Journal Of Electronical & Cyber Defence
Vol. 2, No. 4, 2015, Serial No. 8

A new approach for static detection of security vulnerabilities in web applications

M. Sadeghi Yakhdani[1*], M. Agha Sarram[2], F. Adibnia[3]

1- Master Of Science, Yazd University, 2- Assistant Professor, Yazd University

3- Assistant Professor, Yazd University
(Received: 05/08/2014, Accepted: 11/05/2015)

ABSTRACT

Nowadays, due to the increased use of web-based applications and storage and exchange of sensitive inforamtion by this category of programs, it is necessary to detect security vulnerabilities and remove them to keep them secure against the misuse of intrusions. In most cases, the Static Analysis is especially valuable in security assurance and detection of security vulnerabilities, while dynamic analysis goal is finding and debugging the errors. In this paper, we present a new approach that detects common vulnerabilities in web applications by Probable Data Flow Analysis on Vulnerability Probability Graph. VPG is designed to consider the points with more probable to vulnerability and PDF Analysis is designed for the increase of accuracy in vulnerability detection. The proposed approach was tested on a few web applications and the results were compared with a few other tools that we observed improvement in performance in some cases.

Key words: Vulnerabilities Static detection, Web Applications, Vulnerability Probability Graph, Probable Data Flow Analysis.

* Corresponding Author Email: masadeghister@gmail.com

مجله علمی- پژوهشی "پدافند الکترونیکی و سایبری"
سال دوم، شماره ۴، زمستان ۱۳۹۳، ص ۷۴-۶۵

روشی جدید برای تشخیص ایستای آسیب پذیری‌های امنیتی در برنامه‌های کاربردی تحت وب

مائده صادقی یخدانی[1*]، مهدی آقا صرام[2] و فضل‌الله ادیب‌نیا[3]

۱- کارشناس ارشد مهندسی فناوری اطلاعات، دانشکده مهندسی برق و کامپیوتر، دانشگاه یزد

۲- استادیار، دانشکده مهندسی برق و کامپیوتر، دانشگاه یزد

۳- استادیار، دانشکده مهندسی برق و کامپیوتر، دانشگاه یزد

(دریافت: ۹۳/۵/۱٤؛ پذیرش: ۹٤/۲/۲۱)

چکیده

امروزه به‌دلیل افزایش استفاده از برنامه‌های کاربردی تحت وب و ذخیره و تبادل‌اطلاعات حساس و مهم توسط این دسته از برنامه‌ها، بررسی و رفع آسیب‌پذیری‌های امنیتی آن‌ها به جهت تامین امنیت در برابر سوء استفاده نفوذگران دارای اهمیت بالایی می‌باشد. تحلیل ایستا در بیشتر موارد به‌منظور تضمین امنیت و تشخیص آسیب‌پذیری‌های امنیتی مورد استفاده قرار می‌گیرد؛ در حالی‌که هدف اصلی تحلیل پویا، تشخیص و اشکال‌زدایی خطاهاست. در این مقاله یک روش نوین ایستا ارائه می‌گردد که در آن با استفاده از تحلیل جریان داده احتمالی بر روی گراف احتمال آسیب‌پذیری، آسیب‌پذیری‌های رایج در برنامه‌های کاربردی تحت وب شناسایی می‌شوند. گراف احتمال آسیب‌پذیری برای بررسی مسیرهایی با احتمال آسیب‌پذیری بیش‌تر و تحلیل جریان داده احتمالی برای افزایش دقت تشخیص آسیب‌پذیری طراحی شده‌اند. نتایج آزمایش‌ها بر روی چند برنامه کاربردی تحت وب و مقایسه نتیجه روش پیشنهادی با روش‌های دیگر، نشان می‌دهد الگوریتم ارائه‌شده در بهبود تشخیص آسیب‌پذیری‌ها، عملکرد قابل قبولی دارد.

واژه‌های کلیدی: تشخیص ایستای آسیب‌پذیری‌ها، برنامه‌های کاربردی تحت وب، گراف احتمال آسیب‌پذیری، تحلیل جریان داده احتمالی

۱- مقدمه

امروزه برنامه‌های کاربردی وب به‌عنوان واسطی برای بسیاری از سیستم‌های تحت وب، مورد استفاده قرار می‌گیرد. با توجه به حساسیت اطلاعاتی که این دسته از برنامه‌ها ذخیره و تبادل می‌کنند، لزوم امن بودن آن‌ها حائز اهمیت است. نفوذگر با سوء استفاده از نقاط ضعف برنامه‌های کاربردی تحت وب، امنیت سیستم را به خطر می‌اندازد. بنابراین، تشخیص آسیب‌پذیری‌های برنامه‌های کاربردی تحت وب به میزان زیادی از حملات نفوذگران جلوگیری می‌کند.

از برجسته‌ترین و رایج‌ترین آسیب‌پذیری‌های مشاهده شده در برنامه‌های کاربردی تحت وب، تزریق SQL و XSS است. در رتبه‌بندی آسیب‌پذیری‌ها تا ابتدای سال ۲۰۱۳، دو مورد فوق به ترتیب رتبه‌های اول و سوم را به خود اختصاص داده‌اند [۱۱].

روش‌های متعددی جهت تشخیص آسیب‌پذیری‌ها معرفی شده‌اند که هر روش دارای مزایا و معایب خاص خود می‌باشد. به‌عنوان مثال، روش‌هایی که متن برنامه را مورد بررسی قرار می‌دهند از تحلیل ایستا، پویا و یا ترکیبی از هر دو استفاده می‌کنند. در روش ایستا تمامی متن برنامه تحلیل می‌شود و بنابراین، بخش‌های بیشتری از برنامه تحت پوشش قرار می‌گیرد. از آن‌جا که این تحلیل بدون اجرا شدن واقعی برنامه تحت وب انجام می‌گیرد، سربار زمان اجرا وجود ندارد اما همواره نتایج شامل مثبت کاذب است و هیچ‌گاه نتایج واقعی حاصل نمی‌شود. در [۳, ۴, ۹, ۱۲] از تحلیل ایستا جهت تشخیص آسیب‌پذیری‌ها استفاده شده است.

ابزار Pixy [۳] با استفاده از گراف جریان کنترل (CFG)، آسیب‌پذیری‌هایی همچون تزریق SQL و XSS را به روش ایستا تشخیص می‌دهد. این ابزار با تحلیل جریان داده، انتشار آلودگی را در متن برنامه بررسی می‌کند و درصورتی که داده آلوده از یک منبع ورودی ناامن به یک حفره برسد، آسیب‌پذیری تشخیص داده

* رایانامه نویسنده مسئول: masadeghister@gmail.com

می‌شود. مزیت اصلی Pixy مربوط به تحلیل ایستای آن است که ابتدا صحیح بودن سینتکس برنامه بررسی می‌شود و در صورت صحیح بودن آن، درخت تجزیه از کد منبع ایجاد می‌شود تا تحلیل‌های بعدی ساده‌تر گردد. از معایب اصلی Pixy این است که تنها کد نوشته‌شده برای PHP نسخه ۴ را پشتیبانی می‌کند و ویژگی‌های زبان جدید PHP5 شامل مدل شیء طراحی‌شده جدید آن پشتیبانی نمی‌شوند. همچنین می‌توان پیچیده و ناکامل بودن تحلیل alias در Pixy را نیز از جمله معایب آن برشمرد.

