International Journal of Information Security manuscript No.
(will be inserted by the editor)

Mario Kozina · Marin Golub · Stjepan Groš

A method for identifying Web applications

the date of receipt and acceptance should be inserted later

Abstract Web applications are ubiquitous in today's businesses. The security of these applications is of utmost importance since security breaches might negatively impact good reputation, and even result in bankruptcy. There are different methods of assessing security of Web applications, mainly based on some automated method of scanning. One type of scan methods feeds random data to the application and monitors its behavior. The other type uses a database with predefined vulnerabilities that are checked one by one until either a vulnerability is found, or it can be claimed that the application doesn't have any known vulnerabilities. The important step in latter type of scan process is the identification of the application since in that case we are narrowing number of checks and, as a consequence, the scan process is faster. This paper describes a method for Web application identification based on a black box principle. Our method is based on the invariance of certain characteristics of Web applications. We experimentally tested and confirmed the usefulness of this approach.

Keywords Web security · Web application identification · Fingerprinting

This work has been carried out within projects 036-0361994-1995 Universal Middleware Platform for Intelligent e-Learning Systems and 036-0362980-1921 Computing Environments for Ubiquitous Distributed Systems both funded by the Ministry of Science and Technology of the Republic Croatia.

M. Kozina · M. Golub · S. Groš
Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia
E-mail: mario.kozina@gmail.com

M. Golub
E-mail: marin.golub@fer.hr

S. Groš
E-mail: stjepan.gros@fer.hr

1 Introduction

Today's Web applications' complexity ranges from a very simple, few lines of a code, to a complex applications supporting enterprises. Furthermore, the availability of a general purpose Web applications, e.g. Web mail clients, forums, CMS, makes it even more attractive for use because of non-existing development cost and a low deployment cost. Still, all these benefits come with the great security risks [17, 18]. This is particularly true for the Web applications that are used in business critical tasks where compromise might reveal sensitive data and lead to financial loss, and the most importantly, severely impact business reputation and trustworthiness. The further complication with respect to the security is the principle *don't fix if it isn't broken* rule! Because of that rule, companies are very reluctant to regularly update their applications without being forced to do so and this opens potential for unpatched vulnerabilities.

All this makes the security of the Web applications a challenging task. Manual search for vulnerabilities in the applications, while the most thorough, has two drawbacks. The first one is that's labor intensive. The second one is that it requires highly skilled expert which can not be found so easily. So, it would be the best that the process of discovering vulnerabilities is based on some kind of automated procedures. Automated scan of network and network application for known vulnerabilities is not a new idea. There are a number of available tools for that purpose, e.g. Nessus [4], or Satan [6]. The main purpose of these tools is to do automated network security assessment in order to improve security. There are tools of different complexity for assessing Web application security that can do certain part of the security assessment tasks, more or less automated, but this area is not even close to the mentioned tools like Nessus or Satan.

To be able to verify the existence of vulnerabilities in an application it is important to have a list, or database, of all the known vulnerabilities of all the known applications. The process of searching for a vulnerability con-

sists of serial check of each vulnerability in turn. Obviously, it's not a very fast process, and to shorten it, it is useful to know which application we are dealing with. Then, we can check only a subset of all the vulnerabilities in a database related to the given application. The process of identifying the exact type and version of the Web application on a given URL is called *fingerprinting*. In this paper we present fingerprinting method for Web applications we developed and implemented. We also present experimental results done with the implementation.

The paper is structured as follows. In section 2 we describe the idea of Web application identification process. Then, in section 3 we describe scanner developed based on the presented ideas. Experiments we've done with the scanner are given in the section 4. Finally, we give conclusions and overview of future work in section 5.

2 A method for identifying Web applications

In the indentification process we assume that the only available source of the information is via HTTP. Based on that assumption the identification process can be divided in two main phases:

- collecting characteristic information from the known Web application in order to build fingerprint database, and
- collecting characteristic information from an unknown Web application and comparing it to a data in the application database

In the following subsections we first describe what we mean by *characteristic information*. Then, we describe process of collecting the information from known Web applications, and finally, we describe the process of identifying unknown Web application based on gathered data.

2.1 Information used in the identification process

Identification process is based on collecting and comparing information from Web applications. When fingerprinting Web applications, all the gathered information differ in their reliability. *Characteristic information* is the information that is used in the decision process. The ones that are more reliable have greater impact on the final decision. Information is more reliable if it is not easily changed by a person deploying or administering Web application. For example, information collected from the HTML itself, e.g. tables and styles, can be easily changed and are thus less reliable. On the other hand, we assume that the information which is related to the source code of a Web application can't be changed easily, since not many users are proficient in programming tasks or have enough time for program modifications.

So assuming that source code of a Web application is not changed, we can use two types of characteristic information for the identification process: link patterns and forms.

A *link patterns* is a set constructed from all the http URLs collected from a Web application. For the purpose of the set construction process we assume that the http URL consists of the host part, followed by a path component and followed by the optional query part that contains parameter and value pairs. If there is a query part then it is delimited from the path component with the question sign (?), and multiple parameters and value pairs are delimited with the '&' sign. Parameter and value are separated with the equal sign [15]. Link pattern consists of the path component, and all the parameters found in the URL with the order of the parameters preserved, and values ignored. As an example, suppose that we have the following URLs:

```
1: http://www.example.com/index.php
2: http://www.example.com/index.php?i=1&a=2
3: http://www.example.com/index.php?i=2&a=3
4: http://www.example.com/index.php?j=2&a=3
5: http://www.example.com/index.php?a=2&j=3
6: http://www.example.com/index.php?j=2
```

Then we have the following link patterns:

```
1: (index.php)
2: (index.php, i, a)
3: (index.php, i, a)
4: (index.php, j, a)
5: (index.php, a, j)
6: (index.php, j)
```

When comparing two link patterns we say that they match if they have the same number of components and each respective component of each link pattern is the same. Thus, in the given example we can unify 2nd and 3rd link patterns as they are the same.

