from other systems on the active /64 network, filters can be put in place to filter the packet capture down to the target subnet.

Now, when packets are transmitted from the source, those packets arrive at the destination location and the threat actor can extract and process the embedded data from the PCAP files allowing them to reconstitute the data from the source.


“System Requirements”

Source Configuration

1. **Native IPv6 - tunneling acceptable, if permitted, and using it *will not* trigger alerts; other transition technologies (Teredo, etc.) likely to be blocked or trigger alerts.**
2. **Compression tools - zip, rar, etc.**
3. **Format conversion tools - binary to hexadecimal, specifically**
4. **Scapy OR other packet/network request tools (wget, dig, etc.), with limitations**

One note on packet generation tools…. Using Scapy allows for the packet to be crafted to a threat actor's specifications, so in this case, the flow label field could be used to carry segment ordering information. Other tools, such as wget, dig or ping, lack the options to input arbitrary values into that field, so an alternative means for tracking segment order would be needed.

Destination Configuration

1. Native IPv6 - tunneling acceptable; must have an unused /64 subnet configured either through prefix delegation or static configuration
2. Router capable of statically routing PD assigned subnets to an arbitrary destination
3. Compression tools - zip, rar, etc.
4. Format conversion tools - hexadecimal to binary, specifically
5. Packet capture utilities - tcpdump, Wireshark, etc.

Let’s Send a Message

“Welcome to CackalackyCon”

Let's Send a Message

10,000ft Version

"Welcome to CackalackyCon"

↓

1. Convert String to Base-16 (Hexadecimal)

57656C636F6D6520746F204361636B616C61636B79636F6E

Let's Send a Message

10,000ft Version

"Welcome to CackalackyCon"

↓

1. Convert String to Base-16 (Hexadecimal)

57656C636F6D6520746F204361636B616C61636B79636F6E

↓

2. Divide result into 64bit segments

57656C636F6D6520 746F204361636B61 6C61636B79636F6E

Let's Send a Message

10,000ft Version

"Welcome to CackalackyCon"

↓

1. Convert String to Base-16 (Hexadecimal)

57656C636F6D6520746F204361636B616C61636B79636F6E

↓

2. Divide result into 64bit segments

57656C636F6D6520 746F204361636B61 6C61636B79636F6E

↓

3. Associate a numerical order (in hex) to each segment

0000000001 57656C636F6D6520

0000000002 746F204361636B61

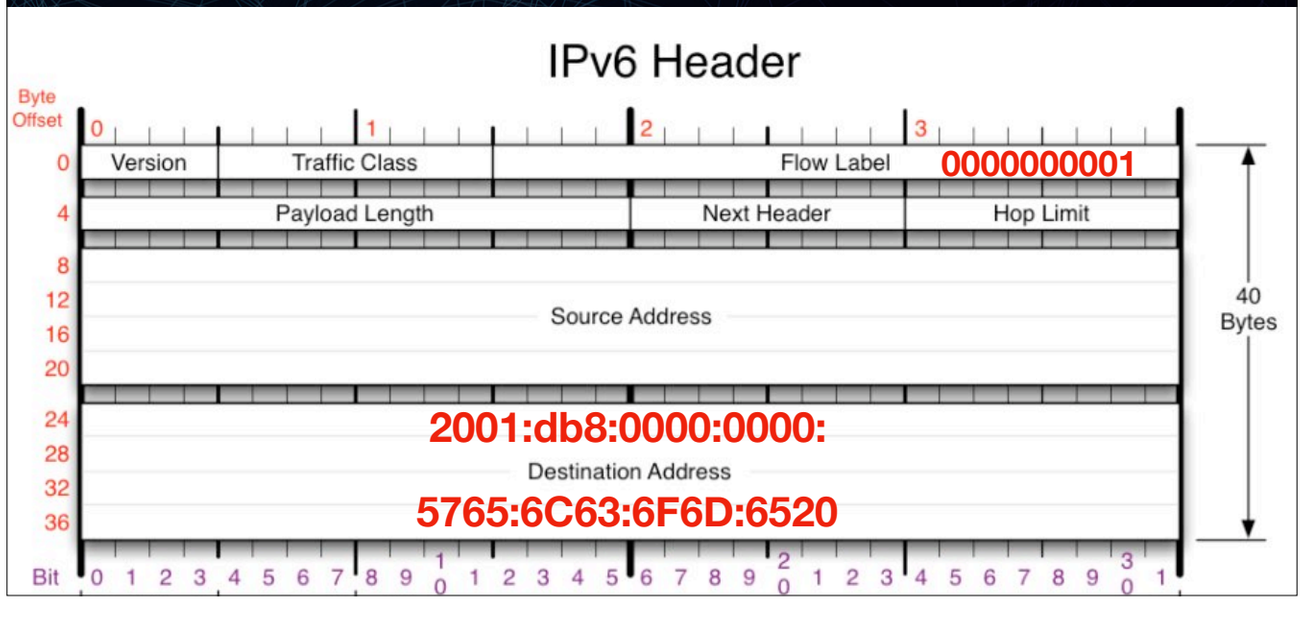
0000000003 6C61636B79636F6E

Let's Send a Message

10,000ft Version

4. Put each segment into the Host ID part of destination IPv6 address and transmit with numerical identifier…

0000000001 **2001:db8::5765:6C63:6F6D:6520**



Transmitter Processing

Step-by-Step

1. Select the target data.
2. Data compression to reduce transmission time. (Optional)
3. Format conversion to Base-16 (Hex)
4. Data slicing - converting a continuous string of hexadecimal characters to 64 bit segments.
5. Data sequencing - associate an ordered hexadecimal identifier with each 64 bit string to be transmitted with the segment for loss detection and reassembly. (Optional)

Transmitter Processing

Step-by-Step

6. Packet Construction: Using SCAPY, insert the associated numerical identifier for the first segment into the “Flow Label” field. Use the target PD subnet as the subnet portion of the destination IPv6 address and the first 8 byte segment as the host identifier. Source and destination port are at your discretion.

