فصلنامه علمی - پژوهشی فرماندهی و کنترل، سال پنجم، شماره یک، بهار 1400

بهینه سازی تشخیص حملات تزریق SQL با استفاده ترکیبی از الگوریتم های جنگل تصادفی و ژنتیک

جواد مرادی[1]، مجید غیوری ثالث[2]

تاریخ دریافت: 1399/06/11 **تاریخ پذیرش:** 1400/02/12

چکیده

علی‌رغم تمام تلاش متخصصان امنیتی برای کشف حملات تزریق SQL، اما بر اساس گزارش OWASP، کماکان حمله تزریق SQL به‌عنوان مهم‌ترین و زیان‌بارترین حمله سایبری توسط مهاجمین مورد استفاده قرار می‌گیرد. به‌منظور تشخیص حملات از دو روش مبتنی بر امضاء و مبتنی بر رفتار استفاده می‌شود. روش‌های مبتنی بر امضاء برای حملات شناخته شده کاربرد دارند و روش‌های مبتنی بر رفتار برای تشخیص حملات ناشناخته مناسب هستند. از آنجایی که حملات به روش‌های مختلفی پیاده‌سازی می‌شوند سیستم‌های تشخیص نفوذ مبتنی بر رفتار، کاربرد بیشتری دارند. رفتار را می‌توان با استفاده از روش‌هایی مانند طبقه‌بندی، خوشه‌بندی و غیره تحلیل کرد. یکی از مهم‌ترین الگوریتم‌های طبقه‌بندی، الگوریتم جنگل تصادفی است که دقت بالایی دارد و از طرفی پیاده‌سازی و تفسیر نتایج با استفاده از این الگوریتم به سادگی قابل انجام است. با توجه به بررسی‌های انجام شده دقت الگوریتم جنگل تصادفی به‌شدت وابسته به پارامترهای ورودی آن است. این پارامترها شامل ۹ مورد ازجمله تعداد درخت‌ها، عمق آن‌ها، نحوه رأی‌گیری، بهره اطلاعاتی و غیره است. تعیین بهینه این پارامترها یک مسئله بهینه‌سازی با فضای حالت بزرگ است. در این پژوهش روشی بر اساس الگوریتم ژنتیک برای تعیین مقادیر بهینه این پارامترها ارائه شده است. در اثر تعیین بهینه پارامترها، نتایج به‌دست‌آمده در مقایسه با حالت پیش‌فرض الگوریتم و سایر تحقیقات، بهبود دقتِ تشخیص را نشان می‌دهد. نتایج ارزیابی حاکی از آن است که دقت تشخیص نفوذ در روش پیشنهادی، ۹۸% بوده است که در مقایسه با الگوریتم جنگل تصادفی با پارامترهای پیش‌فرض حدوداً ۱۱% و در مقایسه با پژوهش‌های قبلی ۵۸% دقتِ تشخیص، افزایش یافته است.

واژه‌های کلیدی: الگوریتم جنگل تصادفی، الگوریتم ژنتیک، حمله تزریق SQL، سیستم تشخیص نفوذ پایگاه داده

[1] *کارشناس ارشد رایانش امن، دانشگاه جامع امام حسین (ع)*، jmoradi@ihu.ac.ir

[2] *استادیار دانشگاه جامع امام حسین (ع)*، ghayoori@ihu.ac.ir

1. مقدمه

تعداد حملات سایبری به‌صورت چشمگیری در حال افزایش است به‌طوری‌که بر اساس گزارش استاتیستا در سال ۲۰۱۹، 668 میلیون حمله‌ی سایبری رخ‌داده است [1].

بر اساس آخرین گزارش‌های مرکز تحقیقاتی OWASP، کماکان حملات تزریق جزء شایع‌ترین و خطرناک‌ترین حملات بر روی برنامه‌های تحت وب هستند. حملات تزریق شامل تزریق SQL، XML، ORM، Command و غیره است که یکی از مهم‌ترین و زیان‌بارترین آن‌ها حمله تزریق SQL است. در حمله تزریق SQL مهاجم با تزریق پرس‌وجوهای پایگاه‌داده در فیلدهای ورودیِ برنامه، می‌تواند داده‌های حساس از پایگاه داده را بخواند، تغییر دهد، عملیات مدیریتی مانند خاموش کردن پایگاه‌داده انجام دهد و در برخی موارد فرمان‌هایی را روی سیستم‌عامل اجرا کند [2].

یکی از راه‌کارهای تشخیص و جلوگیری از این حمله، استفاده از سیستم تشخیص نفوذ پایگاه داده است. سیستم‌های تشخیص نفوذ در دو دسته کلی، مبتنی بر امضاء و مبتنی بر رفتار تقسیم می‌شوند. در رویکرد مبتنی بر امضاء، الگوهای نفوذ توسط افراد خبره و متخصص تشخیص داده می‌شود و حملات در چهارچوب این الگوها شناسایی می‌شوند. ضعف این رویکرد در عدم تشخیص حملات جدید است و همچنین نیازمند به‌روزرسانی مداوم پایگاه‌داده امضاءها می‌باشد؛ اما در رویکرد دوم، سیستم تشخیص نفوذ با ایجاد الگوهای رفتار عادی کاربران، حملات را شناسایی می‌کند. این روش برای پیدا کردن حملات ناشناخته در پایگاه‌داده مناسب است. رفتارهای کاربران را می‌توان با استفاده از روش‌هایی مانند درخت تصمیم، جنگل تصادفی، نزدیک‌ترین همسایه، شبکه‌های عصبی و غیره شناسایی کرد. **الگوریتم جنگل** تصادفی در واقع یک الگوریتم ترکیبی[2] می‌باشد. در این روش مجموعه یا جنگلی از درختان مورد استفاده قرار می‌گیرد. استفاده از مجموعه‌ای از درخت‌ها صحت طبقه‌بندی و پیش‌بینی را افزایش می‌دهد، درحالی‌که بقیه خصوصیات مطلوب یک درخت مثل سادگی تفسیر نتایج را نیز حفظ خواهد شد [۱۷-۱۹].

تحقیقات اخیر نشان داده است که الگوریتم جنگل تصادفی در شناسایی رفتار حملات به خوبی عمل می‌کند [۱۹] اما در بررسی‌هایی که ما انجام دادیم مشخص شد تغییر در پارامترهای ورودی الگوریتم جنگل تصادفی تأثیر بسزایی در نتایج خروجی دارد.

