Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2011, Article ID 569829, 28 pages
doi:10.1155/2011/569829

Research Article

Evaluating Grayware Characteristics and Risks

Zhongqiang Chen,[1] Zhanyan Liang,[2] Yuan Zhang,[3] and Zhongrong Chen[4]

[1] *Yahoo! Inc., Sunnyvale, CA 94089, USA*
[2] *Department of Mathematics, Guangxi University of Finance and Economics, Guangxi 530003, China*
[3] *Department of Mathematics, Florida State University, Tallahassee, FL 32306, USA*
[4] *Corporate Accounts, Shire Pharmaceuticals, Inc. Wayne, PA 19087, USA*

Correspondence should be addressed to Zhongqiang Chen, zqchen97@yahoo.com

Received 8 March 2011; Revised 22 June 2011; Accepted 28 June 2011

Academic Editor: Yueh M. Huang

Grayware encyclopedias collect known species to provide information for incident analysis, however, the lack of categorization and generalization capability renders them ineffective in the development of defense strategies against clustered strains. A grayware categorization framework is therefore proposed here to not only classify grayware according to diverse taxonomic features but also facilitate evaluations on grayware risk to cyberspace. Armed with Support Vector Machines, the framework builds learning models based on training data extracted automatically from grayware encyclopedias and visualizes categorization results with Self-Organizing Maps. The features used in learning models are selected with information gain and the high dimensionality of feature space is reduced by word stemming and stopword removal process. The grayware categorizations on diversified features reveal that grayware typically attempts to improve its penetration rate by resorting to multiple installation mechanisms and reduced code footprints. The framework also shows that grayware evades detection by attacking victims' security applications and resists being removed by enhancing its clotting capability with infected hosts. Our analysis further points out that species in categories *Spyware* and *Adware* continue to dominate the grayware landscape and impose extremely critical threats to the Internet ecosystem.

1. Introduction

Grayware, an umbrella term for software with unwanted or undesirable features and functionalities, imposes serious security threats to Internet activities as it is mainly designed to profile users' computing habits, capture sensitive data, and steal business secrets [1, 2]. The information collected by grayware could be used to obtain financial assets, commit organized crimes, or trade for profits [3]. For instance, the world's largest attempted robbery against London offices of a Japanese bank is conducted via a spyware [4], while the keylogger surreptitiously implanted at some Kinko's stores exposes banking accounts and their passwords to attackers [5]. With its grayware products that track users' surfing behavior, a grayware company has established the seventh largest decision-support database in the world [6]. Motivated by financial gains, grayware typically attempts to infect as many hosts as possible through diversified penetration mechanisms such as drive-by download, deceptive installations, or vulnerability exploitations [7]. To generate a continuous revenue, grayware usually resides permanently on affected machines and deeply permeates into victim systems by all means including modification of autostart configurations or registry databases, so that it can survive system reboots and crashes [8]. To elude detection, grayware routinely uses obfuscation techniques, lowers security levels of infected systems or even terminates security services such as antivirus, firewall, and antispyware products [9, 10]. By attaching itself to a variety of legitimate processes, grayware significantly increases its clotting capability within affected hosts making it extremely difficult to be completely removed and essentially expanding its life span [8].

The incentive for profits has propelled grayware into its drastic proliferation as demonstrated by the rapid expansion of species collected in the Trend Micro grayware encyclopedia [11]. Starting from only 335 strains in 2004, the encyclopedia added 14,437 new species in 2005 and another 49,310 in 2006; upto the first half of 2008, it has 86,834 specimens in its repertoire. It has been established that more than 90% of Internet-connected hosts are infected with grayware and each victim machine accommodates 28 species on average [12]. In addition, the devastating impact on confidentiality, integrity, and availability (CIA) has rendered grayware the second most critical threat to the Internet ecosystem and continue to be the primary risk to cyberspace [13]. The incompatibility of different grayware strains residing on the same victim hosts also significantly affects system stability and productivity. In this regard, more than 12% technical supports in Dell service center are due to grayware; while the grayware-inflicted computer crashes reported to Microsoft cost billions to repair [5]. Moreover, the rampant growth of grayware inevitably hinders the prosperity of e-commerce as 44% of network users have dramatically reduced their Internet activities to avoid identity thefts [14]. The projected worldwide business expenditure on grayware defense will increase from $214 million in 2006 to $1.4 billion by 2010, further indicating the formidable and costly fight against grayware [15].

Falling into the "gray area" between legitimate applications and malicious software (i.e., malware) such as virus and worm, grayware is usually distributed by bundling with other legal software packages including freeware and shareware, which obtain users' consent for installation via an End User License Agreement (EULA) [4]. Unfortunately, the lengthy EULAs presented by such packages are typically vague and deceptive on functionality descriptions. For example, the 9,400-word EULA from the package consisting of *C2 Media, AdIntelligence*, and *Alset* could take hours even for the fastest readers to go through without taking into consideration its narrow display window and small font sizes [16]. It is also quite common for grayware EULAs to provide incomplete disclosures on bundled components as manifested by the Peer-to-Peer (P2P) application *Grokster*: its installation further invites another 14 species including *BullGuard, Cydoor*, and *IGetNet*, all of which are not disclosed in the EULA [17]. Compared to its malware counterpart that is usually created in an underground fashion, grayware is mainly developed by commercial companies, rendering the vulnerability of anti-grayware producers to lawsuits [18]. In this context, Zone Labs is sued by grayware 180Solutions in the pretext of trade libel and unfair practices, simply because products from the latter are labeled as grayware owing to their deceptive installation avenues [19]. Similarly, Symantec is caught into a fierce lawsuit by classifying *Hotbar*, a product from Claria, as grayware and removing it from infected systems [18].

Without doubt, the growing number of lawsuits between grayware and anti-grayware camps stems from the existence of the "gray area" and the lack of a well-formed grayware definition [20]. The complexity of grayware identification and categorization is further exacerbated when business partnership is established between grayware and anti-grayware vendors [21]. By admitting adware companies to be its members, the collapse of the Consortium of Antispyware Technology is unavoidable due to its loss of credibility [21]. It is natural that products from adware producer WhenU are legitimized thanks to its partnership with antispyware company Aluria. Evidently, the business cooperations between grayware and anti-grayware further blur the boundary between legal applications and grayware. Moreover, the widespread of grayware is also driven by the ever-increasing investments from merchants and affiliate networks. To this end, the adware WhenU obtains a large amount of fund directly from its largest customers such as Priceline and J. P. Morgan Chase by displaying their advertisements via adware products [22]. Similarly, Google actually helps grayware purveyors such as 180 Solutions and Ask Jeeves by paying them to advertise through grayware [23, 24]. Clearly, the complication of grayware classification renders it not only a technical challenge, but also a serious issue involving legal, social, economic, and human factors [2].

The explosive population of grayware and its disastrous effects on infected systems make it vitally important to characterize the behavior of grayware and analyze its risks so that effective detection and defense strategies can be developed [25]. We therefore propose a grayware categorization framework termed Grayware Assessor that defines taxonomic features for the entire grayware life cycle, classifies species with respect to a variety of characteristics, and evaluates their threats to the Internet ecosystem. The proposed framework treats grayware categorization as a supervised learning problem and constructs a learning model for each taxonomic feature with the help of support vector machines (SVMs) [26]. Taxonomic features as well as their training data are automatically extracted from the Trend Micro grayware encyclopedia to avoid time-consuming manual operations. The training data can be further expanded with grayware entries that match telltale patterns unique to the taxonomic feature in question. We design word stemming and stopword removal process to reduce the dimensionality of the feature space formed by grayware feature vectors. To further decrease the dimensions of the feature space, we select features based on their potential information gain [27]. The categorization results are visualized with the help of self-organizing maps (SOMs) [28].

The large number of grayware entries in the Trend Micro grayware encyclopedia and their respective very short descriptions inevitably lead to a high-dimensional feature space as well as sparse feature vectors. Thus, we employ SVM techniques, superb for learning tasks with dense concepts and sparse feature vectors [29], to build learning models for taxonomic features. The proposed framework defines two types of taxonomic features, single-label and multilabel; in the former, only one label can be attached to each grayware entry while in the latter, a single grayware can simultaneously belong to multiple categories. For a multilabel feature, the framework reduces the modeling problem into a set of tasks, each of which differentiates a category of the feature from the rest, and trains an SVM binary classifier for each task. For a single-label feature, the Grayware Assessor constructs binary classifiers as well as a single multiclass SVM categorizer in order to attain better classification accuracy.

With the SVM learning models in place, the proposed framework systematically organizes grayware species according to various taxonomic features. For instance, it can yield a grayware hierarchy by initially clustering all strains on feature Grayware Type that includes categories such as *Spyware* and *Adware*, and then classifying each category with respect to feature Risk Level; the latter contains five classes: *Extreme, High, Medium, Mild*, and *Slight*. The resulting hierarchy facilitates the identification of spyware specimens that impose extremely critical threats to the Internet. More importantly, the proposed grayware categorization framework can automatically classify to-be-discovered grayware breeds with the established learning models and help evaluate the grayware evolution and its risks. Using the grayware categorization derived from the proposed framework, we ascertain that grayware typically employ multiple attack avenues to improve its penetration rate and carry a variety of payloads to maximize its exploits on victim systems. It is also revealed by the framework that grayware evades detection by reducing its code footprint and enhancing its clotting capability with victim systems. The grayware trend analysis conducted with the proposed framework indicates that *Spyware, Adware*, and *Cracking Application* are the most serious threats to the Internet.

The rest of our presentation is organized as follows. Section 2 presents related work on grayware as well as its taxonomic features and classifications. Section 3 outlines the proposed framework that automates training data generation and learning model construction for grayware categorizations. The classifications based on taxonomic features associated with various stages of grayware life cycle are discussed in Sections 4, 5, and 6. Section 7 evaluates grayware evolution and analyzes its threats to the Internet. Our main conclusions and future work can be found in Section 8.

2. Related Work

The complexity of grayware characteristics and its ever evolving behavior renders that a widely accepted definition for grayware has not yet been established [2]. In this regard, some definitions consider a program as grayware if it establishes surreptitious communication channels and has negative consequences on end systems [9, 17, 30], while others concentrate on surveillance and information gathering capability of a program and distinguish grayware from legal software mainly based on users consent or permission for its installation [4, 20]. However, the fact that grayware often live in the gray area around the demarcation line between legitimate and malicious software makes it difficult to quantitatively measure the stealthiness of a program as well as the truthful disclosure of its purpose via EULAs or terms of service (TOS) [2]. It has been shown that most users regret and reverse their decision after they are clearly notified the functionalities of installed grayware [31]. The lack of a well-formed grayware definition surely affects the boundary of its universe and categorization schemes of its species. To this end, only 26,517 grayware strains are gathered in the Computer Associates encyclopedia [32], a much smaller population compared with that collected in Trend Micro which has 86,834 species. Along the same line, grayware categorization schemes diversify significantly in different grayware advisories, for instance, *Backdoor, Denial of Service (DoS)*, and *Trojan* are considered as grayware genre in [32] but treated as malware in [11] and therefore excluded from its grayware repertoire.

Grayware categorization and generalization necessitates the development of taxonomies so that the universe of grayware can be organized systematically and its species can be easily identified [13, 33]. To be of value, a grayware taxonomy should demonstrate the properties of objectivity, operability, and repeatability, and its classification scheme should be deterministic and specific [34]. Determinism calls for intrinsic features to specimens under study so that their extraction can be automated independent of observers while specificity ensures both the uniqueness and unambiguity of taxonomic features. Unfortunately, the sophisticated nature of grayware makes it difficult to define unambiguous features that can be applied to entire grayware universe. For instance, taxonomic feature User Consent plays a critical role in grayware characterization schemes of [4, 20]; however, it is rather subjective and may cause a data-gleaning program to be categorized as *Safe Data Collection* or *Data Theft* by different observers. In the same manner, the feature Attacker Intent is used to differentiate *Nuisance Spyware* from *Malicious Spyware* in [11]; clearly, the intent of attackers is difficult to evaluate and measure. The fact that grayware encyclopedias such as Trend Micro and Computer Associates are created manually by domain experts typically lead to incomplete information collection and analysis for many strains. To this end, only 38 out of 86,834 species in the Trend Micro grayware encyclopedia [11] provide information for the feature In The Wild, making any generalization attempt statistically insignificant.

The lack of determinism and specificity definitely affects the usability of grayware classification schemes; however, the main barrier for their applications in real-world environments is their inability to classify newly discovered species without human intervention. Should significant manual intervention be required, for instance, in categorizing all entries in the Trend Micro grayware encyclopedia with respect to a variety of taxonomic features, it is impossible to keep pace with the ever-growing grayware population. Therefore, machine learning and data mining techniques have been applied in an effort to diminish the dependency on domain experts as far as grayware and malware categorization is concerned [35, 36]. The behavior-based malware/spyware classification method built on case-based reasoning techniques stores known malware/spyware in a database and finds the sample from the database with the minimum distance from the program under study, and declares it as malicious, should the distance be within the specified range [37]. With the help of a self-organizing map (SOM), the proposal of [38] automatically groups vulnerabilities into four categories—denial of service, deception, reconnaissance, and unauthorized access—by clustering vulnerability textual descriptions into a two-dimensional array of nodes [28]. Similarly, SOM techniques are also used to automatically discover patterns hidden in vulnerability descriptions by first organizing vulnerabilities into a self-organizing map consisting of multiple nodes, and then labeling each node with categories such as denial-of-service, worm, and buffer overflow [39].

To automate the grayware classification process and offer grayware categorization and generalization capability, we treat grayware classification as a supervised learning problem and employ Support Vector Machines (SVMs) [26] to create learning models for taxonomic features. Derived from the Structural Risk Minimization (SRM) principle of the computational learning theory, SVMs seek learning functions that minimize classification errors on selected examples [26, 40] and have been successfully used in text classification, pattern recognition, and natural language processing [40, 41]. SVM binary classifiers have been proposed for the process of voluminous data with high-dimensional and sparse feature vectors commonly encountered in the real-world applications [42, 43]. For instance, in the context of the sequential minimal optimization (SMO) method [44], the large quadratic programming (QP) optimization problem of an SVM binary classifier is decomposed into a series of tasks that can be solved analytically. Similarly, in SVM-Light [42] and SVM-Torch [43], the model training process is accelerated with the use of working set and data caching. The SVM modeling complexity can be further reduced with the help of heuristics and domain-specific optimizations [45]. Compared with alternative machine learning methods including Naive Bayes, neural networks, and Decision Tree; SVMs achieve significantly better performance in terms of generalization [41].

When a taxonomic feature contains more than two categories, the learning task should differentiate multiple classes calling for a multiclass learning model [46]. The multiclass problem at hand can be decomposed through a multiclass-to-binary reduction method into a set of binary classification tasks, each of which distinguishes a single class from the rest [47]. Similarly, the multiclass learning problem can also be solved by partitioning categories into opposing subsets using error-correcting codes such as Hamming encoding so that binary classifiers can be trained based on subsets instead of individual classes [48, 49]. By treating the multiclass learning problem as a monolithic constrained optimization task with a complex quadratic objective function [50], a single multiclass categorizer can also be built with the multiclass-optimization method. In this regard, the constrained optimization performed by the multiclass categorizer is decomposed into a series of small steps so that each step involves only a subset of training data or constraints and can therefore be solved analytically [51]. In the proposed framework, we build SVM learning models with both multiclass-to-binary reduction and multiclass-optimization methods.

3. The Proposed Grayware Categorization Framework

Threat advisories such as the Trend Micro grayware encyclopedia could play a much valuable role in grayware risk management should they provide the functionalities of categorization and generalization. For instance, prompt incident response demands that predominant grayware penetration mechanism be quickly determined; similarly, it is equally desirable to derive the grayware distribution with respect to specific taxonomic features such as Carried Payload or Risk Level. Unfortunately, such functionalities are unavailable in current security advisories. Resorting to machine-learning and data-mining techniques, the proposed grayware categorization framework empowers the Trend Micro grayware encyclopedia with the capabilities of categorization and summarization. Considering grayware classification as a supervised learning problem, the designed framework collects training data automatically for taxonomic features and builds learning models with support vector machines. The dimensionality of grayware feature space formed by attributes selected according to their information gain is reduced with the help of word stemming and stopword elimination techniques; while classification results are visualized via self-organizing maps.

3.1. The Lifecycle of Grayware. Along with the dramatic explosion of the grayware universe, variants in each individual grayware family also expand at an increasingly fast rate. Sharing similar characteristics and behavior, members of the same grayware family are treated differently by various grayware encyclopedias: the entire grayware family is represented with a single entry as in the Symantec threat advisory or each variant has its own independent entry manifested by the Trend Micro grayware encyclopedia. It is therefore inevitable that boundaries of grayware universe defined in different collections may not coincide. In this regard, Symantec and Trend Micro grayware encyclopedias gather 6,767 and 86,834 species, respectively [11, 52]. The ever-increasing grayware population also renders it extremely difficult for a grayware encyclopedia to thoroughly analyze each specimen. To overcome this issue, grayware encyclopedias typically cross-reference each other to share and correlate information. For instance, Mcafee security advisory usually references to other resources including Trend Micro grayware encyclopedia in addition to its own grayware descriptions.

As demonstrated in Table 1, the Trend Micro grayware encyclopedia not only provides adequate information on species, but also organizes entries with a relatively stable structure making it possible to automatically process the collected data. The descriptions on spyware *TSPY_LINEAGE.GL* and adware *ADW_ALEXA.AK* outlined in Table 1 clearly follow a well-defined template consisting of three parts: *General*, *Description*, and *Detail*. Each part comprises a series of fields that are used to characterize the grayware specimen in question. For example, feature System Impact in part *General* attempts to measure strain's effect on system integrity; while Compression Type of part *Detail* describes the specimen's behavior on packing its files to reduce storage footprints and defeat detection. Compared with other security advisories, the Trend Micro grayware encyclopedia defines a much rich set of features to describe grayware species and is frequently referenced by other security databases. Therefore, we select Trend Micro grayware encyclopedia as the testbed for the proposed framework and transform it into a grayware classifier with the capability of categorization and generalization.

Table 1: *TSPY_LINEAGE.GL* and *ADW_ALEXA.AK* in Trend Micro grayware encyclopedia.

