# SCMP Classification Method Framework for Malicious Code and Multi-label Mechanism for Risk Behaviors

**Authors:** Xiao Xinguang, Li Chenping, Han Yaoguang, Tong Zhiming, Li Qi, Li Chenping

**Date:** 2024-05-09T00:00:00+00:00

## Abstract

In response to the demand from academia and industry for scientific malware classification methods, this research builds upon the foundations of existing work, draws upon the advantages of Kaspersky' s relatively rigorous multi-stage classification nomenclature, follows a methodology emphasizing mutual exclusivity, complete coverage, and convergence, and combines this with "threat-risk behavior tags", thereby forming a malware classification framework that adheres to the MECE principle, achieves classification convergence, and is compatible with de facto industry classifications, effectively supporting security defense and governance.

## Full Text

## Preamble

### SCMP Malware Classification Framework and Multi-Label Risk Behavior Mechanism—A Taxonomy Compliant with MECE Principles and Enhanced Nomenclature for Improved Risk Disclosure

Xiao Xinguang[1], Li Chenping[1]*, Han Yaoguang[1], Tong Zhiming[1], Li Qi[1]
[1](Antiy Technology Group Co., Ltd., Harbin 150028, China)

[**Objective**] To address the academic and industrial demand for a scientific malware classification methodology, [**Methods**] this study builds upon existing work, drawing insights from Kaspersky' s rigorous multi-stage naming conventions. It emphasizes mutual exclusivity, comprehensive coverage, and convergence, integrating "threat risk behavior labels" into a unified framework. [**Results**] The result is a malware classification framework that conforms to

MECE (Mutually Exclusive, Collectively Exhaustive) principles, achieves taxonomic convergence, and remains compatible with de facto industry classifications. [**Conclusion**] This framework effectively supports security defense and governance operations.

**Keywords:** malware, classification methodology, threat behavior, risk labels, engineering foundation
**Classification Number:** TP309.5

**Scientific Classification of Malware from Practice**

Xiao Xinguang[1], Li Chenping[1], Han Yaoguang[1], Tong Zhiming[1], Li Qi[1]
[1](Antiy Technology Group Co., Ltd., Harbin 150028, China)

**Abstract:** [**Objective**] In response to demands from academia and industry for a scientific malware classification system, [**Methods**] this study builds upon existing research and draws from Kaspersky' s rigorous multi-stage naming methodology, emphasizing principles of mutual exclusivity, comprehensive coverage, and convergence, combined with the use of "threat risk behavior labels." [**Results**] This forms a malware classification framework that conforms to MECE principles, achieves taxonomic convergence, and remains compatible with real-world industry classifications. [**Conclusions**] The framework provides effective support for security defense and governance.

**Keywords:** Malware Engineering, Classification Method, Threat Behavior, Risk Label

## 1. Introduction

The 1991 CARO conference established the initial industry consensus on computer virus nomenclature, proposing the original four-segment naming convention known as the "CARO Standard" [1]. At that time, the personal computing environment was dominated by DOS systems without widespread network connectivity; disk copying served as the primary medium for software installation and data exchange. Consequently, infectious viruses represented the predominant threat landscape, with family and variant totals numbering in the thousands. However, with the construction of global information highways, rapid development of Internet applications, and continuous appreciation of virtual and digital assets, traditional infectious viruses have ceased to be mainstream within the broader malware spectrum. Network worms, Trojan horses, and other malware types have successively emerged as dominant threat categories, accompanied by various emerging risks and ambiguous classifications. The overall number of malware variants has now ballooned to tens of millions. Nearly all sophisticated and large-scale attack operations rely on malware deployment and execution.

Therefore, within the security capability spectrum, malware detection, identification, and precise naming capabilities constitute fundamental pillars of security protection. In security operations practice, detection by antivirus engines

and other detection mechanisms, triggering subsequent actions such as removal, quarantine, and blocking, represents the most basic security capability. Malware detection must be transformed into clear alert information and disposition feedback: for desktop systems, this information must be displayed in alert event windows; for enterprise-grade AV, EPP, or EDR management interfaces, it must appear in event lists and TOP statistics; related logs must also be integrated for correlation analysis by SIEM, XDR, and other systems. This necessitates a standardized structure for malware detection events, including timestamps, objects, malware detection status, malware names, and processing results. Simultaneously, standardized and informative malware naming is required. Network administrators need to retrieve relevant information to assess risks based on these names, while SIEM and XDR platforms require them for correlation analysis and risk prioritization. These requirements have made the quality of malware naming increasingly critical. In retrospect, the CARO Standard, while leaving a legacy of precise segmented naming, also suffers from the regrettable omission of a "classification" concept. Since malware family naming inevitably involves substantial conventional "customs" rather than "paradigms," focusing primarily on individualizing malware families, its ability to reveal associated information is often insufficient. Therefore, the complete malware name in alerts requires some information providing common "attributes" of malware to augment informational support for users and IT operations personnel. At the societal level, for security early warning, situational analysis, and emergency response, regulatory and emergency departments need to analyze overall threat trends and patterns beyond the statistical "dimensions" of malware families and variants. Consequently, whether for security protection and response in information asset scenarios or for societal-level security governance, including academic and scientific research treating malware as a resource, more scientific and clear classification standards and naming conventions are needed to provide more precise and informative malware data.

