Chapter 3

System Level Determinants of Availability & the Impact of DoS Attacks.

3.1 Introduction

From the perspective of the user there are two views of an Information System, one is the external view of the system i.e. the set of services and functionalities the system provides to the users of the Information System. The other is the inside view of the Information System i.e. the design and architecture of the system, how the different software/hardware components interact with each other in order to provide the services and functionalities to the users of the information system. The external view is also called the system level (service level) view in the realm of information system technology. The well-established principles / attributes at the service level that determine/impact Availability of an Information System existent in theory and practice are *Reliability, Timeliness* and *Accessibility* [76]. The determinants provide us with a platform to understand, analyse and measure Availability of an information system at the service level with the help of certain well known metrics.

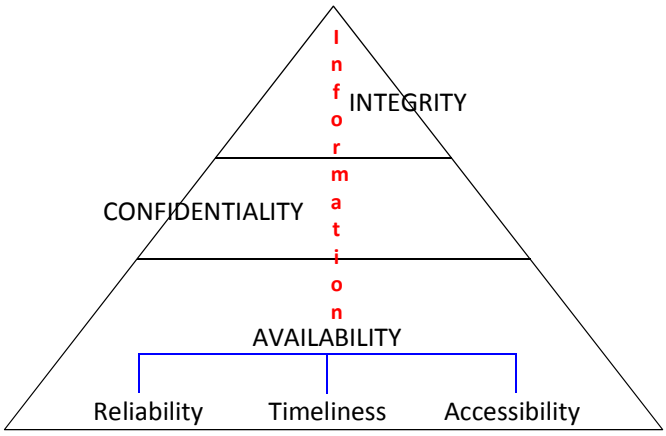

Figure 3.1: System level determinants of Availability and the CIA triad.

Linked to the system level determinants that impact Availability there is also a second line of factors that can impact *Availability* indirectly [76] (also known as the second order determinants) i.e. *Security Policy*, *Physical Security, Auditing and System Effectiveness Evaluation, Redundancy, System Monitoring and Operational Controls, Backups* and *Business Continuity.* Figure 3.1 presents the picture of Availability w.r.t determinants and other security attributes of the CIA triad. The determinants,

Reliability, Timeliness and Accessibility and the respective metrics (metrics described in next section) are very critical in understanding, measuring and analysing *Availability* of an *Information System.* But in practice whenever Availability is discussed the security practitioners and stakeholders are more inclined towards first two determinants i.e. Reliability and Timeliness. Accessibility is certainly not ignored but is discussed the least and is not taken as seriously as a measuring entity as the first two are taken. This surely does not mean Accessibility is not important, Accessibility describes more the behavioural aspect of the system rather than a serious system defining metric. The focus of the chapter is to analyse how the Availability at the system level is impacted by DoS attacks. For this purpose a detailed understanding of the system level factors that impact Availability is presented first and followed by an experimental evaluation of the facts established during the understanding of the factors.

3.2 Determinants

The attributes that determine Availability of an Information System at the service level are:

3.2.1 Reliability

Reliability is the extent to which an information system performs its expected function over a given duration of time [181 and 182]. Reliability is not the only factor or the lead factor that impacts availability and it should be noted that the measurement of reliability alone cannot be taken as the measurement of availability of an information system i.e. 99% reliability of an information system does not mean 99% availability of the information system. Reliability is the ability of an information system to perform its function nonstop while as the goal of Availability is much broader and is the ability of the information system to provide services to the legitimate clients whenever and wherever demanded. Reliability of an information system provides us with a metric that tells us about the failures of a component. The component (Hardware/Software) is most reliable when the component is in its “Useful Life” as shown in the figure 3.2.

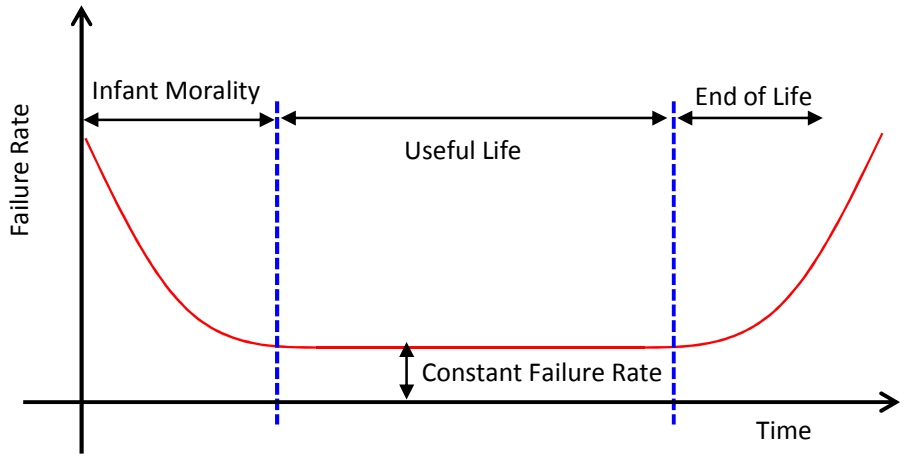

Figure 3.2: Bathtub Curve showing the failure rate of a Component in a System.

It is the most stable time in the life cycle of the component as the component shows a constant failure rate. While as the time periods "Early Life or Infant Morality" and "End of Life" are the most unstable ones, therefore we can say more unreliable as compared to when the component is in the "Useful Life" phase.

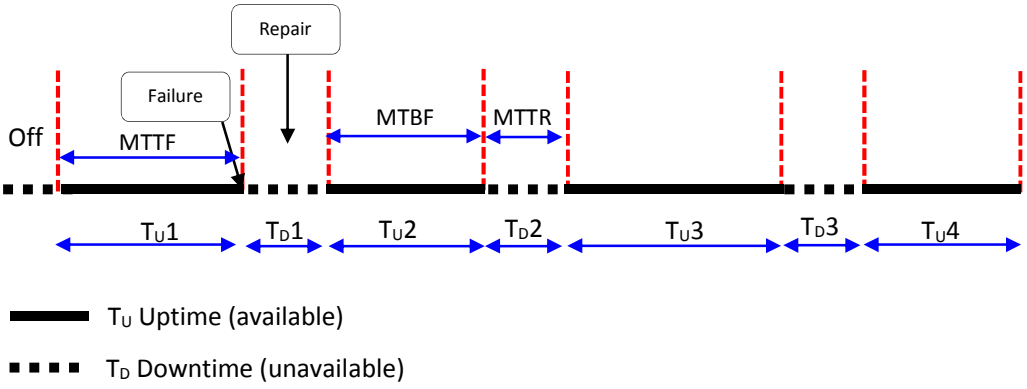

Figure 3.3: Operating times of a Component in a System.

The metric for reliability [96,179 and 180] is trifold *Mean Time between Failures (MTBF), Mean Time to Failure (MTTF)* and *Failure Rate.*

1. *Mean Time between Failures (MTBF):* This measurement is the average time period (usually in hours) between two successive component failures. This metric is used for systems that can be repaired in finite time. Larger the

MTBF, greater the *Availability*. Figure 3.3 shows the *MTBF* of a component in an Information System. One more metric that's is used in conjugation with *MTBF* is *Mean Time to Repair (MTTR)*, which is the average time period to repair or recover the failed component. Going by the figure 3.3, the metric for *MTBF* and *MTTR* is as follows:

$$\text{MTBF} = \frac{T_u1+T_u2+T_u3+T_u4}{4}$$

$$\text{MTTR} = \frac{T_d1+T_d2+T_d3+T_d4}{4}$$

Therefore in general, the above equations respectively can be written as:

$$\text{MTBF} = \frac{\sum_{\mathbf{i=1}}^{\mathbf{n}} T_u(\mathbf{i})}{\boldsymbol{n}}$$

$$\text{MTTR} = \frac{\sum_{\mathbf{i=1}}^{\mathbf{n}} T_d(\mathbf{i})}{\boldsymbol{n}}$$

There are different variations of MTBF [96]:

(a) *Hardware MTBF*: pertains to mean time between successive hardware component failures.

