I. J. Computer Network and Information Security, 2012, 6, 19-25
Published Online June 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2012.06.03

Toward Security Test Automation for Event Driven GUI Web Contents

Izzat Alsmadi[1] and Ahmed AlEroud[2]
[1]Department of Computer Information System, Yarmouk University, Irbid 21163, JORDAN
ialsmadi@yu.edu.jo
[2]Department of Information Systems, University of Maryland, Baltimore County (UMBC)
1000 Hilltop Circle, Baltimore, MD 21250, USA
ahmed21@umbc.edu

Abstract — The web is taking recently a large percentage of software products. The evolving nature of web applications put a serious challenge on testing, if we consider the dynamic nature of the current web. More precisely, testing both blocked contents and AJAX interfaces, might create new challenges in terms of test coverage and completeness. In this paper, we proposed enhancements and extensions of the current test automation activities. In the proposed framework, user interaction with AJAX interfaces is used to collect DOM violation states. A blocked content is accessed through multiple forms' submission with dynamic contents, and in each iteration the vulnerability events databases are modified. Next, the test cases database of possible vulnerable inputs for both AJAX and blocked contents is built. Finally, Coverage assessment is evaluated after executing those test cases based on several possible coverage aspects.

Index Terms — Web Testing, Test Automation, Security Testing, Coverage Metrics

I. Introduction

Testing in general and testing web applications in particular is a challenging and time consuming. It has been shown that most web applications are vulnerable, the reason is that most of attacks are occurred over HTTP protocol [1]. Securing web applications should be taken into account in all organization, and thereby the testing plan should be prepared in order to ensure a secured testing environment. Recently, there has been a wide concentration on web application development using dynamic techniques such as Asynchronous JavaScript and XML (AJAX) in which web applications are created to provide interactivity to users who find more flexibility in carrying out their task in the same manner in the traditional desktop applications.

One of the key advantages of AJAX technology is the possibility of running several mini web widget on the webpage at the same time. Such benefit leads to various useful applications, such as displaying the latest updates of the webpage in a timely manner, this style of development is most likely utilized in social networks and other emerging web applications. Though their advantage, AJAX application might consequence in security challenges, and therefore adding security burden on the web application testers, the reason is that such applications are event driven, accordingly, test automation might not be effortlessly performed. As such, developing a robust testing methodologies with appropriate metric as vital given the rapid growth in the number of event driven web applications. This paper presents a novel framework that can be utilized to automate security testing of event driven web applications. Most of the previous works focus on creating vulnerability scanners to prepare test cases, for instance in [2], a Web Application Vulnerability and Error Scanner was implemented, it consists of a black-box testing framework for Web application security assessment, the scanner was compared with other different tools and it was feasible to for traditional web applications security testing, such tools are nevertheless not effective in handling event driven vulnerabilities. The coverage metrics cannot be easily desgined for such applications. indeed, the presented framework might be effective in tackling dynamic web applications security based testing.

The rest of the paper is organized as follows. In the next section we provide an overview of the web application testing challenges.Section-II provides an overview of the challenges of testing web applications. The research goals and approaches have been discussed in section-III. The proposed test automation framework has been discussed in section IV. Conclusions have been drawn in section-V.

II. Challenges of Testing Web Applications

Compared to different applications, web applications have some specialists [3], The first challenge is the performance challenge. While performance is important in all types of software applications, it is rather vital in web applications. All web applications are distributed and there are several factors besides those related to local machine that may affect communication speed in web

applications. Examples of those factors include the Internet speed, line bandwidth, the number of concurrent users, website database and network readiness, etc. In relation to performance, stress or robustness testing is used to evaluate the website ability to handle requests under abnormal or exceptional situations.

