استفاده از مدل مبتنی بر ریسک امنیتی در خزش برنامه‌های غنی-شده‌ی تحت وب

محمد جعفرنژادقمی[۱]، سید امین حسینی سنو[۲]

[۱] دانشگاه فردوسی مشهد، mo.jafarnezhadghomi@stu.um.ac.ir

[۲] گروه مهندسی کامپیوتر، دانشگاه فردوسی مشهد، hosseini@um.ac.ir

چکیده

توسعه و پیشرفت برنامه‌های تحت وب، چالش‌هایی را در رابطه با بررسی خودکار امنیت این برنامه‌ها به وجود آورده است. در واقع ظهور فناوری‌هایی مانند ارتباط ناهمگام برنامه‌ی وب با سرور، سبب شده است تا خزنده‌های سنتی فاقد کارایی لازم برای خزش برنامه‌های غنی-شده‌ی تحت وب باشند. برای رفع این چالش، خزنده‌های مبتنی بر مدل ارائه شدند. این خزنده‌ها در حال حاضر در حال آزمایش هستند و هیچ یک از ابزارهای کاربردی از آن‌ها استفاده نمی‌کنند. هیچ یک از کارهای انجام شده در زمینه‌ی خزش مبتنی بر مدل، با نگاه امنیتی نبوده است و در نتیجه مدل‌های ارائه شده از برنامه‌های وب، برای انجام تست‌های امنیتی و کشف آسیب‌پذیری‌ها بهینه نیستند. در این مقاله یک مدل ریسک برای خزش برنامه‌های غنی‌شده‌ی وب ارائه شده‌است. در این مدل، خزنده سعی می‌کند تا حالت‌های آسیب‌پذیر را قبل از حالت‌های دیگر موجود در برنامه کشف کند. بدین ترتیب چنانچه به دلایلی مانند زمان محدود، خزنده نتواست خزش را تا انتها انجام دهد، حالت‌های کشف شده، از نظر امنیتی، از اهمیت بیشتری برخوردار خواهند بود. به منظور ارزیابی کارایی راهبردهای خزش مختلف، هزینه‌ی خزش را بر حسب تعداد رویدادهای اجرا شده و همچنین تعداد بارگذاری‌های مجدد انجام شده در طول خزش، محاسبه می‌کنیم. نتیجه‌ی آزمایش‌ها به همراه تحلیل آن‌ها در انتهای مقاله نشان می‌دهد که راهبرد پیشنهادی با صرف هزینه‌ی کمتر توانسته است همه‌ی حالت‌های آسیب‌پذیر را کشف کند.

واژه‌های کلیدی

خزش، برنامه‌های غنی‌شده‌ی تحت وب، امنیت، ریسک

۱- مقدمه

امنیت برنامه‌های کاربردی تحت وب از آن جهت حائز اهمیت است که در حال حاضر اطلاعات حساس سازمان‌ها، شرکت‌ها، و افراد بر روی اینترنت قرار گرفته است و از طریق این برنامه‌ها قابل دسترس است[1]. گونه‌ای از برنامه‌های تحت وب که با عنوان برنامه‌های غنی‌شده‌ی تحت وب[۱] شناخته می‌شوند، از فناوری‌هایی مانند جاوااسکریپت و ای‌جکس برای تعامل با کاربر استفاده می‌کنند[2]. در برنامه‌های سنتی وب، کاربر با دنبال کردن URL ها می‌توانست از صفحه‌ای به صفحه‌ی دیگری برود. این وضعیت در برنامه‌های غنی‌شده‌ی تحت وب، به دو طریق متمایز شده است: ۱) عملیاتی مانند تغییر صفحه و افزودن لینک می‌توانند با استفاده از زبان‌های اسکریپتی نظیر جاوااسکریپت، در سمت کلاینت، و بدون اتصال به سرور انجام گیرند. ۲) ارتباط ناهمگام کلاینت با سرور می‌تواند بدون تغییر آدرس URL، سبب تغییر بخشی از صفحه شود[3]. با پیدایش این تغییرات در توسعه‌ی برنامه‌های وب، چالش‌های جدیدی در زمینه‌ی جستجوپذیری و آزمون‌پذیری به وجود آمده است.