Both academia and industry have conducted long-term explorations and extensive practices toward more scientific malware classification methods. In 2004, Kaspersky Lab proposed a malware classification system based on a "classification tree," using malware behavior on hosts as the primary criterion and following a principle of prioritizing behavioral risk levels to achieve mutual exclusivity [2]. Antivirus vendors such as Symantec, Microsoft, and Trend Micro have also developed their own malware classification methods. Although nearly all mainstream vendors acknowledge the three foundational categories of "virus," "worm," and "Trojan," they exhibit certain differences in defining these categories and establishing category priorities [3][4][5]. The Malware Type Enumeration (CME) led by the U.S. Computer Emergency Response Team and the Malware Attribute Enumeration and Characterization (MAEC) led by MITRE Corporation have made active efforts to promote industry consensus and coordinate security devices [6]. However, in reality, facing the rapid expansion of malware, attempting to unify nomenclature harbors unrealistic fantasies. Due to the lack of clear classification criteria, security vendors, teams, and institu-

tions have relatively arbitrarily added new classifications, while some emerging vendors have created concepts for commercial differentiation, leading to continuous divergence and branching on classification issues that should possess common attributes. For example, Microsoft' s foundational malware categories have reached as many as 31 types to date.

We must emphasize that no "algorithmic solution" exists for malware classification. Although numerous research papers have attempted various approaches using Bayesian methods, nearest neighbor calculations, and graph computing to design so-called "classification algorithms," their granularity actually corresponds to the industry concept of malware families rather than classification categories. Furthermore, without an "engineering filter" for assistance, almost all algorithms based on "entropy" or deep learning cannot address real-world industrial challenges—namely, algorithmic feasibility within a sample space where the baseline 存量 reaches billions and daily increments number in the millions.

Since the fundamental operational activities of malware countermevolution revolve around industry practices of capture, automated and manual analysis, rule extraction, and engine/signature database updates, continuous construction and control of massive sample sets and analysis infrastructure have enabled industry to maintain de facto standards. Each mainstream vendor possesses its own experience and internal specifications for malware naming and classification. Overall, several distinct "schools"have emerged in sample classification and naming styles. The first is the "Family School" represented by American mainstream vendors such as McAfee and Symantec, which largely inherited the original CARO style. Their multi-segment malware naming structure lacks a unified "classification," only supplementing runtime environment information like W32 and adding a few critical behavior suffixes such as @mm. This approach suffers from poor comprehensibility and information disclosure.

The second is the "Popularity School"represented by Microsoft. As a new critical force in security without the burden of historical naming conventions, Microsoft' s malware naming methodology primarily uses prevalent threat types as classification criteria, mainly covering Windows platform threats. This perspective is overly oriented toward Microsoft's customer scenarios and operational conditions as an OS and application vendor, making it difficult to cover the entire malware ecosystem comprehensively. The third is the "Behavior School" represented by Eastern European mainstream vendors like Kaspersky and Bitdefender. This school's malware naming methodology primarily supports classification based on specific malware behaviors, relatively strictly adhering to a four-segment structure of "classification prefix," "environment prefix," "family name," and "variant number." Its structural clarity is significantly better. However, classification expansion lacks clear standards, with arbitrary additions resulting in nearly a hundred classification prefixes at its peak. Not only did the three classic foundational categories of Virus, Worm, and Trojan spawn over twenty "subclasses" as primary prefixes such as P2P-Worm and Email-Worm, but new "primary classification prefixes" like Backdoor, Rootkit, AdWare, and PornWare also con-

tinuously emerged. Although Kaspersky' s threat annual reports and security blog statistics show they have in fact conducted certain classification integration and convergence, demonstrating attempts to adjust their malware behavior classification methodology according to new security threats and attack trends, these changes have never been implemented in engine alert outputs.

Antiy and its research collaborators have drawn upon Kaspersky' s rigorous multi-stage classification naming advantages while emphasizing mutual exclusivity and convergence in classification naming. In the "Antiy Laboratory Malware Classification Standard (2015)," Antiy attempted to propose an eight-type classification method, overall suggesting the absorption of all high-risk samples without active self-propagation capabilities into the Trojan category, and specifically proposing TestFile and JunkFile as two independent categories. This formed a classification framework with mutual exclusivity and complete coverage for the entire sample set, combined with naming conventions based on malware core behaviors and priority levels, capable of effectively covering all malware samples.

Building upon this foundation, the researchers of this paper aim to explicitly propose a set of classification specifications combined with "threat risk behavior labels," which can be integrated with CARO convention naming practices and de facto standards to form a new classification and naming logic. Our working principles and objectives include:
1. The classification methodology conforms to MECE (Mutually Exclusive, Collectively Exhaustive) principles. According to standards of mutual exclusivity and complete coverage, the methodology should encompass malware samples while mutually exclusively assigning each sample to an independent classification space.
2. The overall number of classifications should converge, with clear, uniquely conditional classification criteria between categories.
3. It should be compatible with current industry detection results, enabling integration and transformation of alert information from most mainstream vendors with strict classification naming specifications.
4. It should effectively support protection scenario requirements, threat intelligence sharing, risk early warning and notification, judicial forensics, and sentencing.

## 2. SCMP Classification Formation Process

### 2.1 Foundation: Extending Classic Classification Standards

Based on the fundamental dimension of replication and propagation methods, malware has three most basic categories:
1. **Infectious Virus**: Possesses host infection attributes, using hosts to complete replication and propagation;
2. **Worm**: Does not infect hosts, capable of self-replication and propagation;
3. **Trojan**: Lacks active infection propagation attributes and does not self-

replicate.

**Table 1 Basic Malware Classification Table**

These three categories form foundational classifications based on the same dimension, representing the basic paradigm of malware classification. In reality, malware such as Emotet and Ryuk possess both PE file infection capabilities and network-based propagation, clearly exhibiting dual attributes of infectious viruses and worms. Therefore, we supplement two working principles in our recommended classification naming approach:

**Working Principle 1**: All classifications are assigned different priority levels. For malware samples with cross-category attributes, use the classification with the higher priority level.

**Working Principle 2**: Introduce a tagging mechanism in sample naming to enhance informational value. Multiple tags may appear in a naming structure. When malware exhibits other critical threat behaviors beyond its category characteristics, use more granular behavior tags for identification to provide a multi-dimensional knowledge structure without information loss while satisfying one-dimensional classification requirements.

