Application of String Matching in Internet Security and Reliability

Ali Peiravi

Ferdowsi University of Mashhad
Department of Electrical Engineering, School of Engineering, Mashhad IRAN
Telephone number: (0098) 511-881-5100
Ali_peiravi@yahoo.com

Abstract: In this study the role of string matching algorithms in hardware/software applications in virus scanners or intrusion detection systems in improving data security over the internet is stressed. The author's contribution to an architectural design of a new string matching algorithm implemented on an FPGA for improving hardware based data security over the internet with improved performance over previously published results is reported. This can be used in applications such as hardware virus scanners or intrusion detection systems to further improve internet reliability in case of cyber attacks. After indicating the basic measures of intrinsic internet reliability such as MTTF, MTTR and availability, a new measure is introduced to account for the availability of the internet including the effect of cyber attacks. The new index that is introduced is a measure of downtime due to a cyber attack and is called mean down time due to cyber attack (*MCADT*). Results of empirical measurement of the intrinsic availability of a sample of 159 internet hosts in Iran using a developed software tool are presented and compared with those of a similar study over international hosts. [Journal of American Science 2010;6(1):25-33]. (ISSN: 1545-1003).

Key words: Internet reliability, availability, string matching, data security, cyber attack

I. INTRODUCTION

The Internet is completely unreliable. How can we deal with that? With the rapid growth of the world wide web and increased reliance on the web for almost every aspect of man's life today, Internet reliability is perhaps the most important challenge that researchers and practitioners face today.

The reliability issues of the world wide web stem from various underlying factors. The first and most basic one is the intrinsic reliability that depends on the hardware topology of the network, the various computer and communications systems and devices which make the physical connection possible, and the protocols and operating system software that make it operational. The next element of reliability in the internet is the maintainability of the various subsystems involved as maintainability is a key ingredient of reliability.

The real growth of the internet lies in bandwidth-intensive web content, rich media, and web and IP-based applications. There are many challenges facing internet reliability as businesses move more of their critical functions on-line, and as consumer entertainment shifts to the internet from other broadcast media. Leighton (2009) considered the most serious reliability challenge as the ownership of the heterogeneous internet infrastructure by many competing entities with little incentive to expand capacity.

However, we feel that the most important issue in the reliability of the internet stems from the fact that there are as many potential points of attack in it as there are computers connected to it, making it the most vulnerable system man has ever put to use in such a large scale.

The fact that more and more transactions are being done via the internet has resulted in a wide spread effort by cyber criminals to attempt to earn illegitimate income from the internet. Intrusion of privacy and accessing people's private information has also been attempted with various motives. There have also been attempts at other crimes that are beyond the scope of this paper including espionage, cyber attacks and even cyber war.

There are various approaches to deal with internet reliability problems. One is the use of hardware redundancy. We may consider having alternate ways to get online for the users, or having several mirror sites for the servers to allow more available access of their system to customers as examples of hardware redundancy. Another means of improved hardware reliability is the use of hardware for backing up important information. We may even rely on public services for backing up our important work.

A third important point is having a good connection. This concept refers to what is commonly called carrier hotels, internet peering points or co-location buildings. Such co-location buildings usually sit at major points of internet connectivity,

have redundant power supply connections from the electricity grid plus backup power in case of blackout. Such locations offer high availability in the order of 99.999 percent that is equivalent to 5 minutes of downtime per year [Strom (2008)].

To improve internet intrinsic reliability, three distinct areas require attention:

1- The terminals and user equipment should be more reliable. The personal computers, set-top boxes, cable modems, and routers should have backup batteries to provide service, and the operating system and applications software should be free of bugs.

2- Network infrastructure should be built with a high-availability objective. Individual switches, cross-connects, multiplexers, transmitters, and all associated software should have built-in redundancy such that the network can allow instant rerouting in case of any equipment or cable failure.

3- The protocols on which the networks and services operate should provide perfect access to all features and capabilities of the web.