(b) System MTBF: pertains to mean time between system failures.

(c) Mean Time between Interruptions (MTBI): same as *MTBF* but the only difference is a failure and an interruption. MTBI is just a temporary outage requiring no repairs.

2. *Mean Time To Failure (MTTF):* these metric measures the time of components in a system until the first failure. It is the average time of such components in a system. The difference with *MTBF* lies in the fact that *MTTF* measures reliability of non-repairable systems (systems with infinite repair time).
3. *Failure Rate (FR):* one more metric for measuring reliability is Failure Rate. It is the inverse of MTBF.

$$Failure\ Rate = \frac{1}{MTBF}$$

MTBF can be either Hardware *MTBF* or System *MTBF* and the corresponding failure rates are known as *Hardware Failure Rate* and *System Failure Rate* respectively. Hardware MTBF is the most generally mentioned metric.

3.2.2 Timeliness (Uptime)

Timeliness is the response of an information system to a user request in a suitable amount of time. Delayed response is equivalent to no-response in today's world, given the speed and efficiency at which information processing and communicating systems work these days. Given the criticality of time w.r.t Availability, this metric is the most used and mentioned in determining the Availability of an information system. There are two things to be seen here, one is the individual time of each request/message and the second is the overall time all the requests/messages (includes idle time as well). Generally when it comes to measuring Availability we are interested in the second one i.e. the overall time or better put as the extent (time) to which an information system or resource is processing or working without any interruption or outage (*Uptime*) [100 and 102]. We are also interested in the time when the information system or resource is not processing or working (*Downtime*) i.e. outage, repairing time or the time during up gradation of a system, or any other time when the system is down. A glimpse of *Uptime* and *Downtime* is shown in the figure 3.3. Availability is measured in terms of *Uptime Ratio*, which gives us the nearest approximation of the most commonly quoted availability metric i.e. The *Steady State Availability* [96]. *Uptime Ratio* is the percentage of the system being available without any interruption during the useful life. *Uptime Ratio* is calculated as follows [100] (w.r.t figure 3.3):

$$\textit{Uptime Ratio} = \frac{T_u}{T_u + T_d} \quad \text{or} \quad A = \frac{T_u}{T_u + T_d}$$

Where
T_u: *Uptime,*
T_d : *Downtime,*
A : *Availability.*

Considering figure 3.3 for a repairable system above equation can be written as

$$A = \frac{MTBF}{MTBF+MTTR}$$

Another most commonly used Availability metric related to *downtime* is *downtime per year in minutes* [96], calculated as follows:

$$T_{down} = (\mathbf{1} - \boldsymbol{A}) \times \mathbf{365} \times \mathbf{24} \times \mathbf{60}$$

Based on the results from above equations Availability of an information system is classified as:

Table 3.1: The Availability league (six 9's)

Availability %	Downtime %	Downtime per Year
98%	2%	7.3 days
99%	1%	3.65 days
99.8%	0.2%	17 hours, 30 minutes
99.9%	0.1%	8 hours, 45 minutes
99.99%	0.01%	52.5 minutes
99.999%	0.001%	5.25 minutes
99.9999% (Six 9s)	0.0001%	31.5 seconds

The Information Systems are classified based on the number of 9s given in the table 3.1 [103]. Based on the number of 9s the corresponding downtime per year can be calculated using equation 8 i.e. a company XYZ claims that the services that they provide online have a rating of four 9s, using equation T_{down} the downtime for them per year is therefore 52.6 minutes. Any value of 99% is acceptable in today's world of Information and Communications Technology.

Every component is a system in itself and Availability of an Information System depends upon the availability of the individual components. Information system components can be coupled together serially i.e.

Component 1 $\xrightarrow[\textbf{\textit{depends on}}]{}$ Component 2 $\xrightarrow[\textbf{\textit{depends on}}]{}$ Component 3 and so on.

Availability for such a scenario i.e. serial coupling is [183]:

$$A_{serial} = \prod_{i=1}^{n} A_i$$

Information system components can be coupled parallel as well i.e. component 1 is redundant to component 2 and so on. Availability for such a scenario i.e. parallel coupling is [183]:

$$A_{parallel} = 1 - \prod_{i=1}^{n} (1 - A_i)$$

3.2.3 Accessibility

Accessibility is the extent to which an information system is used concurrently by as many number of users viable without making any changes (like adding new hardware for more users) to the Information System. All the concurrent users should be in active state and should be subscribed and authorized to whatever services the Information System provides. Now to grant access to an Information System we need gate keeping for letting only authorized users to access the resources. Such gatekeeping is provided by means of *Authentication and Authorization.* Authentication has proved to be a well-known critical factor in accessibility [104 and 105] and the domain of accessibility is divided into three levels:

i. *Technical*: at this level there are many control mechanisms to allow access like, *Mandatory Access Control (MAC), Discretionary Access Control (DAC),* and *Role Based Accessed Control (RBAC)* [105 and 106].
ii. *Conceptual*: at this level "who has access to what" is expressed using the *Access Matrix* [107]. Some implementation of Access Matrix used are *Access Control Lists (ACL), Capabilities etc.*
iii. *Organizational:* This is the top most level in the domain of Accessibility. The efforts for secure access consist of Security Policy, use of Biometrics and many more [104].

3.3 Empirical Justification and Analysis

All the metrics mentioned in the preceding sections are used by the security practitioners for the evaluation of Availability of Information Systems. In order to put an Information System into an evaluation process, all the values are measured systematically over some pre-determined course of time or in certain cases until a disruption or a problem occurs. For the empirical justification we won't be following the said approach, because the main motive of this chapter is just to analyse and study how the determinants of Availability are affected by DoS attacks but not to present a measurement or analysis of the determinants of Availability. We will follow a different approach wherein we will empirically justify only the theory but not the mathematical part of the different metrics mentioned i.e. while analysing *Timeliness* we will not measure for how long the system has been up or down (Calculating 'A' or 'T_{down}'), but will only demonstrate how the DoS attack will bring the system (a server) down or bring the system to a level where the system will no longer process legitimate requests of the clients. For the empirical justification we are conducting two small scale experiments under controlled conditions, the experiment-1 will focus on *Reliability* and *Timeliness* and experiment-2 will focus on *Accessibility* of the information system.

3.3.1 Experiment 1 (validating Reliability and Timeliness)

For the illustration of the effect of DoS attack on *Reliability* and *Timeliness* we conduct a small scale experiment using a simple network topology given in figure 3.4. The objective of this experiment it to demonstrate the effect of DoS attack on Windows Server 2012 for the illustration of Reliability and Timeliness. We achieve this by flooding the server using *Sockstress* [110], a TCP based socket stress framework. We do this by exploiting the zero window condition in the TCP handshake process.

