شناسایی باج‌افزارها و خانوادهٔ آن‌ها با بهره‌گیری از روش کاوش الگوهای متوالی در تحلیل پویا

حمید دارابیان[۱]، ستار هاشمی[۲]، سجاد همایون[۳] و کرم‌الله باقری‌فرد[*۴و۵]

[۱و۲] دانشکده مهندسی برق و کامپیوتر، دانشگاه شیراز، شیراز، ایران

[۳] دانشکده مهندسی فناوری اطلاعات و کامپیوتر، دانشگاه صنعتی شیراز، شیراز، ایران

[۴] گروه مهندسی کامپیوتر، واحد یاسوج، دانشگاه آزاد اسلامی، یاسوج ، ایران

[۵]باشگاه پژوهشگران جوان و نخبگان، واحد یاسوج ، دانشگاه آزاد اسلامی، یاسوج ، ایران

چکیده

امروزه باج‌افزارهای رمزکننده تبدیل به یکی از مهم‌ترین تهدیدات حوزه سایبری شده است. یک باج‌افزار رمزکننده با رمزکردن داده‌های با ارزش قربانی، دسترسی به داده‌ها را از بین می‌برد و در ازای رمزگشایی آن‌ها درخواست پرداخت باج می‌کند. به‌علت نوظهوربودن باج‌افزارهای رمزکننده، پژوهش چندانی در جهت شناسایی آن‌ها انجام نشده است و بیش‌تر پژوهش‌های مرتبط روی سیستم فایل و نظارت بر رفتار فرآیندها روی فایل‌ها انجام شده است. از آن‌جایی که سرعت در تشخیص باج‌افزارها اهمیت فراوانی دارد، تمرکز این مقاله روی تشخیص دقیق و زودهنگام باج‌افزارها بر اساس تحلیل لاگ‌های رفتاری است. در این مقاله، ابتدا محیط آزمایشی مناسبی را ایجاد می‌کنیم تا بتوانیم رفتار ۵۷۲ نمونهٔ باج‌افزار از خانوادهٔ TeslaCrypt، ۵۳۵ نمونهٔ باج‌افزار از خانواده Cerber و ۵۱۷ نمونه باج‌افزار از خانواده Locky را ثبت کنیم که محیط مهیا شده قابلیت کاربرد در سایر پروژه‌ها و پژوهش‌های مشابه را دارد. برای دسته‌بندی و شناسایی نمونه‌های باج‌افزار، با بهره‌گیری از روش کاوش الگوهای متوالی، ویژگی‌هایی را به‌دست می‌آوریم تا قابل استفاده برای الگوریتم‌های دسته‌بندی‌کنندهٔ یادگیری ماشین باشد. دقت ۹۹٪ در تشخیص نمونه‌های باج‌افزار و همین طور دقت ۹۶.۵٪ در شناسایی و دسته‌بندی خانوادهٔ آن‌ها روی الگوریتم‌های متداول یادگیری ماشین نشان از کیفیت بالای ویژگی‌های پیشنهادی دارد.

واژگان کلیدی: بدافزار، باج‌افزار، باج‌افزار رمزکننده، شناسایی باج‌افزار، شناسایی خانوادهٔ باج‌افزار

Detecting Ransomware and Identifying their Families Using Sequence Mining in Dynamic Analysis

Hamid Darabian[1], Sattar Hashemi[2*], Sajad Homayoun[3] & Karamollah Bagherifard[*4,5]

[1,2] Faculty of Computer Scince and Engineering, Shiraz University, Shiraz, Iran

[3]Faculty of Information Technology and Computer Engineering, Shiraz University of Technology, Shiraz, Iran

[4]Department of Computer Engineering, Yasooj Branch, Islamic Azad University, Yasooj, Iran

[5]Young Researchers and Elite Club, Yasooj Branch, Islamic Azad University,Yasooj, Iran

Abstract

Nowadays, crypto-ransomware is considered as one of the most threats in cybersecurity. Crypto ransomware removes data access by encrypting valuable data and requests a ransom payment to allow data decryption. The number of Crypto ransomware variants has increased rapidly every year, and ransomware needs to be distinguished from the goodware types and other types of ransomware to protect users' machines from ransomware-based attacks. Most published works considered System File and process behavior to identify ransomware which depend on how quickly and accurately system logs can be obtained and mined to detect abnormalities. Due to the severity of irreparable damage of ransomware attacks, timely detection of ransomware is of great importance. This paper focuses on the early detection of ransomware samples by analyzing behavioral logs of programs executing on the

* نویسندهٔ عهده‌دار مکاتبات

* Corresponding author

● تاریخ ارسال مقاله: ۱۳۹۸/۲/۸ ● تاریخ پذیرش: ۱۳۹۹/۵/۲۸ ● تاریخ انتشار: ۱۴۰۰/۱۰/۲۷ ● نوع مطالعه: کاربردی

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]

[DOI: 10.52547/jsdp.18.3.29]

operating system before the malicious program destroy all the files. Sequential Pattern Mining is utilized to find Maximal Sequential Patterns of activities within different ransomware families as candidate features for classification. First, we prepare our test environment to execute and collect activity logs of 572 TeslaCrypt samples, 535 Cerber ransomware, and 517 Locky ransomware samples. Our testbed has the capability to be used in other projects where the automatic execution of malware samples is essential. Then, we extracted valuable features from the output of the Sequence Mining technique to train a classification algorithm for detecting ransomware samples. 99% accuracy in detecting ransomware instances from benign samples and 96.5% accuracy in detecting family of a given ransomware sample proves the usefulness and practicality of our proposed methods in detecting ransomware samples.

Keywords: malware, ransomware, crypto ransomware, ransomware detection, ransomware family detection.

۱- مقدمه

امروزه جرایم سایبری[1]، تهدیدات بسیار جدی‌ای را برای دولت‌ها، کسب‌وکارها و کاربران حقیقی و حقوقی به‌وجود آورده است. افراد بداندیش روزبه‌روز در حال توسعه ابزارهای خطرآفرین هستند و در طرف مقابل متخصصان امنیتی تلاش می‌کنند که از این فعالیت‌ها جلوگیری به‌عمل آورند [1]. حجم، حوزه و هزینه جرایم سایبری در سال‌های اخیر رشد قابل توجهی داشته است و نرم‌افزارهای بداندیش یا بدافزارها[2] همیشه به‌عنوان یکی از مهمترین ابزارهای جرایم سایبری مطرح بوده‌اند [2]. بر اساس گزارش سیمنتک [3] تعداد برنامه‌های مخرب از ۳۵۵ میلیون در سال ۲۰۱۵ به ۳۵۷ میلیون در سال ۲۰۱۶ افزایش داشته است [3]. توسعه پول‌های الکترونیکی مثل بیت‌کوین[4] منجر به ظهور گونه‌ای از بدافزارها به نام باج‌افزار[5] شده است [4]. باج‌افزار نوعی از بدافزار است که با بهره گیری از تکنیک‌های مخرب از دسترسی کاربران به سیستم‌هایشان یا داده‌هایشان جلوگیری می‌کند [5]. تاریخچه پیدایش باج‌افزارها به سال ۱۹۸۹ برمی‌گردد، زمانی‌که نخستین باج‌افزار تحت عنوان AIDS منتشر شد [6]. اما در آن زمان به‌علت عدم فراهم‌بودن زیرساخت‌های لازم جهت گمنام‌سازی[6] موفقیت چندانی برای آن‌ها حاصل نشد؛ چون زیرساخت‌های گمنام‌سازی ارتباط در کنار پول‌های الکترونیکی گمنام از ملزومات و خواسته‌های حمله‌کننده است. باج‌افزارها از طریق روش‌های مختلفی انتشار پیدا می‌کنند و به‌صورت کلی فرایند انتشارشان شامل چند مرحله مختلف است که البته می‌توان با شناسایی آن‌ها و بلاک‌کردن آنها جلوی آلوده‌سازی را گرفت. برای مثال هنگامی که یک باج‌افزار از طریق رایانامه منتشر می‌شود، ممکن است، صدها هزار رایانامه آلوده به باج‌افزار به‌وسیلهٔ سامانه‌های تشخیص هرزنامه[7] بلاک شوند. بیشتر رایانامه‌های شامل باج‌افزار دارای یک دانلودکننده[8] مخفی درون محتوای پیوست‌شده به رایانامه‌ها هستند که وظیفه دانلود نسخهٔ اجرایی باج‌افزار را بر عهده دارد.

به‌طور کلی باج‌افزارها را می‌توان به دو دستهٔ باج‌افزارهای قفل‌کننده[9] سیستم و باج‌افزارهای رمزکننده[10] تقسیم کرد. دسته نخست رایانامه و سیستم قربانی را قفل می‌کنند و فایل‌های موجود در سیستم را دست‌کاری نمی‌کنند؛ درحالی‌که دسته دوم تمام یا قسمتی از داده‌ها را با استفاده از الگوریتم‌های رمزنگاری مثل AES رمز می‌کنند و سپس یک دستورالعمل پرداخت[11] به قربانی نمایش داده می‌شود [7]. باج‌افزارهای رمزکننده به‌طورمعمول محبوب‌تر از باج‌افزارهای قفل‌کننده هستند چرا که به‌طورمعمول مهندسان امنیت می‌توانند راهی را برای بازکردن قفل سیستم پیدا کند [8]؛ بنابراین تمرکز اصلی این مقاله بر روی باج‌افزارهای رمزکننده است.

در این مقاله تمامی نمونه‌ها از خانواده‌های مختلف باج‌افزارها در یک محیط شبیه‌سازی‌شده اجرا و دنبالهٔ فراخوانی‌های سیستمی آن‌ها ثبت می‌شود. با استفاده از روش کاوش الگوهای متوالی بهترین ویژگی‌ها برای جداسازی نمونه‌های باج افزار از نمونه‌های غیرمخرب، شناسایی می‌شود. از جمله مزایای روش پیشنهادی این است که باج افزارها می‌توانند با تعداد کمی ویژگی (شامل ۲۷ ویژگی) تشخیص داده شوند. دقت ۹۹ درصدی بعد از اعمال تعدادی از الگوریتم‌های یادگیری ماشین و در تشخیص روی ویژگی‌های یافت‌شده، نشان از کیفیت بالای این ویژگی‌ها دارد.

