**به نام خدا**

دانشکده مهندسی کامپیوتر

# فرم خلاصه «تک‌صـفحه‌ای» پیشنهاد پایان‌نامه کارشناسی ارشد

| | |
|---|---|
| **نام دانشجو: مرتضی ذاکری نصرآبادی** | **شماره دانشجویی: ۹۵۷۲۳۰۸۸** |
| **نام استاد راهنما: دکتر سعید پارسا** | **مرتبه علمی: دانشیار** |
| **عنوان پایان‌نامه (فارسی): تولید خودکار داده آزمون در فازرهای قالب فایل** | |
| **عنوان پایان‌نامه (انگلیسی): Automatic test data generation in file format fuzzers** | |
| **کلید واژه‌ها (۳ الی ۵ مورد): آزمون فازی، داده آزمون، پوشش کد، یادگیری ژرف، شبکه عصبی مکرر.** | |

**شـرح مسئله:**

هدف ارایه روشی کارا جهت یافتن آسیب‌پذیری‌ها در نرم‌افزارهایی مثل PDFخوان‌ها بوده که ورودی آنها فایل با ساختار مشخص و معمولا پیچیده است. در فن آزمون فازی یا فازینگ [1] با ارایه مکرر داده‌های ورودی به نرم‌افزار تحت آزمون، سعی می‌شود تا آسیب‌پذیری‌ها مشخص گردند [2]. در فازینگ قالب فایل، چالش عمده تعیین خودکار ساختار فایل ورودی جهت تولید داده‌های آزمون مناسب است. در این راستا تولید خودکار فایل‌های ورودی باتوجه به معیارهای پوشش کد از اهمیت ویژه‌ای برخوردار است. روش‌های تصادفی و نیز روش‌های اکتشافی مبتنی بر جهش، برای قالب‌های فایل، در عمل پوشش کد ضعیف و سطحی دارند [3]. روش‌های مبتنی بر گرامر برای تولید داده‌های آزمون پیچیده، پوشش کد نسبتاً بهتری را ارایه نموده‌اند؛ اما، این روش‌ها کاملاً خـودکار نیستند. همچنین تولید گرامر به‌صورت دستی نیازمند داشتن مشخصات قالب فایل مربوطه است. لذا، این دسته از روش‌ها در عمل بسیار زمان‌بر، پرهزینه و مستعد خطا هستند [4]. با نگاه به مسائل یاد شده، در این پژوهش سعی به ارایه روشی خودکار جهت تولید داده‌های آزمون مبتنی بر گرامر با هدف بهبود پوشش کد و شناسایی آسیب‌پذیری‌های احتمالی موجود در نرم‌افزارهای با ورودی فایل، خواهد شد. بسیاری از نرم‌افزارهای کاربردی دنیای واقعی ورودی خود را به‌شکل فایل با قالب مشخص می‌پذیرند. از این حیث پیدا کردن خطاها و آسیب‌پذیری‌ها در این دسته از نرم‌افزارها حایز اهمیت است.

**نوآوری پژوهش:**

- خودکارسازی فرایند یادگیری ساختار فایل ورودی و تولید داده‌های آزمون برای آزمون فازی قالب فایل با به‌کارگیری روش‌های جدید یادگیری ژرف.
- بهبود پوشش کد (پوشش دستورات و مسیرهای اجرایی) با تولید داده‌ها مبتنی بر گرامر و بد شکل‌سازی داده‌های ورودی جهت یافتن آسیب‌پذیری‌ها.

**محصول پژوهش و روش ارزیابی آن:**

- **محصول:** یک نرم‌افزار جهت تولید خودکار داده آزمون از روی مدل مولد؛ قابل استفاده در فازرهای قالب فایل.
- **روش ارزیابی:** انجام آزمون فازی روی نرم‌افزار(های) هدف با ورودی فایل، سنجش میزان پوشش کد و مقایسه نتایج حاصله با نتایج موجود از فازرهای قالب فایل مشهور، مثل AFL [5].

**مراجع اصلی پژوهش (تا ۵ مورد):**


[1] B. P. Miller, L. Fredriksen, and B. So, “An empirical study of the reliability of unix utilities,” *Commun. ACM*, vol. 33, no. 12, pp. 32–44, 1990.

