MALURLS: A Lightweight Malicious Website Classification Based on URL Features

Monther Aldwairi
Department of Network Engineering and Security, Jordan University of Science and Technology, Irbid, Jordan
munzer@just.edu.jo

Rami Alsalman
Department of Computer Engineering, Jordan University of Science and Technology, Irbid, Jordan
rsalsalman08@cit.just.edu.jo

***Abstract*—Surfing the World Wide Web (WWW) is becoming a dangerous everyday task with the Web becoming rich in all sorts of attacks. Websites are a major source of many scams, phishing attacks, identity theft, SPAM commerce and malwares. However, browsers, blacklists and popup blockers are not enough to protect users. That requires fast and accurate systems with the ability to detect new malicious content. We propose a lightweight system to detect malicious websites online based on URL lexical and host features and call it MALURLs. The system relies on Naïve Bayes classifier as a probabilistic model to detect if the target website is a malicious or benign. It introduces new features and employs self learning using Genetic Algorithm to improve the classification speed and precision. A small dataset is collected and expanded through GA mutations to learn the system over short time and with low memory usage. A completely independent testing dataset is automatically gathered and verified using different trusted web sources. They algorithm achieves an average precision of 87%.**

***Index Terms*— malicious websites, machine learning, genetic algorithm, classification**

I. Introduction

Internet access is an integral part of the modern life and employees in today's fast economy depend on Internet connected smart phones, laptops and personal assistants to perform their jobs on the go. Even regular Joe and young children are becoming techsavvy and cannot live without Internet. Users shop, check movies, bank accounts, email, health insurance, they renew driving licenses, pay bills, make calls over IP, chat and play games, just to name a few daily activities. Such Internet access takes place through web browsers making them the most popular application for most users.

As browser vendors race to introduce more features and new functionalities more vulnerabilities arise and more personal data are put at risk. Browsers have become the main system entry point for many attacks that aims at stealing private data and manipulating users to reveal sensitive information. Unsuspecting web surfers are not aware of the many drive-by-downloads of malwares, ad wares, spywares and Trojans to their devices. Just a single visit to a shady website is sufficient to allow the intruder to detect vulnerabilities in the surfer's computer and inject a malware that might enable the intruder to gain remote access or open a backdoor for future blunders.

Users do not have to visit pornographic or hacker websites to get compromised. Commerce related SPAM such as pharmaceuticals and fake products are one way to coerce users to click and access malicious websites. In addition, they can be redirected to such websites through more organized schemes such as fast flux networks (FFN) [1]. Consequently, users are easily tricked to reveal private information using phishing and pharming attacks [2]. In addition to all of that, browsers collect sensitive data such as favorites, cache files, history file, cookies, form data and passwords. This puts such information at risk and keeping your browser and computer up to date will not cut it. For instance, cache timer sniffing enables intruders to determine websites you have visited.

Finding and identifying such websites is no simple task due to the ever growing World Wide Web and the dynamic nature of malicious websites. Blacklisting services rose to the challenge and were encapsulated into browsers, toolbars and search engines. The lists are constructed through manual reporting, honeypots or web spiders. But blacklists grow uncontrollably and become a performance bottleneck. Incorrect listing is a major problem, due to reporting, analysis and record keeping mistakes. Therefore, legitimate websites may be incorrectly evaluated and listed, while malicious websites are not listed because they are new and haven't been analyzed yet. Researchers have been very active in devising online and offline solutions to classify malicious websites and make web surfing safer. Shortly we give a brief survey of the current state of art techniques for classifying websites.

