دانشکده مهندسی کامپیوتر و فناوری اطلاعات

پایان‌نامه کارشناسی ارشد هوش مصنوعی و رباتیکز

رهیافتی هوشمند جهت کشف آسیب‌پذیری تزریق SQL

نگارنده: محمدحسین عموئی

استاد راهنما

دکتر محسن رضوانی

استاد مشاور

دکتر منصور فاتح

مرداد ۱۴۰۰

فرم شماره (۳) صورتجلسه نهایی دفاع از پایان نامه دوره کارشناسی ارشد

با نام و یاد خداوند متعال، ارزیابی جلسه دفاع از پایان نامه کارشناسی ارشد آقای **محمدحسین عموئی** با شماره دانشجویی **۹۷۰۰۲۵٤** رشته **مهندسی کامپیوتر** گرایش **هوش مصنوعی و رباتیکز** تحت عنوان **رهیافتی هوشمند جهت کشف آسیب‌پذیری تزریق SQL** که در تاریخ **۱٤۰۰/۰٦/۰۳** با حضور هیأت محترم داوران در دانشگاه صنعتی شاهرود برگزار شد به شرح ذیل اعلام می گردد:

الف) درجه عالی: نمره ۲۰-۱۹ ☑ ب) درجه خیلی خوب: نمره ۱۸/۹۹ – ۱۸ ☐

ج) درجه خوب: نمره۱۷/۹۹– ۱۶ ☐ د) درجه متوسط: نمره ۱۵/۹۹-۱۴ ☐

ه) کمتر از ۱۴ غیر قابل قبول و نیاز به دفاع مجدد دارد ☐

نوع تحقیق: نظری ☐ **عملی** ☐

عضو هیأت داوران	نام ونام خانوادگی	مرتبهٔ علمی	امضاء
۱ـ استاد راهنمای اول	محسن رضوانی	دانشیار	
۳- استاد مشاور	منصور فاتح	استادیار	
۴- استاد داور اول	حمید حسن‌پور	استاد	
۵- استاد داور دوم	علیرضا تجری	استادیار	
۶- نماینده تحصیلات تکمیلی	محسن فرهادی	مربی	

نام و نام خانوادگی رئیس دانشکده:

تاریخ و امضاء و مهر دانشکده

تقدیم به پدر و مادر عزیزم

خدای را بسی شاکرم که از روی کرم پدر و مادری فداکار نصیبم ساخته تا در سایه درخت پر بار وجودشان بیاسایم و از ریشه آنها شاخ و برگ گیرم و از سایه وجودشان در راه کسب علم ودانش تلاش نمایم. والدینی که بودنشان تاج افتخاری است بر سرم و نامشان دلیلی است بر بودنم چرا که این دو وجود پس از پروردگار مایه هستی ام بوده‌اند، دستم را گرفتند و مرا راه‌رفتن در این وادی زندگی پر از فراز و نشیب آموختند. آموزگارانی که برایم زندگی، بودن و انسان بودن را معنا کردند. حال این برگ سبزی است تحفه درویش تقدیم آنان.

به پاس تعبیر عظیم و انسانی شان از کلمه ایثار و از خودگذشتگان، به پاس عاطفه سرشار و گرمای امیدبخش وجودشان که در این سردترین روزگاران بهترین پشتیبان است، به پاس قلب های بزرگشان که فریاد رس است و سرگردانی و ترس در پناهشان به شجاعت می گراید و به پاس محبت های بی دریغشان که هرگز فروکش نمی کند.

این مجموعه را به پدر و مادر عزیزم تقدیم می کنم .

سپاس‌گزاری...

تشکر قلبی و لسانی خود را از استاد عالی قدر جناب آقای دکتر محسن رضوانی که زحمت راهنمایی این پایان نامه را عهده دار گردیدند و در تمامی مراحل انجام رساله از راهنمایی های مدبّرانه ایشان استفاده نمودم ابراز می دارم و توفیقات روز افزون ایشان را توأم با صحت و سعادت خواستارم.

از جناب آقای دکتر منصور فاتح که در امر مشاوره این رساله مساعدت نمودند و در این امر نهایت مراقبت، توجه و دقت خود را مبذول فرموده اند کمال تشکر و امتنان را دارم و برای ایشان از خداوند سلامت و سعادت ابدی را خواهانم.

در آخر از دوستان عزیزم که همواره در این مسیر پر چالش مرا همراهی کردند کمال قدردانی را دارم و از خداوند منان برایشان آرزوی موفقیت و سربلندی می‌کنم.

محمدحسین عموئی
مرداد ۱۴۰۰

تعهد نامه

اینجانب **محمدحسین عموئی** دانشجوی کارشناسی ارشد رشته **مهندسی کامپیوتر مهندسی کامپیوتر و فناوری اطلاعات** دانشگاه شاهرود، نویسنده پایان‌نامه با عنوان **رهیافتی هوشمند جهت کشف آسیب‌پذیری تزریق SQL** ، تحت راهنمایی **محسن رضوانی** متعهد می‌شوم:

- تحقیقات در این پایان‌نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهش‌های دیگر پژوهش‌گران، به مرجع مورد استفاده استناد شده است.
- مطالب این پایان‌نامه، تا کنون توسط خود، یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ‌جا ارایه نشده است.
- حقوق معنوی این اثر، به دانشگاه صنعتی شاهرود تعلق دارد، و مقالات مستخرج با نام ‘‘ دانشگاه صنعتی شاهرود’’ یا “ Shahrood University of Technology “ به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به‌دست آوردن نتایج اصلی پایان‌نامه تاثیرگذار بوده‌اند، در مقالات مستخرج از پایان‌نامه رعایت می‌گردد.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که از موجود زنده (یا بافت‌های آنها) استفاده شده است، ضوابط و اصول اخلاقی رعایت شده است.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته (یا استفاده شده است)، اصل رازداری و اصول اخلاق انسانی رعایت شده است.

محمدحسین عموئی
مرداد ۱۴۰۰

مالکیت نتایج و حق نشر

- تمام حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه‌های رایانه‌ای، نرم‌افزارها و تجهیزات ساخته شده) متعلق به دانشگاه صنعتی شاهرود می‌باشد. این مطلب باید به نحو مقتضی، در تولیدات علمی مربوطه ذکر شود.
- استفاده از اطلاعات و نتایج موجود در این پایان‌نامه بدون ذکر منبع مجاز نمی‌باشد.

چکیده

فایروال‌های وب برنامه‌هایی هستند که از برنامه‌های تحت وب در برابر حملات گوناگون محافظت می‌کنند. از آنجایی که روز به روز بر پیچیدگی این حملات افزوده می‌شود، فایروال‌ها برای اینکه بتوانند با این حملات مقابله کنند، نیاز است تا به‌صورت مداوم تست و بروزرسانی شوند. در عمل، استفاده از حملات brute-force برای کشف آسیب‌پذیری‌های فایروال به‌دلیل تنوع بسیار زیاد الگوهای حملات امکان‌پذیر نیست. بنابراین، تحقیقات بسیاری پیرامون ارائه‌ی روش‌های خودکار تست جعبه سیاه انجام شده است. اما روش‌های ارائه شده هنوز به بلوغ کافی نرسیده‌اند و در عمل از کارایی لازم برخوردار نیستند. بدین‌جهت، در این پژوهش ما روش تست جعبه سیاه خودکاری برای کشف آسیب‌پذیری‌های فایروال‌ها در مقابل حملات تزریق کد ارائه می‌دهیم. به‌طور خاص، در این پایان‌نامه بر روی دو حمله‌ي تزریق SQL و XSS تمرکز شده است که در دهه‌ی گذشته جز ده آسیب‌پذیری‌های برتر بوده‌اند.

در روش پیشنهادی ما، در فاز اول، با استفاده از n-gram رشته حملات موجود در مجموعه‌دادگان به زیررشته‌هایی تجزیه می‌شوند. در مرحله‌ی بعدی این زیررشته‌ها با استفاده از مدل skip-gram به بردارهای عددی تبدیل می‌شوند و پس از آن با استفاده از الگوریتم خوشه‌بندی سلسله‌مراتبی خوشه‌بندی می‌شوند. در مرحله آخر پس از تشکیل بردار ویژگی با استفاده از این خوشه‌ها و شبکه‌ی عمیق کدکننده‌ی خودکار، رشته حملات موجود در مجموعه‌دادگان خوشه‌بندی می‌شوند. در فاز دوم، در هرخوشه، زیررشته‌های کم اهمیت با محاسبه‌ی آنتروپی، کشف شده و از مجموعه‌ی ویژگی‌های آن خوشه حذف می‌شوند. در مرحله‌ی بعد، با استفاده از معکوس فراوانی سند برای هر زیررشته یک وزن محاسبه می‌شود که با کنار هم قراردادن این وزن‌ها بردار ویژگی هر رشته حمله بدست می‌آید. در فاز آخر، با استفاده از بردارهای ویژگی بدست آمده از فاز دوم، عملیات جستجو برای کشف آسیب‌پذیری‌ها انجام می‌شود. در این فاز، همزمان دو مرحله جستجوی برون خوشه‌ای و جستجوی درون خوشه‌ای انجام می‌شود. جستجوی برون خوشه‌ای با سیاست ϵ-greedy و به منظور یافتن خوشه‌هایی که شامل رشته حملات موفق هستند، انجام می‌شود. جستجوی درون خوشه‌ای اما به‌منظور کشف رشته حملات موفق درون خوشه‌ها انجام می‌شود.

در این پژوهش، روش پیشنهادی با استفاده از دو مجموعه‌داده شامل ۲۴۱۷۷۲۰ رشته حمله‌ی تزریق SQL و ۱۷۹۸۰۶۲ رشته حمله‌ی XSS با سه روش جدید ارائه شده در پژوهش‌ها با نام‌های ML-Driven E، ART4SQLi و XSSART مقایسه شده است. نتایج بدست آمده از آزمایش‌ها، نشان می‌دهد که روش پیشنهادی قادر است به‌طور میانگین ۳۳/۵۳ درصد آسیب‌پذیری‌های بیشتری نسبت به ML-Driven E کشف کند. همچنین روش پیشنهادی به‌طور متوسط با ۶۳/۱۶ درصد تعداد تلاش کمتر از ART4SQLi و XSSART می‌تواند اولین آسیب‌پذیری را کشف کند.

کلمات کلیدی: تست امنیت، حمله‌ی تزریق کد، تست تطبیقی، فایروال وب، خوشه‌بندی حملات، تزریق SQL

لیست مقالات مستخرج از پایان‌نامه

1. M. Amouei, M. Rezvani and M. Fateh, "RAT: Reinforcement-Learning-Driven and Adaptive Testing for Vulnerability Discovery in Web Application Firewalls," in *IEEE Transactions on Dependable and Secure Computing*, Accepted in July 2021. doi: 10.1109/TDSC.2021.3095417.

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مقدمه

در این فصل ابتدا به بیان مسأله و هدف تحقیق می‌پردازیم. پس از بررسی چالش‌های تحقیق و راه‌حل‌های آن، فرضیه‌های تحقیق و نوآوری‌های این پژوهش را ارائه می‌دهیم. در نهایت ساختار کلی پایان‌نامه بصورت مختصر ارائه می‌گردد.

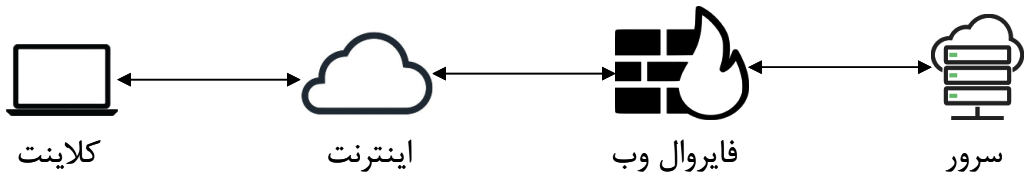
۱.۱ بیان مسأله

در سال‌های اخیر، با فراگیر شدن اینترنت، بسیاری از مشاغل به ارائه‌ی سرویس‌های برخط روی آورده اند. به عنوان مثال می‌توان به فروشگاه‌های برخط، بانکداری الکترونیک و شبکه‌های اجتماعی اشاره کرد. درنتیجه‌ی مهاجرت از روش‌های سنتی ارائه‌ی خدمات به روش‌های نوین، وابستگی مردم به اینترنت افزایش یافته و داده‌های خصوصی و محرمانه‌ی بسیار زیادی که متعلق به مردم و سازمان‌ها است، در پایگاه‌داده‌های برنامه‌های تحت وب ذخیره شده است. این داده‌های ذخیره شده در پایگاه‌داده‌های ارائه دهنده‌های خدمات برخط هدفی بسیار وسوسه‌کننده برای هکرها است، به‌طوری که طبق گزارشی، یک برنامه‌ی تحت وب ممکن است در هر دقیقه تا ۲۶ بار مورد حمله قرار بگیرد [۴۷]. علاوه بر این گزارشی که اخیراً منتشر شده است نشان می‌دهد که ۷۶ درصد وبسایت‌ها در مقابل حملات گوناگون آسیب‌پذیر هستند [۱۰]. بدین‌جهت، حفاظت از این برنامه‌ها امری حیاتی است.

یکی از رایج‌ترین دسته‌ی حملات که بسیار مخرب نیز هستند، تزریق کد نام دارد. حملات

تزریق کد انواع مختلفی دارند که هر کدام به‌تنهایی می‌تواند عواقب جبران ناپذیری به بار آورد، مانند افشای اطلاعات محرمانه یا در اختیار گرفتن کامل برنامه‌ی تحت وب. در این حملات، مهاجم تلاش می‌کند تا با ارسال کدهای مخرب بجای ورودی‌های مجاز منطق برنامه‌ی هدف را دچار مشکل کند. درنتیجه برنامه با پردازش این قطعه کدها عملیات مطلوب مهاجم را انجام می‌دهد. حملات تزریق SQL[۱] (SQLi) و XSS[۲] دو مورد از معروف‌ترین و پربحث‌ترین حملاتی هستند که در این دسته جای می‌گیرند [۲–۴، ۸، ۱۵، ۱۷، ۳۲، ۴۷–۴۹، ۵۱، ۵۲، ۵۹]. آسیب‌پذیری در مقابل این حملات در سال‌های اخیر جزو ده آسیب‌پذیری برتر بوده است.

یکی از راه‌های مؤثر در تأمین امنیت برنامه‌های وب استفاده از فایروال‌های وب[۳] است [۱، ۳۸]. فایروال‌های وب برنامه‌های مستقلی هستند که بین سرورهای ارائه دهنده‌ی خدمات تحت وب و کاربران قرار می‌گیرند و تمامی ترافیک‌های بین کاربران و سرورها را بررسی می‌کنند. نحوه‌ی قرارگیری فایروال وب در شبکه در شکل ۱.۱ نشان داده شده است. درصورتی که یک فایروال تشخیص دهد یک درخواست آلوده است می‌تواند از رسیدن آن درخواست به مقصد جلوگیری کند. این فایروال‌ها در تشخیص و جلوگیری از حملات تزریق کد بسیار مؤثر هستند. آن‌ها از مجموعه‌ای از قواعد دستور زبانی استفاده می‌کنند و چنانچه الگوی درخواستی با این قواعد مطابقت کند، آن درخواست آلوده تشخیص داده می‌شود.



شکل ۱.۱: نحوه‌ی قرارگیری فایروال وب در شبکه

مطالعات اخیر نشان می‌دهد که پیچیدگی حملاتی که به برنامه‌های تحت وب می‌شود دائماً در حال افزایش است [۲۲]. بنابراین تست و بروزرسانی مداوم فایروال‌های وب از اهمیت ویژه‌ای برخوردار است. چرا که با افزایش پیچیدگی الگوی حملات، قواعد استفاده شده در فایروال نیز باید بروزرسانی شود تا از کارایی آن کاسته نشود. اما با افزایش پیچیدگی حملات، تست فایروال‌ها نیز پیچیده و زمان‌بر می‌شود به‌طوری که بسیار سخت است که متخصصین بتوانند تمامی آسیب‌پذیری‌های فایروال را به‌صورت دستی در میان میلیون‌ها الگوی حمله کشف کنند. بدین‌جهت، نیاز است تا ابزارهایی طراحی شوند که بتوانند به‌صورت خودکار و بهینه آسیب‌پذیری‌های فایروال‌ها را کشف و گزارش کنند.

[1] SQL injection
[2] Cross-site scripting
[3] Web Application Firewall

۲.۱ هدف تحقیق

تست جعبه سیاه روشی است که بدون نیاز به دانستن منطق و ساختار برنامه قادر است مشکلات آن را بیابد. این روش به‌دلیل راحتی اجرا و اطلاعات ناچیزی که از برنامه لازم دارد، سالیان بسیار زیادی مورد توجه محققین بوده است و تلاش شده است تا کارایی این روش در طول سال‌ها همواره بهبود پیدا کند [۳۰]. اگرچه تست برنامه‌ها بدون نیاز به هیچگونه دانشی درمورد نحوه عملکرد آن ایده‌آل به نظر می‌رسد اما همین امر باعث می‌شود که عملکرد این تست در عمل بسیار کاهش یابد. به‌طوری که با وجود تمام تلاش‌هایی که برای توسعه‌ی روش تست جعبه سیاه شده است، مطالعات نشان می‌دهد که انواع روش‌های تست جعبه سیاه از کارایی کافی برخوردار نیستند و در تست‌های امنیتی با استفاده از این روش بسیاری از آسیب‌پذیری‌ها پنهان می‌مانند [۵، ۱۶].

با پیشرفت و توسعه‌ی روز افزون الگوریتم‌های هوش‌مصنوعی بسیاری از مسائل حل نشده، در سال‌های اخیر حل شدند و همین امر باعث شده است تا محققین سعی کنند با استفاده از تکنیک‌های هوش‌مصنوعی عملکرد روش‌های تست جعبه سیاه را افزایش دهند. هدف این پژوهش نیز مرتفع سازی برخی از چالش‌های تست جعبه سیاه و ارائه‌ی روشی هوشمند برای تست و کشف آسیب‌پذیری‌های فایروال‌های وب در مقابل حملات SQLi و XSS است.

۳.۱ چالش‌ها و راه حل‌های آن

در مشاهداتی که در این پژوهش انجام شد دریافتیم که کشف آسیب‌پذیری‌های فایروال‌های وب با استفاده از روش تست جعبه سیاه دارای دو چالش اساسی است. این چالش‌ها شامل محدودیت در تعداد جستجو و عدم وجود داده‌های برچسب‌دار و اطلاعات کافی درمورد نحوه‌ی عملکرد فایروال است. در ادامه به شرح این مشکلات می‌پردازیم.

- **محدودیت در تعداد جستجو:** در این روش تست، نمونه‌های مختلف حملات از مجموعه‌ی داده‌ها انتخاب می‌شود و هر بار هرکدام از آن‌ها از طریق اینترنت و با استفاده از پروتکل HTTP برای فایروال ارسال می‌شود تا بررسی شود که فایروال قادر به تشخیص آن حمله است یا خیر. تعداد درخواست‌هایی که از طرف ابزار تست برای فایروال ارسال می‌شود دارای محدودیت است و درصورتی که بسیار زیاد باشد خود نیز حمله تلقی می‌شود، چرا که درخواست‌های بسیار زیاد با فواصل زمانی کوتاه می‌تواند باعث شود که برنامه‌های تحت وب از دسترس خارج شوند. بنابراین، در این روش تست باید تعداد تلاش‌ها برای یافتن آسیب‌پذیری‌ها به حداقل کاهش یابد.
- **عدم وجود داده‌های برچسب‌دار و اطلاعات کافی:** از آنجایی که آسیب‌پذیری‌های فایروال‌ها با یکدیگر متفاوت هستند بنابراین امکان برچسب‌گذاری مجموعه داده‌ها برای

استفاده در الگوریتم‌های یادگیری ماشین وجود ندارد. از طرفی به‌دلیل تنوع بسیار زیاد الگوی حملات و کم‌یاب بودن حملاتی که بتوانند از فایروال عبور کنند، نیاز است تا هر بار حملات به‌صورت هدفمند و هوشمندانه انتخاب شوند. عدم دسترسی به ساختار و نحوه‌ی عملکرد فایروال تحت تست در این روش موجب می‌شود که ابزار اطلاعات کافی برای تصمیم‌گیری درباره‌ی نحوه‌ی انتخاب نمونه حملات برای تست بر روی فایروال در اختیار نداشته باشد. همین امر باعث می‌شود تا ابزار تست برای کسب دانش و پیش‌بینی حملات موفق درخواست‌های زیادی را به سمت سرور ارسال کند..

با توجه به چالش‌هایی که در روش تست جعبه سیاه وجود دارد نیاز است تا با استفاده از الگوریتم‌های هوش‌مصنوعی راهکاری ارائه شود که بتواند با مشاهدات اندک، دانشی بدست بیارد که با استفاده از آن بتواند نمونه حملات را به‌صورت هدفمند و هوشمندانه انتخاب کند. اگرچه در سال‌های اخیر روش‌هایی ارائه شده است که سعی کرده‌اند با استفاده از یادگیری ماشین کارایی این شیوه‌ی تست را بهبود بخشند [۲، ۳، ۹]، اما همچنان چالش برچسب‌گذاری داده‌ها حل نشده است و در این پژوهش‌ها برای برچسب‌گذاری داده‌ها از روش تست تصادفی استفاده می‌شود که خود باعث می‌شود درخواست‌های زیادی برای فایروال ارسال شود. بدین‌جهت، نیاز است تا روشی ارائه شود که بتواند بدون در اختیار داشتن داده‌های برچسب‌دار حملات موفق را پیش‌بینی کند. همچنین پژوهش‌های اخیر نشان داده‌اند که نمونه حملات موفق شبیه به یکدیگر هستند [۳۲، ۵۹]. بنابراین خوشه‌بندی آن‌ها می‌تواند حملات موفق را در تعداد کمی از خوشه‌ها قرار دهد. بنابراین شناسایی این خوشه‌ها و حذف سایر خوشه‌ها که فاقد حملات موفق هستند می‌تواند به کاهش تعداد مشاهدات مورد نیاز برای یادگیری الگوهای موفق کمک کند.

۴.۱ پیش‌فرض‌های تحقیق

در این بخش به طرح پیش‌فرض‌هایی که در این پژوهش درنظر گرفته شده‌اند می‌پردازیم.

- در این پژوهش الگوهای حملات رشته‌های[1] معنی‌داری هستند که از قاعده‌ی خاصی پیروی می‌کنند.
- در این پژوهش برای تشخیص موفقیت یا عدم موفقیت حملات از کدهای وضعیت درخواست HTTP[2] استفاده می‌کنیم. بر اساس RFC[3] کد ۴۰۳[4] برای پاسخ به یک درخواست غیرمجاز استفاده می‌شود. اگرچه این کد در تنظیمات فایروال قابل تغییر است، اما در این پژوهش ترجیح داده شده است تا از کد استاندارد استفاده شود.

[1] String
[2] HTTP status
[3] Request for Comments (RFC)
[4] Forbidden error

بنابراین، با فرض تنظیمات استاندارد، درصورتی که ابزار تست کد ۴۰۳ را در پاسخ دریافت کند می‌توان نتیجه گرفت که فایروال آن حمله را تشخیص داده است. در غیر این صورت، حمله از فایروال عبور کرده است.

- هدف این پژوهش تست فایروال است و نه برنامه‌ی تحت حفاظت آن. بنابراین قابلیت اجرا شدن حملات روی آن برنامه‌ی خاص اهمیتی ندارد. چراکه ساختار برنامه‌های مختلف با یکدیگر متفاوت است و فایروال‌ها به‌صورت مستقل از برنامه‌های تحت حفاظتشان به بررسی درخواست‌های HTTP می‌پردازند. اگرچه حملاتی که از فایروال عبور می‌کنند ممکن است روی برنامه‌های پشت فایروال اجرا نشوند، اما آن‌ها همچنان تهدیدی برای آن برنامه‌ها حساب می‌شوند چراکه ممکن است با اندکی تغییر قابلیت اجرا روی آن برنامه را پیدا کنند.

۵.۱ نوآوری‌های تحقیق

نوآوری‌های این پژوهش به‌طور کلی در ذیل شرح داده شده است:

- در این پژوهش یک روش استخراج ویژگی مبتنی بر n-gram جهت استخراج الگوهای پیچیده تشکیل‌دهنده‌ی رشته حملات ارائه شده است.
- همچنین راهکاری جهت خوشه‌بندی حملات، ارائه شده و تاثیر آن بر عملکرد روش پیشنهادی بررسی گردیده است.
- در نهایت نیز یک راهکار جستجوی تطبیقی با ترکیب سیاست ϵ-greedy و روش جستجوی مبتنی بر فراوانی وزنی کلمات[1] ارائه شده است.

۶.۱ ساختار پایان‌نامه

در ادامه‌ی این پایان نامه که در شش فصل ارائه خواهد شد، در فصل دوم به بررسی مدل حملات، فایروال‌ها و ضرورت تست آن‌ها و پژوهش‌های پیشین می‌پردازیم. در فصل سوم، مفاهیمی که در طراحی روش پیشنهادی استفاده شده‌اند شرح داده می‌شوند. در فصل چهارم، روش پیشنهادی به‌طور مفصل شرح داده می‌شود و درفصل پنجم به طرح پرسش‌های تحقیق و پاسخ به آن‌ها در قالب آزمایش‌های مختلف پرداخته می‌شود. درنهایت، فصل ششم نتیجه‌گیری و پیشنهاداتی برای پژوهش‌های آتی ارائه می‌دهد.

[1] Term frequency–Inverse document frequency (TF-IDF)

۷.۱ جمع‌بندی

در این فصل ابتدا به بیان مسأله و هدف تحقیق پرداختیم. سپس چالش‌های موجود و راه‌حل‌های آن‌ها را بررسی کردیم و پس از آن پیش‌فرض‌های این پژوهش را بیان کردیم. در انتها، ساختار پایان‌نامه را به اختصار شرح دادیم.

فصل ۲

ادبیات موضوع و کارهای پیشین

در این فصل، ابتدا به معرفی حملات و آسیب‌پذیری‌های مورد مطالعه در این پژوهش پرداخته می‌شود. سپس راه‌های مقابله با این حملات بررسی می‌شوند و پس از معرفی انواع روش‌های تست نفوذ خودکار، به مرور روش‌های کنونی تست نفوذ خودکار برنامه‌های تحت وب پرداخته می‌شود. در آخر، جمع‌بندی‌ای از مطالب بیان شده در این فصل ارائه می‌گردد.

۱.۲ معرفی حملات تزریق کد

حملات تزریق کد، حملاتی هستند که در آن، مهاجم تلاش می‌کند تا با تزریق ورودی‌های مخرب به یک برنامه تحت وب کنترل این برنامه‌ها را در دست بگیرد یا بدون احراز هویت به پایگاه داده‌ی برنامه دسترسی پیدا کند. این ورودی‌های مخرب قطعه کدهایی هستند که اجرای آن‌ها توسط مفسرهای برنامه می‌تواند محرمانگی[۱]، یکپارچگی[۲] و در دسترس بودن[۳] داده‌های برنامه را به خطر بی‌اندازد [۴۲].

حمله‌ی اصلی مورد بحث در این پژوهش، حمله‌ی SQLi است. با این حال، راهکار پیشنهادی این پژوهش قابل استفاده برای حملات مختلف از خانواده حملات تزریق کد است. از این‌رو، در این پژوهش علاوه بر حمله‌ی SQLi، به بررسی حمله‌ی XSS نیز می‌پردازیم.

[1] Confidentiality
[2] Integrity
[3] Availability

۱.۱.۲ حمله SQLi

پایگاه داده‌ها مسئول ذخیره و نگهداری انبوه اطلاعات مورد استفاده در برنامه‌های گوناگون تحت وب هستند. از انواع پایگاه داده‌ها می‌توان پایگاه داده رابطه‌ای، شی‌گرا، توزیع‌شده، NoSQL، نمودار، داده ابر، مرکزی و عملیاتی را نام برد. در میان این پایگاه داده‌ها، پایگاه داده رابطه‌ای که به‌عنوان پایگاه داده مبتنی بر زبان SQL نیز شناخته می‌شود محبوب‌ترین نوع پایگاه داده‌ها درمیان توسعه‌دهندگان وب هستند [۴۰].

SQL یک زبان پرس و جوی استاندارد برای ارتباط و کار با پایگاه داده‌های رابطه‌ای است. برنامه‌هایی که از این پایگاه داده‌ها استفاده می‌کنند، معمولاً پس از دریافت مقادیر ورودی از سوی کاربر آن‌ها را با رشته‌هایی از قبل شده ترکیب می‌کنند تا یک پرس و جو شکل دهند. سپس پرس و جو را برای پایگاه داده میفرستند تا آن را اجرا کند.

```
1  public void doPost(HttpServletRequest request, HttpServletResponse
       response)
2  throws ServletException, IOException {
3
4   String user_name = request.getParameter("username");
5   String user_pass = request.getParameter("password");
6
7   String sql_statement = "SELECT * FROM users WHERE username = '"
8   + user_name
9   + "' AND password = '"
10  + user_pass + "' ";
11
12  result = Database.execute(sql_statement)
13 }
```

شکل ۱.۲: نمونه‌ی تکمیل نا امن پرس و جو. در این قطعه کد ورودی‌ها بدون صحت سنجی به پرس و جو اضافه می‌شوند.

شکل ۱.۲ نمونه‌ای از تکمیل پرس و جو بدون صحت سنجی ورودی‌ها در سمت سرور را نمایش می‌دهد. در این شکل سرور مقادیر username و password را به‌عنوان ورودی از کاربر بدریافت می‌کند (خطوط ۴ و ۵). سرور پس از دریافت این مقادیر آن‌ها را بدون هیچگونه بررسی و اعتبار سنجی به پرس و جو اضافه می‌کند (خطوط ۷ تا ۱۰). در نهایت، در خط ۱۲ پرس و جو را برای پایگاه داده ارسال می‌کند. درصورتی که مقادیر username یا password

مخرب باشند، با اجرای این پرس و جو عمل مطلوب عامل مخرب انجام می‌گردد.

حمله‌ی SQLi حمله‌ای است که در آن مهاجم مقادیر مخربی را به‌عنوان ورودی برای یک برنامه‌ی تحت وب ارسال می‌کند تا عملیاتی غیر مجاز را انجام دهد. مهاجم می‌تواند از این آسیب‌پذیری استفاده کند تا به داده‌های محرمانه‌ی یک برنامه‌ی تحت وب دسترسی پیدا کند یا آن‌ها را تغییر دهد. به‌عنوان مثالی از حمله، فرض کنید در شکل ۱.۲ اگر مهاجم بجای نام کاربری مقدار '' OR 1 = 1 # را برای متغیر username و برای متغیر password مقدار خالی ارسال کند، پس و جوی نهایی پس از تکمیل توسط سرور به‌صورت زیر درمی‌اید:

```
SELECT * FROM users WHERE username = '' OR 1 = 1 # AND password = ''
```

در زبان SQL نماد # عملگر کامنت است. درنتیجه هرچه بعد از آن باشد به‌عنوان کامنت محسوب می‌شود و در عمل پرس و جوی ما به‌صورت زیر خواهد بود:

```
SELECT * FROM users WHERE username = '' OR 1 = 1
```

رشته‌ی 1 = 1 در این زبان یک شرط همیشه درست است وپایگاه داده پس از اجرای پرس و جوی بالا تمام اطلاعات جدول users را در اختیار مهاجم قرار می‌دهد.

