The 9th International Conference on Web Research
ICWR2023
May 3th-4th, 2023, Tehran, Iran

Cross-Site Scripting Prevention Using Machine Learning Regarding the JavaScript Esoteric Paradigm

Ali Safari*[1], Farnoush Manavi[2], Ali Hamzeh[3]

[1] Master Student of Computer Engineering, Shiraz University, Shiraz, Iran
a.safari@cse.shirazu.ac.ir

[2]PhD Candidate of Computer Engineering, Shiraz University, Shiraz, Iran
f.manavi@cse.shirazu.ac.ir

[3]Professor of Computer Engineering, Shiraz University, Shiraz, Iran
ali@cse.shirazu.ac.ir

Abstract

In recent years, billions of users are exchanging information at an increasing rate in the vast and interconnected world of the Internet. On the other hand, attackers keep planning various threats in this environment. As a result of these issues, it has become more and more critical to prevent or reduce vulnerabilities. Cross Site Scripting is a well-known vulnerability on the web through which an intruder tries to steal users' vital information or induce malicious activities on behalf of the user. The weakness of the previous methods of detecting XSS attacks based on machine learning lies in not taking the possible changes in the characters of the attack vector into account, or in other words, special encodings, and this reduces the accuracy of these methods. The method presented in this article utilizes an algorithm that translates a kind of obfuscation in the attack vector that increases the accuracy of the detection model on the XSSED dataset to over 98%.

Keywords: Cross Site Scripting, JavaScript, Machine Learning, XSS

The 9th International Conference on Web Research
ICWR2023
May 3th-4th, 2023, Tehran, Iran

پیش‌گیری از XSS به وسیله‌ی یادگیری ماشین با توجه به ویژگی Esoteric زبان جاوااسکریپت

علی صفری*[1]، فرنوش معنوی[2]، علی حمزه[3]

[1] دانشجوی ارشد مهندسی کامپیوتر، دانشکده برق و کامپیوتر، دانشگاه شیراز، شیراز
a.safari@cse.shirazu.ac.ir

[2] دانشجوی دکتری مهندسی کامپیوتر، دانشکده برق و کامپیوتر، دانشگاه شیراز، شیراز
f.manavi@cse.shirazu.ac.ir

[3] استاد، دانشکده برق و کامپیوتر، دانشگاه شیراز، شیراز
ali@cse.shirazu.ac.ir

چکیده

امروزه، میلیاردها کاربر، در فضای گسترده و درهم‌تنیده اینترنت مشغول به رد و بدل اطلاعات با سرعتی بی‌سابقه و با نرخی افزایشی هستند. از طرف دیگر مهاجم‌ها نیز، در این فضا، دست به برنامه‌ریزی تهدیدهای گوناگون می‌زنند. این مسائل دست به دست یکدیگر داده‌اند که جلوگیری از این آسیب‌ها و یا کاهش آن‌ها، هر روز دارای اهمیت بیشتری شود.

یکی از پرتکرارترین حملات شناخته‌شده در سطح وب، حمله‌ی XSS است که مهاجم از طریق آن با تزریق اسکریپت‌های مخرب در وب‌سایت، تلاش به دزدیدن اطلاعات مهم کاربر می‌کند. ضعف روش‌های قبلی شناسایی حملات XSS مبتنی بر یادگیری ماشین، در تلاش آن‌ها برای تشخیص المان‌های مشکوک بدون توجه به تغییر شکل ممکن در کاراکترهای مؤلفه مخرب یا به بیان دیگر کدگذاری خاص است و این باعث کاهش دقت این روش‌ها می‌شود. روشی که در این مقاله ارائه شده است به وسیله‌ی یک الگوریتم که توانایی برگردان نوعی از مبهم‌سازی، یا در موضوع ما همان کدگذاری کمتر شناخته‌شده، در مؤلفه مخرب را دارد باعث افزایش دقت تشخیص شده و دقت مدل تشخیص بر روی دیتاست XSSED را به بالای ۹۸ درصد می‌رساند.

