

# On the Security of a Popular Web Submission and Review Software (WSaR) for Cryptology Conferences

Swee-Won Lo<sup>1</sup> \*, Raphael C.-W. Phan<sup>2</sup> and Bok-Min Goi<sup>3</sup>

<sup>1</sup> Mobile Network Security Technology Research Center (MSRC),  
Kyungpook National University, Sankyuk-dong, Buk-gu, Daegu 702-701, Korea  
`swlo@ee.knu.ac.kr`

<sup>2</sup> Laboratoire de sécurité et de cryptographie (LASEC),  
Ecole Polytechnique Fédérale de Lausanne (EPFL),  
CH-1015 Lausanne, Switzerland

`raphael.phan@epfl.ch`

<sup>3</sup> Centre for Cryptography and Information Security (CCIS)  
Faculty of Engineering, Multimedia University, 63100 Cyberjaya, Malaysia  
`bmgoi@mmu.edu.my`

**Abstract.** Most, if not all, conferences use an online system to handle paper submissions and reviews. Introduction of these systems has significantly facilitated the administration, submission and review process compared to traditional paper-based ones. However, it is crucial that these systems have strong resistance against Web attacks as they involve confidential data and privacy. Some submissions could be leading edge breakthroughs that authors do not wish to leak out and be subtly plagiarized. Also, security of the employed system will attract more submissions to conferences that use it and gives confidence of the quality that the conferences uphold. In this paper, we analyze the security of the Web-Submission-and-Review (WSaR) software - latest version 0.53 beta at the time of writing; developed by Shai Halevi from IBM Research. WSaR is currently in use by top cryptology and security-related conferences including Eurocrypt 2007 & 2008, Crypto 2007, and Asiacrypt 2007, annually sponsored by the International Association for Cryptologic Research (IACR). We present detailed analysis on WSaR's security features. In particular, we first discuss the desirable security features that are designed into WSaR and what attacks these features defend against. Then, we discuss how some untreated security issues may lead to problems, and we show how to enhance WSaR security features to take these issues into consideration. Our results are the first known careful analysis of WSaR, or any type of online submission system for that matter.

**Keywords:** Web submission and review software, security analysis, privacy, passwords, email, protocol.

---

\* Part of work done while the author was at CCIS@Multimedia University (Cyberjaya campus) and iSECURES Lab@Swinburne University of Tech (Sarawak campus).

## 1 Introduction

About two decades ago, authors interested to submit papers to a conference would submit via postal service or airmail. After being reviewed, papers (together with the reviews) would be sent back by similar means. Before the camera-ready deadline, authors of accepted papers would need to race against time to send their camera-ready versions over, and hope that they do not get lost in transit. This method of correspondence is not only costly, it is also time-consuming. More notably, it will be hard to trace lost or delayed papers and reviews since it all depends on the reliability of the postal system. Fast forward a few years, technology advances introduce the use of email (attachments) and facsimile. Although most email services are free of charge, this method is often limited in terms of the size of the attachment(s). Facsimile, on the other hand, can be costly although fast.

Thus, the introduction of online web-based systems significantly facilitates the paper submission and review process [26], and overcomes shortfalls in the traditional paper-based system. Authors and reviewers can track their papers' progress anytime, anywhere, as long as they have an Internet connection. In addition, the conference Chair is now capable of managing papers and reviews more effectively, as well as reacting quickly to feedback and complaints.

According to a survey [26] by ALPSP<sup>4</sup> on web submission and review systems for journals, among the 442 respondents selected at random from the ISI Web of Knowledge database, 81 per cent preferred to use web submission and review systems and 36 per cent said that they would think twice when choosing a journal without online submission for their work. Following the introduction of online submission, there was a 25 per cent increase in submission volumes and publishers reported a 30 per cent decrease in administration time. From this survey, we see that online submission and review systems are playing a significant role. Nevertheless, popular though they be, there is still an issue involved that should be a major concern among the community - how secure are the data handled by these systems?

In this setting, security and privacy can be seen from two opposite perspectives. One, arguably less eminent, is the risk of malicious individuals attempting to obtain unauthorized access to leading edge research results and thus idea theft, or cause unfair dismissal of submitted papers. On the other is the case of honest paper authors desiring that the submission system maintains their privacy and secrecy of research ideas, and be able to verify to themselves and prove later to others that their submissions are properly handled by the system; at least that any errors should be detectable without unnecessary delay. Also desirable to the honest reviewer is that reviewer anonymity is upheld.

In this paper, we analyze the web-submission-and-review system (known as WSaR from here onwards) [10] developed by Shai Halevi from IBM Research. WSaR is currently in use by top cryptology conferences including Eurocrypt 2007 & 2008, Crypto 2007 and Asiacrypt 2007, annually sponsored by the Inter-

---

<sup>4</sup> Association of Learned and Professional Society Publishers

national Association for Cryptologic Research (IACR) [11]. See Appendix B for a longer list.

## 2 WSaR and its Security Features

WSaR is open source and is hosted at SourceForge [23]. While analyzing its HyperText Preprocessor (PHP) scripts, we found some security features that have been designed into WSaR to protect against several common Web attacks. This section will analyze how the features are added and the type of attacks the features defend against.

### 2.1 Password Strength

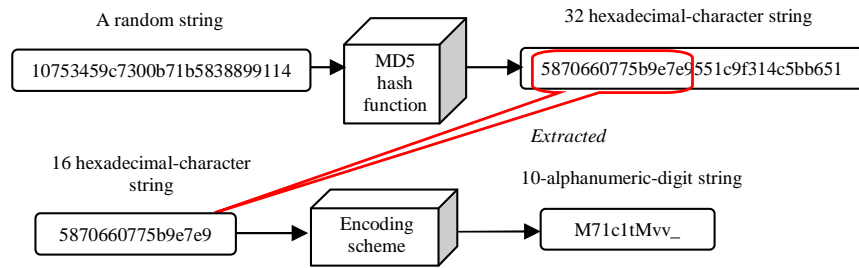
Passwords are often the first defense against intrusion. Relative to online submission and review systems, passwords are the first fortification to ensure the quality of conference proceedings because they are used to safeguard the submissions and their reviews.