We assume that each Web application has a specific set of the link patterns and the link patterns can only be changed by adjusting a source code of the Web application. Based on the assumption that source code is not easily changed, link patterns represent reliable information for fingerprint process. Thus, collected link patterns of the same Web application installed on multiple locations on the Internet/intranet will be very similar. By collecting and comparing link patterns of a different Web application, we can precisely fingerprint a certain Web application. Due to their reliability, we assume that the link patterns have a significant impact on the final decision in the identification process.

Forms are entry points in the Web applications that allow users to supply data. Every form can be identified by its name, id, method, and URL to which data is submitted and by names and values of its input fields. Most

Web applications have characteristic forms in HTML which can be easily identified and thus used in a fingerprint process. On the other hand, of all the possible forms in a single Web application only a subset may be present because administrator, via different control mechanisms, can easily disable certain forms. This leads us to the conclusion that the forms are potentially less reliable than link patterns. Additional reason the forms are less reliable is that fields in the forms can be dynamically added or removed by the backend, depending on the context. For example, if we are entering address inside the United States then we are presented with the *State* field, while for Croatia this field might not be shown.

Apart from the link patterns and forms, we also take into account certain keywords in HTML documents. They could be easily changed, but some licenses require users to embed different keywords into HTML documents, users intentionally leave identifiers, or, in some cases, the specific keywords are generated by code and thus are not easily changed by the user. As an example of a keyword, we can take Mambo Web application which uses keywords *"Mambo"* and *"http://mambo-foundation.org"* in HTML documents. As a conclusion, we assume that keywords have small, but non negligible, impact in the final decision of the identification process.

2.2 Information gathering

Before the identification process, we need to have a database of known Web applications containing their characteristic information. Characteristic information is collected from the HTML documents generated by the target Web application using *crawling* (or *spidering*). To extract a characteristic information from each HTML document, we analyze it's structure. This effectively means searching for `<a>` and `<form>` elements from which we extract links and forms. Crawling ends when all the links in the Web application up to the certain depth were traversed.

During the crawling process we collect and organize links into link patterns, as described in the previous section. As it was already mentioned, the most usual separators are '`&`', '`?`' and '`=`', but today's Web applications can use more complex separators like 'QQ' and 'EE' [1].

Forms are extracted from a `<form>` element in the HTML document. The `<form>` element often has attribute *name* and a list of `<input>` elements. In a crawling process, all the forms are organized into list, where each element of the list is itself again a list with the following elements: form name, URL where data will be submitted, and a list of input name entries. Input values already present in the from, e.g. the default values in the case user doesn't enter them, are not taken because they depend on the localization of a Web application which makes them unreliable information for identification process.

During the process of collecting useful information from a known Web application keywords have to be added manually by the user into a fingerprint database as there is no way for the program to know which words on the page are important and can be used for this purpose. It is important to choose a set of keywords which will best characterize a certain Web application and differentiate it with respect to other Web applications. When collecting characteristic information from an unknown Web application and comparing it to a known application database (fingerprint), keywords are searched anywhere in a HTML document body.

Crawling can be very comprehensive and time-consuming process because of a enormously large number of a very similar links in a certain Web application. For example, there could be a page where some items from the database are shown. In that case there will be as many pages as there are entries in the database, but for our purpose all those pages are the same. As the identification process needs to be done in a reasonable time and the number of almost the same links doesn't bring any new information that might be useful for fingerprint process it is suggested to go through only a certain number of a links in a Web application. So, when the crawler discovers that the traversed links are the same it can stop further processing of the given pages and skip to the next link.

When collecting characteristic information from a known Web application, we collect and structure useful information into a database that will be used to fingerprint unknown Web applications. For the reliability purposes, we use several instances of the Web application for which we generate the fingerprint database. Then, the final step of this phase is to select *the best fingerprint database*, among the several available, for the Web application. The selection is based on the process similar to the identification process, but that is performed on the *known* Web application. The best fingerprint database should have a large number of different forms and link patterns. The selection of the best database thus reduces a chance of failure in the identification process.

2.3 Identification process

In the identification process we collect characteristic information from the Web application that has to be identified and compare collected information with the data stored in the fingerprint database. For each application in the fingerprint database we calculate similarity with the unknown Web application using the following formula:

$$rating = k * keywords_r + l * linkp_r + f * forms_r \qquad (1)$$

In (1) the following variables are used:

- $keywords_r$ is relative number of identical keywords found in all the pages;

- $linkp_r$ is relative number of identical links found in all the pages;
- $forms_r$ is relative number of identical forms found in the pages;
- k, l and f are weight factors in the inclusive range 0 to 1.

Relative numbers are calculated to the best achieved result. E.g. if we collected 30 link patterns, and in the three fingerprint databases we have 20, 16 and 10 matches, then $linkp_r$ for the first database will be 1, 0.8 for the second and 0.5 for the last database. Note that absolute number of link patterns doesn't influence the relative number of link patterns. Thus, in order to reliably identify application there should be at least three link patterns. It is of course advisable that this number is as higher as possible. In our case, number of collected characteristic information per application is shown in the Table 1.

