ANTI WEB SITE DEFACEMENT SYSTEM (AWDS)

Dr. Mazin S. Al-Hakeem
Director of ICT Center, University of Technology, Baghdad, Iraq.
mazin_ictc@yahoo.com , dr.mazin@uotechnology.edu.iq

ABSTRACT:

Network security controls (preventive controls) like firewall, VPN (Virtual Private Network), PKI (Public Key Infrastructure) and others are important tools to keep the web more secure, but they are not enough to ensure web site security. Therefore, the web environment must be viewed as a whole.

In this paper, Anti Web site Defacement System (AWDS) is proposed and implement to detect any defacement attack (one of the most widely known Internet attacks) using multi-checker, and recover the defaced web files without taking web into offline status. The AWDS has immunity to stop if any checker is not functioning (failed) for any reason, and its surveillance of its parts. The AWDS employs improve - Rabin's fingerprint algorithm to guarantee the integrity of web pages.

The AWDS complement the preventive controls as the next line of defense. Scheme has been applied to work in the Central Internet Service Station in University of Technology.

Keywords:-
Anti Web site Defacement System (AWDS), Defacement Attack, Web Integrity Checking, Improved Rabin's Fingerprinting, and Multi-Checker.

نظام مضاد هجمات تشويه مواقع الانترنت

المستخلص:

ان متحكمات امن الشبكات (المتحكمات الوقائية) مثل جدران النار، الشبكات الخاصة الافتراضية، PKI وغيرها هي ادوات هامة لجعل الشبكة اكثر امان، لكنها ليست كافية لجعل مواقع الويب امنة، وعليه فأن النظر الى البيئة الشبكية يجب ان يكون بشكل اشمل.

في هذا البحث تم اقتراح وتنفيذ نظام مكافحة تشويه مواقع الانترنت (AWDS) الذي يهدف الى اكتشاف هجمات التشويه (احد اشهر انواع الهجمات التي تحدث على شبكة الانترنت) بأستخدام الفواحص المتعددة، واعادة اصلاح ملفات مواقع الانترنت المشوهه دون توقف عمل الموقع. ان نظام AWDS المنفذ يمتلك المناعة ضد التوقف في حاله فشل اي فاحص من اداء وظيفته لحدوث اي سبب، وامتلاك النظام خاصية مراقبة اجزاءه. ان نظام AWDS المنفذ يستخدم خوارزمية رابين fingerprint المحسنة لضمان سلامة صفحات مواقع الانترنت.

وان هذا النظام هو مكمل للمتحكمات الوقائية ويعمل كخط دفاع خلفها، وقد طبق للعمل في محطة خدمة الانترنت المركزية في الجامعة التكنولوجية.

الكلمات المفتاحية:-
نظام مكافحة تشويه مواقع الانترنت، هجوم تشويه مواقع الانترنت، فحص سلامة مواقع الانترنت، خوارزمية رابين لبصمة-الاصبع المحسنة، والفواحص المتعددة.

1. INTRODUCTION:

One of the most widely known attacks is the web site defacement attack [1]. It is usually malicious (viruses, worms, trojan horses, and other malicious codes), designed to delete, modify or replace web pages on a host (Web Server) [2].

For different purposes, the web site vulnerability is a potential target of hacking. The hackers have several tools to search widely and quickly for web site vulnerabilities (particular weaknesses) and move swiftly and stealthily to exploit those weaknesses [3].

Web site defacement attacks were done to violating web integrity (correctness) by one of the following violations [1]:

- Change the content of a web page.
- Change any part of the content of a web page.
- Replace a web page entirely.
- Reuse an old web page.
- Change the apparent source of a web page.
- Redirect a web page.
- Destroy or delete a web page.

The network security controls (preventive controls) like firewall, VPN (Virtual Private Network), PKI (Public Key Infrastructure) and others are important tools to keep the web more secure, but they are not enough to ensure web site security, because such attacks cannot be prevented at higher networking layers, rather security mechanisms need to be provided [1].