ابزار Rips [۹]، یک ابزار نوشته‌شده به زبان PHP است که با استفاده از تحلیل ایستا به تشخیص آسیب‌پذیری‌ها می‌پردازد. Rips با توکن‌سازی و تجزیه کد متن برنامه، مدلی از برنامه ایجاد می‌کند و سپس با استفاده از این مدل به تشخیص حفره‌های آسیب‌پذیری می‌پردازد که می‌توانند به‌وسیله منبع ورودی کاربر آلوده شوند. یکی از مزایای Rips توانایی تشخیص محدوده وسیعی از آسیب‌پذیری‌ها است. از طرف دیگر، Rips چندین محدودیت شدید دارد که از مطلوب بودن آن می‌کاهد. Rips، با فرض یک کد برنامه‌نویسی خوب کار خود را انجام می‌دهد، چون Rips کد را به توکن‌هایی تجزیه می‌کند و حتی اگر کد PHP مورد نظر در بین راه قطع شده باشد و نیمه‌کاره باشد، اجازه تحلیل داده می‌شود؛ برخی توکن‌ها مانند پرانتزها به‌طور کامل صرف‌نظر می‌شوند. درحالی‌که این مسئله می‌تواند مزیت دیده شود اما در بیشتر مواقع به‌عنوان یک عیب Rips بیان می‌شود.

روش [۴]، از تحلیل ایستای سطح بایت کد استفاده می‌کند و بنابراین نیازی به کد متن برنامه ندارد. در این روش با استفاده از الگوی آسیب‌پذیری به‌صورت PQL، به تشخیص آسیب‌پذیری‌هایی همچون تزریق SQL و XSS پرداخته می‌شود. این روش تنها آسیب‌پذیری‌های برنامه‌های کاربردی تحت وب نوشته‌شده به زبان جاوا را تشخیص می‌دهد.

ابزار کاشف [۱۲]، به‌منظور تشخیص هرچه بیشتر نقاط آسیب‌پذیر در متن کد برنامه کاربردی وب، از تحلیل جریان داده به‌صورت معکوس استفاده می‌کند. در این روش، تحلیل جریان داده بر روی گراف جریان کنترل معکوس (RCFG) از حفره به سمت منبع ورودی انجام می‌گیرد و آلودگی داده در منبع ورودی، آسیب‌پذیری مربوطه را مشخص می‌کند. مزیت اصلی این ابزار این است که بیشتر مسیرهای برنامه در نظر گرفته می‌شوند و ما در روش خود تا حدودی از روش تحلیل این ابزار بهره گرفته‌ایم.

روش‌های پویا، شامل تحلیل متن یک برنامه در طول زمان اجرا و نظارت بر رفتار آن در زمان اجرای واقعی است. در این روش‌ها هیچ اطلاعات اولیه‌ای از متن برنامه در دسترس نیست، بنابراین تنها نقاط خاصی از برنامه با توجه به داده تولیدی در طول زمان اجرا در نظر گرفته می‌شود. نتایج حاصل از این تحلیل، دقیق‌تر ولی سربار زمان اجرا بیشتر می‌باشد. در [۶ -۵] با استفاده از تحلیل جریان داده به‌صورت پویا، آسیب‌پذیری‌های برنامه کاربردی تحت وب تشخیص داده شده است.

روش ارائه‌شده در [۵]، با استفاده از مفسر PHP اصلاح‌شده، برای هر رشته، اطلاعات آلوده را در سطح کاراکتر ردیابی می‌کند و با تغییر در کد متن برنامه، به تحلیل جریان داده به‌صورت پویا می‌پردازد.

روش [۶]، بدون نیاز به کد متن برنامه، تحلیل پویا را بر روی بایت کد جاوا در زمان اجرا انجام می‌دهد و ورودی‌های کاربر را تحلیل می‌کند. درصورتی‌که این ورودی‌های آلوده به حفره‌ای برسند، آسیب‌پذیری تشخیص داده می‌شود.

برخی روش‌های دیگر برای استفاده از مزیت هر دو تحلیل ایستا و پویا، از ترکیب آن‌ها استفاده می‌کنند و آسیب‌پذیری‌ها را تشخیص می‌دهند. مواردی مانند [۸ -۷] از جمله این روش‌ها هستند.

ابزار Ardilla [۷] با ترکیبی از تحلیل‌های ایستا و پویا، آسیب‌پذیری‌های تزریق SQL و XSS را تشخیص می دهد. این ابزار در ابتدا با استفاده از یک مولد ورودی، تمام مسیرهای اجرایی برنامه کاربردی وب را پیدا می‌کند و سپس دسته‌ای از ورودی‌ها که موجب اجرای مسیرهایی شده‌اند که از حفره عبور می‌کند در نظر گرفته می‌شود. در قدم بعد با توجه به الگوی آسیب‌پذیری مربوطه، متغیرهایی که مقدار مشخصی ندارند و از آن‌ها در حفره استفاده شده است، به‌عنوان کاندید حمله در نظر گرفته می‌شوند. با اجرای برنامه، واقعی بودن این کاندیدها بررسی می‌شود.

ابزار Amnesia [۸]، برای تشخیص آسیب‌پذیری‌های تزریق SQL استفاده می‌شود. در این ابزار ابتدا با اجرای برنامه، مدلی برای پرس‌وجوهای آن تولید می‌شود و سپس به‌صورت ایستا این مدل‌های ایجادشده با مدل قانونی پرس‌وجو تطبیق داده می‌شود. اگر مطابقت وجود نداشت آسیب‌پذیری تزریق SQL تشخیص داده می‌شود.

مزیت اصلی تحلیل ایستا این است که تمام مسیرها و مقادیر متغیرها در تمام کد برنامه مورد بررسی قرار می‌گیرند و نه فقط آن‌هایی که در زمان اجرا درگیر هستند. بنابراین، تحلیل ایستا می‌تواند خطاهایی را تشخیص دهد که ممکن است تا هفته‌ها، ماه‌ها و حتی سال‌ها بعد از انتشار، خودشان را آشکار نکنند. این مزیت تحلیل ایستا می‌تواند در تضمین امنیت برنامه بسیار مفید باشد؛ چون حملات امنیتی، اغلب برنامه کاربردی را در شیوه های پیش‌بینی‌نشده و بررسی‌نشده درگیر می‌کنند. تحلیل پویا می‌تواند در تضمین امنیت برنامه مفید باشد اما هدف اصلی آن، پیدا کردن و اشکال‌زدایی خطاهاست. از آن‌جا که هدف ما پیدا کردن آسیب‌پذیری‌های امنیتی در برنامه‌های کاربردی تحت وب است، تحلیل ایستا را برای این هدف انتخاب کرده‌ایم.

