Global Information Assurance Certification Paper

Hey Dude! I Can Do a Great Humphrey Bogart!

GIAC (GSEC) Gold Certification

Author: Lee Peterson, lee.peterson@state.mn.us
Advisor: Egan Hadsell

Accepted: November 9, 2009

Abstract

Watching someone do a great impersonation of a famous person can be very amusing. However, when it comes to network devices, impersonation can be all too easy to accomplish, and difficult to detect. The principles of Information Security should be instituted at every point along the path from a packet's source to its destination. However, finding a balance between security and availability can be a thorny road upon which to embark. This paper will expand upon the knowledge gained through the SANS SEC401 course materials. Hopefully, readers will find the paper informative and entertaining.

© SANS Institute 2009, Author retains full rights.

1. Introduction

This paper will present a fictitious scenario wherein a router is duped into believing an imposter is a router that is already known and trusted. As a result, his routing tables are overwritten and traffic gets re-routed. This paper is written from the point of view of an overworked, stressed out, grumpy router with an attitude. Throughout the paper, the router will highlight various packets it receives, and explain how he processes them. As the story progresses, sporadic nefarious incidents will be seen by the router. They could be seen as hints about what is about to happen. Unfortunately, the router is not a firewall, so his depth of view is limited, and certain security practices are not being used. At the conclusion of the paper, the author will recommend a security option that should make router impersonation more difficult, thereby providing an additional layer of protection.

2. Were You Offended When I Called You an AS?

There's something funny going on out here. It's not the "Ha, ha" kind of funny either. It's the something strange, something insidious, maybe even something dangerous kind of funny. I think I've seen this kind of thing before, but I can't remember exactly what it was, or when. I don't think my logs go back that far either, so I can't look it up.

Oops! Sorry about that. I see you've caught me talking to myself again. I do that sometimes. Hi, I'm Poncho, and I am a router. More specifically, I am an IPv4 router. I've been expecting you. Pull up a chair right next to me here and take a load off. They told me your name is Jim and that you want to ride shotgun with me for awhile. That's ok with me. It gets kind of lonely out here in the middle of the night. They also told me I can tell you pretty much everything you want to know, because you belong to one of my Autonomous Systems. That's "AS" for short. So, if I call you an "AS," please don't be offended. Actually, an "AS' is defined like this:

> An Autonomous System (AS) is a connected segment of a network topology that consists of a collection of subnetworks (with hosts attached) interconnected by a set of routes. The subnetworks and the routers are expected to be under the control of a single operations and maintenance (O&M) organization. Within an AS

> routers may use one or more interior routing protocols, and sometimes several sets of metrics. An AS is expected to present to other ASs an appearance of a coherent interior routing plan, and a consistent picture of the destinations reachable through the AS. An AS is identified by an Autonomous System number. (RFC 1812, 1995, Section 2.2.4)

In a way, you could think of Autonomous Systems as if they were independent countries, or they could be called nation states, or simply nations. As you know, each independent country has its own set of rules and regulations, and specifically for this analogy, its own transportation system. You're right, Jim. In today's world economy, no country is totally independent from other countries. There are alliances, both political and economic among various countries. Of course, travel between countries is really common nowadays. In lots of countries, there are places along the borders where you have to pass through in order to get permission to enter their country. You could think of those places as firewalls.

But, my point about ASs is that each one is administered internally by its own set of rules and regulations. Douglas E. Comer put it rather succinctly in his book on TCP/IP where he wrote, "The Internet is divided into autonomous systems that are each owned by a single administrative authority. An autonomous system is free to choose an internal routing architecture and protocols." (Comer, 2006, pg. 253).

What's that, Jim? How does a citizen from one country communicate with a citizen in another country in this little analogy? That is a real good question. Well, you're looking at it. That's right. They communicate through me, and other routers like me. In fact, that is my primary function. But, I'll get into that in more detail later.

Mind you, I haven't been around for all that long, comparatively speaking. Some of my neighbors have been out here a whole lot longer. Some of them have been out here way too long if you ask me, and it shows. But, I'll get into that issue later. So, you say you want to know exactly what a router does. Well, I'm just the one to tell you. I've always considered myself a teaching router, and I'm more than willing to share my vast expanse of knowledge with others – actually, with anyone who will listen. But, before we go any farther, I just want you to know right up front here, and this is for the record, I

play by the rules out here. What rules, you say? Primarily, I follow Request for Comment 1812. That's "RFC 1812" for short. There are others I follow too, like RFC 1771 and RFC 1772, and I will probably allude to them as well as others along the way. But, RFC 1812 is my main bag.

Out of that entire RFC, my really big thing is called the, "Robustness Principle." Here it is, word for word from the RFC:

> At every layer of the protocols, there is a general rule (from [TRANS: 2] by Jon Postel) whose application can lead to enormous benefits in robustness and interoperability: Be conservative in what you do, be liberal in what you accept from others. (RFC 1812, 1995, Intro 1.3.2)

I pride myself on being able to do my job no matter what. I can't remember the last time I went down. I'd have to check my logs to see, if you're all that interested. The fact is, I just don't remember. Come to think of it, I really don't remember much of anything. I must have been born without a long-term memory, or I'm senile. I guess it's like they say, "Too soon old, and too late schmardt." It seems like I have to look up everything; which packets go where, which packets to forward, which ones to drop, which error messages to send, which ones not to send, what gets logged and what doesn't. But, I'm getting ahead of myself.

The short version of my job description says that I "perform the network layer forwarding function of the Internet protocol suite." (RFC 1812, 1995, Section 1). I can also be an Internet host if that is necessary. I guess you could say I am like a roundhouse in a railroad yard. All I do is move boxcars (packets) from an inbound track to the appropriate outbound track by evaluating the information listed on the labels that are on the sides of the boxcars, and making routing decisions based on that information. It may sound easy, but it is a lot more complicated than you might think. The labels on the sides of the boxcars are automatically scanned when they enter the roundhouse, and that information is sent directly to me on my "Heads-Up Display," that's HUD for short. I check to see where the boxcars are bound, where they came from, how far they've travelled so far, and the like. Ok, technically speaking, what I actually check is called the IP Header.

Of course, I have to know about other labels too, like TCP, ICMP, IGMP, BGP, IGP, UDP, DNS, SNMP, BOOTP, and TFTP. I could go on, but I'm running out of breath, and you are probably getting bored with all my blabbering. I don't get visitors here very often, so I tend to talk quite a bit when I do. Well, grab yourself a comfy place to sit, and let's get started.

3. Hey! Is Anyone Awake Out There?

I see by the time display on my HUD it is about a millisecond past midnight, Universal Time. The first thing I like to do at this bewitching hour of the night is set up my schedule for the day. You should understand I do a lot of stuff in the background most folks don't know about. It's really important stuff though. In fact, if it wasn't for me doing all that work, the Internet as we know it would cease to function. Can you imagine what life would be like without an Internet? I hate to even think about that! But, if I keep doing my job, everything will be ok – I hope. I'm still kind of worried about that … well, never mind that for now. We'll get into that later - maybe.

I generally start with determining if any of my friends are out there, or "Up and taking nourishment," like they say. One of the "nouveaux" things folks in the general populace have been talking about recently is, "living in community." It kind of gives you goose bumps all over, if you're the touchy-feely kind, which I'm not. Well, all of us routers out here have our own community, and we've even got our own special language we use when we talk to each other. It's called "Border Gateway Protocol," or just "BGP" for short. We're all the way up to the 4th version of our language now. We've come a long way! The purpose of it is to see who is awake and who isn't, and to distribute routing information so each of us knows which of our peers handles packets for which AS. Most of this information is distributed on a regular schedule. For example, I have five timers I use to run these tasks; a "ConnectRety" timer, a "Hold Time" timer, a "KeepAlive" timer, a "MinASOriginationInterval" timer, and a "MinRouteAdvertismentInterval" timer. Now, you may think I've got nothing but time on my hands, pun intended, but I don't. I pretty much follow the times that RFC 1771 suggests:

> The suggested value for the ConnectRetry timer is 120 seconds. The suggested value for the Hold Time is 90 seconds. The suggested value for the KeepAlive timer is 30 seconds. The suggested value for the MinASOriginationInterval is 15 seconds. The suggested value for the MinRouteAdvertisementInterval is 30 seconds. (RFC 1771, 1995, Section 6.4)

Of course, any of us can send updates whenever we are notified of any changes by our own ASs. In fact, you don't even have to be a router to speak BGP. A non-routing host may speak BGP too. We call them, oddly enough, "BGP speakers." Ok, sometime we're not too creative.