به عبارت بهتر، تنظیم پارامترها در الگوریتم جنگل تصادفی مسئله بسیار مهمی است که با تنظیم بهینه آن می‌توان به نتایج خیلی بهتری نسبت به تحقیقات پیشین دست پیدا کرد؛ اما تنظیم پارامترها با توجه به تعداد و مقادیر زیاد این پارامترها منجر به حالات بسیار زیادی در الگوریتم جنگل تصادفی شده است که مسئله انتخاب بهترین پارامترها را به یک مسئله NP-Complete تبدیل کرده است. در این پژوهش ما سعی داریم با استفاده از الگوریتم فرا ابتکاری ژنتیک، مقدار بهینه پارامترهای الگوریتم جنگل تصادفی را به‌منظور بالا بردن دقت سیستم تشخیص نفوذ پایگاه داده تعیین کنیم.

2. پیشینه تحقیق

هنمنتهو و همکارانش [۲۳] در دانشگاه کاکاتیا هند مدلی را بر اساس درخت تصمیم برای جلوگیری از حمله تزریق SQL پیشنهاد دادند. در مدل پیشنهادی، یک پایگاه از انواع حملات تزریق SQL جمع‌آوری‌شده و پرس‌وجوهای جدید بر اساس ساختار درختی تعریف‌شده با این پایگاه مقایسه شده و اگر تطابقی یافت شد، پرس‌وجوی موردنظر را به‌عنوان حمله دسته‌بندی می‌کند. در این پژوهش ابتدا یک پیش‌پردازش بر روی پرس‌وجوها انجام می‌شود و در گام بعدی مقدار بهره اطلاعاتی[3] برای نقطه شکست پرس‌وجوها محاسبه می‌شود و درخت‌ها را بر این اساس می‌سازند. در ادامه هر پرس‌وجوی جدید، به درخت تصمیم‌گیری نگاشت می‌شود و طبقه‌بندی آن مشخص می‌شود. ازجمله مشکلات این روش می‌توان به موارد زیر اشاره

[2] Ensemble

[3] Information_Gain

کرد. الف) در این تحقیق درخت‌ها فقط بر اساس بهره اطلاعاتی ساخته‌شده‌اند، درصورتی‌که با بررسی سایر شاخص‌های اطلاعاتی مانند Gain_Ratio، Gini_Index یا Accuracy ممکن بود به نتایج بهتری دست پیدا کند. ب) عدم استفاده از سایر پارامترهای درخت مانند عمق درخت و تعداد برگ‌ها به‌منظور بهینه‌سازی مدل.

دبابراتا و همکارانش [۲۵] در دانشگاه سیلیکون هند برای تشخیص حملات تزریق SQL از رویکرد میزان شباهت پرس‌وجوها استفاده کردند. هسته کار دبابراتا و همکارانش در دو گام خلاصه می‌شود. گام اول نرمال کردن پرس‌وجوها و گام دوم بررسی میزان شباهت پرس‌وجوهای جدید با پرس‌وجوهای آموزش‌دیده می‌باشد. آن‌ها به‌منظور تحلیل و تشخیص حملات، پرس‌وجوها را قبل از تحلیل در موتور تحلیل‌گر که بر اساس خوشه‌بندی است، پرس‌وجوها را به یک‌شکل یکسان که تمامی آن‌ها دارای نظم خاصی هستند، تبدیل کردند. در جدول 1 به نحوه نرمال کردن کاراکترهای خاص و در جدول ۲ به نحوه نرمال کردن کلمات و رشته‌های مورداستفاده در عبارت‌های SQL اشاره‌شده است. درروش پیشنهادی ما نیز از روش کار دبابرتا برای نرمال‌سازی داده‌ها استفاده شده است.

جدول 1: نرمال کردن کاراکترهای خاص [۲۵]

Symbol	Name	Transform	Symbol	Name	Transform
`	Backtick	Remove	(	Opening Parenthesis	LPRN
!= or <>	Not Equals	NEQ	)	Closing Parenthesis	RPRN
&&	Logical AND	AND	{	Opening Brace	LCBR
||	Logical OR	OR	}	Closing Brace	RCBR
~	Tilde	TLDE	[	Opening Bracket	LSQBR
!	Exclaimation	EXCLM	]	Closing Bracket	RSQBR
@	At-the-rate	ATR	\	Back Slash	BSLSH
#	Pound	HASH	:	Colon	CLN
$	Dollar	DLLR	;	Semi-colon	SMCLN
%	Percent	PRCNT	"	Double Quote	DQUT
^	Caret	XOR	'	Single Quote	SQUT
&	Ampersand	BITAND	<	Less Than	LT
|	Pipe or Bar	BITOR	>	Greater Than	GT
*	Asterisk	STAR	,	Comma	CMMA
-	Hyphen/Minus	MINUS	.	Stop or Period	DOT
+	Addition/Plus	PLUS	?	Question Mark	QSTN
=	Equals	EQ	/	Forward Slash	SLSH

پس از اتمام نرمال‌سازی، پرس‌وجوهای حمله و پرس‌وجوهای عادی را با خاصیت خوشه‌بندی در خوشه‌های متفاوت قرار دادند. مشکل اول این تحقیق، عدم ارائه راه‌کاری برای بهینه‌سازی خوشه‌بندی مانند تعداد خوشه‌ها است. مشکل دوم این تحقیق استفاده از تطبیق الگوها یا عبارت منظم است که با توجه به ماهیت چندشکلی بردارهای حمله به‌عنوان راه‌حل عملی مناسب نیست. مشکل سوم در قسمت نتایج کار می‌باشد که مدت‌زمان مدل‌سازی طرح بیان‌نشده است.

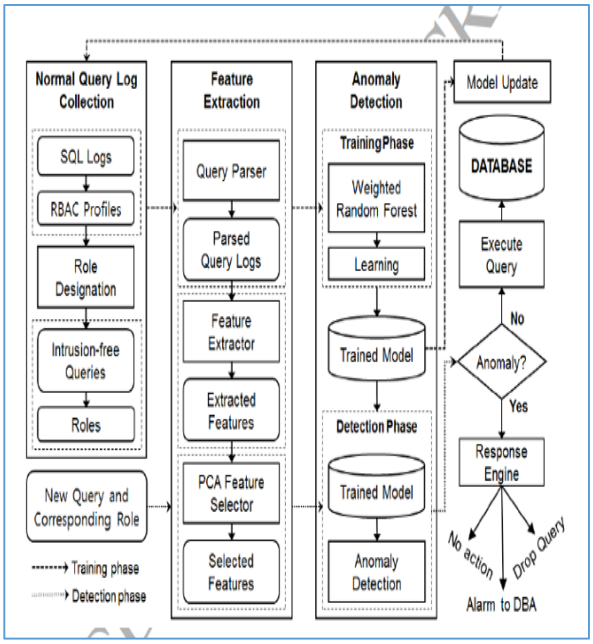
جدول ۲ : نرمال کردن عبارت‌ها و کلمات استفاده‌شده در پرس‌وجوها [۲۵]