TSPY_LINEAGE.GL	*ADW_ALEXA.AK*
General	
Type: Spyware; In the wild: No; Destructive: Yes; Systems Affected: Windows 95, 98, ME, NT, 2000, XP, Server 2003; Encrypted: No; Language: English; System Impact: High; Information Exposure: High;	Type: Adware; In The Wild: No; Destructive: No; Systems Affected: Windows 98, ME, NT, 2000, XP, Server 2003; Encrypted: No; Language: English; System Impact: Medium; Information Exposure: Medium;
Description	
Installation and Autostart Technique: On Windows NT, 2000, XP, and Server 2003, this spyware drops a copy of itself in the Program Files folder as SVHOST32.EXE. It then modifies the following registry entry to ensure its automatic execution at every system startup: . . . On Windows 95 and 98, it copies itself as RUNDLL32.EXE in the Windows folder and INTERNAT.EXE in the. . . *Information Theft*: This spyware steals and logs sensitive information from an affected system and the game . . . *Process Termination*: This spyware also terminates . . .	*Installation and Autostart Technique*: This adware may arrive on a system as a file downloaded by unsuspecting users while visiting Web sites. It may also be dropped by other grayware. Upon execution, it creates the folder Alexa Toolbar in the Program Files folder, then drops the following files . . . It then installs the dropped.DLL files on the infected system. As a result, the routines of ADW_ALEXA.AP are exhibited on the system. It creates the following registry keys: . . . *Other Details*: This adware registers itself as a Browser Helper Object (BHO) and adds additional search
Detail	
Initial Samples Received on: Sep 7, 2005; File Type: PE; Memory Resident: Yes; Compression Type: UPX; File Size: Varies; Payload 1: Moves system files to other folders; Payload 2: Terminates processes; Payload 3: Steals information;	Initial Samples Received on: Nov 2, 2006; File Type: PE; Memory Resident: Yes; Compression Type: No; File Size: 494,672 Bytes; Payload 1: Creates search functionalities on Internet browser; Payload 2: Redirects search queries

Every grayware specimen shares a similar life cycle: after being created, it first penetrates as many machines as possible, then transports malicious code termed payload into infected systems and conducts stealthy activities until it is eventually discovered and eradicated. More specifically, grayware life cycle can be divided into the following stages. (a) *Creation*. Grayware may be developed by script kiddies or commercial companies motivated by profits. Some species are bred in large scale with the help of investments from advertisers, merchants, and affiliate networks. (b) *Penetration*. Newly born grayware attempts to creep into victim systems with arsenal such as freeware/shareware, email attachments, and security vulnerability exploitation. (c) *Activation*. Payloads, the workhorse for continuous revenues, are shipped into infected systems. (d) *Discovery*. Grayware may be eventually identified due to its pernicious activities. (e) *Eradication*. Grayware is contained and removed from infected systems upon detection.

Although undergoing similar life cycle, grayware strains indeed manifest diversified behavior in various stages of their life time ranging from sheerly annoying to extremely malicious. Mainly based on functionality and behavior, species in the Trend Micro grayware encyclopedia are categorized into groups listed in Table 2. Comprising 59.97% of the entire repertoire, category *Spyware* is the largest class and its members are designed to profile users-computing/browsing habits, steal sensitive information, and capture screenshots or event logs. The next two most populated categories *Dialer* and *Adware* contribute 17.50% and 16.27% to the total grayware population, respectively; the former creates revenue for its owners by redirecting phone connections, while the latter generates financial gains through displaying advertisements. In comparison, categories *Toolbar, Keylogger*, and *Hijacker* are only sparsely populated, rendering them the minority of the grayware universe.

3.2. Design Rationale for the Proposed Framework. As demonstrated by spyware *TSPY_LINEAGE.GL* and adware *ADW_ALEXA.AK* of Table 1, each entry in the Trend Micro grayware encyclopedia is constructed with a template mainly consisting of three parts: *General, Description*, and *Detail*. In part *General*, the entry is characterized by a series of taxonomic features including *Type, Systems Affected*, and *Information Exposure*. Part *Description* presents an overview of the specimen in the entry and usually covers installation mechanisms, activities conducted, and symptoms of infected systems. Information in part *Detail* outlines techniques employed by grayware in question with respect to file compression methods, payload peculiarity, and message obfuscation. Although the Trend Micro grayware encyclopedia devises a dozen of taxonomic features to characterize grayware species, only a small portion of entries actually provide information on defined features due to the time-consuming and labor-intensive tagging process. For instance, only 8,301 out of 86,834 strains in the Trend Micro repertoire have information for feature Information Exposure; while less than 9.57% entries are labeled with respect to feature System Impact. Moreover, no taxonomic feature is designed in the Trend Micro grayware encyclopedia to characterize grayware behavior on attack avenues, which is highly important for grayware evaluation.

Grayware classification with respect to taxonomic features helps develop defense policies targeting clustered breeds with similar behavior instead of individual strain. Furthermore, organizing grayware species systematically

Table 2: Categories of species in the Trend Micro grayware encyclopedia.

ID	Name	Description	Num	Pct
1	Spyware	Install on user systems to track activities and collect data	52075	59.97
2	Dialer	Redirect calls to premium 900 numbers for financial purpose	15196	17.50
3	Adware	Display pop-ups/pop-unders and may be active after hosts terminate	14124	16.27
4	Hacking Tool	Various toolkits for malicious purposes	3672	4.23
5	Browser Helper Object	Plugins to browsers and track surfing habits and gather information	691	0.80
6	Cracking Application	Recover passwords from encrypted forms by brute-force/algorithms	624	0.72
7	Trojan Spyware	Stealthy programs to conduct harmful activities	608	0.70
8	Toolbar	Perform searching or file downloading and modify search results	394	0.45
9	Trackware	Track Web browsing activities for targeting ads or malicious purposes	116	0.13
10	Keylogger	Records keystrokes and send to attackers	85	0.10
11	Remote Access Trojan	Abuse for system administration or info theft	75	0.09
12	Hijacker	Manipulate system settings to reroute traffic	45	0.05
13	Dropper	Program that can retrieve and install other malware or grayware	82	0.09
14	Freeloader	Retrieve, install, and execute other software in background	49	0.06
15	Joke Program	Annoy users but do not infect files	425	0.49

according to their characteristics could automate the classification of newly discovered grayware and shed light on the evolution of the grayware universe. However, the large population in the Trend Micro grayware encyclopedia essentially renders it inefficient and impractical to manually categorize species based on a variety of taxonomic features. We therefore propose a grayware categorization framework termed Grayware Assessor that automates the grayware categorization and generalization. The Grayware Assessor not only automatically extracts features from Trend Micro grayware encyclopedia, but also designs new taxonomic features to cover the entire grayware life cycle. For instance, features Attack Avenue and Registry Key are defined to portray grayware characteristics in *Penetration* and *Eradication* stages of its life span.

Each taxonomic feature has its dimensionality—the number of categories, in this context, features Attack Avenue and System Impact contain ten and three classes, respectively, the former includes categories *Dropped by Malware, Drive-by Download*, and *Network File Shares*, while the latter holds classes *Low, Medium*, and *High*. A taxonomic feature is multilabel if a grayware specimen can be simultaneously assigned to multiple categories, and single-label otherwise. In the proposed framework, feature Attack Avenue is multilabel as a grayware specimen could resort to multiple penetration mechanisms at the same time; for instance, adware *ADW_ALEXA.AK* successfully infects victim machines by both *Drive-by Download* and *Dropped by Malware* methods. On the other hand, feature System Impact is single-label, thus spyware *TSPY_LINEAGE.GL* of Table 1 is only put into category *High* while adware *ADW_ALEXA.AK* is considered to impose less serious impact on affected systems compared to *TSPY_LINEAGE.GL* and is therefore assigned to class *Medium*.

The framework carries out grayware categorization on a taxonomic feature in three stages: grayware representation, learning model construction, and unlabeled-sample classification. In the first stage, the description text for a specimen in the Trend Micro grayware encyclopedia is converted into a format feasible for machine learning. More specifically, each entry in the Trend Micro repertoire is considered as a bag of words by ignoring word positions in the text, then it is further represented with a feature vector by treating each word in the bag as a feature (or attribute) and word occurrences in the corresponding entry as its value. To reduce the dimensionality—number of attributes—of the feature space formed by all grayware feature vectors, the framework conflates words into their common roots by a stemming process. The feature space is further decreased by filtering out stopwords that have only grammatical functions and are typically articles and common prepositions. Moreover, the set of attributes is trimmed according to their information gains and only those features with most significant gains are kept in the set. The bias resulting from difference in sizes of grayware entries is eliminated by normalizing feature vectors to have unit length. To improve classification accuracy, the proposed framework scales each attribute of a feature vector with its inverse document frequency (IDF) [53] defined as the ratio between the grayware population and the number of entries containing the attribute (i.e., word) in question.

In the stage of learning model construction, the proposed framework first automatically collects training data for each taxonomic feature from the Trend Micro grayware encyclopedia and builds an SVM learning model. The fact that categorization information on taxonomic features in the Trend Micro grayware encyclopedia is manually created by domain experts usually leads to a very small set of training data being generated. Therefore, the framework expands the training data by identifying entries that match telltale patterns unique to taxonomic features. The sparse grayware feature vectors due to the extremely short description for each grayware entry motivates us to employ SVMs in the proposed framework which are superb for learning tasks with dense concepts and sparse feature vectors [29]. Little noisy or redundant information in grayware texts generated by domain experts also justifies the application of SVMs in the proposed framework as it mitigates the burden on feature selection that affects the performance of learning models. The classification performance of learning models is evaluated through the *n*-fold cross-validation procedure.

With the constructed learning models at hand, the proposed framework categorizes unlabeled grayware that is not part of the training data. For a multilabel feature, the learning model consists of multiple SVM binary classifiers, each of which discriminates one category from the rest, while for a single-label feature, a single multiclass categorizer is usually built. Grayware entries can be organized systematically according to a variety of taxonomic features with the help of the learning models. For instance, a grayware hierarchy can be constructed if we classify species with respect to a series of taxonomic features such as Grayware Type, System Impact, and Discovery Date. Clearly, the resulting hierarchy makes it straightforward to identify 2008-born spyware that seriously affects system integrity. Moreover, learning models built in the designed framework can also be used to categorize grayware strains that may be discovered in the future and assign them into the established hierarchical structure. Finally, categorization results are visualized with self-organizing maps (SOMs).

3.3. Grayware Feature Vectors. To transform a grayware entry into a structured representation suitable for automatic process by machine learning algorithms such as SVMs and SOMs, the proposed framework first treats the text of each grayware in its part *Description* as a sequence of tokens and represents it as a bag of words by ignoring token positions in the text. A feature vector for the grayware in question is then formed with each distinct token as an attribute and its occurrence frequency as value. The grayware feature space is the assembly of feature vectors for all species and its dimensionality is the key determinant for the computational complexity of the learning task. To reduce the dimensions of feature space, the proposed framework resorts to the *Porter Stemming* algorithm frequently used in Information Retrieval (IR) [54], so that tokens are conflated into their common stem roots by stripping plurals, past participles, and other suffixes. The resulting word stems are further converted to their lower-case counterparts to condense the feature space. For instance, after the stemming process, tokens *install, technique*, and *automatic* of *TSPY_LINEAGE.GL* in Table 1 are converted to their stems *instal, techniqu*, and *automat*, respectively, before they are put into the feature vector. The stemming results for entries *TSPY_LINEAGE.GL* and *ADW_ALEXA.AK* are presented in Row "*Word Count*" of Table 3.

The proposed framework devises a stopword-elimination process to further decrease the feature space dimensions. A token is considered as a stopword if it has only grammatical function without adding new meaning to sentences it involves. Stopwords in the proposed framework are typically articles, case particles, and conjunctions [55]. With the stopword-elimination process, tokens *this, in*, and *on* of entry *TSPY_LINEAGE.GL* are excluded from its feature vector; in the same manner, words *may, by*, and *then* of entry *ADW_ALEXA.AK* are also part of the stopword list and removed from the feature vector. The proposed framework also treats a token as a feature candidate only if it appears at least a specified number of times in the grayware repertoire (3 by default). To mitigate the impact by difference in sizes of grayware entries, the proposed framework normalizes every feature vector to unit length. We also improve classification performance by scaling each attribute of a feature vector with its inverse document frequency. With the aforementioned word stemming and stopword elimination process, the dimensionality of the feature space is reduced from 5,881 to 4,910. Row "*Feature Vector*" in Table 3 depicts the feature vectors for entries *TSPY_LINEAGE.GL* and *ADW_ALEXA.AK*. The nonzero attributes in feature vectors of *TSPY_LINEAGE.GL* and *ADW_ALEXA.AK* are 61 and 72, respectively, making them extremely sparse compared to the 4,910 dimensions of the feature space.

3.4. Grayware Classifications with SVM and SOM. For a given set of training data $T = \{\overline{x}_i, y_i\}$, $(i = 1, \ldots, m)$, each data point (or example) $\overline{x}_i \in R^d$ with d features and a true label $y_i \in Y = \{l_1, \ldots, l_k\}$, a supervised learning task is to construct a model that attempts to balance the classification accuracy on T and its generalization capability on unseen examples. A classification error occurs when the label of an example assigned by the model contradicts its true label. Support vector machines (SVMs) attempt to construct models that minimize the classification errors on randomly selected examples [26]. When the label set is $Y = \{l_1 = -1, l_2 = +1\}$, the learning model—a binary classifier—distinguishes data points in positive $(+1)$ category from its negative (-1) counterparts with a separating hyperplane that maximizes the summation of its shortest distances to the closest positive and negative examples. The separating hyperplane can be expressed as $\overline{w} \cdot \overline{x} + b = 0$, here the weight vector $\overline{w} \in R^d$ is normal to the hyperplane, operator $(\cdot)$ computes the inner product of vectors $\overline{w}$ and $\overline{x}$, and b is the bias. The objective function of an SVM binary classifier can be expressed to minimize $\|w\|^2/2 + C_b \sum_i^m \xi_i$ with parameter C_b a penalty to classification errors, and ξ_i $(\xi_i \geq 0,\ i = 1, \ldots, m)$ a nonnegative slack variable for the ith example in T so that $\overline{w} \cdot \overline{x}_i + b \geq +1 - \xi_i$ for positive examples and $\overline{w} \cdot \overline{x}_i + b \leq -1 + \xi_i$ for negative examples. Clearly, the parameter C_b controls the trade-off between training errors and classification accuracy.

In practice, the objective function of a binary model is transformed into its Wolfe dual form to maximize $\sum_i^m \alpha_i - (1/2) \sum_{i,j}^m \alpha_i \alpha_j y_i y_j \overline{x}_i \cdot \overline{x}_j$, subjected to $0 \leq \alpha_i \leq C_b$ $(i = 1, \ldots, m)$ and $\sum_i^m \alpha_i y_i = 0$. Parameter α_i $(i = 1, \ldots, m)$ is a nonnegative multiplier for each constraint. Obviously, the objective function in the Wolfe form is convex and the constraints also form a convex set, rendering it a convex quadratic programming (QP) problem [41]. The solution to the Wolfe dual problem is given by $\overline{w} = \sum_i^m \alpha_i y_i \overline{x}_i$ and

Table 3: Feature vectors for spyware *TSPY_LINEAGE.GL* and adware *ADW_ALEXA.AK*.

Part	*TSPY_LINEAGE.GL*	*ADW_ALEXA.AK*
Word count	instal: 1 autostart: 1 techniqu: 1 window: 7 nt: 3 2000: 2 xp: 2 server: 2 2003: 2 spywar: 4 drop: 2 copi: 5 program: 1 file: 4 folder: 7 modifi: 1 follow: 2 registri: 2 entri: 2 ensur: 1 automat: 1 execut: 1 system: 5 startup: 1 95: 2 98: 2 rundll32.ex: 1 internat.ex: 2 inform: 2 theft: 1 steal: 1 log: 1 sensit: 1 affect: 2 game: 1 lineag: 1	instal: 2 autostart: 1 techniqu: 1 adwar: 2 arriv: 1 system: 6 file: 9 download: 2 unsuspect: 1 user: 1 visit: 1 web: 2 site: 1 drop: 3 graywar: 1 execut: 1 creat: 2 folder: 4 alexa: 2 toolbar: 4 program: 2 file: 9 follow: 2.dll: 1 infect: 3 result: 1 routin: 1 adw: 4 exhibit: 1 registri: 1 kei: 1 detail: 1 regist: 1 browser: 2 helper: 1 object: 1
Feature vector	instal: 0.0010 autostart: 0.0106 techniqu: 0.0101 window: 0.0485 nt: 0.0216 2000: 0.0144 xp: 0.0144 server: 0.0149 2003: 0.0151 spywar: 0.0036 drop: 0.0147 copi: 0.0456 program: 0.0003 file: 0.0281 folder: 0.0553 modifi: 0.0127 follow: 0.0146 registri: 0.0156 entri: 0.0163 ensur: 0.0097 automat: 0.0085 execut: 0.0076 system: 0.0034 startup: 0.0085 95: 0.0286 98: 0.0146 rundll32.ex: 0.0163 internat.ex: 0.0344 inform: 0.0032 theft: 0.0104 steal: 0.0075 log: 0.0105 sensit: 0.0138	instal: 0.00138 autostart: 0.0076 techniqu: 0.0072 adwar: 0.0077 arriv: 0.0060 system: 0.0029 file: 0.0450 download: 0.0116 unsuspect: 0.0091 user: 0.0002 visit: 0.0070 web: 0.0014 site: 0.0031 drop: 0.0157 graywar: 0.0100 execut: 0.0054 creat: 0.0066 folder: 0.0225 alexa: 0.0430 toolbar: 0.0488 program: 0.0004 file: 0.0450 follow: 0.0104.dll: 0.0075 infect: 0.0288 result: 0.0120 routin: 0.0062 adw: 0.0462 exhibit: 0.0117 registri: 0.0056 kei: 0.0076 detail: 0.0086 regist: 0.0098

parameter b can be obtained with equation $\alpha_i(y_i(\overline{w}\cdot\overline{x}_i+b)-1)=0$ for any $\alpha_i\neq 0$. Data points with their corresponding $\alpha_i>0$ form the set of support vectors $S=\{s_1, s_2,\ldots\}$. An unseen example $\overline{x}$ is assigned label +1 if formula $\sum_{i=1}^{|S|}\alpha_i y_i \overline{s}_i\cdot\overline{x}+b$ is positive, and label -1 otherwise.

