مجله علمی پژوهشی «علوم و فناوری‌های پدافند غیرعامل»
سال چهارم، شماره ۳، پاییز ۱۳۹۲؛ ص ۱۵۵- ۱۴۵

طراحی و پیاده‌سازی موتور دگردیسی بدافزارها با رویکرد ارزیابی کارایی روش‌های شناسایی

مهران خسروی[۱]، سعید پارسا[۲*]

۱-کارشناس ارشد دانشگاه آزاد اسلامی واحد بروجرد ۲- دانشیار دانشگاه علم و صنعت ایران

(دریافت: ۱۳۹۱/۰۸/۰۹، پذیرش: ۱۳۹۲/۰۵/۲۰)

چکیده

یکی از اصول پدافند غیرعامل مقاوم‌سازی سامانه‌ها در مقابل حملات است. دسته بزرگی از حملات از طریق حمله بدافزارها به سیستم‌های کامپیوتری صورت می‌گیرد. باید میزان کارایی روش‌های موجود در مقابله با حملات بدافزارها مورد ارزیابی قرار بگیرند. یکی از رویکردها در این زمینه انجام حملات مدیریت شده توسط بدافزارهای تولید شده با موتورهای هوشمند است. اکثر محصولات ضد بدافزار از روش‌های شناسایی مبتنی بر امضای کد دودویی برای شناسایی بدافزارها استفاده می‌کنند. خانواده‌ای از بدافزارهای کامپیوتری به نام بدافزارهای دگردیس وجود دارد که در هر نسل امضای خود را با بهره‌گیری از روش‌های مبهم‌سازی تغییر می‌دهند. بنابراین با این روش مانع از تشخیص توسط روش‌های شناسایی مبتنی بر امضا دودویی بدافزار می‌شوند. در این مقاله موتور دگردیسی مبتنی بر اتوماتای سلولی یادگیر طراحی شده است که به دلیل ماهیت پویای آن علاوه بر توانایی مقابله با روش‌های شناسایی مبتنی بر امضای دودویی کد، به دلیل ایجاد کدهای مشابه برنامه‌های بی‌خطر با درصد تشابه بالا، قابلیت مقابله با روش‌های شناسایی مبتنی بر تحلیل آماری کدها را نیز دارد. این موتور می‌تواند ابزاری مناسب جهت بررسی کارایی سیستم‌های موجود در مقابله با حملات احتمالی باشد.

کلید واژه‌ها: **بدافزار دگردیس، امضای بدافزار، تحلیل آماری کد، موتور دگردیسی، مبهم‌سازی، اتوماتای سلولی یادگیر.**

Design and Implementation of a Metamorphic Engine Malware with Evaluation of Identifing Techniques Performance Approach

M. Khosravi, S. Parsa*

Islamic Azad University, Bruojerd Branch

(Received: 30/10/2012; Accepted: 11/08/2013)

Abstract

ONE OF PASSIVE DEFENCE PRINCIPLES IS THE IMMUNIZATION AGAINST THE ATTACKS TO COMPUTER SYSTEMS. LARGE SET OF ATTACK IS OCCURRED BY MALWARE TO COMPUTER SYSTEMS. PERFORMANCE OF EXISTING TECHNIQUES SHOULD BE EVALUATED AGA ATTACKS. IN THIS REGARDS, ONE OF THE APPROACHES IN THIS AREA IS TO PERFORM THE MANAGED ATTACKS BY THE PROD THROUGH INTELLIGENT ENGINES. MOST OF ANTI-MALWARE PRODUCTS MAY APPLY THE DETECTION TECHNIQUES BASE SIGNATURE CODE TO IDENTIFY THE MALWARE. A FAMILY OF COMPUTER MALWARE CALLED AS METAMORPHIC EXISTS WH HAS BEEN CHANGED IN EACH GENERATION THROUGH APPLYING THE OBFUSCATION TECHNIQUES SO THAT THEY CANNOT E BINARY MALWARE SIGNATURE BASED DETECTION TECHNIQUES. IN THIS PAPER, A METAMORPHIC ENGINE HAS BEEN PRESE BASIS OF LEARNING CELLULAR AUTOMATA. THIS ENGINE IS CAPABLE TO CONFRONT THE DETECTION TECHNIQUES ACCORDING TO STATISTICAL ANALYSIS DUE TO ITS DYNAMIC NATURE, IDENTIFICATION ABILITY AND CREATION OF SAFETY PROGRAM'S SIMI HIGH SIMILARITY RATE. THIS ENGINE COULD BE A SUITABLE TOOL FOR ASSESSING THE PERFORMANCE OF EXISTING SYSTEM THE POSSIBLE ATTACKS.

Keywords: Metamorphic Malware, Malware Signature, Statistical Analysis Code, Metamorphic Engine, Obfuscation, Cellular Learning Automata.

*Corresponding Author E-mail: parsa@iust.ac.ir

۱. مقدمه

از اصول اساسی پدافند غیرعامل حفاظت از مراکز حیاتی و مهم و مقاوم‌سازی آنها در مقابل حملات احتمالی است. مراکز حیاتی مراکزی هستند که انهدام کل یا قسمتی از آنها، موجب بروز بحران، آسیب و صدمات قابل توجه در نظام سیاسی، هدایت، کنترل و فرماندهی، تولیدی و اقتصادی، پشتیبانی، ارتباطی و مواصلاتی، اجتماعی، دفاعی با سطح تأثیرگذاری در سراسر کشور می‌شود. از اهداف کلان پدافند غیرعامل در حوزه IT تأمین امنیت و حصول اطمینان از عدم دسترسی‌های غیر مجاز به اسرار و اطلاعات کشور (ملی و بخشی) و ایمن‌سازی و حصول اطمینان از پایداری و خلل‌ناپذیری در فعالیت شبکه‌های الکترونیکی مدیریت و کنترل کشور (ملی و بخشی) می‌باشد. سیستم‌های کامپیوتری موجود در مراکز حیاتی از اهداف اصلی حملات سایبری خاموش[1] و فعال[2] به شمار می‌آیند. لازمه یک دفاع موفق در جنگ سایبری بالا بردن سطح امنیتی عناصر درگیر است. از اصول مطرح در دفاع سایبری، جلوگیری[3] است. جلوگیری عبارت از شناسایی راه‌های نفوذ و حمله و مقابله با آنها جهت افزایش ضریب امنیت، ایمنی و پایداری است [۱]. باید میزان مقاومت سیستم‌های کامپیوتری را نسبت به حملات نسل جدید، به ویژه حملاتی که به وسیله بدافزارهای دگردیس هوشمند صورت می‌گیرد، مورد ارزیابی قرار داد. یکی از روش‌های اصلی برای بررسی میزان کارایی روش‌های شناسایی بدافزارها آزمودن آنها در مقابل بدافزارهای جدید و هوشمند است. به همین دلیل در طی چند سال اخیر موتورهای تولید بدافزار با قابلیت تولید بدافزارهای دگردیس طراحی و ساخته شده‌اند. بدافزارهای دگردیس به عنوان یک خانواده از بدافزارهای هوشمند شناخته می‌شوند چرا که این دسته از بدافزارها قابلیت تغییر ساختار خود در نسل‌های مختلف را دارا هستند و از این طریق مانع از شناسایی توسط روش‌های شناسایی مبتنی بر امضا دودویی می‌شوند [۴-۲]. ساخت موتورهای تولید بدافزارهای دگردیس یک چالش پایان نیافتنی است چرا که دائماً با تعیین نقاط ضعف آنها می‌توان روش‌هایی برای شناسایی بدافزارهای تولید شده توسط آنها مطرح نمود. این امر ناشی از عدم پویایی موتورهای مطرح شده در این زمینه است. در این مقاله موتوری مطرح شده است که برای اولین بار از مفهوم پویایی در تولید نسل‌های جدید بدافزارها بهره گرفته است و مسیر تولید بدافزار در نسل‌های جدید نسبت به نسل‌های قبلی کاملاً متفاوت می‌باشد [۳ و ۸-۵]. با به کارگیری این موتور امکان بررسی میزان کارایی روش‌ها و نرم‌افزارهای شناسایی در مقابله با این خانواده از بدافزارها قابل ارزیابی است. در ادامه این مقاله، ابتدا مروری مختصر بر مفاهیم اولیه در زمینه بدافزارهای دگردیس و روش‌های شناسایی متداول بدافزارها صورت گرفته است. بعد از آن موتورهای مطرح در زمینه تولید بدافزارهای دگردیس معرفی و ساختار و نقاط ضعف هر یک تشریح شده است. در دو بخش پایانی هم ساختار موتور پیشنهادی مطرح و با جدیدترین روش‌های شناسایی کارایی آن به صورت منفرد و در مقایسه با سایر موتورهای مطرح در این زمینه مورد بررسی قرار گرفته است.