در این مقاله، روش جدیدی به‌منظور تشخیص آسیب‌پذیری‌های برنامه‌های کاربردی تحت وب معرفی شده است که با استفاده از تحلیل ایستای جریان داده احتمالی بر روی گراف‌های احتمال آسیب‌پذیری (VPG) که برای اولین بار در این مقاله مطرح شده‌اند، برنامه کاربردی تحت وب را تحلیل می‌کند. گراف احتمال آسیب‌پذیری به‌منظور پوشش حداکثری نقاط محتمل به آسیب‌پذیری در برنامه با انجام عملیاتی بر روی گراف جریان کنترل، طراحی و پیاده‌سازی شده است. روش تحلیل جریان داده احتمالی جهت کاهش تعداد مثبت‌های کاذب در تشخیص آسیب‌پذیری‌ها استفاده می‌شود.

دربخش دوم این مقاله، آسیب‌پذیری‌های رایج در برنامه‌های کاربردی وب تشریح می‌شود. در ادامه و در بخش‌های سوم و چهارم به‌ترتیب روش پیشنهادی در این مقاله و تست و پیاده سازی آن توضیح داده می‌شود. در نهایت در بخش پنجم نتیجه‌گیری ارائه شده است.

۲. آسیب‌پذیری‌های رایج در برنامه‌های کاربردی تحت وب

OWASP [۱۱] موسسه غیرانتفاعی تحقیقات بر روی بهبود امنیت برنامه‌های کاربردی تحت وب می‌باشد. این موسسه، ده مورد از بحرانی‌ترین آسیب‌پذیری‌های سطح کاربرد در برنامه‌های کاربردی تحت وب را انتشار داده است. آخرین دسته‌بندی از این کلاس های آسیب‌پذیری در سال ۲۰۱۳ نشان می‌دهد که تزریق‌های SQL و XSS، دو مورد از رایج‌ترین آسیب‌پذیری‌ها هستند که در رتبه‌های اول و سوم قرار گرفته‌اند. با استفاده از روش ارائه‌شده در این مقاله، این دو نوع آسیب‌پذیری تشخیص داده می‌شود.

۲-۱- آسیب‌پذیری XSS

XSS نوعی از آسیب‌پذیری است که به مهاجمان اجازه می‌دهد تا داده آلوده‌ای را به داخل یک صفحه از برنامه کاربردی از طریق مرورگر تزریق کنند [۱] و با این کار اطلاعات حساس از جمله کوکی‌ها را به سرقت ببرند. این داده آلوده اغلب کدهای مخرب جاوا اسکریپت هستند. در این آسیب‌پذیری جریان داده آلوده می‌تواند از ورودی به سمت حفره XSS وجود داشته باشد. حفره XSS بخش‌هایی از برنامه هستند که خروجی را برای مرورگر آماده و ارسال می‌کنند. کدهای مخرب جاوا اسکریپت پس از تزریق در برنامه، در حفره‌های XSS برای مرورگر ارسال می‌شود و با اجرای آن‌ها امنیت سیستم دچار مشکل می‌گردد.

آسیب‌پذیری‌های XSS، به دو دسته اصلی آسیب پذیری‌های مرتبه اول و مرتبه دوم تقسیم می‌شوند [۲]. در آسیب‌پذیری‌های مرتبه اول، داده آلوده به طور مستقیم و بدون واسطه، از منبع ورودی به سمت خروجی انتشار پیدا می کند، در حالی‌که در آسیب‌پذیری‌های مرتبه دوم یک وسیله ذخیره‌سازی داده مانند پایگاه داده به عنوان واسطی برای انتشار داده آلوده عمل می کند. آسیب‌پذیری‌های XSS مرتبه دوم از اهمیت بیشتری برخوردار هستند زیرا با ذخیره کد مخرب در وسیله ذخیره‌سازی، می‌تواند افراد زیادی را مورد حمله خود قرار دهد. روش ارائه شده در این مقاله می‌تواند آسیب‌پذیری‌های XSS مرتبه دوم را نیز تشخیص دهد.

۲-۲- آسیب‌پذیری تزریق SQL

آسیب‌پذیری SQLI یا تزریق SQL نوعی از آسیب‌پذیری tain style[۱] است که به موجب آن، یک فراخوانی ناامن به یک پایگاه داده جهت انجام عملیاتی بر روی آن صورت می‌گیرد. در این آسیب‌پذیری جریان آلوده‌ای از ورودی ناامن بدون رفع آلودگی، به حفره SQL پدید می‌آید. با استفاده از این آسیب‌پذیری‌ها، کاربران مخرب می‌توانند ساختار پرس‌وجوها برای دسترسی به پایگاه‌داده را تغییر دهند و پرس‌وجوهای دلخواه خود را اجرا نمایند و بدین وسیله به اطلاعات حساس و مهم برنامه‌های کاربردی وب دسترسی پیدا کنند.

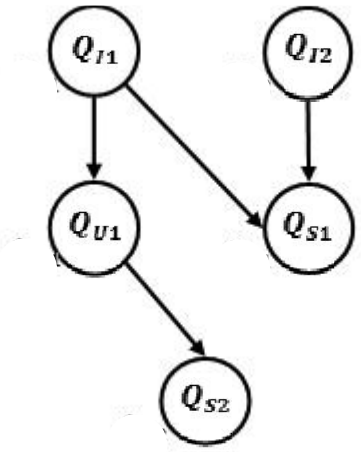
۳- روش پیشنهادی

در این بخش، به تشریح روش تشخیص پیشنهادی پرداخته می‌شود. در این روش، آسیب‌پذیری‌های تزریق SQL و XSS مرتبه اول و دوم با استفاده از تحلیل جریان داده احتمالی بر روی گراف احتمال آسیب‌پذیری تشخیص داده می‌شود. گراف احتمال آسیب‌پذیری به‌منظور پوشش حداکثری مسیرهایی است که در آن‌ها احتمال آسیب‌پذیری به میزان زیادی وجود دارد. گراف احتمال آسیب‌پذیری در دو شکل گراف احتمال آسیب‌پذیری مرتبه اول (VPG1) و گراف احتمال آسیب‌پذیری مرتبه دوم (VPG2) طراحی شده است. این دو گراف به‌ترتیب برای تشخیص آسیب‌پذیری‌های مرتبه اول XSS، تزریق SQL و مرتبه دوم XSS مورد استفاده قرار می‌گیرند و برای اولین بار در روش ما مطرح شده‌اند.