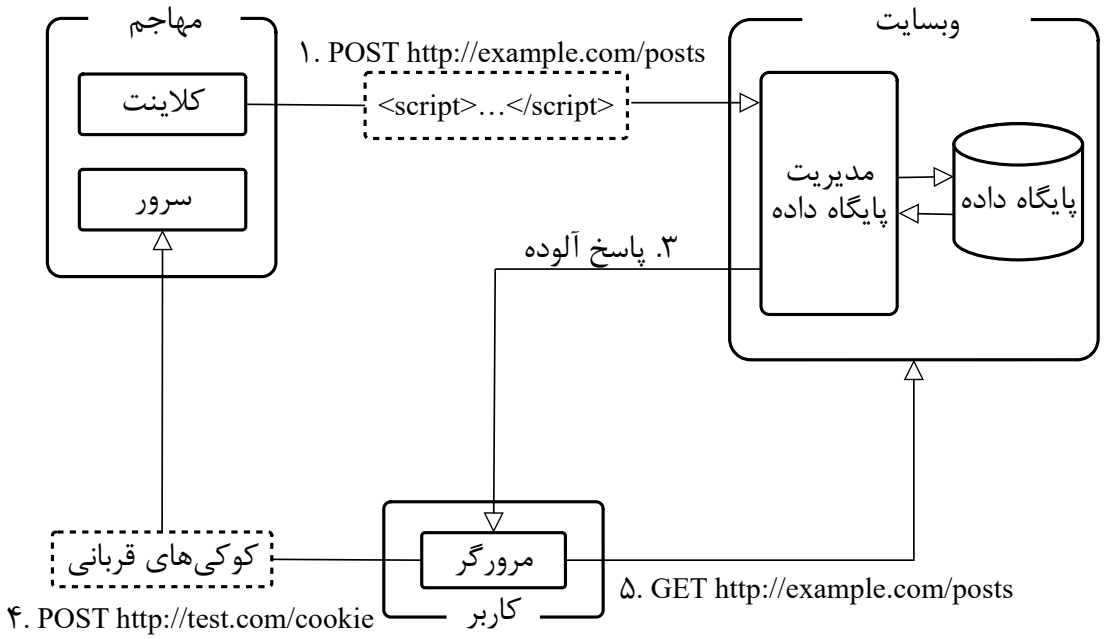
آسیب‌پذیری درمقابل حمله‌ی SQLi بسیار پرریسک است. درصورت موفقیت یک حمله‌ی SQLi، ممکن است اطلاعات محرمانه‌ی یک سازمان یا کاربران حقیقی به سرقت رفته یا حذف یا ویرایش شود. بنابراین، محافظت از برنامه‌های تحت وب دربرابر این حملات امری حیاتی است.

۲.۱.۲ حمله XSS

حمله‌ی XSS نوعی حمله‌ی تزریق کد است که در آن مهاجم سعی می‌کند تا به نحوی کد مخرب خود را در رابط کاربری یک وبسایت قابل اعتماد قرار بدهد. درنتیجه کد مخرب برروی مرورگر کاربر قربانی اجرا می‌شود و مهاجم به اهداف خود دست می‌یابد. تمامی سایت‌هایی که خروجی‌های آن‌ها به نوعی وابسته به ورودی‌هایی است که از کاربران خود دریافت می‌کنند، می‌توانند در مقابل این حمله آسیب‌پذیر باشند. در این حمله به‌دلیل اینکه مرورگر تصور می‌کند وبسایت آلوده مورد اطمینان است، به هیچ عنوان قادر به تشخیص حمله نیست و تمام کدهایی که در رابط کاربری آن وبسایت هست را اجرا می‌کند. درنتیجه‌ی موفقیت یک حمله‌ی XSS، مهاجم می‌تواند به کوکی‌ها[1]، توکن‌های نشست[2] و هر اطلاعات محرمانه‌ی دیگری که توسط مرورگر جمع آوری می‌شود دسترسی پیدا کنید. مهاجم حتی می‌تواند از این طریق اطلاعات حساب اشخاص را به هنگام انجام عملیات پرداخت بانکی به سرقت ببرد. حمله‌ی XSS دو نوع اصلی دارد که در ادامه بصورت مختصر آن‌ها را توضیح می‌دهیم.

[1] Cookie
[2] Session tokens

- **ذخیره شده:**[1] در این نوع حمله کدهای مخرب در سرور وبسایت ذخیره می‌شوند. به‌عنوان مثال مهاجم می‌تواند در قسمت نظرات یک سایت کد مخرب خود را ارسال کند. این کد در پایگاه داده آن وبسایت ذخیره می‌شود و هنگامی‌که سایر کاربران به آن صفحه مراجعه کنند قربانی این حمله می‌شوند.
- **منعکس شده:**[2] در برخی از وبسایت‌ها مقادیر پارامترهای درخواست HTTP مستقیماً به رابط کاربری اضافه می‌شوند (مانند موتورهای جستجو). در این نوع حمله، مهاجم کد مخرب خود را به آدرس درخواست اضافه می‌کند. سپس این آدرس را از راه‌های مختلف مانند ایمیل برای کاربر هدف ارسال می‌کند. درصورتی که کاربر بر روی این لینک به ظاهر مطمئن کلیک کند قربانی این حمله می‌شود.



شکل ۲.۲: مثالی از فرآیند یک حمله‌ی XSS که در آن مهاجم پست آلوده‌ای را در پایگاه داده‌ی یک شبکه‌ی اجتماعی ذخیره می‌کند.

شکل ۲.۲ یک نمونه از حمله‌ی XSS ذخیره شده را نمایش می‌دهد. مراحل این مثال در ادامه توضیح داده شده است.

۱. مهاجم ابتدا بجای محتوای سالم، یک کد مخرب را در یک شبکه‌ی اجتماعی پست می‌کند. این کد سپس در پایگاه داده‌ی این شبکه‌ی اجتماعی ذخیره می‌شود.

۲. کاربری دیگر می‌خواهد محتویات ارسال شده در شبکه‌ی اجتماعی را مشاهده کند. بنابراین، درخواستی برای این وبسایت ارسال می‌کند تا آخرین پست‌ها را دریافت کند.

[1] Stored
[2] Reflected

```
<!DOCTYPE html>
<html>
<h1> Latest Posts </h1>
...
<script>
  $(document).ready(function() {
      var cookie = document.cookie;
      $.post("http://test.com/cookie",
      {cookie: cookie},
      function(result) {});
  });
</script>
...
</html>
```

شکل ۳.۲: نمونه‌ی کد HTML که شامل کد مخربی است که کوکی‌های کاربر را به سرقت می‌برد.

۳. شبکه‌ی اجتماعی بدون آگاهی صفحه‌ای شامل آخرین پست‌ها را برای کاربر ارسال می‌کند. شکل ۳.۲ مثالی از این صفحه را نمایش می‌دهد. در این مثال، خطوط ۵ تا ۱۲ کد مخرب ذخیره شده در پایگاه داده‌ی این شبکه اجتماعی است. این کد پس از اجرا کوکی‌های کاربر را از مرورگر خوانده و برای مهاجم ارسال می‌کند.

۴. کاربر به محض مشاهده‌ی صفحه‌ی دریافتی، کوکی‌های خود را بدون آگاهی در اختیار مهاجم قرار می‌دهد. مهاجم ممکن است از این کوکی‌ها استفاده کند تا بجای این کاربر در شبکه‌ی اجتماعی فعالیت کند.

عواقب تمامی انواع حمله‌ی XSS یکسان است و تنها تفاوت آن‌ها نحوه‌ی پیاده‌سازی آن‌ها است. درصورت موفقیت این حمله، مهاجم می‌تواند اطلاعات محرمانه کاربران را به سرقت ببرد، سیستم کاربران را ویروسی کند و یا محتویات وبسایت‌ها را تغییر دهد. از این‌رو لازم است تا برنامه‌های تحت وب در مقابل این حمله محافظت گردند.

۲.۲ نحوه مقابله با حملات تزریق کد

برای مقابله با حملات تزریق کد راه‌های گوناگونی وجود دارد. این راه‌ها معمولاً در دو دسته‌ی تحلیل و پاکسازی ورودی‌های یک برنامه قرار می‌گیرند. تحلیل ورودی‌ها بدین معنی است که

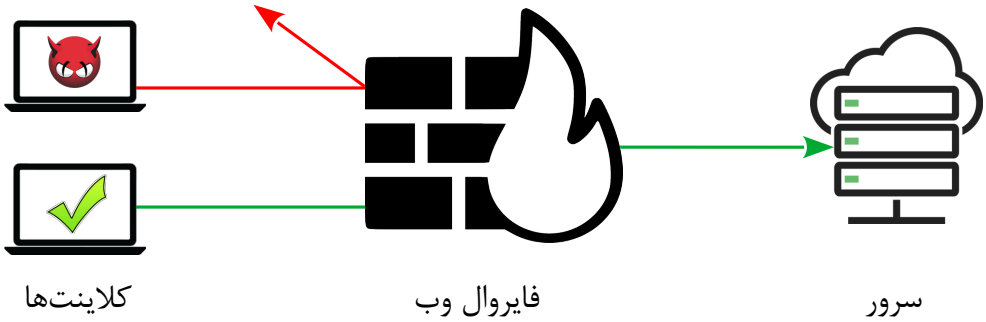
یک برنامه باید ورودی‌های خود را قبل از ذخیره و استفاده پردازش کند تا درصورت مشاهده‌ی ورودی‌های مخرب اقدام‌ها لازم را به عمل آورد. بدین منظور معمولاً برنامه‌ها از دو روش لیست سیاه یا عبارات منظم استفاده می‌کنند. در روش لیست سیاه کلمات کلیدی در یک لیست قرار می‌گیرند و درصورت مشاهده آن‌ها در ورودی، آن ورودی مخرب شناسایی می‌شود. اما در روش عبارات منظم که پیچیده‌تر است، الگوهای حملات با عبارات منظم مدل می‌شوند و در هر بار دریافت ورودی از کاربر این ورودی‌ها با عبارات منظم از پیشتعریف شده مطابقت داده می‌شوند. روش‌های مبتنی بر پاکسازی ورودی‌ها تلاش می‌کنند تا با اصلاح ورودی‌های برنامه آن‌ها را به گونه‌ای تغییر دهند که درصورت وجود کد مخرب، آن کد دیگر قابل اجرا نباشد.

توسعه‌ی برنامه‌های تحت وب که از امنیت بالایی برخوردار باشند نیازمند این است که توسعه دهندگان از دانش امنیت برخوردار باشند. اما بسیاری از توسعه دهندگان از دانش لازم نسبت به انواع آسیب‌پذیری‌ها و راه‌های مقابله با آن‌ها برخوردار نیستند. همین امر سبب شده است تا اکثر از برنامه‌های تحت وب در مقابل انواع مختلفی از حملات آسیب‌پذیر باشند [۱۰]. ویرایش سورس کدهای این برنامه‌ها جهت ایمن سازی آن‌ها کاری پرهزینه، دشوار و زمان‌بر است. بدین‌جهت، سازمان‌ها که عموماً میزبان برنامه‌های تحت وب متعددی هستند، برای تأمین امنیت خدماتشان از برنامه‌هایی تحت عنوان فایروال استفاده می‌کنند. این برنامه‌ها که در بین سرورها و کلاینت‌ها قرار می‌گیرند وظیفه‌ی تشخیص و جلوگیری از حملات گوناگون را بر عهده دارند. در ادامه به‌طور مفصل به بررسی فایروال‌های وب می‌پردازیم.

۱.۲.۲ فایروال وب

فایروال‌های وب برنامه‌هایی هستند که از سرویس‌های تحت وب درمقابل حملات گوناگون از جمله انواع حملات تزریق کد محافظت می‌کنند. این فایروال‌ها همانطور که در شکل ۴.۲ نشان داده شده است، بین سرورها و کلاینت‌ها قرار می‌گیرند. هر درخواستی که از سوی کلاینت‌ها برای سرورها ارسال می‌شود، ابتدا توسط فایروال بررسی می‌شود. چنانچه درخواست سالم و مجاز شناسایی شود به سوی سرور مقصد هدایت می‌گردد. فایروال‌های وب معمولاً به دودسته‌ی فایروال‌های مبتنی بر قواعد[۱] و مبتنی بر یادگیری ماشین[۲] تقسیم‌بندی می‌شوند. فایروال‌های مبتنی بر یادگیری ماشین از الگوریتم‌های هوش‌مصنوعی استفاده می‌کنند تا ناهنجاری‌ها را در درخواست‌ها تشخیص دهند و براساس آن تصمیم بگیرند که یک درخواست سالم است یا آلوده. فایروال‌های مبتنی بر قواعد که هدف آزمایش در این پژوهش نیز هستند از یک سری عبارات منظم استفاده می‌کنند تا حملات را تشخیص دهند. به‌عنوان مثال، عبارات /**/ و # در زبان SQL عملگرهای کامنت هستند. بدین معنی که هرچه بعد از آن‌ها بیاید کامنت درنظر گرفته شده و اجرا نمی‌شود. حال برای یافتن عملگرهای

[1] Rule-based
[2] Machine-learning-based



شکل ۴.۲: نحوه‌ی محافظت از سرورها توسط فایروال وب.

کامنت در درخواست‌های کاربر یک فایروال از قاعده‌ی زیر استفاده می‌کند [۳]. درنتیجه اگر درخواستی با الگوی زیر مطابقت کند می‌تواند به‌عنوان درخواست مخرب درنظر گرفته شود.

```
/\*!?|\*/|[';]--|--[\s\r\n\v\f]|(?:--[^-]*?-)|
                ([^\&])#.*?[\s\r\n\v\f]|;?\\x00
```

یک فایروال وب به دلایل گوناگونی می‌تواند دارای آسیب‌پذیری باشد. مهم‌ترین دلایلی که می‌تواند موجب آسیب‌پذیری یک فایروال وب شود شامل پیاده‌سازی نادرست یا وجود نقص در تنظیمات آن است. از این‌رو، لازم است تا فایروال‌ها همواره آزمایش شوند تا از عملکرد صحیح آن‌ها اطمینان حاصل شود. در گذشته این کار توسط یک متخصص انجام می‌شد. اما با افزایش چشمگیر تنوع و پیچیدگی‌حملات، عملاً آزمایش فایروال‌های وب به وسیله‌ی انسان نمی‌تواند نتایج قابل اعتمادی را فراهم کند [۳]. بنابراین، طراحی روش‌های تست خودکار این فایروال‌ها امری حیاتی است.

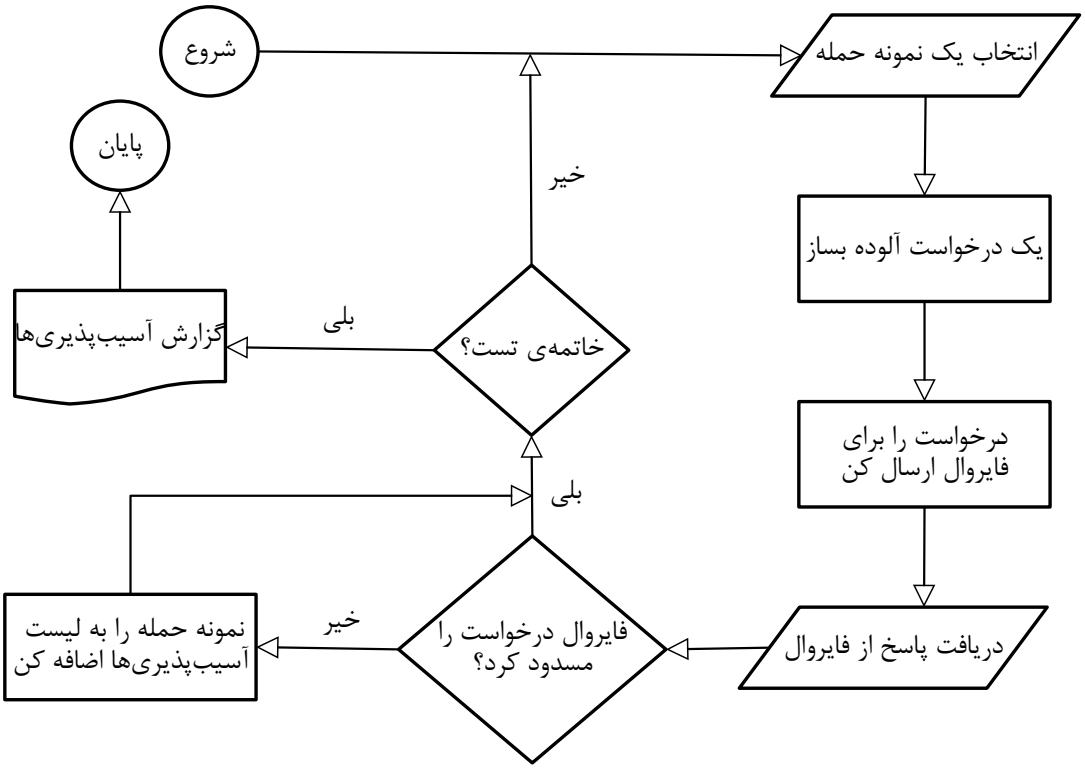
۳.۲ روش‌های کشف خودکار آسیب‌پذیری‌ها

در دهه‌ی اخیر تلاش‌های فراوانی صرف ارائه و توسعه‌ی روش‌های کارآمد جهت یافتن خودکار آسیب‌پذیری‌های برنامه‌ها شده است [۲۹، ۴۶]. این روش‌ها در دسته‌های مختلفی قرار می‌گیرند که سه تا از مهم‌ترین آن‌ها روش جعبه سفید، روش مبتنی بر مدل و روش جعبه سیاه هستند. در ادامه هرکدام از این روش‌ها به‌صورت مختصر توضیح داده شده‌اند.

- **روش جعبه سفید:** در این روش سورس کدهای برنامه‌ها برای یافتن حفره‌های امنیتی بررسی می‌شوند. از آنجایی که دراین روش دسترسی مستقیم به سورس برنامه‌ها وجود دارد بنابراین این روش می‌تواند بسیاری از آسیب‌پذیری‌های موجود در برنامه را بیابد. اما روش جعبه سفید مشکلاتی نیز دارد. یکی از مهم‌ترین مشکلات این روش وابستگی آن به زبان برنامه نویسی استفاده شده برای توسعه‌ی برنامه‌ی مورد آزمایش هست [۳۱].

یکی دیگر از معایب این روش این است که در بسیاری از موارد دسترسی به سورس کد برنامه به دلایل مختلفی از حمله محرمانگی امکان‌پذیر نیست [۱۹].

- **روش مبتنی بر مدل:** این روش نیاز دارد تا مدلی از نحوه‌ی پیاده‌سازی و سیاست‌های مورد استفاده برای تأمین امنیت برنامه را در اختیار داشته باشد. سپس با تحلیل و بررسی این مدل سعی در یافتن حفره‌های امنیتی موجود می‌کند. متأسفانه این مدل‌ها معمولاً در دسترس نیستند و طراحی آن‌ها به‌صورت دستی امری بسیار زمان‌بر و دشوار است [۳۱].



شکل ۵.۲: فرآیند انجام تست جعبه سیاه روی فایروال وب.

- **روش جعبه سیاه:** در این روش سعی می‌شود تا با بررسی و اعمال الگوهای مختلف حملات آسیب‌پذیری‌های موجود در برنامه‌ها نمایان گردند. شکل ۵.۲ فرآیند تست فایروال وب با استفاده از روش جعبه سیاه را نمایش می‌دهد. این روش هیچکدام از محدودیت‌های دو روش قبلی را ندارد. اما همین امر باعث می‌شود تا در این روش اطلاعات بسیار محدودی از برنامه‌ی تحت آزمایش در اختیار باشد. بنابراین دقت این روش در یافتن آسیب‌پذیری‌ها بسیار پایین است [۲۸، ۳۴]. با این حال، عدم وجود محدودیت‌های مذکور در این روش سبب جذابیت هرچه بیشتر روش تست جعبه سیاه

شده است و تحقیقات زیادی جهت افزایش کارایی این روش انجام شده است. روش ارائه شده در این پژوهش نیز در این دسته قرار می‌گیرد.

۴.۲ کارهای پیشین

در یک دهه‌ی گذشته حملات تزریق کد بخصوص حملات XSS و SQLi بسیار مورد توجه محققین قرار گرفته است. درنتیجه، تلاش‌های بسیاری برای ارائه‌ی راهکارهای کشف خودکار آسیب‌پذیری مقابل این حملات شده است [۱۹]. درمیان این روش‌ها روش‌های متعددی برای تست جعبه سیاه نیز ارائه شده‌اند که انواع گوناگونی دارند. در این فصل ما به بررسی چندتا از مهم‌ترین انواع این روش‌ها از جمله روش‌های تست ترکیبی[۱]، جستجوی تطبیقی[۲] و مبتنی بر هوش‌مصنوعی[۳] می‌پردازیم.

۱.۴.۲ روش تست ترکیبی

تست ترکیبی که به آن تست τ-way نیز گفته می‌شود روشی است که برای تست ترکیب‌های مختلف از پارامترهای یک سیستم استفاده می‌گردد. به‌طور دقیق‌تر، اگر سیستمی دارای تعداد n ورودی باشد، در این روش تمامی ترکیب‌های τ تایی ورودی‌های این سیستم ارزیابی می‌شود. این روش می‌تواند به صورت قابل ملاحظه‌ای تعداد تست‌ها را کاهش دهد. به‌عنوان مثال، اگر سیستمی دارای چهار ورودی $C_i(i = ۱,۲,۳,۴)$ باشد و هر ورودی سه مقدار $V = \{a, b, c\}$ را قبول کند. درصورتی که بخواهیم تمام ترکیب‌های ممکن این ورودی‌ها را تست کنیم باید تعداد ۸۱ تست انجام دهیم. درصورتی که اگر بخواهیم فقط ترکیب‌های دوتایی از ورودی‌ها ($\tau = ۲$) را درنظر بگیریم کافی است فقط تعداد ۹ تست انجام دهیم. در ادامه به شرح جزئیات این مثال می‌پردازیم.

در مثالی که پیش‌تر مطرح شد، در هربار تست تمامی‌پارامترهای ورودی سیستم مقدار دهی می‌شوند. اما این مقادیر به گونه‌ای انتخاب می‌شوند که تمامی ترکیب‌های دوتایی پوشش داده شوند. جدول ۱.۲ نحوه‌ی ترکیب مقادیر ورودی‌های مثال مطرح شده را برای $\tau = ۲$ نمایش می‌دهد. در این مثال، همانطور که مشاهده می‌شود با انجام ۹ تست می‌توان تمام حالات دوتایی زیر را مورد آزمایش قرار داد.

$$T = \{(C_1, C_2), (C_1, C_3), (C_1, C_4), (C_2, C_3), (C_2, C_4), (C_3, C_4)\}$$

در سال‌های اخیر با توجه به عملکرد خوب روش تست ترکیبی در بررسی عملکرد سیستم‌های نرم‌افزاری راهکارهای متنوعی مبتنی بر این روش برای کشف خودکار آسیب‌پذیری‌های XSS

[1] Combinatorial testing
[2] Adaptive search
[3] Artificial-Intelligence-Driven

جدول ۱.۲: نمونه‌ی ترکیب ورودی‌های یک سیستم به گونه‌ای که تمام ترکیب‌های دو تایی را پوشش دهد.

ترکیب‌های ممکن از مقادیر ورودی‌ها									پارامترهای ورودی
c	c	c	b	b	b	a	a	a	C_1
b	a	c	a	c	b	c	b	a	C_2
a	c	b	b	a	c	c	b	a	C_3
c	b	a	c	b	a	c	b	a	C_4

و SQLi ارائه شده است. در این روش‌ها دستور زبانی[1] برای تولید الگوهای حملات طراحی می‌شود و قواعد این دستور زبان‌ها به‌عنوان ورودی‌های سیستم درنظر گرفته می‌شوند. در ادامه به بررسی چند نمونه از این روش‌ها می‌پردازیم.

در سال ۲۰۱۵، بوزیچ[2] و همکاران دستور زبان‌های موجود برای تولید حملات XSS را طوری بازطراحی کردند که کارایی روش تست ترکیبی برای کشف خودکار آسیب‌پذیری‌های XSS در برنامه‌های تحت وب افزایش یابد. همچنین آزمایش‌های آن‌ها نشان داد که هرچه مقدار τ در تست ترکیبی بیشتر باشد تعداد آسیب‌پذیری‌های بیشتری را می‌توان کشف کرد [۸].

حملات تزریق کد رشته‌هایی هستند که از تعدادی زیر رشته‌هایی با الگوهای منظم و معین تشکیل شده اند. معمولاً ترکیب‌های مختلفی از زیر رشته‌ها، که فقط بخشی از حمله‌ی تزریق کد را تشکیل داده باشند، ممکن است باعث موفقیت آن حمله شود. از این‌رو شناسایی این ترکیب‌ها می‌تواند در رفع حفره‌های امنیتی بسیار کارآمد باشد. در پژوهشی در سال ۲۰۱۶، سیموس[3] و همکاران با گسترش مدلی که یک سال قبل در مقاله‌ی بوزیچ ارائه شده بود، راهکاری جهت شناسایی ترکیب‌های τ تایی از زیر رشته‌های تشکیل‌دهنده‌ی حملات XSS، که موجب موفقیت حملات شود، ارائه دادند [۴۸].

همانند تلاش‌هایی که برای کشف آسیب‌پذیری‌های XSS با روش تست ترکیبی شد، راهکارهایی نیز برای کشف آسیب‌پذیری‌های SQLi با استفاده از این روش ارائه شده است. در سال ۲۰۱۹، در [۴۹] سه دستور زبان تولید حملات SQLi برای استفاده در روش تست ترکیبی ارائه شد. یک سال پس از آن، ژائو[4] و همکاران روشی مبتنی بر تست ترکیبی برای یافتن آسیب‌پذیری‌های SQLi پیشنهاد دادند [۶۰]. در این روش، بجای اینکه قواعد دستور زبان به‌عنوان پارامترهای تست ترکیبی درنظر گرفته شود، تعدادی عملگر خاص بدین منظور درنظر

[1] Grammar
[2] Bozic
[3] Simos
[4] Zhao

گرفته شدند. وظیفه‌ی این عملگرها تغییر نمونه حملات موجود است، به‌گونه‌ای که احتمال موفقیت آن‌ها افزایش یابد.

اگرچه روش تست ترکیبی با ترکیب‌های دو یا سه تایی برای آزمایش عملکرد ورودی‌های یک نرم‌افزار از کارایی بالایی برخوردار است، اما به‌دلیل پیچیدگی مقابله با حملات تزریق کد افزایش مقدار τ می‌تواند به‌صورت قابل ملاحظه‌ای کارایی تست را در کشف آسیب‌پذیری‌ها بهبود بخشد [۸]. با این‌حال اساسی‌ترین مشکل این روش این است که افزایش مقدار τ سبب ازدیاد تعداد حالت‌ها و در نتیجه کاهش بهینگی آن می‌شود.

۲.۴.۲ روش جستجوی تطبیقی

در روش‌های مبتنی بر جستجوی تطبیقی تلاش می‌شود تا در هر بار انتخاب یک نمونه حمله برای تست روی سیستم تحت آزمایش، آن نمونه با توجه به موفقیت یا عدم موفقیت نمونه حملات تست شده‌ی قبلی انتخاب شود. بدین‌منظور، معمولاً در این روش‌ها، که فقط کمی از روش جستجوی تصادفی بهتر هستند، از یک معیار فاصله استفاده می‌شود تا نمونه‌ی بعدی طوری انتخاب شود که بیشترین فاصله را از حملات ناموفق قبلی داشته باشند.

ژنگ[1] و همکاران در سال ۲۰۱۹ روشی را تحت عنوان ART4SQLi [۵۹] ارائه دادند. در این روش، در هر بار آزمایش تعدادی نمونه رشته‌ی SQLi از مجموعه داده به‌صورت تصادفی انتخاب می‌شود. از میان این نمونه‌ها آن نمونه که کمترین فاصله‌ی آن از نمونه‌های ناموفق قبلی بیشتر از سایر نمونه‌های انتخاب شده باشد، انتخاب می‌شود. درصورتی که نمونه‌ی جدید بتواند از تمهیدات امنیتی سیستم تحت آزمایش عبور کند، آن نمونه یک حمله‌ی موفق شناخته شده و الگوریتم خاتمه می‌یابد. درصورت عدم موفقیت نمونه‌ی منتخب، آن نمونه به مجموعه حملات ناموفق اضافه می‌شود و تست ادامه پیدا می‌کند.

همانند روش ART4SQLi، روشی برای تست آسیب‌پذیری XSS نیز تحت عنوان XSSART [۳۲] ارائه شده است. در این روش نیز از معیار فاصله برای رتبه بندی نمونه‌های منتخب و در نهایت انتخاب یک نمونه با بهترین رتبه از میان آن‌ها برای تست روی سیستم تحت آزمایش استفاده شده است. همچنین در آزمایش‌های این روش نیز پس از کشف اولین حمله‌ی موفق الگوریتم خاتمه می‌یابد.

هدف هر دو روش ART4SQLi [۵۹] و XSSART [۳۲] افزایش سرعت پیداکردن اولین حمله‌ی موفق است. یافتن اولین حمله‌ی موفق بدین‌جهت اهمیت دارد که می‌توان از آن برای یافتن حملات موفق بعدی استفاده کرد. همچنین نتایج بدست آمده در این دو پژوهش بیان داشت که حملات موفق از یکدیگر فاصله‌ی کمی دارند. این یافته اساس ایده‌ی ارائه شده در پژوهش ما مبنی بر خوش‌بندی حملات است. بدین‌جهت، در این پژوهش تأثیر خوشه‌بندی حملات را بر عملکرد تست جعبه سیاه را به‌طور مفصل بررسی می‌نماییم.

[1] Zhang

۳.۴.۲ روش‌های مبتنی بر هوش مصنوعی

در سال‌های اخیر با توجه به قابلیت‌ها و توانایی‌های الگوریتم‌های هوش‌مصنوعی تلاش‌های گسترده‌ای به‌منظور ارائه‌ی راهکارهای بهینه‌تر از روش‌های سنتی تست خودکار آسیب‌پذیری، با بکارگیری الگوریتم‌های هوش‌مصنوعی شده است. به‌عنوان مثال، توم و همکاران در [۵۱] تابع برازندگی‌ای جهت اندازه‌گیری اینکه چقدر احتمال دارد که یک توکن تزریق SQL باعث موفقیت حمله شود، ارائه دادند. در [۴] نیز با استفاده از الگوریتم ژنتیک راهکاری جهت کشف ورودی‌های آسیب‌پذیر صفحات وب پیشنهاد شده است. همچنین راهکاری مبتنی بر الگوریتم ژنتیک برای تولید حملات XSS در [۱۷] ارائه شده است. علاوه بر این روش‌ها، دو دسته‌ی مهم از روش‌های مبتنی بر هوش‌مصنوعی تحت عناوین روش‌های تخاصمی[1] و روش‌های مبتنی بر یادگیری[2] در مقالات ارائه شده است، که در ادامه به بررسی آن‌ها می‌پردازیم.

- **روش‌های تخاصمی:** روش‌های تخاصمی تکنیک‌هایی هستند که از آن‌ها برای دور زدن و گمراه کردن فایروال‌های مبتنی بر یادگیری ماشین استفاده می‌شود. در این روش، ابزار تست سعی می‌کند تا با استفاده از یادگیری ماشین ورودی‌ها را طوری تغییر دهد که الگوریتم یادگیری ماشین استفاده شده در فایروال گمراه شود و نتواند آن را به‌درستی دسته بندی کند. با توجه به اینکه هدف این پژوهش تست فایروال‌های مبتنی بر قواعد هست و نه مبتنی بر یادگیری ماشین، در ادامه فقط جهت آشنایی معرفی مختصری از دو نمونه از روش‌های تخاصمی ارائه می‌دهیم.

  الدرمن[3] و همکاران در [۱۸] یک بازی امنیتی شبیه‌سازی کردند که در آن دو عامل مهاجم و مدافع با یکدیگر به رقابت می‌پردازند. در این بازی، هدف مهاجم نفوذ به مدافع، و هدف مدافع اصلاح و تقویت خود است. در این بازی که با استفاده از الگوریتم‌های یادگیری تقویتی پیاده‌سازی شده است، هم مدافع و هم مهاجم پس از گذشت زمان تقویت می‌شوند.

  در سال ۲۰۲۰ در [۱۵] راهکاری تخاصمی برای کشف آسیب‌پذیری‌های SQLi ارائه شد. در این روش ابتدا تعدادی از حملات ناموفق انتخاب می‌شوند. سپس با تعدادی عملگر جهش سعی می‌شود تا آن حملات طوری تغییر داده شوند که آن حمله بدون تغییر معنایش بتواند از فایروال عبور کند. به عبارتی دیگر، در این روش فقط ظاهر حملات به‌گونه‌ای تغییر داده می‌شود که الگوریتم یادگیری ماشین استفاده شده در فایروال دیگر قادر به شناسایی درست آن نباشد.

- **روش‌های مبتنی بر یادگیری:** روش‌های مبتنی بر یادگیری روش‌هایی هوشمند، جدید و کارآمد هستند. در این روش‌ها از الگوریتم‌های یادگیری ماشین جهت شناسایی و

[1] Adversarial
[2] Learning-based
[3] Elderman

یادگیری الگوهای مؤثر استفاده می‌شود. اگرچه این روش‌ها از عملکرد بسیار بهتری نسبت به روش‌های سنتی برخوردار هستند، اما همچنان نیاز به توسعه و بهبود دارند. در ادامه به بررسی چند نمونه از این روش‌ها می‌پردازیم.