کلمات کلیدی

جاوا اسکریپت، لینک‌های مخرب، مرورگر، یادگیری ماشین، XSS(Cross-Site Scripting)

۱- مقدمه

حمله XSS به عنوان یک بردار حمله ناشی از اسکریپت‌های مخرب در سیستم کاربر یا سرور تعریف می‌شود که در آن داده‌های ورودی کاربر توسط برنامه نویسان وب‌سایت به درستی پاکسازی نشده‌است [1]. حمله XSS شامل انواع مختلفی است که عبارت اند از: XSS منعکس شده، XSS ذخیره شده و XSS کورکورانه. در XSS منعکس شده کد مخرب بدون دخالت مستقیم کاربر و بدون آگاهی او بر روی مرورگر اجرا می‌شود. این قطعه کد مخرب می‌تواند از طریق کلیک بر روی یک لینک اجرا شود. در XSS ذخیره شده مهاجم یک قطعه کد مخرب در وب‌سایت وارد می‌کند و این کد در سرور ذخیره شده و کاربران معمولی دیگر که درخواست اطلاعات از سرور می‌کنند، آن کد مخرب را نیز دریافت کرده و از این طریق اطلاعاتشان در معرض خطر قرار می‌گیرد. در حملات XSS کورکورانه فرد مهاجم به صورت مستقیم به نقطه آسیب‌پذیر دسترسی ندارد [2] بدون دانش از اینکه XSS در کدام قسمت و یا در چه زمانی اتفاق می‌افتد، کد مخرب را در ورودی‌های منجر به ذخیره در سرور وب‌سایت تزریق می‌کند.

زبان جاوا اسکریپت به عنوان یک زبان برنامه‌نویسی بسیار رایج سمت «کِلاینت» و همچنین اخیرا سمت «سِرور» یک زبان تابع‌گرا به همراه ویژگی‌های شیءگرایی است، اما ویژگی کمتر شناخته شده این زبان، پارادایمی به نام Esoteric است. هدف این پارادایم، امکان نوشتن برنامه با کاراکترهای غیرالفبا اما قابل‌تفسیر برای مفسر جاوا اسکریپت است. برای مثال می‌توان عبارت "alert" را به صورت کدگذاری شده همانند عبارت (۱) درآورد.

```
(![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]+(!![]+[])[+[]]
```
(۱)

برای ارزیابی این فرآیند، طبق شکل (۱) با وارد کردن عبارت (۱) به کنسول جاوا اسکریپت که به راحتی در مرورگر کروم قابل دسترسی است، عبارت "alert" دریافت می‌شود که نشان دهنده‌ی ویژگی esoteric حتی در جدیدترین نسخه مرورگر کروم(نسخه ۱۱۲) است.

این ویژگی جاوا اسکریپت می‌تواند برای اهداف XSS استفاده شود و مدل‌های جلوگیری از XSS را دور بزند. در روش‌های ارائه شده قبلی توجهی به این بُعد از مسئله نمی‌شود و نمی‌توان با این حمله مقابله کرد. در این مقاله، با الگوریتمی که عمل جایگذاری انجام می‌دهد لینک‌های مبهم‌سازی شده به حالت غیر مبهم برگردانده می‌شود و با استفاده از مدل یادگیری ماشین، حملات XSS تشخیص داده می‌شود.

ساختار این مقاله با ذکر کارهای پیشین در بخش ۲، شرح دقیق روش پیشنهادی در بخش ۳، ارزیابی و نتایج در بخش ۴ و در نهایت، در بخش ۵ نتیجه‌گیری ارائه شده است.

۲- مروری بر کارهای پیشین

در پژوهش‌های [3] و [4] روش‌هایی تحت عنوان جعبه سفید و فرآیندهای متوالی مرتبط[1] ارائه شده است که در آن تلاش شده است چارچوبی طراحی و پیاده سازی شود که طی آن نرم‌افزار تحت‌وب به صورت خودکار قبل از انتشار، بازنویسی شود و تمام منطق کد برنامه با داده‌ها، جدا شود. این روش البته غیر از عدم پشتیبانی از نرم‌افزارهای مختلف و تنها پشتیبانی از نرم‌افزارهای تحت‌وب ویندوز با ضعف‌هایی اعم از، نیاز به اصلاح و تغییر کد سمت سرور و افزایش بار محاسباتی و اعتماد کامل مرورگر به سرور که خود یک آسیب‌پذیری محسوب می‌شود، روبه رو است. بر اساس پژوهش [5] روشی در قالب افزونه مرورگر معرفی شده است که سعی در استفاده از ویژگی‌های ارتقاء یافته HTML 5 در ترکیب با CORS(Cross-origin Resource Sharing Policy) برای جلوگیری از حمله‌هایXSS، دارد. در این روش تمام درخواست‌های جاوا اسکریپتی به غیر از آن‌هایی که از سیاست منبع مشترک پیروی می‌کنند، نادیده گرفته می‌شوند. ضعف این روش، افزایش مدت زمان بارگذاری صفحه‌های وب است که در این روش نزدیک به دو برابر می‌شود.

