Event based Alert Correlation System to detect SQLI Activities and Prevention using Stored Procedure Mechanism

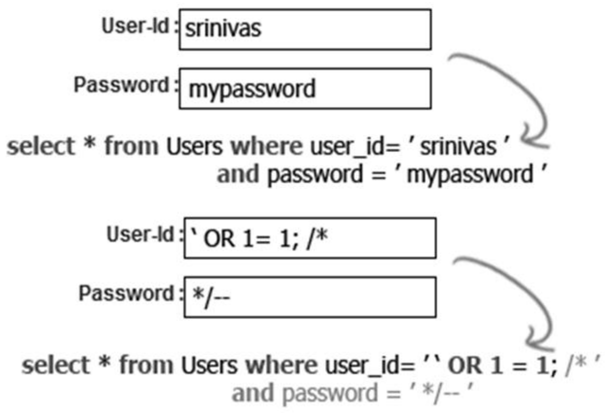
Rahul Patil[1] Aniket Deshpande[2] Nilesh Kasurde[3]
[1] Project Guide [2, 3]Student
[1, 2, 3]Bharti Vidyapeeth College of Engineering, Kharghar, Navi Mumbai-400614.

Abstract---Alerts correlation techniques have been widely used to provide intelligent and stateful detection methodologies. This is to understand attack steps and predict the expected sequence of events. However, most of the proposed systems are based on rule –based mechanisms which are tedious and error prone. Other methods are based on statistical modeling, These are unable to identify causal relationships between the events. In this paper, we have identified the limitations of the current techniques and propose a model for alert correlation that overcomes the shortcomings. The proposed model has been implemented in real-time and has successfully generated security events on establishing a correlation between attack signatures. The system has been evaluated to detect one of the most serious multi-stage attacks in Cyber-Crime – SQLIA (SQL Injection Attack). Typical SQLIA steps are analyzed within the realm of simulated malicious activities normally used by cyber criminals. SQL Injection attacks target databases that are accessible through a web front-end, and take advantage of flaws in the input validation logic of Web components such as CGI scripts. SQL Injection attacks can be easily prevented by more secure authentication schemes in login phase itself.

Keywords: Network intrusion detection systems; Alerts correlation; multi-stage attac; SQL Injection, stored procedures;

I. INTRODUCTION

SQL injection is a basic attack used either to gain unauthorized access to a database or to retrieve information directly from the database. They are used most often to attack databases and for extracting any confidential information such as Credit card information, Social Security numbers etc. Web applications are at highest risk to attack since often an attacker can exploit SQL injection vulnerabilities remotely without any proper database or application authentication.


User-Id : srinivas
Password : mypassword
select * from Users where user_id= ' srinivas '
and password = ' mypassword '
User-Id : ' OR 1= 1; /*
Password : */--
select * from Users where user_id= '' OR 1 = 1; /* '
and password = ' */-- '

An application is vulnerable to SQL injection for only one reason – end user input string is not properly validated and is passed to a dynamic SQL statement without any such validation. If we are sanitizing the user input, then indirectly we are restricting them to not entering single quotes and double quotes in the input. SQL injection is too much vulnerable that it can bypass many traditional security layers like Firewall, encryption, and traditional intrusion detection systems.

1) $ stmt =”SELECT * FROM users where username=’username’ and
2) Password= ’password’
3) For example if user enters “shrinivas‘or 1=1--“ and “any password”
4) In username and password fields instead of correct user name and
5) Password, the resulting query will be:
6) $stmt=”SELECT * FROM users where username=’shrinivas’ or
7) 1=1 --‘ and password=’any password’
8) After --, the rest of the sentence will be treated as comment and
9) Because of ‘1=1’ is always true.

The effectiveness of any NIDS depends on its ability to recognize different variations of cyber attacks. The current implementation of intrusion detection systems (commercial and open-source) is employing signature-based detection mechanisms. The main task of signature-based systems is to inspect the network traffic and perform pattern matching to detect attacks and generate alerts. The systems generates large number of alerts everyday and make the job of administrator difficult as the person has to sift the entire alert log to find out actual attacks. For this reason, high-level and real-time analysis techniques are needed. It has been practically identified that most of attacker activities consists of multiple steps (attack scenario) and occur in a certain time (attack window). Identification of such strategy can lead to the recognition of attack intensions and also prediction of unknown attacks.

Till now we have so many solutions to prevent SQLIA. Those are using bind variables, proper input validation, customized error messages, limiting database permissions. Any program or application may be vulnerable to SQL injection including stored procedures executed with a direct database connection. Write the stored procedure in one way, you can prevent SQL injection. Write it in another way, and you are still vulnerable to SQL injection. Stored procedures are an important part of modern-day web applications. It is an operation set that is stored in the database. Since stored procedures are stored on the server side, they are available to all clients. They add an extra layer of abstraction in to the design of a software system. This layer of abstraction also helps put up an extra barrier to potential attackers. The benefits of stored procedures are encapsulation of business logic in a single entity, faster execution, exception handling, create it once and store it in database and can be called any no. of times.