Many other challenges can arise due to Continuous evolution and frequent changes in websites [4]. Due to requirements changes for websites, and the continuous growth of their size over time, such evolution causes extra overhead in testing as test cases and expected results may change whenever a website is evolved. In addition, pressure in deadlines of completing web applications, testing has extra pressure to be completed as early as possible. The applications' compatibility is another challenge in web testing, as the web is a wide open platform for the different kinds of applications. Developers use different programming languages, scripts, databases, etc. Compatibility can be a serious problem when different applications are communicating for data transfer or service request-consumption. We dedicated this section to discuss such challenges as follows:

Security and access challenges. In security, there are several related challenges. Websites are more vulnerable to hacking and illegal intrusions in comparison to applications in other environments. However, in this paper, we will focus on one aspect of security related challenges, access to web pages, HTML forms and other dynamic resources. Website pages are typically accessed through user names, passwords and sessions. Simulating those scenarios, especially through test automation tools is problematic. One example of security challenges is related to client-side input validation, where end users can bypass this validation, this might result security problems for Web applications, accordingly it might lead to unauthorized access to data. A new technique called bypass testing was proposed in [5] to create client-side tests for Web applications, a model was created to support more input validation testing, and rules were defined to bypass and input validation, the authors conclude that web application developers should check data on the server.

Testing effectiveness is usually evaluated through coverage. Coverage indicates the parts of the software or the website that are tested through those test cases which are relative to the overall software or website. For example, code coverage calculates the percentage of code visited by the test cases relative to the overall coverage. Following are examples of some of the coverage aspects or criteria that can be evaluated in web site testing. Path and branch coverage, this technique was used in [6] the work proposed web testing model for web application testing.

A page flow diagram was used to extract paths and then translated them into XML syntax which is then utilized as an input of test engine. A graph for path or branch coverage can be created from a website based on several possible aspects. A GUI graph is drawn from a website that represents the hierarchical relation between web pages.

Graph coverage can be also created from branches or decisions in the code behind website pages. Example of code branching keywords include: if, while, for, else, etc. They can be also drawn for user different choices or options when using or visiting the website. Function or service coverage. In each website, several services are provided; several requirements and functions are expected to be fulfilled.

Code coverage is also one of the main challenges in web testing; web applications are extremely large, making complete code coverage very difficult a study by the author of [7] has shown that only 50%-80% of web application code is covered even in good testing situations in testing web applications. A large amount of code exists behind web pages. Some code is automatically generated by the development tools while others are manually created. Statement coverage similar to code or Lines of Code (LOC) coverage, statement coverage calculates the number of statements covered by testing. Such coverage is usually used in applications for platforms rather than in web applications.

Resources or time coverage: Unlike website attributes based coverage, it is possible sometime to stop testing based on the consumed time or resources. In [8] it has been shown that 80% of the web applications usages tend to use only 20% of the system services.

Model-based testing of Web applications was proposed in [9]. The authors defined the coverage criteria (e.g. page and link coverage) regarding to navigational model. The link coverage deals with several links pointing to other web pages in the website or outside it. In Page coverage, Similar to links, pages is a major element in the structure of the website. Since a webpage is a container for other elements, testing a web page means usually testing all elements in that webpage. Besides links and pages, websites include several other types of elements that can be evaluated for coverage. This include: forms, frames, buttons, labels, textboxes, etc.

Figure 1. Typical Form page.

Deep web (also called hidden or invisible web) content can only be accessed via submitting HTML forms that retrieve information from these hidden documents. In[10] Raghavan and Garcia-molina described a deep web crawler to crawl deep web content. The crawler can perform customized forms' submission. Initially, the crawler has to build an internal representation when it receives the form page. Figure 1 shows a typical content of a form page.

III. GOALS AND APPROACHES

We will try to utilize some of the techniques used by search engines to access web pages with access restrictions to be used in websites testing. We have paid some attention also to Asynchronous JavaScript and XML (AJAX) interfaces in websites as such contents are not readily accessible by crawlers, some specials tools and techniques should be situation like this. Since AJAX-based applications depend on asynchronous client/server communication and client-side processing of the DOM tree, testing web applications with AJAX content is pretty challenging compared to traditional web content [11]. (AJAX) is getting more and more popular in web applications for building client side interactive web applications.

AJAX is a family of applications which utilizes other web related applications such as: HTML, DOM Cascading Style Sheets (CSS), and Java Script. The main advantage of AJAX is the speed of interaction regarding events carried by user.

The Ajax engine works within the Web browser through JavaScript and the DOM to render the Web application and handle any requests that the customer might have of the Web server. Ajax engine works within the Web browser to render the Web application and handle any requests that the customer might have of the Web server.