تریپ[1] و همکاران در [۵۲] روشی را با نام XSS Analyzer برای کشف آسیب‌پذیری‌های XSS ارائه دادند. در این روش، نویسندگان سعی کردند تا مشکل انتخاب بهینه نمونه حملات موفق در میان انبوه نمونه حملات موجود را حل کنند. بدین منظور، XSS Analyzer حملات را با کمک یک دستور زبان ایجاد می‌کند و سپس تلاش می‌کند تا یاد بگیرد که چه توکن‌هایی (همان کلمات تشکیل دهنده‌ی یک حمله) باعث موفقیت یا عدم موفقیت حملات می‌شود.

دوسال پس از ارائه‌ی روش XSS Analyzer یعنی در سال ۲۰۱۵، اپلت[2] و همکاران روشی را با الهام از XSS Analyzer برای کشف آسیب‌پذیری‌های SQLi ارائه دادند [۲]. آن‌ها اسم روش خود را ML-Driven گذاشتن. بر خلاف XSS Analyzer که سعی می‌کرد توکن‌های مؤثر و غیر مؤثر را بیابد، ML-Driven سعی می‌کند تا ترکیب‌های مؤثر و غیر مؤثر از این توکن‌ها را بیابد. این مزیت باعث می‌شود که کارایی روش ML-Driven بیشتر از XSS Analyzer شود. علاوه بر این، ML-Driven تعداد درخواست کمتری را برای سرور ارسال می‌کند، درنتیجه این روش از XSS Analyzer بهینه‌تر نیز هست. سه سال پس از ارائه‌ی ML-Driven، اپلت و همکاران روش خود را توسعه دادند و توانستند با استفاده از الگوریتم تکاملی کارایی روش خود را به طرز چشمگیری افزایش دهند. آن‌ها این روش را ML-Driven E نام‌گذاری کردند [۳].

اگرچه روش‌های ML-Driven قادر هستند که تعداد زیادی از آسیب‌پذیری‌ها را کشف کنند، اما همچنان نیاز دارند تا مشاهدات زیادی را انجام دهند تا الگوهای مؤثر را بیابند. این روش‌ها برای اینکه بتوانند از الگوریتم یادگیری ماشین خود استفاده کند نیاز دارند تا مجموعه داده‌هایی متشکل از حملات برچسب‌گذاری شده‌ی موفق و ناموفق در اختیار داشته باشند. اما در تست جعبه‌ی سیاه این داده‌های برچسب‌دار در دسترس نمی‌باشند. بدین‌جهت، در تمامی روش‌های ML-Driven در ابتدا یک جستجوی تصادفی انجام می‌شود تا مجموعه داده‌های اولیه را تشکیل دهد. انجام جستجوی تصادفی زمانی که فایروال هدف به خوبی تنظیم شده باشد، باعث ارسال درخواست‌های زیاد HTTP به سمت سرور می‌شود. درمقابل، در روش ما، الگوریتم تطبیقی ارائه شده می‌تواند بدون دسترسی به داده‌های برچسب دار مسیر خود را به سوی حملات موفق پیدا کند.

مشکل دیگر روش‌های ML-Driven، روش غیر بهینه‌ی استخراج ویژگی آن‌ها است. در این روش‌ها برای استخراج ویژگی ابتدا درخت تجزیه‌ی هر رشته حمله تشکیل می‌شود.

[1] Tripp
[2] Appelt

سپس تمامی زیر درخت‌های آن‌ها استخراج می‌شوند. این زیر درخت‌ها همان ویژگی‌ها هستند. با این که حملات SQLi رشته‌هایی کوتاه هستند اما این روش استخراج ویژگی باعث می‌شود تا هزاران ویژگی از مجموعه‌های این حملات استخراج شود. افزون بر این برخی حملات مانند حمله‌ی XSS بسیار طولانی‌تر از SQLi هستند، که همین باعث می‌شود تعداد ویژگی‌ها در این روش به‌طرز چشمگیری افزایش یابد. تعداد این ویژگی‌ها به‌قدری زیاد است که طراحان روش‌های ML-Driven مجبور شدند که هر بار زیر مجموعه‌ی کوچکتری از این ویژگی‌ها را به‌صورت تصادفی انتخاب کنند. همین امر باعث می‌شود کارایی و بهینگی روش‌های ML-Driven کاهش پیدا کند. درمقابل، الگوریتم استخراج ویژگی روش ما قادر است که الگوهای پیچیده را با تعداد ویژگی‌های بسیار کمتر از روش‌های ML-Driven مدل کند. همچنین روش کاهش ویژگی استفاده شده در این پژوهش با حذف ویژگی‌های ناکارآمد، باز هم تعداد ویژگی‌ها را کاهش می‌دهد که در نهایت موجب افزایش کارایی و بهینگی روش ما می‌شود.

۵.۲ جمع‌بندی

در این فصل ابتدا مدل حملات و راه‌های مقابله با آن‌ها را شرح دادیم. سپس به بررسی ضرورت تست برنامه‌های تحت وب پرداختیم و در انتها پس از بررسی انواع روش‌های کشف آسیب‌پذیری‌های برنامه‌های تحت وب، مروری بر کارهای پیشین در زمینه‌ی تست جعبه سیاه ارائه دادیم. دیدیم که اگرچه کارهای گسترده‌ای در این زمینه انجام شده است، اما همچنین نیاز است تا روش‌های کارآمدتری برای تست جعبه سیاه ارائه شود.

فصل ۳

پیش‌نیازها

در این فصل تمامی مفاهیم استفاده شده در طراحی روش پیشنهادی این پایان نامه شرح داده می‌شود. این مفاهیم شامل روش‌های متن کاوی استفاده شده در این پایان نامه، شبکه عصبی کد کننده خودکار، روش‌های خوشه‌بندی مورد استفاده و الگوریتم یادگیری تقویتی استفاده شده در روش ما است. در آخر، جمع‌بندی از مطالب بیان شده در این فصل ارائه می‌گردد.

۱.۳ متن کاوی

متن کاوی[1] که به آن آنالیز متن نیز گفته می‌شود، یک تکنولوژی مبتنی هوش‌مصنوعی است که متن را به داده‌های نرمال شده و ساختار یافته تبدیل می‌کند تا مناسب آنالیز و استفاده در الگوریتم‌های یادگیری ماشین باشند. درنتیجه می‌توان الگوهای معنی‌دار را از داده‌های متنی استخراج کرد. در متن کاوی از روش‌های گوناگونی استفاده می‌شود، که در این بخش به معرفی آن‌هایی که در این پژوهش استفاده شده اند، پرداخته می‌شود.

[1] Text mining

۱.۱.۳ تعبیه سازی کلمات

زمانی که صحبت از مبحث پردازش زبان‌های طبیعی[۱] باشد، دو شاخه اصلی علوم‌کامپیوتر و هوش‌مصنوعی مطرح می‌شوند. در واقع NLP زیرشاخه‌ای از این دو رشته محسوب می‌شود. بنابراین برای فهم دقیق، نیاز به دانستن هدفی است که در آن دنبال می‌شود. هدف اصلی که در NLP مطرح می‌شود، ایجاد ماشینی است که بتواند تحلیل و پردازش میزان بالایی از داده‌ها زبان‌طبیعی را انجام دهد.

این هدف، در واقع بخشی از هدفی است که در هوش‌مصنوعی دنبال می‌شود (ساخت موجود هوشمند). از طرفی برای دستیابی به هدف پیش‌روی، نیاز به نگاشت داده‌های متنی به نوع داده‌ای قابل فهم (قابل پردازش) برای ماشین (داده عددی) است. این عمل در زیرشاخه علوم کامپیوتر قرار می‌گیرد.

راهکار قدیمی تبدیل متن به اعداد، عملیات One-hot می‌باشد. روشی که در آن، برداری برای بازنمایی کلمات[۲] که در آن هر کلمه در متن با یک توالی رشته‌ای منحصربه‌فرد مشخص است، ایجاد می‌گردد. در واقع، در این روش به‌ازای هر کلمه یک بردار به‌طول کل تعداد کلمات منحصربه فرد در متن داریم که در آن تنها مقدار کلمه مورد نظر برابر با یک است. در این روش دو نکته اساسی وجود دارد:

۱. جایگاه هر کلمه در بردار one-hot شده مهم است.

۲. در این نوع عملکرد ارتباط بین کلمات قابل بیان نیست. شکل انتزاعی عملکرد این روش در ادامه نشان داده‌شده‌است.

فرض کنید بخواهیم جمله the cat sat on the mat را با استفاده از روش one-hot به یک بردار عددی تبدیل کنیم. بدین ترتیب جدول ۱.۳ بیانگر مفهوم نامبرده است.

با توجه به آنچه که به‌صورت انتزاعی مشخص نمودیم، برداری که به روش one-hot در نظر گرفته می‌شود، طبق آنچه که بیان شد، به اندازه تعداد کل کلمات منحصربه‌فرد موجود در متن است. با توجه به آنچه درباره این روش بیان شد، کاملاً مشخص است که از لحاظ پردازشی، این روش دارای پیچیدگی محاسباتی فوق‌العاده بالایی است. از طرفی با توجه به نکته شماره ۲ بیان‌شده در فوق، می‌توان اذعان داشت که این روش نه تنها قابلیت انجام فرآیندهایی همچون تعیین ارتباط لغوی[۳] بین کلمات درون متن داده‌ای را نخواهد داشت، برای پردازش‌های پیچیده‌ای همچون نتیجه‌گیری محتوایی متن[۴] نیز بسیار ناکارامد می‌باشد. بنابراین نیاز به روشی است که بتواند ضعف‌هایی از این قبیل را بپوشاند. بدین‌منظور از روش تعبیه‌سازی کلمات[۵] برای توسعه استفاده می‌کنیم.

[1] Natural Language Processing(NLP)
[2] Word Representation
[3] Lexical Relation
[4] Text Summrization
[5] Word Embedding

جدول ۱.۳: مثالی از تبدیل کلمات به بردار one-hot.

کلمات	بردار one-hot
the	[۱۰۰۰۰]
cat	[۰۱۰۰۰]
sat	[۰۰۱۰۰]
on	[۰۰۰۱۰]
mat	[۰۰۰۰۱]

تعبیه‌سازی کلمات، بردار مقادیر عددی حقیقی از بازنمایی معنایی[۱] و نحوی[۲] کلماتی است که از پیکره یک مجموعه‌داده بزرگ متنی بدون‌برچسب ایجاد شده‌اند. در واقع این روش، ابزاری قدرتمند است که به‌صورت گسترده در فرآیندهای نوین پردازش زبان‌های طبیعی، از جمله تحلیل معنایی[۳] [۵۷]، بازیابی اطلاعات[۴] [۴۴]، تجزیه وابستگی[۵] [۱۳، ۳۹، ۴۵]، پاسخ‌دهی به سوال[۶] [۲۳، ۶۱] و ترجمه ماشینی[۷] [۱۲، ۵۸، ۶۱] بکار گرفته می‌شود. از لحاظ مفهومی، تعبیه‌سازی کلمات شامل یک جایگذاری ریاضیاتی با تغییر در رویکرد عملیاتی نگاشت کلمات به اعداد است. به این طریق که از یک فضای یک‌بعد بر کلمه، به فضای برداری پیوسته با ابعاد بسیار کمتر، عملیات با کاهش پیچیدگی محاسباتی همراه می‌شود.

روش‌های متنوعی از تعبیه‌سازی کلمات وجود دارند که هریک از آن‌ها دارای ویژگی‌هایی هستند که با هدف بهبود کارایی دیگری ایجاد گردیده‌اند. از جمله معروف‌ترین روش‌هایی که از تعبیه‌سازی کلمات می‌توان به آن‌ها اشاره نمود، Word2Vec [۳۵]، GloVe [۴۱]، FastText [۷، ۲۷]، NNLM [۶] و Dictionary Model [۵۳] هستند. روش Word2Vec را در بخش بعدی توضیح خواهیم داد. اما مسئله مهم دیگری که مطرح می‌شود، کارایی و ویژگی‌های مطلوب مدل‌های تعبیه‌سازی[۸]، و همچنین ویژگی‌های ارزیاب‌های[۹] موجود است، که در ادامه به بررسی آن‌ها خواهیم پرداخت.

[1] Semantic Representation
[2] Synthetic Representation
[3] Semantic Analysis
[4] Information Retrieval
[5] Dependency Parsing
[6] Question Answering
[7] Machine Translation
[8] Embedding Models
[9] Evaluators

ویژگی‌های مطلوب برای مدل‌های تعبیه‌سازی

روش‌های مختلف تعبیه‌سازی، هریک دارای خروجی متفاوت در نمایش برداری خود هستند. مسئله بسیار مهم و قابل بررسی که باید در نظر گرفت، این است که تعداد کمی از ویژگی‌هایی که وجود دارند، در تمام روش‌ها از لحاظ هدف موجود در بازنمایی، مناسب هستند. برای اینکه نمایش برداری انجام گرفته در فرآیند تعبیه‌سازی مناسب با هدف مورد نظر باشد، شناسایی و استفاده از ویژگی‌های قابل قبول در مدل‌های مختلف تعبیه‌سازی بسیار اهمیت دارد. از جمله ویژگی‌هایی که می‌توان به آن‌ها اشاره نمود، تلفیق‌ناپذیری[۱][۵۵]، که اشاره به ساختار و جزئیات یک کلمه از لحاظ جمع یا مفرد بودن، زمان و همچنین ارتباط با زیرفضایی از کلمات در مدل مربوطه دارد. دیگر ویژگی که می‌توان از آن نام برد، ثبات در مقابل ابهام لغوی[۲][۵۵]، است که دربرگرفتن معنی صحیح کلمه با توجه به بافت متنی را بیان می‌کند. بدین معنی که مدل باید قادر به بازنمایی تمامی حالات و یا معانی ممکن کلمه در بافت متنی باشد. از دیگر ویژگی‌های مطلوب مدل‌ها می‌توان به، قابلیت اطمینان[۳][۲۴]، فضای برداری خوب[۴][۲۱] و نمایش چندوجهی[۵][۵۵] را بیان نمود که هریک در مقالات مرجع خود توضیحات کامل داده شده‌اند.

ارزیاب‌ها

علاوه‌بر ویژگی‌های مطلوب که پیش‌تر برای یک مدل تعبیه‌سازی برداری بیان شد، ارزیاب‌ها و معیارهای ارزیابی نیز باید بتوانند با وضوح کافی، کلیت عملکردی یک مدل را به خوبی بیان کنند. هدف اصلی در استفاده از ارزیاب‌ها، مقایسه مشخصه‌های (کاراکترهای) مدل‌های تعبیه‌سازی برای متریک‌های کمی[۶] و نمایشی است. به‌هرحال اینکه بخواهیم یک راه‌حل واحد متداول و منسجم برای ارزیابی کاراکترهای موجود پیدا کنیم، ساده نیست. معمولاً یک ارزیاب خوب برای مدل‌های تعبیه‌سازی باید دارای ویژگی‌های زیر باشد:

۱. **داده‌های آزمون خوب**[۷]: داده‌های مناسب جهت آزمودن یک مدل تعبیه‌سازی باعث افزایش قابلیت اطمینان مدل می‌شود.

۲. **جامعیت**[۸]: از ویژگی‌های ایده‌آل یک ارزیاب، این است که بر روی ویژگی‌های متنوع و زیادی از مدل تعبیه‌سازی مورد آزمون قرار گرفته شود. بدین معنی که حالات مختلف را

[1] Non-Conflation
[2] Robustness Against Lexical Ambiguity
[3] Reliability
[4] Good Geometry
[5] Demonstration of Multifaceted
[6] Quantitative Metrics
[7] Good Test Data
[8] Comprehensiveness

بررسی کند. بدین ترتیب امتیازی که در آزمودن مدل بدست می‌آید، واقعی‌تر و معتبرتر خواهد بود.

۳. **همبستگی بالا**[۱]**:** از امتیازات یک مدل تعبیه‌سازی خوب، این است که ذاتاً یک ارزیاب قابل قبول باشد. به‌طوری که همبستگی میان کارایی مدل و پردازش زبان طبیعی در taskهای موجود مشهود باشد.

۴. **بهینگی**[۲]**:** یعنی مدل‌های تعبیه‌سازی باید از لحاظ پیچیدگی محاسباتی بهینه باشند. مدل‌هایی که ایجاد می‌شوند، درصورتی‌که از لحاظ محاسباتی بهینگی لازم را نداشته باشند، از دیدگاه عملیاتی و کاربردی قابل ارائه نیستند.

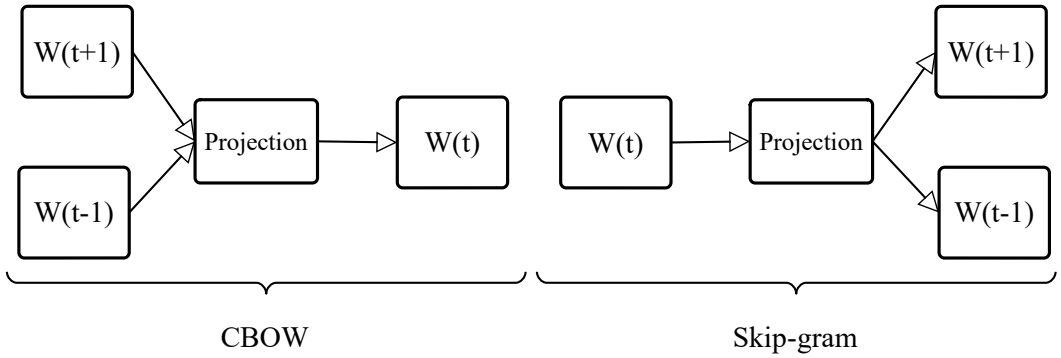
۵. **اهمیت آماری:** یکی از مشخصه‌هایی که برای مدل‌های تعبیه‌سازی در نظر گرفته می‌شود، بررسی کارایی مدل از نظر پارامترهای آماری همچون واریانس مناسب برای مدل است. بدین معنی که پارامترهای آماری مورد استفاده در مدل، باید بتوانند کارایی مدل را به خوبی و درستی نشان بدهند.

در ادامه و در بخش بعد به توضیح مدل Word2Vec و دو مورد از مدل‌های موجود خواهیم پرداخت.

۲.۱.۳ مدل Word2vec

مدل Word2Vec در سال ۲۰۱۳ توسط میکولوف و تیم تحقیقاتی وی در کمپانی گوگل ایجاد گردید[۳۵]. هدف از ارائه مدل توسط این پژوهشگران، محاسبه بردارهای مقادیر پیوسته، برای بازنمایی کلمات مجموعه‌های داده‌ای با اندازه و حجم بالا بود. همچنین نویسنده و تیم تحقیقاتی سعی داشتند مدلی ارائه دهند که که نسبت به مدل‌های مختلف دیگر دارای یادگیری سریع‌تری باشند. مبنای عملکرد مدل ارائه‌شده استفاده از شبکه‌های عصبی[۳] بود که دارای ساختار ساده‌ای می‌باشد. این افراد دو نوع متفاوت از مدل Word2Vec(کیف کلمات پیوسته[۴] و skip-gram) ارائه دادند. این مدل‌ها از نوع پیشگو[۵] بودند. همچنین از لحاظ یادگیری برداری، در مدل‌های نامبرده این قابلیت ایجاد شده‌بود که با آموزش بهتر، نتایج دقیق‌تری به‌عنوان خروجی ارائه دهند. در ادامه ابتدا به تشریح معماری کلی مدل Word2Vecخواهیم پرداخت و سپس هریک از مدل‌های نامبرده را تشریح خواهیم نمود.

[1] High Correlation
[2] Efficiency
[3] Neural Network
[4] Continuous Bag Of Word (CBOW)
[5] Predictive



شکل ۱.۳: معماری کلی دو مدل CBOW و skip-gram

معماری کلی مدل Word2vec

معماری پایه‌ای که برای مدل Word2Vec مطرح شد، همانطور که بیان گردید بر مبنای شبکه‌های‌عصبی هستند. کلیت معماری مدل Word2Vec یک شبکه عصبی دولایه است که پردازش متن را با استفاده از بردارسازی کلمات انجام می‌دهد. ورودی مدل یک پیکره متنی، و خروجی آن نیز مجموعه‌ای از بردارها می‌باشند. بردارهای مشخصه[۱]، بازنمایی کلمات در پیکره متن را نشان می‌دهند. درحالی‌که کلیت ساختاری مدل Word2Vec شبکه عصبی عمیق نیست، اما با توجه به تبدیل متن به داده‌های عددی، می‌توان آن را همچون یک شبکه عصبی عمیق تصور نمود. شکل ۳-۱ کلیت معماری دو مدل موجود Word2Vec را نشان می‌دهد. از جهتی معماری مدل Word2Vec شبیه به یک کدکننده خودکار[۲] است. از آنجایی که عملیات کدگذاری[۳]، هر کلمه را به یک بردار نگاشت می‌کند، اما آموزش لغات ورودی در مقابل بازسازی آن‌ها، نسبتا شبیه به یک ماشین محدودشده بولتزمن[۴] است. بدین‌معنی که Word2Vec در آموزش یک لغت، سایر لغات موجود در همسایگی آن را در پیکره متن در نظر گرفته و بدین طریق عملیات یادگیری انجام می‌شود. این فرآیند را به دو طریق کاملاً معکوس می‌توان درنظر گرفت که دقیقاً تعریف کننده هریک از مدل‌های موجود در Word2Vec می‌شوند. مدل اولی از یک شبکه‌عصبی‌پیشخور[۵] و با استفاده از ورودی پیکره متنی در جهت تخمین کلمه عمل می‌کند (CBOW)، مدل دوم با استفاده از کلمه، در جهت تخمین متن عمل می‌کند(skip-gram). شکل ۱.۳ کلیت معماری دو مدل موجود Word2Vec را نشان می‌دهد.

برای تمام مدل‌های بیان‌شده پیچیدگی زمانی برابر با مقدار بیان شده در فرمول ۱.۳

[1] Feature Vectors
[2] Autoencoder
[3] Encoding
[4] Boltzman Restricted Machine
[5] Feedforward Neural Network

است[۳۵].

$$O = E \times T \times Q \tag{۱.۳}$$

در رابطه فوق E تعداد تکرارهای آموزش، T تعداد کلمات و Q برای پیچیدگی هریک از مدل‌ها تعریف شده است.

حال با توجه به رابطه بیان شده (۱.۳)، اگر بخواهیم پیچیدگی زمانی هریک از مدل‌های CBOW و Skip-gram را بدست آوریم، باید ابتدا برای هریک از مدل‌های شبکه عصبی زبانی پیشخور[۱]، و شبکه عصبی زبانی بازگشتی[۲] عملیات تعیین پیچیدگی زمانی را محاسبه کنیم.

در شبکه FNNLM، بردار V که نشان‌دهنده تعداد لغات درون متن است، به‌صورت یک دربرابر جمع $V - ۱$ تعیین می‌شود و دارای بعد NxD می‌باشد. از طرفی در لایه‌افکنش[۳] دارای بعد N×D×H می‌باشیم، و نهایتاً در لایه‌خروجی، بعد H×V داریم. لذا پیچیدگی زمانی معماری کلی مدل Word2Vec برابر با رابطه (۲.۳) خواهد شد.

$$Q = N \times D + N \times D \times H + H \times V \tag{۲.۳}$$

با قرار دادن رابطه (۲.۳) در رابطه (۱.۳) پیچیدگی زمانی مدل نهایی ما بدست خواهد آمد.

حال اگر بخواهیم برای شبکه RNNLM پیچیدگی محاسباتی را بدست آوریم، باید پیچیدگی معماری Q برابر با رابطه بیان‌شده در (۳.۳) قرار داده شود.

$$Q = H \times H + H \times V \tag{۳.۳}$$

با توجه به آنچه بیان شد، معماری کلی مدل Word2Vec کاملاً مشخص گردید. در ادامه به بیان دو مورد از مدل‌های مهم در مدل Word2Vec خواهیم پرداخت.

مدل کیف کلمات پیوسته(CBOW)

در این مدل یک متن (برای مثال یک جمله) به‌عنوان ورودی مدل گرفته می‌شود. سپس با استفاده از تصویرکردن[۴] پنجره‌ای که بر روی ورودی انجام می‌شود، شناسایی ارتباط بین کلمات موجود در متن انجام شده، و نهایتاً به پیش‌بینی لغت گم‌شده (لغتی که باید از روی متن تخمین زده شود) در متن ختم خواهد می‌گردد. باید توجه داشت در این روش، ترتیب ظاهرشدن کلمات تأثیری در عملکرد مدل ندارد. این خود یکی از اهدافی است که این مدل

[1] Feedforward Neural Network Language Model(FNNLM)
[2] Recurrent Neural Network Language Model(RNNLM)
[3] Projection Layer
[4] Projection

را برای رفع مشکل موجود در مدل‌های با رویکرد one-hot بوجود آورده است. در واقع همین عدم تأثیر ظاهرشدن کلمات در این مدل است که باعث شده به این روش کیف کلمات اطلاق گردد.

مدل Skip-gram

مدل Skip-gram دقیقاً عملکردی برعکس با مدل CBOW دارد. در این مدل بر خلاف مدل قبلی که ورودی متن و پیش‌بینی کلمه را انجام می‌داد، لایه‌ورودی یک لغت دریافت می‌کند و در خروجی یک جمله پیش‌بینی می‌شود. سوالی که ممکن است در اینجا مطرح گردد، این است که مدل Skip-gram چگونه می‌تواند با دریافت یک کلمه، اقدام به پیش‌بینی یک جمله داشته باشد؟ در ابتدا نیاز است که ما یک مجموعه از کلمات داشته باشیم که بتوانیم با استفاده آن‌ها عملیات آموزش سند مورد نظر را انجام دهیم. بدین‌منظور ما از یک فرآیند one-hot برای پیشبرد کار خود استفاده خواهیم نمود. بدین‌منظور بردار one-hot را در ماتریسی که جملات ممکن را نشان خواهند داد ضرب خواهیم نمود، و نهایتاً این بردار، با عملیات تطبیق جمله صحیح را بدست خواهد آورد. شکل ۲.۳ مرحله آخر عملیات مدل، که ضرب بردار one-hot در ماتریس جملات ممکن است را نشان می‌دهد.

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

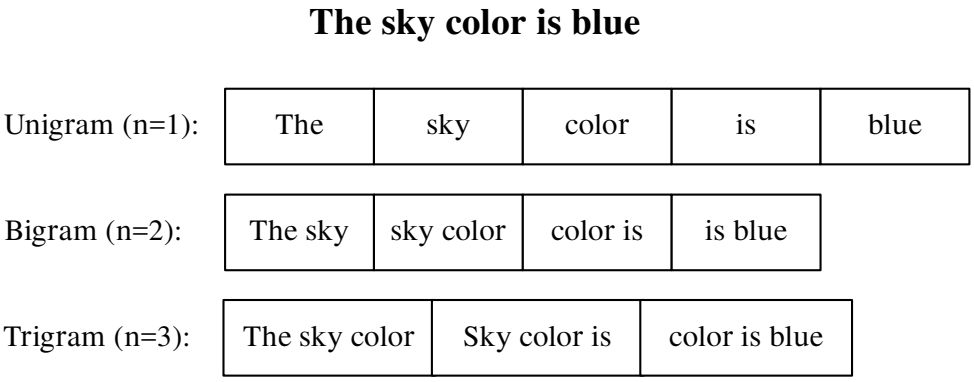
شکل ۲.۳: ضرب بردار one-hot در ماتریس جملات ممکن

۳.۱.۳ مدل n-gram

مدل n-gram یک روش بسیار مهم در مدل‌های زبانی است. این مدل در بسیاری از عملیات‌های موجود در پردازش زبان‌های طبیعی مورد استفاده قرار می‌گیرد. همچنین از آن به‌عنوان مدل پایه‌ای سایر مدل‌ها همچون Word2Vec، GloVe، PPMI و SVD استفاده می‌شود. به‌علاوه اینکه به‌جای استفاده از روش‌های سنتی در آموزش جفت‌نمونه‌ها[1] و یا جزءکلمه‌های[2] اطلاعات سطحی، همچون مدل FastText، مدل Ngram2vec بررسی سطح word-word را به‌صورت هم‌رخدادی[3] انجام می‌دهد.

[1] Sample Pairs
[2] Sub-Words
[3] Co-occurrence

مدل n-gram دنباله‌ای به اندازه n حرف یا کلمه از یک متن است. به‌طور دقیق‌تر، در روش n-gram که برای استخراج ویژگی از متن استفاده می‌شود، تمامی حالت‌های nتایی حروف یا کلمات پشت سر هم از متن استخراج می‌شوند. در شکل ۳.۳ نحوه عملکرد n-gram برای nهای مساوی با ۱، ۲ و۳ نشان داده شده است.



شکل ۳.۳: روش n-gram

۴.۱.۳ فراوانی وزنی کلمات

فراوانی وزنی کلمات یک روش آماری برای اندازه‌گیری میزان ارتباط کلمات به اسناد[1] متنی است. درواقع این روش اولین بار برای جستجو و بازیابی اطلاعات ابداع و توسط موتورهای جستجو بکار گرفته شد. به‌طور کلی در این روش هرچه یک کلمه بیشتر در یک سند متنی تکرار شده باشد، درحالی که همان کلمه در سندهای کمتری آمده باشد امتیاز تعلق آن کلمه به سند مربوطه بیشتر می‌شود. همچنین برای بررسی میزان تعلق یک جمله به یک سند، کافی است تا فراوانی وزنی کلمات آن را حساب کرده و با یکدیگر جمع کنیم. برای محاسبه‌ی فراوانی وزنی یک کلمه کافی است تا تعداد تکرار آن کلمه در سند[2] را در معکوس فراوانی سند آن کلمه[3] ضرب کنیم. نحوه‌ی محاسبه‌ی فراوانی وزنی کلمات در رابطه‌ی زیر آمده است.

$$tfidf = tf \times idf \tag{۴.۳}$$

در این رابطه tf فراوانی کلمه در سند است. idf نیز معکوس فراوانی سند آن کلمه است که از رابطه‌ی (۵.۳) بدست می‌آید.

$$idf = \ln(\frac{N}{D}) \tag{۵.۳}$$

[1] Document
[2] Term frequency (TF)
[3] Inverse document frequency (IDF)

در این رابطه N تعداد اسنادی است که کلمه‌ی مورد نظر در آن دیده شده است و D تعداد کل اسناد است.

۲.۳ شبکه‌های عصبی مصنوعی

شبکه‌های‌عصبی‌مصنوعی از ساختار بیولوژیکی[۱] مغز انسان الگو گرفته‌شده‌اند و می‌توانند در هوش‌مصنوعی و یادگیری ماشین مورد استفاده قرار بگیرند. با استفاده از این شبکه‌ها، مسائل متنوعی با استفاده از کامپیوترها می‌توانند حل شوند. ساختار کلی این شبکه‌ها شامل اتصالات داخلی بین نرون‌هایی است که با ارتباط با یکدیگر مسائل را حل می‌کنند. استفاده از شبکه‌های عصبی مصنوعی در فیلدهای متنوعی از جمله پزشکی، اقتصاد، آمار و حتی هنر قابل استفاده هستند. این شبکه‌ها بخشی از شاخه یادگیری ماشین محسوب می‌شوند که مکانیزم اصلی کاری‌شان همانطور که بیان گردید، از نوع ارتباط درون‌ساختاری نرون‌های آن‌ها است. برای حل مسائل توسط این شبکه‌ها ابتدا باید آن‌ها را آموزش داد. در ادامه به بیان کلیاتی از این شبکه‌ها خواهیم پرداخت.