### 2.2 Classification Supplement and Extension Process

The extension process beyond the three basic categories of Virus, Worm, and Trojan corresponds to the researchers' investigation of threats that existing classifications could not encompass.

**(1) Extending "HackTool" Classification Based on Runtime Location** The traditional malware observation perspective focuses on the attacked host scenario, centering on malware diffusion, propagation, deployment, execution, and post-execution derivatives. Since the late 20th century, tools like packet generators in OOB and SMBDIE attacks have been widely used in attacks. Since these do not affect the security of the host environment where they run but rather impact receiving systems, they cannot be contained within traditional malware classification scopes. However, in security incident response and forensics, these tools must be detectable. Therefore, using runtime location as a new differentiator, we conducted the first fundamental extension, encompassing tools running on the attacker side that do not compromise the integrity, availability, or confidentiality of the current host (otherwise they should be classified as infectious viruses, worms, or Trojans).

**(2) Extending "Grayware" Classification Based on Low-Impact Risk** Traditional malware and hacking tools typically correspond to cybercrime activities or APT intrusions with national or regional backgrounds. However, in actual network activities, some software and tools are used to achieve low-impact behaviors such as adware, pornware, and rogue software. We propose using grayware to cover this category, which mostly consists of advertising derivatives delivered via the internet, often accompanying related software downloads. Some

antivirus vendors use "PUA (Potentially Unwanted Application)"for similar sample objects. Since classifications directly manifest as primary prefixes—a focal point for network administrators—grayware, despite having many "behavioral subtypes" that cause severe primary prefix bloat in some strictly classifying vendors, overall resides in the low-risk zone. Allowing primary prefix proliferation would create an overwhelming proportion of low-risk threat primary prefixes, causing attention resource imbalance. Converging them into a major classification reduces interference and panic among users and network management operators from alert names and minimizes consolidation interference for SIEM and XDR systems.

**(3) Extending "Riskware" Classification Based on Uncertainty from Non-Malicious Development**

The essential tenet of the term "malware"lies in "maliciousness"—the developer's intent to compromise information system integrity, availability, or confidentiality. This makes it impossible to define within the malware concept boundary the following scenario: software written by legitimate organizations and individuals to support actual system functions and business operations that could potentially be used by attackers as attack payloads. This creates situations where related software tools are normal applications in most scenarios but malicious tools in 少数 scenarios. Based on this, we believe the entire malware concept scope requires extension, adding the "Riskware" category. For example, since around 2002, security response operations have encountered numerous scenarios where legitimate remote management tools were used as remote control Trojans, such as the open-source network management software VNC (remote management software), a typical riskware. The Riskware classification prompt helps network administrators make judgments based on whether the tool was deployed by themselves or users. Simultaneously, for Riskware classification alerts, antivirus software adopts an alert-and-wait approach for user and administrator disposition, avoiding false positives on normal applications.

**(4) Extending "TestFile" Classification Based on Self-Test Samples from Testing Organizations**

Another classification challenge we encountered was how to classify "EICAR," a standard test file created by the European Institute for Computer Antivirus Research for legally testing and validating antivirus software and other security products. By clearly distinguishing actual malware from test specimens, we derived the independent "TestFile" classification. "TestFile" explicitly identifies malware used for testing antivirus engines and security products, commonly appearing in adversarial testing of samples.

**(5) Extending "JunkFile" Classification for Meaningless Samples**

In multiple customer testing scenarios, we encountered interference from invalid test samples. These invalid samples were often downloaded from the internet by customers. Although claimed to be sample resource packages, large numbers of files were not actual samples but meaningless binary garbage files. To avoid excessive discussion about customer sample quality, we provided a unified

classification identifier for such samples based on the principle that "alerting and disposing of related samples will not cause consequences or impact on the system and other vendors also alert on them."

## 3. SCMP Classification Methodology Framework

The extension process described in the previous chapter initially formed eight malware classification categories, making this classification standard a MECE-compliant taxonomy through the methodological framework shown in Figure 1 [Figure 1: see original paper]. The framework introduces five dimensions—threat risk, threat classification, differentiation criteria, specific classification separation methods, and authorship—to constitute this framework. The overall classification objects are: all objects captured and discovered to date for which mainstream security products or detection engines produce naming output.

**Figure 1 SCMP Malware Classification Methodology Framework**

This process forms eight classifications through seven classification cut points:
1. **First differentiation criterion**: Whether the sample is meaningful data—files lacking functional or valid significance are designated as junk files;
2. **Second differentiation criterion**: Whether the sample' s construction purpose is for user scenario testing/verification or functional implementation—samples created for testing/verification are designated as test files;
3. **Third differentiation criterion**: The actual runtime location of the sample—samples running on the attacker side rather than the victim side are designated as hack tools;
4. **Fourth differentiation criterion**: Maliciousness of intent—samples without malicious intent, written to perform normal business functions but potentially used by attackers to create risk, are designated as riskware;
5. **Fifth differentiation criterion**: Separation based on infringement severity, i.e., the distinction between criminal and illegal acts. Samples with weaker infringement that does not constitute crime but may be illegal are classified as grayware;
6. **Sixth differentiation criterion**: Among objects with strong infringement running on victim hosts, separation based on self-propagation capability—samples with strong infringement but lacking autonomous replication propagation are classified as Trojans;
7. **Seventh differentiation criterion**: Among samples with autonomous propagation capability, those not relying on host infection for self-replication are designated as worms, while those dependent on host propagation are designated as infectious viruses.

This classification standard completely covers the existing full malware sample set, satisfies mutual exclusivity conditions, and establishes relatively complete mapping with other dimensions.

For example, mapping to authorship: infectious viruses, worms, Trojans, and hack tools correspond to attack organizations, individuals, and attack enablers;

grayware and riskware correspond to software vendors and developers; test files correspond to testing organizations; junk files, due to complex causes and no specific authorship, correspond to no entity.