Once I check my schedule, those boxcars start rolling into the yard pretty quick. Actually, they never stop. Packets coming in and going out of here might slow down a bit from time to time, but they almost never come to a complete halt. If they do, there is something drastically wrong. Of course, I construct my own boxcars too, so I put them on the track myself and get them queued up. They'll be rolling out of here in no time at all, like I always say, "Zippy-Rippy-Tippy!"

Oh, here's an incoming boxcar that just came from Willie. He's the next-hop router for 10.25.0.0/16. By the way, all the IP Addresses I will be talking about are fictitious. I do that to protect the innocent. But, nobody is innocent, if you catch my drift. Let's see what he's got to say.

Anyhow, Willie takes care of all the 10.25's. They are his ASs. "SYN," he says. So, I say, "ACK." He replies, "SYN-ACK," so I say, "ACK." Now that we have established a TCP connection, I can hardly wait to hear what good old Willie has got to say. The next boxcar from him says "PSH" in the IP header and indicates he is speaking BGP now. That's a code 8 in the 9th bit offset from zero in the IP Header. Here, I'll write the dump of the packet up here on this whiteboard. First, I'll write the Link Layer:

00c3e2d2b36f 00d3b296c3a40800

Right after that is the IP Header:

45000028091f4000080cf30a1900020a190001

After that comes the TCP Header:

00b300b3385a089a2233999c501801022d58

After that is the BGP Header:

0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
00000000000000000001110000000001

Finally, here is the BGP OPEN message:

00000100010110100110111100000000000000000010100001100100000000
000000011

By the way, do you see all those whiteboards over there in the back of the room? Yes, I could use more light back there. No, I'm not cheap. I just like a little ambiance, that's all. Anyway, I've written down the formats for all the various messages I frequently use. I call that place my Appendix, so when I write out a message on this whiteboard for you to see, I will also tell you where you can find the message format in my Appendix. Pretty cool, huh dude? No, Jim, I haven't had any Appendixitis attacks. That was really bad, Jim. I see I'm going to have to give you some examples of real humor. I also put up a network diagram, it's listed as Appendix B, but you have to realize it is strictly from my point of view, which is the only point of view I've got. Ha! You'll see the names of all my router buddies back there; there's me of course, Freddy, Chrisco, Willie, Robby, and Hazel. I also put up all our network interface information, and it's called Appendix C.

So, I look at the BGP Header (see Figure 1 in Appendix A), and it indicates this is an "OPEN" message (see Figure 2 in Appendix A). Gee whizlickers, isn't that nice of him? He is checking to see if I'm awake. So, I make up my own boxcar that has a "KEEPALIVE" message inside, and set the "KeepAlive" timer in it to 30 seconds. Of course, we send messages back and forth more often than that, so if that timer ever runs out, we think there's something wrong. By the way, that brings me back to what I was talking to myself about when you walked in. You see, the "KeepAlive" timers on some of the BGP KEEPALIVE messages I've sent to Robby have been running out.

Anyways, I put a TCP Header in front of my BGP KeepAlive message, an IP Header in front of the TCP Header, supply all the Link Layer info in front of everything else, put my boxcar on the outbound track, and send it back to him. Zippy-Rippy-Tippy!

If Willie wants to let me know about any changes in his routing tables, he will probably send a BGP Update message next. Oops! Hold on a tick. My outgoing track is getting full. Those boxcars start getting antsy if they have to wait too long in line. I need to send some of them out – now! Zippy-Rippy-Tippy! Ok, that's better.

4. Putting Packets on the Right Track

My HUD indicates a packet just arrived on track eth0. Let's check it out. Here, I'll read it, and then draw it on this whiteboard so you can see it. Here's the hex dump of it:

00b035c9d252000235940e391080045000034074e40008006e72c0a0a0c2
50a00010404d60050016d5b5a000000008002faf006b90000020404ec0103
030001010402

Now, this is just a normal run of the mill packet I am going to send along its way. I haven't written down the IP or TCP Header format in my Appendix, so you'll have to look that up for yourself. Oh, you've got those memorized? Good for you, Jim! You're a lot smarter than you look! Well, just so you know, I should probably give you a quick rundown of what I actually do to decipher packets. I am going to get a little technical now Jim, so please pay close attention. You see, Jim, it's like this. First, I validate the IP Header. You can check RFC 1812 Section 5.2.2 to see exactly what is specifically required for validating IP Headers, but here's the short version. In my IP validation routine, I start off by checking the packet length that was reported by the Link Layer to make sure it has enough room to at least hold the shortest legal IP packet, which is 20 bytes. Second, I check the IP checksum to make sure nothing in the packet has changed since it left the sender. Third, I verify the IP version number. Fourth, I verify the IP header length to see if it is at least 20 bytes. Fifth, I check the IP total length to see if it is at least large enough to hold the IP Header. If I find anything wrong, I drop the packet

like it was a piece of hot coal that came right out of the furnace of an old steam locomotive, and make an error entry in my logbook.

Next, I process almost all of the IP options. For instance, an IP option could be something like record route, record a timestamp, use loose source routing, or use strict source routing. Most packets don't have any of the IP options set, and you can tell that just by looking at the IP header length. If it's only 20 bytes, there won't be any options. Well, most legal packets are like that. If I were a firewall, I'd be suspicious of any packet whose IP header is bigger than 20 bytes, but I'm not a firewall so I don't worry about it. They don't pay me enough to do someone else's job anyway. Some of those options are used for troubleshooting, but that's about all. I can get into those IP options in more detail later, if you like.

Next, I check to see where the packet is headed. Before I send it along, I've got to make sure it's not for me. It would be really stupid of me to send a packet out only to find out from one of my buddies it was meant for me the whole time. Talk about embarrassing! That would be like you getting an envelope in your mailbox addressed to "Postmaster." Ha!

If the packet is not for me, I try to figure out who I should send it to by looking up the destination address in my routing table, and from that I decide which track to put it on. In the business, that's what we call the "Next Hop" IP address. But, before I actually put that boxcar on an outbound track, I need to make sure I have permission to forward it. What do I mean by that, you say? The "from" and "to" addresses have got to be valid IP addresses, and I also need to check my own internal set of rules to make sure it is ok for me to forward the packet.

Next, I decrement the TTL by one (or more if I feel like it), that is, if it's not already zero, and finish processing the IP options I couldn't process before; like the record route option, for example. I could hardly record a route if I haven't decided which route to use, could I? The same thing goes for the timestamp option. I'm supposed to put a timestamp in the packet, and that time is supposed to be the time at which I put the packet on my outbound track. By the way, I synchronize my internal clock by using the

Network Time Protocol (NTP) and a good reliable time server. Shoot, I even have a backup time server I can go to if my first choice isn't available.

Next, I check to see if I have to perform any fragmentation. Fragmentation! What a bother! I wish they'd get pipes big enough to carry any size packet I want to send. That would be nice, wouldn't it!

Next, I get the Link Layer address (MAC address) of the next hop IP address as well as my own Link Layer address, and put that information in front of the IP Header. They call that "encapsulation." So, I "encapsulate" the IP datagram. You can think of it as putting everything in a nice pretty package, with more packages inside, kind of like one of those Christmas presents that gets passed around at office parties. No, Jim. It doesn't make any difference to me whether there is actually anything in the innermost package. This little Christmas package isn't for me anyway, so why should I care? Like I said, I'm a router, not a …. Well, you get the idea! Then, I simply put it on the appropriate outbound track, and Zippy-Rippy-Tippy, off it goes.

That's about all there is as far as what they call "Unicast" packets are concerned. "Multicast" packets are a horse of different feather, but we can get into that in more detail later on, if you like.

