PACKET ATTACKS - VERSION 1.1

Let me start by saying the internet is full of wonderful tools and papers like this one. Alot of these things
can help you
increase your knowledge, perhaps your job or more. But just as easily as you can learn from them, people read
into them to
much and decide to harm other peoples work for no apparent reason. Let it be known that is in no way the
purpose of this
paper. A true hacker is one who strives to attain the answers for themselves through curiosity. Its the path
we take to
those answers that makes us hackers, not destruction of other peoples work. So with that said, please enjoy my
work, as I
have enjoyed writing it.

The flow of data has always captured my interest. Just how does it work, how can we dissect it and use it to
our advantage.
Well I have spent a long time studying all of this, and that is why I wrote this paper. It's a collection of
run on s
entences on different packet attacks and how they work. Now we all know you can learn all you ever wanted to
know about the
specifications of a protocol by reading its 30 page RFC document. But that is the protocol according to
design, in the wild
its a different story all together. 'Packet Attacks' covers everything from basic DOS attacks to TCP/IP
hijacking. Hence the
name "Packet Attacks". This paper also focuses not just on attacks but practical ways to prevent such attacks
and ideas on
new methods to help us stop them and secure our networks.

Introduction.

Well I assume most of you reading this paper already have a good understanding of TCP/IP and how it works so I wont get to
much into detail on that, but I will scrape the surface on the parts we NEED to discuss. The internet is a MASSIVE web of
machines all connected to one another through a series of hardware devices known as routers, switches, hubs, bridges and
lots more. All of these devices (although some are smarter then others) push along packets. Our operating systems and
applications craft these packets in order to send data to one another over the wire. Each packet, although varying in size,
carries a small bit of data to and from one host to another. Each packet must also carry its own personal information such
as where it came from and where its headed. Of course there is a lot more to a packet then just this information. But as far
as attacks go this is the crucial information we need to look at. Now there are many many different types of protocols that
craft many different types of packets. And they are all read differently when they are received at the other end. Where as
an ARP packet may tell a host who has this MAC address on this subnet, a TCP packet might transfer the last

few bits in that
MP3 your downloading. Regardless the data, all of these packets use the same wire to move to and from locations. I couldn't
possibly discuss every protocol and packet structure in this one paper. The average end user takes for granted all of this
running in the background while they surf the net. Most people dont understand the complexity of this internet we are all so
familiar with, the chat rooms etc. But there are people who do, and there are people who take advantage of that. Reverse
engineering has led to the creation of attacks using the basic fundamentals these protocols rely on. And since TCP/IP is so
embedded in our infrastructure we must adapt and learn to defend each new attack.

OSI MODEL

Open Systems Interconnection model, is a seven layered networking design. Its an industry standard that defines exactly how
data is transffered between protocol to protocol. Not every protocol follows the OSI model exactly and some do. TCP the
internets main mode of data transport does not follow it exactly. Let me take you through a brief over view of the OSI model.

Layer Seven : Application Layer
This layer is obviously application specific, it provides everything from authentication to email to ftp and telnet, the
list goes on. Its specifically for end user processes, what we input into our applications we can see on our screens.

Layer Six : Presentation Layer
This layer changes and possibly encrypts the data so that the application layer can understand it. (you will understand what
this means in a few minutes)

Layer Five : Session Layer
Think of this layer as Establishment, Control and Termination of the sessions formed by the application(client) to a remote host(server).

Layer Four : Transport Layer
This layer is responsible for the invisible transfer of data between host to host. It is there to ensure all data transfer
goes accordingly. The protocols used are, UDP and TCP.

Layer Three : Network Layer
This layer is for error correction, packet sequencing, and for transmitting data from node to node. Addressing is also
another function of this layer in inter-networking.

Layer Two : Data Link Layer
This layer decodes and encodes packets into bits so they are ready for the physical layer. It also handles error correction
in the physical layer. This layer is also divided into two different sub-layers. The LLC (logical link control) and MAC
(media access control) sub layers. The LLC sub layer provides control for frame synchronization and error checking. The MAC

sub layer controls how a computer on your network has access to data.