یادگیری عمیق به منظور دفاع در برابر انواع بدافزار، می‌تواند ابزاری مناسبی باشد که رویکردی متفاوت با الگوریتم‌های سنتی را دربر می‌گیرد. در پژوهش [6] رویکردی با استفاده از یادگیری عمیق برای تشخیصXSS پیشنهاد شده است. این روش، ابتدا از یک ربات خزنده وب برای جمع‌آوری داده‌های مخرب و داده‌های بی‌خطر استفاده می‌کند، سپس از منبع داده XSSED [7] برای داده‌های مخرب و از منبع داده DMOZ برای جمع‌آوری داده‌های بی‌خطر استفاده می‌کند. در نهایت، داده‌ها را نرمال می‌کند تا اطلاعات بی‌فایده را حذف نماید و داده‌ها را با استفاده از عبارات منظم طراحی شده خودشان برچسب گذاری کند. سپس از word2vec، برای بدست آوردن ویژگی‌ها و ایجاد یک ارتباط بین هر قلم داده با هر بردار ویژگی

```
> (![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![]+[])[!+
  []+!+[]+!+[]]+(!![]+[])[+!+[]]+(!![]+[])[+[]]
<- 'alert'
>
```

شکل (۱): نمونه عبارت مبهم

استفاده می‌کند. سپس، نتایج مرحله قبل را به یک شبکه عصبی عمیق وارد می‌کند. این روش دارای نرخ دقت و نرخ یادآوری بالایی است که برای حل مسئله قابل قبول و قابل توجه است اما یک مشکل اساسی این روش اتکای زیاد بر مجموعه داده‌های ارائه شده به مدل است و در صورت آزمایش مدل با داده‌های دیده نشده با خطا روبرو می‌شود. بنابراین، رسیدن به مدل کارآمد دارای چالش‌های فراوانی است.

یکی دیگر از مسائلی که تحلیل‌گران امنیتی با آن روبرو هستند بحث مبهم‌سازی بدافزار است [8] که این امر، تاثیر تخریب حمله را بیشتر می‌کند؛ زیرا نه تنها این حملات ممکن است شناسایی نشده باقی بمانند بلکه این موضوع امکان افزایش مدت زمان آسیب را بیشتر می‌کند. بنابراین، برای جلوگیری کارآمد از حملات، مدلی لازم است که توانایی شناسایی کدهای مخرب جاوا اسکریپت و به طور موثر توانایی شناسایی کدهای مبهم را نیز داشته باشد. پژوهش [9] به نتایج بهتری نسبت به روش‌های پیشین رسیده است که از منبع داده XSSED استفاده کرده است و ویژگی‌های مورد نظر را از اطلاعات مختلف در مورد صفحه وب بر اساس تجزیه و تحلیل ویژگی‌های مبتنی بر اسکریپت مانند: اندازه اسکریپت، اندازه رشته کاراکترها، اشیاء خاص DOM به خصوص آن مواردی که قابلیت انتقال و ارسال داده را دارند، استخراج کرده است. در نهایت ویژگی‌های دیگر از جمله فراخوانی‌های توابع جاوا اسکریپت مانند: مدیریت رویدادها[2]، کلید واژه‌های حساس و غیره با هدف ساختن مدل یادگیری ماشین کارآمد استخراج شده است.

با مروری بر روش‌های پیشین، مشخص می‌شود که ضعف اکثر این روش‌ها عدم مقابله در برابر تکنیک‌های مبهم‌سازی است. بنابراین، در این مقاله در بخش ۳ روشی ارائه شده است که بر این مشکل تمرکز دارد و سبب شناسایی دقیق‌تر حملات XSS می‌شود [10].