3.3.1.1 Experiment Setup

The experiment is carried out under controlled conditions on a local area network consisting of a server and three client computers. The configuration of the machines are presented in the table 3.2. The server is the victim machine (192.168.0.10) which

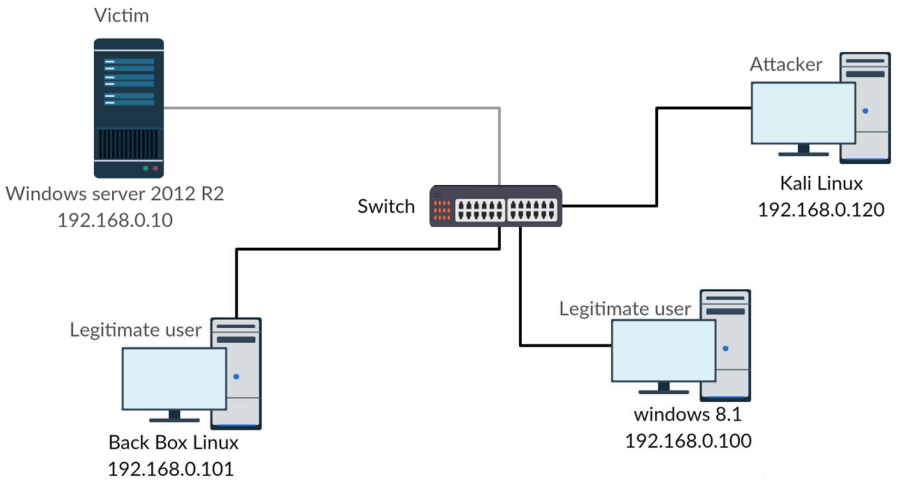

Figure3.4: Experiment 1 setup.

is at the receiving end of the traffic generated by the attacker machine (192.168.0.120). There are two legitimate users as well who want to access the services of the server machine. The server machine is configured as an application server, web server and a FTP server, hosting the EES Web_Controls web application of the EES system (section 4.5, chapter 4).

Table 3.2: System Configurations used in the Experiment.

Machine	Operating System	Hardware Configuration
192.168.0.10 (victim)	Windows server 2012 R2 (6.3 build 9600)	Intel® Core™ 2 Duo 2GHz, 1 GB RAM
192.168.0.100 (user)	Windows 8.1 (6.3 build 9600)	Intel® Core™ i3 2.4 GHz, 2 GB RAM
192.168.0.101 (user)	Back Box 4.1	Intel® Core™ i3 2.4 GHz, 2 GB RAM
192.168.0.120 (Attacker Machine)	Kali Linux 1.1.0	Intel® Core™ i5 2.8 GHz, 4 GB RAM

Besides HTTP, the server is running a number of well-known services like i.e. HTTPS, DHCP, FTP, SMTP, Telnet, DNS, NETBIOS, POP3, and MSRPC etc. The server is running on *VMware Player V7* [128], hosted on windows 8.1 machine (6.3 build 9600) with Intel® Core™ i5 2.8 GHz, 4 GB RAM.

The attacker launches the DoS attack from his machine running Kali Linux using a socket stress testing framework Sockstress. Sockstress exploits the TCP handshake process by creating a huge number of legitimate zero window TCP connections. Sockstress creates a zero window when the client sends back the last ACK for the completion of the connection request. The TCP handshake process by Sockstress is shown in the figure 3.5. Now this situation keeps the windows server in a waiting state. The windows server will probe the client until the size of the window is greater than zero. Now during this waiting state the server receives thousands of requests from the system running Sockstress, which results in performance degradation of the windows server and this process ultimately brings the server down to a level where it no longer entertains new TCP connections from requesting clients and therefore results in a DoS attack. The service requested by the client running

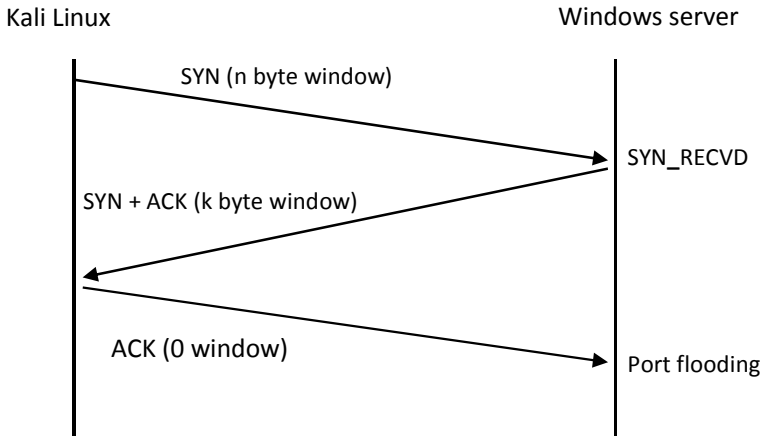

Figure 3.5: TCP Handshake with zero window size.

Back Box is a simple control message calculating the delay from the client to the windows server i.e. an ICMP request. The results of the ICMP request are recorded before and during the DoS attack. We will be analysing the ICMP data from the Back Box client and the RAM depletion data from the windows server.

3.3.1.2 Results and Discussion

The attacker machine first scans for all the open TCP ports on the windows server. This is accomplished using a tool Nmap [129]. The tool returns the following list of open TCP ports on the target machine:

Table 3.3: Intense Scan using Nmap on Windows Server.

PORT	SERVICE	STATE
7 TCP	Echo Protocol	Open
9 TCP	Discard Protocol	Open
13 TCP	Daytime Protocol	Open
17 TCP	Quote of the Day (QOTD)	Open
19 TCP	CHARGEN	Open
20 TCP	FTP data transfer	Open
21 TCP	FTP control	Open
23 TCP	Telnet	Open
25 TCP	SMTP	Open
53 TCP	DNS	Open
80 TCP	HTTP	Open
135 TCP	MSRPC	Open
137 TCP	NetBIOS name service	Open
139 TCP	NetBIOS session service	Open
445 TCP	MS-DS Active Directory	Open
47001 TCP	Unknown	Open
49152 TCP	Unknown	Open
49153 TCP	Unknown	Open
49154 TCP	Unknown	Open
49155 TCP	Unknown	Open
49156 TCP	Unknown	Open
49157 TCP	Unknown	Open
49158 TCP	Unknown	Open

After configuring Sockstress on Kali Linux for attacking the windows server, we use the following command to bring havoc on the target machine:

```
./Sockstress -A -c -1 -d 192.168.0.10 -m -1 -Ms -p 7,9,13,17,19,20,21,23,25,53,80,135,137,139,445,47001,49152,49153,49154, 49155 ,49156 ,49157 ,49158 -r 100000 -s 192.168.0.150/175 -vv
```

As soon as the DoS attack starts, we start capturing data at two places:

1. ICMP response time on 192.168.0.101 (Back Box).
2. RAM decay on 192.168.0.10 (Windows server).

The attack and the corresponding data capturing continues until the server freezes or is brought down to a state where it no longer responds to new incoming TCP connections. Now before going into the aftermath of the attack, let's first mention the state of ICMP response time and RAM in the respective machines before the attack. Figure 3.6 shows the state of memory of the server before the attack and figure 3.7 shows the response time of an ICMP message from the Back Box machine before the attack.