یکی از ابزارها جهت کشف خودکار آسیب‌پذیری‌های امنیتی در برنامه-های وب، اسکنرها[۲] هستند که به دو دسته‌ی جعبه‌سیاه[۳] و جعبه‌سفید[۴] تقسیم می‌شوند. اسکنرهای جعبه‌سیاه برخلاف انواع جعبه‌سفید آن، به کد برنامه دسترسی ندارند. این اسکنرها به طور کلی فعالیت خود را در سه مرحله انجام می‌دهند: خزش[۵]، تست، و بررسی پاسخ. اهمیت مرحله‌ی اول (خزش)، که وظیفه‌ی کشف صفحات مختلف برنامه‌ی وب را بر عهده دارد،

[1] Rich Internet Applications

[2] Scanner

[3] Black-box

[4] White-box

[5] Crawling

از آنجا مشخص می‌شود که اگر صفحه‌ای از برنامه‌ی وب یافت نشود، در مرحله‌ی بعد هرگز بررسی نخواهد شد[4].

در فرایند خزش تلاش می‌شود تا تمام صفحات برنامه‌ی وب، و همچنین اطلاعات مربوط به رسیدن به آن صفحات، بدست آید. به ابزاری که عمل خزش را انجام می‌دهد، خزنده می‌گویند[2]. اسکنرهای امنیتی به خزنده‌ی وب نیاز دارند تا حالت‌های مختلف برنامه را کشف کنند[5].

هیچ یک از کارهای انجام شده در زمینه‌ی خزش، با نگاه امنیتی نبوده است. لذا نتایج حاصل از خزش، برای انجام آزمون‌های امنیتی و کشف آسیب‌پذیری‌ها بهینه نیستند. هدف خزنده‌های مبتنی بر مدل که تا کنون ارائه شدند، پیدا کردن سریع‌تر حالت‌های جدید است. در این پژوهش، با اعمال راهکاری جدید، یک مدل ریسک ایجاد کردیم، و خزنده را برای پیدا کردن هرچه سریع‌تر حالت‌های پرخطر، هدایت می‌کنیم. که در نتیجه‌ی آن، اسکنر مربوطه قادر خواهد بود تا آسیب‌پذیری‌های بیشتری را در زمان کمتری کشف کند.

ادامه‌ی مقاله به صورت زیر است: در بخش دوم پیشینه‌ای از خزش در برنامه‌های وب مورد بررسی قرار گرفته است. در بخش سوم راهکار پیشنهادی مطرح می‌گردد. و سپس در بخش چهارم نحوه‌ی پیاده‌سازی و ارزیابی مدل ارائه شده تشریح می‌شود. در انتها نیز در بخش پنجم به نتیجه‌گیری پرداخته شده است و همچنین محدودیت‌ها و مسائل باز موجود برای ادامه‌ی کار بیان شده است.

۲- پیشینه

خزش برنامه‌های وب را می‌توان به طور کلی به دو دسته‌ی خزش برنامه-های سنتی، و خزش برنامه‌های غنی‌شده تقسیم کرد. چالش‌های موجود در خزش برنامه‌های سنتی وب، به خوبی مورد مطالعه قرار گرفته است و راه-حل‌های مناسبی برای آن‌ها ارائه شده است. اما پژوهش‌های ارائه شده در زمینه‌ی خزش برنامه‌های غنی‌شده‌ی وب، در حال بررسی چالش اولیه، یعنی کشف صفحات و حالت‌ها هستند. خزش این برنامه‌ها برخلاف خزش برنامه‌های سنتی، صرفا با پیمایش URL ها امکان‌پذیر نیست. زیرا در این برنامه‌ها، دور از ذهن نیست که تنها یک URL وجود داشته باشد و تمام قسمت‌های دیگر برنامه با استفاده از رویدادها پیمایش شوند. بدنه‌ی اصلی پژوهش‌های مربوط به خزش برنامه‌های غنی‌شده، از سه منبع اصلی برگرفته شده است: ۱) گروه پژوهشی در دانشگاه سوئیسی ETH Zurich [6]، ۲) گروه پژوهشی دیگر که ابزاری به نام Crawljax را توسعه داده-اند[7,8]. ۳) گروه پژوهشی Amalfitano و همکارانش[12-9].