Layer One : Physical Layer
This layer is the actual movement of the data. Using electrical impulse or some other form of data movement is pushes the
bit stream towards the other host. This layer is the hardware level, the ethernet card, the wire etc. There are many
protocols within this layer.

You may ask yourself why I listed these from 7 to 1. Well I did to show you how the OSI model really works. Layer Seven
really comes first, the end user types something into his instant messenger (for example) and the data flows down through
the OSI model being encapsulated and changed at every level it has to be changed or corrected at. The data travels the wire
and at the other end it moves back up the OSI model all the way back up to layer seven where the other host can read it in
the original form it was sent. So theres a VERY basic understanding of the OSI model and how it works to transmit data from
host to host. There is alot more protocols and parts to the OSI model but this basic representation should provide a firm
understanding.

To understand all of this more in depth please get your hands on a few RFC (request for comment) documents and start reading.
Because it will take you a very long time to understand exactly how TCP/IP works. If your very knowledgeable in the way
TCP/IP works then this paper should make alot of sense to you, perhaps even bore you! :(On the other hand if you dont
understand TCP/IP as well as you would like to, you still might get something out of this. I try and explain all of the
technical writing as easily as I can. Feel free to email me if you have a question or comment. Thanks :)
Data_Clast

Chapter 1.

Section a.

The most common attack on the internet today is a denial of service attack. There are many programs on the internet today
that will assist anyone in crafting one of these attacks. The sad part is for as easy as they are to make their power can be
destructive when used properly. No matter what kind of packet attack it may be most are based on the same principal, volume.
Thousand and thousands of spoofed packets will eat up network resources within minutes, choking and essentially 'killing'
any network. There are many types of packet attacks. Some are more sophisticated then others. I will also talk about TCP/IP
hijacking and your typical port and vulnerability scans among other things.

Why do people launch these attacks? How are they launched? How do they exactly (technically speaking) 'choke a network'?!
Hold tight im getting to that. The lower end of these attacks are usually launched by what the hacker

community calls a
script kiddie. You see a hacker isnt a mindless web defacing juvenile (please see the mentors manifesto). A
hacker is a
person of true intellect and would never craft such an attack for no reason. But these lower end attacks are
usually
launched at peoples individual machines. Their IP address's may come from an IRC chat room, yahoo messenger,
AOL, ICQ, or
whatever other messenger you might use. Although not as sophisticated, these 'lower end' attacks can still
knock an
individual machine offline in minutes. The slightly more advanced attacks may be aimed at a business
competitor in order to
slow their sales or disrupt their outgoing internet connection. Whatever the reason may be they are usually
launched for a
reason. Attacking a box for no reason is typically useless and will only take up your own bandwidth.

The more sophisticated attacks are aimed at government and root points of the internet. Such as the attacks on
the root DNS
servers in October of 2002. These attacks were sophisticated in the way they were crafted. The attacks lasted
for over an
hour and successfully took out a few of the servers. If the attack had lasted just a few more minutes who
knows the damage
it could have caused. The possibility of the authorities solving these attacks and apprehending the offenders
is slim to
none because they are created and launched by skilled malicious individuals. They were also distributed denial
of service
attacks. Which means the 'zombie' machines that attacked the servers were spread out all over the world. We
will touch more
on that later though.

Section b.

You will learn more about how these individual attacks are crafted and how they work later in this paper but
this is
small introduction so you can get a vague idea. Creating spoofed packets requires an open socket. This socket
binds to an
IP and a port and allows you to inject a packet onto the wire or accept any incoming packets to that IP and
port. *NIX
openly supports open socket programming (many tutorials on this type of programming). Which means you can code
programs that
create packets and then inject them into the network with ease. An example of this would be a program called
"SENDIP" which
allows you to create custom packets, and it supports many protocols (another good program is nemesis). I have
written a few
tutorials using SENDIP, I think its a great program for both advanced and new network engineers to use. It
will help you
learn about packet structure and the different protocols it supports. Microsoft is not an open source company,
which pretty
much makes it even harder to find help in creating these sorts of programs for Windows. But it is possible to
craft these
attacks from within a Windows environment. Its referred to 'Winsock' programming. Infact most of these DDOS
attacks are
because of vulnerable Windows boxes out on the net. They are sitting ducks for trojan horses and other
programs that craft
these attacks on servers when commanded from a client program to do so. Most end users do not understand

