ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 12(47), DOI: 10.17485/ijst/2019/v12i47/149364, December 2019*

Audit Trail Generator and Pattern Analyzer for Secure Online Transactions in Academic Institutions

Mark Godfrey D. Torres[1,*] and Consorcio S. Namoco, Jr.[2]

[1]Commission on Higher Education – Region 10, Cagayan de Oro City, Philippines; markgodfrey.torres@gmail.com
[2]University of Science and Technology of Southern Philippines, Cagayan de Oro City, Philippines; consorcio.namoco@ustp.edu.ph

Abstract

Objectives: This study focused on creating a three (3) point Audit Trail Generator (ATG) to be integrated into the system, and a Trail Pattern Analyzer (TPA) that uses the Teiresias algorithm as a data mining technique for analyzing audit trails for secure online transactions in academic institutions. **Methods/findings:** The audit trail can be used to enhance the security through monitoring the pattern of user activities within the bounds of the system. It was shown that the mechanism provided an in-depth monitoring of user activities without degrading the performance regarding the proceedings in doing the secure online transactions. **Application:** The overall compliance status of the study resulted to be higher than the required standards. The design is therefore considered usable and secured.

Keywords: Teiresias Algorithm, Web Mining, Security, Activity Monitoring

1. Introduction

Academic institutions use the internet to provide access to online information systems that support its processes, for example, student and faculty information systems. The need for accessing the internet itself is growing and with such growth and the broadness of its accessibility, the demand to use security mechanisms increases. Academic institutions provide students and faculty services in information dissemination. The rapid growth of the internet would necessitate the use of some securing mechanism to protect users and data.[1]

The rapidly growing interconnectivity of IT systems, and the convergence of their technology, renders these systems increasingly vulnerable to malicious attacks. Universities and academic institutions also face concerns about the security of computing resources and information; however, traditional security architectures are not effective for academic or research environments.[2]

In Refs.,[3,4] Tak et al. conducted studies on four security issues (*non-repudiation*, *integrity*, *authorisation*, and *confidentiality*) in implementing a secure and efficient software framework. Furthermore, designed a software framework for a secure online transaction in academic institution focused on the implementation of these four security issues. With this, the security of the transmission of the data increases. However, the framework lacks the monitoring on the transaction file itself. An example would be a situation on One User-Multiple Session scenarios, especially if the account is shared to multiple individuals; the owner of the account has the chance to deny an activity related to the account, in which also shakes the credibility of the authentication security requirement. Another threat would be the attempts to infiltrate the system through hacking the password of a certain user. Or if the hacking is successful, multiple attempts to decrypt a certain file by an unauthorised person. Another is trying to use access processes within the system without following the standard procedure. A strategy must be incorporated with

*Author for correspondence

the current system to help monitor the activities of the user of the system, thus heighten the implementation of the four mentioned security requirements.

The audit trail can be used to detect unusual or suspicious user actions and identify the specific users who performed those actions. It can also be used to detect unauthorized access attempts, assess potential security damage, provide evidence in investigations if necessary, and provide a passive deterrent against unwanted activities, as long as users know that their actions might be audited[5–8] and according to Omar[9] in information technology and particularly in communication systems, observation of the communication process necessitates the collection of such traces, their classification and even their organisation. We can then use the analysis of the audit trail to monitor the performance of the gateway, for improvement of the process, and for security support. For example, a user would have gained access to another account and will try to download the secured transaction file without currently having the valid private key. Even if the user does not a valid private key, the user can still try to create fabricated keys and try to decrypt the transaction file as many times as possible. If not monitored, such activity could result to a successful decryption.

The objective of this study is to come up with an Audit Trail Generator (ATG) for secure online transaction in academic institution anchored on the study to ensure the careful monitoring of users' actions and address serious security issues, and a Trail Patterns Analyzer (TPA) that can analyse the audit trail generated by the ATG. The ATG and the TPA are designed in the context of the audit trail areas of consideration in this study namely: *definition, structure, capture retention and storage, analysis, reasons of use,* and *context*, which must comply with the four security requirements of *confidentiality, integrity, authorisation*, and *non-repudiation*. The ATG and the TPA are developed using the Teiresias algorithm as a Data Mining Technique. The ATG and the TPA are evaluated in two ways: through the performance of the framework with the ATG; and the usability of the audit trail with its analysis by the TPA.

2. Materials and Methods

2.1. Design and Development of Audit Trail Generator and Trail Pattern Analyzer

The study focuses on the integration of an ATG and TPA to the software framework developed. The framework was designed for the online transaction of electronic versions of academic documents (web browser based) to ensure that the documents retain the factors of secured online transactions such as Confidentiality, Authorisation, Non-Repudiation, and Integrity. These documents are as follows: Transcript of Records, Honorable Dismissal, Grade Release, and Statement of Account. The enhanced program resulted in a software for a secure online transaction for academic institution patterned with the software of without using Joomla, but still using JavaScript, to control user's activities within the system, additional scripts are injected on the original system's major process to activate the *Audit Trail Generator* and a *Trail Pattern Analyzer* module to analyse the audit trail.