In this paper we propose lightweight statistical self-learning scheme to classify websites based on their features. It is fast and designed to run online to protect the users. We use a Naïve Bayes classifier to classify the websites into two classes: malicious or benign. The number of features used is small and they fall under one of three categories: lexical, host-based or special features. Features include those suggested by McGrath et al. [3]

and Ma et al. [4]. We add special features to improve the classification accuracy such as JavaScript Enable/Disable, Document Frequency, and Title Tag. In addition, Genetic Algorithm is used to expand the training dataset through mutations to learn the Naïve Bayes classifier better and faster without the need to deal with huge datasets. The authors presented preliminary results in a previous paper [5] and in this paper they expand the original work by including more implementation details and adding additional results.

The rest of the paper is organized as follows. Section II surveys the related work. Section III details the methodology followed to classify websites, this includes the list features, collecting the training and testing datasets and how the Naïve Bayes and GA are used. Section IV presents the experimental results.

II. Related Work

Blacklisting was and still is a popular technique. Whittaker et al. [6] offline analyzed millions of pages daily from the noisy Google's phishing blacklist. Their main contribution was achieving 90% classification accuracy for phishing pages after a three weeks training. PhishNet [7] used approximate pattern matching algorithm to match URL components against blacklist entries. Though the above techniques tried to automatically manage blacklists and increase their accuracy, they are still insufficient and suffer from their growing size and incorrect listing. Blacklists can be combined with other techniques that uses machine learning to classify malicious websites.

One of the earliest classification systems for malicious websites was concerned with the detection of SPAM in blog posts. Blog identification and splog detection by Kolari et al. [8] used the activity and comments generated by a blog post as the main classification feature in addition to ping update services. Support Vector Machines (SVM) was used with only linear kernel in all experiments and reported moderate results. Subsequent work focused on detecting Phishing URLs in SPAM emails. Garera et al. [9] main contribution was identifying eighteen features to detect phishing URL embedded in SPAM. They used linear regression compare millions of Google's toolbar URLs to identify 777 phishing pages a day and 9% of the users that visit them are potential victims. McGrath et al. [3] studied phishing infrastructure and the anatomy of phishing URLs. They pointed out the importance of features such as the URL length, linked-to domains age, number of links in e-mails and the number of dots in the URL. PhishDef [10] used features that resist obfuscation and suggested used the AROW algorithm to achieve higher accuracy.

To further increase the accuracy several approaches focused on page content statistics such as Seifert et al. [11]. They added features derived from JavaScript and HTML tags such as redirects, long script code lines and shell code. Seifert et al. used features from the page contents such as the number of HTML script tags and size of *iframe* tags. Cova [12] et al. went too far by profiling the normal JavaScript behavior and applying anomaly detection which is prone to high false positives. Anomaly detecting works by extracting features during the normal learning phase based on a specific model. In the testing phase the new feature values for the websites to be tested are checked against the training models representing the normal behavior. The features used include: the number of code executions, code length, number of bytes, shell codes and the difference in returned pages for different browsers and the number of redirections. The Prophiler by Canali [13] used HTML tag counts, percentage of the JavaScript code in the page, percentage of whitespace, entropy of the script, entropy of the of the strings declared, number of embed tags, presence of meta refresh tags, the number of elements whose source is on an external domain and the number of characters in the page. While improving accuracy the Prophiler significantly increased the number of features to eighty eight. In addition to the increased overhead due to the statistically processing the page content, those techniques suffered from the inherent danger of having to access the malicious page and download the content before deciding it was malicious.

Ma et al. [4] used Yahoo−PhishTank dataset and validated their work using three machine learning models Naïve Bayes, SVM with an RBF kernel and regularized logistic regression. Later, Ma et al. [14] [15] developed a light weight algorithm for website classification based on lexical and host-based features while excluding page properties. It was designed as real-time, low-cost and fast alternatives for black listing. They reported 3.5% error rates and 10–15% false negatives, but the tradeoff was between memory usage and accuracy. However, the main disadvantage was the fact that they use tens and hundreds of thousands of features to achieve their results. Another disadvantage shared among all previous approaches is collecting and handling a large number of websites and features which makes it hard to run them online.