Step	Token/Symbol	Transform
1.	Newline characters (\r or \n) if any	Remove
2.	Inline Comments (/*...*/) if any	Remove
3.	Anything within single/double quotes	
	(a) Hexadecimal value	HEX
	(b) Decimal value	DEC
	(c) Integer value	INT
	(d) IP address	IPADDR
	(e) Single alphabet character	CHR
	(f) General string (none of the above)	STR
4.	Anything outside single/double quotes	
	(a) Hexadecimal value	HEX
	(b) Decimal value	DEC
	(c) Integer value	INT
	(d) IP address	IPADDR
5.	System objects	
	(a) System databases	SYSDB
	(b) System tables	SYSTBL
	(c) System table column	SYSCOL
	(d) System variable	SYSVAR
	(e) System views	SYSVW
	(f) System stored procedure	SYSPROC
6.	User-defined objects	
	(a) User databases	USRDB
	(b) User tables	USRTBL
	(c) User table column	USRCOL
	(d) User-defined views	USRVW
	(e) User-defined stored procedures	USRPROC
	(f) User-defined functions	USRFUNC
7.	SQL keywords, functions and reserved words	To Uppercase
8.	Any token/word not transformed so far	
	(a) Single alphabet	CHR
	(b) Alpha-numeric without space	STR
9.	Other symbols and special characters	As per Table 2
10.	The entire query	To Uppercase
11.	Multiple spaces	Single Space

کریسا آن رنو و همکارانش [۱۹] در دانشگاه یونسی کره جنوبی طی ارائه مقاله‌ای به بررسی نحوه‌ی تشخیص پرس‌وجوهای غیر نرمال در پایگاه داده با استفاده از الگوریتم جنگل تصادفی و PCA[4] پرداختند. ازآنجاکه این تحقیق، مرجع روش پیشنهادی ما است در این قسمت به بررسی دقیق‌تر آن پرداخته‌ایم. آن‌ها سیستم تشخیص نفوذ پایگاه داده مبتنی بر

[4] Principal Component Analysis

الگوریتم جنگل تصادفی با رأی‌گیری وزنی[5] و در کنار آن استفاده از الگوریتم PCA به‌عنوان یک تکنیک انتخاب ویژگی را پیشنهاد نمودند.



شکل 1: معماری سیستم تشخیص نفوذ پایگاه داده مبتنی الگوریتم جنگل تصادفی [۱۹]

معماری روش مذکور در شکل 1 نمایش داده‌شده است. در این پژوهش برای هر نقش در پایگاه داده یک پروفایل به‌عنوان رفتار عادی ایجاد کرده و تراکنش‌هایی که با پروفایل‌های ساخته‌شده مطابقت نداشته باشد به‌عنوان حمله طبقه‌بندی می‌شود. برای این کار در مرحله اول، ورودی‌های کاربران عادی از پایگاه‌داده جمع‌آوری‌شده و پروفایل‌های متناسب هر نقش ساخته می‌شود. در مرحله دوم از طریق تجزیه‌کننده (پارسر)، ویژگی‌های پرس-وجوها را استخراج کرده و این ویژگی‌ها را معادل‌سازی عددی می‌کردند، سپس با استفاده از الگوریتم PCA ابعاد ویژگی‌ها را کاهش داده و در مرحله آخر با استفاده از الگوریتم جنگل تصادفی مدل‌سازی را به‌منظور تشخیصِ آنومالی ساختند. مدل پیشنهادی فوق دو ایراد مهم دارد، اول اینکه فقط از یک پارامترِ الگوریتم جنگل تصادفی (درختان وزن‌دار) به‌منظور بهینه‌سازی این الگوریتم استفاده‌شده است، درصورتی‌که می‌توان از سایر پارامترهای الگوریتم جنگل تصادفی مانند تعداد درختان، عمق درختان، بهره اطلاعاتی و غیره به‌منظور بهینه‌سازی کامل این الگوریتم استفاده کرد. ایراد دوم مربوط به استفاده نادرست از الگوریتم PCA است. در این مقاله برای استفاده از الگوریتم PCA به‌منظور کاهش ابعاد، باید پرس‌وجوها به مقادیر عددی تبدیل شوند که این کار نهایتاً منجر به کدگذاری پرس‌وجو به مقادیر عددی می‌شود و باعث مدل‌سازی نادرست توسط الگوریتم جنگل تصادفی می‌شود.

3. روش پیشنهادی

با توجه به موارد ذکرشده در بیشینه تحقیق، تصمیم گرفتیم در گام اول به‌منظور پیش‌پردازش مناسب از ایده کار دبابراتا و همکارانش [۲۵] برای نرمال کردن پرس‌وجوها استفاده کرده و در گام دوم به‌منظور حداکثر کارایی الگوریتم جنگل تصادفی در مسئله تشخیص حملات تزریق SQL تمامی پارامترهای الگوریتم جنگل تصادفی را به روش سیستماتیک و نه به روش آزمون‌وخطا، با استفاده از الگوریتم ژنتیک، بهینه‌سازی کنیم.

در روش پیشنهادی، یک سامانه تشخیص نفوذ مبتنی بر رفتار ارائه کرده‌ایم. در این مدل، از الگوریتم جنگل تصادفی به‌عنوان موتور تحلیل‌گر سیستم تشخیص نفوذ و الگوریتم فرا ابتکاری ژنتیک به‌منظور بهینه‌سازی الگوریتم جنگل تصادفی استفاده کرده ایم. دلیل اصلی انتخاب الگوریتم جنگل تصادفی برای موتور تحلیلگر سیستم تشخیص نفوذ، کارایی بالا در عین سادگی این روش می‌باشد. در الگوریتم جنگل تصادفی ۹ پارامتر قابل تنظیم دارد که این پارامترها عبارت‌اند از: تعداد درخت[6] بهره اطلاعاتی[7] حداکثر عمق درخت‌ها[8] حداقل سود[9] حداقل سایز برگ‌ها[10] انجام پردازش[11] حدس زدن حد زیرمجموعه[12] نحوه

[5] WRF
[6] Number of Trees
[7] Criterion
[8] Maximal Depth
[9] Minimal Gain
[10] Minimal Leaf Size
[11] Apply Pruning
[12] Guess Subset Ratio

ی رأی‌گیری درختان[13]و اجرای موازی الگوریتم. بنابراین تعداد حالات پارامترهای قابل تنظیم 8,000,000 حالت است که این عدد (تعداد حالات قابل تنظیم) از ضرب تعداد حالات تک‌تک پارامترها به‌دست‌آمده است. تعداد حالات هرکدام از پارامترها عبارت‌اند از:

Number of Trees = ۱۰۰
Criterion = ۵
Maximal Depth = ۱۰
Minimal Gain = ۱۰
Minimal Leaf Size = ۱۰
Apply Pruning = ۲
Guess Subset Ratio =۲
Voting Strategy =۲
parallel execution =۲