The ATG is carefully designed to ensure the following considerations: the audit trail generated is based on its *definition*; the audit trail's design follows the standard *structure* of an audit trail; and *capture retention and storage*; usability for *analysis*; implementable for its *reason/s of use*; and usability on the appropriate *context.*

The main reason for implementing audit trail in the system is to keep track all the activities being done by the user within the system. To maintain the confidentiality of the data within the system, the trail cannot mine the data contents being manipulated in the system. The activities to be monitored are mainly the triggers of user activities within the scope of the framework. These activities are triggered by user events during interaction like clicks on a button or on a link, or a loading of a page itself. The page requests, due to these activities, will be coming from a web browser over a network.

Each record of the audit trail contains information pertaining to the event that happened. Table 1 shows the list of fields of one record in the audit trail. A *SessionID* is added which will later help group the different events. A *Status* field is also added to identify status of an event whether the event has been Initialized, Processing, Failed, or Completed. This field will change depending on the flow of the execution. During the attempt to make the event, the record would automatically be created with the *Status* field set to *Initialized*. The *Processing* status is given after the *Initialized* status is given and the session is valid. Right before the event would end, the same event would be logged but with a different status, whether it would be *Failed* or *Completed* status. The *Failed* status is given only during a caught exception. The *Completed* status is given only if the event finishes without any exceptions. Therefore, each event would be given three logs to

Table 1. The field list of the audit trail record

Field	Description
SessionID	The unique session ID that will be acquired from the user's browser.
Event	The code for the event/activity
User	The user responsible for the request of the session
Address	The IP and MAC address of the originator
Context	Where the event took place.
TimeStamp	The date and time of the event.
Status	The status of the event (initialized, processing, failed, completed)
Before	The value before the modification (if a modification took place)
After	The value after the modification (if a modification took place)

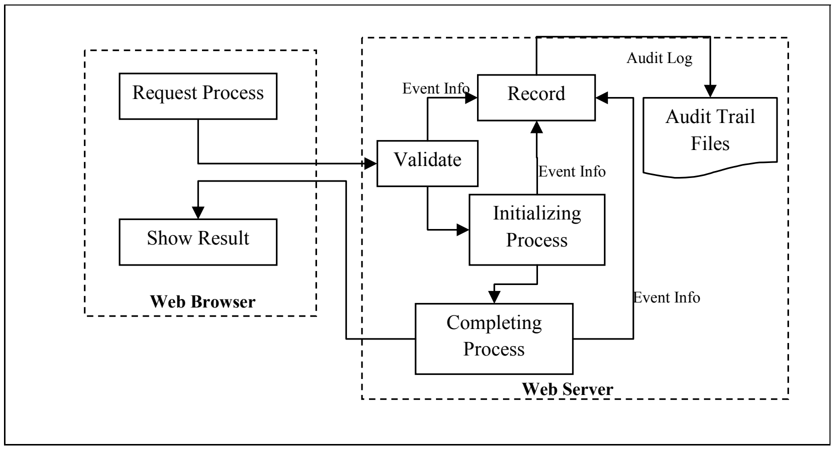


Figure 1. The audit trail generator.

identify starting, middle and end point of an event. This strategy can also be used to calculate the execution time of an event thus used to identify the current performance of the system based on execution time.

To make sure that the events itself will trigger the ATG to generate a log based on an event, a sensor-like procedure is be placed in each process. The sensor is typically a mechanism where when triggered, an event can occur without notifying the one that triggered the sensor. This sensor checks if the request for the session is valid to proceed to the requested event. At the same time, the sensor triggers the ATG to generate a log referring to the attempt to do such process. If the *sessionID* is valid, the sensor will then redirect the session to the actual event. The event itself would still have to authenticate the *sessionID*, if the *sessionID* is not valid, it will still not proceed, but if the *sessionID* is valid, the event again triggers the ATG to generate a log to show that the event is about to process, and then the event continues. When the event completes, the event then triggers the ATG to generate a log referring to the completion of the event. Such multiple sensors are considered to enforce the proper procedure to follow in doing an event. Such setup will create at most three logs on an event as shown in Figure 1.

Table 2 shows the major processes involved in within the framework. After careful study, the processes that can be applied with a three-point log will only be the processes where the context of the processing is within the control of the server. Processes such as link clicks for redirection to other pages can only have one log at a time since no inner processing is involved within the server side scripts.