Another example is mapping to threat risk. Since infectious viruses, worms, and Trojans run on victim hosts, their risk level is "strong direct risk zone"; hack tools run on attacker or compromised terminals without affecting their own runtime environment, belonging to "strong associated risk zone" ; grayware has weaker infringement, belonging to "weak risk zone" ; riskware, being normal software in most scenarios but used for attacks in 少数 cases, belongs to "uncertain risk zone" ; test files and junk files neither run nor cause substantive impact, belonging to "no risk zone."

Comparing the framework's cut point activation sequence with the classification extension sequence in previous chapters reveals that the classification cutting order is essentially the reverse of the extension process, demonstrating that this classification methodology has undergone practical testing through threat confrontation and security operations.

## 4. SCMP' s Eight Foundation Categories

Based on the SCMP classification methodology framework, malware includes the following eight foundation categories. When developing English names for each category, we referenced Pascal case naming conventions.

### 4.1 Virus (Infectious Virus)

**Definition**: Infectious viruses are malware that complete self-propagation by infecting hosts.
**Classification Priority**: 0 (highest).
**Explanation**: Hosts for infectious viruses include but are not limited to: disk files, boot sectors, and other carriers enabling malware self-propagation. Infectious viruses represent the initial mainstream malware form, with core characteristics being self-replication that depends on hosts.

Since host infection is a behavior that compromises system fundamental runtime environment and basic runtime unit (program) integrity, the classification attribute of infectious viruses should be the highest priority attribute. All malware possessing active host infection and host-based propagation attributes, regardless of other behaviors, should be classified into the infectious virus category.

**Special Cases and Exception List**: Due to historical reasons, some samples without infectious capabilities have been assigned the Virus classification prefix by security vendors, such as numerous COM, DOS_{MZ}, and BAT format samples from the DOS era. Some vendors' naming prefixes include Macro Virus, which still belongs to the infectious category. Meanwhile, Trojan binders, despite quasi-infectious behavior, are primarily used for deployment processes rather than persistence within attack scenarios, and most do not compromise the

integrity of bundled programs but rather add independent file headers. Therefore, they remain classified under the Trojan category.

### 4.2 Worm

**Definition**: Worms are malware capable of autonomous propagation without host assistance. Self-replication methods include storage media-based and network-based approaches.
**Classification Priority**: 1.
**Explanation**: Worms can typically exploit system or software vulnerabilities, email, instant messaging, file sharing, social networks, network sharing, or removable storage devices for propagation and diffusion. Some worms propagate as network packets. Their core characteristic is self-replication capability without host infection dependency.

**Special Cases and Exception List**: For other malware components and modules deployed through worm frameworks, if they are not other already-named malware, they should 原则上 be treated as components or samples of that worm.

### 4.3 Trojan

**Definition**: Trojans are malware aimed at severely compromising the availability, integrity, or confidentiality of running systems, or achieving equivalent effects after execution.
**Classification Priority**: 2.
**Explanation**: Although Trojans and hack tools share consistent development purposes, Trojans run in victim hosts. Their core characteristic is operating in victim environments, constituting strong threat risk.

**Special Cases and Exception List**: Security vendors have many classification prefixes for high-risk threats, such as ransomware, miners, and backdoors. These classifications lack autonomous propagation capabilities but possess strong malicious intent and significant infringement. Such threats can be categorized under the Trojan scope.

### 4.4 HackTool

**Definition**: Hack tools are malware written to achieve objectives of compromising computer availability, integrity, or confidentiality, but running on the attacker side to serve auxiliary attack functions.
**Explanation**: Although hack tools share consistent development purposes with Trojans, their operation does not pose corresponding threat risks to the current host environment.
**Special Cases and Exception List**: The control side of remote control tools 符合 our definition of hack tools, but due to the need for mapping relationships with controlled sides in naming, they are typically classified under the Trojan category, simultaneously using "Backdoor" and "Client" behavior tags as modifiers.

### 4.5 Grayware

**Definition**: Grayware is software or plugins running on victim hosts, occupying victim host resources, potentially causing host and user information leakage, but insufficient to constitute major risks.
**Explanation**: Key distinctions between grayware and Trojans include: 1) Development purpose: Trojans are written for pure infringement purposes, while grayware may include some user-desired functions beyond infringing functionalities; 2) Infringement behavior: Trojans steal relatively critical user information, collecting environmental information to facilitate further attack actions, whereas grayware collects information for user profiling, ad pop-ups, and other lightweight monetization; 3) Organization: Trojans are typically released by attack organizations, criminal groups, and individuals, while grayware is usually written and released by legitimate vendors and developers; 4) Infringement qualification: Trojan development constitutes criminal behavior, while grayware development typically constitutes illegal behavior.

**Special Cases and Exception List**: Many classification prefixes for lightweight threats from security vendors, such as adware, pornware, can actually be categorized under grayware.

### 4.6 Riskware

**Definition**: Riskware refers to programs written to implement certain defined computer business functions. Although not written for malicious purposes, they may be transformed into attack tools in attack scenarios. In other words, their security risk is related to "who installs or deploys" and "for what purpose," not the publication purpose.
**Explanation**: Typical riskware includes commercial remote tools like PcAnywhere and VNC. These software display icons in the computer status bar during normal use, are perceivable by remote controllers as normal management tools, but have been extensively used as remote control tools in attack activities.
**Special Cases and Exception List**: In attack event capture, if tools clearly modified by attackers are discovered, products will explicitly mark them as Trojans. Alerting these tools as riskware is somewhat an expedient measure by antivirus companies to ensure they can detect tool exploitation while avoiding false positives on users' normal tools that could cause business impact or legal liability. In some antivirus engines, independent switches are set for whether to alert on such software.