۱۰۰ × ۵ × ۱۰ × ۱۰ × ۱۰ × ۲ × ۲ × ۲ ×۲ = ۸۰۰۰۰۰۰

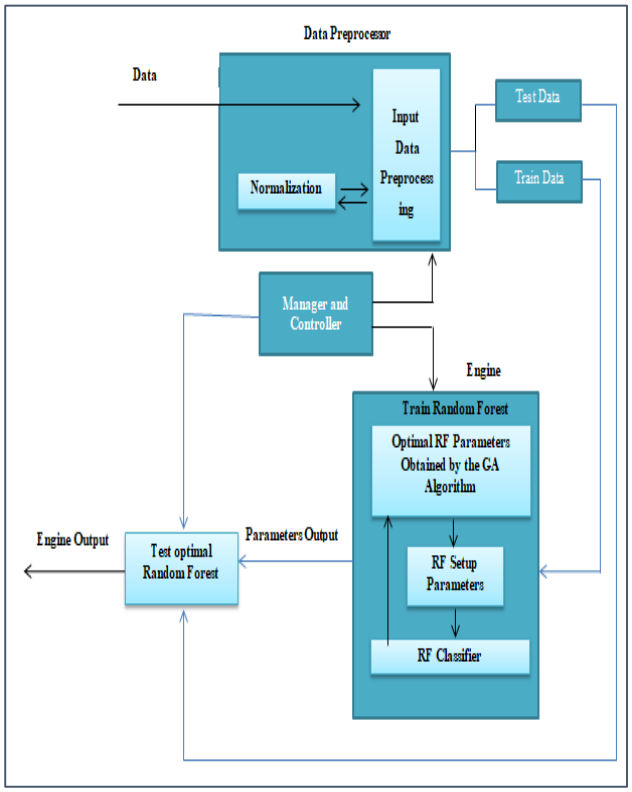
با بررسی اکثر مدل‌هایی که از الگوریتم جنگل تصادفی استفاده کرده‌اند، متوجه شدیم که این پارامترها به‌صورت پیش‌فرض و با به‌صورت سعی و خطا توسط مدیر تنظیم‌شده‌اند. در مدل پیشنهادی قصد داریم که این پارامترها را به‌صورت سیستماتیک تنظیم کنیم به نحوه‌ی که سیستم تشخیص نفوذ حداکثر کارایی را داشته باشد.

به‌منظور بهینه‌سازی پارامترهای ارائه‌شده، استفاده از الگوریتم فرا ابتکاری ژنتیک را پیشنهاد کردیم. الگوریتم ژنتیک به‌عنوان یکی از روش‌های مؤثر جهت حل مسائل بهینه‌سازی یا مسائلی با فضای جستجوی بزرگ شناخته‌شده است، در واقع فرا ابتکاری ژنتیک برای تعیین پارامترهای الگوریتم جنگل تصادفی، نسبت به سایر الگوریتم‌های فرا ابتکاری دیگر، اولاً برای حل مسائل گسسته و غیرخطی با فضای جست‌وجوی بزرگ بسیار کارا می‌باشد، ثانیاً به علت خصلت تصادفی آن، مشکل گیرکردن در بهینه‌های محلی را به‌شدت کاهش می‌دهد، ثالثاً پیاده‌سازی آن ساده بوده و نیازی به روال‌های پیچیده حل مسئله ندارد.

روش پیشنهاد شده متشکل از ۳ مؤلفه اصلی است. این مؤلفه‌ها عبارت‌اند از:

1. مؤلفه پیش پردازنده[14]
2. مؤلفه موتور تحلیل‌گر مبتنی بر الگوریتم جنگل تصادفی[15]
3. مؤلفه مدیر و کنترل‌کننده[16]

در شکل 2 شمای کلی روش پیشنهادی ترسیم شده است.



شکل 2: معماری طرح پیشنهادی

در این سیستم، ورودی شامل مجموعه‌ای از پرس‌وجوهای پایگاه داده می‌باشد که به‌صورت رکوردهایی در جریان هستند. این رکوردهای وارد مؤلفه‌ی اول، یعنی مرحله پیش‌پردازش می‌شوند. در مرحله پیش‌پردازش، پرس‌وجوها نرمال شده که خروجی حاصل، شامل پرس‌وجوهایی با شکل یکسان می‌باشد. سپس پرس‌وجوهای نرمال شده را به مؤلفه تحلیلگر مبتنی بر الگوریتم جنگل تصادفی می‌فرستد. تحلیل‌گر بر اساس فرمان دریافتی از سوی مدیر و کنترل‌کننده‌ی مرکزی، به ترتیب ابتدا پارامترهای الگوریتم جنگل تصادفی را با استفاده از الگوریتم ژنتیک تعیین می‌کند. سپس با دریافت هر پرس‌وجو از مجموعه

[1] Voting Strategy [3]
[1] Data Preprocessor [4]
[1] Random Forest Based Analyzer [5]
[1] Manager and Controller [6]

داده‌ی موردنظر، بر اساس تجربیات کسب‌شده در فاز آموزش، نرمال یا حمله بودن ورودی را مشخص می‌نماید.

۱،٤ مولفه پیش‌پردازنده

وظیفه‌ی پیش‌پردازش دریافت پرس‌وجوهای مربوط به پایگاه داده و انجام عمل نرمال‌سازی بر روی آن‌ها می‌باشد. از آنجایی که بردارهای حمله تزریق SQL به‌صورت مختلفی پیاده‌سازی می‌شود، اگر بتوان بردارهای حمله به یک شکل یکسان تبدیل نمود، انجام تحلیل و تشخیص حملات به راحتی انجام می‌شود. یکی از تحقیقات خوب در زمینه پیش‌پردازش تراکنش‌های پایگاه‌داده، ایده کار دبابارتا و همکارانش [۲۵] می‌باشد که در طی آن تمامی پرس‌وجوی پایگاه داده را به شکل یکسان تبدیل می‌کند. در جدول ۱ به نحوه نرمال کردن کاراکترهای خاص و در جدول ۲ به نحوه نرمال کردن کلمات و رشته‌های مورداستفاده در عبارت‌های SQL اشاره‌شده است. هدف از این کار تأمین ورودی مناسب برای مرحله بعدی، یعنی یادگیری مدل می‌باشد.

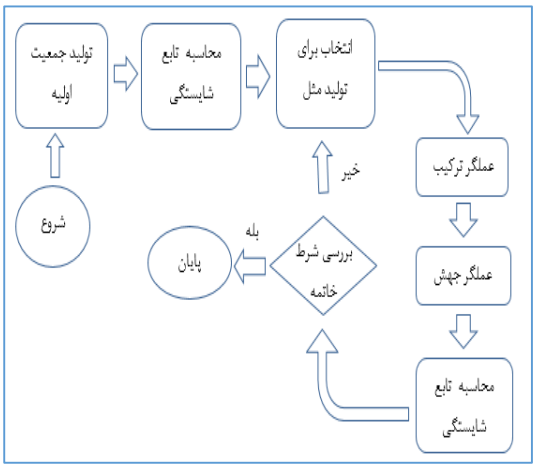
۲،٤ تحلیلگر مبتنی بر الگوریتم جنگل تصادفی

در این مرحله مهم‌ترین مؤلفه این معماری قرار دارد که بخش موتور تحلیلگر[1] آن است. این مرحله وظیفه پردازش مجموعه داده و درنهایت دسته‌بندی آن‌ها را بر عهده دارد.

