Global Journal of Information Technology: Emerging Technologies

Volume 9, Issue 1, (2019) 020-028

www.gjit.eu

Intrusion tolerance model against higher institution database

Bala Mohammed Yakubu*, Department of Computer Science, School of Science and Technology, Abubakar Tatari Ali Polytechnic, Bauchi, Nigeria
Hussaini DanAzumi, Department of Computer Science, School of Science and Technology, Abubakar Tatari Ali Polytechnic, Bauchi, Nigeria
Mohammed Bulama, Department of Computer Science, School of Science and Technology, Abubakar Tatari Ali Polytechnic, Bauchi, Nigeria
Abba Hassan, Department of Computer Science, School of Science and Technology, Abubakar Tatari Ali Polytechnic, Bauchi, Nigeria

Suggested Citation:
Yakubu, B. M., DanAzumi. H., Bulama, M. & Hassan, A. (2019). Intrusion tolerance model against higher institution database. *Global Journal of Information Technology: Emerging Technologies*. *9*(1), 020–028.

Received date August 2, 2018; revised date January 2, 2019; accepted date March 5, 2019;
Selection and peer review under responsibility of Prof. Dr. Dogan Ibrahim, Near East University, Cyprus.

Abstract

Privacy and security are the two major concerns of keeping and accessing data on the internet. The rate of intruding organisation's database by unauthorised users is on the increase. Thus, the affected organisation's data confidentiality is lost; it can be viewed, modified, deleted and/or make it inaccessible to authorised users. Intrusion detection and tolerance techniques help in recognising malicious attacks as well as supports the websites to survive the attack. A quantitative approach was used in this study even though numerous attempts of quantitative evaluation of the survivability of intrusion tolerant systems, especially in database field have been made. Study on survivability of intrusion tolerant systems has being done, taking behaviour of attack, prediction of scale, speed of database damage propagation and its degree of spreading as facilitators. This paper provides the intrusion tolerant database system as a series of state transition model (Zumkas Model) based on the hidden Markov model.

Keywords: Intrusion, survivability, model, patterns.

* ADDRESS FOR CORRESPONDENCE: **Bala Mohammed Yakubu**, Department of Computer Science, School of Science and Technology, Abubakar Tatari Ali Polytechnic, Bauchi, Nigeria *E-mail address:* rosese0079@gmail.com / Tel.: +2348067039020

1. Introduction

Privacy of information is the relationship between collection and dissemination of data, technology, the public expectation of privacy and the legal and political issues surrounding them (Hasty, Trevor & Subjally, 2013). Wherever there is personally identifiable information that includes sensitive information, which is collected and stored—in digital form, concerns in privacy must be considered. The root cause of privacy issues is improper or non-existent disclosure control (Hasty et al., 2013). Information security is the practice of defending information from unauthorised access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction (Herdantseva & Hilton, 2013).

The strength of any higher institution is its database. The security concerns of these databases are confidentiality, integrity and availability of information put in storage in a database. A broad span of research addresses primarily how to protect the secrecy of a database, namely, its confidentiality (Asankhaya, Govindarajan & Srivatsan, 2007). However, some of the higher institutions websites were not designed to successfully survive a database attack. This can seriously impair the integrity and availability of a database. Larose and Rifon (2007) stressed that the experience have shown that a variety of attacks do succeed in fooling conventional database protection mechanisms with data-intensive applications, such as credit card billing, banking and online stock trading. These attacks include but are not limited to malicious transactions through insider attacks, identity theft attacks, SQL injection attacks and cross-site scripting flaws (Wang, 2007).

There are numerous forms of online attack, of which is trending with time. The patterns of threats online are similar; however, there are variations in the pattern. The forms of online attack are not limited to phising, pharming, eavesdropping, malware, Trojan, rootkits, keyloggers, SQL injection/Code injection, denial of service, virus and worms.

The focus of this paper is for higher institutions to survive an attack against their websites. Using state transition model and hidden Markov model (HMM), Zumkas model was designed. Zumkas model enables higher institution website to anomaly detects any irregularity of observations or events that are different from the expected patterns of the items in the dataset of the database.