در مطالعات انجام شده، به غیر از راهبرد خزش مبتنی بر مدل[6][13]، و راهبرد حریصانه[7][14]، توجه کافی به کارایی راهبردهای خزش، نشده است. اکثر روش‌های موجود از راهبردهای استاندارد اول‌سطح یا اول‌عمق استفاده می‌کنند. اگرچه این راهبردها برای برنامه‌های سنتی مناسب هستند، اما برای خزش برنامه‌های غنی شده‌ی وب کارامد نیستند[2]. زیرا هیچ کدام از راهبردها سازوکاری ندارند تا پیش‌بینی کنند کدام رویداد با احتمال بیشتری حالت جدیدی را پیدا می‌کند. برای حل این مشکل در سال ۲۰۱۱، Benjamin و همکارانش [13] اولین نسخه‌ی راهبرد خزش مبتنی بر مدل را ارائه داده‌اند. در این مقاله او از راهبردی به نام Hypercube استفاده کرده است. Choudhary و همکارانش [15] یک راهبرد مبتنی بر مدل دیگر، به نام راهبرد Menu، معرفی کردند. در این مدل رویدادها به دسته‌های مختلف تقسیم بندی می‌شوند و هر دسته اولویت ایستای خود را دارد. در حال حاضر جدیدترین راهبرد ارائه شده برای خزش برنامه‌های غنی شده‌ی وب به صورت کارامد، راهبرد احتمال[8] است[16]. این راهبرد مبتنی بر مدل آماری است و بر پایه‌ی این فرضیه است: رویدادی که در گذشته صفحه‌های جدید بیشتری پیدا کرده است، در آینده نیز با احتمال بیشتری منجر به کشف صفحه‌ی جدید خواهد شد. در جدول ۱ راهبردهای مبتنی برمدل، و ویژگی‌های آن‌ها آورده شده است:

جدول ۱: راهبردهای مبتنی بر مدل

راهبرد	ویژگی
Hypercube [13]	- نخستین نسخه‌ی کاوش مبتنی بر مدل - فرض‌های غیرواقعی در مورد برنامه‌های وب
Menu [15]	- دسته‌هایی با اولویت ایستا - فرض‌های واقعی‌تر، نتایج بهتر - مشاهدات اندک برای دسته‌بندی رویدادها
Probability [16]	- جدیدترین راهبرد ارائه شده - مبتنی بر داده‌های آماری، سابقه‌ی رویداد - فرض‌های بسیار ساده‌تر

۳- راهکار پیشنهادی

قبل از توضیح راهکار پیشنهادی، باید چگونگی عملکرد راهبرد مدل احتمال را بررسی کنیم. در این راهبرد، احتمال یک رویداد (احتمال کشف حالت جدید در اجرای بعدی) را می‌توان با استفاده از رابطه‌ی بیز[9] زیر محاسبه نمود[16]:

$$P(e) = \frac{S(e) + p_s}{N(e) + p_n} \qquad (۱)$$

در این رابطه، S(e) تعداد حالت‌های جدید کشف شده توسط رویداد e است. و N(e) تعداد دفعات اجرای رویداد e است. همچنین به p_s و p_n مقادیر پیش‌فرض نسبت داده می‌شود، تا مقدار احتمال اولیه مشخص گردد.

در دنیای واقعی اغلب برنامه‌های غنی شده‌ی تحت وب شامل تعداد بسیار زیادی حالت هستند. لذا نمی‌توان در زمان معقول همه‌ی حالت‌ها را کشف کرد. علاوه بر آن، هیچ یک از کارهای انجام شده در زمینه‌ی خزش

[6] Model based crawling

[7] Greedy

[8] Probability model

[9] Bayesian

مبتنی بر مدل، با نگاه امنیتی نبوده است و در نتیجه مدل‌های ارائه شده از برنامه‌های وب، برای انجام تست‌های امنیتی و کشف آسیب‌پذیری‌ها بهینه نیستند. هدف خزنده‌های مبتنی بر مدل ارائه شده پیدا کردن سریع‌تر صفحات جدید است. در این مقاله با اعمال راهکارهایی جدید، یک مدل ریسک ایجاد می‌کنیم و خزنده را به سمت پیدا کردن هرچه سریع‌تر صفحه‌های پرخطر هدایت می‌کنیم. که در نتیجه‌ی آن، اسکنر مربوطه قادر خواهد بود تا آسیب‌پذیری‌های بیشتری را در زمان کمتر کشف کند. به منظور ارائه‌ی یک راهبرد خزش مناسب، راهبرد مدل احتمال را به گونه‌ای ارتقا می‌دهیم تا برای تست امنیتی مناسب باشد.