۳- روش پیشنهادی

روش‌های قبلی شناسایی حملات XSS مبتنی بر یادگیری ماشین، تلاش برای تشخیص المان‌های مشکوک می‌کنند [11]، اما توجهی به تغییر شکل ممکن در حرف‌های مولفه مخرب ندارند و این باعث کاهش دقت این روش‌ها می‌شود. بنابراین، برای افزایش دقت در شناسایی حملات XSS در روش پیشنهادی این نکته مورد توجه قرار گرفته شده است. در روش پیشنهادی، ابتدا یک پیش پردازش، قبل از انجام فرآیند طبقه‌بندی لینک مورد نظر انجام می‌شود که از به اشتباه افتادن مدل یادگیری ماشین در موارد مبهم‌سازی شده جلوگیری می‌کند.

در این پژوهش فرآیند شبیه سازی ساخت لینک مخرب و مراحل مورد نظر روش پیشنهادی برای جلوگیری از آن در شکل (۳) نمایش داده شده است و جزییات الگوریتم ابداعی و پیش‌پردازش در ادامه بیان می‌شود.

در قسمت بالایی شکل (۳)، مراحل طی شده در طراحی حمله XSS توسط مهاجم بیان شده است. در این فاز، قسمت‌های اسکریپتی لینکی مخرب به الگوریتم مبهم‌ساز داده می‌شود و خروجی این الگوریتم، عبارتی مبهم است که توانایی اجرا بر روی مفسر مرورگر را دارد اما به صورت منفرد به ظاهر معنی مشخصی ندارد. درفاز بعدی قسمت اصلی روش پیشنهادی وجود دارد که سعی در مقابله با این روش حمله را دارد.

اگر مانند پژوهش‌های پیشین، مدل یادگیری ماشین با توجه به قسمت‌های اسکریپتی آموزش داده شود مدل در برابر لینک‌های مبهم شده کارایی خود را از دست خواهد داد. لذا در روش پیشنهادی، در مرحله شبیه سازی روش حمله نوین، از آنجایی که الگوریتم مبهم‌ساز را در اختیار داریم، ابتدا قسمت‌های اسکریپتی لینک‌هایی که در مجموعه داده XSSED [7] موجود بودند به صورت مبهم در آورده می‌شوند. سپس به وسیله یک الگوریتم مترجم ابداعی که از طریق یادگیری عملی و تجربه و خطا بدست آمده است، عبارت مبهم به حالت غیر مبهم در آورده می‌شود و در نهایت به مدل یادگیری ماشین برای طبقه‌بندی داده می‌شود.

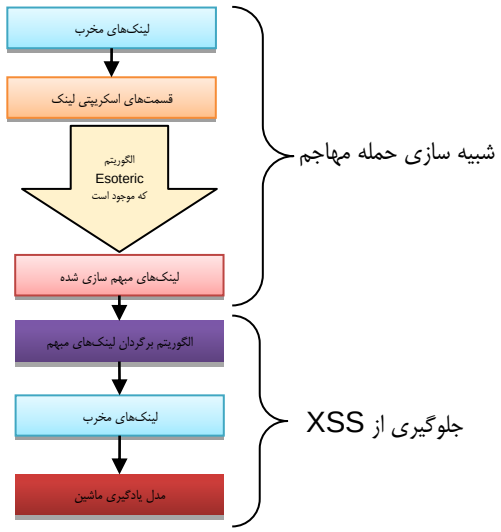
این الگوریتم به وسیله آزمون و خطا، با آزمون هر حرف لاتین در قالب کلمات مختلف و استنتاج یک رفتار تکراری از فرآیند مبهم‌سازی، طراحی شده‌است که بتواند عبارت متناظر هر حرف در هر کلمه را پیش‌بینی کند. به علاوه این الگوریتم، توانایی ترجمه قسمت‌های مبهم در لینک و نادیده گرفتن قسمت‌های دیگر را دارد. چالشی که در این راستا با آن رو به رو بوده‌ایم یک به یک نبودن تناظر حرف‌های یک کلمه یا یک عبارت در چارچوب کلمه یا عبارتی دیگر است. به بیان دیگر الگوریتم مبهم‌ساز به سادگی برگشت‌پذیر نیست و ممکن است کاراکتر a در ali به عبارت متفاوتی از همان کاراکتر در کلمه salam متناظر شود. با در اختیار داشتن الگوریتم مبهم‌ساز و آزمایش به وسیله‌ی طراحی ورودی‌های مناسب و بررسی خروجی‌ها، لغت‌نامه‌ای برای ترجمه عبارات مبهم ساخته شده است. هر بار اشتباهی توسط این الگوریتم صورت بگیرد لغت‌نامه‌ای جدید ساخته می‌شود که با این روش، اشتباهات احتمالی الگوریتم در چند مرحله مختلف به صورت خودکار تصحیح می‌گردد.