III. Mthodology

To address the drawback of previous wok we need to identify malicious websites. We define a malicious webpage as a page that downloads a file, uploads a file, collects data, installs an application, opens a pop window(s), displays an advertisement or any combination of the above without the knowledge or consent of the user. We manually construct our own dataset and label websites as either benign or malicious based on trusted web directories. The complete MALURLs framework is shown in Fig. 1. In step 3 the features for the training dataset are calculated through various sources. We collect 100 benign and 100 malicious sites. In steps 5 and 6 the dataset is expanded to 10000 records using Genetic Algorithm (GA). GA self learns the classifier through mutations on the dataset. Based on a fitness function we can use mutations and crossovers from the current dataset to generate a larger dataset and grantee not to learn our classifier based on specific domain. In step 7 Naïve Bayes are trained using part of the collected features. Finally, a completely different dataset of 200 websites is

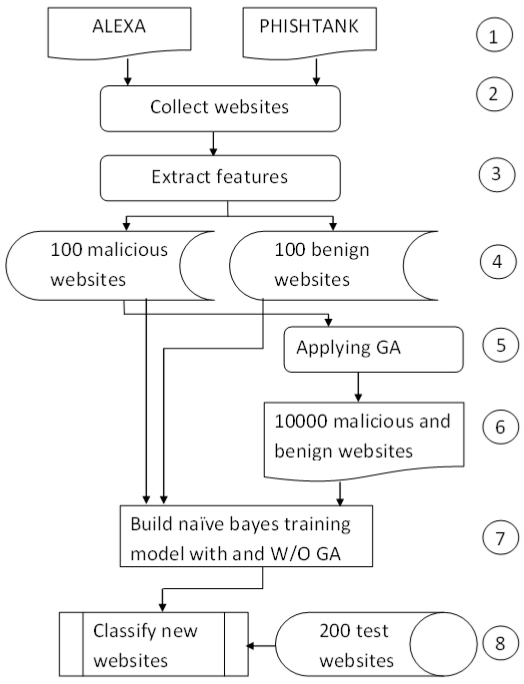


Figure 1. MALURLs framework.

used for testing. The testing dataset is collected and classified automatically into benign and malicious. The features are calculated based on a same web sources used in training. It is worth mentioning that the testing and training datasets are completely independent and are verified and classified using different trusted web sources to eliminate any chance of data poisoning. Unlike most of previous approaches which used the same source, mainly PhishTank.

The following Subsections explain the three feature groups and the basis for their selection, how the training dataset is collected, how the Naïve Bayes classifier works, the Genetic Algorithm and finally how the testing dataset is built.

A. Features

Features used fall into one of three categories: lexical, host-based features and special features.

1) Lexical Features

URL stands for uniform resource locator or formerly the universal resource locator. URL and uniform resource identifier (URI) are equivalent and identify any document retrieved over the WWW. The URL has three main parts: the protocol, hostname and path. Consider the following URL for example: "http://www.just.edu.jo/~munzer/Courses/INCS741/Lec/ch3.ppt". The protocol is: http://", the hostname is: "www.just.edu.jo" and the path is: "~munzer/Courses/ INCS741/Lec/ch3.ppt".

Lexical features are the properties of the URL itself and do not include content of the page it points to. The URL properties include the length of the top level domain (TLD), other domains, the hostname, URL length, as well as the number of dots in the URL. In addition, lexical features include each token in the hostname (delimited by '.') and tokens in the path URL delimited by '/', '?', '+', ' – ', '%', '&', '.', '=', and '_'. Those feature groups are known as a "bag-of-words". The features above tell a lot about a webpage. The domain might indicate a blacklisted malicious content provider. A large number of NULL token probably attributed to too many slashes might indicate an http denial of service attack (DOS) on Microsoft Internet Information Server (IIS).