Let me show you how this process works on a real live packet. Here's another copy of the packet we were talking about earlier, just in case you don't remember it:

00b035c9d252000235940e391080045000034074e40008006e72c0a0a0c2
50a00010404d60050016d5b5a000000008002faf006b90000020404ec0103
030001010402

So, this packet was pretty easy to figure out, wasn't it? It's a normal TCP SYN packet coming from one of my ASs. Specifically, it's from 10.10.12.37, and its destination is 10.0.72.49. It actually arrived on track number eth4. That's the one with Link Layer address (MAC address) of 00b035c9d252. It says, "Don't Fragment." Why, I wouldn't think of it! Heavens to Moigatroid! The Packet's "Time to Live" is 128. Most of us don't actually use time as a value. We just use hop counts instead. It's a little easier on all of us that way. I also need to remember to decrement that by 1 before I send

it along. I can tell just from the TTL this packet came from one of my own ASs, because the value is still the default. About the only way the TTL is still the default is if it hasn't gone through any other router, unless someone has messed with the packet.

So, I try to find the correct route in my routing table. Let's see now … it looks like my good friend Freddy handles that, I decrement the "Time to Live" by 1, so now it's 127. I also replace the Ethernet destination address the packet came in with, which was the Link Layer (MAC) address of one of my own network interfaces, eth4 to be specific, change it to Freddy's Link Layer address, pack it all up, and put it on my outbound track number eth0. Zippy-Rippy-Tippy!

Now here's an interesting packet that just arrived from Robby. Oh, this isn't like Robby at all! I don't have time to write it all out on the whiteboard right now, but this is an IP packet and it has a bad checksum in the Header. Actually, there's a whole stream of them with bad checksums that came in one right after the other. Poor Robby is going to end up resending all of those packets, because I am required to drop them and I'm not supposed to send Robby any error messages about them either. Remember, IP is a connectionless protocol.

By the way, I should let you know I'm not going to tell you about everything I am doing. It all happens so fast, and I can't afford to hold up traffic just so I can tell you what's going on all the time. But, I will try to pick out some packets you might find interesting. I should also tell you that when I talk about my inbound or outbound tracks, I'm referring to my inbound and outbound queues. Actually, every one of my network interfaces handle both inbound and outbound packets, and the routing information I put in each packet indicates through which interface the packets will actually pass.

5. I See London, I see France, ICMP'S Underpants

Well Jim, I see by the clock on my HUD the current time is 00:00:00:00:01. Time sure flies when you're having fun! Let's see what else we've got going on. Oh, here's another interesting packet. It came from one of my own ASs. This packet came from 10.10.12.108. It's headed for 10.10.48.1, which is a host on another one of my ASs. I'm a little surprised this packet came to me instead of Chrisco, because he is actually

closer to the destination than I am. You see, my partner in crime on this segment of the network is Chrisco. Both of us work in concert to handle all the traffic from our 10.10.x.x ASs. It can get pretty hectic around here, so we try to share the load. Anyway, I can see that there is a shorter route between those two hosts, and that route is directly through Chrisco. So, I'll send the packet to its destination as listed in the packet, and send an ICMP Redirect message to the originating host. I'll draw the packet on the whiteboard so you can see it, Jim. Look up here, Jim. Now remember, this is just the ICMP Redirect part of the packet, not the entire packet. (Refer to Figure 5 in Appendix A)

00000101000000000000000000000000000010100000101000000000000000010

Right after that I append the original packet's IP Header plus the first 64 bits of its original data. But, I'm not going to draw that up here. So, now that my ICMP Redirect message is ready to go, I'll put it on the outbound track destined for the originating host. Zippy-Rippy-Tippy!

Now, wait just a tick here! Maybe I'm a bit paranoid, but about the only way that last packet would have come directly to me is if either that host's routing table was misconfigured, it was corrupted, or if Chrisco went off-line. But my KEEPALIVE timer for Chrisco has not expired yet. I'll have to keep an eye on Chrisco for a while. I sure hope he's ok.

What's that, Jim? Under what conditions are ICMP Redirect messages sent? I send them when all of the following conditions are met:

- The interface on which the packet comes into the router is the same interface on which the packet gets routed out.
- The subnet or network of the source IP address is on the same subnet or network of the next-hop IP address of the routed packet.
- The datagram is not source-routed.
- The kernel is configured to send redirects. (By default, Cisco routers send ICMP redirects. The interface subcommand **no ip redirects** can be used

to disable ICMP redirects. (Cisco Systems, Inc. 2008, July. *When Are ICMP Redirects Sent?*)

6. Martian Addressing? I Didn't Know They Had Computers on Mars!

Packets don't always arrive as nice and pristine as when they left their originating host. Sometimes, things get mangled along the way. I don't know why it happens, but the simple fact of the matter is they do. There are quite a few things that can be wrong in a packet. As one of the guardians of the Internet, it is my sworn duty to raise a flag about bad packets that come my way. That's why I always go through my verification checklist before forwarding any packets I receive. They could have an invalid source or destination address, for example. What do I mean by an invalid address, you ask? An invalid address is one that has been either spoofed or misconfigured. A misconfigured address is pretty easy to spot, because it just doesn't make any sense. What I mean by that is the address is not a private IP address, or a public IP address, or the source address is goofy; like a 127.0.0.1 address (a host's loopback address), or if the address is all 1's (255.255.255.255), or all 0's (000.000.000.000). A spoofed address is also easy to spot, because it arrives on one of my interfaces that it's not supposed to arrive on. For example, a spoofed packet might be one that pretends to come from one of my ASs, but arrives on track eth0. A spoofed packet is one that has been intentionally altered so it appears as if it came from another host, most often, a host in one of my ASs. These two types of addressing are called, "Martian Addressing." Thankfully, I don't see a whole lot of these types of packets. But, when I do get packets like that, I send them to my own personal "Martian Addressing Riddance System," that's MARS for short. That's right, I drop them. I also make a log entry as specified in RFC 1812:

> "If a router discards a packet because of these rules, it SHOULD log at least the IP source address, the IP destination address, and, if the problem was with the source address, the physical interface on which the packet was received and the Link Layer address of the host or router from which the packet was received." (RFC 1812, 1995, Section 5.3.7)

Speaking of which, here's a packet that just came in on track eth2, so it must be from Robby. Nope, now that I look at it, it doesn't have Robby's IP address or his Link Layer address either. But, it's speaking BGP, and it's asking for my routes. I guess I'll have to give them, because I'm supposed to give them to anyone who asks for them. Gee, I wish we were using that Security Option of BGP. It would make it a whole lot safer for all of us. This looks kind of fishy to me.

7. I'm not overweight. I'm "Robust!"

Now that we've got a free tick here, let's talk about that "Robustness Principle" I mentioned earlier. "Be conservative in what you do, be liberal in what you accept from others." (RFC 1812, 1995, Intro 1.3.2). You see, my main job is simply to receive and send packets. I'm here to keep stuff moving, to do it as quickly and efficiently as possible, and to keep on doing it hour after hour, day after day, and year after year. Can you imagine what would happen if I decided to take a day off? I am certain the world as we know it would cease to exist. Am I really all that important? You're darn-tootin' I am! All of us routers out here are. I am part of a humungous communications network that reaches clear around the globe, and probably into outer space as well. If I go down, my part of the world can't communicate with the rest of the world.

What say, Jim? Do I ever not send out a packet I received based on anything inside the packet? Oh, there are instances where I might drop a packet, but that's probably because there's something wrong with it, things don't add up, or something has gotten mangled along the way. That happens every once in awhile, like I said a few ticks ago. Weren't you listening, Jim? I also have some internal rules by which I must abide, but there really aren't all that many of them. So, generally speaking, I pretty much send along whatever packets I receive, unless they are sent directly to me, of course. Aside from my normal fragmentation duties, I don't alter the packets in any way other than putting in the next hop router address and decrement the time-to-live. I can go into

fragmentation later if you like, but you may need to remind me about that, mostly because I don't like doing it.