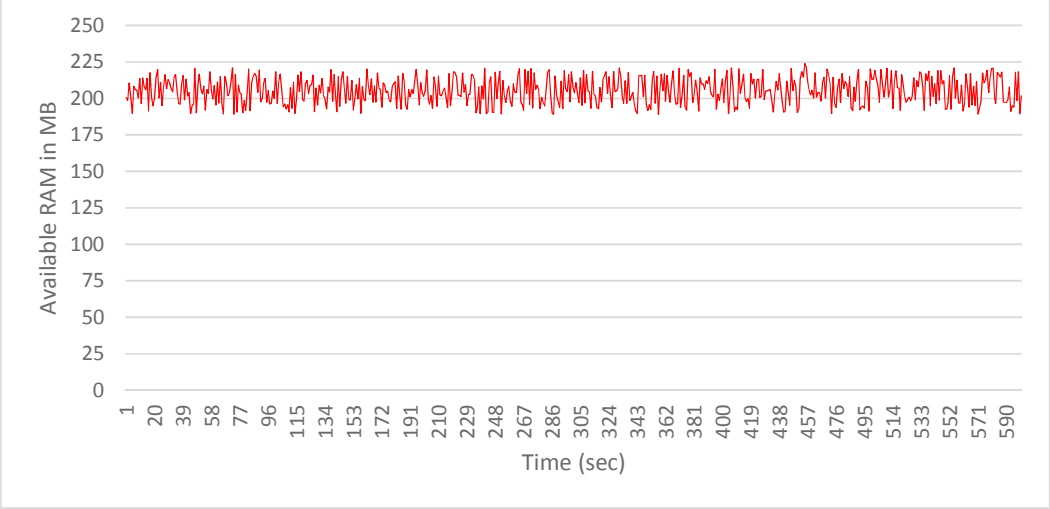

Figure 3.6: Available memory before the attack

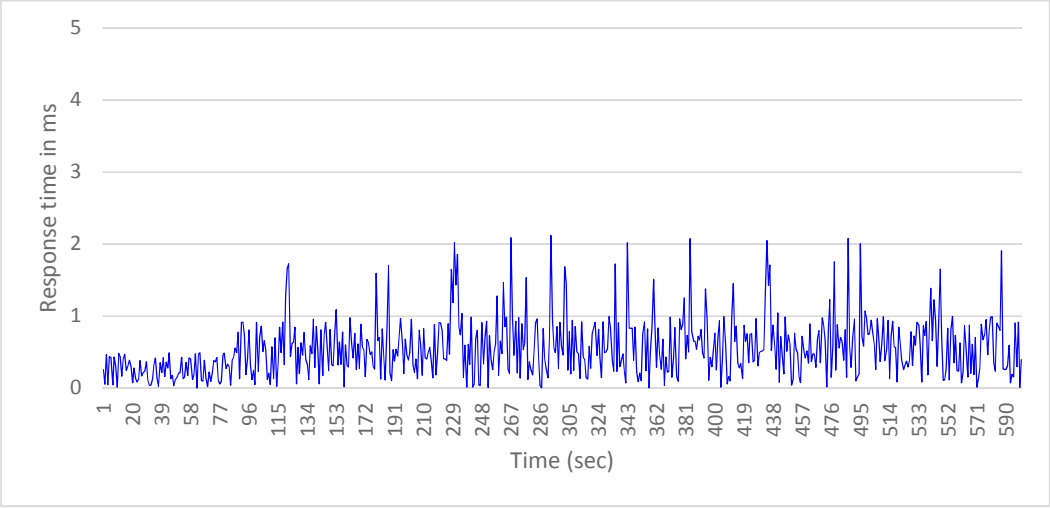

Figure 3.7: ICMP response time before the attack

Now before going further let's not forget to mention the universally accepted standard for Quality of Service (QoS) requirements defined for ICMP protocol and other applications [130]. Table 3.4 shows the universally accepted requirements for various applications.

Table 3.4: QoS requirements of various applications.

Application Category	Request/Response delay
E-mail (user-server)	any < 4 s or maxRTT < 4 s
Telnet	any < 250 ms or maxRTT < 250 ms
DNS	maxRTT < 4 s
ICMP	maxRTT < 4 s
File-sharing, FTP	any < 10 s or maxRTT < 10 s
e-commerce, web	any < 4 s or maxRTT < 4 s

Going by the table 3.4 and the figure 3.7, our measurement of ICMP response time before the attack is well within the boundaries of the universally accepted value of ICMP under normal communication conditions. The accepted value for ICMP maximum *Round Trip Time* (request-response delay) is less than 4 seconds. The available memory before the attack is hovering around the value of 215 Mega Bytes.

As soon as the attack starts, the memory size of the windows server starts to decay and moves towards zero. On the other hand the ICMP response time of the Back Box machine starts to increase and with the passage of time both the values continue in their respective directions. After some time the server ceases down to a state where the memory is as low as 3MB and the ICMP response time is infinite (server not reachable). The table 3.5 and 3.6 represent the recorded values of Available RAM and ICMP response times. The experiment was repeated 5 times for the trueness of the results and in both the cases we are considering only the mean values. The Standard deviation in the case of Available RAM is also calculated and expectedly the dispersion of the values is closer to the mean.

Figure 3.8 reflects the true nature of the attack. The left axis displays the state of the memory during the attack. The axis on the right displays the status of the ICMP requests from the back box machine. The bottom axis represents time. From the start

of the attack the status of the server was captured every second. The memory of the server shows a linear trend in the fall from the start position until when the size of the available RAM touches the 20 MB mark. From here onwards the server starts to behave abnormally and firstly for some time the size of the memory fluctuates between 20 MB and 50 MB and then it fluctuates in the zone between 0 MB and 20 MB. This is the state where the server is most unreliable and unpredictable. On the other side the ICMP response time shows a linear progressive increase and goes well beyond the permissible limits mentioned in the table 3.4.

Table 3.5: Available Memory in Mega Bytes of the server over time[1]

Time (Seconds)	Run 1	Run 2	Run 3	Run 4	Run 5	μ Available RAM	σ Available RAM
1	201	209	218	206	212	209	6
30	124	121	130	119	126	124	4
60	106	101	107	104	100	104	3
90	101	92	86	90	89	92	5
120	74	76	79	82	80	78	3
150	69	65	68	71	69	68	2
180	61	59	57	64	63	61	3
210	58	55	53	51	58	55	3
240	54	50	42	47	46	48	4
270	48	39	30	32	43	38	7
300	43	30	25	30	37	33	6
330	39	27	22	28	33	30	6
360	29	24	21	23	29	25	3
390	24	18	28	34	24	26	5
420	19	13	24	32	21	22	6
450	16	10	21	23	27	19	6
480	23	23	19	11	22	20	5
510	20	16	15	7	19	15	5
540	19	13	8	15	15	14	4
570	9	8	16	8	5	9	4
600	27	26	22	6	13	19	8
630	15	19	18	15	11	16	3
660	8	12	6	10	7	9	2

[1] For representation of data in the recorded samples after every 30 seconds are in table 3.5 and table 3.6. The full set of values are reflected in the graphs in Figures 3.6, 3.8 and 3.9.

During the Unreliable period (0-20 MB) the progression is steep and response time goes to infinity (unreachable), as clearly evident from the plot. During the five runs of the same experiment the lowest recorded available memory size was 3 MB. During the 5 runs the server froze twice and needed hard reset both the times. Rest during the unreliable state the size of the memory hovered between 0 – 20 MB.

Table 3.6: ICMP Response time in ms[2]

Time (Seconds)	Run 1	Run 2	Run 3	Run 4	Run 5	μ Response Time
1	0.340	1.000	1.130	0.952	1.000	0.884
30	0.230	1.100	1.120	0.952	1.000	0.880
60	0.960	1.12	1.120	1.170	1.290	1.132
90	0.907	1.06	5.15	57.50	1.32	13.187
120	0.941	1.18	4.24	11.80	15.60	6.752
150	0.949	1.06	2.70	97.30	1.25	20.652
180	0.928	1.07	1.13	26.7	12.7	8.506
210	0.940	10.30	185.00	158.00	11.00	73
240	19	11.70	11.20	11.90	87.00	28
270	41.40	1142.00	113.40	93.30	119.80	302
300	1837	1040	187	1421	176	932
330	2045	3056	193	1115	1950	1672
360	2643	2356	1136	1314	1942	1878
390	Unreachable	2757	1079	1652	1593	2416
420	2141	Unreachable	Unreachable	Unreachable	Unreachable	4428
450	1038	Unreachable	Unreachable	1867	Unreachable	3581
480	1903	Unreachable	2177	2050	Unreachable	3226
510	Unreachable	2041	2992	Unreachable	Unreachable	4007
540	Unreachable	1032	2867	Unreachable	3182	3416
570	2897	1932	2867	2078	2182	2391
600	2933	Unreachable	Unreachable	Unreachable	Unreachable	Unreachable
630	3216	Unreachable	Unreachable	Unreachable	Unreachable	Unreachable
660	Unreachable	Unreachable	Unreachable	Unreachable	Unreachable	Unreachable

[2] For representation of data in the recorded samples after every 30 seconds are in table 3.6. The full set of values are reflected in the graphs in Figures 3.7, 3.8 and 3.10.