2) Host-based Features

Host-based features are derived from the host properties such as the IP address, geographic properties, domain name properties, DNS time to live (TTL), DNS A, DNS PTR and DNS MX records as well as WHOIS information and dates. Those features are very important and can help any classifier better the detection process. They help address a lot of important questions such as: does the IP address belong to a geographical location associated with malicious content? Does the PTR record resolve an IP that belongs to the host? Do the IP addresses of the DNS records belong to one autonomous system (AS)?

3) Special Features

Those are new features that do not fit under the aforementioned categories or reported good results with previous systems. Some are simple to get a value for such as JS Enable/Disable, HTML Title tag content (<title></title>), 3-4-5 grams (n-grams) and Term Frequency and Inverse Document Frequency (TF-IDF). JavaScript code usually is downloaded and run on the client's browser which can be very dangerous. Term frequency is the number of times a term occurs in a document. The inverse document frequency is the logarithm of the number of documents divided by the number of documents containing the term and it measures the importance of a term. TF-IDF is commonly used in search engines, classification and data mining and finally 3-4-5 grams take longer to calculate than the other features.

Other features require significant computation time such as Anchors or bag-of-anchors which are extracted from all URLs in Anchor tags on the page being examined. Table I below shows a list of features' groups used for training and testing purposes.

B. Training Dataset

The dataset is composed to 200 websites, half benign and the other half is malicious. The websites are chosen randomly and ALEXA Web Information Company's

TABLE I.
FEATURE GROUPS

Features		
JS-Enable-Disable	DNS PTR record	Path tokens
Document Frequency DF	WHOIS info	Last token of the path
Title tag <title>??</title>	Connection speed	Spamassassin plugin
3-4-5 grams	TLD + domain	TLD
TF-IDF weighting	DNS A record	DNS TTL
Blacklists	Geographic	DNS MX record
WHOIS dates	Hostname	Bag-of-words
IP address misc	Words+URLs	URLs
Lexical misc	Meta+link	Anchors
4grams	URLs+anchors	Meta tags
URLs+anchors+meta		

website [16] is used to determine benign websites while PhishTank dataset [17] is used to determine malicious websites. ALEXA is one of the most influential and trusted WWW information companies. It provides information about websites including Internet traffic and top sites rankings. MALURLs uses IP tracer website [18] to extract the URL, DNS, IP address and the geographic properties for benign websites.

PhishTank is an open source anti-phishing website that is widely used by almost all major browsers and WWW vendors such as Mozilla, Yahoo and McAfee. It offers phish pages verification tool through a voting system and issues annual phishing reports. PhishTank dataset contains partial information about malicious websites such as URL, DNS, IP-address and the geographic information. However, not all features are available on PhishTank, particularly the new special features. Therefore, we use Sphider [19] which is an open source web spider and a search engine. Sphider performs full text indexing, for example it finds links anywhere in a document whether in href or even in JavaScript strings. Sphider is able to calculate the Term Frequency (TF), Document Frequency (DF) and Inverse Document Frequency (IDF) features.

C. Naïve Bayes classifier

Bayes [20] is a probabilistic model based on Bayesian theorem. Though it is simple but often outperforms most of the other classifiers especially if trained using supervised learning methods. Naïve Bayes classifiers assume that the effect on a class from a feature is independent of the values of other features. This conditional independence simplifies the computation without sacrificing the accuracy. This makes Naïve Bayes a perfect match for our lightweight online algorithm. Bayesian theorem for example, calculates the probability that a website is malicious from the independent probabilities that a website is from a geographic location that generates fake traffic, has a random and very long hostname, has an IP address that does not match the DNS A recor and so on.