[2] M. Sutton, A. Greene, and P. Amini, *Fuzzing brute force vulnerability discovery*, 1st ed. Addison-Wesley, 2007.

[3] S. Rawat, V. Jain, A. Kumar, L. Cojocar, C. Giuffrida, and H. Bos, “VUzzer : Application-aware evolutionary fuzzing,” no. March, 2017.

[4] P. Godefroid, H. Peleg, and R. Singh, “Learn&Fuzz: Machine learning for input fuzzing,” *Microsoft Res.*, 2017.

[5] “American fuzzy lop.” [Online]. Available: http://lcamtuf.coredump.cx/afl/. [Accessed: 11-Oct-2017].


**امضاء استاد راهنما:**

| | |
|---|---|
| **نظر اولیه مدیر گروه:** قابل طرح در گروه ☐ نیاز به اصلاح ☐ | **نظر گروه:** تأیید ☐ تأیید مشروط ☐ رد ☐ |

- **نکته ۱:** این فرم به‌عنوان جلد فرم اصلی پیشنهاد پایان‌نامه استفاده شده و به‌همراه آن تهیه و تحویل داده شود.
- **نکته ۲:** فرم به‌صورتی پر شود که تحت هیچ شرایطی بخش‌های آن تغییر داده نشده و بیش از یک صفحه نشود.

به نام خدا

دانشکده مهندسی کامپیوتر

# فرم پیـشنهاد پروژه پایانی کارشـناسی ارشد

شماره ثبت پیشنهاد:

تاریخ ارائه پیشنهاد به دفتر تحصیلات تکمیلی دانشکده:

## ۱- مشخصات دانشجو

نام و نام‌خانوادگی: مرتضی ذاکری نصـرآبادی — شماره دانشجـویی: ۹۵۷۲۳۰۸۸

گرایش: مهندسی نرم‌افزار — تمرکز: سیـستم

## ۲- مشخصات استادان راهنما و مشاور

نام و نام‌خانوادگی استاد راهنما: دکتر سعید پارسا — مرتبه دانشگاهی: دانشیار

محل خدمت: دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران. — گرایش: مهندسی نرم‌افزار

نام و نام‌خانوادگی استاد راهنما: — مرتبه دانشگاهی:

محل خدمت: — گرایش:

نام و نام‌خانوادگی استاد مشاور: — مرتبه دانشگاهی:

محل خدمت: — گرایش:

## ٤- عنوان پایان‌نامه

**به فارسی: تولید خودکار داده آزمون در فازرهای قالب فایل**

**به انگلیسی: Automatic test data generation in file format fuzzers**

**۵- نوع پروژه:** بنیادی ☐ نظری ☐ کاربردی ☑ توسعه‌ای ☑

**٦- تعداد واحد در نظر گرفته شده پیشنهاد دهنده برای پروژه:** ۶ واحد ☑ ۹ واحد ☐

## ۷- واژگان کلیدی

**به فارسی: آزمون فازی، داده آزمون، پوشش کد، یادگیری ژرف، شبکه عصبی مکرر.**

**به انگلیسی: Fuzz testing, test data, code coverage, deep learning, recurrent neural network**

## ۸- خلاصه پیشنهاد پروژه

آزمون نرم‌افزار بخش مهم، زمان‌بر و پرهزینه‌ای از فرایند توسعه و ساخت هر سیستم نرم‌افزاری را تشکیل می‌دهد که مسئول یافتن خطا[۱]های موجود در آن سیستم بوده و با فنون مختلفی قابل انجام است. *آزمون فازی*[۲] یا *فازینگ*[۳]، یک فن مؤثر آزمون نرم‌افزار به‌منظور کشف زود هنگام خطاها، قبل از تبدیل شدن آنها به آسیب‌پذیری[۴] است. آزمون فازی برای آزمودن و نمایان‌سازی خرابی[۵]ها در نرم‌افزارهای بزرگ‌مقیاس که ورودی‌هایی با ساختار(ها)ی پیچیده می‌پذیرند؛ از قبیل مرورگرهای وب، ویرایشگرهای متن، پخش‌کننده‌های چندرسانه‌ای و غیره، بسیار مناسب ظاهر شده است [1] و [2]. در این فن ورودی‌هایی خاص توسط یک برنامه دیگر، یعنی با روش خودکار، تولید شده و به نرم‌افزار تحت آزمون (SUT)[۶] تزریق می‌شود. برنامه در عین حال، به امید یافتن خطا بر اثر پردازش ورودی تزریق شده، پایش می‌شود. ورودی که به برنامه داده می‌شود نقش داده آزمون را داشته و عامل اصلی نمایان‌سازی خطا(ها)ی احتمالی موجود در برنامه با بردن آن به یک حالت خرابی است. به‌همین علت مهم‌ترین قسمت در فرایند آزمون فازی را می‌توان تولید خودکار داده‌های آزمون دانست، به‌نحوی که بیشترین خطاها و ایرادات از طریق آن شناسایی گردند.

