استخراج توکن‌های رمزنگاری جستجوپذیر از ترافیک فشرده‌شده HTTPS به‌منظور بازرسی محتوایی

زینب اسکندری[۱]، دانشجوی کارشناسی ارشد؛ مرجان کائدی[۲]، استادیار؛ علی بهلولی[۳]، استادیار

۱- دانشکده مهندسی کامپیوتر - دانشگاه اصفهان – اصفهان – z.eskandari.2015@eng.ui.ac.ir
۲- دانشکده مهندسی کامپیوتر دانشگاه اصفهان -– اصفهان - kaedi@eng.ui.ac.ir
۳- دانشکده مهندسی کامپیوتر - دانشگاه اصفهان – اصفهان - bohlooli@eng.ui.ac.ir

چکیده: بازرسی محتوایی بسته‌های شبکه امری ضروری برای جلوگیری از حملات تحت شبکه است. در حجم زیادی از ترافیک وب، از پروتکل HTTPS استفاده می‌شود. برای بازرسی محتوایی ترافیک HTTPS، از رمزنگاری جستجوپذیر استفاده می‌شود تا این امر بدون رمزگشایی ترافیک HTTPS و با حفظ محرمانگی انجام شود. برای رمزنگاری جستجوپذیر باید از ابرمتن آشکار، توکن استخراج شود. از طرفی درصد قابل توجه‌ای از ترافیک HTTPS، قبل از رسیدن به لایه SSL فشرده می‌شوند که شامل دو مرحله فشرده‌سازی LZ77 و کدگزاری هافمن است. برای ترافیک فشرده‌شده، توکن‌های مورد نیاز برای رمزنگاری جستجوپذیر، بدون فشرده‌گشایی ابرمتن قابل استخراج نیستند. در این شرایط، استخراج توکن با پیمایش ماشین متناهی نامعین(NFA) بر ابرمتن فشرده‌گشایی شده انجام می‌گیرد. هدف این پژوهش کاهش پیچیدگی زمانی بالای پیمایش NFA است. در روش پیشنهادی، به جای فشرده‌گشایی کامل ابرمتن، ابتدا با اعمال کدگشایی هافمن روی آن، ابرمتن فشرده‌شده با LZ77 به دست می‌آید. سپس با استفاده از اشاره‌گرهای LZ77، توکن‌های تکراری در ابرمتن تشخیص داده می‌شوند و می‌توان در NFA ازروی آن‌ها پرید تا استخراج توکن سرعت یابد. ارزیابی‌ها نشان می‌دهد که روش پیشنهادی، با پرش از ۴۴ درصد کاراکترها، زمان استخراج توکن‌ها را ۶۵ درصد نسبت به روش فشرده گشایی کامل، کاهش می‌دهد.

واژه‌های کلیدی: بازرسی محتوایی بسته، HTTPS فشرده‌شده، رمزنگاری جستجوپذیر، LZ77، ماشین متناهی نامعین.

Searchable Encryption Token Extraction for Deep Packet Inspection over Compressed HTTPS

Zeinab Eskandari[1],MSc Student; Marjan Kaedi[2], Assistant Professor; Ali Bohlooli[3], Assistant Professor

1-Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran, Email: z.eskandari.2015@eng.ui.ac.ir
2-Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran, Email: kaedi@eng.ui.ac.ir
3-Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran, Email: bohlooli@eng.ui.ac.ir

Abstract: Deep packet inspection is an unavoidable task to prevent the web-based attacks. HTTPS traffic constitutes a significant percentage of web traffic. In order to perform the deep packet inspection on HTTPS traffic, searchable encryption is used. Using searchable encryption, there is no need to decrypt the packet contents; hence the privacy is preserved. Besides, a significant percentage of HTTPS traffic is compressed before reaching the SSL layer. Compression consists of two stages: LZ77 compression and Huffman encoding. For compressed traffic, required tokens for performing searchable encryption are not accessible and the full decompression should be applied to access the tokens. In this case, token extraction is done by parsing a nondeterministic finite automata (NFA) on the decompressed contents. The goal of this paper is to decrease the time complexity of NFA parsing process. In the proposed method, instead of full decompression of the compressed traffic, at first Huffman decoding is applied on the packets to access the LZ77 compressed content. Then, the LZ77 pointers can be used to recognize the repeated substrings and to skip the NFA parsing process. Hence, the token extraction process is accelerated. The evaluations show that the proposed method decreases 65% of time of compressed HTTPS token extraction by skipping the 44% of characters.

Keyword: Deep packet inspection, Compressed HTTPS, Searchable encryption, LZ77, Nondeterministic finite automata.

تاریخ ارسال مقاله: ۱۳۹۶/۰۴/۱۴
تاریخ اصلاح مقاله: ۱۳۹۶/۰۸/۱۷
تاریخ پذیرش مقاله: ۱۳۹۷/۰۱/۱۰
نام نویسنده مسئول: مرجان کائدی
نشانی نویسنده مسئول: ایران - اصفهان – خیابان هزارجریب - دانشگاه اصفهان - دانشکده مهندسی کامپیوتر.

۱- مقدمه

استفاده از برنامه‌های کاربردی بازرسی محتوایی بسته‌ها[۱] (DPI)، یک امر ضروری برای جلوگیری از حملات تحت شبکه و بدافزارها است. DPI به‌منظور بررسی دقیق محتوای بسته (و در برخی حالات، سرآیندهای آن) طراحی شده‌است و محتوای بسته را از لایه کاربرد تا لایه شبکه مورد بازرسی محتوایی قرار می‌دهد. بازرسی محتوایی براساس قوانین منطقی صورت می‌گیرد؛ قوانین از پایگاه‌دانشی از رفتارهای مخرب ساخته می شوند. برای مثال می‌توان قوانین را از مجموعه قوانین منبع-باز Snort[۱] دریافت کرد. قوانین براساس عبارات منظم[۲] با محتوای بسته مطابقت پیدا می‌کنند. بنابراین DPI دارای مجموعه‌ای از قوانین است و ترافیک عبوری را با این قوانین تطبیق می‌دهد و در مورد عبور دادن/عبور ندادن و یا اخطار و ثبت گزارش ترافیک، تصمیم مناسبی می‌گیرد. با توجه به ضروری بودن تشخیص و جلوگیری از حملات تحت شبکه، این گونه ابزارها باید قادر به بازرسی محتوایی انواع ترافیک شبکه باشند.

به منظور حفظ امنیت و محرمانگی، در سال‌های اخیر رمزنگاری ترافیک شبکه، رشد فزاینده‌ای پیدا کرده است. رمزنگاری به عنوان بخشی از استانداردهای برنامه‌های کاربردی عرضه می‌شود. در این میان استفاده از پروتکل‌های لایه دریچه امن[۳] (SSL) و امنیت لایه انتقال[۴] (TLS) برای رمزنگاری ترافیک شبکه، رشد چشمگیری داشته که شرکت گوگل مشوق این موضوع بوده است. به‌منظور امن کردن پروتکل HTTP با استفاده از SSL، پروتکل امن HTTPS طراحی شد که از آن به عنوان یک امتیاز مثبت در وبگاه‌ها استفاده می‌شود و امنیت، محرمانگی، حریم‌خصوصی و صحت را برای کاربران به همراه دارد. طبق آخرین آمار موجود، در اولین ماه سال ۲۰۱۷ بیشتر از ۷۵ درصد از صفحات وب از HTTPS استفاده می‌کنند [۲]. در این میان بیش از نیمی از ترافیک وب، به صورت ترافیک فشرده‌شده HTTPS هستند که ابزارهای بازرسی محتوایی ترافیک شبکه فعلی، قادر به پشتیبانی کردن این نوع ترافیک نیستند.

ترافیک فشرده‌شده HTTPS حاصل صفحات وبی است که فشرده شده‌اند و سپس با پروتکل HTTPS رمزنگاری و ارسال شده‌اند. برای فشرده‌سازی صفحات وب، از الگوریتم GZIP استفاده می‌شود که شامل دو مرحله است. در مرحله اول فشرده‌سازی LZ77 انجام می‌شود و در مرحله بعدی، کدگذاری هافمن[۵] اعمال می‌گردد. به منظور افزایش سرعت بارگیری صفحات وب، فشرده‌سازی ابرمتن در سمت سرویس‌دهنده وب و بازسازی آن در مرورگر پیاده‌سازی می‌شود. سرویس دهنده‌های وب از قبیل یاهو، گوگل، یوتیوب و فیسبوک نیز از این فناوری پشتیبانی می‌کنند. همچنین استفاده از فناوری فشرده‌سازی ابرمتن منجر به کاهش ۵۰ تا ۹۵ درصدی محتوای وب می‌شود. در حال حاضر، حدود ۷۱ درصد از وبگاه‌ها از فناوری فشرده‌سازی ابرمتن[۶] استفاده می‌کنند [۴،۳]. بنابراین ترافیک فشرده‌شده HTTPS به منظور بهره گیری از مزایای هر دو بخش تشکیل‌دهنده‌اش (یعنی فشرده سازی و رمزنگاری)، مورد توجه قرار گرفته است و در حال حاضر۵۳ درصد از ترافیک وب، به صورت ترافیک فشرده‌شده HTTPS می‌باشد [۴،۲]. با توجه به این درصد قابل توجه، نمی‌توان بازرسی محتوایی این ترافیک را نادیده گرفت.

اولین پژوهشی که در زمینه بازرسی محتوایی ترافیک رمزشده عبوری از شبکه، با حفظ محرمانگی و حریم خصوصی کاربران، صورت گرفت، توسط شری و همکارانش در سال ۲۰۱۵ انجام شده است[۵]. این پژوهشگران برای روش پیشنهادی خود نام BlindBox را انتخاب کردند. این روش برای پروتکل‌هایی که از طریق SSL رمز می‌شوند، طراحی شده است و در آن از پروتکل رمزنگاری جستجوپذیر[۷] برای تطبیق قوانین با محتوای بسته استفاده می‌شود [۵].