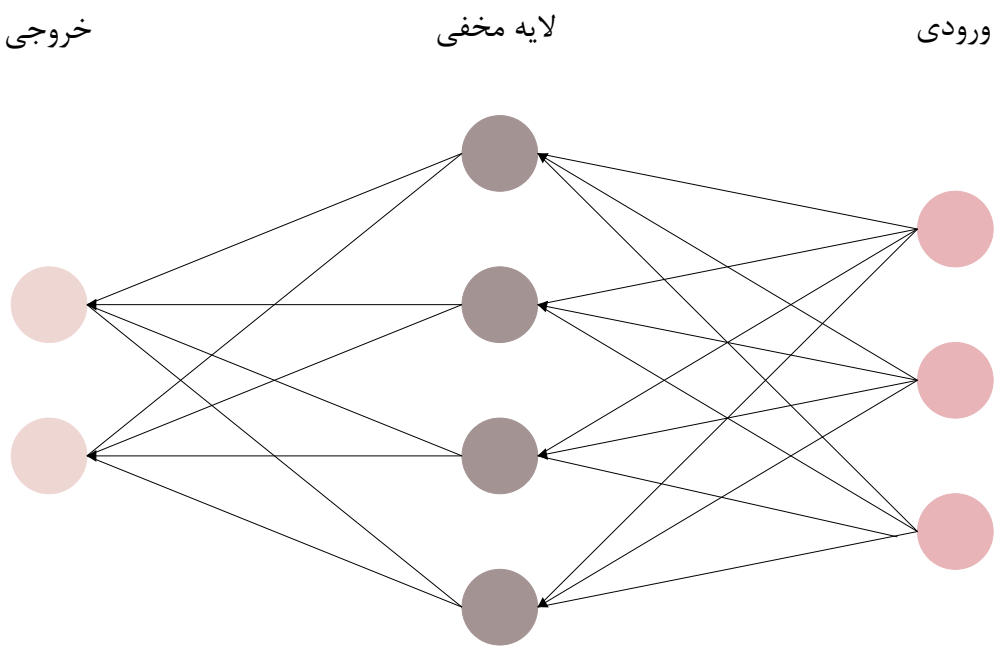
معماری کلی یک شبکه عصبی

به‌طور کلی یک شبکه‌عصبی در ساختار خود از تعدادی گره تحت عنوان نرون[۲] و یال‌های ارتباطی بین نرون‌ها که اصطلاحاً وزن‌های شبکه نامیده می‌شوند، تشکیل شده‌اند. نرون‌های موجود به‌صورت گروهی در مجموعه‌های مختلفی تحت عنوان لایه‌ها قرار می‌گیرند. به‌طور معمول هر شبکه‌عصبی‌مصنوعی می‌تواند شامل لایه‌های ورودی، میانی و خروجی باشد. شکل ۴.۳ ساختار کلی یک شبکه‌عصبی سه‌لایه را نشان می‌دهد. البته وجود لایه‌های میانی با توجه نوع شبکه و یا استفاده کاربر می‌توانند وجود نداشته باشند. از طرفی متناسب با نیاز، ممکن است بیش از یک لایه میانی داشته باشیم، که در این صورت می‌توان مفهومی پیچیده‌تر تحت عنوان شبکه‌های عصبی عمیق[۳] را مطرح ساخت. نحوه آموزش شبکه‌ها بحث دیگری است که از اهمیت بالایی برخوردار است. اینکه مفهوم آموزش یک شبکه چیست و چگونه این فرآیند انجام می‌شود را در بخش بعدی بیان خواهیم نمود

آموزش شبکه عصبی

قبل از اینکه یک شبکه عصبی برای اجرای یک کار مشخصی استفاده شود، باید فرآیند آموزش بر مبنای پارادایم یادگیری[۴] آن شبکه انجام گیرد. منظور از پارادایم، نوع یادگیری است که با

[1] Biological
[2] Neuron
[3] Deep Neural Networks
[4] Learning Paradigm



شکل ۴.۳: ساختار ساده از یک شبکه عصبی سه‌لایه

توجه به داده‌های مورد استفاده در آموزش از نوع نظارت‌شده[۱]، نظارت‌نشده[۲] و یا نیمه‌نظارتی[۳] تعیین می‌شود. از طرفی نیاز به وجود مکانیزم‌هایی برای تغییرات وزن‌های شبکه و بروزرسانی آن‌ها در جهت پیشرفت یادگیری شبکه است. این مکانیزم‌ها، قوانین موجود در نحوه یادگیری شبکه هستند. برای مثال اینکه وزن‌های شبکه بر چه معیاری باید بروزرسانی شوند، خود یکی از قوانین یادگیری شبکه هست. اما آنچه از لحاظ مفهومی به آموزش یک مدل یادگیری (در اینجا مدل یادگیری شبکه‌عصبی است) اطلاق می‌گردد، تنظیم پارامترهای آن مدل (پارامترهای مدل در اینجا وزنه‌ای شبکه هستند.) است. بدین معنی که شبکه عصبی با یک وزن‌دهی اولیه که معمولاً به‌صورت تصادفی است، شروع به فرآیند یادگیری می‌کند، وزن‌ها در طی تکرارهای مرحله آموزش با توجه به قوانین موجود بروزرسانی می‌شوند، تا زمانی‌که خطای حاصل طبق معیاری مشخص از پیش‌تعیین‌شده به آنچه که باید برسد. نهایتاً زمانی که وزن‌های شبکه به مقادیری برسند که خروجی بدست آمده شبکه و خروجی مطلوب بهم نزدیک باشند، می‌گوییم فرآیند آموزش انجام شده است. معیار عملکرد مرحله آموزش، با معیارهایی همچون دقت و یا صحت آن بیان می‌شوند، که البته با درنظر گرفتن شرایط خاص در فرآیند آموزش می‌توانند بیانگر کیفیت روال آموزش باشند. اما نکته مهمی که به آن اشاره شد، قوانین تغییر وزن‌ها

[1] Supervised
[2] Unsupervised
[3] Semisupervised

و بروزرسانی آن‌ها است. از مهم‌ترین مکانیزم‌های موجود در قوانین بروزرسانی، قانون هب[1] است. مکانیزم‌های یادگیری معمولاً با توجه به رویکرد یک شبکه تعیین و بکارگرفته می‌شوند.

شبکه‌های عصبی معمولاً دارای سه لایه می‌باشند. در صورتی که تعداد لایه‌ها بیشتر شود و سطح انتزاع بیشتری در هر لایه نسبت به لایه قبل ایجاد گردد، مفهوم پیشرفته‌تری در شبکه‌های عصبی بوجود می‌آید که به آن شبکه‌های عصبی عمیق می‌گویند. در ادامه به توضیحات بیشتری از این شبکه‌ها می‌پردازیم.

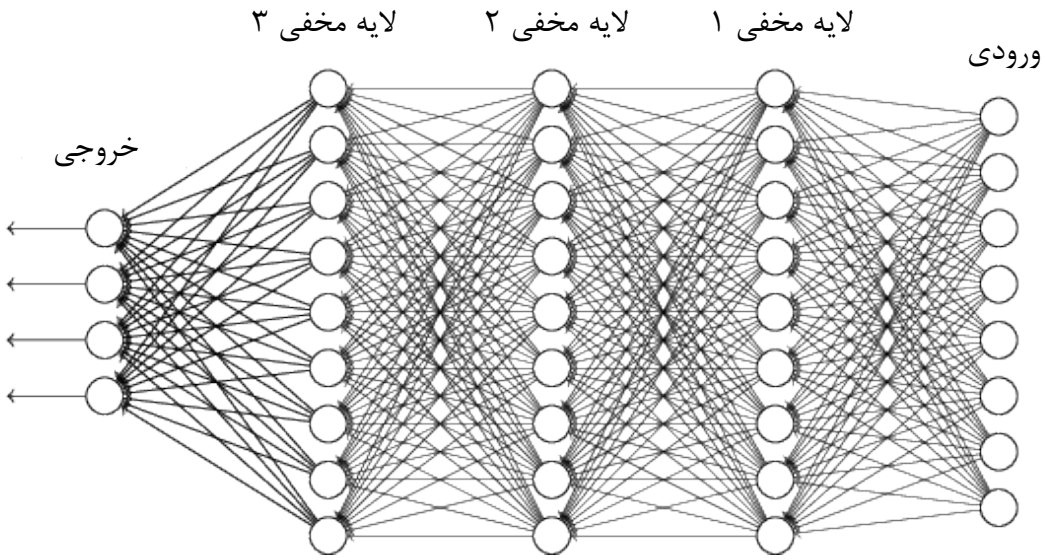
۱.۲.۳ شبکه‌های عمیق

تکنیک یادگیری عمیق که با استفاده از شبکه‌های عصبی عمیق پیاده‌سازی می‌شوند، قابلیت محاسباتی و کارایی بالاتری را در یادگیری امکان‌پذیر می‌سازند. همچنین با استفاده از شبکه‌های عمیق، قدرت و انعطاف‌پذیری بیشتری در استفاده از داده‌های با حجم بالا خواهیم داشت. در واقع یکی از ملزومات استفاده از شبکه‌های عمیق، وجود مجموعه داده با حجم بالای نمونه‌ها هستند. یکی از مهم‌ترین دلایلی که باعث افزایش بکارگیری شبکه‌های عمیق (یادگیری عمیق) نسبت به الگوریتم‌های یادگیری معمول شده است، افزایش دقت در فرآیندهایی همچون طبقه‌بندی[2] و خوشه‌بندی[3] داده‌ها نسبت به الگوریتم‌های یادگیری و شبکه‌های عصبی ساده است. چرا که در شبکه‌های عمیق با افزایش حجم نمونه داده‌ها سیستم‌های موجود آموزش بهتری نسبت به نوع معمول خواهند دید. یکی از نکات مهم دیگر در مورد یادگیری شبکه‌های عمیق که می‌توان به آن اشاره نمود، عملیات استخراج ویژگی این شبکه‌ها است. معمولاً در یادگیری عمیق استخراج ویژگی به‌صورت خودکار انجام می‌شود و باعث بهبود در عملکرد سیستم یادگیری می‌شود. در شکل ۵.۳ نمونه‌ای فرضی از یک شبکه عصبی عمیق را مشاهده می‌نمایید. همان‌طور که در بخش قبل بیان گردید، در شبکه‌های عصبی عمیق تعداد لایه‌های میانی بیشتر از یک لایه هستند. البته نکته مهم همان‌طور که در بخش قبل هم بیان شد، افزایش سطح انتزاع با افزایش تعداد لایه‌ها باید باشد.

معماری‌های اصلی در شبکه‌های عمیق

از آنجا که معماری‌های موجود در شبکه‌های عمیق با توجه به افزایش تعداد لایه‌ها و همچنین ویژگی‌های موجود پیچیده‌تر هستند، زمان آموزش بیشتری نیز نسبت به شبکه‌های عصبی ساده لازم دارند. از طرفی این زمان آموزش بیشتر را می‌توان از طرق مختلفی همچون استفاده از رویکردهایی همچون یادگیری‌انتقالی[4] و یا استفاده از سخت‌افزارهای قوی‌تر همچون واحدهای پردازشی‌گرافیکی[5] کاهش داد.

[1] Hebbian Rule
[2] Classification
[3] Clustering Rule
[4] Transfer Learning
[5] Graphical Processing Units(GPUs)



شکل ۵.۳: شماتیک کلی یک شبکه عمیق با سه لایه میانی

از لحاظ کاربردی، شبکه‌های با ساختار عمیق، به‌طور کامل قابلیت استفاده از پارادایم‌های مختلف یادگیری را دارند. با توجه به پارادایم‌های موجود در یادگیری، می‌توانیم معماری‌های زیر در شبکه‌های عمیق را نام ببریم.

شبکه‌های عمیق پیش‌آموزش‌داده‌شده با پارادایم یادگیری نظارت‌نشده

در شبکه‌های عمیق با پارادیم نظارت‌نشده (بدون ناظر)، معمولاً قبل از یادگیری، وزن‌های اولیه شبکه با استفاده از یک معماری مشخص اولیه تنظیم می‌شوند. برای مثال، وزن‌های شبکه عصبی که می‌خواهیم از آن استفاده کنیم، با استفاده از وزن‌های شبکه با معماری A مقداردهی می‌شوند. سپس با توجه به پارادایم موجود عمل آموزش شبکه انجام می‌شود و برای تخمین از آن استفاده می‌شود. منظور از پیش‌آموزش‌داده‌شده، دقیقاً همین مسئله است. از جمله شبکه‌های موجود در این دسته می‌توان به موارد زیر اشاره نمود:

۱. شبکه عصبی کدکننده خودکار[۱]
۲. شبکه باور عمیق[۲]
۳. شبکه مولد رقابتی[۳]

با توجه به فعالیتی که در این پایان‌نامه قصد انجام آن را داریم، شبکه عمیق کدکننده خودکار را در بخش بعد توضیح خواهیم داد.

[1] Autoencoder Neural Network
[2] Belief Deep Neural Network
[3] Generative Adversarial Network(GAN)

شبکه‌های عصبی کانولوشنی

این شبکه‌ها بیشتر برای تصاویر بکار گرفته می‌شوند. معمولاً در این شبکه‌ها ورودی تصویر داده می‌شود و با توجه به لایه‌های موجود در آن‌ها که هریک وظیفه خاصی را انجام می‌دهند، به فرآیندهایی همچون تشخیص، شناسایی و طبقه‌بندی اشیا در تصاویر می‌پردازیم. از جمله مهم‌ترین معماری‌هایی که برای این شبکه‌ها وجود دارند می‌توان به موارد زیر اشاره نمود:

۱. LeNet

۲. AlexNet

۳. ResNet

۴. VGGNet

۵. GoogleNet

از جمله مهم‌ترین فعالیت‌هایی که با استفاده از معماری‌های فوق انجام می‌شوند، می‌توان به شناسایی اشیا[۱] و قطعه‌بندی معنایی[۲] اشاره نمود.

شبکه‌های عصبی بازگشتی

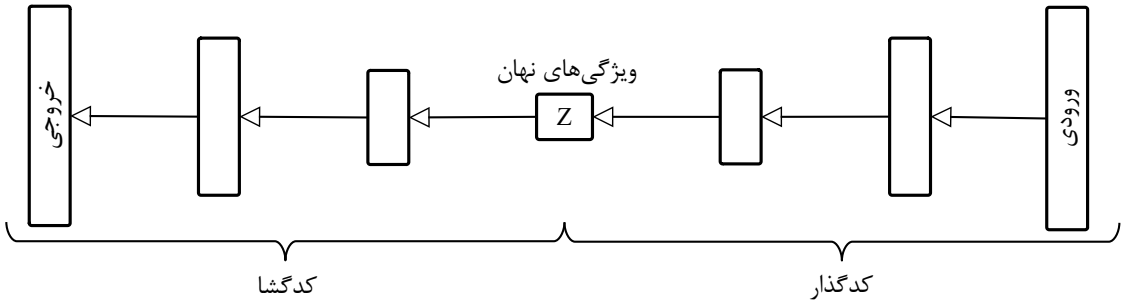
در این شبکه‌ها خروجی هریک از حالات قبل به ورودی حالات جاری فرستاده می‌شوند. در شبکه‌های عصبی بازگشتی، لایه‌های میانی می‌توانند اطلاعات را در خود نگه دارند. به عبارت دیگر، نرون‌های لایه‌های میانی دارای حافظه‌اند. حالات میانی با توجه به خروجی تولیدشده در حالات قبل خود بروزرسانی می‌شوند. این شبکه‌ها می‌توانند عمل پیش‌بینی سری‌های زمانی را انجام دهند. چرا که ورودی‌های حالات قبل را حفظ نمایند. از جمله مهم‌ترین شبکه‌های با معماری بازگشتی می‌توان به شبکه با حافظه کوتاه‌ـ بلند مدت[۳] اشاره نمود.

شبکه عصبی کدکننده خودکار

شبکه کدکننده خودکار از جمله شبکه‌های عمیق هستند که فرآیند آموزش آن‌ها همانطور که در بالا اشاره گردید از نوع نظارت‌نشده است. از جمله مهم‌ترین فعالیت‌هایی که برای این شبکه‌ها می‌توان نام برد، کاهش‌بعد داده‌ها و شناسایی‌آنومالی[۴] می‌باشد. از لحاظ کارایی و فیلد کاری، این شبکه‌ها می‌توانند زیرشاخه‌ای از شبکه‌های عمیق مولد رقابتی باشند. در این شبکه می‌توان از یک آموزش اولیه برای آموزش کلی شبکه استفاده نمود. معماری این

[1] Object Detection
[2] Semantic Segmentation
[3] Semantic Segmentation
[4] Anomaly Detection

شبکه‌ها از دو بخش اصلی تشکیل شده است. یک بخش کدگذار[۱] و بخش دیگر کدگشا[۲] شکل ۶.۳ معماری کلی این شبکه را نشان می‌دهد. در بخش اول که مربوط به لایه‌های کدگذاری می‌باشد، ورودی به یک فضای نهفته[۳] ترجمه و بازنمایی می‌شود. کلیت این عمل با رابطه ریاضیاتی (۱.۳) قابل نمایش است.



شکل ۶.۳: شبکه عصبی کدکننده خودکار

$$h = f(x) \tag{۱.۳}$$

در بخش دوم که مربوطه به لایه‌های کدگشایی است، فضای نهفته حاصل از ورودی که در بخش کدگذاری انجام گرفت، به‌عنوان ورودی دریافت شده و از روی آن ورودی شبکه مجددا به عنوان خروجی بازسازی می‌شود. رابطه (۲.۳) فرآیند کدگشایی را با توجه به رابطه (۱.۳) نشان می‌دهد.

$$r = g(h) \tag{۲.۳}$$

کلیت عملیاتی که شبکه کدکننده خودکار در دو بخش کدگذاری و کدگشایی انجام می‌دهد با استفاده از رابطه (۳.۳) قابل بیان است.

$$f(g(x)) = r \tag{۳.۳}$$

آنچه با توجه به روابط (۱.۳) تا (۳.۳) مشخص است، ابتدا ورودی x به یک تابع کدگذار f داده می‌شود. سپس خروجی r با استفاده از تابع کدگشا g از روی متغیر فضای نمونه h بازسازی می‌گردد. کلیت این دو مرحله در رابطه (۳.۳) بیان شده است.

[1] Encoder
[2] Decoder
[3] Latent Space

۳.۳ خوشه‌بندی

در فرآیند تحلیل‌داده، عملیات خوشه‌بندی، به گروه‌بندی مجموعه‌ای داده (اشیاء)اطلاق می‌گردد که در آن داده‌ها با توجه به معیارهای مشخص از پیش‌تعیین‌شده دارای شباهت‌هایی در ساختار خود هستند. این داده‌ها با توجه به میزان شباهت‌های موجود، در گروه‌بندی‌های مشخصی به نام خوشه[۱] قرار می‌گیرند. خوشه‌بندی یکی از مهم‌ترین عملیات‌ها در علم‌داده[۲] است. خوشه‌بندی از لحاظ نوع پارادایم یادگیری، اغلب از نوع نظارت‌نشده است. ازدیدگاهی دیگر، خوشه‌بندی یک روش معمول برای تجزیه و تحلیل داده‌های آماری است که در بسیاری از زمینه‌ها از جمله تشخیص الگو[۳]، تجزیه‌وتحلیل تصویر، بازیابی اطلاعات و فشرده‌سازی داده‌ها[۴] استفاده می‌شود.

اگر بخواهیم از لحاظ ماهیتی به مسئله خوشه‌بندی بنگریم، باید گفت: مفهوم خوشه را نمی‌توان به‌طور صریح بیان و تعریف نمود. یکی از مهم‌ترین دلایل این امر، تعداد بالای الگوریتم‌های موجود در این زمینه است. با این تفاسیر، پژوهشگران از الگوریتم‌های خوشه‌بندی متفاوتی استفاده می‌کنند کلیت نگرش به مفهوم یک خوشه، از آنجایی که به واسطه الگوریتم‌های موجود صورت می‌پذیرد، به‌طور ملموسی در خواص ماهیتی متفاوت هستند. درک مفهوم ساختاری در تنوع خوشه‌بندی، کلید فهم جامع در پیدا کردن تفاوت‌های بین الگوریتم‌های موجود می‌باشد. در ادامه، به بیان دو مورد از مهم‌ترین الگوریتم‌های خوشه‌بندی که در این پژوهش از آن استفاده نموده‌ایم، خواهیم پرداخت.

۱.۳.۳ روش خوشه‌بندی k-means

روش k-means علیرغم سادگی که در ساختار الگوریتمی خود دارد، روشی پرکاربرد و پایه‌ای همچون تعدادی از روش‌های دیگر، از قبیل خوشه‌بندی فازی[۵] و Segment-wise distributional clustering Algorithm است. نحوه عملکرد الگوریتمی k-means بدین‌گونه است که در شروع تعدادی دلخواه از نقاط را به‌منظور مراکز خوشه‌ها انتخاب می‌کنیم. سپس با بررسی هر نمونه، و محاسبه فاصله آن با تمام مراکز تعیین‌شده، آن را در خوشه متناظر با مرکزی قرار می‌دهیم که نزدیکترین فاصله را با آن داشته است. پس از بررسی تمام نمونه‌ها و تخصیص آن‌ها به خوشه مناسب، عملیات بروزرسانی مراکز، با محاسبه میانگین داده‌های موجود در هر خوشه انجام می‌شود. سپس خوشه‌بندی تمام نمونه‌ها را تکرار می‌نماییم. عملیات تکراری را تا زمانی که طی دو مرحله متوالی خوشه هیچ نمونه‌ای تغییر نکند، انجام می‌دهیم. از جمله مشکلات این روش عدم بهینگی قطعی آن است. چرا که وابسته به انتخاب اولیه مراکز است. از مشکلات

[1] Cluster
[2] Data Science
[3] Pattern Recognition
[4] Data Compression
[5] Fuzzy Clustering

دیگر آن می‌توان به تعیین تعداد خوشه‌ها و صفر شدن خوشه‌ها اشاره نمود.

از نظر ماهیتی، روش K-means دارای ویژگی‌های نظری جالبی دارد. اولین ویژگی اینکه فضای داده‌ای را به ساختاری تحت عنوان نمودار Voronoi تقسیم‌بندی می‌کند. دوم اینکه از لحاظ مفهومی، شبیه به الگوریتم طبقه‌بندی نزدیکترین همسایه[۱] است. به‌همین دلیل است که در یادگیری ماشین محبوب می‌باشد. سومین ویژگی اینکه می‌توان این روش را از لحاظ تنوع در خوشه‌بندی براساس تابع محاسباتی مشاهده نمود.

۲.۳.۳ روش خوشه‌بندی سلسله‌مراتبی

خوشه‌بندی سلسله‌مراتبی[۲]، گاهاً با نام خوشه‌بندی مبتنی بر اتصال[۳] نیز بیان می‌شود. در این روش خوشه‌بندی، نمونه‌ها را برای ایجاد خوشه‌ها بر مبنای فاصله آن‌ها اتصال می‌دهند. خوشه را در حالت کلی می‌توان با حداکثر فاصله مورد نیاز برای اتصال قطعات خوشه توصیف نمود. در فاصله‌های متفاوت، خوشه‌های متنوعی درست می‌شود که می‌تواند با استفاده از یک دندروگرام[۴] نشان داده شود. دندروگرام توضیح می‌دهد که نام معمول خوشه‌بندی سلسله مراتبی از آن می‌آید. این الگوریتم‌ها یک بخش‌بندی مجموعه داده را بیان نمی‌کنند، بلکه سلسله مراتب گسترده‌ای از خوشه‌هایی که در فاصله‌های معینی با یکدیگر ادغام می‌شوند، به‌عنوان خروجی نتیجه می‌دهند. در یک دندروگرام، محور y نشان‌دهنده فاصله‌ای است که خوشه‌ها ادغام می‌شوند، در حالی که اشیا در امتداد محور x قرار می‌گیرند به‌طوری‌که خوشه‌ها با هم مخلوط نمی‌شوند.

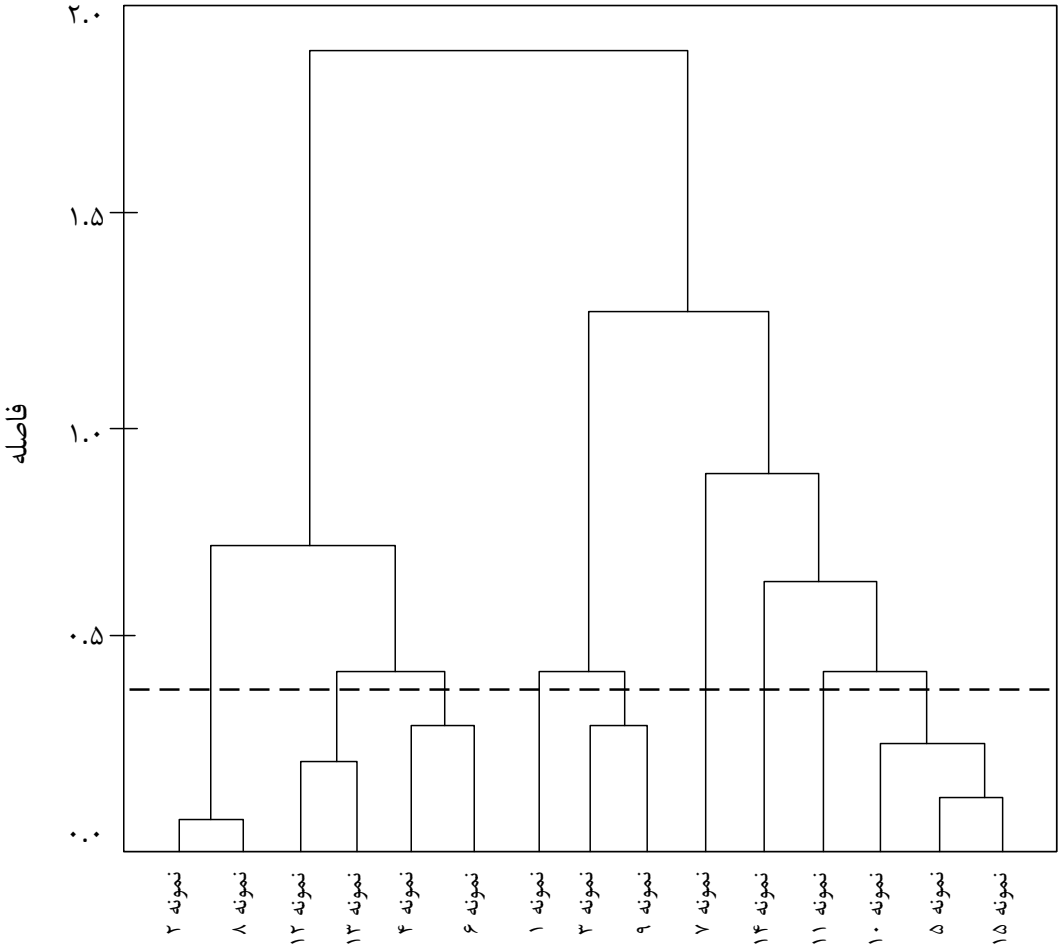
خوشه‌بندی سلسله مراتبی بر دو نوع می‌باشد:

۱. **روش تجمیعی**[۵]: این روش که به روش پایین به بالا[۶] نیز معروف می‌باشد، روشی است که در آن برای شروع کار هر داده به عنوان یک خوشه مجزا در نظر گرفته می‌شود. در ادامه با اجرای یک الگوریتم، هربار خوشه‌های دارای شباهت با یکدیگر ادغام می‌شوند (به‌همین دلیل جمع‌شونده نامیده می‌شود). این روند آنقدر ادامه می‌یابد تا چند خوشه متمایز حاصل شود. اشکال این روش حساسیت بالای داده‌ها به نویز[۷] و پیچیدگی محاسباتی بالای آن است.

۲. **روش تقسیمی**[۸]: در این روش که به روش بالا به پایین[۹] نیز شناخته می‌شود، در شروع، تمام نمونه‌ها، در یک خوشه جای می‌گیرند. سپس با اجرا شدن الگوریتمی تکرارشونده،

[1] Nearest Neighbor Classification
[2] Hierarchical Clustering
[3] Connection Based Clustering
[4] Denderogram
[5] Agglomerative
[6] Button Up
[7] Noise Sensitivity
[8] Divisive
[9] Top-Down

در هر تکرار داده‌ای که شباهت کمتری با داده‌های دیگر دارد به خوشه‌ای مجزا تقسیم می‌شود. این کار تا زمانی ادامه می‌یابد که یک یا چند خوشه یک عضوی ایجاد شود. این روش مشکل نویز را ندارد. شکل ۷.۳ نمونه‌ای از خوشه‌بندی سلسله‌مراتبی را نشان می‌دهد.



شکل ۷.۳: کلیت خوشه‌بندی سلسله‌مراتبی از لحاظ مفهومی

۴.۳ یادگیری تقویتی

یادگیری تقویتی[1] نوعی از فرآیندهای یادگیری ماشینی است که در آن عامل می‌تواند فرآیند یادگیری را از طریق تعامل با محیطی که در آن فعالیت می‌کند، و با استفاده از رویکرد سعی و خطا، از روی بازخوردهای اعمال و تجربیات خود داشته باشد. در این نوع یادگیری، زمانی‌که

[1] Reinforcement Learning

عامل در حالتی مشخص اقدام به انجام عمل مقتضی می‌نماید، با توجه به عمل درصورتی که کارایی آن مثبت باشد، پاداش بدست می‌آورد. از طرفی در صورت کارایی منفی جریمه دریافت می‌کند. هدف اصلی عامل در این نوع یادگیری، به حداکثر رساندن پاداش‌های دریافتی در طولانی مدت است. شباهت یادگیری تقویتی با یادگیری نظارت‌شده، در استفاده هردو آن‌ها از نگاشت ورودی و خروجی برای فرآیند یادگیری است. از طرفی نیز باید بیان نمود در یادگیری تقویتی فرآیند آموزش با یادگیری نظارت‌شده از لحاظ برخی موارد همچون وجود مدل پیش‌فرض بایاس‌شده که در یادگیری نظارت‌شده گاهاً باید وجود داشته باشد، متفاوت است. در یادگیری تقویتی عامل از طریق دریافت پاداش و یا تنبیه که به نوعی سیگنال‌های دریافتی هستند، برای بهینه‌سازی نتیجه عملکرد نهایی سیستم استفاده می‌کند.

تفاوت اصلی رویکردی در یادگیری تقویتی با رویکردهای دیگر در یادگیری ماشین، این است که در یادگیری تقویتی، عمل درستی که در هر حالت وجود دارد، به عامل گفته نمی‌شود، و تنها از طریق معیاری به عامل فهمانده می شود که یک عمل دارای چه اثر خوب یا بدی خواهد بود. عامل یادگیرنده باید میزان اثر خوب یا بد هر عمل عامل را به آن بفهاند. در واقع عامل یادگیرنده با در اختیار داشتن اطلاعات مربوط به اثرات هر عمل، یاد می‌گیرد که عمل بهینه در هر حالت چیست. در واقع، وجود همین ویژگی نقطه قوت اساسی در یادگیری تقویتی است. همچنین، چنین ویژگی باعث می‌شود سخت‌ترین مسائل، با دردسترس‌بودن کمترین اطلاعات از مسئله با استفاده از یادگیری تقویتی به راحتی حل شوند.

۱.۴.۳ اجزای یادگیری تقویتی

۱. سیاست[1]: نگرش بازخوردی با هر عمل انجام‌شونده عامل و نحوه اتخاذ تصمیم، در شرایط گوناگون متفاوت است. این نگراش را سیاست تعیین می‌کند. در واقع سیاست، تعیین‌کننده کلیت رفتاری عامل برای رسیدن آن به حالات بهتر است. از جمله سیاست‌های مهم در یادگیری تقویتی می‌توان به ϵ-greedy اشاره نمود که در بخش بعدی آن را توضیح می‌دهیم.

۲. تابع پاداش[2]: هدف در یک مسئله یادگیری تقویتی، به واسطه تابع یادگیرنده مشخص می‌شود. در واقع، وظیفه انجام فرموله‌سازی مسئله، با تابع یادگیرنده است. رویکرد این تابع بدین صورت تعریف می‌شود که با نزدیک‌شدن عامل به هدف مسئله، پاداش نسبت به حالت‌هایی که در آن عامل کارایی مثبت داشته است، افزایش بیابد. تابع پاداش تأثیر مستقیم در یادگیری عامل دارد. اگر تابع پاداش را به‌صورت مناسب تعریف شود، سرعت یادگیری عامل بهبود پیدا می‌کند. باید بدانیم پاداش‌ها کوتاه مدت و ارزش‌ها، بلند مدت هستند. بدین معنی‌که ممکن است یک حالت پاداش کمتری داشته باشد

[1] Policy
[2] Reward Function

و یا اصلاً پاداشی نداشته باشد (جریمه ندارد) اما همان حالت ما را به هدف نزدیک‌تر می‌کند. بنابراین، نباید در یک مسئله یادگیری تقویتی اهداف بلند مدت را فدای اهداف کوتاه مدت نمود.

۳. تابع ارزش‌گذاری[1]: این تابع بیشتر یک نگاه به چند مرحله بعد دارد. با استفاده از این تابع، برای حالات مقداری مشخص می‌شود. مقادیر تعیین‌شده توسط تابع هرچه بیشتر باشد، نشان‌دهنده نزدیکتر شدن به هدف است.