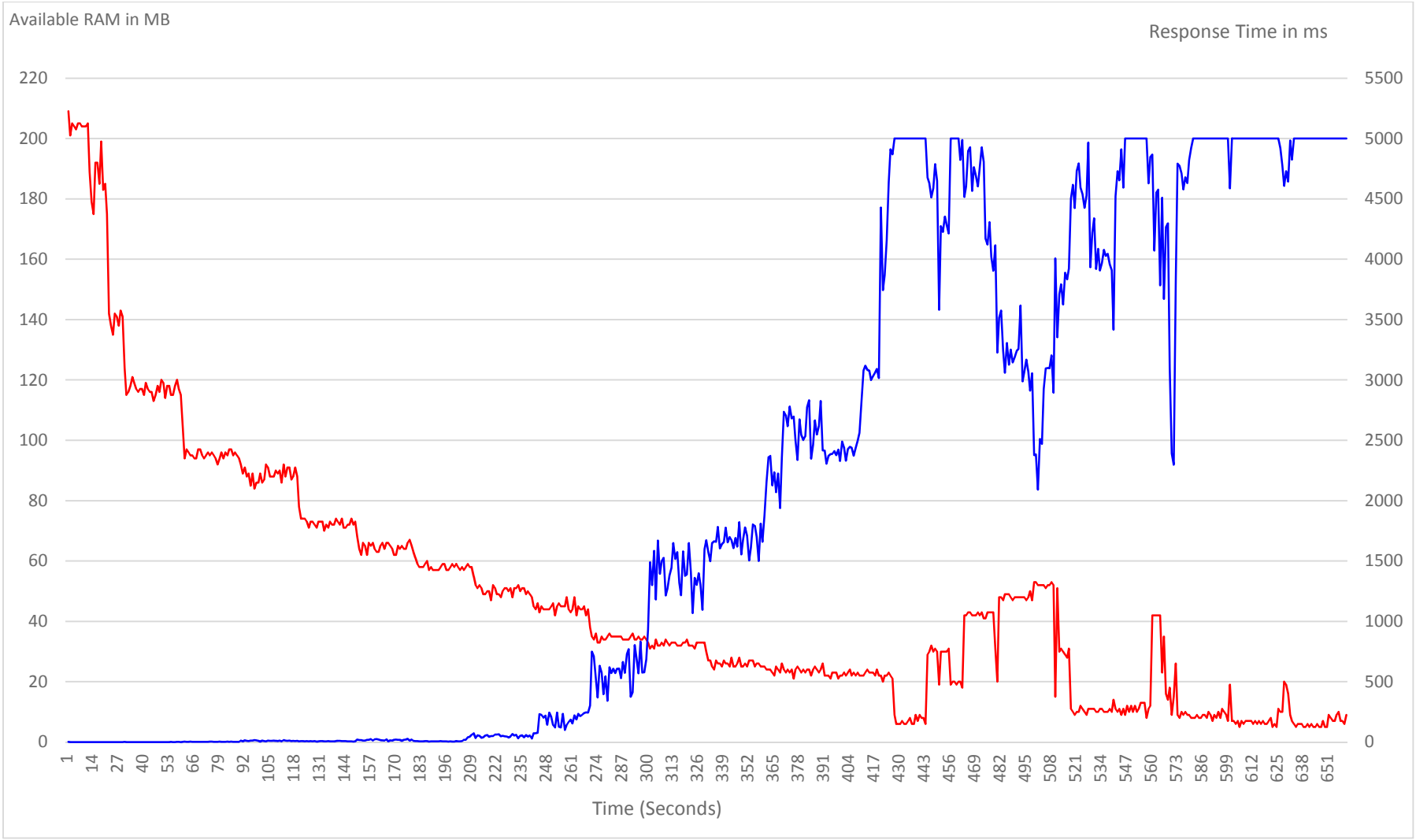

Figure 3.8: Memory Decay and ICMP response time[1], during the DoS attack.

[1] 5000 = Unreachable, for the representation of "Unreachable" value mentioned in the table 3.6

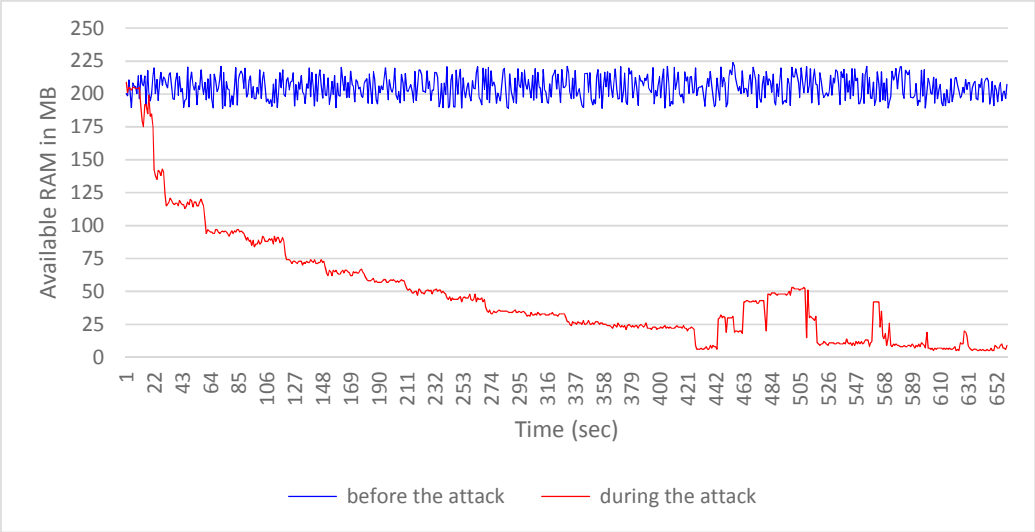

Figure 3.9: Memory status before and during the attack.

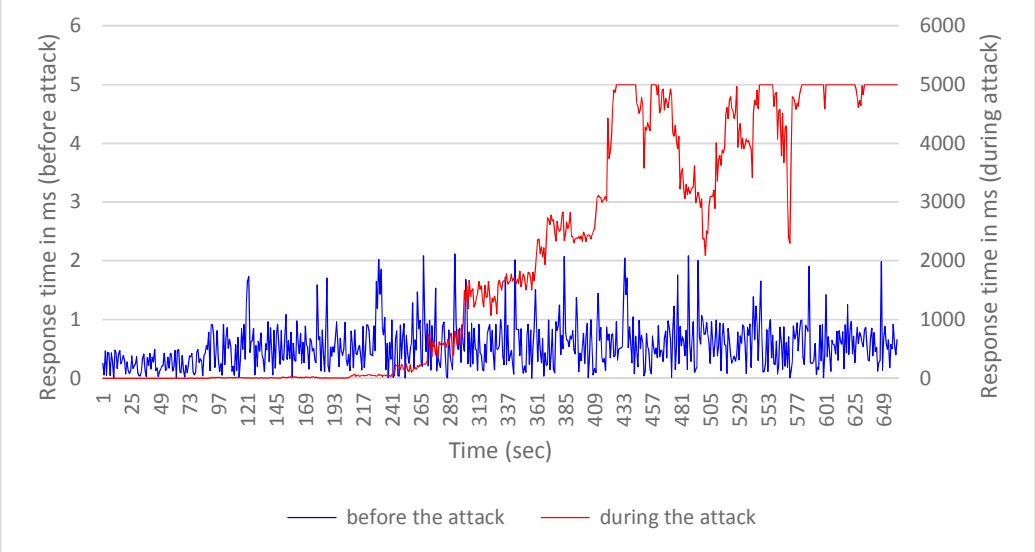

Figure 3.10: ICMP response time before and during the attack.