II. REQUIRES/PROVIDES MODEL

It has been proposed by [1] in inspiration from network management systems to deal with network faults. Cyber attack is described in two components: capabilities and concepts. The idea behind this model is that multi-stage intrusion consisting of a sequence of steps performed by an attacker; the later steps are prepared by the early ones. Target system information collected from scanning or port mapping, are advantages acquired to choose which exploit can be used. Capabilities are defined as general description of the conditions required or provided by each stage of intrusion. In other words, the system state that must be satisfied in order to launch an attack. For instance, a successful Trojan injection requires some particular services running in the target systems and an existence of vulnerabilities Formally, capabilities are a higher level of intrusion abstraction that specifies the system state after each attack attempt. Concepts are abstracts of system states that involved in multi-stage attack scenarios. Attacker uses the capabilities gained by some of his early actions to generate some new capabilities. System state incorporates in attack scenarios if instances of concepts have “required” and “provided” conditions matched.

III. MARS MODEL REVIEW

This section presents briefly the knowledge base of MARS model which automatically gives the attack related information to the admin in terms of Graph representation . As stated earlier, our model is derived from “provides/requires” model using different definitions of the model components. The proposed model for the knowledge base consists of three sets:

Capability C: This specifies a higher level of abstraction of intrusion model. Intrusion attempts are expressed in terms of a set of “required”, “provided”, and extensional “provided” conditions of a given alert.

Meta Alert (M-Alert) concept MC: This specifies the related capabilities of a given Meta-Alert. “Required” and “provided” conditions for each M-Alert are coded in language of capabilities.

Meta-Alert M: a higher level of abstraction of an alert. This can be generated from various IDS sensors So Meta-Alert will be elementary alert received. However, different M-Alerts will be aggregated in different occasions during the correlation process.

A M-Alert concept MC is an abstraction of elementary alerts generated by IDS defined by a set of (Arguments, Required Conditions, Provided Conditions, Extensional Provided Conditions, Vulnerability, Intrusion direction, and Experience) where:

Arguments [r1, r2 ,…ri]→r : are a set of associated attributes such as source and destination IP addresses.

Required Conditions R: are a set of pre-conditions specified in a form of capabilities with variable of Arguments.

Provided Conditions P: is a set of post-conditions specified in the form of capabilities with variable of Arguments.

Extensional Provided Conditions EP: are a set extended Provided Conditions as a result of implicit relations between capabilities in a form of capabilities with variable of Arguments.

Vulnerability V: is a description of state of the target host loor network with variable of Arguments.

Intrusion Direction D: is a description of attack direction (0: source address, 1: destination address, 2: bidirectional)

Experience EX: is description of the security officer’s Feedback in different situations.

```
Algorithm: Alert Correlation Graph
Input: elementary alerts generated by the IDS
Output: Correlated Attack Graph CAG(N,G)
Methods:
 1) Let CAG(N,G) = null
 2) Map elementary alerts to M-Alerts instances
    (m0,m1,……, m i)
 3) Let m0 an instance of isolated M
 4) For k=1 to I
    If
   a) at least one R(mi+1) □□P(mi)
      R (mi+1) □□EP (mi)
   b) V(m i+1), V(m), EX(m), and D(m) are satisfied.
   c) P(mi).End_time >= R (m i+1).Start_time
      EP(mi).End_time >= R (mi+1).Start_time        Then
      Add CAG (nmi, nmi+1)
 5) Return CAG(N,G)
```