[1] Cyber Criminals
[2] Malware
[3] Symantec
[4] Bitcoin
[5] Ransomware
[6] Anonymity

[7] Spam Detection Systems (Anti-Spam Systems)
8 Downloader
[9] Locker
[10] Crypto
[11] Payment Instruction

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]

[DOI: 10.52547/jsdp.18.3.29]

بخش ۲ این مقاله به بررسی پژوهش‌های انجام‌شده به‌وسیلهٔ سایر پژوهش‌گران می‌پردازد. بخش ۳ مراحل جمع‌آوری داده‌ها را به‌عنوان یک مرحله کلیدی شرح می‌دهد. فرآیند پیشنهادی جهت استخراج ویژگی‌ها با جزئیات در بخش ۴ شرح داده شده است. بخش ۵ متریک‌های مورد نظر جهت ارزیابی نتایج را معرفی و شرح می‌دهد و در بخش ۶ نتایج به‌دست‌آمده روی تفکیک باج‌افزارها از نرم‌افزارهای غیرمخرب مورد بحث قرار گرفته‌اند در حالی که بخش ۷ نتایج اجرای روش پیشنهادی برای تعیین خانواده باج‌افزارها را ارائه می‌دهد. درنهایت بخش ۸ به جمع‌بندی و نتیجه‌گیری می‌پردازد.

۲- پژوهش‌های پیشین

بسیاری از رویکردهای تشخیص باج‌افزارها وابسته هستند به سیستم فایل و تغییراتی که در کلیدهای رجیستری[1] انجام می‌شود. *Kharaz* [9] و همکاران تعداد ۱۳۵۹ نمونه از باج‌افزارها را بررسی کردند و نشان دادند که بیش‌تر باج‌افزارها از توابع کتابخانه‌ای مشابهی استفاده می‌کنند و به‌طور کلی استراتژی‌های یکسانی را برای آلوده‌سازی و حمله به فایل‌های کاربران دنبال می‌کنند. CryptoDrop [10] به‌عنوان یک سیستم شناسایی فرآیندهای مخرب رمزکنندهٔ فایل‌های کاربر معرفی شده است. CryptoDrop با یک محدودیت کلیدی مواجه است که دست‌کم باید حدود ده فایل کاربر به‌وسیلهٔ باج‌افزار رمزشده باشند تا توانایی تعیین مخرب‌بودن آن مهیا شود که این خود منجر به ازدست‌رفتن داده‌های این فایل‌ها می‌شود که ممکن است در بردارنده داده‌های حساسی باشند. ShieldFS [11]روشی برای شناسایی باج‌افزارها براساس نحوهٔ استفاده از توابع رمزنگاری و همچنین فایل‌های رمز شده است و درنهایت تصمیم می‌گیرد که یک نمونه فرآیند در حال اجرا مخرب است یا بی‌خطر است. DAD [12] با مقایسه آنتروپی[2] فایل‌ها قبل و بعد از عملیات آلوده‌سازی (رمزگذاری فایل) قصد شناسایی یک باج‌افزار را دارد که بر اساس ارزیابی‌های انجام‌شده توسط پیشنهاددهندگان برای رسیدن به دقت ۹۹/۳۷ درصد شناسایی باج‌افزارها به‌طورتقریبی هفتاد مگابایت از داده‌های فایل‌ها را از دست داده است. روش EldeRan [13] از ترکیب آنالیز ایستا و پویا برای شناسایی باج‌افزارها بهره برده است. در این روش برای آنالیز ایستا از رشته‌های موجود در فایل‌های دودویی و برای آنالیز پویا از فعالیت‌های کلیدهای رجیستری، عملیات روی فایل‌ها و دایرکتوری‌ها، عملیات ساخت فایل‌ها به‌وسیلهٔ فرآیندها در طول اجرای سی ثانیه استفاده می‌کند و درنهایت حضور یا عدم حضور هر رشته و فعالیت‌های پویا را به‌صورت ویژگی‌های باینری در نظر می‌گیرد که باعث می‌شود تعداد ویژگی‌ها به‌طور قابل ملاحظه‌ای افزایش پیدا کند. Redemption [14] به‌عنوان یک سیستم شناسایی باج‌افزارها سعی در استخراج ویژگی‌هایی مبتنی بر محتوا و مبتنی بر رفتار دارد که این ویژگی‌ها بیشتر روی سیستم فایل تمرکز دارند، ویژگی‌هایی نظیر آنتروپی، بازنویسی فایل‌ها و عملیات حذف فایل‌ها در نظر گرفته شده‌اند.

تشخیص به‌موقع باج‌افزار در زمان اجرا بسیار اهمیت دارد و سیستم‌هایی که در کمتر از ده ثانیه نتوانند بدافزارها را شناسایی کنند به‌صورت مؤثر در نظر گرفته نمی‌شوند [5]. همین‌طور این‌که شناسایی به موقع خانواده باج‌افزارها می‌تواند متخصصان را در ساختن عامل‌های هوشمند کاربردی یاری کند.

در این مقاله، ما از تکنیک کاوش الگوهای دنباله‌ای[3] استفاده می‌کنیم تا بتوانیم ویژگی‌های[4] قابل استفاده برای الگوریتم‌های دسته‌بندی‌کنندهٔ[5] یادگیری ماشین[6] را به‌دست بیاوریم که برای جداکردن نمونه‌های باج‌افزار از نمونه‌های غیرمخرب[7] و همین‌طور شناسایی خانواده‌ی باج‌افزارها مناسب باشند. این تکنیک را بر روی الگوی فعالیت‌های رجیستری[8]، سیستم‌فایل[9] و DLL[10] نمونه‌ها در زمان اجرا اعمال کردیم که مدت زمان اجرای نمونه‌ها کمتر از ده ثانیه بوده است. با انجام آزمایش بر روی ۵۱۷ عدد نمونه باج‌افزار از خانوادهٔ Locky، ۵۳۵ عدد باج‌افزار از خانوادهٔ Cerber، ۵۷۲ عدد از خانوادهٔ TeslaCrypt و ۲۲۰ عدد نمونه غیرمخرب ویندوزی، همین‌طور با بهره‌گیری از الگوریتم‌های دسته‌بندی‌کنندهٔ متداول یادگیری ماشین شامل J48، Random Forest، Bagging و MLP، کیفیت ویژگی‌های به‌دست‌آمده را مورد سنجش قرار می‌دهیم. دقت ۹۹ درصد در جداسازی بدافزارها از نمونه‌های غیر مخرب و دقت ۹۶/۵ درصد در شناسایی خانوادهٔ آن‌ها، نشان از مناسب‌بودن کیفیت ویژگی‌های به‌دست‌آمده دارد.

[1] Registry Keys
[2] Entropy
[3] Mining Sequential Patterns
[4] Feature
[5] Classification
[6] Machine Learning
[7] Non-malicious
[8] Registry Activities
[9] System File
[10] Dynamic Linked Library

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]
[DOI: 10.52547/jsdp.18.3.29]

۳- آماده‌سازی داده‌ها

در ابتدا به تعداد ۱۶۲۴ عدد نمونه باج‌افزار را از تارنمای virustotal.com دانلود کردیم که مخرب‌بودن این نمونه‌ها در تارنمای RansomwareTracker.abuse.ch در بازه زمانی فوریه ۲۰۱۶ تا مارس ۲۰۱۷ گزارش شده است. مجموعه باج‌افزارها شامل ۵۱۷ عدد نمونه باج‌افزار از خانوادهٔ Locky، ۵۳۵ عدد باج‌افزار از خانوادهٔ Cerber، ۵۷۲ عدد از خانوادهٔ TeslaCrypt که همگی فایل‌های اجرایی ویندوز هستند. همین‌طور تعداد ۲۲۰ عدد نرم‌افزار بی‌خطر از فهرست portableapps.com جمع‌آوری شده است که تمام این نرم‌افزارها، برنامه‌های قابل حمل[1] exe هستند.

محیطی به‌طورکامل کنترل‌شده طراحی و پیاده‌سازی شده است تا بتوانیم رفتارهای نمونه‌های باج‌افزارها و نمونه‌های بی‌خطر را ثبت کنیم. معماری بستر آزمایش[2] در شکل (۱) نمایش داده شده است. بستر آزمایش دارای دو قسمت اصلی کنترل‌کننده[3] و عامل اجراکننده[4] است. وظایف عامل کنترل‌کننده شامل مشخص‌کردن مدت زمان اجرای آزمایش، ارسال نمونه به عامل اجراکنندهٔ مستقر در ماشین مجازی[5]، بازگرداندن اسنپ‌شات[6] ماشین مجازی بعد از هر بار اجرای برنامه نمونه است. همین‌طور عامل اجراکننده وظیفه دریافت نمونه را از مخزن موجود در FTP دارد تا با اجرای نمونه باج‌افزار، فعالیت‌های ناشی از نمونه مورد آزمایش در محیط سیستم‌عامل با استفاده از ابزار لاگ‌گیری ثبت شود. برای لاگ‌گیری از ابزار Process Monitor استفاده می‌کنیم که قابلیت ثبت فعالیت‌های یک فرایند روی فایل‌ها، کلیدهای رجیستری ویندوز، مانیتورکردن[7] زیرفرایند[8]ها/زیر نخ[9]ها و غیره را دارد. جدول (۱) نشان‌دهنده فهرستی از سه نوع اصلی از فعالیت‌های انجام‌شده به‌وسیلهٔ فرآیندهای سیستم عامل از قبیل بارگذاری[10] DLL ها، رفتارهای مختلف روی فایل‌ها و فعالیت روی کلیدهای رجیستری است که قصد داریم با کمک ابزار Process Monitor به ثبت فعالیت‌های آن‌ها بپردازیم. نکته اینکه فهرست موجود در جدول (۱) شامل تمام رفتارهای منحصربه‌فرد انجام‌شده به‌وسیلهٔ نمونه‌های مورد آزمایش در این پروژه است و این یعنی رفتارهایی که هرگز دیده نشده‌اند در جدول (۱) آورده نشده‌اند؛ زیرا هیچ کمکی به‌صورت مسأله این مقاله نخواهند داشت. اجرای نمونه‌های باج‌افزارها با نمونه‌های بی‌خطر با یکدیگر تفاوت عمده‌ای دارد، چرا که نمونه‌های باج‌افزارها پس از اجرا و بدون نیاز به دخالت کاربر فعالیت‌های مخرب خود را انجام می‌دهد؛ درحالی‌که به‌طورمعمول نمونه‌های غیر مخرب به‌علت وجود پنجره گرافیکی نیاز به تعامل کاربر وجود دارد تا بتوانیم فعالیت‌های برنامه را ثبت کنیم. به همین‌علت ابزاری به نام PyWinMonkey[11] طراحی و پیاده‌سازی کرده‌ایم که به‌صورت تصادفی[12] رفتار کاربر را در تعامل با پنجره گرافیکی برنامه بی‌خطر شبیه‌سازی می‌کند و این کار را تا زمانی انجام می‌دهد که یا مدت زمان آزمایش به پایان برسد یا خطاهای برنامه در اجرای صحیح نرم‌افزار یافته شود.