2. Systematic framework

2.1. Background on intrusion forbearance and detection

An intrusion is defined as ‘any set of activities that attempt to compromise the integrity, confidentiality or availability of a resource or data’ (Zhang, 2003; Zhang, Lee & Huang, 2003). When an intrusion takes place, intrusion prevention techniques, such as encryption and authentication (e.g., using passwords or biometrics), are usually the first line of defence. However, intrusion prevention alone is not sufficient enough because as systems become ever more complex, and as security is still often the after-thought, there are always utilizable weaknesses in those systems due to some design and programming errors, or various ‘socially engineered’ (Zhang, 2000), penetration techniques. For instance, even though they were first reported many years ago, exploitable ‘buffer overflow’ security holes, which can lead to an unauthorised root shell, still exist in some recent system softwares. Furthermore, as illustrated by the distributed denial-of-services (DDoS) attacks launched against several major Internet sites where security measures were in place, the protocols and systems that are designed to provide services (to the public) are inherently subject to attacks, such as DDoS (Portnoy, 2001).

Intrusion detection can be used as a second wall to protect network systems because once an intrusion is detected, e.g., in the early stage of a DDoS attack, response can be put into place to minimise damages, gather evidence for prosecution and even launch counter-attacks. The primary assumptions of intrusion detection are: user and program activities are observable, for example via system auditing mechanisms; and more importantly, normal and intrusion activities have distinct behaviour. Intrusion detection, therefore, involves capturing audit data and reasoning about the evidence in the data to determine whether the system is under attack. Based on the type of audit data used, intrusion detection systems (IDSs) can be categorised as network-based or host-based (Lee, 1998). A network-based IDS normally runs at the gateway of a network and 'captures' and examines network packets that go through the network hardware interface (Zhang, 2000). A host-based IDS relies on operating system audit data to monitor and analyse the events generated by programs or users on the host. Intrusion detection techniques can be categorised into misuse detection and anomaly detection. Misuse detection systems, e.g., IDIOT (Ilgun, 1995) and STAT (Lee, 1998).

While prior studies (Liu, 2001) stressed that very limited research has been done on how to survive successful database attacks, which can seriously impair the integrity and availability of a database. Experience with data intensive applications, such as credit card billing, banking, air traffic control, logistics management, inventory tracking and online stock trading, has shown that a variety of attacks do succeed to fool traditional database protection mechanisms (Heady, 1990). In fact, we must recognise that not all attacks—even obvious ones—can be averted at their outset. Attacks that succeed, to some degree at least, are unavoidable. With cyber-attacks on data-intensive internet applications, i.e., ecommerce systems, becoming an ever more serious threat to our economy, society and everyday lives, attack resistant database systems that can survive malicious attacks are a significant concern. One critical step towards attack resistant database systems is intrusion detection, which has attracted many researchers (Zhang, 2003). IDSs monitor system or network activity to discover attempts to disrupt or gain illicit access to systems. The methodology of intrusion detection can be roughly classed as being either based on statistical profiles (Zhang, 2000) or on known patterns of attacks, called signatures (Ilgun, 1995). Intrusion detection can supplement protection of network and information systems by rejecting the future access of detected attackers and by providing useful hints on how to strengthen the defence. However Liu (2001) stressed that the intrusion detection has several inherent limitations, such as: (a) Intrusion detection makes the system attack-aware but not attack-resistant, that is, intrusion detection itself cannot maintain the integrity and availability of the database in face of attacks. (b) Achieving accurate detection is usually difficult or expensive. The false alarm rate is high in many cases and (c) The average detection latency in many cases is too long to effectively confine the damage. To overcome the limitations of intrusion detection, a broader perspective is introduced, in addition to detecting attacks, countermeasures to these successful attacks should be planned and deployed in advance. In the literature, this is referred to as survivability or intrusion tolerance. In this paper, we will study a critical database intrusion tolerance problem beyond intrusion detection, namely, damage confinement, and present a set of innovative algorithms to solve the problem.