Through the web, there are too many web pages and HTML forms, as well as AJAX interfaces, that can't be accessed by search engines. Those pages are usually called Deep web. To be able to access the deep web, search engines use different techniques. A vertical search engine is used to crawl pages for a particular domain. This requires custom forms to work as mediators with specific information to access each page. Such approach can be very expensive, time consuming and labor intensive. Another approach that is usually called "surfacing" tries to utilize the existing infrastructure of search engines and generate extensions for deep web pages upon request.

Search engines may find different types of elements in web pages. Some elements need special methods for crawling. For example, unbounded fields (such as password and search textboxes) are text based fields that the user has to enter text in. The possible values are unbounded and therefore are more challenging to automatically generate. Some other fields require encrypted information or information transferred from previous pages. In the following we will discuss some security testing guidelines for web application.

A. To Expose Or Not To Expose

When testers have to test a website from security perspective, they are in a two conflicting requirements. In one hand, they need to expose most or all website elements or components for the purpose of testing them and their functionalities. On the other hand, to test them from security perspective, they need to make sure that such components are not accessed to externals except in certain constraints or pre-conditions.

In other words, if we were able to expose functionality for a supposedly blocked component, while such test may pass from functionality perspective, it fails from security perspective. Security testing in most cases goes against goals of typical testing, since for testing purpose, we want to expose all elements, while for security testing, we want to make sure that certain elements are not exposable. To summarize this paragraph, testing a security requirement can lead to one of those possible results:

- If the test case fails, it fails for one of two reasons. It either fails since the element under test was not successfully exposed, or since the results was not expected.
- If the test case passes, it may pass from a functionality perspective while fails from a security perspective. Table 1 summarizes test cases and possible results.

TABLE 1. SECURITY TESTING POSSIBLE RESULTS

SEQ	Web elements are open (no security preconditions)	Test results: Functionality is correct	Test results: web element are exposed	Final test results Test case pass ?
1	YES	YES	YES	YES
2		YES	NO	NO
3		NO	YES	NO
4		NO	NO	NO
5	NO	YES	YES	NO
6		YES	NO	YES
7		NO	YES	NO
8		NO	NO	NO

Table 1 show that out of 8 possible scenarios, only two scenarios should consider the test case as pass. Those are in sequences 1 and 6 in which functionality of the service under test passes and for security testing, if the component is exposable, then it should be exposed and vice versa.

The testing challenge however is in how to test an un-exposable component?!Since testing is usually accomplished internally testers have user privileges that help them access such components for constructive testing. For solely security testing, scenarios from 5 to 8 in Table 1 should all fail.

In all traditional testing methods, security testing is usually overlooked and it is assumed that components are open for testing. If not, such components are either modified to open (temporary for testing) or simply ignored. For example an external testing engine that is testing the class customer in Figure 2, will not be able to expose private attributes and methods.


```
public class Customer {
    private int m_id = -1;
    private int ABC()
    {
        return 6;
    }
    public void SetID(int id)
    {
        m_id = id;
    }
    private string name = string.Empty;
    public string GetName()
    {
        return m_name;
    }
    public void SetName(string name)
    {
        m_name = name;
    }
}
```

Figure2: Code security testing example

B. *Automatic Logging And Captcha*

Many websites try to avoid robots or automatic logging through different techniques. Using "CAPTCHA" is one of the widely used techniques in this regard. CAPTCHA is a randomly generated picture of letters and numbers that are partially distorted. The goal of that image is to allow only humans – not robots – to be able to know and type those letters and numbers. Other websites try to block robots using timer, or session information. Users who frequently and continuously try to open one or more session to the website are tracked and blocked on that basis. For example, Google search engine will track users who submit too many queries in a small amount of time through several techniques. They used a combination of CAPTCHA, session or temporary blocking, etc.

The goal of those methods is to block users who are trying to flood the website with too many calls. Such users can do this for destructive purposes: e.g. Denial of Service (DOS) attack or they may do it through robots to collect information or build a dataset. In all cases, each website is interested to service as many users as possible and block single or a limited number of users to occupy the whole bandwidth of the website and block it from servicing other users. As such, some websites control or regulate bandwidth usage for insiders and outsiders.

C. *SSL And Secured Pages*

Websites who offer e-business services, emails, accounts for any purposes, should protect those accounts and protect those pages with extra security elements. The major element is an encrypted Security Socket Layer (SSL) that allows encrypted transmission of information between clients and web server.