In our case the number of features and their values range are large. If *C* represents the class and *F* represents a feature then the conditional probability (P_r) of *C* given *F* is calculated according to (1).

$$P_r(C|F) = \frac{P_r(F|C)P_r(C)}{P_r(F)} \quad (1)$$

D. Genetic Algorithm

To generate a larger dataset from the initial dataset we use Genetic Algorithm. GA is a biologically inspired algorithm that applies the principles of evolution and natural selection. The algorithms starts with an initial population encoded as a chromosome structure which is composed of genes encoded as numbers or characters. In our case the initial population represents the group of features for the training dataset. Chromosome goodness is evaluated using a fitness function that uses mutations and crossovers to simulate the mutation of species. The fittest chromosomes are selected and the process is repeated till we converge to a solution [21].

The initial population is the dataset collected as specified in Subsection B and used in the learning phase. Mutations are applied on the initial dataset which is composed of a number of features or genes. Mutations are simply applying changes to certain features such as changing JS-Enable-Disabled from True to False (binary encoding) or to add a random amount between 0.2-0.3 to DF and TF (value encoding).
In the testing step the fitness function is calculated by multiplying the probability values for the all features as shown in (2). The fitness function is used to calculate malicious and benign probabilities. The website is classified based on the highest probability.

$$f(x) = \prod \begin{array}{c}(DF, JS_{Enable}, Title_{tag}, TF-IDF, IP,\\ Geographic, Path_{tokens}, DNS,\\ Blacklists, Meta, Malicious)\end{array} \quad (2)$$

E. Testing Dataset

For testing we collect 200 URLs (100 malicious, 100 benign) using WOT Mozilla Plug-in [22]. The features values are collected the same way as in the training dataset. WOT is a traffic-light style rating system where green means (benign) and red means stop (malicious). The rating of a website depends on a combination of user ratings and data from trusted sources such as Malware Patrol [23], Panda [24], PhishTank, TRUSTe [25], hpHosts [26] and SpamCop [27]. In addition, WOT enables users to evaluate the trustworthiness of a website and incorporate their ratings in calculating the reputation of a website.

IV. IMPLEMENTATION AND RESULTS

We implement MALURLs using PHP programming language and MySQL database. Equation (3) defines the precision metric used to evaluate the relative accuracy of MALURLs using different features. Precision (*P*) is a measure of the usefulness of the retrieved documents.

$$P = \frac{Number\ of\ websites\ classified\ correctly}{Total\ number\ of\ websites} \quad (3)$$

The testing dataset of 200 instances was divided into five different subsets and the average precision with and without Genetic Algorithm was calculated as shown in Table II. The use of GA to expand the training dataset results in a significant improvement in the classification precision. The average precision for classifying a website as benign or malicious using GA is 87% when using all feature groups.

We run experiments to measure the improvement in

TABLE II.
OVERALL PRECISION WITH AND WITHOUT GA

Dataset Number	**Precision (%)**	
	Without GA	***With GA***
1	70	85
2	75	90
3	80	95
4	80	80
5	65	85
Average	74	87

TABLE III.
OVERALL PRECISION WITH AND WITHOUT TF-IDF

Dataset Number	Precision using GA (%)	
	Without JS	*With JS*
1	60	70
2	65	60
3	90	95
4	55	80
5	60	75
Average	66	76

TABLE IV.
PRECISION WITH AND WITHOUT JS-ENABLE-DISABLE

Dataset Number	Precision using GA (%)	
	Without JS	*With JS*
1	55	65
2	50	60
3	75	75
4	70	65
5	65	80
Average	63	69

TABLE V.
PRECISION WITH AND WITHOUT 3-4-5 GRAMS

Dataset Number	Precision using GA (%)	
	Without n-grams	*With n-grams*
1	80	80
2	70	70
3	85	85
4	60	75
5	90	90
Average	77	80

precision attained by adding the new individual features to MALURLs. The features added include TF-IDF, JS-Enable-Disable and 3-4-5 grams. The addition of TF-IDF results in a significant increase in classification precision from 66% to 76% as shown in Table III. This is expected because of the volume of information presented by this feature. Adding the JS-Enable-Disable did show a very good increase in average precision from 63% to 69% as illustrated by Table IV. The experiments to measure the improvement in MALURLs precision achieved by adding 3-4-5 grams show a small increase in average precision from 77% to 80% as illustrated by Table V. 3-4-5 grams calculation is complex, takes a long time and puts the user at risk due to the need to download the document. Therefore n-grams can be deemed irrelevant because of the high overhead and small improvement which is consistent with our goal of keeping the algorithm light weight.