فنون گوناگونی برای آزمون نرم‌افزار وجود دارد؛ از جمله فنون ایستا شامل مرور کد منبع و فنون پویا نظیر اجرای نمادین، اجرای واقعی و تحلیل آلودگی. مزیت آزمون فازی سادگی، خودکاربودن و قابل استفاده بودن در نرم‌افزارهای کاربردی دنیای واقعی است. در عین حال این آزمون برای برنامه‌هایی که به‌عنوان ورودی فایل با ساختار مشخصی را می‌پذیرند، با چالش‌هایی مثل پوشش کد ضعیف و سرعت کم مواجه است؛ زیرا ساختار اغلب فایل‌ها پیچیده است. تولید خودکار این فایل‌ها اگر کاملا تصادفی باشد فقط مسیرهای سطحی و اولیه کد برنامه را هدف آزمون قرار می‌دهد. به‌همین دلیل روش‌های مبتنی بر تولید[۷] ابداع شدند که از روی گرامر به تولید ورودی می‌پردازند. مشکل عدیده این روش‌ها نیز خودکار نبودن مرحله تنظیم گرامر یا قالبی است که بایستی در اختیار تولید کننده مورد آزمون قرار گیرد. تولید دستی گرامر یا قالب برای فایل زمان‌بر، پرهزینه و مستعد خطا است؛ لذا، تلاش در راستای خودکارسازی مرحله تولید داده آزمون در این فازرها می‌تواند منجر به کاهش هزینه و زمان آزمون، پوشش کد بیشتر و در نهایت کشف خطاها و آسیب‌پذیرهای پنهان درSUT شود [3]. در این پژوهش روی ایجاد یک سازوکار عملی برای درک خودکار ساختار فایل‌ها و سپس تولید داده مناسب آزمون فازی از طریق آن متمرکز خواهیم شد که گامی پیـشرو در آزمون نرم‌افزار و کشف خطاها خواهد بود. در این راستا از پیشرفت‌های اخیر در روش‌های یادگیری ژرف[۸] برای یادگیری ساختار فایل ورودی و تولید داده آزمون بهره خواهیم گرفت.

## ۹- مروری بر کارهای انجام شده (پیشـینه‌ی کار)

**شرح پژوهش و اهداف.** راهکارهای آزمون فازی نرم‌افزار برای شناسایی خطاها و آسیب‌پذیری‌ها [4]، [5]، [6] و [7] نیاز به داده‌های خاص آزمون دارند [2] و [8]. ورودی اغلب برنامه‌های کاربردی در قالب یک فایل با ساختار مشخص است. مشکل اساسی در آزمون فازی قالب فایل، پیچیده بودن ساختار ورودی، ساختار کد SUT و به تبع آن تعداد بیش از حد مسیرهای اجرایی در کد برنامه است. از یکسو اگر

---

[۱] fault

[۲] fuzz testing

[۳] fuzzing

[۴] vulnerability

[۵] failure

[۶] software under test

[۷] generation based

[۸] deep learning

بیشتر مسیرهای اجرایی پوشش داده نشوند، نمی‌توان انتظار داشت که آسیب‌پذیری‌ها مشخص گردند؛ از دیگر سوی، آزمودن کلیه مسیرهای اجرایی برنامه بسیار سخت، زمان‌بر و پرهزینه خواهد بود [3] و [9]. روش‌های تصادفی محض و مبتنی بر جهش[1] برای این ساختارها اصلا مناسب نیستند. روش‌های مبتنی بر تولید پوشش کد بالاتری فراهم می‌کنند، اما خودکار نبوده و نیازمند داشتن مشخصه‌های قالبِ فایل هستند. روش‌های اکتشافی و آگاه از برنامه که در حاضر رایج‌اند، برخی مشکلات را حل کرده، اما معمولا کُند هستند.