گراف احتمال آسیب‌پذیری با انجام عملیاتی بر روی گراف جریان کنترل به دست می‌آید. برای این کار ابتدا با استفاده از کامپایلر phc [۱۰]، گراف جریان کنترل برنامه کاربردی وب ایجاد می‌شود. در این گراف ، هر گره نشان‌دهنده یک دستور در برنامه می‌باشد و گره‌های متوالی‌که با یال‌های جهت‌دار به هم متصل هستند جریان دستورهای متوالی در برنامه را نشان می‌دهند. از مزایای استفاده از کامپایلر phc ساده‌تر شدن دستورات و عملیات محاسباتی و هم‌چنین تبدیل دستورات به فرمت سه آدرس می‌باشد. به عنوان نمونه، حلقه‌های for و while توسط این کامپایلر به دستورات پایه‌ای و ساده‌تری تبدیل می‌شوند. پس از ایجاد گراف جریان کنترل برای کد منبع هر صفحه از برنامه کاربردی وب، این گراف‌ها در ادامه برای تولید گراف‌هایی که آن‌ها را گراف‌های احتمال آسیب پذیری نام نهاده‌ایم مورد استفاده قرار می‌گیرند.

۳-۱- گراف احتمال آسیب‌پذیری مرتبه دوم (VPG2)

گراف احتمال آسیب پذیری مرتبه دوم به منظور تشخیص آسیب‌پذیری‌های XSS مرتبه دوم در برنامه، ایجاد می‌شود. برای ساختن گراف احتمال آسیب‌پذیری مرتبه دوم، در ابتدا با تحلیل بر روی گراف‌های جریان کنترل ایجادشده برای هر صفحه برنامه کاربردی، ارتباط بین دستورات درج، به‌روزرسانی و واکشی مربوط در یک وسیله ذخیره‌سازی داده مانند پایگاه داده در صفحات مختلف برقرار می‌شود. از آن‌جا که دستورات درج، به‌روزرسانی و واکشی داده لزوما در یک صفحه قرار ندارند، ارتباط بین صفحات مختلف برنامه کاربردی وب در نظر گرفته می‌شود و می‌توان آسیب پذیری‌های مربوط به ارتباط چند صفحه را نیز تشخیص داد. این مرحله در نهایت گرافی را ایجاد می‌کند که در مرحله بعدی برای طراحی گراف احتمال آسیب‌پذیری مرتبه دوم استفاده می‌شود. وجود این گراف در برنامه، همچنین لزوم ایجاد گراف احتمال آسیب‌پذیری مرتبه دوم را مشخص می‌کند. درصورتی که در یک برنامه کاربردی تحت وب نتوان این‌چنین گرافی را ایجاد کرد و یا آسیب پذیری‌ای که قصد تشخیص آن را داریم، شامل آسیب‌پذیری مرتبه دوم نباشد، گراف احتمال آسیب‌پذیری مرتبه دوم ایجاد نمی‌شود. شکل(۱) نمایی از گراف ایجادشده در این مرحله را نشان می‌دهد.



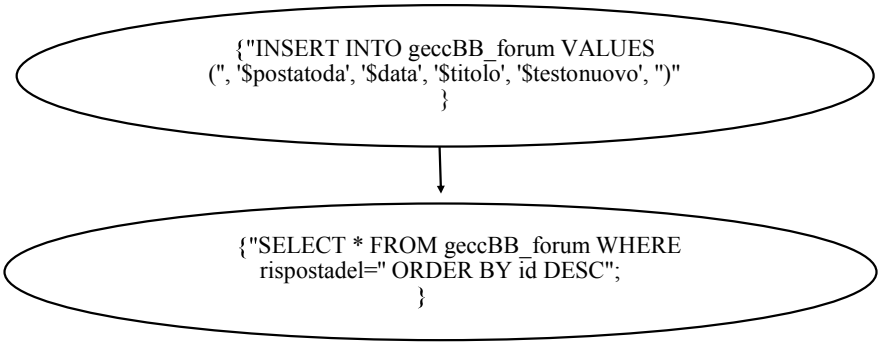
شکل(۱). گراف مرحله اول برای ایجاد گراف احتمال آسیب‌پذیری مرتبه دوم

در شکل(۱)، نودهای Q_I، Q_S و Q_U به‌ترتیب دستورات درج، واکشی و به‌روزرسانی داده را در یک وسیله ذخیره‌سازی داده نشان می‌دهد. در شکل(۲) بخشی از قطعه کد از دو فایل مختلف یک برنامه کاربردی تحت وب آورده شده است و گراف ایجادشده در این مرحله مشابه با شکل(۳) ایجاد می‌شود. داده واردشده به پایگاه داده geccBB_forum از طریق دستور Select در یک فایل دیگر از طریق دستور insert واکشی می‌شود و بنابراین، گراف این مرحله به صورت شکل(۳) ایجاد می‌شود. در ایجاد گراف مورد نظر در این مرحله شرط where در دستور select نیز در نظر گرفته می‌شود. اگر در جایی از برنامه بین دستور واکشی داده با توجه به شرط where و مقدار درج‌شده، هماهنگی وجود نداشته باشد، ارتباطی بین این دستورها برقرار نمی‌شود.

```
echo "<hr />\n";
$query_selecta_topic="SELECT * FROM geccBB_forum WHERE rispostadel=" ORDER BY id DESC";
$r=mysql_query($query_selecta_topic) or die ("devi eseguire prima <a href=\"install.PHP\">install.PHP</
a>\n");
echo "<ul>\n";
```

```
elseif($postatoda && $titolo && $testo)
{
   $data=date("U");
   $query_ins_post="INSERT INTO geccBB_forum VALUES(", '$postatoda', '$data', '$titolo', '$testonuovo',
")";
   $i=mysql_query($query_ins_post);
}
```

شکل(۲). بخشی از قطعه کد از دو فایل مختلف یک برنامه کاربردی تحت وب



شکل(۳). گراف نشان‌دهنده ارتباط بین دستورات پایگاه داده برای قطعه کدهای شکل ۲

در مرحله بعد با توجه به نوع آسیب‌پذیری مورد نظر، لیست حفره‌هایی که در آن‌ها از متغیرها استفاده شده است ایجاد شده و در گراف جریان کنترل نشانه‌گذاری می‌شوند. بنابراین تنها حفره‌هایی در نظر گرفته می‌شوند که احتمال آسیب‌پذیری در آن‌ها وجود دارد و بنابراین در مرحله تحلیل، از بررسی حفره‌های نامربوط جلوگیری می‌شود. پس از نشانه‌گذاری حفره‌های مناسب در گراف جریان کنترل برنامه، در یک فرآیند بازگشتی از حفره‌ها به سمت منابع ورودی ناامن و با کمک گراف حاصل از مرحله قبل، گراف احتمال آسیب‌پذیری مرتبه دوم ایجاد می‌شود. این گراف جهت تشخیص آسیب‌پذیری‌های مرتبه دوم و همچنین آسیب‌پذیری‌های مربوط به ارتباط چند صفحه از برنامه کاربردی وب ایجاد می‌شود.