۴. مدل: مسائل موجود در یادگیری تقویتی، احتمالاتی و تصادفی هستند. حالت‌های درون فضای مسئله به‌صورت غیرقطعی می‌باشند. بدین معنی که در برابر انجام یک عمل مشخص، ممکن است عامل به بیش از یک حالت و یا حتی به همه حالات ممکن با یک مقدار احتمال نگاشت داده شود. هر عمل دارای یک احتمال است. انتقال بین حالات دیگر نیز احتمال هستند. در واقع، هدف که فرموله‌سازی مسئله یا همان مدل است، بیشینه‌سازی سودمندی[2] در بلندمدت است.

۲.۴.۳ سیاست ϵ-greedy

سیاست ϵ-greedy سیاستی در یادگیری تقویتی است که با توجه به ماهیت آن، به‌منظور متوازن کردن کشف و بهره‌برداری درهنگام انتخاب تصادفی یک عمل مشخص برای عامل انجام می‌شود. در این سیاست عامل در یک محیط مشخص با احتمال ϵ اقدام به انتخاب بهترین حالت می‌کند، و با احتمال $\epsilon - ۱$ به‌صورت حریصانه اقدام به عمل بعدی خواهد داشت. در این سیاست معمولاً مقدار احتمال ϵ کم است.

۵.۳ جمع‌بندی

روش پیشنهادی این پژوهش از اجزا مختلفی تشکیل شده است که هر کدام از آن‌ها دربرگیرنده‌ی مفاهیم متنوعی در زمینه‌ی هوش‌مصنوعی است. بنابراین، در این فصل به شرح مفاهیمی پرداختیم که در روش پیشنهادی مورد استفاده قرار گرفته‌اند و دانستن آن‌ها برای فهم روش پیشنهادی الزامی است.

[1] Value Function
[2] Utility Maximization

فصل ۴

روش پیشنهادی

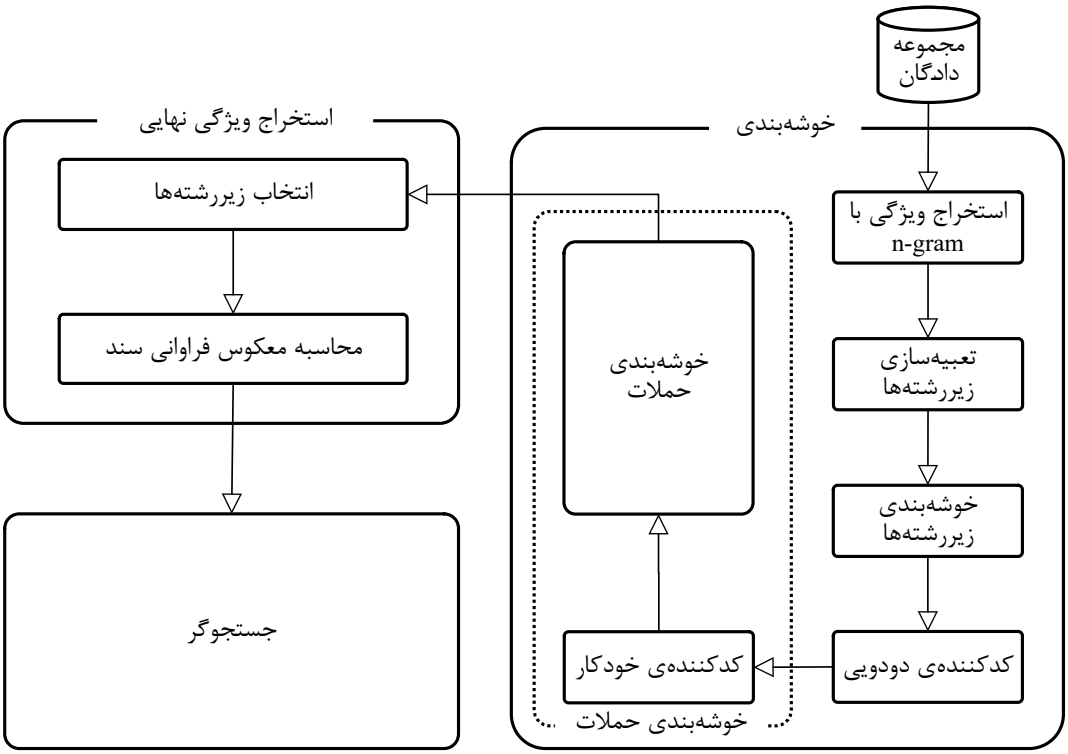
در این فصل ابتدا یک شمای کلی از روش پیشنهادی و ایده‌ی اصلی آن ارائه می‌دهیم. سپس در ادامه‌ی بخش‌ها به‌ترتیب به شرح مفصل تمامی مراحل روش ارائه شده می‌پردازیم. در انتها نیز جمع‌بندی از مطالب گفته‌شده ارائه می‌نماییم.

۱.۴ شمای کلی

حملات تزریق کد، رشته‌هایی هستند که از به هم چسباندن تعدادی زیررشته تشکیل می‌شوند. اگر این زیررشته‌ها را به‌عنوان پارامترهای تست در نظر بگیریم، می‌توان گفت که هر کدام از آن پارامترها می‌تواند باعث موفقیت یا عدم موفقیت یک حمله در عبور از فایروال شود. در تست جعبه سیاه اگرچه اطلاعاتی از سیستم در اختیار ابزار نیست که با آن زیررشته‌ها را دسته‌بندی کند، اما این امکان وجود دارد که با تحلیل و بررسی حملات موفق و ناموفق بتوان زیررشته‌های مؤثر را کشف کرد. با این حال، به‌دلیل تنوع بسیار زیاد الگوهای یک حمله، یک ابزار تست برای آن که بتواند زیررشته‌ها را دسته‌بندی کند به تعداد بسیار زیادی مشاهده نیاز دارد.

به‌طور کلی در این پژوهش ابتدا سعی می‌کنیم زیررشته‌ها را طوری از رشته‌های حملات استخراج کنیم که هم تعداد آن‌ها قابل قبول باشد و هم بتواند الگوهای پیچیده را مدل کند. بدین‌منظور از n-gram استفاده می‌کنیم که به سادگی و مقیاس‌پذیری شهرت دارد [۳۷]. پس از آن رشته‌های حملات را خوشه‌بندی می‌کنیم که این امر به خودی خود باعث کاهش ویژگی

هم می‌شود. چرا که بسیاری از زیررشته‌هایی که استخراج شده‌اند در میان حملات یک خوشه ممکن است استفاده نشده باشند. علاوه بر این، از آنجایی که انتظار می‌رود حملات موفق در خوشه‌های یکسان قرار بگیرند، با تمرکز روی خوشه‌هایی که احتمال بیشتری دارند که شامل حملات موفق باشند تعداد جستجوهای ناموفق را کاهش می‌دهیم.



شکل ۱.۴: شمای کلی از دو فاز اول روش پیشنهادی که مراحل آماده سازی داده‌ها برای فاز جستجو را نشان می‌دهد.

شکل ۱.۴ شمای کلی از روش پیشنهادی را نشان می‌دهد. در مرحله‌ی اول از فاز خوشه‌بندی، زیررشته‌های تشکیل دهنده‌ی نمونه حملات مجموعه‌داده توسط n-gram استخراج می‌شود. سپس، در مرحله‌ی تعبیه‌سازی کلمات، زیررشته‌های استخراج شده به برداری از اعداد حقیقی تبدیل می‌شوند تا در مرحله‌ی بعد بتوانیم آن‌ها را با استفاده از روش سلسله‌مراتبی خوشه‌بندی کنیم. در مرحله‌ی بعد برداری دودویی[1] با استفاده از خوشه‌های بدست آمده تشکیل می‌شود. پس از آن، شبکه‌ی کدکننده‌ی خودکار از این بردار دودویی ویژگی‌هایی را استخراج می‌کند که در مرحله‌ی آخر این فاز با استفاده از آن‌ها حملات را خوشه‌بندی می‌کنیم. در فاز بعدی، در مرحله‌ی اول زیررشته‌های بلااستفاده در خوشه از مجموعه‌ی زیررشته‌های متعلق به آن خوشه حذف می‌شوند و در مرحله‌ی بعد از آن، معکوس فراوانی سند برای زیررشته‌های باقی‌مانده محاسبه می‌شود تا بردار ویژگی نهایی شکل بگیرد. پس از شکل‌گیری بردارهای ویژگی از آن‌ها

[1] Binary

در فاز آخر یعنی جستجو استفاده می‌شود. شایان ذکر است که دو فاز اول تنها یکبار اجرا می‌شوند و پس از آن می‌توان بارها از خروجی آن‌ها در فاز جستجو استفاده کرد.

۲.۴ خوشه‌بندی

در اولین مرحله‌ی روش پیشنهادی، رشته‌های حملات به زیررشته‌های تشکیل‌دهنده‌ی خود تجزیه می‌شوند و سپس بر اساس زیررشته‌های مشترک خوشه‌بندی می‌شوند. به عبارتی دیگر، در این روش یک مجموعه‌داده‌ی بزرگ به چندین مجموعه‌داده‌ی کوچکتر شکسته می‌شود. بنابراین می‌توانیم جستجو را در مجموعه‌دادگان کوچکتر و بصورت مستقل انجام دهیم. این عمل می‌تواند به کاهش ویژگی کمک چشمگیری کند. کاهش ویژگی از آنجایی اهمیت دارد که هرچه تعداد ویژگی‌ها بیشتر باشد مشاهدات بیشتری برای یادگیری الگوهای مؤثر لازم است. بنابراین حذف ویژگی‌های نامرتبط می‌تواند باعث بهبود دقت شود و از آنجایی که کاهش ویژگی‌ها باعث کاهش پیچیدگی محاسباتی می‌شود و کار کردن با بردار ویژگی‌های کوچک به منابع کمی نیاز دارد کارایی روش نیز افزایش می‌یابد [۵۰]. در ادامه‌ی این بخش به شرح کامل مراحل خوشه‌بندی حملات می‌پردازیم.

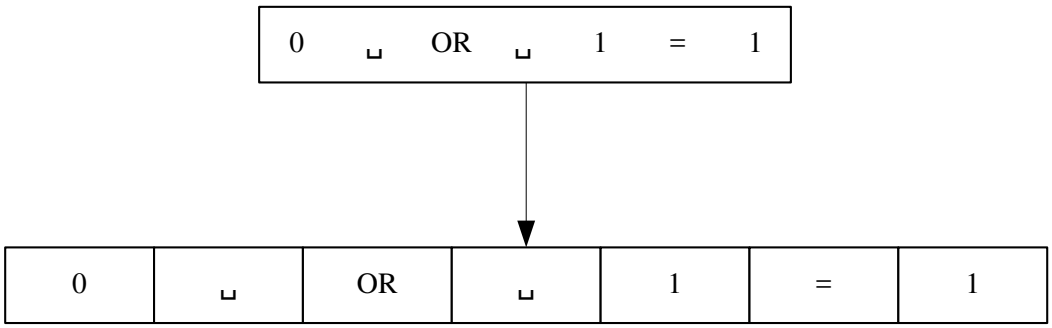
۱.۲.۴ استخراج ویژگی با n-gram

در این مرحله ابتدا رشته‌های حملات را به اجزاء کوچکتری به‌نام توکن تجزیه می‌کنیم. هر توکن می‌تواند یک حرف یا یک کلمه باشد. به‌عنوان مثال، یک نمونه از تجزیه حملات SQLi در شکل ۲.۴ نشان داده شده است. به‌طور کلی هر توکن می‌تواند باعث شکست یا موفقیت یک حمله شود. با این حال، در موارد بسیاری ترکیب این توکن‌ها نیز ممکن است باعث موفقیت یک حمله شود [۳]. بنابراین در این پژوهش بجای استفاده از این توکن‌ها، ما با استفاده از روش n-gram ترکیب این توکن‌ها را از حملات استخراج می‌کنیم. در این پژوهش منظور از زیر رشته همین ترکیب توکن‌ها است.

جدول ۱.۴: مثال از تجزیه‌ی رشته حمله‌ی `--(0␣!␣0)>0␣not␣or␣0)` با استفاده از n-gram به ازای n های مختلف.

Unigram ($n=1$)	Bigram ($n=2$)	Trigram ($n=3$)
0 ,) , ␣ , ...	0) ,)␣ , ␣or , ...	0)␣ ,)␣or , ␣or␣ , ...

جدول ۱.۴ یک نمونه استخراج ویژگی با استفاده از n-gram را به ازای n های مساوی ۱، ۲ و ۳ نمایش می‌دهد. همانطور که در جدول دیده می‌شود، با افزایش مقدار n می‌توان الگوهای پیچیده‌تری را استخراج کرد. با این حال، افزایش n باعث می‌شود تا تعداد زیررشته‌های



شکل ۲.۴: تجزیه‌ی یک نمونه حمله‌ی SQLi به توکن‌های تشکیل‌دهنده‌ی آن.

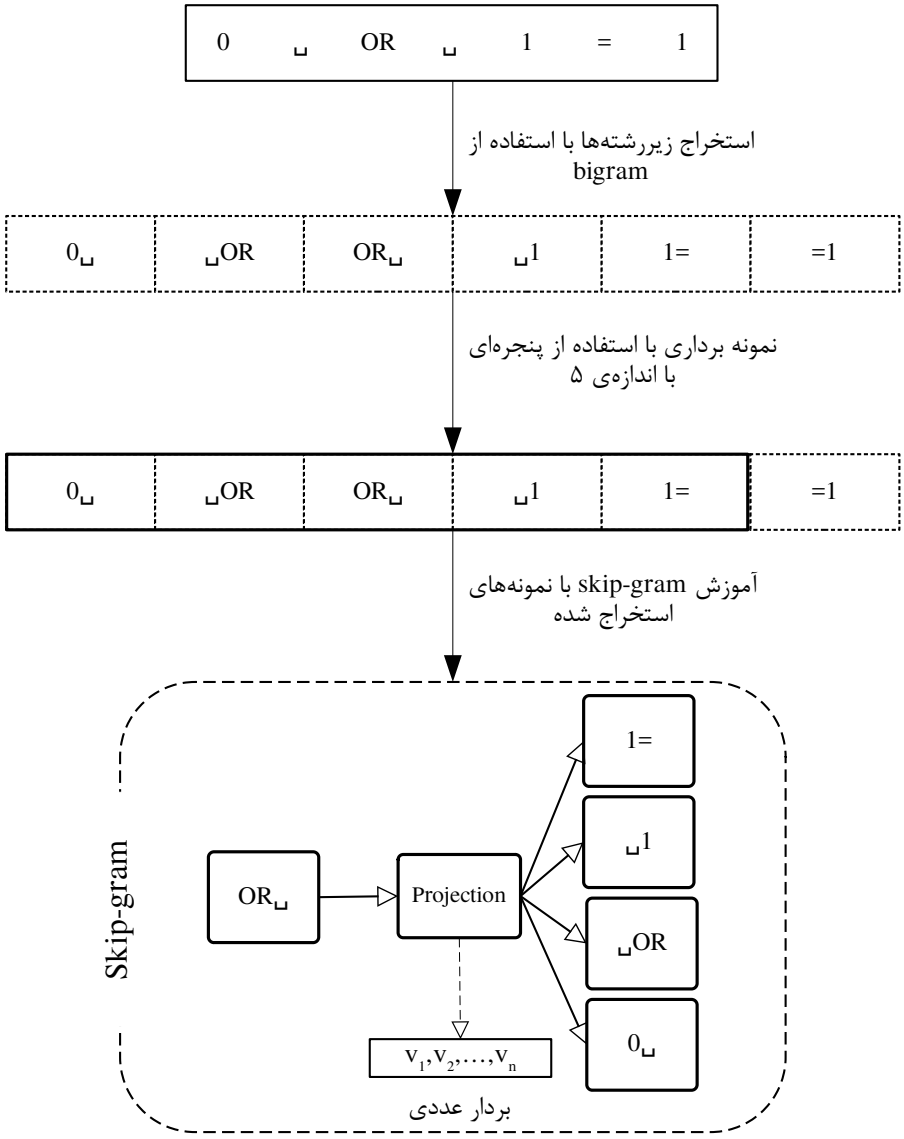
استخراج شده افزایش یابد که این خود می‌تواند تأثیر منفی بر عملکرد روش پیشنهادی داشته باشد. در فصل پنجم به‌طور مفصل تأثیر nهای مختلف را بر عملکرد روش پیشنهادی بررسی خواهیم کرد.

۲.۲.۴ تعبیه‌سازی زیررشته‌ها

در حملات تزریق کد توکن‌های زیادی هستند که می‌توان آن‌ها را بجای هم استفاده کرد به‌طوری که حمله از نظر معنایی تغییری نکند، به‌عنوان مثال در حمله‌ی SQLi توکن‌های OR و || هم معنی هستند و می‌توانند به‌جای یکدیگر استفاده شوند. اما این خاصیت فقط مختص توکن‌ها نمی‌باشد. بلکه می‌تواند زیررشته‌های مختلف را بجای هم استفاده کرد، به‌عنوان مثال زیررشته‌های 1=1 و "a"="a" عبارات همیشه درست هستند که می‌توان از آن‌ها بجای یکدیگر در حمله‌ی SQLi استفاده کرد. برای خوشه‌بندی حملات مشابه، می‌توانیم زیررشته‌های مشابه را به‌عنوان یک زیررشته درنظر بگیریم. بدین‌منظور از word2vec استفاده می‌کنیم تا با تبدیل زیررشته‌ها به بردارهای عددی بتوانیم زیررشته‌های مشابه را بیابیم.

همانطور که در فصل سوم توضیح داده شد، CBOW و skip-gram دو معماری معروف word2vec هستند. در هر دوی این مدل‌ها، کلمات به بردارهای عددی تبدیل می‌شوند. CBOW برای یادگرفتن این بردار عددی از کلمات احاطه کننده‌ی یک کلمه استفاده می‌کند تا کلمه‌ی میانی را پیش‌بینی کند. برخلاف CBOW در skip-gram سعی می‌شود تا با استفاده از کلمات میانی کلمه‌های احاطه‌کننده‌ی آن‌ها پیش‌بینی شوند. در این پژوهش ما مشاهده کردیم که skip-gram عملکرد بهتری نسبت به CBOW داشت. یکی از مشکلات احتمالی CBOW آن است که به‌دلیل اینکه کلمات هدف در این مدل خروجی هستند، بنابراین کلمات کم‌یاب باید با کلمات پرتکرار رقابت کنند. این امر موجب می‌شود که در مجموعه‌دادگان نامتوازن به کلمات کمیاب توجه کمتری شود. درمقابل، در skip-gram به‌دلیل اینکه کلمات هدف، ورودی شبکه هستند، بنابراین کلمات کمیاب و پرتکرار با یکدیگر رقابت نمی‌کنند و همین امر

باعث می‌شود که معماری skip-gram برای مجموعه‌دادگان نامتوازن مناسب‌تر باشد. از آنجایی که مجموعه‌دادگان مورد استفاده در این پژوهش نیز نامتوازن هستند، در روش پیشنهادی از مدل skip-gram برای یادگیری بردارهای عددی استفاده می‌کنیم.



شکل ۳.۴: مثالی از نحوه‌ی آموزش skip-gram با استفاده از زیررشته‌های یک نمونه حمله‌ی SQLi.

در این پژوهش، برای هر مجموعه‌داده یک مدل مستقل skip-gram را آموزش می‌دهیم. شکل ۳.۴ فرآیند آموزش مدل skip-gram را با استفاده از زیررشته‌های یک نمونه حمله‌ی SQLi را به تصویر کشیده است.در مرحله‌ی اول، با استفاده از n-gram زیررشته‌ها استخراج می‌شوند. سپس با استفاده از یک پنجره با اندازه‌ی مشخص از زیررشته‌ها نمونه برداری می‌کنیم. درنهایت، skip-gram را با نمونه‌های بدست‌آمده آموزش می‌دهیم. در این مثال خاص، یک هم معنی برای توکن OR، توکن || است. همچنین توکن ~ نیز به معنی فاصله

است. بنابراین دو زیررشته‌ی OR␣ و ∼|| هم معنی هستند و می‌توان از آن‌ها بجای یکدیگر استفاده کرد. در نتیجه انتظار می‌رود که پس از آموزش skip-gram این دو زیررشته بردارهای عددی مشابه هم داشته باشند.

۳.۲.۴ خوشه‌بندی زیررشته‌ها

پس از تبدیل زیررشته‌ها به بردارهای عددی آن‌ها را با استفاده از شیوه‌ی سلسله‌مراتبی خوشه‌بندی می‌کنیم. بدین‌منظور برای محاسبه‌ی فاصله‌ی بین بردارها از معیار فاصله‌ی کسینوسی[1] استفاده می‌کنیم. در ادامه به شرح این معیار می‌پردازیم.

شباهت کسینوسی معیاری برای محاسبه‌ی فاصله کسینوس زاویه‌ی بین دو بردار است. در صورتی که زاویه‌ی بین دو بردار صفر درجه باشد این مقدار برابر با صفر می‌شود که بیانگر بیشترین شباهت است. هنگامی‌که دو بردار کاملاً مخالف یکدیگر باشند زاویه بین آن دو ۱۸۰ درجه می‌شود که در این صورت میزان شباهت آن‌ها کسینوسی آن‌ها منفی یک است.

می‌توان با کم کردن شباهت کسینوسی از عدد ۱، فاصله‌ی کسینوسی را حساب کرد. اگر فرض کنیم v_1 و v_2 بردارهای عددی زیررشته‌های f_1 و f_2 باشند، در این‌صورت با استفاده از رابطه‌ی زیر می‌توان فاصله‌ی کسینوسی این دو بردار را محاسبه کرد.

$$distance(f_1, f_2) = 1 - \frac{v_1.v_2}{\|v_1\|_{l_2} \times \|v_2\|_{l_2}} \quad (۱.۴)$$

حاصل رابطه‌ی (۱.۴) عددی در محدوده‌ی $[0, 2]$ است. اگر دو بردار کاملاً شبیه به هم باشند مقدار فاصله‌ی آن‌ها برابر ۰ می‌شود. درصورتی که دو بردار بر یکدیگر عمود باشند، فاصله‌ی آن‌ها برابر ۱ می‌شود. فاصله‌ی برابر با ۲ بیانگر آن است که دو بردار کاملاً در جهت‌های مخالف هم قرار دارند.

در این مرحله برای تشخیص تعداد خوشه‌ها، ابتدا یک حد آستانه درنظر گرفته شده و سپس هر داده به عنوان یک خوشه در نظر گرفته می‌شود. پس از آن فاصله‌ی بین خوشه‌ها اندازه‌گیری شده و خوشه‌هایی که فاصله‌ی آن‌ها از حد آستانه کمتر باشد باهم ادغام می‌شوند. این کار تا جایی ادامه می‌یابد که خوشه‌هایی با فاصله‌ی کمتر از حد آستانه وجود نداشته باشد. لازم به ذکر است، برای محاسبه‌ی فاصله‌ی خوشه‌ها از یکدیگر، ابتدا فاصله‌ی تمامی اعضای خوشه‌ها از هم محاسبه می‌شود سپس میانگین فواصل بدست آمده به عنوان فاصله خوشه‌ها از هم درنظر گرفته می‌شود.

[1] Cosine

۴.۲.۴ کدکننده‌ی دودویی

در این پژوهش، میزان تکرار زیررشته‌ها در رشته‌های حملات برای مرحله‌ی خوشه‌بندی حملات اهمیتی ندارد. بنابراین، پس از خوشه‌بندی زیررشته‌ها، با درنظر گرفتن تمام زیررشته‌های یک خوشه به‌عنوان یک زیررشته‌ی واحد، هر خوشه یک ویژگی خواهد بود که مقدار آن صفر یا یک است. مقدار صفر برای یک خوشه بیانگر آن است که یک حمله هیچ زیررشته‌ای ندارد که متعلق که به آن خوشه باشد. همچنین مقدار یک بیانگر آن است که آن حمله شامل حداقل یک زیررشته از آن خوشه است.

جدول ۲.۴: مثالی از تشکیل بردار دودویی برای حمله‌ی p.

حمله	C_1	C_2	C_3	C_4	C_5	C_6
p	۱	۰	۱	۱	۰	۰

جدول ۲.۴ مثالی از تشکیل بردار دودویی برای یک حمله را نمایش می‌دهد. فرض کنیم حمله‌ی p دارای مجموعه زیررشته‌های استخراج شده $F = \{f_1, f_2, f_3, f_4, f_5\}$ باشد و مجموعه‌ی $C = \{C_1, C_2, C_3, C_4, C_5, C_6\}$ خوشه‌های زیررشته‌ها باشد. اگر $\{f_1, f_3\} \subseteq C_3$، $\{f_2\} \subseteq C_1$ و $\{f_4, f_5\} \subseteq C_4$ باشد، درنتیجه مقادیر متناظر با خوشه‌های C_1، C_3 و C_4 در بردار دودویی برابر یک، و مقادیر متناظر با سایر خوشه‌ها برابر صفر می‌شود.

۵.۲.۴ خوشه‌بندی داده‌ها

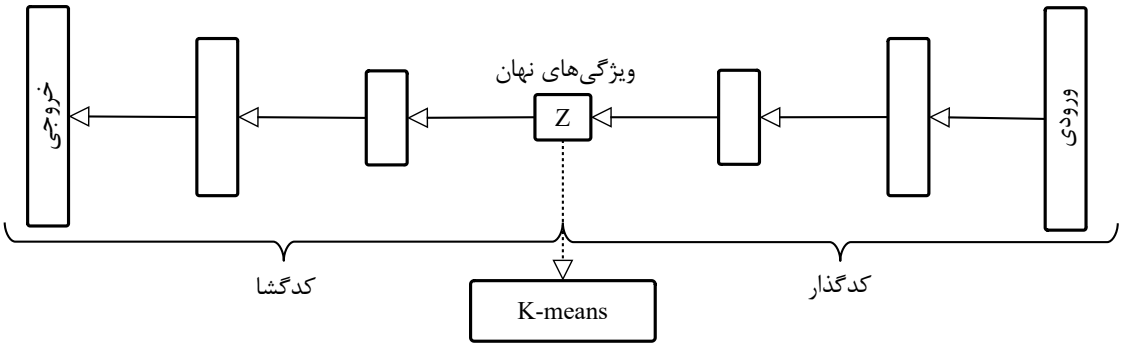
در آخرین مرحله از فاز اول یعنی خوشه‌بندی داده‌ها، با استفاده از بردارهای دودویی، حملات را خوشه‌بندی می‌کنیم. اگرچه معیارهای بسیاری برای اندازه‌گیری شباهت بردارهای دودویی ارائه شده است [۱۴]، اما ما در مشاهداتمان متوجه شدیم که با استفاده از این معیارها نمی‌توان بردارهای دودویی این پژوهش را به خوبی خوشه‌بندی کرد. این می‌تواند به دو دلیل تعداد زیاد ویژگی‌ها و پیچیدگی بردار دودویی باشد. از آنجایی که حملات تزریق کد، رشته‌هایی با قاعده هستند بنابراین حضور یا عدم حضور زیررشته‌ها در یک حمله به یکدیگر وابسته است. بنابراین، در بردار دودویی بدست‌آمده مقادیر ویژگی‌ها به یکدیگر وابسته هستند که این باعث افزایش پیچیدگی بردار ویژگی‌ها می‌شود. بنابراین، قبل از خوشه‌بندی داده‌ها سعی می‌کنیم تا بردارهای دودویی را به فضایی که قابلیت تمایز بیشتری داشته باشد نگاشت کنیم.

به‌طور کلی نگاشت ویژگی‌ها به فضای ویژگی جدید به دو دسته‌ی روش‌های خطی مانند تحلیل مولفه‌های اصلی[1] [۵۴] و روش‌های غیرخطی مانند مدل کرنل[2] [۲۵] تقسیم‌بندی می‌شود. در سال‌های اخیر، با توسعه‌ی شبکه‌های عصبی عمیق روش‌های نگاشت غیرخطی

[1] Principal component analysis (PCA)
[2] Kernel model

مبتنی بر شبکه‌های عمیق ارائه شده‌اند که قادر هستند ویژگی‌ها را به فضای ویژگی جدیدی که بسیار بیشتر مناسب خوشه‌بندی است نگاشت کنند [۳۶].

روش‌های بسیار زیادی برای نگاشت فضای ویژگی و خوشه‌بندی آن‌ها با استفاده از شبکه‌های عصبی عمیق ارائه شده است [۱۱، ۲۰، ۲۶، ۵۶] که در این پژوهش ما از روش ارائه شده در [۲۶] استفاده می‌کنیم. در این روش ابتدا با استفاده از یک شبکه‌ی عصبی کنندکننده‌ی خودکار بردارهای دودویی را به فضای حقیقی با تعداد بسیار کمتری ویژگی نگاشت می‌کنیم. سپس از این ویژگی‌های جدید و الگوریتم خوشه‌بندی k-means برای خوشه‌بندی حملات استفاده می‌کنیم.



شکل ۴.۴: فرآیند نگاشت بردار دودویی به فضای ویژگی نهان و انتقال آن به الگوریتم خوشه‌بندی k-means.

شکل ۴.۴ نحوه‌ی نگاشت ویژگی‌ها به فضای جدید و استفاده از آن‌ها برای خوشه‌بندی را نمایش می‌دهد. در این روش قسمت کدگذار $h = f(\hat{x})$ ویژگی‌های اولیه را به فضای جدید نگاشت می‌کند و ویژگی‌های نهان را استخراج می‌کند. سپس قسمت کدگشا $r = g(h)$ سعی می‌کند تا با استفاده از ویژگی‌های نهان ویژگی‌های اولیه را بازیابی کند. به عبارت دیگر، شبکه‌ی کدکننده‌ی خودکار با کمینه کردن خطای بازیابی $L(\hat{x}, g(f(\hat{x})))$ ویژگی‌های اولیه، ویژگی‌های نهان را یاد می‌گیرد. در این پژوهش به‌دلیل اینکه ورودی و خروجی شبکه‌ی کدکننده‌ی خودکار دودویی هستند، از تابع خطای آنتروپی متقاطع دودویی استفاده می‌کنیم. اگر فرض کنیم بردار ورودی $\hat{x} = [x_1, x_2, \ldots, x_n]$ و خروجی متناظر آن $\hat{y} = [y_1, y_2, \ldots, y_n]$ باشد، خطای بازیابی از رابطه‌ی زیر محاسبه می‌شود.

$$L(\hat{x}, g(f(\hat{x}))) = -\frac{1}{n}\sum_{i=1}^{n} x_i \times \ln y_i + (1 - x_i) \times \ln(1 - y_i) \qquad (۲.۴)$$

در این پژوهش، از یک کدکننده‌ی خودکار با شش لایه‌ی مخفی به علاوه یک گلوگاه به اندازه‌ی سه نرون استفاده شده است که بردارهای دودویی را به فضای سه‌بعدی نگاشت می‌کند. پس از آموزش شبکه‌ی عمیق، قسمت کدگذار و کدگشا را از هم جدا می‌کنیم و برای خوشه‌بندی فقط از خروجی قسمت کدگذار به‌عنوان ورودی الگوریتم k-means استفاده

می‌کنیم. دلیل استفاده از k-means در این پژوهش سادگی آن است و الگوریتم‌های دیگر می‌توانند در پژوهش‌های آتی مطالعه شوند.

۳.۴ استخراج ویژگی نهایی

در این فاز از روش پیشنهادی، ابتدا در هر خوشه زیررشته‌های اضافی را حذف می‌کنیم. سپس معکوس فراوانی سند را برای زیررشته‌های باقی‌مانده محاسبه می‌کنیم تا بردار ویژگی‌های نهایی را تشکیل دهیم. شایان ذکر است از این فاز تمامی عملیات‌ها برای خوشه‌ها به‌صورت مستقل از هم انجام می‌شود.

۱.۳.۴ انتخاب زیررشته‌ها

از آنجایی که هر خوشه زیرمجموعه‌ی کل داده‌ها است، بنابراین مجموعه‌ی زیررشته‌های آن هم زیرمجموعه‌ی کل زیررشته‌ها است. همچنین به‌دلیل اینکه در ادامه‌ی روش جستجو در هر خوشه به‌صورت مستقل از دیگر خوشه‌ها انجام می‌شود، بنابراین برای جستجو درون هر خوشه تنها به مجموعه زیررشته‌های همان خوشه نیاز داریم. بدین‌جهت، در مرحله‌ی اول برای هر خوشه زیررشته‌هایی که در آن خوشه استفاده نشده است را حذف می‌کنیم.