به نظر می‌رسد خودکارسازی فرایند درک ساختار فایل و سپس ایجاد داده‌ها بر اساس ساختار مورد انتظار برنامه بتواند از هزینه و زمان زیاد روش‌های مبتنی تولید کاسته و در عین حال منجر به افزایش پوشش کد گردد. هدف اصلی در این پژوهش ارایه یک سازوکار برای یادگیری ساختار فایل‌های پیچیده و استفاده از آن در تولید داده آزمون مورد نیاز برای انجام آزمون فازی است. اهداف دیگر عبارتند از: گردآوری مجموعه داده اولیه مناسب برای آزمون فازی قالب فایل، افزایش پوشش کد و نفوذ به مسیرهای اجرایی عمیق و بلأخره یافتن خطاها و آسیب‌پذیری‌های احتمالی نهفته در SUT با فاز کردن مناسب داده‌های تولیدی.

**کارهای مرتبط.** آزمون فازی نخستین بار توسط دکتر میلر[2] و همکاران برای آزمون تعدادی از ابزارهای سیستم‌عامل یونیکس طراحی و اجرا شد [4] که نتیجه موفقیت آمیزی در کشف خطاها داشت. برای خودکارسازی فرایند آزمون فازی، ابزاری به‌نام *فازر*[3] توسعه داده می‌شود. اولین فازرهای ساخته‌شده رویکردی مبتنی بر جعبه سیاه داشتند. برخلاف ساده بودن و نیز ناآگاهی از SUT، این دسته از فازرها مانند Peach، Sulley و Radamsa توانسته‌اند خطاهای بسیاری را در برنامه‌های کاربردی پیدا کنند که تا قبل از این نهفته مانده‌بود [9]. به‌تدریج فازرها در دو رویکرد جعبه سفید [10] و جعبه خاکستری [11] هم مورد توجه قرار گرفتند. به موازات آن روش‌های مختلفی برای تولید خودکار داده آزمون در فازرها استفاده شد؛ به‌نحوی که می‌توان گفت آنچه یک فازر را از فازر دیگر متمایز می‌کند، روش به‌کار رفته در آن برای تولید داده آزمون است. دو شیوه کلی تولید داده آزمون فازی با نام‌های *مبتنی بر تولید* و *مبتنی بر جهش یا جابه‌جایی* شناخته می‌شوند [2].

تولید مبتنی بر جهش با گذر زمان از نگرش کاملا تصادفی و کورکورانه‌ی رویکرد جعبه سیاه به روشی هوشمند در رویکرد جعبه خاکستری ترفیع یافته است. در روش‌های جدید، الگوریتم‌های اکتشافی برای تولید هدفمند و هوشمندانه داده آزمون بعدی به‌کار گرفته شده‌اند. به‌طور خلاصه SUT ابزارگذاری می‌شود و رفتار داده آزمون تزریق شده روی معیاری مثلا پوشش دستورات مشاهده گردیده، سپس از بازخورد این رفتار در تولید داده‌های جدید استفاده می‌گردد. این بازخورد عمدتا شامل میزان خوب بودن ورودی آزمون قبلی است. در این دیدگاه خوب بودن بیشتر به کشف مسیرهای اجرایی جدید تفسیر می‌شود. سیستم فازینگ تکاملی (EFS) [11] از الگوریتم ژنتیک برای جهش داده‌ها روی یک مجموعه داده تصادفی اولیه، استفاده می‌کند. AFL [12] نیز از الگوریتم‌های تکاملی و جهش‌هایی با ترتیب قطعی استفاده می‌کند. به‌طور دقیق‌تر AFL هر مورد آزمونی را که مسیر(های) جدیدی کشف کند، در یک صف نگهداری کرده و آن را برای تولید مورد آزمون بعدی جهش می‌دهد. هیچکدام از این فازرها به معنای واقعی از درون SUT اطلاع ندارند. برای همین منظور روش‌های جدیدتر از ترکیب تحلیل ایستای SUT و فنون مهندسی معکوس با الگوریتم‌های یاد شده بهره برده‌اند [9].