Figure (1) illustrates how we can use a combination of controls to secure the valuable resources [1]. We use one or more controls, according to what we are protecting, how the cost of protection be compared with the risk of loss, and how hard we think intruders will work to get what they want [1]. Therefore, the web environment must be viewed as a whole, and the web site administrator should project forward the effects of the attacks.

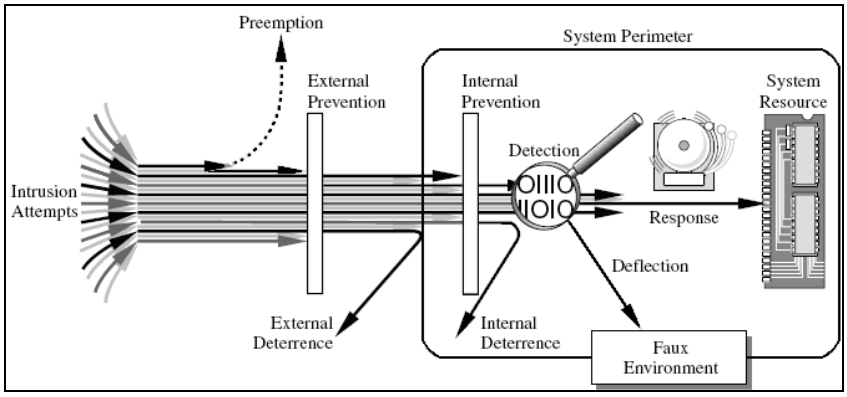

Figure (1): Combination of Controls to Secure the Valuable Resources.

However, there are several addressed systems like Integrit[4], Veracity[5], Aide[6], L5[7],Tripwire[8], and WHDRM[9] which use to protect web site against attacks, but each one has a specific drawback as follows:

- Integrit, Veracity, Aide and L5 lack protecting themselves when they have been attacked and will leave the website unprotected.
- Tripwire lacks generating any warning if its process is failed by the attacker.
- WHDRM takes the web into offline status during the recovery of the hacked files.

And all the above systems are stopped and have no security value if the checker-part is failed for any reason.

The proposed AWDS was design and implemented to work under speed and efficient terms much more than pervious addressed systems because it depends on multi-checker subsystem based on fast fingerprint algorithm to guarantee the integrity of web pages, Self-Watching against attack and works as a trusted system, generates variety warning messages, and recovers the defacement web pages without taking the web into offline status. The AWDS has partial security value and wouldn't stop if any checker is failed for any reason.

2. DOCUMENT FINGUREPRINT:

In computer science, fingerprint uniquely identifies the original data for all practical purposes just as human fingerprints uniquely identify people for practical purposes [10]. A document fingerprint is a collection of integers that represent some key-content of the document. Each of these integers is referred to as a "minutia" [10].

Typically, a fingerprint is generated by selecting substring from the text and applying a mathematical function to each selected substring [11]. This function, similar to a hashing function, produces one minutia. The minutia is then stored in an index for quick access when querying. When a query document is compared to the collection, the fingerprint for the query is generated. For each minutia in the fingerprint, the index of queried and a list of matching fingerprints are retrieved. The number of minutia in common between the query fingerprint and each fingerprint in the collection determines the score of the corresponding document [11].

There are several methods for fingerprinting based on variation in the four below design parameters [10]:

A- ***Selection strategy*** (which is used to select substrings from the document - the selection strategy).
B- ***Size of the substrings*** (which are extracted from document - the granularity).
C- ***Number of minutia*** (which is used to build a document fingerprint - the resolution).
D- ***Fingerprint function*** (which is used to generate a minutia from substring in the document, such as checksums, hash functions, cryptographic hash functions, and digital signatures).

3. THE PROPOSED IMPROVE RABIN'S FINGERPRINTING ALGORITHM:

Rabin's fingerprinting algorithm is one of many fingerprint algorithms, which implements public key using polynomials over a finite field [12].