In order to gain initial access to the administration or review sites, WSaR computes and generates unique passwords for the conference Chair and reviewers respectively. Firstly, after the customization phase, the Chair will be given a 10-alphanumeric-character password to log in to the administration site. Once logged in, he has the option to change the default password.

The same goes to the reviewers (in most security conferences, a reviewer with access to the online system is called a program committee member) - soon after the Chair grants the reviewers access to the review site, they will receive a notification email that gives them the password to log on to their own review sites. Here, they will be able to view the list of submissions (for which they have no conflict of interest), change their reviewing preferences, post their reviews, participate in paper discussions or take part in a ballot. On the other hand, throughout the submission phase, a distinct submission-ID and password will be generated for every paper. Authors will need these parameters to revise or withdraw the papers. However, they do not have the option to change the submission password.

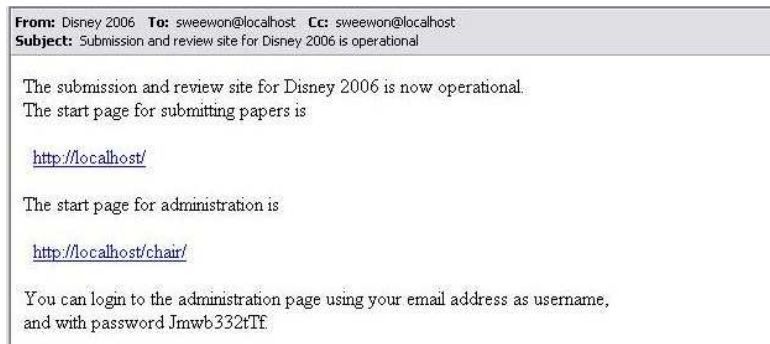
These “WSaR-generated” passwords are 10-alphanumeric-character strings. Each character is either uppercase (A-Z), lowercase (a-z), numerals (0-9), or sometimes a tilde (~) or an underscore (\_). Figure 1 illustrates how passwords are generated in WSaR:

1. For the Chair and PC members: A long random string is generated using PHP functions such as `uniqid()`, `mt_rand()`, and `rand()`. This random string has length between 15 to 28 digits.  
For the submissions: A long random string is generated using the similar functions as above. This string is then appended by the submission’s title and the author’s name.
2. The hash of the resulting string is then computed via PHP’s MD5 function.



**Fig. 1.** Password generation process for the reviewer

3. The first 16 hexadecimal digits of the resulted message digest are extracted.
4. A custom WSaR encoding function is used to compress these extracted digits into a 10-digit alphanumeric string.
5. The resulting string (which is the password) will be emailed to the user (see Figure 2).



**Fig. 2.** An example email where generated password is emailed to the conference Chair

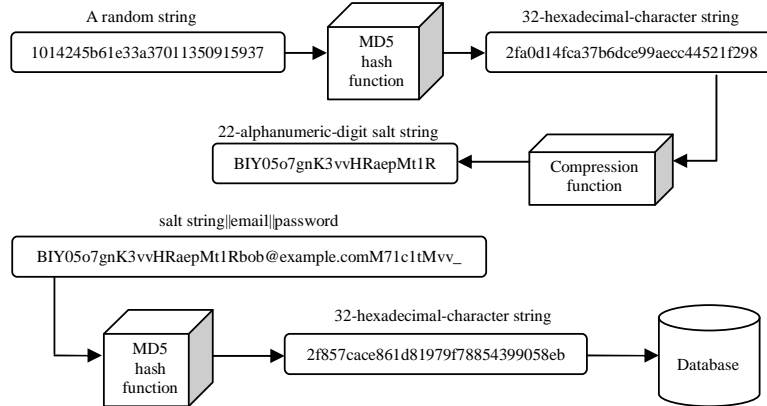
Note that an alphanumeric character corresponds to 6-bit entropy, so the entire 10-alphanumeric-character string requires an exhaustive effort of  $2^{60}$  to brute force. Therefore, the passwords generated by WSaR are deemed to be secure.

## 2.2 Password Storage for Conference Chair and PC Members

No matter where passwords are stored on the online system server, they should be stored in encrypted or hashed form, similar to multi-user operating systems like Unix, Linux etc. The most popular way to seal passwords in a database is via password hashing. There are currently two most commonly used hash functions, namely the Secure Hash Algorithm-1 (SHA-1) and the Message Digest 5 (MD5).

MD5 produces a hash that is 128 bits long (equivalent to 32 hexadecimal characters) while SHA-1 computes a hash that is 160 bits long (equivalent to 40 hexadecimal characters). Among these two, MD5 is the more commonly used hash function to safeguard passwords as well as to ensure message or software integrity. Likewise, this function is also employed in WSaR when it comes to storing passwords.

Here, we look at how WSaR protects the Chair and PC members' passwords stored in the system database (see Figure 3):



**Fig. 3.** Generation of salt string and password storing process

1. Upon customization, a random string is generated using PHP functions such as `uniqid()`, `mt_rand()`, and `rand()`. The string's message digest is computed and the digest is then compressed using a custom WSaR function into a 22-alphanumeric-character salt string. This salt string is saved and is constant throughout the entire conference.
2. The user's email address and password are retrieved.
3. The salt string is then appended by the user's email address and password, and it is fed to the MD5 hash function.
4. The resulting digest is stored into the database table (see Figure 4).

```
mysql> select revId,revPwd,name from committee;
+-----+-----+-----+
| revId | revPwd | name |
+-----+-----+-----+
| 1 | e81bf9d3b0cf5bc462f41b0221fbc0b9 | Disney2006 Chair |
| 2 | 8285ced9fc7aa1339dcf0caf93e76fa | John Smith |
| 3 | ec739037e723790de8b4f30212e82c2d | Jane Doe |
+-----+-----+-----+
```

**Fig. 4.** The digests are stored in the database instead of the passwords

In this case, users' passwords are not stored in the clear in the database, and it is infeasible for an attacker to reverse on the hash values to retrieve the pre-images (passwords). In Section 3.5, we will discuss an issue with the storage of submissions' passwords.