In case that the label set $Y=\{l_1=1,\ldots, l_k=k\}$ and $k>2$, a multiclass SVM learning model is built with two approaches in the proposed framework: *multiclass-to-binary reduction* and *multiclass-optimization* methods. In the multiclass-to-binary reduction method, the learning problem in question is reduced to a set of binary classification tasks and a binary classifier is built independently for each label l_k with the one-against-rest training technique [47]. More specifically, in constructing training data for the classifier designated to label l_k, data points with label l_k are considered as positive while the remaining examples are treated as negative, then an SVM binary learner is built. Therefore, the resulting learning model by the multiclass-to-binary reduction method consists of k binary classifiers. In comparison, the multiclass-optimization method defines a monolithic objective function with complex constraints covering all classes so that a single multiclass categorizer is constructed. Similar to an SVM binary classifier, the objective function for a multiclass categorizer is to minimize $\|W\|^2/2+C_m\sum_{i=1}^{m}\xi_i$ subject to $\overline{W}_{y_i}\cdot\overline{x}_i+\delta_{y_i,r}-\overline{W}_r\cdot\overline{x}_i\geq 1-\xi_i$ (for all i,r). Here, W is a matrix of weights with size $k\times n$, $\overline{W}_r$ is the rth row of W, $\delta_{i,j}$ is a loss function that generates an output of 1 if $i=j$ and 0 otherwise, and parameter C_m controls the balance between training errors and classification accuracy. Similar to the multiclass-to-binary reduction method, the objective function of the multiclass categorizer is also converted to its Wolfe dual problem for the derivation of its solution [51]. As the multiclass optimization method treats classes to be mutually exclusive, it is therefore mainly used to build learning models for single-label features. In comparison, the multiclass-to-binary reduction method can be used for both multilabel and single-label taxonomic features.

From the Wolfe dual form of the objective function and its solution for an SVM classifier, we can observe that data points appear only with the form of inner product (i.e., $x_i\cdot x_j$). By specifying a mapping function $\Phi: R^d\mapsto H$, we can transform feature vectors of training data from R^d into a space H with higher or even infinite dimensions so that the model constructed in H only depends on data points through functions of the form $\Phi(x_i)\Phi(x_j)$. With a kernel function $K(x_i,x_j)=\Phi(x_i)\Phi(x_j)$, we can replace $x_i\cdot x_j$ by $K(x_i,x_j)$ in objective functions and their constraints, and build the learning model in space H by using K without explicit computation of Φ. In the same manner, the label of an unseen example can be obtained by computing inner products of its feature vector and parameter w (or W) via function K instead of Φ. The kernels used in the designed framework can be polynomial, radial basis functions (RBFs), or sigmoid functions [26].

Compared to SVM-supervised learning techniques, self-organizing maps (SOMs) can be considered as unsupervised learning methods that transfer data points from a high-dimensional space into its low-dimensional counterpart so that instances with similar features in the former are spatially clustered together in the latter. Consisting of multiple components called neurons that are arranged into a hexagonal or rectangular grid, a SOM map associates each neuron with a weight vector having the same dimensionality d as the feature vectors of input data. The neurons in the map are trained in such a way that different parts of the map are activated by distinct input patterns; meanwhile, adjacent nodes of the map respond similarly to the same stimulus. The SOM training is conducted with competitive learning methods that compute the Euclidean distances between each input and the weight vectors of all neurons in the map and designates the cell with the smallest distance as the best matching unit (BMU). The weights of the BMU and its neighboring neurons in the SOM lattice are then adjusted so that they resemble the input vector and become the winner with high probability when encountering similar instances in the future.

The magnitude adjustment of a weight vector w_i for node i decreases along with time as well as its distance from the BMU according to formula $w_i(t+1) = w_i(t) + h_{ic}(t)\alpha(t)\{x(t) - w_i(\mathrm{t})\}$, where t is the training epoch, $x(t)$ is the input vector, $h_{ic}(t)$ is the neighborhood function with c the BMU, and α is the learning rate monotonically decreasing with t. The neighborhood function $h_{ic}(t)$ depends on the lattice distance between node i and the BMU c. In its simplest form termed *bubble*, a neighborhood set $N_c(t)$ is defined for each node c and $h_{ic} = 1$ if node $i \in N_c(t)$ or $h_{ic} = 0$ otherwise. Another neighborhood kernel used in the proposed framework is the *gaussian* function expressed as $h_{ic}(t) = \exp(-(\|r_c - r_i\|^2/2\sigma^2(t)))$ with r_c and r_i the radius vectors of nodes c and i, respectively. Evidently, the neighborhood for a neuron diminishes over time and vanishes completely at the end of the training process. The fact that the SOM not only adjusts the winner but also its neighboring cells during the training process leads to a spatial clustering of instances in adjacent parts of the map, rendering its topology preservation capability. By calibrating the SOM lattice with labeled input data so that the BMU for an input sample inherits the label of its corresponding input, the map can act as a classifier that tags each unknown data point with the label of its BMU. Furthermore, the calibrated map is also an excellent visualization utility due to its clustering ability and topology preservation property.

3.5. Construction of Training Data and Learning Models. The learning model construction by a supervised learning method requires the availability of training data expressed as $T = \{\overline{x}_i, y_i\}$, $(i = 1, \ldots, m)$, here $\overline{x}_i \in R^d$ is the feature vector for the ith data point and $y_i \in Y = \{l_1, \ldots, l_k\}$ is its true label. The construction of T is labor intensive and time consuming should it be prepared and labeled manually. The proposed framework therefore automates the training data collection by taking advantage of taxonomic features defined in the Trend Micro grayware encyclopedia as shown in parts *General* and *Detail* of spyware *TSPY_LINEAGE.GL* and adware *ADW_ALEXA.AK* of Table 1. The fact that only a small portion of entries provide information on taxonomic features in the Trend Micro grayware encyclopedia renders that the resulting training data may not be sufficient to build a reliable learning model. To overcome this issue, the proposed framework can be configured to enlarge the set of training data with entries that match keywords unique to the taxonomic feature in question. Keywords for a taxonomic feature are presented with regular expressions in the designed framework and pattern matching process is only performed on entries that are not yet in the training data.

With the training data for a taxonomic feature at hand, a learning model can be built with the help of SVMs. When the size of the label set Y for the taxonomic feature is two, a binary learning model is materialized. For a taxonomic feature with $|Y| > 2$, the learning problem is decomposed into $|Y|$ binary classification tasks with the multiclass-to-binary reduction method, and subsequently $|Y|$ binary classifiers are built with the one-against-rest training method. For instance, in constructing training data for class *High* of the taxonomic feature System Impact that consists of three categories *Low, Medium*, and *High*, spyware *TSPY_LINEAGE.GL* is treated as a positive sample but adware *ADW_ALEXA.AK* as a negative sample even though the latter has a true label *Medium*. For a single-label taxonomic feature, the proposed framework also creates a multiclass categorizer with the multiclass-optimization training method.

The training data for a taxonomic feature may not cover all entries in the Trend Micro grayware encyclopedia, leaving some grayware unlabeled. In addition, the rapid expansion in the grayware population also renders that most newly discovered species wait for being categorized. The proposed framework saves the learning models built for all taxonomic features and uses them to classify unlabeled grayware entries or newly identified species. For a multilabel taxonomic feature, an unlabeled grayware could be assigned to multiple categories as long as their corresponding binary classifiers output positive values for the grayware in question. In comparison, for a single-label taxonomic feature, the Grayware Assessor simply puts a grayware into the category with the largest output if the model is composed of multiple binary classifiers or the class with the highest confidence when a multiclass model is used.

The proposed framework employs an n-fold cross-validation method to evaluate the performance of a learning model in terms of classification accuracy, precision, recall, and F_β measure. The set of training data T for a taxonomic feature is first partitioned into n equally sized groups $\{\mathrm{T}_i, i = 1, \ldots, n\}$, and each group assumes the grayware distribution that is similar to T so that each partition contains examples from all possible classes. Then the cross-validation process carries out the following steps in the ith of its n iterations: (a) *training phase*: partition T_i is held out to act as the validation set while the remaining $(n-1)$ partitions are combined together to form a new training set $A_i = \bigcup_{j \neq i} T_j$. A learning model L_i is built with training data set A_i; (b) *labeling phase*: data points in validation set T_i are labeled with the learning model L_i; an example in T_i is classified correctly if its assigned label by L_i is the same as its true label; (c) *measuring phase*: performance metrics such as classification accuracy, precision, and recall for learning model L_i are computed.

In addition to the classification accuracy that is defined as the ratio of the number of correctly classified examples over the size of the validation set, the proposed framework also resorts to the F_β-measure that computes the weighted harmonic mean of precision P and recall R with the formula $F_\beta = (1+\beta^2)PR/(\beta^2 P + R)$; here, the precision P for a class is defined as the ratio between the number of correctly labeled samples and the total number of samples that are assigned to the class; on the other hand, the recall R is the ratio of correctly labeled samples over the total number of samples that actually belong to the class. By setting $\beta = 1$ so that the precision P and the recall R are considered to be equally important, we obtain the F_1-measure as $F_1 = 2PR/(P+R)$. The performance metrics such as classification accuracy and precision attained by a learning model are the average of measures obtained in the n iterations of the above-described cross-validation process.

4. Penetration Mechanisms Utilized by Grayware

The first step for grayware to transform a victim host into a profit resource is to find its entry point to the latter. The motivation for financial gains drives grayware to infect as many machines as possible by trying out every conceivable means including social engineering, file sharing, and security loophole exploitation. Successful system penetration heavily depends on the compatibility between grayware code and computing platforms on target machines. It is equally vital for grayware to ensure the operability of its executables on target environments. In this section, we define taxonomic features to categorize the behavior manifested by grayware in the stage *Penetration* of its life cycle.

4.1. Computational Platforms Targeted by Grayware.

A computing platform defines the architecture for a computer system which mainly consists of hardware, operating system (OS), and runtime libraries. Acting as the mainstay of a computing platform, an OS manages and coordinates computer resources by providing a pool of services accessed via system calls or application programming interfaces (APIs). Like legitimate applications, grayware typically materializes its functionalities via OS services. Moreover, the homogeneity demonstrated by OSs derived from the same code base renders that grayware could successfully penetrate virtually all variants of an OS family. For instance, spyware *TSPY_LINEAGE.GL* is capable of compromising seven members of the Windows family including *Windows 95* and *Server 2003*. To improve penetration rate, grayware attempts to adjust its attack strategy and tailor its installation mechanisms according to OS types on target hosts, leading to divergent behavior across different platforms. In this regard, spyware *TSPY_LINEAGE.GL* masquerades itself as *SVHOST32.EXE* when attacking *Windows NT* but impersonates as *RUNDLL32.EXE* for *Windows 95* platform.

Taxonomic feature Affected Platform defined in the framework characterizes OSs attacked by grayware. Grayware usually targets computing platforms with large user base or rich applications. For example, *Linux* is only attacked by a few grayware species including hacking tool *HKTL_CALLBACK* variants. In contrast, a vast majority of grayware species target *Windows 95, 98, ME, NT, XP, 2000*, and *2003*. Thus, we mainly focus on the above seven *Windows* members and designate them as the categories of the feature Affected Platform; for convenience, we also assign them identifiers 1 to 7 in the given order. As demonstrated by spyware *TSPY_LINEAGE.GL* of Table 1, grayware can successfully penetrate multiple OSs at the same time, we therefore treat Affected Platform as a multilabeled feature. The training data are constructed automatically by extracting entries from the Trend Micro grayware encyclopedia which have information on the field *System Affected*.

With the training data in place, we build a learning model termed *Stemming* by using the multiclass-to-binary reduction method and enabling stemming and stopword removal processes. The grayware distribution generated by model *Stemming* is described in Figure 1, and it clearly indicates that the majority of grayware attack *Windows 98, ME, NT, XP, 2000*, and *2003*; on the contrary, only a few species intrude *Windows 95* mainly because it is a legacy and out-of-fashion OS. Compared to the size 86,834 of the grayware repertoire, categories *Windows 98, ME, NT, XP*, and *2000* have nearly the same grayware population, indicating that they usually are attacked simultaneously and vulnerable to almost all grayware species. The categorization performance by the seven binary classifiers is presented in Figure 2. All binary learning models achieve similar categorization accuracies with the highest 99.80% attained by the classifier for *Windows XP* and the lowest 97.48% by the learner for *Windows ME*. On the other hand, the precision, recall, and F_1-measures delivered by the categorizer for *Windows 95* are much lower than those for other classes. The average classification accuracy attained by the seven binary classifiers is 98.34%, meanwhile, average precision, recall, and F_1-measure are 97.50%, 98.72%, and 0.981, respectively.

To investigate the impact on classification accuracy by word-stemming and stopword-elimination operations, we build another two learning models, *No-Stemming* and *No-Stemming-Stopword*, the former is obtained by skipping the word stemming process, while the latter is created by further leaving out the stopword-elimination procedure. Figure 3 outlines classification accuracies attained by the three models. Obviously, most binary classifiers in model *No-Stemming* outperform their counterparts in model *Stemming* with respect to categorization accuracy. In addition, average classification accuracy can be derived as 98.34%, 98.48%, and 98.46%, respectively, for models *Stemming, No-Stemming*, and *No-Stemming-Stopword*, further indicating model *No-Stemming* the best performer. The fact that models *No-Stemming* and *No-Stemming-Stopword* achieve better classification accuracies than *Stemming* clearly reveals the negative effect of the stemming process. In the same manner, the slight deterioration in the categorization accuracy by model *No-Stemming-Stopword* compared to model *No-Stemming* manifests the harmful impact by stopwords. However, without the stemming process, model *No-Stemming* generates a 5,719-dimensional feature space, much larger than 4,910 for model *Stemming*; while model *No-Stemming-Stopword* further enlarges its feature space to contain 5,881 attributes by treating stopwords as features. A large feature space obviously increases the model training time, the proposed framework carries out word stemming and stopword elimination by default.

4.2. Types of Files Smuggled by Grayware.

To successfully penetrate infected systems and effectively execute on victim machines afterwards, grayware should pack itself in file formats compatible with target environments. It is typical that OSs define their own file formats and refuse to process files in incompatible forms. In this regard, the executable and link format (ELF) is mainly recognized by *Linux*, while the dynamic linked library (DLL) objects are unique to the *Windows* family. The proposed framework introduces taxonomic feature File Type to identify the file formats utilized

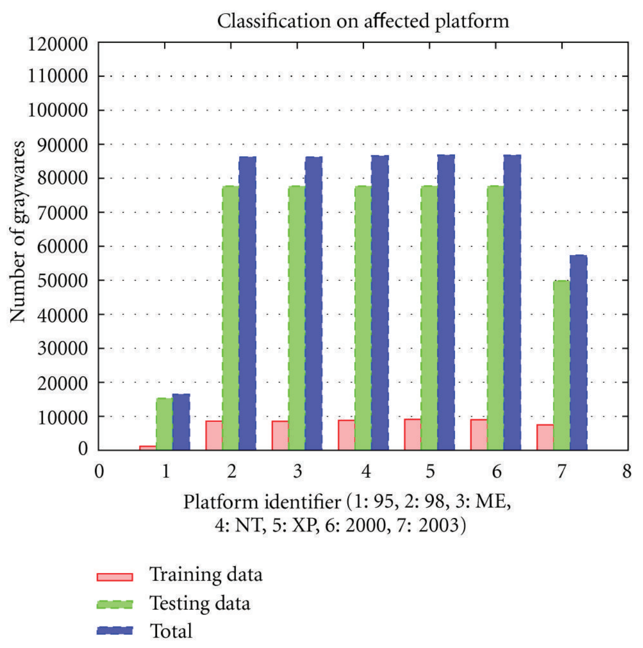

FIGURE 1: Classification on Affected Platform with stemming and stopword elimination.

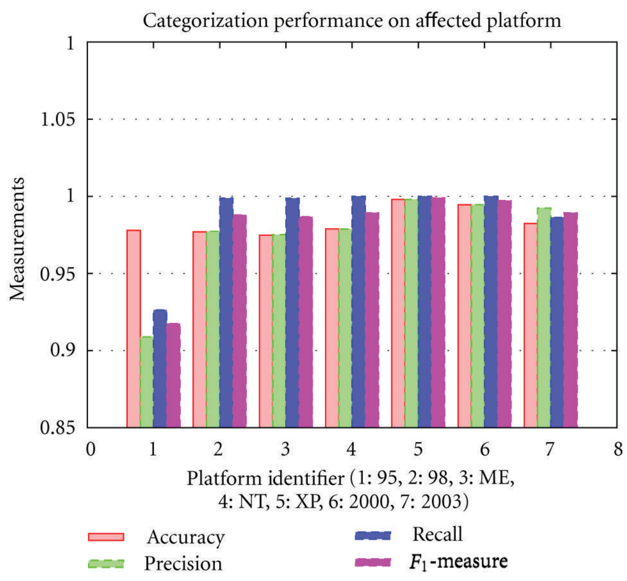

FIGURE 2: Classification accuracy, precision, recall, and F_1-measure for Affected Platform.

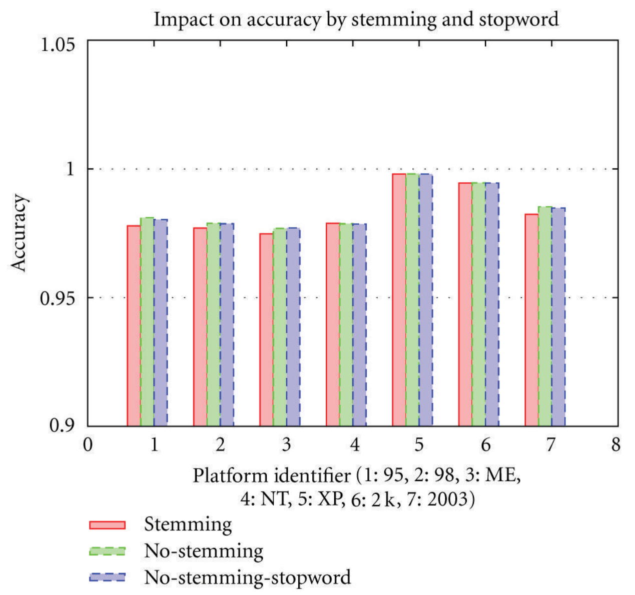

FIGURE 3: Accuracy on Affected Platform with/without stemming and stopword elimination.

by grayware to transport its code into target hosts. Based on file type popularity and information provided by the Trend Micro grayware encyclopedia, we designate four classes to the feature File Type: *EXE, DLL, PE*, and *Other*, and assign them identifiers 1–4, respectively. The *PE* file format defines a basic data structure to encapsulate information for OS loader so that the wrapped code can be executed in environments with different software architectures. Recognizable to *Windows 95, NT*, and other new versions, the *PE* file format is frequently used to represent object code and API import/export tables. The *Dynamic Link Library (DLL)* format describes shared libraries, ActiveX controls, or system drivers that are used by *Windows* and *OS/2*. Files in the *Executable (EXE)* format can be executed in *Windows* and *OS/2* families; in addition, objects such as bitmaps and icons associated with executables can also be represented in *EXE* files.