قبل از اینکه ترافیک به لایه SSL برسد و رمز شود، پروتکل رمزنگاری جستجو پذیر اجرا می‌شود. این پروتکل با استخراج توکن[۸] و رمزکردن آن‌ها این امکان را فراهم می‌کند تا موجودیت‌های مجاز بتوانند در این توکن‌ها جستجو انجام دهند. توکن‌ها پس از استخراج رمزنگاری می‌شوند و به صورت موازی با ترافیک HTTPS ارسال می‌شوند. پاسخ جستجو در توکن‌های رمزشده به صورت بله یا خیر است؛ یعنی در این جستجو تشخیص داده می‌شود آیا کلید واژه مورد جستجو، در توکن‌های رمزشده وجود دارد یا خیر. بدین ترتیب بازرسی محتوایی با حفظ محرمانگی ترافیک صورت می‌گیرد.

روش BlindBox [۵] برای حالتی طراحی شده است که صفحات وبِ ارسالی/دریافتی قبل از ورود به لایه SSL فشرده نشده باشند. اگر صفحات وبی که قرار است با پروتکل SSL رمزنگاری و ارسال شوند، فشرده‌شده باشند، معادلات پژوهش شری و همکارانش قابل استفاده نخواهد بود و BlindBox قادر نخواهد بود به درستی عملیات بازرسی محتوایی ترافیک را انجام دهد. زیرا در پژوهش شری و همکارانش [۵]، فشرده بودن ابرمتن درنظر گرفته نشده است و محتوای متنی فشرده‌شده صفحات وب با روش‌های رمزنگاری جستجوپذیر، قابل جستجو نیست.

بنابر توضیحات مذکور، برای حل مسئله بازرسی محتوایی ترافیک فشرده‌شده HTTPS، باید به مسئله استخراج توکن‌های رمزنگاری جستجوپذیر از ابرمتن فشرده پاسخ داد. در این پژوهش هدف استخراج توکن، از ابرمتن فشرده می‌باشد. روش اولیه ما برای حل این مسئله، فشرده‌گشایی کامل ابرمتن، سپس استخراج توکن است. این روش را «Blindbox مبتنی بر فشرده‌گشایی کامل» می‌نامیم. در روش Blindbox [۵] از متن آشکار (غیرفشرده و غیررمز) توکن استخراج می‌شود. روش پیشنهادی اولیه این پژوهش از همان روش Blindbox [۵] برای استخراج توکن استفاده می‌کند با این تفاوت که قبل از استخراج، ابرمتن فشرده گشایی کامل می‌شود. اما این روش از نظر زمان کارا نیست. دلیل این موضوع در بخش ۴ بحث خواهد شد. در این مقاله، روشی پیشنهاد می شود که در آن به منظور کاهش زمان استخراج توکن، به جای فشرده گشایی کامل (که شامل دو مرحله فشرده‌گشایی هافمن و رمزگشایی LZ77 است)، تنها کدگشایی هافمن بر روی ترافیک انجام شود. سپس الگوریتمی ارائه می‌گردد که بتواند توکن‌ها را از روی متن فشرده‌شده

LZ77 استخراج کند. این کار با استفاده از ماشین متناهی نامعین[۹] (NFA) و پرش از روی توکن‌های تکراری انجام خواهد شد.

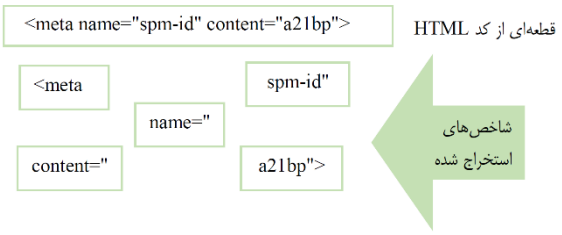
در ادامه در بخش ۲ مفاهیمی چون رمزنگاری جستجوپذیر، نحوه استخراج توکن‌های آن و الگوریتم فشرده‌سازی ابرمتن معرفی خواهند شد. در بخش ۳ پژوهش‌های پیشین مرتبط با مسئله پژوهشی این مقاله معرفی خواهند شد. سپس در بخش ۴ چالش و رهیافت مسئله پژوهشی مطرح می‌شود. روش پیشنهادی این مقاله برای استخراج بهینه توکن‌ها از ابرمتن فشرده شده به تفصیل در بخش ۵ توضیح داده می‌شود. بخش‌های ۶ و ۷ به بررسی و ارزیابی روش پیشنهادی پرداخته می‌شود. در نهایت، در بخش ۸ پژوهش انجام شده نتیجه‌گیری می‌شود.

۲- پیش‌زمینه

در این بخش، ابتدا مفهوم توکن در رمزنگاری جستجوپذیر و همچنین نحوه استخراج توکن‌ها معرفی می‌شوند. سپس الگوریتم فشرده‌سازی ابرمتن معرفی می‌شود.

۲-۱- توکن‌ها در رمزنگاری جستجوپذیر

در رمزنگاری جستجوپذیر برای بازرسی محتوایی ترافیک رمز شده، لازم است که توکن‌ها از محتوای آشکار بسته، استخراج شوند. همانطورکه قبلاً بیان شد، منظور از محتوای آشکار، محتوای غیرفشرده و غیررمز می باشد. بنابراین استخراج توکن قبل از اینکه ابرمتن به لایه SSL ارسال شود، انجام می‌شود. به واسطه این توکن‌های استخراج شده، جستجو بر روی ترافیک رمزشده صورت می‌گیرد. برای مثال در شکل ۱، قطعه‌ای از کد HTML نشان داده شده است که توکن‌های معناداری از آن استخراج شده است.



شکل ۱: قطعه کد HTML وتوکن‌های استخراج شده از آن

کدهای HTML از تگ‌ها و مشخصه‌های[۱۰] تعریف شده‌ای تشکیل شده‌اند. بنابراین باید برای استخراج توکن‌های معنادار، تگ‌ها و مشخصه‌ها تشخیص داده شوند. در بعضی از قسمت‌های یک کد HTML رشته‌هایی غیر از این تگ‌ها و مشخصه‌ها وجود دارند که آن‌ها نیز باید به نحوی استخراج شوند. برای مثال در شکل ۱ مشاهده می‌شود که دو توکن "smp-id و <"a21bp جز تگ‌ها و مشخصه‌ها نیستند، ولی به نحو معناداری استخراج شده‌اند. برای اینکه بتوان توکن‌ها را به این صورت تولید کرد، ماشین حالتی تعریف می‌شود که تگ‌ها و مشخصه‌های کد HTML را پذیرش کند و برای توکن‌های غیر از این دو دسته نیز حالت هایی در نظر داشته باشد.

۲-۲- چگونگی استخراج توکن‌ها

ماشین‌های متناهی نامعین و معین (NFA و DFA[۱۱])، ماشین‌های حالتی هستند که برای پردازش رشته‌ها به کار می‌روند[۶]. با توجه به اینکه استخراج توکن نوعی پردازش رشته است، ماشین‌های NFA و DFA را می‌توان در شرایط مختلف برای استخراج توکن‌ها به‌کار برد.

ماشین حالت DFA سرعت اجرای بالاتری نسبت به ماشین حالت NFA دارد. اما از طرفی حافظه قابل توجهی برای ذخیره ماشین حالت نیاز دارد[۷]. این موضوع برای ماشین‌های کوچک به سرعت اجرا لطمه ای وارد نمی‌کند. اما زمانی که حافظه مورد نیاز ماشین بیشتر از حافظه نهان سیستم[۱۲] باشد، ماشین حالت در حافظه اصلی ذخیره می‌شود. با توجه به اینکه زمان مورد نیاز برای واکشی حافظه اصلی از زمان واکشی حافظه نهان سیستم بسیار زیادتر است، زمان مورد نیاز برای پویش ماشین حالت افزایش می‌یابد [۸]. از این موضوع در بخش ۲-۵ برای استخراج توکن‌ها استفاده خواهد شد.

۲-۳- الگوریتم فشرده‌سازی ابرمتن

در این مقاله استخراج توکن‌ها از ترافیک فشرده شده HTTPS صورت می‌گیرد. ترافیک فشرده‌شده HTTPS، ترافیک HTTPS است که در سرآیند بسته HTTP آن content-encoding=GZIP باشد و بدنه بسته HTTP آن، با الگوریتم GZIP فشرده‌شده باشد؛ سپس بسته HTTP به لایه SSL/TLS داده می‌شود تا رمزنگاری صورت گیرد [۹]. عملیات استخراج توکن‌ها باید قبل از رمزنگاری لایه SSL انجام شود[۵]. همانطورکه گفته شد این بسته‌ها قبل از اینکه به لایه SSL برسند، فشرده هستند و باید از این محتوای فشرده توکن‌های معنادار استخراج شوند. در این بخش الگوریتم GZIP معرفی می‌شود.

فهرستی از الگوریتم‌های فشرده‌سازی که برای فشرده‌سازی ابرمتن استفاده می‌شوند، توسط IANA [۱۳] معرفی شده‌اند. اما از میان این فهرست دو مورد DEFLATE [۱۰] و GZIP [۹] بیشتر مورد توجه قرار گرفته‌اند، به طوریکه GZIP، ۹۸/۹ درصد و DEFLATE، ۱/۱ درصد را به خود اختصاص داده‌اند [۱۱]. الگوریتم فشرده‌سازی GZIP نیز از دو الگوریتم فشرده‌سازی هافمن [۱۲] و LZ77 [۱۳] تشکیل شده است. در الگوریتم GZIP، ابتدا متن آشکار توسط الگوریتم فشرده‌سازی LZ77 فشرده می‌شود. سپس با استفاده از الگوریتم هافمن کدگذاری می‌شود.

الگوریتم LZ77 رشته‌های تکراری در متن را با استفاده از اشاره‌گری مانند <destination ,length> نشان می‌دهد. معنی اشاره‌گر این است که اگر از موقعیت کنونی اشاره‌گر، به اندازه destination به عقب برگردیم، به اندازه length از رشته اصلی، عیناً در محلی که اشاره‌گر وجود دارد، تکرار می‌شود. در هر اشاره‌گر مقدار destination عددی بین ۱ تا ۳۲۷۶۸ است. بنابراین هر اشاره‌گر می‌تواند حداکثر به ۳۲ کیلو بایت قبل از خود اشاره کند. مقادیر ممکن برای length نیز از ۳ تا ۲۵۸ می‌باشد. به عنوان مثال، رشته "abcdefabcd" به فرم LZ77 به صورت "abcdef<6, 4>" فشرده می‌شود که در آن اشاره‌گر <۴و۶> به این معنا است: ۶ کاراکتر

به عقب بازگرد و ۴ کاراکتر را از رشته اصلی، عیناً بجای اشاره‌گر تکرار کن تا به رشته غیرفشرده دست پیدا کنی. در شکل ۲ فرم غیرفشرده و فشرده‌شده طبق الگوریتم LZ77 برای صفحه اصلی گوگل نشان داده شده است.