Those secured web pages require extra levels of security testing. The first quick test is to make sure that the SSL is working and that the digital certificate accompanied is current, valid and working. Network and vulnerability tests can be implemented to make sure that information can't be hijacked after sending it from the client.

There are several types of attacks related to secured logins such as: session hijacking, tampering, DNS poisoning, etc. Security testers should be aware of those types of tests. In those scenarios testing can be divided into two major classes:

The first class where the user has insider knowledge and will try to use it to break or tamper sessions.

Zero knowledge attacks are intended to simulate external users who have no previous knowledge and who will try to break through the website.

D. *Logging Testing*

Websites log different types of activities for different purposes. Logs can be useful for marketing purposes to evaluate the nature of visitors, time of visiting, website areas that users usually visit, etc. Logs can be also important for maintenance and testing purposes looking for any possible abnormal event or sequence of events. For security, logs can be important for several reasons. Logs can include all information need to investigate a successful intrusion.

Logs themselves can be also attacked or tampered by some types of attacks (e.g. to cover attacks). Log testing should include scanning the logs to make sure that there are no gaps or inappropriate data. Since logs are large in size, special tools are used to scan through them looking for specific query or information.

E. *Other Types Of Security Testing*

Security testing is large and can include several other types or areas to test in a website. Examples of those other types of security testing include: code, operating system, network, access security testing, etc.

Code testing includes techniques to make sure that written code is not vulnerable and do not allow intruders to expose private elements in the website. Network security testing include testing the data, packet, ports, and hardware level elements to make sure only specific ports in the network are open and those ports are only used for normal data. Inward and outward packets are also screened for any abnormal traffic. Passwords and logins are tested and roles are enforced for periodic modifications.

Other challenges that crawlers may face is the issue of dealing with the huge amount of data. A large percent of this data is old (i.e. same as that crawled in previous cycle). It is estimated that 8 % of web pages are new in every week period. In order to improve speed, differential crawling is necessary to be able to crawl only new data. However, it will be further challenging for the crawler to differentiate new from old pages without through analysis.

Some websites include useful information for this purpose (i.e. last modified date). Differentiation can be in batch or incremental mode. In the batch mode that is used for small websites, the crawler starts every indexing stage with an empty storage. In incremental mode, the crawler updates rather than erase the old storage [12].

Web contents vary in nature between text, images, video where each type needs special methods to handle or crawl. Unlike the data in the database that is structured, the data in the web is unstructured and hence structured queries are not effective for mass information retrieval. Based on a free text query in the web, the definition of relevant results is vague and open for discussion.

Exact same query may retrieve different results in different search engines. In some cases, different results in the same search engine in different times. Relevancy between a query and a document depends on user and time.

IV. THE TEST AUTOMATION FRAMEWORK

Figure 3 shows a high level proposed framework for testing websites (taking into consideration to access blocked contents). Besides the generic elements, three elements are inspired from crawlers. Those are the form page in the forward link and the response page and analysis in the feedback link.

In addition, the framework takes into account, Ajax interfaces so we have added three new components to record and store all user interactions with AJAX contents, recording all Ajax interactions as an event driven approach.