جهت اجرای بستر آزمایش از ویندوز ۱۰ با سریال تولید ۱۰۲۴۰ روی سیستم رایانه‌ای با پردازندهٔ هشت هسته‌ای با قدرت ۴ گیگاهرتز و حافظه اصلی شانزده گیگابایت استفاده شده است. از نسخهٔ ProcessMonitor V3.31 جهت لاگ برداری فعالیت‌های فرآیندها استفاده شده است. از آن‌جا که رفتار کلی هر فرآیند از طریق دنباله‌ای[13] از فعالیت‌های پشت سر هم تشکیل می‌شود؛ لذا در این مقاله از مفهوم دنباله جهت نمایش فعالیت‌های یک فرآیند بهره گرفته می‌شود. گفتنی است که داده‌های اولیه (لاگ‌ها) استخراج‌شده از بستر آزمایش به‌صورت یک فایل به‌ازای هر آزمایش است که در مرحله نخست نیاز به ساخت دنباله فعالیت‌ها از فایل لاگ‌ها است و این یعنی پس از تبدیل به‌ازای هر فایل لاگ یک دنباله افقی نشان‌دهنده فعالیت‌های انجام‌گرفته به‌صورت ترتیبی موجود خواهد بود که درنهایت مجموعه این دنباله‌ها مجموعه‌دادگان $D = \{S_1, S_2, \dots, S_n\}$ را می‌سازند که S_i نشان‌دهنده دنباله ترتیبی مرتب‌شده در واحد زمان از فعالیت‌ها است.

هر دنباله S_i که به‌صورت عبارت ۱ است، نشان‌دهندهٔ یک دنباله است که دارای m فعالیت است و m می‌تواند برای Sهای مختلف متفاوت باشد که نشان‌دهنده تعداد فعالیت‌های انجام‌شده متفاوت به‌وسیلهٔ فرآیندها است. $A_{i,j}(argA_j)$ نشان‌دهنده فعالیت j اُم از دنباله i اُم (S_i) است. هر فعالیت انجام‌شده روی یک مسیر

[1] Portable
[2] Test Bed
[3] Controller
[4] Launcher
[5] Virtual Machine
[6] Snapshot
[7] Monitoring
[8] Sub-process
[9] Sub-thread
[10] Load

[11] https://github.com/sajadhomayoun/PyWinMonkey
[12] Random
[13] Sequence

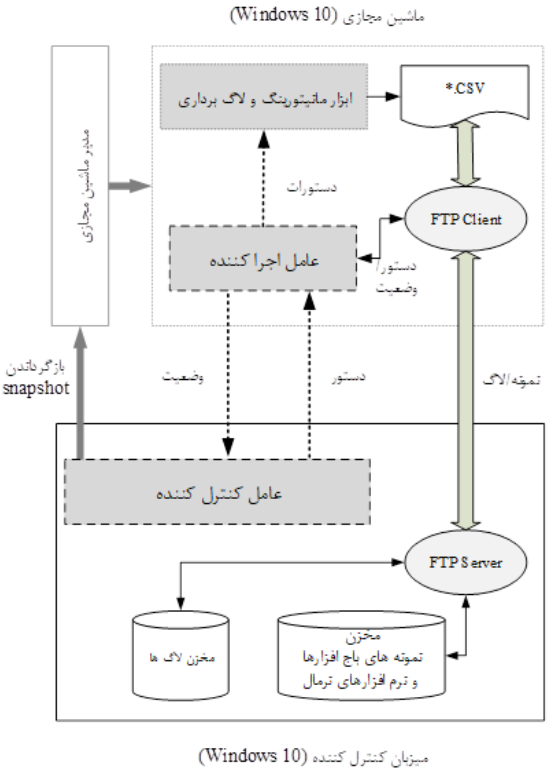
[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]
[DOI: 10.52547/jsdp.18.3.29]

در سیستم انجام می‌گیرد که با $argA_j$ نمایش داده شده و بیان‌کننده این است که فعالیت A_j روی مسیر $argA_j$ که تداعی‌کننده آرگمان[1] ورودی به فعالیت است، انجام گرفته است.

$$S_i = \{A_{i,1}(argA_1), A_{i,2}(argA_2), \dots, A_{i,m}(argA_m)\} \quad (۱)$$

به‌عنوان مثال عبارت:

{LoadImage(system32\gdi32.dll),ReadFile(Windo ws\SysWOW64\wininet.dll)} نشان‌دهنده یک دنباله از دو فعالیت که نخستین فعالیت LoadImage است که فایل system32\gdi32.dll را در حافظه فرآیند فراخوانی‌کننده بارگذاری می‌کند و سپس فایل wininet.dll از مسیر Windows\SysWOW64\ خوانده شده است. اندازه هر دنباله وابسته به تعداد فعالیت‌هایی است که در طول اجرای برنامه انجام شده و بین برنامه‌های مختلف، متفاوت است.



(شکل-۱): بسترآزمایش طراحی شده جهت اجرای نمونه‌ها

(Figure-1): Designed test bed to execute samples

جدول (۲) نشان‌دهنده مجموعه‌دادهٔ نهایی و تعداد دنباله در هر مجموعه‌داده است. به‌صورتی‌که D_x به مجموعه‌داده x اشاره می‌کند که شامل دنباله‌های رفتاری توسط نمونه‌های از خانواده x تولید شده‌اند.

D_{Locky} نشان‌دهنده تمام دنباله‌های به‌دست‌آمده از اجرای نمونه‌های باج‌افزارهای از خانواده Locky در بسترآزمایش است. D_{Cerber} نمایش‌گر دنباله‌های تولیدشده به‌وسیلهٔ باج‌افزارهای خانواده Cerber است. $D_{TeslaCrypt}$ دربر گیرنده دنباله‌های به‌دست‌آمده از اجرای نمونه‌های TeslaCrypt در بسترآزمایش است. درنهایت D_{Benign} نشان‌دهنده دنباله‌های تولیدشده از اجرای نرم‌افزارهای بی‌خطر است. جهت اینکه تخمینی واقع‌بینانه از مدل‌های پیش‌بینی ساخته‌شده در این بخش وجود داشته باشد و از عدم مشکل به‌وجودآمدن بیش‌برازش[2] اطمینان حاصل کرد از مجموعه‌داده‌هایی استفاده می‌شود که شامل دنباله‌هایی باشند که به‌طورکامل برای مدل ساخته‌شده جدید باشند. از همین رو به‌ازای هر کدام از خانوادهٔ باج افزارها و برنامه‌های غیرمخرب از همان ابتدا به‌صورت تصادفی تعدادی از دنباله‌ها انتخاب و در دیتاستی جداگانه به نام D_{OP} قرار داده شدند.

(جدول-۱): فهرست فعالیت‌های استخراج‌شده از رفتار نمونه‌های تحت آزمایش

(Table-1): List of activities can be captured by process montitor

ردیف	نوع فعالیت	لیست
۱	Registry Key	RegQueryKey, RegOpenKey, RegCloseKey, ,RegQueryValue RegCreateKey, RegSetInfoKey, RegEnumKey, RegQueryKeySecurity, RegEnumValue, RegSetValue, RegDeleteValue, RegQueryMultipleValueKey, RegLoadKey, ,RegDeleteKey RegFlushKey
۲	Filesystem	QueryNameInformationFile, ReadFile, CreateFile, QueryBasicInformationFile, CloseFile, QueryStandardInformationFile, CreateFileMapping, QuerySizeInformationVolume, FileSystemControl, QueryDirectory, WriteFile, QueryNetworkOpenInformationFile, QueryRemoteProtocolInformation, QuerySecurityFile, LockFile, UnlockFileSingle, DeviceIoControl, SetEndOfFileInformationFile, FlushBuffersFile, SetAllocationInformationFile, SetBasicInformationFile, QueryAttributeTagFile, QueryFileInternalInformationFile, QueryInformationVolume, QueryAttributeInformationVolume, SetRenameInformationFile, QueryNormalizeNameInformationFile, NotifyChangeDirectory, QueryFullSizeInformationVolume, SetSecurityFile, QueryStreamInformationFile, SetDispositionInformationFile,

[1] Argument

[2] Overfit

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]

[DOI: 10.52547/jsdp.18.3.29]

		QueryEaInformationFile, QueryAllInformationFile, QueryIdInformation, SetPositionInformationFile, QueryPositionInformationFile, SetValidDataLengthInformationFile
۳	DLL	LoadImage

حال که مجموعهٔ دنباله‌های هر برنامه مخرب و غیر مخرب به‌دست آمد، روش معرفی‌شده در بخش بعدی به شرح نحوهٔ استخراج ویژگی‌ها می‌پردازد.

(جدول-۲): مجموعه‌داده‌های ساخته‌شده
(Table-2): Created datasets

شماره ردیف	مجموعه داده	تعداد دنباله در هر مجموعه داد
۱	D_{Locky}	۴۵۰
۲	D_{Cerber}	۴۷۰
۳	$D_{TeslaCrypt}$	۵۰۷
۴	D_{Benign}	۲۰۰
۵	D_{OP}	۱۷۴

۴- استخراج ویژگی و بردارسازی

با بهره‌گیری از بستر آزمایش طراحی‌شده در قسمت پیش تمامی نمونه‌های بی‌خطر بارگیری شده و تمامی نمونه‌های باج‌افزارها و نرم‌افزارهای بی‌خطر اجرا شده و فعالیت‌های آن‌ها ثبت و ضبط شده است و تمامی فعالیت‌های هر نمونه تحت قالب یک دنباله نشان داده می‌شود. پس از اتمام فرآیند ساخت دنباله‌ها، از الگوریتم FindFPOF [15] جهت شناسایی و حذف دنباله‌های پرت استفاده شده است. تعداد کمی از الگوریتم‌های شناسایی نقاط پرت[1] توانایی کار روی دنباله‌ها را دارند که FindFPOF[2] یکی از آن روش‌ها به شمار می‌آید و از روش کاوش دنباله‌ها بهره می‌گیرد [16]. FindFPOF از معیار FPOF[3] جهت شناسایی و حذف نقاط پرت استفاده می‌کند؛ بدین صورت که دنباله‌های با مقادیر FPOF پایین به‌عنوان نقاط پرت معرفی می‌کند.

جهت شناسایی بهترین ویژگی‌ها برای دسته‌بندی باج‌افزارها و نرم‌افزارهای بی‌خطر نیاز به تعریف الگوهایی از فعالیت‌ها که قابل شناسایی در دنباله‌ها باشند، است؛ بنابراین از الگوریتم‌های کاوش الگوی متوالی برای به‌دست‌آوردن الگوهای متوالی بیشینه (MSP)[4] بهره گرفته، سپس عملیات پیمایش دنباله‌ها در مجموعه‌دادگان انجام می‌شود و با توجه به MSPها عملیات بردارسازی و محاسبه ویژگی‌ها انجام می‌گیرد.