II. BASIC MEASURES OF RELIABILITY

Reliability is usually defined as the probability of successful operation of a mission under predefined operating conditions and for a specified mission time. There are many different measures used to measure reliability as presented below.

II.1. *Failure Rate* and MTTF

The most basic measure of reliability is the failure rate that indicates the average number of failures per unit time as follows:

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} P[System\ Down\ \ in\ (t, t+\Delta t) \mid System\ Up\ at\ \ t)] \quad (1)$$

In cases where the failure rate is constant

$$\lambda(t) = \lambda \quad (2)$$

Reliability is found from the failure rate function as follows:

$$R(t) = e^{-\int \lambda(t)dt} \quad (3)$$

For constant failure rate, the reliability is

$$R(t) = e^{-\int \lambda dt} = e^{-\lambda t} \quad (4)$$

The next measure of reliability is the mean time to failure, or the expectation of the stochastic variable T_U that defines the uptime of the system.

$$MTTF = E[T_U] = \int_0^\infty t f_U(t)dt = \int_0^\infty R(t)dt \quad (5)$$

For a system with exponential probability density function for T_U we have:

$$MTTF = E[T_U] = \int_0^\infty t f_U(t)dt = \int_0^\infty e^{-\lambda t}dt = \frac{1}{\lambda} \quad (6)$$

II.2. *Mean Time to Repair (MTTR)*

The next important measure affecting a system's reliability is its maintainability indicated by mean time to repair as follows:

$$MTTR = E[T_D] = \int_0^\infty t f_D(t)dt \quad (7)$$

II.3. *Mean time between failures (MTBF)*

Another reliability index used in repairable systems is the mean time between failures as:

$$MTBF = MTTF + MTTR \quad (8)$$

This index shows the average time between successive failures or repairs. Table 1 indicates typical MTBF values for computers and related equipment usually used in the internet.

TABLE 1

MTBF FOR COMMERCIAL COMPUTER EQUIPMENT

Equipment	**MTBF (Hours)**
Personal Computer	5000-50000
Monochrome Display	20000-30000
Color Display	5000-30000
Hard Drive	30000-90000
Floppy Drive	20000-40000
Tape Drive	7500-12000
Compact Disk Drive	30000-60000
DVD Drive	75000-125000
Keyboard	30000-60000
Dot Matrix Printer	2000-4000
Plotter	30000-40000
Modem	20000-30000
Router	50000-500000
Power Supply	20000-40000

II.4. *Availability*

Another measure of reliability for repairable

systems is availability which takes into account both MTTF and MTTR as follows:

$$Availability = \frac{MTBF}{MTBF + MDT} \quad (9)$$

And

$$MDT = MTTR + MCADT \quad (10)$$

where MCADT is a new index proposed in this study to denote the mean downtime due to cyber attacks such as denial of service or distributed denial of service, or any other attack which hinders the normal operation of the system.

II.5. *Intrinsic Availability of the Internet*

If we consider ideal conditions and no presence of cyber abnormal activity such as hacking or denial of service attacks, then

$$MCADT = 0 \quad (11)$$

Then the intrinsic system availability that is the highest possible level of system reliability is:

$$Ao = \frac{MTTF}{MTTF + MTTR} \quad (12)$$

Therefore, the factors that can affect the availability of the internet in these conditions are just related to technical and maintenance problems. However, any attacks on the internet can drastically affect its performance and such attacks should be considered in internet reliability by inclusion in MCADT.

III. RELIABILITY OF THE INTERNET

We can obtain a model for the reliability of the internet if we assume the following simple model in which the user's facilities, the web server with which he is communicating, and the rest of the network which is involved in this connection are modeled as three separate entities as shown in the reliability block diagram of Figure 1.

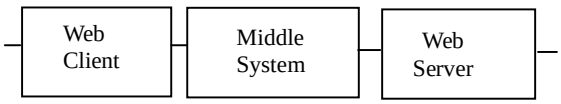

Fig 1- The simplified reliability block diagram of the internet

Notice that the internet relies on packet switching to deliver each packet from one point to another point through various nodes and branches. The availability of the middle system for each packet transmitted would be the product of the availabilities of the facilities in the middle that are involved in the transmission of that specific packet. The use of redundancy in the internet and the packet switching protocol helps the packet travel through some available path. This helps increase the availability of the middle system. However, this is not the only factor involved in the overall availability of internet service for a given user.