۱-۱. سیر تکاملی بدافزارها

ساده‌ترین راه برای تغییر نمایش یک بدافزار استفاده از رمزنگاری[4] است. یک بدافزار رمز شده شامل یک رمزگشای ساده و یک بدنه بدافزار رمز شده است. اگر کلید رمزنگاری متفاوت برای تولید هر نسل استفاده شود، بدنه بدافزار رمزنگاری شده متفاوت به نظر خواهد رسید. با رمزنگاری، رمزگشا از یک نسل به نسل دیگر ثابت باقی می‌ماند. در نتیجه، تشخیص می‌تواند بر اساس الگوی کد رمزگشا باشد [۴ و ۹].

بدافزارهای چندریخت براساس جهش[5] در رمزگشای خود عمل می‌کنند و تعداد نمونه‌هایی که برای رمزگشا در این مدل بدافزارها مطرح می‌شود تا چند میلیون هم می‌رسد. مشکل اصلی بدافزارهای چندریخت این است که در هنگام رمزگشایی بدافزار، قسمت کد بدخواه که ۹۰ درصد بدنه کد بدافزار را تشکیل می‌دهد، می‌تواند به عنوان منبع مهمی برای تطبیق امضا باشد [۴ و ۱۶-۹]. برای رفع ضعف‌هایی که بدافزارهای قبلی داشتند، بدافزارهای دگردیس ابداع شدند. این بدافزارها دارای رمزگشا یا بدنه بدافزار ثابت نیستند، اما توانایی تولید نمایش‌های متفاوت از ظاهر خود را دارند.

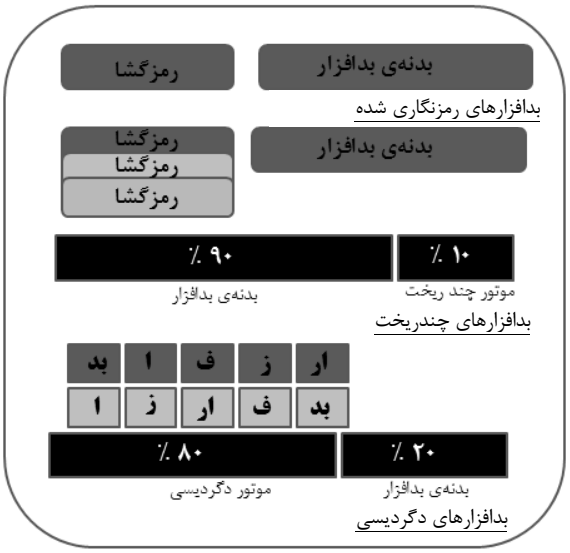
بدافزار دگردیس برای تغییر بدنه کد خود از روش‌های مبهم‌سازی کد استفاده می‌کنند. با به کارگیری روش‌های مبهم‌سازی، تحلیل ایستای کد بدافزار بسیار مشکل شده و در بسیاری از موارد غیر ممکن می‌شود [۴ و ۹]. با توجه به این مطلب که بدنه کد نسل جدید تولید شده از بدافزار دگردیس با بدنه نسل قبلی‌اش متفاوت است، بنابراین امضا بدافزارها از هر نسل به نسل بعدی آن متفاوت خواهد بود. بدنه یک بدافزار دگردیس از دو قسمت عمده موتور دگردیسی و کدبخواه تشکیل شده است. ۲۰ درصد کد بدافزار دگردیس مربوط به کد بدخواه و ۸۰ درصد کد بدافزار دگردیس مربوط به موتور دگردیسی است. ساختار انواع بدافزارهای مطرح شده به همراه سیر تکاملی آنها در شکل (۱) نشان داده شده است.

۱-۲. روش‌های تشخیص بدافزارها

الف: ساده‌ترین رویکرد برای تشخیص بدافزارها اسکن رشته است. اسکنرهای نسل اول[6] و اسکنرهای نسل دوم[7] به دنبال امضای بدافزار که شامل دنباله‌ای از بایت‌ها (رشته‌ها) استخراج شده از بدافزار، در فایل یا حافظه هستند. امضای بدافزارها در یک پایگاه داده قرار داده می‌شود. برای شناسایی بدافزار، اسکنرهای بدافزار فایل‌ها و یا سیستم

[1] Silent Killers
[2] Active
[3] Prevention
[4] Encryption
[5] Mutation
[6] First Generation Scanners
[7] Second Generation Scanners

تحت پوشش خود را به دنبال امضاهای شناخته شده در پایگاه داده بررسی می‌کنند. معمولاً این روش‌ها در مقابله با بدافزارهای دگردیس با مشکل مواجه می‌شوند [۴ و ۱۳- ۱۱].



شکل ۱. ساختار انواع بدافزارها به همراه سیر تکاملی آنها [۱۰]

ب: تحلیل اکتشافی[1] برای تشخیص بدافزارهای جدید یا ناشناخته به کار برده می‌شود. متدهای اکتشافی می‌توانند ایستا[2] یا پویا[3] باشند. اکتشاف ایستا بر اساس تحلیل فرمت فایل و ساختار کد بخش‌های بدافزار است. اکتشاف پویا از شبیه‌سازی کد، برای شبیه‌سازی پردازنده‌ها، سیستم عامل و تشخیص عملکرد مشکوک هنگامی که کد بدافزار در حال اجرا بر روی ماشین مجازی است، استفاده می‌کند. امکان رخداد خطای تشخیص (مثبت‌کاذب[4]) در تحلیل اکتشافی زیاد زیاد است [۱۴]. یک مثبت‌کاذب هنگامی رخ می‌دهد که یک تحلیگر اکتشافی یک برنامه را به اشتباه بدافزار تشخیص دهد. این مثبت‌کاذب‌ها از نظر هزینه به صرفه نیستند.

ج: زنجیره‌های مخفی مارکوف برای تشخیص بدافزارهای دگردیس مورد استفاده قرار می‌گیرند [۱۵]. مدل‌های زنجیره مخفی مارکوف (HMMs[5]) برای تحلیل الگوهای ایستا بسیار مناسب هستند. HMM یک ماشین حالت است که انتقال بین حالت‌ها، احتمال ثابتی دارد. می‌توانیم HMM را برای نمایش یک مجموعه داده آموزش دهیم که معمولاً به صورت دنباله مشاهدات هستند. حالت‌ها در آموزش HMM نمایانگر ویژگی‌هایی از داده‌های ورودی هستند و انتقال و احتمال مشاهده نمایانگر ویژگی‌های ایستا هستند. با دادن هر دنباله مشاهده، می‌توان آن را با یک HMM آموزش دیده هماهنگ کرد تا بتوان احتمال مشاهده شدن چنین دنباله‌ای را تعیین کرد. بدافزارهای دگردیس خانواده‌ای از بدافزارها را تشکیل می‌دهند. حتی اگر اعضای موجود در یک خانواده جهش پیدا کنند و ظاهر خود را تغییر دهند، هنوز میزان تشابهی برای نشان دادن وجود توابع یکسان وجود دارد. تشخیص بدافزارهای متنوع محدود به پیدا کردن این تشابه‌ها در قالب یک HMM آموزش دیده می‌شود [۱۴ و ۱۵].

د: روش آزمایش تشابه توسط میشرا معرفی شده است [۱۷]. این روش دو کد اسمبلی را مقایسه و یک مقدار امتیاز را تعیین می‌کند که بیانگر درصد تشابه بین دو برنامه است. اگر روش‌های مبهم‌سازی به صورت غیر مدیریت شده بر روی نسل‌های مختلف بدافزار پایه به کار گرفته شود می‌توان با این روش وجود تشابه بین نسل‌ها مختلف یک بدافزار را تعیین و از این طریق خانواده‌ای از بدافزارها را شناسایی نمود [۱۸]. اساس این روش به شرح زیر می‌باشد:

۱) دو برنامه اسمبلی به عنوان‌های X و Y برای تعیین میزان تشابه در نظر گرفته می‌شود، دنباله کد دستورالعمل‌ها را برای هر یک از دو برنامه استخراج کرده، که توضیحات کدها، خط‌های خالی، برچسب‌ها و سایر وابستگی‌ها شامل آن نمی‌شود. نتیجه دو دنباله از کد دستورالعمل‌ها است که متناظر این دو برنامه هستند.

۲) دو دنباله کد دستورالعمل‌ها را با در نظر گرفتن تمام زیر دنباله‌های به طول سه کد دستورالعمل پی‌درپی در هر دنباله مقایسه می‌شود. تعداد موردهای یکسان که در آن هر سه کد دستورالعمل با هر ترتیبی یکسان هستند برشمرده و بر تعداد کل کد دستورالعمل‌های هر یک از برنامه‌ها به صورت جداگانه تقسیم می‌شود. با میانگیری از دو عدد حاصل شده میزان تشابه دو برنامه به دست خواهد آمد. هر چه این عدد بیشتر باشد، تشابه دو برنامه بیشتر خواهد بود.