The events that will be logged by the ATG takes place in the Web as the system itself is web based. The appropriate Data Mining Technique used would be the Web Usage Mining.[10] Figure 2 shows the diagram for the Audit Trail

Table 2. List of processes of the current software framework

Processes	Description	Context
Login	Authenticating the user for a new valid session	Server
Logout	Releasing the current session	Server
FileSecure	Uploading a file using encryption	Server
Genkeys	Generating keys for encryption	Server
EnableAccount	Enable a user's account	Server
DisableAccount	Disable a user's account	Server
Decrypt	Decrypting a secured file before downloading	Server
Register	Registering a new account	Server
LinkAccess	Transfer from one page to another page	Client

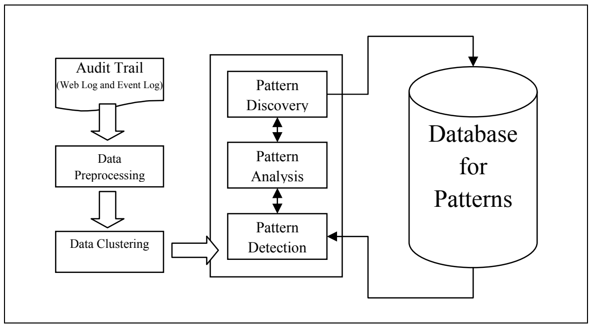


Figure 2. Trail pattern analyzer.

Pattern Analyzer anchored with Web usage mining but with pattern detection. It illustrates the various processes or procedures to be done to arrive with an audit trail to having the patterns discovered and detected to having it analysed. The audit trail will be going through data preprocessing. After preprocessing, it will then be clustered for possible discovery of patterns. The user can then select some of these patterns for further detection of related activities. These patterns can then be analysed through the use of visualisation techniques.

The Audit Trail shows only events that happened at a particular time and does not directly shows the connectivity between events to make a pattern. The techniques from the study of Wespi et al.[11] are adapted for the extraction of the patterns of the audit trail as the trail of activity will be extracted as a sequence of events. After preprocessing, these sequences are inputted to the pattern-extraction module. A variable-length pattern is then defined as a subsequence that has a minimum of two and occurs at least twice, be it in the same or in different sequences. Furthermore, only maximal variable-length patterns are considered. Teiresias algorithm is used in this technique to get the maximal patterns as it is well suited for the following reasons[12]: it finds the maximal variable-length patterns by avoiding the generation of non-maximal intermediate patterns during the pattern-extraction process; and its performance scales quasilinearly with the size of the output. And it follows that Teiresias vary efficiently finds all the maximal variable-length patterns in the set of sequences.

Relevant patterns will be identified by the administrator so that the next time an activity with a similar pattern occurs, the system will be able to notify the administrator. The patterns identified during the preprocessing will be compared with the existing patterns stored in the database.

Newly discovered patterns and existing discovered patterns are then identified. The purpose of this phase is the monitoring of the patterns itself. Included in the discovery and detection is the statistical analysis of the occurrence of these patterns. Such analysis will be graphically shown by the system for the administrator's better understanding. Examples of graphical representations will be line graphs and bar graphs.

Moreover, the Pattern Discovery involves comparing the base patterns during the clustering phase with the identified patterns in the database to see whether a new pattern has appeared. The Pattern Detection

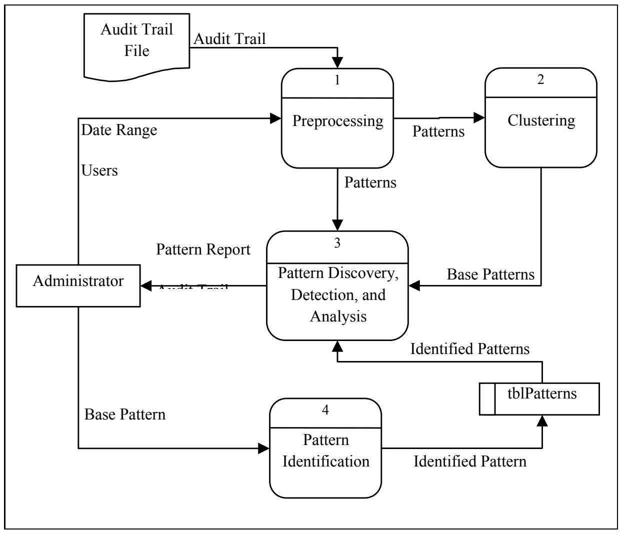


Figure 3. DFD of TPA major processes.

shows the occurrence of the already identified patterns in the currently associated set of audit trails. This set of information also includes warnings for patterns detected that are identified as a potential risk. With the patterns identified, it can then be classified as a *Normal* or *Suspicious* activity with a specified limit. If the limit has been exceeded on a *Normal* pattern, then a *Warning* signal will be issued. If the limit has been exceeded on a *suspicious* pattern, then an *Alert* signal will be issued. Part of the analysis will not be just on identifying patterns, but also instances of a given data. Such as identifying the usernames used with one session, or identifying the IP addresses used in one session.