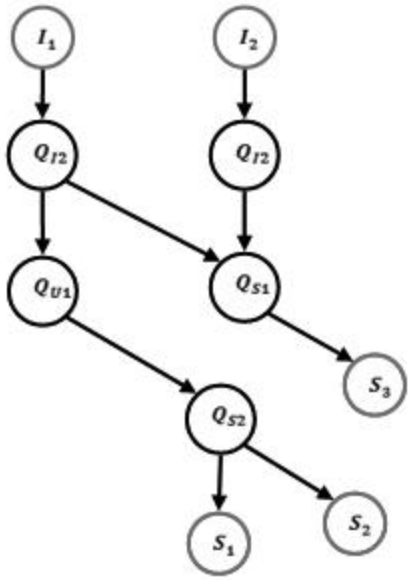
از مزایای شناسایی مسیرها به‌صورت پایین به بالا یعنی از سمت حفره‌ها به سمت منابع ورودی ناامن که ما در روش خود به کار می‌بریم، این است که حفره‌های مربوط به هر آسیب‌پذیری، اغلب توابعی هستند که داده آن‌ها از منابع ورودی خاصی دریافت می‌شوند و این مسئله تشخیص منابع ورودی و مسیرهای مناسب در گراف احتمال آسیب‌پذیری را سریع‌تر و ساده‌تر می‌کند و بنابراین از بررسی مسیرهای بیهوده در گراف‌ها جلوگیری می‌شود. شکل(۴) نمایی از گراف احتمال آسیب‌پذیری مرتبه دوم را نشان می‌دهد. گره‌های I که با رنگ آبی نشان داده شده‌اند، منبع ورودی ناامن به برنامه را نشان می‌دهند و گره‌های S که قرمزرنگ هستند، نشان‌دهنده حفره‌های برنامه می‌باشند.

همان‌طور که بیان شد، در پیاده‌سازی گراف احتمال آسیب‌پذیری مرتبه دوم، ابتدا گره‌های Q تولید می‌شوند که ارتباط بین دستورات درج، واکشی و به‌روزرسانی داده مربوط به یک وسیله ذخیره‌سازی را برقرار می‌کنند. سپس در ادامه بین دستورات مربوط به ورودی ناامن و حفره‌ها در صفحات مختلف با واسطه این گره‌های Q، ارتباط مناسب برقرار می‌شود.

۳-۲- گراف احتمال آسیب‌پذیری مرتبه اول (VPG1)

گراف احتمال آسیب‌پذیری مرتبه اول، برای تشخیص آسیب‌پذیری‌های XSS مرتبه اول و همچنین تزریق SQL ایجاد می‌شود. برای تشخیص آسیب‌پذیری‌های XSS مرتبه اول، پس از

ایجاد گراف احتمال آسیب‌پذیری مرتبه دوم، به تحلیل جریان داده برای تشخیص آسیب پذیری های مرتبه دوم پرداخته می‌شود. در صورتی که حفره‌ای شامل این نوع آسیب‌پذیری باشد از لیست حفره‌های نشانه‌گذاری‌شده برای ایجاد گراف‌های احتمال آسیب‌پذیری مرتبه اول حذف می‌شود.



شکل(۴). نمایی از گراف احتمال آسیب‌پذیری مرتبه دوم

برای ایجاد گراف های احتمال آسیب پذیری مرتبه اول، پس از نشانه‌گذاری حفره‌های باقی مانده در گراف جریان کنترل، تنها مسیرهای منتهی از حفره‌ها به سمت ورودی‌های ناامن مربوطه در نظر گرفته می شوند و بقیه مسیرها از گراف جریان کنترل حذف می شوند. بنابراین تحلیل جریان داده در مسیرهای نامربوط که احتمال آسیب‌پذیری در آن‌ها وجود ندارد، انجام نمی‌شود. در نهایت با حذف مسیرهایی که به حفره‌های نشانه‌گذاری‌شده ختم نمی‌شوند، گرافی به‌دست می‌آید که در حقیقت، برای آسیب‌پذیری مرتبه اول XSS محتمل می‌باشد.

شکل(۷) گراف احتمال آسیب‌پذیری مرتبه اول را برای قطعه کد شکل(۵) نشان می دهد. در این بخش از گراف، پایین ترین گره، مربوط به چاپ مقداری در خروجی با استفاده از تابع echo است که حفره XSS را نشان می‌دهد و با رنگ قرمز مشخص شده است. بالاترین گره که با رنگ آبی نشان داده شده است، یک منبع ورودی ناامن با تابع GET می‌باشد. همان‌طور که در شکل (۷) مشاهده می‌شود کامپایلر phc تابع echo را با print جایگزین می‌کند.

```
<?php
$head = $_GET['head'];
?>
<html>
<head><?php echo $head; ?></head>
```

شکل(۵). بخشی از متن ساده‌شده برنامه کاربردی وب

گراف احتمال آسیب‌پذیری مرتبه اول برای هر صفحه در برنامه کاربردی وب ایجاد و سپس تحلیل جریان داده بر روی آن‌ها انجام می‌شود.

برای تشخیص آسیب‌پذیری‌های تزریق SQL نیز از گراف احتمال آسیب پذیری مرتبه اول استفاده می شود؛ به این‌صورت که حفره هایی که نشان دهنده این نوع آسیب پذیری است در گراف جریان کنترل نشانه‌گذاری می‌شوند و مسیرهای منتهی از حفره‌های تزریق SQL به سمت ورودی‌های ناامن در نظر گرفته می‌شوند. به این صورت گراف احتمال آسیب‌پذیری مرتبه اول یا به عبارتی آسیب‌پذیری تزریق SQL ایجاد می‌شود.

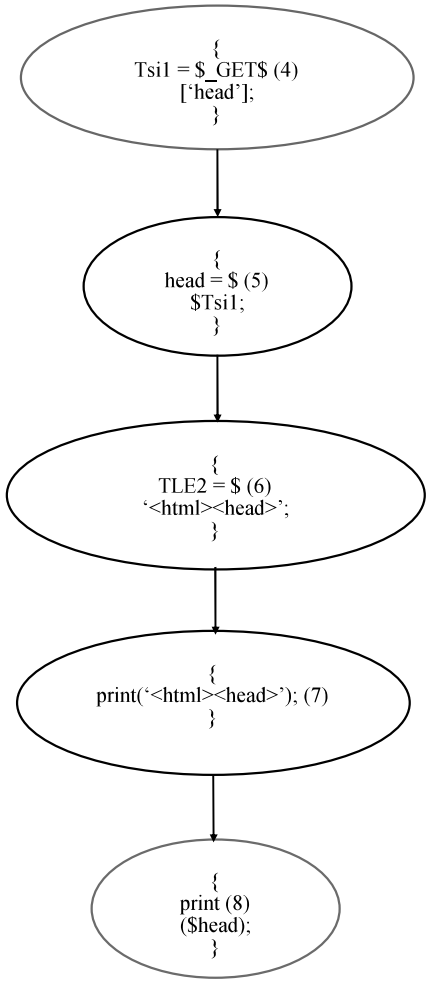
شکل(۸) گراف احتمال آسیب‌پذیری مرتبه اول برای قطعه کد شکل(۶) را نشان می‌دهد که دارای آسیب‌پذیری تزریق SQL است.

```
<?php
$id = $_GET['id'];
$query = "SELECT * FROM users WHERE id =
'" . $id . "'" ;
mysql_query( $query ) ; /* BAD */ /* SQL */
```