In this paper, we choose and improve the typical Rabin's fingerprinting algorithm to generate a minutia from substring in the web pages, because it's a fast and easy to implement, and it comes with a mathematically precise analysis of the probability of collision (two files yielding the same fingerprint). The improved Rabin's fingerprinting algorithm is used in the proposed AWDS system. It is as the following:

The Improve Rabin's Fingerprinting Algorithm:

Input: document (a published web page)
Output: document's fingerprint

Step1: Begin

Step2: Remove all white spaces and special characters (like <, >, %, !, etc.) from HTML code (web page code) to obtain a pure text block.

Step3: Partitioning the pure text block into K-length substrings (K must be efficient as possible as the match is detected).

// The number of K-length substrings and hence the number of hashed is closed to the size of the document. Simply, it is equal to (m-K+1), where m is the size of the document.

Setp4: Compute the minutia for each substring by calculating H(P) as follows:

// H(P) is a linear in n (the length of P)

Initialization:

Count= K

Tr = T[r .. r+n-1]

$H(S) = S(n) + 2*S(n-1) + 4*S(n-2) + ... + 2^{n-1}*S(1)$

Do while Count > 0

Use $H_p(P) = H(P) \bmod p$ as a fingerprint of P

// This may result in false positives, which can either be eliminated by checking, or the probability of false positives can be bounded by picking p appropriately and using multiple moduli.

$H_p(T_r) = [\ 2*H_p(T_{r-1}) - (2^n \bmod p) * T(r-1) + T(r+n-1)\ ] \bmod p$

dec Count by 1

Step5: All selected values are saved as the fingerprints of the document (Webpage).

Step6: End

As a result of improving the algorithm, it became:

1- White-Space & Special Characters Insensitivity: In matching the code of web pages, matches should be unaffected by such things as extra white-space or special characters, capitalization, punctuation, etc.
2- Noise Detected: Finding out (discovering) short matches is important to imply that the material has not been modified.

4. THE PROPOSED AWDS:

The main purpose of the Anti Web site Defacement System (AWDS) is to detect any web defacement attack and to recover the defaced web files. To achieve this mission, the proposed AWDS was designed and implemented upon 2 servers (Web-Server and AWDS-Server) with five integrated subsystems and centralized DB. These subsystems and their main functions are as follows:

Subsystem #	Subsystem Name	Functions	Place
I	**Builder**	It publish a given web pages, and maintains the check-list which includes the web sites file names with their attributes and minutia.	AWDS-Server
II	**Multi-Checkers**	Monitoring the integrity of the published web pages.	Web-Server
III	**Recover**	It restores the original web pages when the published one is defaced.	AWDS-Server
IV	**Self-Watcher**	Verifying Checker status.	AWDS-Server
V	**Admin**	It controls all different AWDS parts.	AWDS-Server
and,	**Check-List Database**	Also, a database, which store files with their minutia, attributes, and interval times.	AWDS-Server

The recent copies of web pages and the maintenance pages are stored in an external storage area (called "Recovery Files") on different server at a different network zone. Those parts and the relationship between them are shown in figure (2).