If we assume the availabilities of the user's facilities or the web client, middle system facilities and the web server facilities as A_1, A_2, A_3 respectively, then the internet reliability measure or intrinsic internet availability for this user will be

$$A = A_1 A_2 A_3 \quad (13)$$

The typical values of availability are from 0 to 1 with numbers close to 0.99 or better expected. If we assume availabilities of 0.99 for each of these parts, then

$$A = A_1 A_2 A_3 = 0.99^3 = 0.970299$$

However, when there is a cyber attack, the MCADT will become important, and the associated value of availability will be reduced drastically. Whether this happens at the web client, the middle system or the web server, the overall effect is the same as far as the user is concerned. He will experience a very low availability rate that is almost the same as no availability at all. For example, if the web server's availability is low, or the web client's availability is low, we would get a drastic drop in internet availability:

$$A = A_1 A_2 A_3 = 0.99^2 0.1 = 0.09801$$

Another internet reliability issue is related to content reliability and security. It should be noted that the content that we may find available on the internet may not be reliable. A lot of information is posted on the internet. However, this does not guarantee that the content we find is reliable. One should be very careful and check for the accuracy of what he may find on the internet. It may also be not secure and contain viruses, worms, Trojans or other malicious codes embedded in it.

IV. STRING MATCHING AND INTERNET RELIABILITY

IV.1. String Matching Algorithms

String matching is simply defined as how to check whether or not a given string occurs in a given text. This is a common problem in many applications such as text processing, virus detection, molecular biology, intrusion detection, web searching, genetics, etc.

There are many reported string matching algorithms. The problem addressed in string matching is defined as follows. Given a text, T, where T is an array T[1..n] of characters and a pattern P, where P is an array P[1..m] of characters with $m < n$, and typically $m << n$, we wish to know whether P appears as a substring of T, and if so, where.

The most basic algorithm for string matching is the naive algorithm or the brute force method in which there is no need for any preprocessing. In the brute force algorithm there is checking, at all positions in the text between 0 and n-m, whether an occurrence of the pattern starts there or not. Then the pattern is shifted by exactly one position to the right. During the search phase, the text character comparisons can be done in any order.

There are also many other algorithms such as the Morris-Pratt, Knuth-Morris-Pratt, Galil-Seiferas, Boyler-Moore, Turbo-Boyler-Moore, Tunnel-Boyler-Moore, Zhu-Takaoka, Berry-Ravidran, Smith, Raita, Horsepool, etc. to name only a few. The details of these algorithms are presented in many papers and could be pursued by interested readers.

The basic concept behind the naive approach is character by character matching, shifting the whole string down by one character against the text when there is a mismatch and starting again at the beginning. This requires too many repetitions of matching of characters making this algorithm have a time complexity of $O((n-m+1)m)$.

Since the naive algorithm forgets all information about previously matched symbols after a shift of the pattern, it may recompare a text symbol with different pattern symbols again and again leading to a worst case complexity of $O(mn)$ where m denotes the length of the pattern and n denotes the length of the text in which we are searching.

This could be improved upon by noting that when a mismatch is detected somewhere in the text, we have already detected some matched characters. This could be used to decide where to restart matching. This is the basis of the Knuth-Morris-Pratt (1977) algorithm and it solves the problem with n-m comparisons. The time complexity of Knuth-Morrsi-Pratt algorithm is $O(m+n)$. This algorithm uses the information gained by previous symbol comparisons and never compares a text symbol that has matched a pattern symbol again. The complexity of the searching phase of the Knuth-Morris-Pratt algorithm is $O(n)$. Of course, this algorithm requires pattern preprocessing with complexity of $O(m)$ to analyze structure making the overall complexity $O(m+n)$. Since usually $m < n$, the complexity of the Knuth-Morris-Pratt algorithm is approximately $O(n)$.