Furthermore, this technique of appending a salt string to the email address and password before hashing it increases the difficulty in cracking the passwords using brute-force attack even if an attacker has access to the hash table. In Section 3.4, we discuss how using different salt strings (instead of a constant one) can increase the difficulty in cracking users' passwords.

### 2.3 Input Sanitization

As discussed in [8], SQL injection is considered one of the most dangerous threats to Web applications because it allows an attacker to connect to the back-end

database and extract any data as he wants. An example of an SQL injection attack is the log in form where a user enters his username and password to be authorized, and the server will retrieve the user’s ID and credit card number, for example. Generally, the SQL query is shown in Table 1:

**Table 1.** SQL statement to retrieve user’s ID and credit card number

SELECT	ID, CREDIT_NUM
FROM	users
WHERE	username = '\$username'
AND	password = '\$password'

Assuming an attacker enters “Jane” in the username field and provides the string “anything’ OR ‘a’=‘a” in the password field. The SQL query would become:

**Table 2.** SQL statement as “modified” by the attacker

SELECT	ID, CREDIT_NUM
FROM	users
WHERE	username = 'Jane'
AND	password = 'anything' OR 'a'='a'

The ‘a’=‘a’ part is always true regardless of what the first part of the query contains, thus the attacker would be able to trick the application to obtain database data that is not supposed to be returned by the application. Successful SQL injection attack will also result in authentication bypass and database modification [15].

The “one rule” to defend against SQL injection (as well as cross-site scripting, buffer overflows etc.) is input sanitization. If this is done, the Web application will be 80 per cent more secure.

There are two commonly used ways to validate input: (1) strip off any undesirable characters (such as meta-characters) and (2) check input data for expected data type [12]. Both ways are implemented in WSaR. We note that the potential SQL injection characters include: (“\*;&<>/’^”) [7]. In WSaR, a possible exploitation of special characters for an SQL Injection attack is in the “Submission/Revision Receipt” page where, as an example, the receipt page’s URL for submission A with password ‘ABC’ will be “<http://localhost/receipt.php?subId=A&subPwd=ABC>”. In this case, query to the database will be as in Table 3 (as an example).

**Table 3.** SQL statement to retrieve submission with subId=A and subPwd=ABC

SELECT	title, authors, abstract
FROM	submissions
WHERE	subId='A'
AND	subPwd='ABC'

Firstly, WSaR uses the `my_addslashes()` function in PHP to remove undesirable characters. If an attacker happens to be one of the submitters to the conference, she would know the pattern of the receipt page's URL. Now, she is interested in finding out the paper submitted by her rival, so she launches an SQL Injection attack on the receipt page by changing the URL to "`http://localhost/receipt.php?subId=1&subPwd=1'%20R%20'1'='1'`", where `%20` represents a space in its HTML entity. This time, the query will be as follows:

**Table 4.** SQL query for an SQL Injection attack

SELECT	title, authors, abstract
FROM	submissions
WHERE	subId='1'
AND	subPwd='1' OR '1'='1'

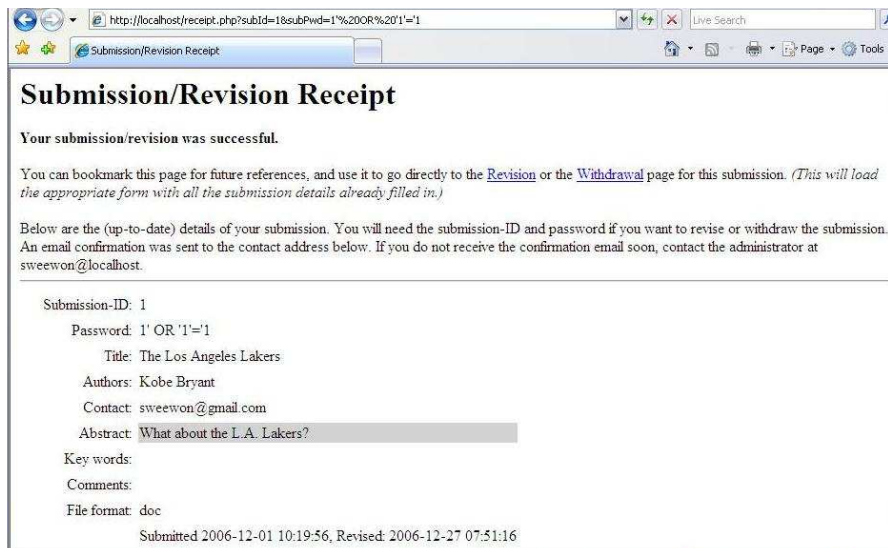
However, since all special characters are removed by the `my_addslashes()` function - single quotes are ignored using backslashes (commented out), the `'1'='1'` part no longer makes sense. Thus, the attacker will receive a generic error message as shown in the screen shot in Figure 5.



**Fig. 5.** Attacker receives an error message

If the software does not sanitize user's input in all PHP scripts (i.e. all `my_addslashes()` functions are removed from every WSaR scripts), the URL constructed by the attacker will return submission 1's receipt page and from there, she can redirect to the revision page and revise her rival's paper or even withdraw it from the conference without her rival knowing it (see Figure 6).

Secondly, developers should specify the type of input that is expected for certain form fields, and that the unexpected input will be removed. As an example, a form field that requests for user's phone number should accept only numbers as input. WSaR always has a specific input type that is expected for the submission ID - it processes only integers for the submission ID field. In spite



**Fig. 6.** An attacker can click on the Revision or Withdrawal link to revise or withdraw the submission

of that, developers can make use of regular expressions to tell the program the type of pattern in the text that it should look for [19]. If the data submitted does not match the regular expression, it will be ignored [7] or error messages will be generated.

Both the above mentioned methods are employed in WSAE. Therefore, this system is not categorized as one of the 60 per cent of Web applications that is vulnerable to SQL injection (or other attacks due to invalidated input) [24].

## 2.4 Resistance to Bypass of Access Control Checks through Forced Browsing

Forced browsing refers to a technique used by attackers to access resources that are not referenced, but are nevertheless accessible [25]. Access control checks are normally performed after a user gets authenticated and it monitors what authorized users are allowed to do. As an example, if a reviewer is blocked from reviewing certain submissions due to conflict of interest, he should not be able to bypass the access control checks by requesting the review form of that submission directly in the URL.