کاوش الگوهای متوالی، تمامی زیردنباله‌ها[5] (الگوهای متوالی) موجود در یک مجموعه‌دادگان شامل داده‌های دنباله‌ای را طوری شناسایی می‌کند که تعداد تکرار آن‌ها در مجموعه‌دادگان بزرگتر مساوی کمینه مقدار آستانه[6] تعیین‌شده از سوی کاربر است. یک دنباله α را که به‌صورت $\{a_1, a_2, \dots, a_{Len(\alpha)}\}$ است، در نظر بگیرید که a_i نمایش‌گر عنصر iاُم باشد و $Len(\alpha)$ نمایش‌گر طول دنباله α است. زیردنباله یک دنباله دیگر $\beta = \{b, b_2, \dots, b_{Len(\beta)}\}$ است؛ درصورتی‌که اعداد صحیحی به‌صورت $1 \le j_1 < j_2 < \cdots < j_{Len(\alpha)} \le Len(\beta)$ وجود داشته باشند که $a_1 \subseteq b_1, a_2 \subseteq b_2, \dots, a_{Len(\alpha)} \subseteq b_{Len(\alpha)}$ باشد. اگر min_{sup} نشان‌دهنده کمینه تکرار مورد نظر جهت مشخص‌کردن یک الگوی مکرر باشد و شرط $sup_\alpha \ge min_{sup}$ برقرار باشد، می‌توان از α به‌عنوان یک الگوی متوالی مکرر (MSP) نام برد، جایی که sup_α نمایش‌گر تعداد رخداد α در مجموعه‌دادگان D است.

روش‌های مختلفی برای پیاده‌سازی روش کاوش الگوهای متوالی ارائه شده [17] مثل GSP، AprioriAll، SPADE و PrefixSpan که این روش‌ها اغلب به‌دلیل طولانی‌بودن زمان پردازش و مصرف حافظه زیاد ممکن است برای مجموعه‌داده‌های بزرگ ناکارامد باشند [18]. در این مقاله برای تولید MSPها از راه‌کاری به نام MG-FSM استفاده شده است که یک راهکار پردازش موازی[7] مبتنی بر Map-Reduce [19] ارایه می‌کند که به‌راحتی قابلیت پیاده‌سازی روی زیرساخت ابری را با هدف مقیاس‌پذیری در کاوش مجموعه‌دادگان بزرگ فراهم می‌آورد. مهم‌ترین ورودی این الگوریتم مجموعه‌دادگان دنباله‌ها و مقدار min_{sup} است [16]. در الگوریتم‌های کاوش الگوهای دنباله‌ای مقدار min_{sup} روی نتیجهٔ نهایی تأثیر فراوانی دارد. مقادیر پایین برای min_{sup} ممکن است، مجموعهٔ عظیمی از MSPها تولید کند (MC بزرگ) که این امر زمان اجرای الگوریتم بردارسازی[8] پیشنهادی را افزایش

[1] Outlier
[2] Find Frequent Pattern Outlier Factor
[3] Find Pattern Outlier Factor
[4] Maximum Sequential Pattern
[5] Sub-sequence
[6] Threshold
[7] Parallel Processing
[8] Vectorization

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]
[DOI: 10.52547/jsdp.18.3.29]

می‌دهد. از سوی دیگر مقادیر بالا برای min_{sup} ممکن است MSPهای با ارزشی را حذف کند که می‌توانند در شناسایی و تفکیک باج‌افزارها مؤثر باشند. از همین رو در این مقاله min_{sup} به مقدار پنجاه درصد تنظیم شده است تا کارایی قابل قبولی در شناسایی MSPها به‌دست آید.

به‌کاربردن جداگانه MG-FSM روی مجموعه‌دادگان D_{Locky}، D_{Cerber} و $D_{TeslaCrypt}$ موجب تولید $MC_{D_{Locky}}$، $MC_{D_{Cerber}}$، $MC_{D_{TeslaCrypt}}$ می‌شود؛ به‌طوری‌که MC_D به شکل فرمال در فرمول (۲) قابل مشاهده است و P_x و P_y نشان‌دهنده یک زیردنباله هستند.

$$MC_D = \left\{ \left(P_x, sup_{P_x}\right) \middle| \begin{matrix} sup_{P_x} \geq min_{sup} \\ \wedge \\ \forall P_x \left(\nexists P_y \left(P_x \subseteq P_y\right)\right) \end{matrix} \right\} \qquad (۲)$$

جهت تعریف انواع MSPها در ابتدا سه نوع MSP اتمیک[1] تعریف شده است و سپس شش نوع MSP تک‌انتقالی[2] را در جدول (۳) تعریف می‌شود. MSPهای اتمیک نشان‌دهنده فعالیت‌های از یک نوع به‌صورت پشت سر هم است؛ به عبارت دیگر یک MSP اتمیک است، اگر تمامی فعالیت‌های موجود در آن از یک نوع باشند. به‌عنوان مثال در جدول (۳) مشخص است که F نشان‌دهنده MSPهایی است که تمامی فعالیت‌هایشان از نوع کار با سیستم فایل‌ها باشد.

MSPهای تک‌انتقالی نشان‌دهنده انتقال (گذار)[3] از یک نوع فعالیت به نوع دیگری از فعالیت‌ها است. به‌عنوان مثال RD نشان می‌دهد که ابتدا دنباله از یک یا چند فعالیت رجیستری (R) رخ داده و سپس یک یا چند فعالیت DLL (D) انجام شده است. به‌عنوان مثالی از انتقال دوتایی FRD را در نظر بگیرید که نشان‌دهنده انتقال از F به R و سپس از R به D است. هر چند به‌طورعمومی تعداد ویژگی‌های بیشتر ساخت مدل پیش‌بینی‌کننده و تفکیک‌کنندهٔ نمونه‌ها ساده‌تر و امکان‌پذیرتر می‌کند؛ اما تعداد ویژگی‌های بیش از اندازه، داده‌های مجموعه‌داده را در فضای ابعاد اسپارس می‌کند و در نتیجه یافتن بهترین صفحه جداکننده نمونه‌ها امری دشوار می‌شود. این موضوع تحت عنوان مسأله نفرین ابعاد[4] [20] مطرح می‌شود و بیان می‌کند که افزایش تعداد ابعاد موجب افزایش فضا شده به‌طوری که داده‌های موجود به‌سرعت در محیط نمودار پراکنده می‌شوند.

(جدول-۳): فهرست انواع MSP
(Table-3): List of MSP types

نوع MSP	شرح
R	تمام فعالیت‌ها از نوع رجیستری باشد
F	تمام فعالیت‌ها از نوع سیستم فایل باشد
D	تمام فعالیت‌ها از نوع بارگذاری فایل‌های DLL باشد
RF	MSP دارای یک یا چند نوع انتقال بین فعالیت‌های انواع مختلف است در حالی که اولین انتقال رخ داده در آن انتقالی از فعالیت رجیستری به فعالیت سیستم فایل می‌باشد.
RD	MSP دارای یک یا چند نوع انتقال بین فعالیت‌های انواع مختلف است در حالی که اولین انتقال رخ داده در آن انتقالی از فعالیت رجیستری به فعالیت DLL می‌باشد.
FR	MSP دارای یک یا چند نوع انتقال بین فعالیت‌های انواع مختلف است در حالی که اولین انتقال رخ داده در آن انتقالی از فعالیت سیستم فایل به فعالیت رجیستری می‌باشد.
FD	MSP دارای یک یا چند نوع انتقال بین فعالیت‌های انواع مختلف است در حالی که اولین انتقال رخ داده در آن انتقالی از فعالیت سیستم فایل به فعالیت DLL می‌باشد.
DR	MSP دارای یک یا چند نوع انتقال بین فعالیت‌های انواع مختلف است در حالی که اولین انتقال رخ داده در آن انتقالی از فعالیت DLL به فعالیت رجیستری می‌باشد.
DF	MSP دارای یک یا چند نوع انتقال بین فعالیت‌های انواع مختلف است در حالی که اولین انتقال رخ داده در آن انتقالی از فعالیت DLL به فعالیت سیستم فایل می‌باشد.

با توجه به مطالب مطرح‌شده می‌توان تعداد کل ویژگی‌های موجود در بردار را با استفاده از فرمول (۳) به‌دستآورد. جایی که عدد ۳ نشان‌دهنده تعداد نوع فعالیت‌های در نظر گرفته شده است (F، R و D) و x بیان‌گر تعداد گذار مورد نظر در عملیات بردارسازی است. در توضیح این فرمول باید گفت که اگر $t = a, b, c$ یک MSP باشد و برای درنظرگرفتن انتقال‌های ۲تایی a می‌تواند یکی از سه حالت ممکن (F، R یا D) باشد و از آن جا که برای به‌وجودآمدن یک انتقال باید نوع فعالیت جدید با قبلی متفاوت باشد ($b \neq a$) یعنی دو نوع فعالیت مجاز برای b موجود است؛ درحالی‌که برای انتقال بعدی باید $c \neq b$ باشد و این یعنی دو نوع فعالیت ممکن برای c. در نتیجه بخش 3×2^i از فرمول (۳) تشکیل می‌شود. به‌عنوان مثال برای $x = 3$ که به معنای درنظرگرفتن تک انتقالی، دو انتقالی و سه انتقالی‌ها است براساس فرمول عبارت ۳ باید ۴۵ ویژگی در بردار باشد. بنابر مطالب گفته‌شده جهت جلوگیری از اسپارس‌شدن نمونه‌ها در این مقاله فقط انتقال‌های یگانه و دوتایی در استخراج ویژگی‌ها در نظر گرفته شده‌اند.

[1] Atomic
[2] One Step Transition
[3] Transition
[4] Curse of Dimensionality Problem

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]
[DOI: 10.52547/jsdp.18.3.29]

$$TotalFeatures = 3 + \sum_{i=1}^{x}(3 \times 2^i) \quad (۳)$$

با درنظرگرفتن $P = \{E_1, E_2, \dots, E_n\}$ به‌عنوان یک MSP اتمیک تعریف می‌شود اگر شرط $\forall_{E_x,E_y \in P \wedge E_x \neq E_y}(EventType(E_x) = EventType(E_y))$ برقرار باشد. همچنین P یک MSP تک‌انتقالی است؛ اگر شرط $\exists_{E_x,E_y \in P \wedge E_x \neq E_y}(EventType(E_x) \neq EventType(E_y))$ برقرار باشد. $EventType(x)$ نشان‌دهنده یک روال است که با دریافت یک فعالیت x نوع آن را با توجه به جدول (۳) مشخص می‌کند؛ به‌علاوه می‌توان مجموعه‌ای از انواع ممکن MSP در $MSPTypeSet = \{F, R, D, FR, FD, RF, RD, DF, DR\}$ مجموعهٔ تعریف کرد که الگوریتم MSPType() در شکل (۲) به‌عنوان خروجی یکی از عناصر مجموعه EventTypeSet را باز می‌گرداند.