Hashing was first proposed by Harrison (1971) and provides a simple method that avoids the quadratic number of character comparisons in most practical situations, and runs in linear time under reasonable probabilistic assumptions. He proposed a fast implementation of a test to determine if one string contains a specified substring. The proposed hashing technique used the ability to do many Boolean operations in parallel on a standard computer. Later, Karp and Rabin (1987) presented an algorithm where a hash function h is used for strings of length m. Instead of checking at each position of the text if the pattern occurs, it is more efficient to check only if the contents of the window look like the pattern. Hashing is used to check the resemblance between these two words. In this algorithm, the preprocessing has a time complexity $O(m)$ and searching has a time complexity $O(mn)$ making the overall expected number of text character comparisons $O(n+m)$.

Other algorithms were presented later. For example, in the Boyer-Moore Algorithm both character-jump heuristic and looking glass heuristic are applied. Here the worst case run-time is $O(n*m+|\Sigma|)$ where Σ denoted the alphabet, and the runtime can be improved to $O(m+n)$ by using the good-suffix heuristic.

IV.2. String Matching and Data Security over the Internet

String matching can be effectively used to improve data security over the internet to improve reliability. Goel and Bush (2003) presented a distributed model for security based on biological paradigms of epidemiology and immunology whereby each node in the network has an immune system that identifies and destroys pathogens in the incoming network traffic as well as files resident on the node. In this scheme, each node compiles a list of pathogens that are perceived as threats by using information from all other nodes. The signatures are incorporated into the detector population of the immune systems to increase the probability of detection. They clearly state that the detection scheme is the most critical part for the success of the proposed system. They examined three separate schemes for detecting pathogens namely, contiguous string matching, Hamming distance, and Kolmogorov Complexity. Brönnimann et al. (2005) studied string matching in a stream of network packets as part of a larger system for facilitating network forensics across and within networks. The proposed system monitored

network traffic, created hash-based digests of payload, and archived them periodically. A user-friendly query mechanism provides the interface to answer post-mortem questions about the payload.

Another application of string matching is in data mining, mirroring, storage, and content distribution. Managing large collection of replicated data in centralized or distributed environments is very important. It is true that redundancy can be used to increase the reliability. However, uncontrolled redundancy would aggravate the performance of the system in the retrieval phase. It may even be useless if the returned documents are obsolete. Document similarity matching algorithms do not provide the information on the differences of documents. Moreover, file synchronization algorithms are inefficient since they ignore the structural and syntactic organization of documents.

Another application of string matching is in detection of plagiarism which is of interest to publishers. There are a variety of methods used in plagiarism detection. However, the usual trade-off between speed and reliability still remains. Mozgovoy et al. (2007) introduced a two-step approach to plagiarism detection that combines high algorithmic performance with the quality of pairwise file comparison. In their proposed system, a fast detection method is used to select suspicious files first and then more precise and naturally slower algorithms are used to get reliable results.

Ayqun (2008) proposed a matching approach called S2S that is composed of structural and syntactic phases to compare documents. In this approach, the documents are first decomposed into components by syntax in the structural phase and are compared at the coarse level. The decomposed documents are processed in the structural mapping phase based on syntax without actually mapping at the word level. Then a syntactic matching algorithm uses a heuristic look-ahead algorithm for matching consecutive tokens with a verification patch.

V. INTERNET SECURITY

The World Wide Web is constructed from programs called Web servers and Web browsers. Many companies use the web for electronic commerce, but it poses profound security challenges such as

1- Possible unauthorized access to other files in the computer system using bugs in the Web server or CGI scripts.
2- Unauthorized distribution of confidential information on the server.
3- Interception of transmission of confidential information.
4- Access to confidential information on the client.
5- Potential threat of specially licensed software meant to combat internet security issues.

As more corporate computer systems are connected to the internet and more transactions take place over computerized systems, the identification and prevention of cyber misuse becomes increasingly critical. Owens and Levary (2006) presented an adaptive expert system for intrusion detection that uses fuzzy sets with the ability to adapt to the type and/or degree of the threat.