### 4.7 TestFile

**Definition**: Test files refer to public documents released by testing organizations to allow users to detect whether antivirus software works properly in their own scenario environments.
**Explanation**: Test files are not files used by testing organizations to test secu-

rity software effectiveness, but rather files usable by ordinary users to self-test security software in their own system environments through simple, safe methods. Principally, files serving as test tools should have meaning and should not be executable programs.

**Special Cases and Exception List**: To date, the only file clearly meeting this standard is eicar released by testing organizations. Despite being a single file, its characteristics can constitute an independent branch.

### 4.8 JunkFile

**Definition**: Junk files refer to files without actual execution capability or data significance but used as test samples by some users or testing organizations. To avoid interference with normal security product operation, they must be treated as a separate classification.
**Explanation**: False positives and false selections of normal executable programs or data files should not serve as alert bases. Although junk files do not 符合 malware definitions, they must be treated as a special classification due to their actual existence in antivirus product event alerts. This classification essentially represents a compromise antivirus companies must make due to user and testing organization capability deficiencies or operational errors.

**Special Cases and Exception List**: The core element for judging junk files is whether the file itself has meaning. Virus remnant files left due to incomplete infectious virus removal by antivirus software should not be treated as junk files but should be stored following malware naming paradigms with crushed tags added.

## 5. Formal Verification

This chapter describes the malware SCMP classification methodology as a formal system and verifies its compliance with MECE principles.

### 5.1 Universe of Discourse

The object domain discussed in this formal system is: all objects captured and discovered to date for which mainstream security products or detection engines produce naming output.

### 5.2 Symbols

**Table 2 SCMP Classification Methodology Formal System Symbols**

| Symbol Type | Notation | Semantic Interpretation |
|---|---|---|
| Object Symbol | Single lowercase English letter | A malware file object to be classified |

| Symbol Type | Notation | Semantic Interpretation |
|---|---|---|
| Assertion Symbol | English word with initial capital | Virus(x) is an assertion about object x named Virus, interpreted as: x belongs to the "Virus" classification |
| Quantifier | xF(x) | All objects x satisfying proposition F(x) |
| Implication | A→B | When assertion A is true, assertion B is also true; if A then B. But B cannot imply A |
| Equivalence | A B | Assertions A and B are simultaneously true/false and mutually derivable |
| Negation | ¬A | Negation of assertion A |
| Conjunction | A&B | Only when both assertions A and B are true is A&B true |
| Disjunction | A B | When either assertion A or B is true, A B is true |
| Joint Denial | A↓B↓C↓D | Only when assertions A, B, C, D are all false is A↓B↓C↓D true |
| Priority Brackets | ¬[A&B C]↓D | Items in [] have higher operation priority than items outside []. In the example, logical operation order is &, , ↓ sequentially |
| Comments | /* / /sometext*/ | Text between comment symbols is explanatory content |

### 5.3 Assertion Symbols and Semantics

- **Virus(x)**: Means x belongs to the "Virus" classification.
- **Worm(x)**: Means x belongs to the "Worm" classification.
- **Trojan(x)**: Means x belongs to the "Trojan" classification.
- **HackTool(x)**: Means x belongs to the "HackTool" classification.
- **Grayware(x)**: Means x belongs to the "Grayware" classification.
- **Riskware(x)**: Means x belongs to the "Riskware" classification.
- **TestFile(x)**: Means x belongs to the "TestFile" classification.
- **JunkFile(x)**: Means x belongs to the "JunkFile" classification.
- **Nomean(x)**: Means x is a file without functional or valid significance data.
- **Test(x)**: Means x is a sample file formed for testing/verification purposes certified by authoritative testing organizations.
- **Environment(x)**: Means x infringes upon its current runtime environment.
- **Male(x)**: Means x is constructed with malicious intent to execute malicious functions.

- **Substantive(x)**: Means x can constitute deterministic, substantive infringement.
- **Copy(x)**: Means x possesses self-replication attributes.
- **Inject(x)**: Means x possesses active host file infection attributes.
- **MECE(E1,E2,E3,⋯,En)**: Means in the assertion set composed of E1,E2,E3,⋯,En, exactly one assertion is true.

### 5.4 Axioms

**Axiom 1**:

Nomean(x) $\rightarrow$ Test(x)↓Environment(x)↓Male(x)↓Substantive(x)↓Inject(x)↓Copy(x)
Test(x) Environment(x) Male(x) Substantive(x) Inject(x) Copy(x) $\rightarrow$ ¬Nomean(x)

/* Meaningless files have no behavior decision points; conversely, if a file hits any decision point, it is meaningful */

**Axiom 2**:

Test(x) $\rightarrow$ Nomean(x)↓Environment(x)↓Male(x)↓Substantive(x)↓Inject(x)↓Copy(x)
Environment(x) Male(x) Substantive(x) Inject(x) Copy(x) $\rightarrow$ ¬Test(x)

/* TestFiles certified by testing organizations have no actual behaviors or infringement consequences. If a sample exhibits infringement characteristics, it is not a TestFile */

**Axiom 3**:

Substantive(x) $\rightarrow$ Male(x)

/* Software developed for non-malicious purposes cannot form deterministic threats (since developers cannot predict exploitation methods). Malware with deterministic substantive infringement must be maliciously designed or modified */

**Axiom 4**:

Inject(x) Copy(x) $\rightarrow$ Environment(x)&Substantive(x)

/* Active file infection and self-replication have already caused substantive infringement to the current environment */

**Axiom 5**:

x[¬Nomean(x)]&[¬Test(x)] $\rightarrow$ Environment(x) Male(x)

/* Malware classification work takes real-world occurred threats as prior experience. Software developed for normal purposes that does not infringe upon the current environment typically does not fall under antivirus work scope, or is on a whitelist. In either case, antivirus engines will not output malware naming. Therefore, objects entering the universe of discourse (except junk files and test

files as real-world compromises) are either deterministically maliciously developed or non-maliciously developed but with prior facts of malicious exploitation causing infringement */