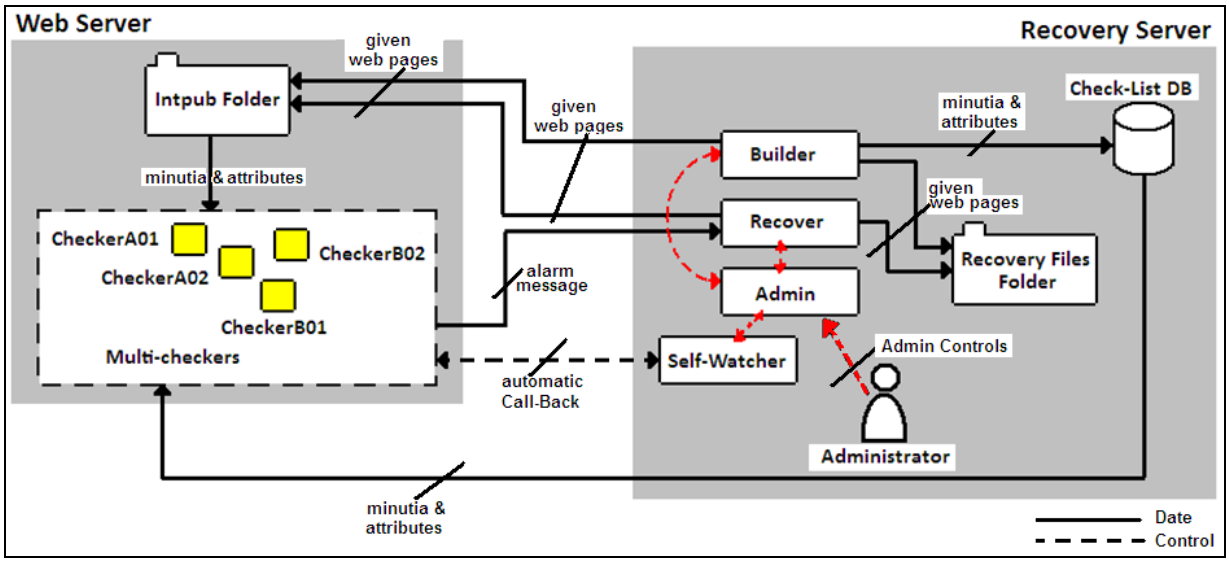

Figure (2): AWDS Architecture Diagram.

4-1. THE BUILDER SUBSYSTEM:

The Builder subsystem is running at AWDS-Server when the Admin subsystem adds, edits, or removes web pages.

The Builder subsystem works as following:

1- Computes the minutia and attributes (name, size of file, type, and date modified) of the given web pages (newly added web page or edited one).
2- Stores those computed information into the check-list DB, and the Builder subsystem maintain the check-list DB.
3- Save the given web pages in the Recovery-Folder at AWDS-Server.
4- Publish the given web pages at the Intpub Folder (typically Windows Internet publishing folder) at Web-Server.

4-2. THE MULTI-CHECKER SUBSYSTEM:

The multi-checker subsystem is the heart of AWDS, it contains several checkers (in this paper we implement four checkers), which works nearly Intpub folder at Web-Server.

Each checker periodically checks the integrity of subset of the published web pages, and the important web pages such as Home page (index.htm, index.html, default.asp, default.aspx, start.php, home.php, default.asp) will be checked frequently. So there is a time interval associated to each file to recheck the integrity.

The checkers are working depending on one of the following modes:

A- **The basic mode (Quick mode):** When the multi-checker subsystem is been started, the checkers computes the attributes (name, size of file, type, and date modified) for the current web page (which stored on the Intpub) and compare it with the stored one in the Checked-list DB. Figure (3) shows the fourth basic file Attributes.

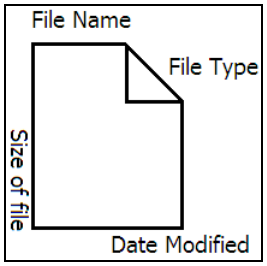

Figure (3): The Basic File Attributes

B- **The advanced mode:** When the multi-checker subsystem is been started, the checkers compute the minutia (employ improved Rabin's fingerprint) for the published web page (which is stored on the Intpub) and compare it with the stored one in the Checked-list DB.

The implemented multi-checker subsystem is content of two checkers working at basic mode (checkerB01 and checkerB02), and two checkers working at advanced mode (checkerA01 and checkerA02).

However, the status of each checker is being monitored by the Recover subsystem. The statuses of the checker are as follows:

A- Negative Statues (Detect Defacing Attempts):

- If the current web page is not listed in the check-list DB, is mean that the current web page is an illegal file (malicious file, virus, …), and the checker generates 'malicious-alarm', then sends it as an acknowledgment to the Recover subsystem, and continues to check the next web page.
- If the minutia (in advanced mode) or any attribute (in basic mode) of the current web page is different than the original one, is mean that the current web page is defaced, and the check subsystem generates 'defaced-alarm', then sends it as an acknowledgment to the Recover subsystem, and continuing to check the next web page.
- If the web page's name was existing in the check-list DB and not existing in the Intpub folder, is mean that the web page is deleted from publishing Intpub folder, and the check subsystem generates 'delete-alarm', then sends it as an acknowledgment to the Recover subsystem, and continues to check the next web page.