Taking a look at the URL of a review form; to review submission A, reviewer will access the review form at the URL "http://localhost/review/review.php?subId=A" (see Figure 7).

Assume that the insider attacker is one of the reviewers in a conference and she is blocked from reviewing submission 1 since she is one of the authors of that submission. In order to make sure that her submission is accepted to the conference, the attacker needs good reviews for her paper. Thus, she tries to change submission A's review form URL to which she has access, from "http://localhost/review/review.php?subId=A" to "http://localhost/review/



Fig. 7. Review form for submission 1

review.php?subId=1” since she is prohibited to access the link directly from her review page. If the system then displays the review form for submission 1, the attacker has bypassed the access control checks through forced browsing.

Fortunately, the attempt to bypass WSaR’s access control check is forbidden in the review form page, as well as the voting page and the reviewers’ discussion forum. Whenever a reviewer attempts to access a blocked submission by directly specifying the submission-ID in the URL, WSaR will firstly perform an authorization check in the reviewer table; if the reviewer is blocked from that submission, it will display an error message indicating that the submission is not found, or the reviewer has a conflict as shown in Figure 8.



Fig. 8. WSaR prevents broken access control

### 3 Security Issues and Enhancements

In addition to the security features already designed into WSaR discussed in Section 2, we have also discovered some security issues not treated in WSaR and in this section, we discuss them in detail and then describe ways to enhance WSaR security by taking them into consideration.

For any security issue, we will discuss its implications from the two opposing perspectives motivated in Section 1. We also highlight whether exploitation of issues can be traceable or uniquely pointing the finger to the culprit. This has devastating consequences if the attacker cannot be traced since it means even a curious (if not malicious) researcher from the scientific community could have mounted the attack without any counter-incentives i.e. no adverse effects on his reputation. Furthermore, issues that lead to attacks for which the culprit cannot be unambiguously accused will cause disputes for which a malicious attacker could deny his involvement or an honest user be unfairly thought by peers to have mounted an attack.

### 3.1 Browser Caching

Browser caching is categorized as one of the critical areas in OWASP's Top Ten projects under "Broken Authentication and Session Management" [25].

In WSAr, the submission-ID and password are sent by the HTTP GET method. Information sent in such way will be displayed in the browser's URL [18] and it is extremely undesirable (see Figure 9).

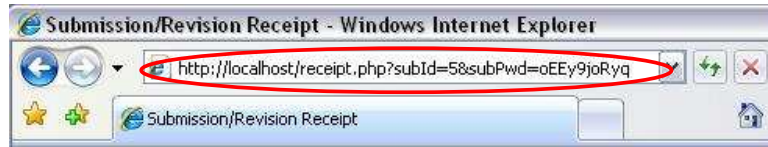


Fig. 9. Submission-ID and password are displayed in the page's URL

This means the submission-ID and password are part of the URL. As has been highlighted in [25], authentication and session data should not be submitted as part of a GET to prevent someone malicious from using the "Back" button in an authorized user's browser to backup the page and hence obtain the password from the URL.

Alternatively, even if a browser window is closed, the URL info can also be obtained from the browser's cache and history. Recall that browsers *cache* most of the contents of frequently visited pages so that the pages will load faster the next time they are visited, including images, sounds, cookies, web pages and their URLs. Information stored in browsers' cache is not encrypted [1] and it can be obtained by anyone who accesses the computer. Both Netscape Navigator and Mozilla Firefox, by default require users to clear the browser's cache (and disk cache - also known as "Temporary Internet Files") manually. On the other hand, Internet Explorer will clear the browser's cache but not the "Temporary Internet Files", once the browser is restarted. In addition, previously visited URLs are typically stored in the browser's *history*; for instance the latest versions (at the time of writing) of Netscape Navigator and Mozilla Firefox by default will only clear browser's history every nine days, while the default for Internet Explorer is 20 days.

Figure 10 shows the receipt URLs stored in the "Temporary Internet Files" folder. In spite of that, if the user does not clear the browser's cache or history after he uses the computer, the URLs will also be displayed in the browser's history pane as shown in Figure 11.

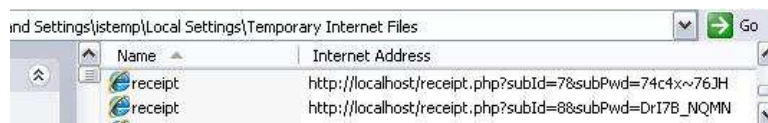
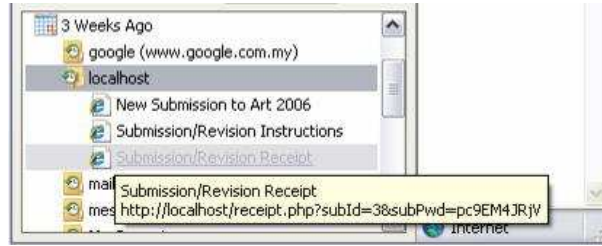


Fig. 10. Internet Explorer's temporary internet files that stores the receipt page's URL



**Fig. 11.** Submission ID and password are exposed to the attacker

This threat can be launched by any individual having access to a common machine previously utilized by a WSaR user, and upon retrieving the login information (ID and password), he can login as the WSaR victimized user. This attack is non-traceable in the sense that the attacker cannot be later pinpointed, so he can mount the attack without any risk of jeopardizing his reputation. Thus, the counter-incentive is non-existent that it will indeed be tempting for any individual to abuse this issue to the victim's disadvantage.

Browser caching can be prevented by submitting the submission-ID and password as part of a HTTP POST method [25] instead. If the POST method is used, we can employ the PHP's predefined variable "\$\_POST" to retrieve the values entered in the submission-ID and password fields respectively. In this case, both submission-ID and password will not be displayed in the page's URL.

Other than this, WSaR users are advised to clear the browser's cache before they leave the computer. In Netscape Navigator, users would have to clear both the memory and disk cache; in Mozilla Firefox, users should check all fields when clearing their private data [4]; in Internet Explorer, users should clear the history and all temporary internet files [21]. This way, users can ensure the security of their private information.