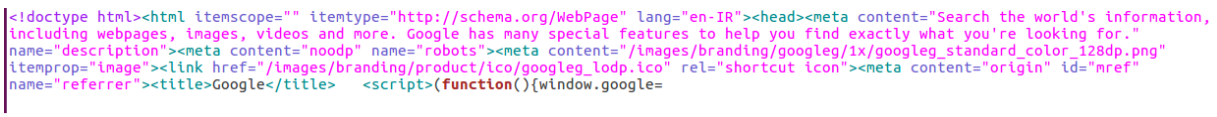
۳- کارهای گذشته

در این بخش مروری بر پیشینه پژوهش‌های صورت گرفته در زمینه بازرسی محتوایی ترافیک رمز شده انجام می‌شود. مشاهده خواهد شد که در این پژوهش‌ها مسئله بازرسی محتوایی ترافیک رمز شده با استفاده از رمزنگاری جستجوپذیر صورت می‌گیرد.

در پژوهش شری و همکارانش [۵]، در سال ۲۰۱۵، اولین راه‌حل در زمینه بازرسی محتوایی ترافیک رمز شده، با حفظ حریم خصوصی کاربران، ارائه شد. در این پژوهش یک روش بنیادی به نام BlindBox، برای بازرسی محتوایی ترافیک HTTPS ارائه شده است. این روش در برنامه‌های کاربردی سرند ترافیک کودکان[۱۴]، سرند ترافیک خروجی[۱۵] و سیستم تشخیص نفوذ[۱۶] قابل استفاده است. در روش BlindBox بدون اینکه محرمانگی بسته از بین برود، بر بدنه بسته‌های HTTPS بازرسی ارسال توکن‌های رمزنگاری شده، ترافیک HTTPS نیز برای گیرنده ارسال می‌شود.

شری و همکارانش در پژوهشی دیگر [۱۴] روشی متفاوت از BlindBox، به نام Embark را ارائه کرده‌اند. در روش Embark برنامه کاربردی DPI به فضای ابری منتقل شده است. در این روش یک دروازه وجود دارد که همه اعمال رمزنگاری و ارتباط با برنامه کاربردی موجود در فضای ابری را به تنهایی انجام می‌دهد. در این روش نیز از رمزنگاری جستجوپذیر استفاده شده است، با این تفاوت که همه اعمال رمزنگاری در دروازه انجام می‌شود.

یان و همکارانش [۱۵] به طور عمده به بررسی مشکلات BlindBox پرداخته‌اند و سعی در رفع آن‌ها داشتند. در این پژوهش به نامناسب بودن روش BlindBox در بازرسی محتوایی ترافیک‌های خروجی، عدم محافظت از مجموعه قوانین در حالتی که سرویس دهنده BlindBox غیرقابل اعتماد است و پرداخت هزینه گزاف پروتکل محاسبات امن[۱۷]، اشاره شده است. هسته اصلی روش ارائه شده در پژوهش آن‌ها، استفاده از سرند رمز شده[۱۸] است. سرند رمز شده در برنامه کاربردی بازرسی محتوایی بسته‌ها پیاده‌سازی می‌شود و محرمانگی بسته‌ها را تضمین می



الف)

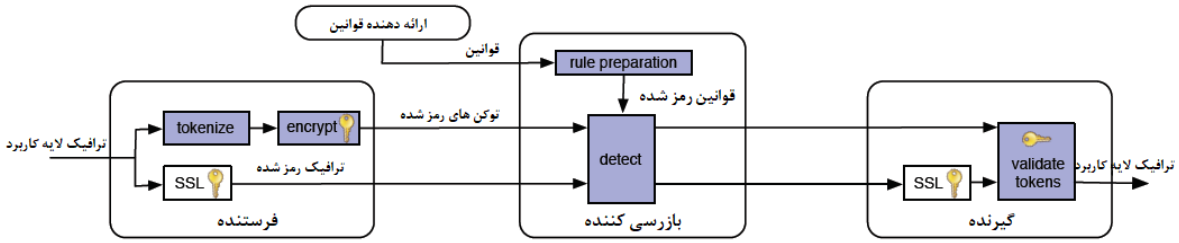
ب)

شکل ۲: بخشی از کد HTML مربوط به صفحه اصلی google.com الف- غیرفشرده ب- فشرده شده با LZ77

محتوایی انجام می‌شود. ایده اصلی برای بازرسی محتوایی،

شمای کلی روش Blindbox [۵] در شکل ۳ نشان داده شده است. فرستنده، گیرنده، بازرسی‌کننده و ارائه دهنده قوانین، اجزای اصلی این معماری هستند. ایده اصلی این روش، استفاده از پروتکل رمزنگاری جستجوپذیر است که به واسطه آن امکان بازرسی محتوایی بر ترافیک رمز شده بدون رمزگشایی آن، فراهم می‌شود. در بازرسی‌کننده، یک درخت جستجوی دودویی از قوانین رمز شده وجود دارد و توکن‌های رمز شده در این درخت جستجو می‌شوند. طبق پروتکل رمزنگاری جستجوپذیری که در این روش استفاده شده است، در سمت فرستنده از محتوای لایه کاربرد (HTTP)، توکن استخراج می‌شود. سپس توکن‌ها در همان فرستنده رمز می‌شوند و برای گیرنده ارسال می‌گردند. بازرسی کننده که در مسیر فرستنده و گیرنده است، بسته‌های حاوی توکن‌های رمزنگاری شده را دریافت می‌کند و در درخت جستجو می‌کند. بعد از کند. بنابراین دراین روش نیز مانند روش BlindBox بعد از برقراری ارتباط HTTPS، ابتدا توکن‌های رمزنگاری جستجوپذیر، از روی محتوای بسته استخراج و رمز می‌شوند و به سرویس دهنده ارسال می‌شوند. سپس در سمت سرویس‌دهنده با استفاده از سرند رمز شده، قوانین و توکن‌های رمز شده با هم مطابقت پیدا می‌کنند و تصمیم مناسبی در مورد عبور یا عدم عبور بسته گرفته می‌شود. از پژوهش‌های مذکور می‌توان نتیجه گرفت که هسته اصلی بازرسی محتوایی ترافیک رمز شده، بر اساس پروتکل رمزنگاری جستجوپذیر است. اما در تمام این پژوهش‌ها تنها به مسئله ترافیک HTTPS پرداخته شده است و ارسال ابرمتن فشرده‌شده از طریق پروتکل HTTPS در نظر گرفته نشده است.

از طرفی پژوهش‌هایی در زمینه بازرسی محتوایی بر ترافیک فشرده شده نیز صورت گرفته است. در این پژوهش‌ها برای بازرسی محتوایی، قوانین به صورت عبارت منظم توصیف می‌شوند و پیاده‌سازی در قالب ماشین حالت، انجام می‌شود. هدف این روش‌ها، بهبود پویش ماشین



شکل ۳: شمای کلی از بازرسی محتوایی ترافیک HTTPS طبق روش Blindbox [۵]

حالت بر ترافیک فشرده شده است. در ادامه مختصری از این پژوهش‌ها ارائه می‌شود.

نیشاد و سانکار [۱۶] برای داده‌های بزرگ روش جدیدی را ارائه داده‌اند. این روش علاوه بر اینکه فشرده‌ترین حالت ممکن را برای داده های بزرگ ایجاد می‌کند، این امکان را فراهم می‌کند که تنها قسمت مورد نظر بازسازی شود و مورد جستجو قرار گیرد.

پژوهش بکی و همکارانش [۸]، در زمینه تطبیق عبارت منظم بر ترافیک ابرمتن فشرده‌شده به منظور بازرسی محتوایی ترافیک، صورت گرفته است. فشرده‌سازی که HTTP استفاده می‌کند معمولاً فشرده‌ساز GZIP است. GZIP از ترکیب دو فشرده‌ساز استفاده می‌کند: ابتدا با الگوریتم LZ77 به جای رشته‌های تکراری، اشاره‌گرهایی به فرمت <length, destination> قرار می‌دهد. سپس رشته نیمه فشرده‌شده جدید را با الگوریتم هافمن کاملاً فشرده می‌کند. در واقع این روش با استفاده از NFA و DFAهایی که برای رشته‌های عادی (فشرده نشده) طراحی شده‌اند، عبارات منظم را بر داده نیمه فشرده LZ77 تطبیق می‌دهد [۸].

پژوهش برملر بار [۱۷] اولین راه‌حل سریع ارائه شده در زمینه تطبیق چندالگویی بر ترافیک ابرمتن فشرده‌شده بدون بازسازی کامل است. فشرده‌سازی ابرمتن از طریق الگوریتم فشرده‌سازیGZIP صورت می‌گیرد. بازسازی کامل ابرمتن فشرده‌شده به فضا و زمان زیادی احتیاج دارد. بنابراین برملر بار به دنبال روشی سریع بوده است که بدون بازسازی کامل بتواند بر ابرمتن فشرده‌شده، تطبیق چندالگویی را انجام دهد. برملر بار روی رشته فشرده LZ77 تطبیق چند الگویی انجام می‌دهد.

در پژوهشی دیگر از برملر بار و همکاران [۱۸]، سرعت تطبیق چندالگویی بر ابرمتن فشرده‌شده با استفاده از پرش wu-manber [۱۹] افزایش یافته است. برملربار و همکاران روش پیشنهادی خود را تطبیق الگوی پرش-مبنا نامیده‌اند. این روش در مقایسه با روش قبلی [۱۷] از نظر زمان و فضای ذخیره‌سازی بهتر عمل می‌کند. در این روش الگوریتم wu-manber به‌گونه‌ای تغییر داده می‌شود که برای رشته‌های فشرده LZ77 مناسب باشد. بنابراین این الگوریتم جدید wu-manber تغییر یافته با پرش از رشته‌های غیرمهم، سرعت جستجو بر ابرمتن فشرده‌شده را افزایش می‌دهد.