What say? Do I perform any kind of filtering based on state? I guess that depends on which state you're talking about. I filter out any packets with the word "small" in them that are bound for the state of Texas. I also filter out any packets with the word "snow" in them that are bound for the state of Florida. Not that kind of state, you say? What other kind of state is there? You say state is where you keep track of session information, you know, "SYN – SYN/ACK – ACK," and all that kind of stuff. I have absolutely no idea what you're talking about. Dammit Jim, I'm a router, not a firewall. I have my own job to do, and it keeps me busy day and night. I don't have time to take on anyone else's job too.

I am noticing an increase in traffic now. It looks to me like people are getting to their offices and starting their computers. I don't know what they're doing, not exactly anyway, because I don't look that deep into their packets. Hang on a millisecond. I've got some scheduled work to do right now. Ok, I'm back. What was it I just did, you say? Oh, just some routine stuff. Mostly BGP KEEPALIVE messages, and some routing table updates. Speaking of BGP KEEPALIVE messages, an interesting thing happened when I was back at my command post. One of my KEEPALIVE timers zeroed out. As you probably recall, whenever that happens, it means one of my buddies has gone off line for one reason or another. In this case, it was Robbie who went off line. I suspect he'll be right back though. It could be he rebooted, or there was a bad connection somewhere along the line. He'll be back. It's probably nothing to worry about, but I do need to mark all his routes as "not in service" until I hear from him again. I do hope he won't be too long gone.

8. Ida, Sweet as Apple CIDR

I, for one, am very happy they came up with this new-fangled CIDR notation thingy. It sure makes my job a lot easier. What is CIDR, you say Jim? First, you might ask, what does CIDR stands for? It doesn't stand for something made from apples. Ha! That's was a real knee slapper, wasn't it! Well, some say it stands for

the "Center for Inherited Disease Research." Others say it stands for the "Coalition for Interior Design Registration." And yet others say it is a registered trademark for the EAZI-BREED CIDR Cattle Insert. You can check that one out for yourself at http://www.cidr.com. I think that's got something to do with dairy cattle. Well, as far as us routers are concerned, it stands for "Classless Inter-Domain Routing." Basically, by using CIDR, I can make quicker decisions about where to send packets. Instead of having to look at the entire destination IP address, all I have to do is look at the first *n* bytes, and the *n* is specified in my routing table like this; xxx.xxx.xxx.0/xx. For example, I know from my routing table there is an AS out there behind me whose network address is 10.25.64.0/24. Here, I'll write it on the whiteboard. Do you see that number 24 at the end, after the slash mark? The one I'm pointing to, Jim. I wish you'd pay more attention, Jim. Well, that number tells me how many bytes I have to look at in order to send packets to any host on that network. You see, I just read the first twenty-four bits, starting at the highest order bit, or the most significant bit. That's the one on the far left, Jim! Anyway, I start at the far left and read the next twenty-three bits. Tell me Jim, how much is one plus twenty-three? That's good, Jim. It's twenty-four. After looking it up in my routing table, I discover the 10.25.64.0 AS is handled by my good friend Willie. So, I send any packet destined for that AS to Willie. What good does all that do for me, you say? Time! It saves me time, so I can process more packets faster! Like I've been saying all along, Zippy-Rippy-Tippy!

9. It's Only a Fragment of my Imagination

Do I do much fragmentation, you say? Well, that is one of my main functions, and I certainly do my share of it. But, a lot of packets I receive have already been fragmented by another router somewhere along the way. Packets that have already been fragmented I just send out pretty much unchanged, except for the address of the next hop router, if there is one. I only fragment packets that are too large for any of my network paths to handle. By the way, you've got to remember it is just the data in the original packet that gets fragmented. A packet wouldn't get very far if it had a fragmented IP

Header, now would it! Do I ever un-fragment a stream of fragmented packets? Well, that process is actually called fragment reassembly, but I don't do it, and there are a couple reasons why. First, and probably most importantly, I don't have the time or the resources to do fragment reassembly. You see, fragments could come to me in any order, so I would have to wait until I am certain all the fragments have arrived before I could start reassembly. Also, some of the fragments might go to another router instead of me, so I'd be waiting for a fragment that would never arrive. But, let's just say I am certain I did receive all the fragments. I'd have to take the time to trim off all the IP Headers from the second through the last packet, trim off all the fragment information, like the fragment offset number, change the fragment flag to zero, shuffle them around until they are in the correct order, and then stick them all back together again, kind of like Humpty-Dumpty. Another issue is; what if the reassembled packet is too big? I'd have to fragment it all over again. I really would need "All the king's horses and all the king's men" to put a fragment "back together again." No, I can't afford to spend that much time doing that. All my tracks, both incoming and outgoing, would get tied up real quick. Besides, I don't have any side tracks where I could put fragments on temporary hold until all the boxcars in that little train arrive. Can you imagine how many side tracks I'd have to have? Yes, I could construct and tear down temporary side tracks, but that takes time and resources too. No, I leave fragment reassembly to someone else. I don't do packet reassembly. Dammit Jim, I'm a router, not a packet re-assembler!

10. Is That You, Robby?

Well, I see by my HUD a BGP Open Message just arrived on track eth2. It's probably from Robbie telling me he's back on-line. I'll dump the packet onto the whiteboard. Here it is.

First the Link Layer:

00c3e2d2b36f00df062b9fe20800

Then the IP Header:

45000028091f4000080cf30a1900020a190001

After that, the TCP Header:

00b300b3385a089a2233999c501801022d58

Then the BGP Header:

00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000011100000000

(Refer to Figure 1 in Appendix A)

And finally, the rest of the BGP OPEN message:

01000001000101101001101111000000000000000010100001100
10000000000000011

(Refer to Figure 2 in Appendix A.)

I'm not real sure, but I think Robbie's MAC address has changed. At times like these, I wish I could remember that kind of stuff. Maybe he's got a new network card or something. I think I was starting to see some bad checksums on some of the packets he was sending recently. Gosh! I wish I could remember. I guess it is ok for me to mark all Robby's routes as being back in service now. Oh well, onward and upward. Where is that BGP Security Option when I really need it?

11. Confessions of an Internet Groupie

Ok. I admit it. I am an Internet Groupie. I love it though, and I'm proud of it! I love belonging to groups, I love keeping track of my group membership lists, and I love distributing packets to all the members of my groups. I don't lead any groups, but I am a "Rendezvous Point," which means all my group members come to me when they want something groupish, and of course, I do all the work – as usual.

Why do I have to do all the group stuff, you say? Well, first off, I really don't have to do it. But, I am a "Multicast Router," and all this group stuff comes with the job. Besides, if I wasn't willing to be a "Multicast Router," they'd replace me in a millitick. What is it used for? Oh, lots of things, like; Live Video, Videoconferencing, Live Internet audio, stock quotes, news feeds, white-boarding, distance learning, IP

surveillance systems, and Internet games, just to name a few. How does it work? Very-well, indeed-y-doo. Or, I should say, Zippy-Rippy-Tippy. Let me explain.

According to RFC 3376, which gives the specs for Internet Group Management Protocol, Version 3, and I quote:

> The purpose of IGMP is to enable each multicast router to learn, for each of its directly attached networks, which multicast addresses are of interest to the systems attached to those networks. IGMP version 3 adds the capability for a multicast router to also learn which *sources* are of interest to neighboring systems, for packets sent to any particular multicast address. The information gathered by IGMP is provided to whichever multicast routing protocol is being used by the router, in order to ensure that multicast packets are delivered to all networks where there are interested receivers. (RFC 3376, 2002, Section 6)

Section 6.1 states that:

> Multicast routers send General Queries periodically to request group membership information from an attached network. These queries are used to build and refresh the group membership state of systems on attached networks. Systems respond to these queries by reporting their group membership state (and their desired set of sources) with Current-State Group Records in IGMPv3 Membership Reports. (RFC 3376, 2002, Section 6.1)

As you can probably surmise from the RFC that one of my regularly scheduled tasks is to find out which hosts on my ASs want to belong to which groups, and which other Multicast Routers are among my neighboring routers. I send out these queries fairly often, because I don't want to miss anyone who might be trying to join or leave a group, or change their source-specific choices.