In order to normalize the final result (rating) the sum of all parameters has to be equal to 1. In our case identification of Web applications reduces to evaluation of three parameters: keywords (k), forms (f) and link patterns (l). So the following equation holds:

$$k + l + f = 1 \tag{2}$$

Web applications which have rating 1 (100 percent) are exact match by the all three criteria, i.e. forms, link patterns and keywords. The Web application with rating 100 percent is chosen as the final result of an identification process. Moreover, in general, Web application with the best rating, not necessarily 100 percent, is chosen as the final result. On the other hand, if we have the same best ratings for several Web applications, final decision of the identification process can be determined only with a certain probability, and to come to a final decision we need to manually check those Web applications with best ratings.

3 Framework design

The Web Security Assessment Tool[14] (WSAT) is the framework we are developing for experimenting with new ideas in a Web application security assessment. Web security assessment is a complex process which we divided into several phases. So, to represent each phase of the process and to provide a capability of future enhancements, WSAT is implemented as a modular system whose architecture is shown in the Figure 1. Of all the shown modules, the exploit module will not be described as it is not interesting for this work. As an implementation language for WSAT we selected very popular, and widely supported, high level object oriented language Python.

Crawler module is used to abstract communication details between WSAT and a Web application. It gathers Web pages from the Web application and extracts different information for it's own purpose, e.g. links to fetch.

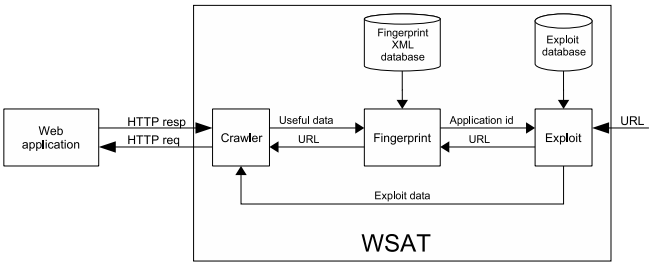

Fig. 1 The Web Security Assessment Tool architecture

It also gives fetched pages to the other modules in the WSAT framework (fingerprint, exploit). Crawler implementation is based on Wapiti security tool programming code [13], on which various communication structures and iterative crawling method were based. The Crawler module goes through these phases:

- collecting HTML,
- analyzing HTML,
- building Web application link structure, and
- organizing links.

Crawler is the only module which communicates directly with a Web application and it's purpose is to fetch HTML document starting from the URL given by fingerprint module. It contains communication handlers for different types of the communication protocols and mechanisms (e.g. HTTP, HTTPS, proxy, cookies). After fetching an HTML document, tag parser is used to analyze HTML and grab important information from the HTML: forms and links. The web application link structure is then built by using iterative crawling process. The process of crawling uses three lists of URLs. In the first list there are all as of yet unvisited URLs. This is list is at the start of the crawling process initialized to a starting URL of a Web application. The second list is a list of visited links, which at the end of the crawling process is composed of all the gathered links. From this list the link patterns for the given application are generated. Finally, the third list is a list of banned URLs that contains forbidden links defined at the start of the crawling process.

The crawling process takes new links from HTML, checks if any of them is in the banned list and discards those that are. Then, the remaining links are added into the non visited list and the process is repeated for the links in the non visited list until the list becomes empty. If a link has already been visited, i.e. it is in the visited list, then it is skipped. As we said before, this process can be time consuming, so we use *smart module* to optimize it. Smart module organizes links into link patterns and checks when a link with a certain pattern repeats a certain number of times, defined by some user defined threshold. In case when this threshold is reached, a patterns that matches the link group is added into the banned list which prevents further crawling of that link

Table 1 Number of characteristics

Characteristic information	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB	Total
Keywords	4	3	4	4	3	3	3	3	3	30
Forms	11	11	5	4	14	7	4	4	3	63
Link patterns	40	27	17	8	18	50	37	16	13	226

group. The additional functionality of the smart module is to generate link patterns that are handed to the fingerprint module.

Fingerprint module implements the identification process described in the previous section. As we stated before, data collected from a known Web application should be structured in a fingerprint database. To simplify gathering and comparing phases of the identification process and to standardize structure of a useful information, we have chosen XML format for the database. The XML fingerprint file consist of some general information and information needed for the identification process. The general structure of the XML database is:

```
<?xml version="1.0" encoding="utf-8"?>
<fingerprint>
  <name></name>
  <id></id>
  <keywords>
    <keyword></keyword>
    ...
  </keywords>
  <link_patterns>
    <separator></separator>
    <pattern >
      <weight></weight>
      <url></url>
      <params>
        <param name=""></param>
        ...
      </params>
    </pattern>
    ...
  </link_patterns>
  <forms>
    <form name="" location="">
      <from></from>
      <to></to>
      <field name=""></field>
      ...
    </form>
    ...
  </forms>
</fingerprint>
```

Some of the most important elements included in a XML fingerprint file are: `<name>`, `<id>`, `<keywords>`, `<link_patterns>`, and `<forms>`.

The element `<name>` is used to store Web application's full name and it's version while `<id>` element contains a unique number that identifies a certain Web application under the WSAT framework. The element `<keywords>` contains one or more `<keyword>` elements which contain different keywords for Web application. Link patterns are organized into `<link_patterns>` element which consist of a variable number of `<pattern>` elements, one for each link pattern found during the crawling process. Pattern has an `<url>` element which represents relative URL to the document (e.g. `/index.html`) and adequate number of parameters, each included into `<param>` element and described by name and all values of parameters. Forms are placed in a `<forms>` top level element which contains a `<form>` element for each form that was found in the Web application. Element `<form>` has a name attribute and adequate number of `<field>` elements used to describe field (input) name and content.