شکل(۶). بخشی از متن ساده‌شده برنامه کاربردی وب

همان‌طور که از شکل(۶) مشخص می‌شود، نوع حفره در این نوع آسیب‌پذیری، متفاوت با XSS است. نود قرمز رنگ، حفره مربوطه را نشان می‌دهد که در ابتدا در گراف جریان کنترل نشانه‌گذاری می‌شود. تابع mysql_query یک حفره SQL است که می تواند منجر به آسیب‌پذیری تزریق SQL در برنامه شود. مشابه با ایجاد گراف احتمال آسیب پذیری برای آسیب‌پذیری‌های XSS، مسیر اجرایی از این حفره به سمت منبع ورودی نا امن در برنامه، بررسی و برای ایجاد گراف احتمال آسیب‌پذیری مرتبه اول ذخیره می‌شود. در قطعه‌کد آسیب پذیر شکل(۶) این مسیر از حفره mysql_query ($query) به تابع ورودی نا امن GET['id'] می‌رسد. در نهایت گراف احتمال آسیب پذیری مرتبه اول برای قطعه‌کد آسیب پذیر۶ به‌صورت شکل (۸) ایجاد می شود. سپس، در ادامه، این گراف به بخش تحلیل‌گر داده می‌شود و این بخش با تحلیل بر روی گراف،

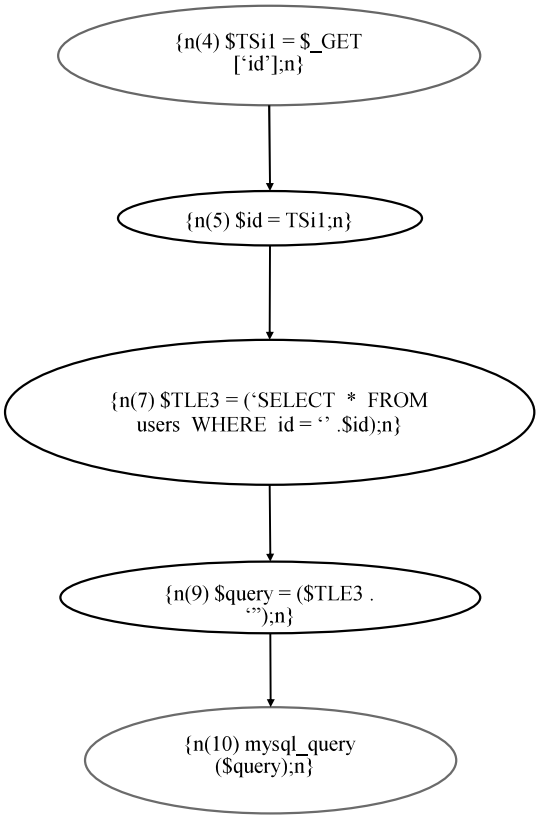
آسیب‌پذیری حفره مربوطه را تشخیص می‌دهد.



شکل(۷). گراف احتمال آسیب‌پذیری مرتبه اول برای قطعه‌کد شکل

۳-۳- تحلیل جریان داده پیشنهادی

در هر مرحله با ایجاد گراف های احتمال آسیب پذیری، تحلیل جریان داده آلوده برای این گراف‌ها انجام می‌شود. تحلیل جریان داده استفاده‌شده در این روش را تحلیل جریان داده احتمالی نامیده‌ایم که جریان داده را از هر حفره به سمت منابع ورودی تحلیل می‌کند و با توجه به نسبت مسیرهای آسیب‌پذیر به کل مسیرهای بررسی‌شده برای یک حفره، آسیب پذیری آن تشخیص داده می‌شود. این روش تحلیل برای اولین‌بار در این مقاله مطرح شده است. مزیت تحلیل گراف‌های احتمال آسیب‌پذیری به صورت پایین به بالا، به کارایی آن برمی‌گردد. در هنگام تحلیل یک مسیر از حفره به سمت منبع ورودی، در صورتی که ما در یک نود به تابع رفع آلودگی داده برسیم، دیگر از ادامه تحلیل بر روی آن مسیر صرف‌نظر می‌کنیم. بنابراین از آن‌جایی که توابع رفع آلودگی داده اغلب نزدیک به حفره‌ها قرار داده می‌شوند، تا حدودی سربار زمان اجرا برای تحلیل کاهش می یابد. پس از ساخت گراف های احتمال آسیب پذیری، هر گره‌ای که نشان‌دهنده حفره است به عنوان گره شروع‌کننده تحلیل در گراف مشخص شده است. در روش تحلیل ما، هر متغیر برچسبی دارد که این برچسب می تواند مقدار درست، نادرست و یا نامشخص داشته باشد. در ابتدا برچسب تمام متغیرها مقدار نامشخص گرفته‌اند. با آغاز تحلیل، تمام متغیرهای استفاده‌شده در حفره مورد بررسی، آلوده در نظر گرفته می‌شوند و برچسب آلودگی آن‌ها مقدار درست می‌گیرد. در ادامه، تمام جریان‌های داده از این حفره به سمت منابع ورودی مورد تحلیل قرار می‌گیرند. اگر جریان داده‌ای از حفره مورد نظر به سمت یک منبع ورودی، آلوده شناخته شد به این معنا که با بررسی مسیر حفره تا منبع ورودی در نهایت برچسب آلودگی متغیر مورد استفاده در منبع ناامن، مقدار درست گرفته بود، تحلیل پایان نمی‌گیرد بلکه این جریان داده به تعداد جریان داده‌های آلوده برای حفره مربوطه اضافه می‌شود و سپس مسیر دیگری از همان حفره به یک منبع ورودی دیگر بررسی می‌شود. پس از این‌که مسیرها از یک حفره به تمام منابع ورودی مورد تحلیل قرار گرفت، با استفاده از رابطه [۱]، آلودگی حفره مربوطه تشخیص داده می‌شود.