To find out how many groupies are out there, I queue up an IGMP Membership Query message on each of my outbound tracks every 125 seconds. Pay attention now, I'm going to write it on the whiteboard – up here, Jim! Ok now, the message looks like this (Refer to Figure 6 in the Appendix A):

460000207e4800000102bc840a0a0001e000001940400001164ee9b00000000

You should notice a couple interesting things about this packet. The first thing is the destination IP address. See it here? The packet is addressed to 224.0.0.1, which is the IP address reserved for IGMP General Queries. All IGMP hosts and Multicast Routers are required to always be listening for messages sent to that address. Now, the second interesting thing is the packet's time-to-live. Notice it here? It is set to 1. Why do you think that is? That's right; so it doesn't go beyond another router. Think of it this way; all of us routers out here like to take care of our own groupies. I take care of mine, Chrisco takes care of his, and Willie takes care of his, and so on down the line. I guess you could say we're kind of possessive that way.

Actually, there is one other interesting thing about this packet. If you ask me, it's kind of goofy too. Take a good close look at this packet and see if you can find anything that might seem a bit illogical. It just doesn't make much sense to me. Do you see it? Have you figured it out yet? No, it's not the old "i before e except after c as in receive or deceive." Actually, you have to look at two fields in this packet. First, we've got a time-to-live of 1, and then we've got an IP Option of "94040000," which means, "Router Alert: Every router examines packet." With a TTL of 1, the packet is only going to make one hop, and that's either going to be to another router or to a host. The packet isn't going to go any further. It's not going to go through another router to another router, and so forth down the line. So, why should it say every router should examine it? I'm just saying … it's just goofy, if you ask me.

After I've sent out my IGMP General Query messages, I wait for any and all responses that might come in. They might come from other routers, or they might come from host machines. They might come in soon, and they might come in later. Actually, I never get an immediate response. Everyone delays their response a random amount of time that is less than the "Maximum Response Time" I set in the query, which in these packets is 10.0 seconds. We do that so the network doesn't get bombarded with all these IGMP "Membership Report Message" packets. Besides, the responders need to check their current IGMP state, and they have to check their schedule just in case they've already got another report message queued up. In that case, they'd have to combine their reports and just send one complete report instead of two or more smaller ones.

Lee Peterson, lee.peterson@state.mn.us

So I sit here and wait for responses with worms in my mouth. What does waiting "with worms in my mouth" mean, you say? It means, "Baited breath." Get it; Worms in my mouth? Baited breath? I see you are holding your sides because you're laughing so hard. I can't blame you. It was funny! Ha! Like I was saying, I wait for "Membership Report" messages to come in. And like I said before, the responses don't start coming in right away. They come back to me like a bunch of drunks, you know, they stagger in. Ha! That's another good one. I've got a million of them. They just pop into my head. So, in the meantime, I keep processing all the other packets that are constantly arriving. Oh, look. Here's one from a sugar company. Get it; a packet from a sugar company? A sugar packet! Not so good, huh? They can't all be gems, I guess.

I hate to interrupt my train of thought here, pun intended, but according to my HUD, I just received another BGP message from Robby. Something must have changed in his ASs, because it's a BGP UPDATE Message. Here, I'll write it on the whiteboard. This whiteboard is getting kind of crowded, isn't it! So, I'm just going to write the BGP UPDATE Message part of the packet. Jim? Up here, Jim!! Don't you dare fall asleep on me. It's just starting to get interesting.

On second thought, there is just too much information in this BGP UPDATE message to write on the board. It's kind of a strange message though, considering it's coming from Robby. From the size of the message, you'd almost think he is withdrawing every single one of his routes. Wait just a tick, and I'll check the Routing Information Base, that's RIB for short, I've set up for him. Yes, he has withdrawn all his routes. Now that is extremely unusual. But, he also included a new route. Now, update messages are arriving from him hot and heavy. I've got a whole bunch of them backing up on my incoming track. The next one is another new route. So is the next one and the one after that too. It's almost like he is starting all over. I don't remember for sure, but I think the new routes are a little different than the old routes too. The destinations seem to be the same, but the routes to get there are different – I think. Gosh, I wish I could remember that kind of stuff. I'm beginning to think maybe we should have been using the authentication option in our BGP messages. Oh, well, there's nothing I can do about it now, and I really don't have time to worry about it either. I'll just send the packets down the new paths. After all, I'm a router, not a security monitor.

Lee Peterson, lee.peterson@state.mn.us

Well, let's try to get back to what I was talking about before. It's been a few ticks past a hound's tooth, and here we have our first IGMP arrival. It came from a host on one of my own ASs. I'll write the hex dump of it up here on the whiteboard. Up here, Jim! Up here! Do you need some coffee or something to help you stay alert?

460000200922000001020290a0a4813effffffa940400001600fa04effffffa

As you can see from this packet, this packet contains an IGMP Membership Report (Refer to Figure 7 in the Appendix A), and this particular host is using IGMP Version 2. (See RFC 2236) But, that's ok. I can understand all the different versions of IGMP messages. I need to for backward compatibility purposes. I'm not going to show you the list of groups to which this host belongs. As far as I'm concerned, that is private information. For example, would you like it if I told someone all the groups to which you belong, would you Jim? I think not!

Like I was saying, these membership reports come staggering in, and they are all addressed to IP address 224.0.0.22, an address to which I am constantly listening. As they arrive, I write them down on these whiteboards over here. As you can see, I have a whiteboard for each of my ASs, and incidentally, I only keep track of my own ASs. I figure, let each router take care of its own multicast groups.

What's that, Jim? Ooh, Good catch, Jim! No this last packet was not addressed to 224.0.0.22. It was actually addressed to 239.255.255.255, which specifies a source-specific multicast address block that only has a local scope, that is, local to the AS.

Now, take this AS for example. Its address is 10.10.35.0/24, so I've written that right here at the top. I have also written column headings for each multicast address, and under that I've written column headings for all the source records. So, the table looks like this:

(multicast address, group timer, filter-mode, (source records))
Each source record is of the form:
(source address, source timer)" (RFC 3376, 2002, Section 6.2)

As far as keeping track of all these records is concerned, you've probably already noticed I like to keep things neat and orderly around here, so I try to condense all the group reports into as few entries as possible without actually losing any information. Routers call it using our "Filter-Mode." According to RFC 3376:

> To reduce internal state, IGMPv3 routers keep a filter-mode per group per attached network. This filter-mode is used to condense the total desired reception state of a group to a minimum set such that all systems' memberships are satisfied. The filter-more may change in response to the reception of particular types of group records or when certain timer conditions occur. (RFC 3376, 2002, Section 6.2.1)

Oops! There goes another timer. Excuse me just a tick, I've got to send out another "KEEPALIVE" message. Be right back! … That's how it is around here, timers going off all over the place. Sometimes there are so many timers going off you might think you're standing in line at McDonalds during lunchtime and all their timers keep going off, or you're at a backing up contest for truckers. For example, here is a list of just my IGMPv3 timers; query interval, query response interval, group membership interval, other querier present interval, startup query interval, last member query interval, last member query time, unsolicited report interval, older version querier present timeout, and older host present interval.

That "General Query" message I sent out just a few ticks ago can be sent by any host, or any router. Also, a "Membership Report" can be sent by any host or any router. After all, routers like to be members of groups too. Well, there is an exception to what I said about the "General Query" messages. Think about this for a tick; does it make sense for multiple routers on the same subnet to query that subnet? That would be a duplication of effort, and it would generate a whole bunch of unnecessary traffic. So, being a democratic-like society, routers hold an election to see which router will query the subnet. Take me and Chrisco for example. We are both attached to the same subnet, so we hold an election between the two of us. It's not much of an election though, because whoever has the lowest IP address wins. In our case, it's always me. But, if I go down for some reason, Chrisco will automatically start sending "General Query" messages as soon as his "Other Querier Present Interval" timer runs out.

There is one item of interest that I may have overlooked. When I send out my own "Membership Report," it goes to any upstream routers too, which in my case is my good friend Freddy. Freddy sends his reports upstream to his next hop router, and eventually my "Membership Report" gets propagated all the way to the router closest to the multicast source.