در اینجا می‌خواهیم یک مدل ریسک را برای برنامه‌ی وب ارائه دهیم. در کارهای موجود اولویت اجرا به رویدادی داده می‌شود که سابقه‌ی بهتری در پیدا کردن حالت‌های جدید داشته باشد، اما در مدل پیشنهادی، به منظور پیدا کردن آسیب‌پذیری‌های بیشتر، اولویت اجرا به رویدادی داده خواهد شد که سابقه‌ی بهتری در پیدا کردن صفحه‌های پرخطر داشته است. بدین ترتیب تعداد صفحات پرخطر بیشتری پیدا می‌شوند که در نتیجه‌ی آن با اعمال تست‌های امنیتی می‌توان آسیب‌پذیرهای بیشتری را در زمان کم‌تر پیدا کرد.

در رابطه‌ی (۱)، مقدار S(e) از جمع نتایج اجراهای قبلی به دست می‌آید، که هر اجرا یا منجر به پیدا شدن حالتی جدید شده است (برابر با یک) و یا حالتی تکراری را پیدا کرده است (برابر با صفر). در نتیجه به عنوان مثال S(e) می‌تواند حاصل جمع زیر باشد:

$$S(e) = 1 + 1 + 0 + 1 + \dots$$

در مدل ریسک که ارائه کردیم، هر بار اجرای رویداد منجر به پیدایش حالتی می‌شود که مخاطره‌ی آن حالت عددی بین صفر و یک محاسبه خواهد شد (محاسبه‌ی مخاطره‌ی حالت در ادامه توضیح داده می‌شود). لذا محاسبه‌ی S(e) به عنوان مثال می‌تواند به شکل زیر باشد (مقدار صفر به معنی تکراری بودن صفحه است):

$$S(e) = 0/7 + 0/4 + 0 + 0/6 + \dots$$

مسئله‌ی بعدی که مطرح شده است محاسبه‌ی میزان مخاطره‌ی امنیتی یک صفحه است. چندین رویکرد برای انجام این‌کار وجود دارد که در این پژوهش از رویکرد ایستا استفاده خواهد شد. بدین ترتیب میزان مخاطره‌ی یک صفحه را می‌توان با بررسی کد سمت کاربر[۱۰] آن صفحه محاسبه نمود. به عنوان مثال وجود عناصر پرخطر مانند فیلد کلمه‌ی عبور، یا فیلد بارگذاری فایل در یک صفحه، می‌تواند میزان خطر و آسیب‌پذیری آن صفحه را بالا ببرد. این نوع محاسبه که تنها از بررسی کد صفحه به دست می‌آید را محاسبه‌ی ایستای میزان مخاطره‌ی صفحه می‌نامیم.

۴- پیاده‌سازی و ارزیابی

[۱۰] منظور از کد سمت کاربر کدهایی نظیر HTML، جاوااسکریپت و غیره است که برای مرورگر کاربر ارسال می‌شود و از سمت کاربر به طور کامل قابل دسترس است.

در این قسمت چگونگی پیاده‌سازی مدل ریسک و همچنین برنامه‌های تست بررسی می‌شوند. سپس نحوه‌ی محاسبه‌ی هزینه‌ی اجرای راهبرد و نحوه‌ی ارزیابی آن ارائه شده است.

۴-۱ نحوه‌ی پیاده‌سازی

به منظور پیاده‌سازی راهبردهای مختلف در این مقاله از زبان برنامه‌سازی C#، و کتابخانه‌ی QuickGraph[۱۱] استفاده شده است. در راهبرد مدل ریسک، پس از اجرای هر رویداد، مقدار successCount_Se به صورت زیر تغییر می‌کند:

```
IF (Current page is new) {
   pageRisk = GetCurrentPageRisk();
   lastExecutedEdge.successCount_Se += pageRisk;
}
ELSE {
   lastExecutedEdge.successCount_Se += 0;
}
lastExecutedEdge.Calculate_Pe();
```

بدین ترتیب پس از کشف یک حالت جدید، ابتدا مقدار ریسک این حالت توسط تابع GetCurrentPageRisk() محاسبه می‌گردد و سپس این مقدار به successCount_Se اضافه می‌گردد. این امر سبب می‌شود رویدادهایی که صفحات آسیب‌پذیرتری پیدا می‌کنند، اولویت بالاتری نسبت به سایر رویدادها داشته باشند.