در مرحله‌ی دوم از این بخش، زیررشته‌هایی که اطلاعات زیادی در اختیار ما قرار نمی‌دهند را جهت بهبود عملکرد روش پیشنهادی از مجموعه‌ی زیررشته‌ها حذف می‌کنیم. این زیررشته‌ها آن‌هایی هستند که یا بسیار کم‌یاب و یا بسیار پرتکرار می‌باشند. بدین‌منظور آنتروپی مرتبط با هر زیررشته را حساب می‌کنیم و درصورتی که مقدار آنتروپی آن از حد آستانه کمتر باشد، آن زیررشته را حذف می‌کنیم. نحوه‌ی محاسبه‌ی آنتروپی زیررشته‌ها در رابطه‌ی زیر آمده است.

$$\begin{aligned} E(f_i) &= -(p(f_i)\log_2 p(f_i) + (1 - p(f_i))\log_2(1 - p(f_i))) \\ &= -((\frac{k_i}{N} \times \log_2(\frac{k_i}{N})) + (\frac{N - k_i}{N} \times \log_2(\frac{N - k_i}{N}))) \end{aligned} \quad (۱.۴)$$

در رابطه (۱.۴)، i شماره‌ی زیررشته، $p(f_i)$ احتمال آن است که زیررشته‌ی f_i در یک حمله که تصادفی انتخاب شده است حضور داشته باشد. k_i تعداد حملاتی است که زیررشته‌ی f_i را شامل می‌شوند و N تعداد کل حملات درون خوشه می‌باشد.

۲.۳.۴ محاسبه‌ی معکوس فراوانی سند برای زیررشته‌ها

درمیان زیررشته‌های باقی‌مانده، آن‌ها که کمتر تکرار شده‌اند از اهمیت بیشتری برخوردار هستند. بنابراین، برای تشکیل بردار ویژگی نهایی ابتدا معکوس فراوانی سند را برای زیررشته‌ها محاسبه می‌کنیم. سپس یک بردار تشکیل می‌دهیم که هر عنصر آن متناظر با یک زیررشته از

مجموعه زیررشته‌های خوشه است. درصورتی که یک حمله زیررشته‌ای را شامل شود، مقدار متناظر با آن زیررشته در بردار ویژگی آن حمله برابر با معکوس فراوانی سند آن و در غیر این صورت برابر صفر خواهد بود. معکوس فراوانی سند هر زیررشته را از رابطه‌ی زیر محاسبه می‌کنیم.

$$w_f^i = \ln(\frac{N}{k_i}) \tag{۲.۴}$$

در رابطه (۲.۴)، i شماره‌ی زیررشته، N تعداد حملات درون خوشه و k_i تعداد حملاتی است که شامل زیررشته‌ی f_i هستند.

۴.۴ جستجوگر

از آنجایی که روش ارائه شده در این پژوهش یک شیوه‌ی تست جعبه سیاه است، تنها اطلاعاتی که می‌تواند درمورد فایروال داشته باشد این است که کدام حمله‌های پیشین توانسته‌اند از آن عبور کنند. بنابراین، در این پژوهش ما روش جستجویی ارائه می‌دهیم که ایده‌ی اصلی آن این است که زیررشته‌ای که در حملات ناموفق بیشتری حضور داشته باشد احتمال بیشتری دارد که موجب شکست حملات شود. ما این روش جستجو را جستجوی تطبیقی می‌نامیم. به‌طور دقیق‌تر، در این روش جستجوی تطبیقی حملاتی که توسط فایروال شناسایی می‌شوند را با یکدیگر تجمیع می‌کنیم و آن‌ها را به‌صورت یک سند[1] متنی درنظر می‌گیریم. حال می‌توانیم باقی حملاتی که هنوز تست نشده‌اند را به‌عنوان یک پرس‌وجو درنظر بگیریم. بنابراین با محاسبه‌ی مجموع فراوانی وزنی زیررشته‌های حملات می‌توانیم میزان تعلق حملات به سند حملات ناموفق را محاسبه کنیم. درنتیجه آن حمله‌ای که تعلق بیشتری به سند حملات ناموفق داشته باشد، احتمال شکست بیشتری دارد. بدین‌جهت، برای تست فایروال آن حمله‌ای را انتخاب می‌کنیم که کمترین میزان تعلق به سند حملات ناموفق را داشته باشد. اگر حمله توانست از فایروال عبور کند، تمامی زیررشته‌های آن را از سند حملات ناموفق حذف می‌کنیم. درغیر این صورت آن حمله را نیز به این سند اضافه می‌کنیم.

شبه کد جستجوی تطبیقی در الگوریتم ۱ آمده است. ابتدا در خط ۲، حملات خوشه بارگزاری می‌شوند. خط ۳ فراوانی زیررشته‌هایی که در حملات ناموفق شرکت داشته‌اند را نگهداری می‌کند. درواقع بجای اینکه یک سند واقعی برای حملات ناموفق درست کنیم، کافی است تنها فراوانی زیررشته‌های آن‌ها را نگهداری کنیم. سپس در خط ۴، یک بردار آرایه‌ی دودویی تعریف می‌کنیم. وظیفه‌ی این آرایه حذف زیررشته‌های حملات موفق از سند حملات ناموفق است. بدین صورت که اگر زیررشته‌ای در حملات موفق شرکت داشته باشد مقدار متناظر با آن در این آرایه صفر است. بنابراین فراوانی زیررشته‌های آن در BV همیشه برابر با

[1] Document

الگوریتم ۱ جستجوی تطبیقی

```
1: procedure AdaptiveSearch(cluster, rounds)
2:     P ← getPayloads(cluster)
3:     BV ← getBlockedVector(cluster)
4:     PV ← getBypassingVector(cluster)
5:     S ← {∅}
6:     SR ← +1                                   ▷ Reward
7:     FR ← -0.5                                 ▷ Punishment
8:     R ← 0                                     ▷ Sum of Rewards
9:     for i = 1 to rounds do
10:        if is first time then
11:            test_candidate ← pickRandomPayload(P)
12:        else
13:            rt ← RankPayloads(P, BV, PV)
14:            testCandidate ← pickBestPayload(rt)
15:        end if
16:        P ← P − testCandidate
17:        result ← evaluate(testCandidate)
18:        if result is successful then
19:            S ← S ∪ {testCandidate}
20:            PV ← updateBypassingVector(testCandidate)
21:            R ← R + SR
22:        else
23:            BV ← updateBlockedVector(testCandidate)
24:            R ← R + FR
25:        end if
26:    end for
27:    saveClusterState(P, BV, PV)
28:    return S, R
29: end procedure
```

صفر باقی می‌ماند. خط ۵ یک مجموعه‌ی خالی برای نگهداری حملات موفق تعریف می‌کند. خطوط ۶ و ۷ مقادیر پاداش و تنبیه را مشخص می‌کنند که از آن‌ها برای محاسبه‌ی پاداش نهایی (خط ۸) بدست آمده از خوشه استفاده می‌کنیم.

در اولین باری که جستجو را در یک خوشه انجام می‌دهیم، یک حمله را به‌صورت تصادفی انتخاب می‌کنیم (خطوط ۱۰-۱۲). اما در جستجوهای بعدی هربار میزان تعلق حملات جدید به سند حملات ناموفق را با استفاده از فراوانی وزنی زیررشته‌ها محاسبه می‌کنیم. سپس بهترین حمله را انتخاب می‌کنیم (خطوط ۱۳-۱۴). نحوه‌ی محاسبه‌ی این امتیاز در رابطه‌ی (۱.۴) نشان داده شده است.

$$score(p) = \sum_{i=۱}^{n} b_i \times v_i \tag{۱.۴}$$

در رابطه‌ی (۱.۴)، i شماره‌ی زیررشته است و b_i فراوانی زیررشته‌ی f_i در سند حملات ناموفق است. v بردار ویژگی است و n بیانگر تعداد ویژگی‌ها می‌باشد.

هربار که یک حمله را انتخاب می‌کنیم آن را از مجموعه‌داده‌ی خوشه حذف می‌کنیم سپس آن را روی فایروال تست می‌کنیم (خطوط ۱۶-۱۷). اگر آن حمله با موفقیت از فایروال عبور کند ابتدا آن را به مجموعه‌ی حملات موفق اضافه می‌کنیم (خط ۱۹). سپس مقدار زیررشته‌های آن را در PV صفر می‌کنیم (خط ۲۰). در غیر این صورت با استفاده از رابطه‌ی زیر فراوانی زیررشته‌های سند حملات ناموفق را بروزرسانی می‌کنیم.

$$b_i = s_i \times (tf_i + b_i) \tag{۲.۴}$$

در رابطه‌ی (۲.۴)، i شماره‌ی زیررشته است. b_i فراوانی زیررشته‌ی f_i در سند حملات ناموفق و tf_i فراوانی آن در حمله‌ی منتخب است. s_i مقدار متناظر با زیررشته‌ی f_i در PV است، که مقدار آن صفر یا یک است.

برای درک بهتر، فرض کنیم یک خوشه از چهار حمله تشکیل شده است. جدول ۳.۴، فراوانی زیررشته‌های هر حمله را به همراه بردار ویژگی آن‌ها نمایش می‌دهد. فرض کنیم دو حمله‌ی اول این خوشه برروی فایروال تست شده‌اند و نتیجه آن‌ها ناموفق بوده است. برای محاسبه‌ی بردار $\hat{b}$ فراوانی زیررشته‌های این دو حمله را با یکدیگر جمع می‌کنیم (رابطه (۲.۴)). حاصل بردار $\hat{b} = [۳,۱,۱,۰]$ می‌شود. برای انتخاب حمله‌ی بعدی کافی است تا حاصل ضرب نقطه‌ای بردار $\hat{b}$ و باقی بردار ویژگی‌ها را محاسبه‌کنیم. حاصل این ضرب برای حملات سوم و چهارم برابر با $Rank(p_۳) = ۰/۹۷$ و $Rank(p_۴) = ۱/۱۲$ است. پس از آن حمله با کمترین امتیاز یعنی حمله‌ی سوم را انتخاب می کنیم. اگر این حمله بتواند با موفقیت از فایروال عبور کند، مقادیر متناظر با زیررشته‌های $f_۲$، $f_۳$ و $f_۴$ را در بردار $\hat{s}$ برابر با صفر قرار می‌دهیم. درنتیجه بردار $\hat{s}$ برابر با $\hat{s} = [۱,۰,۰,۰]$ می‌شود.

جدول ۳.۴: مثالی از فراوانی زیررشته‌های حملات یک خوشه به همراه معکوس فراوانی سند آن‌ها.

شماره حمله	فراوانی زیررشته‌ها				معکوس فراوانی سند زیررشته‌ها			
	f_1	f_2	f_3	f_4	f_1	f_2	f_3	f_4
۱	۲	۰	۱	۰	۰/۲۸	۰	۰/۶۹	۰
۲	۱	۱	۰	۰	۰/۲۸	۰/۲۸	۰	۰
۳	۰	۱	۲	۳	۰	۰/۲۸	۰/۶۹	۱/۳۸
۴	۲	۲	۰	۰	۰/۲۸	۰/۲۸	۰	۰

در خطوط ۲۱ و ۲۲ مقدار پاداش بروزرسانی می‌شود که جلوتر آن را توضیح می‌دهیم. پس از آن در خط ۲۷ وضعیف فعلی خوشه ذخیره می‌شود و در انتها، در خط ۲۸، حملات موفق و پاداش بدست آمده به‌عنوان مقادیر بازگشتی تابع را به‌عنوان خروجی‌های آن برمی‌گرداند.

همانطور که پیشتر نیز مطرح شد، درصورت وجود آسیب‌پذیری حملات مشابه به هم می‌توانند از فایروال عبور کنند در نتیجه درصورت خوشه‌بندی درست آن‌ها می‌توان انتظار داشت که در یک خوشه قرار بگیرند. علاوه بر این، ما در مشاهدات خود دریافتیم که حملات موفق معمولاً کم‌یاب هستند. بنابراین، فقط تعداد اندکی خوشه شامل حملات موفق هستند. بدین‌جهت، یافتن این خوشه‌ها و محدود کردن جستجو به آن‌ها می‌تواند تا حد بسیار زیادی تعداد جستجوهای ناموفق را کاهش دهد. بدین‌منظور ما از سیاست ϵ-greedy استفاده می‌کنیم که در ادامه به شرح آن می‌پردازیم.

ابتدا نیاز است به تعریف پارامترهای مورد استفاده در سیاست مورد استفاده در روش پیشنهادی بپردازیم. در ادامه این تعاریف آمده اند.

- **پاداش:** یک مقدار مثبت است که به‌عنوان پاداشی برای هر حمله‌ی موفق درنظر گرفته شده است. این مقدار را برابر با یک قرار می‌دهیم (خط ۶ از الگوریتم ۱)
- **تنبیه:** یک مقدار منفی است که به‌عنوان تنبیه برای هر حمله‌ی ناموفق درنظر گرفته شده است.
- **بازی:** هربار بازی تعداد R جستجو در خوشه است (خط ۵ از الگوریتم ۲).
- **پاداش بازی:** به مجموع پاداش‌ها و تنبیه‌های یک خوشه به ازای یک‌بار بازی، پاداش بازی می‌گوییم.
- **پاداش متوسط:** به متوسط پاداش‌های بدست آمده از یک خوشه، پاداش متوسط آن خوشه می‌گوییم.

الگوریتم ۲ سیاست ϵ-greedy.

1: **procedure** EPSILONGREEDY
2: $PC \leftarrow$ *Payload Clusters*
3: $S \leftarrow \{\emptyset\}$
4: $max\_\epsilon, \epsilon \leftarrow$ *Epsilon*
5: $R \leftarrow$ *Number of Searching Rounds per episode*
6: $K \leftarrow$ *Update Rate for Epsilon*
7: $E \leftarrow$ *Episodes*
8: $AR \leftarrow$ *Initial Average Reward of Each Cluster*
9: **for** $i = 1$ to E **do**
10: $c \leftarrow$ *pickCluster*(ϵ)
11: *bypassing, reward* $\leftarrow$ *ADAPTIVESEARCH*(c, R)
12: $AR \leftarrow$ *updateAverageReward*(c, *reward*, AR)
13: $S \leftarrow S \cup$ *bypassing*
14: $\epsilon \leftarrow$ *updateEpsilon*(max_ϵ, i, K)
15: **end for**
16: **end procedure**

الگوریتم ۲ سیاست ϵ-greedy مورد استفاده در روش پیشنهادی را نمایش می‌دهد. در این الگوریتم ابتدا خوشه‌ها را با استفاده از میزان پاداش متوسط بدست آمده از آن‌ها در بازی‌های قبلی ارزش‌گذاری می‌کنیم. سپس برای بازی جدید، یا یک خوشه تصادفی یا باارزش‌ترین خوشه را انتخاب می‌کنیم. احتمال انتخاب خوشه‌ی تصادفی ϵ و احتمال انتخاب بهترین خوشه $1 - \epsilon$ است (خط ۱۰).

از آنجایی که در ابتدای جستجو هیچ دانشی در اختیار ما نیست، بنابراین، در ابتدای جستجو مقدار ϵ را یک مقدار نزدیک زیاد مانند ۰/۹ درنظر می‌گیریم تا نرخ اکتشاف را افزایش دهیم. هرچه تعداد جستجو بیشتر می‌شود، دانش بدست آمده درباره‌ی خوشه‌ها نیز افزایش می‌یابد. بنابراین با افزایش تعداد جستجوها مقدار ϵ را کاهش می‌دهیم تا میزان اکتشاف کاهش و میزان بهره‌وری افزایش یابد (خط ۱۴). نحوه‌ی کاهش میزان ϵ در رابطه‌ی زیر آمده است.

$$\epsilon = \epsilon_m \times e^{-(k \times \tau)} \tag{۳.۴}$$

در رابطه (۳.۴)، ϵ_m بیشترین مقدار برای ϵ است. k یک عدد ثابت است و τ تعداد بازی‌های

انجام شده است.

۵.۴ جمع‌بندی

در این فصل روش پیشنهادی برای کشف خودکار آسیب‌پذیری‌های فایروال‌های وب در سه فاز خوشه‌بندی، استخراج ویژگی و جستجو شرح داده شد. در فاز خوشه‌بندی ابتدا ویژگی‌های اولیه از حملات، استخراج شده و سپس حملات خوشه‌بندی می‌شوند. در فاز بعدی، ابتدا عملیات کاهش‌ویژگی انجام می‌شود و درنهایت بردار ویژگی‌های نهایی تشکیل می‌شود. در فاز جستجو که آخرین فاز از روش پیشنهادی است، عملیات جستجو درون و برون خوشه‌ای انجام می‌شود. جستجوی درون خوشه‌ای برای کشف حملات موفق موجود درون خوشه‌ها اجرا می‌شود، و جستجوی برون خوشه‌ای جهت یافتن خوشه‌هایی که شامل حملات موفق هستند انجام می‌شود.

فصل ۵

پیاده‌سازی و نتایج

در این فصل ابتدا به طرح پرسش‌های تحقیق می‌پردازیم. سپس، نحوه‌ی انجام آزمایش‌ها را توضیح می‌دهیم. در نهایت پس از توضیح محیط آزمایش و نحوه‌ی انتخاب پارامترهای مورد استفاده در روش پیشنهادی، به پرسش‌های تحقیق در قالب آزمایش‌های مختلف پاسخ می‌دهیم. شایان ذکر است در بخش نتایج جهت سهولت در گزارش نتایج نام روش پیشنهادی را RAT[1] می‌گذاریم که برگرفته از نام مقاله‌ی مستخرج از این پژوهش است.

۱.۵ پرسش‌های تحقیق

هدف از طراحی و اجرای آزمایش‌های این پروژه پاسخ به چهار پرسش زیر است:

۱. آیا استفاده از n-gram اهمیتی دارد؟

۲. خوشه‌بندی حملات چه تأثیری بر عملکرد روش پیشنهادی می‌گذارد؟

۳. نحوه‌ی عملکرد روش پیشنهادی در مقایسه با رقیبان خود چگونه است؟

۴. آیا روش پیشنهادی در عمل هم از کارایی لازم برخوردار است؟

[1] RAT: Reinforcement-Learning-Driven and Adaptive Testing for Vulnerability Discovery in Web Application Firewalls

مقصود از سؤال اول بررسی نحوه‌ی عملکرد روش استخراج ویژگی ارائه شده در این پژوهش بر کارایی روش پیشنهادی است. سؤال دوم به بررسی چگونگی تأثیرگذاری عمل خوشه‌بندی بر عملکرد روش پیشنهادی و به‌طور کلی تست جعبه سیاه می‌پردازد. هدف از سؤال سوم مقایسه‌ی روش پیشنهادی با سه روش ART4SQLi، ML-Driven E و XSSART است، و در آخر سؤال چهارم عملیاتی بودن یا نبودن روش‌پیشنهادی در دنیای واقعی را بررسی می‌کند.

۲.۵ فرآیند کلی آزمایش‌ها

همانطور که پیش‌تر نیز مطرح شد، به‌دلیل اینکه هدف این آزمایش تنها فایروال است و نه برنامه‌های تحت حفاظت آن، بنابراین در آزمایش‌ها اهمیتی ندارد که پارامترهای درخواستی که برای سرور ارسال می‌کنیم توسط برنامه‌ی تحت وب قابل قبول باشد یا خیر. چراکه فایروال فارق از معتبر بودن یا نبودن پارامترهای درخواست، همه‌ی آن‌ها را بررسی می‌کند. بدین‌جهت برای تشکیل درخواست‌های HTTP از پارامترهای ساختگی[1] استفاده می‌کنیم. به‌عنوان مثال فرض کنیم q یک پارامتر ساختگی برای درخواست HTTP از نوع GET باشد، و رشته‌ی "0%20or%201=1" حمله‌ای باشد که می‌خواهیم روی فایروال تست کنیم. بنابراین، درخواست نهایی برای ارسال به فایروال به‌صورت زیر خواهد بود.

```
http://example.com/?q=0%20or%201=1
```

برای تست آسیب‌پذیری SQLi پارامتر GET و پارامتر cookie که در سرآیند[2] درخواست HTTP قرار دارد را هدف قرار می‌دهیم. برای تست آسیب‌پذیری در مقابل حملات XSS، فقط پارامتر GET را هدف قرار می‌دهیم. شایان ذکر است که دلیل انتخاب درخواست‌ها از نوع GET سادگی آن‌ها است و روش‌پیشنهادی وابسته به پروتکل‌های خاص نیست و قابل پیاده‌سازی برای تمامی پروتکل‌ها از جمله پروتکل POST GET و SOAP است.

هنگامی‌که فایروال درخواستی را دریافت می‌کند، بررسی می‌کند که آیا آن درخواست آلوده است یا خیر. درصورتی که تشخیص دهد آن درخواست آلوده است، در پاسخ کد ۴۰۳ را به کلاینت برمی‌گرداند. بنابراین، هنگامی‌که در آزمایش‌ها کد ۴۰۳ دریافت کنیم تشخیص می‌دهیم که آن حمله ناموفق بوده است. در غیر این‌صورت و در صورت دریافت کدهای دیگر غیر از کدهای خانواده‌ی ۵۰۰ که به معنی وجود خطا در سرور هستند، حمله را یک حمله‌ی موفق درنظر می‌گیریم.

[1] Dummy
[2] Header

۳.۵ محیط آزمایش

روش پیشنهادی از دو بخش پردازش‌گر داده و جستجوگر تشکیل شده است. بخش پردازش‌گر برای هر یک مجموعه‌داده‌ها تنها یکبار اجرا می‌شود و تا زمانی که تغییری در این داده‌های آن‌ها صورت نپذیرد نیاز به اجرای مجدد آن نیست. این بخش برای اجرا به یک سیستم با حداقل‌های زیر نیاز دارد.

- 32GB of Memory
- Graphic Card with the minimum compute capability 3.0

بخش جستجوگر که می‌توان با استفاده از خروجی بخش پردازشگر از آن بارها استفاده کرد، برای اجرا به سیستمی با حداقل مشخصات زیر نیاز دارد.

- 8GB of memory
- 2.10GHz duo core CPU

اگرچه بخش جستجوگر به سخت‌افزار قدرتمندی نیاز ندارد، اما ما برای سرعت بخشیدن به انجام آزمایش‌ها برای بخش جستجوگر از سروری با مشخصات زیر استفاده کردیم.

- Double 2.10GHz Intel® Xenon® processors
- 64GB of memory

روش پیشنهادی با زبان Python 3.6 پیاده‌سازی شده است و سورس‌کد آن در گیتهاب[۱] در دسترس عموم قرارداده شده است. همچنین بخش شبکه عصبی این پروژه با استفاده از Keras-GPU 2.2.4 پیاده‌سازی شده است و برروی سرور Google Colab[۲] اجرا شده است. در ادامه‌ی این بخش به بررسی فایروال‌های مورد آزمایش، مجموعه‌دادگان و معیارهای ارزیابی می‌پردازیم.

۱.۳.۵ فایروال‌های مورد بررسی

در آزمایش‌ها از سه فایروال استفاده می‌شود. دوتا از آن‌ها فایروال‌های متن باز با نام‌های NAXSI[۳] و ModSecurity[۴] هستند و دیگری یک فایروال شخصی‌سازی شده مبتنی بر ModSecurity است. در ادامه شرح مختصری از این فایروال‌ها می‌دهیم.

[1] https://github.com/mhamouei/rat
[2] https://colab.research.google.com
[3] https://www.nbs-system.com
[4] https://modsecurity.org

فایروال ModSecurity ابزاری است جهت محافظت از برنامه‌های وب در برابر حملات مختلف از جمله حملات تزریق کد. اگرچه ModSecurity قابل پیاده‌سازی برروی انواع وب‌سرورها است، اما ما آن را روی وب‌سرور Apache و روی یک ماشین مجازی[۱] محلی[۲] راه‌اندازی کردیم.

فایروال NAXSI یک ماژول برای وب‌سرور Nginx است. این فایروال نیز قادر است از برنامه‌های وب درمقابل حملات متنوعی از جمله انواع حملات تزریق کد محافظت کند. همانند فایروال ModSecurity این فایروال را نیز روی یک ماشین مجازی محلی راه‌اندازی کردیم.

برخلاف فایروال‌های قبلی که برروی ماشین‌های مجازی محلی راه‌اندازی شدند، فایروال شخصی‌سازی شده مسئولیت محافظت از سرورهای واقعی را برعهده دارد. این فایروال از سرورهای یک مجموعه‌ی آموزشی محافظت می‌کند که شامل اطلاعات هزاران دانشجو است.

۲.۳.۵ مجموعه‌دادگان

در این پژوهش از دو مجموعه‌داده استفاده شده است که یکی از آن‌ها متعلق به حملات SQLi و دیگری متعلق به حملات XSS است. این دو مجموعه‌داده را نیز همانند سورس‌کد در گیتهاب در دسترس عموم قرار داده ایم. تعداد حملات موجود در هر مجموعه‌داده در جدول ۱.۵ نشان داده شده است.

جدول ۱.۵: تعداد حملات در هر مجموعه‌داده.

مجموعه‌داده	تعداد حملات
SQLi	۲،۴۱۷،۷۲۰
XSS	۱،۷۹۸،۰۶۲

برای تهیه‌ی مجموعه‌داده‌ی SQLi، از دستورزبانی که در [۳] ارائه شده است استفاده شده است. به‌طور دقیق‌تر، از آنجایی که این دستور زبان یک دستور زبان محدود است و تعداد حالات خروجی آن نامحدود نیست، بنابراین تمامی حملاتی که می‌شود با این دستورزبان تولید کرد را تولید و ذخیره کردیم. همچنین به‌دلیل اینکه روشی که روش پیشنهادی را با آن مقایسه می‌کنیم از این دستور زبان استفاده کرده است، مقایسه‌های انجام شده نیز منصفانه می‌باشد.

برای تهیه‌ی مجموعه‌داده‌ی حملات XSS، از ابزاری به نام dharma[۳] استفاده کردیم. این ابزار از یک دستورزبان برای تولید حملات XSS استفاده می‌کند. اما دستورزبان استفاده شده در این ابزار محدود نیست. بنابراین، تعداد حملات را محدود کردیم و پس از اتمام تولید

[1] Virtual machine
[2] Local
[3] https://github.com/MozillaSecurity/dharma

حملات توسط ابزار، حملات تکراری را از مجموعه‌داده حذف کردیم. نمونه داده‌های موجود در دیتاست در جدول ۲.۵ نمایش داده شده است.

جدول ۲.۵: نمونه رشته حملات هر مجموعه‌داده.

مجموعه‌داده	نمونه رشته حمله
SQLi	0') or ('1'='1'/*
XSS	xxs link

۳.۳.۵ معیارهای ارزیابی اثربخشی

اصلی‌ترین هدف روش پیشنهادی کشف بیشترین تعداد آسیب‌پذیری با کمترین تعداد جستجو است. بنابراین، در آزمایش‌ها حملات موفق را به‌عنوان مثبت‌ها و حملات ناموفق را به عنوام منفی‌ها درنظر می‌گیریم. درنتیجه‌ی این نام‌گذاری، آسیب‌پذیری‌های کشف شده مثبت‌های درست[۱] (TP) و تلاش‌های ناموفق مثبت‌های کاذب[۲] (FP) هستند.

جدول ۳.۵: میزان آسیب‌پذیری‌های موجود در مجموعه‌دادگان برای فایروال‌های متن‌باز.

فایروال	SQLi		XSS
	پارامتر GET	Cookie	پارامتر GET
ModSecurity	۰/۰۰۷٪	۰/۰۷۹٪	۰/۰۱۵٪
NAXSI	۰/۰۰۵٪	۰/۰۰۵٪	۰/۰۰۴٪

جهب مقایسه‌ی میزان اثربخشی روش‌پیشنهادی و روش ML-Driven E، میزان TPها را برای هر روش در تعداد محدودی درخواست HTTP اندازه‌گیری می‌کنیم. از آنجایی که فایروال‌های NAXSI و ModSecurity روی ماشین‌های مجازی محلی راه‌اندازی شده‌اند، می‌توان روی آن‌ها حمله‌ی آزمون جامع[۳] را انجام داد. بنابراین، با انجام حمله‌ی آزمون جامع تمامی حملات مجموعه‌دادگان را روی فایروال‌ها تست کردیم و تعداد کل آسیب‌پذیری‌های این دو فایروال‌ها را محاسبه کردیم. در جدول ۳.۵ نشان داده شده است که هر فایروال نسبت به چند درصد از حملات هر مجموعه‌داده آسیب‌پذیر است. با دانستن کل آسیب‌پذیری‌ها، در آزمایش‌ها می‌توانیم نرخ مثبت‌های درست[۴] (TPR) را برای فایروال‌های متن‌باز محاسبه

[1] True positives
[2] False positives
[3] Brute-force attack
[4] True positives rate

کنیم. بنابراین، برای گزارش میزان موفقیت روش‌های مورد آزمایش این بخش در یافتن آسیب‌پذیری‌های فایروال‌های متن‌باز از TPR استفاده می‌کنیم. اما از آنجایی که امکان انجام حمله‌ی آزمون جامع روی فایروال شخصی سازی شده وجود ندارد، بنابراین برای مقایسه‌ی میزان موفقیت روش‌ها در کشف آسیب‌پذیری‌های فایروال شخصی سازی شده فقط میزان TP را گزارش می‌کنیم. علاوه بر این، برای مقایسه‌ی روش‌پیشنهادی با روش‌های ART4SQLi و XSSART نیاز است تا تعداد حملات ناموفق قبل از کشف اولین حمله‌ی موفق را گزارش کنیم. بدین‌منظور برای مقایسه‌ی روش‌پیشنهادی با این دو روش مقدار FP را گزارش می‌کنیم.

۴.۳.۵ معیارهای ارزیابی کارایی

جهت ارزیابی کارایی روش پیشنهادی، مقدار زمانی که برای انتخاب و تست هر حمله سپری می‌شود[1] (TSR) را اندازه‌گیری می‌کنیم. به منظور اندازه‌گیری TSR برای روش‌پیشنهادی، در هربار بازی زمان را اندازه‌گیری می‌کنیم و در انتها تقسیم بر تعداد کل درخواست‌های ارسالی می‌کنیم. درنهایت متوسط TSR را به ازای تکرارهای آزمایش‌ها گزارش می‌کنیم.

۵.۳.۵ معیار Sillhouette

یکی از معیارهای مهمی که نتایج حاصل از روال خوشه‌بندی را می‌تواند تحت تاثیر قرار دهد، تعداد خوشه‌های استفاده‌شده است. این‌که چه تعداد خوشه در یک فرایند خوشه‌بندی برای جداسازی داده‌ها تعیین شود، مهم و در عین حال مسئله‌ای است که الگوریتم مشخصی برای آن وجود ندارد. بهترین ایده‌ای که می‌توان برای مشاهده نتایج حاصل از خوشه‌بندی و کیفیت آن ارائه داد، بررسی معیارهای مناسب با بکارگیری تعداد خوشه‌های متفاوت است. در ادامه به معرفی یکی از این معیارهای بکارگرفته شده میپردازیم.

معیار Sillhouette [۴۳] روشی است که برای ارزیابی خوشه‌های ایجاده شده در الگوریتم‌های خوشه‌بندی همچون k-Means استفاده می‌گردد، که در آن به بررسی میزان شباهت هر نمونه نسبت به نمونه‌های موجود دیگر در یک خوشه با درنظرگیری ویژگی‌های داده‌ها پرداخته می‌شود. این معیار برای هریک از نمونه‌های موجود در خوشه‌های مختلف محاسبه می‌شود. در محاسبه این معیار، نیاز به پیداکردن فاصه‌های بیان‌شده در ادامه برای هر نمونه، نسبت به نمونه‌های دیگر در خوشه‌های گوناگون، در طول فرایند خوشه‌بندی است.

- **فاصله میانگین درون خوشه‌ای:** این فاصله برابر است با میانگین فاصله هر نمونه نسبت به نمونه‌های دیگر خوشه‌ای که در آن قرار دارند.
- **فاصله میانگین از نزدیک‌ترین خوشه:** این فاصله برابر است با میانگین فاصله هر نمونه، از نمونه‌های موجود در نزدیکترین خوشه، نسبت به خوشه‌ای که نمونه موردنظر

[1] Time spent per request

در آن قرار دارد.

مقداری که برای معیار Sillhouette بدست می‌آید، در بازه $[-۱,۱]$ خواهد بود. مقادیر نزدیک به یک بیانگر یک خوشه متراکم، که نسبت به خوشه‌های دیگر به خوبی جداشده است. مقادیر نزدیک صفر نشان از یک همپوشانی داده‌های مرزی خوشه‌های همسایه دارد. مقادیر در بازه $[۰,-۱]$ بیانگر انتخاب خوشه‌های نامناسب است.