3.3.1.3 Experiment Conclusion

Comparing the QoS standards from table 3.4 and the data in tables 3.5 and 3.6 we come to the conclusion that how badly the said DoS attack can damage the server, as the ICMP request/response values go well beyond the permissible values mentioned in the table 3.4. Recall from section 3.2, the metrics for Reliability and Timeliness. Reliability got affected here because the system froze twice (hence downtime) and the effect on timeliness is evident from the linear increase in the ICMP values from 0 to infinity (server being unreachable) and timeliness also can be evaluated by considering the downtime, the server in this case was down twice.

Now in the normal metric evaluation the *downtime* and the *uptime* are measured in a systematic and timely manner over some period of time to measure the effects on Availability. Since our aim was to prove the effect rather than measure the effect, that's why just bringing the system down once or twice and the demonstration of the increase in ICMP values suffices our need and proves the objective using the above mentioned tabular data and facts. A comparison of memory state before and during the course of the attack is given in the figure 3.9. Similarly a comparison of the ICMP response time before the attack and during the attack is also given in the figure 3.10.

3.3.2 Experiment 2 (validating Accessibility)

For the explanation of the effect of DoS attack on *Accessibility* we conduct a small scale experiment using a simple network topology given in figure 3.11. We demonstrate this by stress testing the windows server using *Siege* 2.70 [131], a HTTP/HTTPS based stress testing framework. The objective is to demonstrate how much data or requests the target system can handle concurrently and at the same time give an indication about the systems *Availability* (Accessibility). Overloading the system with requests and data generated through siege may also result in a DoS attack, but primarily here we are more interested in the number of connections that the server can handle concurrently. The tool allows us to strike the server with pre-configured number of concurrent simulated users. The tools give us various performance measures which will be discussed further in the experiment.

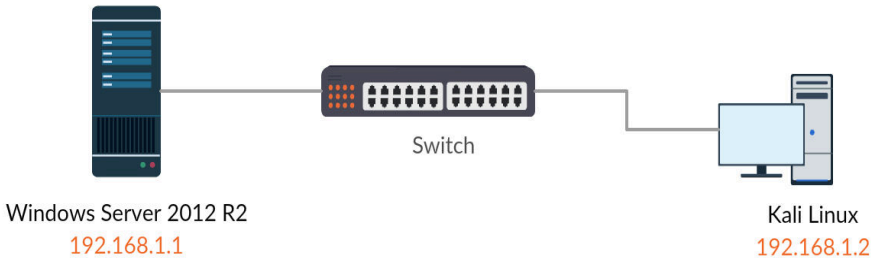

Figure 3.11: Experiment 2 setup.

3.3.2.1 Experiment Setup

The experiment is carried out under controlled conditions on a Local Area Network consisting of a server and a client computer. The configuration of the machines are presented in the table 3.7. The server (192.168.1.1) which is at the receiving end of the traffic generated by the attacker machine (192.168.1.2). The server machine is configured as an application server and web server, hosting the EES Web_Controls

web application of the EES system (section 4.5, chapter 4). Besides HTTP, the server is running a number of well-known services like i.e. HTTPS, DHCP, FTP, SMTP, Telnet, DNS, NETBIOS, POP3, and MSRPC etc. The windows server is running on *VMware Player V7* [128], hosted on windows 8.1 machine (6.3 build 9600) with Intel® Core™ i5 2.8 GHz, 4 GB RAM, with Intel® Core™ i5 2.8 GHz, 4 GB RAM.

Table 3.7: System Configurations used in Experiment.

Machine	Operating System	Hardware Configuration
192.168.1.1 (victim)	Windows server 2012 R2 (6.3 build 9600)	Intel® Core™ 2 Duo 2GHz, 1 GB RAM
192.168.1.2 (Attacker Machine)	Kali Linux 1.1.0	Intel® Core™ i5 2.8 GHz, 4 GB RAM

The Siege load testing framework is launched from the machine running Kali Linux (192.168.1.2). Siege has three modes of operation, *internet simulation*, *regression testing* and *brute force*. We will be using brute force for validating the accessibility component of *Availability*. The tools tests the server and benchmarks the server for various performance measurements carried out during the load testing.

3.3.2.2 Results and Discussion

After configuring Siege on Kali Linux for load testing we use the following configuration of the tool to test the strength of the target machine:

```
siege -c500 -t10 192.168.1.1
** SIEGE 2.70
** Preparing 500 concurrent users for battle.
The server is now under siege...      done.
```

We created 5 instances of the above configuration and each configuration prepares 500 concurrent simulated users to test the strength of the server for 10 seconds. That means when all the five configurations run concurrently, we are actually striking with a force of 2500 concurrent users. After the wait of 10 seconds (in each configuration)

the server prints the measurements shown in the table 3.8. The experiment was repeated 3 times with same configurations under same conditions. The measurements returned are Transactions, Availability, Elapsed time, Data Transferred, Response time, Transaction rate, Concurrency and Failed Transactions, out of which we are only interested in 3 measurements, Availability, Response time and Concurrency. Availability here is different than *Availability* [123] we have been discussing so far, here in *Siege* testing framework it is the percentage of successfully handled socket connections by the server. It is the result of socket failures (including timeouts) divided by the sum of all connection attempts. Response time is the average processing time it took to process each simulated user's requests. Concurrency is the average number connections from each simulated user. The experiment was run with same configurations in all the instances across all the 3 runs.

Table 3.8 displays the data collected after putting the windows server under siege. In the first instance of the first run, we have 277 successful transactions done with the server by 500 concurrent simulated users. The availability value measured in this configuration is 18%, the average response time of every connection is 6.26 seconds and the number of concurrent connections for the same is 208. Important thing to observe here is the response time, which is well above the permitted Round Trip Time (RTT) [130] is case of a HTTP web request (see table 3.4). The availability measured in the second instance is higher at 33%, the average response time of every connection is 5.07 seconds and the number of concurrent connections for the same is 246. Response time is still above the permissible limit in case of HTTP web request. If we go on and analyse all the values of these three parameters (Availability, Response time and Concurrency) in the first run, we find an interesting trend, decrease in response time leads to increase in availability and concurrency as well. Two observation where the response time was under the permissible limit was in instance 4 and instance 5 and in both the cases the concurrency was highest among the other entries in the group. From the first run we deduce that high availability percentage and a lower value of response time produces a higher number of concurrent connections for the server and low availability percentage and a higher

value of response time produces a lower number of concurrent connections for the server.