پژوهشی دیگر [۲۰]، تطبیق چند الگویی بر ترافیک فشرده‌شده شبکه را امکان‌پذیر کرده است. از آنجایی که این روش، بر ترافیک فشرده شبکه کار می‌کند، الگوریتم فشرده‌سازی مورد استفاده در آن، GZIP است. بنابراین عمل تطبیق چندالگویی را بر رشته فشرده LZ77 انجام می‌دهد. ایده اصلی، استفاده از دو نوع پرش است که منجر به کاهش پویش در رشته فشرده‌شده می‌شود. ابتدا رشته LZ77 به اسلایدهایی تقسیم می‌شود اما فقط بعضی از آن‌ها پویش می‌شوند و از روی اسلایدهای غیرمهم پرش می‌شود. همچنین با استفاده از الگوریتم wu-manber، رشته‌های غیرمهم نادیده گرفته می‌شوند و پویش نمی‌شوند. به این ترتیب، میزان رشته‌ای که پویش می‌شود بسیار کاهش می‌یابد و در نتیجه در زمان صرفه‌جویی خواهد شد.

۴- چالش و رهیافت

در این بخش، ابتدا به چالشی که در رابطه با استخراج توکن‌ها از محتوای فشرده شده وجود دارد، پرداخته می‌شود. سپس، رهیافت پیشنهادی برای مقابله با این چالش معرفی می‌گردد.

۴-۱- چالش

همانطورکه گفته شد، ساده‌ترین روش برای استخراج توکن‌ها از محتوای فشرده‌شده عبارت است از فشرده‌گشایی کامل ابرمتن فشرده‌شده و اعمال NFA، برای استخراج توکن‌های معنادار[۵]. این روش، Blindbox مبتنی بر فشرده‌گشایی کامل نامیده شد. بنابراین برای فشرده‌گشایی ابرمتن فشرده‌شده با GZIP می‌بایست ابتدا کدگذاری هافمن را از ابرمتن برداشت، سپس عکس الگوریتم LZ77 را اجرا کرد تا به ابرمتن غیرفشرده رسید؛ آنگاه NFA را برای استخراج توکن‌ها اعمال نمود. این کار سربار زمانی زیادی دارد. در ارزیابی‌های بخش ۶ و جدول ۲ مشاهده خواهد شد که عمل فشرده‌گشایی، در مقایسه با فرآیند استخراج توکن‌ها بخش خیلی ناچیزی از زمان را به خود اختصاص می‌دهد.

بنابراین علت سربار زمانی زیاد روش فشرده‌گشایی کامل، پویش NFA برای استخراج توکن‌های معنادار است. این موضوع در یکی از پژوهش‌های پیشین [۷] نیز تأیید شده است. در جدول ۱ دو ماشین NFA و DFA از نظر سربار زمان و حافظه مقایسه شده‌اند. در این جدول مشاهده می‌شود که ماشین DFA از نظر پیچیدگی زمانی نسبت به ماشین NFA بهتر است و سربار زمانی پیمایش NFA بسیار بالا است. ماشینی که در این پژوهش برای استخراج توکن‌های معنادار طراحی شده است از دو بخش تشکیل شده است: یک DFA برای پذیرش تگ‌ها و مشخصه‌ها و یک NFA برای پذیرش رشته‌هایی غیر از این دو دسته. بنابراین وجود NFA در این ماشین باعث می‌شود که سربار زمانی استخراج توکن‌های معنادار زیاد شود. این موضوع در بخش ۶ نشان داده

خواهد شد. همچنین نحوه طراحی این ماشین حالت در بخش ۵-۲ توضیح داده خواهد شد.

۴-۲- رهیافت

هدف این پژوهش، کاهش سربار زمانی برای استخراج توکن‌های مورد نیاز پروتکل رمزنگاری جستجوپذیر است. پیش‌تر ذکر شد که در روش Blindbox مبتنی بر فشرده‌گشایی کامل[۵]، ابتدا ابرمتن به طور کامل فشرده‌گشایی می‌شود. سپس توکن‌ها با استفاده از NFA به روش Blindbox [۵] از متن آشکار استخراج می‌شوند. این NFA در بخش ۲-۵ معرفی می‌شود.

در بخش ۴-۱ مشاهده شد که عمل فشرده‌گشایی زمان زیادی را به خود اختصاص نمی‌دهد، اما پویش NFA و استخراج توکن‌ها به زمان زیادی نیاز دارد. بنابراین برای کاهش زمان استخراج توکن‌ها و بهبود روش فشرده‌گشایی کامل، به دنبال روشی هستیم که زمان مورد نیاز برای استخراج توکن‌ها را کاهش دهد.

در این مقاله پیشنهاد می‌شود که به‌جای فشرده‌گشایی کامل ابرمتن و استخراج توکن‌ها، فقط کدگذاری هافمن برداشته شود تا به یک متن فشرده شده با LZ77 دست یابیم. سپس از ویژگی‌های ذاتی ابرمتن فشرده با LZ77 برای تسریع فرآیند استخراج استفاده شود. به معنای دیگر، در روش پیشنهادی از متن فشرده شده با LZ77 (غیرآشکار) توکن استخراج می‌شود.

در بخش ۲-۳ گفته شد فشرده‌ساز LZ77 به این صورت عمل می‌کند که به جای زیر رشته‌های تکراری، اشاره‌گرهایی قرار می‌دهد. از این ویژگی می‌توان برای بهبود سرعت NFA استفاده نمود. اگر یک توکن به دفعات در کد HTML تکرار شده باشد (مانند تگ‌ها)، به غیر از اولین بار، دفعات بعدی تکرار آن توکن در متن، توسط اشاره‌گرهای LZ77 نشان داده می‌شوند. بنابراین توکن‌های تکراری با وجود این اشاره‌گرها قابل تشخیص هستند. در روش پیشنهادی این مقاله با استفاده از اشاره‌گرهایی که در متن وجود دارد توکن‌های تکراری یافته می‌شوند و از اعمال NFA بر آن‌ها جلوگیری می‌شود.

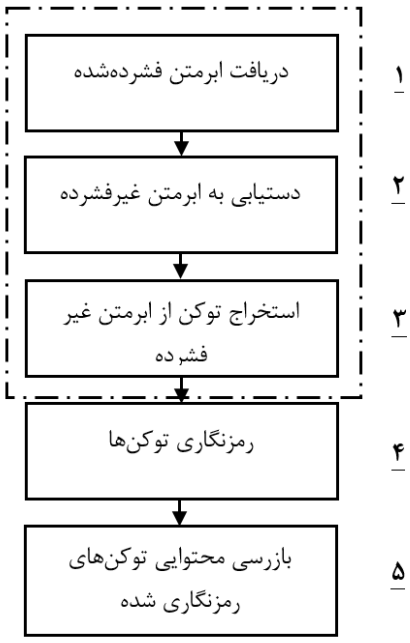
انتظار می‌رود که این کار در نهایت موجب کاهش سربار زمانی برای استخراج توکن‌های مورد نیاز پروتکل رمزنگاری جستجوپذیر شود. جزئیات این کار در بخش ۵-۱ توضیح داده می‌شود.

جدول ۱ : مقایسه ماشین حالت NFA و DFA [۷]

	یک عبارت منظم به طول n		m عبارت منظم در قالب یک عبارت منظم	
	پیچیدگی زمانی	حافظه مورد نیاز	پیچیدگی زمانی	حافظه مورد نیاز
NFA	$O(n^2)$	$O(n)$	$O(n^2m)$	$O(nm)$
DFA	$O(1)$	$O(\sum^n)$	$O(1)$	$O(\sum^{nm})$

۵- روش پیشنهادی

مسئله پژوهشی این مقاله استخراج توکن از ترافیک فشرده‌شده است. این توکن‌ها در پروتکل رمزنگاری جستجوپذیر استفاده می‌شوند که برای بازرسی محتوایی ترافیک فشرده شده HTTPS کاربرد دارد. شمایی از مراحل اصلی بازرسی محتوایی ترافیک فشرده‌شده HTTPS در شکل ۴ نشان داده شده است. در این شکل، مراحل ۱ تا ۴ در فرستنده انجام می شوند و مرحله ۵ که بازرسی توکن‌های رمزنگاری شده است در بازرسی کننده انجام خواهد شد. کادر خط‌چین در شکل ۴، محدوده هدف پژوهش، یعنی استخراج توکن از محتوای فشرده‌شده، را نشان می‌دهد. طبق این شکل، در روش پیشنهادیِ این پژوهش، در فرستنده برنامه‌ای وجود دارد که ابتدا ابرمتن فشرده را قبل از ارسال به لایه SSL فشرده گشایی می‌کند؛ سپس توکن‌ها را استخراج می‌کند. چالش و رهیافت این مسئله در بخش ۴ مطرح شد. در این بخش با تشریح جزئیات مراحل ۱ تا ۳ که در شکل ۴ مشخص شده است، نحوه طراحی ماشین NFA و ذکر مثال، روش پیشنهادی مقاله برای فشرده‌گشایی و استخراج توکن بیان می‌شود.



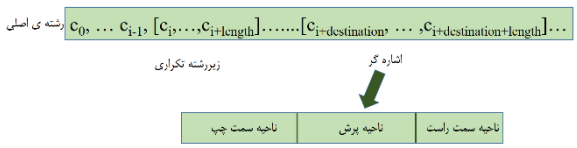
شکل ۴: مراحل اصلی بازرسی محتوایی ترافیک فشرده شده HTTPS (مراحل داخل کادر خط‌چین در فرستنده و مرحله ۵ در بازرسی‌کننده انجام می‌شود)

۵-۱- الگوریتم پیشنهادی (مراحل کار)

روش پیشنهادی این مقاله بر مبنای تشخیص توکن‌های تکراری است. توکن‌های تکراری را می‌توان در زیررشته‌های تکراری یافت که با اشاره گر نشان داده می‌شوند. رشته‌های موجود در یک متن LZ77 به دو دسته تقسیم می‌شوند: کاراکترهای آشکار و اشاره‌گرها. پیمایش کاراکترهای آشکار به صورت عادی توسط NFA (که در بخش ۵-۲ طراحی خواهد شد) انجام می‌شود اما نحوه پیمایش اشاره‌گرها متفاوت خواهد بود. روش

پیشنهادی باید توکن‌های تکراری را تشخیص دهد و از پیمایش آن‌ها جلوگیری کند. برای این کار نیاز به اطلاعاتی است که وجود توکن‌های تکراری در زیررشته را نشان دهد. سپس با توجه به این اطلاعات، زیررشته تکراری، به سه بخش تقسیم می‌شود. ناحیه سمت چپ، پرش و ناحیه سمت راست.