به منظور ارزیابی مدل پیشنهادی، دو برنامه‌ی وب مبتنی بر ای‌جکس طراحی و توسعه داده شده است. اگرچه این برنامه‌ها بزرگ نیستند، اما به گونه‌ای طراحی شده‌اند که منطق کاری آن‌ها مشابه برنامه‌های دنیای واقعی باشد. برنامه‌ی اول به منظور بررسی رفتار خزنده در راهبردهای مختلف نوشته شده است، لذا رویدادهای اضافه در آن قرار نگرفته است تا پاسخ آزمون‌ها بیشتر قابل درک و تحلیل باشد. این برنامه شامل سه صفحه‌ی اصلی است که با دکمه‌های Page1 و Page2 و Page3 می‌توان به آن‌ها دسترسی داشت. در Page1 فهرستی از کالاها قرار دارد که کاربر می‌تواند با کلیک بر روی دکمه‌ی Next Product، مشخصات کالای بعدی را مشاهده نماید. در اینجا ۶ کالا درنظر گرفته شده است. در Page2 چند فرم برای ورود اطلاعات از طرف کاربر وجود دارد. هر فرم دارای آسیب‌پذیری مخصوص به خود است. در اینجا نیز کاربر با کلیک بر روی دکمه‌ی Next Form می‌تواند فرم‌ها را پیمایش کند. در Page3 کاربر می‌تواند اطلاعاتی را در موارد مختلف مشاهده کند. در اینجا نیز دکمه‌ی Next Info برای پیمایش اطلاعات است. برنامه‌ی تست دوم شامل تعداد حالت‌ها و رویدادهای بیشتری است. این برنامه یک فروشگاه اینترنتی ساده را شبیه‌سازی می‌کند که شامل صفحاتی مانند حساب کاربری، جستجو و ارتباط با ما است.

۴-۲ نحوه‌ی محاسبه‌ی هزینه‌ی راهبرد

[1] https://quickgraph.codeplex.com/

در اینجا برای تعیین هزینه‌ی یک راهبرد، بجای محاسبه‌ی زمان اجرای آن، تعداد اجرای رویدادها شمرده خواهند شد. شمارش تعداد اجرای رویدادها از آن جهت قابل قبول است که زمانی که برای اجرای یک رویداد صرف می‌شود، زمان غالب در فرایند خزش است. و این تعداد فقط و فقط به تصمیماتی که راهبرد خزش می‌گیرد بستگی دارد. درواقع این راهبرد خزش است که تعیین کننده‌ی تعداد نهایی اجرای رویدادها است. بدین ترتیب نتیجه‌ی حاصل شده مستقل از سخت‌افزار و تاخیرات شبکه خواهد بود.

همچنین علاوه بر شمارش تعداد رویدادهای اجرا شده، تعداد دفعاتی که خزنده وادار به reset می‌شود نیز باید محاسبه گردد. بدین ترتیب هزینه‌ی کشف حالت‌ها از ترکیب تعداد reset ها و تعداد اجرای رویدادها بدست خواهد آمد. لذا هزینه‌ی کشف حالت‌ها از رابطه زیر محاسبه خواهد شد[16]:

$$\cos t = n_e + n_r \times c_r \qquad (۲)$$

در این رابطه n_e و n_r به ترتیب تعداد اجرای رویدادها و تعداد reset های انجام شده در راهبرد مورد نظر است. c_r هزینه‌ی انجام reset است که با تقسیم میانگین زمان اجرای هر reset بر میانگین زمان اجرای هر رویداد به دست می‌آید.

۴-۳ نحوه‌ی ارزیابی

به منظور ارزیابی روش پیشنهادی، در ابتدا مدل احتمال طبق مقاله‌ی [16] پیاده‌سازی شد. پس از پیاده‌سازی مدل احتمال و کسب نتایج و مشاهدات مطلوب، تغییرات ذکر شده در قسمت سوم، در راهبرد مدل احتمال اعمال شد و مدل ریسک با امکان اولویت‌بندی رویدادها بر حسب ریسک آن‌ها تهیه گردید.