در الگوریتم جنگل تصادفی تعیین پارامترهای بهینه در عملکرد این الگوریتم بسیار حائز اهمیت است؛ برای رسیدن به این هدف در این مقاله، از الگوریتم بهینه‌سازی ژنتیک برای تعیین پارامترهای الگوریتم جنگل تصادفی استفاده کرده‌ایم. این الگوریتم با در نظر داشتن تمام ابعاد مسئله به‌مرورزمان، فضای جستجوی خود را به سمت بهترین راه‌حل‌ها محدود می‌کند که این ویژگی منحصربه‌فرد باعث کارایی بالای این الگوریتم در عین حفظ کارایی و دقت می‌شود. همان‌طور که در شکل 3 مشاهده می‌شود چرخه الگوریتم ژنتیک به شرح زیر است.

1. تولید جمعیت اولیه
2. محاسبه تابع شایستگی
3. انتخاب دو کروموزوم از جمعیت به‌عنوان والد
4. اجرای عملگر ترکیب به‌منظور تولید فرزند.
5. در صورت مهیا بودن شرایط، اجرای عملگر جهش
6. محاسبه تابع شایستگی فرزندان
7. بررسی شرط خاتمه

- اگر شرط خاتمه صحیح بود اجرای الگوریتم متوقف می‌شود و بهترین راه‌حل انتخاب می‌شود.
- اگر شرط خاتمه صحیح نبود به گام 2 برگرد.



شکل 3: فلوچارت الگوریتم ژنتیک

در حل هر مسئله‌ای توسط الگوریتم ژنتیک باید الزامات و توابع این الگوریتم من‌جمله شیوه کدگذاری، تابع شایستگی، تابع انتخاب، عملگر ترکیب، عملگر جهش و شرط خاتمه الگوریتم تبیین شود. در ادامه به تشریح هر بند پرداخته‌شده است.

۱،۲،٤ شیوه کدگذاری

الگوریتم ژنتیک بجای اینکه بر روی پارامترها یا متغیرهای مسئله کار کند، با شکل کد شده آن‌ها سروکار دارد. روش‌های کدگذاری متداول متغیرها در الگوریتم ژنتیک عبارت‌اند از

[1] Analyzer

کدگذاری باینری،[18] کدگذاری جهشی[19]کدگذاری ارزشی[20]و کدگذاری درختی.[21]

در روش پیشنهادی از کدگذاری جهشی استفاده می‌کنیم و هر پارامتر را معادل یک ژن در نظر می‌گیریم که ۹ ژن در کنار یکدیگر، یک کروموزوم را تولید می‌کند.

۲,۲,٤ تابع شایستگی

زمانی که در فضای مسئله به دنبال بهترین جواب‌ها هستیم باید معیاری را به‌منظور مناسب بودن جواب تعریف کنیم. در الگوریتم ژنتیک، معیاری که این کار را انجام می‌دهد تابع شایستگی[22]ام دارد. ورودی تابع شایستگی، کروموزوم‌ها و خروجیِ تابع، میزان دقت مدل می‌باشد. می‌توان از محاسبه‌ی دقتِ مدل به‌عنوان، تابع شایستگی استفاده کرد که هدف غایی الگوریتم ژنتیک، بیشینه کردن تابع شایستگی می‌باشد. دقتِ مدل، حاصل تعداد رکوردهایی که به‌درستی توسط مدل پیشنهادی تشخیص داده‌شده (TP + TN) تقسیم بر تعداد کل رکوردها (TP + TN + FP + FN) می‌باشد. دقتِ مدل را با استفاده از فرمول ۱ محاسبه می‌شود.

$$Accuracy = \frac{TP+TN}{(TP+FN)+(TN+FP)}$$ فرمول ۱

در ابتدا از فضای حل مسئله، ۱۰۰ کروموزوم را به‌صورت تصادفی انتخاب می‌کنیم که این کروموزوم‌هایِ انتخابی، جمعیت اولیه را تشکیل می‌دهند. سپس تابع شایستگی را بر روی کروموزوم‌های جمعیت اولیه اجرا می‌کنیم و میزان شایستگی هر کروموزوم را محاسبه کرده و در گام بعدی بر اساس تابع انتخاب، دو کروموزوم را به‌عنوان والد برای عمل ترکیب و جهش انتخاب می‌کنیم.

۳,۲,٤ تابع انتخاب

همان‌طور که گفته شد پس از محاسبه میزان شایستگی کروموزوم‌ها باید دو کروموزوم را به‌عنوان والد برای عملیات ترکیب و جهش انتخاب کنیم. در اینجا از مکانیزم چرخ رولت به‌عنوان تابع انتخاب استفاده کرده‌ایم. مهم‌ترین مزیت استفاده از چرخ رولت این است که هر کروموزومی از جمعیت، شانس انتخاب شدن به‌عنوان والد را دارد، درواقع در مکانیزم چرخ رولت هر یک از کروموزوم‌ها بسته به میزان مناسب بودنش (بر اساس تابع شایستگی) احتمال انتخاب شدنش وجود دارد. به‌عبارت‌دیگر هر چه یک کروموزوم مقدار تابع شایستگی‌اش بیشتر باشد احتمال انتخاب شدنش برای تولید نسل بعدی بیشتر است و برعکس هر چه مقدار تابع شایستگی کروموزوم کم‌تر باشد، احتمال انتخاب شدنش برای تولید نسل بعدی کمتر است.

شیوه پیاده‌سازی چرخ رولت به این صورت است که با توجه به مقادیر تابع شایستگی هر کروموزوم، احتمال انتخاب کروموزوم بر اساس فرمول ۲ محاسبه می‌شود.

$$\text{Probability} = \frac{\text{Fitness(chromosome X)}}{\text{Sum Fitness (All chromosome)}}$$ فرمول ۲

با توجه به اینکه مجموعه احتمالات کروموزوم‌ها برابر ۱ است در قدم بعدی احتمال انتخاب کروموزوم‌ها را بر روی یک بردار به‌اندازه ۱ نگاشت می‌دهیم و در گام آخر یک عدد تصادفی بین ۰ تا ۱ تولید می‌کنیم. این عدد در هر بازه‌ای قرار بگیرید یعنی آن کروموزوم انتخاب‌شده است.