دو ناحیه سمت چپ و راست نواحی هستند که باید توسط NFA پیمایش شوند. از آنجایی که قبل از وقوع یک اشاره‌گر ممکن است ماشین حالت NFA هنوز به حالت پایانی نرسیده باشد، باید تعدادی از کاراکترهای زیررشته تکراری توسط NFA پیمایش شوند تا NFA به حالت نهایی برسد (یک توکن استخراج شود). این ناحیه، ***ناحیه سمت چپ*** است. سپس با توجه به آن اطلاعات، توکن‌های تکراری و محدوده وقوع آن‌ها تشخیص داده می‌شوند. به ناحیه‌ای که توکن‌های تکراری وجود دارد، ***ناحیه پرش*** می‌گویند. آنگاه، اگر از زیررشته هنوز کاراکترهایی باقی مانده باشد، ***ناحیه سمت راست*** را تشکیل می‌دهد که باید توسط NFA پیمایش شود. در شکل ۵ این نواحی نشان داده شده‌اند. با مشخص شدن ایده اصلی مقاله، روند کلی روش پیشنهادی توضیح داده می‌شود. در شکل ۶، روش پیشنهادی به صورت روندنما نشان داده شده است. یادآوری می‌شود که از ابرمتن فشرده شده قبل از ارسال به لایه SSL برای استخراج توکن استفاده می‌شود. روندنمای شکل ۶، همان کادر قرمز رنگ در شکل ۴ می‌باشد که با جزئیات بیان شده است. ابتدا متن فشرده‌شده با GZIP کدگشایی می‌شود و به متن فشرده‌شده با LZ77 تبدیل می‌شود. می‌دانیم که متن فشرده شده با LZ77 حاوی اشاره گرهایی است. بنابراین در یک چرخه، رشته‌ای از متن فشرده شده با LZ77، انتخاب می‌شود و بررسی می‌شود که آیا یک اشاره‌گر است یا رشته‌ای از کاراکترهای آشکار (اگر پایان متن نبود). اگر رشته ورودی اشاره‌گر نبود، توسط NFA پیمایش می‌شود. در این مرحله اگر NFA به حالت نهایی رسید، به این معنی است که یک توکن تشخیص داده شده است و این توکن استخراج می‌شود.



شکل ۵: نمایی از رشته اصلی، زیر رشته تکراری، اشاره‌گر و نواحی مربوط به اشاره‌گر

اگر رشته ورودی اشاره‌گر بود، ابتدا فشرده‌گشایی می‌شود. به عبارت دیگر، زیررشته‌ای که اشاره‌گر به آن اشاره می‌کند، در محل اشاره‌گر کپی می‌شود. سپس ***ناحیه سمت چپ*** زیررشته توسط NFA پیمایش می‌شود. بعد از پیمایش ناحیه سمت چپ، بررسی می‌شود که از این زیررشته‌ای که کپی شده، قبلاً توکن(هایی) استخراج شده یا خیر. اگر زیررشته شامل توکن تکراری بود، به تعداد تکرار آن یک واحد اضافه می‌شود و از روی آن پرش می‌شود، این کار تا جایی ادامه می‌یابد که همه توکن‌های تکراری درون زیررشته تشخیص داده شوند. به این قسمت از زیررشته ***ناحیه پرش*** می‌گویند.

زمانی که توکن تکراری در زیررشته وجود داشته باشد، توسط NFA پیمایش نمی‌شود و این امر باعث می‌شود که سربار زمانی استخراج توکن ها کاهش یابد. زمانی که همه توکن‌های تکراری در یک زیررشته تشخیص داده شدند، کاراکترهای باقیمانده از زیررشته (در صورت وجود) توسط NFA پیمایش می‌شوند. این ناحیه همان ***ناحیه سمت راست*** است. آنگاه با پایان یافتن کاراکترهای اشاره‌گر، به متن فشرده شده با LZ77 بازمی‌گردد و همین روند ادامه می‌یابد تا زمانی که متن فشرده شده با LZ77، به پایان برسد.

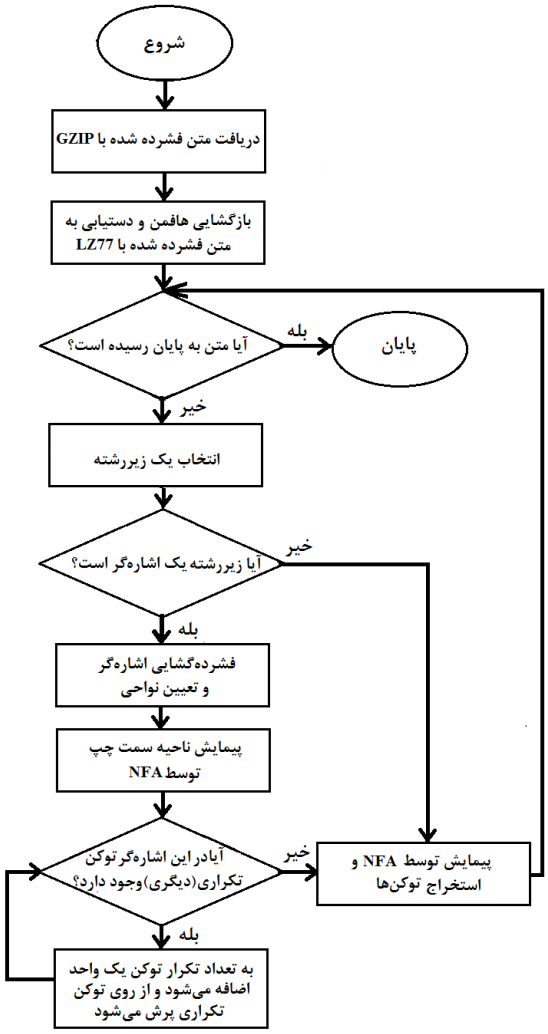
۵-۲- ماشین NFA طراحی شده

برای استخراج توکن‌های معنادار باید ماشین حالتی طراحی شود که تگ ها و مشخصه‌های کد HTML را پذیرش کند. همانطورکه در بخش ۲-۲ گفته شد، اجرای ماشین DFA سریع‌تر از ماشین NFA است. اما از طرفی حافظه قابل توجهی برای ذخیره ماشین حالت نیاز دارد. از آنجایی که تعداد تگ‌ها و مشخصه‌ها محدود است، برای دستیابی به سرعت بیشتر برای استخراج این توکن‌ها از DFA استفاده می‌شود. در بعضی از قسمت‌های یک کد HTML رشته‌هایی غیر از تگ‌ها و مشخصه‌ها وجود دارند که تعداد نامحدودی دارند و اگر از DFA برای استخراج آن‌ها استفاده شود، حافظه بسیار زیادی جهت ذخیره ماشین نیاز است که در نهایت موجب سربار زمانی نیز می‌شود. به همین دلیل برای تشخیص و استخراج این نوع توکن‌ها از ماشین NFA استفاده می‌شود.

به طور خلاصه، ماشین DFA را می‌توان برای استخراج توکن‌ها از تگ‌ها و مشخصه‌ها استفاده کرد و برای توکن‌های غیر از این دو دسته از ماشین حالت NFA استفاده می‌شود. در نتیجه برای استخراج کل توکن‌های معنادار باید ترکیبی از دو ماشین حالت NFA و DFA طراحی کرد و آن را به عنوان یک ماشین حالت یکپارچه پیاده‌سازی نمود. از آنجایی که هر DFA نوع خاصی از NFA محسوب می‌شود، ترکیب این دو ماشین NFA و DFA، در مجموع به عنوان یک NFA محسوب می شود. در شکل ۷-الف بخشی از ماشین DFA نشان داده شده است که چند تگ HTML را می‌پذیرد. شکل ۷-الف تنها بخش محدودی از ماشین را نشان می‌دهد و در واقع، این ماشین به همین ترتیب توسعه می‌یابد تا همه تگ‌ها و مشخصه‌ها را بپذیرد.

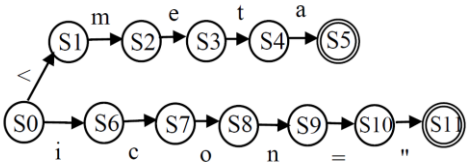
برای طراحی NFA جهت تشخیص توکن‌هایی غیر از تگ‌ها و مشخصه‌ها، ابتدا یک عبارت منظم تعریف می‌شود و سپس عبارت منظم به ماشین NFA تبدیل می‌گردد. برای تشخیص این توکن‌ها ابتدا ۲۵۶ کاراکتر اسکی را به سه گروه تقسیم می‌کنیم. گروه α از حروف کوچک و بزرگ الفبا و اعداد تشکیل شده است و گروه δ شامل کاراکترهایی مانند فاصله، \n، \t، \r است. باقی کاراکترها در گروه β قرار می‌گیرند. یک توکن باید به کاراکترهای گروه δ ختم شود یا اگر تعداد کاراکترهای طی شده بیشتر از ۳۰ شد، باید پس از مواجهه با اولین کاراکتر از گروه

β، به حالت نهایی برویم. در شکل ۷-ب ماشین NFA برای پذیرش این نوع توکن‌ها نشان داده شده است.

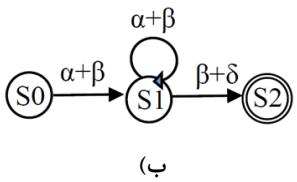


شکل ۶: روندنمای روش پیشنهادی

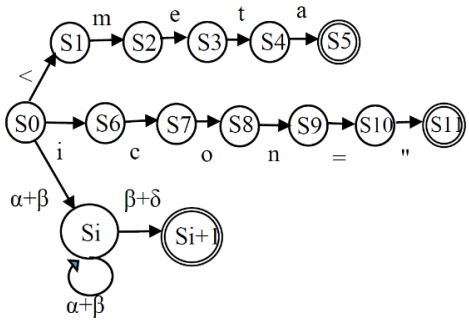
در نهایت، ماشین DFA و ماشین NFA با هم ترکیب می‌شوند و در قالب یک ماشین NFA یکپارچه برای پذیرش توکن‌های معنادار استفاده می‌شوند (شکل ۷-ج). با استفاده از این NFA، هر رشته‌ای که مورد پذیرش قرار گیرد، به عنوان یک توکن ذخیره می‌شود. این ماشین NFA، همان ماشینی است که در بخش‌های ۴-۲، ۵-۱ و ۵-۳ مورد اشاره قرار گرفته است.