7. Transmit packet.

8. Repeat steps 6 and 7 for each segment.

Padding

Data will NOT always end on a 64 bit boundary

How to handle this case?

Prepend padding bits?

2001:db8::XXXX:XX48:656C:6C6F

Append padding bits?

2001:db8::4865:6C6C:6FXX:XXXX

Padding

How is not that important, so
long as the padding method
is deterministic and reversible.

Receiver Processing

Step-by-Step

1. Start a packet capture filtered for the destination PD subnet.
2. Once the final segment has been received, extract the Flow Label field and host identifier values from each packet and order the segments.
3. Concatenate the 8 byte segments into the original order.
4. Convert from hexadecimal back to the original format.
5. Decompress, if needed.

Receiver Processing

Step-by-Step

6. Tent your fingers and say…

Receiver Processing

Step-by-Step

Weaknesses and Challenges

Traceable

All the addresses are visible and less likely to change. Pointing fingers at another user is unlikely to fly.

Hurry up & wait

Transmitting only 8 bytes at a time is not particularly fast. Good for an APT-style low and slow attack. Blasting away is an option, but more likely to attract attention.

Disinformation

A technically adept victim could discover and poison the data being exfiltrated without detection. They could embed their own malicious payloads or just waste time with garbage data allowing them time for IR.

Weaknesses and Challenges

Protocol Limits

The IPv6 flow label is field is only 20 bits. This allows for only ~1MM segments, about 8MB, before the counter rolls.

Packet Loss

Without some sort of a flow control, lost packets are detectable but not recoverable. This results in corrupted data.

One possible packet loss mitigation (which was mistakenly not covered during the presentation), would be to randomize the transmission order of the packets while leaving the associated numerical order identifiers for each segment unchanged. So, in an eight segment example, instead of transmitting in order - 1,2,3,4,5,6,7,8 - the segments would be transmitted out of order - 8,3,5,7,6,1,2,4. Once all of the packets have been transmitted, the segment transmission order would be randomized and all segments transmitted again.

This method relies on the statistical unlikelihood that the same packet would be lost both times. If a packet was lost, it could be recovered based on the data received in the first or second transmission.

Unfortunately, this requires every segment be transmitted twice, extending the length of time the exfiltration activity takes which increases the possibility of detection.

Potential Detection Methods

Netflow Anomaly Detection

1:1 Netflow with Anomaly Detection is able to detect single host to many host conversations that exceed configurable thresholds. Low and slow transmission could thwart this method of detection assuming sufficiently long periods of time between transmissions.

Full Packet Capture

A victim armed with full packet capture would be able to run anomaly detection while also being able to reconstruct the data transmitted by the adversary. Encryption prior to transmission would thwart reconstruction of exfiltrated data.

So could this be detected?

Unfortunately for Blue Teams, detection is unlikely today.

IDS/IPS?

No, not currently. Signatures currently examine source/destination IP for rough filtering or blacklist matches, but do not examine those fields for embedded payloads. Doing so would be computationally intensive and potentially prohibitive to implement.

DLP?

No, these systems also do not inspect source/destination IP for payloads and doing so would have the same issues as implementing this functionality in an IDS/IPS.

So could this be detected?

Unfortunately for Blue Teams, detection is unlikely today.

Full Packet Capture and 1:1 Netflow Anomaly Detection is expensive and out of reach for many potential victims.

The rate of false positives with any detection method would be high because even legitimate source/destination addresses will contain patterns mimicking embedded data.

Single source to many destination pattern of exfiltration traffic also mimics the normal one to many pattern of legitimate traffic.

ACK

Related Works

Giving credit where credit is due…

Miller, Barrett. "*Steganography in IPv6*". University of Arkansas. 2008

Qasim, Mustafa Alaa and Dipak Pawar. "*Encryption & Steganography in IPv6 Source Address*". International Journal of Computer Engineering & Technology. Vol 4, Issue 2, March – April (2013), pp. 315-324

Both papers encode data in the Source Address field of the IPv6 header field directed to a single destination address.

The drawback to this method is it generates more noise at the source by increasing the number of neighbor advertisements. Rapid turnover of source addresses, inconsistent with normal churn, can be detected and may trigger incident response.

An additional drawback to embedding data in source addresses is the possibility of an address collision with host on the network. While unlikely, due to the number of addresses in a /64, there is no way it could be prevented since any valid IPv6 address could also be a valid data pattern. Address collisions are highly unusual in properly run networks and would likely trigger some sort of investigation.

FIN

Questions, Comments, Flames?

@__wavelength__

darkwavelength@protonmail.ch

Email messages preferred for clarification, comments or questions, however Twitter DMs are also open.