۱-۳- استخراج ویژگی

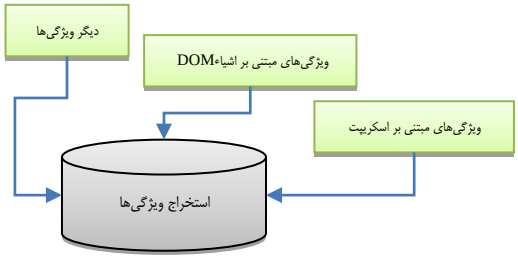
شکل (۴) منابعی که برای استخراج ویژگی‌ها استفاده شده است را نشان می‌دهد. منظور از این ویژگی‌ها کلیدواژه‌هایی است که وجود آن‌ها امکان خطر بالقوه حمله XSS را افزایش می‌دهد که عبارت اند از:

- ویژگی‌های مبتنی بر اسکریپت: محتوای اسکریپت غالبا به عنوان ظرفی برای رساندن محتوای مخرب به کاربر استفاده می‌شود که ممکن است با روش‌های مختلفی مبهم‌سازی شده باشد. بعضی از ویژگی‌هایی که یک اسکریپت مخرب را از یک اسکریپت معمولی متفاوت می‌کند عبارت است از: اندازه اسکریپت و اندازه ورودی‌های توابع اسکریپت. در روش پیشنهادی ۲۵ ویژگی از این دسته، استخراج شده است.
- ویژگی‌های مبتنی بر اشیاءDOM: برای مثال توابعی مانند document.URL، document.URLUnencoded، document.location، document.referrer و window.location نمایانگر حمله XSS احتمالی است، لذا از این توابع برای استخراج ویژگی در روش پیشنهادی استفاده شده است و از این دسته، حدودا ۷۰ ویژگی بدست آمده است.
- دیگر ویژگی‌ها: برای مثال توابعی مانند ESAPI.encoder، encodeForJavaScript و eventHandler می‌توانند نمایانگر حمله XSS احتمالی باشند، لذا از آنها نیز برای استخراج ویژگی استفاده شده است. در مراحل تصفیه و استخراج ویژگی‌ها از الگوریتم‌های انتخاب ویژگی CfsSubsetEval و Ranker استفاده شده است ویژگی‌هایی که دارای بیشترین تاثیر در شناسایی XSS هستند به عنوان ویژگی نهایی برای آموزش مدل یادگیری ماشین انتخاب شده‌اند.

شکل (۲) :دیاگرام طراحیXSS توسط مهاجم



شکل (۳) :دیاگرام روش پیشنهادی



شکل (٤): منابع مورد نظر برای استخراج ویژگی‌ها

۴- آموزش مدل

در این مرحله با هدف آموزش مدل، ویژگی‌های بدست آمده را به مدل درخت تصمیم‌گیری می‌دهیم. دلیل انتخاب درخت تصمیم‌گیری به عنوان مدل طبقه‌بندی مورد نظر برای این مسئله به غیر از آموزش دیدن سریع و قابلیت طبقه‌بندی سریع با توجه به مجموعه داده‌های محدود، قابل درک و قابل تحقیق بودن چرایی نتیجه طبقه‌بندی یا به عبارت دیگر همان تفسیرپذیر بودن است [12].