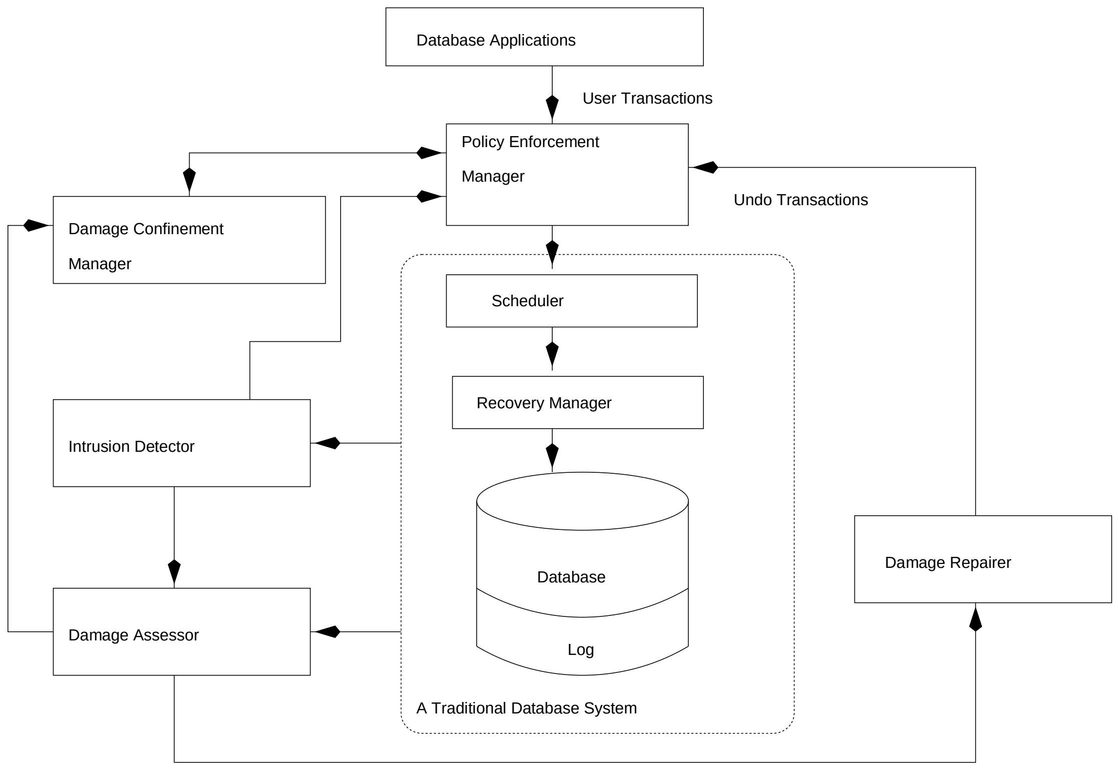

Figure 1. An intrusion tolerant database system architecture (Liu, 2007)

2.2. Threats

According to open web application security project (OWASP) (2013) one of the top 10 online threats is SQL injection (SQLi) in 2010 and 2007 (Top 10 Threats, 2013). In 2013, SQL injection was rated the number one attack on the OWASP top 10 (Top 10 Threats, 2013).

According to Deltchev (2010), SQL injection has five sub-classes of SQL injection: 1

Classic SQLi

Blind or Inference SQL injection

Database management system-specific SQLi

Compounded SQLi

SQL injection + insufficient authentication

SQL injection + DDoS attacks

SQL injection + DNS hijacking

SQL injection + XSS

Compounded SQLi, which is represented as Storm Worm (Deltchev, 2010).

SQL injection mostly occurs when a user is asked for an input into the system, like the username or userId, and instead of a name or userId, the user supplied an SQL statement that would be unknowingly run in to the system’s database.

The following example creates a SELECT statement by adding a variable (txtUserId) to a select string. The variable is fetched from user input (getRequestString):

```
txtUserId = getRequestString(“UserId”);
txtSQL = “SELECT * FROM Users WHERE UserId = " + txtUserId;
```

Another illustration of this injection using java statements below written by Zorabedian (2016)

```
String accountBalanceQuery =
“SELECT accountNumber, balance FROM accounts WHERE account_owner_id = "
+ request.getParameter("user_id”);
try {
Statement statement = connection.createStatement();
ResultSet rs = statement.executeQuery(accountBalanceQuery);
while (rs.next()) {
page.addTableRow(rs.getInt(“accountNumber”), rs.getFloat(“balance"));
}
} catch (SQLException e) { ... }
```