شکل(۸). گراف احتمال آسیب‌پذیری مرتبه اول برای قطعه‌کد شکل (۶)

اگر $\frac{\text{تعداد جریان داده های آلوده برای حفره x}}{\text{مجموع جریان داده های بررسی شده برای x}} \geq \frac{1}{2}$

آنگاه x یک حفره آسیب پذیراست[۱]

استفاده از تحلیل جریان داده احتمالی بر روی گراف‌های احتمال آسیب‌پذیری که در این مقاله پیشنهاد شده است، در حقیقت جهت کاهش مثبت‌های کاذب و افزایش دقت تشخیص آسیب‌پذیری است؛ زیرا برای تشخیص آسیب پذیری، با کمک روش پیشنهادی تنها حفره‌هایی را در نظر می‌گیریم که احتمال آسیب پذیری در آن ها بالاست و این خود باعث کاهش میزان مثبت‌های کاذبی می‌شود که ابزارهای دیگر آن‌ها را به عنوان آسیب‌پذیری تشخیص می‌دهند. در بخش پایانی این فصل، نتایج حاصل از به‌کارگیری این روش را بر روی برنامه‌های نوشته‌شده به زبان PHP نشان می‌دهیم.

نکته‌ای که در مورد این رابطه باید بیان کرد این است که نسبتی که برای تشخیص آسیب‌پذیری یک حفره در نظر گرفته شده است بر اساس نصف مسیرهایی است که تحلیل جریان داده آلوده بر روی آن‌ها صورت می‌گیرد. این نسبت به صورت کلی در این مقاله مطرح شده است و در ادامه، میزان صحت آن با ارزیابی روش پیشنهادی سنجیده می شود. در حقیقت آن چه که در تحلیل جریان داده احتمالی در این روش می تواند میزان دقت روش را افزایش دهد، تعیین همین نسبت است. این نسبت باید به گونه‌ای تعیین شود که علاوه‌بر اینکه میزان مثبت کاذب در تشخیص آسیب پذیری را تا حدودی کاهش می دهد، منفی کاذب زیادی هم تولید نکند. ما می‌توانیم با بررسی بیشتر برنامه‌های کاربردی وب و ارزیابی آن‌ها با روش پیشنهادی، برای این نسبت به صورت تجربی مقدار دقیق‌تری را تعیین کنیم. تعیین تجربی نسبت مطرح‌شده در رابطه می تواند جزو کارهای آتی ما محسوب شود. در حال حاضر در این مقاله این نسبت به این صورت تعیین شده است و تا حدودی برای برخی برنامه‌های کاربردی وب ارزیابی‌شده، پاسخ قابل قبولی گزارش می‌دهد.

۴- پیاده‌سازی و ارزیابی

روش ارائه‌شده در این مقاله در پلتفرم‌های لینوکس و ویندوز و به زبان جاوا پیاده‌سازی شده است. این روش آسیب‌پذیری‌های تزریق SQL و XSS را در برنامه های کاربردی تحت وب که به زبان PHP نوشته شده‌اند، تشخیص می‌دهد و برای سایر آسیب‌پذیری‌های مشابه قابل توسعه است.

برای پیاده‌سازی این روش، ابتدا گراف جریان کنترل برای برنامه کاربردی تحت وب، توسط کامپایلر phc دریافت می‌شود. این گراف به عنوان ورودی برای ایجاد گراف های احتمال آسیب‌پذیری مرتبه اول و دوم به کار می‌رود. پیاده‌سازی مورد نظر از طریق این ورودی، گراف های احتمال آسیب پذیری را ایجاد می کند و سپس تحلیل جریان داده پیشنهادی را بر روی این گراف ها انجام می دهد. در بخش ۲ و ۳ نحوه عملکرد روش پیشنهادی شرح داده شد. جهت نشان دادن اینکه روش پیشنهادی ما می‌تواند آسیب‌پذیری‌ها را تشخیص دهد و تا حدودی به افزایش دقت تشخیص آسیب‌پذیری‌ها کمک می‌کند، این روش را در دو مرحله مورد ارزیابی قرار دادیم. در مرحله اول، روش خود را بر روی چندین فایل تست کوچک نوشته‌شده به زبان PHP که دارای آسیب‌پذیری‌های تزریق SQL و همچنین XSS بودند اجرا کردیم. برخی از این قطعه برنامه‌ها برای ارزیابی روش، از [۱] انتخاب شده‌اند. نتایج تحلیل نشان می‌دهد که روش ما می تواند درصورت وجود آسیب‌پذیری در یک قطعه برنامه نوشته‌شده به زبان PHP، آن را به درستی تشخیص دهد.

در گام دوم ارزیابی روش پیشنهادی، پیاده‌سازی خود را بر روی چند برنامه کاربردی تحت وب منبع باز اجرا کرده و نتایج آن را با چند ابزار تشخیص آسیب‌پذیری منبع باز مقایسه نمودیم. جدول (۱) نتیجه اجرای روش پیشنهادی بر روی دو برنامه کاربردی تحت وب متن باز را نشان می‌دهد.

همان‌طور که در جدول (۱) مشاهده می‌کنید، روش ما می‌تواند چندین آسیب‌پذیری را برای دو برنامه کاربردی تحت وب به همراه

جدول(۱). نتیجه اجرای روش پیشنهادی بر روی دو برنامه متن باز

		XSS		SQLI	
برنامه کاربردی تحت وب	تعداد فایل‌های ورودی	مجموع آسیب‌پذیری‌ها	مثبت‌های کاذب	مجموع آسیب‌پذیری‌ها	مثبت‌های کاذب
GeccBBlite	۱۱	۳	۱	۲	۰
WackoPicko	۵۰	۵	۰	۱	۰

جدول(۲). نتیجه اجرای ابزار Rips بر روی دو برنامه متن باز

		XSS		SQLI	
برنامه کاربردی تحت وب	تعداد فایل‌های ورودی	مجموع آسیب‌پذیری‌ها	مثبت‌های کاذب	مجموع آسیب‌پذیری‌ها	مثبت‌های کاذب
GeccBBlite	۱۱	۴	۴	۴	۲
WackoPicko	۵۰	۸	۳	۳	۲

جدول(۳). نتیجه اجرای ابزار کاشف بر روی دو برنامه متن باز

		XSS		SQLI	
برنامه کاربردی تحت وب	تعداد فایل‌های ورودی	مجموع آسیب‌پذیری‌ها	مثبت‌های کاذب	مجموع آسیب‌پذیری‌ها	مثبت‌های کاذب
GeccBBlite	۱۱	۴	۲	۳	۰
WackoPicko	۵۰	۴	۰	۰	۰

تنها ۱ مثبت کاذب برای برنامه کاربردی GeccBBlite تشخیص دهد. در ادامه، ما ابزارهای Rips و کاشف را بر روی همان برنامه‌های کاربردی اجرا کردیم. نتایج اجرای این ابزار در جدول‌های (۳-۲) نشان داده شده است.

ابزار Rips آسیب‌پذیری‌های بیشتری را نسبت به روش پیشنهادی ما تشخیص می‌دهد اما با مقایسه تعداد مثبت‌های کاذب می‌توان تا حدودی بهبود را در روش ما مشاهده کرد. جدول (۴) اطلاعات تشخیص آسیب‌پذیری توسط سه روش معرفی شده را نشان می‌دهد و شکل (۹)، یک مقایسه کلی از تعداد آسیب‌پذیری‌های تشخیص داده شده و تعداد مثبت های کاذب در این ابزار را بیان می‌کند. روش پیشنهادی و ابزار کاشف، تعداد آسیب‌پذیری‌های یکسانی را تشخیص داده‌اند اما روش ما در تعداد مثبت‌های کاذب بهتر از کاشف عمل کرده است.