هر یک از راهبردهای موردنظر را می‌توان بر روی بسترهای آزمون اجرا کرد و نتیجه‌ی حاصل شده را با یکدیگر مقایسه کرد. این راهبردها عبارت‌اند از:

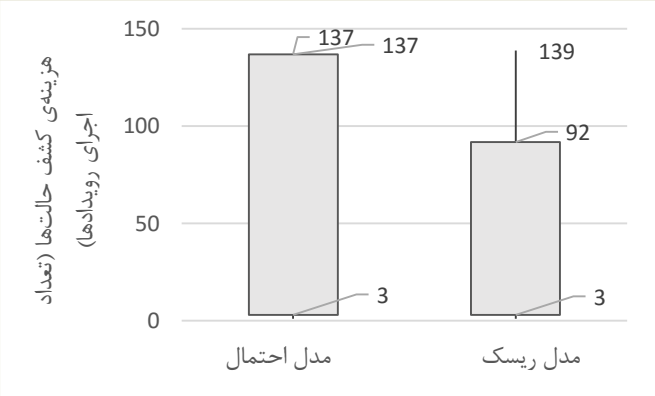
- راهبرد مدل احتمال: در راهبرد احتمال به هر رویداد یک مقدار احتمال نسبت داده می‌شود. رویدادی که سابقه‌ی بهتری در کشف حالت‌های جدید داشته باشد از احتمال بالاتری برخوردار خواهد بود. انتخاب رویداد بعدی برای اجرا بر اساس احتمال آن‌ها انجام می‌گیرد.

- راهبرد مدل پیشنهادی، مدل ریسک: در این راهبرد نیز به هر رویداد یک مقدار احتمال نسبت داده می‌شود. اما رویدادی از احتمال بالاتری برخوردار خواهد بود که سابقه‌ی بهتری در کشف حالت‌های آسیب‌پذیر داشته باشد. در اینجا نیز انتخاب رویداد بعدی برای اجرا بر اساس احتمال آن‌ها انجام می‌گیرد.

۴-۴ ارزیابی راهبرد

راهبرد ارائه شده در این پژوهش ابتدا حالت‌هایی را کشف می‌کند که از نظر امنیتی دارای اهمیت بیشتری باشند. ارزیابی ما نیز مبتنی بر همین موضوع است. درواقع کارایی راهبردها را در کشف حالت‌های آسیب‌پذیر بررسی می‌کنیم. و یا به عبارت دیگر با انجام آزمایشات مشخص می‌کنیم که هر راهبرد با چه هزینه‌ای حالت‌های آسیب‌پذیر را کشف خواهد کرد. در برنامه‌ی تست اول، سه آسیب‌پذیری تعریف شده است: تزریق SQL، XSS، و آپلود فایل بدون محدودیت. در برنامه‌ی تست دوم نیز تعداد بیشتری از همین آسیب‌پذیری‌ها قرار گرفته است. نتایج حاصل‌شده از راهبردهای مختلف در ادامه بر روی دو نمودار مجزا آورده شده و مورد تحلیل قرار گرفته است. محور افقی در نمودارها بیان‌گر راهبردهای مختلف، و محور عمودی بیان‌گر هزینه‌ی مربوط به کشف حالت‌ها است. راهبردی کاراتر است که حالت‌های آسیب‌پذیر را سریع‌تر پیدا کند. بدین منظور هزینه‌ی کشف حالت‌های آسیب‌پذیر در نمودار با رنگ خاکستری مشخص شده است.

تصویر ۱: نمودار مقایسه‌ی راهبردهای مختلف در برنامه‌ی تست اول



تصویر ۲: نمودار مقایسه‌ی راهبردهای مختلف در برنامه‌ی تست دوم

تحلیل نمودارها

در این نمودارها نقطه‌ی شروع هر نمودار (نقطه‌ای با هزینه یک) به معنی هزینه‌ی کشف اولین حالت، و همچنین نقطه‌ی انتهایی نمودار بیان‌گر هزینه‌ی مورد نیاز برای کشف آخرین حالت است. نکته‌ی حائز اهمیت در این نمودار، که هدف اصلی این پژوهش نیز بوده است، قسمت خاکستری رنگ نمودار است. شروع هر یک از قسمت‌های خاکستری نشان‌دهنده‌ی هزینه‌ی کشف اولین حالت آسیب‌پذیر است. و به همین ترتیب نقطه‌ی پایانی بیان‌گر هزینه‌ی کشف آخرین حالت آسیب‌پذیر خواهد بود. به عبارت