در حوزه فازینگ جعبه سفید فنون اجرای نمادین، اجرای واقعی–نمادین و تحلیل آلودگی پویا برای تعیین مسیرهای جدید و مکان‌های احتمالی خطا وارد فازرها شده است [9] و [13]. در حالی که ترکیب فازینگ و اجرای نمادین رضایت بخش بوده و زمینه تحقیقاتی مورد علاقه‌ای محسوب می‌شود، این فنون با چالش اساسی مقیاس‌پذیری مواجه هستند. یعنی برای برنامه‌های خیلی بزرگ با ورودی‌های پیچیده مناسب نیستند؛ چرا که محدودیت‌های زیاد و پیچیده‌ای تولید می‌شود که حل آنها زمان‌بر یا غیر ممکن است [9]. در این حالت‌ها معمولا فازرهای مبتنی بر گرامر کارآمدتر ظاهر شده‌اند.

کار بر روی تولید داده آزمون خودکار مبتنی بر گرامر در دهه ۱۹۷۰م. شروع گردید و با آزمون مبتنی بر مدل در ارتباط نزدیک هست [3]. بیشتر روش‌های مبتنی بر تولید در آزمون فازی به نحوی گرامر را لحاظ می‌کنند. این روش‌ها اما به‌طور کامل خودکارسازی نشده‌اند. به عبارت دیگر یک گرامر نیاز است که قالب فایل ورودی SUT را تصریح کند. این گرامر معمولا با دست نوشته می‌شود. تلاش‌هایی در جهت خودکارسازی درک و استخراج گرامر و سپس تولید ورودی آزمون از روی آن انجام شده است. مقاله یادگیری و فاز [3] روش جدیدی را برای

---

[1] mutation based

[2] Barton. P. Miller

[3] fuzzer

تولید داده آزمون جهت استفاده در آزمون فازی بر مبنای شکبه‌های عصبی مکرر ژرف[1] و مدل کدگذار–کدگشا[2] ارایه کرده است. در این مقاله ساختار فایل PDF برای یادگیری انتخاب شده است. ایده اصلی، یادگیری یک مدل مولد زبان روی مجموعه‌ای از ویژگی‌های اشیای PDF با داشتن مجموعه‌ای از نمونه‌های اولیه است. مدل کدگذار–کدگشا اجازه یادگیری متن با طول دلخواه را برای پیش‌بینی توالی بعدی کاراکترها، می‌دهد. این مدل در مقایسه با رویکردهای سنتی n-gram که محدود به طول رشته متناهی هستند، بهتر است.

در حال حاضر یادگیری و فاز، تنها بر روی اشیای داده‌ای قالب فایل PDF و در حالت بسیار ابتدایی پیاده‌سازی شده است. این امکان وجود دارد که آن را بر روی سایر بخش‌های فایل PDF یا حتی دیگر قالب‌های فایل نیز اعمال کرد. به‌علاوه الگوریتم فاز بایستی به سمت بدشکل کردن هدفمند ورودی بهبود یابد تا افزون بر دستیابی به پوشش کد بالاتر، امکان کشف خطا هم میسر شود. الگوریتم فعلی، بدون حلقه بازخورد، ناآگاه از برنامه و در نتیجه مبتنی بر یک رویکرد جعبه سیاه بوده که حاکی از امکان بهبود آن در موارد ذکر شده است.

روش‌های یادگیری ژرف به تازگی، با پیشرفت سخت افزارها و برداشتن برخی مشکلات نظری بر سر راه آموزش شبکه‌های عصبی ژرف فراگیر شده است [14]. در پژوهش‌های اخیر تمایل زیادی به استفاده از این زمینه برای تحلیل و تولید برنامه‌ها به‌وجود آمده‌است. به بیان بهتر شبکه‌های عصبی ژرف در انجام وظایف ساده برای انسان، سخت برای ماشین، مثل وظایف حوزه پردازش زبان طبیعی بسیار امیدوار کننده ظاهر شده‌اند. یادگیری ساختار فایل را می‌توان جزو این رده از وظایف به‌شمار آورد که البته یادگیری به این شیوه، نو محسوب می‌شود. برای اعمال آزمون فازی، این یادگیری به تنهایی کافی نیست و باید بتوان ورودی‌های خوش–شکل تولید شده را فاز (بد–شکل) نیز کرد. در نگاه اول اهداف آزمون فازی و یادگیری با یکدیگر در تناقض هستند. یادگیری بهتر یعنی تولید خوش–شکل‌تر داده‌های ورودی و فازینگ بهتر یعنی تولید داده‌های مخرب. هر الگوریتمی که ارایه می‌گردد در نهایت باید با این چالش به‌نحو مقتضی کنار بیاید.