There is a need for a more intuitive, automated systems-level approach to determining the overall security characteristics of a large network. Given the complex nature of security tools and their general lack of interoperability, it is difficult for system designers to make definitive statements about the nature of their network defense. Rasche et al. (2007) presented an approach for automatically verifying the correctness of cyber security applications through formal analysis guided by hierarchical models of the network, applications, and potential attacks.

VI. INTRUSION DETECTION SYSTEMS

The use of Network Intrusion Detection System (NIDS) has been increasing due to the rising trend in cyber crimes. Software based solutions for NIDS are inefficient when they are employed on high speed high volume networks. Many researchers have studied hardware solutions hoping to acquire a much higher efficiency. However, these solutions pose major problems of flexibility, reliability, scalability, efficiency, speed and cost.

Intrusion detection systems (IDS) are software and/or hardware solutions meant to detect unwanted attempts at accessing, manipulating or disabling computer systems through networks. An IDS consists of several components including sensors to generate security events, a console to control the sensors and monitor events and alerts. It also includes a central engine to record sensed events in a database. The IDS uses a system of rules to generate alerts from security events received.

It is very likely that an intruder who breaks into a computer system may behave much different from a legitimate user. Lunt et al. (1990) designed and developed a real-time intrusion-detection expert system (IDES) that observes user behavior on one or more monitored computer systems and flags suspicious events. It monitors the activities of individual users, groups, remote hosts and entire systems to detect suspected security violations. The main feature of IDES is that it adaptively learns

users' behavior patterns over time and detects any deviation from this behavior. Their next step was the development of NIDES that performs real-time monitoring of user activity on multiple target systems connected via Ethernet to analyze audit data collected from various interconnected systems and search for unusual user behavior.

Popular Intrusion detection systems include Snort as an open source IDS, OSSEC HIDS as an open source host based IDS, Fragroute as a network intrusion detection evasion toolkit, BASE as a basic analysis and security engine, and Sguil as the analyst console for network security monitoring.

The most popular one is Snort that is an open source network intrusion prevention and detection system (IDS/IPS) written by Martin Roesch at Sourcefire. It can perform real-time traffic analysis and packet logging on IP networks. Snort can also perform protocol analysis and content searching/matching, detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and OS fingerprinting attempts. Snort uses a flexible rules language to describe traffic that it should collect or pass plus a detection engine that utilizes a modular plug-in architecture. OSSEC is an Open Source Host-based Intrusion Detection System that performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response, and can run on most operating systems. Fragroute has a simple ruleset language. It can delay, duplicate, drop, fragment, overlap, print, reorder, segment, source-route, etc. all outbound packets.

Choi and Lee (2005) presented a parallel coordinate attack visualization tool called PCAV for detecting unknown large-scale attacks including worms, DDOS, and network scanning. They used hashing to develop nine attack signatures and their detection mechanism.

The effectiveness and precision of network-based intrusion detection signatures can be evaluated either by direct analysis of the signatures or by using black-box testing. Recently, several techniques have been proposed to generate test cases by automatically deriving mutations of attacks. Kruegel et al. (2007) proposed an approach for test case generation by using the information gathered by analyzing the dynamic behavior of the intrusion detection system. They applied dynamic data flow analysis techniques to the intrusion detection system to identify which parts of a network stream are used to detect an attack and how these parts are matched by a signature.

Intrusion detection is an indispensable part of cyber security. Bhatia et al. (2008) presented the integration of Host-based Intrusion Detection System (HIDS) with existing network based detection on Gen 3 Honeynet architecture involving the stealth mode operation of HIDS sensor, code organization to generate HIDS alerts, enhancement of the functionality of data fusion, and further visualization on Graphical Analysis Console.