دیگر در این نقطه تمام حالت‌های آسیب‌پذیر برنامه کشف شده‌اند. بدین ترتیب همان‌گونه که مشاهده می‌شود راهبرد مدل ریسک توانسته است با هزینه‌ی کمتر تمام حالت‌های آسیب‌پذیر را کشف کند.

راهبرد مدل ریسک به گونه‌ای طراحی شده است که رویدادها را بر اساس ریسک آن‌ها اولویت‌بندی می‌کند. تحلیل نمودار مربوط به این راهبرد در برنامه‌ی تست اول بدین شرح است: در راهبرد مدل احتمال و راهبرد مدل ریسک، رویدادهای Page1 و Page2 و Page3 اجرا می‌شوند. خزنده در حالت Page3 برای ادامه‌ی کار از رویداد موجود در این حالت استفاده می‌کند و آن را اجرا می‌کند. پس از چند بار اجرای رویداد Next Info، و پیمایش کامل حالت Page3، خزنده باید به یکی از صفحات Page1 یا Page2 بازگردد و ادامه‌ی خزش را انجام دهد. در مدل احتمال، هردو رویداد ذکر شده دارای اهمیت یکسان هستند، لذا ممکن است هرکدام از آن‌ها انتخاب شوند. اما در مدل ریسک، رویداد Page2 از اهمیت بالاتری برخوردار خواهد بود. زیرا این رویداد در اجرای قبلی خود حالتی را کشف کرده بود که دارای فرم بوده است، و آن فرم نیز دارای آسیب‌پذیری تزریق SQL بوده است. بدین ترتیب مدل ریسک، رویداد Page2 را برای اجرا انتخاب می‌کند، در حالی که مدل احتمال رویداد Page1 را برگزیده است. پیمایش حالت Page2 قبل از حالت Page1 منجر به کشف سریع‌تر حالت‌های آسیب‌پذیر شده است.

در برنامه‌ی تست دوم نیز مشاهده می‌شود که مدل ریسک توانسته است با هزینه‌ی کمتر، تمام حالت‌های آسیب‌پذیر در برنامه را کشف کند.

۵- نتیجه‌گیری و کارهای آینده

راهبردهای استاندارد خزش، مانند راهبردهای اول‌سطح و اول‌عمق، برای خزش برنامه‌های غنی‌شده‌ی تحت وب مناسب نیستند، زیرا هیچ یک از این راهبردهای مکانیزمی برای پیش‌بینی رفتار رویدادها ندارند. برای حل این مشکل راهبردهای مبتنی بر مدل ارائه شد که از کارایی بالاتری برخوردار هستند. پس از بررسی راهبرد مدل احتمال و راهبردهای دیگر از لحاظ تحلیل‌های امنیتی، مشخص شد که تا کنون هیچ یک از راهبردهای خزش در جهت تست امنیتی بهینه نشده‌اند. بدین جهت در این پژوهش همان‌گونه که تشریح شد، تغییراتی را در راهبرد مدل احتمال اعمال کردیم به گونه‌ای که راهبرد جدید برای انجام تست‌های امنیتی مناسب باشد. در مدل ارائه شده که مدل ریسک نام دارد، اولویت‌بندی رویدادها بر اساس ریسک آنها صورت می‌گیرد، که این امر منجر به پیدا کردن سریع‌تر حالت‌های آسیب‌پذیر می‌شود. از نقاط قوت روش پیشنهادی می‌توان به سادگی روش به دلیل عدم استفاده از روابط پیچیده، و همچنین عدم افزودن سربار اضافه‌ی محسوس اشاره کرد.

یکی از موضوعاتی که باید در کارهای آینده مورد توجه قرار گیرد، استفاده از رویکرد پویا در محاسبه‌ی میزان مخاطره‌ی صفحه‌ها است. بدین ترتیب که به جای استفاده از کدهای سمت کاربر، در طول فرایند خزش، از تست‌های امنیتی نظیر تست فاز یا تست نفوذ برای تشخیص میزان مخاطره‌ی صفحه استفاده کرد.