**فرضیه‌ها.** چندین فرضیه برای این پژوهش در نظر گرفته شده است، که تدوین سازوکارهایی برای پاسخ به آنها از اهدافِ پژوهش پیشرو است:

- استفاده از فنون یادگیری ژرف بالاخص شبکه‌های عصبی مکرر ژرف در یادگیری خودکار ساختار فایل بسیار مؤثر خواهد بود.
- استفاده از تولید مبتنی بر گرامر منجر به رسیدن به مسیرهای اجرایی جدید و بهبود میزان پوشش کد آزمون فازی در برنامه‌هایی که فایل با ساختار پیچیده را به عنوان ورودی می‌پذیرند، می‌گردد.
- خودکارسازی کامل فرایند آزمون فازی مبتنی بر گرامر با ترکیب مدل یادگیری و روش‌های فازینگ یا همان بد–شکل‌سازی به ورودی، به خوبی میسر می‌شود.
- در نهایت امکان کشف خطاها و آسیب‌پذیری‌های احتمالی موجود در SUT به صورت کاملا خودکار فراهم می‌شود.

در مجموع به نظر می‌رسد آزمون فازی ترکیب شده با یادگیری ژرف، برای بهبود دیگر پارامترهای موجود در این فن آزمون نیز مؤثر باشد. [15]. آزمون فازی به نوعی یک آزمون امنیت و نفوذ به برنامه محسوب می‌شود که برای کشف خطاها و آسیب‌پذیری‌ها هم در جهت اصلاح آنها و هم برای بهره‌برداری از آنها می‌تواند مورد استفاده قرار بگیرد. از این حیث پرداختن به روش‌های جدید کشف آسیب‌پذیری و بهبود روش‌های پیشین مهم و قابل توجیه است.

## مراجع.


[1] M. Sutton, A. Greene, and P. Amini, *Fuzzing brute force vulnerability discovery*, 1st ed. Addison-Wesley, 2007.

[2] A. Kettunen, “Test harness for web browser fuzz testing,” University of Oulu, 2014.

[3] P. Godefroid, H. Peleg, and R. Singh, “Learn&Fuzz: Machine learning for input fuzzing,” *Microsoft Res.*, 2017.

[4] B. P. Miller, L. Fredriksen, and B. So, “An empirical study of the reliability of unix utilities,” *Commun. ACM*, vol. 33, no. 12, pp. 32–44, 1990.

[5] B. P. Miller *et al.*, “Fuzz revisited - A re-examination of the reliability of unix utilities and services,” *October*, vol. 1525, no. October 1995, pp. 1–23, 1995.

[6] J. E. Forrester and B. P. Miller, “An empirical study of the robustness of Windows NT applications using random testing,” *Proc. 4th USENIX Wind. Syst. Symp.*, no. August, pp. 59–68, 2000.

[7] B. P. Miller, G. Cooksey, and F. Moore, “An empirical study of the robustness of MacOS applications


[1] deep recurrent neural networks

[2] encoder-decoder

using random testing," *Proc. 1st Int. Work. Random Testing, RT'06*, vol. 2006, no. March 2017, pp. 46–54, 2006.
[8] S. Veggalam, S. Rawat, I. Haller, and H. Bos, "IFuzzer: An evolutionary interpreter fuzzer using genetic programming," pp. 581–601, 2016.
[9] S. Rawat, V. Jain, A. Kumar, L. Cojocar, C. Giuffrida, and H. Bos, "VUzzer : Application-aware evolutionary fuzzing," no. March, 2017.
[10] P. Godefroid, A. Kiezun, and M. Y. Levin, "Grammar-based whitebox fuzzing," *ACM SIGPLAN Not.*, vol. 43, no. 6, p. 206, 2008.
[11] J. D. Demott, "Revolutionizing the field of grey-box attack surface testing with evolutionary fuzzing," 2007.
[12] "American fuzzy lop." [Online]. Available: http://lcamtuf.coredump.cx/afl/. [Accessed: 11-Oct-2017].
[13] N. Stephens *et al.*, "Driller: Augmenting fuzzing through selective symbolic sxecution," *Ndss*, no. February, pp. 21–24, 2016.
[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.
[15] M. Rajpal, W. Blum, and R. Singh, "Not all bytes are equal: Neural byte sieve for fuzzing," pp. 1–10, 2017.