نرخ پشتیبانی[1] (SR) برای یک MSP مقداری در بازه [0,1] است که نشان‌دهنده احتمال رخداد آن MSP در مجموعه‌داده باجافزارها است و از طریق تقسیم تعداد رخداد MSP (sup_{MSP}) بر تعداد کل (۷) دنباله‌های باجافزارهای مجموعه‌داده D به‌دست می‌آید. برای هر دنباله S یک بردار با ۹ ویژگی تعریف می‌شود که هر ویژگی مقدار SR برای آن نوع MSP و با توجه به مجموعه MSPها (MC) محاسبه می‌شود. قالب بردار ویژگی یادشده به‌صورت فرمول (۴) است.

$$Vector(S)_{MC} = \{SR_R, SR_F, SR_D, SR_{FR}, SR_{RF}, SR_{RD}, SR_{DF}, SR_{DR}\} \quad (۴)$$

مقدار SR برای هر نوع MSP از یک دنباله با توجه به MC مورد نظر می‌تواند با استفاده از الگوریتم $CalculateSR()$ در شکل (۳) محاسبه می‌شود؛ پس از آن که یک بردار به‌ازای هر دنباله مجموعه‌داده محاسبه شد درنهایت مجموعه‌داده برداری $VD_{D,MC}$ را موجود خواهد داشت که $VD_{x,y}$ نشان‌دهندهٔ نسخه‌برداری شده مجموعه‌داده x براساس مجموعه MSPهای موجود در y است. همچنین این پروژه جهت شناسایی خانواده‌های باجافزارها مفهوم آَبربردار[2] (SV) را مطرح می‌کند که از قرارگرفتن چند بردار در کنار یکدیگر بردار بزرگتری تشکیل می‌دهد. به عبارت دیگر اگر بردارسازی با توجه به MCهای مختلف انجام گیرد می‌توان آن‌ها را کنار یکدیگر قرار داد و بردار بزرگتری ساخت که شکل (۴) نشان‌دهنده ساختار یک آَبربردار است.

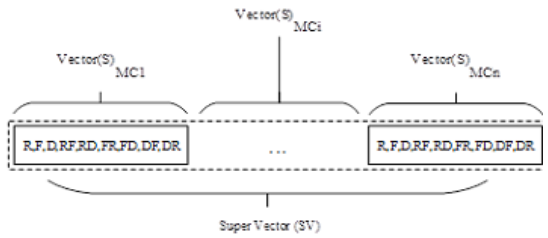
[1] Support Ratio
[2] Super Vector

```
1.  Procedure MSPType(MSP P)
2.      for all (E_x, E_y ∈ P) ∧ (x ≤ i) ∧ (i, y ≤ n)
3.          if EventType(E_x) == EventType(E_y)
            then
4.              if EventType(E_x) == R return R
5.              if EventType(E_x) == F return F
6.              if EventType(E_x) == D return D
7.          Else
8.              if EventType(E_x) == R &
                EventType(E_x) == F return RF
9.              if EventType(E_x) == R &
                EventType(E_x) == D return RD
10.             if EventType(E_x) == F &
                EventType(E_x) == R return FR
11.             if EventType(E_x) == F &
                EventType(E_x) == D return FD
12.             if EventType(E_x) == D &
                EventType(E_x) == R return DR
13.             if EventType(E_x) == D &
                EventType(E_x) == F return DF
14.         end if
15.     end for
16. end procedure
```

(شکل-۲): الگوریتم MSPType جهت تعیین نوع یک MSP
(Figure-2): MSPType algorithm to determine a given MSP

```
1.  Procedure CalculateSR(Sequence S, MSP Collection
    MC, MSPType_Set)
2.  SR_{P_Total} = 0
3.      for all (P ∈ MC) do
4.          if (P ⊆ S) & (MSPType(P) == T) then
5.              SR_{P_Total} = SR_{P_Total} + (sup_P / γ)
6.          end if
7.      end for
8.      return SR_{P_Total}
9.  end procedure
```

(شکل-۳): الگوریتم محاسبه SR برای دنبالهٔ S
(Figure-3): SR Calculation algorithm for sequence S



(شکل-۴): شکل یک ابر بردار ساخته شده از سه MC مختلف
(Figure-4): A super vector made of three different MC

در شکل (۴)، $Vector(S)_{MC_i}$ نشان‌دهنده بردار به‌دست‌آمده از S با توجه به مجموعه MSPهای موجود در MC_i است. از آن جا که در این پروژه سه خانواده باجافزار در نظر گرفته شده‌اند، بنابراین برای هر S سه بردار ساخته می‌شود: $Vector(S)_{MC_{Locky}}$، $Vector(S)_{MC_{Cerber}}$ و $Vector(S)_{MC_{TeslaCrypt}}$ که تشکیل یک آَبربردار می‌دهند. با

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]

[DOI: 10.52547/jsdp.18.3.29]

توجه به اینکه هر بردار ۹ ویژگی دارد پس آبربردار نهایی دارای $9 \times 3 = 27$ ویژگی خواهد بود. پس از محاسبه آبربردارها برای Sهای موجود در مجموعه‌داده D، مجموعه‌داده ابربرداری شده SVD_D تشکیل خواهد شد.

۵- متریک‌های مورد استفاده جهت ارزیابی روش پیشنهادی

در این مقاله برای ارزیابی کارایی روش پیشنهادی از معیارهای شاخص کارایی دسته‌بندی‌کننده‌ها [21] شامل مثبت واقعی[1]، منفی واقعی[2]، مثبت کاذب[3] و منفی کاذب[4] هستند. معیار دقت[5] که به‌عنوان نرخ آشکارسازی مثبت هم شناخته می‌شود، میزان حساسیت مدل یا مقدار نتایج مرتبط برگردانده‌شده را نشان می‌دهد و برابر است با تعداد باج‌افزارهای تشخیص داده‌شده به‌وسیله مدل تقسیم بر کل تعداد باج‌افزارهای عضو در مجموعه آزمایش که از طریق فرمول (۵) محاسبه می‌شود. معیار بازخوانی[6] نشان‌دهندهٔ نسبت داده‌های مثبتی (باج‌افزارهایی) که به‌درستی به‌وسیلهٔ مدل تشخیص داده شده‌اند می باشد که از رابطهٔ (۶) محاسبه می‌شود. اندازه F-measure، میانگین هارمونیک معیارهای دقت و بازخوانی است و میزان کارایی دسته‌بندی‌کننده را نشان می‌دهد (فرمول ۷).

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \qquad (۵)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \qquad (۶)$$

$$\text{F-measure} = 2 * \frac{\text{Precision}*\text{Recall}}{\text{Precision+Recall}} \qquad (۷)$$

مشخصهٔ عملیاتی گیرنده[7] (ROC) یک معیار قوی برای مقایسه دسته‌بندی‌کننده‌ها به حساب می‌آید چرا که در برابر ناهم‌گونی طبقه‌های مجموعه‌داده ثابت و در این مقاله نیز گزارش شده است. در یک نمودار ROC نرخ مثبت واقعی بر اساس نرخ مثبت کاذب با آستانه‌ها مختلف کشیده می‌شود. سطح زیر نمودار ROC که به‌عنوان AUC شناخته می‌شود، پارامتر خوبی است که می‌تواند دو دسته‌بندی‌کننده را مقایسه کند. معیار AUC درواقع یک تک‌مقدار است که می‌توان گفت نمودار ROC را در یک عدد خلاصه کرده است. AUC مقداری بین صفر و یک دارد که مقدار یک بیان می‌کند که دسته‌بندی‌کننده عملکردی دقیق داشته است.

[1] True Positive
[2] True Negative
[3] False Positive
[4] False Negative
[5] Precision
[6] Recall
[7] Receiver Operating Characteristic

ضریب همبستگی متیوز[8] (MCC) [22] پارامتر دیگری از میزان کیفیت است که برای مقایسه دسته‌بندی‌کننده‌های مختلف به کار می‌رود. این اندازه برای زمانی که طبقه‌های مجموعه‌داده متعادل نیستند بهتر از دیگر معیارها عمل می‌کنند [23]. به‌عنوان مثال معیار F-measure زمانی که مدل عملکرد تصادفی دارد مقدار آنها ممکن است بیشتر ۰/۵ و این مقدار به سمت طبقه با داده‌های بیشتر هم‌گرا می‌شود درحالی‌که این مقدار برای ضریب MCC دقیقا صفر خواهد شد. مقدار این ضریب بین ۱- تا ۱+ تغییر می‌کند. در این‌جا مقدار ۱+ نشان‌دهندهٔ این است که دسته‌بندی‌کننده به‌صورت کاملا دقیق نمونه‌های آزمایش از هم تشخیص می‌دهد و مقدار ۱- بیان‌گر این است که دسته‌بندی‌کننده به‌صورت کاملا برعکس جواب داده است. مقدار صفر نشان می‌دهد دسته‌بندی‌کننده مورد ارزیابی دارای عملکردی تصادفی است؛ بنابراین برای نشان‌دادن آن‌که دسته‌بندی‌کننده‌های آموزش دیده‌شده، عملکردی به دور از تصادفی‌بودن داشته‌اند، مقدار MCC از طریق فرمول (۸) گزارش داده می‌شود.

$$\text{MCC} = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \qquad (۸)$$

۶- نتایج روش بردارسازی برای شناسایی باج‌افزارها

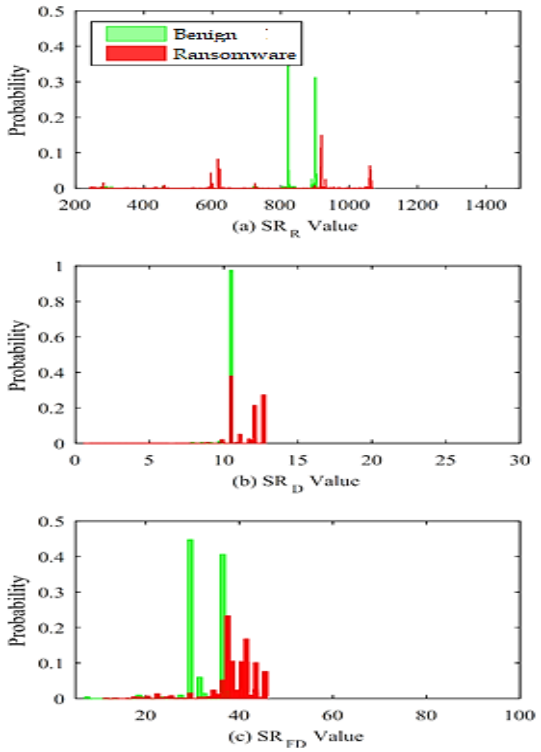
برای پیاده‌سازی قسمت پیش‌پردازش داده‌ها، بردارسازی و ساخت دسته‌بندی‌کننده‌ها از کتاب‌خانه‌های موجود در Python3.5 استفاده شده است. همین‌طور تمام ارزیابی‌های این مقاله به روش 10-Fold انجام شده است.