در گام دوم برای جلوگیری از حذف بهترین کروموزوم‌های نسل قبل از خاصیت نخبه‌گرایی استفاده کرده‌ایم و همیشه بهترین جواب نسل قبل را بدون هیچ تغییری به نسل جدید منتقل می‌کنیم.

٤,۲,٤ عملگر ترکیب

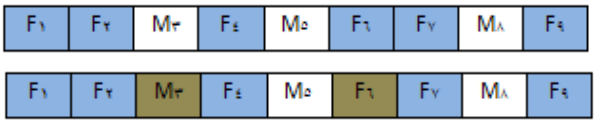
[1] Binary Encoding [8]
[1] Permutation Encoding [9]
[2] Valu Encoding [0]
[2] Tree Encoding [1]
[22] fitness function

این عملگر اطلاعات دو کروموزوم والد را گرفته و با ترکیب آن‌ها یک جفت کروموزوم جدید تولید می‌کند. درواقع در جریان عمل ترکیب، بخش‌هایی از کروموزوم‌ها با یکدیگر تعویض می‌شوند. این موضوع باعث می‌شود که فرزندان، ترکیبی از خصوصیات والدین خود را به همراه داشته باشند و از طرفی دقیقاً مشابه یکی از والدین نباشند. به عبارت بهتر هدف عملگر ترکیب تولید فرزند جدید می‌باشد به این امید که خصوصیات خوب دو موجود در فرزندشان جمع شده و موجود بهتری را تولید کنند. عملگر ترکیب می‌تواند به‌صورت تک نقطه‌ای، چندنقطه‌ای و یکنواخت باشد.

در میان روش‌های فوق روش تک نقطه‌ای دارای تنوع ژنتیکی کم‌تری نسبت به سایر روش‌ها می‌باشد و از طرفی ترکیب یکنواخت دارای تنوع ژنتیکی بیشتری نسبت به ترکیب‌های تک نقطه‌ای و چندنقطه‌ای می‌باشد به همین دلیل این نوع ترکیب در جمعیت‌هایی که اعضای کمی دارند اثر بهتری دارد تا جمعیت‌هایی که تعداد اعضای زیادی دارند؛ بنابراین با توجه به فضای مسئله، ما عملگر چندنقطه‌ای را انتخاب کردیم.

۵،۲،٤ عملگر جهش

مهم‌ترین وظیفه عملگر جهش جلوگیری از قرار گرفتن الگوریتم ژنتیک در بهینه‌های محلی می‌باشد. جهش نباید زیاد صورت بگیرد زیرا در این صورت الگوریتم ژنتیک به جستجوی کاملاً تصادفی تبدیل خواهد شد. در الگوریتم ژنتیک بعد از اینکه یک عضو در جمعیت جدید به وجود آمد، هر ژن آن با احتمال جهش[23]جهش می‌یابد. احتمال جهش $P_{Mutation}$ مقداری است که توسط کاربر تعیین می‌شود. در الگوریتم استاندارد ژنتیک مقدار این پارامتر، بسیار کوچک، مثل $P_{Mutation} = 0.01$ یا حتی $P_{Mutation} = 0.001$ در نظر گرفته می‌شود. در فرزندی که توسط عملگر ترکیب به وجود آمده است، به ترتیب مقداری تصادفی بین ۰ و ۱ به هر ژن اختصاص می‌یابد. اگر این مقدار اختصاص داده شده از $P_{Mutation}$ کمتر باشد، ژن جهش می‌یابد و اگر بیشتر باشد ژن تغییر نمی‌کند. در شکل ۸-4 نمونه‌ای از یک عمل جهش را مشاهده می‌کنید.



شکل 4: نمونه‌ای از عملگر جهش

٦،۲،٤ شرط خاتمه الگوریتم

برای اینکه تشخیص دهیم چه موقع الگوریتم از اجرا متوقف شود، از شیوه‌های مختلفی می‌توان استفاده کرد. به‌عنوان نمونه تولید نسل‌ها بیشتر N (مثلاً ۱۰۰۰ نسل) شود یا اگر در طی Y نسل (مثلاً ۵۰ نسل) حداکثر تابع برازندگی تغییر نکرد را به‌عنوان شرط خاتمه در نظر گرفت. در این پژوهش به علت بار زمان پردازش سریع‌تر و همچنین بالا رفتن ضریب دقت مدل، شرط خاتمه الگوریتم را این گذاشتیم که اگر در طی ۵۰ نسل حداکثر تابع برازندگی تغییر نکند.

ساخت مدل مبتنی بر الگوریتم جنگل تصادفی

بعد از تعیین پارامترهای الگوریتم جنگل تصادفی، وارد روال آموزش الگوریتم جنگل تصادفی می‌شویم. در این روال، الگوریتم جنگل تصادفی که شامل مجموعه‌ای از درختان تصمیم است بر اساس پارامترهای تعیین‌شده توسط الگوریتم ژنتیک، به یک جنگل بهینه برای آموزش می‌رسیم که این کار درنهایت منجر به افزایش نرخ تشخیص صحیح در سیستم تشخیص نفوذ می‌شود.

۳،٤ مدیر و کنترل‌کننده

در مؤلفه سوم از معماری سیستم تشخیص نفوذ پایگاه داده، مدیر و کنترل‌کننده قرار دارد. در این مؤلفه مقداردهی اولیه یک سری پارامترها، مدیریت فضای کاربر و اجرای فرمان‌ها صادرشده از سوی کاربر با برقراری ارتباط با مؤلفه‌های سیستم می‌باشد. لذا مشخص نمودن مقدار تابع شایستگی و شرط خاتمه الگوریتم

[23] Presumption Mutation - P_m

ژنتیک، برای تعیین پارامترها توسط کاربر انجام می‌گیرد که لازم است همان ابتدا تعیین شوند.

4. نتایج

مجموعه داده

با توجه به این‌که مجموعه داده مناسبی برای ارزیابی طرح پیشنهادی پیدا نکردیم، تصمیم بر این شد که مجموعه داده را خودمان تولید کنیم. برای این منظور ابتدا به بررسی ابزارهای معتبر آسیب‌پذیری تزریق SQL پرداختم و به‌صورت ویژه بر روی ابزار SQL Map که کدباز[24]می‌باشد تمرکز کردیم و با مراجعه به کد و تنظیمات نرم‌افزار توانستیم بردارهای حمله تزریق SQL را از این نرم‌افزار استخراج کنیم با این کار توانستیم تعداد ۲۰۰ بردار حمله به دست بیاوریم. در ادامه به چند مورد از این بردارهای حمله اشاره‌شده است.