الف)



ب)



ج)

شکل ۷: مراحل طراحی ماشین حالت NFA الف) مثال کوچکی از طراحی DFA برای پذیرش تگ‌ها و مشخصه‌ها ب) طراحی NFA برای پذیرش توکن‌هایی با طول نامعلوم ج) ماشین حالت NFA نهایی برای پذیرش توکن‌های معنادار

۵-۳- چند مثال

در ادامه، حالت‌های مختلف از نواحی یک اشاره‌گر و نحوه کار روش پیشنهادی با ذکر چند مثال توضیح داده می‌شوند. ابتدا ساختمان داده‌هایی که برای پیاده‌سازی روش پیشنهادی استفاده شده‌اند، معرفی و سپس مثال‌ها تشریح می‌شوند.

- ساختمان داده ***Info*** به ازای هر کاراکتر پیمایش شده توسط NFA، در خود اطلاعاتی را ذخیره می‌کند. در مراحل بعد با استفاده از این اطلاعات، توکن‌های تکراری تشخیص داده می‌شوند.
- ***Len*** به ازای هر کاراکتر پویش شده توسط NFA، مقدارش یک واحد اضافه می‌شود و در ***Info*** متناظر با کاراکتر ذخیره می‌شود.
- یک لیست از توکن‌های استخراج شده وجود دارد و هر کدام از توکن‌ها در یک سطر ذخیره می‌شوند. ***Line*** شماره سطری است که توکن استخراج شده در آن ذخیره می‌شود و به نحوی مقداردهی می‌شود که قابل تمایز با مقدار ***Len*** باشد. زمانی که یک توکن استخراج می‌شود، مقدار ***Line*** در ساختمان‌داده ***Info*** متناظر با کاراکتر آخر توکن (حالت نهایی NFA)، ذخیره می‌شود. این مقدار در ارجاع‌های بعدی توسط اشاره‌گرهای LZ77، مشخص می‌کند که یک توکن تکراری وجود دارد و قبلاً در سطر ***Line*** از لیست توکن ها ذخیره شده است.
- هنگامی‌که یک اشاره‌گر در رشته اصلی کپی می‌شود، مقدار ***Info*** در حین کپی شدن، بررسی می‌شود تا مشخص شود که آیا مقدار ***Line*** در آن ذخیره شده است یا ***Len***؛ اگر مقدار ***Line*** ذخیره شده بود یعنی در این موقعیت قبلاً یک توکن استخراج شده و در سطر

Line ذخیره شده است. برای ذخیره موقعیت‌هایی که قبلاً در آن‌ها توکن استخراج شده است، از ساختمان‌داده آرایه‌ای ***match_index*** استفاده می‌شود و اندیس آن موقعیت‌ها در این ساختمان داده ذخیره می‌شود.

در شکل ۸ چند مثال آورده شده است که حالت‌های مختلف از نواحی یک اشاره‌گر را نشان می‌دهد. با مثال ۱ نحوه تقسیم بندی نواحی چپ، پرش و راست تشریح می‌شود و در مثال‌های بعدی حالات دیگری از تقسیم بندی نواحی نشان داده شده‌اند. طبق روندنمای شکل ۶، بعد از کدگشایی هافمن، ابتدا یک زیر رشته از ابرمتن فشرده شده با LZ77 انتخاب می‌شود. شکل ۸ حالتی را نشان می‌دهد که زیررشته انتخاب شده، اشاره‌گر باشد. زیررشته‌ای که اشاره‌گر به آن اشاره می‌کند، کپی شده و با رنگ سبز نشان داده شده است. زمانی که کاراکترهای تکراری در موقعیت اشاره‌گر کپی می‌شود، اطلاعات ذخیره شده در ***Info***ی زیررشته تکراری نیز در موقعیت اشاره‌گر کپی می‌شوند. در حین کپی کردن ساختمان داده ***Info***، موقعیت‌هایی که با L پر شده‌اند در ساختمان داده ***match_index*** ذخیره می‌شوند. اطلاعات ذخیره شده در ساختمان داده ***Info*** در صورت پیمایش کاراکترها توسط NFA، به‌روز رسانی می شوند.

مثال ۱: در این مثال مشاهده می‌شود که یک کاراکتر قبل از زیررشته تکراری(کاراکتر z)، مقدار ۲ را در ***Info*** نشان می‌دهد. اگر مقدار ***Info*** غیرصفر باشد یعنی قبل از زیررشته تکراری، NFA در حال پیمایش بوده و در یکی از حالت‌های میانی قرار دارد. بنابراین طبق روندنمای شکل ۵، از شروع زیررشته تا کاراکتری که توکن استخراج شود، ***ناحیه سمت چپ*** است و باید توسط NFA پیمایش شود. برای تعیین ناحیه پرش، به ساختمان داده ***match_index*** مراجعه می‌شود. در حین کپی شدن زیررشته، سه موقعیت در ساختمان داده ***match_index*** ذخیره شده است. یکی از آن‌ها (L_i) قابل استفاده نیست، زیرا در ***ناحیه سمت چپ*** قرار گرفته است. اما دو موقعیت دیگر وجود دارد. این دو به عنوان توکن‌های تکراری در نظر گرفته می‌شوند و توسط NFA پیمایش نمی شوند، بلکه فقط به تعداد تکرار آن‌ها یک واحد اضافه می‌شود. اگر بعد از ناحیه پرش، قسمتی از زیررشته باقی مانده باشد، توسط NFA پیمایش می‌شود و به آن ***ناحیه سمت راست*** گفته می‌شود.

مثال ۲: در این مثال ناحیه ***سمت راست*** وجود ندارد. زیرا آخرین کاراکتر زیررشته تکراری، L_k است.

مثال ۳: در این مثال ناحیه ***سمت چپ*** وجود ندارد. زیرا آخرین کاراکتر قبل از شروع زیررشته‌ی‌تکراری، L_i است. یعنی NFA به حالت نهایی رسیده و در حال حاضر در حالت اولیه قرار دارد. بنابراین مستقیم وارد ***ناحیه پرش*** می‌شود.

مثال ۴: اگر در زیررشته تکراری هیچگونه توکن تکراری وجود نداشته باشد، ***ناحیه پرش*** نیز وجود ندارد و همه زیر رشته توسط NFA پیمایش می‌شود.

با توجه به روندنمای روش پیشنهادی که توضیح داده شد، انتظار می‌رود که وجود ***ناحیه پرش*** موجب بهبود سرعت استخراج توکن‌های رمزنگاری جستجوپذیر شود. در ادمه در بخش ۶ این موضوع بررسی و تفسیر خواهد شد.

۶- ارزیابی نرم‌افزاری

در این بخش ابتدا مجموعه داده‌هایی که برای آزمایش الگوریتم روش پیشنهادی استفاده شده، معرفی می‌شوند. سپس نتایج به دست آمده از پیاده‌سازی روش پیشنهادی تحلیل و ارزیابی می‌شوند.

برای آزمایش الگوریتم روش پیشنهادی مقاله، از کد HTML صفحات ۵۰۰ وبگاه برتر دنیا استفاده شده است که لیست آن‌ها از وبگاه Alexa [۲۱] بدست آمده است. حجم این صفحات وب در حالت غیرفشرده 66MB و در حالت فشرده با الگوریتم GZIP به 15MB می‌رسد. برای اجرای برنامه‌های پیاده‌سازی شده از رایانه‌ای با پردازنده corei7 و حافظه داخلی 4GB استفاده شده است؛ برنامه‌ها به زبان C و با بهره‌گیری از کتابخانه zlib نوشته شده است [۲۲]. کتابخانه zlib برای فشرده‌گشایی ابرمتن فشرده استفاده می‌شود. برای پیاده‌سازی، با ایجاد تغییراتی در این کتابخانه، ابرمتن فشرده شده با LZ77 حاصل می‌شود.

برای انجام ارزیابی‌ها سه روش پیاده‌سازی می‌شوند.

- روش پیشنهادی: در روش پیشنهادی، همانطورکه ذکر شد، بعد از کدگشایی هافمن، الگوریتم معرفی شده در بخش ۵-۱ و NFA طراحی شده در بخش ۵-۲ بر روی رشته‌های LZ77 پیاده‌سازی می‌شود.
- روش Blindbox مبتنی بر فشرده‌گشایی کامل[۵]: در این روش، NFA طراحی شده در بخش ۵-۲ پیاده‌سازی می‌شود و برای ۵۰۰ وبگاه برتر اجرا می‌شود. این روش در حقیقت همان روش Blindbox [۵] برای استخراج توکن است که قبل از اجرای آن ابرمتن، فشرده‌گشایی کامل می‌شود.
- روش بدون پرش از توکن‌های تکراری: در این روش، ابتدا کد هافمن از ابرمتن فشرده برداشته می‌شود و NFA بر متن فشرده‌شده با LZ77، بدون پرش از توکن‌های تکراری اعمال می‌شود.

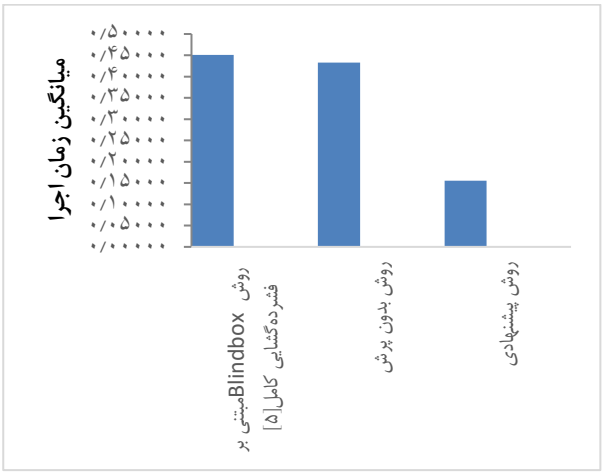
با پیاده‌سازی این سه روش، به مقایسه و ارزیابی روش پیشنهادی پرداخته می‌شود. جدول ۲ میانگین زمان اجرای مراحل مختلف روش‌های پیاده‌سازی شده برای ۵۰۰ وبگاه برتر را نشان می‌دهد. در این جدول مشاهده می‌شود که زمان فشرده‌گشایی در مقایسه با زمان استخراج توکن‌ها بسیار ناچیز است. شکل ۸ نمودار نتایج حاصل از پیاده سازی روش پیشنهادی (پرش از توکن‌های تکراری)، Blindbox مبتنی بر فشرده‌گشایی کامل[۵] و بدون پرش را نشان می‌دهد.