در روش بردارسازی برای شناسایی بهترین ویژگی‌ها در تفکیک باج‌افزارها از نرم‌افزارهای بی‌خطر، مجموعه‌داده D_{Total} با ادغام $D_{Ransomware}$ و D_{Benign} ساخته می‌شود؛ سپس $MC_{Ransomware}$ را با به‌کارگیری MG-FSM روی $D_{Ransomware}$ به‌دست آورده تا مهیای عملیات بردارسازی شود. اکنون با روش بردارسازی مطرح شده در بخش ۴ $VD_{D_{Total},MC_{Ransomware}}$ ساخته شده و آماده الگوریتم‌های آموزش دسته‌بندی‌کننده می‌شود. جهت انتخاب بهترین ویژگی‌ها در تفکیک باج‌افزارها و نرم‌افزارهای بی‌خطر از روش جستجوی گام به گام حریصانهٔ[9] CfsSubsetEval [24] استفاده شده است. پس از اجرای CfsSubsetEval سه ویژگی SR_R، SR_D و SR_{FD}

[8] Matthews Correlation Coefficient
[9] Greedy Stepwise Search

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]
[DOI: 10.52547/jsdp.18.3.29]

به‌عنوان بهترین ویژگی‌ها انتخاب شدند که توانایی ایجاد بهترین جداسازی بین نمونه‌های مثبت (باج‌افزار) و نمونه‌های منفی (نرم افزارهای بی‌خطر) را داشته‌اند. شکل (۵) نشان‌دهنده هیستوگرام ویژگی R است که مشخص می‌کند باج‌افزارها تمایل بیشتری به استفاده از کلیدهای رجیستری در ده ثانیه نخست اجرایشان داشته‌اند. شکل (۵) نشان می‌دهد بیشتر نرم‌افزارهای بی‌خطر رفتار DLLی مشابهی از خود نشان داده‌اند؛ در حالی که برای نمونه‌های باج‌افزار تغییرات بیشتری برای فعالیت‌های DLL مشاهده می‌شود؛ زیرا نمونه‌های بی‌خطر مقدار نرخ پشتیبانی (SR) کمتری نسبت به نمونه‌های باج افزار داشته‌اند. شکل (۵) بیان می‌کند که باج‌افزارها تمایل بیشتری در MSPهای دارای انتقال از فعالیت‌های مرتبط با سیستم فایل به فعالیت‌های مرتبط با DLLها دارند.



(شکل-۵): هیستوگرام‌های به‌دست‌آمده از ویژگی‌های انتخاب‌شده به‌وسیلهٔ الگوریتم CfsSubsetEval (SR: نرخ پشتیبانی)

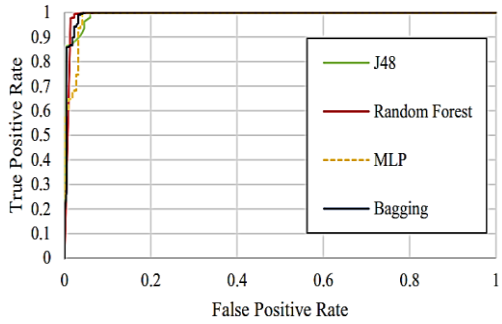
(Figure-5): Obtained histograms from CfsSubsetEval algorithm (SR: Support Ratio)

جدول (۴) نشان‌دهنده مقایسه کارایی دسته‌بندی‌کننده دودویی به‌دست‌آمده روی مجموعه‌داده $VD_{D_{Total},MC_{Ransomware}}$ است که فقط سه عدد ویژگی شامل R، D و FD دارد. تمامی الگوریتم‌ها به مقدار F-measure ۰/۹۹ درصد رسیده‌اند؛ درحالی‌که نرخ پیش‌بینی غلط نمونه‌های بی‌خطر برای همه الگوریتم‌ها کمتر از ۰/۰۴ بوده است (کمتر از ۴٪). به‌علاوه شباهت‌های موجود بین نمودارهای ROC ترسیم‌شده برای هر کدام از الگوریتم‌ها در شکل (۶) اثبات می‌کند که تفاوت بسیاری بین کارایی دسته‌بندی‌کننده‌های مختلف آموزش داده‌شده وجود ندارد که این خود نشان‌دهنده کیفیت ویژگی‌های به‌دست‌آمده در عملیات بردارسازی است. به‌عبارت دیگر نتایج مشابه برای دسته‌بندی‌کننده‌های مختلف در مسأله حاضر اثبات می‌کند که ویژگی‌های با کیفیت در یک مسأله یادگیری ماشین از اهمیت ویژه‌ای برخوردار هستند و تکنیک بردارسازی ارایه شده توانایی استخراج این ویژگی‌ها را دارد. شکل (۷) مقادیر AUC تمام دسته‌بندی‌کننده‌های مورد مطالعه را نشان می‌دهد که همگی دسته‌بندی‌کننده‌های مورد مطالعه مقادیر بیش از ۰/۹۹ دارند و دسته‌بندی‌کننده ساخته‌شده به‌وسیلهٔ Bagging مقدار AUC برابر با ۰/۹۹۵ دارد.

(جدول-۴): عملکرد دسته‌بندی‌کننده‌ها روی مجموعه‌داده $VD_{D_{Total},MC_{Ransomware}}$

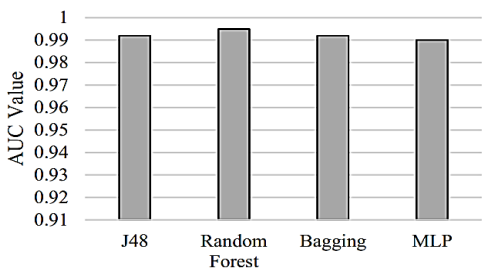
(Table-4): Performance classifiers on the $VD_{D_{Total},MC_{Ransomware}}$ dataset

F-Measure	FPR	TPR	دسته‌بندی‌کننده
0.994	0.040	0.994	J48
0.993	0.040	0.993	Random Forest
0.997	0.039	0.994	Bagging
0.997	0.035	0.994	MLP



(شکل-۶): نمودارهای AUC برای دسته‌بندی‌کننده‌ها

(Figure-6): AUC diagams of classifiers



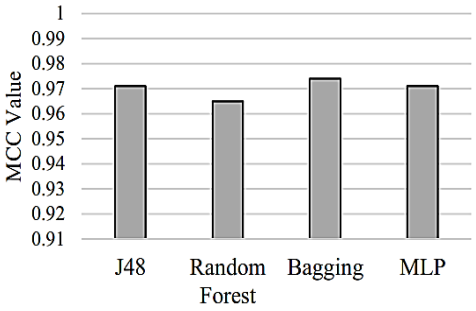
(شکل-۷): مقادیر AUC برای دسته‌کننده‌ها جهت شناسایی باج‌افزارها

(Figure-7): AUC values of the classifiers for ransomware detection

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]

[DOI: 10.52547/jsdp.18.3.29]

مقدار MCC در شکل (۸) برای تمامی دسته‌بندی‌کننده‌ها بیشتر از ۰/۹۶ است و RandomForest و Bagging به مقداری نزدیک به ۱+ رسیده‌اند که حکایت از نزدیکی به حالت ایده‌آل پیش‌بینی دارد.



(شکل-۸): مقادیر MCC برای دسته‌کننده‌ها جهت شناسایی باج‌افزارها

(Figure-8): MCC values of the classifiers for ransomware detection

جهت نمایش کارایی روش پیشنهادی با سایر پژوهش‌های انجام‌شده در حوزه شناسایی باج‌افزارها به مقایسه نتایج به‌دست‌آمده از دسته‌بندی‌کننده‌های ساخته‌شده در روش بردارسازی با روش ElderRan پرداخته می‌شود که در سال ۲۰۱۶ پیشنهاد شده است. بر اساس جدول (۴)، جدول (۵) روش MLP بهترین نتایج دسته‌بندی‌کننده دودویی روی ویژگی‌های استخراج‌شده به‌وسیلهٔ روش بردارسازی را تولید کرده است که نتایج MLP به‌عنوان بهترین نتیجه در جدول (۴) جهت مقایسه با نتایج به‌دست‌آمده در جدول (۵) درج شده‌است. نتایج به‌دست‌آمده از اجرای روش ElderRan نیز در جدول (۵) نشان داده شده است.

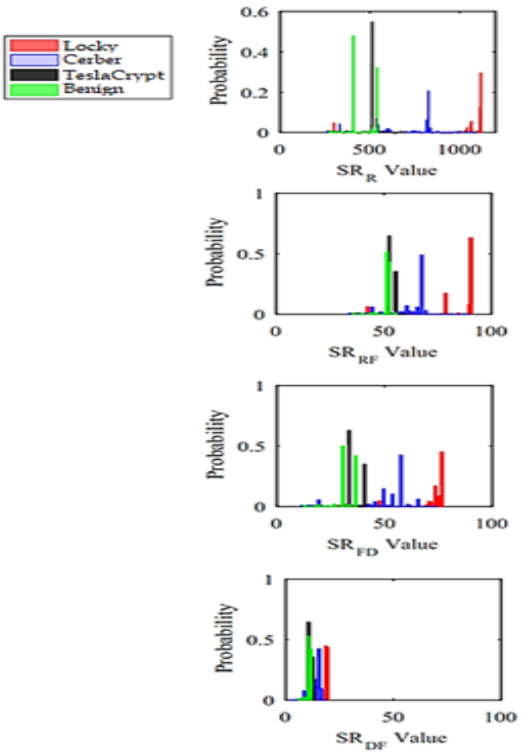
روش پیشنهادی بردارسازی قصد شناسایی باج‌افزارها در ۱۰ ثانیه نخست اجرای آن‌ها را دارد؛ درحالی‌که روش ElderRan شناسایی را در ۳۰ ثانیه نخست انجام می‌دهد. برای اینکه مقایسه صحیحی بین روش پیشنهادی و روش ElderRan قابل انجام باشد، روش ElderRan روی ۱۰ ثانیه نخست اجرای برنامه‌ها اعمال شده و نتایج به‌دست‌آمده در جدول (۵) نشان داده شده است. همان‌طور که مشخص است روش پیشنهادی بردارسازی مقدار F-Measure بالاتری نسبت به روش ElderRan به‌دست آورده است. هر چند مقادیر F-Measure به‌دست‌آمده نزدیک به یکدیگر هستند؛ اما با بررسی مقادیر FPR مشخص می‌شود که روش ElderRan در صورتی که روی ده ثانیه نخست اعمال شود مقدار FPR بسیار بالاتری نسبت به روش بردارسازی تولید کرده که این به معنای آن است که به‌خوبی توانایی تفکیک نمونه‌های بی‌خطر را ندارد و حدود ۱۴/۱ درصد از نمونه‌های بی‌خطر را به‌عنوان نمونه باج‌افزار شناسایی کرده که مقدار محسوسی در حوزه امنیت و دفاع است.