1- [RANDNUM] = [RANDNUM]
2- NOT (RANDNUM]=[RANDNUM]
3- ELT([RANDNUM]=[RANDNUM], [ORIGVALUE])
4- RLIKE SLEEP([SLEEPTIME])
5- SLEEP([SLEEPTIME])

تولید پرس‌وجوهای نرمال

به‌منظور تولید پرس‌وجوهای نرمال، یک خزش‌گر پیاده‌سازی کردیم و برای خوراک اولیه خزش‌گر با دادن کلمات کلیدی مانند SQL، QUERY، TSQL و غیره صفحاتی که حاوی این کلمات کلیدی بوده را پیدا کرده و از میان محتوای صفحات دریافت شده با استفاده از چند عبارت منظم به استخراج پرس‌وجوهای موجود در آن‌ها پرداختیم. با این کار توانستم تعداد 4 هزار پرس‌وجوی نرمال غیرتکراری تهیه کنم.

تولید پرس‌وجوهای حمله

از بین 4 هزار پرس‌وجوی مرحله قبل یک لیست ۱۲۰ تایی از پرس‌وجوهای نرمال دارای قسمت Where Clause تهیه کردیم که تکراری هم نیستند و در گام بعدی با ترکیب بردارهای حمله با پرس‌وجوهای نرمال توانستیم درمجموع، 2 هزار پرس و جوی حمله غیرتکراری تولید کنیم.

پیش‌پردازش و نرمال‌سازی مجموعه داده‌ها

قبل از تحلیل مجموعه داده توسط الگوریتم جنگل تصادفی بهینه‌شده، مجموعه داده را بر اساس روش دبابارتا و همکارانش [۲۵] نرمال کردیم. درواقع با این کار تمامی پرس‌وجوها به شکل یکسان تبدیل شدند که این کار منجر به برقراری نظمی خاص در مجموعه داده شده که سرعت و دقت الگوریتم جنگل تصادفی را بالا برد.

آزمایش و ارزیابی

در پیاده‌سازی روش پیشنهادی، قبل از اجرای الگوریتم ژنتیک بر روی الگوریتم جنگل تصادفی، برخی از پارامترهای الگوریتم ژنتیک مانند شرط خاتمه و تابع انتخاب که از بقیه پارامترهای الگوریتم ژنتیک مهم‌تر هستند را تعیین کرده و مابقی پارامترهای این الگوریتم را در حالت پیش‌فرض اجرا کرده‌ایم. نتایجی که در ادامه نشان داده‌شده، در ۳ حالت شرط خاتمه را بررسی کردیم. حالت اول حداکثر تعداد نسل ۱۰۰۰ باشد. حالت دوم تابع شایستگی بعد از ۲۰ نسل بهبودی نداشته باشد و حالت سوم بعد از ۵۰ نسل تابع شایستگی بهبودی نداشته باشد.

جدول 2: تعیین شرط خاتمه در الگوریتم ژنتیک

معیار / شرط خاتمه حلقه	دقت	زمان
تولید حداکثر 1000 نسل	93/56	7m:15s
عدم بهبود تابع شایستگی پس از 20 نسل	90/83	2m:16s
عدم بهبود تابع شایستگی پس از 50 نسل	92/74	3m:28s

با توجه به‌دقت و مدت‌زمان طی شده تصمیم گرفتیم شرط خاتمه الگوریتم ژنتیک را بر روی عدم بهبودی تابع شایستگی در ۵۰ نسل بگذاریم.

به‌منظور تنظیم تابع انتخاب، روش چرخ رولت و حالت پیش‌فرض را امتحان کردیم.

[2] open source [4]

جدول 3: تعیین تابع انتخاب در الگوریتم ژنتیک

معیار / تابع انتخاب	دقت	زمان
حالت پیش‌فرض	98/60	14m:48s
روش چرخ رولت	98/60	12m:56s

با توجه به اینکه دقت یکسان است ولی مدت‌زمان چرخ رولت کم‌تر است، در نتیجه از تابع انتخاب چرخ رولت استفاده کردیم. در موارد فوق شرط خاتمه و تابع انتخاب الگوریتم ژنتیک را تنظیم کردیم، در ادامه الگوریتم ژنتیک را بر روی الگوریتم جنگل تصادفی به‌منظور تعیین پارامترهای بهینه اجرا کردیم و پارامترهای زیر به‌عنوان پارامترهای بهینه تشخیص داده شده است.

```
Random Forest.number_of_trees   = 21
Random Forest.criterion = accuracy
Random Forest.maximal_depth     = 19
Random Forest.minimal_gain      = 40.0
Random Forest.minimal_leaf_size = 22
Random Forest.apply_pruning     = false
Random Forest.guess_subset_ratio        = true
Random Forest.voting_strategy   = confidence vote
Random Forest.enable_parallel_execution = true
```

شکل 5: مقدار پارامترهای بهینه

در ادامه، روش پیشنهادی را با مقاله [۱۹] و اجرای الگوریتم در حالت پیش‌فرض را مورد بررسی و مقایسه قرار دادیم.

جدول 4: مقایسه روش پیشنهادی با سایر روش‌ها

معیار / الگوریتم	accuracy	precision	Recall	f_measure	Time
اجرای الگوریتم با پارامترهای پیش‌فرض	87/52	99/66	62/75	76/99	8m:12s
پیاده‌سازی مقاله مرجع [29]	90/30	97/59	72/70	83/25	1h:22m:30s
روش پیشنهادی	98/32	97/75	99/77	98/75	3h:10m:18s

آنچه از تحلیل‌ها و نتایج می‌توان استنباط کرد، برتری کامل روش پیشنهادی نسبت به روش ارائه‌شده در [۱۹] و همچنین روش الگوریتم جنگل تصادفی با پارامترهای پیش‌فرض در سه معیار Accyracy، Recall و F-measure است اما در معیار Precision روش الگوریتم جنگل تصادفی با پارامترهای پیش‌فرض عملکرد بهتری داشته است. اگرچه اکثر پارامترهای ارزیابی Accyracy، Recall و غیره نشان از برتری روش پیشنهادی دارد اما مدت‌زمان اجرای روش پیشنهادی تقریباً 1h:12m بیشتر از مقاله مرجع [۱۹] و 3h:2m بیشتر از اجرای الگوریتم با پارامترهای پیش‌فرض طول کشید. لازم به ذکر است که پس از تنظیم پارامترهای بهینه و اجرای مجدد الگوریتم جنگل تصادفی (پس از تعیین پارامترهای بهینه، الگوریتم ژنتیک را از مدل پیشنهادی حذف کنیم) با زمان اجرای 16m:21s به‌دقت حالت قبل می‌رسیم. در روش پیشنهادی با استفاده از الگوریتم فرا ابتکاری ژنتیک، پارامترهای بهینه مشخص شد که نتایج نشان از تأثیر بسیار زیاد این بهینه‌سازی بر روی سیستم تشخیص نفوذ پایگاه داده مبتنی بر الگوریتم جنگل تصادفی دارد. آنچه قابل‌توجه است دقت سیستم پیشنهادی نسبت به روش [۱۹] و الگوریتم جنگل تصادفی با پارامترهای بهینه است که به‌طور متوسط ۸ درصد دقت تشخیص نسبت به روش [۱۹] بهبود داده‌شده و حدود ۱۱ درصد نسبت به روش الگوریتم جنگل تصادفی با پارامترهای بهینه بهبود حاصل‌شده است و از ۸۷% به حدود ۹۸% در آزمایش-ها دست پیدا کردیم.