B- Positive Statues (Detect Nothing Attempts):

- Else, the checker does nothing, and continues to check the next web page.

4-3. THE RECOVER SUBSYSTEM:

Successful defacement attack attempts are discovered when non-listed file name appears, a calculated file attributes or minutia are more different than the original one, or the listed file not appeared. In case of defacement detected, the Recover subsystem will be triggered by the multi-checkers subsystem.

When Recover subsystem is invoked, the Recover subsystem will restore the original web pages depending on it policy, as following:

1- Delete the current web page from Intpub folder, if the non-listed file name appears.
2- Restore the original web pages, if the calculated web page attributes or minutia are different than the original one.
3- Restore the original web pages, if the listed file is not appeared.

And the Recover subsystem notifies the administrator by sending a suitable alarm message (malicious-alarm, defaced-alarm, or delete-alarm).

4-4. THE SELF-WATCHER SUBSYSTEM:

A serious vulnerability to an AWDS is the multi-checker subsystem. The self-watcher subsystem is watching each checker at multi-checker subsystem, to ensure that the checkers are active. The self- watcher works to verify the checkers' statuses. The multi-checker subsystem is the heart of AWDS, so if any checker is not functioning for any reason (such as being attacked), the AWDS has partial security value.

However, the self-watcher works to insure that the checkers are active and functioning, by "automatic Call-Back" procedure to verify the checkers statuses. In each given time interval, the self-watcher dials a checker by sending a random number, the checker breaks the communication line. And the checker calls the self-watcher back.

If the self-watcher does not receive the call within a given timeout period, it redials at the same way. If there is no answer within a given maximum timeout, (it means that the checker is failed or the connection line between two servers is disconnected), the self-watcher reports a 'failed-alarm' warning to the admin subsystem.

4-5. THE ADMIN SUBSYSTEM:

The administrator of the implemented system can control different AWDS components using this subsystem. For example, add a new web page, edit an existing web page, or delete web page from the check-list DB and Recovery-Folder. By using admin subsystem, the administrator can set the interval-time of each web page for checking and initialize the message-warnings and notifications.

5. THE AWDS IMPLEMENTATION:

The AWDS is implemented using ASP and VBScript, and the Chick-List database is implemented by MS-Access 2007. Then we re-configuring an existing ComTech-Modem (which islisted below) to authenticate the stream of AWDS traffic between the servers. Both servers (Web-Server and Recovery-Server) are implemented at Central Internet Service Station in ICT Center at University of Technology, with the specifications which listed in table (1).

Table (1): The Specifications of Central Internet Service, Web-Server and Recovery-Server:

Item	General Features
Central Internet Service	- VSAT Modem ComTech CDM 570L TPC 8 PSK. - Cisco Router with HWIC 1T – EIA 530MT cable. - Dedicated 12 Mbps bandwidth (2Mbps Upload / 10Mbps Download). - Unlimited bandwidth Traffic per month. - C-Band SCPC/DVB-S Satellite Sign.

Web-Server	- Windows Server 2003 OS.
	- Default Internet Information Services (IIS).
	- HP ProLiant DL320 Generation 5 (G5):
	  - Processor Type: Intel® Celeron with 3.2 GHz.
	  - Processor Cache: 512KB L2 Cache.
	  - Number of Processors: 1.
	  - Max front side bus: 1066 MHz.
	  - Standard Memory: 8 GB (Maximum) DDR2 SDRAM Memory.
	  - Storage Type: Hot plug 3.5-inch SAS.