(جدول-۵): مقایسه بهترین نتیجه روش بردارسازی با روش ElderRan

(Table-5): Comparison the best result of proposed method with ElderRan approach

F-Measure	FPR	TPR	دسته‌بندی‌کننده
0.997	0.035	0.994	روش پیشنهادی بردارسازی (MLP)
0.980	0.141	0.980	ElderRan

برای نشان‌دادن اینکه دسته‌بندی‌کننده‌های ساخته‌شده روی داده‌های آزمایش دچار مشکل بیش‌برازش نشده‌اند از دیتاست D_{OF} بهره گرفته شده است که شامل نمونه‌هایی از دنباله هاست که به‌طورکامل برای مدل پیش‌بینی ساخته‌شده جدید و تازه هستند و حتی در فرآیند تولید MSPها توسط الگوریتم MG-FSM نیز شرکت نداشته‌اند. جدول (۶) نشان‌دهنده دقت شناسایی باج‌افزارها روی $VD_{D_{OF},MC_{Ransomware}}$ می‌باشد که نسخه بُرداری شده D_{OF} است. واضح است که همه دسته‌بندی‌کننده ۹۹/۴ درصد از باج‌افزارهای موجود در D_{OF} را شناسایی کرده‌اند.



(شکل-۹): هیستوگرام‌های ویژگی‌های به‌دست‌آمده از $VD_{D_{TotalFamily},MC_{Locky}}$ با الگوریتم CfsSubsetEval (SR: نرخ پشتیبانی)

(Figure-9): Histograms of obtained features from $VD_{D_{TotalFamily},MC_{Locky}}$ by using CfsSubsetEval algoithm (SR: Supprot Ratio)

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]
[DOI: 10.52547/jsdp.18.3.29]

(جدول-۶): نتایج به‌دست‌آمده از به کار بستن دسته‌بندی‌کننده‌های آموزش داده‌شده روی داده‌های دیده نشده

(Table-6): Obtained results of trained classifiers on the unseen dataset

دسته‌بندی‌کننده	J48	Random Forest	Bagging	MLP
TPR	0.994	0.994	0.994	0.994

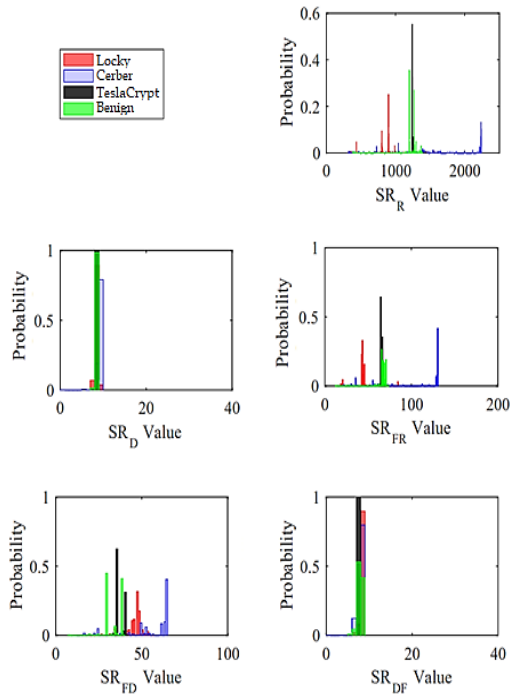
۷- شناسایی خانوادهٔ باج‌افزارها

درواقع شناسایی خانواده باج‌افزارها دارای اهمیت است؛ زیرا یکی از فعالیت‌های هوشمندی تهدید به‌شمار می‌آید که در آن در کنار شناسایی باج‌افزاربودن یک برنامه مخرب، هدف کسب اطلاعات بیشتر از جزئیات کارکردی از جمله اینکه باج افزار کشف‌شده متعلق به چه نوع از خانواده‌های درقبل ثبت‌شده است که بتوان از روش‌های درقبل طراحی‌شده برای مقابله و بازیابی از حمله استفاده کرد. ممکن است باج‌افزار از یک خانواده جدید باشد که این امر می‌تواند نشان‌دهنده ظهور یک نسل جدید و اعلام هشدارهای جدی جهت آمادگی مقابله را در پی داشته باشد.

برای ارزیابی کارایی دسته‌بندی‌کننده‌ها در شناسایی خانوادهٔ باج‌افزارها ابتدا $D_{TotalFamily}$ ساخته شده که ترکیبی از D_{Locky}، D_{Cerber}، $D_{TeslaCrypt}$ و D_{Benign} است و برچسب‌های نمونه‌ها به‌ترتیب Locky، Cerber، TeslaCrypt و Benign تنظیم شده‌اند.

$VD_{D_{TotalFamily},MC_{Locky}}$، $VD_{D_{TotalFamily},MC_{Cerber}}$ و $VD_{D_{TotalFamily},MC_{TeslaCrypt}}$ مجموعه‌داده‌های برداری هستند که به‌صورت جداگانه به الگوریتم انتخاب ویژگی CfsSubsetEval تحویل داده شده‌اند و در کل ۱۳ ویژگی به‌عنوان بهترین ویژگی‌ها در شناسایی و جداسازی خانواده‌های مختلف باج‌افزارها و نرم‌افزارهای بی‌خطر انتخاب شده‌اند. شکل‌های (۸، ۹ و ۱۰) به‌ترتیب هیستوگرام‌های ویژگی‌های انتخابی برای $VD_{D_{TotalFamily},MC_{Locky}}$، $VD_{D_{TotalFamily},MC_{Cerber}}$ و $VD_{D_{TotalFamily},MC_{TeslaCrypt}}$ را به تصویر کشیده‌اند.

با مقایسه شکل‌ها می‌توان دریافت که MSPهای اتمیک رجیستری (R) از اهمیت ویژه‌ای در تفکیک باج‌افزارهای خانواده‌های مختلف برخوردار است؛ زیرا در هر سه مورد Locky، Cerber و TeslaCrypt به‌عنوان یک ویژگی تأثیرگذار وجود دارد. فعالیت‌های انتقالی از سیستم فایل به DLL (FD) نیز در هر سه شکل به‌عنوان ویژگی مهم نشان داده شده است.



(شکل-۱۰): هیستوگرام‌های ویژگی‌های به‌دست‌آمده از $VD_{D_{TotalFamily},MC_{Cerber}}$ با الگوریتم CfsSubsetEval

(SR: نرخ پشتیبانی)

(Figure-10): Histograms of obtained features from $VD_{D_{TotalFamily},MC_{Cerber}}$ by using CfsSubsetEval algorithm (SR: Support Ratio)

از آن جا که شناسایی خانواده باج‌افزارها یک عملیات دسته‌بندی‌کننده با چند رده[1] است هر کدام از نمونه‌ها دارای یکی از برچسب‌های Locky، Cerber، TeslaCrypt یا Benign هستند. در انتها J48، Random Forest، Bagging و MLP با استفاده از $SVD_{D_{TotalFamily}}$ و با ۱۳ ویژگی انتخاب‌شده نهایی آموزش داده شده‌اند که جدول (۷) نشان‌دهنده کارایی دسته‌بندی‌کننده‌های مورد نظر در شناسایی خانواده‌های باج‌افزارها بر اساس ارزیابی 10-Fold است. F-measure به‌دست‌آمده ۰/۹۸۳ با FPR کوچکتر مساوی ۰/۰۰۶ حکایت از مناسب‌بودن ویژگی‌های محاسبه‌شده در شناسایی نمونه‌های سه نوع خانواده مورد بررسی است. مقادیر MCC بیشتر از ۰/۹۵ به‌دست‌آمده برای دسته‌بندی‌کننده‌های مختلف نشان از کیفیت بالای بردارهای ویژگی به‌دست‌آمده دارد.

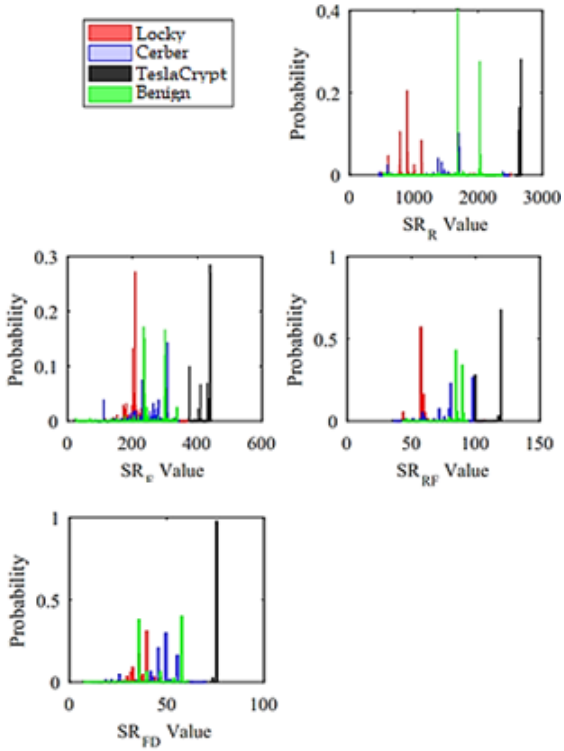
(جدول-۷): نتیجه دسته‌بندی‌کننده‌ها روی دیتاست $SVD_{D_{TotalFamily}}$

(Table-7): Results of classifiers on $SVD_{D_{TotalFamily}}$

دسته‌بندی‌کننده	TPR	FPR	F-Measure	MCC
J48	0.981	0.006	0.981	0.974
Random Forest	0.983	0.006	0.983	0.978
Bagging	0.980	0.007	0.980	0.974
MLP	0.980	0.007	0.980	0.973

[1] Multi-class Classification

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]
[DOI: 10.52547/jsdp.18.3.29]