Fingerprint module operates in two modes. The first mode is the generator mode, which is used to collect a characteristic information from a known Web application and to store it into XML fingerprint file. These XML files are used in the second mode of the Fingerprint module to compare and determine which XML file best describes unknown Web application. The second mode is also used to determine the best fingerprint database for a certain Web application. The determination process is done by comparing each type of information from the XML fingerprint file to each type of characteristic information collected from an unknown Web application. There are certain rules that are enforced when comparing the information:

- To count keyword as matched, the complete string has to be identical including case.
- Link pattern must have the same relative URL, same number of parameters and identical names of parameters,
- Forms must have same name and a certain, minimum, number of a identical field names. The minimum number of identical field names in the form comparison is used because some forms have dynamically added or removed fields. This minimum number is determined by the user of the fingerprint module.

After comparing and calculating each type of information, (1) is used to rate each fingerprint database and, indirectly, to assess which application was fingerprinted.

4 Experimental verification

After developing the fingerprint module inside WSAT framework we performed experiments to validate our approach of fingerprinting Web applications. Furthermore, we wanted to determine approximate optimal values of the weight factors k, f, and l, used in the rating (1) which forms the base of the identification process. In other words, it is necessary to confirm hypothesis that link patterns, forms and keywords can be used to fingerprint a Web application and to find best measures (weight factor values) to accurately fingerprint a Web application.

Experiments were performed in three phases, each phase using a separate set of URLs:

- Collecting fingerprint databases from the known Web applications;
- Rating and selecting the best fingerprint XML database for each Web application;
- Identification process of unknown Web applications and the result analysis.

In the first phase we collected characteristic information from the URLs that contain known Web application. Then, in the second phase we compared collected fingerprint databases with another set of the URLs with the known Web applications to select the best XML fingerprint database for each Web application that will be detected. The selected XML fingerprint database will be used in the identification process. Finally, in the third phase, a few tests with different weight factor values were tried to determine where the optimal values for the weight factors are placed and to determine how well the process of identification of unknown Web application performs.

To perform all the experiments, we selected the following open source Web applications:

- Content Management Systems (CMS): Joomla[2], Mambo[3], PHP-nuke[5], Post Nuke[10]
- Forums: MyBB[8], PhpBB[9], UseBB[12], BBpress[7], PunBB[11]

Of all the selected applications, Joomla and Mambo are specific since Joomla is a fork of Mambo. Thus we expect them to be very similar in terms of forms and links which is an additional test for our method of recognizing Web applications.

4.1 Collecting and rating fingerprint XML files

Before fingerprinting unknown Web applications, we need to have the best possible fingerprint XML database describing each Web application selected for the experiment. The good XML file should have large amount of characteristic information to reduce chance of making wrong decision when identifying unknown Web applications. To collect XML files, the threshold of the smart module was set to high a level, i.e. 40 for a CMS applications and 10 for forums. After collecting a number of fingerprint XML databases per the Web application, we rated each XML file on a test applications by using rating formula (1).

Table 2 Choosing the best Joomla XML fingerprint file

Sites/XML files	A	B	C	D
Site 1	84	100	53	7
Site 2	100	100	60	10
Site 3	100	100	57	7
Site 4	100	90	60	10
Site 5	100	90	63	9

For example, we gathered four fingerprint databases for Joomla Web application. Three XML files (A,B,C) were gathered from different sites where the same Web application was running and fourth (D) was gathered from a fresh, locally installed Web application. In next step, we used fingerprint module to rate these XML files on five different Web sites with Joomla to see how well can they identify Web application type. To simplify this process, we used weight factor with values $l = 1$, $k = 0$, and $f = 0$, i.e. the decision was based on a link patterns only. The obtained results are shown in the Table 2.

The table shows that A and B variant XML files have the best ratings, but A has the best overall rating, so we selected XML file A to fingerprint Joomla applications. Surprisingly, XML file D, which is from the fresh install, has the worst rating for all the sites. The main reason for such result is that fresh installations of Web applications don't have all features installed, thus they don't have enough usable information like link patterns and forms which could be gathered and used in the identification process. To support this hypothesis, we checked internal structure of A and D fingerprint files and revealed that XML file A had 25 link patterns and D file had only 9 link patterns.

4.2 Characteristic data

After collecting and selecting the best XML fingerprint data for every Web application in the list, the next step is the identification of unknown application and result analysis. In this part we conducted three different tests for each Web application listed above on the randomly chosen URLs. Every test had a different values of the weight factors which depends on assumed characteristic data reliability. When examining characteristic data reliability, we assume:

- Link patterns are the most reliable type of useful information of a Web application. We assume that link patterns are least susceptible to changes from a Web

Table 3 Weight factors used in experiments

Test/weight factor	k	f	l
Test 1	0.1	0.2	0.7
Test 2	0	0	1
Test 3	0	0.4	0.6

developer or a Web administrator. Thus, functionality of most Web applications is based on links, so they can be easily gathered.
- Forms are reliable type of useful information, but not so as link patterns. Forms are susceptible to changes and they can easily be eliminated from the Web application using some kind of administrative interface available to the Web administrator.
- Keywords are not very reliable type of useful information. They are susceptible to changes and can be easily changed or even eliminated from the HTML. Furthermore, keywords can easily guide to a wrong conclusion as it is possible that they are used in some other context in a Web application content, e.g. discussing Mambo on the Web forum based on phpBB.