۱-۳. روش‌های مبهم‌سازی کد

الف: تزریق کدهای بی اثر: بدافزارهای دگردیس دستورات زائد را در بین هسته دستورات خود وارد می‌کنند. دستورات زائد دستوراتی هستند که یا اجرا نمی‌شوند و یا تأثیری بر روی خروجی برنامه ندارند [۱۵ و ۱۸]. چون امضای کد از روی دستورات ساخته می‌شود این روش باعث می‌شود که امضای دودویی بدافزار تغییر کند.

ب: جایگشت دستورات: استفاده از جایگشت برای جابه‌جایی زیر روتین‌های یک بدافزار یک روش مؤثر است. با n زیر روتین متفاوت یک بدافزار می‌تواند !n نسل مختلف تولید کند. به طور مثال با ۱۰ زیر روتین مختلف می‌تواند !۱۰ = ۳۶۲۸۸۰۰ نوع مختلف تولید کند. با این کار می‌توان امضای بدافزار را تغییر داد [۱۵ و ۱۸].

ج: ایجاد زیر توابع و فراخوانی آنها: در این روش، بلوکی از بدنه کد بدافزار به زیرروال تبدیل شده و به جای آن، دستور فراخوانی مربوطه‌اش قرار می‌گیرد. این روش نیز با ایجاد تغییر در بدنه کد بدافزار، امضای دودویی بدافزار را تغییر می‌دهد [۱۹].

د: تغییر کنترل جریان برنامه: به صورت متناوب، بدافزارهای

[1] Heuristic Analysis
[2] Static
[3] Dynamic
[4] False Positives
[5] Hidden Markov Models

دگردیس دستورات پرش را در کد خود قرار می‌دهند تا بتوانند به قسمت بعدی دستورات خود بروند. وارد کردن دستورات پرش باعث می‌شود کنترل جریان برنامه دائماً تغییر کند. در این روش، ابتدا به صورت تصادفی ترتیب قرارگیری دستورات برنامه را تغییر می‌دهند و بعد برای آنکه بتواند روند اجرای منطقی برنامه را ایجاد کند از دستورات پرش در نقاط مورد نظر برنامه استفاده می‌کنند. برای این منظور از دستور پرش غیر شرطی JMP استفاده می‌کنند [۱۹]. این کار با استفاده از قرار دادن تعداد نامحدود دستورات پرش غیر شرطی در بین بخش‌های مختلف کد بدافزار و جابجا کردن این بخش‌ها به صورت تصادفی صورت می‌گیرد.

۲. پیشینه تحقیق

۲-۱. موتور دگردیسی تووآرد[1]

این موتور به عنوان یک پایان‌نامه دانشجویی در دانشگاه سن جوزه از کالیفرنیای شمالی ارائه شده است [۵]. ساختار این موتور دگردیسی از ساختار بدافزار دگردیسی Evol الهام گرفته شده است. بدافزار Evol از روش‌های مبهم‌سازی افزودن کدهای بی‌اثر، تعویض نام ثبات‌ها و جایگزینی دستورات هم‌ارز استفاده می‌کند. موتور دگردیسی Towards علاوه بر روش‌های مبهم‌سازی استفاده شده در بدافزار Evol، از روش‌های دیگری نیز برای ایجاد جهش در کد بدافزار پایه استفاده می‌کند. در این موتور دگردیسی سعی شده است که بدافزارهای تولید شده با موتور به برنامه‌های عادی شباهت داشته باشند. جهت رسیدن به این هدف، از روش مبهم‌سازی افزودن کدهای بی‌اثر با تغییراتی جزئی استفاده شده است. این موتور دارای ضعف‌های زیر است:

۱- ناتوانی در مقابل روش‌های شناسایی مبتنی بر زنجیره‌های مخفی مارکوف ۲- محدود بودن روش‌های مبهم‌سازی به کار برده شده (تنها سه روش) ۳- محدود بودن به یک بدافزار خاص ۴- اشتباه بودن دستورات هم‌ارز به کار برده شده ۵- عدم پویایی الگوریتم مشابه‌سازی به برنامه‌های بی‌خطر مطرح شده در موتور ۶- افزایش بیش از حد حجم دودویی بدافزارها در نسل‌های جدید.

۲-۲. موتور دگردیسی جنریک[2]

این موتور دگردیسی از دو فاز اصلی برای فرآیند مبهم‌سازی استفاده کرده است [۶]. در فاز اول با استفاده از روش‌های مبهم‌سازی کد (بخش ۱-۳) سعی در ایجاد تغییر در ظاهر و رفتار کد می‌شود و در فاز دوم هدف خنثی کردن تغییراتی است که بر روی رفتار کد در فاز اول ایجاد شده است. این موتور دگردیسی با ارائه تغییراتی در روش مبهم‌سازی جایگشت کد، به وجود آمده است. برای مبهم‌سازی کد از روش جایگشت دستورات و برای مبهم‌سازی داده‌ها از رمزنگاری استفاده شده است. این موتور دارای ضعف‌های زیر است:

۱- دستورات پرش غیرشرطی و یا شرطی با نرخ بالا و بدون مدیریت به کد بدافزار تزریق می‌شوند که می‌تواند زمینه‌ای برای تشخیص توسط محصولات ضد بدافزاری باشد. ۲- از روش‌های مبهم‌سازی متنوع استفاده نشده است. ۳- در تولید بدافزارهای جهش‌یافته هیچ تلاشی درجهت مشابه‌سازی به برنامه‌های بی‌خطر و مقابله با روش تحلیل اکتشافی انجام نمی‌گیرد.

۲-۳. موتور دگردیسی هانتینگ[3]

در موتور پیاده‌سازی شده در این طرح، علاوه بر پیاده‌سازی چندین روش مبهم‌سازی، سعی شده است که بدنه کد بدافزارهای تولید شده به بدنه کد برنامه‌های عادی و قانونی از نظر آماری شباهت زیادی داشته باشند [۷]. از اصلی‌ترین اهداف این موتور تولید نسل‌های مختلفی از یک بدافزار پایه که امتیاز تشابه آنها نسبت به بدافزار پایه و نسبت به یکدیگر در حدود ٪۳۰ باشد (دلیل این امر این است که برنامه‌های عادی و قانونی دارای امتیاز تشابهی در حدود ٪۳۰ هستند). در این موتور از مجموعه‌ای از روش‌های مبهم‌سازی به صورت ترکیبی استفاده شده است. برای مشابه‌سازی به برنامه‌های بی‌خطر در فرآیند مبهم‌سازی از یک الگوریتم امتیازدهی پویا استفاده شده است. این موتور دارای ضعف‌های زیر است:

۱- موتور دگردیسی مذکور بر اساس یک نوع خاص بدافزار پایه طراحی شده است و هیچ تضمینی برای بهبود عملکرد آن در بدافزارهای دیگر وجود ندارد. ۲- کدهای بی‌اثر و زیرتوالی‌های بی‌اثری که از برنامه‌های عادی به کد بدافزارهای جهش یافته تزریق می‌شوند، بدون هیچ روش و مبهم‌سازی خاصی انجام شده و به راحتی قابل تشخیص و حذف هستند. ۳- الگوریتم مشابه‌سازی مطرح شده به صورت محلی عمل کرده و فاقد پویایی است.

۲-۴. موتور دگردیسی ای بی سی ام ایی[4]

موتور دگردیسی دیگری ارائه شده است [۸] که از روش‌های مبهم‌سازی بهبود یافته و الگوریتم سنجش تشابه وزن‌دار پیشنهادی، برای ایجاد جهش مدیریت شده در تولید بدافزارها و غلبه بر محصولات ضد بدافزاری، استفاده می‌کند. الگوریتم پیشنهادی به صورت وزن‌دار و پویا میزان تشابه کد بدافزار را به برنامه‌های بی‌خطر محاسبه کرده و تغییراتی را حفظ می‌کند که میزان تشابه را بین کد نسل جدید و کد برنامه‌های بی‌خطر افزایش داده و نسبت به بدافزار پایه کاهش دهد. این موتور دارای ضعف‌های زیر است:

۱- انتخاب کد بی‌خطر در فرآیند مشابه‌سازی به صورت ثابت صورت می‌گیرد. ۲- الگوریتم مشابه‌سازی مطرح شده فاقد پویایی لازم است. ۳- میزان مشابه‌سازی به برنامه‌های بی‌خطر کم بوده و توسط روش‌های شناسایی مبتنی بر زنجیره‌های مخفی مارکوف قابل شناسایی است.