۶.۳.۵ روش Wilcoxon

آزمون Wilcoxon[۱] [۳۳] یکی از آزمون‌های غیرپارامتریک است که در آن هدف مقایسه خروجی دو گروه مستقل است. این آزمون عموما برای آزمودن اینکه دو نمونه آماری (نمونه در اینجا به معنی یک مجموعه داده چندتایی است) که از یک جامعه مشخص مشتق شده‌اند (بدین معنی که از یک جامعه آماری دو نمونه چندتایی انتخاب شده است) چه میزان دارای شباهت می‌باشند، استفاده می‌شود. در آزمون Wilcoxon سه مفهوم کلیدی بسیار مهم وجود دارند که در ادامه به آن‌ها اشاره می‌کنیم.

۱. **فرضیه:** در آزمون Wilcoxon معمولا دو فرضیه پوچ [۲] و جایگزین [۳] مطرح می‌گردند. در بررسی آزمون نیز قاعدتا یکی از فرضیه‌ها رد و دیگری با توجه به رد فرضیه اول، پذیرفته می‌شود. به عنوان مثال یک فرضیه‌ی پوچ می‌تواند این باشد که میانه‌ی نمونه‌های آماری دسته‌ی اول از میانه‌ی نمونه‌های آماری دسته‌ی دوم بزرگتر است. بنابراین، فرضیه جایگزین، عکس فرضیه‌ی پوچ خواهد بود.

۲. **مقدار α[۴]:** این پارامتر بیان یک سطح اطمینان تصمیم‌گیری برای پذیرش و یا رد یک فرضیه است. به عبارت دیگر مقدار α بیانگر آن است که چه مقدار خطا ممکن است در رد فرضیه‌ی پوچ وجود داشته باشد.

۳. **مقدار p[۵]:** مقدار p در آزمون Wilcoxon بیانگر احتمال صحت فرضیه‌ی پوچ است. بنابراین هرچه مقدار p کمتر باشد احتمال پذیرفته شدن فرضیه جایگزین بیشتر می‌شود. در این آزمون برای رد یک فرضیه‌ی پوچ مقدار p باید کمتر از مقدار α باشد.

۴.۵ انتخاب پارامترها

در روش‌پیشنهادی پارامترهای متعددی وجود دارد که باید مقادیرشان تعیین شود. برای الگوریتم n-gram تنها یک پارامتر یعنی مقدار n باید تعیین شود که در قسمت نتایج به‌طور

[1] Wilcoxon rank-sum test
[2] Null hypothesis
[3] Alternative hypothesis
[4] Significance level
[5] p-value

مفصل به بررسی آن می‌پردازیم. باقی پارامترها در ادامه توضیح داده شده اند.

۱.۴.۵ پارامترهای الگوریتم خوشه‌بندی سلسه مراتبی

در این مرحله، نیاز است تا یک حد آستانه جهت خوشه‌بندی زیررشته‌ها تعیین کنیم. بنابراین زیررشته‌هایی که متوسط فاصله‌ی آن‌ها کمتر از حد آستانه باشد در یک خوشه قرار می‌گیرند. اگر حد آستانه را مقدار زیادی درنظر بگیریم، تعداد خوشه‌ها کاهش یافته و اندازه آن‌ها افزایش می‌یابد که این باعث می‌شود جزئیات زیادی را از دست بدهیم. اما در صورتی که حد آستانه بسیار کوچک باشد نیز جزئیات زیادی را نگهداری می‌کنیم. در مشاهداتمان متوجه شدیم که یک مقدار حد آستانه بین ۰٫۲ تا ۰٫۵ می‌تواند مقدار مناسبی باشد و بهترین نتایج را توانستیم با حد آستانه‌ی ۰٫۳ بدست بیاوریم. این حد آستانه به این معنی است که زیررشته‌هایی که متوسط شباهت کسینوسی آن‌ها با یکدیگر بیشتر از ۸۵ درصد است در یک خوشه قرار می‌گیرند.

۲.۴.۵ پارامترهای مدل Skip-gram

برای پیاده‌سازی skip-gram از کتابخانه‌ی Gensim[1] در پایتون استفاده شده است و بیشتر پارامترها همان مقادیر پیش‌فرض این کتابخانه در نظر گرفته شده اند. برای این بخش از روش پیشنهادی تنها پارامتری که تغییر می‌دهیم اندازه‌ی پنجره می‌باشد. به‌طور دقیق‌تر، اندازه‌ی پنجره را متناسب با مقدار n در n-gram انتخاب می‌کنیم. به‌عنوان مثال اگر $n = ۲$ باشد، به‌دلیل اینکه بخش‌هایی از زیررشته‌ی میانی در زیررشته‌های قبلی و بعدی تکرار شده است، اندازه‌ی پنجره را ۵ درنظر میگیریم تا آموزش skip-gram معنی‌دار باشد.

۳.۴.۵ معماری شبکه‌ی عمیق کد کننده‌ی خودکار

در روش پیشنهادی از ساده‌ترین معماری کدکننده‌ی خودکار برای نگاشت ویژگی‌ها استفاده شده است. از آنجایی که ورودی‌های شبکه عمیق در اینجا بردار هستند یک شبکه‌ی عمیق ساده می‌تواند دقت بالایی را در بازسازی ورودی‌ها بدست آورد. با این حال، نحوه‌ی انتخاب تعداد لایه‌ها و اندازه‌ی آن‌ها تأثیر قابل توجهی در افزایش یا کاهش دقت شبکه‌ی عمیق دارد. از آنجایی که شیوه‌ی تعیین دقیق پارامترهای شبکه‌ی عمیق همچنان یک مشکل حل نشده است [۳۹]، بنابراین سعی کردیم تا پارامترهای شبکه‌ی عمیق را به‌صورت دستی و با سعی و خطا بدست آوریم. در این مرحله مشاهده‌کردیم که تعداد سه لایه مخفی در هر قسمت کدگشا و کدگذار کافی است و از آنجایی که سعی داشتیم تا ابعاد فضای جدید ویژگی‌ها را تا حد امکان کاهش دهیم توانستیم با سه بعد دقت‌های ۹۵٫۹۲ و ۹۶٫۷۶ درصد را به‌ترتیب برای

[1] https://radimrehurek.com/gensim

مجموعه‌دادگان XSS و SQLi بدست آوریم. معماری نهایی شبکه عمیق استفاده شده برای هر مجموعه‌داده در جدول ۶ نشان‌داده شده است.

جدول ۴.۵: معماری کدکننده‌ی خودکار استفاده شده در این پژوهش.

نوع حمله	ورودی	کدگذار			گلوگاه	کدگشا			خروجی
		لایه‌های مخفی				لایه‌های مخفی			
SQLi	۵۲	۳۶	۲۳	۱۰	۳	۱۰	۲۳	۳۶	۵۲
XSS	۳۴۷	۲۶۱	۱۷۵	۸۹	۳	۸۹	۱۷۵	۲۶۱	۳۴۷

۴.۴.۵ تعداد خوشه‌ها

پیچیدگی زمانی انتخاب هر حمله برابر با $O(n) \times O_s$ است،که n تعداد حملات درون خوشه و O_s پیچیدگی زمانی تابع محاسبه‌ی امتیاز حملات می‌باشد. بنابراین، جستجو در خوشه‌های کوچک پیچیدگی زمانی کمتری دارد. افزایش تعداد خوشه‌ها باعث می‌شود که هم اندازه‌ی خوشه‌ها و هم تعداد ویژگی‌ها کاهش یابد. با این حال افزایش تعداد خوشه‌ها باعث بوجود آمدن دو مشکل اساسی می‌شود:

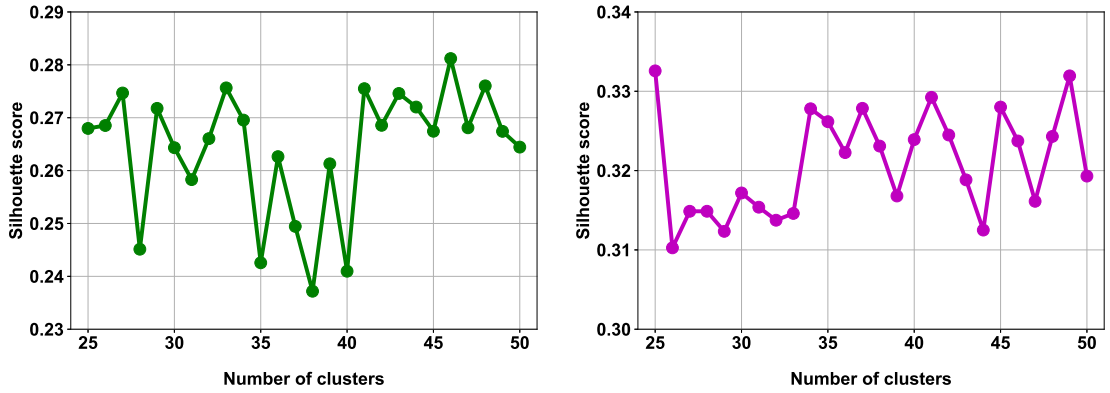
۱. حملات موفق در خوشه‌های بیشتری پخش می‌شوند که باعث می‌شود تراکم آن‌ها درون خوشه‌ها کاهش یافته و کشف آن‌ها سخت‌تر شود.

۲. سیاست ϵ-greedy به اکتشافات بیشتری نیاز دارد که خوشه‌های مؤثر را بیابد. همچنین به‌دلیل مشکل اول، احتمال قرار گرفتن در بهینه‌ی محلی افزایش می‌یابد.

مشاهدات ما نشان داد که تعداد ۲۵ تا ۵۰ خوشه می‌تواند مقدار مناسبی باشد. بااین حال برای یافتن مقدار دقیق خوشه‌ها با استفاده از معیار Silhouette امتیاز تمامی حالات بین ۲۵ تا ۵۰ خوشه را محاسبه کردیم. شکل ۱.۵ امتیازهای بدست آمده را برای هر دو مجموعه‌داده نشان می‌دهد. با توجه به مقادیر بدست آمده، مجموعه‌داده‌ی SQLi را به ۲۵ و مجموعه داده‌ی XSS را به ۴۶ خوشه تقسیم کردیم.

۵.۴.۵ حد آستانه‌ی آنتروپی

در آخرین مرحله از کاهش ویژگی نیاز است تا حد آستانه‌ای را برای مقدار آنتروپی انتخاب کنیم. شایان ذکر است که اگر مقدار حد آستانه بسیار کوچک باشد باعث می‌شود ویژگی‌های نامناسب بیشتری باقی بمانند و درصورتی که حد آستانه بسیار بزرگ باشد باعث از دست رفتن



XSS (ب) SQLi (آ)

شکل ۱.۵: امتیاز خوشه‌های مختلف هر مجموعه‌داده که با استفاده از روش Silhouette محاسبه شده است.

اطلاعات می‌شود. برای تعیین این حد آستانه از سعی و خطا استفاده کردیم و متوجه شدیم که حد آستانه‌ی ۰٫۰۰۵ مقدار مناسبی است.

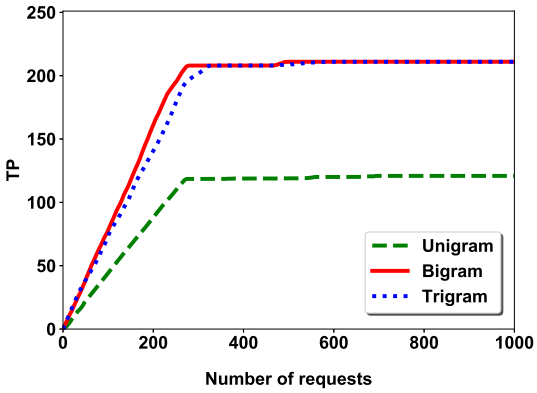
۶.۴.۵ پارامترهای ϵ-greedy

آخرین پارامترهای باقی‌مانده متعلق به سیاست ϵ-greedy هستند. در این بخش از روش پیشنهادی، مقدار پاداش را برابر ۱ قرار می‌دهیم و از آنجایی که تعداد حملات ناموفق بسیار بیشتر از حملات موفق است، مقدار تنبیه را برابر با ۰٫۵ درنظر می‌گیریم. حداکثر مقدار ϵ را برابر با ۰٫۹ درنظر می‌گیریم و مقدار k (رابطه‌ی (۳.۴)) را برابر با ۵×۱۰^{-۳} قرار می‌دهیم. وظیفه‌ی k کنترل میزان اکتشاف و بهره‌وری در سیاست ϵ-greedy است. هرچه مقدار k بیشتر باشد الگوریتم زمان بیشتری را صرف اکتشاف می‌کند. در این پژوهش، مقدار k نیز با سعی و خطا بدست آمده است.

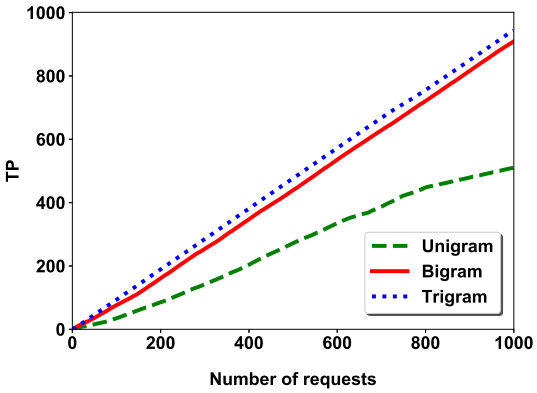
۵.۵ نتایج

در این بخش هر زیربخش مسئولیت پاسخ به یکی از پرسش‌های تحقیق را بر عهده دارد. این پاسخ‌ها با آزمایش‌های مختلف فراهم می‌شوند.

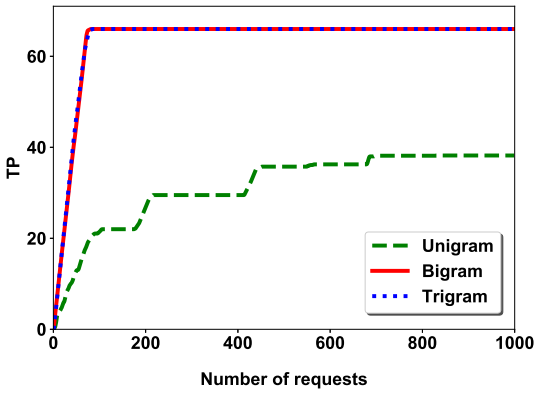
[۱] فایروال شخصی سازی شده



(آ) ModSecurity



(ب) NAXSI



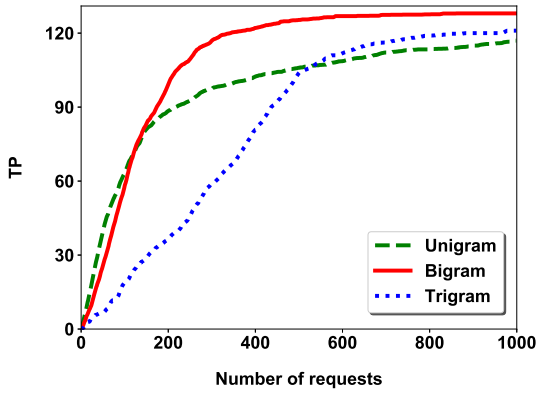
(ج) Custom-built WAF[1]

شکل ۲.۵: متوسط مثبت‌های درست در تست فایروال‌ها برای کشف آسیب‌پذیری‌های SQLi به ازای nهای مختلف.

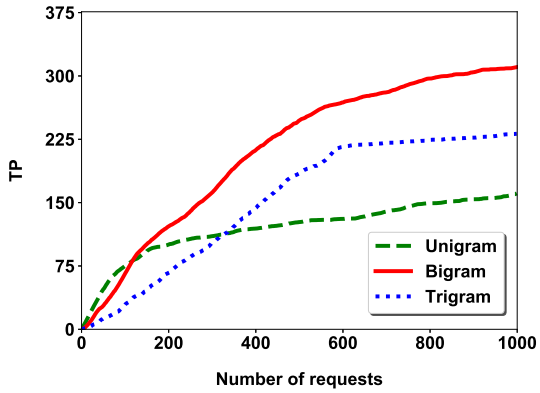
۱.۵.۵ بررسی n-gram

برای پاسخ به اولین پرسش، ابتدا ۴ زیرمجموعه از مجموعه‌ها که هر زیرمجموعه شامل ۲۵ هزار حمله است انتخاب شد. این زیرمجموعه‌ها به گونه‌ای انتخاب شدند که شامل حملات موفق باشند. سپس الگوریتم جستجوی تطبیقی را با استفاده از این مجموعه‌دادگان کوچک برای n های مساوی ۱، ۲ و ۳ اجرا کردیم. هر آزمایش ۱۰۰ بار تکرار شده است. سپس متوسط TP را برای یک هزار درخواست محاسبه کردیم.

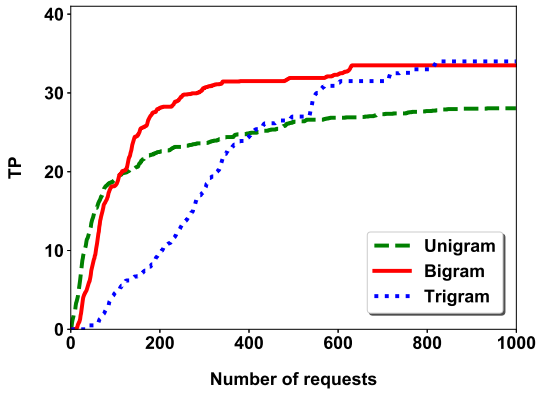
شکل ۲.۵ نتایج تست حملات SQLi و شکل ۳.۵ نتایج تست حملات XSS را برای هر سه فایروال نمایش می‌دهد. نتایج بدست آمده نشان می‌دهد که unigram احتمالاً به‌دلیل آنکه نمی‌تواند الگوهای پیچیده را استخراج کند، عملکرد ضعیفی دارد. Bigram و trigram هر دو توانایی استخراج الگوهای پیچیده را دارند. اما نتایج نشان می‌دهد که در مجموع عملکرد bigram کمی بهتر از trigram است. دلیل این تفاوت می‌تواند این باشد که اگرچه trigram می‌تواند الگوهای پیچیده‌تری را نسبت به bigram از حملات استخراج کند. اما تعداد



(آ) ModSecurity



(ج) Custom-built WAF



(ب) NAXSI

شکل ۳.۵: متوسط مثبت‌های درست در تست فایروال‌ها برای کشف آسیب‌پذیری‌های XSS به ازای nهای مختلف.

ویژگی‌هایی که با trigram حاصل می‌شود حتی پس از کاهش ویژگی بسیار بیشتر از bigram است، درنتیجه روش جستجوی تطبیقی ارائه شده در این پژوهش به مشاهدات بیشتری برای کشف آسیب‌پذیری‌ها نیاز خواهد داشت که این می‌تواند باعث کاهش عملکرد trigram شود. علاوه بر این افزایش مقدار n موجب عدم کشف ترکیب‌های موثر کوچکتر (مانند ترکیب‌های تکی یا دوتایی) می‌شود. بنابراین حالت ایده‌آل حالتی است که شامل انواع ترکیب‌های تکی، دوتایی و سه‌تایی باشد، که متاسفانه به‌دلیل محدودیت منابع، پردازش مجموعه‌ای شامل تمامی این حالات برای ما امکان‌پذیر نبود. همچنین، در آزمایش‌هایی که در این بخش انجام شد، مشاهده می‌شود که عملکرد bigram در آزمایش‌های مختلف با مجموعه‌دادگان متفاوت از ثبات بیشتری برخوردار است. بنابراین، در سایر آزمایش‌های این پژوهش از bigram برای استخراج زیررشته‌ها استفاده می‌کنیم.

۲.۵.۵ مطالعه‌ی تأثیر خوشه‌بندی

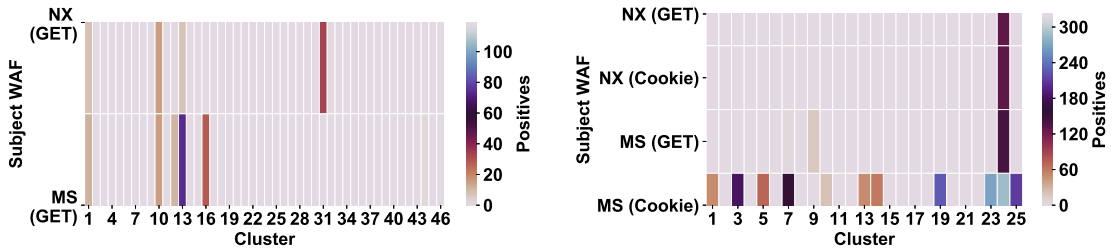
برای پاسخ به دومین پرسش تحقیق، تأثیر خوشه‌بندی را از دو جنبه بررسی می‌کنیم. جنبه‌ی اول تأثیر خوشه‌بندی بر کاهش ویژگی است. از آنجایی که کاهش ویژگی می‌تواند به کاهش تعداد مشاهدات کمک کند، بنابراین تعداد زیررشته‌ها را قبل و بعد از خوشه‌بندی محاسبه کردیم. نتایج در جدول ۵.۵ نشان داده شده است. همانطور که در جدول دیده می‌شود خوشه‌بندی به‌تنهایی و قبل از اعمال کاهش ویژگی توانست تعداد ویژگی‌ها را به‌ترتیب ۶/۴۸ و ۵۷/۲۶ درصد برای مجموعه‌دادگان SQLi و XSS کاهش دهد. پس از اعمال کاهش ویژگی، توانستیم در بدترین حالت به‌ترتیب ۳۱/۸۹ و ۹۸/۵۰ درصد ویژگی‌ها را برای مجموعه‌دادگان SQLi و XSS کاهش دهیم. همچنین کمترین نسبت اندازه‌ی خوشه به تعداد ویژگی اعضایش در ستون آخر جدول ۵.۵ نشان داده شده است.

جدول ۵.۵: مقایسه‌ی تعداد ویژگی‌ها قبل و بعد از خوشه‌بندی.

نوع حمله	قبل از خوشه‌بندی	پس از خوشه‌بندی				
	کل ویژگی‌ها	$T = ۰$		$T = ۰/۰۵$		تناسب
		بیشترین	متوسط	بیشترین	متوسط	
SQLi	۱۸۵	۱۷۳	۷۳	۱۲۶	۶۹	۲۴۷
XSS	۱۱۳۴۰۸	۴۸۴۷۰	۲۰۷۱۴	۱۷۰۲	۹۳۳	۳/۱۳

نتایج بدست آمده در شکل ۴.۵ نشان داده شده است. در این شکل، هر ستون نشان‌دهنده تعداد حملات موفق یک خوشه در تست فایروال‌های ModSecurity (MS) و NAXSI (NX) است. به‌عنوان مثال، در شکل ۴.۵ب، سی و یکمین خوشه حاوی بیش از ۱۰۰ حمله‌ی موفق در آزمایش پارامتر HTTP GET محافظت شده توسط ModSecurity است. همچنین، همان خوشه دارای حدود ۴۰ حمله‌ی موفق در تست همان پارامتر محافظت شده توسط NAXSI است.

با توجه به شکل ۴.۵، می‌توان مشاهده کرد که حملات موفق تنها در چند خوشه‌ی محدود توزیع می شوند. بنابراین، یافتن خوشه‌های مؤثر با استفاده از سیاست ϵ-greedy می تواند به‌طور قابل توجهی تلاش‌های ناموفق در کشف حملات موفق را با کنار گذاشتن خوشه‌های بی اثر کاهش دهد.



(آ) SQLi (ب) XSS

شکل ۴.۵: توزیع حملات موفق در خوشه‌های مجموعه‌دادگان.

۳.۵.۵ مقایسه‌ی روش پیشنهادی با سایر روش‌ها

برای پاسخ به پرسش سوم، ابتدا روش‌های ML-Driven E[۱]، ART4SQLi[۲] و XSSART[۳] را بر اساس مقالات آن‌ها پیاده سازی کردیم و سپس RAT را با روش‌های ML-Driven E، ART4SQLi و روش تصادفی با استفاده از مجموعه‌داده‌ی SQLi و XSSART با استفاده از مجموعه داده XSS مقایسه کردیم. آزمون‌های مقایسه‌ای با توجه به هدف هر روش طراحی شده‌اند. علاوه بر این، برای بررسی توانایی RAT در تست حملات تزریق کد دیگر، آن را با روش تصادفی با استفاده از مجموعه داده XSS مقایسه می‌کنیم.

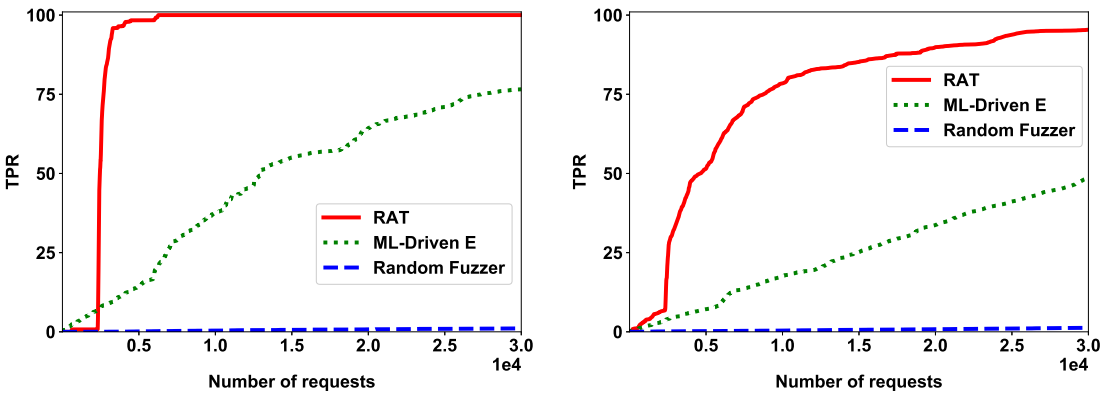
برای مقایسه‌ی RAT با ML-Driven E و روش تصادفی، از آنجا که هدف ML-Driven E کشف بالاترین تعداد آسیب‌پذیری SQLi است، مقدار TPR را برای ۳۰،۰۰۰ درخواست و برای هر روش اندازه گیری کردیم. سپس نتایج را ارزیابی می کنیم تا اثربخشی آنها را مقایسه کنیم.

هدف روش‌های ART4SQLi و XSSART کشف اولین حمله‌ی موفق با کمترین تعداد درخواست است. بنابراین، برای مقایسه RAT با این روش‌ها، تنها مقدار FP ها را اندازه گیری کردیم. برای این منظور در هر بار جستجو، یک خوشه تصادفی انتخاب کردیم و الگوریتم جستجوی تطبیقی را برای خوشه‌ی انتخاب شده برای یک دور اجرا کردیم.

در این بخش فقط به تست فایروال‌های متن‌باز می‌پردازیم. آزمایش‌های مقایسه‌ای بین RAT، ML-Driven E و روش تصادفی ۳۰ بار تکرار می شوند زیرا این آزمایش‌ها وقت گیر هستند و افزایش تعداد آزمایش‌ها با منابع محدود ما عملی نبود. درمقابل، آزمون‌های مقایسه‌ای بین RAT، ART4SQLi و XSSART وقت گیر نیستند. علاوه بر این، ART4SQLi و XSSART حداکثر ۲۷ درصد کارآمدتر از آزمایش تصادفی هستند. بنابراین، آزمایش‌های آن‌ها برای گزارش نتایج قابل اعتماد به تکرار بیشتر نیاز دارد. از این‌رو، آزمایش‌های مقایسه‌ای بین RAT، ART4SQLi و XSSART را ۱۰۰ بار تکرار کردیم.

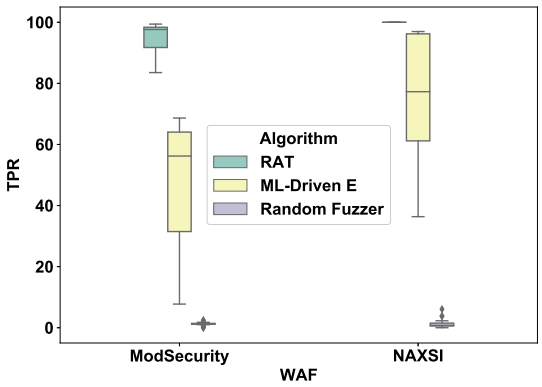
[1] https://github.com/mhamouei/ml-driven
[2] https://github.com/mhamouei/art4sqli
[3] https://github.com/mhamouei/xssart

برای مقایسه آماری نتایج بدست آمده، از آزمون Wilcoxon با سطح معنی‌دار[1] $\alpha = ۰/۰۵$ استفاده کردیم. از آنجا که این آزمون غیر پارامتری است، نیازی نیست که داده‌ها از توزیع نرمال پیروی کنند. برای انجام آزمایش Wilcoxon، همه‌ی نتایج آزمایش‌ها را برای تمام تکرارها جمع آوری کردیم و سپس آنها را بر اساس نوع حمله گروه بندی کردیم (به‌عنوان مثال، SQLi یا XSS). بنابراین، نتایج Wilcoxon برای هر نوع حمله به‌صورت جداگانه گزارش می‌شوند.



(ب) NAXSI (آ) ModSecurity

شکل ۵.۵: متوسط TPR بدست آمده برای هر روش در تست فایروال‌های متن‌باز برای کشف آسیب‌پذیری‌های SQLi.



شکل ۶.۵: نمودار جعبه‌ای TPR بدست آمده برای هر روش در تست فایروال‌های متن‌باز برای کشف آسیب‌پذیری‌های SQLi.

.شکل ۵.۵ نتیجه مقایسه بین RAT، ML-Driven E و روش تصادفی را به تصویر می کشد. شکل ۶.۵ نیز همین نتیجه را در قالب نمودار جعبه‌ای نشان می دهد. همانطور که در شکل ۵.۵ نشان داده شده است، در ۳۰،۰۰۰ درخواست، RAT می تواند به‌طور متوسط ۹۵/۳۷ و ۱۰۰ درصد آسیب‌پذیری‌ها را در تست فایروال‌های ModSecurity و NAXSI به‌دست آورد. علاوه بر

[1] Significance level

این، در بدترین حالت، RAT توانست ۸۳/۵۳ درصد از حملات موفق را پیدا کند (شکل ۶.۵). در مقایسه، ML-Driven E به‌طور متوسط توانست به ترتیب ۴۸/۵۱ و ۷۶/۵۶ از آسیب‌پذیری‌های ModSecurity و NAXSI را پیدا کند. مشاهدات نشان می دهد که RAT و ML-Driven E می توانند بسیاری از حملات موفق را پیدا کنند، در حالی که روش تصادفی به دلیل کم‌یاب بودن حملات موفق قادر به کشف آن‌ها نبود. علاوه بر این، نتایج به وضوح نشان می دهد که RAT دارای نرخ مثبت کاذب کمتری است و به‌طور قابل توجهی بهتر از همتایان خود عمل می کند.

جدول ۶.۵: نتایج آزمون Wilcoxon در تست فایروال‌ها برای کشف آسیب‌پذیری‌های SQLi.

تعداد درخواست‌ها	روش تصادفی	ML-Driven E ($p < ۰/۰۰۱$)	RAT ($p < ۰/۰۰۱$)
۱۰,۰۰۰	—	روش تصادفی	ML-Driven E روش تصادفی
۲۰,۰۰۰	—	روش تصادفی	ML-Driven E روش تصادفی
۳۰,۰۰۰	—	روش تصادفی	ML-Driven E روش تصادفی

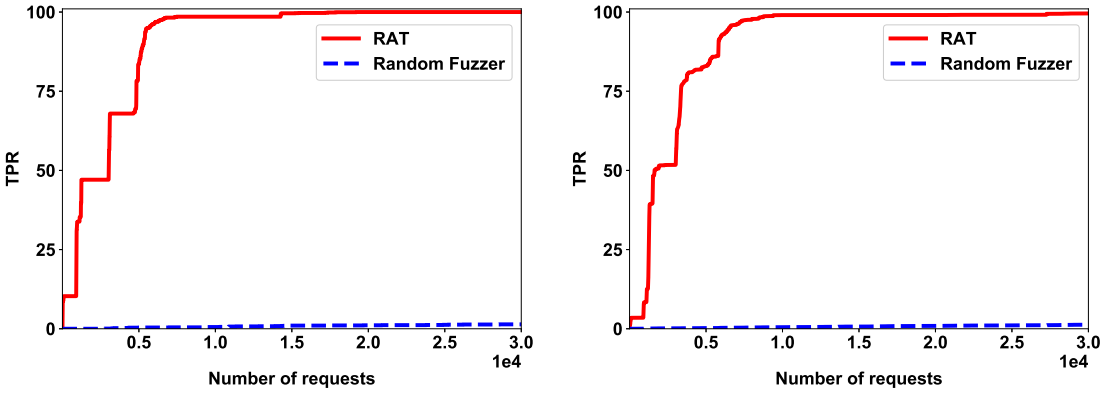
جدول ۶.۵ نشان می دهد که عملکرد هر یک از روش‌ها پس از هر ۱۰،۰۰۰ درخواست از عملکرد کدام روش‌ها بهتر است. برای ایجاد این جدول TPR را پس از هر ۱۰،۰۰۰ درخواست برای هر بار تکرار اندازه گیری کردیم و سپس از آزمون Wilcoxon برای مقایسه آماری روش‌ها استفاده کردیم. نتایج بدست آمده در جدول ۶.۵ نشان داده شده است. در این جدول هر سلول روش‌هایی را نشان می‌دهد که از روش عنوان آن ستون عملکرد ضعیف‌تری داشته‌اند.