### 5.5 Inference Rules

```
Nomean(x)   JunkFile(x)
Test(x)   TestFile(x)
[¬Nomean(x)]&[¬Test(x)]&[¬Environment(x)] $\rightarrow$ HackTool(x)
[¬Nomean(x)]&[¬Test(x)]&Environment(x)&[¬Male(x)] $\rightarrow$ Riskware(x)
[¬Nomean(x)]&[¬Test(x)]&Environment(x)&Male(x)&[¬Substantive(x)] $\rightarrow$ Grayware(x)
[¬Nomean(x)]&[¬Test(x)]&Environment(x)&Male(x)&Substantive(x)&Inject(x) $\rightarrow$ Virus(
[¬Nomean(x)]&[¬Test(x)]&Environment(x)&Male(x)&Substantive(x)&[¬Inject(x)]&Copy(x) $\rightar
[¬Nomean(x)]&[¬Test(x)]&Environment(x)&Male(x)&Substantive(x)&[¬Inject(x)]&[¬Copy(x)] $\righ
```

**MECE Principle Proof**

**Proof Objective**:

```
x MECE(Virus(x),Worm(x),Trojan(x),HackTool(x),Grayware(x),Riskware(x),TestFile(x),JunkFile(
```

**Logical Use Case Table** :
The table demonstrates that based on Axiom 1, when a file lacks meaning, there' s exactly one valid use case; based on Axiom 2, when a file is test-specific, there' s exactly one valid use case; other cases are excluded by Axioms 3-5, ultimately proving that exactly one classification assertion holds true for any object, satisfying MECE.

## 6. Logic and Concept of Threat Behavior Risk Labels

The core of using MECE-based malware classification principles is to satisfy coverage and mutual exclusivity. On another level, due to malware' s functional complexity and software code' s inherent flexibility, it' s impossible to cover all malware attributes with a finite, convergent classification set, let alone effectively represent critical threat information. A malware sample may possess multiple threat behaviors, necessitating a structural form supporting multi-expression coexistence that remains sufficiently concise. Clearly, using tags is a better choice.

Multi-segment malware naming structures are based on mutual exclusivity logic at each segment. For example, Kaspersky' s first segment is the malware classification prefix, second is environment prefix, third is family name, and subsequent content includes variant numbers and other modifiers. Each segment completes selection of the next branch node in a tree structure to achieve mutual exclusivity. We uniformly introduce a risk behavior suffix domain in sample naming, which can either output a single highest-level risk behavior or multiple delimiter-separated risk behaviors—the former helps relevant parties using alert

information focus on the most response-worthy behavioral risk, while the latter provides stronger information disclosure.

This basic approach aims to maintain detection log two-dimensionalization while ensuring sample naming and behavior information are still output as single strings, providing a foundation for transforming sample naming into multi-dimensional data structures in fine-grained event and knowledge information queries.

Basic considerations when establishing behavior tags include:
1. Covering security risks unsuitable as classifications but currently popular, high-risk, or of high user concern, such as ransomware, exploit, fraud, kernel masquerading, etc.;
2. Mapping and digesting small categories or primary prefixes output by mainstream security software under original non-MECE standard systems;
3. Covering several dimensions of malware critical behaviors: propagation methods, attack purposes and targets, attack techniques, concealment methods, etc.

Guided by this behavior tag system, Antiy has developed static and dynamic sample behavior judgment mechanisms in static analysis and dynamic sandbox development, enabling these behavior tags to be generated during automated analysis and provided to third parties.

The introduction of object behavior tags not only compensates for information disclosure deficiencies in convergent classification methods but also essentially enables transformation of all valid malware naming from mainstream vendors into corresponding naming based on foundation classification, runtime environment, family name, variant number, and behavior suffix. Simultaneously, when different engines can supply different object information for the same sample, this information can also be transformed into effective naming information, supporting emergency response organizations' attempts to unify alert formats and styles.

## 7. Application and Integration

### 7.1 Consolidated Classification Absorption Table

SCMP can completely absorb existing classifications and primary prefixes from precisely classifying vendors like Kaspersky through its eight categories, as detailed in the table below.

**Table 4 SCMP Classification Absorption Table**

| SCMP Category | Absorbed Vendor Classifications and Primary Prefixes |
|---|---|
| Virus (Infectious) | • Kaspersky: Virus • Microsoft: Macro virus |
| Worm | Kaspersky: Worm, Email-Worm, IM-Worm, Net-Worm, P2P-Worm |

| SCMP Category | Absorbed Vendor Classifications and Primary Prefixes |
|---|---|
| Trojan | • Kaspersky: Trojan, Trojan-Ransom, Backdoor, Trojan-Rootkit, Trojan-Banker, Trojan-Clicker, Trojan-Downloader, Trojan-Dropper, Trojan-ArcBomb, Trojan-Spy, Trojan-DDoS, Trojan-Botnet, Trojan-Miner, Trojan-Proxy, Trojan-Dailer, Trojan-Keylogger, Trojan-PWS, Trojan-FakeAV, etc. • Bitdefender: Trojan, Exploit, Keylogger, Backdoor, Downloader • McAfee: Trojan, PWS • Symantec: Trojan, Backdoor, Miner, Downloader, Ransom • Microsoft: Trojan, Ransomware, Exploit, Backdoor, Downloader, Dropper, Rogue security software, Password stealer, Trojan-Clicker, Command and Control |
| HackTool | • Kaspersky: Hacktool, Constructor, VirTool • Microsoft: Hacktool, Obfuscator, VirTool |
| Grayware | • Kaspersky: Adware, Pornware • Bitdefender: Spyware, Porn, Adware |
| Riskware | • McAfee: PUP • Microsoft: PUA • AVG: PUP • Kaspersky: RemoteAdmin, Monitor, NetTool, RiskTool |
| TestFile | • Kaspersky: EICAR-Test-File • Bitdefender: EICAR-Test-File • Microsoft: RemoteAccess |
| JunkFile | • Not covered by other vendors |

### 7.2 Clarification of Typical Classification Absorption and Related Cognitive Misconceptions

Since SCMP significantly reduces the number of malware classifications, we must clarify some typical cognitive misconceptions about malware. These misconceptions arise from both insufficient malware cognition and translation issues plus historical engineering factors. Below we introduce several common classification misconceptions and explain why they are unscientific.