V. CONCLUSIONS

This paper presents a new website classification system based on URL, host-based and special feature. We experiment with various features and determine the ones that improve the precision with minimum overhead. The data is collected using WOT Mozilla plug-in and the features are calculated using various web resources. MALURLs system reduces the training time using GA to expand the training dataset and learn the Naïve Bayes classifier. The experimental results show the average system precision of 87%. The additional features proved valuable to improve the overall classification precision. TF-IDF improved precision by up to 10%, JS-Enable-Disable improvement was about 6% while 3-4-5 grams improvement was limited to 3%.

REFERENCES

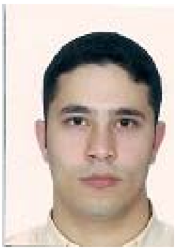
[1] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke and G. Eaton, “Real-time Detection and Classification of Fast Flux Service Networks”, In Proceedings of the Cybersecurity Applications and Technology Conference for Homeland Security (CATCH), Washington, DC, Mar 2009.
[2] J. Zdziarski, W. Yang, P .Judge, “Approaches to Phishing Identification using Match and Probabilistic Digital Fingerprinting Techniques”, In Proceedings of the MIT Spam Conference, 2006.
[3] D. McGrath and M. Gupta, “Behind Phishing: An Examination of Phisher Modi Operandi”, In Proceedings of the USENIX Workshop on Large-scale Exploits and Emergent Threats (LEET), San Fransicso, CA, Apr 2008.
[4] J. Ma, L. Saul, S. Savage, and G. Voelker, “Beyond Blacklists: Learning to Detect Malicious Websites from Suspicious URLs”, In Proceedings of the ACM SIGKDD Conference, Paris, France, Jun 2009.
[5] M. Aldwairi, R. Alsalman. “MALURLs: Malicious URLs Classification System”. In Proceedings of the Annual International Conference on Information Theory and Applications (ITA), Singapore, Feb 2011.
[6] C. Whittaker, B. Ryner, and M. Nazif, “Large-Scale Automatic Classification of Phishing Pages”, In Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS’10), San Diego, CA, Mar 2010.
[7] P. Prakash, M. Kumar, R. R. Kompella and M. Gupta, “PhishNet: Predictive Blacklisting to Detect Phishing Attacks”, In Proceedings of INFOCOM ’10, San Diego, California, Mar 2010.
[8] P. Kolari, T. Finin, and A. Joshi, “SVMs for the Blogosphere: Blog Identification and Splog Detection”, In AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs, Mar 2006.
[9] S. Garera, N. Provos, M. Chew, and A. D. Rubin, “A Framework for Detection and Measurement of Phishing Attacks”, In Proceedings of the ACM Workshop on Rapid Malcode (WORM), Alexandria, VA, Nov 2007.
[10] A. Le, A. Markopoulou and M. Faloutsos, “PhishDef: URL Names Say It All”, In Proceedings of the 30th IEEE INFOCOM 2011 (Mini Conference), Shanghai, China, April 10-15, 2011.
[11] C. Seifert, I. Welch and P. Komisarczuk, “Identification of Malicious Web Pages with Static Heuristics”, In Proceedings of the Australasian Telecommunication Networks and Applications Conference (ATNAC), 2008.
[12] M. Cova, C. Kruegel and G. Vigna, "Detection and Analysis of drive-by-Download Attacks and Malicious JavaScript Code", In Proceedings of the 19th International Conference on World Wide Web (WWW'10), Raleigh, NC, Apr 2010.
[13] D. Canali, M. Cova, G. Vigna and C. Kruegel, “Prophiler: a Fast Filter for the Large-Scale Detection of Malicious Web Pages", In Proceedings of the 20th International World Wide Web Conference (WWW), Hyderabad, India, Mar 2011.
[14] J. Ma, L. Saul, S. Savage, and G. Voelker, “Identifying Suspicious URLs: An Application of Large-Scale Online Learning”, In Proceedings of the International Conference