security and how
easy it is to break into someones home computer, so they lack firewalls and virus scanners. This leads to many zombie
machines available to hackers disposal on the net. All one has to do is scan a class C subnet for open trojan ports and
hack their way into those trojans and use them as a backdoor, another zombie is created for attacking remote targets. Almost
every program that interacts with TCP/IP generates packets to and from places, this is valid traffic. As you read you will
distinguish the difference between valid and non valid, as it easy pretty easy to understand what I am explaining when I say
"attack". When creating an open socket and crafting spoofed packets these programs tell the kernel they are going to
construct their own IP headers. Usually this information is put on by the kernel before exiting the machine. But in this
instance we are telling the kernel we want to specify our own information. Not all operating systems will allow this. And
no I dont have a detailed list of which do and which dont. Most of the experiments I have conducted on my network used
different versions of RedHat Linux, Mandrake Linux, and Windows XP.

Chapter 2.

Section a.

There are several different types of packet attacks. Theres the simple brute flood of ICMP packets which floods a network
and eats up all the available bandwidth. And then there are more sophisticated attacks like the Smurf or SYN/ACK attack.
All of these attacks target different things. While the SMURF attack may target the general network its attacking, the
SYN/ACK attack targets a specific host or service running on a host. We also must take into consideration when a target is
attacked it may not be the only machine affected. There are many routers and other boxes transfering the data between point
A and point B. Other peoples legitimate data is flowing between them, and may be disrupted by the packet flood. Even a top
of the line router can only handle so much data. And unfortunately it is very easy to attain soure code for these attacks
all over the web. Lets take a more detailed look at each attack.

ICMP brute flood attack.

ICMP works on top of TCP. The ICMP protocol is simple yet very effective. Its used for error correcting and testing network
connectivity. Your average PING program uses ICMP packets to test network connectivity. By sending a small amount of
arbitrary data in an ECHO_REQUEST packet it waits for a reply from the target host, simple right? A typical ICMP packet is
called an ECHO_REQUEST. You send 4 or 5 of these at a target machine and when it arrives there it requests an ECHO_REPLY.
Thats when everything is done according to design. If you want more info on an ICMP packet and how it works

then read my
tutorial on that!
http://www.theory-x.org/dataclast/_content/MPS.txt

In this attack the source IP address is spoofed. So now hundreds, thousands of ECHO_REQUEST packets rush towards their
destination. They reach point B, request an ECHO_REPLY for every ECHO_REQUEST sent. Point B says OK, reads the source IP.
The source IP ends up being unreachable. But point B is waiting a small amount of time (milliseconds) to determine that for
every packet thats hitting it. It will be a few more moments before the process relinquishes this small bit of memory back
to the system. This adds up to a great deal of packets and memory allocation building up. Now if these packets are coming
from multiple source zombies (DDOS) then this means there each coming from different routes. So even if one ISP stops one
attack, there are still many more zombie machines attacking the victim. All of this is eating up time and bandwidth, because
with every millisecond that passes more and more bandwidth is being taken up. Eventually point B can no longer keep up with
the ECHO_REQUESTS and his connection is completely flooded and of no use. On an unprotected system or router this attack
can be very consuming. This attack is also sometimes referred to a bandwidth attack. Even if the target is running an
advanced firewall it cannot protect the wire it connected to from being flooded with packets. There have been changes in
this attack as well. On the net there are what we call amplifiers. On every network there are the network and subnet
addresses. In many default configurations when you ping either one of these addresses they multiply the echo requests by 4
or more. So a zombie would attack a vulnerable network (.0) or subnet address (.255) with a spoofed source IP, being the
victims real IP. So even tho the traffic becomes valid as far as IP addresses go. The victim gets bombarded with massive
ECHO_REPLY packets. You will see more of this description in other attacks, as it works for some of those to.

```
[zombie machine] -->ICMP ECHO_REQUEST (source IP = 1.1.1.1) -->-->--> [target]
[??????????????]          ICMP ECHO_REPLY (destination 1.1.1.1 ?)<-- [target]
```

Hopefully that simple drawing shows you exactly how this attack works. Its very very simple, massive ICMP packets with
spoofed address's taking up network resources. The simplest of attacks.