5. مراجع (References)

[۱] **https://www.statista.com**

[۲] **https://www.owasp.org**

[۳] **Umar, Kabir, Abu Bakar Sultan, Hazura Zulzalil, Novia Admodisastro, and Mohd Taufik Abdullah. "Formulation of SQL Injection Vulnerability Detection as Grammar Reachability Problem." In ۲۰۱۸ International Conference on Information and**

[13] N. Shone, T. Nguyen Ngoc, V. Dinh Phai, Q. Shi, “A Deep Learning Approach to Network Intrusion Detection”, TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, VOL. ٢, NO. ١, FEBRUARY, IEEE ٢٠١٨.

[14] Tang, TA, Mhamdi, L, McLernon, D et al, “Deep Learning Approach for Network Intrusion Detection in Software Defined Networking”, International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, Morocco. IEEE, ٢٠١٦.

[15] Bockermann, Christian, Martin Apel, and Michael Meier. "Learning sql for database intrusion detection using context-sensitive modelling." In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. ١٩٦-205. Springer, Berlin, Heidelberg, ٢٠٠٩.

[16] Resende, Paulo Angelo Alves, and André Costa Drummond. "A survey of random forest based methods for intrusion detection systems." ACM Computing Surveys (CSUR) ٥١, no. ٣ (٢٠١٨): ٤٨.

[17] Vorobeva, Alisa A. "Influence of features discretization on accuracy of random forest classifier for web user identification." In ٢٠١٧ ٢٠th Conference of Open Innovations Association (FRUCT), pp. ٤٩٨-504. IEEE, ٢٠١٧.

[18] Farnaaz, Nabila, and M. A. Jabbar. "Random forest modeling for network intrusion detection system." Procedia Computer Science ٨٩ (٢٠١٦): ٢١٣-217.

[19] Ronao, Charissa Ann, and Sung-Bae Cho. "Mining SQL queries to detect anomalous database access using random forest and PCA." In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. ١٥١-160. Springer, Cham, ٢٠١٥.

[20] Ronao, Charissa Ann, and Sung-Bae Cho. "Anomalous query access detection in RBAC-administered databases with random forest and PCA." Information Sciences ٣٦٩ (٢٠١٦): ٢٣٨-250.

[21] Aung, Yi Yi, and Myat Myat Min. "An analysis of random forest algorithm based network intrusion detection system." In ٢٠١٧ ١٨th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. ١٢٧-132. IEEE, ٢٠١٧.

Communication Technology for the Muslim World (ICT٤M), pp. ١٧٩-184. IEEE, ٢٠١٨.

[٤] Berk, Arslan, Rustam Olexandrovich Gamzayev, Ertuğrul Karaçuha, and Mykola Vyacheslavovich Tkachuk. "Algorithms and software solutions for SQL injection vulnerability testing in web applications." Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології ٢٢ (٢٠١٨): ٣-10.

[5] El Hayat, Soumiya Ain. "Intrusion Detection Systems: To an Optimal Hybrid Intrusion Detection System." In Smart Data and Computational Intelligence: Proceedings of the International Conference on Advanced Information Technology, Services and Systems (AIT٢S-١٨) Held on October ١٧–١٨, ٢٠١٨ in Mohammedia, p. ٢٨٤. Springer, ٢٠١٩.

[6] S. Khanna, and A. Verma, "Classification of SQL Injection Attacks Using Fuzzy Tainting", Progress in Intelligent Computing Techniques: Theory, Practice, and Applications. Springer, pp. ٤٦٣-469, ٢٠١٨.

[7] S. Steiner, D.Leon, and J. Alves-Foss, "A Structured Analysis of SQL Injection Runtime Mitigation Techniques", Proceedings of the ٥٠th Hawaii International Conference on System Sciences, ٢٠١٧.

[8] Liu, Guozhen. "SQL Injection Behavior Mining Based Deep Learning." In Advanced Data Mining and Applications: ١٤th International Conference, ADMA ٢٠١٨, Nanjing, China, November ١٦–١٨, ٢٠١٨, Proceedings, vol. ١١٣٢٣, p. ٤٤٥. Springer, ٢٠١٨.

[9] Dewa, Zibusiso, and Leandros A. Maglaras. "Data mining and intrusion detection systems." International Journal of Advanced Computer Science and Applications ٧,١ (٢٠١٦): ٦٢-71.

[10] M. Belhor, F Jemili, “Intrusion Detection Based on Genetic Fuzzy Classification System”, ٩٧٨-1-5090-4320-0/١٦, IEEE ٢٠١٦.

[11] S. Duquea, D. M. N. b. Omar, “Using Data Mining Algorithms for Developing a Model for Intrusion Detection System (IDS)”, Conference Organized by Missouri University of Science and Technology, San Jose, CA IEEE ٢٠١٥.

[12] S. E. Benaicha, L. Saoudi, S. E. B. Guermeche, O. Lounis, “Intrusion Detection System Using Genetic Algorithm”, Science and Information Conference August ٢٧-29, London, UK IEEE ٢٠١٤.

[22] **Sukumar, JV Anand, I. Pranav, M. M. Neetish, and Jayasree Narayanan. "Network Intrusion Detection Using Improved Genetic k-means Algorithm." In ۲۰۱۸ International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. ۲٤٤١-2446. IEEE, ۲۰۱۸.**

[23] **Hanmanthu, B. B. Raghu Ram, and P. Niranjan. "SQL Injection Attack prevention based on decision tree classification." In ۲۰۱۵ IEEE ۹th International Conference on Intelligent Systems and Control (ISCO), pp. ۱-5. IEEE, ۲۰۱۵.**

[24] **Sahasrabuddhe, Atmaja, Sonali Naikade, Akshaya Ramaswamy, Burhan Sadliwala, and Pravin Futane. "Survey on intrusion detection system using data mining techniques." Int Res J Eng Technol ٤, no. ٥ (۲۰۱۷): ۱۷۸۰-4.**

[25] **Kar, Debabrata, Suvasini Panigrahi, and Srikanth Sundararajan. "SQLiDDS: SQL injection detection using query transformation and document similarity." In International Conference on Distributed Computing and Internet Technology, pp. ۳۷۷-390. Springer, Cham, ۲۰۱۵.**

[۲٦] امینی، م : "تشخیص تهاجم بااستفاده از شبکه های عصبی" ، پایان نامه کارشناسی ارشد، دانشگاه صنعتی شریف، 19-7 ، 1382