We collect training data for feature File Type by retrieving entries from the Trend Micro grayware encyclopedia. With the training data at hand, we construct a self-organizing map (SOM) that comprises a 10×10 grid of neurons, each of which is in hexagonal shape by employing the following procedure: (a) *initialization*: weight vectors of neurons, which have the same dimensions as feature vectors of input samples in training data, are initialized with random values; (b) *training*: weight vectors are updated with competitive learning techniques described in Section 3.5. Neighborhood function *bubble* and learning rate $\alpha = 0.05$ are utilized to identify adjacent cells for each node; (c) *refinement*: the map is further refined with a fine-granular learning rate $\alpha = 0.02$; (d) *calibration*: the best matched unit (BMU) for each input sample is identified and the BMU inherits the sample's label. The ultimate label of a neuron is determined with the simple majority vote mechanism.

The U-matrix of the SOM termed *SOM-Hexa* constructed according to the above procedure is obtained by representing each node with its average distance to its closest neighbors. Figure 4 depicts the U-matrix of the *SOM-Hexa* model for feature File Type, here, the origin of the map is at the upper-left corner, and each value of the matrix (i.e., distance) is converted into a gray level in [0, 100] with the darkest 0 gray scale denoting the largest distance. Figure 4 clearly demonstrates that SOM tends to cluster together samples with the same labels so that they reside at adjacent neurons. For instance, the six consecutive nodes locating at the left portion of the 8th row are occupied by category *EXE*;

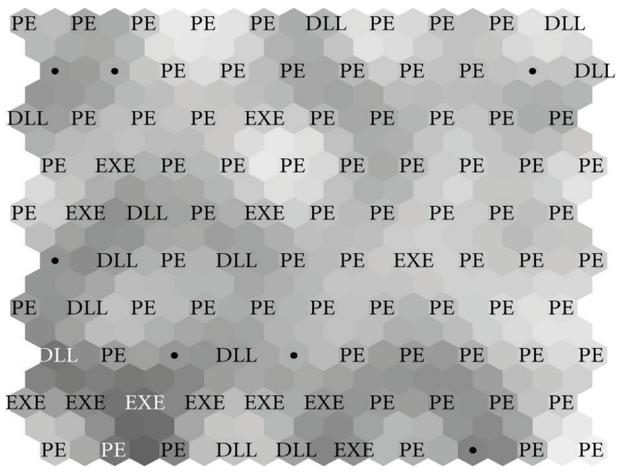
PE PE PE PE PE DLL PE PE PE DLL
• • PE PE PE PE PE PE • DLL
DLL PE PE PE EXE PE PE PE PE PE
PE EXE PE PE PE PE PE PE PE PE
PE EXE DLL PE EXE PE PE PE PE PE
• DLL PE DLL PE PE EXE PE PE PE
PE DLL PE PE PE PE PE PE PE PE
DLL PE • DLL • PE PE PE PE PE
EXE EXE EXE EXE EXE EXE PE PE PE PE
PE PE PE DLL DLL EXE PE • PE PE

Figure 4: Distances between reference vectors of hexagonal neighboring units on File Type.

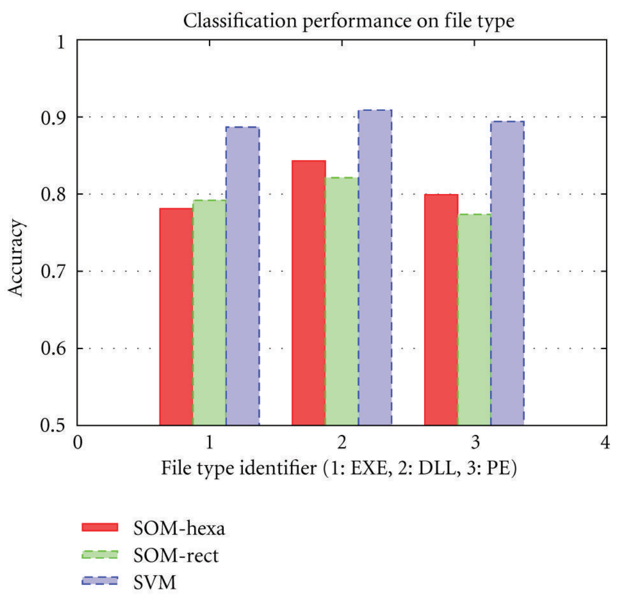

Figure 5: Classification accuracies on File Type by SVM, SOM-Hexa, and SOM-Rect.

similarly, neighbors of a cell with label *PE* are highly likely to share the cell's label. Moreover, the darker color at the left part of the map signifies that neurons at the right half of the SOM are much closer to each other. The fact that the majority of the map estate is occupied by category *PE* points out that feature vectors for classes *DLL* and *EXE* are much homogeneous.

By changing the neuron lattice from hexagonal to rectangular shape, we obtain another SOM map termed *SOM-Rect*. In addition, based on the same training data, we also build an SVM learning model with the help of the *multiclass-to-binary reduction* method. By using the cross-validation process, we evaluate the categorization accuracies of the three models, *SOM-Hexa, SOM-Rect*, and *SVM*, and present the results in Figure 5. Clearly, model *SOM-Hexa* outperforms *SOM-Rect* on categories *DLL* and *PE*, while *SOM-Rect* achieves a better classification accuracy on category *EXE*. By averaging the categorization accuracies of the three categories for each model, we obtain that *SOM-Hexa, SOM-Rect*, and *SVM* models offer classification accuracies of 80.78%, 79.56%, and 88.03%, respectively. It is evident that SVM model performs significantly better than its SOM counterparts. Nevertheless, the visualization capability of SOMs is also valuable to the grayware characteristics evaluation. According to the grayware classification on feature File Type by the SVM model, 53.36% grayware are transported in *DLL* format, while 25.80% and 20.84% ship as *PE* and *EXE*, respectively.

4.3. Attack Avenues Utilized by Grayware. Grayware transfers itself to victim machines via diverse attack avenues existing in various services and applications. The widespread of file-sharing applications including instant messaging (IM) and peer-to-peer (P2P) not only offers convenient installation channels for grayware due to their superb anonymity, but also provides a large infectable user base attributed to their ubiquity. The enormous amount of security loopholes harbored in networks and systems facilitates the automatic delivery of malicious code, while the flexibility of macro- and scripting languages further accelerates grayware dissemination. Similar to worm and malware, grayware also frequently employs social engineering as an effective mechanism tricking users to open grayware-carrying documents or visit malicious web sites. In the proposed framework, we define the taxonomic feature Attack Avenue that consists of grayware installation mechanisms outlined in Table 4.

With the help of pattern matching techniques, we form the set of training data for the feature Attack Avenue by retrieving entries in the Trend Micro grayware encyclopedia that match any patterns specified in Column *Sample Patterns* of Table 4. For example, according to the description of *ADW_ALEXA.AK* in Table 1, the strain in question is dropped into a target system by malware or installed manually by unsuspecting users. The pattern matching process performed by Grayware Assessor results in the categorization of *ADW_ALEXA.AK* into class *Dropped by Malware* due to the existence of keyword *dropped by* in its description. We then build an SVM learning model for the feature with the *multiclass-to-binary reduction* method and use it to classify the entire grayware repertoire. The grayware distribution generated by the model reveals that 58.56% species can piggyback in malicious software such as worm and virus, rendering *Dropped by Malware* the most favorite grayware installation mechanism. The next two frequently used attack avenues are *Components of Software* and *Bundle with Software*, the former packs grayware as components of other legitimate software, while the latter treats grayware as an independent program in packages. The drive-by download is also employed by many grayware species including both *TSPY_LINEAGE.GL* and *ADW_ALEXA.AK* of Table 1. In contrast, attack channels such as *Vulnerability Exploit, IM*, and *Network File Share* are only utilized occasionally by grayware.

Table 4: Installation mechanisms employed by grayware species.

ID	Channel	Description	Sample patterns
1	Bundle with software	Independent utilities distributed with other software	Bundled with
2	Components of software	Implemented as components of other legitimate software	Component of, as part of
3	Drive-by download	Visit web sites or download pages containing grayware	Drive by
4	Dropped by malware	Spread by other grayware such as downloaders	Dropped by
5	Vulnerability exploits	Exploits vulnerabilities such as buffer overflow	Vulnerability, loophole
6	Instant messengers	Install via Instant messengers (IM) such as AOL and MSN	Instant messaging
7	Email and attachments	Embedded in emails and attachments to trick recipients	Email
8	Manually installation	Directly download from web sites or through FTP services	Manually install
9	Peer-to-peer	Transport in P2P applications such as Gnutella and KaZaA	Peer-to-peer, p2p
10	Network file shares	Trusted network share folders as propagation channels	Network share

To evaluate the impact on categorization performance by parameter C_b that is used to train learning models, we construct a series of classifiers by varying parameter C_b in the range of [0, 300] and compute their classification accuracy, precision, recall, as well as F_1-measure. By controlling the balance between training errors and the margin of the separating hyperplane, parameter C_b eventually affects the classification performance as demonstrated in Figure 6. Apparently, the classification recall improves monotonically along with increasing parameter C_b, and the enhancement is significant when C_b changes in the range [1, 100]. In the same manner, the F_1-measure, which is the weighted harmonic mean of the precision and recall, mainly follows the trend of the recall and therefore witnesses noticeable improvement as well when C_b falls in the range of [1, 100]. As expected, a larger C_b imposes heavier penalty on any training error committed by models and forces the latter to make less categorization mistakes in order to minimize the objective function, consequently delivering better classification performance but at the cost of a smaller margin of separating hyperplane.

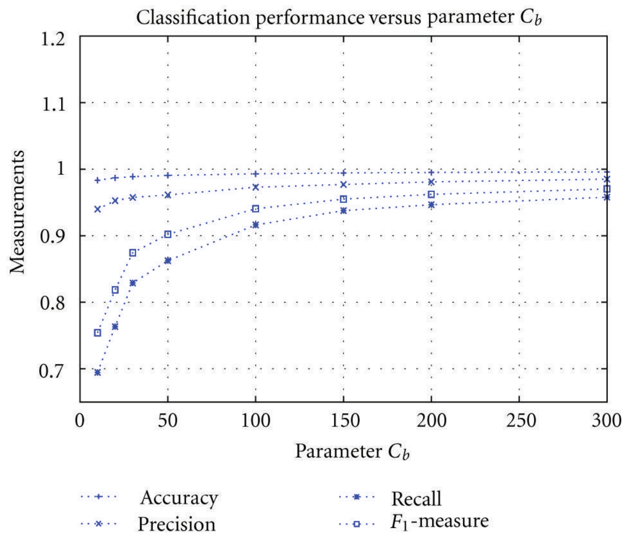

Figure 6: Performance measures of SVM model for Attack Avenue.

5. Grayware Payloads and Their Effects

The deleterious grayware activities are instantiated by the code transported into victim machines after the latter are successfully penetrated. As the behavior of the grayware code can only be constrained by the imaginations of its creators, the proposed framework makes no attempt to enumerate all possible grayware payloads, instead, it mainly focuses on grayware impact on confidentiality, integrity, and availability (CIA) of infected systems. The confidentiality of affected hosts is inevitably jeopardized by the grayware-committed information theft; while its integrity is destroyed when its security applications are incapacitated, and its availability is compromised after its resources such as CPU cycles and network bandwidths are usurped by intruders. In this Section, we define taxonomic features to delineate grayware characteristics manifested in *Activation* stage of its life cycle.

5.1. Information Exposure. Confidential information such as passwords, financial documents, or sensitive data stored on user systems is the source for grayware to generate profits. The proposed framework uses the taxonomic feature Information Exposure to describe grayware impact on information confidentiality. The feature consists of three categories: *Low, Medium*, and *High*, and is treated as single-label so that each grayware strain has only one label. The training data for the feature in question are constructed with the help of the Trend Micro grayware encyclopedia. For instance, spyware *TSPY_LINEAGE.GL* and adware *ADW_ALEXA.AK* depicted in Table 1 are assigned to categories *High* and *Medium*, respectively. The collected training data help us build a SOM with 10 × 10 hexagonal lattice, and the U-matrix of the resulting map is presented in Figure 7. It is evident that the map has strong clustering capability. In this regard, neurons with label *Medium* mainly locate at the middle-left of the map; while nodes of category *Low* occupy the upper- and lower-left corners, and the remaining cells are the territory of class *High*.

In the Trend Micro grayware encyclopedia, the adware family *ALEXA* has 158 members, each of which has its own unique name consisting of three parts: grayware type,

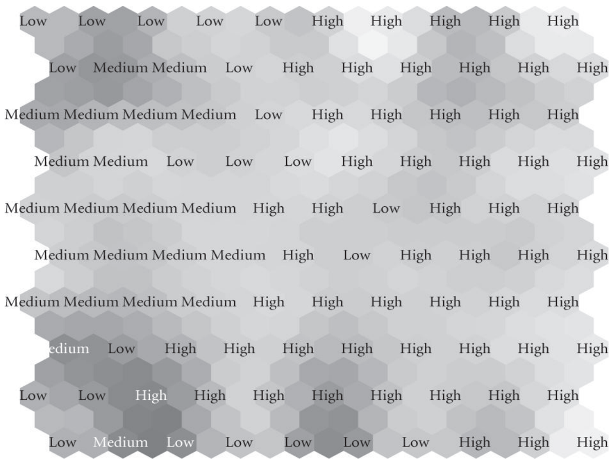

Figure 7: Distances between reference vectors of neighboring units on Information Exposure.

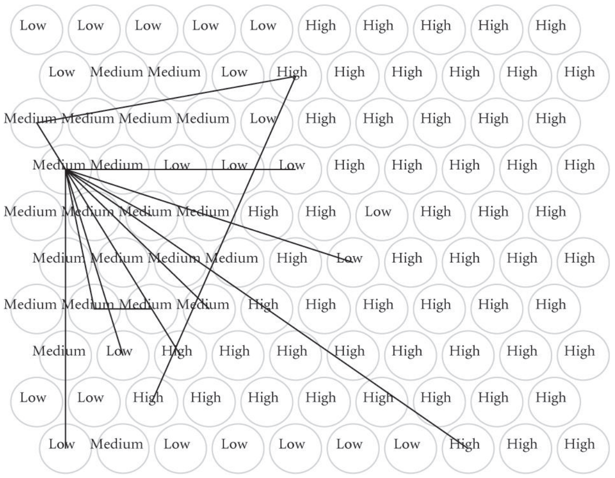
Low Low Low Low Low High High High High High
Low Medium Medium Low High High High High High High
Medium Medium Medium Medium Low High High High High High
Medium Medium Low Low Low High High High High High
Medium Medium Medium Medium High High Low High High High
Medium Medium Medium Medium High Low High High High High
Medium Medium Medium Medium High High High High High High
Medium Low High High High High High High High High
Low Low High High High High High High High High
Low Medium Low Low Low Low Low High High High

Figure 8: Trajectory of adware family *ADW_ALEXA* on Information Exposure.

family name, and suffix. The grayware type is typically an acronym for the genre described in Column *name* of Table 2. For instance, the identifier *ADW* is designated to adware, while *TSPY* is assigned to Trojan spyware. The family name identifies a group of grayware specimens that share the same code base and, therefore, have similar behavior. Members in the same family are differentiated by suffixes that are assigned in alphabetical or numerical order according to their discovery date. For example, supposed that adware family *ALEXA* has an instance *ADW_ALEXA.AA*, its two immediate descendants are then named as *ADW_ALEXA.AB* and *ADW_ALEXA.AC*, respectively. Obviously, the ages of grayware strains in the same family have close relationship with their names, and their chronological arrangement can be obtained by sorting their names in ascending alphabetical order. In this regard, the specimen *ADW_ALEXA.GS* is a descendant of *ADW_ALEXA.AK* shown in Table 1. By identifying best matching units (BMUs) in a SOM map for members of a specific grayware family and connecting them with lines according to their chronological orders, we obtain a curve termed genealogical trajectory, which describes the evolution of the grayware family in question with respect to feature Information Exposure. Figure 8 outlines the genealogical trajectory for family *ADW_ALEXA*.

It can be observed from Figure 8 that 158 *ALEXA* family members only occupy 14 out of 100 neurons in the map, indicating that many strains share the same BMUs. To this end, 145 *ALEXA* members select node (0, 3) as their BMU, while the remaining 13 specimens have their own unique cells. The fact that neuron (0, 3) bears label *Medium* for the feature Information Exposure clearly points out that most *ALEXA* family members impose medium threat on information confidentiality. In comparison, only 4 *ALEXA* strains including *ADW_ALEXA.BI* hit neurons with label *Low* and another 4 family members such as *ADW_ALEXA.A* reside on territory belonging to category *High*. Interestingly, *ALEXA* members labeled *High* are ancestors of *ADW_ALEXA.AK* while those with tag *Low* its descendants, implying its declining impact on information confidentiality. On the contrary, the genealogical trajectory for *TSPY_LINEAGE* family reveals that most of its members land on the area occupied by category *High* of feature Information Exposure, therefore imposing serious threat on the Internet ecosystems.

Each neuron in SOM has a weight vector with the same dimensionality as feature vectors of input data. In the SOM construction process, weight vectors of neurons are adjusted according to input samples so that characteristics of the latter could be preserved in certain cells of the map. It is therefore reasonable to expect that a dominant attribute in the feature space should still remain its significance in the SOM lattice with high probability. We define the plane map for the SOM's *i*th attribute, which is a token appeared in the Trend Micro repertoire, as the lattice formed by the *i*th component in the weight vector of every neuron. Figure 9 presents the plane map associated with token *steal*, here, the attribute value is converted into a gray scale of [0, 1]: the brighter the gray-scale is, the larger value the attribute assumes. Obviously, the brightest neuron locating at (8, 9) has gray scale 0.950, and it happens to be in category *High*. Similarly, its two neighbors at (9, 9) and (9, 8) also belong to class *High* and have gray scales 0.946 and 0.776, respectively. Furthermore, neurons with significant values for token *steal* form a tight community, an indicator for the SOM's clustering capability. Figure 9 clearly manifests that large values for token *steal* only associate with neurons with label *High*, rendering the attribute in question a unique differentiator for category *High*. In this context, spyware *TSPY_LINEAGE.GL* outlined in Table 1 can be considered with high confidence to put high risk on information confidentiality as its description contains token *steal*. On the contrary, plane maps for tokens *export* and *system* disqualify them to be dominant features as they appear in neurons with labels *Low* and *High* simultaneously. Clearly, the characteristics of attributes' plane maps, such as mean and variance, are helpful to feature selection. Our experiments show that the performance of model based on top-70% attributes with highest gray-scale variance is similar to that of model with full set of features. Therefore, top-70% tokens with highest gray-scale variance are the significant contributors to classification performance.