برای آموزش و ارزیابی مدل مورد نظر از پایتون نسخه ۳.۵ و از کتابخانه sklean استفاده شده است که امکانات بسیاری برای امور یادگیری ماشین دارد. برای ایجاد مدل از DecisionTreeClassifier از پکیج sklearn.tree استفاده شده است و پارامترهای پیش‌فرض این کتابخانه برای آموزش مدل درخت تصمیم‌گیری درنظر گرفته شده است.

۴-۱- ارزیابی مدل

بعد از آموزش مدل، با توجه به شکل ۳ برای هر رکورد داده به منظور طبقه‌بندی، ابتدا داده مورد نظر که در این حوزه یک لینک محسوب می‌شود، در الگوریتم برگردان مبهم‌ساز به حالت یک لینک با کاراکترهای متعارف و معمولی تبدیل می‌شود. سپس این لینک به مدل طبقه‌بندی کننده وارد شده و برچسب آن مشخص می‌شود.

۵- نتایج

در این بخش ابتدا به بیان مشخصات مجموعه داده‌ای که به وسیله‌ی آن روش پیشنهادی بررسی شده است، می‌پردازیم. سپس معیارهای ارزیابی مورد استفاده در این مقاله را شرح می‌دهیم و در آخر نتایج حاصل از ارزیابی روش پیشنهادی را شرح خواهیم داد.

۵-۱- مجموعه داده و جمع آوری اطلاعات

در این قسمت مجموعه داده‌ی استفاده شده در این تحقیق توضیح داده می‌شود. مجموعه داده‌ی مورد نظر از وب‌سایت (www.xssed.com) [13] استخراج شده است که برای یادگیری و سنجش مدل طبقه‌بندی‌کننده از آن استفاده می‌شود. در این وب‌سایت، لیستی از لینک‌ها که هر کدام با id مشخص به سمت یک صفحه وب ایستا هدایت می‌شوند قرار دارند. به وسیله‌ی یک ربات خزنده متناسب با این وب‌سایت لینک‌های مورد نظر از تاریخ ۲۰۰۹ تا ۲۰۲۲، برای آموزش و آزمایش مدل انتخاب و استخراج شده است. هر نمونه از این مجموعه داده یک لینک است که به صورت یک بردار نمایش داده شده است. هر بردار از این مجموعه داده شامل ویژگی‌هایی با مقادیر عدد (برای مثال تعداد کاراکترها) و بولین (وجود یک عبارت مشخص در لینک) می‌باشد. هر نمونه داده در این مجموعه داده شامل یک برچسب است که مخرب یا غیر مخرب بودن لینک را مشخص می‌کند. لازم به ذکر است بیان شود، در فرآیند استفاده از این مجموعه داده با چالش‌هایی مانند: حذف لینک‌های تکراری، حذف ناهنجاری‌های متنی مانند استفاده غیر منتظره از حروف بزرگ و کوچک مانند عبارت (۲) که باید به عبارت (۳) تبدیل شود و غیره روبرو بوده‌ایم. برای مثال، می‌توان به عبارت (۴) به عنوان یک نمونه از این قبیل چالش‌ها و ناهنجاری‌ها اشاره کرد که دارای قسمت‌هایی بی‌معنی نسبت به کاربری ما هستند و تاثیری در روش پیشنهادی ندارند که باید حذف شوند.

web.site/?user=John <script> <sCriPt>
(۴) (۳) (۲)

۵-۲- معیار ارزیابی

در فرآیند ارزیابی مدل یادگیری ماشین، از معیار precision و kappa statistic که در این حوزه رایج هستند استفاده شده است. به صورت کلی معیار precision با توجه به رکوردهای طبقه‌بندی‌شده به عنوان positive کارآمدی مدل را می‌سنجد. در عبارت (۵) نحوه محاسبه این معیار نمایان شده است.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (۵)$$

برای سنجش توانایی پیش‌بینی مدل، معیار precision نسبت به معیار accuracy ارجع است. دلیل آن این است که ممکن است در یک سنجش، نرخ دقت مدل نزدیک به ۱۰۰٪ باشد اما دلیل آن صرفا غالب بودن تعداد رکوردهای داده از یک کلاس خاص باشد و معیار سنجش به اشتباه و بدون توجه به این مسئله، دقت بالای مدل را نشان دهد.