(شکل-۱۱): هیستوگرام‌های ویژگی‌های به‌دست آمده از $VD_{D_{TotalFamily},MC_{TeslaCrypt}}$ با الگوریتم CfsSubsetEval

(SR: نرخ پشتیبانی)

(Figure-11): Histograms of obtained features from $VD_{D_{TotalFamily},MC_{TeslaCrypt}}$ by using CfsSubsetEval algorithm (SR: Support Ratio)

۸- نتیجه‌گیری

این مقاله به‌طور خاص روی شناسایی زود هنگام باج‌افزارها تمرکز داشته است و سعی بر این است تا عملیات شناسایی در بازه زمان تنظیم[1] باج‌افزارها که به‌طورمعمول در ابتدای شروع فعالیت انجام می‌گیرد انجام شود. نکته مهم دیگری که در این مقاله مطرح شده شناسایی خانواده باج‌افزارها با توجه به فعالیت‌های انجام‌شده است که اجرای نمونه‌ها (شامل باج‌افزارها و نمونه‌های بی‌خطر) در فضای بستر آزمایش پیاده‌سازی انجام می‌شود و این بستر توانایی اجرای خودکار برنامه‌ها در محیطی امن را دارد. این امر می‌تواند به خبرگان فضای امنیت سایبری در تحلیل رفتار بدافزارها کمک بسیار زیادی کند که تحت عنوان عملیات هوشمندی تهدید مطرح است. روش بردارسازی پیشنهادشده با ملاحظهٔ دسته‌های فعالیت‌ها از قبیل عملیات سیستم فایل، رجیستری و کار با DLL سعی در پیشنهاد ویژگی‌هایی با کیفیت بالا در دسته‌بندی باج‌افزارها دارد. به دلیل نوظهوربودن بدافزارهای باج‌گیر طبیعتا جای کار و نوآوری هنوز هم وجود دارد. به‌عنوان پیشنهاد ادامه کار می‌توان به کابرد روش‌های خوشه‌بندی در شناسایی و دسته‌بندی خانواده‌های مختلف باج‌افزارها اشاره کرد. باج‌افزارهای قفل‌کننده سیستم نیز می‌تواند مورد مطالعه قرار گیرند هرچند روند رشد و توسعهٔ آن‌ها مانند باج‌افزارهای رمزکننده نیست. روش‌هایپیشنهادی این مقاله می‌تواند روی باج‌افزارهای تلفن‌های هوشمند نیز مورد مطالعه قرار گیرند. همچنین با توجه به پیش‌بینی‌ها رشد و تهدید حوزه باج‌افزارها ادامه خواهد داشت و در سال‌های آینده قطعا به‌عنوان تهدیدی برای دنیای اینترنت اشیا[2] محسوب خواهد شد که نحوه برخورد با باج‌افزارها جذابیت زیادی برای پژوهش‌گران حوزه امنیت سایبری خواهد داشت. همین‌طور شرکت‌های تولیدکننده ضد بدافزار و حتی پژوهش‌گرانی که قصد مقابله با نسل‌های جدید باج‌افزارها و بدافزارها را دارند، می‌توانند از روش‌های پیشنهادی و نتایج این مقاله بهره ببرند.

9- References ۹- مراجع

[1] M. Hopkins and A. Dehghantanha, “Exploit Kits: The production line of the Cybercrime economy?,” *in 2015 2nd International Conference on Information Security and Cyber Forensics*, InfoSec 2015, 2016, pp. 23–27.

[2] Hosseini F, Mirzarezaee M, Sharifi A, "Malware Detection using Classification of Variable-Length Sequences," *JSDP*, vol. 16 (2), pp.137-146, 2019

[۲]حسینی فاطمه، میرزارضایی میترا، شریفی آرش. آشکارسازی بدافزارها با استفاده از دسته‌بندی دنباله‌های با طول متغیر .پردازش علائم و داده‌ها. ۱۳۹۸;۱۶(۲);۱۴۶-۱۳۷

[3] Symantec, “Internet Security Threat Report (ISTR),” no. April. p. 10, 2017.

[4] D. Palmer, “How Bitcoin helped fuel an explosion in ransomware attacks,” 2016. [Online]. Available: http://www.zdnet.com/article/how-bitcoin-helped-fuel-an-explosion-in-ransomware-attacks/.

[5] A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, “Detecting crypto-ransomware in IoT networks based on energy consumption footprint,” J. Ambient Intell. Humaniz. Comput., Aug. 2017.

[6] R. Richardson and M. M. North, “Ransomware : Evolution , *Mitigation and Prevention,” Int. Manag. Rev.*, vol. 13, no. 1, pp. 10–21, Jan. 2017.

[1] Setup Time

[2] Inter of Things

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]

[DOI: 10.52547/jsdp.18.3.29]

[7] K. Savage, P. Coogan, and H. Lau, "The Evolution of Ransomware," *Res. Manag.*, vol. 54, no. 5, pp. 59–63, 2015.

[8] Monika, P. Zavarsky, and D. Lindskog, "Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization," *in Procedia Computer Science*, 2016, vol. 94, pp. 465–472.

[9] E. Kirda, "UNVEIL: A large-scale, automated approach to detecting ransomware (keynote)," in usenix.org, 2017, pp. 1–1.

[10] N. Scaife, H. Carter, P. Traynor, and K. R. B. Butler, "CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data," *in Proceedings - International Conference on Distributed Computing Systems*, 2016, vol. Aug2016, pp. 303–312.

[11] A. Continella et al., "ShieldFS," *in Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC* 16, 2016, pp. 336–347.

[12] A. Palisse, A. Durand, H. Le Bouder, C. Le Guernic, and J. L. Lanet, "Data aware defense (DaD): Towards a generic and practical ransomware countermeasure," *in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017, vol. 10674 LNCS, pp. 192–208.

[13] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection," undefined, 2016.

[14] A. Kharraz and E. Kirda, "Redemption: Real-Time Protection Against Ransomware at End-Hosts," *in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017, vol. 10453 LNCS, pp. 98–119.

[15] Z. He, X. Xu, J. Z. Huang, and S. Deng, "A Frequent Pattern Discovery Method for Outlier Detection," Springer, Berlin, Heidelberg, 2010, pp. 726–732.

[16] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *in Proceedings of the Eleventh International Conference on Data Engineering*, 1995, pp. 3–14.

[17] C. H. Mooney and J. F. Roddick, "Sequential pattern mining -- approaches and algorithms," *ACM Comput. Surv.*, vol. 45, no. 2, pp. 1–39, Feb. 2013.

[18] Andrej Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," 2015. [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/. [Accessed: 30-May-2019].

[19] "What is Apache MapReduce? | IBM." [Online]. Available: https://www.ibm.com/analytics/hadoop/mapreduce. [Accessed: 30-May-2019].

[20] J. A. K. Suykens, "Introduction to Machine Learning," 2014, pp. 765–773.

[21] M. Sohrabi, M. M. Javidi, and S. Hashemi, "Detecting intrusion transactions in database systems: A novel approach," *J. Intell. Inf. Syst.*, vol. 42, no. 3, pp. 619–644, Jun. 2014.

[22] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric," *PLoS One*, vol. 12, no. 6, pp. e0177678, Jun. 2017.

[23] D. M. W. Powers, "Evaluation: From precision, recall and fmeasure to roc, informedness, markedness and correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2007.

[24] A. Hall, Mark, "Correlation-based feature selection for machine learning۱۹۹۹ ",.



حمید دارابیان، در سال ۱۳۹۳ مدرک کارشناسی در رشتهٔ مهندسی مهندسی فناوری اطلاعات از دانشگاه صنعتی شیراز و در سال ۱۳۹۷ مدرک کارشناسی ارشد خود را در رشته مهندسی فناوری اطلاعات-امنیت اطلاعات از دانشگاه شیراز دریافت کرد. زمینه‌های پژوهشی مورد علاقه ایشان امنیت فضای سایبری، رمزنگاری، کاربرد یادگیری ماشین و یادگیری عمیق در امنیت سیستم‌ها و شبکه‌های کامپیوتری است.

نشانی رایانامه ایشان عبارت است از:

h.darabian@cse.shirazu.ac.ir



ستار هاشمی، در سال ۱۳۷۷ مدرک کارشناسی در رشتهٔ مهندسی کامپیوتر-سخت‌افزار از دانشگاه صنعتی اصفهان و در سال ۱۳۸۰ مدرک کارشناسی ارشد خود را در رشته مهندسی کامپیوتر-هوش مصنوعی از دانشگاه علم و صنعت ایران و هم‌چنین مدرک دکترای خود را در رشته مهندسی کامپیوتر-هوش مصنوعی از دانشگاه علم و صنعت و موناش استرالیا دریافت کرد. وی هم‌اکنون عضو هیئت علمی و دانشیار دانشگاه شیراز و زمینه‌های پژوهشی مورد علاقه ایشان

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]

[DOI: 10.52547/jsdp.18.3.29]

یادگیری ماشین، داده‌کاوی، نظریه بازی‌ها و امنیت سایبری است.
نشانی رایانامه ایشان عبارت است از:
s_hashemi@shirazu.ac.ir



سجاد همایون، در سال ۱۳۹۶ از پایان‌نامه خود روی موضوع شناسایی باج افزارها با بهره‌گیری از تکنیک‌های مبتنی بر یادگیری ماشین و یادگیری عمیق در دانشگاه صنعتی شیراز دفاع کرده است. از ایشان تاکنون مقالات متعددی در مجلات و همایش‌های معتبر داخلی و خارجی را به چاپ رسیده است. زمینه‌های پژوهشی ایشان امنیت فضای سایبری و کاربرد یادگیری ماشین و یادگیری عمیق در امنیت سیستم‌ها و شبکه‌های کامپیوتری است.
نشانی رایانامه ایشان عبارت است از:
s.homayoun@sutech.ac.ir



کرم‌الله باقری‌فرد، در سال ۱۳۸۴ مدرک کارشناسی در رشتهٔ مهندسی کامپیوتر از دانشگاه اصفهان و در سال ۱۳۸۷ مدرک کارشناسی ارشد خود را در رشته مهندسی کامپیوتر- نرم‌افزار را دریافت کرد. ایشان مدرک دکترای خود را در سال ۱۳۹۶ در گرایش نرم‌افزار در دانشگاه اراک به اتمام رساند. زمینه‌های پژوهشی مورد علاقه ایشان یادگیری ماشین، داده‌کاوی و سیستم‌های پیشنهاددهنده است.
نشانی رایانامه ایشان عبارت است از:
k.bagheri@iauyasooj.ac.ir

[Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]

[DOI: 10.52547/jsdp.18.3.29]

[DOI: 10.52547/jsdp.18.3.29] [Downloaded from jsdp.rcisp.ac.ir on 2026-11-22]