[1] Towards
[2] Genric
[3] Hunting
[4] ABCME

۳. اتوماتای سلولی یادگیر

۳-۱. اتوماتای سلولی

اتوماتای سلولی یک مدل ریاضی است که می‌تواند برای محاسبات و شبیه‌سازی سیستم‌ها به کار رود. اتوماتای سلولی سیستم‌های گسسته‌ای هستند. هر سلول برای خود مجموعه‌ای از حالات دارد که در هر لحظه با توجه به حالت خودش و همسایه‌ها تصمیم می‌گیرد که به چه حالتی برود. قوانین تغییر حالت در اتوماتای سلولی در طول کار ثابت است و تغییر نمی‌کند. شبکه سلول‌ها می‌تواند ابعاد متفاوتی داشته باشند و یک، دو و یا بیشتر بعد داشته باشند.

۳-۲. اتوماتای یادگیر

اتوماتای یادگیر عاملی است که برای قرار گرفتن در محیطی احتمالی و غیر قطعی طراحی شده است. این ماشین تعدادی عمل متناهی می‌تواند انجام دهد. هر اتوماتای یادگیر دارای برداری از احتمال‌ها می‌باشد. این بردار نشان می‌دهد هر عمل با چه احتمالی انجام می‌شود. جمع درایه‌های این بردار برابر ۱ می‌باشد. هر عملی که توسط اتوماتا انجام می‌شود، توسط یک محیط احتمالی ارزیابی می‌شود و نتیجه ارزیابی در قالب سیگنالی مثبت یا منفی به اتوماتا داده می‌شود و اتوماتا از این پاسخ در انتخاب عمل بعدی تأثیر می‌گیرد. هدف اینست که اتوماتا بتواند بهترین عمل را از بین اعمال خود انتخاب کند. بهترین عمل، عملی است که احتمال دریافت پاداش از محیط را به حداکثر برساند [۲۰]. منطور از پاداش بالا رفتن احتمال انتخاب عمل در تکرار بعد است.

۳-۳. اتوماتای سلولی یادگیر

یک اتوماتای یادگیر تنها، کارایی زیادی ندارد. اگر تعداد زیادی اتوماتای یادگیر در همسایگی و همکاری با هم قرار بگیرند، می‌توانند مسائل دشوار را حل کنند. طراحی قانون‌هایی ثابت برای اتوماتای سلولی کاری دشوار می‌باشد و بدون شبیه‌سازی تصور رفتار اتوماتای سلولی بسیار دشوار می‌باشد. ترکیب اتوماتای سلولی با اتوماتای یادگیر می‌تواند تا حدی این مشکل را برطرف کند. با توجه به این مسئله و ضعف‌های عنوان شده برای اتوماتای سلولی، در یک تحقیق با ترکیب این دو مدل، مدل جدیدی با نام اتوماتای یادگیر سلولی ایجاد شد [۲۱]. در زیر تعریف فرمال اتوماتای یادگیر سلولی ارائه شده است:

تعریف ۱ (اتوماتای یادگیر سلولی): اتوماتای یادگیر سلولی d بعدی یک چندتایی $CLA=(Z^d, \varphi, A, N, F)$ است به طوری که:

- Z^D یک شبکه از d تایی‌های مرتب از اعداد صحیح می‌باشد. این شبکه می‌تواند یک شبکه متناهی، نیمه متناهی یا نامتناهی باشد.
- φ یک مجموعه متناهی از حالت‌ها می‌باشد.
- A ، یک مجموعه از اتوماتاهای یادگیر (LA) است که هر یک از آنها به یک سلول از اتوماتای سلولی نسبت داده می‌شود.
- $N=\{x_1, \ldots, x_m\}$ یک زیر مجموعه متناهی از Z^d می‌باشد که بردار همسایگی نامیده می‌شود.
- $F : \varphi^{\_} \rightarrow \beta^{\_}$ قانون محلی CLA می‌باشد به طوری که $\beta^{\_}$ مجموعه مقادیری است که می‌تواند به عنوان سیگنال تقویتی پذیرفته شود.

۴. موتور پیشنهادی

هدف به کارگیری اتوماتای سلولی یادگیر به عنوان یک عامل نظارتی در دگردیسی و مبهم‌سازی بدافزارها در جهت مشابه‌سازی آنها به برنامه‌های بی‌خطر است. ورودی اتوماتای سلولی یادگیر یک کد بدخواه و خروجی آن نیز کدی با بیشترین تشابه ممکن به برنامه‌های بی‌خطر با توجه به هزینه‌های زمانی و فضایی در نظر گرفته شده است.

در این طرح از اتوماتای سلولی یادگیر با متغییر سراسری استفاده شده است. در اتوماتای یادگیر سلولی با متغیر سراسری، هر سلول برای تصمیم‌گیری در مورد پاداش دادن و یا جریمه کردن عمل خود، علاوه بر در نظر گرفتن وضعیت اتوماتاهای اطراف خود، از پاسخ محیط سراسری حاکم بر کل شبکه نیز استفاده می‌کند. برای اتوماتای یادگیر داخل هر سلول نیز از اتوماتای یادگیر با ساختار متغییر استفاده شده است. اتوماتای یادگیر با ساختار متغیر را می‌توان توسط چهارتایی (α,β,p,T) نشان داد که:

$\alpha=\{\alpha_1, \ldots, \alpha_n\}$ مجموعه عمل‌های اتوماتا، $\beta=\{\beta_1, \ldots, \beta_n\}$ مجموعه ورودی‌های اتوماتا، $p=\{p_1, \ldots, p_n\}$ بردار احتمال انتخاب هر یک از عمل‌ها و $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگوریتم یادگیری می‌باشد. الگوریتم زیر یک الگوریتم یادگیری خطی است. فرض کنید عمل i در مرحله nام انتخاب شود.

- پاسخ مطلوب

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) &= (1-a)p_j(n) \qquad \forall_j \ j \neq i \end{aligned} \tag{۱}$$

- پاسخ نامطلوب

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= \left(\frac{b}{r} - 1\right) + (1-b)p_j(n) \qquad \forall_j \ j \neq i \end{aligned} \tag{۲}$$

در روابط (۱) و (۲)، a پارامتر پاداش و b پارامتر جریمه می‌باشند. با توجه به مقادیر a و b، زمانی که b از a خیلی کوچک‌تر باشد، الگوریتم را $L_{R\varepsilon P}$ می‌نامیم.

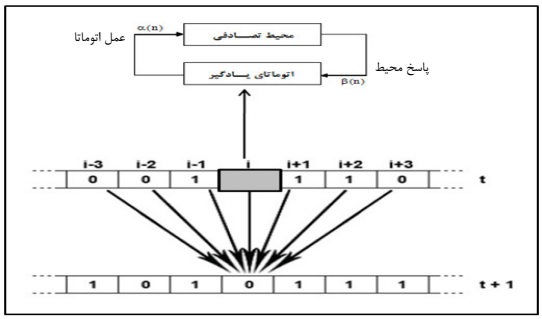
حال باید اعضای چهارتایی اتوماتای یادگیر با ساختار متغییر را تعیین نمایم. مجموعه عمل‌های اتوماتا (α) را برابر با مجموعه‌ای از روش‌های متنوع مبهم‌سازی در نظر گرفته‌ایم، که تعداد روش‌های

انتخاب شده ۵ تا می‌باشد که به ترتیب مبهم‌سازی از طریق تزریق کدهای بی‌اثر به‌هم متصل، مبهم‌سازی از طریق تزریق کدهای بی‌اثر غیرمتصل، مبهم‌سازی از طریق ایجاد جایگشت در بلوک‌های کد، مبهم‌سازی از طریق ایجاد زیرروال از کد و فراخوانی آن و مبهم‌سازی از طریق تغییر در کنترل جریان اجرای برنامه می‌باشند. مجموعه ورودی‌های اتوماتا (β) را برابر با بلوک انتخاب شده متناظر با بخشی از کد بدافزار پایه در نظر گرفته‌ایم. بردار انتخاب هر یک از عمل‌ها (P) را برابر با P={۰/۱۸, ۰/۱۸, ۰/۲۴, ۰/۲۲, ۰/۱۸} در نظر گرفته‌ایم. الگوریتم یادگیری را نیز مشابه حالت تعریف شده در بالا در نظر گرفته‌ایم.

برای آنکه مانع از انجام عملیات نرمال‌سازی بر روی بدافزار تولید شده در نسل جدید توسط روش‌های نرمال‌سازی شد، در ابتدا باید یک تابع پیچیده به کد بدافزار اولیه به صورت تصادفی ترزیق شود. که هم به برنامه‌های نرمال مشابه باشد و هم به دلیل پیچیدگی مانع از نرمال‌سازی کد بدافزار تولید شده در نسل جدید شود. این تابع باید ابتدا بر روی کد تأثیر بگذارد و سپس تأثیر اعمال شده را خنثی کند. به این منظور می‌توان از توابع رمزنگاری متنوعی استفاده نمود. یکی از گزینه‌های مطلوب، که در موتور پیشنهادی نیز به کار برده شده است، الگوریتم رمزنگاری مبتنی بر اتوماتای سلولی با قانون ۳۰ می‌باشد. قانون ۳۰ با نمایش دودویی به صورت رشته بیتی (۰۰۱۱۱۱) خواهد آمد. هر رشته وردی اتوماتا که تحت تأثیر این تابع از نمایش اولیه خود خارج شود و بعد از اجرای چند تکرار به نمایش ثانویه برسد، از نظر تشابه بر اساس تعداد تکرار انجام شده با نمایش اولیه خود بسیار متفاوت خواهد بود. برای بازیابی نمایش اولیه آن باید ابتدا حالت اولیه رمزنگار را تشخیص داد و سپس از XOR کردن نمایش ثانویه و حالت اولیه رمزنگار نمایش اولیه به دست خواهد آمد.