شکل ۸: چند مثال از نحوه اجرای روش پیشنهادی

با پیاده‌سازی این سه روش، به مقایسه و ارزیابی روش پیشنهادی پرداخته می‌شود. جدول ۲ میانگین زمان اجرای مراحل مختلف روش‌های پیاده‌سازی شده برای ۵۰۰ وبگاه برتر را نشان می‌دهد. در این جدول مشاهده می‌شود که زمان فشرده‌گشایی در مقایسه با زمان استخراج توکن‌ها بسیار ناچیز است. شکل ۸ نمودار نتایج حاصل از پیاده سازی روش پیشنهادی (پرش از توکن‌های تکراری)، Blindbox مبتنی بر فشرده‌گشایی کامل[۵] و بدون پرش را نشان می‌دهد.

از شکل ۹ نتایج مهمی بدست می‌آید:

- تحلیلی که در بخش ۴-۱ مبتنی بر سربار زمانی اجرای NFA انجام شد، در نتایج مربوط به «روش Blindbox مبتنی بر فشرده‌گشایی کامل[۵]» و «روش بدون پرش» کاملاً مشهود است. در این نتایج مشاهده می‌شود که بیشتر از ۹۹ درصد از کل زمان اجرا، صرف پیمایش NFA برای استخراج توکن‌ها شده است. بنابراین اگر از پیمایش تعدادی از کاراکترها صرف‌نظر شود باید در زمان کل اجرا بهبود مشاهده شود.
- تفاوت قابل ملاحظه‌ای میان نتایج «روش Blindbox مبتنی بر فشرده‌گشایی کامل[۵]» و «روش بدون پرش» مشاهده نمی‌شود، زیرا در هر دو روش همه کاراکترها توسط NFA پیمایش می‌شوند. بنابراین می‌توان اطمینان یافت که با کاهش تعداد کاراکترهای پیمایش شده بهبود خوبی در استخراج توکن‌ها حاصل می‌گردد.
- در نتایج «روش پیشنهادی» به خوبی مشاهده می‌شود که با جلوگیری از پیمایش توکن‌های تکراری، به میزان قابل توجهی سرعت استخراج توکن‌ها افزایش یافته است. با توجه به اعداد جدول ۲ زمان اجرای کل نسبت به روش بدون پرش ۶۳/۷۶ درصد و نسبت روش Blindbox مبتنی بر فشرده‌گشایی کامل[۵] ۶۵/۲۰ درصد کاهش یافته است.
- همچنین برای بررسی معنادار بودن اختلاف میان دو روش «Blindbox مبتنی بر فشرده‌گشایی کامل[۵]» و «روش پیشنهادی» از آزمون t دو نمونه مستقل [۲۳-۲۴] استفاده شده است. سطح معناداری برابر با ۰/۰۵ تعریف شده است. بنابراین اگر سطح معناداری کمتر از ۰/۰۵ به دست بیاید، یعنی اختلاف میان دو روش معنادار است و هر چه این عدد کوچک‌تر شود درصد معنادار بودن اختلاف بیشتر می‌شود که نتایج آن در جدول ۳ نشان داده شده است.



شکل ۹: نمودار مقایسه میانگین زمان اجرا

در شکل ۱۰ نسبت نرخ وقوع اشاره‌گر در هر فایل LZ77 به نرخ کاراکترهای پویش شده توسط NFA در روش پیشنهادی مقاله نشان داده شده است. نرخ وقوع اشاره‌گر، به معنی نسبت تعداد کاراکترهای نمایش داده شده با اشاره‌گر، به تعداد کل کاراکترها است. نرخ کاراکترهای پویش شده نیز از نسبت کاراکترهای پویش شده توسط NFA، به تعداد کل کاراکترها محاسبه می‌شود. این دو پارامتر به ازای

هر ۵۰۰ فایل محاسبه شده‌اند و در شکل ۱۰ نشان داده شده‌اند. از این نمودار می‌توان نتیجه گرفت هرچه نرخ وقوع اشاره‌گر در یک فایل بیشتر باشد و به ۱ نزدیک‌تر شود، نرخ کاراکترهای پویش شده توسط NFA کاهش می‌یابد. در این نمودار نشان داده می‌شود که رابطه خطی میان نرخ وقوع اشاره‌گر و نرخ کاراکترهای پویش شده وجود دارد. در شکل ۱۱ هیستوگرام نرخ وقوع اشاره‌گر نمایش داده شده است. با مشاهده دو نمودار شکل ۱۰ و ۱۱ این نتیجه حاصل می‌شود که در اکثر فایل‌ها نرخ وقوع اشاره‌گر بالای ۰/۸ است که در آن محدوده نرخ کاراکترهای پویش شده(در شکل ۱۰) بین ۰/۶ الی ۰/۴ است.

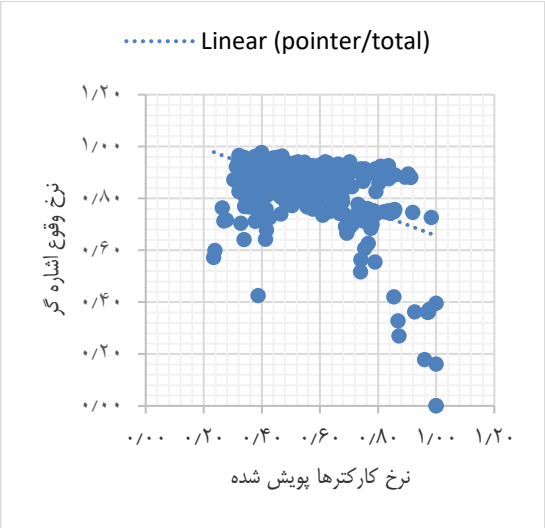
با محاسبه نسبت تعداد کاراکترهای پویش نشده به تعداد کاراکترهای نمایش داده شده با اشاره‌گر، این نتیجه حاصل می‌شود: ۵۱ درصد از کاراکترهایی که با اشاره‌گر نشان داده شده‌اند، پویش نمی‌شوند. به معنای دیگر به طور متوسط هر اشاره‌گر ۵۱ درصدش ناحیه پرش است و ۴۹ درصد آن ناحیه سمت چپ یا راست است که توسط NFA پیمایش می‌شود.

جدول ۲: میانگین زمان اجرای روش‌های پیاده‌سازی شده

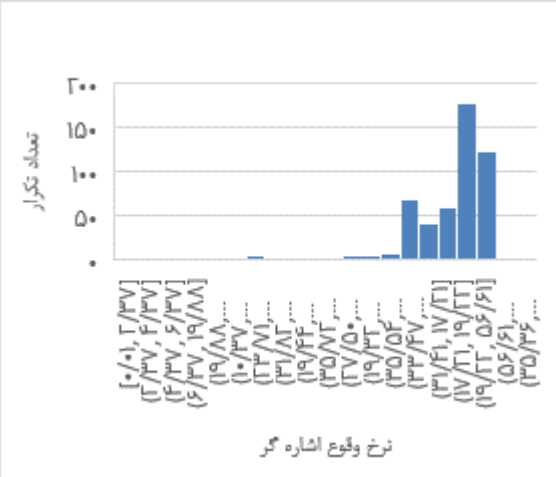
	روش Blindbox مبتنی بر فشرده‌گشایی کامل [۵]	روش بدون پرش	روش پیشنهادی	بهبود روش پیشنهادی نسبت به فشرده‌گشایی	بهبود روش پیشنهادی نسبت به روش بدون پرش
زمان فشرده‌گشایی	۰/۰۰۱۶۶۲ ثانیه	۰/۰۰۱۹۷۳ ثانیه	۰/۰۰۲۰۹۰ ثانیه	-	-
زمان استخراج توکن‌ها	۰/۴۵۱۱۳۳ ثانیه	۰/۴۳۲۸۵۴ ثانیه	۰/۱۵۵۴۵۶ ثانیه	۶۵/۰۵۴٪	۶۴/۰۸٪
زمان کل	۰/۴۵۲۷۹۵ ثانیه	۰/۴۳۴۸۲۸ ثانیه	۰/۱۵۷۵۴۵ ثانیه	۶۵/۲۰٪	۶۳/۷۶٪

جدول ۳: نتایج آزمون t زمان اجرا

	روش Blindbox مبتنی بر فشرده‌گشایی کامل [۵]	روش پیشنهادی
نتیجه آزمون معنا داری- زمان اجرا	بله	بله



شکل ۱۰: نسبت نرخ وقوع اشاره‌گر به نرخ کاراکترهای پویش شده به ازای هر ۵۰۰ فایل



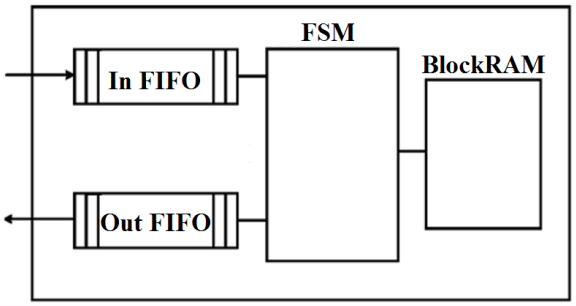
شکل ۱۱: هیستوگرام نرخ وقوع اشاره‌گر

برای درک بهتر از عملکرد روش پیشنهادی، طی یک آزمایش برای ۵۰۰ وبگاه برتر، نرخ کاراکترهای پیمایش شده توسط NFA و نرخ کاراکترهایی که از روی آن‌ها پریده می‌شود، محاسبه شده است. این آزمایش نشان می‌دهد که ۵۶ درصد از کاراکترها توسط NFA پیمایش شده و ۴۴ درصد از کاراکترها پیمایش نشده‌اند و جزء توکن‌های تکراری بوده‌اند. این آمار نشان می‌دهد که تنها با پرش از روی کمتر از نصف کاراکترها، زمان اجرای استخراج توکن‌های رمزنگاری جستجوپذیر نزدیک به ۶۵ درصد کاهش می‌یابد.