مراجع

[1] Sandulescu, Mihai. "Techniques for Finding Vulnerabilities in Web Applications." Journal of Mobile, Embedded and Distributed Systems 6, no. 1 (2014): 44-51.

[2] Dincturk, Mustafa Emre, Guy-Vincent Jourdan, Gregor V. Bochmann, and Iosif Viorel Onut. "A model-based approach for crawling rich internet applications." ACM Transactions on the Web (TWEB) 8, no. 3 (2014): 19.

[3] Garrett, Jesse James. "Ajax: A new approach to web applications." (2005): 1-6.

[4] Doupé, Adam, Ludovico Cavedon, Christopher Kruegel, and Giovanni Vigna. "Enemy of the State: A State-Aware Black-Box Web Vulnerability Scanner." In USENIX Security Symposium, pp. 523-538. 2012.

[5] Mirtaheri, Seyed M., Mustafa Emre Dinçktürk, Salman Hooshmand, Gregor V. Bochmann, Guy-Vincent Jourdan, and Iosif Viorel Onut. "A brief history of web crawlers." arXiv preprint arXiv:1405.0749 (2014).

[6] Duda, Cristian, Gianni Frey, Donald Kossmann, Reto Matter, and Chong Zhou. "Ajax crawl: Making ajax applications searchable." In Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on, pp. 78-89. IEEE, 2009.

[7] Mesbah, Ali, Arie van Deursen, and Stefan Lenselink. "Crawling Ajax-based web applications through dynamic analysis of user interface state changes." ACM Transactions on the Web (TWEB) 6, no. 1 (2012): 3.

[8] Mesbah, Ali, Engin Bozdag, and Arie Van Deursen. "Crawling Ajax by inferring user interface state changes." In Web Engineering, 2008. ICWE'08. Eighth International Conference on, pp. 122-134. IEEE, 2008.

[9] Amalfitano, Domenico, Anna Rita Fasolino, and Porfirio Tramontana. "Reverse engineering finite state machines from rich internet applications." In Reverse Engineering, 2008. WCRE'08. 15th Working Conference on, pp. 69-73. IEEE, 2008.

[10] Amalfitano, Domenico, Anna Rita Fasolino, and Porfirio Tramontana. "Experimenting a reverse engineering technique for modelling the behaviour of rich internet applications." In Software Maintenance, 2009. ICSM 2009. IEEE International Conference on, pp. 571-574. IEEE, 2009.

[11] Amalfitano, Domenico, Anna Rita Fasolino, and Porfirio Tramontana. "Rich internet application testing using execution trace data." In Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on, pp. 274-283. IEEE, 2010.

[12] Amalfitano, Domenico, Anna Rita Fasolino, and Porfirio Tramontana. "Techniques and tools for rich internet applications testing." In *Web Systems Evolution (WSE), 2010 12th IEEE International Symposium on*, pp. 63-72. IEEE, 2010.

[13] Benjamin, Kamara, Gregor Von Bochmann, Mustafa Emre Dincturk, Guy-Vincent Jourdan, and Iosif

internet applications crawling: menu and probability models." Journal of Web Engineering 13, no. 3-4 (2014): 243-262.

Viorel Onut. A strategy for efficient crawling of rich internet applications. Springer Berlin Heidelberg, 2011.

[14] Peng, Zhaomeng, Nengqiang He, Chunxiao Jiang, Zhihua Li, Lei Xu, Yipeng Li, and Yong Ren. "Graph-based ajax crawl: Mining data from rich internet applications." In Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on, vol. 3, pp. 590-594. IEEE, 2012.

[15] Choudhary, Suryakant, Mustafa Emre Dincturk, Seyed M. Mirtaheri, Guy-Vincent Jourdan, Gregor V. Bochmann, and Iosif Viorel Onut. "Building rich internet applications models: Example of a better strategy." In *Web Engineering*, pp. 291-305. Springer Berlin Heidelberg, 2013.

[16] Choudhary, Suryakant, Emre Dincturk, Seyed Mirtaheri, Ggregor V. Bochmann, Guy-Vincent Jourdan, and Iosif Viorel Onut. "Model-based rich