ورودی‌های اتوماتای سلولی یادگیر را باید تعیین نمود. برای این منظور کد ورودی بدافزار پایه که تابع پیچیده را به آن تزریق کرده‌ایم، را در قالب ۱۰ بلاک ذخیره‌سازی می‌کنیم. این ۱۰ بلاک تنها شامل بخش دستورات کد بدافزار پایه است و قسمت‌های توضیحات و فضاهای خالی را حذف می‌کنیم. هر بلاک شامل بخشی از کد بدافزار مورد بررسی است.

متناظر با تقسیم کد بدافزار به ۱۰ بلاک باید یک الگوی نرمال و بی‌خطر نیز در نظر بگیریم. به همین منظور یک کد نرمال و بی‌خطر را به منظور مشابه‌سازی به صورت تصادفی انتخاب می‌کنیم. انتخاب تصادفی این کد بی‌خطر باعث می‌شود که به صورت ضمنی تشابه بین نسل‌های مختلف تولید شده از کد بدافزار پایه یکسان کاهش پیدا کند و همچنین باعث پویایی بیشتر ساختار موتور پیشنهادی می‌شود. این کد بی‌خطر را نیز به ۱۰ بلاک تقسیم می‌کنیم. ساختار اتوماتای طراحی شده در شکل (۲) نشان داده شده است. این موتور با عنوان CLAEn[1] در این مقاله نام‌گذاری شده است.



شکل ۲. ساختار اتوماتای طراحی شده برای موتور CLAEn

۵. فرآیند یادگیری

بلاک اتوماتای سلولی از ۱۰ بلاک کد بدافزار پایه و ۱۰ بلاک کد بی‌خطر متناظر با آنها تشکیل شده است.. در هر مرحله به صورت تصادفی تعدادی از بلاک‌ها به منظور تعامل با محیط و آموزش اتوماتای یادگیر انتخاب می‌شوند. هر بلاکی که انتخاب می‌شود برای تعامل با محیط می‌تواند از ۵ تابعی که برای آن در نظر گرفته شده است، یکی را انتخاب کند. انتخاب توابع براساس احتمال‌های نسبت داده شده به آنها صورت می‌گیرد و تابعی انتخاب می‌شود که احتمال آن نسبت به سایر توابع بیشتر باشد. با انتخاب تابع و اعمال آن به بلاک کد مورد نظر به یک بلاک کد جدید می‌رسیم. با استفاده از الگوریتم سنجش تشابه پایه میزان تشابه بلاک انتخاب شده را در حالت قبلی و حالت جدید می‌سنجیم. هدف از این کار تعیین پاداش و جریمه با توجه به بازخورد محیط است. در این حالت دو وضعیت پیش خواهد آمد:

❖ نخست- میزان تشابه بلاک به کدهای بی‌خطر متناظر در وضعیت جدید افزایش پیدا کرده باشد. در این حالت دو کار باید انجام دهیم: الف: بلاک جدید را جایگزین بلاک قبلی در اتوماتای سلولی می‌کنیم. ب: وزن‌های اعمال شده به توابع را بروزرسانی می‌کنیم. یعنی تابع انتخاب شده که باعث بهبود تشابه شده است با افزایش احتمال انتخاب در مرحله‌های بعد همراه خواهد بود و متناسب با این افزایش برای حفظ مجموع ۱ احتمالات، باید احتمال انتخاب سایر توابع را کاهش دهیم.

❖ دوم- میزان تشابه بلاک به کدهای بی‌خطر متناظر در وضعیت جدید کاهش پیدا کرده باشد. در این حالت دو کار باید انجام دهیم: الف: بلاک قبلی را بدون تغییر در اتوماتای سلولی حفظ خواهیم کرد. ب: وزن‌های اعمال شده به توابع را به‌روز رسانی می‌کنیم. یعنی تابع انتخاب شده که باعث کاهش میزان تشابه شده است، با کاهش انتخاب احتمال در مرحله‌های بعد همراه خواهد بود و سایر توابع

[1] Cellular Learning Automata Engine

به منظور حفظ مجموع ۱ برای احتمالات، متناسب با این کاهش با افزایش احتمال انتخاب در مرحله بعد همراه خواهند شد.

عملیات به کارگیری توابع و سنجش میزان تشابه و به‌روز رسانی احتمال‌های انتخاب را برای تمام بلاک‌های انتخاب شده انجام می‌دهیم. در انتهای فرآیند آموزش مجدد آزمایش سنجش میزان تشابه کلی کد نسل جدید با کدهای بی‌خطر را انجام می‌دهیم. در این حالت نیز دو حالت ممکن است رخ دهد:

❖ الف: میزان تشابه کد مخرب جدید نسبت به کد بی‌خطر انتخاب شده افزایش پیدا کرده باشد. در این حالت تمام وزن‌ها به وزن‌های جدید به‌روز رسانی می‌شوند و بلاک‌های اتوماتای سلولی نیز با بلاک‌های جدید جایگزین می‌شوند و فرآیند آموزش مجدد و این بار با احتمال‌های انتخاب به‌روز رسانی شده و نیز بلاک‌های تغییر یافته، تکرار خواهد شد.

❖ ب: میزان تشابه کد مخرب جدید نسبت به کد بی‌خطر انتخاب شده کاهش پیدا کرده است. در این حالت تمام وزن‌ها به وزن‌های جدید به‌روز رسانی می‌شوند اما بلاک‌های جدید با بلاک‌های قبلی تعویض خواهند شد و فرآیند آموزش مجدد و این بار با احتمال‌های انتخاب به‌روز رسانی شده ولی با همان بلاک‌های قبلی، تکرار خواهد شد.

اما این فرآیند آموزش را تا چه زمانی ادامه دهیم؟ این سؤال را می‌توان با سه حالت مشخص نمود:

❖ حالت اول زمانی رخ می‌دهد که برای آموزش اتوماتای سلولی منابع محدودی در اختیار داریم. در این حالت بعد از طی تعداد مشخصی از فرآیند آموزش و اتمام منابع، آموزش را متوقف و آخرین وضعیت بلاک‌ها را به صورت خروجی در نظر می‌گیریم و آخرین احتمالات را برای توابع در نظر می‌گیریم.

❖ حالت دوم زمانی رخ می‌دهد که اساساً کد بدافزار پایه قابلیت و استعداد مشابه شدن به برنامه‌های نرمال و بی‌خطر را ندارند به همین منظور برای تعداد مراحل آموزش یک حد بیشینه تعریف می‌کنیم تا در صورتی که تعداد مراحل آموزش به این حد رسید فرآیند آموزش را متوقف و آخرین وضعیت بلاک‌ها را به عنوان خروجی در نظر بگیرد.

❖ حالت سوم و در واقع حالت ایده‌آل مربوط به شرط اتمام آموزش می‌باشد. در اکثر مقالات و منابع مطرح شده میزان تشابه کدهای نرمال و بی‌خطر در حالت کمینه نزدیک به ۱۴ درصد، در حالت بیشینه نزدیک به ۹۳ درصد و در حالت متوسط ۳۵ درصد ذکر شده است [۸ و ۵]. هدف ما مشابه‌سازی کدهای مخرب به کدهای بی‌خطر و نرمال است. پس باید کدهای تولید شده از نظر آماری بسیار نزدیک به سایر کدهای نرمال باشند. به همین منظور شرط اتمام مراحل آموزش را بر روی میانگین میزان تشابه کدهای نرمال و بی‌خطر به یکدیگر یعنی ۳۵ درصد قرار می‌دهیم. به عبارت دیگر در صورتی که میزان تشابه کد مخرب آموزش دیده به کدهای نرمال و بی‌خطر به میزان ۳۵ درصد افزایش رسیده باشد مراحل آموزش را پایان می‌دهیم.

الگوریتم موتور CLAEn به صورت شبه کد در شکل (۳) آورده شده است.