Table 3.8: Data assemblage from Siege http load test

Run1					
	instance1	instance2	instance3	instance4	instance5
Transactions:	277	592	708	1710	2477
Availability: %	18.19	33.18	39.25	61.29	68.39
Elapsed time: secs	5.67	12.19	11.79	20.05	20.9
Data transferred: MB	0.57	0.77	0.81	1.48	2.01
Response time: secs	6.27	5.07	4.71	3.61	2.96
Transaction rate: /sec	48.85	48.56	60.05	85.29	118.52
Concurrency	208.71	246.18	283	307.71	351.24
Failed transactions	1246	1192	1096	1080	1145
Run2					
Transactions:	2637	801	1466	3691	7074
Availability: %	67.46	38.7	54.34	75.62	85.5
Elapsed time: secs	23.99	13.75	17.85	16.93	23.3
Data transferred: MB	2.16	0.93	1.36	2.84	5.1
Response time: secs	3.14	5.64	4.45	1.3	0.69
Transaction rate: /sec	109.92	58.25	82.13	218.02	303.61
Concurrency	344.96	328.4	365.57	382.99	410.3
Failed transactions	1272	1269	1232	1190	1200
Run3					
Transactions:	138398	142039	1132	1922	86627
Availability: %	100	100	51.71	64.05	98.74
Elapsed time: secs	599.78	599.81	10.84	16.71	221.25
Data transferred: MB	92.52	94.96	1.09	1.62	58.25
Response time: secs	1.66	1.61	2.96	3	0.76
Transaction rate: /sec	230.75	236.81	104.43	115.02	391.53
Concurrency	383.61	381.27	309.37	345.27	396.21
Failed transactions	3	0	1057	1079	1102

The graphical analysis of the above mentioned facts is done is figure 3.12 and 3.13. In the second run of the experiment a similar trend is observed in the three parameters i.e. when the availability if high, the response time is low and low response time also

means higher rates in concurrency and the vice versa as well. The response times in instance 2 and 3 are above the permissible limits and in both the cases the availability

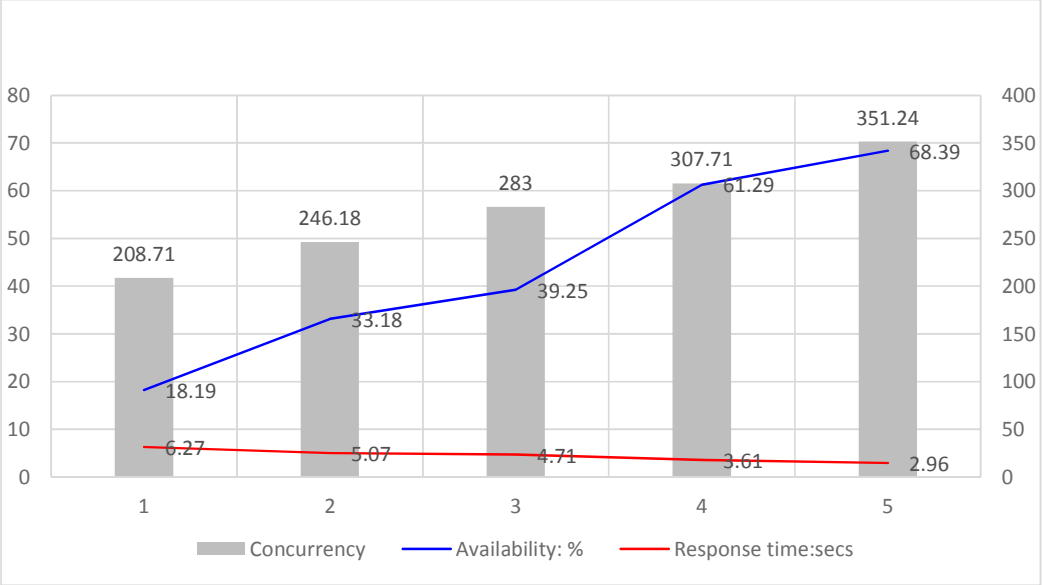

Figure 3.12: Run 1, measurements of Concurrency, Availability and Response time.

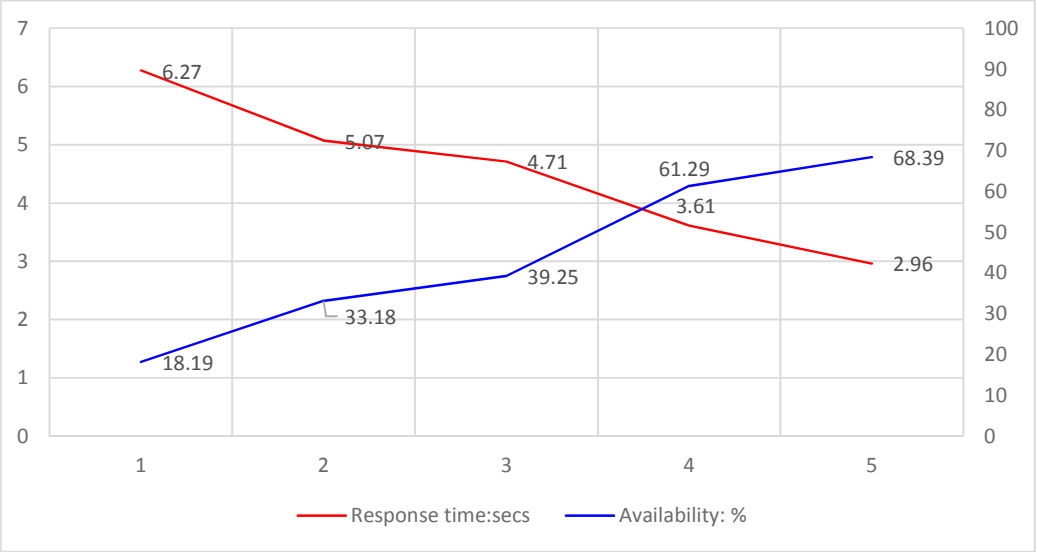

Figure 3.13: Run 1, comparison of Availability and Response time.

and concurrency is at lower ends in the group. The highest rate of concurrency is achieved in instance 5 and the lowest response time is also from the same instance. The graphical analysis of these facts is done is figure 3.14 and 3.15.

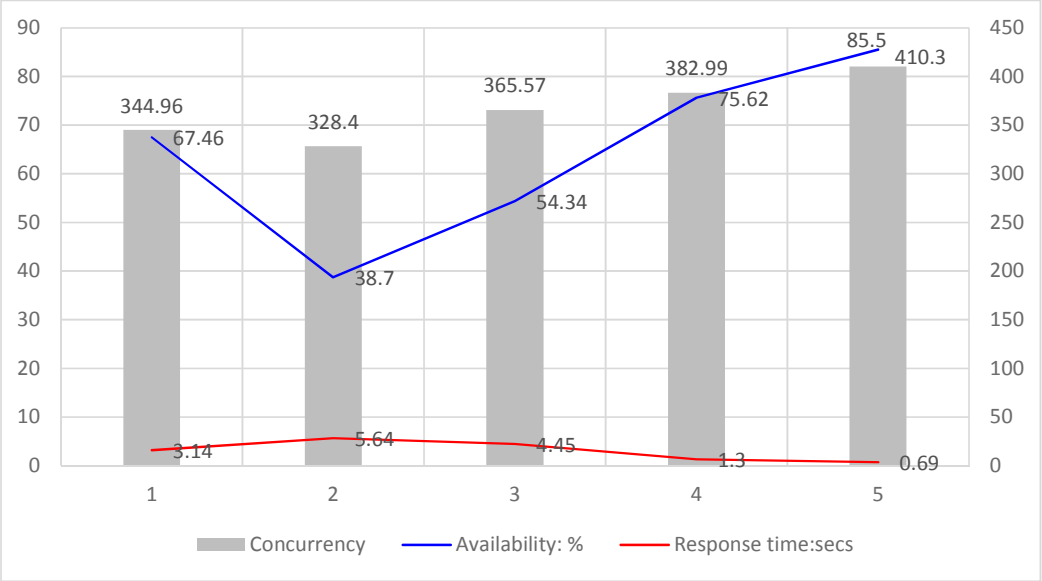

Figure 3.14: Run 2, measurements of Concurrency, Availability and Response time.