From the above code, the user with ID 117 might be logged in with URL: https://bankingwebsite/show_balances?user_id=117. This indicates accountBalanceQuery would end up being: *SELECT accountNumber, balance FROM accounts WHERE account_owner_id = 117.* It’s then passed into database, and the accounts and balances for user 117 are returned, and rows are added to the page to show them. The parameter ‘*user_id*’ can be changed by the attacker to be interpreted as: *0 or 1 = 1* and the balance query would be:

SELECT accountNumber, balance FROM accounts WHERE account_owner_id = 0 OR 1 = 1

If the query is passed into database, all the balances of account numbers stored and the added rows will be displayed for the attacker. Thus, the attacker knows full details of the bank accounts.

3. Model framework

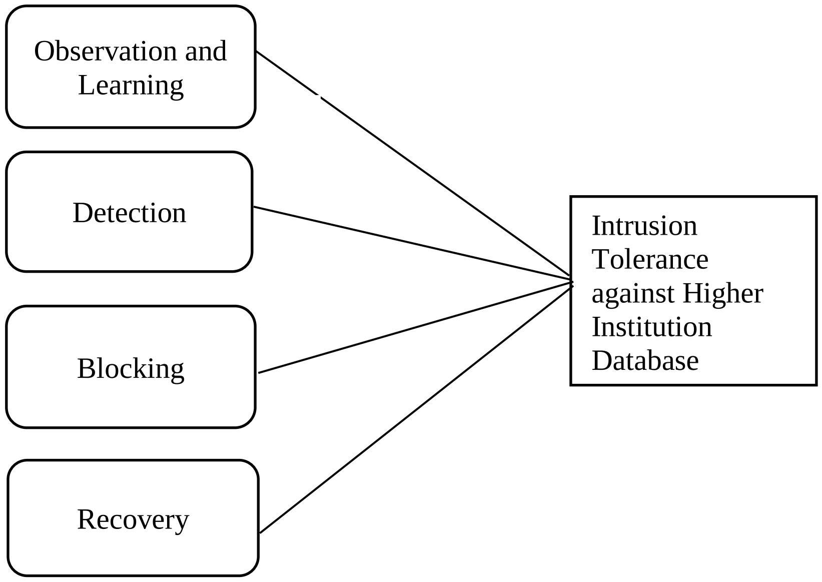

Figure 2. Zumkas tolerance models

As shown in Figure 2 above, the Zumkas tolerance model has four features, namely

1. Observation & Learning
2. Detection
3. Blocking
4. Recovery.

3.1. Observation and learning

These features were achieved using HMM as well as state e transition model.

The HMM is a variant of a finite state machine having a set of hidden states, *Q*, an output alphabet (observations), *O*, transition probabilities, *A*, output (emission) probabilities, *B* and initial state probabilities, Π. The current state is not observable. Instead, each state produces an output with a certain probability (*B*). Usually the states, *Q*, and outputs, *O*, are understood, so a HMM is said to be a triple, (*A*, *B* and Π) (Lawrence, 1989; Nikolai, 2017).

As described by Nikolai (2017), Hidden states *Q* = {qi}, *i* = 1,..., *N*. Transition probabilities *A* = {*aij* = *P*(qj at *t* + 1 | qi at *t*)}, where *P*(*a* | *b*) is the conditional probability of a given *b*, *t* = 1,..., *T* is time, and qi in *Q*. Informally, *A* is the probability that the next state is qj given that the current state is qi. Observations (symbols) *O* = {ok}, *k* = 1,..., *M*.

Emission probabilities *B* = {bik = bi(ok) = *P*(ok | qi)}, where ok in O. Informally, *B* is the probability that the output is ok given that the current state is qi.

Initial state probabilities Π = {pi = *P*(qi at *t* = 1)} (Nikolai, 2017).