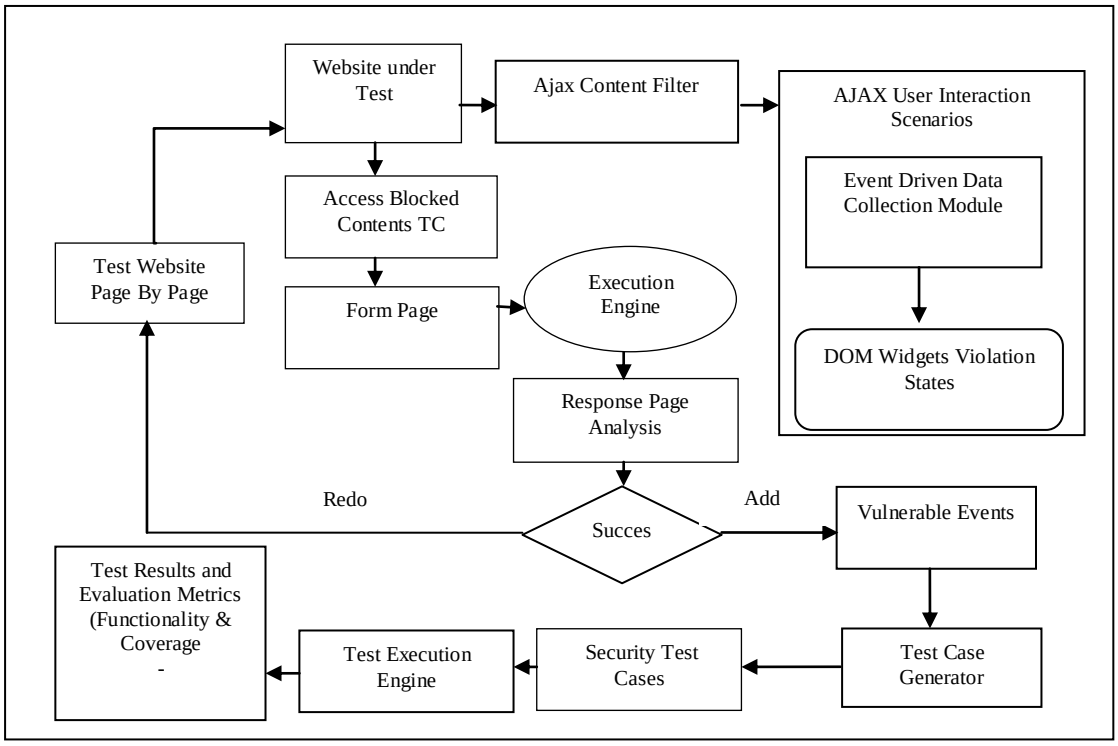

Figure 3: Security Test Automation Framework

Ambiguity in natural language can also cause problem for defining relevant documents especially for synonyms words (e.g. car, and vehicle).Spam content is another major challenge or problem in information retrieval and search engines where irrelevant information is injected in relevant ones (usually for marketing purposes).

While such problem may not be applicable to typical website testing, hence we can use some of the techniques for spam detection to test specific aspects of a website as spam detection techniques tried to distinguish normal from spam data based on normal data typical characteristics.

Some websites use techniques to raise their ranking and be more visible for search engines in techniques usually called Search Engine Optimization (SEO) Coverage in search engine crawling is measured based on the number of pages crawled and indexed by the search engine.

Indeed, we have added a protocol driven data collection module, to parse some vulnerability related contents extracted from user interaction with web AJAX web forms, similarly, in the proposed framework, a protocol crawler is used to crawl other possible blocked web page components.

A. *Blocked Documents and Contents (Non-Ajax Content)*

The goal of the form page is to prepare a typical initial form page (as that shown in Figure 1). Such form can be dynamically changed in next loops or cycles based on the analysis from the response page (i.e. successful or failed trial to access blocked documents or contents). Accessing blocked contents will be used to collect some possible vulnerable inputs, in that case, a new record will be added to the vulnerable events that may causes some security problems.

In crawlers, a serious challenge is to determine which form inputs to choose and finding applicable values to fill them with. This should not be the case for testing own website where the testers have insider knowledge of the

website and may also have certain user privilege to use for testing. Such information can be used to define the default form page. An example of a problem that both crawlers and web test automation tools has to deal with is the fact that many services in the web are performed based on interactive dialogue with the user and is dependent on user response.

In a previous test automation tool built by the first author of [13], a full test automation framework is proposed for testing software products in Windows environment. In similar dialogue scenarios, we simulate default or typical scenarios that may work in many cases.

However, some other problems may arise such as time synchronization and the ability to simulate all types of user actions and responses. Websites are updated asynchronously which means that only small parts of the page are updated which is referred to as delta-communication. It is not trivial to just retrieve these changes because often these delta updates become meaningful not until being injected to the DOM on the client-side.

B. Event Driven Dom Vulnerability Testing

The Approach of examining AJAX pages is a slightly bit similar to that used in [11]. There are three major differences between our test automation framework and the one proposed in [11], first we extracted DOM widgets vulnerable states based on different user interaction scenarios, second, our concentration in this framework is not only AJAX interfaces we are building test cases for both possible blocked contents and Dynamic Ajax interfaces, while in [11] the focus was on Ajax interface only. Our test Automation framework included a dynamic cycles to feed new inputs for those blocked content, eventually, the goal is to have as many security testing scenarios as possible.