۷- ارزیابی سخت‌افزار مورد نیاز برای روش پیشنهادی

علاوه بر ارزیابی نرم‌افزاری روش پیشنهادی که در بخش قبل به آن پرداخته شد، الگوریتم معرفی شده در بخش ۵-۱ و NFA طراحی شده در بخش ۵-۲ بر روی رشته‌های LZ77 بر روی بورد ML605 شرکت زایلینکس با استفاده از ISE 14.7 پیاده‌سازی سخت افزاری شدند. شِمای کلی سخت‌افزار پیاده‌سازی شده در شکل ۱۲ نمایش داده شده است. در این شکل، رشته‌های LZ77 به صورت بایت به بایت از ورودی دریافت

شده و در صف FIFO ورودی (In FIFO) ذخیره می‌گردند. سپس ماشین حالت، آن‌ها را به ترتیب دریافت کرده و توکن‌ها را استخراج و از طریق صف FIFO خروجی (Out FIFO) ارسال می‌کند. به منظور افزایش سرعت دسترسی به حافظه، از حافظه BlockRAM داخل FPGA استفاده شد. سخت‌افزار مورد نیاز برای پیاده‌سازی بخش استخراج توکن‌ها در سه روش BlindBox، روش بدون پرش و روش پیشنهادی در جدول ۴ نمایش داده شده است. از بررسی نتایج جدول ۴ مشاهده می‌شود که تعداد بلوک حافظه استفاده شده توسط سه روش با هم برابر است و حجم سخت‌افزار مورد نیاز در روش پیشنهادی کاهش یافته است هرچند که این کاهش چندان چشم‌گیر نیست. اما به علت پرش از روی توکن‌های تکراری، تعداد کلاک‌های مورد نیاز برای استخراج توکن به میزان قابل توجهی کاهش یافته است و در نتیجه، میزان گذردهی به میزان زیادی افزایش یافته است.



شکل ۱۲: بلاک دیاگرام سخت افزار پیشنهادی

جدول ۴: نتایج پیاده‌سازی سخت‌افزاری

	روش Blindbox مبتنی بر فشرده گشایی کامل[۵]	روش بدون پرش	روش پیشنهادی
تعداد LUT	۴۴۸۷	۴۳۴۵	۴۰۳۲
تعداد بلوک حافظه	۱۰۰	۱۰۰	۱۰۰
میزان گذردهی (Mbps)	۴۸	۵۵/۰۴	۲۸۸

۸- نتیجه‌گیری

ترافیک رمز شده بخش قابل توجهی از ترافیک شبکه را تشکیل می‌دهند. برای بازرسی محتوایی ترافیک رمز شده از پروتکل رمزنگاری جستجوپذیر استفاده می‌شود. ۵۳ درصد از ترافیک وب، ترافیک فشرده شده HTTPS است. در این نوع ترافیک، بدنه بسته قبل از رمزنگاری، فشرده شده است. بنابراین بدون فشرده‌گشایی نمی‌توان توکن‌های مورد نیاز برای رمزنگاری جستجوپذیر را استخراج نمود. برای استخراج توکن‌ها از ماشین حالت NFA استفاده می‌شود. سربار زمانی استخراج توکن‌ها ناشی از پیمایش NFA است. در این مقاله روشی برای بهبود زمان استخراج توکن‌ها ارائه شد.

در روش پیشنهادی مقاله از وجود توکن‌های تکراری برای بهبود استفاده شد. توکن‌های تکراری به واسطه اشاره‌گرهای LZ77 و ذخیره اطلاعاتی در حین پیمایش NFA تشخیص داده شدند. بدین ترتیب با تشخیص موقعیت توکن‌های تکراری، از پیمایش NFA بر آن‌ها خودداری شد که در نهایت موجب کاهش سربار زمان استخراج توکن شد. برای ارزیابی، روش پیشنهادی و دو روش دیگر پیاده‌سازی شدند و برای ۵۰۰ وبگاه برتر معرفی شده توسط Alexa، آزمایش شدند. نتایج نشان داد در بیش از ۸۰ درصد فایل‌ها نرخ وقوع اشاره‌گر LZ77 بیش از ۰/۸ است و همانطورکه گفته شد، توکن‌های تکراری در این اشاره‌گرها نهفته‌اند. همچنین تعداد اشاره‌گرها با تعداد کاراکترهای پیمایش شده رابطه خطی دارد. طبق ارزیابی‌ها بطور متوسط ۴۴ درصد از کاراکترها به دلیل وجود توکن‌های تکراری در اشاره‌گر، توسط NFA پویش نمی‌شوند و این امر باعث می‌شود که زمان مورد نیاز برای استخراج توکن‌های رمزنگاری جستجوپذیر ۶۵ درصد نسبت به روش Blindbox مبتنی بر فشرده گشایی کامل[۵]، کاهش یابد.

در پژوهش‌های آینده می‌توان از نتایج این پژوهش برای بهبود سرعت استخراج توکن‌های رمزنگاری جستجوپذیر برای محتوای فشرده نشده نیز استفاده نمود. به این ترتیب که محتوای فشرده نشده را به فرم LZ77 فشرده نمود و سپس با استفاده از روش پیشنهادی این پژوهش، از پیمایش NFA بر توکن‌های تکراری جلوگیری کرد. در نهایت با داشتن توکن‌های منحصربه‌فرد (غیرتکراری) و ارائه پروتکل رمزنگاری جستجوپذیر مناسب می‌توان سرعت بازرسی محتوایی و پهنای باند مصرفی را نیز کاهش داد.

مراجع

[1] Snort: The Open Source Network Intrusion Detection System, https://snort.org/ [Accessed September 20, 2016].

[2] HTTPS at Google, https://www.google.com/transparencyreport/https/ [Accessed September 20, 2016].

[3] Speed Web Delivery with HTTP Compression, https://www.ibm.com/developerworks/library/wa-httpcomp/ [Accessed January 6, 2018].

[4] Usage of Site Elements for Websites, https://w3techs.com/technologies/overview/siteelement/all [Accessed January 6, 2018].

[5] J. Sherry, C. Lan, R. Ada Popa and S. Ratnasamy, "BlindBox: Deep Packet Inspection over Encrypted Traffic," *SIGCOMM '15 Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, New York, NY, USA, 2015.

[6] P. Linz, *An Introduction to Formal Languages and Automata*, Jones & Bartlett Learning, 5th Edition, 2011.

[7] F. Yu, Z. Chen, Y. Diao, T. V. Lakshman and R. H. Katz, "Fast and memory-efficient regular expression matching for deep packet inspection," *ACM/IEEE Symposium on Architecture for Networking and Communications systems*, San Jose, CA, USA, 2006.

[8] M. Becchi, A. Bremler-Barr, D. Hay, O. Kochba and Y. Koral, "Accelerating regular expression matching over compressed HTTP," *IEEE Conference on Computer Communications (INFOCOM)*, Hong Kong, 26 April - 1 May 2015.

[9] P. Deutsch, "*RFC 1952: GZIP file format specification version 4.3,"* Aladdin Enterprises, Network Working Group, May 1996.

[10] P. Deutsch, "RFC 1951: *DEFLATE compressed data format specification version 1.3*," Aladdin Enterprises, Network Working Group, May 1996.

[11] Usage Statistics of Compression for Websites, https://w3techs.com/technologies/details/ce-compression/all/all [Accessed September 20, 2016].

[12] D. E. Knuth, "Dynamic huffman coding," Journal of Algorithms, vol. 6, no. 2, pp. 163-180, 1985.

[13] Z. Jacob and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337-343, 1977.

[14] C. Lan, J. Sherry, R. Ada Popa, S. Ratnasamy and Z. Liu, "Embark: securely outsourcing middleboxes to the cloud," *13th USENIX Symposium on Networked Systems Design and Implementation*, Santa Clara, CA, March 16–18, 2016.

[15] X. Yuan, X. Wang, J. Lin and C. Wang, "Privacy-preserving deep packet inspection in outsourced middleboxes," *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, San Francisco, CA, 10-15 April 2016.

[16] P. Nishad and S. Sankar, "Efficient random sampling statistical method to improve big data compression ratio and pattern matching techniques for compressed data," *International Journal of Computer Science and Information Security*, vol. 14, no. 9, pp. 179-184, 2016.

[17] A. Bremler-Barr and Y. Koral, "Accelerating multipattern matching on compressed HTTP traffic," *IEEE/ACM Transactions on Networking* , vol. 20, no. 3, pp. 970 - 983, 2012.

[18] A. Bremler-Barr, K. Yaron and Z. Victor, "Shift-based pattern matching for compressed web traffic," *12th International Conference on High Performance Switching and Routing.*, Cartagena, Spain, 4-6 July 2011.

[19] W. Sun and U. Manber, "*A fast algorithm for multi-pattern searching*," Technical Report TR-94-17, Department of Computer Science, University of Arizona, 1996.

[20] H. Peng, J. Li, B. Li and M. H. Arif, "Fast multi-pattern matching algorithm on compressed network traffic," China Communic, vol. 13, no. 5, pp. 141-150, 2016.

[21] Alexa Site, http://www.alexa.com/, [Accessed January 6, 2018].

[22] Zlib Site, https://zlib.net/, [Accessed January 6, 2018].

[۲۳] عاتکه گشوارپور، عطاله عباسی و عاطفه گشوارپور، «بررسی تفاوت‌های پاسخ به تحریکات تصویری دارای بار احساسی در زنان و مردان با استفاده از آزمون آماری ویلکاکسون». مجله مهندسی برق دانشگاه تبریز، دوره ۴۷، شماره ۲، صفحات ۶۹۵-۶۸۷، ۱۳۹۶.

[۲۴] وحید رافع و سجاد اسفندیاری، «راهکاری نوین جهت تولید دنباله آزمون کمینه در فرآیند آزمون نرم افزار با ترکیب الگوریتم‌های جستجوی تپه نوردی و جستجوی خفاش»، مجله مهندسی برق دانشگاه تبریز، دوره ۴۶، شماره ۳، صفحات ۳۵-۲۵، ۱۳۹۵.

زیرنویس‌ها

[1] Deep Packet Inspection(DPI)
[2] Regular Expression
[3] Secure Socket Layer(SSL)
[4] Transport Layer Security (TLS)
[5] Huffman
[6] http compression
[7] Searchable Encryption
[8] Tokens
[9] Nondeterministic Finite Automata
[10] Attributes
[11] Deterministic Finite Automata
[12] cache
[13] Internet Assigned Numbers Authority
[14] Parental filtering
[15] exfiltration
[16] Intrusion detection system(IDS)
[17] Secure computation
[18] Encrypted Filter