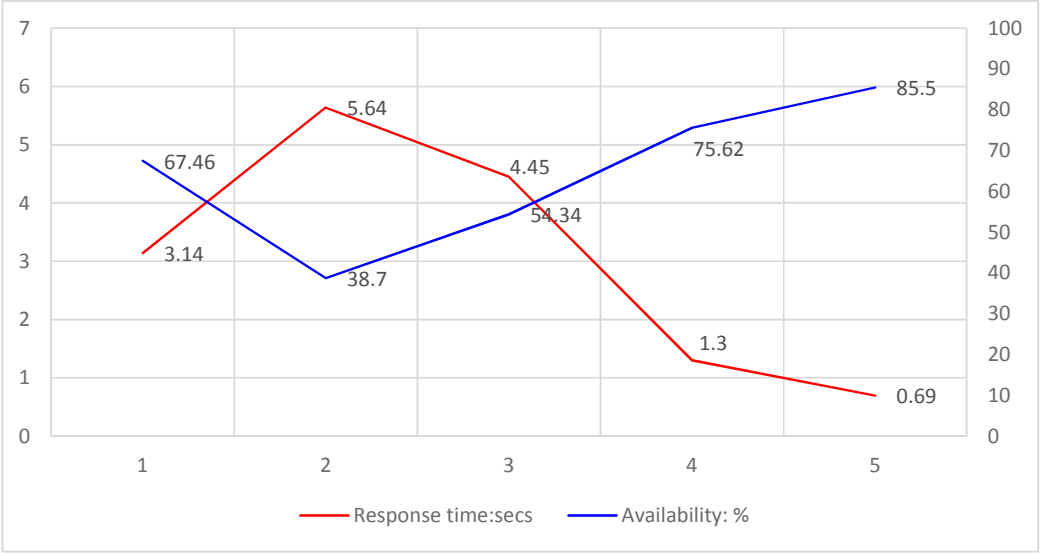

Figure 3.15: Run 2, comparison of Availability and Response time.

The third run of the experiment showed a similar trend as the preceding experiments, Availability and Concurrency showed maximum growth when the response time was lowest.

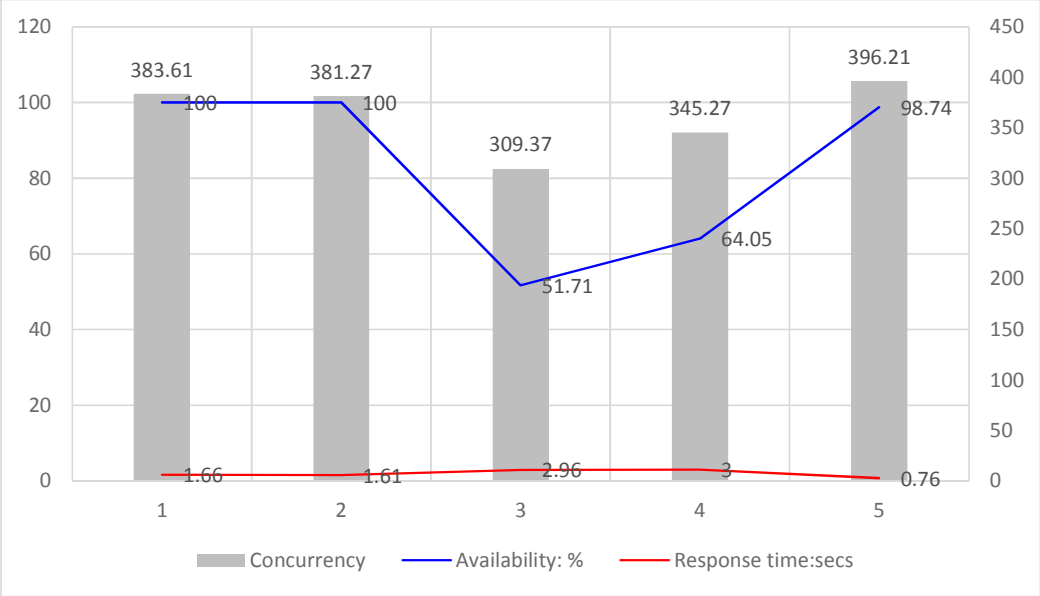

Figure 3.16: Run 3, measurements of Concurrency, Availability and Response time.

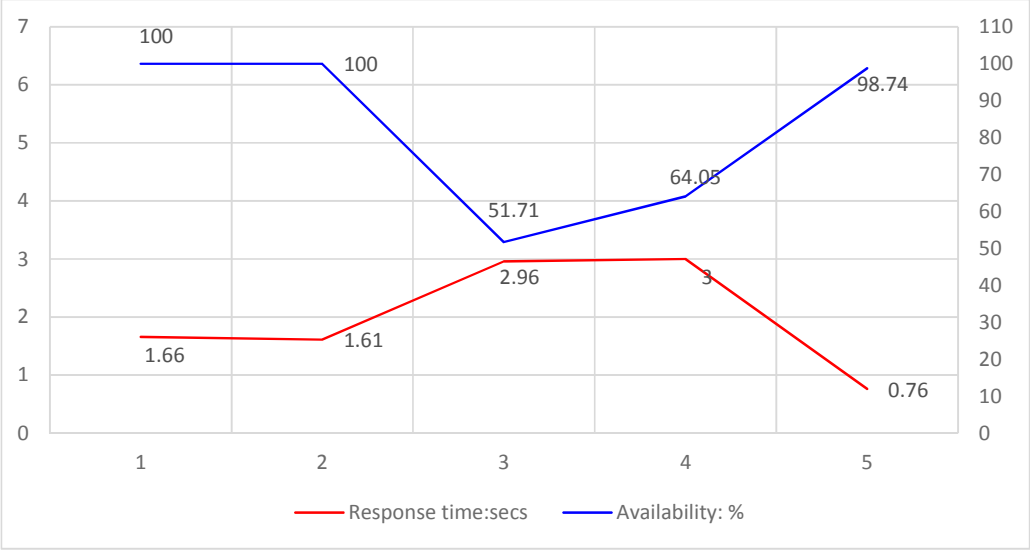

Figure 3.17: Run 3, comparison of Availability and Response time.

The first two instances achieved 100% availability and in both the cases the response time was well within the limits of universally accepted values. The concurrency was highest in the fifth instance and no surprises for response time being the lowest here among the group. The graphical analysis of these facts is done is figure 3.16 and 3.17. Also this run produced the highest number of successful transactions and the lowest number of failed transactions.

3.3.2.3 Experiment Conclusion

With respect to using the siege framework for evaluating the *Accessibility* of an information system i.e. the number of concurrent connections that a server supports, we conclude with the fact that there exists a relation between *Concurrency, Response Time and Availability*. Higher number of concurrent connections are possible only when the response time of every user request is low, preferably below the universally accepted mark (refer to table 3.4 for the universal standard operation requirement). The vice versa is true as well, when the response time is high, the concurrency is low. Now under normal conditions in the system the response time will mostly be under permissible limits, which therefore won't affect the number of concurrent connections that a server can support. But going by the results of experiment 1 in table 3.6, a DoS attack can severely impact the response time (ICMP response time or RTT) and in the table we have seen how the response time jumped beyond the permissible limits once the attack was launched. It even reached to infinite (server unreachable). Now once the response time starts increasing, the availability and concurrency start decreasing. In other words the increase in response time leads to decrease in the number of concurrent connections that a server can support. In worst cases very high response time will lead to no concurrent connections or no connections at all, leading to what we call as a Denial of Service Attack and therefore in the process affecting *Accessibility*.