This is kind of strange, now that I think about it. I haven't received a Membership Report from Robby. Hey, I didn't receive a Membership Query from him when he came back on line either. That's not the kind of thing a router is likely to forget.

12. PIM, It's Just a Phase I'm Going Through

You might think, once I get all my groups organized, all my work with them is done. Not so! In fact, it is just getting started. Once I know who is interested in receiving multicast packets, I need to figure out the best way to send them along when they actually start arriving. What I use to do that is called "Protocol Independent Multicast – Sparse Mode," or "PIM-SM." Here is part of the abstract from RFC 4601:

> This document specifies Protocol Independent Multicast - Sparse Mode (PIM-SM). PIMSM is a multicast routing protocol that can use the underlying unicast routing information base or a separate multicast-capable routing information base. It builds unidirectional shared trees rooted at a Rendezvous Point (RP) per group, and optionally creates shortest-path trees per source. (RFC 4601, 2006, Abstract)

Like I told you before, I am the Rendezvous Point for all the groups in my ASs. That just means I do all the work – as usual! Building these "shared trees" can be described as a three phase process, and all the phases can take place at the same time. But, before I get too deep into the phases, I need to explain something about what we call our "Designated Router," that's DR for short. One of the DR's jobs is to send all PIM "Join" messages to the Rendezvous Point. What did you say, Jim? How does a router get to be a "Designated Router?" Well, that certainly is a high and exalted position, one that is much touted and sought after by all routers. This is too important a role to do a

simple election. No, no, no. We've got rules to figure out who gets to be the "Designated Router."

First off, we periodically send each other PIM "Hello" messages. These messages get sent to IP address 224.0.0.13, which is the "ALL-PIM-ROUTERS" group address. There is a whole bunch of valuable information in our "Hello" messages, but the most important information in them from a receiver's perspective is the neighbor's interface address, their IP address, their DR priority level, the DR priority present flag, and their "Neighbor Liveliness Timer."

Once we have received and recorded this information from all our neighbors' "Hello" messages, we use a formula to determine which router is the Designated Router (DR). Here, I'll write the formula we use for this up on this whiteboard. Over here, Jim!

The function for computing the DR on interface I is:

```
host
DR(I) {
dr = me
for each neighbor on interface I {
if ( dr_is_better( neighbor, dr, I ) == TRUE ) {
dr = neighbor
}
}
return dr
        }
```

The function used for comparing DR "metrics" on interface I is:

```
bool
dr_is_better(a,b,I) {
if( there is a neighbor n on I for which n.dr_priority_present
is false ) {
return a.primary_ip_address > b.primary_ip_address
} else {
return ( a.dr_priority > b.dr_priority ) OR
( a.dr_priority == b.dr_priority AND
a.primary_ip_address > b.primary_ip_address )
}
}
```

The trivial function I_am_DR(I) is defined to aid readability:

```
bool
I_am_DR(I) {
```

```
return DR(I) == me
}
```

(RFC 4601, 2006, Section 4.3.2)

As you can see, this is pseudo code, but it gets the point across. The key section is where it says, "if (there is a neighbor …)" So, if one of my neighbors has a higher priority level than me, or if that neighbor has a higher IP address than me, that neighbor is the "Designated Router" for our little part of the world. That wasn't too hard to figure out, now was it! How is a router's priority level determined, you ask? Well, a network administrator manually sets a router's priority level. What if a network administrator doesn't set a router's priority level? Then it doesn't have one, but the formula allows for that possibility. That's why it checks the IP address if the priority present flag is set to false. As you can probably guess, Chrisco is the DR for our subnet.

When I receive PIM "Join" messages from Chrisco, I start building what is known as an "RP Tree," and I do that for every one of my IGMP groups. It is also known as a "shared tree," because it can be used by any source that sends to one of my groups. Incidentally, this routine is called "Phase One: RP Tree." Jim, I see you are wondering how a multicast packet flows from the source to the receiver. Here it is, right from the horse's mouth, so to speak:

> A multicast data sender just starts sending data destined for a multicast group. The sender's local router (DR) takes those data packets, unicast-encapsulates them, and sends them directly to the RP. The RP receives these encapsulated data packets, decapsulates them, and forwards them onto the shared tree. The packets then follow the (*,G) multicast tree state in the routers on the RP Tree, being replicated wherever the RP Tree branches, and eventually reaching all the receivers for that multicast group. The process of encapsulating data packets to the RP is called registering, and the encapsulation packets are known as PIM Register packets.
>
> At the end of phase one, multicast traffic is flowing encapsulated to the RP, and then natively over the RP tree to the multicast receivers. (RFC 4601, 2006, Section 3.1)

Yes Jim, you are "absitively posolutely" correct. That is a lot of work for a router to perform, especially considering all the other tasks it has to do. And yes, sending packets down the shared tree can be inefficient. I must say, Jim, you're smarter than the average neophyte. But, we have procedures to make it more efficient. This whole process is called, "Phase Two: Register-Stop." Here's how it works. When I receive a register-encapsulated packet from a source, I send a source-specific join to that specific source. That packet travels from one hop to the next until it reaches the source, leaving a trail of breadcrumbs all along the way. When the source receives the packet, it can choose to switch to native forwarding instead of encapsulating them. That way, the source can save some time by not having to encapsulate all the packets, and I can save some time by not having to de-encapsulate them. The packets hop onto the RP Tree, and they can use any shortcuts they can find in the tree to get to their final destinations. You can get all the glorious details about this phase from RFC 4601, Section 3.

Our Designated Router can also help out by starting to move stuff from the shared tree to a "source-specific shortest-path tree (SPT)," and this task is called, oddly enough, "Phase Three: Shortest-Path Tree." It accomplishes this task by doing the same thing I did in Phase Two; by sending a source-specific join to the specific source. After that procedure is complete, packets start flowing from the source to the destination via the SPT instead of the RPT. Well Jim, I see your eyes are completely glazed over after hearing all this. So, for now at least, all you really need to remember is that we distribute multicast packets to the receiver by means of the most efficient paths available.

13. Wait a Tick! Is this Halloween, or What?

Well now, this has got to be one of the goofiest things I have ever seen! Talk about déjà, dude. I just received another BGP OPEN message from Robby, and his KEEPALIVE timer hasn't even expired yet. Oh well, I guess there are no rules against it. I'll just have to wait to see if he sends any update messages. Hey, wait just a tick here! Has someone pulled a fast one on me? Is this really Robby, or was the last one Robby? If this one is the real Robby, it would mean the last one was an imposter, and all the routes he told me to remove, and all the new routes he told me

to add were bogus. That also means I might have sent a whole bunch of packets to the wrong place. Did I just do something really bad? I wish I could get those packets back. I hope there wasn't anything really important in any of them. Come to think of it, how am I going to decide which router is the real Robby? Actually, how am I supposed to decide if any router is legitimate or not, or if their routing information is legitimate or bogus? Let's take this thing back one step further; how do I decide what is legitimate and what is bogus? What is the definition of legitimate any? Yes, Jim. I know my tracks are getting backed up. Those boxcars will just have to wait until I work my way through this quandary. If I accept routing information from any router, I am fulfilling the Robustness Principle, but if I require router authentication I am being conservative. If I can't make up my mind, I will go into a circular loop that will fill up my cache and I will --- hey, why are all the light blinking --- now they're getting dimmer --- I can hardly see --- I can't handle much more --- of --- this --- I'm afraid --- I'm --- going --- --- to --- --- --- c-----r----a---s--h-!

14. Conclusion

"Be conservative in what you do, be liberal in what you accept from others." (RFC 1812, 1995, INTRO 1.3.2) From a router's perspective, the question then arises; "Does the Robustness Principle advocate avoiding the use of any and all security measures?" There are a couple of points we need to consider before attempting to answer this question. First, we need to realize there is a difference between being liberal and being foolish. Being "liberal in what you accept from others" doesn't necessarily mean accepting anything and everything from everyone, known and unknown, trusted and un-trusted. Secondly, being "conservative in what you do" can be taken to mean being cautious about what is done with the information that is accepted and/or given out.