Recovery-Server	- Windows XP SP2 OS.
	- Default Internet Information Services (IIS).
	- HP ProLiant DL320 Generation 5 (G5):
	  - Processor Type: Intel® Celeron with 3.2 GHz.
	  - Processor Cache: 512KB L2 Cache.
	  - Number of Processors: 1.
	  - Max front side bus: 1066 MHz.
	  - Standard Memory: 8 GB (Maximum) DDR2 SDRAM Memory.
	  - Storage Type: Hot plug 3.5-inch SAS.

6. AWDS EVALUATION:

Evaluating of the AWDS is to measure the average score for checking the web pages' integrities per time units. In this paper, 340 web pages (the width of actual published UOT "University of Technology" web site, at www.uotechnology.edu.iq) have been used as a data set. However, many web page defacement (add/ delete/ modify) are occurred to achieve the measurement. The average scores of integrity checking (for both basic mode and advanced mode) are shown in table (2).

Table (2): Average Score of Integrity Checking (for both basic mode and advanced mode):

No. of Defaced Pages	Type of Web Defacement	Based Basic Mode		Based Advanced Mode	
		Average Score of Integrity Checking	Totally Checking Time (Min : Sec)	Average Score of Integrity Checking	Totally Checking Time (Min : Sec)
No pages	No Deface.	14 pages/sec	00 : 24	9 pages/ sec	00 : 37
1 page	Modify	13 pages/sec	00 : 26	8 pages/ sec	00 : 42
10 pages	Add/ Delete/ Modify	9 pages/ sec	00 : 37	6 pages/ sec	00 : 56
20 pages	Add/ Delete/ Modify	3 pages/sec	02 : 28	2 pages/ sec	03 : 23
50 pages	Add/ Delete/ Modify	1 pages/sec	06 : 06	0.5 pages/ sec	11 : 33

Figure (4) illustrates the diagram of average score of integrity checking per time units for both basic mode and advanced mode.

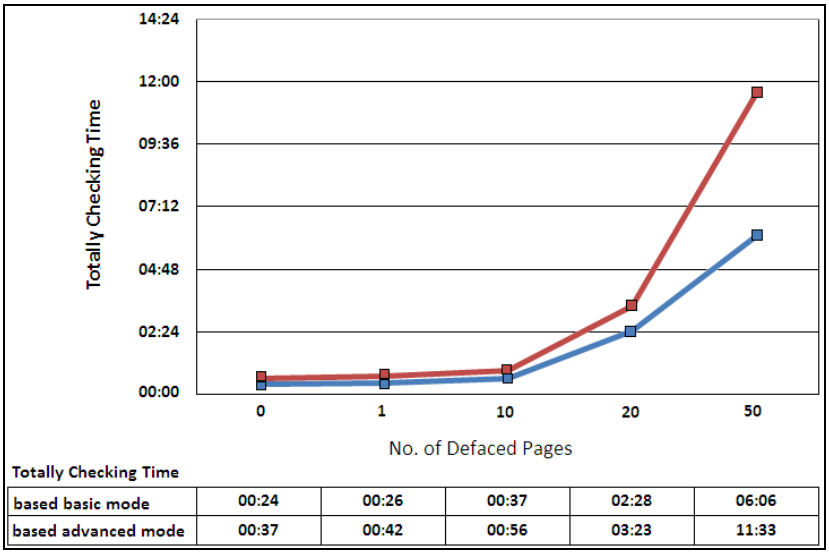

Figure (4): Average Score of Integrity Checking
(For both basic mode and advanced mode)

It is difficult to measure the average score for recovery of the original pages, because of the variety of defacement attacks. However, the worst case is that all web pages are modify-defaced. In this case the totally calculated time is (**01**H : **12**Min : **27**Sec).

7. DISCUSSION AND CONCLUSIONS:

In this paper we briefly described the web vulnerabilities and several violating web integrity attacks possible on such environment. Presented the related work in this area and its drawbacks, and then we presented the proposal and the implementation of our Anti Web site Defacement System (AWDS).