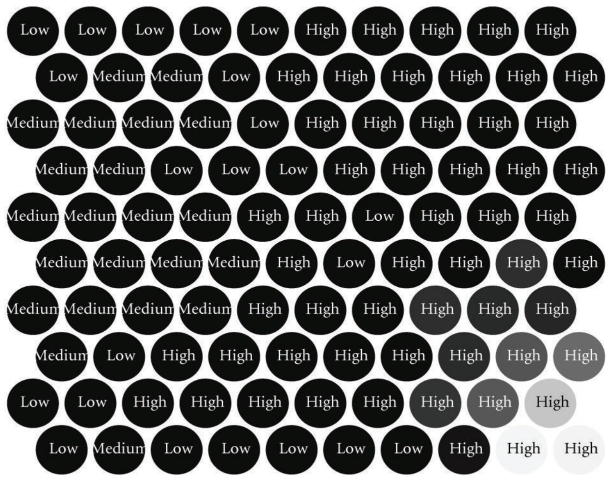
Low Low Low Low Low High High High High High
Low Medium Medium Low High High High High High High
Medium Medium Medium Medium Low High High High High High
Medium Medium Low Low Low High High High High High
Medium Medium Medium Medium High High Low High High High
Medium Medium Medium Medium High Low High High High High
Medium Medium Medium Medium High High High High High High
Medium Low High High High High High High High High
Low Low High High High High High High High High
Low Medium Low Low Low Low Low High High High

FIGURE 9: Plane map for token *steal* on Information Exposure.

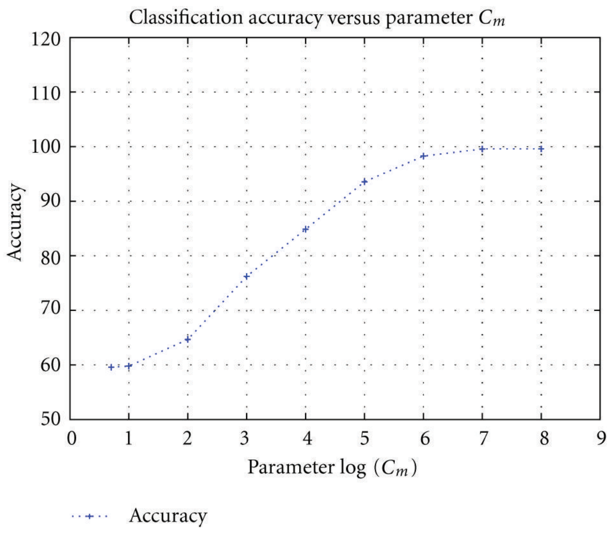

FIGURE 10: Relation between parameter C_m and accuracy on Integrity Impact.

5.2. Grayware Impact on Integrity. The feature Integrity Impact is designed to characterize the grayware effect on the integrity of infected systems. To hide its malicious activities from users, grayware typically alters configurations and lowers security settings on infected systems in an unauthorized manner, compromising the integrity of the latter. Some grayware species even aggressively disable or terminate security applications installed on end systems, essentially making grayware attacks undetected. Grayware is classified with respect to feature Integrity Impact into three categories, *Low, Medium*, and *High*. We gather training data from entries of the Trend Micro grayware encyclopedia having information on field *System Impact* and build an SVM learning model with the multiclass optimization method. The classification accuracy attained by the learner heavily depends on parameter $\log(C_m)$ and its relationship with $\log(C_m)$ is depicted in Figure 10. It is obvious that the classification accuracy improves monotonically with the increasing $\log(C_m)$ and reaches 90% when $\log(C_m) \geq 4.5$. Similar to C_b in binary SVM models, parameter C_m balances between training errors and margins of separating hyperplanes, and a large C_m imposes heavy punishment on training errors committed by SVM models, compelling the latter to reduce training errors. Compared to the effect of C_b on classification accuracies of binary learners for feature Attack Avenue shown in Figure 6, parameter C_m in the multiclass model for Integrity Impact demands a much larger magnitude to achieve a commensurable performance due to the complex objective function and constraints it controls.

With the same training data, we build a hexagonal SOM map with a 10 × 10 neuron lattice for feature Integrity Impact, and present its U-matrix in Figure 11. Apparently, neurons with the same labels tend to cluster together in the map: nodes for class *Medium* mainly reside at the upper edge of the map, the territory of category *Low* is at the upper-left corner and the middle of the grid, while the remaining estate belongs to category *High*. The relatively dark color for the region occupied by category *Low* reveals that the average distance between its neurons are much larger than that for classes *Medium* and *High*. The clique formed by cells with label *Medium* clearly points out that members of the category *Medium* share similar features. In comparison, species in class *High* scatter in a vast territory of the SOM map, manifesting their diversified features. By performing analysis on plane maps for feature Integrity Impact, we can find that tokens such as *sensitive* and *stolen* are strong differentiators for feature in question.

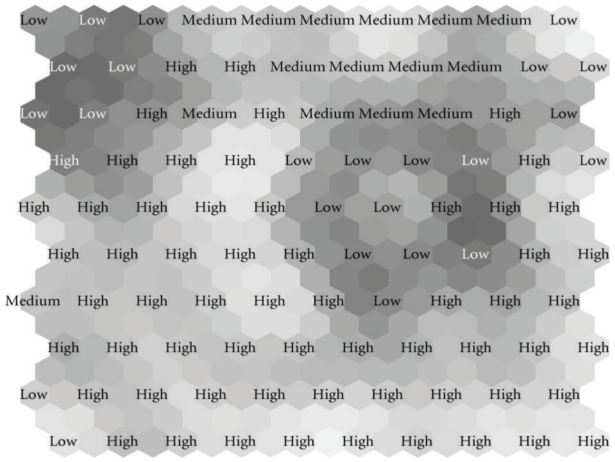
Low Low Low Medium Medium Medium Medium Medium Medium Low
Low Low High High Medium Medium Medium Medium Low Low
Low Low High Medium High Medium Medium Medium High Low
High High High High Low Low Low Low High Low
High High High High High Low Low High High High
High High High High High Low Low Low High High
Medium High High High High High Low High High High
High High High High High High High High High High
Low High High High High High High High High High
Low High High High High High High High High High

FIGURE 11: The U-matrix for the feature Integrity Impact.

To analyze the effect on the classification performance by the size of feature set selected by information gain, we construct different sets of attributes which contain 60%, 70%, 80%, and 100%, respectively, of the features with the highest information gains and build learners based on these attribute sets. The performance of the resulting models is depicted in Figure 12. It is clear that the relationship between the size of attribute set and categorization performance is monotonic, that is, a larger set of features leads to a better classifier. However, the marginal difference in performance

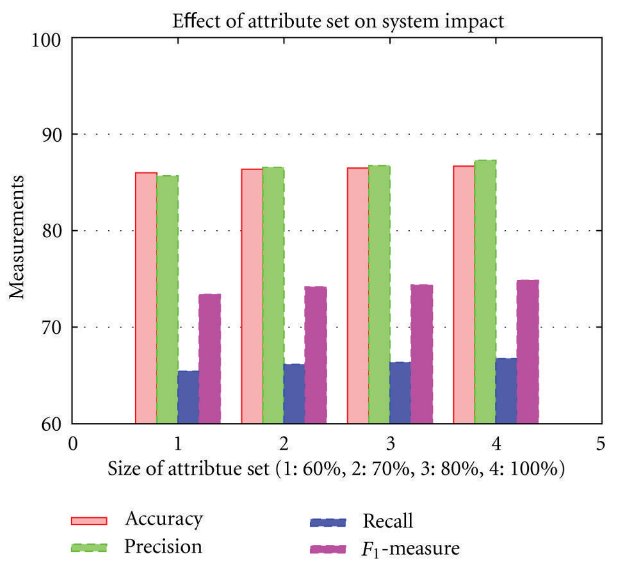

Figure 12: Relation between attribute set and categorization performance for Integrity Impact.

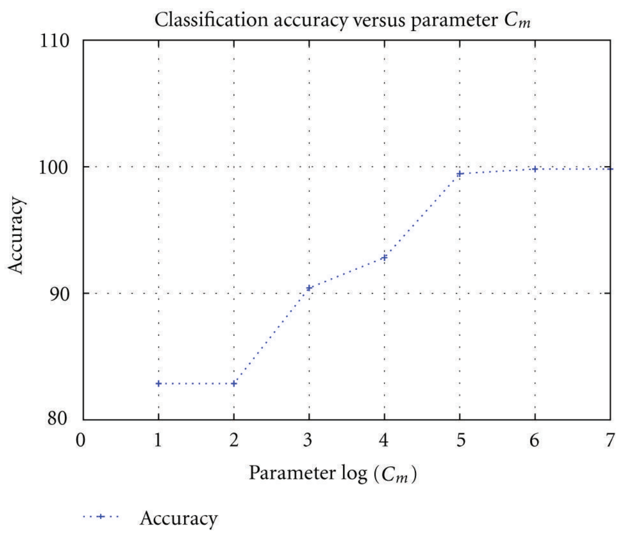

Figure 13: Relation between parameter C_m and accuracy on Destructiveness.

between the learner with top-70% attributes and the model with full feature set also indicates that attributes with low information gain actually contribute little to the classification performance.

5.3. The Destructiveness of Grayware. The direct damage on victim systems' availability inflicted by grayware is measured with taxonomic feature Destructiveness in the proposed framework. A grayware is considered as destructive if it corrupts victim's file system or even formats the entire hard drive; it is also destructive when grayware excessively consumes target system's resources affecting the stability and productivity of the latter. By using infected machines as launch pads for denial of service (DoS) attacks against other hosts, grayware further affects the availability of DoS-targeted systems as well. The single-label feature Destructiveness defined in the framework consists of two categories: *Yes* and *No*. The training data for the feature are collected automatically from the Trend Micro grayware encyclopedia. For instance, spyware *TSPY_LINEAGE.GL* and *ADW_ALEXA.AK* described in Table 1 are samples for categories *Yes* and *No*, respectively. By using the *multiclass-optimization* method, we build a sequence of classifiers for the feature with varying parameter $\log(C_m)$ in the range of [1, 7] and present their categorization accuracies in Figure 13.

The staircase-like curve in Figure 13 clearly indicates the nonlinear relationship between classification accuracy and $\log(C_m)$. The improvement on categorization accuracy is marginal when $\log(C_m)$ changes in the range [1, 2]; however, it jumps from 82.87% to 90.42% by increasing $\log(C_m)$ from 2 to 3. After encountering a relatively insensitive $\log(C_m)$ range in [3, 4], the classification accuracy once again boosts significantly from 92.82% to 99.45% if $\log(C_m)$ advances from 4 to 5. The classification accuracy saturates as parameter $\log(C_m) > 5$. To evaluate the impact on grayware categorization by parameter $\log(C_m)$, we classify species in the grayware repertoire with the series of learners obtained above and compute grayware distributions on classes *Yes* and *No* of feature Destructiveness. Our analysis points out that 0.19% grayware species are tagged as *Yes* for the feature in question when parameter $\log(C_m) = 3$; and it becomes 0.34% and 0.95% when parameter $\log(C_m)$ is 4 and 5, respectively. Therefore, grayware categorization is only loosely sensitive to parameter $\log(C_m)$.

To evaluate the performance by using the model trained based on data extracted from Trend Micro to classify entries in other grayware encyclopedias, we randomly select 1000 grayware species collected in the Symantec encyclopedia to be the test set. As taxonomic feature Destructiveness is not used in the Symantec encyclopedia, instead, a feature termed Risk Impact is defined, which has 5 levels: very low, low, moderate, high, and very high. We tag an entry in the test set as destructive if it is high or above in its feature Risk Impact, or nondestructive otherwise. For comparison, we also randomly pick 1000 entries from Trend Micro to obtain another test set, and with both test sets in hand, we evaluate the performance of the learner presented in Figure 13 with $\log(C_m) = 4.0$, which is described in Figure 14. Evidently, the learner performs a little better on the test set collected from Trend Micro and achieves a much higher recall. The fact that the model delivers satisfactory performance on Symantec encyclopedia seems to imply that similar tokens are used to delineate the same grayware in different encyclopedias.

5.4. Payloads Carried by Grayware. The grayware effect on confidentiality, integrity, and availability (CIA) of infected systems has been measured with the help of taxonomic features Information Exposure, Integrity Impact, and Destructiveness described previously. Another dimension to characterize grayware activities relevant to CIA is to identify key payload types carried by species even though it is unrealistic

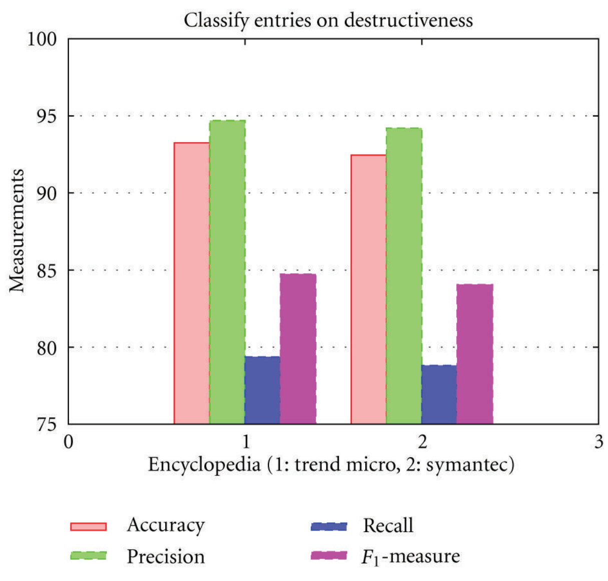

FIGURE 14: Using learner from Trend Micro to classify entries in Symantec on Destructiveness.

to enumerate all possible payloads. Our analysis on the Trend Micro grayware encyclopedia indicates that many grayware species deliver advertisements and other popups to end systems, while some strains routinely modify system configurations or security settings on infected machines to avoid detection. In addition, the excessive amount of processes and network connections spawned by grayware on victim systems not only consumes precious resources, but also degrades user productivity. The surreptitious installation of multiple grayware species on a victim host by different intruders in a greedy and uncoordinated manner further deteriorates the performance of the targeted host. The proposed framework therefore introduces taxonomic feature Carried Payload to classify grayware payloads into categories depicted in Table 5. The feature Carried Payload is treated as multivariate as grayware usually possesses multiple payloads simultaneously.

We resort to pattern matching techniques in generating training data for feature Carried Payload: we first extract entries from the Trend Micro grayware encyclopedia which have information on the field *Payload*, then search the field in question for any patterns specified in Column *Keyword* of Table 5 to assign entries into corresponding categories. For instance, spyware *TSPY_LINEAGE.GL* of Table 1 performs three different activities, and its first task—move system files to other folders—causes it to be put into category *File Manipulation* due to keyword *file*, while its other two tasks lead it to classes *Terminate Process* and *Information Theft* as well. In the same manner, adware *ADW_ALEXA.AK* depicted in Table 1 is assigned to class *Hijack Session* as it redirects search queries via usurping user connections. The above-described pattern matching process allows us to form a set of training data consisting of 2,418 samples, among which 400 belong to the category *Download Software*, while 350 and 300 are from classes *File Manipulation* and *Terminate Process*, respectively. With the help of the *multiclass-to-binary reduction* method, we obtain a series of SVM learning models obtained by varying parameter C_b in [10, 500], and every learner comprises 10 binary classifiers, each of which identifies a category shown in Table 5.

The categorization performance of the learners obtained is shown in Figure 15. When parameter $C_b = 10$, the model attains average classification accuracy, precision, recall, and F_1-measure 92.22%, 93.47%, 47.09%, and 0.58, respectively. The recall and F_1-measure can be enhanced dramatically with incremental parameter C_b. More specifically, by changing C_b from 10 to 100, the recall increases from 47.09% to 73.09% and the corresponding F_1-measure jumps from 0.58 to 0.82; the recall and F_1-measure continue to improve significantly by further adjusting C_b from 100 to 300, but saturates after C_b is beyond 300. In comparison, the classification accuracy and precision remain above 92.00% within the entire C_b spectrum, in this regard, they are 98.95% and 98.20%, respectively, when $C_b = 500$. The grayware distribution generated by the learner with $C_b = 500$ is depicted in Figure 16. Apparently, the largest category *Popup Advertisements* (with ID = 3 in Figure 16) accounts for 51.39% of the entire grayware population. The contribution from the next two most populated classes *File Manipulation* and *Network Connection* (IDs = 10 and 9) is 21.12% and 19.15%, respectively. It can be concluded that a vast majority of grayware manipulates victims' file systems to store collected user profiles, sends them back to attackers via network connections, and fetches targeted advertisements to display on infected systems.

6. Extend Life Span by Eluding Detection

Profits generated by grayware heavily depend on the time span of its *discovery* stage, motivating it to extend its life expectancy at all costs. To avoid detection so that financial gains can be maximized, grayware typically employs cryptographic techniques to encrypt its communication channels and collected information. Grayware can also increase the difficulty of being identified by security applications through compressing its executables and data generated. By residing at main memory of infected systems and injecting itself into legitimate processes, grayware essentially lives a parasite life and becomes invisible to routine administration utilities, further expanding its life time. The diversified grayware behavior in the *Discovery* stage of its life cycle is characterized in this Section.

6.1. Information Encryption. One way for grayware to defeat pattern-based anti-grayware products is to scramble its files and network communications with cryptographic techniques as it is nearly impossible for security devices to transform ciphertexts into plaintexts to make pattern match feasible without the knowledge of encryption algorithms and keys involves. The proposed framework dedicates a feature Information Encryption to characterize grayware cryptographic behavior, and it contains two categories: *Yes* for species employing encryption methods and *No* for

Table 5: Malicious activities carried out by grayware.