Therefore, the three listed assumptions give the additional constrain on parameters k, l, and f:

$$k < f < l \tag{3}$$

4.3 The weight factors influence analysis

4.3.1 Parameter setings

Based on the previous discussions we selected three sets of weight factors, show in the Table 3.

As it can be seen, keywords are taken into consideration in the first test where they participate in the final decision with minimal 10 percent. Forms haven't effect the rating in the second test, have less effect in first (20 percent) and moderate effect in the third test (40 percent). Link patterns are used in every test with largest effect, especially in the second test where final decision is based on link patterns only (100 percent).

All nine selected Web applications were being tested three times, each time with different weight values, where testing was made on average 6-10 URLs where certain Web application was installed. For instance, we had 10 URLs with Joomla and 6 URLs for MyBB. We were certain that these sites had a certain Web application installed but for our experiments (identification), we acted as they are unknown Web applications. In the following subsection we will show a representative results of the experiments.

4.3.2 The results for the CMS Web applications identification

First, we shall examine results from CMS Web applications. CMS Web applications don't have large amount of content and useful information gathering isn't time consuming. So, the threshold value in the crawler module was set to 20 in order to collect large amount of characteristic information. Larger amount of characteristic information allows fingerprint module to make better decisions and thus give better results.

As a representative for the analysis of the CMS group, we present Joomla Web application. In the Table 4, the results of the first test of Joomla are shown with the weights $k = 0.1$, $f = 0.2$, and $l = 0.7$.

The results of the first test show a similarity between Joomla and Mambo. Other Web applications have substantially smaller ratings or they even don't have a rating (0 percent), so we can, with great certainty, say that these application weren't present at tested URLs. Although we knew that on the tested URLs was Joomla, URLs S1-URL2 and S1-URL9 gave us surprising results for Joomla and Mambo. At first URL, Joomla has a slightly better result (100) than Mambo (90), and at the second URL Joomla had better rating (100) than Mambo (70). S1-URL6 is also interesting because it shows absence of keywords, thus, maximum rating is decreased by 10 percent which is rating influence of keywords. To gain a better understanding at the influence of the selected weights, we also show the results of second and third test. The Table 5 shows the results for the weights $k = 0$, $f = 0$, and $l = 1$, and the Table 6 results for the third test with the weights $k = 0$, $f = 0.4$, and $l = 0.6$.

Results of the second test show that Joomla and Mambo have a very similar link patterns, especially if we look at S1-URL9 and S1-URL2. For these URLs, both Joomla and Mambo have high ratings (100), so final decision can't be made when looking at link patterns only.

If we look at the results of the third test, we can notice that there is also similarity between Mambo and Joomla in the forms. Looking at all the results, starting from weight set 1 to set 3, the difference between ratings have increased in Joomla's favor. This tells us that Mambo and Joomla are less similar in the forms than in the link patterns. The confirmation of this can be found in the difference between the results of the second and third test at S1-URL9 where Mambo rating decreased from 100 percent to 60 percent. The cause of this decrease is that Web application at the given URL doesn't contain any form which can be found in Mambo XML fingerprint file. On other hand, if we look at S1-URL2 we notice that ratings haven't changed from second test which means that the same number of forms was found in the Mambo and the Joomla XML fingerprint files. By exploring S1-URL2 we determined that there is only one form in the Web application. When comparing login forms in Joomla and Mambo XML fingerprint files, great similarity can be noticed. Because we use flexible rating of forms, where 80 percent of fields must be the same, forms of the both Web applications are adequate. So, in this case (third test) for S1-URL2 final decision can't be made.

Table 4 The results for Test 1 of Joomla Web application detection

Test URL	Total score									Detected?
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB	
S1-URL1	100.0	0.0	51.5	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL2	100.0	0.0	90.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL3	100.0	0.0	40.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL4	100.0	0.0	14.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL5	100.0	0.0	58.2	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL6	90.0	0.0	23.8	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL7	100.0	0.0	30.8	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL8	100.0	3.3	52.8	3.3	3.3	3.3	3.3	3.3	3.3	Y
S1-URL9	100.0	0.0	70.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL10	100.0	0.0	25.2	0	0	10.0	0.0	0.0	0.0	Y

Table 5 The results for Test 2 of Joomla Web application detection

Test URL	Total score									Detected?
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB	
S1-URL1	100.0	0.0	45	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL2	100.0	0.0	100	0.0	0.0	0.0	0.0	0.0	0.0	N
S1-URL3	100.0	0.0	50	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL4	100.0	0.0	20	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL5	100.0	0.0	76	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL6	100.0	0.0	34	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL7	100.0	0.0	44	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL8	100.0	0.0	66	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL9	100.0	0.0	100	0.0	0.0	0.0	0.0	0.0	0.0	N
S1-URL10	100.0	0.0	36	0.0	0.0	0.0	0.0	0.0	0.0	Y

Table 6 The results for Test 3 of Joomla Web application detection

Test URL	Total score									Detected?
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB	
S1-URL1	100.0	0.0	67.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL2	100.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	N
S1-URL3	100.0	0.0	30.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL4	100.0	0.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL5	100.0	0.0	55.6	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL6	100.0	0.0	20.4	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL7	100.0	0.0	26.4	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL8	100.0	0.0	39.6	0.0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL9	100.0	0.0	60.0	0,0	0.0	0.0	0.0	0.0	0.0	Y
S1-URL10	100.0	0.0	21.6	0.0	0.0	20.0	0.0	0.0	0.0	Y

By analyzing the results, we came to the conclusion that first test performs the best when identifying unknown Web application. Keywords helped us in solving the final doubt for S1-URL2. Although our initial hypothesis given in the Section 2 assumes that link patterns and forms are the most important when making final decision, sometimes it also beneficial to take keywords in consideration.