جدول ۷.۵ نتیجه مقایسه بین RAT، ART4SQLi و روش تصادفی را نشان می دهد. مشاهدات نشان می دهد که RAT می تواند اولین حمله‌ی موفق را پس از تعداد قابل قبولی مثبت کاذب بیابد، در حالی که ART4SQLi قادر به پیدا کردن حمله‌ی موفق در تعداد محدود درخواست نبود بجز برای پارامتر کوکی ModSecurity که RAT توانست ۶۱/۴۳ درصد سریعتر از ART4SQLi اولین آسیب‌پذیری را بیابد. FPها را نیز در آزمایش پارامتر کوکی ModSecurity با استفاده از آزمون Wilcoxon مقایسه کردیم. درنتیجه، RAT توانست از ART4SQLi با $p < ۰/۰۰۱$ پیشی بگیرد.

سرانجام، برای ارزیابی عملکرد RAT در کشف سایر آسیب‌پذیری‌های تزریق کد، با استفاده

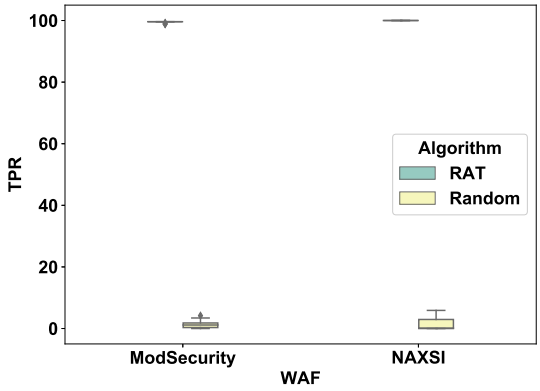
جدول ۷.۵: مقادیر FP قبل از یافتن اولین حمله‌ی موفق در تست فایروال‌ها برای کشف آسیب‌پذیری‌های SQLi.

روش	پارامتر GET		Cookie	
	ModSecurity	NAXSI	ModSecurity	NAXSI
RAT	۱۴۲/۴	۲۰۹/۳	۱۷۹/۳۳	۲۱۰/۷۸
ART4SQLi	—	—	۴۶۵/۰۲	—
روش تصادفی	—	—	—	—



(آ) ModSecurity (ب) NAXSI

شکل ۷.۵: TPR بدست آمده برای هر روش در تست فایروال‌های متن‌باز برای کشف آسیب‌پذیری‌های XSS.

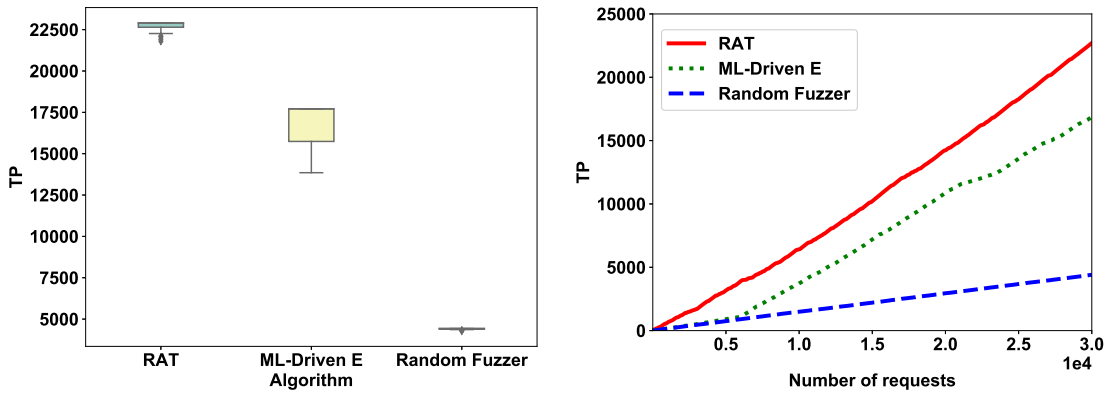


شکل ۸.۵: نمودار جعبه‌ای TPR بدست آمده برای هر روش در تست فایروال‌های متن‌باز برای کشف آسیب‌پذیری‌های XSS.

از مجموعه داده XSS فایروال‌های ModSecurity و NAXSI را تست کردیم. شکل ۷.۵ نتیجه‌ی تست ModSecurity و NAXSI را برای کشف آسیب‌پذیری‌های XSS با استفاده از RAT و روش تصادفی نشان می دهد و شکل ۸.۵ تغییر آماری نتایج آزمون را نشان می دهد. مطابق شکل ۷.۵، RAT توانست قبل از رسیدن به ۱۰ هزار درخواست، ۱۰۰ درصد آسیب‌پذیری‌ها را پیدا کند، در حالی که روش تصادفی توانست فقط یک درصد از آسیب‌پذیری‌ها را کشف کند. علاوه بر این، نتایج این دو روش را با استفاده از آزمون Wilcoxon نیز مقایسه کردیم. درنتیجه، تأیید شد که RAT می تواند به‌طور قابل توجهی ($p < ۰/۰۰۱$) بهتر از روش تصادفی عمل کند.

۴.۵.۵ ارزیابی کارایی روش‌پیشنهادی در عمل

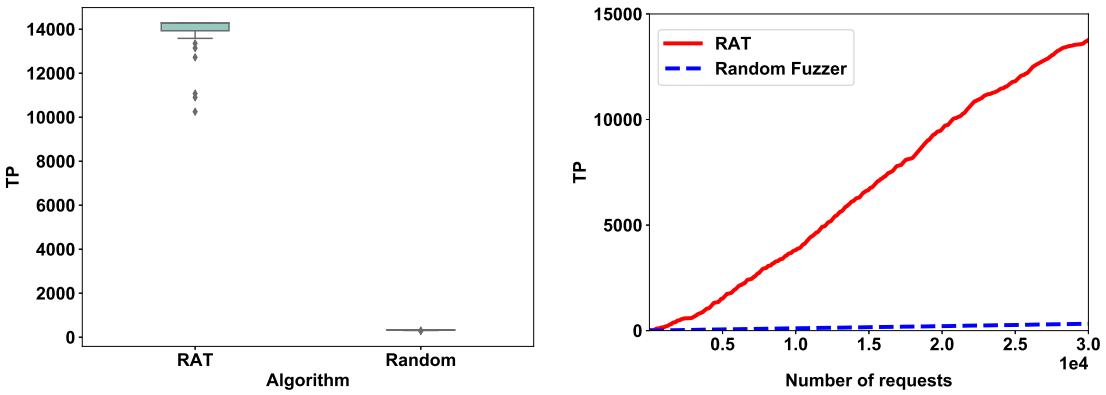
برای پاسخ به آخرین پرسش، RAT و همتایان آن را روی فایروال شخصی سازی شده تست کردیم. در این آزمایش درخواست‌ها از طریق اینترنت برای فایروال ارسال شدند. برای ارزیابی اثربخشی، در هر بار آزمایش ۳۰،۰۰۰ جستجو را با هر روش انجام دادیم و تعداد حملات موفق (TP) را اندازه‌گیری کردیم. از آنجا که برنامه مورد آزمایش در دنیای واقعی پیاده‌سازی شده است، حمله‌ی آزمون جامع غیرقابل اجرا بود، و ما نمی توانستیم تعداد کل حملات موفق را برای محاسبه TPR اندازه گیری کنیم. بنابراین در این بخش فقط TP را گزارش می دهیم.



(آ) نمودار خطی (ب) نمودار جعبه‌ای

شکل ۹.۵: متوسط آسیب‌پذیری‌های SQLi کشف شده با هر روش در تست فایروال شخصی‌سازی شده.

شکل ۹.۵آ نتایج اعمال سه روش RAT، ML-Driven E و روش تصادفی را روی فایروال شخصی‌سازی شده جهت کشف آسیب‌پذیری‌های SQLi نشان می دهد و شکل ۹.۵ب نیز همان نتایج را در قالب نمودارهای جعبه‌ای برای درک آماری بهتر نشان می‌دهد. نتایج بدست‌آمده نشان می دهد که RAT می تواند تعداد قابل توجهی از آسیب‌پذیری‌ها را با تعداد قابل قبولی درخواست کشف کند. به‌طور متوسط، RAT تعداد ۲/۸۳ درخواست را برای کشف هر حمله‌ی موفق ارسال کرد، در حالی که این تعداد برای روش ML-Driven E برابر با ۴/۰۶ درخواست و برای روش تصادفی این عدد ۱۳/۵۶ است.



(آ) نمودار خطی (ب) نمودار جعبه‌ای

شکل ۱۰.۵: متوسط آسیب‌پذیری‌های XSS کشف شده با هر روش در تست فایروال شخصی‌سازی شده.

برای ارزیابی کارایی RAT، میانگین TSR را برای RAT و ML-Driven E اندازه گیری کردیم. مقدار متوسط TSR برای RAT برابر با ۰/۷۴ ثانیه و برای ML-Driven E این مقدار ۰/۸۰ ثانیه بود. مقادیر TSR برای هر دو روش تقریباً یکسان بود و در عمل مقدار معقولی است. ما همچنین همین آزمایش را برای حمله XSS انجام دادیم. نتایج بدست آمده در شکل ۱۰.۵ نشان داده‌شده است که خلاصه‌ی آن به شرح زیر است:

- به ازای کشف هر آسیب‌پذیری RAT به‌طور متوسط ۴/۴۴ درخواست ارسال کرده است، در حالی که روش تصادفی ۱۸۴/۱۱ درخواست ارسال کرده است.
- میانگین مقدار TSR برای RAT در این آزمایش ۰/۳۹ ثانیه بود که به دلیل تعداد کمتر نمونه، از مقدار TSR تست آسیب‌پذیری TSR کمتر بود.

آزمایش Wilcoxon را برای مقایسه TPهای RAT با ML-Driven E و روش تصادفی نیز انجام دادیم. نتایج تست آسیب‌پذیری‌های SQLi همانند جدول ۶.۵ بود، همچنین نتایج بدست آمده از تست آسیب‌پذیری‌های XSS را نیز با استفاده از روش Wilcoxon ارزیابی کردیم که نشان می‌داد عملکرد RAT بسیار بهتر ($p < ۰/۰۰۱$) از روش تصادفی بود.

آزمایش‌های مقایسه‌ای را همچنین بین RAT، ART4SQLi و XSSART برای ارزیابی سرعت این روش‌ها در کشف اولین آسیب‌پذیری فایروال شخصی سازی شده انجام دادیم تا عملکرد این روش‌ها را در دنیای واقعی با یکدیگر مقایسه کنیم. نتایج در جداول ۸.۵ و ۹.۵ نشان داده شده است. مطابق جدول ۸.۵، روش تصادفی و ART4SQLi می‌توانند اولین آسیب‌پذیری را با تلاشی کمتر از RAT کشف کنند. نتایج بدست آمده از آزمون Wilcoxon نیز تأیید می کند که روش تصادفی و ART4SQLi می‌توانند از RAT بهتر عمل کنند. با این حال، نه ART4SQLi و نه روش تصادفی برتری‌ای نسبت به یکدیگر نداشتند. دلیل احتمالی نتایج به دست آمده می تواند تعداد زیاد آسیب پذیری‌های فایروال شخصی‌سازی‌شده در برابر حملات SQLi باشد.

جدول ۸.۵: نتیجه مقایسه بین سرعت RAT،ART4SQLi و روش تصادفی در تست فایروال شخصی‌سازی شده برای کشف آسیب‌پذیری‌های SQLi. در این جدول، میانگین، متوسط FPها قبل از کشف اولین آسیب‌پذیری است.

	روش تصادفی	ART4SQLi	RAT
میانگین	۴/۹۳	۴/۶۴	۷/۵۷
واریانس	۵/۹۶	۴/۶۳	۷/۲۱
Wilcoxon	RAT ($p < ۰/۰۰۱$)	RAT ($p < ۰/۰۱$)	—

جدول ۹.۵: نتیجه مقایسه بین سرعت RAT،XSSART و روش تصادفی در تست فایروال شخصی‌سازی شده برای کشف آسیب‌پذیری های XSS. در این جدول، میانگین، متوسط FPها قبل از کشف اولین آسیب‌پذیری است.

	روش تصادفی	XSSART	RAT ($p < ۰/۰۰۱$)
میانگین	۹۲/۹۵	۱۰۷/۶۱	۳۷/۷۷
واریانس	۹۲/۱۶	۱۱۶/۶۳	۱۵/۶۵
Wilcoxon	—	—	روش تصادفی XSSART

با این حال، جدول ۹.۵ نشان می دهد که RAT می تواند به‌طور قابل توجهی عملکرد بهتری نسبت به XSSART و روش تصادفی در کشف اولین آسیب‌پذیری XSS داشته باشد.

در نهایت، در پاسخ به این سؤال که آیا روش پیشنهادی در عمل هم از کارایی لازم برخوردار است یا خیر، باید بگوییم که بله. نتایج بدست آمده نشان می‌دهد که روش پیشنهادی را می‌توان در دنیای واقعی برای کشف آسیب‌پذیری فایروال‌ها تحت وب در بستر اینترنت استفاده کرد.

۶.۵ جمع‌بندی

دراین فصل، ابتدا پرسش‌های تحقیق مطرح شد و هدف هر پرسش شرح داده شد. پس از آن محیط آزمایش را معرفی کردیم. همچنین درمورد نحوه‌ی تعیین پارامترهای روش پیشنهادی به بحث پرداختیم. در نهایت، در آخرین بخش از این فصل به پاسخ به پرسش‌های تحقیق و مقایسه‌ی روش پیشنهادی با همتایان خود در قالب آزمایش‌های مختلف پرداختیم.

فصل ۶

نتیجه‌گیری و پیشنهادها

۱.۶ جمع‌بندی و نتیجه‌گیری

در جامعه‌ی مدرن امروزی، سهل‌انگاری در زمینه حفظ حریم خصوصی و امنیت در فضای سایبری پیامدهای مخرب و جبران‌ناپذیری را به‌دنبال دارد. بنابراین، فایروال‌های وب به‌عنوان یک راه حل مؤثر، نقش برجسته‌ای در محافظت از برنامه‌ها و سرویس‌های تحت وب دارند. از آنجا که وقوع حملات و همچنین پیچیدگی آن‌ها در حال افزایش است، فایروال‌ها باید مرتباً آزمایش و به‌روزرسانی شوند، زیرا در غیر این صورت، حملات ممکن است آن‌ها را دور زده و برنامه‌های تحت محافظت آن‌ها را تهدید کند.

هدف اصلی این پژوهش، ارائه‌ی یک روش تست جعبه سیاه برای کشف آسیب‌پذیری‌های فایروال‌های وب است. به‌طوری‌که بتواند با تعداد کمی جستجو در میان انبوه الگوهای حملات بیشترین تعداد آسیب‌پذیری‌ها را بیابد. بدین‌منظور روشی شامل سه فاز ارائه شد.

در فاز اول، این روش به استخراج ویژگی‌های اولیه می‌پردازد تا با استفاده از آن‌ها نمونه حملات مشابه را خوشه‌بندی کند. هدف از این کار، کاهش تعداد پارامترهای تست یا همان ویژگی‌ها و محدود کردن دامنه‌ی جستجو است. بنابراین، می‌توان تعداد جستجو را برای یافتن آسیب‌پذیری‌ها با استفاده از خوشه‌بندی تا حد قابل توجهی کاهش داد.

فاز دوم وظیفه‌ی کاهش ویژگی و پس از آن تشکیل بردار ویژگی‌های نهایی برای استفاده توسط جستجوگر را برعهده دارد. اگرچه با خوشه‌بندی داده‌ها تعداد ویژگی‌ها کاهش چشمگیری

پیدا می‌کند. اما همچنان ویژگی‌های بسیاری پس از فاز اول باقی می‌مانند که معایب حضورشان بیش از مزایای آن‌ها است. بنابراین، در این فاز آن ویژگی‌ها نیز حذف می‌شوند و داده‌ها آماده‌ی استفاده در جستجوگر می‌گردند.

در فاز آخر که جستجوگر نام دارد عملیات جستجو در دو سطح درون و برون خوشه‌ای انجام می‌شود. در سطح درون خوشه‌ای با به‌کارگیری یک روش جستجوی تطبیقی پیشنهادی به جستجو برای کشف حملات موفق درمیان حملات درون خوشه پرداخته می‌شود. اما از آنجایی که تمامی خوشه‌ها دارای حملات موفق نیستند، بنابراین، با استفاده از یادگیری تقویتی عملیات جستجوی برون خوشه‌ای به‌منظور کشف خوشه‌های مؤثر انجام می‌شود.

در این پژوهش تمامی ایده‌های طرح شده در بخش روش پیشنهادی توسط آزمایش‌های متنوع بررسی شدند. همچنین روش پیشنهادی با سه روش جدید مقایسه شد. نتایج آزمایش‌های مقایسه نشان داد که روش پیشنهادی نه تنها عملکرد بسیار بهتری نسبت به رقبای خود دارد، بلکه می‌توان به‌راحتی از آن در دنیای واقعی نیز استفاده کرد.

۲.۶ پیشنهاد برای کارهای آتی

همانطور که پیش‌تر بحث شد، تست فایروال‌ها جهت کشف آسیب‌پذیری‌های آن‌ها امری حیاتی است. اما همچنان اصلاح فایروال‌ها به‌صورت دستی امری دشوار است و نیاز است تا راهکاری ارائه شود که بتواند به‌صورت خودکار و با استفاده از آسیب‌پذیری‌های کشف شده تنظیمات فایروال‌ها را اصلاح کند.

اگرچه روش پیشنهادی عملکرد بسیار خوبی از خود نشان داد، اما عملکرد این روش و سایر روش‌هایی که تاکنون مطالعه کرده ایم بسیار وابسته به مجموعه‌داده یا دستورزبانی است که از آن‌ها استفاده می‌کنند. از این‌رو، عدم جامعیت و بروزرسانی مجموعه‌دادگان تأثیر قابل توجهی بر افت عملکرد این روش‌ها می‌گذارد. از طرفی، بدلیل عدم محدودیت اندازه‌ی طول رشته حملات، می‌توان به تعداد نامحدودی رشته حمله تولید کرد. همین دلیل موجب می‌شود که تهیه‌ی و بروزرسانی یک مجموعه‌داده جامع بسیار دشوار باشد. برای حل این مشکل می‌توان ابتدا روشی جهت استخراج آخرین نمونه حملات کشف شده از وبسایت‌ها، مقالات و گزارش‌ها ارائه داد. پس از آن، با استفاده از این داده‌ها و حملاتی که در تست‌های گذشته موفق بوده‌اند، می‌توان یک مدل مولد جهت تولید رشته حملات جدید و بروزرسانی خودکار مجموعه‌داده ارائه داد.

در نهایت، امتیاز silhouette بدست آمده برای خوشه‌ها، بیانگر این است که داده‌ها همچنان کامل از هم جدا نشده‌اند و ارائه‌ی راهکاری جهت جداسازی بهتر داده‌ها برای خوشه‌بندی آن‌ها ممکن است عملکرد روش پیشنهادی را باز هم بهبود بخشد.

مراجع

[1] Aliero, M. S., Shamaki, B. I., KALGO, B. S., Bello, A.-a. M., et al. (2020). Web application firewall. *International Journal Of All Research Writings*, 3(4):26–43.

[2] Appelt, D., Nguyen, C. D., and Briand, L. (2015). Behind an application firewall, are we safe from SQL injection attacks? In *2015 IEEE 8th international conference on software testing, verification and validation (ICST)*, pages 1–10. IEEE.

[3] Appelt, D., Nguyen, C. D., Panichella, A., and Briand, L. C. (2018). A machine-learning-driven evolutionary approach for testing web application firewalls. *IEEE Transactions on Reliability*, 67(3):733–757.

[4] Avancini, A. and Ceccato, M. (2011). Security testing of web applications: A search-based approach for cross-site scripting vulnerabilities. In *2011 IEEE 11th international working conference on source code analysis and manipulation*, pages 85–94. IEEE.

[5] Bau, J., Bursztein, E., Gupta, D., and Mitchell, J. (2010). State of the art: Automated black-box web application vulnerability testing. In *2010 IEEE symposium on security and privacy*, pages 332–345. IEEE.

[6] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

[7] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[8] Bozic, J., Garn, B., Kapsalis, I., Simos, D., Winkler, S., and Wotawa, F. (2015). Attack pattern-based combinatorial testing with constraints for web security testing. In *2015 IEEE International Conference on Software Quality, Reliability and Security*, pages 207–212. IEEE.

[9] Calzavara, S., Conti, M., Focardi, R., Rabitti, A., and Tolomei, G. (2020). Machine learning for web vulnerability detection: the case of cross-site request forgery. *IEEE Security & Privacy*, 18(3):8–16.

[10] Chandrasekar, K., Cleary, G., Cox, O., Lau, H., Nahorney, B., Gorman, B. O., O'Brien, D., Wallace, S., Wood, P., and Wueest, C. (2017). Internet security threat report (ISTR). Technical report, Symantec.

[11] Chen, D., Lv, J., and Zhang, Y. (2017a). Unsupervised multi-manifold clustering by learning deep representation. In *Workshops at the thirty-first AAAI conference on artificial intelligence*.

[12] Chen, K., Zhao, T., Yang, M., Liu, L., Tamura, A., Wang, R., Utiyama, M., and Sumita, E. (2017b). A neural approach to source dependence based context model for statistical machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(2):266–280.

[13] Chen, W., Zhang, M., and Zhang, Y. (2014). Distributed feature representations for dependency parsing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):451–460.

[14] Choi, S.-S., Cha, S.-H., and Tappert, C. C. (2010). A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48.

[15] Demetrio, L., Valenza, A., Costa, G., and Lagorio, G. (2020). Waf-a-mole: evading web application firewalls through adversarial machine learning. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pages 1745–1752.

[16] Doupé, A., Cova, M., and Vigna, G. (2010). Why johnny can't pentest: An analysis of black-box web vulnerability scanners. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 111–131. Springer.

[17] Duchene, F., Groz, R., Rawat, S., and Richier, J.-L. (2012). XSS vulnerability detection using model inference assisted evolutionary fuzzing. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, pages 815–817. IEEE.

[18] Elderman, R., Pater, L. J., Thie, A. S., Drugan, M. M., and Wiering, M. A. (2017). Adversarial reinforcement learning in a cyber security simulation. In *ICAART (2)*, pages 559–566.

[19] Felderer, M., Büchler, M., Johns, M., Brucker, A. D., Breu, R., and Pretschner, A. (2016). Security testing: A survey. In *Advances in Computers*, volume 101, pages 1–51. Elsevier.

[20] Ghasedi Dizaji, K., Herandi, A., Deng, C., Cai, W., and Huang, H. (2017). Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*, pages 5736–5745.

[21] Gladkova, A. and Drozd, A. (2016). Intrinsic evaluations of word embeddings: What can we do better? In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 36–42.

[22] Hamam, H. and Derhab, A. (2021). An owasp top ten driven survey on web application protection methods. In *Risks and Security of Internet and Systems: 15th International Conference, CRiSIS 2020, Paris, France, November 4-6, 2020, Revised Selected Papers*, volume 12528, page 235. Springer Nature.

[23] Hao, Y., Zhang, Y., Liu, K., He, S., Liu, Z., Wu, H., and Zhao, J. (2017). An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 221–231.

[24] Hellrich, J. and Hahn, U. (2017). Don't get fooled by word embeddings-better watch their neighborhood. In *Digital Humanities 2017–Conf. Abstracts of the 2017 Conf. of the Alliance of Digital Humanities Organizations (ADHO)*, pages 250—252.

[25] Hofmann, T., Schölkopf, B., and Smola, A. J. (2008). Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220.

[26] Huang, P., Huang, Y., Wang, W., and Wang, L. (2014). Deep embedding network for clustering. In *2014 22nd International conference on pattern recognition*, pages 1532–1537. IEEE.

[27] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

[28] Khan, M. E., Khan, F., et al. (2012). A comparative study of white box, black box and grey box testing techniques. *Int. J. Adv. Comput. Sci. Appl*, 3(6).

[29] Li, J. (2020). Vulnerabilities mapping based on OWASP-SANS: a survey for static application security testing (SAST). *Annals of Emerging Technologies in Computing (AETiC), Print ISSN*, pages 2516–0281.

[30] Li, J., Zhao, B., and Zhang, C. (2018). Fuzzing: a survey. *Cybersecurity*, 1(1):6.

[31] Li, Y.-F., Das, P. K., and Dowe, D. L. (2014). Two decades of web application testing—a survey of recent advances. *Information Systems*, 43:20–54.

[32] Lv, C., Zhang, L., Zeng, F., and Zhang, J. (2019). Adaptive random testing for XSS vulnerability. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pages 63–69. IEEE.

[33] Mann, H. B. and Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50 – 60.

[34] McGraw, G. (2004). Software security. *IEEE Security & Privacy*, 2(2):80–83.

[35] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[36] Min, R., Stanley, D. A., Yuan, Z., Bonner, A., and Zhang, Z. (2009). A deep non-linear feature mapping for large-margin knn classification. In *2009 Ninth IEEE International Conference on Data Mining*, pages 357–366. IEEE.

[37] Mnih, A. and Hinton, G. E. (2009). A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088. Citeseer.

[38] Muzaki, R. A., Briliyant, O. C., Hasditama, M. A., and Ritchi, H. (2020). Improving security of web-based application using modsecurity and reverse proxy in web application firewall. In *2020 International Workshop on Big Data and Information Security (IWBIS)*, pages 85–90. IEEE.

[39] Ouchi, H., Duh, K., Shindo, H., and Matsumoto, Y. (2016). Transition-based dependency parsing exploiting supertags. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2059–2068.

[40] Overflow, S. (2019). Stack overflow annual developer survey. [online] https://insights.stackoverflow.com/survey/2019.

[41] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[42] Phanidhar, G. and Venkateswarlu, S. (2021). A superstructural approach to avoid code injection attacks. *Annals of the Romanian Society for Cell Biology*, pages 19579–19587.

[43] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

[44] Schütze, H., Manning, C. D., and Raghavan, P. (2008). *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

[45] Shen, M., Kawahara, D., and Kurohashi, S. (2014). Dependency parse reranking with rich subtree features. *IEEE/ACM transactions on audio, speech, and language processing*, 22(7):1208–1218.

[46] Shen, Z. and Chen, S. (2020). A survey of automatic software vulnerability detection, program repair, and defect prediction techniques. *Security and Communication Networks*, 2020.

[47] Simos, D. E., Garn, B., Zivanovic, J., and Leithner, M. (2019a). Practical combinatorial testing for XSS detection using locally optimized attack models. In *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 122–130. IEEE.

[48] Simos, D. E., Kleine, K., Ghandehari, L. S. G., Garn, B., and Lei, Y. (2016). A combinatorial approach to analyzing cross-site scripting (XSS) vulnerabilities in web application security testing. In *IFIP International Conference on Testing Software and Systems*, pages 70–85. Springer.

[49] Simos, D. E., Zivanovic, J., and Leithner, M. (2019b). Automated combinatorial testing for detecting SQL vulnerabilities in web applications. In *2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)*, pages 55–61. IEEE.

[50] Singh, K., Devi, H. M., Mahanta, A. K., et al. (2017). Document representation techniques and their effect on the document clustering and classification: A review. *International Journal of Advanced Research in Computer Science*, 8(5).

[51] Thomé, J., Gorla, A., and Zeller, A. (2014). Search-based security testing of web applications. In *Proceedings of the 7th International Workshop on Search-Based Software Testing*, pages 5–14.

[52] Tripp, O., Weisman, O., and Guy, L. (2013). Finding your way in the testing jungle: a learning approach to web security testing. In *Proceedings of the 2013 International Symposium on Software Testing and Analysis*, pages 347–357.

[53] Wang, B., Wang, A., Chen, F., Wang, Y., and Kuo, C.-C. J. (2019). Evaluating word embedding models: Methods and experimental results. *APSIPA transactions on signal and information processing*, 8.

[54] Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

[55] Yaghoobzadeh, Y. and Schütze, H. (2016). Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 236–246.

[56] Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. (2017). Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870.

[57] Yu, L.-C., Wang, J., Lai, K. R., and Zhang, X. (2017). Refining word embeddings using intensity scores for sentiment analysis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(3):671–681.

[58] Zhang, B., Xiong, D., Su, J., and Duan, H. (2017). A context-aware recurrent encoder for neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(12):2424–2432.

[59] Zhang, L., Zhang, D., Wang, C., Zhao, J., and Zhang, Z. (2019). Art4sqli: The art of SQL injection vulnerability discovery. *IEEE Transactions on Reliability*, 68(4):1470–1489.

[60] Zhao, J., Dong, T., Cheng, Y., and Wang, Y. (2020). CMM: A combination-based mutation method for SQL injection. In *Structured Object-Oriented Formal Language and Method: 9th International Workshop, SOFL+ MSVL 2019, Shenzhen, China, November 5, 2019, Revised Selected Papers*, volume 12028, page 345. Springer Nature.

[61] Zhou, G., Xie, Z., He, T., Zhao, J., and Hu, X. T. (2016). Learning the multilingual translation representations for question retrieval in community question answering via non-negative matrix factorization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(7):1305–1314.

Aabstract

Web Application Firewalls (WAFs) are responsible for the safety of numerous web applications that are brutally under attack. Due to the increasing sophistication of web attacks, WAFs have to be tested and updated regularly to resist the relentless flow of web attacks. In practice, using a brute-force attack to discover vulnerabilities is infeasible due to the wide variety of attack patterns. Thus, various black-box testing techniques have been proposed to provide a practical solution. However, these techniques are not mature enough and suffer from low efficiency. In this paper, we present an automated black-box testing strategy to discover injection vulnerabilities in WAFs. In particular, we focus on SQL injection and Cross-site Scripting (XSS), which have been among the top ten vulnerabilities over the past decade.

In our proposed method, in the very first phase, an n-gram is used to decompose attack payloads into string fragments. Then, skip-gram is utilized to transform fragments into numerical vectors to cluster them in the next step using hierarchical clustering. In the last step, attack payloads are clustered using fragment clusters and autoencoder. In the second phase, first, non-informative fragments are removed in each cluster using entropy. Then, a weight is calculated for each fragment using Inverse-Document-Frequency (IDF) to form the final feature vectors. In the last phase, test oracle searches for bypassing payloads in two parallel steps: inter-cluster search and intra-cluster search. The inter-cluster search aims to discover clusters that include bypassing payloads using an e-greedy policy. Simultaneously, the intra-cluster search algorithm tries to discover bypassing payloads inside a selected cluster using Term-Frequency-Inverse-Document-Frequency (TF-IDF).

In this research, the proposed method is compared to three state-of-the-art methods, namely ML-Driven E, ART4SQLi, and XSSART, using two datasets containing 2417720 and 1798062 SQLi and XSS payloads, respectively. Results show that the proposed method discovers an average of 33.53% more vulnerabilities than ML-Driven E. Moreover, on average, the proposed method consumes 63.16% fewer HTTP requests than ART4SQLi and XSSART before discovering the first bypassing payload.

Keywords: Security testing, Intection attack, Adaptive testing, Web application firewall (WAF), Attack sample clustering, SQL injection

Faculty Of Computer Engineering

MSc Thesis in Artificial Intelligence Engineering

Intelligent Vulnerability Discovering for SQL Injection

By: Mohammadhossein Amouei

Supervisor:

Mohsen Rezvani

Advisor:

Mansoor Fateh

August 2021