Figure 3 shows the processes going on inside the TPA. The Audit Trail File is a file that holds the Audit Trails of the users of the system. Notice that the TPA is only capable of reading from the Audit Trail File and not modify it. The only module that can modify the Audit Trail File is the ATG. The modification of the ATG is only for appending new records and not altering or removing records. This is done to prevent the system from doing unwanted modifications from the Audit Trail File. The Audit Trail will always go through the preprocessing phase to convert these trails to patterns. But if the purpose is only to retrieve a copy of the audit trail, the preprocessing phase will be used to filter the Audit Trail based on the specifications of the administrator. The patterns from the Preprocessing phase will then be used to create Base Patterns in the Clustering phase. The patterns from the Preprocessing and the Base Patterns from the Clustering will then be used in the Pattern Discovery, Detection, and Analysis phase along with the identified Patterns from the database.

Each record of the audit trail is stored on a flat file using XML as its format. Figure 4 shows the template of the log file using XML. One element of a log contains all the data required for the trail. The log will be written to two audit trail files, a main audit trail file and a temporary audit trail file. The query retrieved and will be shown to the user on a logout will be from the temporary file. The two files have similar contents: the difference is that the temporary file contains only the logs of a user at a specific *sessionID*. This means the only connection when a user retrieves a log file is only through a temporary file for that user only. Only the administrator can have access to the main audit trail file. During analysis part, another temporary file will

```
<log>

<SessionID>[The unique session ID that will be acquired from the user’s
browser.]</SessionID>

<Event>[The code for the event/activity]</Event>

<User>[The user responsible for the request of the session]</User>

<Address>[The IP and MAC address of the originator]</Address>

<Context>[Where the event took place.]</Context>

<TimeStamp>[The date and time of the event.]</TimeStamp>

<Status>[The status of the event(Initialized, Processing, Failed, Com-
pleted)]</Status>

<Before>[The value before the modification (if a modification took
place)]</Before>

<After>[The value after the modification (if a modification took
place)]</After>

</log>
```

Figure 4. Sample log file in XML form.

be created, which contains copies of the logs based on the given time span. Such strategy imposes *authorisation* and *integrity* as a security requirement.

The log will be written to two audit trail files, a main audit trail file and a temporary audit trail file. The query retrieved and will be shown to the user on a logout will be from the temporary file. The two files have similar contents: the difference is that the temporary file contains only the logs of a user at a specific *sessionID*. This means the only connection when a user retrieves a log file is only through a temporary file for that user only. Only the administrator can have access to the main audit trail file. During analysis part, another temporary file will be created, which contains copies of the logs based on the given time span. Such strategy imposes *authorisation* and *integrity* as a security requirement.

To increase security to the audit trail, we limit the local of access to the file. The analysis will only be within the context of the server only thus using PHP server-side scripts. One problem with this is that the analysis of the audit trail requires heavy use of the web server's resources, to avoid overusing the resource of the web server just by analysis; the analysis is then broken down into subprocesses and will be invoked separately.

2.2. Evaluation for the Usability of the ATG and the TPA

The system was deployed in a controlled network for testing with multiple users to simulate the actual use of the system. A controlled network refers to a network where no external connections, aside from the computers within the network, to interfere with the test. These individuals for the testing must have the following qualifications: must be able to understand the System/Software Requirements Specification (SRS); must be able to implement the policies in letter and in spirit; must be preferably security conscious and aware; should also be of the opinion that security assurance can largely be started and implemented at the requirements level. Before the testing, an orientation took place with a demonstration of the system. Then it was the respondent's options to test it further themselves. If a respondent deems to do so, they are to take down their activity and rationale in the paper.

The system was then evaluated using an event log and audit trail checklist adapted from Pandey and Mustafa.[13] The checklist enables the assessment of the appropriateness of 'Event Log and Audit Trails' and lead to counter/additional measures for security assurance. 'Event Log and Audit Trails' is globally accepted as one of the prominent security requirements. Appropriate level of this requirement may well enforce security features and hence, ensure security for deployed software. One should try to achieve the maximum depending upon the various constraints like time, cost, efforts, etc. After a careful walkthrough of the checklist, with three (3) points noted as NA or Not Applicable, the checklist was then given to twelve (12) *Requirement Engineers* to use for the evaluation of the system. The following scaling are considered: If Overall Compliance Status (OCS) less than 50%, it is considered incomplete and insecure; if OCS is greater or equal to 50% but less than or equal 75%, it is considered as marginally complete and secure; if it is greater than 75%, it is considered as secure. The final result of checklist assessment is computed on the basis of the total compliant, non-compliant, and 'N/A' checkpoints. The system will be stronger if it satisfies all or most of the checklist items given in the checklist.