VII. A Contribution to Data Security on the Internet

Proodfoot et al. (2008) proposed a system that uses a modified version of Snort. Their proposed system runs Snort in software until it gets to the pattern matching function and then offloads that processing to the FPGA. The designed hardware is claimed to be able to process data at up to 1.7GB/s on one Xilinx XC2VP100 FPGA. Since the rules are not coded in hardware, the proposed system is more flexible than other FPGA string matching designs. Since their proposed design allows parallel use of FPGAs to increase speed, it is claimed to be scalable.

A scalable high performance content processor was designed by the author for storage disks which could be easily installed in any host as an interface between the hard disk and the system bus to improve internet reliability. Moreover, a novel and powerful exact string matching architecture was presented to search for several thousand strings at high rates. The proposed architecture was implemented on a Xilinx XC4VFX100 Field Programmable Gate Array (FPGA) and it was shown that the system can search for over sixteen-thousand 32 byte strings with a speed near the maximum stated in ATA-7 standard. The design showed a much better performance as measured in Throughput/(LogicCells/Char) when compared with the best existing designs. Later, this design was upgraded and the matching architecture was implemented on a Virtex4 FX 100 -11 chip using Xilinx ISE 10.1i that supports up to 16K single size signatures of 32 bytes. The design achieves a Throughput/(LogicCells/Char) of 31.6 that is much better than any existing system since it only uses embedded on-chip memory blocks, and the logic resources required for implementing it are independent of the number of strings. This contribution has been reported elsewhere and is accepted for publication [Peiravi and Rahimzadeh (2009)].

VIII. Measurement of Internet Availability

Peiravi and Shahraeeni (2004) developed a Web Availability Analyzer Software Tool to measure the

availability of the internet as shown in Figure 2. The tool was run for 90 days to measure reliability data for 159 Iranian hosts. The hosts chosen included 18 universities, 6 news agencies, 36 internet service providers, 29 government agencies and the rest were other public sites. This mix was chosen so as to obtain an average measure of intrinsic internet availability in Iran. This unavailability data measurement was executed from two different points of connection to the internet to remove any unavailability data related to the facilities of the measurement site itself, and purely obtain the behavior of the hosts under study.