Smurf attack.

(first part is repeat from ICMP attack) There have been changes in the ICMP attack. On the net there are what we call
amplifiers. On every network there are the network and subnet addresses. In many default configurations when you ping either
one of these addresses they multiply the echo requests by 4 or more. So a zombie would attack a vulnerable network (.0) or
subnet address (.255) with a spoofed source IP, being the victims real IP. So even tho the traffic becomes valid as far as

IP addresses go. The victim gets bombarded with massive ECHO_REPLY packets. You will see more of this description in other
attacks, as it works for those to.

You can try this attack on your home network by simply opening a packet sniffer on each machine that is on. Pick a machine,
any machine and ping your broadcast address. Mine is 192.168.0.255 Immediately you see each machine receiving a broadcast
packet. Now imagine its several hundred and each one has a spoofed source IP address. Its a brute ICMP attack on a massive
scale, this possibilities to this attack are endless. You could easily implement this attack in anyway you chose. You could
spoof the victims real IP as your source IP and create massive volumes of legit ECHO_REPLY packets. Even though its valid
traffic, its 4x or more times the normal load of valid traffic. This consumes the connection and valid traffic cant pass,
or passes so slowly it makes no difference to the end user.

[zombie machine] --> ICMP ECHO_REQUEST source ip = 10.2.2.2 --> to: broadcast router 4.1.0.255 (router multiplies the
ECHO_REPLY packets by 4x! --> --> --> --> [victim 10.2.2.2]

SYN/ACK attack.

The SYN/ACK attack is a very powerful attack. SYN/ACK packets are also used in TCP hijacking, and the TCP/IP three way
handshake. When an application wants to connect with a server somewhere over the net via a TCP connection (connection vs
connectionless data transfer (UDP)) it first sends a SYN packet. The SYN packet tells the target machine he wants to make
a connection on a certain specified port, and then send data. When the target machine read the SYN packet it replies to
the original host with a SYN packet of his own and an ACK (acknowledgement) packet with sequence and ack numbers. These SEQ
and ACK numbers are used to synchronize the data transfer, incase one or two packets gets lost or slowed down along its
route, it can be assembled again in the correct order. The orignal machine replies again with another SYN ACK packet
combination acknowledging the sequencing numbers and then it starts to send data. When it creates this connection a tiny
piece of memory is allocated to hold the connection while the packets are in route. Now a SYN/ACK attack would consist of
spoofing the source IP address on the original SYN packet. The target receives the request for a connection, reads the
spoofed source IP and tries to send its own SYN and ACK packet to a destination that does not exist. Most operating systems
will continue to send SYN/ACK packets if they dont receive a reply as a method of error correction and guaranteed data
delivery. Just like in the ICMP attack the machine has to wait a few milliseconds before abandoning all hope of reaching
the machine. So these tiny allocated spaces of memory are building up with every spoofed packet that arrives at the target.
This attack is very powerful and can disable a service running on the target machine in a matter of minutes.

Not to mention
all the available bandwidth is eaten with thousands and thousands of spoofed packets. So there is the SYN/ACK attack in a
brief description.

[zombie machine] --> SYN packet (source IP 1.1.1.1, port = 23 telnet) (seq = 100) --> [target]
[??????????????] <-- SYN/ACK packets sent (seq = 300) (ack = 101) <-- [target]

As you can see from the simple drawing above the target machine has no idea who is sending the SYN packets and the telnet
server he is running on port 23 would most likely crash. At best the telnet daemon would not allow any other legitimate
traffic through, as it could not gather enough resources (memory, bandwidth) to make the connection due to all the spoofed
packets.

Another use of this attack is to disconnect a user from their current TCP session. By spoofing SYN/ACK packets to a server
a client is currently using. An attacker would place a "FIN" flag in the packets, this tells the server the client is done
sending data. Client uses his connection and attacker walks away undetected, because it only took one packet to accomplish
this.