### 3.2 Constant Salt String for Reviewer and Chair Passwords

In Section 2.2, we discussed how WSaR uses a salt string appended to the reviewer's email address and password to build a stronger defense in securing his passwords. However, this salt string is constant throughout the entire conference for any party, thus all users of the same conference will have the same salt appended to their password. So what differentiates each user is just the email address (which is typically public information) and his password. This indicates that the salt string does not really provide better strength against insider attackers (users of the same conference) than conventional password-based authentication systems. Therefore, we recommend the use of different salt strings for different users.

These salt strings will be stored in a single PHP script upon customization of WSaR, and they will be labeled in the sense that "SALT\_2" will be used for PC member with the ID of 2 and so on. In this case, since the salt strings will be random and of different lengths, the attacker would have a much harder time

trying to guess the exact salt that is appended to a specific user’s email address and password.

Again, this threat is non-traceable as it allows a malicious insider individual to brute-force the password, upon which he can login as the victim without any evidence pointing back to him.

### 3.3 Storage of Submission Passwords

Throughout our analysis, we discovered that although WSaR stores the hashes of reviewers’ passwords, it does not do the same when it comes to storing submissions’ passwords (see Figure 12). For someone who manages to gain access to the database, he has the ID and password for every submission as well. We recommend that the submission passwords to be salted and hashed as well to secure them so that even if a hacker manages to penetrate the database’s security, he would face the prospect of a potentially expensive search for the exact password.

```
mysql> select subId,title,subPwd from submissions;
+----+-----+-----+
| subId | title                                     | subPwd |
+----+-----+-----+
| 1 | Hardening Network Security              | 1yZrHxZs4o |
| 2 | Multiband PIFAs (planar inverted-F antenna) for Internal Mobile Phone Antennas | KluGRUFaCb |
| 3 | Implementation of Cryptographic Pseudorandom Generator in 8-bit Microcontroller | fCbK_Mkf1_ |
| 4 | Intelligent Health System              | ZEqje44Rk |
| 5 | Trellis coding                          | 2H0xeSxMcC |
| 6 | Chess-Playing Robot                     | Ady4i4D0_8 |
| 7 | Intelligent Car Alarm System            | x08qBfvk3q |
+----+-----+-----+
7 rows in set (0.00 sec)
```

Fig. 12. Submissions’ passwords are stored in the clear in the database

### 3.4 Password Policy and Strength Checking

Using a good password by every user is vital to defending a system. The components of a good password should be at least eight characters long, should not consist of dictionary words (would be vulnerable to dictionary attack otherwise), should never be the same as the user’s log in name, should not consist of any item that is easily identified with the user, and have at least three of the following elements [5]:

- One or more uppercase letters (A-Z)
- One or more lowercase letters (a-z)
- One or more numerals (0-9)
- One or more special characters or punctuation marks

However, it is worth noting that some systems do not permit the use of certain special characters in user’s password string.

Although the passwords supplied by WSaR fulfill all the mentioned requirements, users might find the password hard to remember and opt to change it. Therefore, we analysed WSaR’s password change mechanism. WSaR does not have any password policies imposed to ensure the strength of new passwords, thus a careless user may happen to employ a password which is easily cracked

by any password-cracking tools, without being reminded not to do so. Hence, we suggest that WSaR should force its users to adopt passwords that cannot be easily cracked. Google mail [9] uses a scale to show its users the strength of their passwords with parameters such as “Too short”, “Weak”, “Fair”, “Good” or “Strong”. This is a feature that could be added to WSaR.



**Fig. 13.** Google mail’s password strength evaluation

However, careless users tend to ignore the password strength evaluation and proceed to employing the new password. Consequently, the system’s security could still be breached.

Having said that, WSaR’s password changing mechanism should perform an assessment on the new password - whether it is too short or it contains dictionary words or even a row of letters from a standard keyboard layout (e.g., ertyui) [14]. If a user’s new password happens to be a weak password, a pop up window should be issued to require him to re-enter a new password.

Besides that, a password policy should also be implemented to require frequent change of passwords. If an attacker uses brute-force attack to crack a password, it is possible that he would be taking a long time (depending on the length of the password, speed of both the machine and network connection) to complete the attack [14]. In this case, if the user changes his password frequently, he could avoid his password from being cracked via brute-force attack.

### 3.5 Absence of File Integrity and Binding

Whenever we download any files or software, the first thing we would want to do is to check the file’s integrity. Presently, the most popular method used to verify a file’s integrity is via the use of the MD5 function. If the file content is modified, it can be easily detected by re-computing its hash value. In an online submission and review system, calculating a submission’s message digest is important both to ensure message integrity and to bind the author to the submitted paper due to absence of face-to-face communication.

We note that WSaR does not have this feature, i.e. upon submission the file’s message digest is not computed and thus not supplied to both the author and the Chair. Therefore, considering the case of malicious users, if an author happens to submit a not-so-perfect paper in order to meet the submission deadline, he can later deny that he submitted that version of the paper, then claim that the file is corrupted and request for a second-time submission, thus gaining advantage over other authors. More devastatingly, an honest author could have submitted a proper paper but the file became corrupted by the server machine; in which case it is desirable to be able to unambiguously prove that the corrupted version is not the submitted file, or at least be able to check during submission that it was properly received. This avoids ambiguity when a Chair notes during review phase that a file is corrupted and is unsure if it was intentionally submitted

in that form in order to buy authors some time, or if it was indeed submitted properly but was corrupted in transit.

Indeed, this issue is often not so much to guard against malicious authors but rather so as to allow a scientifically honest author in this situation to be able to prove his innocence beyond any doubt.

To counter this issue, we recommend that authors be presented with a 128-bit hash of his submission file and that this hash value is kept as a record for the Chair. Since WSaR is developed in PHP, we can apply the “`md5_file()`” function where it calculates the message digest of the given file [18].