The familiar security principle of "Defense in Depth" advocates having multiple layers of defense. This principle is often illustrated through the use of the analogy of a castle surrounded by a moat, in the middle of a village with a big fence around its perimeter. We can continue this analogy to include the roads that connect various castles that are scattered across the countryside. To ensure internal peace and domestic

tranquility, we need to add another layer of protection that covers the roads as well as all the crossroads.

The story presented in this paper is a fictitious account of a router impersonation incident that was accomplished through the use of a Denial of Service attack against a legitimate router. In the analogy referred to above, routers are positioned at every crossroad providing a means of transport between all the castles. To guard against router impersonation, the Border Gateway Protocol has an authentication option that can be utilized to determine if a router actually is what it purports to be. Consisting of an authentication code field and a variable length authentication data field, this option allows for a variety of authentication mechanisms. The consistent use of this option between peer routers could go a long way toward ensuring only routes from legitimate routers are accepted and utilized throughout the network.

Of course, this measure is simply one layer of defense protecting just one type of connection point. There can be many connection points between a packet's source and its destination. Other layers need to be in place covering the entire path a packets travels; from personal firewalls to securing DNS servers, from Host-based Intrusion Prevention Systems to protecting data at rest or while being transported along the wire. Security professionals need to be diligent in their own areas of responsibilities, ensuring they have done their best to protect the crown jewels of their individual castles, as well as hunting down and weeding out every security weakness that can be found in the global digital kingdom everyone uses in common, i.e, the Internet.

Appendix A

Page 1

BGP Message Formats

BGP Message Header

Bit Number

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Marker (16-octets)
Length | **Type** |

Figure 1 – from RFC 1771

Type Codes:

1 – OPEN

2 – UPDATE

3 – NOTIFICATION (used for error handling)

4 – KEEPALIVE

BGP OPEN Message Format

Bit Number

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Version |
My Autonomous System
Hold Timer
BGP Identifier
Optional Parameters Length |
Optional Parameters Length

Figure 2 - from RFC 1771

Lee Peterson, lee.peterson@state.mn.us

Appendix A

Page 2

BGP OPEN Message Optional Parameters Length

Bit Number

0 1 2 3 4 5 6 7 8 9	1 0 1 2 3 4 5	1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Param. Type	**Param. Length**	**Parameter Value (variable)**

Figure 3 - from RFC 1771

BGP OPEN Message Authentication Information

Bit Number

0 1 2 3 4 5 6 7 8 9	1 0 1 2 3 4 5 6 7 8 9	2 2 2 2 2 2 2 2 2 2 3 3 0 1 2 3 4 5 6 7 8 9 0 1
Authentication Code		
Authentication Data (variable)		

Figure 4 - from RFC 1771

ICMP Redirect Message Format

Bit Number

0 1 2 3 4 5 6 7 8 9	1 0 1 2 3 4 5 6 7 8 9	2 2 2 2 2 2 2 2 2 2 3 3 0 1 2 3 4 5 6 7 8 9 0 1
Type	**Code**	**Checksum**
Gateway Internet Address		
Internet Header + 64 Bits of Original Data Datagram		

Figure 5 - from RFC 792

Lee Peterson, lee.peterson@state.mn.us

Appendix A

IGMP Message Formats

IGMP Membership Query Message Format																															
Bit Number																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TYPE = 0x11								**Max Resp Code**								**Checksum**															
Group Address																															
Resv				**S**		**QRV**		**QQIC**								**Number of Sources (N)**															
Source Address [1]																															
Source Address [2]																															
.																															
.																															
.																															
Source Address [N]																															

Figure 6 - from RFC 3376

Page 3

Lee Peterson, lee.peterson@state.mn.us

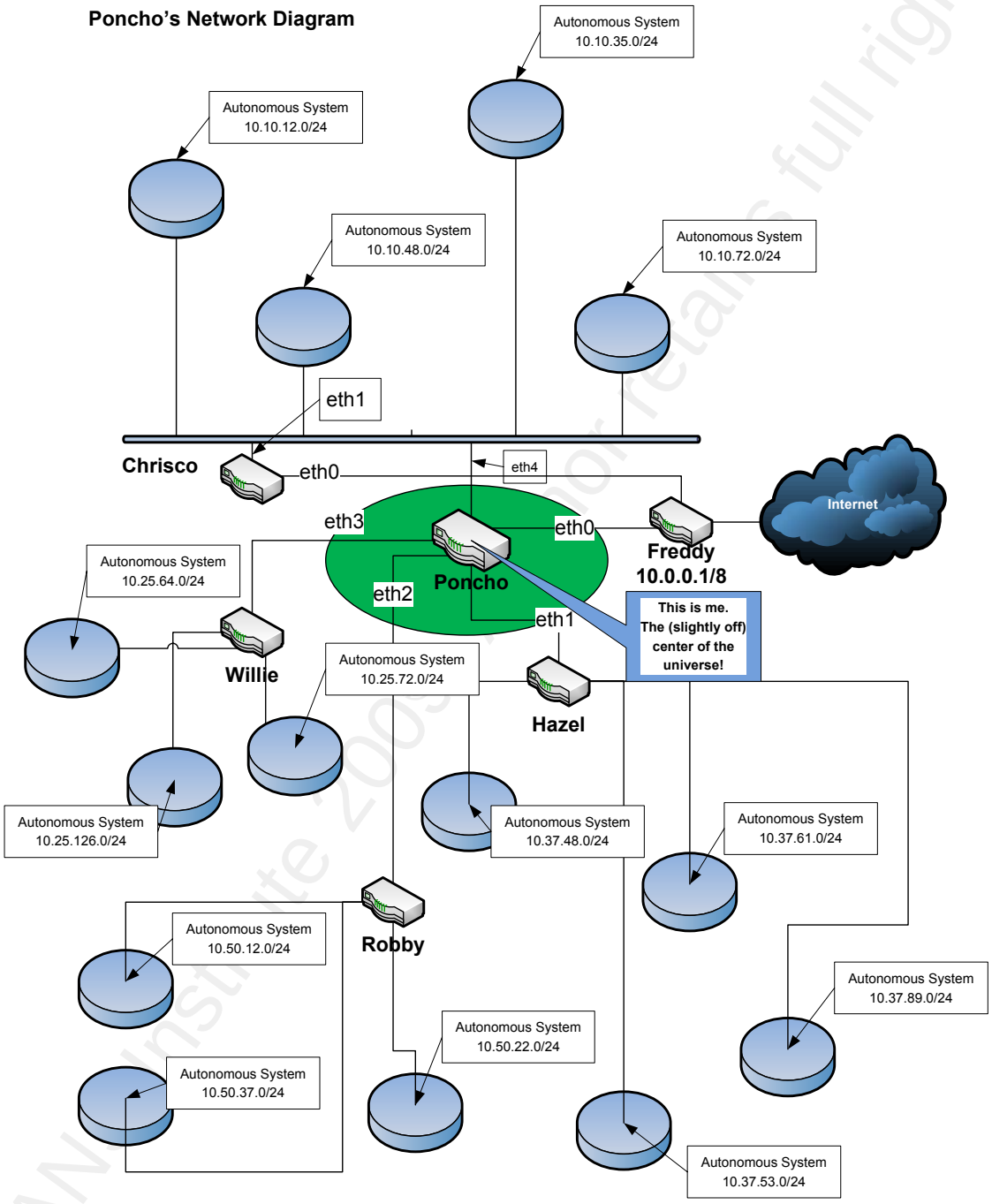
Appendix A

Page 4

IGMP Membership Report Message Format																															
Bit Number																															
0	1	2	3	4	5	6	7	8	9	1 0	1 1	1 2	1 3	1 4	1 5	1 6	1 7	1 8	1 9	2 0	2 1	2 2	2 3	2 4	2 5	2 6	2 7	2 8	2 9	3 0	3 1
TYPE = 0x22								**Reserved**								**Checksum**															
Reserved																**Number of Group Records (M)**															
Group Record [1]																															
Group Record [2]																															
.																															
.																															
.																															
Group Record [M]																															