معیار بعدی که برای ارزیابی روش پیشنهادی استفاده شده kappastatistic است. این معیار با مقایسه‌ی احتمال مشاهده و امید ریاضی، همچنین مانند معیار precision سعی در نمایش دقت مدل به دور از مشکلات رایج accuracy دارد. در عبارت (۶) نحوه محاسبه این معیار آمده است.

$$Kappa = \frac{P_{observed} - P_{expected}}{1 - P_{expected}} \quad (۶)$$

به منظور بررسی مدل تحت عواملی مانند بیش‌برازش از اعتبارسنجی متقابل ۱۰–دسته‌ای استفاده شده است. در این روش برای اعتبارسنجی کارایی مدل، مجموعه داده به ۱۰ زیرمجموعه مساوی تقسیم می‌شود که اصطلاحا به آن‌ها دسته (فلد) گفته می‌شود. در هر بار یکی از این زیرمجموعه‌ها به عنوان مجموعه داده آزمایشی استفاده می‌شود و بقیه زیرمجموعه‌ها برای آموزش مدل به کار گرفته می‌شود و میانگین نتایج بدست‌آمده برای ارزیابی روش استفاده می‌شود. در نتیجه، بر اساس میانگین اعتبارسنجی ۱۰–دسته، ارزیابی مدل درخت تصمیم انجام شده است و نتایج جدول ۱ بر این اساس بدست آمده است.

۵-۳- ارزیابی روش مقابله از طریق طراحی یک آزمایش

پارادایم Esoteric جاوا اسکریپت [14] قسمت‌های اسکریپتی این مجموعه داده بازنویسی شده است. با تغییر مجموعه داده، مدل‌های قبلی [9] که با دقت ۹۹ درصد XSS را شناسایی می‌کردند، با کاهش دقت روبه رو شده‌اند و بدیهی است که به خوبی توانایی شناسایی این مدل از تهدیدات را نداشته باشند. الگوریتم پیشنهادی این مقاله، برای برگردان لینک‌های مبهم با توجه به آزمون و خطاهای مداوم با چالش‌هایی از جمله تغییر عبارت متناظر هر حرف با توجه به ترتیب قرارگیری آن و مخصوصا تغییر عبارت با توجه به حروف دیگر موجود در جمله، در ابتدا در مواردی به اشتباه دچار می‌شد، که به خاطر ماهیت تکراری الگوریتم این اشتباهات قابل پیش‌بینی بود و با آن مقابله شد و در نهایت الگوریتم به حالتی که همیشه جواب صحیح دهد ارتقاء داده شد و کارآمدی مدل به کارایی قابل قبول رسانده شد. در نهایت برای ارزیابی درخت تصمیم‌گیری تولید شده از اعتبارسنجی متقابل ۱۰–دسته‌ای استفاده شده است.

۵-۴- ارزیابی نهایی

به منظور مقایسه عادلانه‌ی روش پیشنهادی و روش پیشین [9]، لینک‌های مبهم شده را به هر دو مدل آموزش داده شده، وارد می‌کنیم و مدل‌ها را ارزیابی می‌نماییم. نتیجه این مقایسه در جدول (۱) بیان شده است. دقت بالاتر مدل پیشنهادی در این آزمایش کنترل شده، در مقایسه با کارهای قبلی محرز است. اضافه شدن مرحله برگردان لینک‌های مبهم در روش پیشنهادی سبب دقیق‌تر شدن روش تشخیص شده است و این مرحله نوآوری روش پیشنهادی است.

۶- نتیجه گیری

پیدا کردن قسمت‌های آسیب‌پذیر و یا تهدیدآمیز برای امنیت اطلاعات کاربر در وب‌سایت‌ها امری بسیار پیچیده است که نیازمند دانش بسیار، در زمان طراحی و تولید نرم‌افزار وب و در فاز نگهداری از آن است. در این مقاله تمرکز بر روی یافتن و جلوگیری از وقوع موارد XSS در صفحات وب از طریق استخراج ویژگی‌های مناسب است. این مقاله با استفاده از درخت تصمیم‌گیری و ویژگی‌های مناسب استخراج شده، روشی با کارایی قابل‌قبول برای جلوگیری از حملات XSS ارائه کرده است.