Firstly, a new column has to be created in the “`submissions`” table to record the message digest by adding a “`msgDigest varchar(255) BINARY NOT NULL`” statement in the “`create_table( )`” function, which can be found in WSaR’s `database.php` script. Subsequently, we added a statement to calculate the file’s message digest in the `act-submit.php` and `act-revise.php` scripts, which are used in WSaR to process all parameters entered in the submission and revision form respectively. The resulted digest is inserted into the database under the “`msgDigest`” column and authors will be redirected to the receipt page. An email will also be generated to the author, and carbon-copied to the Chair with all submission details listed (including the message digest). Now, we can be assured of the file’s integrity as well as binding the author to his submission. The screenshots are shown as Figure 14 and Figure 15 in Appendix A for further illustration.

## 4 Protocol Sketch for Password Distribution via Email

In general (not specific to WSaR), passwords for reviewers are commonly sent via emails to the reviewers’ email addresses. This is indeed the most practical way to distribute passwords, although it is known that email formats by default (and this is the setting that users commonly use) do not provide any form of confidentiality nor authenticity, unless explicit email clients or plug-ins like PGP are applied.

We take what we view as a concrete step towards securing online systems by motivating here and sketching the basic idea of our ongoing work: a proposal for an email-based password distribution protocol for security conferences. Having this kind of protocol in place will prevent reviewer passwords from being easily compromised through attacks mounted not on the system itself but on how this password is distributed. Indeed, it is well known that the study of password-based key exchange protocols is a long-standing research topic in cryptology, thus we should avoid having security conferences using in practice the email-based password distribution protocols that are not securely designed.

The setting for an email ID-based password distribution protocol is different from those in typical key exchange protocols [3]. The protocol involves two parties, the program chair  $C$  and the reviewer  $R$ . Rather than requiring any public-key infrastructure (PKI), only a trusted web site bulletin board is used, for instance the IACR website (<http://www.iacr.org>), where URLs for differ-

ent IACR conferences or workshops are hosted or mirrored. On this website is listed the email address  $ID_C$  of the program chair. Meanwhile, it is common that the chair invites program committee members (the reviewers) who are experts in the field and for whom the chair knows the authentic email addresses  $ID_R$  either himself, or for which he can ask from other experts through some out of band mechanism. Alternatively, the email addresses can be obtained from the IACR membership database.

Thus, when a reviewer  $R$  receives an email from the chair  $C$ , it can check to be certain that the email came from the chair; vice versa a chair knows if an email came from a particular reviewer. This provides source authentication without resorting to any PKI.

Then a general sketch of this kind of protocol proceeds as follows:

1.  $C$  generates an ID-based public and private key pair based on his  $ID_C$ . Denote these as  $pk_C$  and  $sk_C$ . Then  $C$  computes the message  $sig_{sk_C}(ID_C, ID_R, \text{INVITE})$  where  $sig_{sk}$  denotes signing under a person's private key  $SK$ , and INVITE contains a one-way (e.g. hashed) representation of a typical invitation email stating the conference name etc.
2.  $C$  sends  $m_1 = sig_{sk_C}(ID_C, ID_R, \text{INVITE})$  to  $R$ .
3.  $R$  generates an ID-based public and private key pair based on his  $ID_R$ . Denote these as  $pk_R$  and  $sk_R$ . Then  $R$  computes the message  $sig_{sk_R}(ID_R, ID_C, \text{ACCEPT})$ .
4.  $R$  sends  $m_2 = sig_{sk_R}(ID_R, ID_C, \text{ACCEPT})$  to  $C$ .
5.  $C$  generates a password  $pwd_R$  for  $R$ .
6.  $C$  sends  $m_3 = \langle sig_{sk_C}(ID_C, ID_R, m_1, m_2, Enc_{pk_R}(pwd_R)), Enc_{pk_R}(pwd_R) \rangle$  to  $R$ , where  $Enc_{pk}(\cdot)$  denotes encryption under a person's public key  $PK$ .
7.  $R$  obtains  $pwd_R$  by decrypting  $Enc_{pk_R}(pwd_R)$ .

In fact, this can be a concrete step to a long-term setting where one submission and review system is used for all conferences that use WSaR, so only a one-time setup cost e.g. the ID-based password distribution protocol as above, is incurred for new users, while existing users can update their passwords online through the system at any time; and new membership in conference program committees of existing users only require the program chair to email to a PC member  $R$  asking him to update his password himself rather than having to email newly generated passwords every time  $R$  is involved in a new conference program committee. Indeed, this centralization is possible since conferences using WSaR are typically hosted on a central machine e.g. at <http://s1.iacr.org>, unlike some other submission software that need to be locally set up. Furthermore, conferences using WSaR commonly involve program committee members who are involved in multiple conferences so the setup cost will be amortized over time and this centralization makes sense compared to treating each conference system separately.

We emphasize here that the above is a sketch of a protocol design that we are currently working on, whose formal security we are in the process of proving. This should therefore preclude the sprouting of future papers that propose informal "breaks" on the above enumerated steps. The above sketch should only be taken as a basic template and taken for the general idea that it is sketching and nothing

more. We do welcome comments for which will be acknowledged in future work, or collaborations in this direction.

## 5 Concluding Remarks

In this paper, we have seen that WSaR is built with a strong defense against a number of known attacks in the Web. We have also discovered and discussed the absence of several features that could jeopardise the security of WSaR users and we proposed suggestions to further enhance WSaR's security that address these issues. As a side remark, we recommend that besides strengthening the defense of the software itself e.g. WSaR, Web administrators should secure the Web server as well as the back-end database and constantly monitor the activities going on in the Web server to detect any malicious behaviour.

In addition, as a further step to securing these types of systems, we have also motivated the design of an email-based password distribution protocol for WSaR users and argued that together with such a design, there are advantages in centralizing conferences that use WSaR.