3. Results

3.1. The Audit Trail Generator

The result of the activities logged by the ATG in comparison with what is to be expected by the pen and paper approach to be logged showed that 100% activities were successfully logged by the ATG.

Looking at Figures 5 and 6, there is not much difference in the performance. After careful study, the figures show that there is only a minimal difference when a system leads in performance on a certain process. It is therefore evident that the ATG does not have a much of an effect on the performance of the system. This is due to the fact that

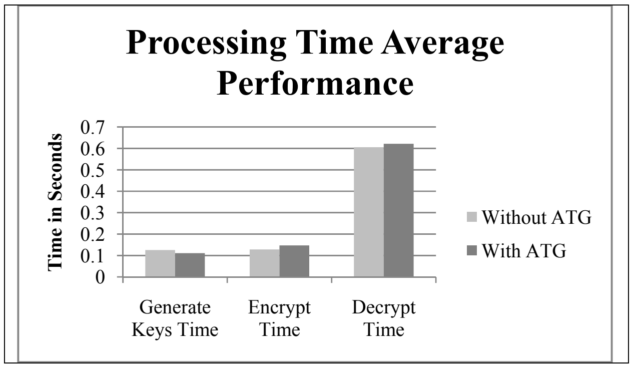


Figure 5. Bar chart on processing time average performance.

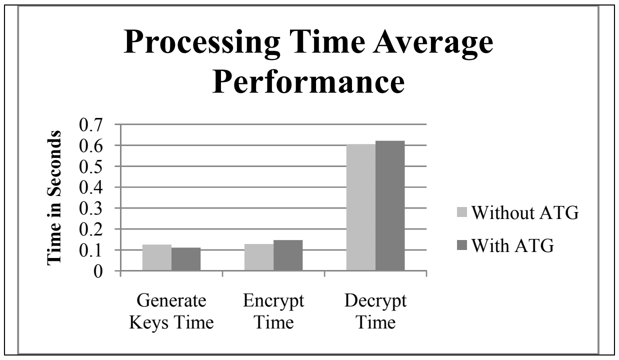


Figure 6. Bar chart on CPU utilisation average performance.

the function of the ATG is only to record the activity and that there is no real time detection involved in the design.

3.2. The Trail Pattern Analyzer

Table 3 shows the summary of the sessions and the sequences of events involved in the session extracted by the TPA. The sessions will expire if no current activity will be detected within 20 minutes. The sequences are composed of characters where each character corresponds to an event and the status of the event.

With the data found in Table 3, an initial detection can happen as shown in Figure 7 after clicking its Load button. Here, base patterns from the database will be compared with the extracted patterns and evaluated. The module will then show if how many sessions within the collection of sessions has an *Alert* flag, *Warning* flag, or *Normal* flag. Inside the box contains the details of the sequence with base patterns detected in that sequence and which category of flag these base patterns belong to.

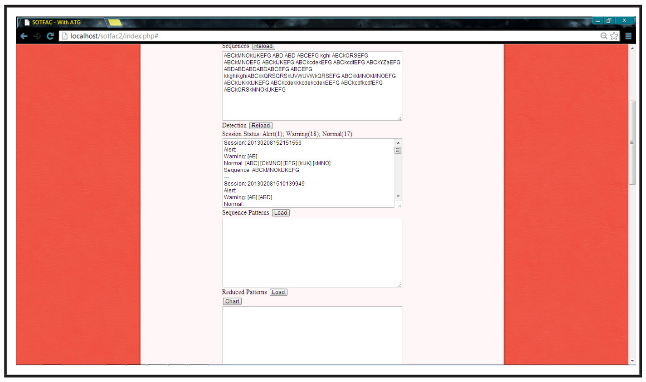
Figure 8 is the actual result of patterns extracted from the sequence separated with a space between the patterns. Each pattern is separated with a pipe "|" symbol where the actual pattern is on the left side of the pipe and the numeric value on the right side represents the number of times the pattern occurred within the sequences of events. The order of arrangement is based on the longest pattern first, then arranged in lexicographical order where capital letters are first in the order.

The new patterns will then be stored in the database automatically, where the TPA gives them default values to note that the pattern is still undefined. The new patterns can then be registered through a registration module.