جدول (۱) : نتایج

روش	Precision	Kappa Statistics
مدل پیشین[7]دربرابر حمله‌های جدید	۹۶.۹٪	۹۳.۷۱٪
روش پیشنهادی این تحقیق	۹۸.۹٪	۹۸٪

مراجع

[1] Vernotte, A., Dadeau, F., Lebeau, F., Legeard, B., Peureux, F., & Piat, F. (2014). Efficient Detection of Multi-step Cross-Site Scripting Vulnerabilities. In Information Systems Security (pp. 358–377). Springer International Publishing. https://doi.org/10.

[2] Mattia Fazzini, Prateek Saxena, and Alessandro Orso. 2015. AutoCSP: automatically retrofitting CSP to web applications. In Proceedings of the 37th International Conference on Software Engineering - Volume 1 (ICSE '15). IEEE Press, 336–346.

[3] Wang, C.-H., & Zhou, Y.-S. (2016). A New Cross-Site Scripting Detection Mechanism Integrated with HTML5 and CORS Properties by Using Browser Extensions. In 2016 International Computer Symposium (ICS)..

[4] Mokbal, F. M. M., Dan, W., Xiaoxi, W., Wenbin, Z., & Lihua, F. (2021). XGBXSS: An Extreme Gradient Boosting Detection Framework for Cross-Site Scripting Attacks Based on Hybrid Feature Selection Approach and Parameters Optimization. In Journal of Informat.

[5] Song, X., Chen, C., Cui, B., & Fu, J. (2020). Malicious JavaScript Detection Based on Bidirectional LSTM Model. In Applied Sciences (Vol. 10, Issue 10, p. 3440). MDPI AG. https://doi.org/10.3390/app10103440.

[6] Sarmah, U., Bhattacharyya, D. K., & Kalita, J. K. (2020). XSSD: A Cross-site Scripting Attack Dataset and its Evaluation. In 2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP). 2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP).

[7] Buyukkayhan, A.S. et al. (1970) What's in an exploit? an empirical analysis of reflected server {XSS} exploitation techniques, USENIX. Available at: https://www.usenix.org/conference/raid2020/presentation/buyukkayhan (Accessed: April 19, 2023).

[8] R. Banerjee, A. Baksi, N. Singh and S. K. Bishnu, "Detection of XSS in web applications using Machine Learning Classifiers," 2020 4th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 2020, pp. 1-5, doi: 10.1109/IEMENTech51367.2020.9270052.

[9] Buyukkayhan, A.S. et al. (1970) What's in an exploit? an empirical analysis of reflected server {XSS} exploitation techniques, USENIX. Available at: https://www.usenix.org/conference/raid2020/presentation/buyukkayhan (Accessed: April 19, 2023).

[10] I. Kareem Thajeel, K. Samsudin, S. Jahari Hashim, and F. Hashim, “Dynamic feature selection model for adaptive cross site scripting attack detection using developed multi-agent deep Q learning model,” Journal of King Saud University - Computer and Information Sciences. Elsevier BV, Jan. 2023. doi: 10.1016/j.jksuci.2023.01.012.

[11] J. Kaur, U. Garg, and G. Bathla, “Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review,” Artificial Intelligence Review. Springer Science and Business Media LLC, Mar. 23, 2023. doi: 10.1007/s10462-023-10433-3.

[12] C. Li, Y. Wang, C. Miao, and C. Huang, “Cross-Site Scripting Guardian: A Static XSS Detector Based on Data Stream Input-Output Association Mining,” Applied Sciences, vol. 10, no. 14. MDPI AG, p. 4740, Jul. 09, 2020. doi: 10.3390/app10144740.

[13] Z. Liu, Y. Fang, C. Huang, and Y. Xu, “MFXSS: An effective XSS vulnerability detection method in JavaScript based on multi-feature model,” Computers & Security, vol. 124. Elsevier BV, p. 103015, Jan. 2023. doi: 10.1016/j.cose.2022.103015..

[14] L. L. Custode and G. Iacca, "Evolutionary Learning of Interpretable Decision Trees," in IEEE Access, vol. 11, pp. 6169-6184, 2023, doi: 10.1109/ACCESS.2023.3236260.

زیرنویس ها

[1]Communicating Sequential Processes (CSP)

[2]Event Handlers