on Machine Learning (ICML), Montreal, Quebec, Jun 2009.
[15] J. Ma, L. Saul, S. Savage, and G. Voelker, "Learning to Detect Malicious URLs", ACM Transactions on Intelligent Systems, vol. 2, no. 3, Apr 2011.
[16] ALEXA Web Information Company's website, http://www.alexa.com/, last access Feb 2011.
[17] PhishTank, http://www.phishtank.com/, last access Feb 2011.
[18] IP Tracer Website. http://www.ip-adress.com/ip_tracer/, last access Feb 2011.
[19] Sphider, http://www.sphider.eu/, last access Feb 2011
[20] G. John, G. and P. Langley, Estimating continuous distributions in Bayesian classifiers, In proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (pp. 338–345), San Francisco, CA, 1995.
[21] J. Koza, M. Keane, M. Streeter, W. Mydlowec, J. Yu and G. Lanza "Genetic Programming IV: Routine Human-Competitive Machine Intelligence", Kluwer Academic Publishers, 2003.
[22] Web of Trust Mozilla Plug-in, http://www.mywot.com/en/download, last access Feb 2011.
[23] Malware Patrol. http://www.malware.com.br/, last access Feb 2011.
[24] Panda WOT. http://www.pandasecurity.com/, last access Feb 2011.
[25] TRUSTe – WOT. http://www.truste.org/, last access Feb 2011.
[26] hpHosts online. http://www.hosts-file.net/, last access Feb 2011.
[27] SpamCop. http://www.spamcop.net/, last access Feb 2011.

Monther Aldwairi was born in Irbid, Jordan in 1976. Aldwairi received a B.S. in electrical engineering from Jordan University of Science and University (JUST) in 1998, and his M.S. and PhD in computer engineering from North Carolina State University (NCSU), Raleigh, NC in 2001 and 2006, respectively.

He is an Assistant Professor of Network Engineering and Security Department at Jordan University of Science and Technology, where he has been since 2007. He is also the Vice Dean of Faculty of Computer and Information Technology since 2010 and was the Assistant Dean for Student Affairs in 2009. In addition, he is an Adjunct Professor at New York Institute of Technology (NYiT) since 2009. He worked as Post-Doctoral Research Associate in 2007 and as a research assistant at NCSU from 2001 to 2006. He interned at Borland Software Corporation in 2001. He worked as a system integration engineer for ARAMEX from 1998 to 2000. His research interests are in network and web security, intrusion detection and forensics, artificial intelligence, pattern matching, natural language processing and bioinformatics. He published several well cited articles.

Dr. Aldwairi is an IEEE and Jordan association of engineers' member. He served at the steering and TPC committees of renowned conferences and he is a reviewer for several periodicals. He organized the Imagine cup 2011-Jordan and the national technology parade 2010.



Rami Alsalman was born in Irbid, Jordan in 1986. He received his B.S. in computer information systems from Jordan University of Science and University in 2008. He received his M.S. degree in computer engineering JUST in 2011.

He is currently working on his PhD in cognitive systems from the computer science department at the University of Bremen, Germany. He has three publications in AI and security areas.

He received a three months research scholarship at the Heinrich-Heine University of Düsseldorf, Germany during the summer of 2010. His research interests include security, data mining, information assurance and data reasoning.

Eng Alsalman is an editorial board at an International Journal of Web Applications. Additionally, he was a committee member in International Conference on Informatics, Cybernetics, and Computer Applications (ICICCA2010).