ID	Payload	Description	Keyword
1	Attack security software	Lower security level, disable security applications	Antivirus, firewall, security
2	Hijack Session	Intercept connections or communication channels	Hijack, affiliate, redirect
3	Popup advertisements	Show ads out of contexts or overlap others	Pop-up, pop-under
4	Information theft	Collect sensitive data and keystrokes, send to attackers	Passwords, information
5	Configuration change	Modify homepage, preference, bookmarks, registry	Folder, registry, config
6	Arbitrary commands	Execute arbitrary programs by attackers	Arbitrary code, execute, run
7	Download software	Act as downloaders or droppers for extra programs	Download, drop
8	Terminate process	Kill system daemons or network applications	Terminate, kill, stop
9	Network connection	Open network connections to grant attacker full control	Connect, proxy
10	File manipulation	Add, modify, move, or delete system/data files	File, overwrite, load, move

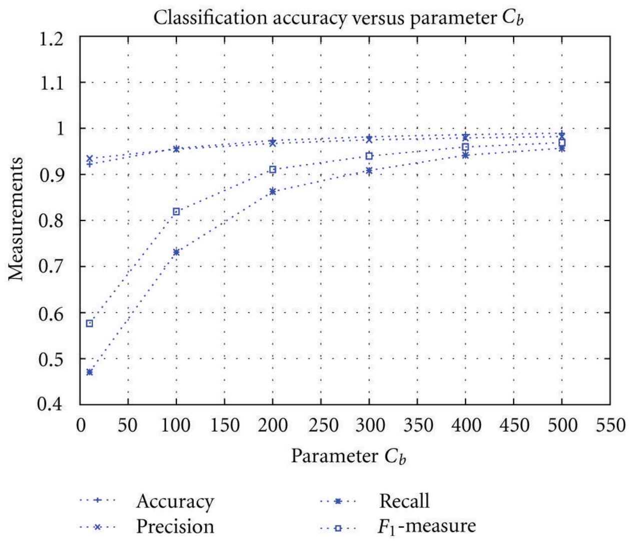

Figure 15: Relation between parameter C_b and accuracy on Carried Payload.

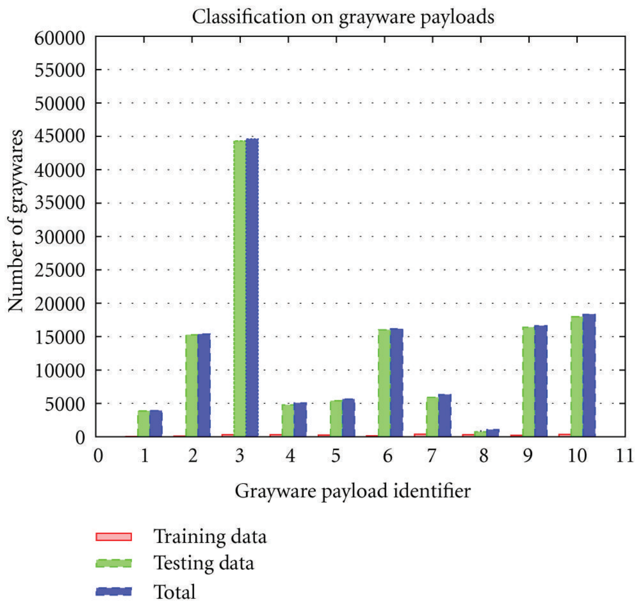

Figure 16: Classification on Carried Payload when $C_b = 500$.

others. For instance, both the spyware *TSPY_LINEAGE.GL* and adware *ADW_ALEXA.AK* depicted in Table 1 transport and store their files in plaintext; while *TSPY_LINEAGE.BZY* camouflages its data in an encrypted form. The training data for the feature Information Encryption are gathered by extracting entries from the Trend Micro grayware encyclopedia having information on the field *Encrypted*.

With the training data at hand, we evaluate the impact on the grayware classification performance by the lattice sizes of SOMs. For brevity, we only consider hexagonal maps assuming dimensionality of $x \times y$ with $x = y \in [5, 10]$ and present the categorization accuracy, precision, recall, and F_1-measure of the corresponding SOMs in Figure 17. Generally speaking, the classification accuracy can be improved steadily by enlarging the SOM grid size. In this regard, the categorization accuracy boosts to 80.62% from 74.16% by changing the lattice dimensionality from 5×5 to 7×7, and it further advances to 83.97% for the 10×10 SOM grid. In the same manner, the F_1-measure also increases monotonically along with the augmenting dimensions of SOM maps. To this end, the F_1-measure can be lifted from 0.63 to 0.72 when the 5×5 map is replaced with the 7×7 lattice, and it reaches 0.78 if a 10×10 SOM grid is used instead.

The U-matrix of the 10×10 hexagonal SOM lattice outlined in Figure 18 demonstrates the map's clustering capability. A SOM with a large grid size may result in unlabeled neurons as manifested by cells at (5, 1) and (8, 1). On the other hand, a large map estate indeed provides more flexibility for weight vectors adjustment and refinement leading to a better grouping power. In this context, the range formed by the lowest and highest gray scales of neurons occupied by class *No* in the 10×10 map is [27, 94], while it is a much narrower spectrum [30, 72] for the 5×5 grid. Compared to the highest categorization accuracy 83.97% by SOMs, the SVM learner built on the same training data with the multiclass-optimization method offers classification accuracy 90.00%.

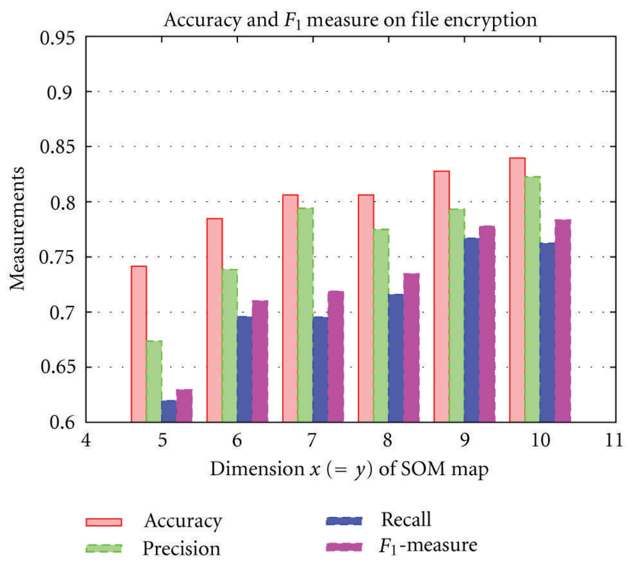

FIGURE 17: Performance on Information Encryption by SOMs.

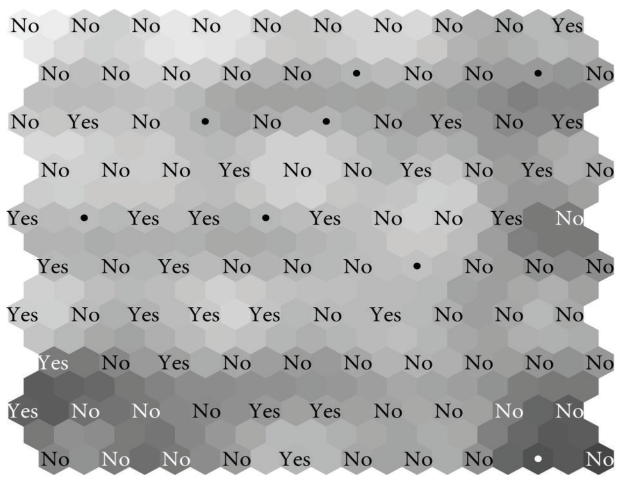
No No No No No No No No No Yes
No No No No No No No No
No Yes No No No Yes No Yes
No No No Yes No No Yes No Yes No
Yes Yes Yes Yes No No Yes No
Yes No Yes No No No No No No
Yes No Yes Yes Yes No Yes No No No
Yes No Yes No No No No No No No
Yes No No No Yes Yes No No No No
No No No No Yes No No No No

FIGURE 18: U-matrix for feature Information Encryption.

6.2. File Compression. Executable file compression performed by grayware not only reduces its storage footprint and consequently diminishes its exposure to scrutiny, but also obfuscates its file content helping deter reverse engineering and elude detection by pattern-based anti-grayware devices. To achieve the same effect as uncompressed executables, grayware usually transforms its executable into a self-extracting archive consisting of the compressed file and a piece of decompression code that unpacks the file into its original form and transfers control to it on the fly in execution. Although contemporary security devices are capable of unpacking files that are compressed in a variety of publicly known algorithms for grayware detection, they are still ineffective when it comes to software packed with proprietary compression methods.

In the proposed framework, we design a feature File Compression to characterize the grayware behavior on packing executables. As it is nearly impossible to enumerate all potential compression algorithms, especially when taking proprietary packing mechanisms into considerations, we only focus on the most grayware-favorite compression methods listed in Table 6. The training data for the feature File Compression are constructed by retrieving entries from the Trend Micro grayware encyclopedia. For instance, spyware *TSPY_LINEAGE.GL* described in Table 1 compacts its files in UPX format before shipping to affected systems, and it is therefore marked as a positive sample for category *UPX*. In contrast, adware *ADW_ALEXA.AK* transfers its files in an uncompressed fashion excluding it from the training data. By using the *multiclass-to-binary reduction* method with parameter $C_b = 10$, we build an SVM model consisting of 8 binary classifiers, each of which identifies a category of the feature File Compression. The performance of the learner obtained can be derived with the cross-validation procedure as 90.36% for categorization accuracy, and 96.33%, 19.09%, and 29.00% for precision, recall, and F_1-measure, respectively.

The extremely low recall and F_1-measure of the above-described learner leads us to improve its performance by adjusting parameter C_b. By sweeping C_b in the range of [10, 500], we observe that the recall can be improved dramatically from 19.09% to 57.94% when C_b is changed from 10 to 100, and it further increases to 94.00% with $C_b = 500$. In the same manner, the F_1-measure also reaches 96.17% when C_b is set to 500. By analyzing the grayware distribution for feature File Compression generated by the learner with $C_b = 500$, we observe that about half of grayware species resort to *Petite* compression method, and *Aspack* and *UPX* are also heavily used by grayware. In comparison, compression approaches *Upack*, *PECompact*, and *FSG* are only occasionally employed by grayware.

To evaluate the impact on categorization performance by SVM kernel functions, we train learning models with the following types of kernels, and present their performance measures in Figure 19. (a) Polynomial kernels in the form of $(\overline{x}_i \cdot \overline{x}_j + 1)^d$ with variable exponent d and for brevity, we only describe results with $d = 1$ or 2 (termed poly-1 and poly-2 in Figure 19). (b) Radial basis functions (RBFs) expressed as $\exp(-\gamma \|\overline{x}_i - \overline{x}_j\|^2)$ with various γ, and Figure 19 only shows the result for $\gamma = 1$ or 2. (c) Sigmoid functions in the format of $\tanh(\overline{x}_i \cdot \overline{x}_j + c)$ with tunable parameter c and the results are for $c = 0.5$ or 1 (denoted as sigmoid-1 and sigmoid-2 in Figure 19). It can be observed from Figure 19 that sigmoid functions deliver the worst categorization performance, while RBFs marginally outperform other kernel types even though the difference is insignificant. Compared to linear kernels, learning models with other kernel types such as RBFs demand much more CPU cycles to train; therefore, the proposed framework employs linear kernel functions by default in its learning model generation.

6.3. Memory Residency. It is desirable for grayware to reside at main memory of infected systems in order to continuously track user activities, inject itself into active processes,

Table 6: File compression types by grayware in encyclopedia of Trend Micro.

ID	Compression	Description
1	Aspack	A Win32 executable compressor capable of reducing file size and resisting reverse engineering
2	FSG	A file compressor especially suitable for small *EXE* or *ASM* files.
3	PECompact	A compressor for code, data, and import/export table with proprietary compression algorithm.
4	Petite	A utility to compress and encrypt files, automatically expand files in memory when execution.
5	SFX	The Self extractor (SFX) compresses a file and transports it to a remote system where decompression is performed.
6	UPX	Ultimate Packer for eXecutables is an open-source packer performing in-place decompression.
7	Upack	Upack is a file packer based on LZMA compression.
8	Other	There are many other file compression packages such as Neolite, Nullsoft, PEPack, or RAR.

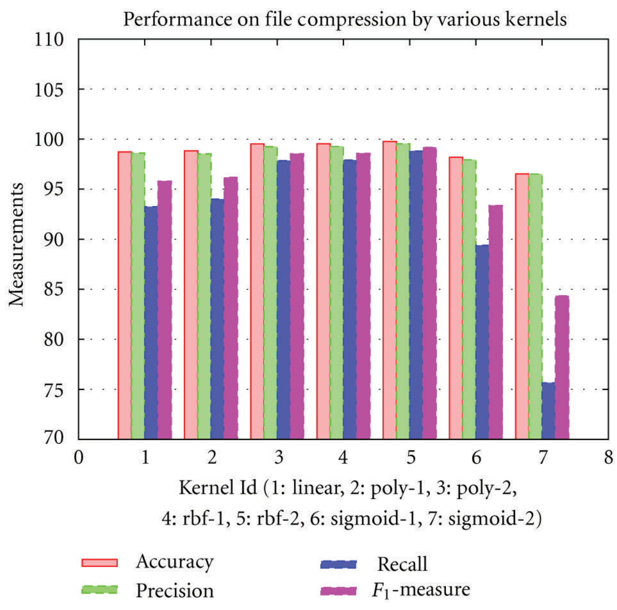

Figure 19: Performance on feature Compression Type by different kernels.

and hide from administration utilities. Another benefit of memory residency is that files associated to running grayware programs cannot be modified or deleted unless they are first removed from main memory. We designate feature Memory Resident in the proposed framework to characterize the property of grayware memory residency. Two categories are defined for the feature in question: *Yes* for species capable of staying at memory, while *No* for others. For instance, both spyware *TSPY_LINEAGE.GL* and adware *ADW_ALEXA.AK* depicted in Table 1 are memory resident strains, while hacking tool *HKTL_HIDEOUT.A* and cracker *CRCK_REALVNC.A* are not.

The training data for the feature Memory Resident are automatically generated from the Trend Micro grayware encyclopedia. With the training data at hand, we construct four learning models named *bubble-SOM*, *inverse-SOM*, *gaussian-SOM*, and *linear-SVM*. The *bubble-SOM* and *inverse-SOM* models are built by using SOM techniques with neighborhood function *bubble* but different learning rates: the rate linearly decreases with time for the former while inversely changes with respect to time in the latter; meanwhile the *gaussian-SOM* model is obtained with neighborhood function *gaussian* and linear learning rate. On the other hand, the *linear-SVM* is an SVM-based learner generated with a linear kernel by the *multiclass-to-binary reduction* method. The classification performance in terms of categorization accuracy, precision, recall, and F_1-measure of the four models is depicted in Figure 20. With classification accuracy 83.78%, the *linear-SVM* learner significantly outperforms its SOM counterparts as the best SOM model can only achieve 73.84%. Among the three SOM-based models, the *inverse-SOM* delivers the worst performance, while *bubble-SOM* and *gaussian-SOM* are comparable in classification accuracy and precision. The grayware categorization on feature Memory Resident reveals that 56.85% species are memory resident.

7. Trends on Grayware Characteristics and Risk

The proposed framework can also be used to evaluate grayware threats and shed light on grayware evolution leading to more effective prevention strategies and site-specific defense policies.

7.1. Grayware with Compact Footprints. The storage footprint of a grayware strain characterized with the feature File Size in the proposed framework clearly affects its functionality and installation methods. The transportation of a grayware specimen to infected hosts is inevitably slowed down if it assumes a large footprint. The grayware footmark is also constrained when it penetrates targets by taking advantage of security vulnerabilities such as buffer overflows as the latter necessitate specific size of input data for successful exploitations. Furthermore, a sizeable grayware consumes a large amount of storage space in victim systems and leaves a visible trace leading to its detection. On the other hand, a small grayware footprint does affect its payloads carried and consequently its functionalities. Therefore, it is expected that grayware creators would put much effort on optimizing their products to balance between storage consumption and functionality.

The sophisticated grayware behavior and its customized functionalities complicates the definition of taxonomic feature File Size. First, grayware could tailor its footprint

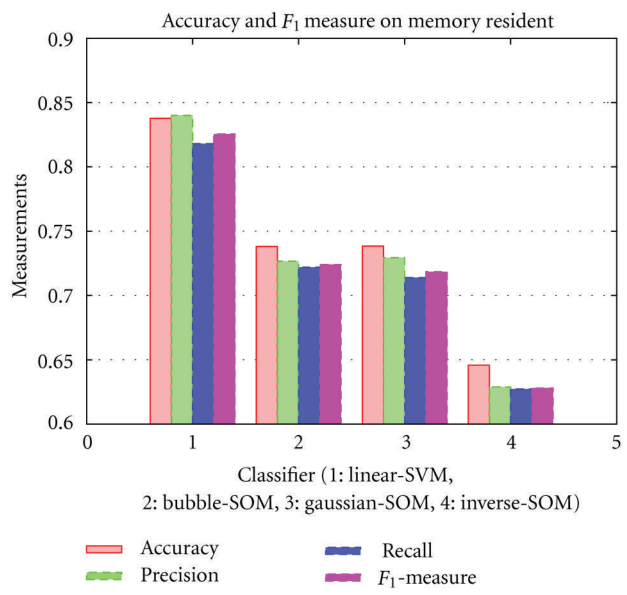

Figure 20: Classification performance on Memory Resident by different learners.

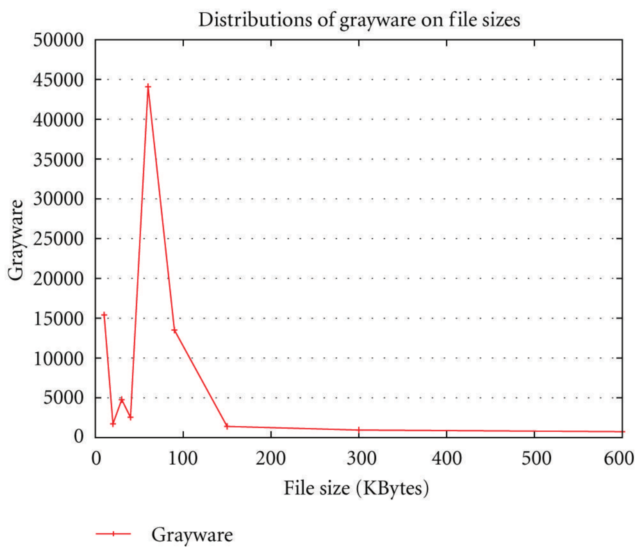

Figure 21: Grayware population with various file sizes.