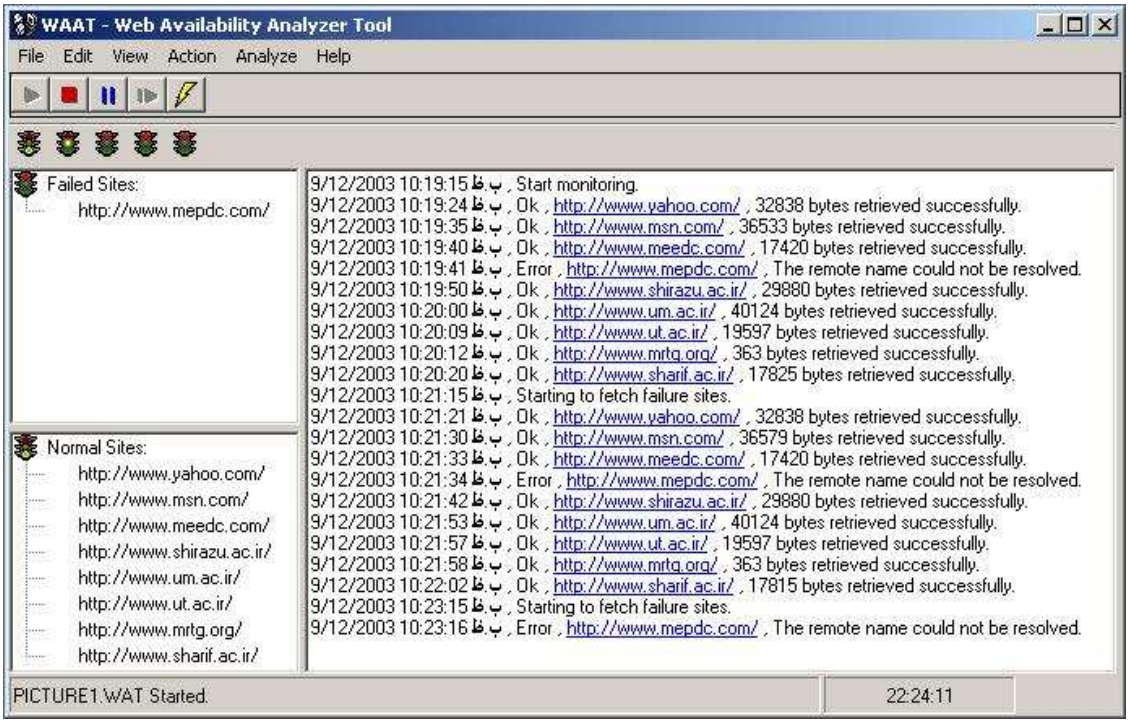

Fig. 2. A sample of WAAT running

The tool generated a log file for each 24 hours. Later analysis of these log files revealed the following results as shown in Tables 2, 3 and 4. We used ping to exclude failures related to intermediate lines and nodes, and thus eliminated any failures due to the internet backbone, too.

Table 2 - The mean and median MTTF values

Mean MTTF	21.56 days
	±0.68 (50% confidence)
	±1.82 (95% confidence)
	±2.46 (99% confidence)
Median MTTF	14.24 days
Number of hosts	159

Availability can be computed from the data shown in Tables 2 and 3 as shown in Table 4.

Table 3 - The mean and median MTTR values

Mean MTTR	3.375 days
	±0.310 (50% confidence)
	±0.668 (95% confidence)
	±0.908 (99% confidence)
Median MTTR	1.012 days
Number of hosts	159

Table 4 - The mean and median Availability values

Mean availability	0.865
	±0.007 (50% confidence)
	±0.012 (95% confidence)
	±0.016 (99% confidence)
Median availability	0.934
Number of hosts	159

Long et al. (1995) had reported a similar study with the following results after surveying 1170 hosts that were uniformly distributed over the name space and could respond to RPC polls for 90 days. Their results are shown in Tables 5 and 6.

Table 5 - The mean and median MTTF values

Mean MTTF	15.92 days
	±0.28 (50% confidence)
	±0.82 (95% confidence)
	±1.08 (99% confidence)
Median MTTF	5.53 days
σ	28.86

Table 6 - The mean and median MTTR values

Mean MTTR	1.201 days
	±0.021 (50% confidence)
	±0.062 (95% confidence)
	±0.082 (99% confidence)
Median MTTR	0.3394 days
n	3500 intervals
σ	1.885

Availability was computed from the data shown in Tables 5 and 6 as shown in Table 7.

Table 7 - The mean and median Availability values

Mean MTTR	0.9260
	±0.002 (50% confidence)
	±0.007 (95% confidence)
	±0.009 (99% confidence)
Median MTTR	0.9723
Number of hosts	1162

Table 8 – A comparison of results for the survey on Iranian hosts and international hosts

Hosts	**MTTF (days)**	**MTTR (days)**	**Availability**
159 Iranian Hosts	21.56	3.375	0.865
1162 International Hosts	15.92	1.201	0.9260

Table 8 shows a comparison of the results of the two studies. This indicates an average lower level of reliability for Iranian hosts, compared with international hosts indicating that a lot more work is needed for Iranian hosts to reach the average international availability levels.

IX. CONCLUSIONS

In this paper the various issues related to the reliability of the internet were reviewed. Hardware redundancy and software issues were presented as means of improving internets intrinsic reliability. A new measure was proposed to include down time due to cyber attacks in internet availability. The issue of data security that is the next most important aspect of internet reliability was addressed. Results of the development of a new string matching architecture implemented on an FPGA for implementation in fast hardware based data security applications such as intrusion detections systems or virus scanners was presented. Results of actual measurement of internet availability for hosts in Iran were presented and compared with international hosts indicating lower availability for the Iranian hosts.

Acknowledgement

I would like to express my appreciation for the sincere efforts of Mr. Muhammad Shahraeeni and Mr. Muhammad Javad Rahimzadeh for the good work they performed on their Master's thesis under my supervision. This work is a continuation of research carried out here at the Ferdowsi University of Mashhad and was supported by the Grant No. 4270 (1388/4/28) Project of the Vice Chancellor of Research and Technology of the Ferdowsi University of Mashhad.