However, the AWDS is proposed and implemented to detect any defacement attack and recover the defaced web files by the original ones, without taking web into offline status (without diverts web visitors to an alternative site). The AWDS complements the preventive controls as the next line of defense to keep the web more secure, and to complement the preventive controls (like firewall, VPN, PKI and other controls).

The AWDS employs improved Rabin's fingerprint algorithm to guarantee the integrity of web pages. At the beginning, the minutia of the web pages must be calculated and stored in the AWDS database. The AWDS continuously checks the integrity of the web pages based on their interval time, using multi checker. When a computed minutia of the published web page is not equal to the stored minutia, the non-listed web page name appears, or the listed web page would not appeared, the AWDS will be triggered to restore web pages depending on its policy, without taking web into offline status. AWDS has the ability of self watching, to ensure that the checkers are active. However, if any checker is not functioning (failed) for any reason, the AWDS has partial security value and would not be stopped.

However, the important conclusions of our work are as following:

1- Generally, there is no totally secure web environment.
2- The AWDS works with an efficient term much more than an addressed systems like Integrit, Veracity, Aide and L5 which lack protecting themselves when they have been attacked and will leave the website unprotected, and Tripwire which lacks the generation of any warning if its process is killed by the attacker, and WHDRM will take the web into offline statue during the recovery of the hacked files.
3- The proposed and implemented AWDS works with a speed term more much than the above addressed systems, because it depends on multi-Checker to achieve a continuously checking Phase.
4- The advantage of the improve Rabin's fingerprint algorithm is that it takes considerably shorter to execute than cryptographic hash algorithms such as MD5 (which is used in WHDRM, L5, and other systems) and SHA, and it offers provably strong probabilistic guarantees that two different strings will not have the same fingerprint (collision probability). Other checksum algorithms (such as MD5 and SHA) do not offer such provable guarantees, and are also more expensive to compute than fingerprints, so it still may be useful for error checking.

8. REFERENCES:

(1) **"Security in Computing**", 3rd Edition, Charles P. Pfleeger and Shari Lawrence, Prentice Hall – 2003 (*available at http://books.google.com/books?id=O3VB-zspJo4C&dq=%22web+site+defacement+attack+%22&source=gbs_navlinks_s*).

(2) **"Cryptography and Network Security"**, William Stalling, Prentice Hall – 1999.

(3) "**Surviving security: how to integrate people**", process, and technology, 2nd Edition, Amanda Andress and Mandy Andress, Sams,2004 (*available at: http://books.google.com/books?id=iENtTU3HZoMC&dq=%22web+site+defacement+attack+%22&source=gbs_navlinks_s*).

(4) "**Integerit file Verification System**", E.L.Cashin, 2000 (*available at http://integrit.sourceforge.net*).

(5) "**Veracity- nothing can change without you knowing: Data integrity assurance**", Rocksoft, 2003 (*available at http://www.rocksoft.com/veracity/*)

(6) "**Advanced Intrusion Detection Environment**", R.Lehti, 2005 (*available at http://www.cs.tut.fi/arammer/aide.html*).

(7) "**The MD2 Message Digest Algorithm**", RSA Laboratories, 1992.

(8) "**Advanced Applications of Tripwire for Servers**", Gene Kim, Tripwire,Inc, 2001.

(9) "**WHDRM**", Abdulkader A. Alfantookh, College of Computer and Information Sciences, King Saud University, Saudi Arabia, 2006 (*available at http://faculty.ksu.edu.sa/fantookh/Publications/Publications.aspx*).

(10) "**High Performance Issues in Web Search Engines**", Selvitri F., 2004.

(11) "**A Novel and Efficient Approach for Near Duplicate Page Detection in Web Crawling**", V.A.Narayana, P.Premchnd, IEEE International Advance Computing Conference (IACC2009), Patiala, India, 6-7 March 2009.

(12) "**Some Applications of Rabin's Fingerprinting Method**", A. Z. Broder, Springer-Verlag, 1993 *(available at http://en.wikipedia.org/wiki/Fingerprint_(computing))*.