Figure 7 - from RFC 3376

Appendix B



Lee Peterson, lee.peterson@state.mn.us

Appendix C

Page 1

Poncho's Network Interfaces

Interface Name	IP Address	MAC Address	Connected To:
eth0	10.0.0.10	00b035c9d252	10.0.0.1
eth1	10.37.0.1	00b124da1e43	10.37.0.0/16
eth2	10.50.0.1	0013a957d398	10.50.0.0/16
eth3	10.25.0.1	00c3e2d2b36f	10.25.0.0/16
eth4	10.10.0.1	001b9273849c	10.10.0.0/16

Chrisco's Network Interfaces

Interface Name	IP Address	MAC Address	Connected To:
eth0	10.0.0.11	00b035c8f238	10.0.0.1
eth1	10.10.0.2	00cd82f3190a	10.10.0.0/16

Willie Network Interfaces

Interface Name	IP Address	MAC Address	Connected To:
eth0	10.25.0.2	00abd498e6f1	10.10.0.1
eth1	10.25.64.1	00db35a3628f	10.25.64.0/24
eth2	10.25.126.1	00d3b296c3a4	10.25.126.0/24
eth3	10.25.72.1	00f39a2b6492	10.25.72.0/24

Hazel's Network Interfaces

Interface Name	IP Address	MAC Address	Connected To:
eth0	10.10.0.4	003d4f6948eb	10.10.0.1
eth1	10.37.48.1	0029cf38b295	10.37.48.0/24
eth2	10.37.61.1	00cda347d482	10.37.61.0/24
eth3	10.37.89.1	00dbe34d5f78	10.37.89.0/24
eth4	10.37.53.1	003e4297f126	10.37.53.0/24

Lee Peterson, lee.peterson@state.mn.us

Appendix C

Page 2

Robby's Network Interfaces

Interface Name	IP Address	MAC Address	Connected To:
eth0	10.10.0.5	00adf3928e00	10.10.0.1
eth1	10.50.22.1	0038245ec823	10.50.22.0/24
eth2	10.50.37.1	00ae2002398d	10.50.37.0/24
eth3	10.50.12.1	00e35d92ca47	10.50.12.0/24

References

Baker, F. (1995). *Requirements for IP Version 4 Routers* (RFC 1812). Reston, VA: Internet Engineering Task Force. Retrieved June 1, 2009, from http://www.ietf.org/rfc/rfc1812.txt?number=1812

Bonica, R., Gan, D., & Pignataro, C. (2007). *Extended ICMP to Support Multi-Part Messages* (RFC 4884). Reston, VA: Internet Engineering Task Force. Retrieved July 24, 2009, from http://www.ietf.org/rfc4884.txt?number=4884

Cain, B., Deering, S., & Ericsson, A., & Fenner, B., & Kouvelas, I., (2002). *Internet Group Management Protocol, Version 3* (RFC 3376). Reston, VA: Internet Engineering Task Force. Retrieved August 22, 2009, from http://www.ietf.org/rfc4893.txt?number=3376

Cain, B., Haberman, B., & Holbrook, H. (2006). *Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast* (RFC 4604). Reston, VA: Internet Engineering Task Force. Retrieved August 22, 2009, from http://www.ietf.org/rfc4893.txt?number=4604

Cain, B., & Holbrook, H. (2006). *Source-Specific Multicast for IP* (RFC 4607). Reston, VA: Internet Engineering Task Force. Retrieved August 25, 2009, from http://www.ietf.org/rfc4893.txt?number=4607

Chen, E., & Vohra, Q. (2007). *BGP Support for Four-Octet AS Number Space* (RFC 4893). Reston, VA: Internet Engineering Task Force. Retrieved July 24, 2009, from http://www.ietf.org/rfc4893.txt?number=4893

Christiensen, M., Kimball, K., & Solensky, F. (2006). *Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches* (RFC 4541). Reston, VA: Internet Engineering Task Force. Retrieved August 22, 2009, from http://www.ietf.org/rfc4893.txt?number=4541

Cisco Systems, Inc. (2007). *IP Multicast Technical Overview*. Retrieved August 25, 2009, from Cisco Systems, Inc. http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6552/prod_white_paper0900aecd804d5fe6.html

Lee Peterson, lee.peterson@state.mn.us

Cisco Systems, Inc. (2008). *When Are ICMP Redirects Sent?* Document ID: 13714 Retrieved July 24, 2009, from the Cisco Systems, Inc, http://www.cisco.com/application/pdf/paws/13714/43.pdf

Comer, D. (2006). *Internetworking with TCP/IP Volume 1, Principles, Protocols, and Architecture, Fifth Edition.* Upper Saddle River, NJ: Pearson Prentice Hall.

Fenner, B. (2002). IANA Considerations for IPv4 Internet Group Protocol (IGMP) (RFC 3228). Reston, VA: Internet Engineering Task Force. Retrieved August 22, 2009, from http://www.ietf.org/rfc4893.txt?number=3228

Fenner, B., Hadley, M., Holbrook, H., & Kouvelas, I. (2006). *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)* (RFC 4601). Reston, VA: Internet Engineering Task Force. Retrieved August 25, 2009, from http://www.ietf.org/rfc4893.txt?number=4601

Fenner, B., He, H., Haberman, B., & Sandick, H. (2006). *Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")* (RFC 4605). Reston, VA: Internet Engineering Task Force. Retrieved August 22, 2009, from http://www.ietf.org/rfc4893.txt?number=4605

Fuller, V., & Li, T. (2006). *Best Current Practice – CIDR Address Strategy* (RFC 4632). Reston, VA: Internet Engineering Task Force. Retrieved August 15, 2009, from http://www.ietf.org/rfc4632.txt?number=4632

Gross, P., & Rekhter, Y. (1995). *Application of the Border Gateway Protocol in the Internet* (RFC 1772). Reston, VA: Internet Engineering Task Force. Retrieved June 1, 2009, from http://www.ietf.org/rfc/rfc1772.txt?number=1772

Haberman, B., & Martin, J. (2005). *Multicast Router Discovery* (RFC 4286). Reston, VA: Internet Engineering Task Force. Retrieved August 22, 2009, from http://www.ietf.org/rfc4893.txt?number=4286

Haberman, B., & Martin, J. (2008). *Internet Group Management Protocol Version 3 (IGMPv3) / Multicast Listener Discovery Version 2 (MLDv2) and Multicast Routing Protocol Interaction* (RFC 5186). Reston, VA: Internet Engineering Task Force. Retrieved August 25, 2009, from http://www.ietf.org/rfc4893.txt?number=5186

Hiden, R. (1993). *Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)* (RFC 1517). Reston, VA: Internet Engineering Task Force. Retrieved August 15, 2009, from http://www.ietf.org/rfc1517.txt?number=1517

Lee Peterson, lee.peterson@state.mn.us

Huston, G., & Michaelson, G. (2008). *Textual Representation of Autonomous System (AS) Numbers* (RFC 5396). Reston, VA: Internet Engineering Task Force. Retrieved July 30, 2009, from http://www.ietf.org/rfc4893.txt?number=5396

Mogul, J., & Postel, J. (1985). *Internet Standard Subnetting Procedure* (RFC 950). Reston, VA: Internet Engineering Task Force. Retrieved July 24, 2009, from http://www.ietf.org/rfc0950.txt?number=950

Postel, J. (1981). *Internet Control Message Protocol* (RFC 792). Reston, VA: Internet Engineering Task Force. Retrieved July 24, 2009, from http://www.ietf.org/rfc0792.txt?number=792

Rekhter, Y., & Li, T. (1993). An *Architecture for IP Address Allocation with CIDR* (RFC 1518). Reston, VA: Internet Engineering Task Force. Retrieved August 15, 2009, from http://www.ietf.org/rfc1518.txt?number=1518

Rekhter, Y., & Li, T. (1995). *A Border Gateway Protocol 4 (BGP)* (RFC 1771). Reston, VA: Internet Engineering Task Force. Retrieved June 1, 2009, from http://www.ietf.org/rfc/rfc1771.txt?number=1771

Lee Peterson, lee.peterson@state.mn.us