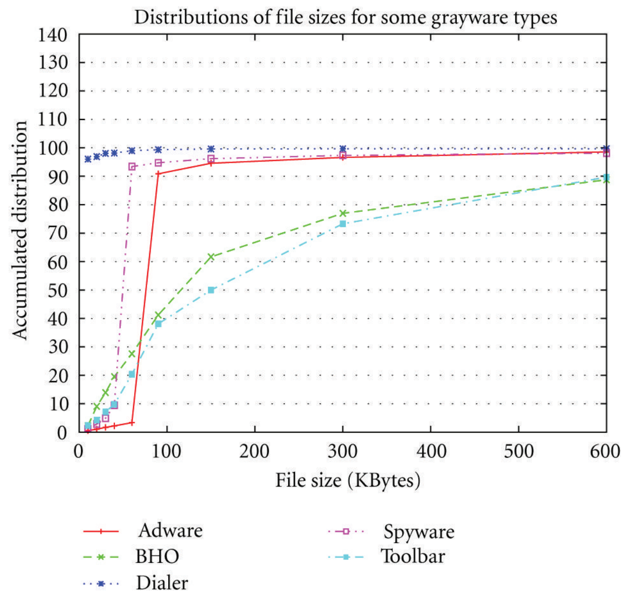

Figure 22: Accumulated grayware distribution on Grayware Type and File Size.

according to dynamics of affected systems such as OSs and network bandwidth. Next, grayware is also capable of transferring files in an accumulated manner, in addition, periodical updates on files can change file sizes and extra data/files may be downloaded on demand. Finally, it is a common practice for grayware to compress its files to facilitate transportation and storage. We therefore select the smallest uncompressed footprint for a grayware if it manifests diversified behavior on the feature File Size.

We specify ten categories for feature File Size: 10, 20, 30, 40, 60, 90, 150, 300, 600, and >600 (in KBytes). The training data are constructed by extracting entries of the Trend Micro grayware encyclopedia that have information in field *File Size* and conversing their file sizes into the buckets. AN SVM model is then built and used to classify the grayware repertoire, resulting in the grayware distribution depicted in Figure 21. Evidently, about 50.77% of the grayware population specimens fall into the bucket with 60 KBytes, making it the most populated category, while buckets with 10 and 90 KBytes contribute 17.74% and 15.56%, respectively, to the grayware universe. By considering storage consumption less than 100 KBytes to be footprint compact, we can easily derive that 94.49% of grayware species are compact in their footprints.

To investigate the relationship between grayware genre and storage footprint, we classify grayware according to feature Grayware Type and then categorize each group with respect to feature File Size, and present part of the classification results in Figure 22. The similar accumulated distributions for grayware types *Spyware* and *Adware* result in 94.76% and 90.86% of spyware and adware strains to be compact in their storage footprints (i.e., <100 KBytes). In the same manner, grayware *Toolbar* and *Browser Helper Object (BHO)* share similar accumulated distributions; however, they are quite different from *Spyware* and *Adware*. More specifically, only 38.10% and 41.21% of *Toolbar* and *BHO* are footprint compact; while the remaining species bear large storage footprints. Mainly functioning as plugins for web browsers, *Toolbar* and *BHO* are confined by the programming paradigms imposed by the host applications such as *Internet Explorer*.

To further get insight into the trend on grayware storage footprint, we classify grayware with respect to features File Size and Discovery Date and obtain the categorization results of Figure 23. Here, we aggregate together grayware species discovered prior 2004 due to their sparse samples. The evolution that grayware tends to shrink its storage footprint chronologically can be easily observed. To this end, 74.03%

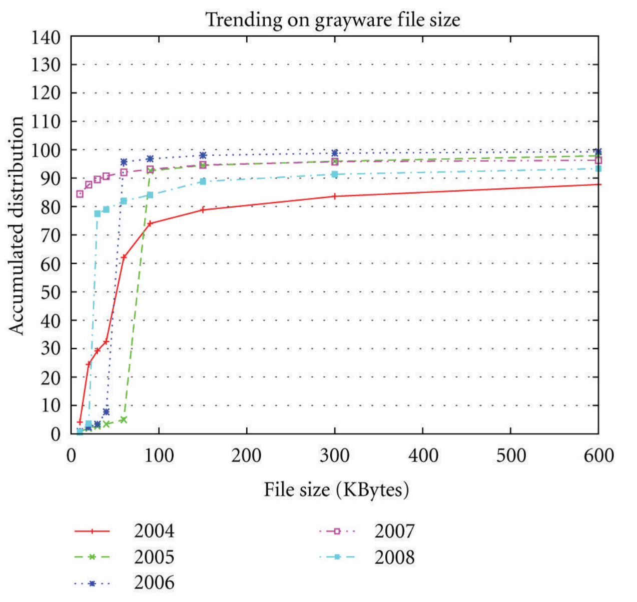

Figure 23: Accumulated grayware distribution on Discovery Date and File Size.

of grayware species created before 2004 are compact in their storage footprints, it increases to 92.56% for 2005 and further advances to 96.79% and 93.10% in 2006 and 2007, respectively. The heavy-tailed distribution of storage footprint for 2004-born strains is also evident in Figure 23 as 12.24% of its members fall outside the range of [0, 600] KBytes. In contrast, only 2.13%, 0.70%, and 3.70% of grayware species discovered in 2005, 2006, and 2007 consume storage larger than 600 KBytes.

7.2. Diversified Attack Avenues and Multiple Payloads. A grayware typically carries a variety of payloads in order to perform multiple activities including those listed in Column *Payload* of Table 7. Meanwhile, grayware payloads also develop and evolve along with time as demonstrated by the categorization with respect to features Carried Payload and Discovery Date shown in Table 7. Evidently, 181 species, which are 33.21% of the 545 strains discovered prior 2004, mainly manipulate files on affected systems, while 83 modify system configurations. Similarly, a vast majority of 2005-born grayware focus on file manipulation; however, an increasing number of grayware strains are designed to steal sensitive information. Advertisement delivery and software download are the dominant payload types of species detected in 2006. The three key payloads contained by specimens discovered in 2007, *Network Connection, Arbitrary Commands*, and *Hijack Session*, are clearly inseparable for any full-fledged grayware; attackers-initiated commands are executed on hijacked user sessions to collect sensitive information, which is sent back to attackers via network connections.

The classification in Table 7 also manifests that payload type *Attack Security Software* is an important weapon in grayware's arsenal used to fight against anti-grayware products and subsequently extend its life span. Compared to the four 2004-born grayware strains that detect and disable security protection applications on affected systems; the number of species with the payload in question reaches 60 and 458, respectively, in 2005 and 2006; it further advances to 3,358 in 2008 after decreasing to 33 in 2007. There is no doubt that improved detection capabilities of anti-grayware products definitely weaken the effectiveness of grayware attacks and force the latter to discard its ineffective payloads temporarily until new and effectual attack mechanisms are invented. As the battle between grayware and anti-grayware progresses, the cycle formed by the growth and decline of the relative strength of the two sides will continue. Similar observations can also be applied to other payload types, for instance, the favorite payload type *File Manipulation* in 2004 and 2005 is overshadowed by others in 2006 and 2007 but resurges as a major player in 2008.

The summation of grayware species presented in rows and columns of Table 7 is 133,114, much larger than the population of the Trend Micro repertoire (i.e., 86,834). Therefore, some grayware strains in fact carry multiple payloads simultaneously. Our grayware classification according to the number of carried payloads reveals that 73.96% of the population possess only one payload; while 3.52% and 18.02% encapsulate two and three payloads, respectively. Although it is rare for a grayware to carry more than six payloads, 23 grayware strains are still found to pack six payloads within their footprints. The evolution on payloads carried by grayware can be further analyzed by classifying grayware with respect to features Carried Payload and Discovery Date depicted in Figure 24. Most grayware species created in 2004–2006 are single-payload carriers (i.e., singleton); more specifically, 65.67% of 2004-born strains carry only one payload, and it is 92.48% and 95.46% for 2005 and 2006, respectively. In contrast, only 12.26% and 27.47% of grayware detected in 2007 and 2008 are singletons, while 83.67% 2007-discovered species carry three payloads and 64.49% 2008-created strains contain four payloads. By taking into account the fact that grayware tends to reduce its footprint in exchange for propagation speed and penetration rate, we clearly perceive grayware achievement to pack more payloads within the ever reduced footprint, indicating the formation of professional grayware development process.

The evolving grayware attack avenues can be analyzed with the help of the categorization according to features Discovery Date and Attack Avenue shown in Figure 25. The curve "*Attack Avenues* = 2" depicts the distribution of grayware that penetrates target systems by utilizing two attack mechanisms. Apparently, its peak clearly points out that most two-attack-channel graywares are created in 2005. In the same manner, the curve "*Attack Avenues* = 3" forms its pinnacle in 2006, while spikes of curves "*Attack Avenues* = 4" and "*Attack Avenues* = 5" coincide in 2007. The observation that peaks of the above-described curves shift chronologically signifies that newly-created grayware prefers to penetrate targets with multiple installation mechanisms. Again, the combined objectives of reduced footprint, multi-payloads, and diversified attack avenues attained by grayware can indeed act as a strong indicator for the maturity of the grayware industry.

Table 7: Grayware classification based on payload and discovery date.

#	Payload	2004	2005	2006	2007	2008
1	Attack security software	4 (0.005)	60 (0.069)	458 (0.527)	33 (0.038)	3358 (3.867)
2	Hijack Session	39 (0.045)	183 (0.211)	397 (0.457)	14660 (16.883)	86 (0.099)
3	Popup advertisements	58 (0.067)	594 (0.684)	43219 (49.772)	611 (0.704)	141 (0.162)
4	Information theft	49 (0.056)	821 (0.946)	621 (0.715)	152 (0.175)	3429 (3.949)
5	Configuration Change	83 (0.096)	362 (0.417)	1033 (1.190)	602 (0.693)	3578 (4.121)
6	Arbitrary commands	58 (0.067)	404 (0.465)	758 (0.873)	14789 (17.031)	151 (0.174)
7	Download software	37 (0.043)	350 (0.403)	3532 (4.068)	1603 (1.846)	794 (0.914)
8	Terminate process	10 (0.012)	166 (0.191)	597 (0.688)	165 (0.190)	105 (0.121)
9	Network connection	26 (0.030)	313 (0.361)	1307 (1.505)	14829 (17.077)	150 (0.173)
10	File manipulation	181 (0.208)	12806 (14.748)	1102 (1.269)	241 (0.278)	4009 (4.617)

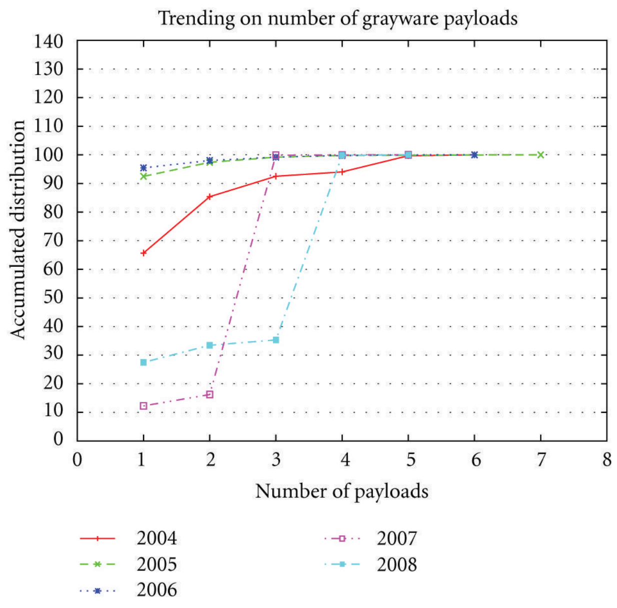

Figure 24: Accumulated distribution of grayware with respect to number of payloads.

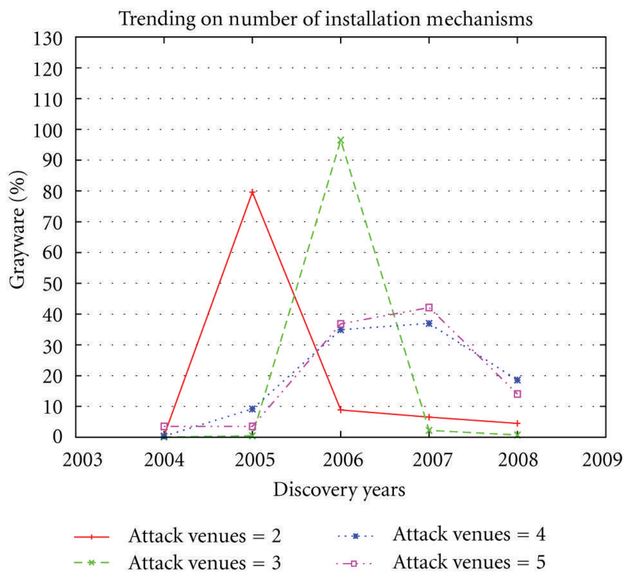

Figure 25: Number of installation venues employed by grayware.

7.3. Strong Clotting Capability. By infiltrating deeply into victim systems and intertwining tightly with other applications, grayware could increase the difficulty of being removed by security products and therefore expand its life span. One way for grayware to resist elimination is to create multiple files in infected systems and make them work in a coordinated manner so that any file removal could trigger the self-healing process: survival files automatically reinstall any missing files. Clearly, the self-healing effect can be enhanced by increasing the number of files involved; however, a large set of created files do leave more visible traces on affected systems, exposing grayware to detection. The grayware evolution on the number of created files can be evaluated based on the classification with respect to features Discovery Date and Created Files outlined in Figure 26. Here, grayware with ≥10 generated files is aggregated into the category *Number of Files* = 10. Figure 26 shows that 2004-born grayware strains have no preference in the numbers of created files; while species discovered in 2005 tend to contaminate affected systems with a large number of files, increasing the difficulty to restore infected systems to a clean state. Grayware discovered in 2006–2008 seems to neatly balance between created files and detection probability as most species create 3-4 files so that a self-recovery scheme can still be feasible while visible traces are marginal.

Another way for grayware to enhance its coherence with victim systems is to pollute registry databases of the latter with multiple keys so that it is automatically activated in every system reboot and survives system crashes, essentially making it a permanent resident on affected systems. By defining the taxonomic feature Registry Key and categorize grayware accordingly, we observe that 52.44% of grayware species create only one registry key, while 18.29% and 8.92% generate two and three registry keys, respectively. Although 90% grayware strains contaminate registry databases with less than 7 entries, some outliers actually insert a significant

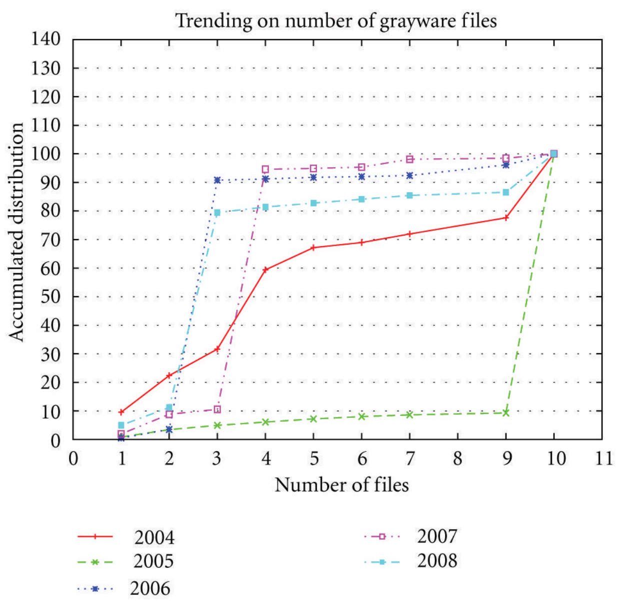

Figure 26: Number of new files created by grayware.

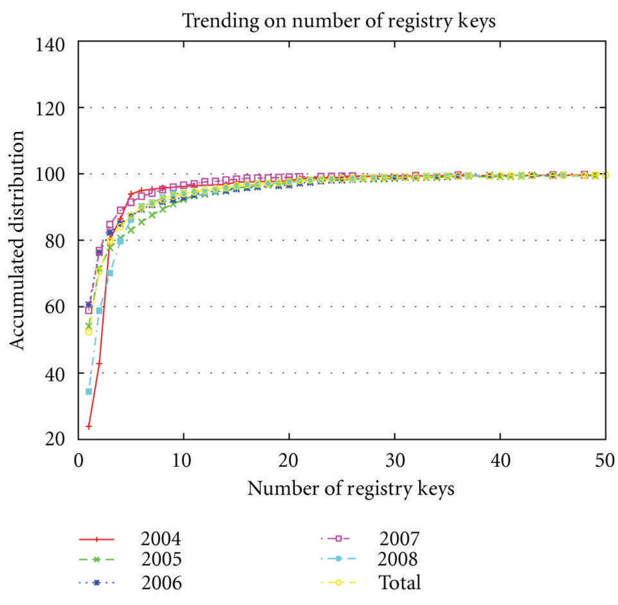

Figure 27: Number of registry keys created by grayware.

amount of registry keys, for instance, 2% grayware specimens generate more than 20 registry keys, a dozen of which including *ADWARE_180SOLUTIONS* and *ADW_ALEXA.BS* even scatter more than 100 registry keys in victim systems.

By further categorizing grayware with respect to feature Discovery Date in addition to Registry Key as demonstrated in Figure 27, we can analyze grayware evolution on registry key manipulation. First, the maximum number of registry keys created by grayware increases gradually: it is 84 in 2004, but changes to 167, 119, 189, respectively, for the subsequent three years. Next, the amount of grayware with multiple registry keys also expands annually, in this regard, only 4% of the 2004-created grayware generate >10 registry keys; however, more than 8% of strains discovered in 2005–2008 insert at least 10 entries into registry databases. For the feature Registry Key, there is a trade-off between coherence with infected systems and exposure to detection: a large amount of registry keys certainly helps improve the clotting capability but at the cost of excessive traces left by the created keys that may eventually betray the grayware. Similar to feature Created Files, Registry Key can achieve a self-healing effect by monitoring the status of registry database and rematerializing the keys if any change is detected.

7.4. Risk Imposed by Grayware. Efficient incident responses and effective defense strategies necessitate grayware risk assessment. However, the diverse grayware behavior and its sophisticated characteristics make it challenging to thoroughly evaluate the grayware threat on the Internet ecosystems. Based on the grayware characteristics defined in the proposed framework and summarized in Table 8, we design the feature Risk Level to measure the threat imposed by grayware. In addition to listing taxonomic features in Table 8, which clearly cover the entire grayware life cycle, we also quantify the contribution to the feature Risk Level by each characteristic with the configurable scoring mechanism defined in Column *Default Scoring Method* and compute a risk score for each grayware. For instance, the spyware *TSPY_LINEAGE.GL* outlined in Table 1 attacks seven different OSs causing 7 points to be added to its risk score; in addition, each of its high information exposure and high system impact contributes 3 points to its risk score as well. By summing up the contributions from all features described in Table 8 for *TSPY_LINEAGE.GL*, we obtain its risk score 27.