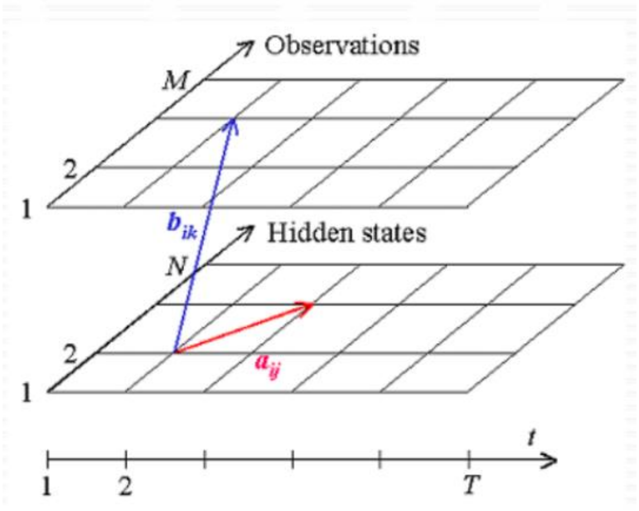

Figure 3. HMM (Nikolai, 2017)

Several models of Markov model exist, in this paper; forward algorithm was used. The forward algorithm, in the context of a HMM, was used to calculate a ‘belief state’: the probability of a state at a certain time, given the history of evidence, thus compute the probability of observation sequences (Norvig’s, 2003). Nikolai (2017) describe the Forward Algorithm as a recursive algorithm for calculating *αt*(*i*) for the observation sequence of increasing length *t*. First, the probabilities for the single-symbol sequence are calculated as a product of initial *i*th state probability and emission probability of the given symbol o(1) in the *i*th state. Then the recursive formula is applied. Assume we have calculated *αt*(*i*) for some *t*. To calculate *αt* + 1(*j*), we multiply every *αt*(*i*) by the corresponding transition probability from the *i*th state to the *j*th state, sum the products over all the states, and then multiply the result by the emission probability of the symbol o(t + 1). Iterating the process, we can eventually calculate $\alpha T(i)$, and then summing them over all states, we can obtain the required probability.

Initialisation:

$$\alpha 1(i) = pi\ bi\left(o\left(1\right)\right), i = 1,\ldots,N$$

Recursion:

$$here i = 1, \ldots, N, t = 1, \ldots, T - 1$$

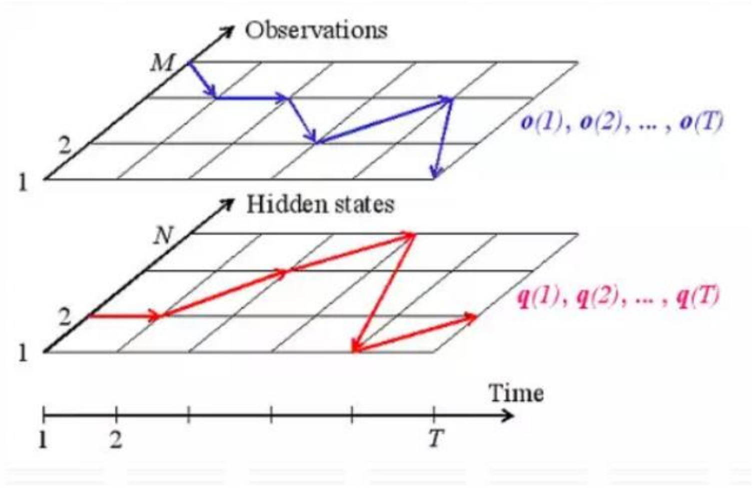

Figure 4. Forward algorithm (Nikolai, 2017)

Let αt (i) be the probability of the partial observation sequence Ot = {o(1), o(2),..., o(t)} to be produced by all the possible state sequences that end at the ith state.

$$\alpha t(i) = P(o\left(1\right), o\left(2\right), \ldots, o(t) \mid q(t) = qi).$$

Then, the unconditional probability of the partial observation sequence is the sum of $\alpha t(i)$ over all N states (Nikolai, 2017).