## References

1. AICT Security - Empty your Cache. Available online at <https://www.ualberta.ca/AICT/Security/BrowserCache.html#private>
2. Archer, T. Are Hash Codes Unique?. Available online at <http://blogs.msdn.com/tomarcher/archive/2006/05/10/594204.aspx>
3. Bellare, M., and Rogaway, P. Entity Authentication and Key Distribution. Advances in Cryptology - CRYPTO '93, LNCS, Vol. 773, pp. 232-249, 1993
4. CIBC - Clear Your Browser's Cache. Available online at <http://www.cibc.com/ca/legal/clear-browsers-cache.html>
5. Conklin, W.A., White, G.B., Cothren, C., Williams, D., and Davis, R.L. Principles of Computer Security: Security+ TM and Beyond. McGraw-Hill, 2005.
6. EasyChair Conference System. Available online at <http://www.easychair.org/>
7. Foster, J.C. Defense Tactics for SQL Injection Attacks. Available online at [http://searchappsecurity.techtarget.com/tip/0,289483,sid92\\_gci1219912,00.html](http://searchappsecurity.techtarget.com/tip/0,289483,sid92_gci1219912,00.html)
8. Fyre, C. One Simple Rule to Make your Web Apps more Secure, 2006. Available online at [http://searchappsecurity.techtarget.com/qna/0,289202,sid92\\_gci1225425,00.html](http://searchappsecurity.techtarget.com/qna/0,289202,sid92_gci1225425,00.html)
9. Google Mail. Available online at <http://gmail.google.com>.
10. Halevi, S. Web Submission and Review Software. Available online at <http://theory.csail.mit.edu/~shaih/websubrev>
11. IACR Conferences. Available online at <http://www.iacr.org/conferences/>
12. McClure, S., Shah, S., and Shah, S. Web Hacking: Attacks and Defense. Addison-Wesley, 2003.
13. Microsoft Corporation. Microsoft's Conference Management Toolkit. Available online at <http://msrcmt.research.microsoft.com/cmt/>
14. Password Cracking: Information from Answers.com, 2006. Available online at <http://www.answers.com/topic/password-cracking>

15. Peikari, C., and Chuvakin, A. Security Warrior. O'Reilly, 2004.
16. Phan, R.C.-W., and Goi, B.-M. Flaw in IEEE Trans on Consumer Electronics Online Submission System. *Presented at the rump session of the International Conference on Cryptology in Malaysia (Mycrypt '05)*, 2005.
17. Phan, R.C.-W., and Ling, H.-C. On the Insecurity of the Microsoft Research Conference Management Tool (MSRCMT) System. *Proceedings of International Conference on IT in Asia (CITA '05)*, pp. 75-79, 2005. *Also presented at the rump session of Asiacrypt 2004, Jeju Island, Korea.*
18. PHP Manual. Full version available online at <http://www.php.net/manual/en/>
19. Regular Expressions, 2006. Available online at [http://searchappsecurity.techtarget.com/sDefinition/0,290660,sid92\\_gci517740,00.html](http://searchappsecurity.techtarget.com/sDefinition/0,290660,sid92_gci517740,00.html)
20. ScholarOne, Inc. Manuscript Central: About Manuscript Central. Available online at [http://www.scholarone.com/products\\_manuscriptcentral\\_aboutMC.shtml](http://www.scholarone.com/products_manuscriptcentral_aboutMC.shtml)
21. Security Information: Clearing Browser Cache and History. Available online at [http://www.hlasset.com/files/Clearing\\_Cache\\_History.pdf](http://www.hlasset.com/files/Clearing_Cache_History.pdf)
22. SoftConf.com - Software for Conferences. Available online at <http://www.softconf.com/index.html>
23. SourceForge.net: Web Submission and Review Software. Available online at <http://sourceforge.net/projects/websubrev>
24. What is SQL Injection?, 2006. Available online at [http://searchappsecurity.techtarget.com/sDefinition/0,290660,sid92\\_gci1003024,00.html](http://searchappsecurity.techtarget.com/sDefinition/0,290660,sid92_gci1003024,00.html)
25. The Ten Most Critical Web Application Security Vulnerabilities, 2004. Available online at <http://osdn.dl.sourceforge.net/sourceforge/owasp/OWASPTopTen2004.pdf>
26. Ware, M. Online Submission and Peer-Review System, 2005. Available online at [www.zen34802.zen.co.uk/Learned\\_Publishing\\_offprint.pdf](http://www.zen34802.zen.co.uk/Learned_Publishing_offprint.pdf)

## Acknowledgement

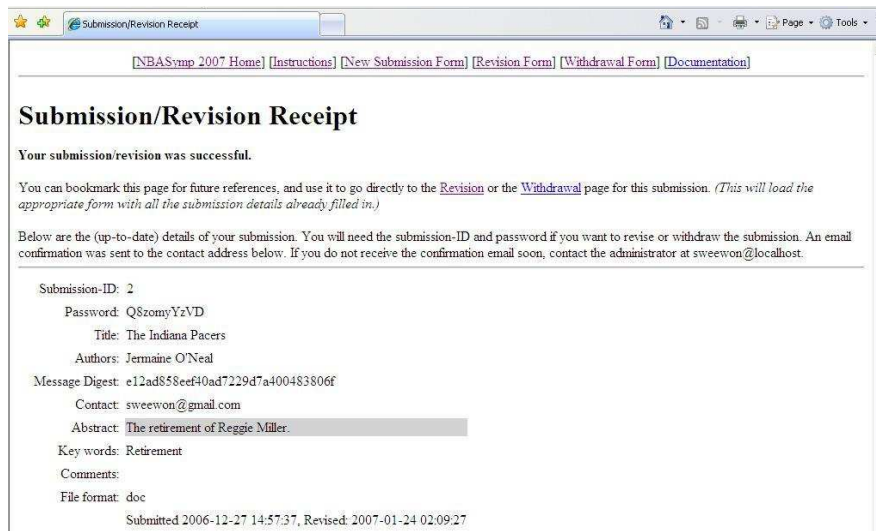
The first author thanks Shai Halevi for encouragement during the initial stages of this research, for patiently answering queries about WSaR and for detailed comments on an early version of this paper. The second author thanks Thomas Baignères for stimulating discussions on another popular submission and review system iChair. Nous avons eu un temps merveilleux pendant la pause de café, ou bien? We thank an anonymous referee for motivational comments especially on the need to make clear the distinction between pro- and counter-incentives for an attacker; and for acknowledging the fun of this research work :-)