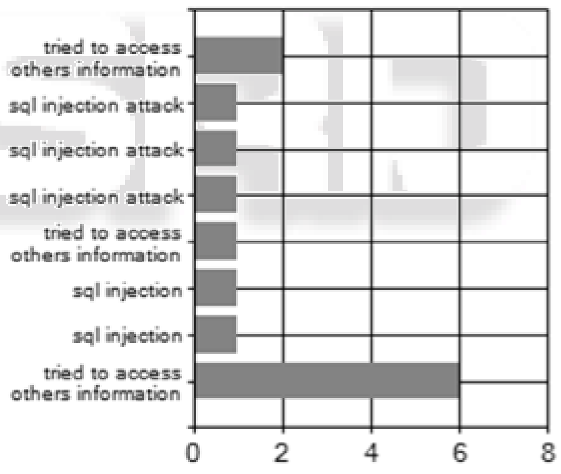


Fig. 1: Analysis According to Attacker Activities

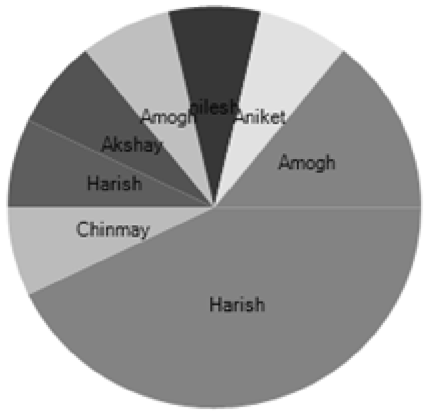


Fig. 2: Graph Based Representation of Detection Of Attack Activities By Insiders

IV. PREVENTION OF SQL INJECTION ACTIVITIES

Many techniques have been proposed to prevent SQL injection Attacks for example, dynamic monitoring tools. Various SQLIA detection techniques for the application layer have been proposed in literature, but none of them pay enough attention to SQLIA in stored procedures by providing enough security. Many existing techniques, such as filtering, information-flow analysis, penetration testing, and defensive coding, can detect and prevent subset of vulnerabilities that lead to SQLIAs. A number of techniques are in use for securing the web applications. The most common way is the authentication process through the username and password. One of the major problems in the authentication process is the input validation checking. We propose an SQL-Injection Attack prevention technique that addresses all types of SQLIAs. This technique works by combining encryption of user entered data within the stored procedure

V. CONCLUSION

We have presented our proposed correlation model to achieve high quality recognition of multistage attack in real time. The proposed approach is mainly based on improved version of "requires/provides" model which is basically used in plan recognition models. Novel methods have been presented to overcome the limitation of current systems: vulnerability, extensional conditions, attack direction, and administrator experience. It has been demonstrated that this mechanism can applied to detect complex multi-stage attack. The advantage with the proposed system is that the vulnerabilities in stored procedures can be avoided. We need not to sanitize the user input and therefore not restricting the user from entering special characters

REFERENCES

[1] S. J. Templeton and K. Levitt. A requires/provides model for computer attacks. In NSPW '00: Proceedings of the 2000 workshop on New security paradigms, 31-38, New York, NY, USA, 2000. ACM Press.

[2] F. Alserhani, M. Akhlaq, I. Awan, A. Cullen, and P. Mirchandani, "MARS: Multi Stage Attack Recognition System, In Proc. of the International Conf. on Advanced Information Networking and Applications (AINA), Perth, Australia, 2010 , 753-759.

[3] MeiJunji, An approach for SQL injection vulnerability detection. Sixth International Conference on Information Technology, (2009

[4] Indrani Balasundaram An Authentication Mechanism to prevent SQL Injection Attacks, International Journal of Computer Applications Volume 19– No.1, April 2011.

[5] Halfond, W. G. J. and A. Orso (2005). AMNESIA: analysis and monitoring for Neutralizing SQL-injection attacks. . ASE'05. Long Beach, California, USA.

[6] W. G. J. Halfond and A. Orso. Combining static analysis and runtime monitoring to counter sql-injection attacks. WODA, 2005.

[7] G.Wassermann and Z. Su. An analysis framework for security in web applications. SAVCBS, 2004.

[8] F. Cuppens and A. Miege. Alert correlation in a cooperative intrusion detection framework. In SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy, page 202, Washington, DC, USA, 2002. IEEE Computer Society.

[9] Jie Ma, Zhi-tang Li, Wei-ming Li, Real-Time Alert Stream Clustering and Correlation for Discovering Attack Strategies, fskd, vol. 4, 379- 384, 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, 2008

[10] K. Julisch. Clustering intrusion detection alarms to support root cause analysis. ACM Trans. Inf. Syst.Secur., 6, 4, 443-471, 2003.

[11] Lincoln Labs Information Systems Technology, http://www.ll.mit.edu/mission/communications/ist/corp ora/ideval/dat a/index.html

[12] Li, Z., A. Zhang, et al. Real-Time Correlation of Network Security Alerts. Proceedings of the IEEE International Conference on e- Business Engineering, IEEE Computer Society, 2007

[13] Nessus: Security Scanner; http://www.nessus.org

[14] Open Web Application Security Project: OWASP Top Ten Injection Flaws. http://www.owasp.org/index.php/Top_10_2007/injectio n_Flaws, Aug 2010

[15] Peng Ning, Yun Cui, Douglas Reeves, and Dingbang Xu, Tools and Techniques for Analyzing Intrusion Alerts, in ACM Transactions on Information and System Security, 7, 2, 273--318, May 2004.

[16] Peng Ning, Yun Cui, Douglas S. Reeves, Constructing Attack Scenarios through Correlation of Intrusion Alerts, in Proceedings of the 9th ACM Conference on Computer & Communications Security, 245--254, Washington D.C., November 2002.