In the CMS group, it is also interesting to look at the results of the third test for Mambo, shown in the Table 7.

Although this test was made on URLs where Mambo was installed, the results of the test show again a similarity between Mambo and Joomla when looking at characteristic information. This behavior is expected as we already noted that Joomla is a fork of Mambo. Besides that, it is also interesting to look at S2-URL4, S2-URL7, and S2-URL10 where the rating is only 60 percent in favor of Mambo. This rating shows that not even one form from Mambo XML fingerprint file was found on these URLs. By exploring these URLs, we noticed that there is only one voting form present on each URL. Although Mambo has voting form stored in his XML fingerprint file, a number of voting fields in the forms present at the tested URLs is larger than number of voting fields in forms in XML file. So, by comparing these forms, the fingerprint module rated them as unequal.

For the other applications from the CMS group, PHP-Nuke and PostNuke, it is important to say that they have unique link patterns. As a consequence, in the second

Table 7 The results for Test 3 of Mambo Web application detection

Test URL	Total score									Detected?
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB	
S2-URL1	45.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S2-URL2	62.6	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S2-URL3	39.6	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S2-URL4	34.2	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S2-URL5	77.2	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S2-URL6	77.2	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S2-URL7	42.6	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S2-URL8	70.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	Y
S2-URL9	91.0	0.0	100.0	0.0	20.0	0.0	0.0	0.0	0.0	Y
S2-URL10	60.0	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0	N

test both Web applications for their URLs have maximum ratings (100), while all the other Web applications don't have even a one adequate link pattern (rating 0). There are also a few results in third test where a rating is 60 percent, which shows absence of forms in the Web application.

4.3.3 The results for the forums identification

The next group of the Web applications we tested were forums. Forums have relatively bigger amount of data than the CMS Web applications. Reason lies in their functionality, to enable user to exchange information by using different categories and topics of forum. If forum has a large number of users, then amount of data with reference to amount of characteristic information can also be very large. So, it is necessary to customize crawler module with a relatively small threshold in the smart module to gather useful information in a reasonable time. But, as a consequence of the small threshold, there is a greater chance of introducing errors and uncertainties in the identification process. We set the treshold value to 5 during the fingerprint process.

As a representative for the forum group, we took UseBB forum. In the Table 8 results of first test with the wight factors $k = 0.1$, $f = 0.2$, and $l = 0.7$ for UseBB are shown.

From the results of the first test for UseBB, shown in Table 8, we can notice that for the most URLs, UseBB is properly identified using all three criteria - it has rating 100 percent. Other Web applications have ratings which are less then or equal to 15%, so those Web applications can be excluded from consideration. But, we can also notice few exceptions, especially on S3-URL4 and S3-URL5 where ratings are relatively low (30%). To explain this exceptions it is necessary to analyse results from the second and third test, which are shown in the Tables 9 and 10.

From the results of the second test, we can notice that a cause for the exceptions in the first test lies in absence of correct link patterns in UseBB XML fingerprint file. It we look at the other results from the same test, the ratings are 100 percent, which is at first a little confusing. But, by exploring UseBB forums on S3-URL4 and S3-URL5, we determined that administrator of UseBB forum has the ability to change names of parameters in the links. For example, S3-URL4 has parameter off_sid instead of standard UseBB parameter $usebb\_sid$. As a consequence, link patterns are not adequate and comparable, so it is logical that ratings for these two URLs in second test are 0. This also shows that in some situations, link patterns are not so reliable.

The results of the third test reveal the cause of the exception in the second test for URL S3-URL9 is the absence of adequate forms. By further examination, we determined that the crawler didn't find any form at the given URL, which is very likely because crawler used very low value for the treshold in the smart module to successfuly gather enough forms. It is also interesting to notice, relatively high ratings in third test for PHP-Nuke at two URLs. This shows that some forms in PHP-Nuke fingerprint file are very similar to the ones in UseBB XML fingerprint file.

Other results for Web applications from forum group are mostly consistent and expected. This particulary refers to the second test, where is no doubt, because ratings were 100% for one Web application and other application have rating of 0%. In the third test, there are some higher ratings for the Web applications which are not present at a certain URLs, but bellow 20%.

4.4 Final results

Analyzing all nine instances of Web applications, we determined that the first test, with the weights $k = 0.1$, $f = 0.2$, and $l = 0.7$, gave the best results. The results of all the tests have shown that we need to take into considerations all three types of characteristic information to get better results and especially to resolve doubts where final decision can't be made. It is also shown that it is necessary to respect the condition (3), because the major difference between Web applications is in the link patterns, followed by the difference in the forms with keywords at the end.