In our proposed framework, many user interaction clickable events will be used as a collaboration scenario at the client level, hence any possible malicious change on the DOM (Document object Model) will be calculated based on DOM stable boundary [11], then a decision is made to consider event as malicious or not. Next a set of malicious events will be added to the vulnerable events database, ultimately, the complete test cases list will be composed of both vulnerable events of DOM AJAX interfaces and the blocked web contents. Security test is finally executed with such test cases and test coverage on both functionality and statement may be executed.

V. CONCLUSION

In this paper, challenges and problems of current web security testing are discussed. We believe that security testing activities are largely avoided. Security testing does not mean only to scan for website or network vulnerabilities. It has several other important elements. This paper includes an example of several areas to test in security in which test automation can be implemented. However, automation and security comes in many cases in conflict specially as many intrusions are implemented through automation and hence testing for security should verify in many cases that the website disables automatic or robotic users. We proposed a high level framework for testing websites taking into consideration to access blocked contents). Besides the generic elements, three elements are inspired from crawlers. Those are the form page in the forward link and the response page analysis in the feedback link

REFERENCES

[1] A. Danny ,"Web Application Security: Automated Scanning Versus Manual Penetration Testing". Web application security White paper, January, 2008.

[2] W. Yao, T .Chung, L .Tsung, "A Testing Framework for Web Application Security Assessment", Journal of Computer Networks, vol. 48, PP (739–761), 2005

[3] C.Kallepalli, and J.Tian, "Measuring and Modeling Usage and Reliability for Statistical Web Testing", IEEE Trans Software Engineering, 27(11): PP (1023-1034), 2001

[4] L. Xu, B. Xu, and Z. Chen, "A Scheme of Web Testing Approach", Journal of Najing in Chinese, 38(11): PP (182-186), 2002.

[5] J. Offutt, Y. Wu, X. Du and H. Huang,"Bypass Testing of Web Applications", In Proc. of the 15th International Symposium on Software Reliability Engineering (ISSRE'04), Saint-Malo, Bretagne, France, 2004.

[6] Z. Qia, M. Ko, and H. Zen, "A Practical Web Testing Model for Web Application Testing", in the 3d International IEEE Conference on Signal-Image Technologies and Internet-Based System, Bali, Indonesia, (2008), PP(434-441)

[7] B. Marín, Tanja, G. Giachetti, A. Baars, , "Towards Testing Future Web Applications", Fifth International conference in Research Challenges in Information Science (RCIS), Valencia, Spain, 2011, PP (1 – 12).

[8] X. Luo, F. Ping, and M. Hwa, "Clustering and Tailoring User Session Data for Testing Web Applications", International Conference on Software Testing Verification and Validation, Berlin, 2009, PP(336 – 345)

[9] F. Ricca and P. Tonella. "Analysis and Testing Of Web Applications". In Proc. of ICSE 2001, International Conference on Software Engineering, Toronto, Ontario, Canada, May, 2001, 12-19, pages 25–34.

[10] S. Raghavan, And H. Garcia-molina, "Crawling the Hidden Web". In Proc. of the 27th VLDB Conference, Roma, Italy, 2001, PP (129–138).

[11] A. Mesbah, A. Deursen and D. Roest," Invariant-Based Automated Testing of Modern Web Applications". IEEE Transactions on Software Engineering, vol. 40, no. 8, 2012.

[12] H. Alessandro, and P. Tonella, "Improving Web Application Testing Using Testability Measures", 11th IEEE International Symposium on Web Systems

Evolution (WSE), Issue Date: 25-26 Sept. 2009, PP (49 - 58).
[13] I. Alsmadi, and K. Magel, "An Object Oriented Framework for User Interface Test Automation". Midwest Instruction and Computing Symposium, ND, USA 2007.

Izzat Mahmoud Alsmadi is an assistant professor in the department of computer information systems at Yarmouk University in Jordan. He obtained his Ph.D degree in software engineering from NDSU (USA). His second master in software engineering from NDSU (USA) and his first master in CIS from University of Phoenix (USA). He had B.sc degree in telecommunication engineering from Mutah university in Jordan. He has several published books, journals and conference articles largely in software engineering and information retrieval fields.

Ahmed Aleroud is a doctoral student in the department of information systems at the university of Maryland, Baltimore County, USA. He obtained his master degree in computer information systems from Yarmouk University Jordan. He had a B.sc degree in software engineering from Hashemite University in Jordan. His research interests are primarily in information security field