Based on risk scores, we define five categories for feature Risk Level according to the following criteria.

(1) *Extremely Critical.* This category accommodates the most dangerous grayware species that possess significant damaging power and are very difficult to eliminate completely. Specimens of this category typically have risk scores >40 (configurable).

(2) *Highly Critical.* This group contains grayware strains that are highly dangerous and difficult to contain, and assigned risk scores are larger than a threshold 36.

(3) *Moderately Critical.* Grayware in this class may have multiple payloads or attack channels and impose medium system impact or create >2 registry keys. The default threshold for risk scores is 26.

(4) *Mildly Critical.* This category holds grayware specimens with risk scores greater than a specified threshold (20 by default).

(5) *Slightly Critical.* Species pose little threat to affected systems and end users.

The grayware categorization, with respect to the feature Risk Level based on the above-described

Table 8: Features used in the gray risk evaluation by the proposed framework.

Stage	Feature	Description	Default scoring method
Penetration	Affected Platform	Operating systems (OSs) vulnerable to grayware	Number of OSs affected
	Attack Avenue	Installation mechanisms to infect systems	Number of attack channels
Activation	Information Exposure	Expose confidential information	Low: 1, Medium: 2, High: 3
	Integrity Impact	Impact on system integrity and availability	Low: 1, Medium: 2, High: 3
	Destructiveness	Damage file systems, stability, and productivity	No: 1, Yes: 2
	Carried Payload	Malicious activities after invading victims	Number of payloads
Discovery	Information Encryption	Cryptographic methods employed by grayware	No: 1, Yes: 2
	Memory Resident	Stay at main memory after execution	No: 1, Yes: 2
Eradication	Registry Key	Create registry keys to survive reboot	Number of registry keys
	Manipulated Files	Create/modify files to customize victim systems	Number of created files

criteria, allows us to derive that both the spyware *TSPY_LINEAGE.GL* and the adware *ADW_ALEXA.AK* depicted in Table 1 are in category *Moderately Critical*. In comparison, *TSPY_QQPASS.AXY, ADW_HOTBAR.P*, and *HKTL_IPSCAN.C* are assigned to class *Extremely Critical*, while *CRCK_QIQI.A, ADWARE_HUNTBAR.C*, and *SPYWARE_TRAK_ACTLOG.16* are considered highly critical. Furthermore, the grayware classification according to Grayware Type and Risk Level presented in Table 9 helps us investigate the risks imposed by different grayware types. Generally speaking, the majority of grayware with label *Extremely Critical* come from grayware types *Cracking Application, Adware*, and *Spyware*. For category *Highly Critical*, grayware types *Adware, Hacking Tool*, and *Remote Access Trojan (RAT)* are the key contributors; while most members of class *Moderately Critical* belong to *Spyware, Adware, Trojan Spyware*, and *Browser Helper Object*. On the contrary, most strains in *Spyware, Trojan-Spyware*, and *Toolbar* are moderately critical, and the majority of *Dialer, Hacking Tool*, and *Dropper* are labeled as *Mildly Critical*.

To analyze the evolution of risks imposed by grayware, we categorize grayware according to features Risk Level and Discovery Date to obtain the results of Figure 28. The majority of the 2005-born grayware are highly critical, while most species created in 2006 are considered as moderately critical, and many strains discovered in 2007 and 2008 are labeled as *Mildly Critical*. By computing the annual ratio between species in extremely/highly and moderately/mildly/slightly groups, we can observe that the ratio reaches its peak in 2005, then falls into a valley in 2007, but rises again in 2008. It is therefore expected that grayware risk fluctuates along with time, demanding constant monitoring on its evolution and continuous effort on its containment and mitigation.

8. Conclusions and Future Work

By chronologically enumerating discovered spyware, adware, and other grayware, the Trend Micro grayware encyclopedia provides critical information for grayware analysis and incident responses. The encyclopedia is further enhanced by the proposed framework Grayware Assessor that offers classification and generalization capabilities. Treating grayware classification as a supervised learning problem, the proposed framework builds learning models for taxonomic features with the help of support vector machines. Each entry in the grayware encyclopedia is collapsed into a bag of words by ignoring word positions in the entry text and is further represented as a feature vector with each word as an attribute and the word occurrence frequency in the corresponding entry as its value. We reduce the dimensionality of feature space formed by grayware entries which is reduced via feature selection, word stemming, and stopword removal. The training data for learning models are automatically extracted from the encyclopedia, and SVM learning models are built with both multiclass-to-binary reduction and multiclass-optimization methods, while categorization results are visualized with self-organizing maps.

The classifications on taxonomic features covering entire grayware life cycle demonstrate that the proposed framework can classify grayware with high performance in terms of accuracy, precision, recall, and F_1-measure. The trend analysis conducted in the proposed framework helps us understand the grayware evolution and the development of grayware characteristics. To this end, the proposed framework reveals that grayware species are sparing no effort on shrinking their storage footprints to improve propagation speed and therefore reduce the probability of being detected. By finding entry points into victim systems with diversified attack avenues and subsequently transporting multiple payloads into infected hosts, grayware effectively increases its penetration rate and possesses much versatile functionalities. The proposed framework also exposes the ever-improving grayware clotting capability manifested by its deep infiltration into file systems and registry databases on affected machines, making it extremely difficult to be completely eliminated. Furthermore, attacks on security protection applications have become an effective weapon to defeat anti-grayware products. Finally, the threat assessment conducted by the framework points out that grayware types, *Cracking Application, Adware*, and *Spyware*, are the major risks to the Internet ecosystem.

To further enhance both the functionality and flexibility of the proposed framework, we intend to work with

Table 9: Risk levels of grayware assigned by the proposed framework.

#	Genre	Extremely	Highly	Moderately	Mildly	Slightly
1	Spyware	223 (0.257)	44 (0.051)	47244 (54.407)	3954 (4.554)	609 (0.701)
2	Dialer	9 (0.010)	5 (0.006)	95 (0.109)	14931 (17.195)	156 (0.180)
3	Adware	413 (0.476)	12165 (14.010)	774 (0.891)	475 (0.547)	297 (0.342)
4	Hacking Tool	4 (0.005)	68 (0.078)	86 (0.099)	3444 (3.966)	70 (0.081)
5	Browser Helper Object	70 (0.081)	18 (0.021)	351 (0.404)	160 (0.184)	92 (0.080)
6	Cracking Application	461 (0.531)	0 (0.000)	100 (0.115)	27 (0.031)	36 (0.042)
7	Trojan-Spyware	6 (0.007)	0 (0.000)	516 (0.594)	68 (0.078)	18 (0.020)
8	Toolbar	46 (0.053)	16 (0.018)	233 (0.268)	66 (0.076)	33 (0.020)
9	Trackware	63 (0.073)	0 (0.000)	39 (0.045)	9 (0.010)	5 (0.004)
10	Keylogger	8 (0.009)	5 (0.006)	36 (0.042)	25 (0.029)	11 (0.011)
11	Remote Access Trojan	2 (0.002)	58 (0.067)	9 (0.010)	4 (0.005)	2 (0.001)
12	Hijacker	4 (0.005)	2 (0.002)	30 (0.035)	5 (0.006)	4 (0.004)
13	Dropper	2 (0.002)	1 (0.001)	24 (0.028)	34 (0.039)	21 (0.019)

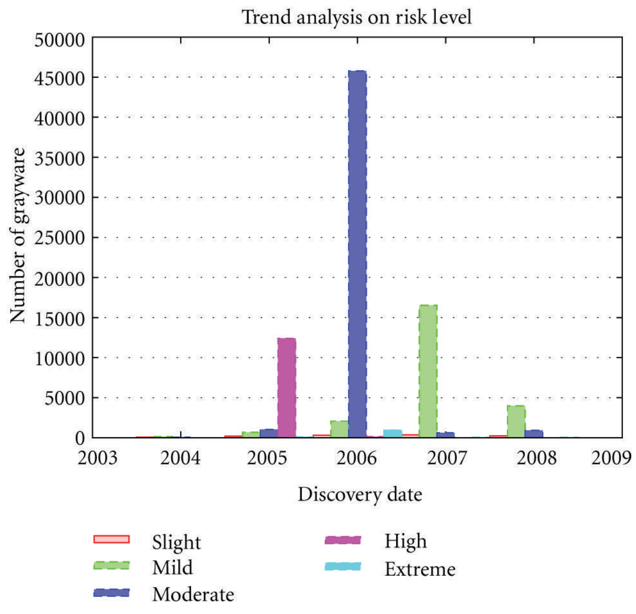

Figure 28: Classification on Risk Level and Discovery Date.

a much larger set of taxonomic features having the properties of determinism and specificity. We plan to strengthen the visualization capability provided by SOM techniques so that the intricate structures formed by grayware categorizations with respect to diversified features can be effectively projected. We are currently integrating into the framework other classification and clustering techniques including Decision Tree and Hierarchical classification methods, so that much more efficient categorization algorithms may be discover in terms of classification performance. We are exploring the applicability of the methodologies in the proposed framework to other grayware encyclopedias and other types of security-related specimens such as malware. We also attempt to evaluate the feasibility and performance of using models learned from one encyclopedia to classify species collected in other encyclopedias. Our preliminary evaluation presented in this paper indicates that such a research direction is promising although manual intervention is still needed due to inconsistencies in feature taxonomies, connotation, and granularity among different encyclopedias. Finally, we aim at providing toolkits that are able to monitor in real time the Internet ecosystem, evaluate the grayware evolution, and forecast the development trend of various grayware species.

Acknowledgments

The authors are grateful to the invaluable comments of the anonymous reviewers that helped us significantly improve the presentation of our work. They are also indebted to Professors Alex Delis and Mema Roussopoulos at University of Athens for their suggestions on the draft of the paper.

References

[1] Grayware, 2011, http://en.wikipedia.org/wiki/Grayware .

[2] N. Good, R. Dhamija, J. Grossklags et al., “Stopping spyware at the gate: a user study of privacy, notice and spyware,” in *Proceedings of the Symposium on Usable Privacy and Security (SOUPS ’05)*, vol. 93, pp. 43–52, 2005.

[3] M. Warkentin, X. Luo, and G. F. Templeton, “A framework for spyware assessment,” *Communications of the ACM*, vol. 48, no. 8, pp. 79–84, 2005.

[4] Tenebril, “Spyware: A Dangerous New Threat,” November 2005, http://www.tenebril.com/pdf/SpywareProfiling.pdf .

[5] Webroot Software, Inc. “Monitoring Software on Your PC: Spyware, Adware, and Other Software,” 2004, http://www.ftc.gov/os/comments/spyware/040521webrootsoftware.pdf.

[6] B. Edelman, “Claria’s Misleading Installation Methods—Ezone.com,” 2005, http://www.benedelman.org/spyware/installations/ezone-claria/.

[7] N. F. Awad and K. Fitzgerald, “The deceptive behaviors that offend us most about spyware,” *Communications of the ACM*, vol. 48, no. 8, pp. 55–60, 2005.

[8] T. F. Stafford and A. Urbaczewski, “Spyware: the ghost in the machine,” *Communications of AIS*, vol. 14, pp. 291–306, 2004.

[9] S. S. M. Chow, L. C. K. Hui, S. M. Yiu, K. P. Chow, and R. W. C. Lui, "A generic anti-spyware solution by access control list at kernel level," *Journal of Systems and Software*, vol. 75, no. 1-2, pp. 227–234, 2005.

[10] I. You and K. Yim, "Malware obfuscation techniques: a brief survey," in *Proceedings of the International Conference on Broadband, Wireless Computing Communication and Applications, (BWCCA '10)*, pp. 297–300, IEEE, Fukuoka, Japan, 2010.

[11] Trend Micro, "Spyware/Grayware Encyclopedia," April 2009, http://www.trendmicro.com/vinfo/grayware .

[12] EarthLink, "Most Dangerous Types Of Spyware Increasing, States SpyAudit Survey," 2005, http://ir.earthlink.net/releasedetail.cfm?ReleaseID=249692.

[13] H. Lee, C. A. Christiansen, and B. E. Burke, "Worldwide Spyware 2004–2008 Forecast and Analysis: Security and System Management Sharing Nightmares," 2004, http://www.idc.com/getdoc.jsp?containedId=32229.

[14] Osterman Research, "Survey on Messaging Issues," 2005, http://www.ostermanresearch.com/results/orresults_2005-01.pdf.

[15] The Radicati Group, "Inc. Corporate Anti-Spyware Market, 2006–2010," 2006, http://www.radicati.com/ .

[16] E. Howes, "The Anatomy of a 'Drive-by-Download'," 2004, http://www.spywarewarrior.com/uiuc/dbd-anatomy.htm .

[17] D. J. Stang, "Internet Intruders: Spyware, Adware, Hijackers and Other Pests," 2004, http://Research.PestPatrol.com .

[18] B. Edelman, "Threats Against Spyware Detectors, Removers, and Critics," 2005, http://www.benedelman.org/spyware/threats.

[19] CNet, "Symantec Sued for Labeling Product 'Adware'," 2004, http://news.com.com/2100-1023 3-5293992.html .

[20] Federal Trade Commission, "Monitoring Software on Your PC: Spyware, Adware, and Other Software," 2005, http://www.ftc.gov/os/comments/spyware/040521webrootsoftware.pdf.

[21] E. Howes, "The FTC Spyware Workshop: One Year Later," April 2005, http://netfiles.uiuc.edu/ehowes/www .

[22] B. Edelman, "Advertisers Using WhenU.," June 2004, http://www.benedelman.org/spyware/whenu-advertisers .

[23] B. Edelman, "How Google's Blogspot Helps Spread Unwanted Software," 2005, http://www.benedelman.org/news/022205-1.html.

[24] B. Edelman, "More on Google's Role: Syndicated Ads Shown Through Ill-Gotten Third-Party Toolbars," 2005, http://www.benedelman.org/news/060605-1.html .

[25] R. K. Shazhad, S. I. Haider, and N. Lavesson, "Detection of spyware by mining executable files," in *Proceedings of the 5th International Conference on Availability, Reliability, and Security (ARES '10)*, pp. 295–302, 2010.

[26] V. N. Vapnik, *The Nature of Statistical Learning Theory: Information Science and Statistics*, Springer, New York, NY, USA, 1999.

[27] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of Machine Learning Research*, vol. 3, pp. 1289–1305, 2003.

[28] T. Kohonen, *Self-Organizing Maps*, Springer, Berlin, Germany, 2000.

[29] T. Joachims, "A support vector method for multivariate performance measures," in *Proceedings of the International Conference on Machine Learning (ICML '05)*, pp. 377–384, ACM Press, 2005.

[30] G. Goth, "Spyware: Menace, Nuisance, or Both?" *IEEE Security & Privacy*, vol. 1, no. 3, pp. 10–11, 2003.

[31] D. T. Gilbert, C. K. Morewedge, J. L. Risen, and T. D. Wilson, "Looking forward to looking backward: The misprediction of regret," *Psychological Science*, vol. 15, no. 5, pp. 346–350, 2004.

[32] Computer Associates, "Spyware Encyclopedia" August 2011, http://gsa.ca.com/pest/browse.aspx.

[33] P. J. Bruening and M. Steffen, "Spyware: technologies, issues, and policy proposals," *Journal of Internet Law*, vol. 7, no. 9, pp. 3–8, 2004.

[34] U. Lindqvist and E. Jonsson, "How to Systematically Classify Computer Security Intrusions," in *Proceedings of the 1997 IEEE Symposium on Security & Privacy*, pp. 154–163, IEEE Computer Society Press, Oakland, Calif, USA, 1997.

[35] M. Fredrikson, S. Jha, M. Christodorescu, R. Sailer, and X. Yan, "Synthesizing near-optimal malware specifications from suspicious behaviors," in *Proceedings of the 31st IEEE Symposium on Security and Privacy*, pp. 45–60, 2010.

[36] D. Babic, D. Reynaud, and D. Song, "Malware analysis with tree automata inference," in *Proceedings of the 23rd International Conference on Computer Aided Verification*, Springer, Snowbird, Utah, USA, July 2011.

[37] T. Lee and J. Mody, "Behavioral Classification," in *Proceedings of the EICAR Conference*, pp. 1–17, 2006.

[38] L. L. DeLooze, "Classification of computer attacks using a self-organizing map," in *Proceedings fron the Fifth Annual IEEE System, Man and Cybernetics Information Assurance Workshop, SMC*, pp. 365–369, 2004.

[39] H. S. Venter, J. H. P. Eloff, and Y. L. Li, "Standardising vulnerability categories," *Computers and Security*, vol. 27, no. 3-4, pp. 71–83, 2008.

[40] B. Schoelkopf and A. J. Smola, *Learning with Kernels*, MIT Press, Cambridge, Mass, USA, 2002.

[41] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[42] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods—Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola, Eds., chapter 11, pp. 169–184, MIT Press, Cambridge, Mass, USA, 1999.

[43] R. Collobert and S. Bengio, "SVMTorch: Support Vector Machines for large-scale regression problems," *Journal of Machine Learning Research*, vol. 1, no. 2, pp. 143–160, 2001.

[44] J. C. Platt, "Fast training of Support Vector Machines using sequential minimal optimization," in *Advances in Kernel Methods—Support Vector Learning*, B. Sch.olkopf, C. Burges, and A. Smola, Eds., chapter 12, MIT Press, 1999.

[45] S. S. Keerthi and D. Decoste, "A modified finite Newton method for fast solution of large scale linear SVMs," *Journal of Machine Learning Research*, vol. 6, 2005.

[46] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers," *Journal of Machine Learning Research*, vol. 1, no. 2, pp. 113–141, 2001.

[47] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," *Annals of Statistics*, vol. 26, no. 2, pp. 451–471, 1998.

[48] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1995.

[49] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification," *Advances in Neural Information Processing Systems*, vol. 12, pp. 547–553, 2000.

[50] E. J. Bredensteiner and K. P. Bennett, "Multicategory classification by support vector machines," *Computational Optimization and Applications*, vol. 12, no. 1-3, pp. 53–79, 1999.

[51] K. Crammer and Y. Singer, “On the algorithmic implementation of multiclass kernel-based vector machines,” *Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2001.

[52] Symantec, “Virus Encyclopedia of Symantec,” 2009, http://www.sarc.com/avcenter/venc.

[53] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, 1988.

[54] M. F. Porter, “Celebrating 40 years of ICT in libraries, museums and archives: An algorithm for suffix stripping,” *Program*, vol. 40, no. 3, pp. 211–218, 2006.

[55] “Snowball: a String Processing Language for Creating Stemming algorithms in Information Retrieval,” 2008, http://snowball.tartarus.org.