UDP attack

UDP is a protocol that is used to transfer data. Short for USER DATAGRAM PROTOCOL. UDP offers very little error correction
and is used as an alternative means for data transfer. It doesn't require the 3 way handshake such as the SYN/ACK method,
so its initial attack may not take down a remote daemon as quickly. UDP is generally used to broadcast messages over a
network. A UDP attack would consist of spoofing the source IP addresses and specifying a port number like in the SYN attack
above. UDP packets are generally large because they are usually used on closed 100mb subnets (LANS). So an attack would set
flags in the packets and fragment them (break them up and flag where in the packet they broke, so they can be reassembled
on the receiving end). For example in Windows 2000 there was a remote UDP DOS exploit that used the IKE service running on
port 500. All an attacker had to do was connect to port 500 on a random machine with that port open. Start sending massive
UDP packets (above 500 bytes) to that service and the CPU usage would hit 99% and the machine would lock up. The typical
ports that accept UDP packets are 7, 13, 19 and 37 on a Windows box.

DNS attack

The DNS attack is a special one. Not as easily crafted as the others, there arent that many tools readily available to the
average script kiddie to construct such an attack. The DNS protocol is used for name resolution, 216.239.35.100 = google.com,
simple as that? Well not really. A DNS attack is based on the fact that a DNS query takes very little data and

bandwidth to
create, but a DNS response is much bigger. So this is how a DNS attack would look like.

10.10.10.10 = victims IP

[dns query packet (who is google.com)] --> source IP is 10.10.10.10 --> [dns server]
[dns server] --> --> --> [dns response] [dns response] [dns response] --> [victim]

As you can see the attack is sort of relayed from a legitimate DNS server. Although the DNS response packets are 'legit'
there is a massive flood of them because the DNS server that is sending them is a very good machine on a very good
connection. The end user, most likely a home pc, gets flooded with these huge DNS response packets it never asked for.

ARP attack

The arp attack is a special one, it can be used to 'hijack' a tcp connection currently in session or it can be used to
sniff the legitimate traffic on a wire other then your own. Which is a very dangerous thing in the information world we
live in today. There are a few methods of this attack. Lets say person1, attacker, and server are all on the same subnet.
Person1 and server currently have an FTP session open. Attacker sends both server and person1 an ARP packet containing an
invalid MAC address. Now both of their arp tables are messed up for atleast 30 seconds. Server and person1 cant find that
invalid MAC address so they send their data to the IP its associated with, the attacker. So in this case the attacker has a
sniffer setup and hes collecting a ton of data. Now the attacker (an advanced one at that) can issue commands as person1 to
the server. This attack takes timing and skill to pull off on the internet, but on a LAN its very easy. It only allows for
maybe 30 or so seconds of sniffing, until their arp table is constructed properly again.

DRDOS attack

A DRDOS attack uses a little of other attacks to inflict damage. This attack spoofs the source IP address of SYN packets
to the IP of the victim. It requires a third party. This is the part of the attack that makes it so easy. All it needs is
some ftp, webserver, telnet.. ANY service that will reply with an ACK packet, anywhere on the internet. Could be angelfires
free ftp servers, could be your neighbors web server running off his 233mhz compaq with IIS 4.0. It doesn't matter! The SYN
packets are sent to that services IP address and they of course reply with a steady stream of SYN/ACK packets to the victim.
Most likely directed towards an open port on the victims machine, crashing that service and the system. These attacks are
near impossible to track down. This attack is quite possibly the strongest DOS attack in my opinion. For every SYN packet
you send the middle man, it sends out up to 4 SYN/ACK combinations to the victim. And each time the victim doesn't respond
the middle man sends even more (error correction). This allows the attacker to contruct a massive attack from

just one
machine with a broadband connection. There are more dangers to this attack as well, there are hundreds of thousands of FTP,
webservers and many more services running on the net today that will deflect these SYN/ACK packets at the victim. So in
theory this attack could use any number of 'middle man' servers to bombard your network with packets.