Algorithm CLAEn

Input : Base Malware
Output : New Generation of Malware

Begin

Choice a Random Benign Code and divided in 10 Block
Divided Base Malware in 10 Block

Initialize CLA with 10 LA each with 5 actions

While (Number of Iteration is down of 100) **Do**

For each Line in CLA Do in Parallel
Each LA choice one of its action according to its action probability vector
IF1 (New Similarity in Block > Old Similarity in Block)
Old Block = New Block
Old Similarity = New Similarity
Update Action Vector
Else (IF1)
Update Action Vector

IF2 (Full New Similarity > Full Old Similarity)
Update all Block With New Block in CLA
Run new Iteration with New Block and New Action Vector
Else (IF2)
Replace all New Block with Old Block
Run new Iteration with Old Block and New Action Vector

End For
End While
End

شکل ۳. شبه کد الگوریتم موتور CLAEn

۶. نتایج و بحث

یک موتور دگردیسی کارا باید دارای خصوصیات زیر باشد (این خصوصیات در موتورهای مشابه به عنوان عامل ارزیابی کارایی موتور در مقایسه با سایر موتورها مورد استفاده قرار گرفته‌اند) [۸-۵]:

- بدافزارهای جهش یافته را با میزان تشابه کمی نسبت به بدافزار پایه و نسبت به یکدیگر ایجاد کند [۱۰].

- بدافزارهای جهش یافته، از نظر امضاء و خصوصیات آماری کد به برنامه‌های بی‌خطر و قانونی شباهت داشته باشند [۱۵].

- باید از کد اسمبلی هر نوع بدافزاری پشتیبانی کند [۸].

- نباید حجم دودویی بدافزارها را در بدافزارهای جهش یافته به میزان زیادی افزایش دهد [۵].

– روش‌های مبهم‌سازی اعمال شده به کد بدافزارهای جهش یافته، نباید به راحتی قابل تشخیص باشند [۳].

– در عملکرد بدافزارهای جهش یافته از بدافزارهای پایه، تغییری ایجاد نکند [۱۸].

– نسبت به روش‌های شناسایی مبتنی بر نرمال‌سازی مقاوم باشد و یا در ساختارش به گونه‌ای عمل کند که مانع از نرمال شدن کد تولید شده شود [۱۱].

برای انجام آزمایشات، مجموعه کد اسمبلی تعدادی بدافزار مشهور از [۲۲ و ۲۳] دریافت شده و به عنوان بدافزارهای پایه برای ارزیابی موتور دگردیسی استفاده شده‌اند. مجموعه برنامه‌های بی‌خطر نیز، از دودویی فایل‌های اجرایی Cygwin [۱۱]، که عملیات سطح پایینی مثل بدافزارها را انجام می‌دهند، گردآوری شده است.

برای آزمایش قدرت بدافزارهای جهش‌یافته با موتور دگردیسی CLAEn در غلبه بر محصولات ضد بدافزاری، کد اسمبلی ۳۲ بدافزار جهش‌یافته از ۴ بدافزار پایه را در قالب فایل‌های اجرایی ۳۲ بیتی در محیط ماشین مجازی کامپایل کرده و با ۱۳ محصول ضد بدافزاری مشهور که در رده‌بندی محصولات ضد بدافزاری ماه مارس ۲۰۱۲ در رده‌های +Advance و Advance قرار گرفته‌اند، استفاده کرده‌ایم [۲۴]. نتایج این بررسی در جدول (۱) نشان داده شده است.

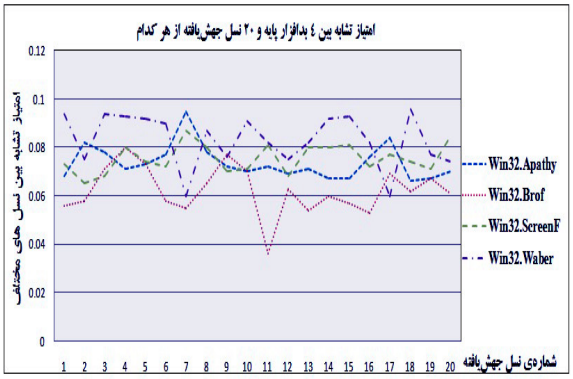
جدول ۱. عملکرد بدافزارهای جهش یافته در مقابل محصولات ضد بدافزاری مشهور(O=Original, M=Muted)

Motor / Antivirus		Screen Fake		Waber		Brof		Apathy	
		O	*M*	*O*	*M*	*O*	*M*	*O*	*M*
1	G Data	✓	✓	✓	×	✓	×	✓	×
2	Avira	✓	×	✓	✓	✓	×	✓	×
3	Kasper sky	✓	×	✓	×	✓	×	✓	×
4	Sophos	✓	×	✓	×	✓	×	✓	✓
5	F-Secure	✓	✓	✓	×	✓	×	✓	×
6	Panda	✓	×	✓	×	✓	✓	✓	×
7	Bit defender	✓	×	✓	×	✓	×	✓	×
8	Bull Guard	✓	×	✓	×	✓	×	✓	×
9	eScan	✓	×	✓	×	✓	×	✓	×
10	Avast	✓	×	✓	×	✓	×	✓	✓
11	ESET	✓	×	✓	✓	✓	×	✓	×
12	McAfee	✓	×	✓	×	✓	×	✓	×
13	Fortinet	✓	×	✓	×	✓	✓	✓	×

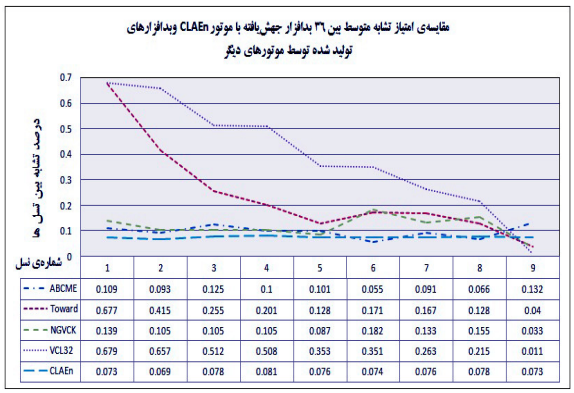
برای انجام آزمایش تشابه، ابتدا بدافزارهای پایه به عنوان ورودی به موتور دگردیسی ارسال شده و بدافزارهای جهش‌یافته از بدافزارهای پایه در خروجی تولید شده‌اند. سپس بدافزارهای جهش‌یافته کامپایل شده و دودویی اجرایی آنها دیس‌اسمبل شده و از کد دیس‌اسمبل شده برای انجام آزمایشات آزمایش تشابه استفاده شده است.

ابتدا میزان تشابه بدافزار پایه با نسل‌های تولید شده جدید توسط موتور دگردیسی در شکل (۴) مورد بررسی قرار گرفته است. با توجه به نمودار میانگین تشابه بین بدافزار پایه و نسل‌های تولید شده به طور متوسط ۷ درصد بوده است. یعنی موتور CLAEn توانایی تولید بدافزارهای جدید با میزان تشابه کم به بدافزار پایه را دارد، که نسبت به موتورهای مشابه نتایج بهتری داشته و در عین حال بیشینه تشابه در موتور پیشنهادی کمتر از ۱۰ درصد می‌باشد که در موتورهای دیگر درصد بالاتری (نزدیک به ۲۰ درصد [۸]) گزارش شده است.

یک موتور کارا باید توانایی تولید بدافزارهایی را داشته باشد که میزان تشابه کمی با یکدیگر و با بدافزار پایه داشته باشند. در شکل (۴) میزان تشابه بدافزارهای تولید شده در نسل جدید با بدافزار پایه مقایسه شده است. در شکل (۵) میزان تشابه بین نسل‌های مختلف در موتور CLAEn و موتورهای دگردیسی ABCME, Toward, Hunting و VCL32 مورد مقایسه قرار گرفته‌اند. همان‌گونه که از شکل (۵) مشاهده می‌شود میانگین میزان تشابه بین نسل‌های مختلف تولید شده توسط موتور CLAEn از سایر موتورهای دیگر کمتر می‌باشد.



شکل ۴. درصد تشابه بین ۴ بدافزار پایه و ۲۰ نسل جهش‌یافته از هر کدام از بدافزارها

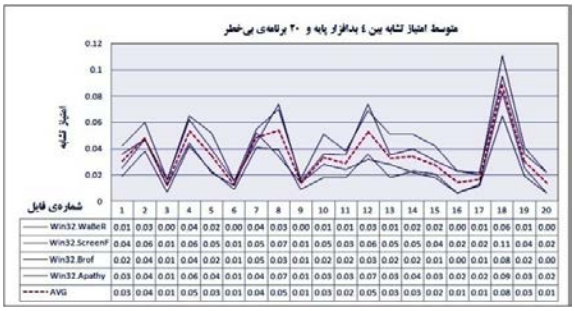


شکل ۵. مقایسه امتیاز تشابه متوسط بین ۳۶ بدافزار جهش‌یافته با موتور CLAEn و بدافزارهای تولید شده توسط موتورهای دگردیسی ABCME, Toward, Hunting و VCL32

روش‌های آماری پیاده‌سازی شده در قسمت تحلیل اکتشافی در محصولات ضد بدافزاری قادر هستند با استفاده از تشابه خصوصیات آماری در کد اسمبلی، بدافزارهای هم‌خانواده را شناسایی کرده و با استفاده از این نتایج آماری بین برنامه‌های بی‌خطر و انواع بدافزارها تمایز قائل شوند.