Table 8 The results for Test 1 of UseBB Web application detection

Test URL	Total score									Detected?
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB	
S3-URL1	0.0	10.0	0.0	5.0	15.0	0.0	0.0	100.0	0.0	Y
S3-URL2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	Y
S3-URL3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	Y
S3-URL4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	30.0	0.0	Y
S3-URL5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	30.0	0.0	Y
S3-URL6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	Y
S3-URL7	0.0	0.0	0.0	6.6	10.0	0.0	6.6	100.0	0.0	Y
S3-URL8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	Y
S3-URL9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	80.0	0.0	Y

Table 9 The results for Test 2 of UseBB Web application detection

Test URL	Total score									Detected?
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB	
S3-URL1	0	0	0	0	0	0	0	100	0	Y
S3-URL2	0	0	0	0	0	0	0	100	0	Y
S3-URL3	0	0	0	0	0	0	0	100	0	Y
S3-URL4	0	0	0	0	0	0	0	0	0	N
S3-URL5	0	0	0	0	0	0	0	0	0	N
S3-URL6	0	0	0	0	0	0	0	100	0	Y
S3-URL7	0	0	0	0	0	0	0	100	0	Y
S3-URL8	0	0	0	0	0	0	0	100	0	Y
S3-URL9	0	0	0	0	0	0	0	100	0	Y

Table 10 The results for Test 3 of UseBB Web application detection

Test URL	Total score									Detected?
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB	
S3-URL1	0.0	20.0	0.0	10.0	30.0	0.0	0.0	100.0	0.0	Y
S3-URL2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	Y
S3-URL3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	Y
S3-URL4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	40.0	0.0	Y
S3-URL5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	40.0	0.0	Y
S3-URL6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	Y
S3-URL7	0.0	0.0	0.0	13.2	20.0	0.0	13.2	100.0	0.0	Y
S3-URL8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	Y
S3-URL9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	60.0	0.0	Y

We are left with the final question of the optimal values for the parameters k, f, and l, i.e. how to determine their optimal ratio. We suggest that the ratio $k : f : l$ is approximately in the ratio of the collected characteristic information. According to the Table 1 this ratio is $30 : 63 : 226$ which matches weight factors used in the test 1, i.e. $1 : 2 : 7$.

In the additional experiments we significantly extended the number of tested Web sites which included some two hundred samples. Some of them were not accessible all the time. This left us with a 129 reliable cases, in addition to the already presented tests, on which we now evaluate WSAT recall and precision. The results of the additional set of tests are summarized in Table 11.

Looking into the table we can make several interesting observations. First, the results for Mambo and Joomla (recall and precision rows) are less then 1. By closer inspection we noticed that in a few cases one was mistaken for the other, and vice versa. This is easly explainable as Mambo and Joomla are very similar applications, i.e. Joomla is a fork of Mambo. This fact has one significant implication. Namely, if the experiments were performed only with Joomla fingerprint database, then all Mambo Web applications would be mistakenly identified as Joomla. Therefore, the conclusion is that care should be taken in order to avoid such misidentification. The second observation is that phpBB's precision and useBB's recall is 0.93. The reason is that one instance of useBB was mistakenly identified as phpBB. Upon closer inspection of the given instance we determined that the administrators of useBB site changed URLs in such a way that it uses different link patterns, i.e. instead of `topic.php&id=9` they use `topic_9.html`. Finally, we note that the fingerprinting method achieves

Table 11 Recall and precision of WSAT identification system

	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB	Total
Number of applications	18	14	18	13	18	13	10	14	11	129
Number of detections	21	14	16	13	18	14	10	13	11	130
Relevant applications	17	14	15	13	18	13	10	13	11	124
Recall	0.94	1.00	0.83	1.00	1.00	1.00	1.00	0.93	1.00	0.96
Precision	0.81	1.00	0.94	1.00	1.00	0.93	1.00	1.00	1.00	0.95

the possible results in a majority of the cases (5 out of 9 Web applications).

Collecting characteristic information from the known Web applications and generating fingerprint databases is a critical step. The better fingerprint databases are, less sensitive the decision process is to the variations in the parameter values. Even with equal values of the parameters, the method gives good results, as can be seen from Figures 2 and 3. The figures are the result of exhaustive search with parameter step 0.01 and restrictions given by equations (2) and (3). The exhaustive search found several optimal values for the parameters in which recall is 0.969 and precision is 0.962, e.g. $k = 0.22$, $f = 0.34$ and $l = 0.44$. This is very close to the parameter values determined by our heuristic method which gives recall 0.961 and precision 0.954 ($k = 0.1$, $f = 0.2$ and $l = 0.7$). The difference in optimal value and the value obtained with our parameters is in a single misdetected Web application of 129 tested applications. This shows that our heuristic method for estimating optimal parameter values is good enough without necessity of performing exhaustive search.

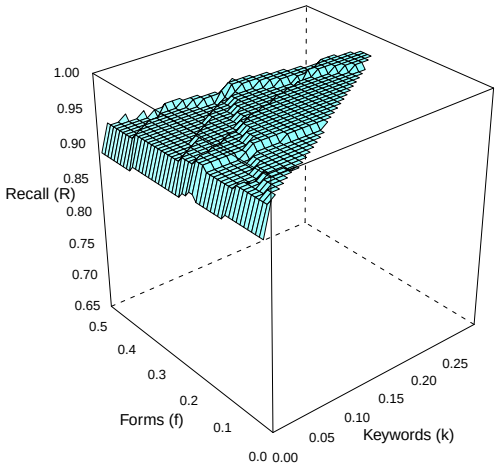

Fig. 2 The results of the exhaustive search for the optimal value of R

5 Conclusions and future work

In this paper we described a method for fingerprinting Web applications. Fingerprinting is a process of identification of unknown Web applications by comparing their characteristic data to the database with the data from the known Web applications. Successful identification of a Web application can make vulnerability scanning faster and more precise as the scanning process is concentrated on only a subset of all the known vulnerabilities.