۵- نتیجه‌گیری و فعالیت‌های آتی

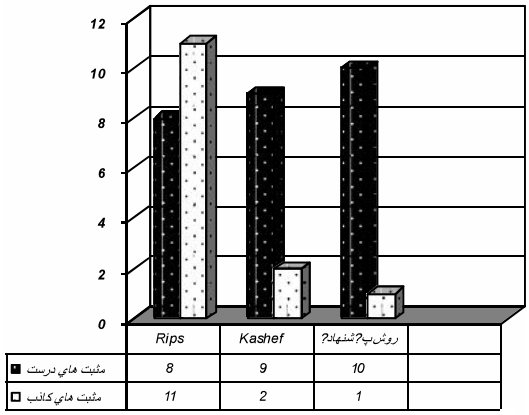
با توجه به نقش مهم برنامه‌های کاربردی تحت وب در پیشرفت فناوری اطلاعات و ذخیره و تبادل اطلاعات حساس و مهم توسط این دسته‌برنامه‌ها، امنیت آن ها از جایگاه ویژه‌ای برخوردار می‌باشد. تشخیص آسیب‌پذیری‌های برنامه‌های کاربردی تحت وب، تهدید حملات نفوذگران برای سرقت اطلاعات حساس و مهم را به میزان قابل توجهی کاهش می‌دهد. در این مقاله یک روش جدید برای تشخیص آسیب پذیری‌ها ارائه شد که به‌صورت ایستا تحلیل جریان داده آلوده احتمالی را بر روی گراف احتمال آسیب‌پذیری انجام می‌دهد. این روش برای اولین‌بار در این مقاله معرفی شده است که می‌تواند تا حدودی دقت تشخیص آسیب پذیری را افزایش دهد. گراف احتمال آسیب‌پذیری به‌منظور پوشش حداکثری نقاط محتمل به آسیب پذیری در برنامه طراحی شده است و تحلیل جریان داده احتمالی جهت کاهش میزان مثبت‌های کاذب در تشخیص آسیب‌پذیری‌ها اجرا می‌شود.

در روش پیشنهادی، چندین قطعه‌کد آسیب‌پذیر و دو برنامه متن باز مورد تست و بررسی قرار گرفت. نتایج حاصل از تحلیل و تست نشان‌دهنده توانایی بالای روش ارائه‌شده در تشخیص آسیب‌پذیری‌های ممکن بوده است. با افزایش دقت در تحلیل جریان داده آلوده، نتایج مثبت کاذب کاهش خواهند یافت.

جدول(۴). مقایسه روش‌های تشخیص آسیب‌پذیری

ابزار	مجموع آسیب‌پذیری‌ها		مثبت‌های کاذب	
	XSS	SQLI	XSS	SQLI
روش پیشنهادی	۸	۳	۱	۰
Rips	۱۲	۷	۷	۴
Kashef	۸	۳	۲	۰

روش تشخیص معرفی‌شده در این مقاله، تنها آسیب‌پذیری‌های تزریق SQL و XSS را تشخیص می دهد که می تواند برای تشخیص آسیب‌پذیری‌های مشابه دیگر توسعه یابد. همچنین، گراف احتمال آسیب‌پذیری مرتبه دوم باید برای برنامه‌های کاربردی وب که از پیچیدگی زیادی برخوردار است- به‌عنوان نمونه شامل دستورهای پیچیده واکشی داده از وسایل ذخیره‌سازی داده است- بهبود یابد.



شکل(۹). مقایسه دقت سه ابزار تشخیص آسیب‌پذیری

همچنین نسبت تعیین‌شده برای تحلیل جریان داده احتمالی در روش تشخیص پیشنهادی می تواند برای افزایش دقت تشخیص، به‌صورت دقیق‌تری با ارزیابی برنامه‌های کاربردی وب بیشتری تعیین شود. از طرف دیگر از آن‌جایی که برای هر متغیر در برنامه کاربردی وب تنها سه برچسب آلوده، غیرآلوده و نامشخص در نظر گرفته می‌شود و مقادیر احتمالی رشته‌ها در زمان اجرا در نظر گرفته نشده است، نتایج حاصل از تحلیل می‌تواند مثبت کاذب داشته باشد. برای کاهش میزان مثبت کاذب، لازم است تا تحلیل جریان داده احتمالی مورد استفاده در این روش با در نظر گرفتن تحلیل مقادیر احتمالی رشته ها دقیق‌تر شود که این موارد جزو کارهای آتی می‌باشد.

۶- مراجع

[1] L. D. P. Nico, "Authomated Security Review of PHP Web Applications with Source Code Analysis," Thesis, University of Groningen, pp. 14-14, 2010.

[2] C. Korscheck, "Automatic Detection of Second-Order Cross-Site Scripting Vulnerabilities," Thesis, University of Tubingen, pp. 2-4, 2010.

[3] J. Nenad, K. Christopher, and K. Engin, "Static analysis for detecting taint-style vulnerabilities in web applications," Journal of Computer Security, vol. 18, no. 5, 2010.

[4] L. Benjamin and S. L. Monica, "Finding Security Vulnerabilities in Java Applications with Static Analysis," in USENIX Security Symposium, vol. 14, pp. 18-18, 2005.

[5] N. Anh, G. Salvatore, G. Doug, Sh. Jeff, and E. David, "Authomatically hardening web applications using precise tainting," Security and Privacy in the Age of Ubiquitous Computing, Springer US, pp. 295-307, 2005.

[6] H. Vivek, Ch. Deepak, and F. Michael, "Dynamic taint propagation for java," in Annual Computer Security Applications Conference, vol. 21, pp. 303 -311, 2005.

[7] K. Adam, J. G. Philip, J. Karthick, and D. E. Michael, "Authomatic Creation of SQL Injection and Cross-Site Scripting Attacks," in International Conference on Software Engineering, vol. 31, pp. 199-209, 2009.

[8] G. H. William, O. Alessandro, "AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks," in IEEE/ACM International Conference on Authomated Software Engineering, vol. 20, pp. 174-183, 2005.

[9] "RIPS-A static source code analyser for vulnerabilities in PHP scripts," http://rips-scanner.sourceforge.net/ 2011.

[10] "phc – the open source PHP compiler," [Online], Version 3.0.1, http://www.phpcompiler.org/July 2013.

[11] "OWASP Top Ten Project," [Online], https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project/June 2013.

[12] M. Ghorbanzadeh and M. R. Shahriari, "Static detection of web applications vulnerabilities using inverse data flow analysis for covering mejority of sensitive points to vulnerability," In international ISC conferenec 9, Tehran, 1391. (In persian)