References

[1] Strom, D., Where businesses go for internet reliability, The New York Times, October 1, 2008.

[2] Leighton, T., Improving performance on the internet, Communications of the ACM, Vol. 52, No. 2, pp. 44-51, 2009.

[3] Knuth, D., Morris, J. H., Pratt, V., Fast pattern matching in strings, SIAM Journal on Computing, Vol. 6, No. 2, pp.323–350, doi:10.1137/0206024, 1977.

[4] Harrison, M. C., Implementation of the substring test by hashing, Communications of the ACM, Vol.14, No.12, pp. 777-779, 1971.

[5] Karp, R. M., Rabin, M. O., Efficient randomized pattern-matching algorithms, IBM J. Res. Dev., Vol. 31, No. 2, pp. 249-260, 1987.

[6] Goel, S. Bush, S. F., Kolmogorov complexity estimates for detection of viruses in biologically inspired security systems: A comparison with traditional approaches, Complexity, Vol. 9, No. 2, pp.54-73, 2003.

[7] Brönnimann, H., Memon, N., Shanmugasundaram, K., Lecture Notes in Computer Science 3405, pp. 75-89, 2005.

[8] Mozgovoy, M., Karakovskiy, S., Klyuev, V., Fast and reliable plagiarism detection system, Proceedings - Frontiers in Education Conference, FIE, pp. S4H11-S4H14, 2007.

[9] Ayqun, R. S., S2S: Structural to syntactic matching similar documents, Knowledge and Information Systems, Vol.16, No. 3, pp. 303-329, 2008.

[10] Owens, S. F., Levary, R. R., An adaptive expert system approach for intrusion detection, International Journal of Security and Networks, Volume 1 , Issue 3/4, pp.206-217, 2006.

[11] Rasche, G., Allwein, E., Moore, M., Abbott, B., Model-based cyber security, Proceedings of the International Symposium and Workshop on Engineering of Computer Based Systems, pp. 405-412, 2007.

[12] Lunt, T. F., Tamaru, A. Gilham, F. Jagannathan, R.Neumann, P. G. Jalili, C., IDES: A progress report, Proc. of the Sixth Annual Computer Security Applications Conference, 1990.

[13] Choi, H., Lee, H., PCAV: Internet attack visualization on parallel coordinates, ICICS 2005, LNCS 3783, pp. 454~466, Dec. 2005.

[14] Kruegel, C., Balzarotti, D., Robertson, W., Vigna, G., Improving signature testing through dynamic data flow analysis, Proceedings - Annual Computer Security Applications Conference, ACSAC, pp. 53-63, 2007.

[15] Bhatia, J. S. Sehgal, R. Bhushan, B. Kaur, H., Multi layer

cyber attack detection through honeynet, Proceedings of New Technologies, Mobility and Security Conference and Workshops, NTMS 2008, 2008.
[16] Proodfoot, R., Kent K., Aubanel, E., Chen, N., Flexible software-hardware network intrusion detection system, Proceedings of the 19th IEEE/IFIP International Symposium on Rapid System Prototyping - Shortening the Path from Specification to Prototype, RSP 2008, pp. 182-188, 2008.
[17] Peiravi, A., Rahimzadeh, R. A novel scalable and storage-efficient architecture for high speed exact string matching, accepted for publication in ETRI Journal.
[18] Peiravi, A., Shahraeeni, M, Implementation of a measurement tool for the reliability measurement of the Web server of Ferdowsi University of Mashhad, A paper published in Farsi in the Proceedings of the ICEE2004 (12th Conference on Electrical Engineering, Ferdowsi University of Mashhad, May 11-13, 2004, pp.50-55, 2004.
[19] Long, D., Muir, A., Golding, R., A longitudinal survey of internet host reliability, Proc. Symposium on Reliable Distributed Systems, Sept. 1995.