The fingerprint process is based on comparison of characteristic information: link patterns, forms and keywords. Fingerprint process assumes that the harder to change some parts of the application the more reliable related information is and thus has a bigger impact on the final decision in the identification process. To verify this hypothesis and to determine where the optimal impact factors of a certain types of a characteristic information are, we used experimental verification. The results of the experimental verification validated our hypothesis that link patterns have the greatest impact, followed by forms and finally keywords with the lowest impact on final decision. Although the results indicated that Web applications mostly differ by their link patterns, there were cases where single comparison of link patterns (without usage of other types of information) left some uncertainties. To resolve those cases it is important to include other two types of characteristic information in the final decision, but with lesser impact. Final results were achieved with the following values of impact (weight) factors:

- link patterns impact is 70%,
- forms impact is 20% and
- keywords impact is 10%,

that are in accordance with the ratio of the collected characteristic information in fingerprint databases.

The framework we used (WSAT) and in which the described fingerprint technique was implemented, can identify a particular Web application with high precision. Except for some rare cases, for most URLs, the system

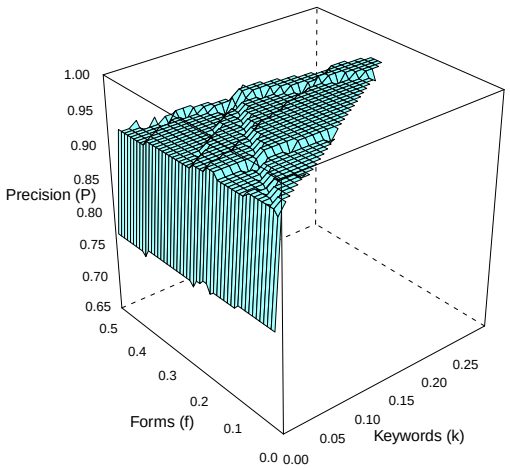

Fig. 3 The results of the exhaustive search for the optimal value of P

correctly identified Web application with the ratings of 100%.

The implementation has some shortcomings that prevent it from being used on the Web applications with following characteristics:

- Web applications that don't use standard link separators in links. These separators are composed of characters which are hard to distinguish from link content itself, so they cause difficulties in link analysis and link organization into link patterns.
- Web applications that don't use links for passing parameters and its values. This is often the case with Web applications developed in ASP.NET environment where parameters and its values are passed through session variables with POST method.
- Web application that use JavaScript or Ajax technology for link generation. In this case, links cannot be gathered by simple analysis of HTML documents.

The given limitations are implementation problem, which can be relatively easy resolved by further development. The system's design is such that the majority of the upgrades should be done in the crawler module to make it capable of gathering information from different sources like JavaScript, Ajax and other new Web technologies.

Fingerprint databases should also be extended to include other types of characteristic information which would made identification process even more precise. By adding these information, the identification method could be also applied to a wider set of Web application technologies like ASP.NET, JavaScript and Ajax. For example, using JavaScript fragments of the code embedded into the pages. There is also a possibility to experiment with different fingerprint procedures, like algorithms from the pattern recognition or data mining fields.

Finally, we note that there is a possibility that someone could purposely try to deceive the application by manipulating keyword, and also links and forms. This could be a problem but it is a matter of further research to evaluate how the technique presented in this paper could be made more robust against such cases.

References

1. eBay (2007). URL `http://www.ebay.com`
2. Joomla! (2007). URL `http://www.joomla.org/`
3. Mamboserver.com - Home (2007). URL `http://www.mamboserver.com/`
4. Nessus vulnerability scanner (2007). URL `http://www.nessus.org`
5. PHP-Nuke (2007). URL `http://phpnuke.org/`
6. Security Administrator Tool for Analyzing Networks (SATAN) (2007). URL `http://www.porcupine.org/satan/`
7. bbPress (2008). URL `http://bbpress.org/`
8. MyBB - Free PHP and MySQL Forum Software (2008). URL `http://www.mybboard.net/`
9. phpBB - Creating Communities Worldwide (2008). URL `http://www.phpbb.com/`
10. PostNuke CMS :: A Flexible Open Source Content Management System (2008). URL `http://www.postnuke.com/`
11. PunBB (2008). URL `http://punbb.org/`
12. UseBB — The Usable Forum Software (2008). URL `http://www.usebb.net/`
13. WAPITI - Web application vulnerability scanner / security auditor (2008). URL `wapiti.sourceforge.net/`
14. WSAT (2008). URL `http://sourceforge.net/projects/wsat`
15. Berners-Lee, T., Fielding, R., Masinter, L.: Uniform Resource Identifier (URI): Generic Syntax. RFC 3986 (Standard) (2005). URL `http://www.ietf.org/rfc/rfc3986.txt`
16. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2009). URL `http://www.R-project.org`. ISBN 3-900051-07-0
17. Rubin, A.D., Geer, D., Ranum, M.J.: Web security sourcebook. John Wiley & Sons, Inc., New York, NY, USA (1997)
18. Rubin, A.D., Jr., D.G.: A survey of web security. Computer **31**(9), 34–41 (Sep 1998). DOI 10.1109/2.708448

M. Kozina obtained dipl. ing. degree at the University of Zagreb in 2007. He is currently employed in the KING ICT, Zagreb working in the field of information security.

M. Golub Ph.D. is an associate professor at the University of Zagreb. His research interests include security and evolutionary algorithms.

S. Groš Ph.D. is a research assistant at the University of Zagreb. His research interests include computer networks with

the emphasis on the Internet protocols, operating systems, security and software engineering.