**(1) Backdoor**

In computer virus classification nomenclature, "Backdoor" first appeared around 2000. The earliest malware marked as Backdoor was a remote control tool released by The Cult of the Dead Cow organization. Subsequently, similar remote control malware like "NetThief" and "IceRiver" were marked as backdoors by mainstream antivirus vendors. Clearly, the "Backdoor" classification prefix does not refer to code defects in software but rather originates from the earliest control tool using Back Orifice, making it a "backdoor" providing direct system access. As related malware evolved to penetrate internal networks and upgrade remote management technologies, samples continued being named backdoors. The actual connotation of "backdoor program" refers to Trojans with remote control capabilities.

Therefore, in SCMP classification nomenclature, such malware is uniformly clas-

sified as "Trojan," while "Backdoor" appears as one of the highest-risk-level tags, without affecting information disclosure.

**(2) Spyware**

In some domestic and international literature, "Spyware" is interpreted as "malware with espionage capabilities." This interpretation is actually literal and does not 符合 the term' s original connotation. The era background when antivirus workers first proposed "Spyware"was the large-scale prevalence of internet clients. Some plugins were installed in users' startup directories without consent, becoming hidden default plugins. "Spy" conveyed the installation characteristic of software programs "silently entering," not the threat behavior of "information theft." Later interpretations of spyware were 事实上 misreadings.

Therefore, in SCMP classification nomenclature, such malware is classified as "Grayware,"while "Spy-Install"serves as a malware behavior tag without affecting information disclosure. Malware with clear sensitive and critical information theft behavior may be 酌情 classified under the "Trojan" category.

**(3) Botnet**

Some later detection software naming includes categories like Botnet, but botnet is not a sample concept; to some extent, it encompasses sample utilization and organizational methods. Introducing it as a category adds a new classification dimension to the malware taxonomy, clearly violating fundamental malware classification principles and 事实上 interfering with classification standards.

From the virus sample perspective, botnet programs running on victim ends share mechanisms with remote-control-capable Trojans, differing only in that the former has relatively stronger automated control capabilities by instruction. Therefore, in SCMP classification nomenclature, such malware is uniformly classified as "Trojan,"while "Botnet"appears as a high-risk-level tag without affecting information disclosure.

**(4) Adware, Pornware, Rogue**

Adware and pornware have also been treated as independent categories by some antivirus software. However, since internet-based gray and black industries are actually joint operation systems, and adware pop-up content often contains large amounts of pornographic files, these two categories are difficult to clearly differentiate. Their essence is attracting user attention through specific content to guide clicks and downloads, facilitating subsequent illegal information collection and other infringing activities. Rogue software installs on computers or mobile devices without explicit user authorization or knowledge. Rogue, adware, and pornware typically spread through bundling with other software, downloaders, or shareware, or through deceptive download links, phishing emails, etc. While these software types may cause user inconvenience, they generally do not cause direct system or data damage or harm.

Therefore, in SCMP classification nomenclature, such malware is uniformly classified as "Grayware," while "Ad, Porn" appear as high-risk-level tags without affecting information disclosure.

**(5) Ransomware**
Undoubtedly, ransomware is currently the most serious security threat, and alerting and disposing of prevalent ransomware is crucial. However, this does not mean "ransomware" is suitable as an independent malware foundation category. Because regardless of ransomware' s working mechanisms and modes, it essentially does not escape the definition of "Trojan."

Therefore, in SCMP classification nomenclature, such malware is uniformly classified as "Trojan," but "Ransom" appears as the highest-risk-level behavior tag without affecting information disclosure.

**(6) Constructor and Obfuscator**
Virus construction kits are relatively ancient concepts, primarily generating virus samples through functional module combination and obfuscation operations under backgrounds where DOS executable files (like COM files) lacked file format specifications and DOSMZ lacked strict format validation. Meanwhile, since infectious virus code segments are not independent executable programs, developers wanting to achieve initial infection must first construct hosts, hence the constructor concept emerged.

Clearly, constructor is an early DOS-era concept whose significance in current environments is minimal. Similarly, some modular Trojans have configuration interfaces for customization and processing. Their configurators can be classified under "HackTool"according to previous definitions, or classified under the Trojan category based on the principle of "matching with payloads."

The obfuscator concept emerged during the DOS-era polymorphic engine era, with core mechanisms achieving fusion between polymorphic engines and virus payloads, enabling non-obfuscated, non-polymorphic samples to gain polymorphic capabilities after combining with polymorphic engines. As malware forms evolved from infectious virus-dominated to independent and even firmware-dominated, there are 事实上 only sample packing concepts, no longer malware obfuscator concepts.

Since most packing tools serve copyright protection scenarios, only being used as detection evasion behavior features in specific attack scenarios, rashly alerting on packers would cause false positives on normal application software. Currently, antivirus vendors alert on some underground packers used only in attack scenarios, but their alerts often appear as named event prompt information (rather than alert information) to demonstrate caution. Overall, all mainstream antivirus products do not alert on packers alone, making it inappropriate to treat obfuscators as a malware classification category.

## 8. Final Effects and Practical Application

We apply the overall malware classification framework using "Category" / "Environment Prefix," "Family Name," "Variant Number," and [Behavior Tags]. For example, the sample naming: **Trojan/Win32.Akira[Ransom]**. In actual

security operations, we can clearly see this is a Trojan running on Windows 32-bit platform with core risk behavior being ransomware attacks. In automated early warning notifications, structural parsing of this string can transform into security announcements like "Ransomware Risk Alert: Beware of Akira Trojan."