3.2. Detection

The system already knows the original pattern and behaviour of the site and the database. So also the system keeps track of all SQLi attempts. Similarly, the basic idea is to monitor the standard operations on a target system such as; logins, command and program execution's, file and device accesses, etc., looking only for deviations in usage. The system does not contain any special features for dealing with complex actions that feat a known or suspected security flaw in the target system; indeed, it has no information of the target system's security mechanisms or its deficiencies. Although a flaw-based detection mechanism may have some value, it would be considerably more complex and would be unable to cope with intrusions that exploit deficiencies that are not suspected or with personnel-related vulnerabilities. By detecting the intrusion; however, the security officer may be better able to locate vulnerabilities. Hence, all similar and new attempts by the intruders will not be accepted or allowed by the system.

3.3. Blocking

When the system detects an intrusion attempt, it blocks the intruder and marked it in the intruder database. In this situation, the system can be in only two states, either Allowed or Denied. State transition diagram below:

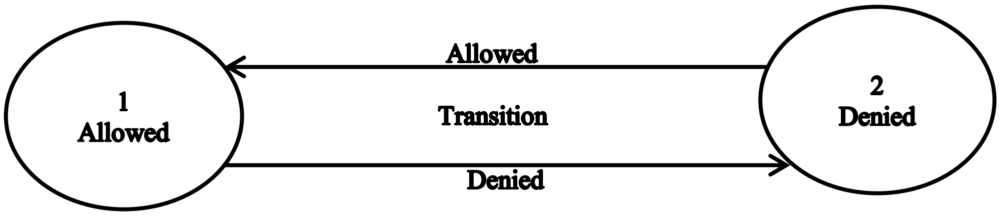

Figure 5. State machine diagram

3.4. Recovery

In computer science, state machine replication or state machine approach is a general method for implementing a fault-tolerant service by replicating servers and coordinating client interactions with server replicas. The approach also provides a framework for understanding and designing replication management protocols.

4. Conclusion

The study discovered that the intrusion prevention alone is not sufficient enough to protect the database of higher institutions because as the systems become ever more complex, and as security is still often the after-thought, there are always utilizable weaknesses in those systems due to some design and/or programming errors. Similarly, the study also discovered that the intrusion detection only involves capturing audit data and reasoning about the evidence in the data to determine whether the system is under attack but it has no tolerance with regards to learning the patterns of the of intrusion.

Finally, the study propose a model called Zumkas model which if properly implemented would provide intrusion tolerance against higher institution database by observing and learning the patterns of intrusion. Any learned pattern can be stored by the system and in case of a new pattern of attack the system observed its state and learns then updates itself. It goes to second state, i.e., Detection. Once the attack is detected it is immediately blocked by the system as third state. Finally, the system rolls back its database to previous state before the attack.

References

Asankhaya, S., Govindarajan, S. & Srivatsan, V. (2007). *DIDAR—database intrusion detection with automated recovery* (Thesis). NITW.

Dark Reading. (2012). *Security management*. Retrieved from http://www.darkreading.com/security/management/showArticle.jhtml?articleID=211201482

Deltchev, K. (2010). *New Web 2.0 attacks* (Thesis). Ruhr-University Bochum, Bochum, Germany.

Hasty, R., Trevor, W. N. & Subjally, M. (2013). Data protection law in the USA. *Advocates for International Development*.

Herdantseva, Y. & Hilton, J. (2013). Organizational, legal, and technological dimensions of information system administrator. In F. Almeida (Ed.), *Information security and information assurance.* Portela: IGI Global Publishing.

Larose, R. & Rifon, N. J. (2007). Promoting i-Safety: effects of privacy warnings and privacy seals on risk assessment and online privacy behavior. *Journal Of Consumer Affairs, 41*(1), 127–149.

Lawrence, R. R. (1989). A tutorial on Hidden Markov models & slelected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257286.

Nikolai, S. (2017). *Hidden Markov models*. Retrieved from http://www.shokhirev.com/nikolai/abc/alg/hmm/hmm.html

Norvig's, R. (2003). *Artificial intelligence, a modern approach* (2003 ed).

OWASP. (2013). *Top ten threats*. Retrieved from https://www.owasp.org/index.php/Top_10_2013-Top_10

Wang, H. (2007). *Modeling and evaluating the survivability of an intrusion tolerant database system* (Thesis). Pennsylvania State University, Information Sciences and Technology, State College, PA.

Zorabedian, J. (2016, November 22). *SQL injection*. Retrieved August 2, 2017, from https://www.veracode.com//sql-injection