## ۱۰- روش(ها) و مراحل انجام پروژه

طرح پیشنهادی این پژوهش به‌طور قطع از مرجع [3] ایده گرفته است. همچنین ایرادشناسی اخیر موجود در [9] به شکل‌گیری موضوع کمک شایانی کرد. مرجع [1] در بیان مفاهیم و اصول اولیه آزمون فازی و فازرهای مختلف حایز اهمیت است. مرجع [14] در اقتباس مفاهیم یادگیری ژرف استفاده می‌گردد. مراجع [4] الی [7] حاصل کار ابداع کنندگان فن آزمون فازی بوده و از حیث نظری و عملی ارزشمند هستند. مرجع [2] یک راهکار خودکارسازی کامل را برای آزمون فازی مرورگرهای وب ایجاد کرده که معماری کاملا پیمانه‌ای داشته و امکان جایگزینی پیمانه‌های مختلف در آن به خوبی فراهم شده است. مراجع [8]، [11] و [12] فازرهای مبتنی بر جهشِ اکتشاف‌محور هستند. مرجع [10] آزمون فازی مبتنی بر گرامر را بحث می‌کند و بلأخره در مرجع [15] که به تازگی منتشر شده از روش‌های یادگیری ماشینی برای انتخاب مکان جهش بایت‌ها استفاده شده است.

مراحل کلی انجام این پژوهش به‌صورت زیر است. هرچند جزئیات مربوطه در روند انجام پروژه بیشتر مشخص می‌شوند، اما امیدوار هستیم این گام‌ها آن‌گونه که در ذهن ماست تحقق پذیرند. یک ریززمانبندی منطبق بر مراحل ذیل، در بخش ۱۱ این طرح پیشنهادی به صورت مصور ارایه شده است.

۱. ابتدا مفاهیم اولیه مربوط به آزمون فازی بررسی می‌شوند. فازرهای لبه پژوهش قالب فایل، مانند AFL به‌صورت عملی راه‌اندازی شده و SUTهای مناسبی (دارای ساختار ورودی پیچیده) مثل MuPDF روی بستر آن آزموده می‌شوند. میزان پوشش کد اندازه‌گیری و گزارش می‌شود.
۲. مفاهیم اولیه مربوط به یادگیری ژرف و شبکه‌های عصبی ژرف و پیاده‌سازی مدل‌های آن در چهارچوب‌هایی مثل Keras بررسی می‌شود. به خصوص مدل یادگیری کدگذار-کدگشا که برای توالی‌های با طول متفاوت مثل یک ساختار فایل مناسب است باید کاملا مطالعه گردند.
۳. مجموعه داده آموزشی برای قالب فایل (پیکره‌ای از PDFها) جمع‌آوری خواهد شد و یک مدل برای آموزش آن طراحی و پیاده‌سازی می‌گردد. فرایند آموزش و ارزیابی انجام می‌شود. مدل مستقل از یک پیکره خاص است و می‌تواند برای قالب‌های دیگر نیز در صورت وجود زمان کافی مهیا شود.
۴. یک الگوریتم فاز جدید طراحی می‌گردد و با استفاده از مدل یادگیری شده داده‌های آزمون برای این انجام فازینگ تولید می‌شود. SUT با استفاده از داده‌های جدید فاز شده و نتایج گزارش می‌شود. فازرها معمولا در تشخیص خرابی‌های حافظه مثل سرریز میانگیر مناسب ظاهر می‌شوند. خرابی‌های حافظه بیشتر در مقادیر مرزی رخ می‌دهد. برای مثال مقادیر عددی و رشته‌ای خیلی بزرگ یا خیلی کوچک. الگوریتم فاز ارایه شده در این مرحله باید راهکاری برای جایگزینی مقادیر مرزی، جهت آشکارسازی خطا داشته باشد.
۵. یافته‌های حاصله از آزمایش‌های تجربی مدل مرحله ۴ با نتایج مرحله ۱ مقایسه و گزارش می‌گردد. در صورت امکان این نتایج با نتایج مندرج در مراجع مشابه نیز مقایسه و علل تفاوت بیان خواهد شد. در نهایت کلیه مراحل در قالب پایان‌نامه مستندسازی می‌شوند.