## A Storage and Display of Submissions' Digests

We present below the screen shots of the “submissions” table, and the submission/revision receipt after the calculation of submission’s digest is incorporated.

```
mysql> select subId,title,authors,msgDigest from submissions where subId=2;
+-----+-----+-----+-----+
| subId | title           | authors      | msgDigest                                     |
+-----+-----+-----+-----+
| 2     | The Indiana Pacers | Jermaine O'Neal | e12ad858ef40ad7229d7a400483806f |
+-----+-----+-----+-----+
```

Fig. 14. Digest of submission is stored in the database for Chair’s reference



**Fig. 15.** Digest of submission is presented in the receipt for the author

## **B Conferences that have Used or are Using WSAr**

In reverse chronological order:

1. EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques
2. CT-RSA 2008: RSA Conference 2007, Cryptographers' Track
3. LATIN 2008: 8th Latin American Theoretical Informatics
4. TCC 2008: 5th Theory of Cryptography Conference
5. PKC 2008: 11th International Conference on Theory and Practice in Public-Key Cryptography
6. ASIACRYPT 2007: 13th Annual International Conference on the Theory and Application of Cryptology and Information Security
7. ISC 2007: 10th Information Security Conference
8. CRYPTO 2007: 27th Annual International Cryptology Conference
9. ICALP 2007: Track C of the 34th International Colloquium on Automata, Languages and Programming
10. GOCP 2007: 1st International Workshop on Group-Oriented Cryptographic Protocols
11. ACNS 2007: 5th International Conference on Applied Cryptography and Network Security
12. EUROCRYPT 2007: 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques
13. USEC 2007: Usable Security Workshop
14. TCC 2007: 4th Theory of Cryptography Conference
15. CT-RSA 2007: RSA Conference 2007, Cryptographers' Track
16. HVC 2006: 2nd Annual Haifa Verification Conference
17. PKC 2006: 9th International Conference on Theory and Practice of Public-Key Cryptography

## C Related Work

Many online paper submission and review systems are emerging. For the context of cryptology and information security, the predecessor to WSaR is the collection of PHP/Perl scripts written progressively by Chanathip Namprempre, Andre Adelsbach, Andrew Clark and the Computer Security and Industrial Cryptography (COSIC) group at Katholieke Universiteit Leuven.

These scripts were used for almost all mainstream cryptology and information security conferences till 2006 when building on ideas in these scripts, two successor systems were developed independently: WSaR by Shai Halevi of IBM Research and iChair by Thomas Baignères and Matthieu Finiasz of LASEC at EPFL. These two systems are now used by almost all mainstream cryptology and information security conferences.

A few other major submission and review systems used in other fields deserve some mention here:

1. Manuscript Central

Manuscript Central [20], developed by ScholarOne, Inc., is the online submission and peer review system used to handle manuscript submissions to journals. It manages over 44,000 submissions per month and its comprehensive and user-friendly features result in reports that most journals using Manuscript Central achieve gains in submissions of 20 to 40 per cent per annum. This system is currently used by most IEEE and Association of Computing Machinery (ACM) journals. Manuscript Central is a fully-developed software with 24-hour support on weekdays. By contacting the sales representative, one would be able to obtain and understand the system's functionalities, features, pricing and get the system running in two weeks' time.

2. Microsoft Research Conference Management Tool (MSRCMT)

Firstly developed for ACM SIGKDD 1999, the MSRCMT [13] is an academic conference management service sponsored by Microsoft Research. Surajit Chaudhuri, a Research Area Manager at Microsoft Research is the architect of MSRCMT. Since the year 1999, this system has been used in over 500 conferences, among them are the International Conference on Security of Information and Networks (SIN 2007) and the ACM SIGCOMM 2007 Data Communication Festival. Similar to Manuscript Central, the MSRCMT is also a fully-developed system. It is free and hosted by Microsoft Research, but with limited support since it is developed and managed by a small team.

3. EasyChair

Developed in the year 2002 by Andrei Voronkov, a Professor from the University of Manchester, EasyChair [6] is used by over 600 conferences in year 2007 alone. EasyChair is free and it is currently hosted by University of Manchester's Computer Science Department. EasyChair is capable of supporting two models: (1) the standard model for conferences having one program committee and (2) the multi-track version for conferences having multiple tracks that have their own program committee. There are a number of ACM and

IEEE conferences/workshops that have used or are using EasyChair. Among them are the 8th IEEE/ACM International Conference on Grid Computing (Grid 2007), the 2nd ACM Workshop on Scalable Trusted Computing (STC'07) and the 20th IEEE Computer Security Foundation Symposium (CSF 20).

4. START V2 ConferenceManager

START V2 [22], written by Rich Gerber, is a product from SoftConf.com. Apart from EasyChair, several IEEE and ACM conferences have, or are employing START V2 as the submission and peer review system since the year 2002. The IEEE Symposium on Intelligence and Security Informatics (ISI 2007), the 2007 IEEE Symposium on Security and Privacy, the 5th ACM Workshop on Recurring Malcode (WORM 2007) and the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007) are among the many IEEE and ACM conferences using START V2. Users would need to contact the developer to order the system; the pricing depends on whether one needs the system to be hosted at `softconf.com`, and on the different types of licensing arrangements. This system is constantly improving based on the users' feedback.

It is worth to note two earlier work related to the security of online submission and review systems, although our work here is the first detailed analysis of this type of system. Phan and Goi [16] pointed out the lack of privacy in a system used by an IEEE Transactions where the URL of pages that disclose paper information and that allow paper revision for a particular submitted paper, differ from pages of other papers by an ID counter. Thus if the URL to revise author *A*'s submitted paper is uniquely identified by ID 100, then *A* can also view the revision page for the paper submitted right after (resp. before) his, which he knows will be uniquely identified by ID 101 (resp. 99). Correspondence with the administrator of the system obtained the response that this was not a significant issue.

Phan and Ling [17] discovered by accident when submitting their paper to a conference using the Microsoft Research Conference Management Tool (MSRCMT) that the system automatically creates an account for co-authors of a corresponding author who submitted a paper, where email addresses of these co-authors are used as login usernames and the numeric 0 is used as the default password. This applied for any co-author(s) of any paper. Correspondence with developers of MSRCMT obtained the response that this was an exercise of regression testing, though the flaw was present for some months in the actual online system used by several conferences. As of 2005 it was verified that both the above systems no longer exhibit those issues.