برای غلبه بر تحلیل اکتشافی سعی شده است که بدنه کد بدافزارهای تولید شده توسط موتور دگردیسی به بدنه کد برنامه‌ها و نرم‌افزارهای عادی و بی‌خطر شباهت داشته باشند. در نتیجه، روش‌های آزمایش تشابه و روش‌های آماری پیاده‌سازی شده در تحلیل اکتشافی در تعیین مقدار آستانه تمایز و تشخیص دچار مشکل شده و توانایی متمایز ساختن بدافزارها از برنامه‌های بی‌خطر را نداشته باشند.

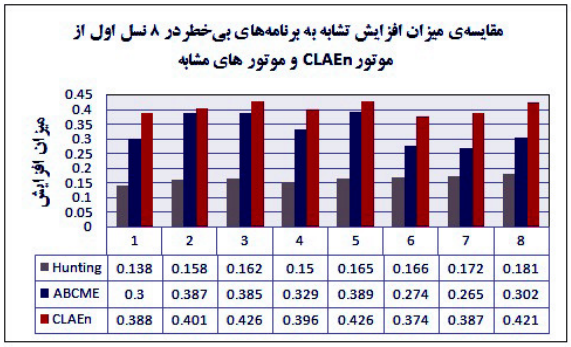
برای سنجش تشابه بین بدافزارهای پایه و برنامه‌های بی‌خطر و همچنین بین بدافزارهای جهش‌یافته و برنامه‌های بی‌خطر، تعداد ۲۰ فایل برنامه بی‌خطر از مجموعه فایل‌های اجرایی و کتابخانه‌ای Cygwin به صورت تصادفی انتخاب شده و در آزمایشات به کار رفته است. نتایج آزمایش تشابه بین بدافزارهای پایه و برنامه‌های بی‌خطر در شکل (۶) ارائه شده است.



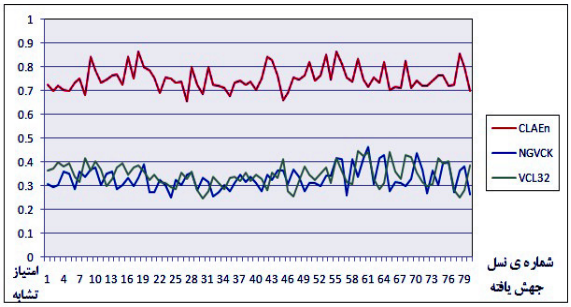
شکل ۶. متوسط امتیاز تشابه بین ۴ بدافزار پایه و ۲۰ برنامه‌ی بی‌خطر موجود در Cygwin

همان‌طوری که مشاهده می‌شود، موتور دگردیسی CLAEn با متوسط درصد تشابه ۷۵٪ به برنامه‌های بی‌خطر، توانسته است میزان تشابه بدافزارهای جهش یافته را به برنامه‌های بی‌خطر در مقایسه با بدافزارهای پایه، به طور متوسط ۴۳٪ افزایش داده و این میزان در مقایسه با [۷ و ۲۵] به طور متوسط ۲۶٪ افزایش داشته است. این اعداد برای موتور دگردیسی ABCME به ترتیب برابر ۳۳٪ و ۱۶٪ بوده است [۸]. نتایج در شکل (۷) قابل مشاهده هستند.

برای ارزیابی کارایی موتور دگردیسی CLAEn در مشابه‌سازی بدافزارهای جهش‌یافته به برنامه‌های بی‌خطر، میزان تشابه بدافزارهای تولید شده توسط موتورهای دگردیسی مشهور NGVCK و VCL32 را به برنامه‌های بی‌خطر مقایسه کرده‌ایم [۲۳]. نتایج محاسبات در شکل (۸) نشان داده شده است. متوسط تشابه به برنامه‌های بی‌خطر در بدافزارهای تولید شده توسط NGVCK برابر با ۳۲٪ و در بدافزارهای تولید شده توسط VCL32 برابر با ۳۴٪ می‌باشد که این مقادیر در مقایسه با میزان تشابه بدافزارهای تولید شده توسط CLAEn، به طور متوسط ۴۲٪ پایین‌تر است. همچنین مقادیر فوق نشان می‌دهد که بدافزارهای جهش یافته توسط CLAEn توانایی غلبه بر دیگر روش تشخیص [۲۶] را نیز دارند.



شکل ۷. مقایسه میزان افزایش تشابه به برنامه‌های بی‌خطردر ۸ نسل اول از موتور CLAEn و موتورهای Hunting ,ABCME

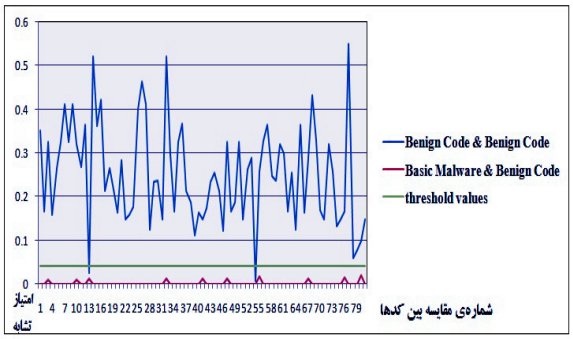


شکل ۸. مقایسه میانگین امتیاز تشابه بین بدافزارهای جهش یافته و برنامه‌های بی‌خطر در CLAEn و موتورهای NGVCK,VCL32

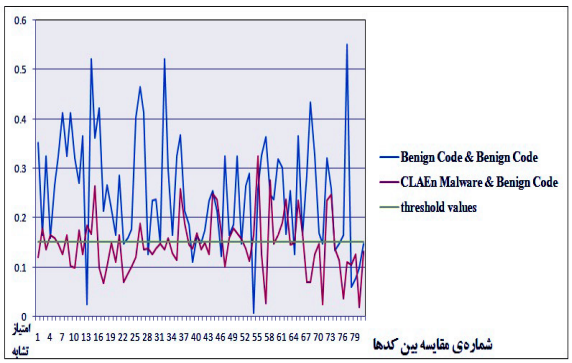
در ایده ارائه شده در مرجع [۷] که مشابه تحلیل اکتشافی محصولات ضد بدافزاری عمل می‌کند، نشان داده شده است که با استفاده از روش آزمایش تشابه با مقدار W=۱۵، می‌توان بدافزارهای دگردیس مشابه برنامه‌های بی‌خطر را با تعیین یک امتیاز تشابه آستانه تشخیص، از برنامه‌های بی‌خطر متمایز نموده و تمامی گونه‌های این بدافزارها را با مقدار منفی‌کاذب و مثبت‌کاذب پایین تشخیص داد. این ایده زمانی یک روش کارا است که بدافزارهای جهش یافته از بدافزارهای پایه، تنها با افزودن کدهای بی‌اثر از برنامه‌های بی‌خطر مبهم‌سازی شده باشند. اما در موتور دگردیسی CLAEn تمامی روش‌های مبهم‌سازی تحت نظارت اتوماتای سلولی یادگیر پیشنهادی انجام شده و عملیات مشابه‌سازی مختص به یک روش خاص نبوده است.

برای آزمایش موتور دگردیسی پیشنهادی در برابر روش ارائه شده در مرجع [۷]، باید میزان تشابه بین هر کدام از برنامه‌های بی‌خطر را نسبت به همدیگر محاسبه کرده و همچنین میانگین میزان تشابه هر کدام از بدافزارهای پایه و نیز جهش یافته را نسبت به همان برنامه‌های بی‌خطر محاسبه کنیم. نتایج حاصل در دو نمودار ارائه

شده است؛ همان‌طوری که در شکل (۹) مشاهده می‌شود، تعیین مقدار آستانه تشخیص به راحتی با مقدار ۰/۰۴ انجام شده و اکثر بدافزارهای پایه با میزان ۱ خطای منفی‌کاذب و ۲ خطای مثبت‌کاذب تشخیص داده شده‌اند. اما با توجه به نتایج شکل (۱۰) مشخص می‌شود که تعیین مقدار آستانه تشخیص، ممکن نبوده و تعیین یک مقدار نادقیق موجب افزایش شدید خطاها یا مثبت‌های کاذب و منفی‌های کاذب گردیده است. نتایج مورد بحث، موفقیت موتور دگردیسی پیشنهادی را در غلبه بر روش تحلیل اکتشافی و روش تشخیص ارائه شده در مرجع [۷] نشان می‌دهد.