After registering some of the patterns, it is then reflected pattern discovery and detection phase where a sample of

Table 3. Resulting event sequences from the CHS

SessionID	Event sequence
20130208152151556	ABCkMNOkIJKEFG
201302081510139949	ABD
201302081511029997	ABD
201302081511164097	ABCEFG
201302081511324751	Kghi
201302081513262873	ABCkQRSEFG
201302081514321186	ABCkMNOEFG
201302081516205250	ABCkIJKEFG
201302081518381859	ABCkcdekEFG
201302081520425598	ABCkcdfEFG
201302081523473842	ABCkYZaEFG
201302261449222445	ABDABDABDABDABCEFG
201302261457385260	ABCEFG
201302261458158983	kkghikghiABCkkQRSQRSkU VWUVWkQRSEFG
201302261514049123	ABCkkMNOkMNOEFG
201302261517143109	ABCkIJKkkIJKEFG
201302261518568041	ABCkcdekkkcdekcdekEEFG
201302261523238982	ABCkcdfkcdfEFG
201302261525246717	ABCkQRSkMNOkIJKEFG


Sequences [Reload]

ABCkMNOkIJKEFG ABD ABD ABCEFG kghi ABCkQRSEFG ABCkMNOEFG ABCkIJKEFG ABCkcdekEFG ABCkcdfEFG ABCkYZaEFG ABDABDABDABDABCEFG ABCEFG kkghikghiABCkkQRSQRSkUVWUVWkQRSEFG ABCkkMNOkMNOEFG ABCkIJKkkIJKEFG ABCkcdekkkcdekcdekEEFG ABCkcdfkcdfEFG ABCkQRSkMNOkIJKEFG

Detection [Reload]

Session Status: Alert(1); Warning(18); Normal(17)

```
Session: 20130208152151556
Alert:
Warning: [AB]
Normal: [ABC] [CkMNO] [EFG] [kIJK] [kMNO]
Sequence: ABCkMNOkIJKEFG
---
Session: 201302081510139949
Alert:
Warning: [AB] [ABD]
Normal:
```

Sequence Patterns [Load]

Reduced Patterns [Load]

[Chart]

Figure 7. Detection module (with data).

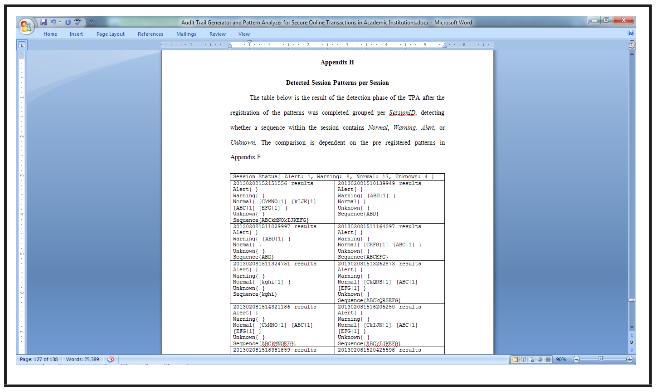
the result is shown in Figure 9. The status in Figure 10 shows the number of sessions that falls under different flags. Where 1 session has an *Alert* flag and requires immediate attention, 8 sessions have been issued with a *Warning* flag that requires some attention, 17 sessions have been issued with a *Normal* flag. And 4 sessions have *Unknown* flags, as some patterns were not registered to the system yet. Session 201302261449222445 was issued with an alert signal due to the patterns with occurrences

```
CkIJKk|1 Ckcdek|2 CkkMNO|1 CkIJK|2 CkMNO|2 CkQRS|2 CkYZa|1
Ckcdf|2 cdekE|2 kcdek|3 CEFG|3 DABC|1 QRSk|2 UVWk|1 cdek|4
kIJK|5 kMNO|5 kQRS|4 kcde|4 kcdf|3 kghi|3 ABC|16 ABD|6 Ckk|2
DAB|4 EFG|16 QRS|5 UVW|2 kcd|7 AB|22 kk|5
```

Figure 8. Reduced list of patterns from Figure 11 (pattern occurrence).

```
QRSQR|5 RSQRS|5 SQRSk|5 cdekE|10 kcdek|15 kkcde|5 kkghi|5 ABCk|52
IJKk|4 UVWk|4 cdek|16 kIJK|20 kMNO|20 kcde|16 kghi|12 ABD|18
EFG|48 QRS|15 UVW|6 YZa|3 cdf|9 kQR|12 kcd|21 kk|10
```

Figure 9. Reduced list of patterns using the original bCover() function (character occurrence).


Audit Trail Generator and Pattern Analyzer for Secure Online Transactions in Academic Institutions.docx - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View

Appendix H

Detected Session Patterns per Session

The table below is the result of the detection phase of the TPA after the registration of the patterns was completed grouped per *SessionID*, detecting whether a sequence within the session contains *Normal*, *Warning*, *Alert*, or *Unknown*. The comparison is dependent on the pre registered patterns in Appendix F.

```
Session Status{ Alert: 1, Warning: 8, Normal: 17, Unknown: 4 }
20130208152151556 results         201302081510139949 results
```

Session Status{ Alert: 1, Warning: 8, Normal: 17, Unknown: 4 }	

20130208152151556 results
Alert{ }
Warning{ }
Normal{ [CkMNO|1] [kIJK|1] [ABC|1] [EFG|1] }
Unknown{ }
Sequence{ABCkMNOkIJKEFG}
	201302081510139949 results
	Alert{ }
	Warning{ [ABD|1] }
	Normal{ }
	Unknown{ }
	Sequence{ABD}