### 8.1 Significance of Scientific Classification for Cybersecurity Management Operations

Cybersecurity management operations require scientific and clear malware classification standards and naming conventions. Malware samples are 海量, their distribution is long-tailed, and most users no longer possess independent malware analysis capabilities, relying on security products for precise identification and fine-grained processing. The ability of security products to precisely name malware and clearly distinguish malware categories correlates with operational issues cybersecurity management operators must consider and how to construct corresponding foundational operation and disposition processes.

The SCMP malware classification method can provide more accurate and higher-information-disclosure malware information, supporting cybersecurity management operators in threat identification, risk assessment, security policy formulation, security incident response, and disposition process construction. It focuses limited energy on several deterministic processing workflows, thereby improving operation disposition effectiveness and efficiency.

Based on several differentiation criteria of the classification standard: if alerts for Virus or Worm with autonomous propagation capabilities are detected, it indicates users may lack basic security compliance capabilities or have not adhered to fundamental security baselines; if Trojan alerts are detected, targeted analysis and disposition are required; if HackTool is detected, it indicates the host may become a 跳板 or be used for lateral movement, sensitive information theft, or further attacks, but could also be internal red team tools; if Grayware is detected, it indicates weak information leakage infringement or governance capability needs improvement, but can also be 选择性地搁置; if Riskware is detected, it requires investigation whether network administrators actively installed and used it—if so, whitelist addition is needed, if not, it may transition to Trojan disposition workflow; network administrators can use JunkFile alerts to judge sample set quality, while JunkFile can also reduce user panic; TestFile generally does not appear unless users actively deploy it—if TestFile is discovered in user systems, investigation is needed to determine whether attackers constructed 伪装 test files. Combined with core behavior tags, this converges 海量 malware disposition into several deterministic workflows.

**Table 5 Malware Category Associated Operational Actions**

| Category | Quarantine | Delete | Alert | Investigate | Whitelist | Sample Analysis | Inquiry |
|---|---|---|---|---|---|---|---|
| Virus | √ | √ | √ | √ | - | * | * |

| Category | Quarantine | Delete | Alert | Investigate | Whitelist | Sample Analysis | Inquiry |
|---|---|---|---|---|---|---|---|
| Worm | √ | √ | √ | √ | - | * | * |
| Trojan | √ | √ | √ | √ | - | * | * |
| HackTool | √ | √ | √ | √ | - | * | * |
| Grayware | √ | - | √ | - | - | - | - |
| Riskware | √ | - | √ | √ | √ | - | - |
| TestFile | NA | NA | NA | NA | NA | NA | NA |
| JunkFile | NA | NA | NA | NA | NA | NA | NA |

*Recommended: √ Appropriate, - Not recommended, NA Not applicable

### 8.2 Practical Application

The above classification and naming fundamental principles have been practically engineered and applied in Antiy' s AVL SDK antivirus engine customers. Nearly 100 partners use our detection engine. According to incomplete statistics, over 1.3 million network devices and security devices, over 2 million PC and cloud nodes, and over 3 billion mobile phones and smart terminals use this antivirus engine.

Simultaneously, relying on the rigorous mutual exclusivity of this naming structure, we have built a computer virus classification and naming knowledge encyclopedia based on massive malware data analysis and with large model assistance. By the end of 2023, we had 收录 53,000+ virus family entries, achieving 100% coverage of known malware family classification naming, currently entering daily incremental family maintenance status.

Beyond our own security practices, this classification and naming structure can basically losslessly absorb all classifications and naming information from other mainstream security vendors, supporting emergency response organizations' attempts to unify alert formats and styles.

Of course, facing continuously expanding security threats and increasingly rich attack patterns, defenders inevitably fall into difficulties and confusion. However, code confrontation remains the most fundamental mode of cyberspace confrontation; malware remains the attack weapon in the vast majority of cyber attack activities. Effective malware identification and containment are even more critical for defenders. It can be said that malware detection has become the "friend-or-foe identification"capability in cyberspace confrontation, and what we do is make our precise classification detection and identification capabilities cover as many attackers' "attack weapons" as possible.

## References

[1] Vesselin Vladimirov Bontchev. Current Status of the CARO Malware Nam-

ing Scheme[EB/OL].[2024-04-29]. https://bontchev.nlcv.bas.bg/papers/naming.html

[2] Kaspersky Lab[EB/OL].(2013-12-10)[2024-04-29].https://encyclopedia.kaspersky.com/knowledge/rules-for-naming/

[3] Symantec[EB/OL].[2024-04-29]https://www.broadcom.com/support/security-center

[4] Microsoft[EB/OL].(2024-04-22)[2024-04-29].https://learn.microsoft.com/en-us/microsoft-365/security/intelligence/malware-naming?view=o365-worldwide

[5] Trend Micro[EB/OL].(2019-12-30)[2024-04-29]. https://success.trendmicro.com/dcx/s/solution/1119738-new-threat-detection-naming-scheme-in-trend-micro?language=en_{US}

[6] The MITRE Corporation.[EB/OL].[2024-04-29].https://maecproject.github.io/

[7] European Institute for Computer Antivirus Research[EB/OL].[2024-04-29].http://www.eicar.org/

(Corresponding Author: Li Chenping E-mail: lichenping@antiy.cn)

## Author Contributions Statement

Xiao Xinguang: Proposed research ideas, designed research plan;
Tong Zhiming, Han Yaoguang: Conducted experiments;
Han Yaoguang, Tong Zhiming: Collected, cleaned, and analyzed data;
Xiao Xinguang, Li Chenping, Han Yaoguang, Tong Zhiming: Drafted paper;
Li Qi: Created illustrations;
Li Chenping: Revised final version.

*Note: Figure translations are in progress. See original paper for figures.*