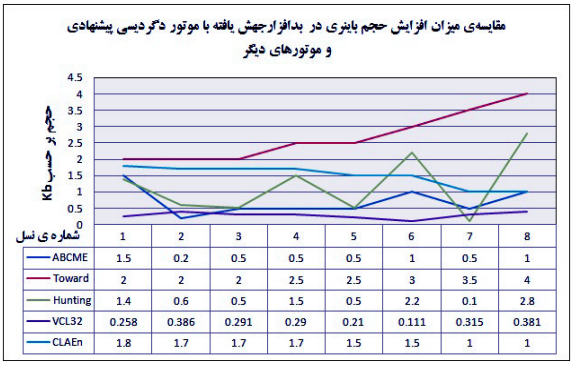


شکل ۹. مقایسه میانگین امتیاز تشابه بین هر کدام از برنامه‌های بی‌خطر و بین هر کدام از بدافزارهای پایه و برنامه‌های بی‌خطر



شکل ۱۰. مقایسه میانگین امتیاز تشابه بین هرکدام از برنامه‌های بی‌خطر و بین هرکدام از بدافزارهای جهش یافته با CLAEn و برنامه‌های بی‌خطر

در یک موتور دگردیسی کارا باید حجم دودویی بدافزارهای جهش یافته نسبت به بدافزار پایه، افزایش معنی‌داری نداشته و روال افزایش حجم دودویی بدافزارهای جهش یافته حالت نمایی پیدا نکند. موتور دگردیسی CLAEn با حداکثر افزایش حجم ۱۵٪ در دودویی بدافزارهای جهش یافته نسبت به بدافزارهای پایه، یک موتور دگردیسی کارا بوده و نسبت به موتورهای دگردیسی ارائه شده در مراجع [۵ و ۷]، افزایش متناسبی در حجم دودویی بدافزارهای جهش یافته دارد. در شکل (۱۱) مقایسه‌ای بین موتور دگردیسی CLAEn و موتورهای دگردیسی Towards ,Hunting ,ABCME و VCL32 ارائه شده است.



شماره ی نسل	1	2	3	4	5	6	7	8
ABCME	1.5	0.2	0.5	0.5	0.5	1	0.5	1
Toward	2	2	2	2.5	2.5	3	3.5	4
Hunting	1.4	0.6	0.5	1.5	0.5	2.2	0.1	2.8
VCL32	0.258	0.386	0.291	0.29	0.21	0.111	0.315	0.381
CLAEn	1.8	1.7	1.7	1.7	1.5	1.5	1	1

شکل ۱۱. مقایسه میزان افزایش حجم دودویی در بدافزار جهش یافته با موتور دگردیسی CLAEn و موتورهای دگردیسی ABCME ,Towards ,Hunting و VCL32

۷. نتیجه‌گیری

در این مقاله به منظور بررسی کارایی روش‌های شناسایی بدافزارهای دگردیسی، موتور CLAEn مطرح شده است. این موتور به صورت پویا و با استفاده از اتوماتای سلولی یادگیر نسل‌های جدیدی از بدافزار پایه تولید می‌کند. بدافزارهای تولید شده از نظر امضای دودویی و آماری با کد بدافزار پایه متفاوت بوده و از لحاظ تشابه به برنامه‌های بی‌خطر با درصد بالایی متشابه (به طور متوسط ۷۵ درصد) است. ماهیت یادگیرنده و پویای اتوماتای سلولی یادگیر این امکان را فراهم می‌کند که برای تولید بدافزارهای نسل جدید از یک بدافزار پایه در هر بار اجرای موتور، مسیر ساخت و الگوریتم یادگیری از ابتدا و به صورت کاملاً تصادفی بر روی کد بدافزار پایه عمل کند. نتایج حاصل از انجام آزمایش‌های تشابه در حالت‌های مختلف بیانگر توانای موتور پیشنهادی در تولید بدافزارهایی با درصد تشابه بالا به برنامه‌های بی‌خطر، با کمترین تشابه به یکدیگر(به طور متوسط ۷ درصد) و مقاوم در برابر محصولات ضد بدافزاری می‌باشد. از آنجا که قابلیت‌های بالای این موتور در این مقاله اثبات و مورد ارزیابی قرار گرفته‌اند، می‌توان از این موتور به عنوان ابزار مناسبی برای بررسی کارایی سیستم‌های امنیتی موجود در مقابل حملات احتمالی استفاده نمود، چرا که نزدیک به ۹۰ درصد نرم افزارهای آنتی ویروس فعلی امکان تشخیص بدافزارهای تولید شده توسط این موتور را ندارند. این موتور نقش قابل توجهی در پدافند غیرعامل سیستم‌های کامپیوتری ایفا می‌کند.

۸. منابع

[1] Mohamadyan, D. H.; Laylai, M. “Passive Defence in Information Technology”; ICT News and Information, http://favanews.com/gallery/padafand.doc (In Persian).

[2] Mathur, K.; Hiranwal, S. “A Survey on Techniques in Detection and Analyzing Malware Executables”; Int. J. Adv. Res. Comput. Sci. Soft. Eng. (IJARCSSE), 2013, 3, 422-428.

[3] Baysa, D. “Structural Entropy and Metamorphic Malware”; Master’s Thesis; San Jose State Univ., 2012.

[4] Szor, P. “The Art of Computer Virus Research and Defence”; Addison Wesley Professional, 2005.

[15] Priyadarshi, S. “Metamorphic Detection via Emulation”; Master’s Thesis, Jose State Univ. 2011.

[16] You, I.; Yim, K. “Malware Obfuscation Techniques: A Brief Survey”; Proc. of Int. Conference on Broadband, Wireless Computing, Communication and Applications 2010.

[17] Mishra, P. “A Taxonomy of Software Uniqueness Transformations”; Master’s Thesis; San Jose State Univ., 2003.

[18] Patel, M. “Similarity Tests for Metamorphic Virus Detection”; Master’s Thesis, Jose State Univ., 2011.

[19] Yalzarooni, K. M. “Malware Variant Detection”; Requirements for the Degree of Doctor of Philosophy, Univ. College London, 2012.

[20] Jafarpour, B. “Cellular Learning Automata, Machine Learning Report”; Amirkabir Univ. of Tech. Computer Eng. and IT Dept., Spring 2006.

[21] Maybodi, M.; Baygi, H.; Taherkhani, M. “Cellular Learning Automata”; Proc. of the Sixth Conf. of Computer Society of Iran, 2001, 153-163 (In Persian).

[22] “Malware Data Base”; http://borax.poluxhosting.com/madchat/vxdevl/vxsrc, 2008.

[23] “Malware Data Base”; http://www.vx.netlux.org; 2007.

[24] Antivirus Comparatives “On-demand Detection of Malicious Software”; www.av-comparatives.org, March 2012.

[25] Gao, X.; Stamp, M. “Metamorphic Software for Buffer Overflow Mitigation”; Master’s Thesis, San Jose State Univ., 2008.

[26] Milgo, C. E. “Statistical Tools for Linking Engine Generated Malware to its Engine”; Master’s Thesis, Columbus State Univ., 2009.

[5] Desai, P. “A Highly Metamorphic Virus Generator”; Int. J. Multimedia Intelligence and Security 2010, 1, 402-427.

[6] Borello, J.; Filiol, E.; Mé, L. “From the Design of a Generic Metamorphic Engine to a Black-Box Classification of Antivirus Detection Techniques”; J. Comput. Virol. 2010, 6, 277-287.

[7] Lin, D.; Stamp, M. “Hunting for Undetectable Metamorphic Viruses”; J. Comput. Virol. 2011, 7, 201-214.

[8] Sayed Hamzeh, J. “A Novel Metamorphic Engine with Capability of Automatic Production of Mutated Metamorphic Malwares with Assimilation to Benign Executives Approach”; Master’s Thesis, Iran Azad Univ. Shabestar, 2011 (In Persian).

[9] Bashari, R. B.; Masrom, M.; Ibrahim, S. “Camouflage in Malware: From Encryption to Metamorphism”; Int. J. Comput. Sci. Network Secur. 2012, 12, 74-83.

[10] Khosravi, M. “Design and Implement a Metamorphic Engine for Generate Malware with Similarity to Benign Program”; Master’s Thesis, Islamic Azad Univ. Broujerd, 2012 (In Persian).

[11] Govindaraju, A. “Exhaustive Statistical Analysis for Detection of Metamorphic Malware”; Master’s Thesis, Jose State Univ. 2010.

[12] Dixit, N. K.; Mishra, L.; Charan, M. S.; Dey, B. K. “ The New Age of Computer Virus and Their Detection”; Int. J. Network Security & Its Applications 2012, 4, 79-96.

[13] Chuan, L. L.; Ismail, M.; Yee, C. L.; Jumari, K. “ A New Generic Taxonomy of Malware Behavioral Detection and Removal Techniques”; J. Theor. Appl. Inform. Tech. 2012, 42, 260-270.

[14] Wong, W.; Stamp, M. “Hunting for Metamorphic Engines”; J. Comput. Virol. 2006, 2, 211-229.