201302081511029997 results
Alert{ }
Warning{ [ABD|1] }
Normal{ }
Unknown{ }
Sequence{ABD}
	201302081511164097 results
	Alert{ }
	Warning{ }
	Normal{ [CEFG|1] [ABC|1] }
	Unknown{ }
	Sequence{ABCEFG}

201302081511324751 results
Alert{ }
Warning{ }
Normal{ [kghi|1] }
Unknown{ }
Sequence{kghi}
	201302081513262873 results
	Alert{ }
	Warning{ }
	Normal{ [CkQRS|1] [ABC|1] [EFG|1] }
	Unknown{ }
	Sequence{ABCkQRSEFG}

201302081514321186 results
Alert{ }
Warning{ }
Normal{ [CkMNO|1] [ABC|1] [EFG|1] }
Unknown{ }
Sequence{ABCkMNOEFG}
	201302081516205250 results
	Alert{ }
	Warning{ }
	Normal{ [CkIJK|1] [ABC|1] [EFG|1] }
	Unknown{ }
	Sequence{ABCkIJKEFG}

201302081518381859 results	201302081520425598 results

Figure 10. Sample detected session patterns per session.

[ABD|4], [DAB|4], and [AB|5], where its sequence shows an exceeding limit of occurrences for suspicious actions, specifically failed login attempts.

Figure 11 shows the occurrences of a pattern on a specific session. The bar graph contains the base patterns detected in that sequence and its occurrences within that sequence. Other analysis can also be applied with the same approach as the previous analysis; the difference is on the focus on how the sequence was retrieved. On the previous analysis, the sequence was retrieved based on it session where logs with the same session are considered as part of one sequence. Instead of using the *sessionID* to group the logs, the user involved in doing the logs where used for creating the sequence. A smaller pattern set resulted on the clustering since only a small number of sequences were used to analyse since a user can have multiple sessions on a span of time and only registered usernames are included in the analysis.

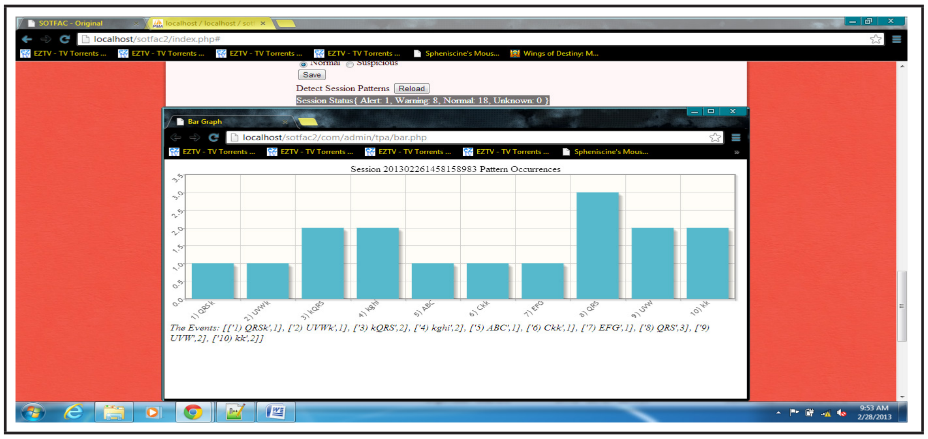
3.3. Initial Evaluation of the ATG and the TPA

The result of the tally of the evaluation is shown in Table 4. The table shows the tally of respondents who answered YES to a requirement, NO to a requirement, and the percentage compliance status per requirement. The compliance status was calculated based on the number of respondents who answered YES over the total number of respondents. After careful study, the lowest status is requirement number 3 (IT Resource Sabotage) where almost half of the respondents did not agree that the system has the capability to detect if such an event is to occur. Requirement 1, 4, and 10 (Employees Accountability, Compliance Monitoring, and Intrusion Detection System) shows that all respondents agreed that such requirements were met. The remaining requirements got 91.67% and 83.33% compliance status.

After consolidating Table 4, Table 5 shows the assessment result of the system. There are 14 checkpoints multiplied by 12 respondents resulting to 168 total

Table 4. Usability evaluation tally sheet

S. no.	Yes	No	Compliance status
1	12	0	100.00%
2	11	1	91.67%
3	8	4	66.67%
4	12	0	100.00%
5	11	1	91.67%
6	11	1	91.67%
7	10	2	83.33%
8	NA		
9	NA		
10	12	0	100.00%
11	10	2	83.33%
12	10	2	83.33%
13	NA		
14	11	1	91.67%



Session Status{ Alert: 1, Warning: 8, Normal: 18, Unknown: 0 }

The Events: [['1) QRSk',1], ['2) UVWk',1], ['3) kQRS',2], ['4) kghi',2], ['5) ABC',1], ['6) Ckk',1], ['7) EFG',1], ['8) QRS',3], ['9) UVW',2], ['10) kk',2]]

Figure 11. Bar graph for session 201302261458158983.