## ۱۱- زمان‌بندی اجرای پژوهش

تاریخ شروع: *نیمه دوم آذرماه ۱۳۹۶* تاریخ اتمام (پیش‌بینی شده): *نیمه دوم شهریورماه ۱۳۹۷*

| مراحل اجرا / زمان (ماه) | ۱ | ۲ | ۳ | ٤ | ٥ | ٦ | ۷ | ۸ | ۹ | ۱۰ | ۱۱ | ۱۲ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **بررسی مفاهیم اولیه فازینگ و آزمایش‌های AFL** | | | | | | | | | | | | |
| **بررسی چارچوب keras و نحوه ایجاد مدل یادگیر** | | | | | | | | | | | | |
| **گردآوری مجموعه آموزش مناسب (پیکره فایل‌ها)** | | | | | | | | | | | | |
| **ایجاد و آموزش مدل عصبی یادگیرنده ساختار فایل** | | | | | | | | | | | | |
| **طراحی و ساخت الگوریتم فاز + انجام آزمون فازی** | | | | | | | | | | | | |
| **ارزیابی روش، مقایسه یافته‌ها و نتیجه‌گیری** | | | | | | | | | | | | |
| **نگارش متن پایان‌نامه، اصلاحات جزیی و غیره.** | | | | | | | | | | | | |

## ۱۲- وسایل و تجهیزات لازم

سیستم فیزیکی یا مجازی با مشخصات حداقلی زیر جهت انجام آزمایش‌های پیش‌بینی شده (عملیات آموزش شبکه ژرف و آزمون فازی):

**CPU:** Intel Core i7 or Intel Xeon Series
**GPU:** NVIDIA GeForce Series
**RAM:** 16 GB DDR4
**SSDs:** 4x Intel DC S3500 480GB

## ۱۳- اعتبار ریالی و ارزی لازم

در حال حاضر اعتبار ریالی و ارزی برای این پژوهش درنظر گرفته نشده است.

## ۱٤- اظهار نامه دانشجو

- قبول می نمایم که این پیشنهاد پروژه، مدارک ضمیمه، آثار و نتایج مادی و معنوی حاصل از انجام پژوهش به دانشکده مهندسی کامپیوتر دانشگاه علم‌وصنعت ایران متعلق بوده و مجاز نیستم بدون موافقت دانشکده، اطلاعات در رابطه با پژوهش را به دیگری واگذار نمایم.
- اظهار می دارم که با توجه به اطلاعات و بررسی های اینجانب تا این تاریخ تحقیق پیشنهادی اصیل بوده و بجز مواردی که در متن پیشنهاد اشاره کرده‌ام بطور کلی و یا جزئی انجام نشده است.
- متعهد می شوم که در مدت اجرای پروژه، بطور تمام وقت انجام وظیفه نموده و بدون مجوز تحصیلات تکمیلی دانشکده و دانشگاه از مرخصی تحصیلی استفاده ننمایم.

**نام و نام‌خانوادگی:** **امضاء:** **تاریخ:**

**۱۵- نظر استادان راهنما**

**نام و نام خانوادگی: دکتر سعید پارسا** **امضاء:** **تاریخ:**

**نام و نام‌خانوادگی:** **امضاء:** **تاریخ:**

**۱٦- نظر مدیر گروه از طرف گروه مربوطه**

**نام و نام‌خانوادگی:** **امضاء:** **تاریخ:**

---

**۱۷- نظر مدیر تحصیلات تکمیلی**

**نام و نام‌خانوادگی:** **امضاء:** **تاریخ:**

---

**۱۸- رئیس دانشکده**

**نام و نام‌خانوادگی:** **امضاء:** **تاریخ:**