Table 5. Overall compliance status assessment result

Total checkpoints	Not applicable (N/A) checkpoints	Total available checkpoints	Non-compliant checkpoints	Compliant checkpoints	Overall compliance status
168	36	132	14	118	**89.39%**

checkpoints. Furthermore, there are 3 Not Applicable checkpoints multiplied by 12, thus 36 N/A checkpoints in total. The total available checkpoints would then be 132, after subtracting the total checkpoints from the total N/A checkpoints. The overall compliance was calculated from the percentage of the compliant checkpoints over the total available checkpoints which resulted in 89.39%. Since the OCS is greater than 75%, the system is considered secure.

4. Discussion

The ATG was capable in recording and tracking user activities and the TPA was capable in analyzing the patterns of these activities while addressing the four security issues namely *confidentiality, authorisation, non-repudiation*, and *integrity*, and addressing the requirements for a usable audit trail namely *employees accountability, security breach reporting, IT resource sabotage, compliance monitoring, record keeping of audit trails, system monitoring, security log reports, intrusion detection system, system monitoring tool, critical data,* and *IT users practice monitoring*. Based on the findings of the study, it is concluded that adopting the ATG as part of the Secure Online Transactions in Academic Institution provides the administrator of the system the ability to track the activity of the users without the issue of losing the systems performance since the analysis is of a separate module thus not in real-time. The adaptation of the TPA for analysis provided also the administrator to not just detect patterns that are possible for the intrusion, but also the ability to categorize the patterns detected for further detailing of the analysis.

The design of the ATG and the TPA looked only to user activities and not inside the confidential data that were transmitted during their transaction thus maintaining confidentiality. Since after every session, the users were given the summary of what they did during the session, non-repudiation is also maintained. As for integrity, it has been established that the logs are within the same location as the system thus giving the files isolation from outside modifications. And authorisation is established since the maintenance of the audit logs is available only within the access of the administrator account.

Furthermore, with the results of the evaluation, the design is considered usable and secured garnering an 89.39% OCS which is 14.39% above the required OCS standard to be considered secured.

For future works, the following directions are suggested: (1) Improve the performance of the pattern analysis by using a separate application to analyse the patterns rather than using server side scripts which depends on the power of the web server application being used. (2) Include additional pattern analysis focus aside from constructing by *sessionID* or by username such analysis for evaluating user activity depending on user account type, and user productivity. (3) Include real-time lightweight pattern analysis feature.

References

1. Eder MS, Doroja GS. Software framework for a secure online local transaction in academic institution. Mindanao J Sci Technol. 2013;11:1–20.
2. Mohajerani M, Moeini A. An approach to a new network security architecture for academic environments. In: Computer safety, reliability and security; 2002. P. 231–54.
3. Tak SW, Lee Y, Park EK. A software framework for non-repudiation service in electronic commerce based on the internet. In: Proceedings eleventh international conference on computer communications and networks; 2002. P. 182–9.
4. Tak SW, Park EK. A software framework for non-repudiation service based on adaptive~secure methodology in electronic commerce. Inf Syst Front. 2004;6(1):47–66.
5. Allison C. Information systems audit trails in legal proceedings as evidence. Comput Secur. 2001;20(5):410–21.
6. Legal recognition of electronic writing or document and data messages. [cited 2019]. https://infograph.venngage.com/p/224083/the-e-commerce-law-of-the-philippines-republic-act-no-8792.
7. Audit trail design concepts in building a secure business application. [cited 2006 Jul 26]. http://www.dotnetspider.com/resources/1851-Audit-Trail-Design-Concepts-building-secure-b.aspx.
8. Srivastava J, Cooley R, Mukund D, Tan PN. Web usage mining: discovery and applications of usage patterns from web data. SIGKDD Explor. 2000;1(2):12–23.
9. Omar L, Zakaria E. Clustering methods applied to the tracking of user traces interacting with an e-learning system. Int J Comput Commun Eng. 2012;6:260–64.
10. Kundu S. An intelligent approach of web data mining. Int J Comput Sci Eng. 2012;4(5):919–28.
11. Wespi A, Dacier M, Debar H. Intrusion Detection using variable-length audit trail patterns. In: RAID '00 proceedings of the third international workshop on recent advances in intrusion detection; 2000. P. 110–29.
12. Rigoutsos I, Floratos A. Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm. Bioinformatics. 1998;14(1):55–67.
13. Pandey SK, Mustafa K. Security assurance through efficient event log and audit trails. J Glob Res Comput Sci. 2012;3(1):27–30.